1. Introduction

This document describes in detail the Object Protocol for the Atmel® mXT540E.

The Object Protocol provides a single common interface across the Atmel touch sensor controllers. This allows the different features in each controller to be configured in a consistent manner. This makes the future expansion of features and simple product upgrades possible, whilst allowing backwards compatibility for the host driver and application code.

2. Overview

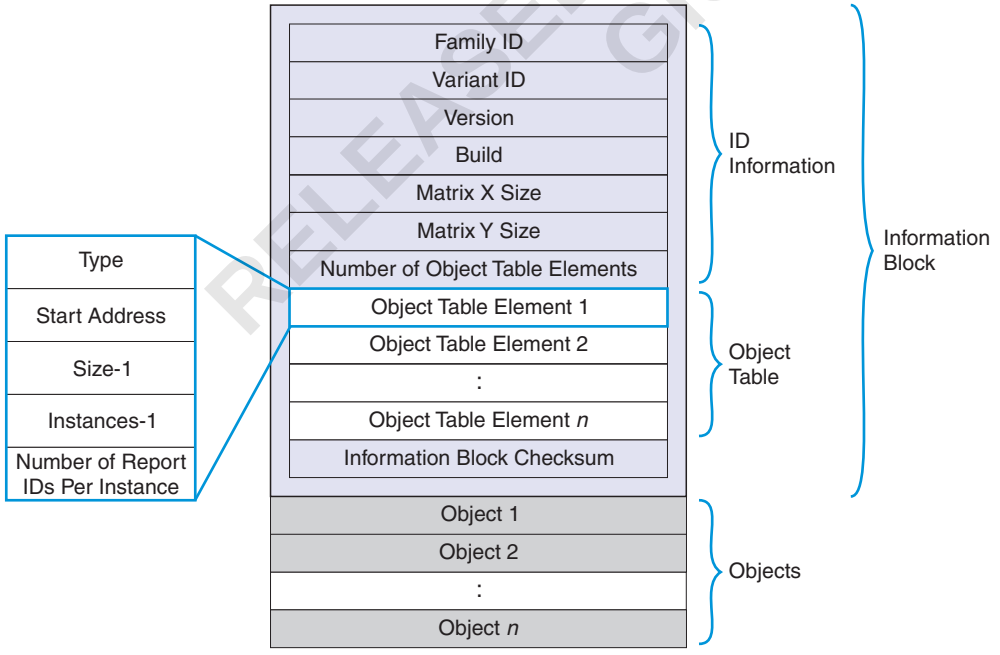
2.1 Memory Map Structure

The protocol is designed to control the processing chain in a modular manner. This is achieved by breaking the features of the device into objects that can be controlled individually. Each object represents a certain feature or function of the device, such as a touchscreen or a key array. Objects can be disabled or enabled as needed.

Each object has its own configuration memory. The objects are stacked together to produce an object-based memory map. A generalized structure of this memory map is shown in Figure 2-1. 20

There are some special objects that can use their memory space for a unique purpose. One example is the Command Processor T6 object, which executes a command when a certain value is written into its memory space. Another example is the Message Processor T5 object, which outputs messages by having the host read its memory space.

Figure 2-1. Generic Memory Map Structure



mXT540E
Revision 1.0

Protocol Guide

maXTouch™

9617CX–AT42–06/11

From Figure 2-1 it can be seen that the memory map contains two main sections:

- An Information Block. This documents which objects are contained in the memory map for the device (see Section 2.2). It is further sub-divided into the ID Information Block and the Object Table (see Section 2.4 on page 3).
- The objects themselves (see Section 2.5 on page 5).

2.2 Information Block

The Information Block allows the host to read information about the layout of objects in the memory map. It contains a list of all the objects in the memory map. This is used by the host driver to know which objects exist, where they are located in the memory map and their sizes. The host driver can therefore read the device's Information Block and gather enough information to be able to communicate with the device.

The Information Block is positioned at the start of the memory map at address zero. This allows it to be read easily by the host as the first operation.

The Information Block contains the following three sections:

- The ID Information fields. These include:
  - Standard ID fields that make up the unique identifier for the device
  - The size of the matrix the device supports
  - The number of objects in the Object Table
- The Object Table itself. This acts as an "index" to the objects in the memory map. Note that one of the objects may be an Information Block object that holds additional Object Table entries (see Appendix D on page 115).
- A checksum for the Information Block. This allows the host to check that the Information Block has been read correctly over the communications interface. See Appendix A on page 105 for details on calculating the checksum.

Table 2-1 shows the contents of the Information Block.

Table 2-1. Information Block Layout

Byte	Description of Field	
0	Family ID	ID Information
1	Variant ID	
2	Version	
3	Build	
4	Matrix X Size	
5	Matrix Y Size	
6	Number of elements in the Object Table	
7 – 12	Object Table element 1 (6 bytes)	Object Table
13 – 18	Object Table element 2 (6 bytes)	
...	...	
...	Last Object Table element (6 bytes)	
(*end*-2) – *end*	24-bit checksum (3 bytes)	Checksum Field

2.3 ID Information

The first four bytes of the Information Block are used to identify the device and its version, as in Table 2-2.

Table 2-2. Device Identifier Fields

Field	Description
Family ID	A unique identifier that indicates the device's family.
Variant ID	A unique identifier that indicates the device's variant.
Version	The current major and minor firmware version of the device. The upper nibble contains the major version and the lower nibble contains the minor version. For example, firmware version 1.0 is stored as 0x10.
Build	The firmware build number.

2.4 Object Table

2.4.1 Introduction

The Object Table is held within the Information Block. The Object Table contains information on all the objects held within the memory map. It indicates which objects are present and their addresses.

IMPORTANT!
Future versions of the chip may include the Information Block T254 object. This object will contain additional Object Table entries. For future compatiblity, Any *current* driver code should be written to allow for the presence of the Information Block T254 object when it is eventually used. The host driver code must parse the Object Table, as at present, and also process the Information Block T254 object, if this is found to be present on the device.
See Appendix D on page 115 for more information.

Each element in the Object Table has the fields listed in Table 2-3.

Table 2-3. Format of an Object Table Element

Byte	Description of Field
0	Type
1	Start position LSByte
2	Start position MSByte
3	Size - 1
4	Instances - 1
5	Number of Report IDs per instance

2.4.2 Type

Each type of object has a unique type code to identify it. This is the number after the "_T" suffix at the end of the object's internal name as given in Section 3 to Section 7. For example, the type code for the Command Processor T6 is **6** (from GEN_COMMANDPROCESSOR_T**6**).

2.4.3 Start Position

Bytes 1 and 2 of the Object Table element holds the start location of the object in the memory map (LSByte and MSByte respectively).

The driver code should ALWAYS read these bytes to find out where in the memory map the object is located and use this address to communicate with the device. The driver code should never use hard-coded addresses for the objects, as these may change with firmware updates.

This means that driver code can be written without making assumptions about the addresses of the objects. This ensures that the code is “future-proof” and will work correctly following firmware updates to the device. It also makes it possible to write common driver code for communication with any Atmel touch controller that uses this object-based protocol approach.

2.4.4 Size

Byte 3 of the Object Table element holds the size (minus 1) of the object in the memory map. This is stored as Size-1, so it is effectively the offset to the end of the object.

2.4.5 Number of Instances

Byte 4 of the Object Table element holds the number of instances of the object in the memory map, minus 1. The number of instances can be calculated by adding 1 to this number (see Table 2-4 on page 6). The different instances of an object are arranged consecutively in the memory map.

2.4.6 Report IDs

If an object sends messages, it is necessary to identify the messages from the object so that they can be correctly interpreted. A report ID is therefore used to identify the source object of a message returned in the Message Processor T5 object (see Section 4.1 on page 11).

Report IDs are numbered sequentially in the order in which the objects are listed in the Object Table, allowing for the appropriate number of instances for each object. Note the following:

- A report ID of zero is a reserved value for use by Atmel. Report IDs from a user’s perspective therefore effectively start from 1.
- A report ID value of 255 is reserved to indicate an “invalid message” response.

If an object has report IDs allocated, each instance of the object will have its own block of report IDs. In the case of a Multiple Touch Touchscreen T9 object, the allocation of the report IDs is determined by the number of touches it reports on. If an instance of a Multiple Touch Touchscreen T9 object reports on 16 touches, it typically has 16 report IDs allocated.

Figure 2-2 on page 5 shows an example of how the number of instances and the Report IDs per instance determine the report IDs for a set of objects. Note that the objects shown in Figure 2-2 are examples only and may not reflect the actual objects present on the mXT540E.

The driver code should build up its own in-memory table of object types and associated report IDs during its initialization. It can do this by parsing the object structure given in the Object Table. This in-memory table can then be used to interpret the messages returned by the device.

A typical algorithm to process the report IDs is as follows:

1. For each element in the Object Table:
   a. Read byte 4 to retrieve the number of instances (remember to add 1 to the value retrieved).
   b. Read byte 5 to retrieve the number of report IDs per instance.
   c. Multiply the figures retrieved in steps 1 and 2 together, and then add this number of "object type/report ID" pairings to the table being built. The report IDs should have sequential values starting with 1 (the zero value is reserved for use by Atmel).
2. Check for the presence of an Information Block T254 object (See Appendix D on page 115). If the object is present on the device, repeat Step 1 for each element held within the Information Block T254 object, continuing to allocate the new reportIDs to those already allocated.

Figure 2-2. Example Assignment of Report IDs

Object	Instances	Report IDs
Command Processor T6	1	1
Multiple Touch Touchscreen T9	2	16
Key Array T15	2	1

Report ID	Object Instance
0	(Reserved)
1	Command Processor T6
2	TouchScreen 0 Touch 0
:	
17	TouchScreen 0 Touch 15
18	TouchScreen 1 Touch 0
:	
33	TouchScreen 1 Touch 15
34	Key Array T15 0
35	Key Array T15 1

2.5 Objects

2.5.1 Classes of Objects

The mXT540E contains the following classes of objects:

- **Debug objects** – provide a raw data output method for development and testing. See Section 3 on page 9.
- **General objects** – required for global configuration, receiving commands and transmitting messages. See Section 4 on page 11.
- **Touch objects** – operate on measured signals from the touch sensor and report touch data. See Section 5 on page 26.
- **Signal processing objects** – process data from other objects (typically signal filtering operations). See Section 6 on page 55.
- **Support objects** – provide additional functionality on the device. See Section 7 on page 88.

2.5.2 Object Instances

Table 2-4 lists the instances of the objects on the mXT540E.

Table 2-4.

Object	Number of Instances	Reference
Debug Objects		
Diagnostic Debug T37	1	Section 3.2 on page 9
General Objects		
Message Processor T5	1	Section 4.2 on page 11
Command Processor T6	1	Section 4.3 on page 13
Power Configuration T7	1	Section 4.4 on page 15
Acquisition Configuration T8	1	Section 4.5 on page 17
Data Source T53	1	Section 4.6 on page 24
Touch Objects		
Multiple Touch Touchscreen T9	2	Section 5.2 on page 26
Key Array T15	2	Section 5.3 on page 45
Proximity Key T52	2	Section 5.4 on page 49
Signal Processing Objects		
One-touch Gesture Processor T24	2	Section 6.2 on page 55
Two-touch Gesture Processor T27	2	Section 6.3 on page 64
Grip Suppression T40	2	Section 6.4 on page 66
Touch Suppression T42	2	Section 6.5 on page 68
Stylus T47	2	Section 6.6 on page 72
Noise Suppression T48	1	Section 6.7 on page 75
Support Objects		
Communications Configuration T18	1	Section 7.2 on page 88
GPIO/PWM Configuration T19	1	Section 7.3 on page 89
Self Test T25	1	Section 7.4 on page 93
User Data T38	1	Section 7.5 on page 99
Digitizer HID Configuration T43	1	Section 7.6 on page 99
Message Count T44	1	Section 7.7 on page 100
CTE Configuration T46	1	Section 7.8 on page 101

2.5.3 Configuration Defaults

The objects are designed such that a default value of zero in their fields is a “safe” value. For example, a value of zero typically disables functionality.

An object must be configured as required with non-zero values before use. Any unused settings can be left at their default zero values. The settings should also be written to the nonvolatile memory using the Command Processor T6 object (see Section 4.3 on page 13).

2.5.4 Compatibility of Object Versions

The Object Protocol described in this datasheet may document fields that are not present in the memory map as it is implemented on a particular firmware version of the device. Over time newer versions of the objects in the Object Protocol may gain additional fields to implement new features.

New fields are added to the end of an object to allow for this situation. This preserves the order of the fields between old and new object versions. A driver designed to work with an older version of the device can safely set any unknown fields located at the end of the object to zero. The device will then behave in the same manner as the older version of the device without the field.

The host driver must always use the Object Table to locate the address of each object and the object's current size. It must also zero any fields that it does not intend to use. This ensures that the host's driver code is compatible with this object expansion scheme.

2.6 Configuration Checks

The device prevents against invalid configurations by performing a sequence of checks. These checks are performed on power-up. They are also performed whenever certain configuration settings are updated. These are typically settings that force a recalculation, a reinitialization or a recalibration when they are changed.

If an error is found during the configuration check, the device pauses until the configuration error is resolved. The device also flags the error to the host by setting the CFGERR bit of the Command Processor T6 message data (see Section 4.3.2 on page 14). This message will be sent to the host every 200 ms until the error is corrected.

Backup requests are allowed when an error has been found during a configuration check. This allows a setting to be corrected and backed up to the nonvolatile memory (NVM). The device can then be reset for the setting to take effect, if the setting requires this. The device will not operate until all errors have been corrected.

Possible causes of configuration errors are:

- Touch objects with overlapping channels or channels outside the maximum matrix dimensions
- Touch objects under the minimum X and Y size, or a Key Array T15 object with a channel size greater than 32
- Field settings outside the permitted range
- A bad checksum for the configuration settings held in the nonvolatile memory

If a configuration error is encountered during product development, it is sufficient for the designer to check the settings used and correct them. In a working product, the device driver should resend the configuration settings and request a backup. The device driver should be written to handle this.

To find the source of the configuration error, first disable all touch, signal processing and support objects. Then re-enable each object in turn until the configuration error is found. Once the error has been corrected, the other objects can be re-enabled and processing can continue as normal. See Appendix B on page 113 for a flow chart showing this method for finding configuration errors.

Notice should be taken of any recommendations in this datasheet concerning configuration checks for the offending object.

2.7 Byte Order

The memory map uses a “little-endian” configuration for its bytes, meaning that all multibyte fields lead with the least significant byte (LSByte).

2.8 Signal and Reference Values

The format used for signal and reference values used by the protocol objects depends on the technology used by a particular touch object. Mutual-capacitance (CTE-based) touch objects use signed 16-bit values in offset binary format; self-capacitance touch objects use unsigned 16-bit values. Table 2-6 lists the format used by the touch objects on the mXT540E.

Table 2-5. Signal and Reference Formats

Object	Technology	Format
Multiple Touch Touchscreen T9 Key Array T15	Mutual capacitance with Capacitive Touch Engine (CTE)	Signed 16-bit values in offset binary format
Proximity Key T52	Self-capacitance with QTouch®	Unsigned 16-bit values

Signal and reference values for mutual-capacitance objects are held as signed 16-bit values using offset binary (lower quarter) format [1]. The offset is 0x4000 (16384), which means that to find the true value this offset must be subtracted from the stored values. Table 2-6 lists some sample values.

Table 2-6. Offset Binary Values

Offset Binary Value	Represents...
0x0000	-16384 (minimal negative value)
0x3FFF	-1
0x4000	0
0x4001	1
0xFFFF	49151 (maximal positive value)

1. Also known as Excess-K, where K = 0x4000 (16384).

3. Debug Objects

3.1 Introduction

Debug objects contain raw data for development and testing purposes. Table 3-1 lists the debug objects on the mXT540E.

Table 3-1. Debug Objects

Object and Name	Description
Diagnostic Debug T37 (DEBUG_DIAGNOSTIC_T37)	Allows access to debug data to aid development. See Section 3.2.

3.2 Diagnostic Debug T37 (DEBUG_DIAGNOSTIC_T37)

The Diagnostic Debug T37 object holds the debug data. The following modes of data are available:

- Delta mode – Signed (two's complement) 16-bit delta values for all channels. These are the raw deltas. They are divided by 8 before use elsewhere (such as for threshold comparisons).
- Reference mode – Values are held in the appropriate format for the touch objects, as given in Section 2.8 on page 8 [1].

The mode is determined by the command written to the DIAGNOSTIC field of the Command Processor T6 object (see Section 4.3 on page 13).

The Diagnostic Debug T37 object works by organizing the data in pages. The value of the PAGE field determines which page of data is currently available, and the object's DATA field holds the data for that page. The pages can be navigated by writing Page Up/Page Down commands to the DIAGNOSTIC field in the Command Processor T6 object (see Section 4.3 on page 13).

A mode or page change command is processed only once per measurement cycle. Note that no command processing is done if the device is in deep sleep mode

3.2.1 Delta and Reference Modes

Figure 3-1 shows the organization of the data in the pages for delta and reference modes. The values for the primary mutual-capacitance matrix channels are listed first, followed by the values from the self-capacitance channels defined by the Data Source T53 object on the next clear page [2]. Note that the channels are numbered along the X lines, that is in the order 0 = X0Y0, 1 = X0Y1, 2 = X0Y2 and so on.

The page number and data index for a given mutual-capacitance channel's data can be calculated as follows:

Page number = FLOOR (channel_number / channels_per_page)

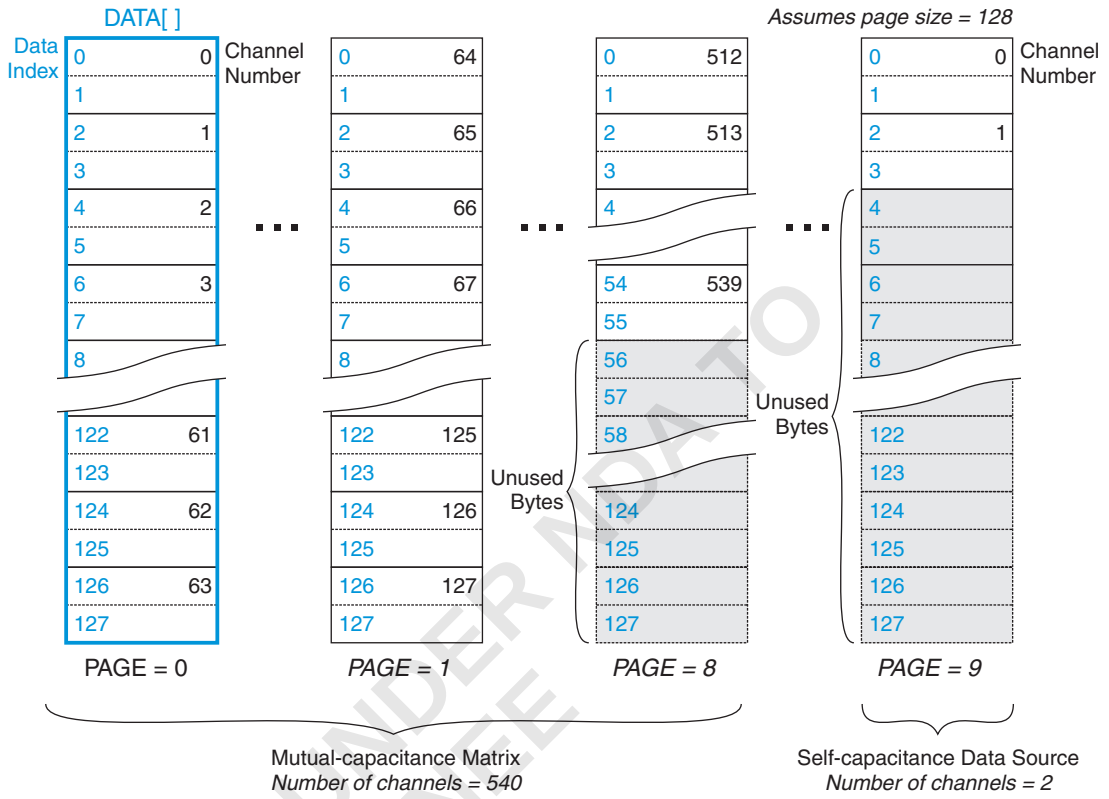
Data index = bytes_per_channel x (channel_number MOD channels_per_page)

where:

- bytes_per_channel is 2
- channels_per_page is the page size divided by bytes_per_channel. In Figure 3-1 this is 128 / 2).

1. That is as unsigned 16-bit values or as 16-bit signed in offset binary format values, as appropriate.
2. See also "Errata" on page 120 for additional information.

Figure 3-1. Diagnostic Debug T37 Fields – Delta and Reference Modes



3.2.2 Configuration

Table 3-2. DEBUG_DIAGNOSTIC_T37

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	MODE	Current mode							
1	PAGE	Page number							
2 – (2+*n*-1)	DATA[*n*]	Data for current page							

Note: *n* = page size

MODE Field

This field contains an indication of the current mode for the data in the DATA[] field. It has the same value as the mode change commands (but not the page change commands) in the DIAGNOSTIC field of the Command Processor T6 object (see Section 4.3 on page 13).

PAGE Field

This field contains the current page number for the data held in the DATA[] field, as controlled by the page change commands in the DIAGNOSTIC field of the Command Processor T6 object (see Section 4.3 on page 13).

DATA[] Field

This field contains the current page of data.

4. General Objects

4.1 Introduction

General objects provide global configuration, such as receiving commands and transmitting messages. Table 4-1 lists the general objects on the mXT540E.

Table 4-1. General Objects

Object	Description
Message Processor T5 (GEN_MESSAGEPROCESSOR_T5)	Handles the transmission of messages. This object holds a message in its memory space for the host to read. See Section 4.2.
Command Processor T6 (GEN_COMMANDPROCESSOR_T6)	Performs a command when written to. Commands include reset, calibrate and backup settings. See Section 4.3.
Power Configuration T7 (GEN_POWERCONFIG_T7)	Controls the sleep mode of the device. Current consumption can be lowered by controlling the acquisition frequency and the sleep time between acquisitions. See Section 4.4.
Acquisition Configuration T8 (GEN_ACQUISITIONCONFIG_T8)	Controls how the device takes each capacitive measurement. See Section 4.5.
Data Source T53 (GEN_DATASOURCE_T53)	Describes a data source. This object is read-only. See Section 4.6.

4.2 Message Processor T5 (GEN_MESSAGEPROCESSOR_T5)

The purpose of the Message Processor T5 object is to relay the latest status information to the host. This object contains the message data from those objects in the memory map that generate messages (for example, the touch objects and the Command Processor T6 object). A message is generated whenever an object's status has changed. For this to happen, the object's report enable bit must be set.

When a device has data to send, it asserts the $\overline{\text{CHG}}$ line to indicate to the host that there is a message. The host should then read the message and use the REPORTID field to determine from which object the message originated. This provides the host with an interrupt-style interface. This has the potential for fast response times and reduces the need for wasteful I^2C-compatible communications.

The host should ALWAYS use the $\overline{\text{CHG}}$ line as an indication that a message is available to read in the Message Processor T5 object. The host should not read the Message Processor T5 object at any other time, such as to poll the device for messages. If the Message Processor T5 object is read when the $\overline{\text{CHG}}$ line is not asserted, an "invalid message" report ID is returned in the Message Processor T5 object.

Multiple messages can easily be read in a continuous read operation using direct memory access (DMA) and message pointer wrapping is implemented to allow this. The Message Count T44 object (see Section 7.7 on page 100) provides a count of pending messages so that the host driver code knows how many messages to read.

Message pointer wrapping occurs if the address pointer is pointing at either the Message Count T44 object (1) or the start of the Message Processor T5 object (that is, the report ID of the next available message).

Note that if checksum mode is enabled, the address pointer wrapping between messages occurs after the checksum byte has been sent. Otherwise the wrapping occurs before the checksum byte to save unecessary reads of the unused checksum byte.

Refer to the following datasheet for more information on DMA reads using the I^2C-compatible interface:

- mXT540E – *540-channel Touchscreen Sensor IC*

An alternative message-reading mechanism is available when using the USB interface. The host can request that messages are sent autonomously whenever an object has generated a message. Refer to the following datasheet for more information:

- mXT540E – *540-channel Touchscreen Sensor IC*

4.2.1 Configuration

Table 4-2. GEN_MESSAGEPROCESSOR_T5

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	REPORTID	Report ID for source object							
1 – *n* (1)	MESSAGE	Message data from source object							
n+1	CHECKSUM	Checksum							

1. The size of the MESSAGE field is set to the length of largest message possible. It is dependent on the objects present in the device at a particular revision. The size should be calculated by subtracting 2 from the size of the Message Processor T5 object retrieved from the Object Table entry (see Section 2.4 on page 3). Any unused bytes in a particular message should be treated as reserved bytes.

REPORTID Field

This field contains the report ID for the message. Messages contain report IDs to allow the host to identify the type of message and its originator. Report IDs are assigned to any object that can send messages. See Section 2.4.6 on page 4 for more information on the assignment of report IDs.

MESSAGE Field

This field contains the message data for the object generating the message.

The size of the MESSAGE field is fixed to the size of the message data for the largest object. For compatibility with future firmware updates, this should *always* be calculated by subtracting 2 from the size of the object recorded in the Object Table entry for the Message Processor T5 object (see Section 2.4 on page 3).

For information on the contents of the MESSAGE field, see the descriptions for each object elsewhere in this datasheet.

CHECKSUM Field

This field contains the 8-bit checksum for the Message Processor T5 object (that is, for the REPORTID and MESSAGE fields) if a communications checksum is requested.

1. Note that the Message Count T44 object occurs immediately before the Message Processor T5 object in the memory map so that the message wrapping mechanism will work.

To request that a checksum is generated, the MSBit of the address of the Message Processor T5 object is set to 1 during a read. For example, if the address of the Message Processor T5 object is 0x0477, specifying the address as 0x8477 will generate a checksum for that read.

If the communications checksum feature is not enabled, this byte should not be read.

See Appendix A on page 105 for details on how to calculate the checksum.

4.3 Command Processor T6 (GEN_COMMANDPROCESSOR_T6)

The Command Processor T6 object allows commands to be sent to the device. This is done by writing an appropriate value to one of its fields.

4.3.1 Configuration

Table 4-3. GEN_COMMANDPROCESSOR_T6

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	RESET	Reset							
1	BACKUPNV	Backup settings							
2	CALIBRATE	Calibrate							
3	REPORTALL	Report current status							
4	Reserved	Reserved							
5	DIAGNOSTIC	Diagnostic debug command							

RESET Field

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

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

BACKUPNV Field

This field backs up settings to the non-volatile memory (NVM). Once the device has processed this command it generates a status message containing the new NVM checksum.

Write value: 0x55

CALIBRATE Field

This field performs a global recalibration on all mutual-capacitance channels. If all the channels are disabled, no message is generated. If the device is in Deep Sleep mode (see Section 4.4), a message is generated when the device wakes from sleep.

Note that this command does not calibrate self-capacitance channels configured using the Proximity Key T52 object). Use the calibrate command provided by the Proximity Key T52 object) to calibrate the proximity keys.

Write value: Nonzero

REPORTALL Field

This field forces all objects that generate messages to report their current status:

- For optional objects, this applies only if they have their report enable bit set and are currently enabled.
- For objects that are always present and generate messages setting this bit will always cause a status message to be reported.

This field is cleared once the command has been processed.

Write value: Nonzero

DIAGNOSTIC Field

This field allows commands to be written to control the use of the Diagnostic Debug T37 object (see Section 3.2 on page 9). Specifically, it allows the pages of debug data to be navigated and sets the data mode. This field is cleared once the command has been processed. The host can poll this field, for example, to determine that a page change has been actioned and that the requested data is now valid.

The valid commands are listed in Table 4-4.

Table 4-4. Diagnostic Debug Commands

Command	Name	Description
0x01	Page Up	Increment page number.
0x02	Page Down	Decrement page number.
0x10	Deltas Mode	The Diagnostic Debug T37 object holds signal deltas.
0x11	References Mode	The Diagnostic Debug T37 object holds reference values.

Note: Changing the mode resets the page number to zero.

4.3.2 Messages

The message data (see Section 4.2) for the Command Processor T6 object is shown in Table 4-5.

Table 4-5. Message Data for GEN_COMMANDPROCESSOR_T6

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	RESET	OFL	SIGERR	CAL	CFGERR	COMSERR	Reserved	
2 – 4	CHECKSUM	Configuration settings checksum							

STATUS Field

Reports the current status and flags errors. A bit is set to indicate the corresponding status/error. Note that there may be more than one status/error reported.

CFGERR, CAL, SIGERR and OFL report ongoing status and error conditions, so once these status/error conditions have terminated, a further message is sent with the appropriate bit cleared.

COMSERR and RESET are one-off reports indicating already terminated conditions. These error conditions do not generate a futher message with a cleared bit.

COMSERR: There is an error with the communications checksum. This error bit is set when the device is being used in communications checksum mode and there has been a checksum error on the bytes that have been written to the device. Note that if there is a checksum error after a write, then the data will still have been written to the device. It is the host's responsibility to take corrective action.

CFGERR: There is a configuration error in one or more of the enabled objects. The device pauses its processing and generates a status message every 200 ms. Note that the device will stop scanning for touches while the error persists.

Note: It is possible to execute a backup command while the device is in this error state.

See Section 2.6 on page 7 for more information on configuration checks.

CAL: The device is calibrating.

SIGERR: There was an error in the acquisition. This error should not normally be seen.

OFL: The acquisition and processing cycle length has overflowed the desired power mode interval. These are controlled by the IDLEACQINT and ACTVACQINT fields in the Power Configuration T7 object (see Section 4.4). Note that the OFL flag is not updated in Free-run or Deep Sleep modes.

RESET: The device has reset.

CHECKSUM Field

Reports the checksum of the configuration settings held in the nonvolatile memory. See Appendix A on page 105 for details on how to calculate the checksum.

4.4 Power Configuration T7 (GEN_POWERCONFIG_T7)

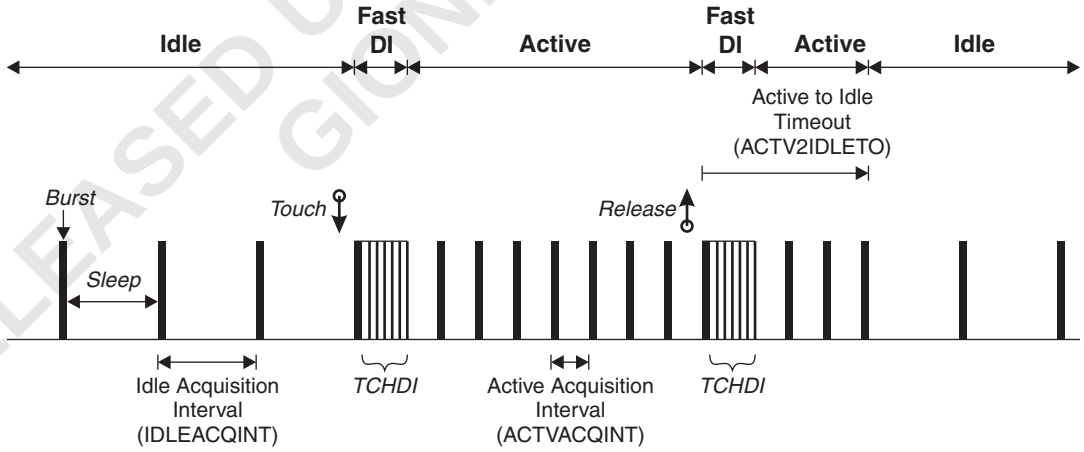
The Power Configuration T7 object controls the active and idle (sleep) times of the device. Current consumption can be lowered by controlling the acquisition frequency and sleep times between measurements.

The device operates in two modes: active (touch detected) and idle (no touches detected).

