

NCB 30x1

Multi-Touch Controller

I2C Interface Protocol

nuvoTon Technology Corporation

NuTouch

Version 0.4 Jun/2013

1. Revision History

Version	Date	Author	Description
0.1	2013/03/01	CSSHIH	First Release
0.2	2013/03/14	CSSHIH	1. Modify the format of point report. 2. Cancel the reading of initial message.
0.3	2013/05/21	CSSHIH	Modify the format of point report.
0.4	2013/06/24	CSSHIH	Correct the sample code

2. Introduction

NCB30x1 is a multi-touch controller IC for capacitive touch panel. Because of integrated powerful noise suppression circuit, charge distribution sensing technic, micro-controller and combining signal analysis with advanced algorithm, NCB30x1 can supplies auto self-calibration for wide operating temperature range, noise rejection for LCM noise, charger noise, and high performance multi-touch tracking function. It can be capable for any touch screen application combined with high performance controller and friendly user interfaces. NCB30x1 support the register mapped interface. User can configure the NCB30x1 or get information like coordinates or raw data by the I2C interface.

3. I2C Interface Protocol

3.1 Default I2C Address

The default I2C Address of NCB30x1 is 0x55(7-bit address)

3.2 Register Read

For reading the registers value from NCB30x1, the I2C host has to

tell the NCB30x1 the “start register address” before reading the corresponding register value.

I2C Start	I2C Addr(W)	Start Reg. Addr.LB(a)	Start Reg. Addr.HB(a)	I2C Stop	I2C Start	I2C Addr(R)	Value of Reg(a)	Value of Reg(a+1)	...	Value of Reg(a+n)	I2C Stop
-----------	-------------	-----------------------	-----------------------	----------	-----------	-------------	-----------------	-------------------	-----	-------------------	----------

3.3 Register Write

For writing the registers of NCB30x1, the host has to tell the NCB30x1 the “start register address”. Register value would be written to the register with address starting from the “start register address”

I2C Start	I2C Addr(W)	Start Reg. Addr. LB(a)	Start Reg. Addr. HB(a)	Value of Reg(a)	Value of Reg(a+1)	...	Value of Reg(a+n)	I2C Stop
-----------	-------------	------------------------	------------------------	-----------------	-------------------	-----	-------------------	----------

4. Register Definitions

NCB30x1 provides a register interface for host to configure device attributes and retrieve information. The registers are listed bellow.

Register Addr	Name	Attribute	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x0102	Reset	R/W	Reset							
0x0103	BackUpNV	R/W	Backup Settings							
0x0104	Calibrate	R/W	Calibrate							
0x0105	ReportAll	R/W	Report Current Status							
0x0106	Reserved	Reserved	Reserved							
0x0107	Diagnostic	R/W	Diagnostic Debug Command							
0x0108	IdleAcqInt	R/W	Idle Acquisition Interval							
0x0109	ActAcqInt	R/W	Active Acquisition Interval							
0x010A	Reserved	Reserved	Reserved							
0x010B	Orient	R/W	Reserved					InvertY	InvertX	Switch
0x010C	XRangeLsB	R/W	X Resolution Low Byte							
0x010D	XRangeMsB	R/W	X Resolution High Byte							
0x010E	YRangeLsB	R/W	Y Resolution Low Byte							
0x010F	YRangeMsB	R/W	Y Resolution High Byte							

4.1 Multi-touch points report

When NCB30x1 has touch data to send, it typically asserts the $\overline{\text{INT}}$ line to indicate to the host that there is a message. Once the host starts to read data, the device clears the $\overline{\text{INT}}$ line. The format of multi-touch points report is described as bellow.

The length of multi-touch points report is 60bytes, 6 bytes for each FID.

Index	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x01	FID/Status	FID					Status		
0x02	XPOSMSB	X position LSByte							
0x03	YPOSMSB	Y position LSByte							
0x04	XYPOSLSB	Y position msbits				X position msbits			
0x05	TCHAREA	Size of touch							
0x06	TCHAMPLITUDE	Touch amplitude(sum of measured deltas)							
...							
0x37	FID/Status	FID					Status		
0x38	XPOSMSB	X position LSByte							
0x39	YPOSMSB	Y position LSByte							
0x3A	XYPOSLSB	Y position msbits				X position msbits			
0x3B	TCHAREA	Size of touch							
0x3C	TCHAMPLITUDE	Touch amplitude(sum of measured deltas)							

FID Field

Indicate the contact ID of touches.

