Synaptics

Synaptics TouchComm Protocol Reference Manual

PN: 511-000746-01 Rev L

Copyright

Trademarks

Notice

Revision History

Revision	Description
A	Release to Production
B	Clarified meaning of “Maximum WRITE FLASH data size in bytes” contents in Table 8 Boot Information Packet Version 1. Updated response payload contents for byte offset 1 in Table 8. In Table 7, updated IDENTIFY packet version to 1.
C	Updated boot configuration/customer OTP descriptions. Updated HOST DOWNLOAD and HOST DOWNLOAD STATUS version payload and field information. Added force measurement entity ($1C) to entity code table. Added status code ($0B) to Command response status code table. Added TDDI-related clarifications.
D	Updated GET DYNAMIC CONFIG and SET DYNAMIC CONFIG command descriptions. Included number of force electrodes in app info packet and revised description of GET APP INFO. Added force electrode info to DELTA DATA and RAW DATA reports. Added CALIBRATE command.
E	Added ROM Bootloader mode value in IDENTIFY packet.
F	Updated ERASE FLASH with the 4-byte version of the payload.
G	Added enable face detection parameter to DESCRIBE DYNAMIC CONFIG field ID list.
H	Added open/short tuning data ($05) data type to GET DATA LOCATION command. Added suggested minimum 10 microseconds wait between SPI transactions. Added open/short configuration packet types to HOST DOWNLOAD command packets. Added hovering object classification value in SET REPORT CONFIG command. Added fingerprint area meet entity ($1D) to SET REPORT CONFIG command.
I	Added open/short configuration required to HOST DOWNLOAD STATUS ($IB) report. Added sensing mode entity ($1E) to SET REPORT CONFIG command.
J	Changed DISABLE REPORT command to allow ID 1 to disable all report types.
K	Added inhibit active gesture ($0F) to DESCRIBE DYNAMIC CONFIG field ID list.
L	Updated Table 20 (Gesture IDs) and Table 21 (Gesture data entity contents).

Contents

1. Introduction

Synaptics® TouchComm is a packet-based data protocol that is used by Synaptics touch (and touch/display) controller products for communications with the host system, typically the main system processor. Synaptics TouchComm-enabled devices use I^{2}C or SPI as the physical layer, and are slave devices on either bus.

Hosts can configure the contents of the data/report packets sent by the device so that only the data of interest needs to be read.

Communication consists of *messages* that are either *read* or *write* messages, from the perspective of the host. Host-to-device writes are *commands*, while device-to-host reads are either *responses* to commands from the host or device-generated *reports*.

Synaptics TouchComm allows only one pending command at a time. The host must wait for a response for each command before sending the next command to the device.

Reports are generated whenever the device has something to say. A device can generate multiple different types of reports depending on its features or modes. This makes Synaptics TouchComm more complicated than a pure command-response protocol, but overcomes some limitations of a single-master bus. More importantly, when a touch report is ready, the master does not have to perform a separate bus transaction to command the device to send a report before reading the report. This reduces the overall touch latency in the system.

1.1. Typical Usage

To use Synaptics TouchComm, a host reads reports from the device whenever they are available. For a typical touch controller in a host system, the device produces touch reports at regular intervals whenever fingers are touching the sensor.

To influence the behavior of the device, the host sends commands to the device, then waits for a response. After a command is sent, it is *pending* until the host reads a response message. Response messages can indicate success or failure of the command.

While a command is pending, the host may still receive reports from the device. Any reports read while a command is still pending may or may not have the command applied to them. For example, if the host commands the device to switch sensing modes, reports read before the response may have been generated in the previous mode. After the response is read, the host knows that future reports will be in the new mode.

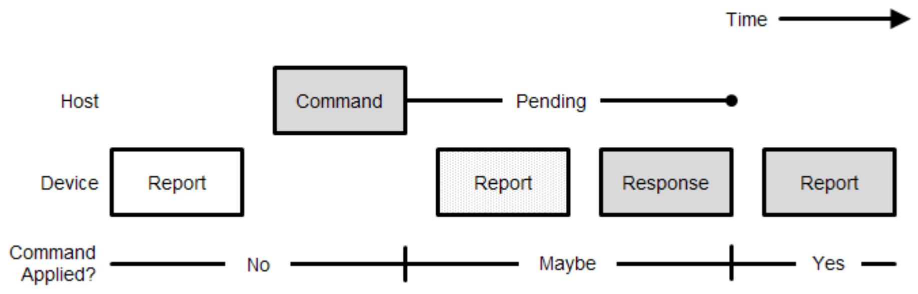


Figure 1. Pending commands

In the example in Figure 1, the host first reads a report from the device. Then, the host writes a command. While that command is pending, the host reads another report from the device. This report may or may not have the command applied. Next, the host reads a response, and the next report the host reads has the command applied.

When no commands are pending and no reports are available, the device responds to reads with an *idle* message. This is most important when the device is used in polling mode.

1.2. Byte Ordering

Some quantities in Synaptics TouchComm messages span multiple bytes. These are always sent least-significant byte first. The bit ordering within each byte is determined by the underlying bus protocol.

1.3. Conventions in This Document

Bits are numbered such that bit 0 is the least significant bit of a byte or other quantity. Subfields of bits are denoted *n*:*m* for the bits *n* down to *m*, inclusive. For example, bits 7:4 comprise the most significant four bits of an 8-bit value. Square brackets are used to indicate the larger value to extract bits from. For example, if LENGTH is a 16-bit value, LENGTH [15:8] denotes a value consisting of the most significant 8 bits of LENGTH.

Values in this document prefixed with a dollar sign ($) are hexadecimal numbers. Other values are decimal numbers unless otherwise noted.

Bits within a byte and bytes within a larger entity are numbered from 0, so the bytes in an 8-byte message are numbered 0-7.

2. Firmware Structure

2.1. Program Memory Organization

As shown in Figure 2, there are four district areas of program memory, normally non-volatile/flash memory:

- Bootloader code
- Bootloader configuration: Includes communication settings such as I^2C address and ATTN polarity.
- Application code: Starts at a fixed address; resizable through a short header.
- Application data: Includes static configuration (all of the settings used to customize or tune a device that do not change at run time), dynamic configuration (settings that can change at run time without stopping touch sensing), and touch report configuration (specifies what is sent in the touch report).

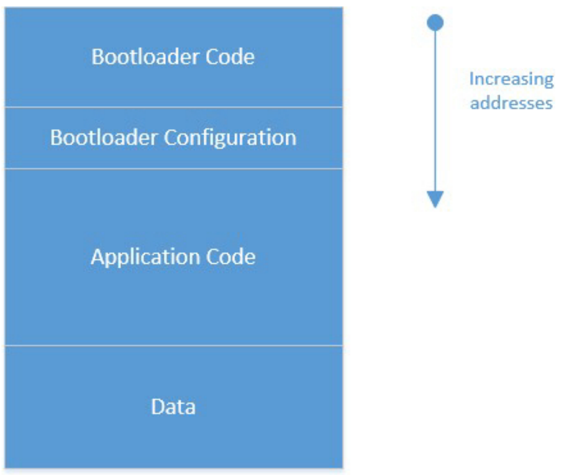


Figure 2. Program Memory Organization

2.2. Bootloader

The bootloader, TouchBoot, has its own command set. The commands do not share the same codes as application firmware because the bootloader mode and application mode coexist on the same device. They use non-overlapping command sets to prevent hosts from accidentally modifying the flash memory contents.

TouchBoot works with four different data sizes:

- Bytes are 8-bit quantities. These are used to specify the length of a read.
- Words are 16-bit quantities. Device addresses refer to word locations.
- Write blocks are the smallest unit of flash that can be written. The size of a write block in words is specified in the boot information packet, but is usually 2 or 4. Write blocks must begin on addresses that are multiples of the write block size.

- Erase pages are the smallest unit of flash that can be erased. The size of an erase page in words is specified in the boot information packet, but is usually 256 or 1024. Erase pages must begin on addresses that are multiples of the erase page size.

Each of the four regions of flash memory has different access rights to TouchBoot.

Table 1. Access rights

Name	Readable	Writable	Erasable
Bootloader code	No	No	No
Bootloader configuration/customer OTP	Yes	Yes	No
Application code	No	Yes	Yes
Application data	Yes	Yes	Yes

These regions are listed from lowest to highest address. They are contiguous, and their extents can be determined from information in the boot information packet.

TouchBoot uses 16-bit word addresses. This means that address 0 contains the first 16-bit value, address 1 contains the next 16-bit value, and so on. Bytes are not individually addressable.

The bootloader code and bootloader configuration/customer OTP regions have fixed sizes. The application code begins at a fixed location for each device but its size is variable. The application data consists of all writeable flash memory at addresses higher than the application code region.