The normal state for the device is idle mode. In this mode the device operates in a series of long burst cycles. Each cycle consists of a short burst (during which measurements are taken to detect a possible touch) followed by an inactive sleep period.

Figure 4-1 shows the power modes for a Multiple Touch Touchscreen T9 object.

Figure 4-1. Power Mode Fields – Multiple Touch Touchscreen T9 object



When the user touches a touchscreen, the device enters a Fast DI (free-run) mode. This consists of a series of quick, short bursts to confirm that a change in the touch state has indeed occurred. The number of bursts is determined by the TCHDI field in the touch objects. If it is a genuine touch, the device enters active mode. In this mode the device operates in a series of burst cycles that intersperse measurement bursts with very short sleep periods. These sleep periods are typically shorter than those in idle mode.

When the user releases the touch, the device again enters a short Fast DI mode to confirm that a change in the touch state has occurred. Then, if it is a genuine release, the device returns to idle mode after a short timeout period. During this timeout, the device continues to run in active mode to allow further touches to keep the device active.

For a Key Array T15 object, the ACTV2IDLETO (and the subsequent Idle mode) applies on a touchdown as well as a release, as in Figure 4-2.

A Proximity Key T52 object has no effect on the cycle time.

Note that the changes to the cycle time happen regardless of whether the touch object is reporting or not.

Figure 4-2. Power Mode Fields – Key Array T15 object

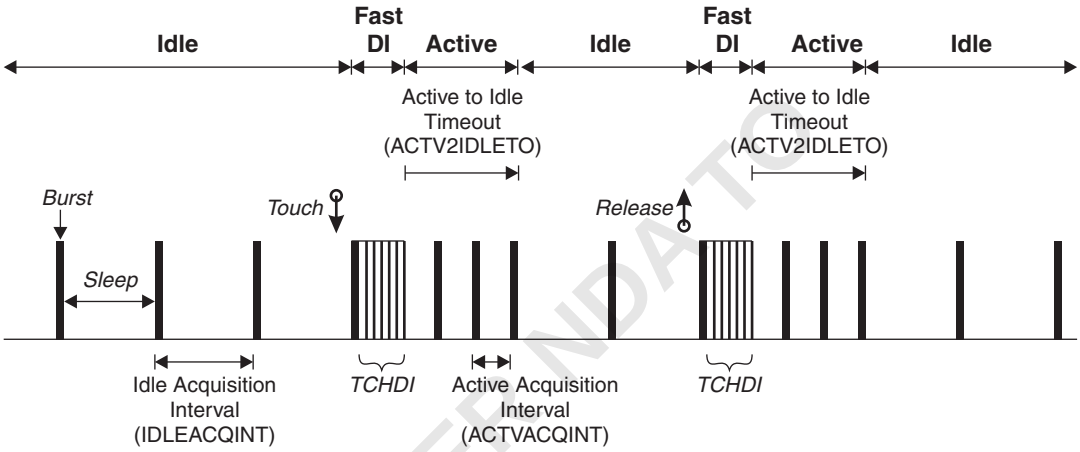


Table 4-6. GEN_POWERCONFIG_T7

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	IDLEACQINT	Idle Acquisition Interval							
1	ACTVACQINT	Active Acquisition Interval							
2	ACTV2IDLETO	Active to Idle Time Out							

IDLEACQINT and ACTVACQINT Fields

The length of the idle and active burst cycles is determined by the Idle Acquisition Interval (IDLEACQINT) and the Active Acquisition Interval (ACTVACQINT) fields respectively (see Figure 4-1 and Figure 4-2).

A setting of 255 forces the device to enter Free-run mode the next time that the appropriate mode (idle or active) is entered. In Free-run mode the device does not sleep between acquisitions. This gives the fastest response time at the expense of power consumption.

A setting of zero forces the device to enter Deep Sleep mode the next time that the appropriate mode (idle or active) is entered. The device remains in Deep Sleep mode until the IDLEACQINT or ACTVACQINT setting is restored. Deep Sleep mode is used to conserve maximum power if the device does not need to be sensing. If Deep Sleep mode is requested, it is advisable to set both IDLEACQINT and ACTVACQINT to zero to avoid indeterminate behavior if one mode is still active. The status flags in the Command Processor T6 (see Section 4.3 on page 13) are not updated when the device is in Deep Sleep mode.

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

Do not set either field to be less than the actual burst time. The device is also designed to sleep as much as possible in order to conserve power. IDLEACQINT should therefore be set longer than ACTVACQINT. Under some circumstances it may be desirable to set IDLEACQINT lower than ACTVACQINT. For example, this might be necessary to minimize the difference between the best-case and the worst-case touchdown latency.

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

IDLEACQINT Typical: 32 (32 ms)

ACTVACQINT Typical: 16 (16 ms)

ACTV2IDLETO Field

The device automatically goes into idle mode whenever possible after each matrix scan to conserve power, unless a touch object is being touched (see Figure 4-1).

The device does not go into idle mode immediately. Instead there is a timeout period. The device runs in active mode during this timeout period to allow further touches to keep the device active. This timeout period is determined by the Active to Idle Timeout (ACTV2IDLETO) field. Under normal operation, the device enters idle mode after the expiry of the Active to Idle Timeout and then remains in idle mode until the next touch is detected. If there is more than one touch present, the Active to Idle Timeout applies only after the last touch has been released. This means that once the device has been awakened by a change, the touch response time is fast for as long as the sensor remains in use. Once channel activity lapses for a period longer than the Active to Idle Timeout, the device returns to idle mode.

The Active to Idle Timeout is specified in 200 ms increments, where 0 means 1 cycle.

Range: 0 (1 cycle), 1 to 255 (in 200 ms increments)

Typical: 50 (10 seconds)

4.5 Acquisition Configuration T8 (GEN_ACQUISITIONCONFIG_T8)

4.5.1 Configuration

The Acquisition Configuration T8 object controls how the device takes capacitive measurements.

Table 4-7. GEN_ACQUISITIONCONFIG_T8

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CHRGTIME	Charge-transfer dwell time							
1	Reserved	Reserved							
2	TCHDRIFT	Touch drift							
3	DRIFTST	Drift suspend time							
4	TCHAUTOCAL	Touch automatic calibration							
5	SYNC	Reserved					DISRISE	DISFALL	ENABLE
6	ATCHCALST	Antitouch calibration suspend time							
7	ATCHCALSTHR	Antitouch calibration suspend threshold							
8	ATCHFRCCALTHR	Antitouch force calibration threshold							
9	ATCHFRCCALRATIO	Antitouch force calibration ratio							

CHRGTIME Field

This setting controls the charge-transfer dwell time. It is specified in clock cycles, one clock cycle equating to 33.3 ns at 30 MHz. Higher charge times result in a slower scan time. The recommended maximum for this field is 150 (5 µs). The default value of 0 means 45 (1.5 µs).

Although the minimum value for this field is 15 (500 ns) [1], the working minimum can be calculated as follows:

Minimum_CHRGTIME = 3.84 * XSLEW * (AVdd / AVdd_nominal) / Clock_cycle

where:

XSLEW is either 500 ns or 350 ns, as set in the XSLEW field in the CTE Configuration T46 object (see Section 7.8 on page 101)

Clock_cycle is 1/30 MHz (= 33.3 ns clock cycle period)

AVdd_nominal is the nominal analog supply (+2.7 Volts)

See Table 4-8 for the limits based on different slew rates and frequencies.

Table 4-8. mXT540E Charge Time Limits – X Slew Rate = 500 ns , AVdd = +2.7V

Base Sampling Frequency (Noise Suppression T48: BASEFREQ)	Nominal Burst Frequency (KHz)	Charge Time (CHRGTIME)
28	300	15
25	293	18
23	285	20
20	279	23
18	272	25
15	267	28
13	261	30
10	255	33
8	250	35
5	245	38
3	240	40
0	236	43

1. Values less than this are automatically set to 15.

Table 4-9. mXT540E Charge Time Limits – X Slew Rate = 350 ns , AVdd = +2.7V

Base Sampling Frequency (Noise Suppression T48: BASEFREQ)	Nominal Burst Frequency (KHz)	Charge Time (CHRGTIME)
13	353	15
10	343	18
8	333	20
5	324	23
3	316	25
0	308	28

See also “Revision History” on page 121.**Range:** 0 (45 = 1.5 µs), 15 to 150 (in clock cycles; 33.3 ns increments)

Default: 45 (1.5 µs)

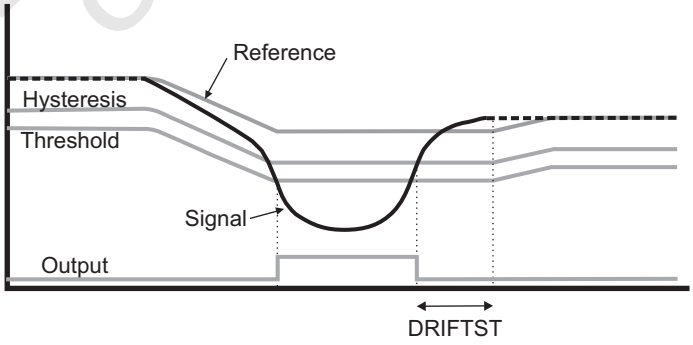
TCHDRIFT Field

The Touch Drift (TCHDRIFT) setting controls the drift interval.

Signals can drift because of changes in the nature of the components and materials over time and temperature. It is crucial that such drift is compensated for, otherwise false detections and sensitivity shifts can occur.

Drift compensation (see Figure 4-3) is performed by making the reference level track the raw signal at a slow rate, but only while there is no detection in effect. The rate of adjustment must be slow, otherwise legitimate detections could be ignored. The TCHDRIFT field can be configured in increments of 200 ms, where a value of 0 means 10 (2 seconds).

Figure 4-3. Thresholds and Drift Compensation



The device drift compensates using a slew-rate limited change to the reference level. The threshold and hysteresis values are slaved to this reference. Any changes to TCHDRIFT affect the rate. The maximum steps per update is fixed at 2. [1]

If the touch is in a Multiple Touch Touchscreen T9 or Key Array T15 object, then once the signal has crossed the respective threshold level for that object, the drift compensation mechanism ceases.

Range: 0 (10 = 2s), 1 to 254 (in 200 ms increments)

1. That is, the drift compensation can drift at a rate of up to 2 steps once every (TCHDRIFT x 200) ms.

Typical: 20

Default: 0 (10 = 2s)

DRIFTST Field

The Drift Suspend Time (DRIFTST) setting controls the time from a touch release on a Multiple Touch Touchscreen T9 or Key Array T15 object until the drift process is re-enabled. DRIFTST is used to restrict drift on all channels while one or more mutual-capacitance channels are activated. It defines the length of time the drift is halted after a touch detection.

Note that the Proximity Key T52 object (see Section 5.4 on page 49) will not itself cause the Drift Suspend Time to be triggered but it is affected by the Drift Suspend Time when it is triggered by another touch object.

This feature is particularly useful in preventing an actual touch – or simply a hovering finger – from causing untouched channels to drift. Without this feature, a sensitivity shift could be created that would ultimately inhibit any further touch detection.

DRIFTST can be configured to a value of between 0 and 255 in increments of 200 ms, where a value of zero means 4 seconds. This gives a range of 200 ms to 51s.

Range: 0 (4 seconds), 1 to 255 (in 200 ms increments)

Typical: 20

Default: 0 (4 seconds)

TCHAUTOCAL Field

A prolonged (usually unintentional) contact from a foreign object may result in a touch detection for a prolonged interval. It is desirable to perform a recalibration in order to restore a touch object's function. This is usually done after a time delay of some seconds.

The Touch Automatic Calibration (TCHAUTOCAL) setting controls the length of time a touch is held until it is considered false and an automatic recalibration is performed to compensate.

The TCHAUTOCAL timer monitors touch detections. If a detection event exceeds the timer's setting, an automatic recalibration occurs. After the recalibration has taken place, normal functionality resumes, even if the touch object is still being contacted by the foreign object. This feature is enabled globally, but the exact mechanism depends on the object being touched:

- For a Multiple Touch Touchscreen T9 object (see Section 5.2) there is a counter per touch, incremented while the touches remain stationary. If a touch is stationary at the end of the TCHAUTOCAL period an automatic recalibration occurs. A touch is considered stationary if it moves less than 1/16 of the screen size (in either axis) within the TCHAUTOCAL period. The automatic recalibration recalibrates all the channels covered by the touchscreen.
- For a Key Array T15 object (see Section 5.3) there is a counter per key, incremented while a touch remains on the key. An automatic recalibration occurs if the touch is still present at the end of the TCHAUTOCAL period. The automatic recalibration recalibrates a single key. Note that a touch detection within a key does not clear the reset counters for any other keys, or the reset counters for any other touch objects.
- For a Proximity Key T52 object (see Section 5.4) there is a counter per key, incremented for as long as a proximity is detected. An automatic recalibration occurs if the proximity is still present at the end of the TCHAUTOCAL period and the Proximity Key T52 object's TCHACEN control is set. Note that a proximity detection does not clear the reset counters for any other touch objects.

TCHAUTOCAL can be disabled by setting it to zero (infinite timeout). In this case the object never autorecalibrates during a continuous detection (but the host could still command it). TCHAUTOCAL above 0 is configured in 200 ms increments.

Range: 0 (infinite), 1 to 255 (in 200 ms increments)

SYNC Field

This field enables the use of a SYNC pin on the device to synchronize entire acquisition cycles with an external clock.

Enabling SYNC mode (using the ENABLE bit) forces the device to wait for an edge on the SYNC pin before performing a measurement. In this case if there is no edge, the device will not acquire. This feature overrules the settings in the Power Configuration T7 object (see Section 4.4 on page 15). If this feature is disabled, the device will run as configured by the Power Configuration T7 object.

If synchronization edges occur more frequently than the acquisition cycles, the device triggers on the next synchronization edge that occurs after the current acquisition cycle. Note that the synchronization edges must not be more than 255 ms apart.

Note: The SYNC pin is shared with the GPIO2 pin and the SNSK0 pin. If SYNC is enabled, this takes precedence over any settings for GPIO2 in the GPIO/PWM Configuration T19 object (see Section 7.3 on page 89) or for Proximity Key 0 in instance 0 of the Proximity Key T52 object.

The use of the “SYNC” pin is controlled by three bits. The ENABLE bit enables the use of the “SYNC” pin. The RISING and FALLING bits turn off synchronization with the appropriate signal edge. This controls whether a rising or falling edge is used for the synchronization feature. Note that on the mXT540E, SYNC mode can be synchronized with either the rising edge or the falling edge, but not both.

ENABLE: The “SYNC” pin is enabled if set to 1, and disabled if set to 0.

DISFALL: Disables synchronization with the falling edge. Synchronization is disabled if set to 1, and enabled if set to 0.

DISRISE: Disables synchronization with the rising edge. Synchronization is disabled if set to 1, and enabled if set to 0.

The values for these three fields are listed in Table 4-10.

Table 4-10. SYNC Field Values

Bit			
DISRISE	**DISFALL**	**ENABLE**	**Effect**
0	0	0	Disabled (default)
X	X	0	Disabled
0	X	1	Enabled: falling edge disabled, synchronized with rising edge
1	X	1	Enabled: rising edge disabled, synchronized with falling edge

Note: X = indeterminate value (0 or 1)

ATCHCALST and ATCHCALSTHR Fields

The standard finger recovery process is intended to allow the sensor to recover when a finger is present on the sensor during calibration and is then subsequently removed. This process attempts to calibrate the sensor when only antitouches are detected on the sensor (see Figure 4-4). These two fields allow this process to be blocked under certain circumstances.

The ATCHCALST field sets the antitouch calibration suspend time. This is the time from the last touch release to when a standard finger recovery recalibration can occur. ATCHCALST can be configured to a value of between 0 and 255 (0 and 51s), in increments of 200 ms. A setting of zero allows the recalibration logic to work immediately after the last finger is removed from the sensor.

Note that the antitouch calibration suspend time should not be set to a period longer than the drift suspend time (DRIFTST).

The ATCHCALSTHR field sets the antitouch calibration suspend threshold. Any channel with a touch delta above this threshold will suspend all standard finger recovery recalibrations. If this field is set to zero, the standard finger recovery calibration is never blocked except by the ATCHCALST period.

Note that the ATCHCALST and ATCHCALSTHR fields control antitouch calibration on the mutual-capacitance channels only and have no effect on the self-capacitance channels controlled by a Proximity Key T52 object. Antitouch calibration on a Proximity Key T52 object is controlled by the ATCHACEN bit in the Proximity Key T52 object itself.

ATCHCALST Range: 0 to 254 (in 200 ms increments)

ATCHCALST Typical: 5 (1 second)

ATCHCALSTHR Range: 0 to 255

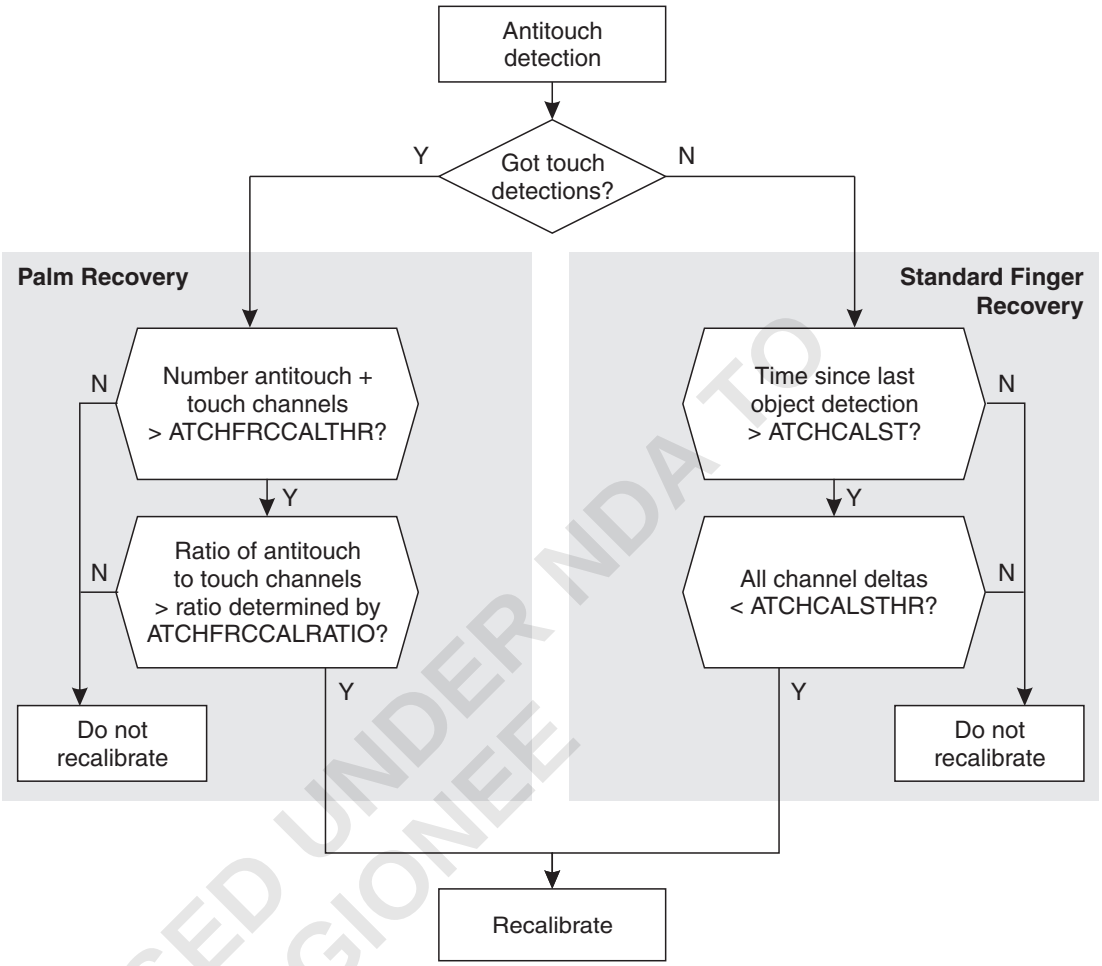
ATCHCALSTHR Typical: 0 (never suspend)

ATCHFRCCALTHR and ATCHFRCCALRATIO Fields

These two fields control the palm recovery process that allows the sensor to recover when a palm is present on the sensor during calibration and is then subsequently removed.

A palm touch during calibration results in a complex pattern of both touch and antitouch detections. The standard finger recovery recalibration process (see the ATCHCALSTHR and ATCHCALST fields), however, is blocked by any channels in touch, so a recalibration would never occur. These two fields ensure that this block is overridden and that a palm recovery calibration occurs (see Figure 4-4).

Figure 4-4. Touch Recovery Processes



The ATCHFRCCALTHR field sets the threshold to allow a palm recovery calibration. If the number of channels in touch or antitouch [1] is greater than or equal to ATCHFRCCALTHR, then the ATCHFRCCALRATIO field is used to determine if a recalibration should occur. Setting the ATCHFRCCALTHR field to zero disables both that field and the ATCHFRCCALRATIO field and a recalibration will not occur.

The ATCHFRCCALRATIO field determines the ratio of antitouch channels to total touch and antitouch channels that must be met for a recalibration to occur. This field takes a signed value that represents the desired ratio (see Table 4-11). Note that negative values should be avoided as they risk rogue calibrations when a palm is placed over the sensor.

Table 4-11. Typical Values for ATCHFRCCALRATIO

Value	Meaning
+1 to 127	Antitouch channels must be greater than 50 percent total (touch + antitouch) channels (where 127 = 100 percent total channels)
-128 to 0	Reserved; only use a value of zero or less if advised to do so by Atmel.

1. A touch channel is one in that is above +TCHTHR (touch threshold). An antitouch channel is one that is below -TCHTHR. See page 29 for more information on the TCHTHR field.

ATCHFRCCALTHR Range: 0 to 255

ATCHFRCCALTHR Typical: 50

ATCHFRCCALRATIO Range: -128 to +127

ATCHFRCCALRATIO Typical: 25 (60 percent)

4.5.2 Configuration Checks

If the CHRGTIME field (charge-transfer dwell time) is changed, the device automatically performs a recalibration and reloads the configuration settings.

Table 4-12. Configuration Checks

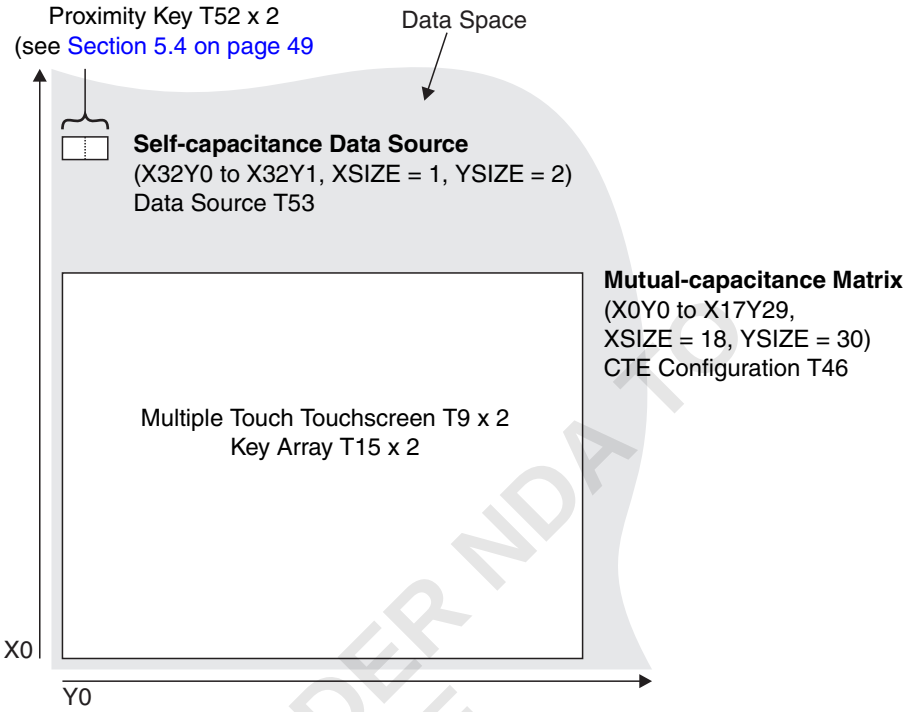
Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CHRGTIME	No	Yes	None
TCHDRIFT	No	No	None
DRIFTST	No	No	None
TCHAUTOCAL	No	No	None
SYNC	No	No	None
ATCHCALST	No	No	None
ATCHCALSTHR	No	No	None
ATCHFRCCALTHR	No	No	None
ATCHFRCCALRATIO	No	No	None

4.6 Data Source T53 (GEN_DATASOURCE_T53)

The Data Source T53 is a read-only object that describes a Data Source as a conceptual two-dimensional XY matrix. It enables touch objects to be defined in terms of XY channels even though they do not use a physical mutual-capacitance sensor. The data from these touch objects can therefore be processed by other objects within the device. This enables data from multiple sources to be described to the host in a compatible manner.

The Data Source is defined to occupy channel numbers that are not used by the mutual-capacitance channels. The arrangement of the channels on the mXT540E is given in Figure 4-5.

Figure 4-5. Mutual-capacitance and Self-capacitance Channels on mXT540E



4.6.1 Configuration

Table 0-1. GEN_DATASOURCE_T53

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	DSTYPE	Data source type = 1							
1	XORIGIN	The X start of the data source in the data space = 32							
2	XSIZE	The X size of the data source in the data space = 1							
3	YORIGIN	The Y start of the data source in the data space = 0							
4	YSIZE	The Y size of the data source in the data space = 2							

DSTYPE Field

Specifies the data source type. On the mXT540E this is always 1 (meaning a self-capacitance sensor).

XORIGIN, YORIGIN XSIZE and YSIZE Fields

Specify the start location and size of the data source within the data space. On the mXT540E the settings are:

XORIGIN = 32, YORIGIN = 0
XSIZE =1, YSIZE = 2

5. Touch Objects

5.1 Introduction

Touch objects operate on measured signals from the touch sensor and report touch data. For example, a Touchscreen object reports XY touch positions. Table 5-1 lists the touch objects on the mXT540E.

Table 5-1. Touch Objects

Object	Description
Multiple Touch Touchscreen T9 (TOUCH_MULTITOUCHSCREEN_T9)	Creates a Touchscreen that supports the tracking of more than one touch. See Section 5.2.
Key Array T15 (TOUCH_KEYARRAY_T15)	Creates a rectangular array of keys. A Key Array T15 object reports simple on/off touch information. See Section 5.3.
Proximity Key T52 (TOUCH_PROXKEY_T52	Configures a self-capacitance channel (key) for use as a proximity sensor. See Section 5.4.

5.2 Multiple Touch Touchscreen T9 (TOUCH_MULTITOUCHSCREEN_T9)

A Multiple Touch Touchscreen T9 object is used to configure a Multiple Touch Touchscreen on the mutual-capacitance sensor matrix. The mXT540E has 2 instances of the Multiple Touch Touchscreen T9 object.

5.2.1 Configuration

Table 5-2. TOUCH_MULTITOUCHSCREEN_T9

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	SCANEN	DISPRSS	DISREL	DISMOVE	DISVECT	DISAMP	RPTEN	ENABLE
1	XORIGIN	X line start position of object							
2	YORIGIN	Y line start position of object							
3	XSIZE	Number of X lines the object occupies							
4	YSIZE	Number of Y lines the object occupies							
5	AKSCFG	Group8	Group7	Group6	Group5	Group4	Group3	Group2	Group1
6	BLEN	GAIN				Reserved			
7	TCHTHR	Touch threshold							
8	TCHDI	Touch detect integration for first touch							
9	ORIENT	Reserved					INVERTY	INVERTX	SWITCH
10	MRGTIMEOUT	Merge timeout							
11	MOVHYSTI	Movement hysteresis, initial							
12	MOVHYSTN	Movement hysteresis, next							
13	MOVFILTER	DISABLE	FILTERLIMIT			ADAPTTHR			
14	NUMTOUCH	Number of reported touches							
15	MRGHYST	Merge hysteresis							
16	MRGTHR	Merge threshold							
17	AMPHYST	Amplitude hysteresis							

Table 5-2. TOUCH_MULTITOUCHSCREEN_T9 (Continued)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
18 – 19	XRANGE	X resolution							
20 – 21	YRANGE	Y resolution							
22	XLOCLIP	X low clipping boundary width							
23	XHICLIP	X high clipping boundary width							
24	YLOCLIP	Y low clipping boundary width							
25	YHICLIP	Y high clipping boundary width							
26	XEDGECTRL	SPAN	DISLOCK	CORRECTIONGRADIENT					
27	XEDGEDIST	X edge correction distance							
28	YEDGECTRL	SPAN	RELUPDATE	CORRECTIONGRADIENT					
29	YEDGEDIST	Y edge correction distance							
30	JUMPLIMIT	Maximum position jump							
31	TCHHYST	Touch threshold hysteresis							
32	XPITCH	X line pitch							
33	YPITCH	Y line pitch							
34	NEXTTCHDI	Touch detect integration for subsequent touches							

CTRL Field

ENABLE: Enables the use of this Multiple Touch Touchscreen T9 object. The object is enabled if set to 1, and disabled if set to 0. The object does not scan for touches if it is disabled, in order to conserve power.

RPTEN: Allows the object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0. Events must be enabled for this bit to have an effect.

DISAMP: Disables the touch amplitude (TCHAMPLITUDE) from being calculated and reported. This saves current and makes for a faster response time. Note, however, that this may get overridden if another process (such as stylus tracking) needs to calculate the amplitude.

DISVECT: Disables vector changes from triggering a message. This also disables the TCHVECTOR field in the message data from being calculated.

DISMOVE: If set to 1, disables position movements from triggering a message.

DISREL: If set to 1, disables release events from triggering a message.

DISPRSS: If set to 1, disables press events from triggering a message.

SCANEN: The device normally has a very efficient mechanism for determining touch separation (that is, distinguishing between multiple fingers). This is sufficient for most purposes. Under some (very unusual) circumstances, however, it is possible for it to fail to distinguish between multiple touches. Setting this bit to 1 causes the device to perform a close scan for touch detections once every 200 ms. Multiple touches are correctly detected, but the checks still occur infrequently enough to prevent excessive current consumption. This scan is done only when the touchscreen is touched.

XORIGIN, YORIGIN, XSIZE and YSIZE Fields

These fields specify the size and position of the touchscreen on the actual matrix, in terms of X and Y lines (see Figure 5-1). The XORIGIN and YORIGIN fields specify the origin and the XSIZE and YSIZE fields specify the size.

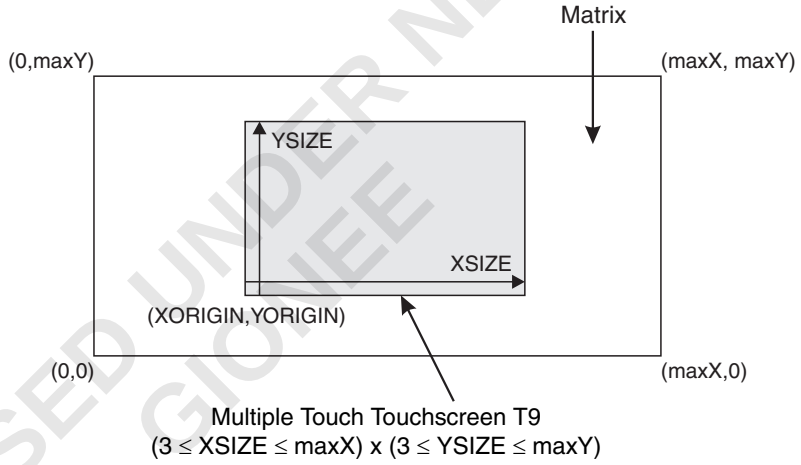
The XORIGIN and YORIGIN can be any value from zero to (maxX - XSIZE + 1) or (maxY - YSIZE + 1), as appropriate.

