

Document ID: EPG-003-151217-1

# I<sup>2</sup>C Programming Guide

For EETI Multi-Touch (EXC7200 / EXC7700)



**EETI CONFIDENTIAL**


For Release Only Under Non-Disclosure Agreement (NDA)

For 威綸科技股份有限公司

**Trademark Acknowledgments:**

I2C is a registered trademark of Philips Electronics.

EETI and EETI logo  and eGalaxTouch® logo 

and eGalaxWorks® logo  are trademarks of eGalax\_eMPIA Technology Inc.

(C) Copyright by EETI 2000, 2015. All rights reserved.sgagaga

Printed in Taiwan.

**EETI CONFIDENTIAL**  
RELEASE UNDER NON-DISCLOSURE AGREEMENT (NDA)  
For 威倫科技股份有限公司



eGalax\_eMPIA Technology Inc.

11F, No 302, Rueiguang Road, Nei Hu District,

Taipei 114, TAIWAN

T: +886 2 8751 5191

F: +886 2 2797 8808

URL: [www.eeti.com](http://www.eeti.com)

Sales : [touch\\_sales@eeti.com](mailto:touch_sales@eeti.com)

FAE : [touch\\_fae@eeti.com](mailto:touch_fae@eeti.com)

## Revision History

Document ID	Date	Revision Description
-	2012/09/24	1.01 (2011/03/17) 1.02 (2011/03/17) 1.03 (2011/03/17) a. Modify the command to query firmware version & controller name. 1.04 (2011/03/21) 1.05 (2011/03/24) 1.06 (2011/04/08) a. Modify the protocol of writing mode. 1.07 (2011/04/14) a. The handle and wakeup timing chart is added. b. Add I2C address 0x2A 1.08 (2011/07/07) a. Add I2C Characteristics for 400K 1.09 (2012/09/024) a. Update document
-	2012/12/12	New layout of a printed document.
-	2013/06/25	Modify the description for diagnostics packet. 2.3.1 page.7, 2.3.2 page.7, 2.3.3 page.8
-	2014/03/17	Modify Figure 3-1 and Table 3-1 (Page.5)
EPG-003-140926-1	2014/09/26	Update EETI document form.
EPG-003-151217-1	2015/12/17	Revise page 6 for typo.

## Index

Revision History .....	3
Introduction .....	5
1 I/O Pin Definition .....	5
1.1 I <sup>2</sup> C AC Waveform .....	6
1.2 Power- on Timing Chart .....	7
2 Software Protocol .....	7
2.1 Slave Address .....	8
2.2 Firmware Power-on Initialization .....	8
2.3 Read/Write Protocol .....	8
3 Power Saving Mechanism .....	13

**EETI CONFIDENTIAL**  
RELEASE UNDER NON-DISCLOSURE AGREEMENT (NDA)  
For 威倫科技股份有限公司

## Introduction

EETI provides a touch controllers designed to optimize the performance of Projected Capacitive Touchscreen (PCAP/PCT). The controller can communicate with system directly through RS232, USB and I2C. This document will assist you to integrate controller through the I2C interface, and describe the related software protocol. The software protocol supports all members of the EXC7200/EXC7700 family.

### 1 I/O Pin Definition

Pin Number	Pin Name	I/O	Description
1	GND	Power	Ground
2	SDA	Open Drain	I <sup>2</sup> C Data
3	SCL	Open Drain	I <sup>2</sup> C Clock
4	VDD	Power	5V/3.3V regulated
5	IRQ	Open Drain	Interrupt Request pin
6	RST	Input/Open Drain	Reset pin to Master Chip

GND : Ground pin.

This pin should be connected with the host GND.

SDA : I2C data pin.

This pin is configured as open drain. It needs a pull up resistor (4.7K) for I2C communication.

SCL : I2C Clock pin.

This pin is configured as open drain pin. It needs a pull up resistor (4.7K) for I2C communication.

VDD : The power supply pin.