Reading, writing, and erasing are performed by address. The image file format specifies values to program at each address, and the reflash software simply sends commands to perform erases and writes at the specified addresses. The only constraint is that the first write block of the application code region must be written before any other because it defines the size of the application code region.

2.3. Bootloader Configuration/Customer OTP

The bootloader configuration/customer OTP is a small region of flash that can hold communications protocol settings and a small amount of customer data. It can be written, but not erased.

The bootloader configuration/customer OTP area is divided into several slots of four 16-bit words each. The starting address and number of slots can be determined with the GET BOOT INFO command. Each slot can contain a bootloader configuration structure. At power-on or reset, the device uses the configuration in the slot with the highest address. A filled slot always has bit 15 of the lowest word set to 1 (or set to 0 for TDDI products because they use external flash and the erase operation results in ones being stored for erased data). Because the slots with lower addresses are ignored, they are not limited to storing valid boot configuration structures and can be used for other purposes as desired.

The bootloader configuration structure format depends on whether the device uses the I^{2}C or SPI protocol. The structures are shown here:

Table 2. Bootloader configuration for I^2C devices

Word/bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	1	ATTN pullup	ATTN drive	ATTN polarity	ATTN pin				ATTN enable	I^2C address						
Word 1	Reserved for Customer Use															
Word 2	Reserved															
Word 3	Reserved															

Table 3. Bootloader configuration for SPI devices

Word/bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	1	ATTN pullup	ATTN drive	ATTN polarity	ATTN pin				ATTN enable	Reserved					CPOL	CPHA
Word 1	Reserved for Customer Use															
Word 2	Reserved															
Word 3	Reserved															

The fields with "ATTN" in their name configure the ATTN interrupt pin. They have the following meanings:

Table 4. ATTN fields

Name	Meaning
ATTN enable	0: disable the ATTN interrupt pin 1: enable the interrupt pin
ATTN pin	0-15: Use specified GPIO pin to drive ATTN interrupt. See the device datasheet to determine valid options.
ATTN polarity	0: ATTN interrupt is active-low 1: ATTN interrupt is active-high
ATTN drive	0: ATTN interrupt is a push-pull output 1: ATTN interrupt is an open-drain output
ATTN pullup	0: An external pull-up is provided for ATTN interrupt 1: Use internal pull-up resistor for ATTN interrupt

The "Reserved for Customer" use field can be used by customers for any 16-bit value. The unused boot configuration slots at addresses below the highest boot configuration structure may also be used for four words of customer data each.

The CPOL and CPHA fields for SPI have their customary meanings.

- For CPOL, 0 means the rising edge of SCLK is the leading edge; 1 means the falling edge of SCLK is the leading edge.
- For CPHA, 0 means data transition is on the falling edge of SCLK; 1 means data transition is on the leading edge of SCLK.

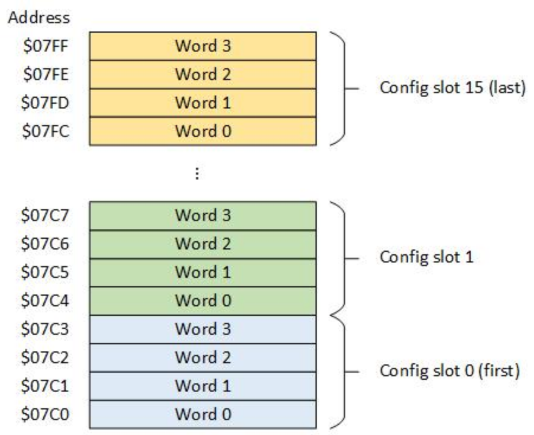


Figure 3 - Bootloader configuration address space

The bootloader configuration may be read/written with the READ FLASH and WRITE FLASH commands. The locations in flash memory for the four bootloader configuration words are able to be written only once. If you try to overwrite with different data, the write command returns an error response.

A bootloader configuration may be 'locked' by writing it at the slot with the highest address. The address organization is shown in Figure 3.

2.4. Application Data

Touch firmware is configured with a configuration ID, a static configuration, a dynamic configuration, and a touch report configuration.

The configuration ID is a 16-byte character string used by customers to label a specific configuration. This allows different configurations to be compared more easily.

The static configuration consists of all the settings used to customize or tune a device and which do not change at run time. It can only be accessed all at once, and touch sensing must restart if it is updated. The contents of the static configuration vary from part to part and are mostly determined by Synaptics tools such as Design Studio™. Hosts treat the static configuration as an opaque binary blob.

The dynamic configuration consists of all the settings that can change at run time without stopping touch sensing. This includes various modes and debugging features. The dynamic configuration is composed of a sequence of 16-bit fields. Random access to individual fields is allowed by specifying a field ID along with the read or write request. The host can query the device to discover the available field IDs in the dynamic configuration on any device, and the meanings of each are available in Design Studio.

The touch report configuration specifies what is sent in the TOUCH report. It is a list of bytecode values that describe how the TOUCH report is assembled and packed into bytes. It is described in more detail in the documentation for the SET REPORT CONFIG command (See 5.4.8.).

The configuration ID, static configuration, dynamic configuration, and touch report configuration are together called the application configuration.

3. Message Formats

Host systems initiate all communication transactions with a Synaptics TouchComm device. A host can:

- Send (write) a command
- Read a command response
- Read a data report

All such read or write packets are of variable length.

All write messages are commands. Reads are either idle messages, command responses, or reports. Reads and writes use similar but slightly different formats.

Message payloads can be split across multiple bus transactions. The first message in such a group uses the normal command code and specifies the full payload length, followed by some portion of the payload. Subsequent messages begin with a continued message code, followed by more of the payload.

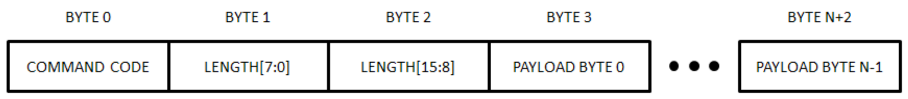
The smallest unit of data transfer is the *bus transaction*. A Synaptics TouchComm message can extend over one or more bus transactions using continued reads and continued writes.

The maximum transfer permitted in one bus transaction can be limited by the host or device. The device's maximum write size for a single bus transaction can vary between bootloader and application mode and is given in the IDENTIFY report (See Section 5.2.1).

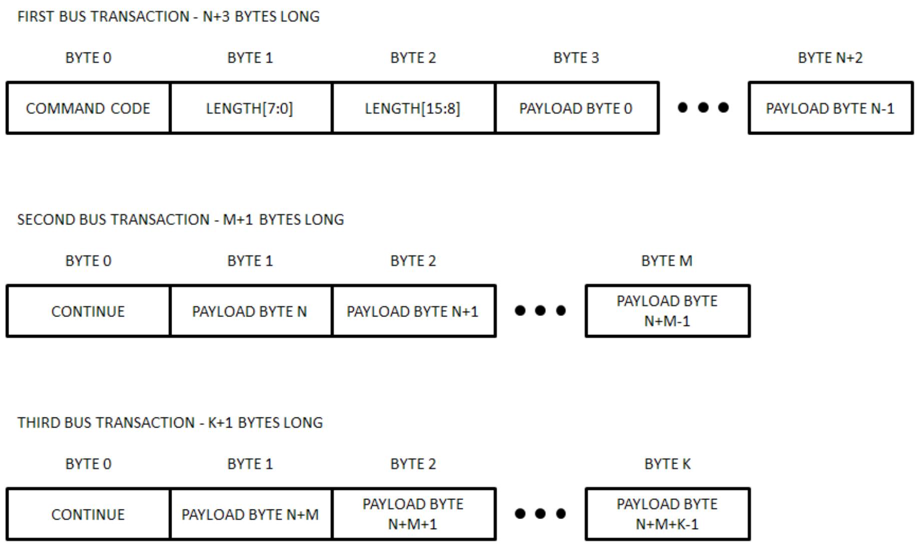
3.1. *Write Messages*

The host sends write messages to the device to control its operation. Write messages consist of an 8-bit command code, followed by a 16-bit payload length in bytes, optionally followed by a payload. If there is no payload, the length can be omitted.

For example, this is the format of a write message N+3 bytes long with an N-byte payload:



Due to limited buffer sizes on the host and device, write messages may be split across multiple bus transactions with the CONTINUE command code for bus transactions after the first one. The length bytes are only included in the first bus transaction. For example, here is a write message split across three transactions:



The device responds to write messages with read messages. A status code in the read message tells the host whether the command succeeded or failed. Until this response is received, the host command is pending and no other commands should be sent.