STATUS Field

Report the current status of the touchscreen; that is, a touch (press) or a release. If the touch is not detected, the FID/Status field would be 0xFF.

PRESS: 0x01

RELEASE: 0x02

MOVE: 0x03

INVALID: 0x07(FID field would be 0x1F)

XPOSLSB, YPOSLSB and XYPOSMSB Fields

These three fields report the X and Y position, with XPOSLSB/YPOSLSB containing the least significant byte of the position and XYPOSMSB containing the most significant bits of the position.

TCHAREA Field

Report the size of the touch area in terms of the number of channels that are covered by the touch.

TCHAMPLITUDE Field

Report a value that is proportional to the signal delta of the channels within a touch.

4.2 Reset

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x0102	Reset	Reset							

Force a reset of the device if a nonzero value is written. If 0xA5 is written to this register, the device resets into bootloader mode.

Write Value: Nonzero (normal) or 0xA5 (bootloader)

4.3 BackUpNV

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x0103	BackUpNV	Backup Settings							

Backs up the RC information to the flash if 0x20 written to this register.

Write Value: 0x20

4.4 Calibrate

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x0104	Calibrate	Calibrate							

Perform a global recalibration (rebuild the RC information) on all

channels.

Write value:0x20

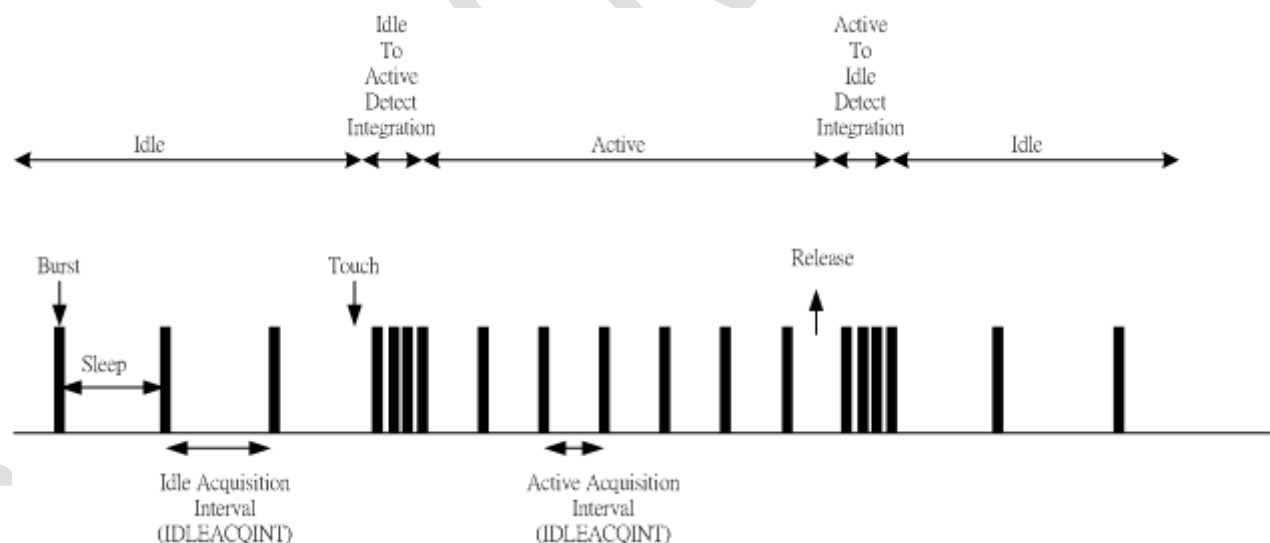
4.5 Diagnostic

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x0107	Diagnostic	Diagnostic Debug Command							

Control the operating mode of the device. In default, device would be in normal mode. This field is reserved for engineering mode.

4.6 Acquisition interval

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x0108	IdleAcqInt	Idle Acquisition Interval							
0x0109	ActAcqInt	Active Acquisition Interval							



The length of the idle and active burst cycles is determined by the Idle Acquisition interval (IDLEACQINT) and Active Acquisition Interval (ACTVACQINT) registers respectively.

The valid range for IDLEACQINT and ACTVACQINT is between 0 and 255.

A setting of 255 makes the device run in Free-run mode, when it does not sleep.

A setting of zero forces the device to enter Deep Sleep mode. The device remains in Deep Sleep mode until the IDLEACQINT or ACTVACQINT setting is restored.