The minimum value for XSIZE and YSIZE is 3 when the XRANGE or YRANGE (as appropriate) is 0 to 1023. The minimum value for XSIZE and YSIZE is 7 when the XRANGE or YRANGE is 1024 to 4095.

Note: If Dual X Drive is enabled for use in the Noise Suppression T48 object, the minimium XSIZE is 4 when the XRANGE or YRANGE is 0 to 1023, and 8 when the XRANGE or YRANGE is 1024 to 4095.

The maximum size (labelled maxX and maxY in Figure 5-1) for a Multiple Touch Touchscreen T9 depends on the number of X and Y lines on a particular device.

Figure 5-1. ORIGIN and SIZE Fields



XORIGIN Range: 0 to (maxX - XSIZE + 1)

YORIGIN Range: 0 to (maxY - YSIZE + 1)

XSIZE Range: 3 to maxX (4 or 8 to maxX if Dual X Drive is enabled)

YSIZE Range: 3 to maxY

AKSCFG Field

This field configures Adjacent Key Suppression® (AKS®) between this Multiple Touch Touchscreen T9 object and any other touch present on the device.

AKS technology is a patented method used to detect which touch object is touched when objects are located close together. A touch in a group of AKS objects is indicated only on the object with the largest signal. This is assumed to be the intended object. Once an object in an AKS group is in detect, there can be no further detections within that group until the object is released.

Group1 to 8: These bits form a bit field that specifies which AKS groups this Multiple Touch Touchscreen T9 object is within. The default value of 0 means that the object is in no AKS groups and the AKS feature is disabled for the Multiple Touch Touchscreen T9.

Range: 0 to 255

BLEN Field [1]

GAIN: Sets the gain of the analog circuits in front of the analog to digital converter (ADC). The range of values for the GAIN setting is 0 to 15.

Range: 0 to 15

TCHTHR Field

The channel detection Touch Threshold value (TCHTHR) defines how much a channel's touch delta [2] must be to qualify as a potential touch detection. The reference level is determined during calibration and adjusted using drift compensation. The final detection confirmation uses the Touch Detect Integration as described in the TCHDI field. Larger values for the threshold desensitize channels, since the signal must change more in order to exceed the threshold level. Conversely, lower thresholds make channels more sensitive.

The setting for TCHTHR for each channel depends on the amount of signal swing that occurs when a channel is touched. Thicker panels or smaller electrode geometries reduce channel sensitivity (that is, signal swing from touch). In this case smaller TCHTHR values are required to detect touch.

Range: 0 to 255

Typical: 30 to 80

TCHDI Field

The Touch Detect Integration (TCHDI) field, together with the NEXTTCHDI field (see page 39), is used to provide detection filtering.

To suppress false detections caused by spurious events like electrical noise, the device incorporates a TCHDI counter mechanism. A per-touch counter is incremented each cycle that a touch is detected. When this counter reaches a preset limit the touch is finally declared to be present. If on any acquisition a delta is not seen to exceed the threshold level, the counter is cleared and the process has to start from the beginning. It takes TCHDI + 1 cycles from touchdown to when the first touch is actually reported via the $\overline{CHG}$ pin, with a minimum time of 2 cycles. Once there is a touch detected, the limit is changed to TCHDI + NEXTTCHDI. This allows a shorter detection integration for the first touch detection.

A similar process is applied when a touch goes out of detection. The counter is decremented each cycle that the delta does not exceed the threshold level, and incremented again if it does exceed the threshold. When the counter reaches zero, the touch is finally declared to be out of detect. It takes TCHDI + 1 + NEXTTCHDI cycles for touches to cease to be reported, with a minimum time of 3 cycles.

The range for this field is 0 to 255, where 0 is the same as 1.

Note: TCHDI and NEXTTCHDI are saturated to 255; that is: TCHDI + NEXTTCHDI ≤ 255

Range: 0 (1), 1 to 255

Typical: 2 to 3

ORIENT Field

The ORIENT field controls the orientation of the touchscreen, such as flipping and rotating the screen display.

1. Despite its name, the BLEN field does not control the burst length.
2. Reference minus the signal.

SWITCH: Switches the X and Y positions; that is, the screen is flipped about the diagonal from (X0, Y0) to (Xmax, Ymax).

INVERTX: Inverts X coordinates; that is: Xnewval = (Xmaxval - X).

INVERTY: Inverts Y coordinates; that is: Ynewval = (Ymaxval - Y).

Note that an INVERTX and/or INVERTY operation takes place before a SWITCH.

The effect of these three bits is shown in Table 5-3.

Table 5-3. ORIENT Field Settings

Bits			Touchscreen Coordinates	Touchscreen Orientation
2 (INVERTY)	1 (INVERTX)	0 (SWITCH)		
0	0	0	0,Ymax Xmax,Ymax 0,0 Xmax,0	Normal orientation
0	1	0	Xmax,Ymax 0,Ymax Xmax,0 0,0	Horizontal flip
1	0	0	0,0 Xmax,0 0,Ymax Xmax,Ymax	Vertical flip
1	1	0	Xmax,0 0,0 Xmax,Ymax 0,Ymax	Rotated 180°
0	0	1	Xmax,0 Xmax,Ymax 0,0 0,Ymax	Diagonal mirror along axis from (X0, Y0) to (Xmax, Ymax)
0	1	1	Xmax,Ymax Xmax,0 0,Ymax 0,0	Rotated 90° counterclockwise
1	0	1	0,0 0,Ymax Xmax,0 Xmax,Ymax	Rotated 90° clockwise
1	1	1	0,Ymax 0,0 Xmax,Ymax Xmax,0	Diagonal mirror along axis from (X0, Ymax) to (Xmax, Y0)

Note: Xmax and Ymax refer to the maximum X and Y screen positions; that is, they relate to the touchscreen resolution and not to the XY channels.

MRGTIMEOUT Field

When two or more touches move too close to each other, they can no longer be distinguished from each other. The touches are therefore merged into a single touch. One touch continues to be reported and the others are no longer reported. When this happens, the device enters a close scan mode that performs a detailed analysis of the touch.

The Merge Timeout (MRGTIMEOUT) field controls the length of time that the device is in close scan mode. It determines how long the device continues to scan the merged touch. If the fingers are pulled apart again, this can be detected as early as possible. The Merge Timeout is specified in increments of 200 ms.

The unreported touch ID is reserved during the Merge Timeout. It can then be reassigned to one of the touches if the touches move apart again during the timeout. If the touches have not moved apart by the end of the period, the touches become one and the touch ID is released for future use.

Range: 0 to 255 (in 200 ms increments)

MOVHYSTI and MOVHYSTN Fields

These fields are applied to the reported touch positions to provide simple filtering.

When the user first touches the touchscreen (that is, when the user's finger has just touched the touchscreen), the Initial Movement Hysteresis (MOVHYSTI) setting is applied to the reported position. Once the user's finger starts to move, the MOVHYSTI must be exceeded [1] for position updates to be reported. This allows the host to differentiate between an intended drag and a simple press.

Once the MOVHYSTI limit has been exceeded, the Next Movement Hysteresis (MOVHYSTN) setting is applied to the calculated positions. This filters direction changes in either axis, allowing a simple jitter filter to be implemented. Higher settings for MOVHYSTN result in a stronger filter.

The MOVHYSTN setting is internally reduced while the touch is moving. This avoids "steps" when drawing, whilst still allowing jitter to be suppressed when the touch is stationary.

If MOVHYSTI is not required, set it to the same value as the MOVHYSTN setting.

The units for these fields are the least significant bits of position.

Range: 0 to 255

MOVHYSTN Typical: 0 to 10

MOVHYSTI Typical: 50

MOVFILTER Field

The Movement Position Filter (MOVFILTER) setting allows the host to change the level of filtering applied to the calculated touchscreen position.

ADAPTTHR: The Adapt Threshold determines the level at which filtering is applied. This value can be tuned, based on the device's programmed cycle time, resolution setting and the level of system noise. Under normal circumstances the default setting is recommended. If the resolution of the touchscreen is modified using the XRANGE and YRANGE fields, the Adapt Threshold may need adjusting to achieve the same performance as at the default resolution.

This value is a signed (two's complement) 4-bit value, where -8 = low filtering, 0 = medium filtering and +7 = high filtering. Some typical settings are shown in Table 5-4.

1. In a positive or negative X or Y direction (that is, in one of four directions).

Table 5-4. Typical Settings for ADAPTTHR

Resolution	Suggested ADAPTTHR Setting
8-bit XRANGE/YRANGE = 255	-2
9-bit XRANGE/YRANGE = 511	-1
10-bit XRANGE/YRANGE = 0 or 1023	0
11-bit XRANGE/YRANGE = 2047	1
12-bit XRANGE/YRANGE = 4095	2

FILTERLIMIT: The Filter Limit limits the maximum amount of filtering that can be applied to the touchscreen position. This field has a range of 0 to 7, indicating the number of steps to reduce the maximum filtering by. A setting of zero (the default) or 7 has no effect. A setting of 1 reduces the filtering by 1 step, a setting of 2 by 2 steps, and so on up to 6.

DISABLE: Set to 1 to disable the filter, set to 0 to enable it.

ADAPTTHR Range: -8 (low) to +7 (high)

ADAPTTHR Typical: See Table 5-4

FILTERLIMIT Range: 0 (no effect),1 to 6 (number of steps), 7 (0 = no effect)

FILTERLIMIT Typical: 0 (no effect)

NUMTOUCH Field

The Number of Touches (NUMTOUCH) setting indicates the number of touches to be reported. Report IDs are not removed. These are fixed to the maximum number of touches (16 on the mXT540E).

Range: 1 to 16 (maximum number of touches)

MRGHYST Field

The Merge Hysteresis (MRGHYST) setting is used in conjunction with the MRGTHR field. It provides a hysteresis to ensure that once two touches have merged (and a single touch is reported), the distance needed for separation before two touches are reported again is slightly larger than for convergence. This can help to stop oscillations between one and two touches being reported at the boundary of convergence. This field is specified in units of signal delta.

Note that the sum of MRGTHR and MRGHYST must not exceed 255. Note also that MRGTIMEOUT must be set up for the touches to stay merged and for MRGHYST to apply.

Range: 0 to 255

Typical: 5

MRGTHR Field

The Merge Threshold (MRGTHR) setting allows the host to tune the amount of unevenness in a measured area that the device still considers as one individual touch. Beyond this threshold the device considers there to be multiple touches present on the sensor matrix.

This value may be used to allow the grouping of a large and uneven touch (such as a user's palm) into a single tracked touch. Larger values allow more uneven touches to be tracked as a single touch, but reduce the minimum separation of two true touches. This field is specified in units of signal delta.

Note that the sum of MRGTHR and MRGHYST should not exceed 255.

Range: 0 to 255

Typical: 5

AMPHYST Field

The Amplitude Hysteresis (AMPHYST) setting is used to filter touch amplitude changes. The Amplitude Hysteresis setting is applied to the reported amplitude of a touch (see "TCHAMPLITUDE Field" on page 43). If the amplitude has changed by more than the AMPHYST value from its last reported value, the amplitude value is updated. This ensures that only deliberate amplitude changes are reported and reduces communications traffic.

Range: 0 to 255

XRANGE/YRANGE, XLOCLIP/YLOCLIP and XHICLIP/YHICLIP Fields

These six fields control the resolution, and thus the reported position, of the touchscreen.

Note: The XRANGE/YRANGE, XLOCLIP/YLOCLIP and XHICLIP/YHICLIP fields operate on the physical ITO matrix of the touchscreen, and are not affected by the logical orientation set by the ORIENT field.

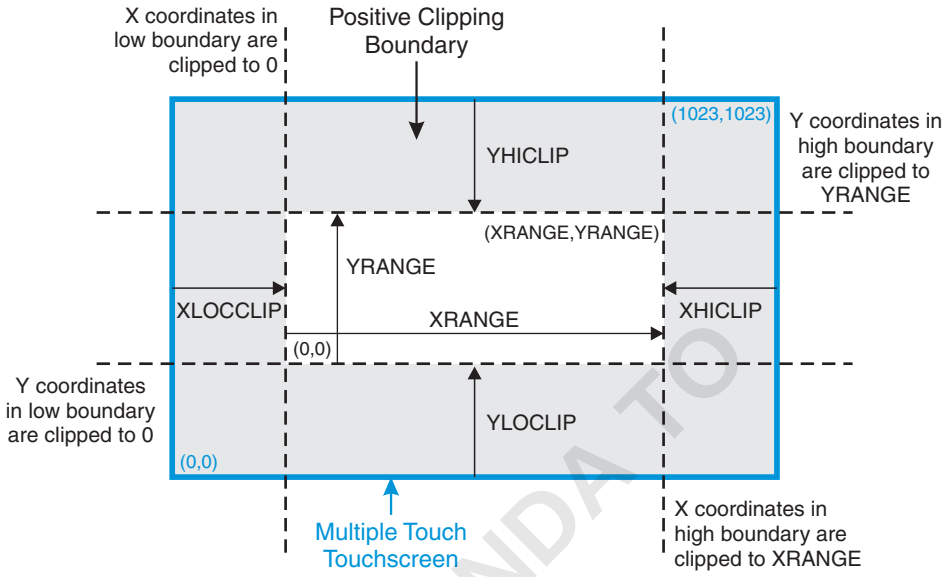
The XRANGE and YRANGE fields set the output resolution for the reported position. These two fields can be set from 0 to 4095 (that is, 12-bit resolution). A setting of 0 (default) sets the resolution to 1023. These fields allow a touchscreen's position to match an LCD's resolution. For example, to set up a touchscreen to match an LCD with a resolution of 800 x 600, XRANGE and YRANGE would be set to 799 and 599 respectively.

XRANGE and YRANGE determine the format of reported positions in messages from Touchscreen and Gesture Processor objects. If XRANGE/YRANGE is below 1024, 10-bit positions are reported. If XRANGE/YRANGE is 1024 or above, 12-bit positions are reported.

The XLOCLIP/YLOCLIP and XHICLIP/YHICLIP fields set up a clipping boundary (illustrated by the dark grey regions in Figure 5-2 and Figure 5-3). Any touch position within this boundary has its X or Y coordinate clipped to the minimum or maximum value. These fields take a signed (two's complement) 8-bit value, allowing settings in the range -128 to +127. These fields work at 10-bit resolution (that is, a touchscreen range of 1023). This means that the maximum clipping values allow a clipping boundary one eighth of the screen width or height.

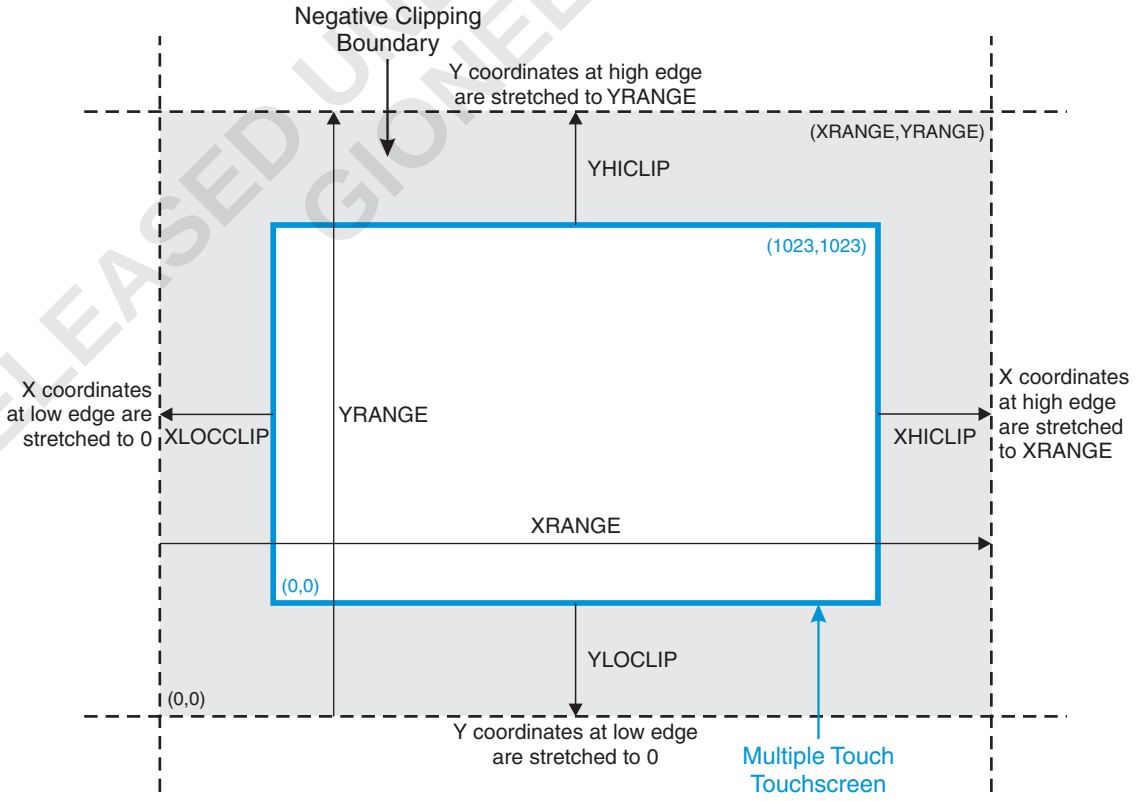
Positive values increase the size of the clipping boundary, moving the 0 and XRANGE/YRANGE reported positions further inside the screen (see Figure 5-2). If the touch enters the clipping boundary, the coordinates are locked at the point at which the touch entered the region. The coordinates are updated only when the touch leaves the clipping boundary. This coordinate locking can be disabled using the DISLOCK field.

Figure 5-2. Resolution Fields – Positive Clipping Boundary



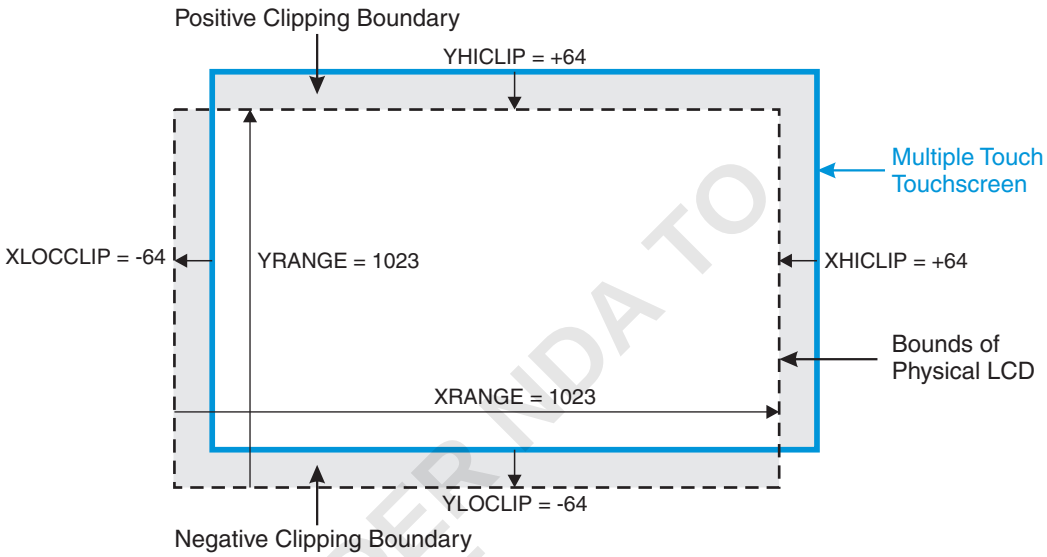
Negative values stretch the 0 and XRANGE/YRANGE positions outside of the screen (see Figure 5-3). This limits the reported minimum and maximum value.

Figure 5-3. Resolution Fields – Negative Clipping Boundary



Adjusting the clipping boundary modifies the accuracy of the reported positions. For example, if the touchscreen is smaller than the physical LCD or is not aligned with the LCD, a combination of positive and negative clipping boundaries help align the reported zero point with the zero point of the LCD. An example of this is shown in Figure 5-4.

Figure 5-4. Example Resolution Fields



This example results in the coordinates shown in Figure 5-5 with the coordinate mapping shown in Figure 5-6.

Figure 5-5. Example Resolution Fields – Resulting Coordinates

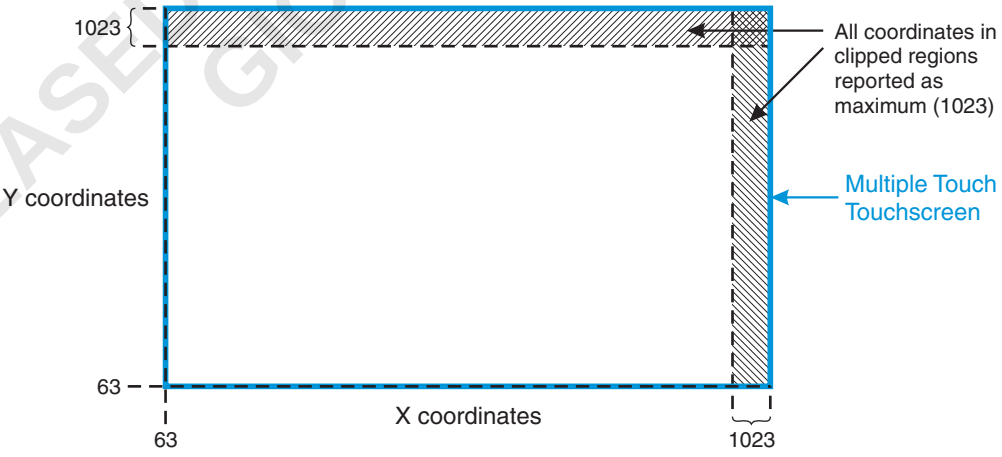
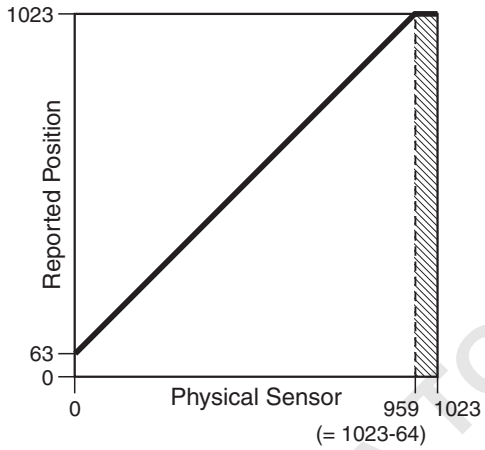


Figure 5-6. Example Resolution Fields – Mapping of Reported Positions



These six resolution controls operate first on the positional data received from the touchscreen. The other touchscreen controls will be relative to the resolution set. For example, a hysteresis setting of 2 will have a different physical meaning on the same screen if the resolution is 200 or 2000. Figure 5-7 shows the processing order of all the touchscreen settings and shows how the resolution controls affect the other settings.

Figure 5-7. Order of Touchscreen Settings

XRANGE/YRANGE Range: 0 (1023; 10-bit resolution), 127 to 4095

XRANGE/YRANGE Default: 0 (1023; 10-bit resolution), 127 to 4095

XLOCLIP/YLOCLIP/XHICLIP/YHICLIP Range: -128 to +127

XEDGECTRL and YEDGECTRL Fields

The XEDGECTRL and YEDGECTRL fields works with the XEDGEDIST and YEDGEDIST fields respectively to perform edge correction.

Note: These fields operate on the physical ITO matrix of the touchscreen, and are not affected by the logical orientation set by the ORIENT field.

CORRECTIONGRADIENT: Specifies the gradient to be used in the edge correction calculation for the appropriate axis. A setting of 0 disables edge correction. A value of 1 to 63 sets the gradient value.

CORRECTIONGRADIENT Range: 0 (disable), 1 to 63

CORRECTIONGRADIENT Typical: 9

DISLOCK (XEDGECTRL only): Disables the coordinate position locking that is applied to touches inside a clipping boundary (see "XRANGE/YRANGE, XLOCLIP/YLOCLIP and XHICLIP/YHICLIP Fields" on page 33 for more information). Set this field to 1 to disable locking, and to 0 to enable locking. This field affects both the X and Y coordinates. (1)

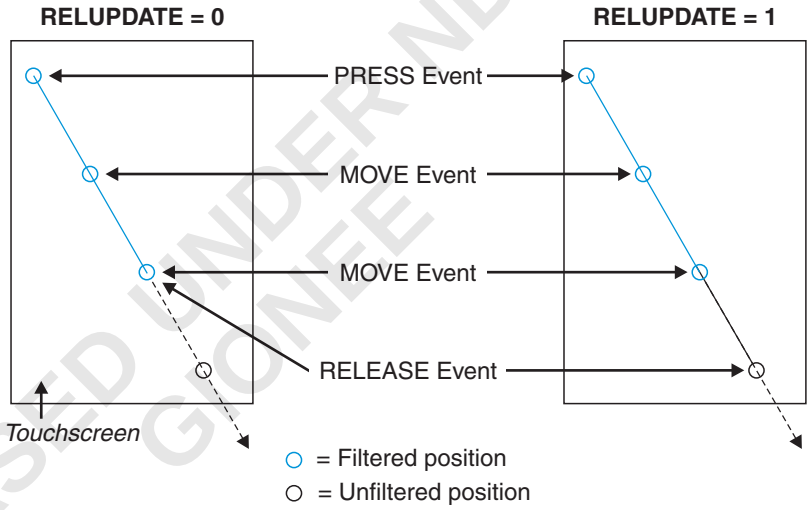
1. Even though it is present in the XEDGECTRL field (only) and not in the YEDGECTRL field.

RELUPDATE (YEDGECTRL only): Enables the touch release update mechanism. This mechanism ensures that the touch position is reported accurately when a touch moves off the edge of the touchscreen.

The Multiple Touch Touchscreen T9 object reports filtered touch positions, not the raw positions. When the touch moves off the touchscreen and is released, the reported position may not be the actual point at which the touch left the touchscreen. This mechanism ensures that the last (raw) position is reported and not the last filtered position for greater accuracy. Set this field to 1 to enable the touch release update mechanism, and to 0 to disable it. This field affects both the X and Y coordinates [1].

Figure 5-8 shows the effect of the RELUPDATE field when a touch moves off the touchscreen and generates a RELEASE event. The position reported by the event depends on the setting of RELUPDATE. If RELUPDATE is set to 0, the reported position is the same as the filtered position reported by the last MOVE event. If RELUPDATE is set to 1, the reported position is the raw position where the touch left the touchscreen.

Figure 5-8. RELUPDATE Field



Note that enabling the touch release update mechanism may affect the positions used by any post-processing objects (for example, gesture processing).

SPAN: At the edges of a touchscreen sensor pattern there should ideally be a half width electrode. Depending on how the screen has been designed, there may actually be an electrode that is somewhere between 50 percent and 100 percent width at the edge. This can cause discrepancies between the physical touch position and the reported touch position. The SPAN field helps correct these discrepancies.

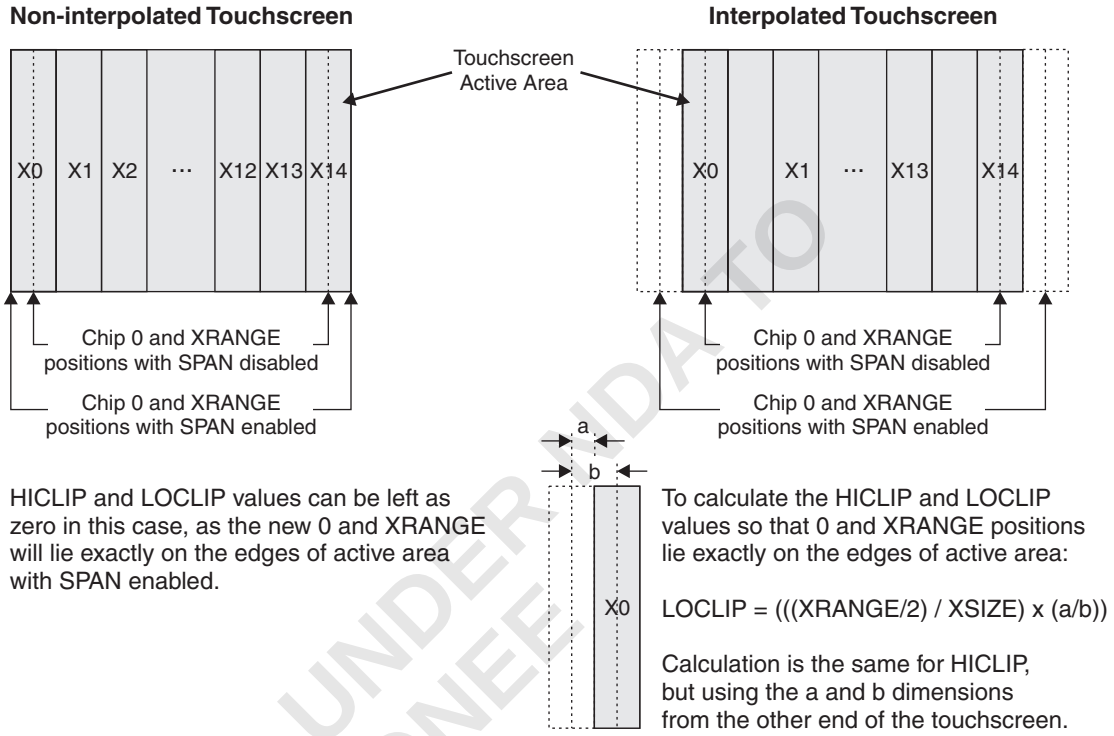
The SPAN field is used in combination with XLOCLIP/YLOCLIP and XHICLIP/YHICLIP settings. This allows the device to correct for accuracy or linearity problems resulting from the practicalities of the ITO design.

The SPAN field moves the theoretical 0 and XRANGE/YRANGE position outwards from the screen center by one half electrode width each. The XLOCLIP/YLOCLIP and XHICLIP/YHICLIP controls can then be used to move the theoretical 0 and XRANGE/YRANGE position back in towards the center in finer steps.

1. Even though it is present in the YEDGECTRL field (only) and not in the XEDGECTRL field.

Figure 5-9 shows an example of how to correct for two screen designs with an X size of 15. Note that although Figure 5-9 shows the SPAN control for the X lines, the same principles apply when the SPAN bit is used with the Y lines.

Figure 5-9. Correcting for Touchscreen Designs (X Coordinates)



XEDGEDIST and YEDGEDIST Fields

The XEDGEDIST and YEDGEDIST fields work with the XEDGECTRL and YEDGECTRL fields to perform edge correction. This improves linearity at the edge of the screen on certain touchscreen designs.

Note: These fields operate on the physical ITO matrix of the touchscreen, and are not affected by the logical orientation set by the ORIENT field.

For low-end positions, the corrected position is calculated as follows (for X):

position - (((XEDGEDIST - position) x X_CORRECTIONGRADIENT) / 16)

OR (for Y):

position - (((YEDGEDIST - position) x Y_CORRECTIONGRADIENT) / 16)

For high-end positions, the corrected position is calculated as follows (for X):

position + (((position - (1023 - XEDGEDIST)) x X_CORRECTIONGRADIENT) / 16)

OR (for Y):

position + (((position - (1023 - YEDGEDIST)) x Y_CORRECTIONGRADIENT) / 16)

Note that these fields work at 10-bit resolution (that is, a touchscreen range of 1023).

Range: 0 to 255

Typical: Half expected touch diameter

JUMPLIMIT Field

This field specifies the maximum touchscreen distance a touch can move in one cycle. If this limit is exceeded, the touch is classified as a new touch and all touchdown logic takes place on the new touch.

Note that low values can hinder fast movements, such as flicks.

The range for this field is 0 to 255, where 0 is off and 1 to 255 is the distance in multiples of 8 (allows an effective range of 8 to 2040). The range is in units of touchscreen position at the configured resolution.