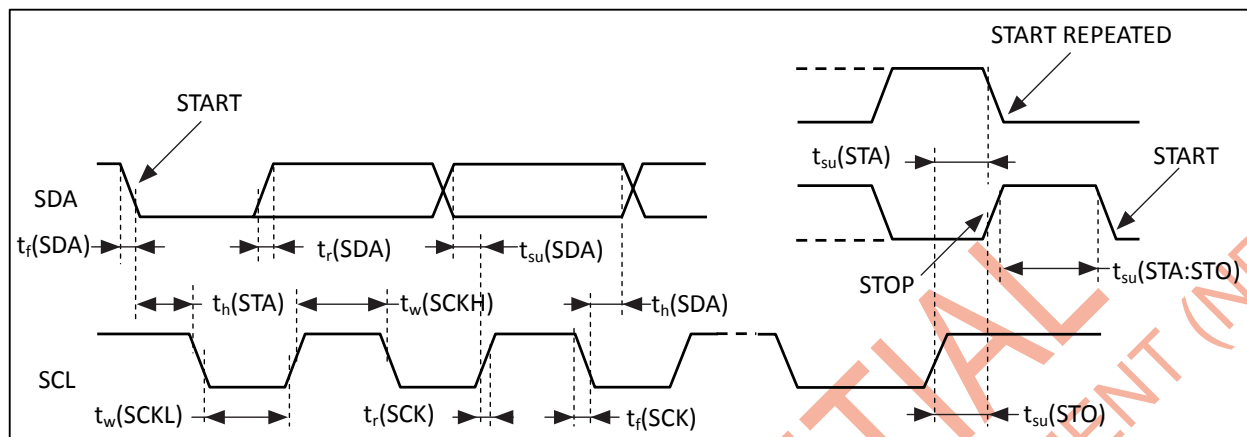
IRQ : Interrupt request pin.

This pin should be kept at logic high at idle state. Whenever the controller has any data to be sent to host computer, controller pulls low it to generate an interrupt to host computer.

RST : Optional reset pin.

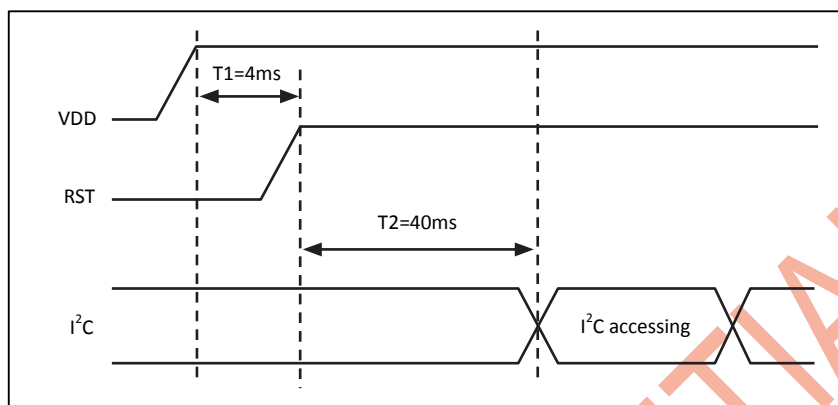
Pull low this pin and up this pin to cause Touch controller reset, the minimum width is 100ms.

When the system works normally, we urge that the RST be set to the input (Open Drain) configuration.

1.1 I<sup>2</sup>C AC Waveform

Symbol	Parameter	SCL = 100KHz		SCL = 400KHz		Unit
		Min	Max	Min	Max	
$t_w(\text{SCKL})$	SCL clock low time	4.7		1.3		$\mu\text{s}$
$t_w(\text{SCKH})$	SCL clock high time	4.0		0.6		
$t_{su}(\text{SDA})$	SDA setup time	250		100		ns
$t_h(\text{SDA})$	SDA data hold time	0		0	900	
$t_r(\text{SDA})$	SDA and SCL rise time		1000		300	
$t_r(\text{SCL})$						
$t_f(\text{SDA})$	SDA and SCL fall time		300		300	$\mu\text{s}$
$t_f(\text{SCL})$						
$t_h(\text{STA})$	Start condition hold time	4.0		0.6		
$t_{su}(\text{STA})$	Repeated Start condition setup time	4.7		0.6		
$t_{su}(\text{STO})$	Stop condition setup time	4.0		0.6		$\mu\text{s}$
$t_w(\text{STO:STA})$	Stop to Start condition time (bus free)	4.7		1.3		