Other values for IDLEACQINT or ACTVACQINT determine the Idle or Active Acquisition Interval in milliseconds. Note that a high value causes more sleep time between acquisitions, resulting in lower power consumption but a slower response time.

Do not set either field to be less than the actual burst time.

Range : 0 (Deep Sleep), 1-254(interval in ms), 255(Free run)

IDLEACQINT Typical : 32(32ms)

ACTVACQINT Typical : 16(16ms)

4.8 Orient

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x010B	Orient	Reserved					InvertY	InvertX	Switch

This register controls the orientation of the touchscreen, such as flipping and rotating the screen display.

SWTICH: Switches the X and Y positions; that is, the screen is flipped about the diagonal.

INVERTX: Inverts X coordinates; that is : $X_{newval} = (X_{maxval} - X)$.

INVERTY: Inverts Y coordinates; that is : $Y_{newval} = (Y_{maxval} - Y)$.

4.9 Resolution

Register Addr	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x010C	XRangeLsB	X Resolution Low Byte							
0x010D	XRangeMsB	X Resolution High Byte							
0x010E	YRangeLsB	Y Resolution Low Byte							
0x010F	YRangeMsB	Y Resolution High Byte							

The XRange and YRange registers set the output resolution for the reported position. These registers can be set from 0 to 4095. These registers allow a touchscreen's position to match an LCD's

resolution. For example, to set up a touchscreen to match an LCD with a resolution of 800x600, XRange and YRange would be set to 799 and 599 respectively.

XRANGE/YRANGE Range: 0-4095

XRANGE/YRANGE Default: 1279/799

5. Samples Codes

5.1 Data Structures and APIs

```
typedef struct{
    u8 contact_id;
    u8 status;
    u16 x;
    u16 y;
    u8 area;
    u8 amplitude;
}MultiTouchMessage_t;
```

//The default I2C address(7-bit) of NCB30x1 would be 0x55

```
#define I2C_Addr 0x55
```

//I2C Master sends count bytes data stored in buf to I2C Slave

//I2C package: |S|I2C Addr|W|Data(buf)|P|

```
extern int i2c_master_send(char I2C_A,const char *buf, int count);
```

//I2C Master reads count bytes data to buf from I2C Slave

//I2C package: |S|I2C Addr|R|Data(buf)|NAK|P|

```
extern int i2c_master_recv(char I2C_A,char *buf, int count);
```


5.2 Read coordinates

```
static uint8_t get_multi_touch_message(MultiTouchMessage_t *
MTouchMessage){
    u8 rx_buf[60], i, touchCount;
    u16 temp;

    i2c_master_recv(I2C_Addr,&rx_buf[0],60);

    //Read message of multi-touch report
    touchCount = 0;
    for(i=0;i<10;i++)
    {
        if(rx_buf[i*6] == 0xFF)
        {
            break;
        }

        MTouchMessage[i].contact_id = rx_buf[i*6]>>3;

        MTouchMessage[i].status = rx_buf[i*6]&0x07;

        MTouchMessage[i].x = (rx_buf[i*6+1] ) |
            ((rx_buf[i*6+3]&0x0F)<<8);

        MTouchMessage[i].y = (rx_buf[i*6+2] ) |
            ((rx_buf[i*6+3]&0xF0)<<4);

        MTouchMessage[i].area = rx_buf[i*6+4] ;

        MTouchMessage[i].amplitue = rx_buf[i*6+5];

        touchCount+=1;
    }
```

```

        return touchCount;
    }

```

5.3 Power Down the NCB30x1

//Write the registers with address 0x0108 and 0x0109 to be 0 would
 //command the NCB30x1 get into power down.

```

static void set_power_down(void){
    u8 tx_buf[4];
    tx_buf[0] = 0x08;
    tx_buf[1] = 0x01;
    tx_buf[2] = 0x00;
    tx_buf[3] = 0x00;
    i2c_master_send(I2C_Addr,&tx_buf[0],4);
}

```

5.4 Wake Up the NCB30x1

//Wake up the NCB30x1 by writing the registers with address
 //0x0108 and 0x0109 with non-zero values.

//Recommended value of active_interval is 10;

//Recommended value of idle_interval is 50;

```

static void wake_up(u8 active_interval,u8 idle_interval){
    u8 tx_buf[4];
    tx_buf[0] = 0x08;
    tx_buf[1] = 0x01;
    tx_buf[2] = active_interval;
    tx_buf[3] = idle_interval;
    i2c_master_send(I2C_Addr,&tx_buf[0],4);
}

```