Range: 0 (off), 1 to 255 (in multiples of 8)

TCHHYST Field

This field controls the level of hysteresis applied to touch detections. For a touch to enter detection the touch delta must be greater than the touch threshold (TCHTHR). For a touch to leave detection the touch delta must be less than (TCHTHR - TCHHYST). This field allows the Multiple Touch Touchscreen T9 to be tuned so that a hovering finger does not cause detection to flicker on and off.

TCHHYST is capped internally to (TCHTHR / 4). A hysteresis greater than 25 percent of the Touch Threshold is therefore not possible.

A setting of 0 means that there is no hysteresis.

Range: 0 (off), 1 to 255 (hysteresis)

XPITCH and YPITCH Fields

These fields specify the physical pitch [1] of the X and Y lines on the touchscreen sensor. These settings are specified in units of 0.1 mm, where a setting of zero means 5 mm.

Range: 0 (5 mm), 1 to 255 (in 0.1 mm units)

NEXTTCHDI Field

This field, together with the TCHDI field, provides Touch Detect Integration. Specifically the NEXTTCHDI field allows for a longer integration period once at least one touch has been detected. This may be needed for particularly noisy touchscreens.

See "TCHDI Field" on page 29 for a more detailed description.

The range for this field is 0 to 255.

Range: 0 to 255

5.2.2 Configuration Checks

A Multiple Touch Touchscreen T9 object causes a configuration check to be performed in the following circumstances:

- When the object is enabled (that is, the ENABLE bit is set in the CTRL field)
- If the object is enabled, when certain fields are changed (as listed in Table 5-5)

In addition, some fields will cause an automatic recalibration to be performed (see Table 5-5).

1. That is, the spacing between the centers of the X or Y lines.

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). [1] This is signaled to the host (see Section 4.3.2 on page 14). The device halts until the error has been corrected. To fix the error, check that all the object settings are within their allowed limits, as stated in the field descriptions.

The XORIGIN/YORIGIN and XSIZE/YSIZE settings should be checked to ensure that:

- Part of the object is not placed off the edge of the sensor matrix. The matrix size is specified in the Information Block (see Section 2.2 on page 2)

  OR
- The object does not overlap with other enabled touch objects

  OR
- The object does not occupy the same Y line as another enabled object

The XSIZE/YSIZE settings should be checked to ensure that:

- Greater than 2 if 10-bit resolution or less is configured

  OR
- Greater than 6 if a higher than 10-bit resolution is configured

Note that this is on a per-axis basis. For example, on a 6 x 9 touchscreen it is possible to have 10-bit resolution on one axis and 12-bit resolution on the other.

Table 5-5. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes [1]	Yes [2]	None
XORIGIN	Yes	Yes	Error if out of range
YORIGIN	Yes	Yes	Error if out of range
XSIZE	Yes	Yes	Error if out of range
YSIZE	Yes	Yes	Error if out of range
AKSCFG	No	No	None
BLEN	No	Yes	None
TCHTHR	Yes	No	None
TCHDI	No	No	None
ORIENT	Yes	No	None
MRGTIMEOUT	No	No	None
MOVHYSTI	No	No	None
MOVHYSTN	No	No	None
MOVFILTER	No	No	None
NUMTOUCH	No	No	None
MRGHYST	No	No	None

1. While the object is disabled, however, potential configuration errors are allowed.

Table 5-5. Configuration Checks (Continued)

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
MRGTHR	No	No	None
AMPHYST	No	No	None
XRANGE	Yes	No	Error if out of range
YRANGE	Yes	No	Error if out of range
XLOCLIP	Yes	No	None
XHICLIP	Yes	No	None
YLOCLIP	Yes	No	None
YHICLIP	Yes	No	None
XEDGECTRL	Yes	No	None
XEDGEDIST	Yes	No	None
YEDGECTRL	Yes	No	None
YEDGEDIST	Yes	No	None
JUMPLIMIT	No	No	None
TCHHYST	Yes	No	None
NEXTTCHDI	No	No	None

1. If the ENABLE bit is toggled on or off.
2. If the ENABLE bit is toggled on or off. Note that if the ENABLE bit is cleared and *all of* the touch objects are now disabled, no calibration takes place.

5.2.3 Messages

Table 5-6 shows the message data for a Multiple Touch Touchscreen T9 object. This message is reported for a single touch on the touchscreen. The report ID in the resulting Message Processor T5 object will indicate to which of the 16 touches this message data refers (see Section 4.2 on page 11). If the reported touch forms a gesture, this message may also be followed by a message from a Gesture Processor object linked to the Multiple Touch Touchscreen T9 object (see Section 6.2.2 on page 62 and Section 6.3.2 on page 65).

Table 5-6. Message Data for TOUCH_MULTITOUCHSCREEN_T9

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	DETECT	PRESS	RELEASE	MOVE	VECTOR	AMP	SUPPRESS	UNGRIP
2	XPOSMSB	X position MSByte							
3	YPOSMSB	Y position MSByte							
4	XYPOSLSB	X position lsbits				Y position lsbits			
5	TCHAREA	Size of touch							
6	TCHAMPLITUDE	Touch amplitude (sum of measured deltas)							
7	TCHVECTOR	Component 1				Component 2			

Note: The format for the XYPOSLSB fields depend on the resolution (10-bit or 12-bit); see page 33.

STATUS Field

Reports the current status of the touch. The touch is active if the DETECT bit is set. The SUPPRESS, AMP, MOVE, VECTOR, PRESS and RELEASE bits indicate which events have occurred to this touch since the last message was read by the host. Note that multiple bits may be set if the host is slow to read the status messages, indicating all the events that have happened.

UNGRIP: The detected touch was previously suppressed because the Grip Suppression T40 object is in nonlocking mode. That is, the touch was previously suppressed in the grip boundary but has now moved into the active region and become detected. See Section 6.4 on page 66 for more details on nonlocking grip suppression mode. This bit remains set until the touch is removed.

SUPPRESS: The detected touch has just been suppressed by the Touch Suppression T42 object linked to this Multiple Touch Touchscreen T9 object (see Section 6.5 on page 68).

AMP: The amplitude of the detected touch has just changed.

VECTOR: A touch vector change has occurred.

MOVE: The detected touch has just been moved (received from a moving touch).

RELEASE: The previously reported touch has just been removed from the sensor.

PRESS: The detected touch has just been put on the sensor.

DETECT: The touch is present on the screen.

XPOSMSB, YPOSMSB and XYPOSLSB Fields

These three fields report the X and Y position. XPOSMSB/YPOSMSB contains the most significant byte of the position. XYPOSLSB contains the least significant bits of the position.

The position has one of two formats, depending on whether it is reporting a 10-bit or 12-bit resolution. This is determined by the setting in the XRANGE or YRANGE field, as appropriate. If the XRANGE/YRANGE setting is less than 1024, the 10-bit format is used. If the XRANGE/YRANGE setting is 1024 to 4095, the 12-bit format is used. Note that the X and Y position formats are independent. For example, a message might contain a 10-bit X position and a 12-bit Y position.

The formats for the X and Y positions are shown in Table 5-7 and Table 5-8.

Table 5-7. X Position Formats

XPOSMSB								XYPOSLSB							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
10-bit Format															
512	256	128	64	32	16	8	4	2	1	N/A		Y position lsbits			
12-bit Format															
2048	1024	512	256	128	64	32	16	8	4	2	1	Y position lsbits			

Table 5-8. Y Position Formats

YPOSMSB								XYPOSLSB							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
10-bit Format															
512	256	128	64	32	16	8	4	X position lsbits				2	1	N/A	
12-bit Format															
2048	1024	512	256	128	64	32	16	X position lsbits				8	4	2	1

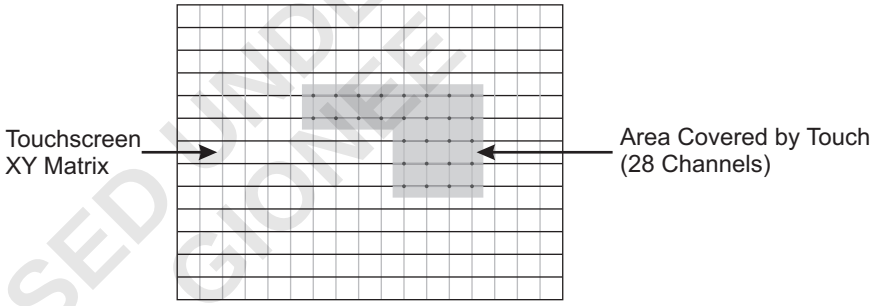
TCHAREA Field

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

A reported size of zero indicates that the reported touch is a stylus from the Stylus T47 object linked to this Multiple Touch Touchscreen T9.

A reported size of 1 or greater indicates that the reported touches are from a finger touch (or a large object, such as a palm or a face). For example, the area covered by the touch in Figure 5-10 is 28 channels.

Figure 5-10. Touch Area



This field can be used to distinguish between a “normal” finger touch and a large object, such as a palm or a face.

TCHAMPLITUDE Field

Reports a value that is proportional to the signal delta of the channels within the touch. This can be used, for example, to detect the size or pressure of the user’s finger touch. Touch amplitudes are reported only if the delta change in the value is greater than the Amplitude Hysteresis (see “AMPHYST Field” on page 33). Amplitude calculation and reporting can be disabled in the CTRL field (see page 27) to reduce processing time and communications traffic.

Note that if this field is disabled, it is considerd invalid. It may still change, however, if another process (such as stylus tracking) needs to calculate this data.

TCHVECTOR Field

The touch vector field gives an indication of the direction of the touch and a confidence level. The DISVECT bit in the CTRL field must be cleared (enabled) for the touch vector to be generated. Setting the DISVECT bit (disabled) will stop both the reporting of the vector and the calculation of the vector.

The byte is a vector made up of two signed (two's complement) 4-bit components:

- Component 1: ratio of (+X,+Y) to (-X,-Y), representing a diagonal (value: -7 to +7)
- Component 2: ratio of X to Y (value: -7 to +7)

The magnitude of the vector represents the confidence. The vector uses the following calculations:

$$\text{Angle} = \tan^{-1}\left(\frac{\text{component}_1}{\text{component}_2}\right) / 2$$

$$\text{Magnitude} = \sqrt{\text{component}_1^{\,2} + \text{component}_2^{\,2}}$$

Note: The touch vector is affected by the orientation of the screen (see "ORIENT Field" on page 29). In the example touch directions shown in Table 5-9 the normal orientation is assumed.

Table 5-9. Touch Direction (Normal Orientation Assumed)

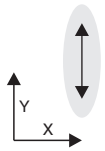
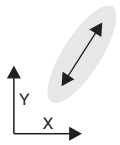
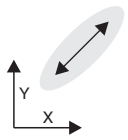
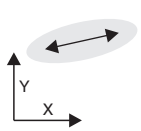
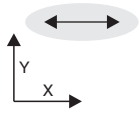
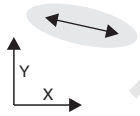
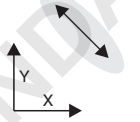
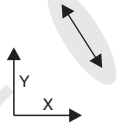
TCHVECTOR Field		
Component 1	**Component 2**	**Touch Direction**
0	Positive	
Negative	Positive	
Negative	0	
Negative	Negative	

Table 5-9. Touch Direction (Normal Orientation Assumed) (Continued)

TCHVECTOR Field		
Component 1	**Component 2**	**Touch Direction**
0	Negative	
Positive	Negative	
Positive	0	
Positive	Positive	

5.3 Key Array T15 (TOUCH_KEYARRAY_T15)

A Key Array T15 object is used to configure a rectangular array of XY channels on the mutual-capacitance sensor for use as keys.

The key numbers are assigned to X lines in Y–X order, as defined by the following equation:

Key number = (Xline - XORIGIN) + ((Yline - YORIGIN) x XSIZE)

For example, for a Key Array T15 of 5 X by 3 Y lines, the key numbers are allocated as in Table 5-10

Table 5-10. Example Key Numbers

	X*n*	**X*n*+1**	**X*n*+2**	**X*n*+3**	**X*n*+4**
Y*m*+2	10	11	12	13	14
Y*m*+1	5	6	7	8	9
Y*m*	0	1	2	3	4

5.3.1 Configuration

Table 5-11. TOUCH_KEYARRAY_T15

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	INTAKSEN	Reserved					RPTEN	ENABLE
1	XORIGIN	X line start position of object							
2	YORIGIN	Y line start position of object							
3	XSIZE	Number of X lines the object occupies							
4	YSIZE	Number of Y lines the object occupies							
5	AKSCFG	Group8	Group7	Group6	Group5	Group4	Group3	Group2	Group1
6	BLEN	GAIN				Reserved			
7	TCHTHR	Touch threshold							
8	TCHDI	Touch detect integration							
9 – 10	Reserved	Reserved							

CTRL Field

ENABLE: Enables the use of this Key Array T15 object. The object is enabled if set to 1, and disabled if set to 0.

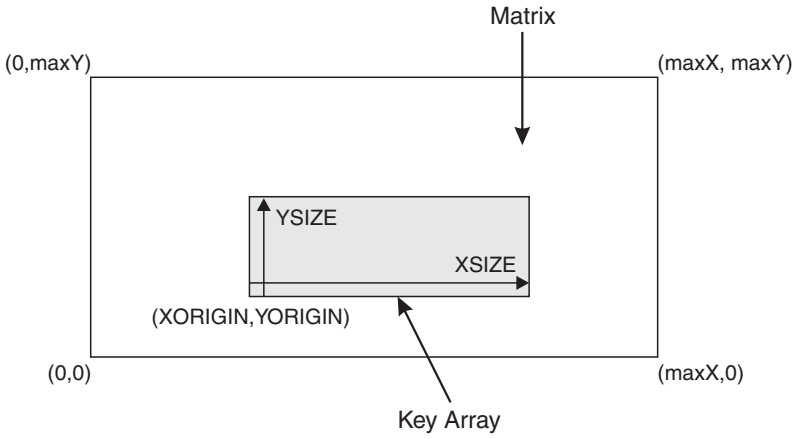
RPTEN: Allows the object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

INTAKSEN: Enables the internal Adjacent Key Suppression® (AKS®) between keys in the array. If internal AKS is enabled, then when one key is touched, touches on all the other keys within the Key Array are suppressed. Set this bit to 1 to enable internal AKS, and to 0 to disable internal AKS.

XORIGIN, YORIGIN, XSIZE and YSIZE Fields

These fields specify the size and position of the Key Array on the actual matrix (Figure 5-11). The XORIGIN and YORIGIN fields specify the origin in X and Y lines. The XSIZE and YSIZE fields specify the size in X/Y channels (keys).

Figure 5-11. Key Array T15 Fields



The minimum size for an enabled Key Array is 1 X by 1 Y lines. The maximum size for a Key Array is such that when XSIZE and YSIZE are multiplied together they obey the rule: XSIZE x YSIZE ≤ 32 channels. Setting XSIZE and YSIZE to values that create more than 32 keys causes a configuration error.

AKSCFG Field

This field configures Adjacent Key Suppression (AKS) between this Key Array T15 object and any other Touchscreen or Key Array T15 objects.

AKS technology is a patented method used to detect which touch object is touched when objects are located close together. A touch in a group of AKS objects is indicated only on the object with the largest signal. This is assumed to be the intended object. Once an object in an AKS group is in detect, there can be no further detections within that group until the object is released.

Group1 to 8: These bits form a bit field that specifies which AKS groups this Key Array is within. The default value of 0 means that the object is in no AKS groups and the AKS feature is disabled for the Key Array.

Range: 0 to 255

BLEN Field [1]

GAIN: Sets the gain of the analog circuits in front of the analog to digital converter (ADC). The range of values for the GAIN setting is 0 to 15.

Range: 0 to 15

TCHTHR Field

The channel detection Touch Threshold value (TCHTHR) defines how much a channel's touch delta [2] must be to qualify as a potential touch detection. The reference level is determined during calibration and adjusted using drift compensation. The final detection confirmation uses the Touch Detect Integration as described in the TCHDI field. Larger values for the threshold desensitize channels since the signal must change more in order to exceed the threshold level. Conversely, lower thresholds make channels more sensitive.

The setting for TCHTHR for each channel depends on the amount of signal swing that occurs when a channel is touched. Thicker panels or smaller electrode geometries reduce channel gain, that is signal swing from touch. In this case smaller TCHTHR values are required to detect touch.

Note that the Touch Threshold has a hysteresis of 2 counts, which should be allowed for in the setting.

Range: 2 to 255

TCHDI Field

To suppress false detections caused by spurious events like electrical noise, the device incorporates a detection integrator (TCHDI) counter mechanism to provide signal filtering. A per-key counter is incremented each cycle that a touch is detected. When this counter reaches the TCHDI setting limit the touch is finally declared to be present. If on any acquisition a delta is not seen to exceed the threshold level, the counter is cleared and the process has to start from the beginning.

1. Despite its name, the BLEN field does not control the burst length.
2. Reference minus the signal.

An opposite process is applied when a key leaves detection. The counter is decremented each cycle that the delta does not exceed the threshold level, and incremented again if it does exceed the threshold. When the counter reaches zero, the touch is finally declared to be out of detect. In this case there is an additional extra cycle (that is, the number of cycles is TCHDI + 1).

The range for this field is 0 to 255, where 0, 1 and 2 is the same as 2.

Range: 0 (2), 1 (2), 2 to 255

5.3.2 Configuration Checks

A Key Array T15 object causes a configuration check to be performed in the following circumstances:

- When the object is enabled (that is, the ENABLE bit is set in the CTRL field)
- When certain fields are changed (as listed in Table 5-12)

In addition, some fields will cause an automatic recalibration to be performed (see Table 5-12).

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14), and the device halts until the error has been corrected. To fix the error, the object settings should be checked to verify that they are all within their allowed limits, as described in the field descriptions.

The following should be checked:

- XSIZE is greater than or equal to 1
- YSIZE is greater than or equal to 1
- There are no more than 32 keys defined, that is: XSIZE x YSIZE ≤ 32

Table 5-12. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes (1)	Yes (2)	None
XORIGIN	Yes	Yes	Error if out of range
YORIGIN	Yes	Yes	Error if out of range
XSIZE	Yes	Yes	Error if out of range
YSIZE	Yes	Yes	Error if out of range
AKSCFG	No	No	None
BLEN	No	Yes	None
TCHTHR	Yes	No	Error if out of range
TCHDI	No	No	None

1. If the ENABLE bit is toggled on or off.
2. If the ENABLE bit is toggled on or off. Note that if the ENABLE bit is cleared and *all of* the touch objects are now disabled, no calibration takes place.

5.3.3 Messages

A Key Array T15 object reports on/off touch information in its message data. The message data for a Key Array T15 object is shown in Table 5-13.

Table 5-13. Message Data for TOUCH_KEYARRAY_T15

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	DETECT	Reserved						
2	KEYSTATE	KEY7	KEY6	KEY5	KEY4	KEY3	KEY2	KEY1	KEY0
3		KEY15	KEY14	KEY13	KEY12	KEY11	KEY10	KEY9	KEY8
4		KEY23	KEY22	KEY21	KEY20	KEY19	KEY18	KEY17	KEY16
5		KEY31	KEY30	KEY29	KEY28	KEY27	KEY26	KEY25	KEY24

STATUS Field

Reports the current status of the object.

DETECT: Set if any key is in a touched state.

KEYSTATE Fields

Report the state of each key, one bit per key in the Key Array; 0 = key is untouched, 1 = key is touched.

5.4 Proximity Key T52 (TOUCH_ PROXKEY_T52)

A Proximity Key T52 object is used to configure a self-capacitance Proximity Key. This object operates on data from a self-capacitance Data Source, as described by a Data Source T53 object (see Section 4.6 on page 24). The Proximity Key T52 object's XORIGIN and YORIGIN fields identify where within the Data Source the Proximity Key resides. The size of a Proximity Key is always 1.

Note: If any Proximity Key T52 object is enabled, power saving is reduced. This means that the proximity functionality works even when the device is in a low power mode.

5.4.1 Configuration

Table 5-14. TOUCH_PROXKEY_T52

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	DISFXDTHR	DISMVRISE	DISMVFALL	ATCHACEN	TCHACEN	CAL	RPTEN	ENABLE
1	XORIGIN	X position of the object in the Data Source							
2	YORIGIN	Y position of the object in the Data Source							
3 – 4	Reserved	Reserved							
5	AKSCFG	Group8	Group7	Group6	Group5	Group4	Group3	Group2	Group1
6	Reserved	Reserved							
7 – 8	FXDDTHR (1)	Fixed detection threshold LSByte							
		Reserved	Fixed detection threshold msbits						
9	FXDDI	Fixed detection integration							
10	AVERAGE	Acquisition cycles to be averaged (power of 2)							

Table 5-14. TOUCH_PROXKEY_T52 (Continued)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
11 – 12	MVNULLRATE (1)	Movement nulling rate LSByte							
		Reserved	Movement nulling rate msbits						
13– 14	MVDTHR (1)	Movement detection threshold LSByte							
		Reserved	Movement detection threshold msbits						

1. See "Errata" on page 120.

CTRL Field

The CTRL field enables the Proximity Key for use and determines which messages will be reported. It also allows the Proximity Key to be sent a calibration command.

ENABLE: Enables the use of this Proximity Key T52 object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Allows the object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

CAL: Setting this bit to 1 sends a calibrate command to the Proximity Key. This bit is automatically cleared once the calibration has been done.

TCHACEN: Enables Touch Automatic Recalibration for this Proximity Key. Touch Automatic Recalibration is controlled by the Acquisition Configuration T8 object (see Section 4.5 on page 17).

ATCHACEN: Enables Antitouch Automatic Recalibration for this Proximity Key. When this is enabled, 2 seconds of antitouch will cause a recalibration.

DISMVFALL: Disables the reporting of movement away from this Proximity Key; disabled if set to 1, enabled if set to 0.

DISMVRISE: Disables reporting of movement towards this Proximity Key; disabled if set to 1, enabled if set to 0.

DISFXDTHR: Disables the reporting of fixed detection threshold messages; disabled if set to 1, enabled if set to 0.

XORIGIN and YORIGIN Fields

These fields specify which channels within the Data Source are occupied by the object (see Section 4.6 on page 24 for an explanation of the Data Source). On the mXT540E XORIGIN will always be 32. YORIGIN can be either 0 or 1. Note that a Proximity Key is assumed to have a size of 1 XY channel.

XORIGIN Range: 32 (always)

YORIGIN Range: 0 to 1

AKSCFG Field

This field configures Adjacent Key Suppression (AKS) between this Proximity Key T52 object and any other touch objects.

AKS technology is a patented method used to detect which touch object is touched when objects are located close together. A touch in a group of AKS objects is indicated only on the object with the largest signal. This is assumed to be the intended object. Once an object in an AKS group is in detect, there can be no further detections within that group until the object is released.

Group1 to 8: These bits form a bit field that specifies which AKS groups this Proximity Key is within. The default value of 0 means that the object is in no AKS groups and the AKS feature is disabled for the Proximity Key.

Range: 0 to 255

FXDDTHR Field

The Fixed Detection Touch Threshold value (FXDDTHR) defines how much the proximity delta [1] must change to qualify as a potential proximity detection. The reference level is determined during calibration and adjusted using drift compensation. The final detection confirmation uses the Touch Detect Integration as described in the FXDDI field.

Larger values for the threshold desensitize channels since the signal must change more in order to exceed the threshold level. Conversely, lower thresholds make channels more sensitive.

The setting for FXDDTHR for each channel depends on the amount of signal swing that occurs when a channel is touched. Thicker panels or smaller electrode geometries reduce channel gain, that is signal swing due to touch. In this case smaller FXDDTHR values are required to detect touch.

There is an in-built hysteresis for the threshold specified by FXDDTHR of 12.5 percent.

FXDDI Field

The device incorporates a detection integrator (FXDDI) counter mechanism to provide signal filtering to suppress false detections caused by spurious events, such as electrical noise. A counter is incremented each cycle that a touch is detected. When this counter reaches a preset limit the touch is finally declared to be present. If on any acquisition a delta is not seen to exceed the threshold level, the counter is cleared and the process starts from the beginning.

A similar process is applied when the Proximity Key leaves detection. The counter is decremented each cycle that the delta does not exceed the threshold level, and incremented again if it does exceed the threshold. When the counter reaches zero, the touch is finally declared to be out of detect. In this case there is an extra cycle (that is, the number of cycles is TCHDI + 1).

The range for this field is 0 to 255, where 0, 1 and 2 is the same as 2.

Range: 0 (2), 1 (2), 2 to 255

AVERAGE Field

This field specifies the number of acquisition cycles (as a power of 2) that are to be averaged together during an acquisition burst.

This averaging ensures that the gain for the Proximity Key is not too high and that the noise level is acceptable. The number of cycles to be averaged can be increased if there is excessive noise, but at the cost of a slower response. In this case, the MVNULLRATE field may also need to be lowered.

The valid values for the AVERAGE field are shown in Table 5-15. Note that it will take a certain number of cycles for a new average to be output (as indicated in Table 5-15). If it takes more than one cycle for the update to be output, the previous value is held.

1. Reference minus the signal.

Table 5-15. AVERAGE Field

AVERAGE Field	Number of Acquisition Cycles (1)	Number of Cycles for Update
0	1	1
1	2	1
2	4	1
3	8	1
4	16	1
5	32	2
6	64	4
7	128	8
≥ 8	256	16

1. These are averaged.

For example, if the AVERAGE field is set to 5, then 32 deltas will be averaged together (that is, the current data and the deltas for the 31 previous cycles). In this case it will take two cycles for the updated average to be output, so the value will change every other cycle and then be held for one cycle.

Values above 8 are capped at 8.

Range: 0 (no averaging), 1 to 8 (number of cycles as a power of 2)

MVNULLRATE Field

The Movement Null Rate (MVNULLRATE) decrements the proximity delta change at a certain rate. The delta change must be greater than the null rate for the resulting change to be nonzero. A higher setting will block slower movements from being detected. A lower setting will allow slower movements to be detected. The units are proximity deltas per 200 ms.

Range: 0 to 32767 (in proximity deltas per 200 ms)

MVDTHR Field

The Movement Detection Threshold (MVDTHR) is the delta required (after MVNULLRATE has been applied) to send a movement message. A higher setting will block slower movements from being detected. A lower setting will allow slower movements to be detected.
Range: 0 to 32767

5.4.2 Configuration Checks

A Proximity Key T52 object causes a configuration check to be performed in the following circumstances:

- When the object is enabled (that is, the ENABLE bit is set in the CTRL field)
- When certain fields are changed (as listed in Table 5-16)

In addition, some fields will cause an automatic recalibration to be performed (see Table 5-16).

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14), and the device halts until the error has been corrected. To fix the error, the object settings should be checked to verify that they are all within their allowed limits, as described in the field descriptions.

If a configuration check occurs, a Proximity Key's internal filtering is reset. Note that this happens whenever *any* object causes a configuration check, not just the actual Proximity Key.

Table 5-16. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes (1)	Yes (2)	None
XORIGIN	Yes	No	Error if out of range
YORIGIN	Yes	No	Error if out of range
AKSCFG	No	No	None
FXDDTHR	No	No	None
FXDDI	No	No	None
AVERAGE	Yes	No	None
MVNULLRATE	No	No	None
MVDTHR	No	No	None

1. If the ENABLE bit is toggled on or off.
2. If the ENABLE bit is toggled on. Note that if the ENABLE bit is cleared, no calibration takes place.

5.4.3 Messages

A Proximity Key T52 object reports proximity information in its message data. The message data for a Proximity Key T52 object is shown in Table 5-17.

Table 5-17. Message Data for TOUCH_PROXKEY_T52

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	FXDDETECT	MVRISE	MVFALL	Reserved				
2 – 3	PROXVALUE	Proximity signal delta							

STATUS Field

Reports the current status of the object.

FXDDETECT: Set if the Proximity Key is in a touched state.

MVRISE: Set if a movement towards the Proximity Key is detected, as defined by the MVNULLRATE and MVD0THR fields.

MVFALL: Set if a movement away from the Proximity Key is detected, as defined by the MVNULLRATE and MVDTHR fields.

PROXVALUE Field

This field reports the delta value for the Proximity Key as a signed 16-bit value. Note that averaging may also be applied to calculate this value, as determined by the AVERAGE field.

6. Signal Processing Objects

6.1 Introduction

Signal processing objects process the data from other objects, for example to provide filtering operations. Table 6-1 lists the signal processing objects on the mXT540E.

Table 6-1. Signal Processing Objects

Object	Description
One-touch Gesture Processor T24 (PROCI_ONETOUCHGESTUREPROCESSOR_T24)	Operates on the data from a Touchscreen object. A One-touch Gesture Processor T24 converts touches into one-touch finger gestures (for example, taps, double taps and drags). See Section 6.2.
Two-touch Gesture Processor T27 (PROCI_TWOTOUCHGESTUREPROCESSOR_T27)	Operates on the data from a One-touch Gesture Processor T24 object. A Two-touch Gesture Processor T27 converts touches into two-touch finger gestures (for example, pinches, stretches and rotates). See Section 6.3.
Grip Suppression T40 (PROCI_GRIPSUPPRESSION_T40)	Suppresses false detections caused, for example, by the user gripping the edge of the touchscreen. See Section 6.4.
Touch Suppression T42 (PROCI_TOUCHSUPPRESSION_T42)	Suppresses false detections caused, for example, by the user placing their face too near the touchscreen on a mobile phone. See Section 6.5.
Stylus T47 (PROCI_STYLUS_T47)	Processes stylus input. See Section 6.6.
Noise Suppression T48 (PROCG_NOISESUPPRESSION_T48)	Performs various noise reduction techniques during touchscreen signal acquisition. See Section 6.7.

6.2 One-touch Gesture Processor T24 (PROCI_ONETOUCHGESTUREPROCESSOR_T24)

A One-touch Gesture Processor T24 object configures the on-chip gesture processing for one-touch gestures (such as taps, double taps, presses, flicks and drags). On the mXT540E there are 2 instances of the One-touch Gesture Processor T24 object, one for each of the 2 Multiple Touch Touchscreen T9 objects.

The device supports the one-touch gestures listed in Table 6-2.

Table 6-2. One-touch Gestures

Gesture	Description
Press	A press occurs when the user touches and holds the touch surface. No significant movement takes place while the user's finger is on the touch surface. This could be used to select a number from a displayed numeric keypad. The same mechanism could be used to autorepeat the selected number if the user continues to press on the displayed number.
Release	A release occurs when the user removes their finger from the touch surface.

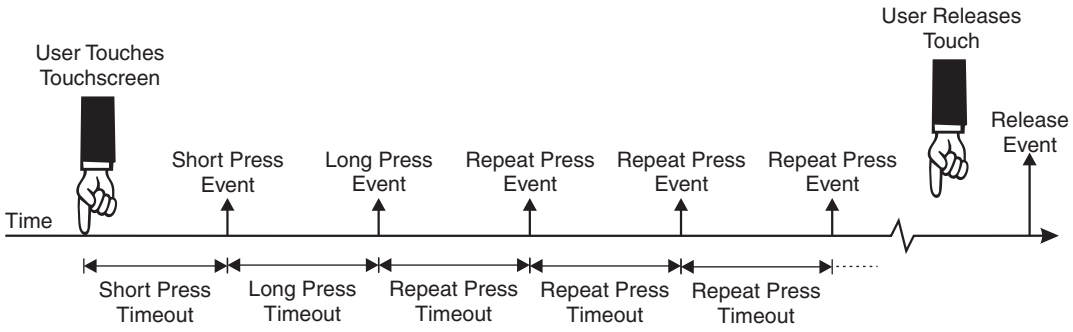
Table 6-2. One-touch Gestures (Continued)

Gesture	Description
Tap	A tap occurs when the user quickly touches and releases the touchscreen. No significant movement takes place while the user's finger is on the touchscreen. It is characterized by a short touch duration. This could be used to activate a hyperlink on a displayed web page.
Double Tap	A double tap occurs when the user quickly touches and releases the touchscreen twice in quick succession. No significant movement takes place while the user's finger is on the touchscreen, or between successive touches. It is characterized by short touch durations, and a short gap between the first release and the second touch. This could be used to select a word in a displayed document.
Flick	A flick occurs when the user quickly touches the touchscreen, moves their finger a short distance across the surface and releases touch. It is characterized by a short touch duration. This could be used to display the next image in a sequence of images.
Throw	A throw occurs when the user touches the touchscreen, moves their finger across the surface, and releases the touch in a quick flick-type motion. Conceptually, it can be considered like a drag motion followed by a flick. It is characterized by a large movement across the touchscreen and a rapid acceleration of the touch just before the release.
Drag	A drag occurs when the user touches the touchscreen, moves their finger across the surface, and releases touch. It is characterized by a large movement across the touchscreen. Depending on the application, multiple drag events may be generated as the user moves their finger. This could be used to select a sentence in a displayed document.
Tap and Press	A tap and press occurs when the user generates both a tap gesture and a press gesture in quick succession.