## 1.2 Power- on Timing Chart



## 2 Software Protocol

I²C Transaction Frame: each I2C transaction frame transfers one I²C packet data. And each packet data have 10 bytes data as following chart. But, each packet data may not be an exact application packet. The detailed application packet format will be described later.

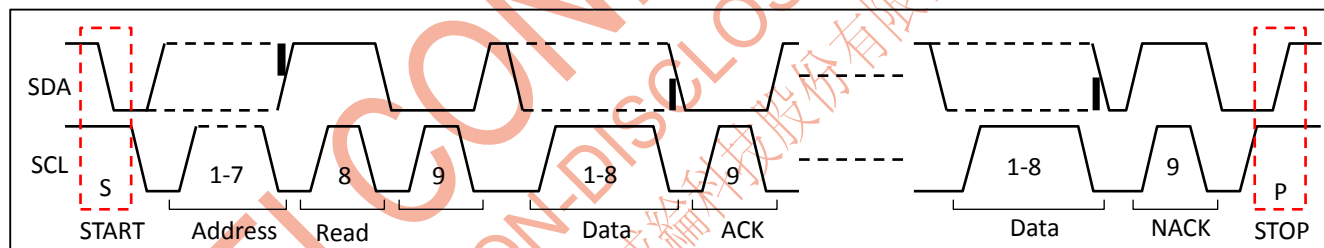


Fig 2-1

The controller interrupt service is LOW LEVEL trigger. The controller will pulls IRQ pin low level until no data in the buffer. The host must retrieve all data until the IRQ pin returns to high level.

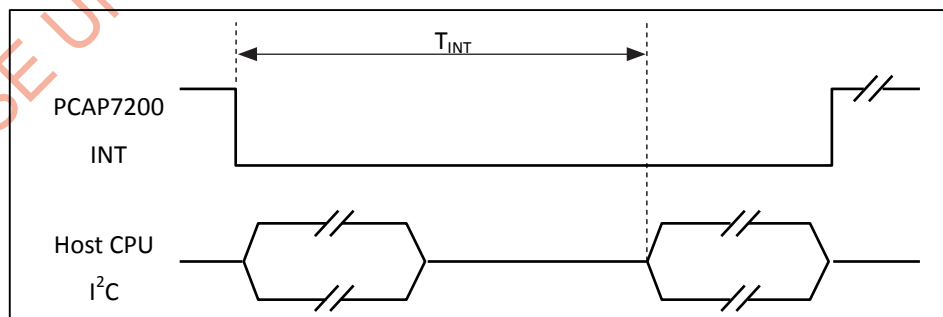


Fig 2-2

Report rate =  $1 / T_{INT}$ , it depends on properties of touch screen such as resistive value, I2C clock rate, channel number, thickness and material of cover lens, etc.

## 2.1 Slave Address

The EETI touch controller makes use of 7-bit addressing.

For EXC7200 controller, the address is 0x04.

For EXC7700 controller, the address is 0x2A.

## 2.2 Firmware Power-on Initialization

When the controller is firstly power on, the host must write a command to enable the device, and then the controller will run at fully working mode immediately. If no initial procedure, the controller had no touch function. In general, host can write a loopback command to do initialization. About the loopback command, please refer 2.3.3 Command List for detailed packet format.