If the host sends a command other than RESET while a previous command is still pending, the device will not execute the second command, but will instead respond with a PREVIOUS COMMAND PENDING status. After this is read, the response for the original pending command can be read when it is available.

If the host splits a write across multiple bus transactions and performs a read before the complete payload has been sent, the continued write is cancelled. The device does not produce a command response for the write, but otherwise the results of a cancelled write are undefined.

3.2. Read Messages

There are two kinds of read messages: *responses* and *reports*. The device generates responses to host commands, and it generates reports periodically to provide touch data to the host. If the host reads from the device when no response or report is available, the device responds with an *idle* response.

If an ATTN interrupt is available (see section 4 - Physical Layers), the device asserts ATTN whenever it has a read message to send to the host.

Responses and reports have the same format but use a different first byte. Responses use codes $00-$0F, and reports use codes $10-$FE. The value $FF is invalid and indicates that the host should retry the read.

Note that it is possible for the host to receive one or more reports between sending a command and receiving the response for that command.

Each read bus transaction begins with an 8-bit start-of-message marker with value $A5, but this is not considered part of the message. The host can use this byte to determine whether the device is responsive. If the first byte read in a transaction is not $A5, this indicates that the device is unable to communicate for some reason (for example, in the middle of a reset). The host can wait briefly and retry or attempt to recover in some other way.

Read messages consist of an 8-bit status code or report type, and a 16-bit payload length in bytes, optionally followed by a payload. If there is no payload, the length is $0000.

If the host reads beyond the end of the payload, the device sends $5A bytes until the end of the transaction. These are not considered part of the message. If a value other than $5A is seen, the host can assume that the packet is corrupt and should be discarded. This helps detect clock synchronization errors due to glitches or device resets due to Electrostatic Discharge (ESD).

For example, this is the format of a read message N+3 bytes long with an N-byte payload:

BYTE 0	BYTE 1	BYTE 2	BYTE 3		BYTE N+2
STATUS/REPORT CODE	PAYLOAD LENGTH[7:0]	PAYLOAD LENGTH[15:8]	PAYLOAD BYTE 0	• • •	PAYLOAD BYTE N-1

Due to limited buffer sizes on the host, read messages may be split across multiple bus transactions with the CONTINUED READ status code for bus transactions after the first one. The length bytes are only included in the first bus transaction. For example, here is a read message split across three transactions:

FIRST BUS TRANSACTION - N+3 BYTES LONG

BYTE 0	BYTE 1	BYTE 2	BYTE 3		BYTE N+2
STATUS/REPORT CODE	PAYLOAD LENGTH[7:0]	PAYLOAD LENGTH[15:8]	PAYLOAD BYTE 0	• • •	PAYLOAD BYTE N-1

SECOND BUS TRANSACTION - M+1 BYTES LONG

BYTE 0	BYTE 1	BYTE 2		BYTE M
CONTINUED READ	PAYLOAD BYTE N	PAYLOAD BYTE N+1	• • •	PAYLOAD BYTE N+M-1

THIRD BUS TRANSACTION - K+1 BYTES LONG

BYTE 0	BYTE 1	BYTE 2		BYTE K
CONTINUED READ	PAYLOAD BYTE N+M	PAYLOAD BYTE N+M+1	• • •	PAYLOAD BYTE N+M+K-1

A continued read may be cancelled with any command. This may be useful if the host loses synchronization with the device and sees a continued read that it is not expecting. Because the payload length is only included on the first bus transaction, the host would either have to read an unknown number of bytes or cancel the read. The device discards the contents of a cancelled read.

3.2.1. Suggested Read Sequence

Hosts should always read one byte beyond the payload. If this byte is not $5A, the host should discard the packet. If this packet was a command response, the host should resend the command. If it was a report, the host should wait for the next report.

Most hosts cannot adjust the size of a read after it has already begun. Because Synaptics TouchComm packets have variable lengths, the host has two options:

1. Perform a 4-byte read to get the payload length, then perform a payload length + 3-byte read to get the payload and a single trailing $5A byte, or
2. Guess the payload length and perform a payload length + 5-byte read.
    - If the read was too short, perform a second read to get the remaining payload + 3 bytes to get the rest of the payload and a single trailing $5A byte.
    - If the read was too long, discard the trailing $5A bytes.

The touch report payload may have a variable length that depends on the number of fingers. To determine how much to read, the host can guess that each touch report contains the same number of fingers as the previous touch report.

4. Physical Layers

Synaptics TouchComm supports I^{2}C and SPI physical layers. Synaptics TouchComm devices are slaves on the bus, they cannot initiate transactions themselves. Instead, the host must determine when to begin a transaction using either interrupt or polling modes.

4.1. Interrupt Mode

Neither I^{2}C nor SPI defines a mechanism for the device to signal the host. Synaptics TouchComm extends both by adding an optional ATTN signal. This is asserted whenever the device has a response or report to be read. The host should treat this as a level-triggered interrupt and read messages from the device until ATTN is no longer asserted. ATTN polarity, drive strength, and internal pullup options are product-specific.

If the host hardware supports only edge-triggered interrupts, the host software can read the value of ATTN after processing each Synaptics TouchComm message, and continue issuing Synaptics TouchComm reads until ATTN is no longer asserted. Figure 4 shows the procedure.

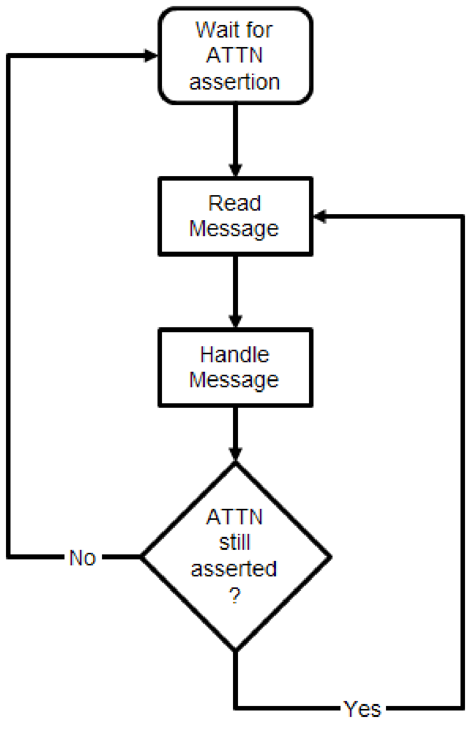


Figure 4. Edge-triggered interrupt procedure

If the host hardware supports level-triggered interrupts, then the final check can be skipped, as shown in Figure 5.

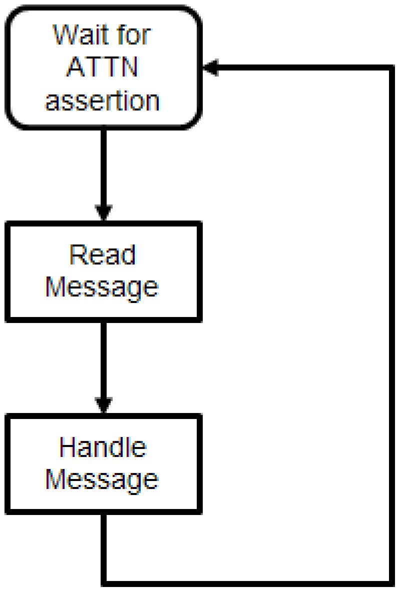


Figure 5. Level-triggered interrupt procedure

4.2. Polling Mode

Because the ATTN signal requires an extra wire from device to host, and this is not always available, hosts may use polling instead to determine when a message is ready. The host reads *idle* responses until a report or command response is available.

Polling requires more power than interrupt-driven operation. The interval at which the host polls must balance the need for latency with power usage. Also, polling the device too often creates processing overhead on the device, reducing the amount of time that it can use to process touch data. With these issues in mind, a reasonable polling interval is 1 ms or greater.

4.3. I^{2}C

Synaptics TouchComm uses three signals for I^{2}C communications:

- SCL: the serial clock
- SDA: the serial data
- ATTN: optional device-to-host interrupt line, usually active low

Synaptics TouchComm over I^{2}C uses the normal I^{2}C 7-bit addressing and transaction direction mechanisms to select the device and distinguish between reads and writes.

A bus transaction consists of the data transferred between the bus address byte and a STOP or REPEATED START condition.

If the host performs several transactions without adequate time between them, the device may stretch the clock until it is ready to handle the transaction. Clock stretching only occurs near the beginning of a transaction addressed to the Synaptics TouchComm device. Synaptics TouchComm devices never stretch the clock during transactions to other devices. Clock stretching can be avoided entirely if the host allows sufficient time.

Figure 6 shows the electrical connection between host and device.