The host can combine one-touch gestures to create two-touch and multitouch gestures, either within the host operating system or in the driver code. For example, two simultaneous tap gestures on the touchscreen can be combined to form a two-touch tap. The mXT540E also provides on-chip touchscreen gesture processing for three specific two-touch gestures: pinch, rotate and stretch (see the Two-touch Gesture Processor T27 in Section 6.3 on page 64).

Gestures are created from a series of press events. These are generated when the touchscreen is touched, the touch is held, and then the touch is released. Figure 6-1 shows the most general case.

Figure 6-1. Sequence of Press Events



In this case:

1. The user touches the touchscreen. If the touch is held for longer than the short Press Timeout, a short press event is generated.
2. If the user continues to hold the touch, and the Long Press Timeout is exceeded, a long press event is generated.
3. If the user continues to hold the touch, and the Repeat Press Timeout is exceeded, a repeat press event is generated.
4. If the user continues to hold the touch, further repeat press events are generated at regular intervals whenever the Repeat Press Timeout is exceeded.
5. When the user releases the key, a release event is generated.

Note that in Figure 6-1 all three press events are enabled. The One-touch Gesture Processor T24 object, however, allows you to enable or disable the generation of the short, long and repeat press events, as required. All touches always generate release events.

6.2.1 Configuration

Table 6-3. PROCI_ONETOUCHGESTUREPROCESSOR_T24

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved						RPTEN	ENABLE
1	NUMGEST	Maximum number of reported gestures							
2	GESTEN	LPRESS	SPRESS	DRAG	FLICK	DBLTAP	TAP	RELEASE	PRESS
3		Reserved						THROW	RPRESS
4	PROCESS	Reserved		THROWEN	FLICKEN	DBLTAPEN	REPEN	LONGEN	SHORTEN
5	TAPTO	Reserved	Tap Timeout						
6	FLICKTO	Reserved	Flick Timeout						
7	DRAGTO	Reserved	Drag Timeout						
8	SPRESSTO	Reserved	Short Press Timeout						
9	LPRESSTO	Reserved	Long Press Timeout						
10	REPPRESSTO	Reserved	Repeat Press Timeout						
11 – 12	FLICKTHR	Flick Threshold							
13 – 14	DRAGTHR	Drag Threshold							
15 – 16	TAPTHR	Tap Threshold							
17 – 18	THROWTHR	Throw Threshold							

CTRL Field

ENABLE: Enables the use of this One-touch Gesture Processor T24 object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Allows the object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

NUMGEST Field

This field specifies the maximum number of touches that are to be reported. For example, if this field is set to 1, only the first touch is reported as a gesture and all other touches do not generate messages.

This field does not remove report IDs. Report IDs are fixed to the maximum number of touch gestures.

Range: 1 to 4

GESTEN Field

Enables one-touch gestures and reporting.

PRESS: Press event reporting is enabled if set to 1, and disabled if set to 0.

RELEASE: Release event reporting is enabled if set to 1, and disabled if set to 0.

TAP: Tap event reporting is enabled if set to 1, and disabled if set to 0.

DBLTAP: Double Tap event reporting is enabled if set to 1, and disabled if set to 0.

FLICK: Flick event reporting is enabled if set to 1, and disabled if set to 0.

DRAG: Drag event reporting is enabled if set to 1, and disabled if set to 0.

SPRESS: Short Press event reporting is enabled if set to 1, and disabled if set to 0.

LPRESS: Long Press event reporting is enabled if set to 1, and disabled if set to 0.

RPRESS: Repeat Press event reporting is enabled if set to 1, and disabled if set to 0.

THROW: Throw event reporting is enabled if set to 1, and disabled if set to 0.

PROCESS Field

This field enables and disables the press events and some of the one-touch gestures that can be processed by the device.

SHORTEN: Enables short presses to be processed by the device. Short presses are enabled if set to 1, and disabled if set to 0.

LONGEN: Enables long presses to be processed by the device. Long presses are enabled if set to 1, and disabled if set to 0.

REPEN: Enables repeat presses to be processed by the device. Repeat presses are enabled if set to 1, and disabled if set to 0.

DBLTAPEN: Enables double taps to be processed by the device. Double taps are enabled if set to 1, and disabled if set to 0.

FLICKEN: Enables flicks to be processed by the device. Flicks presses are enabled if set to 1, and disabled if set to 0.

THROWEN: Enables throws to be processed by the device. Throws are enabled if set to 1, and disabled if set to 0.

TAPTO Field

This field specifies the timeout for a tap gesture. This is the maximum duration of a tap. A tap gesture is when the user touches and releases a touch, without moving their finger. If this sequence takes less time than the tap timeout, a tap event is generated. The timeout is specified in units of 4 ms. A value of zero means 75 (300 ms). The timeout should not be set shorter than the active cycle period.

This value also controls the generation of double tap events. A double tap event consists of a press, release, press, and final release. If the time between each of these actions is less than the tap timeout, a double tap event is generated.

The user must move their finger less than the Tap Threshold (TAPTHR) during a tap to generate a tap or double tap gesture. Also, the distance between consecutive taps must be less than the Tap Threshold to generate a double tap gesture.

If the timeout expires and the touch is still present, the press processing commences.

Range: 0 (75 = 300 ms), 1 to 127 (in units of 4 ms)

FLICKTO Field

This field specifies the timeout for a flick gesture. This is the maximum duration of a flick gesture. A flick gesture is when the user presses the touchscreen, moves their finger, and then releases the touchscreen. A flick event is generated if a touch moves more than the flick threshold (defined by the FLICKTHR field), in less time than the tap timeout (TAPTO), and is then released within the FLICKTO. The flick timeout should not be set shorter than the active cycle period.

The flick timeout is specified in units of 4 ms. A value of zero means 75 (300 ms).

See also “FLICKTHR and TAPTHR Fields” on page 61.

Range: 0 (75 = 300 ms), 1 to 127 (in units of 4 ms)

DRAGTO Field

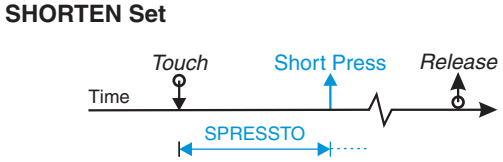
This field specifies the timeout for detecting a new drag event. This is the maximum time between two drag events. A drag gesture is when the user presses the touchscreen and moves their finger. This generates a drag event. If the user moves their finger again within the drag timeout period, another drag event is generated. If the user’s finger remains stationary for longer than the drag timeout period, drag processing stops. In this case, any enabled press events start to be generated. The drag timeout is specified in units of 4 ms. A value of zero means 75 (300 ms). The timeout should not be set shorter than the active cycle period. See also “DRAGTHR Field” on page 62.

Range: 0 (75 = 300 ms), 1 to 127 (in units of 4 ms)

SPRESSTO Field

This field specifies the Short Press Timeout. This is the time between the start of press processing and the generation of a short press event (see Figure 6-2). The Short Press Timeout is used only if short press processing is enabled using the “SHORTEN” bit (see “PROCESS Field” on page 58).

Figure 6-2. Short Press Timeout



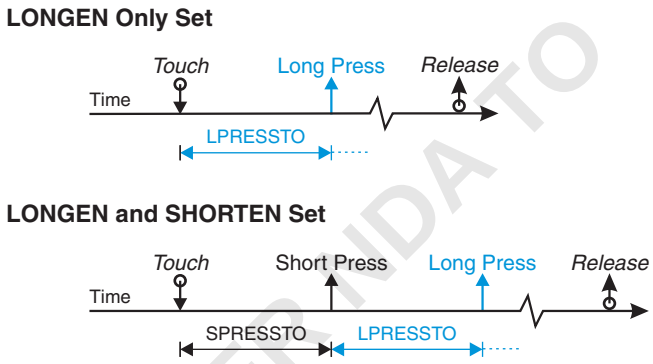
The short timeout is specified in units of 4 ms. A value of zero means 75 (300 ms). The timeout should not be set shorter than the active cycle period.

Range: 0 (75 = 300 ms), 1 to 127 (in units of 4 ms)

LPRESSTO Field

This field specifies the Long Press Timeout. This is the time taken for the generation of a long press event. The start point for the timeout depends on whether or not short events are also enabled. If short presses are enabled, the timeout is from the short press event; otherwise it is from the start of press processing (see Figure 6-3). The Long Press Timeout is used only if long press processing is enabled using the "LONGEN" bit (see "PROCESS Field" on page 58).

Figure 6-3. Long Press Timeout



The long timeout is specified in units of 4 ms. A value of zero means 75 (300 ms). The timeout should not be set shorter than the active cycle period.

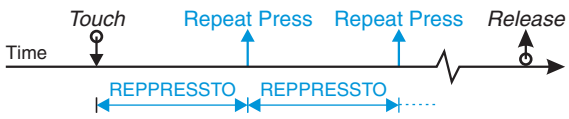
Range: 0 (75 = 300 ms), 1 to 127 (in units of 4 ms)

REPPRESSTO Field

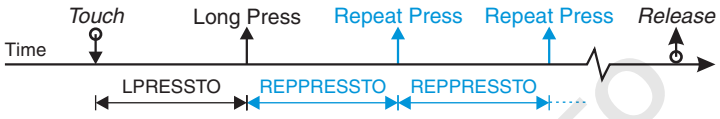
This field specifies the Repeat Press Timeout. This is the time taken for the generation of a repeat press event. The start point for the timeout depends on whether short press and long press events are also enabled (see Figure 6-4). The repeat timeout also specifies the time between the generation of consecutive repeat press events. The Repeat Press Timeout is used only if repeat press processing is enabled using the "REPEN" bit (see "PROCESS Field" on page 58).

Figure 6-4. Long Press Timeout

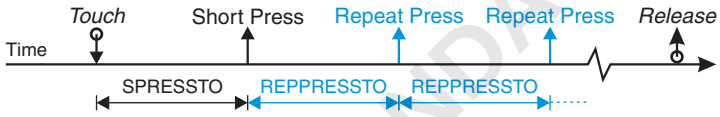
REPEN Only Set



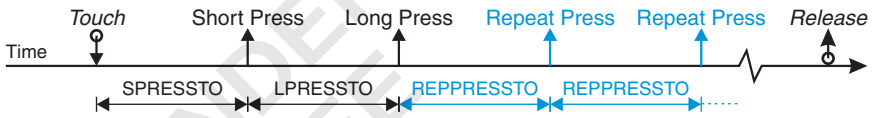
REPEN and LONGEN Set



REPEN and SHORTEN Set



REPEN, LONGEN and SHORTEN Set



The repeat timeout is specified in units of 4 ms. A value of zero means 75 (300 ms). The timeout should not be set shorter than the active cycle period.

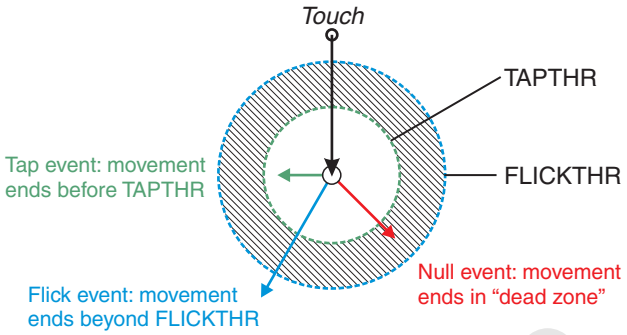
Range: 0 (75 = 300 ms), 1 to 127 (in units of 4 ms)

FLICKTHR and TAPTHR Fields

These fields dictate how a quick touch-and-release action is to be interpreted. This action can be interpreted as either a tap or a flick gesture. These two gestures are essentially the same. The only difference is the amount of movement that occurs between the touch and the release. With a flick, the user's finger moves a significant distance before the touch is release. With a tap there is minimal or no movement.

If the movement is less than or equal to the Tap Threshold (TAPTHR), the gesture is interpreted as a tap. If the movement is greater than or equal to the Flick Threshold (FLICKTHR), the gesture is interpreted as a flick. If the movement is in the "dead zone" between the two thresholds, no gesture is generated for the touch. This is summarized in Figure 6-5.

Figure 6-5. Flick and Tap Thresholds



The thresholds are specified in units of touchscreen distance.

It may be useful to set both the TAPTHR and FLICKTHR values to the same values initially. This means there is no "dead zone" and so either a tap or flick gesture is always reported. If the design needs to be fine tuned, the two controls can later be set to different values to create a "dead zone". Note that FLICKTHR must be greater than or equal to TAPTHR.

See also "TAPTO Field" on page 58 and "FLICKTO Field" on page 59.

DRAGTHR Field

This value specifies how far the user must move their finger across the touchscreen to generate a drag event. See also "DRAGTO Field" on page 59. The drag threshold is specified in units of touchscreen distance.

THROWTHR Field

This field specifies the speed on the drag or release from the drag that is needed to generate a throw gesture. The speed of the user's finger must be greater than or equal to THROWTHR for a throw gesture to be generated. The speed is specified as:

2 x touchscreen_distance_per_ms

A minimum time of 4 ms for the release is assumed. If the release takes less than 4 ms, the One-touch Gesture Processor T24 object uses a time of 4 ms in its calculations.

6.2.2 Messages

The message data for a One-touch Gesture Processor T24 object is shown in Table 6-4.

Table 6-4. Message Data for PROCI_ONETOUCHGESTUREPROCESSOR_T24

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Reserved				Event			
2	XPOSMSB	X position MSByte							
3	YPOSMSB	Y position MSByte							
4	XYPOSLSB	X position lsbits				Yposition lsbits			
5	DIR	Gesture direction							
6 – 7	DIST	Gesture distance							

STATUS Field

This field reports which single-touch event has occurred. Table 6-5 gives the possible values and their meanings.

Table 6-5. Events

Event	Description	Event	Description	Event	Description
0000	Reserved	0101	Flick	1010	Tap and Press
0001	Press	0110	Drag	1011	Throw
0010	Release	0111	Short Press	1100 to 1111	Reserved
0011	Tap	1000	Long Press		
0100	Double-tap	1001	Repeat Press		

XPOSMSB, YPOSMSB and XYPOSLSB Fields

These three fields report the X and Y position of the gesture. XPOSMSB/YPOSMSB contains the most significant byte of the position. XYPOSLSB contains the least significant bits of the position.

The position has one of two formats. This depends on whether it is reporting at 10-bit or 12-bit resolution. The resolution is determined by the XRANGE or YRANGE field of the Multiple Touch Touchscreen T9 object (see Section 5.2 on page 26) linked to this One-touch Gesture Processor T24.

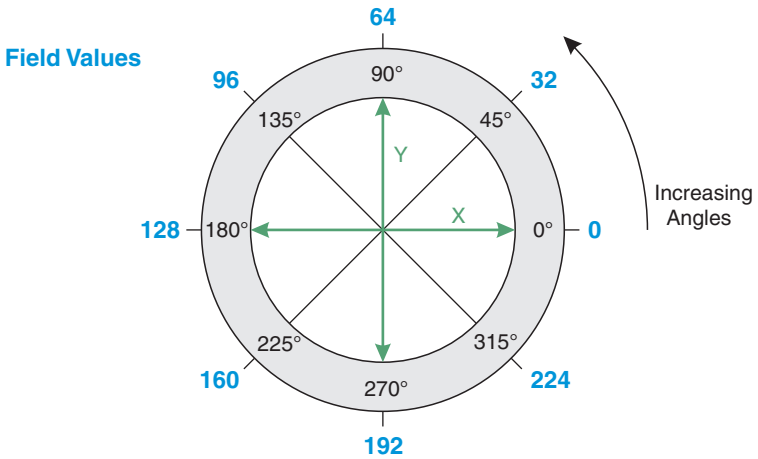
object. The X and Y position formats are independent. For example, a message could contain a 10-bit X position and a 12-bit Y position.

The Formats for the X and Y positions are shown in Table 5-7 and Table 5-8 on page 43.

DIR Field

The DIR field contains the direction of the flick or throw when a flick or throw event is generated. The direction is specified in 1/256 of 360°, as in Figure 6-6.

Figure 6-6. Direction and Angle Values



Note: The X and Y positions are before any processing of the Multiple Touch Touchscreen T9's ORIENT field.

DIST Field

The DIST field contains the distance of the flick or throw when a flick or throw event is generated.

6.3 Two-touch Gesture Processor T27 (PROCI_TWOTOUCHGESTUREPROCESSOR_T27)

A Two-touch Gesture Processor T27 object configures the on-chip touchscreen gesture processing for the two-touch gestures: pinch, rotate and stretch (see Table 6-6). Other two-touch gestures can be generated by the host from one-touch gestures (see Section 6.2 on page 55). On the mXT540E there are 2 instances of the Two-touch Gesture Processor T27 object, one for each of the 2 Multiple Touch Touchscreen T9 objects.

Table 6-6. Two-touch Gestures

Gesture	Description
Pinch	A pinch occurs when the user touches and holds the touch surface with two fingers and brings them together. It is not necessary for both fingers to move. One finger can remain stationary while the other moves towards it. It is the relative (decreasing) distance between the fingers that determines the pinch gesture.
Rotate	A rotate occurs when the user touches and holds the touch surface with two fingers and then moves them in a circle around each other. It is not necessary for both fingers to move. One finger can remain stationary while the other moves around it. It is the relative angle between the fingers that determines the rotate gesture.
Stretch	A stretch is the inverse of a pinch. The user touches and holds the touch surface with two fingers and moves them apart. It is not necessary for both fingers to move. One finger can remain stationary while the other moves away from it. It is the relative (increasing) distance between the fingers that determines the pinch gesture.

Note that the two-touch gestures in Table 6-7 can be combined. For example, combining a rotate gesture with a stretch or pinch gesture will form an increasing or decreasing spiral gesture.

Note: This object operates only if the linked One-touch Gesture Processor T24 object has also been set up correctly and enabled for use (see Section 6.2 on page 55).

6.3.1 Configuration

Table 6-7. PROCI_TWOTOUCHGESTUREPROCESSOR_T27

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved						RPTEN	ENABLE
1	NUMGEST	Maximum number of reported gestures							
2	Reserved	Reserved							
3	GESTEN	STRETCH	ROTATE	PINCH	Reserved				
4	ROTATETHR	Reserved	Rotate Threshold						
5 – 6	ZOOMTHR	Zoom Threshold							

CTRL Field

ENABLE: Enables the use of this Two-touch Gesture Processor T27 object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Allows this object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

NUMGEST Field

This field specifies the maximum number of two-touch gestures to be reported, where each gesture is made up of a pair of touches [1]. For example, if this field is set to 1, the first and second touch are paired as a gesture and are reported; all other touches are ignored by the processor.

Note that this field does not remove report IDs. Report IDs are fixed to the maximum number of touch gestures.

Range: 1

GESTEN Field

This field enables two-touch gestures and reporting.

PINCH: Pinch event reporting is enabled if set to 1, and disabled if set to 0.

ROTATE: Rotate event reporting is enabled if set to 1, and disabled if set to 0.

STRETCH: Stretch event reporting is enabled if set to 1, and disabled if set to 0.

ROTATETHR Field

This field specifies the minimum change in angle between two points for generating two-touch rotate gestures. The rotate threshold is specified in 1/256 of 360° (see Figure 6-6 on page 63) with a maximum allowed angle of less than 180°. The reported angle is affected by the order in which the touches are applied.

Range: 0 to 127 (180°)

ZOOMTHR Field

This field specifies the minimum change in distance (in units of touchscreen distance) required between two simultaneous touches to generate a stretch or pinch gesture.

6.3.2 Messages

The message data for a Two-touch Gesture Processor T27 object is shown in Table 6-8.

Table 6-8. Message Data for PROCI_TWOTOUCHGESTUREPROCESSOR_T27

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	STRETCH	ROTATE	PINCH	ROTATEDIR	Reserved			
2	XPOSMSB	Center of gesture X position MSByte							
3	YPOSMSB	Center of gesture Y position MSByte							
4	XYPOSLSB	X position lsbits				Yposition lsbits			
5	ANGLE	Gesture angle							
6 – 7	SEP	Gesture separation LSByte							
		Gesture separation MSByte							

1. Note that the number of gestures specified represents half the number of touches because the touches are paired.

STATUS Field

This field reports which two-touch events have occurred. A two-touch gesture may consist of more than one two-touch gestures. For example, a rotate may be combined with a stretch to create a spiralling zoom effect.

ROTATEDIR: Indicates the direction of the rotate event: 0 = rotation in the direction of increasing angles, 1 = rotation in the direction of decreasing angles (see Figure 6-6).

PINCH: If set to 1, a pinch event has occurred.

ROTATE: If set to 1, a rotate event has occurred.

STRETCH: If set to 1, a stretch event has occurred.

XPOSMSB, YPOSMSB and XYPOSLSB Fields

These three fields report the X and Y position of the center of the gesture. XPOSMSB/YPOSMSB contains the most significant byte of the position. XYPOSLSB contains the least significant bits of the position.

The position has one of two formats. This depends on whether it is reporting at 10-bit or 12-bit resolution. The resolution is determined by the XRANGE or YRANGE field of the Multiple Touch Touchscreen T9 object linked to this Two-touch Gesture Processor T27 object (see Section 5.2 on page 26). The X and Y position formats are independent. For example, a message could contain both a 10-bit X position and a 12-bit Y position.

The formats for the X and Y positions are shown in Table 5-7 on page 42 and Table 5-8.

ANGLE Field

The ANGLE field reports the angle between two touches in a two-touch gesture. The angle between the two touches, and its direction, is determined by the relative X and Y positions of the second touch from the first touch. The angle is reported in 1/256 of 360° (see Figure 6-6).

SEP Field

The SEP field reports the separation (distance) between the two touches in a two-touch gesture as a 16-bit number. This value could be used by the host, for example, as an indication of the speed of a rotate or stretch gesture.

6.4 Grip Suppression T40 (PROCI_GRIPSUPPRESSION_T40)

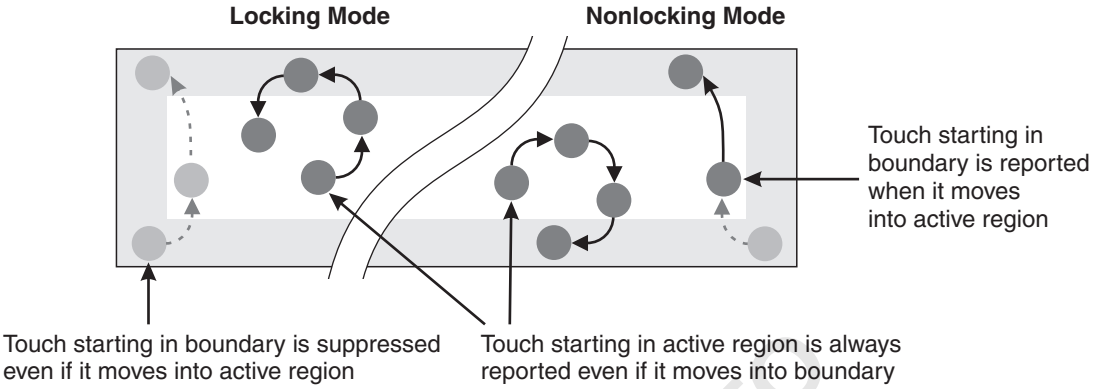
A Grip Suppression T40 object suppresses false detections that are typically caused by the user gripping the touchscreen on a handheld device. It processes the measurement data received from a particular instance of a Multiple Touch Touchscreen T9 object. On the mXT540E there are 2 instances of the Grip Suppression T40 object, one for each of the 2 Multiple Touch Touchscreen T9 objects.

The grip suppression mechanism defines a grip suppression boundary around the edge of the Multiple Touch Touchscreen T9. All touches that originate inside the active region are reported. Touches that start in the boundary are suppressed or reported depending on the grip mode (as defined by the GRIPMODE bit in the CTRL field):

- In locking mode, a touch that starts inside the grip suppression boundary is suppressed. It remains suppressed even if it subsequently moves into the active region.
- In nonlocking mode, a touch that starts inside the grip suppression boundary is suppressed. It remains suppressed while it remains in the boundary. If it subsequently moves into the active region, it is reported even if it later moves back into the grip suppression boundary.

  The effect of grip suppression on detected touches is summarized in Figure 6-7.

Figure 6-7. Effect of Grip Suppression on Detected Touches



In nonlocking mode, the UNGRIP bit is set in the messages for a touch if the touch is detected after having previously been suppressed (see Section 5.2.3 on page 41). The UNGRIP bit remains set while the touch is on the screen and is also set in its release message.

6.4.1 Configuration

Table 6-9. PROCI_GRIPSUPPRESSION_T40

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved			GRIPMODE	Reserved			ENABLE
1	XLOGRIP	Grip suppression X low boundary							
2	XHIGRIP	Grip suppression X high boundary							
3	YLOGRIP	Grip suppression Y low boundary							
4	YHIGRIP	Grip suppression Y high boundary							

CTRL Field

This field enables or disables grip suppression.

ENABLE: Enables the use of this Grip Suppression T40 object. The object is enabled if set to 1, and disabled if set to 0.

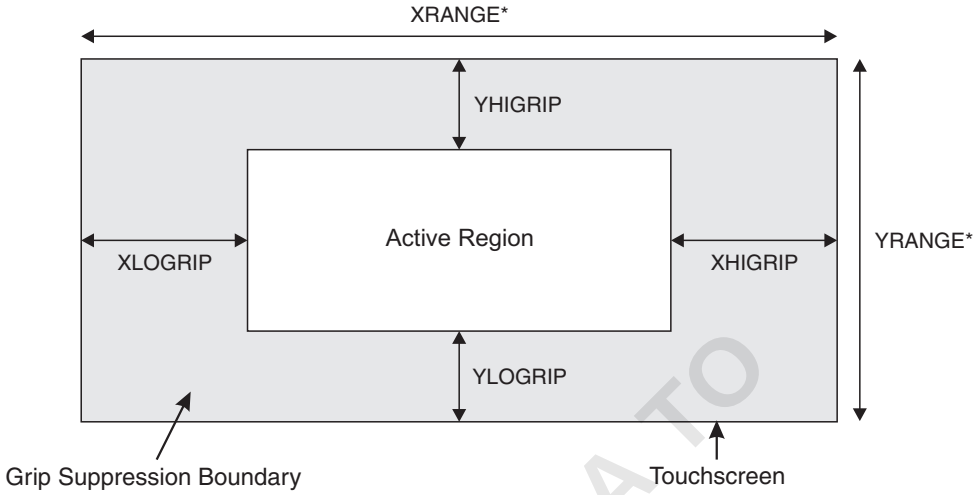
GRIPMODE: Sets the grip mode. Set this bit to 1 for nonlocking mode, and 0 for locking mode.

XLOGRIP, XHIGRIP, YLOGRIP and YHIGRIP Fields

These fields specify a grip suppression boundary around the edge of the Multiple Touch Touchscreen T9 that this object processes. Inside this boundary is the active region of the Touchscreen.

Figure 6-8 shows how the controls define the grip suppression boundary.

Figure 6-8. Grip Suppression Boundary Fields



* XRANGE and YRANGE fields in a Multiple Touch Touchscreen object

Note: The boundary settings are applied after the INVERTX and INVERTY bits of the ORIENT field of the Multiple Touch Touchscreen T9 object take effect, but before the SWITCH bit in the ORIENT field (see Section 5.2 on page 26). Thus the upper and lower boundary settings may be interchanged, but the physical axes remain unaffected.

XLOGRIP/XHIGRIP Range: 0 to 255

YLOGRIP/YHIGRIP Range: 0 to 255

XLOGRIP/XHIGRIP/YLOGRIP/YHIGRIP Default: 0

6.5 Touch Suppression T42 (PROCI_TOUCHSUPPRESSION_T42)

A Touch Suppression T42 object processes the measurement data received from a particular instance of a Multiple Touch Touchscreen T9 object. It suppresses false detections from unintentional touches. On the mXT540E there are 2 instances of the Touch Suppression T42 object, one for each of the 2 Multiple Touch Touchscreen T9 objects.

The Touch Suppression T42 object provides the following types of touch suppression:

- Large object touch suppression – detects unintentional touches from a large body area, such as from a face, ear or palm. If an unintentional touch is detected, the Touch Suppression T42 suppresses the entire Multiple Touch Touchscreen. The suppression is released only when all the touches are released.
- Maximum touch suppression – suppresses all touches if more than a specified number of touches has been detected. Any touches that are already detected will also be suppressed. The touches remain suppressed until all the touches are released or a recalibration occurs.

Note that large object and maximum touch suppression both lock the Touchscreen and with it any Adjacent Key Suppression locks that are currently applied.

6.5.1 Configuration

Table 6-10. PROCI_TOUCHSUPPRESSION_T42

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved				DISLOBJ	SHAPEEN	RPTEN	ENABLE
1	APPRTHR	Approach threshold							
2	MAXAPPRAREA	Maximum approach area threshold							
3	MAXTCHAREA	Maximum touch area threshold							
4	SUPSTRENGTH	Suppression aggressiveness							
5	SUPEXTTO	Suppression extension timeout							
6	MAXNUMTCHS	Maximum touches							
7	SHAPESTRENGTH	Shaped-based aggressiveness							

CTRL Field

This field enables or disables large touch suppression.

ENABLE: Enables the use of this Touch Suppression T42 object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Allows the object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

SHAPEEN: Enables or disables ear suppression. This is a second level of rejection logic that is based upon a touch's shape. Ear suppression is enabled if set to 1, and disabled if set to 0.

DISLOBJ: Disables the large object touch suppression. This allows the maximum touch suppression to be used exclusively. Large object touch suppression is disabled if set to 1, and enabled if set to 0 (the default).

APPRTHR Field

This field specifies the Approach Threshold. Any touch above this setting is assessed to see whether it is an unintentional large-area touch.

The Approach Threshold is specified in delta units, where a value of 0 means use a quarter of the touch threshold specified by TCHTHR field in the Multiple Touch Touchscreen T9 object (see Section 5.2 on page 26). The recommended minimum value for this field is 20.

Range: 0 (TCHTHR / 4), 1 to 255 (threshold in delta units)

Recommended Range: 20 to 255

MAXAPPRAREA Field

This field specifes the Maximum Approach Area Threshold. This is specifed as the number of channels above the Approach Threshold (APPRTHR). If the number of channels exceeds this threshold, the touch is automatically treated as a large unintentional touch and the touch is suppressed without any further checks or suppression calculations.

The maximum Maximum Approach Area Threshold is specified as the number of channels, where a value of 0 means 40 channels.

Range: 0 (40 channels), 1 to 255 (number of channels)

MAXTCHAREA Field

This field specifes the Maximum Touch Area Threshold. This is specifed as the number of channels above the Touch Threshold (TCHTHR) specified in the Multiple Touch Touchscreen T9 object (see Section 5.2 on page 26). If the number of channels in a touch that are above the TCHTHR exceeds this threshold, the touch is automatically treated as a large unintentional touch. In this case the touch is suppressed without any further checks or suppression calculations.