## 2.3 Read/Write Protocol

	From Host to Device
	From Device to Host

S = START condition

Sr = Repeat START condition

P = STOP condition

R = Data direction READ (SDA HIGH)

W = Data direction WRITE (SDA LOW)

Ack = Acknowledge (SDA LOW)

Nak = Not acknowledge (SDA HIGH)

Address = 7-bit (0x2A)

DATA = 8-bit

**Read mode:** Host-receiver, Device-transmitter.

S	ADDRESS	R	A	DATA1	A	DATA2	A	DATA3	A	DATA4	A	DATA5	A	DATA6	A
				DATA7	A	DATA8	A	DATA9	A	DATA10	N	P			

**Write mode:** Host-transmitter, Device-receiver.

S	ADDRESS	W	A	DATA1	A	DATA2	A	DATA3	A	DATA4	A	DATA5	A	DATA6	A
				DATA7	A	DATA8	A	DATA9	A	DATA10	A	P			



## a. From Host to Device:

Report ID = 3 (Diagnostics mode)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
0x03	len	D1	D2	D3	D4	D5	D6	D7	D8

This Report ID = 0x03 is used for diagnostics packet.

len = valid data length in bytes of this current I2C data packet.

D1 to DN totally N bytes ( $N \leq 8$ ) are valid data payload in this I2C data packet. This valid data payload must follow eGalax diagnostics format. The D1 is used as the packet header and should be 0x03 or 0x0A. The programmer should be carefully to handle this. For a long eGalax diagnostics packet may be packaged into several I2C date packet.

## b. From Device to Host:

Host computer poll this I2C device only when the IRQ pulled low by this Touch device. The host computer should not poll this device when this IRQ signal pulled high.

The packet size of each I2C transaction frame is defined as 10 bytes

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
ID	D0	D1	D2	D3	D4	D5	D6	D7	D8

ID is defined as Report ID. The report ID was defined as below

Report ID = 3 (Diagnostics mode)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
0x03	len	D1	D2	D3	D4	D5	D6	D7	D8

This Report ID (0x03) is used for diagnostics packet.

len = valid data length in bytes of this current I2C data packet.

D1 to DN totally N bytes ( $N \leq 8$ ) are valid data payload in this I2C data packet. This valid data payload must follow eGalax diagnostics format. The D1 is used as the packet header and should be 0x03 or 0x0A. The programmer should be carefully to handle this. For a long eGalax diagnostics packet may be packaged into several I2C date packet.

Report ID = 4 (MutliTouch report)

	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
0x04	D0	D1	D2	D3	D4	D5	D6	D7	D8

D0 : B7 = Touch Valid. B7 = 1 is valid touch

[B6:B2] = Contact ID.

B1 = In Range bit, this bit should be always 1

B0 = Down/Up bit, B0 = 1 for Touch Down, B0 = 0 for Lift Off

D1 = Low byte of X coordination

D2 = High byte of X coordination

D3 = Low byte of Y coordination

D4 = High byte of Y coordination

D5 = Low byte of Z coordination

D6 = High byte of Z coordination

D7, D8 = reserved and must be zero.

#### c. Command List

\*Note: The following commands are used the '0x0A' diagnostics packet header as example.

##### ● Loopback Command

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
H→D	0x03	0x03	0x0A	0x01	0x41	0	0	0	0	0
D→H	0x03	0x03	0x0A	0x01	0x41	0	0	0	0	0

When device receive this command, it should respond the same packet to host immediately. Host can send this command to check whether device exist or do the power-on initial procedure.

##### ● Set Idle State

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
H→D	0x03	0x06	0x0A	0x04	0x36	0x3F	0x01	T	0	0

Host computer send the command as above for idle state configuration setting. Where, T means the scanning interval when in idle state. The touch controller will wake up every that period of time to scan touch screen to check if the touchscreen touched or not. Once it detects touchscreen touched, it will response and back to fully working state immediately.

The reasonable value is 0~9. The interval = (T + 1) X 50ms.