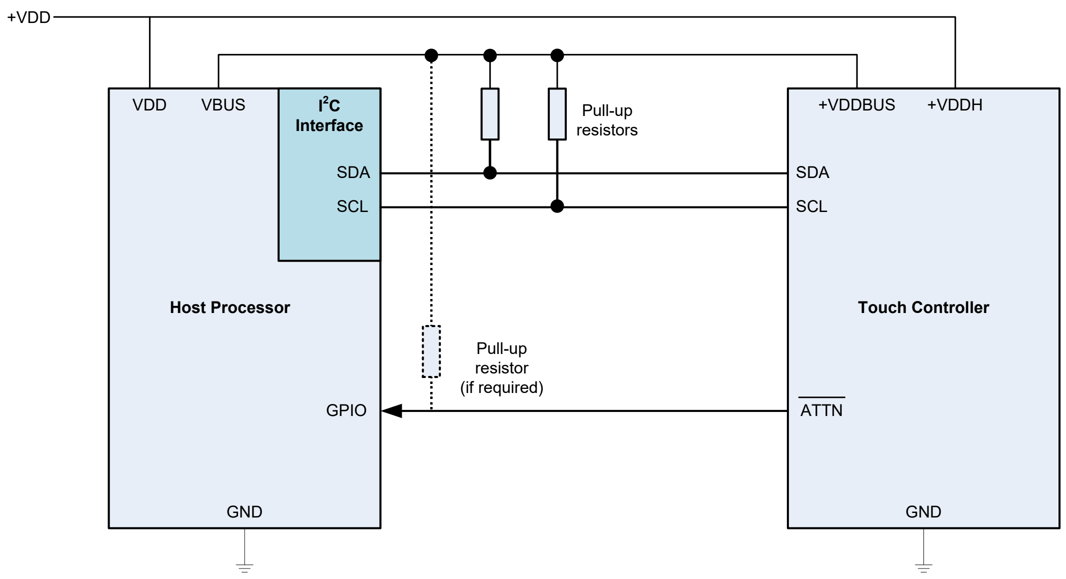


Figure 6. I²C host/device connection

Figure 7 shows real bus waveforms of the start of a touch report packet being read by the host.

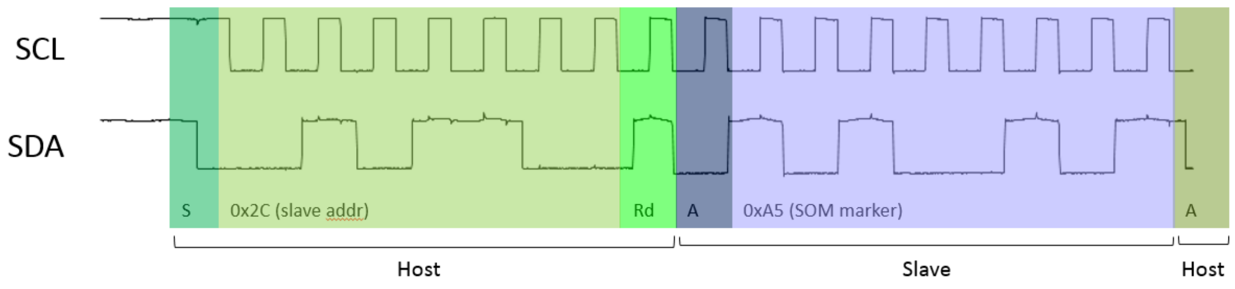


Figure 7. I²C waveforms example

4.4. SPI

Synaptics TouchComm uses five signals for SPI communication:

- SCLK: the serial clock
- MISO: master in, slave out data
- MOSI: master out, slave in data
- SSB: slave select, active low
- ATTN: optional device-to-host interrupt line, usually active low

The host is required to assert the SSB signal to address the device. A bus transaction consists of the data transferred between SSB assertion and SSB deassertion.

Synaptics TouchComm uses the SPI bus in half-duplex mode: each bus transaction is either a read or a write, but not both. If the first byte of a transaction begins with a value other than $00 or $FF on MOSI, that value is a command code and the transaction is a write transaction. Otherwise, the transaction is a read transaction.

Because the device must prepare to send bytes before it can determine whether a particular transaction is a read or write transaction, it may send several bytes worth of data on MISO during a write transaction. These should be ignored by the host.

If the host performs several bus transactions back-to-back without adequate time between them, the device may ignore write messages or send INVALID read messages. An INVALID read message either has an incorrect beginning-of-message marker or uses the INVALID response code. Usually 10 microseconds is a sufficient time to wait between SPI transactions, but consult the device datasheet for a more precise number.

Figure 8 shows the electrical connection between host and device.

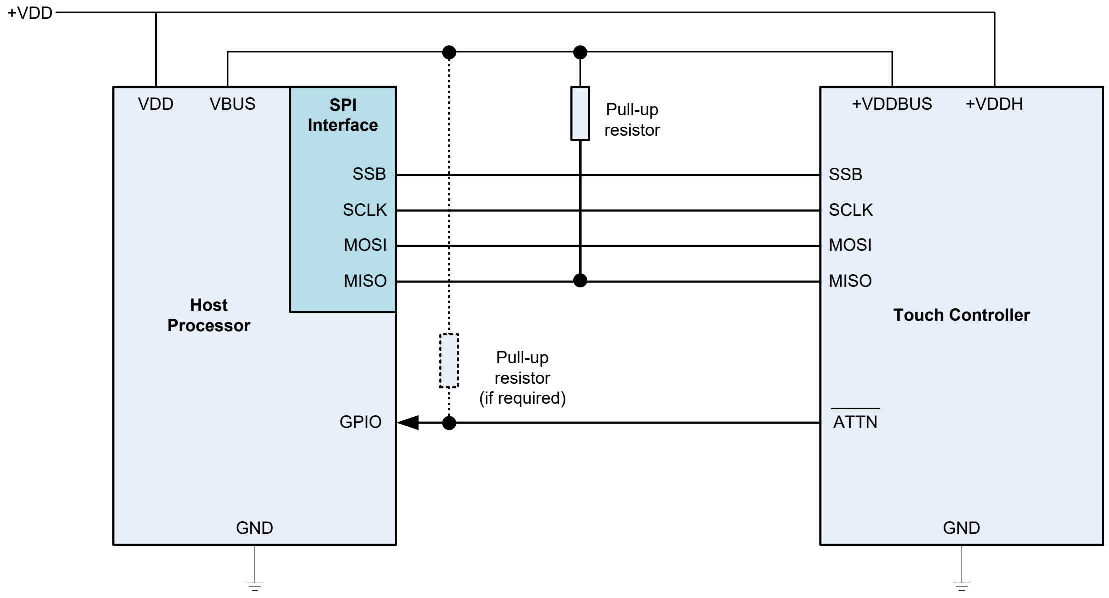


Figure 8. SPI host/device connection

Figure 9 shows real bus waveforms of the start of a touch report packet being read by the host.

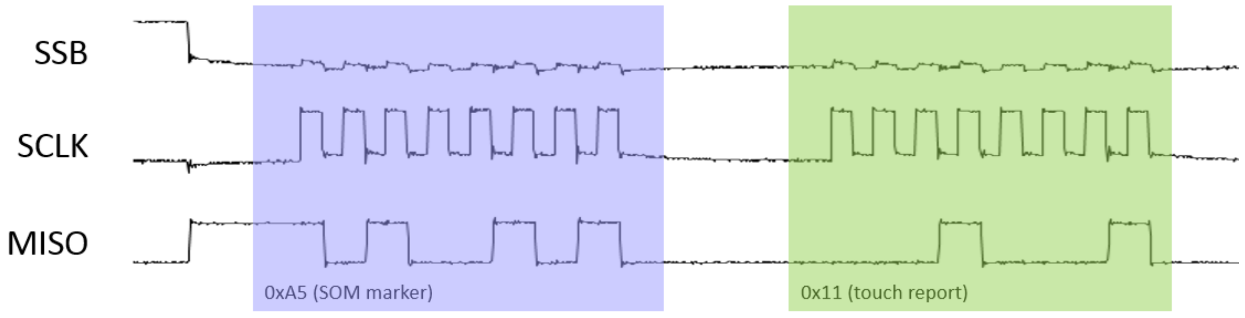


Figure 9. SPI waveforms example (mode 0)

5. Commands and Responses

Command packets have the following structure:

Table 5. Command packet structure

Byte	**1**	**2**	**3**	**…**	**N**
Contents	Command code	Payload	Payload	…	Payload

5.1. Command Response Codes

Synaptics TouchComm uses the same set of status codes for all command responses. Values not in Table 6 are reserved for future use.