The touch area is specified as the number of channels, where a value of 0 means 35 channels.

Range: 0 (35 channels), 1 to 255 (number of channels)

SUPSTRENGTH Field

This field specifies the Suppression Strength. This controls the agressiveness of the Touch Suppression T42 object (see Table 6-11). A higher value reduces the aggressiveness and allows larger touches to be reported, and a lower value increases the aggressiveness and allows only small touches to be reported. A value of 0 means 128; that is, normal aggressiveness.

Table 6-11. Suppression Strength

Value	Meaning
0	Treated as a value 128: normal aggressiveness (default)
1 to 127	Increases aggressiveness (allows only small touches to be reported)
128	Normal aggressiveness
129 to 255	Reduces aggressiveness (allows larger touches to be reported)

Range: 0 (128), 1 to 255

SUPEXTTO Field

Suppression extension timeout extends the time needed to make a decision as to whether a touch is to be suppressed or not. The timeout applies only if the Touch Suppression T42 object cannot determine whether a touch is an actual touch or a rogue touch and requires more evidence to reach the proper decision.

Specifically, this field controls the how long suppression extension is to be performed. The process will keep extending the DI if it cannot determine whether a touch is an actual touch or a rogue touch until this timer expires. When this timer expires, the touch is suppressed.

The SUPEXTTO field is specified as the number of cycles, where a value of 0 disables the timeout (that is the determination process never expires).

Range: 0 (never expires), 1 to 255 (timeout in cycles)

MAXNUMTCHS Field

This field specifies the maximum number of touches that will be reported before touches are suppressed by the maximum touch suppression. If more than MAXNUMTCHS touches are detected, the touchscreen suppresses all touches. This includes those touches already detected. The touches remain suppressed until all touches are released or a recalibration occurs.

A value of 0 ensures that all touches will be reported.

Range: 0 to 15 (maximum number of touches minus 1)

SHAPESTRENGTH Field

This field specifies the shape-based suppression strength. It is valid only if the second-level ear suppression is enabled (that is, the SHAPEEN bit of the CTRL field is set) and controls the shape-based classifier.

The range for SHAPESTRENGTH is 0 to 31, where the default value of 0 is treated as 10.

Values less than 10 result in more aggressive touch suppression; values greater than 10 result in less aggressive touch suppression.

Increasing the shape strength value above the default value of 10 allows more elongated or curved touches to be reported. Decreasing the shape strength value below the default results in the suppression of more touches so that only particularly square-shaped or circle-shaped touches are reported.

For most practical use cases, a SHAPESTRENGTH value between 5 and 20 is adequate.

Range: 0 (10), 1 to 31

6.5.2 Configuration Checks

A Touch Suppression T42 object causes a configuration check to be performed in the following circumstances:

- If the object is enabled, when certain fields are changed (as listed in Table 6-12)

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14). The device halts until the error has been corrected. To fix the error, check that all the object settings are within their allowed limits, as stated in the field descriptions.

Table 6-12. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes	No	None
APPRTHR	Yes	No	None
MAXAPPRAREA	Yes	No	None
MAXTCHAREA	Yes	No	None
SUPSTRENGTH	Yes	No	None
SUPEXTTO	No	No	None
MAXNUMTCHS	No	No	None
SHAPESTRENGTH	Yes	No	None

6.5.3 Messages

The message data for a Touch Suppression T42 object is shown in Table 6-13.

Table 6-13. Message Data for PROCI_TOUCHSUPPRESSION_T42

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Reserved							TCHSUP

TCHSUP: Indicates that touch suppression has been applied to the linked Multiple Touch Touchscreen T9 object. This field is set to 1 if suppression is active, and 0 otherwise.

6.6 Stylus T47 (PROCI_STYLUS_T47)

A Stylus T47 object processes the measurement data received from a particular instance of a Multiple Touch Touchscreen T9 object. It allows the detection of touches that would otherwise by considered too small for the touchscreen. Additionally, there are controls to distinguish a stylus touch from an unwanted approaching finger. On the mXT540E there are 2 instances of the Stylus T47 object, one for each of the 2 Multiple Touch Touchscreen T9 objects.

Stylus touches are reported by the linked Multiple Touch Touchscreen T9 object with a touch area of 0 (see "TCHAREA Field" on page 43). Note that if a stylus touchdown occurs at the edge of the touchscreen, it will be tracked but not reported until it moves away from the edge.

Note that the stylus logic works best if the Touch Threshold (TCHTHR) field in the Multiple Touch Touchscreen T9 is set to a high value (see Section 5.2 on page 26).

For more information on the Stylus T47 object, refer to QTAN0078, *maXTouch Stylus Tuning*.

6.6.1 Configuration

Table 6-14. PROCI_STYLUS_T47

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved							ENABLE
1	CONTMIN	Minimum contact diameter							
2	CONTMAX	Maximum contact diameter							
3	STABILITY	Stability							
4	MAXTCHAREA	Maximum touch area							
5	AMPLTHR	Maximum touch amplitude							
6	STYSHAPE	Stylus shape adjustment							
7	HOVERSUP	Hovering finger suppression							
8	CONFTHR	Confidence threshold							
9	SYNCSPERX	ADC sets per X							

CTRL Field

This field enables or disables the Stylus T47 object.

ENABLE: Enables the use of this Stylus T47 object. The object is enabled if set to 1, and disabled if set to 0.

CONTMIN Field [1]

This field specifies the minimum contact diameter of the stylus that will be used. Note that this is the contact diameter of the tip and not the diameter of the body of the stylus.

This setting should be adjusted to the highest possible value that allows the desired minimum stylus size to be detected when the stylus is touched to the screen. This minimum value depends on the system signal-to-noise ratio (SNR). A low value means that small styluses can be detected, but it may also mean that noise spikes are detected if CONTMIN is too low. Higher values means less noise detections, but only larger styluses will be detected. It is recommended that this field is never set below 20 (2 mm) in regular usage.

The minimum contact diameter is specified in units of 0.1 mm. The default value of 0 means 30 (3 mm).

Range: 0 (30 = 3 mm), 1 to 255 (in 0.1 mm increments)

CONTMAX Field [1]

This field specifies the maximum contact diameter of the stylus that will be used. Note that this is the contact diameter of the tip and not the diameter of the body of the stylus. Objects larger than this size will not be tracked as stylus touches and instead will be treated as finger touches.

This setting should be adjusted to the lowest possible value that allows the desired maximum stylus size to be detected when the stylus is touched to the screen. Note, however, that if CONTMAX is too large, finger touches will not be detected; a touch is reported as a finger only if its diameter is larger than CONTMAX.

The maximum contact diameter is specified in units of 0.1 mm, where a value of 0 means 72 (7.2 mm).

Range: 0 (72 = 7.2 mm), 1 to 255 (in 0.1 mm increments)

STABILITY Field

This field is used to stabilize the stylus detection by applying hysteresis to CONTMIN. Increasing the stability prevents small styluses from dropping out of detection. This causes the algorithm to continue reporting a stylus touch even if it temporarily becomes too small (that is, below CONTMIN). In normal circumstances this setting should not need adjusting. The default value of 0 means that the setting is disabled.

Range: 0 (disabled), 1 to 255

MAXTCHAREA Field

This field specifies the Maximum Touch Area in the linked Multiple Touch Touchscreen T9 object that a touch can have and still be considered a stylus touch. The touch area is specified in channels. The default value of 0 means that only touches that are not detected by the Multiple Touch Touchscreen T9 object are considered as stylus touches.

Range: 0 to 255

1. CONTMIN and CONTMAX are independent of each other. It is possible to set CONTMIN greater than CONTMAX under certain (very rare) circumstances and still have stylus touches reported.

AMPLTHR Field

This field specifies the maximum Touch Amplitude that can still be considered a stylus touch. Any touch with an amplitude larger than this will be reported as a finger. The amplitude is reported by the TCHAMPLITUDE message field in the Multiple Touch Touchscreen T9 object (see Section 5.2.3 on page 41). The default value of 0 means a Touch Amplitude of 40.

Range: 0 (40), 1 to 255

STYSHAPE Field

This field controls how the hovering finger suppression treats the stylus. This is used to adjust for the specific parameters of the touchscreen, such as signal strength and pitch. Increasing STYSHAPE makes touches look “bigger” or “flatter” to the hovering finger suppression algorithm; decreasing STYSHAPE makes the algorithm see touches as “smaller” or “more pointed”. A value of 0 means 16.

Range: 0 (16), 1 to 255

HOVERSUP Field

This field controls the aggresiveness of the hovering finger suppression logic. Higher settings will make this logic less aggressive. In this case a hovering finger is more likely to be tracked as a stylus touch. A value of 0 means 150 and a value of 255 disables this feature.

Range: 0 (150), 1 to 254, 255 (disable)

CONFTHR Field

This field controls the Confidence Threshold. It is specified in units of measurement cycles. Higher settings will reduce the speed of the touch-down detection of a stylus touch but will make a stylus touch more likely to be identified correctly. The default value of 0 means there is no delay in detecting stylus touches.

Range: 0 (no delay), 1 to 255

SYNCSPERX Field

This field specifies the minimum Sync Groups per X line to be used when stylus touches are being detected; that is, it can override the settings in the Noise Suppression T48 and CTE Configuration T46 objects as needed (see Section 6.7 on page 75 and Section 7.8 on page 101). If SYNCSPERX is set to 0, this field has no effect.

See Section 7.8 on page 101 for more details on Sync Groups.

Range: 0 (none), 1 to 63

6.6.2 Configuration Checks

A Stylus T47 object causes a configuration check to be performed in the following circumstances:

- If the object is enabled, when certain fields are changed (as listed in Table 6-15)

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14). The device halts until the error has been corrected. To fix the error, check that all the object settings are within their allowed limits, as stated in the field descriptions.

Table 6-15. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes	No	None
CONTMIN	Yes	No	None
CONTMAX	Yes	No	None
STABILITY	Yes	No	None
MAXTCHAREA	No	No	None
AMPLTHR	No	No	None
STYSHAPE	No	No	None
HOVERSUP	No	No	None
CONFTHR	No	No	None
SYNCSPERX	Yes	No	None

6.7 Noise Suppression T48 (PROCG_NOISESUPPRESSION_T48)

The Noise Suppression T48 object provides an algorithm to suppress the effects of noise (for example, from a noisy charger plugged into the user's product). This algorithm can automatically adjust some of the acquisition parameters and filter the acquisition data received from the sensor. The noise suppression algorithm can make use of the following filters (see Figure 6-9):

- **The Grass Cutter**

  The Grass Cutter operates on individual ADCs (analog-to-digital conversion samples; see Section 7.8 on page 101 for an explanation of ADCs). It rejects any ADCs outside the grass-cutting limit. This limit is adjusted automatically in response to noise levels, but within the extents determined by the GCLIMITMIN and GCLIMITMAX fields. Once the ADCs have been grass-cut, the remaining valid ADCs are averaged. The Grass Cutter can operate in one of five modes (see Table 6-18), as determined by the GCMODE bits in the CFG field.

  When the Grass Cutter is active, the noise suppression algorithm can change the burst frequency if the current burst frequency becomes too noisy. An initial frequency search determines the best frequency to start using for acquisitions and the Grass Cutter then operates at that frequency. Background frequency scans are continually performed during idle acquisition (no touches) if the Grass Cutter is in operation. The burst frequency is adjusted as required. The SELFREQMAX field can be used to limit the maximum range of the initial frequency search and the background frequency scans. Note that if the selected frequency is changed, a message is generated.

- **The Median Filter**

  The Median Filter operates on three sets of accumulated ADCs. The highest and lowest sets of accumulated ADCs are rejected, leaving the center set. The Median Filter acquires the three sets of ADCs at the frequencies specified by the BASEFREQ and the two MFFREQ[] fields. The BASEFREQ field specifies the frequency to use to acquire the first set of ADCs by the Median Filter, and the MFFREQ[] fields specify the second and third frequencies to use.

When the Median Filter is running, the noise suppression algorithm can automatically adjust the number of ADCs per X within the following minimum and maximum bounds:

Minimum ADCs per X: IDLESYNCSPERX, ACTVSYNCSPERX or SYNCSPERX, whichever is currently in effect

Maximum ADCs per X: GCMAXADCSPERX or 63, whichever is smaller

The Median Filter cannot and does not run at the same time as the Grass Cutter. The MFEN bit in the CALCFG field forces the noise suppression algorithm to bypass the Grass Cutter and make direct use of the Median Filter (see Figure 6-9).

Noise suppression is triggered when a noise source is detected, such as when a charger is turned on. There are two types of triggers to which the Noise Suppression T48 object can respond:

- **Hardware trigger** – The CHRG_IN pin can be used as an input to detect the presence of a charger or other potential noise source. Charger Input detection is enabled by the CHRGIN bit in the CFG field.
- **Software trigger** – The host can set the CHRGON bit in the CALCFG field to indicate that a noise source is present. For example, this bit can be set when a noisy charger is plugged into the user's product.

When noise suppression is not in operation (that is, when the noise suppression algorithm is in its off state), the Mean Acquisition Mode can be used for acquisition, if its use has been enabled by the MEANEN bit in the CALCFG field.

- **The Mean Acquisition Mode**

  The Mean Acquisition Mode operates on three sets of accumulated ADCs. It takes the mean of (averages) the three sets of ADCs. The Mean Acquisition Mode acquires the three sets of ADCs at the frequencies specified by the BASEFREQ and the two MFFREQ[] fields. The BASEFREQ field specifies the frequency to use to acquire the first set of ADCs by the Mean Acquisition Mode, and the MFFREQ[] fields specify the second and third frequencies to use.

  The primary purpose for the Mean Acquisition Mode is to help with stylus operation, as the quieter reported position gives better linearity [1]. Note that when noise suppression is triggered, the Mean Acquisition Mode is automatically disabled (see Figure 6-9).

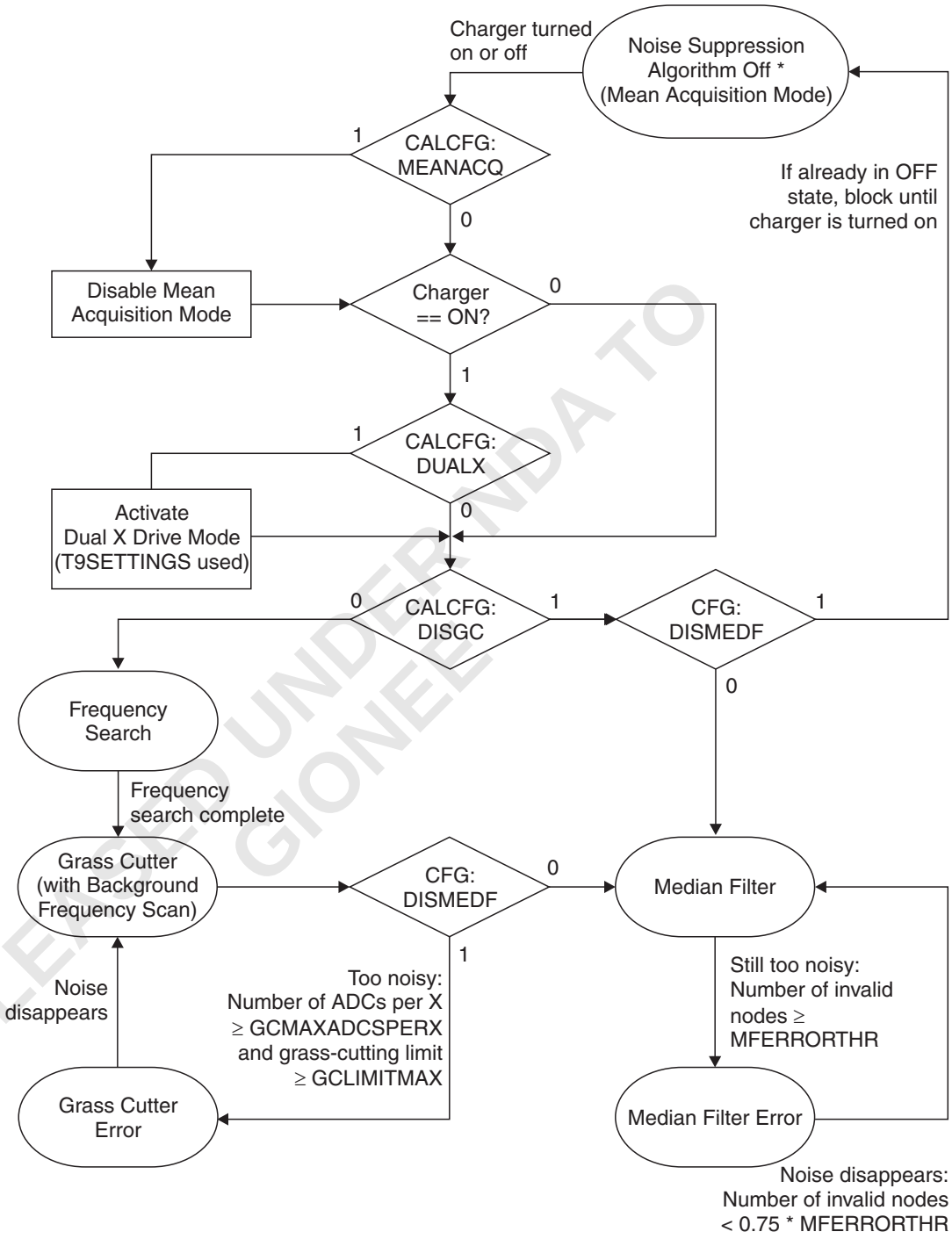
The Noise Suppression T48 object provides an alternative burst mode on the X lines known as Dual X Drive.

- **Dual X Drive**

  With Dual X Drive, X pulses occur on pairs of X lines instead of single X lines. This is useful when finger touches will cover more than one X line on a closely spaced X sensor matrix as it improves the signal-to-noise ratio (SNR). Note, however, that it will affect accuracy and linearity on the X edge. Furthermore, the enlarged touch means that the two-touch separation is increased and the classification of stylus touches is made more difficult.

1. See Section 6.6 on page 72 on for information on the Stylus T47 object.

Figure 6-9. Noise Suppression Algorithm



* The noise algorithm returns to Off state when charger is turned on or off. If the algorithm is already in the Off state, and both DISGC = 1 and DISMEDF = 1, it blocks until charger is turned on

Suggested initial settings for the Noise Suppression T48 object are given in Table 6-16.

Table 6-16. Suggested Initial Settings

Field	Suggested Initial Setting
CTRL	As required by user
CFG	FRQDRFT = 0 GCMODE = 4 (default) DISMEDF = 0 Other bits as required by user
CALCFG	INCBIAS, DUALX = 1 INCRST, MEANACQ, DISGC = 0 CHRGON as required by user
BASEFREQ	0
MFFREQ[]	0
GCACTVINVLDADCS	6 (default)
GCIDLEINVLDADCS	6 (default)
GCMAXADCSPERX	100 (default)
GCLIMITMIN	4 (default)
GCLIMITMAX	64
GCCOUNTMINTGT	10 (default)
MFINVLDDIFFTHR	20 (default)
MFINCADCSPXTHR	6 (approximate rule-of-thumb: number of X lines / 4)
MFERRORTHR	46 (approximate rule-of-thumb: number of X lines x 2)
SELFREQMAX	0

6.7.1 Configuration

Table 6-17. PROCG_NOISESUPPRESSION_T48

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved			RPTNLVL	RPTAPX	RPTFREQ	RPTEN	ENABLE
1	CFG	FRQDRFT	DISMEDF	Reserved		CHRGIN	GCMODE		
2	CALCFG	INCRST	INCBIAS	CHRGON	DUALX	MEANACQ	Reserved	DISGC	Reserved
3	BASEFREQ	Base frequency							
4 – 7	Reserved	Reserved							
8	MFFREQ[]	Median Filter frequency for second set of ADCs							
9		Median Filter frequency for third set of ADCs							
10 – 12	Reserved	Reserved							
13	GCACTVINVLDADCS	Grass cut minimum ADCs required for a valid node in active acquisition							
14	GCIDLEINVLDADCS	Grass cut minimum ADCs required for a valid node in idle acquisition							
15 – 16	Reserved	Reserved							
17	GCMAXADCSPERX	Maximum ADCs per X							
18	GCLIMITMIN	Minimum grass cut limit							
19	GCLIMITMAX	Maximum grass cut limit							

Table 6-17. PROCG_NOISESUPPRESSION_T48 (Continued)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
20	GCCOUNTMINTGT	GC count minimum target LSByte							
21		GC count minimum target MSByte							
22	MFINVLDDIFFTHR	Median filter invalid difference threshold							
23	MFINCADCSPXTHR	Median filter ADCs per X increase threshold LSByte							
24		Median filter ADCs per X increase threshold MSByte							
25	MFERRORTHR	Median filter error threshold LSByte							
26		Median filter error threshold MSByte							
27	SELFREQMAX	Maximum range of frequencies searched							
28 – 33	Reserved	Reserved							
34 – 53	T9SETTINGS[0]	Multiple Touch Touchscreen T9 Settings for Dual X Drive – Touchscreen 0							
54 – 73	T9SETTINGS[1]	Multiple Touch Touchscreen T9 Settings for Dual X Drive – Touchscreen 1							

CTRL Field

This is the main control field.

ENABLE: Enables the Noise Suppression T48 object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Allows this object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

RPTFREQ: Allows the object to send selected frequency change messages. Reporting is enabled if set to 1, and disabled if set to 0.

RPTAPX: Allows this object to send number of ADCs per X change messages to the host. Reporting is enabled if set to 1, and disabled if set to 0.

RPTNLVL: Allows this object to send noise level change messages. Reporting is enabled if set to 1, and disabled if set to 0.

CFG Field

GCMODE: Sets the Grass-cutting mode. See Table 6-18 for possible values. The default value of 0 means 4 (AND mode).

Table 6-18. Grass-cutting Modes

Value	Mode	Description
1	Median 3	Grass-cut around a median value using limits set by the grass-cutting limit. This median value is calculated from the first three ADCs of every acquisition.
2	Median 4	Grass-cut around a median value using limits set by the grass-cutting limit. This median value is calculated from the first four ADCs of every acquisition.
3	Median 5	Grass-cut around a median value using limits set by the grass-cutting limit. This median value is calculated from the first five ADCs of every acquisition.

Table 6-18. Grass-cutting Modes (Continued)

Value	Mode	Description
4	AND	Each ADC is compared with the two adjacent ADCs (one before and after). Both of the adjacent ADCs must be within range (that is, within the grass-cutting limit) for the ADC to be accepted; otherwise the ADC is grass-cut. This is the default mode.
5	OR	Each ADC is compared with the two adjacent ADCs (one before and after). At least one of the adjacent ADCs must be within range (that is, within the grass-cutting limit) for the ADC to be accepted; otherwise the ADC is grass-cut.

CHRGIN: Enables Charger Input detection. When Charger Input detection is enabled, the device can use the CHRG_IN pin as an input to detect the presence of a charger. Note that the CHRG_IN pin is shared with GPIO5, so GPIO5 must not be configured as an output in the GPIO/PWM Configuration T19 object for this to work (see Section 7.3 on page 89). Charger Input detection is enabled if set to 1, and disabled if set to to 0. An alternative method for detecting the presence of a charger is provided by the CHRGON bit in the CALCFG field.

DISMEDF: Disallows use of the Median filter. Use of the Median filter is disallowed if set to 1, and allowed if set to 0.

FRQDRFT: Allows the Noise Suppression T48 object to run 10 touch drift cycles immediately after switching frequency. This allows for a small amount of undercharging of the touchscreen. Freqency drift is enabled if set to 1, and disabled if set to 0.

CALCFG Field

This field configures the other noise suppression techniques (that is, all techniques except the Grass Cutter; see the CFG field). Setting this field will cause a recalibration to occur.

DISGC: Disallow use of the Grass Cutter. Forces the noise suppression algorithm to skip making use of the Grass Cutter. If this bit is set to 1, the Median Filter can be used without the Grass Cutter. If this bit set to 0, the Grass Cutter is used and then the Median Filter if there is too much noise for the Grass Cutter (see Figure 6-9).

MEANACQ: Allows Mean Acquisition Mode to be run when needed. This mode is used when the noise suppression algorithm is in its off state. Mean Acquisition Mode is allowed to run if set to 1, and disallowed if set 0.

DUALX: Allows Dual X Drive to be run when needed. When this bit set to 1, the noise suppression algorithm can use Dual X Drive to generate acquisition pulses on pairs of X lines instead of single X lines (X0 and X1, then X1 and X2, and so on). Note that Dual X Drive occurs only on those X lines that fall within an the area of the sensor covered by an enabled Multiple Touch Touchscreen T9 object. When Dual X Drive is active, then the T9SETTINGS block overrides some of the Multiple Touch Touchscreen T9's settings (see Section 5.2 on page 26).

Note that if Dual X Drive is enabled, an adjusted gain should be used, as calculated from the following formula [1]:

1. This formula means, therefore, that it is not recommended to enable Dual X Drive when the Multiple Touch Touchscreen T9's GAIN setting is already set to 2 or less.

Dual_X_Drive_gain = Multiple_Touch_Touchscreen_T9_GAIN - 3

The calculated Dual X Drive gain should be entered in the BLEN byte of the T9SETTINGS block (see Section 6-19 on page 83).

Note that if Dual X Drive is enabled, the minimium XSIZE in a Multiple Touch Touchscreen T9 object is 4 (that is, the minimium Touchscreen size is 4 by 3).

CHRGON: This bit is set can be set to 1 by the host to indicate that a charger is connected to the product. An alternative method for detecting the presence of a charger is provided by CHRG_IN pin and the CHRGIN bit in the CFG field.

INCBIAS: Increases the biases of the CTE integrators to allow more current through the integrators. This helps to prevent prevents under- and over-saturation. The bias is increased if set to 1, and left unchanged if set to 0.

INCRST: Increases the integrator reset level (that is, the default voltage level that the sampling capacitors are put into before starting a touch acquisition). This bit increases the integrator reset level to prevent under-saturation. The acquisition reset level is increased if set to 1, and left unchanged if set to 0.

BASEFREQ Field

Specifies the amount by which the burst frequency should be decreased, when compared to the burst frequency when the object is not in use. The maximum value for this field is determined by the following formula and is capped if necessary:

Acquisition Configuration T8 CHRGTIME + BASEFREQ ≤ 243

See also the CHRGTIME field in the Acquisition Configuration T8 object (Section 4.5 on page 17) for more details on the relationship between the base frequency and the charge time.

Range: 0 to maximum determined by formula above

MFFREQ[] Fields

These fields specify the Median Filter and/or Mean Acquisition Mode frequencies. The first Median Filter or Mean Acquisition Mode frame in each acquisition is decreased by the current base frequency (as determined by BASEFREQ). The second and third frames are decreased by the values specified in MFFREQ[0] and MFFREQ[1] respectively in addition to the base frequency.

The maximum value for the MFFREQ[] fields is determined by the following formula and is capped if necessary:

Acquisition Configuration T8 CHRGTIME + BASEFREQ + MFREQ[*n*] ≤ 243

Range: 0 to maximum determined by formula above

GCACTVINVLDADCS Field

This field sets the Grass Cut minimum ADCs required for a valid node in active acquisition. If fewer than this number of ADC samples are accepted by the Grass Cutter during an active acquisition, the node is marked as invalid. The range for this field is 0 to 63, where the default value of 0 means 6.

Range: 0 (6), 1 to 63

GCIDLEINVLDADCS Field

This field sets the Grass Cut minimum ADCs required for a valid node in idle acquisition. If fewer than this number of ADC samples are accepted by the Grass Cutter during an idle acquisition, the node is marked as invalid. The range for this field is 0 to 63, where the default value of 0 means 6.

Range: 0 (6), 1 to 63

GCMAXADCSPERX Field

This field specifies the Maximum ADCs per X for use when the Grass Cutter and the Median Filter are running. The Noise Suppression T48 object ensures that the number of ADCs per X does not exceed this number. A setting of 0 means 80.

Range: 0 (100), 1 to 255

Typical: 100

GCLIMITMIN Field

This field determines the lower bound for the Grass-cutting Limit for the Grass-cutting Mode specified by GCMODE (see "CFG Field" on page 79). The range for this field is 0 to 255, where the default value of 0 means 3.

Range: 0 (4), 1 to 255

Typical: 4

GCLIMITMAX Field

This field determines the upper bound for the Grass-cutting Limit for the Grass-cutting Mode specified by GCMODE (see "CFG Field" on page 79). If this field is less than GCLIMITMIN, then GCLIMITMIN is used.

Range: 0 to 255

Typical: 64

GCCOUNTMINTGT Field

The GC Count Minimum is the least number of ADC samples that was accumulated for any node on any touch sensor. The GCCOUNTMINTGT specifies the desired GC Count Minimum target. When the grass cutter is running, the noise suppression algorithm always tries to aim for this desired GC Count Minimum target by automatically adjusting the grass-cutting limit and the number of ADCs per X. The range for this field is 0 to 63, where the default value of 0 means 10.

Range: 0 (10), 1 to 63

MFINVLDDIFFTHR Field

This field sets the Median Filter Invalid Difference Threshold. This lets the Median Filter mark nodes as valid or invalid when they are processed by the Median Filter. If the difference (in delta counts) between the lowest and the highest of the three sets of accumulated ADCs is greater than this threshold, then that node is marked as invalid. The range for this field is 0 to 255, where a value of 0 means 20.

Range: 0 (20), 1 to 255

MFINCADCSPXTHR Field

This field determines if the number of ADCs per X needs increasing by the noise suppression algorithm when the Median Filter is runnning. It specifies a threshold for the number of invalid nodes, as determined by the MFINVLDDIFFTHR field. If the number of invalid nodes exceeds this threshold, then the Median Filter increases the number of ADCs per X. The number of ADCs per X is decreased again when the number of invalid nodes falls below this threshold minus a hysteresis margin. A typical value for this field is approximately the number of X lines divided by 4. The range for this field is 0 to 540 (maximum nodes), where a value of 0 means 6.

Range: 0 (6), 1 to 540 (maximum nodes)

Typical: 6

MFERRORTHR Field

This field sets the Median Filter Error Threshold when using the Median Filter in Median Filter Error state. If the number of invalid nodes is greater than the Median Filter Error Threshold, an error message is emitted and the noise suppression algorithm enters the Median Filter Error state (see the STATE field in Section 6.7.3 on page 86). A typical value for this field is approximately the number of X lines multiplied by 2. The range for this field is 0 to 540 (maximum nodes), where a value of 0 means 46.

Range: 0 (46), 1 to 540 (maximum nodes)

Typical: 46

SELFREQMAX Field

This field limits the number of frequencies that the noise suppression algorithm will search through when attempting to find and select a suitable burst frequency. The maximum range of frequencies searched will be limited to between 0 and SELFREQMAX. A SELFREQMAX of 0 means that there is no limiting by this field.

Range: 0 to 255

T9SETTINGS[] Blocks

These fields override some of the Multiple Touch Touchscreen T9 object settings when the noise suppression algorithm activates Dual X Drive (see Table 6-19). There is one T9SETTINGS block for each Multiple Touch Touchscreen T9 instance present in the device. See Section 5.2 on page 26 for details on the fields within a T9SETTINGS block.