- Sleep State

H→D	0x03	0x05	0x0A	0x03	0x36	0x3F	0x02	0	0	0
-----	------	------	------	------	------	------	------	---	---	---

Host computer send above command packet to touch controller device to make the device enter sleep state for power saving. Once the touch controller device receives this command packet, it enters deep sleep state and the controller does not response until it wakes up from this sleep state. The touch controller device response with above D->H packet data only after it wakes up.

- Wakeup Response Command

D→H	0x03	0x05	0x0A	0x03	0x36	0x3F	0x01	0	0	0
-----	------	------	------	------	------	------	------	---	---	---

Once the controller transfers to wake up from idle and sleep state, it will send the response command as above to HOST.

- Command to Query Firmware Version

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
H→D	0x03	0x03	0x0A	0x01	'D'	0	0	0	0	0
1st D→H	0x03	0x08	0x0A	L1	'D'	Major	.'	Minor1	Minor2	Minor3
2nd D→H	0x03	L2	D1	D2	0	0	0	0	0	0

**L1** : Default value is 0x08 in future, express as length of 1st B4~B9 and 2nd B2~B3.

**L2** : Default value is 0x02 in future, express as length of 2nd B2~B3.

**D1, D2** : Dummy Byte.

Host computer send above H->D command packet to touch controller device to query the firmware version. The device responds with the data packet like above D->H. However, the device responds to this query command with the firmware version ASCII string. The length of the firmware version is variable.

The firmware version is v Major. Minor1 Minor2 Minor3, The Major, Minor1, Minor2 and Minor3 are expressed as characters. For example, the packet of firmware version "1.000" is as follows:

1st 0x03, 0x08, 0x0A, 0x08, 0x44, 0x31, 0x2E, 0x30, 0x30, 0x30

2nd 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

According to previous version "0.990a" is as follows:

1st 0x03, 0x08, 0x0A, 0x09, 0x44, 0x30, 0x2E, 0x39, 0x39, 0x30

2nd 0x03, 0x03, 0x61, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

● Command to Query Controller Name

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
H→D	0x03	0x03	0x0A	0x01	'E'	0	0	0	0	0
1st D→H	0x03	0x08	0x0A	L1	'E'	N1	N2	N3	N4	N5
2nd D→H	0x03	L2	N6	N7	N8	D1	D2	0	0	0

**L1** : Default value is 0x0B, express as length of 1st B4~B9 and 2nd B2~B6.

**L2** : Default value is 0x05, express as length of 2nd B2~B6.

**D1,D2** : Dummy Byte.

Host computer send above H->D command packet to touch controller device to query the controller name. The device responses with the data packet like above D→H packet. However, the device responses to this query command with the controller name ASCII string. The length of the controller name is variable.

For example, the packet of controller name "PCAP7200" is as follows:

1st 0x03, 0x08, 0x0A, 0x0B, 0x45, 0x50, 0x43, 0x41, 0x50, 0x37

2nd 0x03, 0x05, 0x32, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00

### 3 Power Saving Mechanism

EXC7XXX- supports 3 working mode for power saving.

#### Fully working mode:

After reset, the controller module works at full power working state. It needs around 50mA at this mode.

#### Idle mode:

After controller receives a software packet from host computer to request MCU entering idle state, it will send back an idle ready packet data to host computer when it is ready to idle. At idle state, IRQ pin will be released to high state. Host computer can wake up this controller module via generating a falling edge signal at IRQ pin. During idle state, controller scan touch sensor at low speed – around once 100 ms and can be controlled via host computer. Once it detects sensor touched, the controller will back to fully working state automatically. Working at this mode, the power consumption should be around 2mA.

#### Sleep mode:

Whenever the host computer wants to deep sleep, it issues a sleep command packet to controller. Once the controller firmware receives such sleep command, it enters deep sleep state. Only host computer can wake up this device via generating a falling edge signal at IRQ pin. Working at Sleep state, the power consumption should be around or less than 300uA.

The handle and wakeup timing chart is as follows:

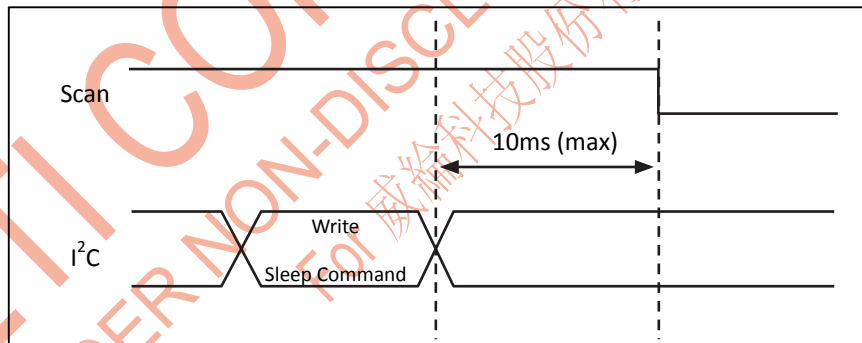


Fig 3-1 Handle Timing

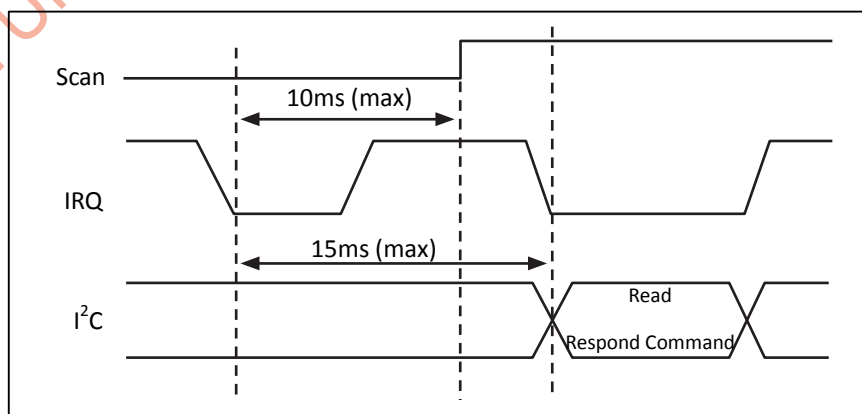


Fig 3-2 Wakeup Timing



eGalax\_eMPIA Technology Inc.

**Headquarters***11F, No 302, Rueiguang Road, Nei Hu District,**Taipei 114, TAIWAN**T: +886 2 8751 5191**F: +886 2 2797 8808***eGalaxTouch****Product Contact**Web Site: [www.eeti.com](http://www.eeti.com)Sales: [touch\\_sales@eeti.com](mailto:touch_sales@eeti.com)FAE: [touch\\_fae@eeti.com](mailto:touch_fae@eeti.com)

EETI (eGalax\_eMPIA Technology Inc.) reserves the right to modify revise or amended this document and/or the content, material, or specification of product at any time without prior notice. EETI takes no responsibility for, and will not be liable for, this document or related information about the suitability or availability being use to the non-EETI's product and using the EETI's product will involve the EETI's software license which including but not limited to source code, program or firmware and is authorized for EETI's product only.

**Disclaimer:**

UNLESS HAVE THE PRIOR NOTICE BY EETI, EETI DOES NOT RECOMMEND THE USE OF ANY OF ITS PRODUCTS IN MEDICINE, MAINTAIN IN HEALTH, EMERGENCY OR OTHER LIFE SUPPORT APPLICATIONS WHERE THE FAILURE OR MALFUNCTION OF THE PRODUCT CAN REASONABLY BE EXPECTED TO CAUSE FAILURE OF A LIFE-SUPPORT SYSTEM OR TO SIGNIFICANTLY AFFECT ITS SAFETY OR EFFECTIVENESS. EETI Products are not authorized for use in such applications as above, so anyone who violate this will bear strictly at your own risk and make representations of this.