Table 6. Command response status codes

Status code		**Name**	**Payload**	**Meaning**
$00		IDLE	No	No commands are pending and no reports are available.
$01		OK	Some OK responses have payloads. For example, the payload of the response to a successful Read Flash command will be the contents of the flash memory at the requested addresses.	The last command completed successfully.
$03		CONTINUED READ	Yes	This read transaction continues the current read message.
$0B		NOT EXECUTED IN DEEP SLEEP	No	A command was received that cannot be processed by the system during deep sleep.
$0C		RECEIVE BUFFER OVERFLOW	No	The previous write bus transaction was too large for the device. The host should resend the command using smaller bus transactions.
$0D		PREVIOUS COMMAND PENDING	No	A command was received when a previous command was still pending.
$0E		NOT IMPLEMENTED	No	The last command is not implemented on this device.
$0F		ERROR	No	The last command did not complete successfully.
$FF		INVALID	No	The packet is invalid and the read should be retried.

5.2. Standard Commands

The standard commands are implemented by all Synaptics TouchComm devices, in either bootloader or application firmware mode.

5.2.1. IDENTIFY

- Command Code: $02
- Payload: None
- Response: Identify Packet

The IDENTIFY command requests an identify packet. The first byte of this packet is a version number that specifies the format of the rest of the packet. The formats for each version are below.

5.2.1.1. Identify Packet Version 1

Table 7. Identify packet contents

Byte offsets	Contents
0	Identify packet version = 1
1	Firmware mode 1: Application firmware 2: Host download firmware 4: ROM Bootloader 11: Bootloader 12: TDDI bootloader 13: TDDI host download bootloader 14: Production test firmware
2-17	Part number string (null-terminated only if less than 16 characters long). The convention is to use *product-version* where *version* refers to the Design Studio software target version. Example: "S3708-10.0" indicates product name S3708, v10.0.
18-21	Firmware build ID
22-23	Maximum write size in bytes

5.2.2. RESET

- Command Code: $04
- Payload: None
- Response: Status (see Table 6) on failure, none on success.

The RESET command resets the device, returning it to the current mode (bootloader or application). An IDENTIFY report is sent after the reboot is complete.

After sending this command, it may take up to 200 ms before the IDENTIFY packet is available to be read. The reset may cause transitions on ATTN before the IDENTIFY packet is ready and these may be handled in the usual way by reading and checking for a leading $A5 byte to indicate a valid message.

5.3. Bootloader Commands

5.3.1. GET BOOT INFO

- Command Code: $10
- Payload: None
- Response: Boot information packet

The GET BOOT INFO command requests a boot information packet. The first byte of this packet is a version number that specifies the format of the rest of the packet. The formats for each version are below.

5.3.1.1. Boot Information Packet Version 1

Table 8. Version 1 response payload contents

Byte offsets	Contents
0	Boot information packet version = 1
1	Status 0 - OK 1 - Booting; 2 - Bootloader request 254 - Bad application code; 255 - Warm boot
2-3	Reserved
4	Write block size in 16-bit words
5-6	Erase page size in 16-bit words
7-8	Maximum WRITE FLASH data size in bytes. (If larger than “Maximum write size in bytes” field of IDENTIFY packet, then write must be split into multiple bus transactions.)
9	Reserved
10-11	Reserved
12-13	Starting write block of boot configuration/customer OTP region
14-15	Size of boot configuration/customer OTP region in write blocks

5.3.1.2. Boot Information Packet Version 2

Table 9. Response payload contents same as version 1 but with changes

Byte offsets	Contents
0	Boot information packet version = 2
1-15	Same as boot information packet version 1
16-19	Display configuration starting write block
20-21	Display configuration length in write blocks
22-25	Backup display configuration starting write block
26-27	Backup display configuration length in write blocks
28-29	Customer OTP region starting address in write blocks
30-31	Customer OTP region length in write blocks

The customer OTP values define the region being used that is part of the boot configuration/customer OTP area (defined in bytes 12-15). If there is no customer OTP, these values report as zero.

5.3.2. ERASE FLASH

- Command Code: $11

Table 10. Command payload contents

Byte offsets	Contents
0	First erase page to erase
1	Number of erase pages to erase

or

Byte offsets	Contents
0-1	First erase page to erase
2-3	Number of erase pages to erase

- Response: Status only (see Table 6)

The ERASE FLASH command erases the specified blocks of flash memory. An error status code is returned if any of the blocks are outside the erasable range. It is undefined whether any blocks are erased in this case.

If any of the specified blocks are in the application code region, the entire application code region is erased.

Some parts may support only the 2-byte version of the payload. To work around this, it is recommended that the host use the 2-byte version of the payload if the first erase page and number of erase pages are both less than 256, and it should otherwise use the 4-byte version.

While this command is executing, the device may be unresponsive. The time taken is product-dependent.

5.3.3. WRITE FLASH

Command Code: $12

Table 11. Command payload contents

Byte offsets	Contents
0-1	First write block to write
2-end	Data to write

The WRITE FLASH command writes the specified data to flash memory. An error status code is returned if any of the blocks are outside the writable range or if any blocks to be written are not blank. It is undefined whether any blocks are written in this case.

If the length of the data to write is not an integer multiple of words, the trailing byte is discarded. If it is not an integer number of write blocks, it is zero-padded to the next write block.

For discrete products, if any write block is requested to be written with all zero values, it is left blank so that it can be programmed later. Conversely for TDDI, any block of ones written is still considered blank so that it can be programmed later.

While this command is executing, the device may be unresponsive. The time taken is product-dependent.

5.3.4. READ FLASH

- Command Code: $13

Table 12. Command payload contents

Byte offsets	Contents
0-3	First word address to read
4-5	Number of 16-bit words to read

- Response: Requested memory

The READ FLASH command reads the specified data from flash memory. For discrete products, reads to the protected bootloader code or application code areas are read as 0. For TDDI, the firmware returns the encrypted data for the application code rather than zeros. If the number of bytes requested is too large, it may be truncated to an implementation-defined maximum read size.

5.3.5. RUN APPLICATION FIRMWARE

- Command Code: $14
- Payload: None
- Response: Status on failure, none on success

The RUN APPLICATION FIRMWARE command requests that the application firmware be run. An error is returned if the application code is not valid for any reason.

While starting the application firmware, the device may become unresponsive briefly. After the application firmware has finished starting, the device sends an IDENTIFY report to indicate that it is ready to receive commands in the new mode.

5.3.6. SPI MASTER WRITE THEN READ

- Command Code: $15
- Payload: None

Table 13. Command payload contents

Byte offset	Contents
0	SPI clock divider
1	Bit 6: CPHA Bit 7: CPOL
2-3	Number of bytes to read
4-end	Values to write

- Response: Status on failure, none on success

The SPI MASTER WRITE THEN READ command sends a command to the SPI flash chip connected to a TDDI chip's SPI master port.

"SPI clock divider" sets the SPI clock rate. The rate is f/(2 - (SPI Clock Divider + 1)), where f is the master clock frequency specified in the part datasheet. "CPOL" sets the polarity of the SPI clock. "CPHA" sets the phase of the SPI clock.

5.3.7. ENTER ROM BOOTLOADER MODE

- Command Code: $16
- Payload: None
- Response: Status on failure, none on success

The ENTER ROM BOOTLOADER MODE command forces the device to restart in ROM bootloader mode. On failure, an error status is returned. On success, there is no response packet.

If the ROM bootloader uses the Synaptics TouchComm protocol, it sends an IDENTIFY report when the ROM bootloader is ready to receive commands. If the ROM bootloader implements the RMI4 protocol, refer to the RMI4 documentation for information about how to determine when the ROM bootloader is ready to receive commands.

5.4. Application Firmware Commands

5.4.1. ENTER BOOTLOADER MODE

- Command Code: $1F
- Payload: None
- Response: Status on failure, none on success

The ENTER BOOTLOADER MODE command requests that the device restart in bootloader mode. This occurs immediately without a response packet.

While starting the bootloader, the device may become unresponsive briefly. After the bootloader has finished starting, the device sends an IDENTIFY report to indicate that it is ready to receive commands in the new mode.

After sending this command, it may take up to 200ms before the IDENTIFY packet is available to be read. The reset may cause transitions on ATTN before the IDENTIFY packet is ready and these may be handled in the usual way by reading and checking for a leading $A5 byte to indicate a valid message.

5.4.2. GET APP INFO

- Command Code: $20
- Payload: None
- Response: App info packet

The GET APP INFO command requests an app info packet. The first word of this packet is a version number that specifies the format of the rest of the packet. The formats for each version are below.

The response from the device may not contain the full packet. The host should assume that all missing fields are zero.