Table 6-19. T9SETTINGS Block

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
n	BLEN	GAIN				Reserved			
n + 1	TCHTHR	Touch threshold							
n + 2	TCHDI	Touch detect integration for first touch							
n + 3	MOVHYSTI	Movement hysteresis, initial							
n + 4	MOVHYSTN	Movement hysteresis, next							
n + 5	MOVFILTER	DISABLE	FILTERLIMIT			ADAPTTHR			
n + 6	NUMTOUCH	Number of reported touches							
n + 7	MRGHYST	Merge hysteresis							
n + 8	MRGTHR	Merge threshold							

Table 6-19. T9SETTINGS Block (Continued)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
n + 9	XLOCLIP	X low clipping boundary width							
n + 10	XHICLIP	X high clipping boundary width							
n + 11	YLOCLIP	Y low clipping boundary width							
n + 12	YHICLIP	Y high clipping boundary width							
n + 13	XEDGECTRL	SPAN	DISLOCK	CORRECTIONGRADIENT					
n + 14	XEDGEDIST	X edge correction distance							
n + 15	YEDGECTRL	SPAN	RELUPDATE	CORRECTIONGRADIENT					
n + 16	YEDGEDIST	Y edge correction distance							
n + 17	JUMPLIMIT	Maximum position jump							
n + 18	TCHHYST	Touch threshold hysteresis							
n + 19	NEXTTCHDI	Touch detect integration for subsequent touches							

6.7.2 Configuration Checks

The Noise Suppression T48 object causes a configuration check to be performed in the following circumstances:

- When the object is enabled (that is, the ENABLE bit is set in the CTRL field)
- When certain fields, including the CTRL field, are changed (as listed in Table 6-20).

In addition, some fields will cause an automatic recalibration to be performed (see Table 6-20).

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14), and the device halts until the error has been corrected. To fix the error, the object settings should be checked to verify that they are all within their allowed limits, as stated in the field descriptions.

If a configuration check occurs, some of the Noise Suppression T48 object's processing is reset. Note that this happens whenever any object causes a configuration check, not just the Noise Suppression T48 object.

Table 6-20. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes (1)	Yes (1)	None
CFG	Yes	No	None
CALCFG	Yes	Yes	None
BASEFREQ	Yes	No	None
MFFREQ[]	Yes	No	None
GCACTVINVLDADCS	Yes	No	None
GCIDLEINVLDADCS	Yes	No	None
GCMAXADCSPERX	Yes	No	None
GCLIMITMIN	Yes	No	None

Table 6-20. Configuration Checks (Continued)

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
GCLIMITMAX	Yes	No	None
GCCOUNTMINTGT	Yes	No	None
MFINVLDDIFFTHR	Yes	No	None
MFINCADCSPXTHR	Yes	No	None
MFERRORTHR	Yes	No	None
SELFREQMAX	No	No	None
T9SETTINGS			
BLEN	Yes	Yes	None
TCHTHR	Yes	No	None
TCHDI	Yes	No	None
MOVHYSTI	Yes	No	None
MOVHYSTN	Yes	No	None
MOVFILTER	Yes	No	None
NUMTOUCH	Yes	No	None
MRGHYST	Yes	No	None
MRGTHR	Yes	No	None
XLOCLIP	Yes	No	None
XHICLIP	Yes	No	None
YLOCLIP	Yes	No	None
YHICLIP	Yes	No	None
XEDGECTRL	Yes	No	None
XEDGEDIST	Yes	No	None
YEDGECTRL	Yes	No	None
YEDGEDIST	Yes	No	None
JUMPLIMIT	Yes	No	None
TCHHYST	Yes	No	None
NEXTTCHDI	Yes	No	None

1. If the ENABLE bit is toggled on or off.

6.7.3 Messages

The message data for the Noise Suppression T48 object is shown in Table 6-21.

Table 6-21. Message Data for PROCG_NOISESUPPRESSION_T48

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Reserved		NLVLCHG	STATCHG	Reserved	ALGOERR	APXCHG	FREQCHG
2	ADCSPERX	Current number of ADCs per X							
3	FREQ	Current selected frequency							
4	STATE	Current noise suppression algorithm state							
5	NOISELEVEL	Current measured noise level							

STATUS Field

This field reports the status of the filters.

FREQCHG: Indicates that the noise suppression algorithm has changed the selected frequency at least once since the previous message. This field is set to 1 if the selected frequency has changed, and 0 otherwise.

APXCHG: Indicates that the noise suppression algorithm has changed the number of ADCs per X at least once since the previous message. This field is set to 1 if the number of ADCs per X has changed, and 0 otherwise.

ALGOERR: Indicates that the noise suppression algorithm cannot suppress the effects of noise. The system will halt until one of the following occurs:

- The noise drops to an acceptable level
- The noise is redefined to be acceptable (for example, by changing this object's settings)
- This object is disabled

STATCHG: Indicates that the noise suppression algorithm state has changed at least once since the previous message. The details of the state change are reported in the STATE field (see below).

NLVLCHG: Indicates that the noise level has changed at least once since the previous message.

ADCSPERX Field

This field reports the current number of ADCs per X being used in acquisitions.

FREQ Field

This field reports the current selected frequency.

STATE Field

This field reports the current noise suppression state (see Table 6-22).

Table 6-22. STATE Field Values

Code	State
0	Off
1	Frequency Search
2	Grass Cutter

Table 6-22. STATE Field Values (Continued)

Code	State
3	Grass Cutter Error
4	Median Filter
5	Median Filter Error

NOISELEVEL Field

This field reports the current measured noise level.

7. Support Objects

7.1 Introduction

Support objects provide additional functionality on the device. Table 7-1 lists the support objects on the mXT540E.

Table 7-1. Support Objects

Object	Description
Communications Configuration T18 (SPT_COMMSCONFIG _T18)	Configures additional communications behavior for the device. See Section 7.2.
GPIO/PWM Configuration T19 (SPT_GPIOPWM_T19)	Creates a bank of digital I/O pins. These pins can then be controlled from the host by writing to the object's configuration memory space. See Section 7.3.
Self Test T25 (SPT_SELFTEST _T25)	Performs self-test routines to find faults on a touch sensor. See Section 7.4.
User Data T38 (SPT_USERDATA_T38)	Provides a data storage area for user data. See Section 7.5.
Digitizer HID Configuration T43 (SPT_DIGITIZER_T43)	Configures the USB Digitizer HID interface and the USB Descriptors associated with it. See Section 7.6.
Message Count T44 (SPT_MESSAGECOUNT_T44)	Provides a count of pending messages. See Section 7.7.
CTE Configuration T46 (SPT_CTECONFIG_T46)	Controls the capacitive touch engine for the device. See Section 7.8.

7.2 Communications Configuration T18 (SPT_COMMSCONFIG _T18)

The Communications Configuration T18 object specifies additional communications behavior for the device.

Table 7-2. SPT_COMMSCONFIG _T18

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	DISMNTR	Reserved				MODE	Reserved	
1	COMMAND	$\overline{\text{CHG}}$ line command code							

CTRL Field

MODE: Selects the $\overline{\text{CHG}}$ line mode. If this bit is set to 0 (the default), the $\overline{\text{CHG}}$ line operates in Mode 0. If this bit is set to 1, the $\overline{\text{CHG}}$ line operates in Mode 1. Refer to the mXT540E datasheet for more information on the $\overline{\text{CHG}}$ line modes.

DISMNTR: Disables the bus monitor. This monitors the I^2C-compatible lines. If either line is low for more than 200 ms, the I^2C-compatible hardware is reset. This ensures that a "stuck" I^2C-compatible bus is detected. The bus monitor is disabled if this bit is set to 1 and enabled if it is set to zero. Note that the bus monitor is not active in deep sleep mode.

COMMAND Field

The COMMAND field provides direct control over the $\overline{\text{CHG}}$ line. This overrides the operation of the $\overline{\text{CHG}}$ line controlled by the MODE bit in the CTRL field. Entering one of the command codes listed in Table 7-3 alters the state of the $\overline{\text{CHG}}$ line.

Table 7-3. Command Codes

Command	Description
0	No command (default)
1	Return $\overline{\text{CHG}}$ line to normal operation, as determined by the MODE bit of the CTRL field
2	Force the $\overline{\text{CHG}}$ line high (inactive)
3	Force the $\overline{\text{CHG}}$ line low (active)

7.3 GPIO/PWM Configuration T19 (SPT_GPIOPWM_T19)

The GPIO/PWM Configuration T19 object sets up the general-purpose I/O pins. The pins can be set as input, output or PWM pins.

Note that the reporting of signals on the input pins occurs only at the appropriate point in the acquisition cycle. This means that the input signal must not occur faster than the Idle or Active Acquisition Interval specified in the Power Configuration T7 object (see Section 4.4 on page 15). Similar criteria apply if an output bit is changed, PWM is enabled or a trigger occurs. These operations do not take place instantaneously and enough time must be allowed for them to happen.

Notes:

1. The GPIO2 function shares a pin with the SYNC function configured by the Acquisition Configuration T8 object. If the SYNC function is enabled, then this takes precedence over any specified GPIO2 function.
2. If GPIO2 is set as an output pin, the burst pulse synchronization functionality of the CTE Configuration T46 object is not available. GPIO2 must be set as an input, or the GPIO/PWM Configuration T19 object disabled, for the CTE Configuration T46 burst pulse synchronization to work.
3. The GPIO5 function shares a pin with the CHRG_IN function (Charger Input detection) configured by the Noise Suppression T48 object. If Charger Input detection is enabled by the Noise Suppression T48 object, then this takes precedence over any specified GPIO5 function. GPIO5 must not be configured as an output, or the GPIO/PWM Configuration T19 object disabled, for Charger Input detection to work.
4. The GPIO1, GPIO2, GPIO4 and GPIO5 functions share pins with the SNS0, SNSK0, SNS1 and SNSK1 pins respectively. GPIO/PWM functionality cannot be used on any of these pins if they are already configured for use as part of a self-capacitance Proximity Key in the appropriate Proximity Key T52 object.

7.3.1 Configuration

Table 7-4. SPT_GPIOPWM_T19

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved					FORCERPT	RPTEN	ENABLE
1	REPORTMASK *	Report mask							
2	DIR *	Pin direction							
3	INTPULLUP *	Internal pull-ups							
4	OUT *	Output value							

Table 7-4. SPT_GPIOPWM_T19 (Continued)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
5	WAKE *	Wake up on bit change							
6	PWM *	Enable PWM							
7	PERIOD	PWM period							
8	DUTY[0]	Duty cycle for PWM0							
9	DUTY[1]	Duty cycle for PWM1							
10	DUTY[2]	Duty cycle for PWM3							
11	DUTY[3]	Duty cycle for PWM4							
12	TRIGGER[0]	Trigger for GPIO0							
13	TRIGGER[1]	Trigger for GPIO1							
14	TRIGGER[2]	Trigger for GPIO3							
15	TRIGGER[3]	Trigger for GPIO4							

* These fields are bit fields with each bit representing a GPIO pin, as in Table 7-5.

Table 7-5. GPIO/PWM Configuration T19 Bit Field Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved		GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

CTRL Field

ENABLE: Enables or disables the use of the GPIO/PWM Configuration T19 object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Enables or disables the GPIO/PWM Configuration T19 object to report messages. If this bit is set, the device sends messages on bit changes. Reporting is enabled if set to 1, and disabled if set to 0.

FORCERPT: Writing a 1 to this bit forces the GPIO/PWM Configuration T19 object to send a message with its current status. The object then clears this bit. The RPTEN bit must be enabled for this to work. See Section 7.3.2 on page 92 for information on the message data.

REPORTMASK Field

Specifies a bit mask that suppresses the sending of messages on bit changes. Set these bits to 1 if the corresponding GPIO pins should not generate a message on a bit change. The pins must be set as input pins in the DIR field. The default value of 0 causes all bit changes to generate messages.

DIR Field

Sets the direction of the pins: 1 = GPIO*n* is an output pin; 0 = GPIO*n* is an input pin. The default value of 0 sets all the pins to input pins.

INTPULLUP Field

Enables the internal pull-up resistors for any input pins. If these bits are set to 1, the internal pull-up resistors for the corresponding pins are enabled. The pins must be set as input pins in the DIR field. If this field is set to 0 (default), all the internal pull-up resistors for the pins are disabled.

OUT Field

Sets the state of the output GPIO pins. Setting these bits to 1 sets the pins high. Clearing these bits to 0 sets the pins low. The pins must already be set as output pins in the DIR field. Note that the OUT settings may be used for any PWM pins when the device enters Deep Sleep (see the PWM field below for more information).

WAKE Field

Causes the device to wake up on bit changes if it is asleep. If these bits are set to 1, the device will wake up on a bit change on the IO pins. When the bit change occurs, the device wakes up, reports the GPIO message and then returns to sleep. The pins must already be set as input pins in the DIR field. If this field is set to 0 (default), the device will remain asleep on bit changes.

Note that enabling the WAKE option will increase the current consumption.

PWM Field

Specifies that the GPIO pins are to be used as PWM pins. If these bits are set to 1, the pins are used as the corresponding PWM pins. In this case, the pins must also be set as output pins in the DIR field. Setting this field overrides the OUT and TRIGGER fields. If this field is set to 0 (default), PWM functionality is disabled and the pins are used as the corresponding IO pins.

Note: PWM functionality can be enabled only on GPI0, GPIO1, GPIO3 and GPIO4.

If the device enters Deep Sleep, [1] PWM functionality is turned off. In this case one of two thing happens to any active PWM pins. If a trigger applies and the sources are active, the pins are inverted (see the TRIGGER field for more details). Alternatively, the pins revert to the corresponding value in the OUT field. When the device awakes from Deep Sleep mode, PWM operation is resumed and the PWM pins become active again.

Note that enabling PWM will increase the current consumption.

PERIOD Field

Sets the period of the PWM waveform for any pins that have been set as PWM pins by the PWM field. This setting applies to all the specified PWM pins. The value in the PERIOD field is 0 to 255, equating to a PWM period of 40 µs to 10.07 ms (25 kHz to 99.30Hz). This can be calculated as follows:

$$\text{PERIOD} = \frac{(p \times 1.5 \times 10^{6}) - 60}{59}$$

where p is the required period in seconds.

A low PERIOD setting with a very low or very high DUTY value may not be accurate due to hardware limitations. If the PERIOD is set to 0, the PWM waveforms are output only if the DUTY setting is greater than 4.

Range: 0 to 255

1. Caused by a setting of ACTVACQINT = 0 or IDLEACQINT = 0 in the Power Configuration T7 object.

DUTY[] Fields

These fields select the duty cycles for the PWM channels. The corresponding pin must be enabled in the PWM field. The range is 0 to 255 (0 to 100 percent), where each increment is equivalent to 100/255 percent.

Range: 0 to 255 (0 to 100 percent)

TRIGGER[] Fields

These fields select a trigger source (see Table 7-6) for the corresponding GPIO pin. The trigger inverts the appropriate GPIO pin if the source object is active. The pin must be set as an output pin in the DIR field.

Note: Trigger functionality can be enabled only on GPI0, GPIO1, GPIO3 and GPIO4.

On entering Deep Sleep mode, any triggered pins stick at their current triggered state until the device wakes again. When the device enters Deep Sleep mode, the PWM is turned off. This happens even though the PWM setting overrides the TRIGGER setting in normal operation. This permits the trigger to function immediately before the device enters Deep Sleep mode. Thus the pin is inverted (triggered) if the source is active. See the PWM field for more information.

Table 7-6 gives the permitted values for the TRIGGER field.

Table 7-6. TRIGGER Field Values

Value	Description
0 (or not listed below)	Trigger disabled.
1	Triggered by the first instance of a Multiple Touch Touchscreen T9 object whenever any touch is detected.
3	Triggered by the first instance of a Touch Suppression T42 object (that is, the TCHSUP bit in the message data) whenever touch suppression has been applied.
4	Triggered by the first instance of a Key Array T15 object whenever any touch is detected.
5	Triggered by the first instance of a Proximity Key T52 object whenever the Proximity Key is in a detect state.

Range: 0 to 4

7.3.2 Messages

The message data for the GPIO/PWM Configuration T19 object is shown in Table 7-7.

Table 7-7. Message Data for SPT_GPIOPWM_T19

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Input pin states							

STATUS: Gives the current state of all GPIO pins that are set as inputs. Any GPIO pin that is set as an output reports as a zero. This field is a bit field with each bit representing a GPIO pin. See Table 7-5 on page 90 for the format of this field.

7.4 Self Test T25 (SPT_SELFTEST_T25)

The Self Test T25 object runs self-test routines in the device to find faults in the sense lines and electrodes. The Self Test T25 object runs a series of test sequences. As soon as the first failure is found, the test run stops and the object reports a message.

The following tests can be run:

- AVdd Power test. This tests that AVdd power is present.
- Pin fault test. This tests the sense pins on the device (X0..X17 and Y0..Y29) so that low-resistance shorts (line-to-line, line-to-Vdd and line-to-GND) can be detected, as well as resistive line-to-line shorts.
- Signal limit tests. These test the signals from each of the touch objects on the device. These tests are run per touch object and not for the entire matrix. The user must specify the expected minimum and maximum level for each touch object under test.
- User-defined sense pin tests. The Self Test T25 object provides a Special Sense Pin Test Mode that allows user-defined tests to be performed on the sense pins.

To run a test, the CMD field is set to the code of the test to be run. The Self Test T25 object then immediately runs the test once and then stops. At the end of the test, the CMD field is cleared. If reporting is enabled (see "CTRL Field"), the Self Test T25 object also sends a report message with the result of the test (see Section 7.4.3 on page 96). The tests can be called at the following times:

- For pin fault testing: when the sense lines are not in use.
- For signal limit testing: at any time after acquisition when the signals are stable (such as not during calibration).
- For all other tests: at any time.

If an error is found, the calibration command in the Command Processor T6 object should be used once the error has been cleared.

7.4.1 Configuration

Table 7-8. SPT_SELFTEST_T25

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved						RPTEN	ENABLE
1	CMD	Test code of test to run							
2 – 5	UPSIGLIM[0]	Higher signal limit LSByte for touch object 0							
		Higher signal limit MSByte for touch object 0							
	LOSIGLIM[0]	Lower signal limit LSByte for touch object 0							
		Lower signal limit MSByte for touch object 0							
...									

Table 7-8. SPT_SELFTEST_T25 (Continued)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(4*n*-2) – (4*n*+1)	UPSIGLIM[*n*-1]	Higher signal limit LSByte for touch object *n*-1							
		Higher signal limit MSByte for touch object *n*-1							
	LOSIGLIM[*n*-1]	Lower signal limit LSByte for touch object *n*-1							
		Lower signal limit MSByte for touch object *n*-1							

Note: *n* = number of touch objects (see Section 5), assigned in the following order:
All Multiple Touch Touchscreen T9 objects
All Key Array T15 objects
All Proximity Key T52 objects

CTRL Field

ENABLE: Enables the object. The object is enabled if set to 1, and disabled if set to 0.

RPTEN: Allows the object to send status messages to the host through the Message Processor T5 object. Reporting is enabled if set to 1, and disabled if set to 0.

CMD Field

This field is used to send test commands to the device. Valid test commands are listed in Table 7-9.

Table 7-9. Test Commands

Code	Test Description
0x00	The CMD field is set to 0x00 after test completed
0x01	Test for AVdd power present
0x12	Run the pin fault test
0x17	Run the signal limit test
0xF0	Enter the special sense pin test mode
0xFE	Run all the tests (except for special sense pin test)

Writing 0x01 to the CMD field causes the device to perform an immediate check for the presence of the AVdd power line. Note that this test is also automatically run every 200 ms if the object is enabled.

Writing 0x12 to the CMD field causes the device to run a pin fault test.

Writing 0x17 to the CMD field causes the device to run a signal limit test.

Writing 0xF0 to the CMD field causes the device to enter the Special Sense Pin Test Mode. This mode can be entered only by this command. It is not triggered by a “test all” command. Once in this mode, the CMD field is used for sending further commands to the device to control the sensor pins. To exit from this mode, the device must be hard reset.

When the device is in the Special Sense Pin Test Mode, a sequence of three command bytes musts be sent to the device to trigger a pin state change, as in Table 7-10.

Table 7-10. Command Bytes for the Special Sense Pin Test Mode

Byte	Name	Description
1	PINCHGCMD	Changes the state of a sense pin (as determined by the PORTNUM and PINNUM bytes): 0x01: Input 0x02: Output low 0x03: Input 0x04: Output high
2	PORTNUM	Indicates the port number (1 or 2) on which the pin will be controlled. Port 1 is for the X pins and Port 2 is for the Y pins.
3	PINNUM	Specifies the pin number (1 to 18 for X pins, 1 to 30 for Y pins) within the port of the pin to be controlled.

Writing 0xFE to the CMD Field causes the device to run all the tests (except for Special Sense Pin Mode).

UPSIGLIM[] and LOSIGLIM[] Fields

These 16-bit fields specify the higher and lower signal limits for the signal tests. UPSIGLIM specifies the upper limit and LOSIGLIM specifies the lower limit. When the test is run, the signals must fall between LOSIGLIM and UPSIGLIM for the test to pass. Any signal limit values outside this range will generate a signal limit error in the test. The limits are specified on a per touch object basis.

The UPSIGLIM and LOSIGLIM values for each touch object are specified using the appropriate format for the touch object, as given in Section 2.8 on page 8 [1]. UPSIGLIM must be greater than LOSIGLIM.

Mutual-capacitance touch objects:

The values chosen for UPSIGLIM and LOSIGLIM should differ depending on whether the unit is a design prototype or a production item.

For design prototypes, UPSIGLIM and LOSIGLIM should each be set to a value between 4160 and 10400 (specified in the appropriate format).

For production use, LOSIGLIM should never be lower than 3360 and UPSIGLIM should never be higher than 12480. The exact values, however, are determined by analyzing the data from actual sensor samples. Using the full range from 3360 to 12480 can cause sensors with too much variation to pass the test.

Note: The gain used for the signal limit tests depend on whether Dual X Drive is enabled. If Dual X Drive is enabled on the Noise Suppression T48 object, the gain specified by the Noise Suppression T48 object is used for the signal limit test (see "T9SETTINGS[] Blocks" on page 83). Otherwise the gain set by the Multiple Touch Touchscreen T9 object is used.

Range for values: 3360 (0x4D20 in offset binary format) to 12480 (0x70C0) (HISIGLIM must be greater than LOSIGLIM)

Recommended Values (Design): 4160 (0x5040) to 10400 (0x68A0)

Recommended Values (Production): Determined on a case-by-case basis

1. That is as unsigned 16-bit values or in offset binary format, as appropriate.

Self-capacitance touch objects:

LOSIGLIM should never be lower than 100 and UPSIGLIM should never be higher than 2200. The exact values, however, are determined by analyzing the data from actual sensor samples. Using the full range can cause sensors with too much variation to pass the test.

Range for values: 100 to 2200
(HISIGLIM must be greater than LOSIGLIM)

7.4.2 Configuration Checks

The object causes a configuration check to be performed in the following circumstances:

- When the object is enabled (that is, the ENABLE bit is set on in the CTRL field)
- When certain fields are changed (as listed in Table 7-11)

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14), and the device halts until the error has been corrected. To fix the error, the object settings should be checked to verify that they are all within their allowed limits, as stated in the field descriptions.

Table 7-11. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	Yes (1)	No	None
CMD	No	No	None
UPSIGLIM[]	Yes	No	Error if UPSIGLIM[*n*] is below LOSIGLIM[*n*]
LOSIGLIM[]	Yes	No	

1. If the ENABLE bit is toggled on or off.

7.4.3 Messages

The Self Test T25 object reports the test results in its message data. The message data for the Self Test T25 object is shown in Table 7-12.

Table 7-12. Message Data for SPT_SELFTEST_T25

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Result code							
2 – 6	INFO	Result data							

STATUS Field

This field contains a result code that indicates the success or failure of the test. Valid codes are given in Table 7-13.

Note: The device must be recalibrated once all errors have been resolved.

Table 7-13. Result Codes

Code	Test Result
0xFE	All tests passed.
0xFD	The test code supplied in the CMD field is not associated with a valid test.
0xFC	The test could not be completed due to an unrelated fault (for example, an internal communications problem).
0x01	AVdd is not present.
0x12	The test failed because of a pin fault. The INFO fields indicate the first pin fault that was detected (see Table 7-14).
0x17	The test failed because of a signal limit fault.

INFO Field

This field contains the result data of the test. The actual data depends on which test was run, as detailed below. Note that if a test does not generate data, the INFO field consists solely of reserved bytes.

INFO Field – AVdd Power Fault

The AVdd Power Fault test generates no data. The INFO field consists of reserved bytes only.

INFO Field – Pin Fault

If the result was a pin fault, the INFO field has the data format shown in Table 7-14.

Table 7-14. Test Result Data for a Pin Fault

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2	SEQ_NUM	Test sequence number							
3	X_PIN	Failing X pin number + 1							
4	Y_PIN	Failing Y pin number + 1							

SEQ_NUM Field

The test sequence number of the test in which a fault is found. The sequence numbers and their meanings are listed in Table 7-15.

Table 7-15. Sequence Numbers

Sequence Number	Description of Test
0x01	All pins are pulled low to Ground. This test detects shorts to Vdd (that is, they fail high). This test can detect resistive shorts, but only up to ~39 kΩ.
0x02	All pins are pulled high. This test detects shorts to Ground (that is, they fail low). This test can detect resistive shorts, but only up to ~68 kΩ.
0x03	Walking 1: All the pins are set to zero. Then, starting with the first pin, each pin in turn is pulled high (set to “1”), leaving all the other pins set to zero. The effect of this operation is that a “1” bit “walks” along the pins. This test detects shorts between sense pins. This test can detect resistive shorts, but only up to ~68 kΩ.

Table 7-15. Sequence Numbers (Continued)

Sequence Number	Description of Test
0x04	Walking 0: All the pins are set to 1. Then, starting with the first pin, each pin in turn is cleared (set to 0), leaving all the other pins set to 1. The effect of this operation is that a “0” bit “walks” along the pins. This test detects shorts between sense pins. This test can detect resistive shorts, but only up to ~68 kΩ.
0x05	Alternate 1’s and 0’s: The pins are set to an alternating pattern of 1’s and 0’s, starting with a 1 on the first pin. Then, the pins are set to an alternating pattern of 0’s and 1’s, this time starting with a 0 on the first pin. The effect of this two-part operation is that the pins are first set to a 10101... pattern and then to a 01010... pattern. This test detects direct shorts between sense pins; this test cannot detect resistive shorts.

X_PIN Field

The number of the X sense pin that failed, where 0 means no pin failed and 1 to 255 is the X line number plus 1.

Y_PIN Field

The number of the Y sense pin that failed, where 0 means no pin failed and 1 to 255 is the Y line number plus 1.

INFO Field – Signal Limit Error

If the result was a signal limit error, the INFO field has the data format shown in Table 7-16.

Table 7-16. Test Result Data for a Signal Limit Error

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2	TYPE_NUM	Touch object type number							
3	TYPE_INSTANCE	Touch object type instance							

TYPE_NUM Field

The type number of the touch object for which a signal limit error was found (see Section 2.4.2 on page 3).

TYPE_INSTANCE Field

The instance number of the touch object for which a signal limit error was found. Note that the touch object is indicated by the TYPE_NUM field.

7.5 User Data T38 (SPT_USERDATA_T38)

The User Data T38 object provides a small area on the device for the user to store their own custom data, such as the user's own version information. The size of this area varies from device to device. On the mXT540E the size of this area is 64 bytes.

Any data stored in this object will be backed up to the non-volatile memory when the Command Processor T6 object processes a backup command (see Section 4.3 on page 13). The data is not included in the checksum reported by the Command Processor T6 after a backup has been processed.

Table 7-17. SPT_USERDATA_T38

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 – 63	DATA	User data							

DATA: The user's data.

7.6 Digitizer HID Configuration T43 (SPT_DIGITIZER_T43)

This object configures the USB Digitizer HID interface and the USB descriptor associated with it.

Note: The logical maximum X and Y sizes in the Digitizer descriptor are linked to the touchscreen resolution settings (XRANGE and YRANGE). These must be set to 4095 (12-bit) if the USB interface is being used. Any other resolution settings may result in indeterminate behavior. The reported digitizer coordinates are also affected by the SWITCH bit in the ORIENT field of the Multiple Touch Touchscreen T9 object.

Table 7-18. SPT_DIGITIZER_T43

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved					RWKEN	Reserved	ENABLE
1	HIDIDLERATE	HID IDLE rate							
2 – 3	XLENGTH	Sensor X length (mm)							
4 – 5	YLENGTH	Sensor Y length (mm)							
6	RWKRATE	Sensor acquisition interval in remote wakeup mode							

CTRL Field

ENABLE: Enables or disables the use of the Digitizer HID interface. If the Digitizer HID is disabled, the Digitizer HID endpoint is enumerated but no touch messages are sent. The Digitizer HID is enabled if set to 1, and disabled if set to 0. This control is useful if the Digitizer HID interface is not being used to receive touch positions, as it can be disabled to avoid wasteful USB communications.

RWKEN: Enables remote wakeup. This allows the device to respond to USB remote wakeup requests. Specifically, the device continues to scan if it is suspended when the operating system enables remote wakeup using the SET_FEATURE (1) command.

If this bit is set to 0 (remote wakeup disabled), the device acts as if a normal suspend had been sent and never sends a resume event.

If this bit is set to 1 (remote wakeup enabled), and the device is suspended with USB remote wakeup enabled using a SET_FEATURE command, the device drops to a sensor refresh rate specified by the RWKRATE field. In this case, the device sends a resume event on the USB bus when a touch on the screen is detected.

HIDIDLERATE Field

This field sets the default HID IDLE rate [1] of the digitizer status updates. The HID IDLE rate determines how frequently the Digitizer interface resends the status of all active touches. If the HID IDLE rate is set to zero, only changes in status will cause the digitizer to update the status of any active touch. This field is read once at power-on and is used to set the default value for the HID IDLE rate. After power-on, the HID IDLE rate setting may be changed and checked by the USB HID SET_IDLE and GET_IDLE [1] commands.

The HID IDLE rate is specified in units of 4 ms, where 0 means off.

Range: 0 (off), 1 to 255 (in 4 ms units)

XLENGTH and YLENGTH Fields

These fields configure the physical size fields in the Digitizer descriptor. These fields are read once at power-on. The reported digitizer coordinates are affected by the SWITCH bit in the ORIENT field of the Multiple Touch Touchscreen T9 object, as are the physical X maximum and Y maximum descriptor fields.

The size is specified in millimeters. If XLENGTH or YLENGTH is set to zero, the Digitizer descriptor uses the size and pitch information specified by the Multiple Touch Touchscreen T9 object (see Section 5.2 on page 26).

XLENGTH/YLENGTH Range: 0 (derived from Multiple Touch Touchscreen T9 object), 1 to 65535 (length in mm)

XLENGTH/YLENGTH Default: 0 (derived from Multiple Touch Touchscreen T9 object)

RWKRATE Field

Specifies the sensor acquisition interval (in ms) in remote wakeup mode. If remote wakeup is enabled, this value is used instead of the IDLEACQINT and ACTVACTINT values in the Power Configuration T7 object (see Section 4.4) when the device is put into suspend. A typical value is 100 ms (10Hz scan rate).

Range: 0 to 255 (ms)

Typical: 100

7.7 Message Count (SPT_MESSAGECOUNT_T44)

This object contains a count of the number of pending messages. This enables the host to use a direct memory access (DMA) transfer to read the messages from the Message Processor T5 object. See Section 4.2 on page 11 for more information.

Table 7-19. SPT_MESSAGECOUNT_T44

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	COUNT	Message count							

COUNT Field

This field specifies the number of pending messages that are ready to send.

1. SET_FEATURE, the HID IDLE rate, SET_IDLE and GET_IDLE are part of the USB HID specification.

7.8 CTE Configuration T46 (SPT_CTECONFIG_T46)

The CTE Configuration T46 object controls how the capacitive touch engine (CTE) in the device samples the capacitive touchscreen signals. Specifically, it determines the generation of the bursts on the X lines and how these are sampled.