5.4.2.1. App Info Packet Version 1

Table 14. Version 1 response payload contents

Byte offsets	Contents
0-1	App info packet version = 1
2-3	Status 0 - OK 1 - Booting 255 - Bad application configuration
4-5	Static configuration length in bytes
6-7	Dynamic configuration length in bytes
8-9	Application configuration starting write block
10-11	Application configuration length in bytes
12-13	Touch report configuration maximum length in bytes
14-15	Touch report payload maximum length in bytes
16-31	Customer configuration ID
32-33	Maximum reported X coordinate
34-35	Maximum reported Y coordinate
36-37	Maximum number of reported objects
38-39	Number of reported buttons
40-41	Number of sensor rows
42-43	Number of sensor columns
44-45	Has profiles
46-47	Number of force electrodes

5.4.3. GET STATIC CONFIG

- Command Code: $21
- Payload: None
- Response: Static configuration

The GET STATIC CONFIG command returns the contents of the static configuration.

5.4.4. SET STATIC CONFIG

- Command Code: $22
- Payload: Static configuration
- Response: Status

The SET STATIC CONFIG command sets the contents of the static configuration. When the write is completed, the device restarts touch sensing with the new settings. If the device is currently reporting any objects, they stop being reported until they have moved.

5.4.5. GET DYNAMIC CONFIG

- Command Code: $23
- Payload: 1-byte field ID
- Response: Field value (or entire dynamic configuration if field ID = 255).

The GET DYNAMIC CONFIG command gets a single field or the entire dynamic configuration. If the field ID is 255, the entire dynamic configuration is returned. Otherwise, the 16-bit value of the field with the specified ID is returned. The IDs for each field are given in the DESCRIBE DYNAMIC CONFIG command.

5.4.6. SET DYNAMIC CONFIG

- Command Code: $24
- Payload: One of the following:

Table 15. Primary command payload contents

Byte offset	Contents
0	Field ID (!= 255)
1-2	Field Value

Table 16. Alternate command payload contents

Byte offset	Contents
0	255
1-end	Dynamic configuration

- Response: Status

The SET DYNAMIC CONFIG command sets either a single field or the entire dynamic configuration. If the field ID is 255, the entire dynamic configuration is set. Otherwise, the selected field is set to the specified 16-bit value. The IDs for each field are given in the DESCRIBE DYNAMIC CONFIG command.

5.4.7. GET REPORT CONFIG

- Command Code: $25
- Payload: None
- Response: TOUCH report configuration

The GET REPORT CONFIG command gets the current TOUCH report configuration.

The TOUCH report configuration describes how TOUCH reports are formatted. The host driver can use this to parse the contents of TOUCH report packets. Its contents are described with the SET REPORT CONFIG command.

5.4.8. SET REPORT CONFIG

- Command Code: $26
- Payload: TOUCH report configuration
- Response: Status

The SET REPORT CONFIG command sets the current TOUCH report configuration.

The TOUCH report configuration describes how TOUCH reports are formatted. The host can set this to specify exactly what items get reported each time a TOUCH report is generated.

The format of the TOUCH report configuration is a list of 8-bit codes. Codes specify either control flow operations or data entities. Data entities are followed by an 8-bit value specifying how many bits the entity occupies. Data types for each data entity are unsigned integers unless otherwise indicated. The core set of codes is as shown in Table 17. Products may define additional codes and these can be determined with the Design Studio tool.

Table 17. Control flow codes

Control	Value
$00	End report
$01	Begin for each active object loop
$02	Begin for each object loop
$03	End for each loop
$04	Zero-pad to next byte boundary. If already at a byte boundary, do nothing.

Table 18. Entity codes

Code	Value
$05	Timestamp (natively 32 bits)
$06	Current object index (natively 4 bits)
$07	Current object classification (natively 4 bits)
$08	Current object X position in display pixels (natively 16 bits)
$09	Current object Y position in display pixels (natively 16 bits)
$0A	Current object Z (natively 16 bits)
$0B	Current object X width (natively 8 bits)
$0C	Current object Y width (natively 8 bits)
$0D	Current object TX position in sensor pixels (natively 16 bits)
$0E	Current object RX position in sensor pixels (natively 16 bits)
$0F	0D buttons state (native width 1 bit per button, number of buttons is in app info packet)
$10	Gesture ID (natively 8 bits, as described in Table 20)
$11	Frame rate (natively 8 bits)
$16	Sensing frequency index (natively 4 bits)
$18	Number of active objects
$1A	Face detected (natively 1 bit)
$1B	Gesture data (natively 16-64 bits, as described in Table 21)
$1C	Force measurement (natively 16 bits)
$1D	Fingerprint area meet (natively 8 bits)
$1E	Sensing mode (natively 8 bits)

Loops can iterate over all objects or only active objects. Active objects are any with a non-zero classification field. Inside of a loop "current object" entities use the current object. Outside of a loop, they use an arbitrary object.

When looping over active objects only, the size of the TOUCH report payload can change size from frame to frame. The host has to infer the number of objects being reported by subtracting the length of the fixed-size elements from the total payload length and dividing by the length of each touch object's data. This is easiest if the ‘for each’ loop is always at the end of the touch report.

The TOUCH report configuration must end with an “End report” code or the SET REPORT CONFIG command returns an error status.

Object classifications are assigned as shown in Table 19.

Table 19. Object classifications

Value	Object Type
0	No object
1	Finger
2	Glove
3	Stylus
4	Eraser
5	Small object
6	Palm
7	Unknown
8	Edge touch
9	Hovering object

Note that not all products will support all of these classifications.

Gesture IDs are assigned as shown in Table 20. A gesture drawn in the shape of a letter receives the ASCII code for that letter.

Table 20. Gesture IDs

Gesture	ID
0	No gesture detected
1	Double tap
2	Swipe
3	Triangle
4	Circle
5	Vee
6	Active Single Tap
7	Active Tap and Hold
8	Active Double Tap
9	Active Early Tap
10	Active Flick
11	Active Press
12	Active Pinch
13	Active Rotation
99	c
101	e
109	m
115	s
119	w
122	z

The contents of the gesture data entity depend on which gesture is detected.

Table 21. Gesture data entity contents

Gesture	Bytes	Contents
Double Tap, Active Single Tap, Active Tap and Hold, Active Double Tap, Active Early Tap	0-1	Tap X position
	2-3	Tap Y position
Swipe	0-1	Swipe X length
	2-3	Swipe Y length
	4-5	Swipe direction 1: +X 2: -X 4: +Y 8: -Y
Active Press	0-1	Press X position
	2-3	Press Y position
Active Pinch	0-1	Pinch finger count
	2-3	Pinch delta position
Active Flick	0-1	Flick finger count
	2-3	Flick X axis delta position
	4-5	Flick Y axis delta position
	6-7	Flick time
Active Rotation	0-1	Rotation direction Positive: clockwise Negative: counterclockwise
	2-3	Rotation distance
Triangle, Circle, Vee, c, e, m, s, w, z	0-1	Gesture speed

Table 22 shows an example touch report configuration.

Table 22. Touch report configuration example

Byte Offset(s)	Contents	Description
0	$01	For each active index...
1-2	$06 $04	Object index, 4 bits
3-4	$07 $04	Object classification, 4 bits
5-6	$08 $0C	Object X position, 12 bits
7-8	$09 $0C	Object Y position, 12 bits
9-10	$0A $08	Object Z, 8 bits
11	$03	…end loop
12	$00	End report

The values are packed in the order the codes are specified, from least to most significant bit. Values can span bytes. This touch report configuration would produce a touch report packet payload encoded as shown in Table 23.

Table 23. Touch report packet payload

	Bits		Object
Byte	**7 6 5 4**	**3 2 1 0**	
0	Classification	Index	1
1	Tx Position<7:0>		
2	Rx Position<3:0>	Tx Position<11:8>	
3	Rx Position <11:4>		
4	Z		
5	Classification	Index	2
6	Tx Position<7:0>		
7	Rx Position<3:0>	Tx Position<11:8>	
8	Rx Position<11:4>		
9	Z		
…	… And so on, for all the active objects….		…

5.4.9. REZERO

- Command Code: $27
- Payload: None
- Response: Status

A REZERO command forces the device to rezero its baseline estimate.

5.4.10. COMMIT CONFIG

- Command Code: $28
- Payload: 16-byte customer configuration ID
- Response: Status

The COMMIT CONFIG command writes the current configuration ID, touch report configuration, static configuration, and dynamic configuration to flash memory to be loaded the next time the device powers on or resets.

The customer configuration ID specified must match the one in the app info packet (set by the SET CONFIG ID command). This is a safeguard to ensure that the flash is not accidentally written to.

While this command is executing, the device may be unresponsive. The maximum length of unresponsive time is listed in the device's data sheet.

5.4.11. DESCRIBE DYNAMIC CONFIG

- Command Code: $29
- Payload: None
- Response: Dynamic configuration description

The DESCRIBE DYNAMIC CONFIG command returns a list of 1-byte codes specifying the order in which fields appear in the dynamic configuration. Fields not in the list are not supported by the device.

Table 24. Possible field IDs

ID	Parameter
$01	Disable doze
$02	Disable noise mitigation
$03	Disable frequency shifting
$04	Requested sensing frequency index (used only if sensing frequency shifting is disabled)
$05	Disable H-sync
$06	Enable rezero when exiting deep sleep
$07	Enable charger mode
$08	Disable baseline relaxation
$09	Enable wakeup gesture mode
$0A	Number of fake fingers to report (for testing)
$0B	Enable grip suppression
$0C	Enable thick glove support
$0D	Enable glove support
$0E	Enable face detection
$0F	Inhibit active gesture

Field IDs greater than 192 are reserved for product-specific dynamic configuration options.

5.4.12. PRODUCTION TEST

- Command Code: $2A
- Payload: 1-byte test ID
- Response: Test result

The PRODUCTION TEST command requests the device to run a production test. Production test IDs and result formats are documented in the Design Studio tool.

5.4.13. SET CONFIG ID

- Command Code: $2B
- Payload: 16-byte config ID
- Response: Status

The SET CONFIG ID command sets the configuration ID to the specified string.

5.4.14. ENABLE REPORT

- Command Code: $05
- Payload: 8-byte REPORT ID
- Response: Status

The ENABLE REPORT command enables a report that was previously disabled. The device does not generate any disabled reports.

5.4.15. DISABLE REPORT

- Command Code: $06
- Payload: 8-byte REPORT ID to disable a single report or 8-bit value 1 to disable all reports
- Response: Status

The DISABLE REPORT command disables a report. The device does not generate reports that are disabled. For example, the TOUCH report can be disabled when communicating with the device so that the host does not have to process TOUCH reports.

The IDENTIFY report sent at startup or after reset cannot be disabled. It and the TOUCH report are always enabled at startup or reset.

5.4.16. ENTER DEEP SLEEP

- Command code: $2C
- Payload: None
- Response: Status

The ENTER DEEP SLEEP command disables touch sensing and puts the device into the low-power deep sleep mode.

5.4.17. EXIT DEEP SLEEP

- Command code: $2D
- Payload: None
- Response: Status

The EXIT DEEP SLEEP command takes the device out of the low-power deep sleep mode and enables touch sensing.

5.4.18. GET TOUCH INFO

- Command code: $2E
- Payload: None
- Response: Touch info packet

The touch info packet varies from part to part and its contents will be specified in the device's datasheet.

5.4.19. GET DATA LOCATION

- Command code: $2F
- Payload:

Table 25. GET DATA LOCATION payload

Byte offset	Contents
0	Data type code, from list below

- Response: Either ERROR if the data type is not supported, or the following response packet:

Table 26. GET DATA LOCATION response packet

Byte offsets	Contents
0-1	Start address, in write blocks
2-3	Length in write blocks

The GET DATA LOCATION command retrieves the address and length of the specified data type.

Table 27. Valid data type codes

Code	Data type
$01	LCM data
$02	OEM data
$03	PPDT data
$04	Force calibration data
$05	Open/short tuning data
$C0-$FF	Reserved for per-product use

5.4.20. HOST DOWNLOAD

- Command code: $30
- Payload:

Table 28. HOST DOWNLOAD version 1 payload

Byte Offsets	Contents
0	Host download version (must be 1)
1	Host download packet type: 0 – invalid 1 – touch configuration 2 – display configuration to PMEM 3 – display configuration to RAM 4 – open/short configuration
2-end	Data

Table 29. HOST DOWNLOAD version 2 payload

Byte Offsets	Contents
0	Host download version (must be 2)
1	Host download packet type: 0 – invalid 1 – touch configuration 2 – display configuration 3 – open/short configuration
2-end	Data

- Response: Status

The HOST DOWNLOAD command supplies configuration data to the device. It is used along with the HOST DOWNLOAD STATUS report.

5.4.21. ENTER PRODUCTION TEST MODE

- Command Code: $31
- Payload: None
- Response: Status on failure, none on success

The ENTER PRODUCTION TEST MODE command requests that the device restart in production test mode. This occurs immediately without a response packet. Some devices require this command before requesting any production test results. For firmware that does not require this command, the response is either an OK status or a COMMAND NOT IMPLEMENTED status.

While entering production test mode, the device may briefly become unresponsive. After production test mode has finished starting, the device sends an IDENTIFY report to indicate that it is ready to receive commands in the new mode. The Mode field of the IDENTIFY report indicates that the device is in production test mode. If the Mode field indicates something else, then this indicates there was an error switching modes.

After sending this command, wait at least 200 ms before reading. The mode switch may cause glitches on ATTN before the IDENTIFY packet is ready.

5.4.22. GET FEATURES

- Command Code: $32
- Payload: None
- Response: Feature description packet

The GET FEATURES command returns a feature description packet that can be used by tools to tell which features are supported.

5.4.23. CALIBRATE

- Command Code: $33
- Payload: 8-bit calibration ID
- Response: Status

The CALIBRATE command runs a calibration routine specified by the calibration ID and returns a status code indicating whether the calibration succeeded.

While the command executes, the device may be unresponsive. The maximum length of unresponsive time is listed in the device's datasheet. If power to the device is removed before this command completes, any previous stored baseline calibration may be lost.

6. Reports

6.1. IDENTIFY

Report ID: $10

The IDENTIFY report payload has the same format as the identify packet received in response to an IDENTIFY command.

This report is sent whenever the device initially powers up, resets, or switch modes to indicate that the device is ready to receive commands in the new mode. The host can read the "Firmware Mode" field to determine if the device is running the bootloader or application firmware mode.

Commands sent prior to reading the IDENTIFY report are ignored. This is to prevent the host from needing to guess whether a command was received before or after the reset.

If Synaptics TouchComm is revised, the IDENTIFY report provides a mechanism for changing the protocol. When the host sees an IDENTIFY report, it can read the first byte to see which version of the protocol to use until the next IDENTIFY report is received.

6.2. TOUCH

Report ID: $11

The TOUCH report is produced periodically when objects are touching the sensor. Its contents are highly configurable and are described in the SET REPORT CONFIG command above.

6.3. DELTA DATA

Report ID: $12

The DELTA DATA report gives the current capacitance measurements with the baseline values subtracted off at each pixel. It always includes the mutual (trans-) capacitance image. If hybrid data is present, the report also includes X and Y self-capacitance profiles. If the sensor has 0D buttons, the report also includes the button capacitance data.

Each value is a two's complement signed 16-bit value with arbitrary capacitance units. The image, X profile, Y profile, and buttons may each have different units.

To decode the payload, use the following fields in the app info packet:

- ROWS = Number of image rows
- COLS = Number of image columns
- BUTTONS = Number of image buttons
- HAS_HYBRID = Hybrid data present flag
- FORCE = Number of force electrodes

The payload format is:

- First ROWS*COLS 16-bit words: mutual capacitance image.
    - Stored row-major, such that consecutive values represent adjacent columns.
- If HAS_HYBRID:
    - Next COLS 16-bit words: X self-capacitance profile.
    - Next ROWS 16-bit words: Y self-capacitance profile.
- Next BUTTONS 16-bit words: button capacitance values.
- Next FORCE 16-bit words: force electrode values.

6.4. RAW DATA

Report ID: $13

The RAW DATA report gives the current capacitance measurements without the baseline values subtracted off at each pixel. It always includes the mutual (trans-) capacitance image. If hybrid data is present, the report also includes X and Y self-capacitance profiles. If the sensor has 0D buttons, the report also includes the button capacitance data.

Each value is an unsigned 16-bit value with arbitrary capacitance units. The image, X profile, Y profile, and buttons may each have different units.

To decode the payload, use the following fields in the app info packet:

- ROWS = Number of image rows
- COLS = Number of image columns
- BUTTONS = Number of image buttons
- HAS_HYBRID = Hybrid data present flag
- FORCE = Number of force electrodes

The payload format is:

- First ROWS*COLS 16-bit words: mutual capacitance image.
    - Stored row-major, such that consecutive values represent adjacent columns.
- If HAS_HYBRID:
    - Next COLS 16-bit words: X self-capacitance profile.
    - Next ROWS 16-bit words: Y self-capacitance profile.
- Next BUTTONS 16-bit words: button capacitance values.
- Next FORCE 16-bit words: force electrodes values.

6.5. HOST DOWNLOAD STATUS

Report ID: $1B

The HOST DOWNLOAD STATUS report is sent on devices that require the host to send configuration data. The report contains two bytes, with the following contents (with 0 being the least-significant bit):

Table 30. HOST DOWNLOAD STATUS contents

Byte	Bit	Contents
0	0	Invalid static configuration detected
	1	Display configuration required
	2	Touch configuration required
	3-6	Host download version: • Zero indicates host download version 1. • Non-zero indicates the version is the specified value.
	7	Open/short configuration required
1	0-7	Reserved

7. Error recovery

7.1. No Command Response

Synaptics TouchComm allows only one pending command at a time. The host must wait for a response for each command before sending the next command to the device. This creates a problem if the device never produces a command response.

If the device fails to produce a command response in a reasonable amount of time but is still responsive to communications, the host can send a RESET command to restore the device to its power-on state. The RESET command is guaranteed to override all pending commands.

7.2. Unexpected IDENTIFY Report

In some cases, due to ESD events or power supply failures, the device may reset while a command is pending. Upon reset, Synaptics TouchComm devices send an IDENTIFY report, defined in section 6.1. If the host reads an identify report while a command is pending, the command is no longer pending, and the host should regard the device as having reset.

7.3. Unresponsive Devices

In some cases, the device may be briefly become unresponsive to communications. For example, at power-on, after a reset, or during writes to flash memory, the device's communications interface may be disabled. When using the I^{2}C protocol, an unresponsive device cannot acknowledge (ACK) its bus address. When using SPI, a device has no way to indicate that it is unresponsive to writes. However, SPI reads to unresponsive devices produce INVALID responses or packets with incorrect beginning-of-message markers. If the device fails to produce a response to a command after a reasonable length of time, the host can attempt to send a RESET command or power cycle the device.

7.4. Dynamic Configuration Settings

Note that in all cases where the device is reset, or assumed to have reset, any dynamic configuration settings explicitly set with SET DYNAMIC CONFIG commands will be lost and need to be resent.

The parameters will be re-initialized with the values in the dynamic configuration stored in the Application Data section of program memory.

8. Comparison with RMI4

Table 31. Protocol comparison

RMI4	Synaptics TouchComm
Register map discovery (page description table scanning)	Not needed. Report payload content may be specified, and discovered, by the host system.
Query registers	Refer to the commands to get the boot information and application information packets (See 5.3.1 and 5.4.2).
Configuration ID	In the app info packet. Includes the Set Config ID that is read from the packet.
F$51 custom registers	Custom commands, reports, and more are defined in the firmware configuration file (JSON format) and shown in Design Studio.
Bootloader configuration	Included in static config blob, generated by Design Studio and described in the Bootloader section (2.3).
Application configuration	Included in static config blob, generated by Design Studio. Run-time control parameters may be read/written using the GET/SET DYNAMIC CONFIG commands.

9. Reference

9.1. Full Command List

This is a list of all standard Synaptics TouchComm commands. Not all commands are supported by all devices or in all modes. Command codes $C0-$FE are reserved for per-product commands.

Table 32. Standard commands

Command code	Name	Payload	Response	Description
$00	NONE	---	--	On SPI, used to request a read. On I^{2}C, not implemented.
$01	CONTINUE WRITE	---	--	Not a command; continues the previous payload.
$02	IDENTIFY	---	Identify packet	Requests device information.
$04	RESET	---	Status on failure only	Resets the device. The reset occurs immediately, with no response packet. An IDENTIFY report is sent after the reset is complete to indicate a successful reset.
$FF	NONE	---	---	On SPI, used to request a read. On I^{2}C, not implemented.

Table 33. Bootloader commands

Command code	Name	Payload	Response	Description
$10	GET BOOT INFO	---	Boot info packet	Request boot information packet.
$11	ERASE FLASH	Byte 0: Start Byte 1: Count or Bytes 0-1: Start Bytes 2-3: Count	Status	Erases flash pages from Start to Start + Count – 1. Until this command completes, the device might be unresponsive. Host should use 2-byte version of payload unless Start or Count is greater than 255.
$12	WRITE FLASH	Bytes 0-1: Start Bytes 2-end: Data	Status	Writes data to flash starting at write block Start. Until this command completes, the device might be unresponsive.
$13	READ FLASH	Bytes 0-3: Start Bytes 4-5: Size	Flash data	Reads Size bytes from flash, starting from word address Start. Protected memory locations read as 0.
$14	RUN APPLICATION FIRMWARE	---	Status	Runs the application firmware. The application firmware does not actually start until after the host has read this command response.

Command code	Name	Payload	Response	Description
$15	SPI MASTER WRITE THEN READ	Byte 0: SPI clock divider Byte 1 bits 6-7: SPI mode Bytes 2-3: Read length Bytes 4-end: Write data	Requested bytes	Writes the provided bytes on the SPI master port, then reads the requested number of bytes in the same bus transaction.
$16	ENTER ROM BOOTLOADER MODE	---	Status only on failure	Forces the device to restart in ROM bootloader mode.

Table 34. Application firmware commands

Command code	Name	Payload	Response	Description
$1F	ENTER BOOTLOADER MODE	---	Status only on failure	Runs the bootloader. This occurs immediately, without a response packet. An IDENTIFY report is sent to indicate successful bootloader entry.
$20	GET APP INFO	---	App info packet	Request app info packet.
$21	GET STATIC CONFIG	---	Static config	Requests static configuration.
$22	SET STATIC CONFIG	Static config	Status	Sets the static configuration.
$23	GET DYNAMIC CONFIG	Field ID	Field value	Gets a dynamic configuration value by Field ID. If the ID = 255, the entire dynamic configuration is returned.
$24	SET DYNAMIC CONFIG	Byte 0: Field ID Byte 1: Value	Status	Sets a dynamic configuration value by Field ID. If the ID = 255, the entire dynamic configuration is set.
$25	GET REPORT CONFIG	---	Touch report config	Gets the touch report configuration.
$26	SET REPORT CONFIG	Touch report config	Status	Sets the touch report configuration.
$27	REZERO	---	Status	Rezeros sensor baseline estimate.
$28	COMMIT CONFIG	16-byte customer config ID	Status	Commits the current configuration to flash. Until this command completes, the device may be unresponsive.
$29	DESCRIBE DYNAMIC CONFIG	---	Dynamic configuration fields	Lists the fields in the dynamic configurations.

Command code	Name	Payload	Response	Description
$2A	PRODUCTION TEST	Byte 0: Test ID	Test result	Performs a production test. The format of the result varies for each test.
$2B	SET CONFIG ID	Bytes 0-15: Config ID	Status	Sets the 16-byte configuration ID. This is read in the app info packet.
$05	ENABLE REPORT	Byte 0: Report ID	Status	Enables a report.
$06	DISABLE REPORT	Byte 0: Report ID	Status	Disables a report.
$2C	ENTER DEEP SLEEP	---	Status	Disables sensing and places the device in low power deep sleep mode.
$2D	EXIT DEEP SLEEP	---	Status	Transitions device from low power deep sleep mode to normal active mode and enables sensing.
$2E	GET TOUCH INFO	---	Touch info packet	Gets information about how the touch sensor is currently configured.
$2F	GET DATA LOCATION	Data type code	Bytes 0-1: Start write block Bytes 2-3: Length in write blocks	Retrieves the address and length of the specified data in flash.
$30	HOST DOWNLOAD	Byte 0: Always 1 Byte 1: Type Byte 2-end: Data	Status	Status touch or display configuration to devices that use host download to provide firmware.
$31	ENTER PRODUCTION TEST MODE	---	Status on failure only	Switches to production test mode. This occurs immediately, without a response packet. An IDENTIFY report is sent to indicate successful production test mode entry.
$32	GET FEATURES	---	Features description packet	Retrieves a feature description packet that describes the features present in the firmware.
$33	CALIBRATE	Byte 0: Calibration ID	Status	Captures baseline shift calibration data and writes it to flash memory.

9.2. Full Report List

Table 35. Standard report types

Report ID	Name	Payload	Description
Standard reports			
$10	IDENTIFY	Identify packet	Sent at reset to tell the host the current mode.
Application Firmware Reports			
$11	TOUCH	Touch report	Sent whenever new touch data is available.
$12	DELTA DATA	Delta image, profile, and button data	Sent whenever new delta data is available.
$13	RAW DATA	Raw image, profile, and button data	Sent whenever new raw data is available.
$1B	HOST DOWNLOAD STATUS	Status bits	Sent to indicate host download is required.

Contact Us

Visit our website at www.synaptics.com to locate the Synaptics office nearest you.