The X lines are sampled one at a time, with 1 or more analog-to-digital conversions (ADCs) performed on each X line in turn. The ADCs are arranged in “Sync Groups”, with one or more ADCs in each Sync Group [1]. The number of ADCs per Sync Group is determined by the ADCSPERSYNC field.

There is one or more Sync Groups in each burst on an X line. For example, if the device is configured to have 10 Sync Groups per X line, there will be 10 Sync Groups on X0, followed by 10 Sync Groups on X1, and so on (see Figure 7-1). The number of Sync Groups per X line is specified by the ACTVSYNCSPERX and IDLESYNCSPERX fields.

Thus, the total number of ADCs per X is equivalent to:

ADCSPERSYNC x ACTVSYNCSPERX or IDLESYNCSPERX

For example, in the third example in Figure 7-1 ADCSPERSYNC is set to 2 and ACTVSYNCSPERX/IDLESYNCSPERX is set to 3, so the total number of ADCs is 6.

Note that the Sync Groups do not have to be synchronized to the SYNC line and a setting of 0 for the ADCSPERSYNC field means that no synchronization required. In this case the ACTVSYNCSPERX/IDLESYNCSPERX field alone determines the number of Sync Groups per X line (see the first example in Figure 7-1).

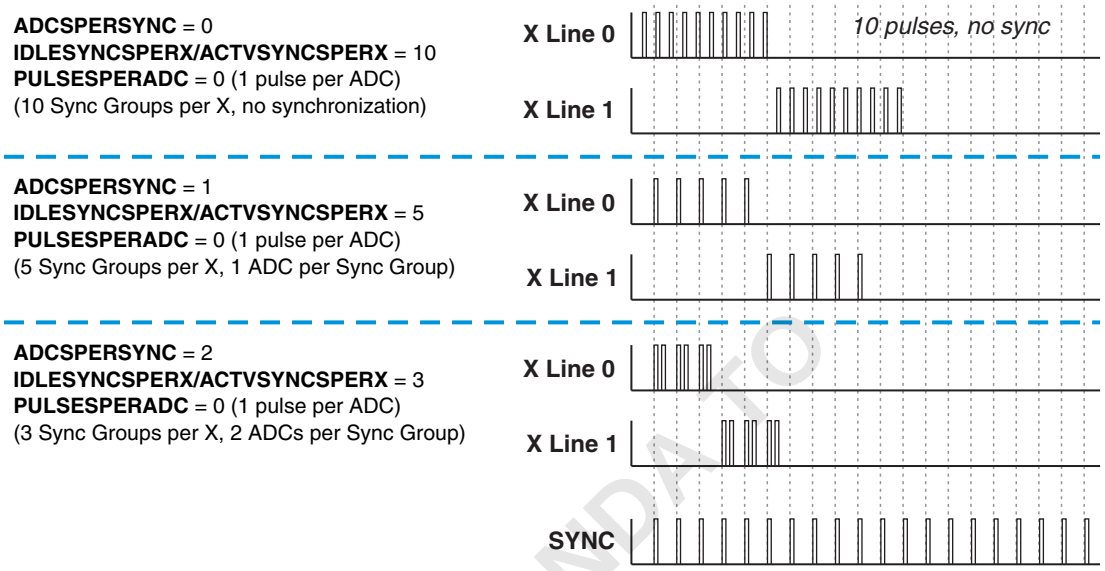
The final aspect that can be configured is the number of pulses for each ADC. By default, the number of pulses per ADC is 1 but this can be changed to up to 4 using the PULSESPERADC field. Thus, the total number of pulses per X line is as follows:

PULSESPERADC x ADCSPERSYNC x ACTVSYNCSPERX or IDLESYNCSPERX

Note: The mXT540E uses multiplexing to measure all available Y lines. This means that twice as many pulses as expected will be seen if a line is probed with a scope.

1. These groups of ADCs are known as Sync Groups because they can be synchronized to the SYNC line (although this is not mandatory).

Figure 7-1. CTE Configuration T46 fields



7.8.1 Configuration

Table 7-20. SPT_CTECONFIG_T46

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved					LCALEN	Reserved	
1	MODE	XY Mode							
2	IDLESYNCSPERX	Number of Sync Groups per X when idle							
3	ACTVSYNCSPERX	Number of Sync Groups per X when active							
4	ADCSPERSYNC	Number of ADC conversions per Sync Group							
5	PULSESPERADC	Number of pulses for each ADC conversion							
6	XSLEW	X pulse slew rate							
7 – 8	SYNCDELAY	Delay from SYNC edge LSByte							
		Delay from SYNC edge MSByte							

CTRL Field

LCALEN: Enables long calibration. Normally, the active Sync Groups per X (ACTVSYNCSPERX) is used for calibration. When long calibration is enabled, the maximum number of Sync Groups per X (that is, 63) is used. Long calibration is enabled if LCALEN is set to 1, and disabled if set to 0

MODE Field

This field specifies which touchscreen XY mode is to be used. The valid codes and their meanings are given in Table 7-21.

Note: The MODE field requires the device to be restarted before the mode change takes place. The device settings must therefore be backed up before the device is restarted.

Table 7-21. Mode Codes

Code	Mode
0	18X by 30Y

IDLESYNCSPERX Field

This field sets the number of Sync Groups per X when idle. The range for this field is 0 to 63, where a value of 0 means 8. (1)

Range: 0 (8), 1 to 63 (number of ADC conversion sets)

Default: 0 (8)

ACTVSYNCSPERX Field

This field sets the number of Sync Groups per X when active. The range for this field is 0 to 63, where a value of 0 means 8. (1)

Range: 0 (8), 1 to 63 (number of ADC conversion sets)

Default: 0 (8)

ADCSPERSYNC Field

This field sets the number of ADCs per Sync Group (that is, the number of ADC conversions that occurs on each sync edge). A value of 0 means no synchronization is required (that is, it is not synchronized to the input signal on the SYNC pin). In this case the IDLESYNCSPERX or ACTVSYNCSPERX field alone determines the number of ADCs per X (2).

Note: If synchronization is turned on (that is, ADCSPERSYNC = 0), GPIO2 cannot be configured as an output in the GPIO/PWM Configuration T19 object. GPIO2 must be set as an input, or the GPIO/PWM Configuration T19 object disabled, for the CTE Configuration T46 burst pulse synchronization to work.

Range: 0 (1, no synchronization), 1 to 255 (number of ADC conversions)

Default: 0 (1, no synchronization)

PULSESPERADC Field

This field sets the number of pulses for each ADC conversion. The range for this field is 0 to 3, which represents 1 to 4 pulses.

Range: 0 to 3 (number of pulses minus 1)

XSLEW Field

This field sets the X slew rate. The valid values for this field are given in Table 7-22.

Table 7-22. XSLEW Values

Value	Meaning
0	500 ns rise and fall time
1	350 ns rise and fall time

1. With synchronization off (that is, ADCSPERSYNC = 0), IDLESYNCSPERX and ACTVSYNCSPERX become equivalent to ADCs per X. This is the same as GCAF Depth in the CTE Configuration T28 object on previous maXTouch chips.
2. In this case the number of ADcs per Sync Group is effectively 1 and the IDLESYNCSPERX or ACTVSYNCSPERX field alone determines the number of ADCs per X.

SYNCDELAY Field

This 16-bit field specifies the additional delay from when the SYNC edge occurs to when the X edge is generated. The delay is specified in clock cycles (33.3 ns), where a value of 0 means no additional delay.

Recommended Range: 0 (no additional delay), 1 to 10,000 (delay in clock cycles)

7.8.2 Configuration Checks

The CTE Configuration T46 object causes a configuration check to be performed in the following circumstances:

- When certain fields are changed (as listed in Table 7-23)

A configuration check may determine that a configuration error has occurred (for example, if a setting is set outside of its allowed range or a conflict has occurred between two settings). This is signaled to the host (see Section 4.3.2 on page 14), and the device halts until the error has been corrected. To fix the error, the object settings should be checked to verify that they are all within their allowed limits, as stated in the field descriptions.

Table 7-23. Configuration Checks

Field	Changing The Field Causes...		Effect of Configuration Checks On Field
	Configuration Check	Automatic Recalibration	
CTRL	No	No	None
MODE	Yes	No	Error if out of range
IDLESYNCSPERX	Yes	No	None
ACTVSYNCSPERX	Yes	No	None
ADCSPERSYNC	No	No	None
PULSESPERADC	Yes	Yes	Error if out of range
XSLEW	No	Yes	None
SYNCDELAY	No	No	None

7.8.3 Messages

The message data for the CTE Configuration T46 object is shown in Table 7-24.

Table 7-24. Message Data for SPT_CTECONFIG_T46

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Reserved							CHKERR

STATUS Field

CHKERR: If set, this error implies that the device has not been correctly calibrated in the factory.

Appendix A. Checksum Calculation

A.1 24-bit CRC

A.1.1 Introduction

The device uses a 24-bit cyclic redundancy check (CRC) in the following places:

- To check the integrity of the main Information Block for the device
- To check the integrity of the Information Block T254 object
- To check the integrity of the configuration settings held in the nonvolatile memory

These CRC checksums allow the host to be confident that the memory map layout has been read over the communications bus correctly.

Note: The C code in this appendix uses the type declarations `uint8_t`, `uint16_t` and `uint32_t` for 8-bit, 16-bit and 32-bit unsigned integers respectively.

A.1.2 24-bit CRC Algorithm

Each checksum is generated by running an algorithm which takes two new bytes of data and combines them with the current checksum to produce a new checksum value.

The sample code below shows how to calculate the checksum for each 16-bit word in a byte stream.

```
uint32_t crc24(uint32_t crc, uint8_t firstbyte, uint8_t secondbyte)
{

  static const uint32_t crcpoly = 0x80001b;
  uint32_t result;
  uint16_t data_word;

  data_word = (uint16_t)((uint16_t)(secondbyte << 8u) | firstbyte);

  result = ((crc<<1u) ^ (uint32_t)data_word);

  if(result & 0x1000000) // If bit 25 is set
  {
    result ^= crcpoly;// XOR result with crcpoly
  }

  return result;
}
```

The checksum routine is called iteratively to calculate the CRC two bytes (16-bit word) at a time. This means that two bytes must be read from the device before performing each iteration of the checksum. Therefore, if an odd number of bytes is read from the device, the host's checksum code should add a zero byte to the end of the byte stream to make the sequence even. For example, if the following stream of 7 bytes is received from the device:

byte1 — byte2 — byte3 — byte4 — byte5 — byte6 — byte7

The checksum could be calculated as follows:

```
uint32_t CRC = 0;
CRC = crc24(CRC, byte1, byte2)
CRC = crc24(CRC, byte3, byte4)
CRC = crc24(CRC, byte5, byte6)
CRC = crc24(CRC, byte7, 0)        /* <- zero added for the last checksum */

CRC = CRC & 0x00FFFFFF;           /* <- mask the 32-bit result to 24-bit */
```

An alternative method of calling the CRC calculation function is to use a loop, as shown in Section A.1.3

Table A-1 shows an example block of 32 bytes and the CRCs these generate. The bytes consist of 31 data bytes plus an additional zero byte to even up the count. You can use the data in this table to verify the validity of any coded CRC routine.

Table A-1. Example CRC Calculations for 24-bit CRC

Byte Pairs		
First Byte	**Second Byte**	**CRC Calculation Result**
0x0	0xFF	0xFF00 (Intermediate CRC – partial calculation)
0x11	0xEE	0x11011 (Intermediate CRC – partial calculation)
0x22	0xDD	0x2FD00 (Intermediate CRC – partial calculation)
0x33	0xCC	0x53633 (Intermediate CRC – partial calculation)
0x44	0xBB	0xAD722 (Intermediate CRC – partial calculation)
0x55	0xAA	0x150411 (Intermediate CRC – partial calculation)
0x66	0x99	0x2A9144 (Intermediate CRC – partial calculation)
0x77	0x88	0x55AAFF (Intermediate CRC – partial calculation)
0x88	0x77	0xAB2276 (Intermediate CRC – partial calculation)
0x99	0x66	0xD6226E (Intermediate CRC – partial calculation)
0xAA	0x55	0x2C116D (Intermediate CRC – partial calculation)
0xBB	0x44	0x586661 (Intermediate CRC – partial calculation)
0xCC	0x33	0xB0FF0E (Intermediate CRC – partial calculation)
0xDD	0x22	0xE1DCDA (Intermediate CRC – partial calculation)
0xEE	0x11	0x43A841 (Intermediate CRC – partial calculation)
0xFF	0x0	0x87507D (Expected CRC – final calculation)

A.1.3 Information Block Checksum

The checksum for the Information Block should be calculated by the host on start-up when the Information Block is first read. This should be compared to the checksum value at the end of the Information Block. If there is a mismatch, an error has occurred.

The following code shows how to use the example `crc24()` function given in Section A.1.2 on page 105 to calculate the CRC checksum for the Information Block.

```
uint32_t crc = 0;           /* Calculated CRC */
uint16_t crc_area_size;     /* Size of data for CRC calculation */
uint8_t *mem;               /* Data buffer */
uint8_t i;
uint8_t status;

/* 7 bytes of version data, 6 * NUM_OF_OBJECTS bytes of object table. */
crc_area_size =  ID_INFORMATION_SIZE +
                 info_block->info_id.num_declared_objects *
                 OBJECT_TABLE_ELEMENT_SIZE;

mem = (uint8_t *) malloc(crc_area_size);
if (mem == NULL)
{
  /* Handle error */
}

/*
 * Read the data using a function written for this purpose
 * Here, it is assumed that the function is named read_mem()
 * and that it returns a status code with the value READ_MEM_OK.
 * It is assumed to have the following prototype:
 * uint8_t read_mem( uint16_t Address, uint8_t ByteCount, uint8_t *Data )
*/
status = read_mem(0, crc_area_size, mem);

if (status != READ_MEM_OK)
{
  /* Handle error */
}

/*
 * Call the CRC function crc24() iteratively to calculate the CRC,
 * passing it two bytes at a time.
*/
i = 0;
```

```
while (i < (crc_area_size - 1))
{
  crc = crc24(crc, *(mem + i), *(mem + i + 1));
  i += 2;
}

/*
 * Call the crc24() with the final byte,
 * plus an extra 0 value byte to make the sequence even.
*/
crc = crc24(crc, *(mem + i), 0);

free(mem);

/* Final result */
crc = (crc & 0x00FFFFFF);  /* <- mask the 32-bit result to 24-bit */
```

A.1.4 Configuration Checksum

The configuration checksum checks the integrity of the memory map when the device's nonvolatile memory is written to. Typically this is when the device is initially set in the factory or during subsequent firmware upgrades. The host should perform a CRC on the entire contents of the memory map from the Power Configuration T7 object onwards. This CRC should then be compared with the expected checksum.

A.2 8-bit CRC

A.2.1 Introduction

An 8-bit CRC can be used in the following places:

- During communications to check that data has been transmitted over the communications bus correctly (see Section A.2.3).
- To check the integrity of data in messages from the Message Processor T5 object when receiving messages (see Section A.2.4 on page 111).

A.2.2 8-bit CRC Algorithm

The sample code below shows how to calculate the 8-bit checksum.

```
uint8_t crc8(unsigned char crc, unsigned char data)
{
  static const uint8_t crcpoly = 0x8c;
  uint8_t index;
  uint8_t fb;
  index = 8;

  do
  {
    fb = (crc ^ data) & 0x01;
    data >>= 1;
    crc >>= 1;
```

```
    if (fb)
      crc ^= crcpoly;
  } while (--index);

  return crc;
}
```

A.2.3 Communications Checksum Mode

A.2.3.1 Introduction

In communications checksum mode an 8-bit CRC is added to all data transmissions. The CRC is sent at the end of the data as the last byte before the STOP condition. The CRC is calculated on all the bytes sent, including the address bytes.

To indicate that a checksum is to be included, the most significant bit of the MSByte of the address is set to 1. In the following examples, therefore, the start address of 0x1234 is sent as 0x9234.

The following sections give examples of I²C-compatible write and read operations.

A.2.3.2 I²C-compatible Write Example

This example sets the address pointer to 0x1234. The address is sent to the device with the most significant bit of the MSByte set to 1 (that is, 0x9234). This indicates I²C-compatible communications checksum mode. The example then writes five bytes to the device: four data bytes plus a checksum byte (see Figure A-1).

Figure A-1. Example I²C-compatible Write With Checksum

		Address							
		LSByte	MSByte	Data	Data	Data	Data	CRC	
START	SLA-W	0x34	0x92	0x96	0x9B	0xA0	0xA5	0x7A	STOP

The example I²C-compatible command in Figure A-1 writes the four bytes to contiguous addresses:

0x96 to address 0x1234

0x9B to address 0x1235

0xA0 to address 0x1236

0xA5 to address 0x1237

The I²C-compatible command sends a checksum (in this case, 0x7A) as the last byte before the STOP condition. The device compares this checksum with its own checksum calculated from the data sent. If the two checksums do not match, the Command Processor T6 object in the device generates a COMMSERR message (1) (see Section 4.3 on page 13).

Table A-2 shows the sequence of the bytes in this example and the intermediate calculations of the CRC.

1. The order of messages is not guaranteed so messages may be out of order. Other intervening messages may be received before this one.

Table A-2. CRC Calculations for Example I²C-compatible Write

Bytes Sent by Host		CRC Calculations in the Device	
0x34	Address LSByte	0xDF	Intermediate CRCs (partial calculations)
0x92	Address MSByte	0xBB	
0x96	Data	0xDE	
0x9B		0x79	
0xA0		0xCB	
0xA5		0x7A*	Expected CRC (final calculation)
0x7A*	Actual CRC		

* These two values should match

A.2.3.3 I²C-compatible Read Example

A checksum can be added to I²C-compatible reads of the Message Processor T5 object. This contains a checksum as its last byte when I²C-compatible communications checksum mode is enabled in the Message Processor T5 object. No other I²C-compatible reads can have a checksum.

The example in Figure A-2 reads the message data from the Message Processor T5 object in the device. It is assumed in this example that the size of the Message Processor T5 is 7 bytes and that its address is 0x1234. The address is sent to the device with most significant bit of the MSByte set to 1 (that is, 0x9234). This indicates I²C-compatible communications checksum mode.

Figure A-2. Example I²C-compatible Read With Checksum

Set Address Pointer

		Address LSByte	Address MSByte	CRC	
START	SLA-W	0x34	0x92	0xBB	STOP

Read Data

		Report ID	Data	Data	Data	Data	Data	Data	Data	CRC	
START	SLA-R	0x01	0x9B	0xA0	0xA5	0xAA	0xAF	0xB4	0xB9	0xA8	STOP

The example consists of two operations. The first operation is an I²C-compatible write of the address to the device. The second operation is an I²C-compatible read of the message data.

Table A-3 shows the sequence of the initial write bytes with the intermediate calculations of the CRC.

Table A-3. CRC Calculations for Example I²C-compatible Write of Address

Bytes Sent by Host		CRC Calculations in the Device	
0x34	Address LSByte	0xDF	Intermediate CRC (partial calculation)
0x92	Address MSByte	0xBB*	Expected CRC (final calculation)
0xBB*	Actual CRC		

* These two values should match

An extra byte that holds the checksum (0xBB in this case) of the start address of the Message Processor T5 object is sent as the last byte before the STOP condition. This prevents a COMMSERR message being generated by the Command Processor T6 object.

Table A-4 gives the sequence of the read bytes with the intermediate calculations of the CRC.

Table A-4. CRC Calculations for Example I²C-compatible Read of Message Data

Bytes Sent by Device		CRC Calculations in Host	
0x01	Report ID	0x5E	Intermediate CRCs (partial calculations)
0x9B	Data	0xF5	
0xA0		0xE4	
0xA5		0x18	
0xAA		0x8E	
0xAF		0x7D	
0xB4		0x56	
0xB9		0xA8*	Expected CRC (final calculation)
0xA8*	Actual CRC		

* These two values should match

A.2.4 Reading the CRC for the Message Processor T5 Data

The following code shows how to use the example `crc8()` function given in Section A.2.2 on page 108 to calculate the CRC checksum for the data in the Message Processor T5 object.

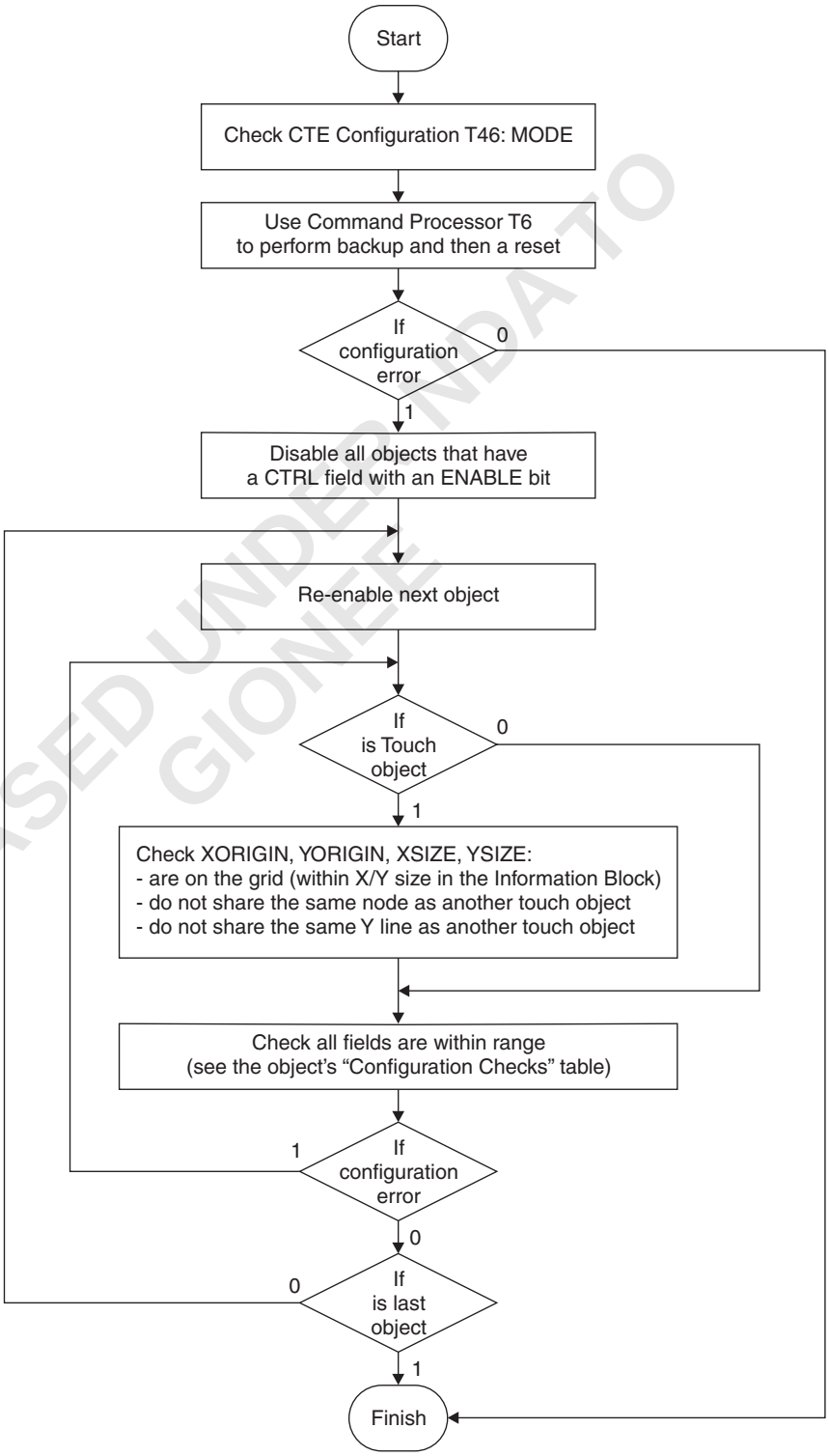
```
uint8_t crc = 0;
uint8_t data_in;

for(i=0; i<MESSAGEPROCESSOR_SIZE; i++)
{
  data_in = read_byte();
  crc = crc8(crc, data_in);
}
```

```
if(crc == 0)
{
  /* CRC is OK - do something appropriate */
  crc_pass();
}
else
{
  /* CRC failed - handle error */
  crc_fail();
}
```

Appendix B. Locating Configuration Errors

Figure B-1 shows the algorithm for locating configuration errors. See Section 2.6 on page 7 for more information on the configuration checks performed by the device.

Figure B-1. Algorithm for Locating Configuration Errors



Appendix C. Bootloading Procedure

Note: To use the bootloader, a legal Reflash Agreement must first be executed.

The maXTouch touch sensors have a common bootloading procedure that allows them to be forced into a special application update mode of operation.

The bootloader ID for the mXT540E is 0x0A.

For more information on the bootloading procedure, refer to the following application note: (1)

- QTAN0051 – *Bootloading Procedure for Atmel® Touch Sensors Based on the Object Protocol*

1. A legal Reflash Agreement is required.

Appendix D. Future Information Block T254 Object

D.1 Introduction

Currently the Object Table entries are held within the Information Block for the device. The current format, however, limits the object type codes to 8 bits. It is anticipated that at some time in the future new objects will be issued with 16-bit type codes (that is, codes above 255). The Information Block T254 object will therefore be added to the protocol to accommodate Object Table elements with 16-bit type codes. This means that if the Information Block T254 object is present on a device, it must be checked for any further Object Table entries.

Note: It is important for future compatibility that any *current* driver code is written to allow for the presence of the Information Block T254 object when it is eventually used. The host driver code must parse the Object Table, as at present, and also process the Information Block T254 object if it is found to be present on the device.

This appendix gives the format of the future Information Block T254 object so that the driver code can be written accordingly.

D.2 Information Block T254 (GEN_INFOBLOCK16BIT_T254)

This object acts as an extension to the Information Block at the start of the chip's memory map and holds additional Object Table entries. Specifically, it hold entries for those objects that have 16-bit type codes (that is, type codes above 255).

Table D-1. GEN_INFOBLOCK16BIT_T254

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 – 6	Object Table	Object Table Element 0							
7 – 13	Object Table	Object Table Element 1							
...									
*m**7 – (*m**7)+6	Object Table	Object Table Element *m*							
(*m*+1)*7 – (*m*+1)*7+2	Checksum	24-bit checksum (3 bytes)							

Object Table Element Fields

These 7-byte fields hold the additional elements in the Object Table. The format of each element is similar to that of the Object Table elements held directly in the main Object Table for the device (see Section 2.4 on page 3), except that the type code occupies two bytes instead of one. The format of each element is given in Table D-2.

Table D-2. Object Table Element Format

Byte	Description
0	Type code LSByte
1	Type code MSByte
2	Start position LSByte
3	Start position MSByte

Table D-2. Object Table Element Format (Continued)

Byte	Description
4	Size - 1
5	Instances - 1
6	Number of Report IDs per instance

Type code (2 bytes): This 16-bit field holds the unique type code for the object to identify it. This is the number after the "_T" suffix at the end of the object's internal name as given in Section 3 to Section 7. For example, the type code for the Command Processor T6 is 6 (from GEN_COMMANDPROCESSOR_T6).

Start position (2 bytes): This 16-bit field holds the start location of the object in the memory map.

Size - 1 (1 byte): This field holds the size (minus 1) of the object in the memory map. This is stored as Size-1, so it is effectively the offset to the end of the object.

Instances - 1 (1 byte): This holds the number of instances (minus 1) of the object in the memory map. The number of instances can be calculated by adding 1 to this number (see Table 2-4 on page 6). The different instances of an object are arranged consecutively in the memory map.

Number of Report IDs per instance (1 byte): This field holds the number of report IDs for each instance of the object. See Section 2.4.6 on page 4 for more information on report IDs.

Checksum

A checksum of the contents of the Information Block T254 object. This allows the host to check that the Object Table elements stored in the Information Block T254 object have been read correctly over the communications interface. See Appendix A on page 105 for details on calculating the checksum.

Appendix E. mXT540E vs mXT224E Gain Settings

The GAIN settings in the mutual-capacitance touch objects have a larger range than those for the mXT224E. This appendix gives a comparison of the gain settings on the mXT540E with those on the mXT224E.

Table E-1. Gain Settings

mXT540E		mXT224E	
Sensor Capacitance Cx (pF) Maximum	**GAIN Setting**	**Sensor Capacitance Cx (pF) Maximum**	**GAIN Setting**
0.54	15		
0.68	14	0.625	7
0.86	13	0.71	6
1.08	12	0.83	5
1.36	11	1	4
1.72	10	1.25	3
2.17	9	1.67	2
2.73	8	2.5	1
3.44	7		
4.33	6	5	0

Table of Contents

Associated Documents

- Datasheet: mXT540E – *maXTouch 540-channel Touchscreen Sensor IC*
- Application Note: QTAN0051 – *Bootloading Procedure for Atmel® Touch Sensors Based on the Object Protocol*
- Application Note: QTAN0059 – *Using the maXTouch Self Test Feature*
- Application Note: QTAN0070 – *Recovering from Palm Touches During Calibration with maXTouch™ Touchscreen Controllers*
- Application Note: QTAN0078 – *maXTouch Stylus Tuning*

Errata

The Object Protocol for mXT540E firmware 1.0 has the errors listed below. These will be corrected in future versions of the device.

- **I²C-compatible bus**

  An I²C-compatible bus error is not handled properly if a STOP condition appears where an ACK/NACK but would normally be received. A very long clock stretch will occur until the Communications Configuration T18 Bus monitor recovers the hardware. The workaround is to ensure that the bus monitor in the Communications Configuration T18 object is enabled to allow the hardware to recover.

- **Proximity Key T52**

  FXDDTHR, MVNULLRATE and MVDTHR fields
  These fields incorrectly mask bit 7 instead of bit 15, which prevents settings with bit 7 set. The workaround is to use values that do not set bit 7 to ensure consistent behavior when this error has been fixed. In addition, the host should never set bit15 to “1”.

- **Acquisition Configuration T8**

  SYNC field
  The SYNC feature can block the GPIO2 pin from being used for alternate features if the Acquisition Configuration T8’s SYNC feature is enabled but no sync edge is ever supplied. The workaround is never to enable Acquisition Configuration T8’s SYNC feature unless a sync signal is present.

Revision History

Revision Number	History
Revision AX – March 2011	Initial release for firmware revision 0.2, die revision B
Revision BX – May 2011	Updated for firmware revision 0.4, die revision E • Section 4.5 (Acquisition Configuration T8): internal name changed to GEN_ACQUISITIONCONFIG_T8. • Section 7.3: GPIO/PWM Configuration T19 object added. • Section 7.4 (Self Test T25): HISIGLIM renamed to UPSIGLIM. • Other minor updates to aid clarity.
Revision CX – June 2011	Updated for firmware revision 1.0, die revision G • Section 4.5 (Acquisition Configuration T8): charge time settings added. Minor updates to reflect addition of Proximity Key T52 object. • Section 4.6: Data Source T53 object added. • Section 5.4: Proximity Key T52 object added. • Section 6.7: Noise Suppression T48 object updated. CTRL, CFG and CALCFG fields updated. Flow diagram updated. NOISELEVEL added to messages. • Section 7.3 (GPIO/PWM Configuration T19): TRIGGER[] fields can now be triggered by Proximity Key T52 objects. • Section 7.4 (Self Test T25): Description of signal limts changed. • Appendix E (mXT540E GAIN settings) added • "Errata" added. • Other minor updates to aid clarity.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 487-2600

International

Atmel Asia
Unit 01-05 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong
Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Touch Technology Division
1560 Parkway
Solent Business Park
Whiteley
Fareham
Hampshire
PO15 7AG
UNITED KINGDOM
Tel: (+44) 844 894 1920
Fax: (+44) 1489 557 066

Atmel Munich GmbH
Business Campus
Parkring 4
D- 85748 Garching b.
MUNICH
Tel.: (+49) 89-31970-111
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
JAPAN
Tel: (+81) 3-3523-3551
Fax: (+81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
touch@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature