MPTT (V40)

MCU Based Maxtouch Framework and Protocol Guide

Version 2.0: T126 definition changed for 4x mode

C 2019 Microchip Technology Inc.

Table of Contents

MCU Based Maxtouch Framework and Protocol Guide 1
Table of Contents 1
To our Valued Customers 3
1.0 OverView 3
1.1 Introduction 3
1.2 MCU Based Maxtouch Framework 3
1.3 Memory Map Structure 4
1.4 Information Block 5
1.5 ID Information 6
1.6 Object Table 6
1.7 Objects 7

1.8 Configuration Checks 8
1.9 Byte Order 9
1.10 Data Values 9
1.11 Configuration and Tuning 9
1.12 Software Tools 9
2.0 Debug Objects 10
2.1 Introduction 10
2.2 Diagnostic Debug T37 Object 10
3.0 General Objects 14
3.1 Introduction 14
3.2 Message Processor T5 Object 14
3.3 Command Processor T6 Object 15
3.4 Power Configuration T7 Object 17
3.5 Acquisition Configuration T8 Object 18
4.0 Touch Objects 20
4.1 Introduction 20
4.2 Key Array T15 Object 21
4.3 Multiple Touch Touchscreen T9 Object 23
5.0 Signal Processing Objects 26
5.1 Introduction 26
5.2 Noise Suppression T72 Object 27
6.0 Support Objects 27
6.1 Introduction 27
6.2 Communication Configuration T18 Object 27
6.3 Self Test T25 Object 28
6.4 Message Count T44 Object 32
6.5 Auxiliary Touch Configuration T104 Object 32
6.6 Self Capacitance Global Configuration T109 Object 34
6.7 Self Capacitance Configuration T111 Object 35
Appendix A: Measurement processing on QTouch 40
Appendix B: CHECKSUM CALCULATION 42
Appendix C: Revision History 43

To our Valued Customers

1.0 OverView

1.1 Introduction

This document describes in detail the Object Protocol for the Microchip MCU with touch module.
The Object Protocol provides a single common interface across the Microchip MCU with touch module. 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.

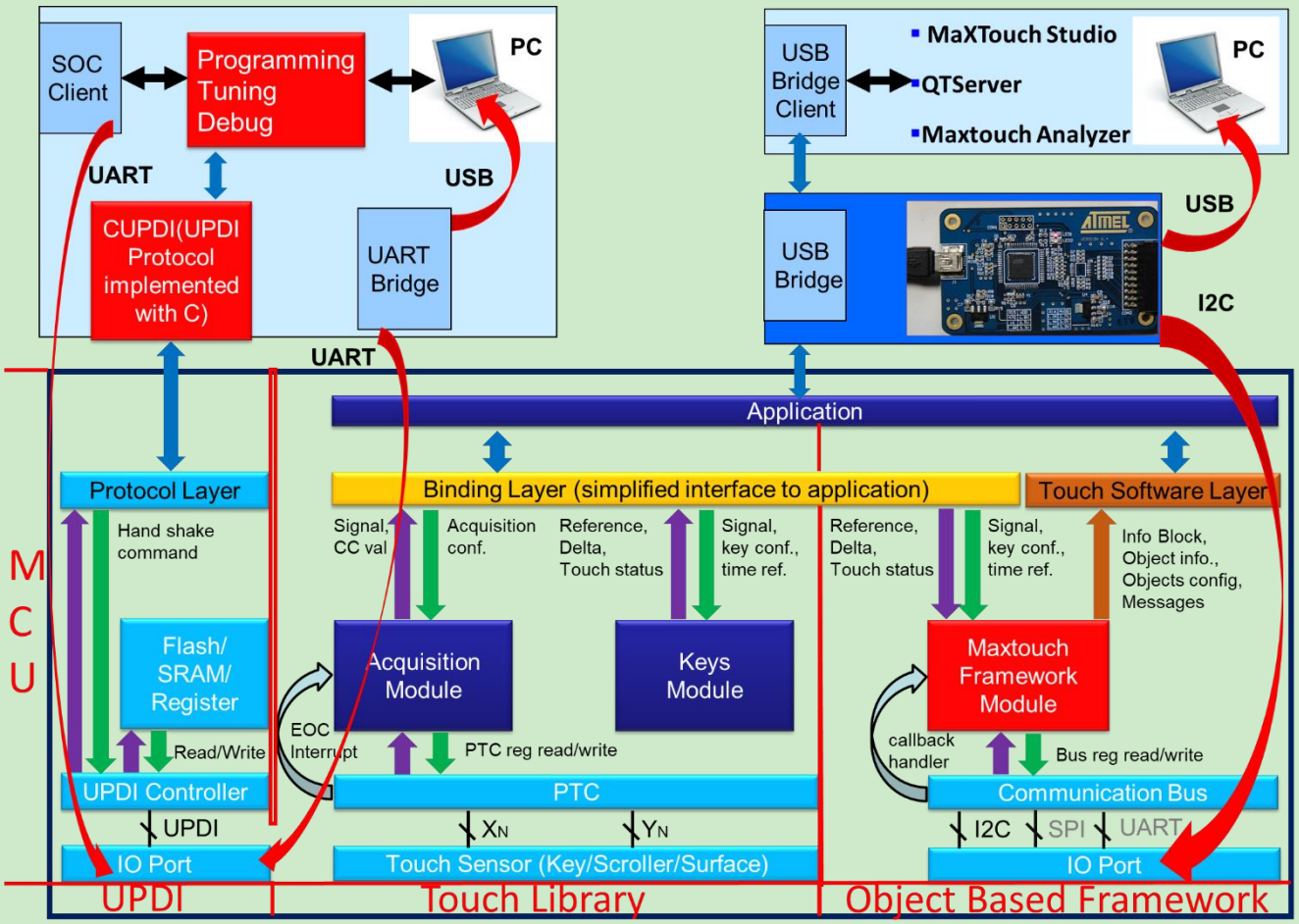
1.2 MCU Based Maxtouch Framework

In order to be compatibility with maxtouch and use a series of maxtouch tools, the MCU based maxtouch framework has been developed. Microchip MCU with touch module uses Touch Library to implement touch function. The framework is an interaction with

Touch Library binding layer. It provides a touch software layer to be used with application and a communication bus to talk with USB bridge board. The relationship between Touch Library and the Object Based Framework is shown in FIGURE 1-1. On the other hand, UPDI is used to program, tune and debug through the UPDI protocol. UPDI protocol is implemented as a hardware module and the software UPDI protocol in SOC client or PC is needed to complete the program, tune and debug. Both the UPDI and the object based framework are used to tune the MCU with touch module.

Users can easily use the maxtouch tools to debug and test the MCU touch solution with Touch Library and Object Based Framework. There're a series of tools that can be reused.

FIGURE 1-1: Framework and Connection



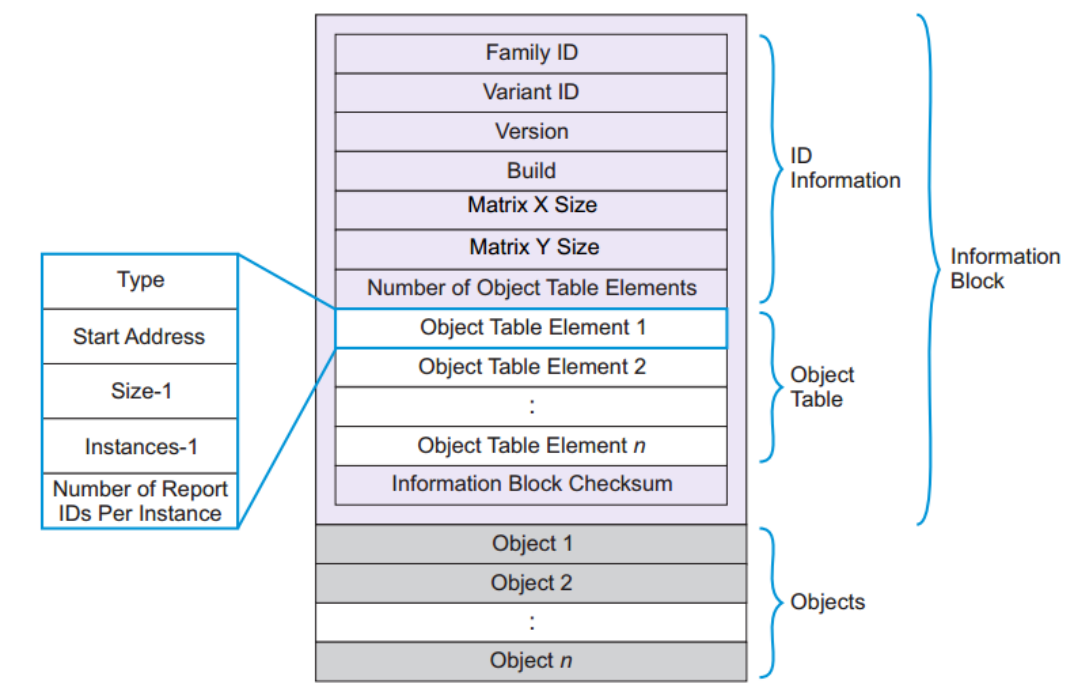
1.3 Memory Map Structure

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. Where appropriate, 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 1-2.

There are some special objects that can use their memory space for a unique purpose. An example is the Command Processor T6 object, which executes a command when a certain value is written into its memory space.

FIGURE 1-2: GENERIC MEMORY MAP STRUCTURE



From Figure 1-2 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 1.4 “Information Block”). It is further sub-divided into the ID Information Block (see Section 1.5 “ID Information”) and the Object Table (see Section 1.6 “Object Table”).
• The objects themselves (see Section 1.7 “Objects”)

1.4 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 determine which objects exist, where they are located in the memory map and their sizes. The host driver can therefore read the device Information Block and gather enough information to be able to communicate with the device.
The Information Block is positioned at the start of the device 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:
• **ID Information Fields** – These include:
- Standard ID fields that make up the unique identifier for the device
- The size of the touchscreen matrix the device supports
- The number of objects in the Object Table
• **Object Table** – This acts as an “index” to the objects in the memory map.
• **Information Block Checksum** – This allows the host to check that the Information Block has been read correctly over the communications interface.
Table 1-1 shows the contents of the Information Block.

TABLE 1-1: INFORMATION BLOCK LAYOUT

Byte	Description of Field	Section
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

1.5 ID Information

The first four bytes of the Information Block are used to identify the device and its version, as in Table 1-2.

TABLE 1-2: DEVICE IDENTIFIER FIELDS

Field	Description
Family ID	A unique identifier that indicates the device family. The family ID for the mXT449TD-AT is 0xA4.
Variant ID	A unique identifier that indicates the device variant. The variant ID for the mXT449TD-AT is 0x39.
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.

1.6 Object Table

1.6.1 Introduction

The Object Table is held within the Information Block. The Object Table contains information on all the objects held withinthe memory map. It indicates which objects are present and their addresses.
Each element in the Object Table has the fields listed in Table 1-3.

TABLE 1-3: 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

1.6.2 Types

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 name, as given in the object descriptions. For example, the type code for the Command Processor T6(GEN_COMMANDPROCESSOR_**T6**) is **6**.

1.6.3 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 object. 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 Microchip device that uses this object-based protocol approach.

1.6.4 Size

Byte 3 of the Object Table element holds the size in bytes (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.

1.6.5 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 1-4). The different instances of an object are arranged sequentially in the memory map.

1.6.6 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 3.2 “Message Processor T5 Object”).
Report IDs are numbered uniquely and 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 Microchip. 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.

Figure 1-3 shows an example of how the number of instances and the Report IDs per instance determine the report IDs for a set of objects.

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:

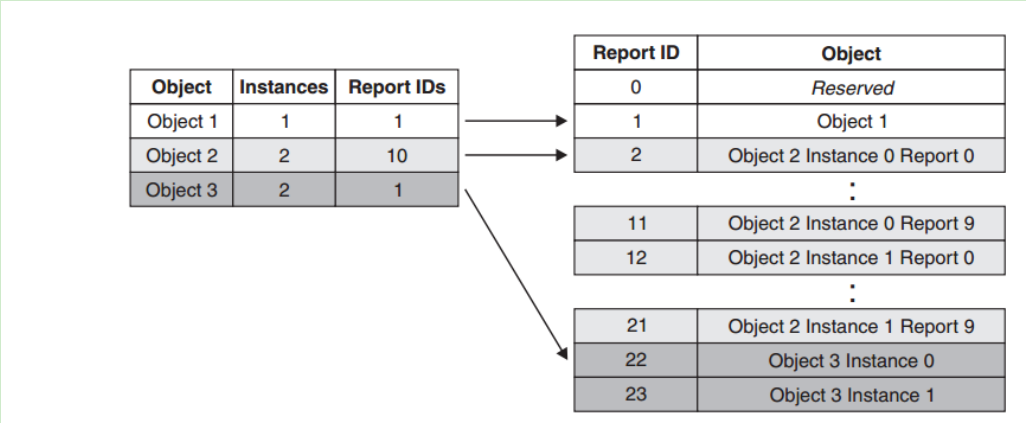
1. For each element in the Object Table:

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 a. and b. 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 Microchip).

FIGURE 1-3: EXAMPLE ASSIGNMENT OF REPORT IDS



The number of report IDs for each object is shown in Table 1-4.

1.7 Objects

1.7.1 Classes of Objects

The mXT449TD-AT contains the following classes of objects:

• **Debug objects** – provide a raw data output method for development and testing.
See Section 2.0 “Debug Objects”.

• **General objects** – required for global configuration, transmitting messages and receiving commands. See Section 3.0 “General Objects”.

• **Touch objects** – operate on measured signals from the touch sensor and report touch data. See Section 4.0 “Touch Objects”.

• **Signal processing objects** – process data from other objects (typically signal filtering operations). See Section 5.0 “Signal Processing Objects”.

• **Support objects** – provide additional functionality on the device. See Section 6.0 “Support Objects”.

1.7.2 Object Instances

TABLE 1-4: OBJECTS ON THE MCU WITH TOUCH MODULE

Object	Number of Instances	Report IDs per Instance	Reference
Debug Objects			
Diagnostic Debug T37	1	0	Section 2.2 “Diagnostic Debug T37 Object”
General Objects			
Message Processor T5	1	0	Section 3.2 “Message Processor T5 Object”
Command Processor T6	1	1	Section 3.3 “Command Processor T6 Object”

Power Configuration T7	1	0	Section 3.4 “Power Configuration T7 Object”
Acquisition Configuration T8	1	0	Section 3.5 “Acquisition Configuration T8 Object”
Touch Objects			
Key Array T15	1	1 or more	Section 4.2 “Key Array T15 Object”
Multiple Touch Touchscreen T9	1	1 or 2	Section 4.3 “Multiple Touch Touchscreen T9 Object”
Signal Processing Objects			
Noise Suppression T72	1	1	Section 5.2 “Noise Suppression T72 Object”
Supported Objects			
Communications Configuration T18	1	0	Section 6.2 “Communications Configuration T18 Object”
Self Test T25	1	1	Section 6.3 “Self Test T25 Object”
Message Count T44	1	0	Section 6.4 “Message Count T44 Object”
Auxiliary Touch Configuration T104	1	0	Section 6.5 “Auxiliary Touch Configuration T104 Object”

TABLE 1-4: OBJECTS ON THE MCU WITH TOUCH MODULE (Continued)

Object	Number of Instances	Report IDs per Instance	Reference
Self Capacitance Global Configuration T109	1	1	Section 6.6 “Self Capacitance Global Configuration T109 Object”
Self Capacitance Configuration T111	1	0	Section 6.7 “Self Capacitance Configuration T111 Object”
Low Power Idle Configuration T126	1	1	Section 6.8 “Low Power Idle Configuration T126 Object”

1.7.3 Configuration Values

The objects are designed such that a value of zero in their fields is a “safe” value. Typically a value of zero means that a default value is used, and this will be indicated within the object’s field descriptions in this protocol guide.
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 non-volatile memory using the Command Processor T6 object (see Section 3.3 “Command Processor T6 Object”).

The field descriptions in this protocol guide specify the recommended ranges for values. Any values outside the specified range for a field should be treated as reserved values and should not be used. Note that in many cases the device will not allow values outside the specified ranges.

1.7.4 Compatibility of Object Versions

The Object Protocol described in this document may contain 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 driver code is compatible with this object expansion scheme.

1.8 Configuration Checks

The device prevents invalid configurations by performing a sequence of checks. These checks are performed on powerup. 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 3.3.3 “Messages”). 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 non-volatile 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 the maximum X and Y size
• Field settings outside the permitted range
• A bad checksum for the configuration settings held in the non-volatile 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 reenabled and processing can continue as normal. See Appendix D “Locating Configuration Errors” for a flow chart showing this method for finding configuration errors.
Notice should be taken of any recommendations in this document concerning configuration checks for the offending object.

1.9 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).

1.10 Data Values

During measurement, the device uses raw 16-bit values for the deltas. These are converted by the Object-based Protocol to 8-bit deltas wherever possible in many of the controls and in the diagnostic debug data. This saves space and, in the case of debug data, allows for significant time savings in data transfer.

For Mutual Capacitance and Self Capacitance Touch deltas, the conversion between 16-bit deltas and 8-bit deltas is as follows:
16-bit delta = 8-bit delta × 8
8-bit delta = 16-bit delta / 8

1.11 Configuration and Tuning

This Protocol Guide provides reference information on the various configuration parameters for the MCU with touch module. For further guidance on configuring and tuning the MCU with touch module, refer to the *MCU with touch module Tuning Guide*.

1.12 Software Tools

The primary interface with the MCU with touch module is through the host driver and application code within the user’s product.
An alternative interface is also available by using maXTouch Studio. This provides a graphical IDE within which the configuration and tuning of a maXTouch device can take place so that the performance of the device can be assessed.
Note that maXTouch Studio Lite is supplied with the maXTouch evaluation kits.
Various plug-in tools are also available to help with configuring and tuning the various objects within the device and can be integrated into the maXTouch Studio development environment. Contact your Microchip representative for more information.

2.0 Debug Objects

2.1 Introduction

Debug objects contain raw data for development and testing purposes. Table 2-1 lists the debug objects on the MCU with touch module.

TABLE 2-1: DEBUG OBJECTS

Object	Description
Diagnostic Debug T37 (DEBUG_DIAGNOSTIC_T37)	Allows access to diagnostic debug data to aid development. See Section 2.2 “Diagnostic Debug T37 Object”.

2.2 Diagnostic Debug T37 Object

2.2.1 Introduction

The Diagnostic Debug T37 object holds the diagnostic debug data. It reports both system data and object-specific data from individual objects. See Section 2.2.1.1 “System Diagnostic Debug Modes” for more information on the system diagnostic debug modes. The object-specific diagnostic debug modes are described elsewhere in this document in the sections for the specific objects.

2.2.1.1 System Diagnostic Debug Modes

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 “DIAGNOSTIC Field”).
The mode in which the data is displayed is determined by the command written to the DIAGNOSTIC field of the Command Processor T6 object (see “DIAGNOSTIC Field”).
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.
The system diagnostic debug modes are shown in Table 2-2. The object-specific diagnostic debug modes are described within the individual object sections.

TABLE 2-2: SYSTEM DIAGNOSTIC DEBUG DATA MODES

T6 Command	Debug Mode	Description
0x01	Page Up	Command Processor T6 debug data page navigation
0x02	Page Down	Command Processor T6 debug data page navigation
0x10	Mutual Capacitance Deltas Mode	The Diagnostic Debug T37 object holds signal deltas. These are signed (two's complement) 16-bit mutual capacitance delta values for all nodes. Note that these are the raw deltas. The firmware divides these by 8 before use elsewhere (such as for threshold comparisons).
0x11	Mutual Capacitance References Mode	The Diagnostic Debug T37 object holds reference values. These are signed (two's complement) 16-bit mutual capacitance reference values for all nodes.
0x12	Mutual Capacitance Signals Mode	The Diagnostic Debug T37 object holds Signal values. These are signed (two's complement) 16-bit mutual capacitance reference values for all nodes.
0x17	Key Delta Mode	The Diagnostic Debug T37 object holds the key deltas. These are signed (two's complement) 16-bit key delta values for all nodes.
0x18	Key Reference Mode	The Diagnostic Debug T37 object holds the key reference values.
0x19	Key Signal Mode	The Diagnostic Debug T37 object holds the key compensation value (CC Value).
0x3B	Low power Mode	The Diagnostic Debug T37 object holds the low power mode data and status.
0x80	Device Information Mode	The Diagnostic Debug T37 object holds information on the device; specifically, the revision identifier.
0xF5	Self Capacitance Signal Mode	The Diagnostic Debug T37 object holds the compensation value (CC Value).
0xF7	Self Capacitance Delta Mode	The Diagnostic Debug T37 object holds the self capacitance deltas. These are signed (two's complement) 16-bit self capacitance delta values for all nodes.
0xF8	Self Capacitance Reference Mode	The Diagnostic Debug T37 object holds the self capacitance reference values.

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

2.2.1.2 Delta Mode (Mutual Capacitance)

Table 2-3 shows the organization of the data in the pages for the mutual capacitance delta mode.
The nodes are numbered along the X lines, that is in the order: element 0 = X0Y0, 1 = X0Y1, 2 = X0Y2 and so on. If there're only self capacitance button/ slider/ surface, then the data in this field will be the average of self capacitance data and the previous self capacitance data.

TABLE 2-3: DIAGNOSTIC DEBUG T37 FIELDS – DELTA MODE (MUTUAL CAPACITANCE)

Page Data	Index	Data[]
Page 0	0 – 31	Deltas for nodes 0 to 15
Page 1	32 – 63	Deltas for nodes 16 to 31
Page 2	64 – 97	Deltas for nodes 32 to 47

Assumes page size = 32

2.2.1.3 Reference Mode (Mutual Capacitance)

Table 2-4 shows the organization of the data in the pages for the mutual capacitance reference mode.
The nodes are numbered along the X lines, that is in the order: element 0 = X0Y0, 1 = X0Y1, 2 = X0Y2 and so on.
If there're only self capacitance button/ slider/ surface, then the data in this field will be the average of self capacitance data and the previous self capacitance data.

TABLE 2-4: DIAGNOSTIC DEBUG T37 FIELDS – REFERENCE MODE (MUTUAL CAPACITANCE)

Page Data	Index	Data[]
Page 0	0 – 31	References for nodes 0 to 15
Page 1	32 – 63	References for nodes 16 to 31
Page 2	64 – 97	References for nodes 32 to 47

Assumes page size = 32

2.2.1.3 Signal Mode (Mutual Capacitance)

Table 2-4 shows the organization of the data in the pages for the mutual capacitance Signal mode.
The nodes are numbered along the X lines, that is in the order: element 0 = X0Y0, 1 = X0Y1, 2 = X0Y2 and so on.
If there're only self capacitance button/ slider/ surface, then the data in this field will be the average of self capacitance data and the previous self capacitance data.

TABLE 2-4: DIAGNOSTIC DEBUG T37 FIELDS – SIGNAL MODE (MUTUAL CAPACITANCE)

Page Data	Index	Data[]
Page 0	0 – 31	Signals for nodes 0 to 15
Page 1	32 – 63	Signals for nodes 16 to 31
Page 2	64 – 97	Signals for nodes 32 to 47

Assumes page size =

2.2.1.4 Delta Mode (Self Capacitance)

Table 2-5 shows the organization of the data in the pages for the self capacitance delta mode.

TABLE 2-5: DIAGNOSTIC DEBUG T37 FIELDS – DELTA MODE (SELF CAPACITANCE)

Page Data	Index	Data[]
Page 0	0 – 31	Touch deltas for Y0 to Ymax, Touch deltas for X0 to Xmax
Page 1	32 – 63	
Page 2	64 – 97	

Assumes page size = 32

2.2.1.5 Reference Mode (Self Capacitance)

Table 2-6 shows the organization of the data in the pages for the self capacitance reference mode.

TABLE 2-6: DIAGNOSTIC DEBUG T37 FIELDS – REFERENCE MODE (SELF CAPACITANCE)

If Xmax is not more than Ymax, the organization is like this.

Page Data	Index	Data[]
Page 0	0 – 31	Touch Reference for Y0 to Ymax, Touch Reference for X0 to Xmax
Page 1	32 – 63	
Page 2	64 – 97	

If Xmax is less than Ymax, the organization is like this.

Page Data	Index	Data[]
Page 0	0 – 31	Touch Reference for Y0 to Ymax, Touch Reference for even X lines Touch Reference for odd X lines (start with Ymax * 2)
Page 1	32 – 63	
Page 2	64 – 97	

Assumes page size = 32

2.2.1.6 Signal Mode (Capacitance Compensation)

Table 2-7 shows the organization of the data in the pages for the capacitance compensation mode.

TABLE 2-7: DIAGNOSTIC DEBUG T37 FIELDS – Signal MODE (CAPACITANCE COMPENSATION VALUE)

If Xmax is not more than Ymax, the organization is like this.

Page Data	Index	Data[]
Page 0	0 – 31	Capacitance compensation value (CC Value) for Y0 to Ymax Capacitance compensation value (CC Value) for X0 to Xmax
Page 1	32 – 63	
Page 2	64 – 97	

If Xmax is less than Ymax, the organization is like this.

Page Data	Index	Data[]
Page 0	0 – 31	Capacitance compensation value (CC Value) for Y0 to Ymax Capacitance compensation value (CC Value) for even X lines Capacitance compensation value (CC Value) for odd X lines (start with Ymax * 2)
Page 1	32 – 63	
Page 2	64 – 97	

Assumes page size = 32

2.2.1.7 Device Information Mode

Page Data	Index	Data[]
Page 0	0 – 31	Device info for `Signature Row Summary`
Page 1	0 – 31	Fuse data

2.2.1.8 Key Delta Mode

Table 2-9 shows the organization of the data in the pages for the key delta mode. The bytes give the 16-bit deltas for the Key Array keys (16 in total). The keys are numbered in node order, as described in Section 4.2.1 “Introduction”.

TABLE 2-9: DIAGNOSTIC DEBUG T37 DATA – KEY DELTA MODE

Page Data	Index	Data[]
Page 0	0 – 31	Key deltas for each channel
Page 1	Reserved	Reserved
Page 2	Reserved	Reserved

Assumes page size = 32

2.2.1.9 Key Reference Mode

Table 2-10 shows the organization of the data in the pages for the key reference mode. The bytes give the 16-bit reference for the Key Array keys (64 in total). The keys are numbered in node order, as described in Section 4.2.1 “Introduction”.

TABLE 2-10: DIAGNOSTIC DEBUG T37 DATA – KEY DELTA MODE

Page Data	Index	Data[]
Page 0	0 – 31	Key Reference for each channel
Page 1	Reserved	Reserved
Page 2	Reserved	Reserved

Assumes page size = 32

2.2.1.10 Key Signal Mode (capacitance compensation)

Table 2-11 shows the organization of the data in the pages for the key signal mode (capacitance compensation). The bytes give the 16-bit data for the Key Array keys (64 in total). The keys are numbered in node order, as described in Section 4.2.1 “Introduction”.

TABLE 2-11: DIAGNOSTIC DEBUG T37 DATA – KEY DELTA MODE

Page Data	Index	Data[]
Page 0	0 – 31	Key Signal for each channel

Page 1	Reserved	
Page 2	Reserved	

Assumes page size = 32

2.2.1.11 Low power Mode

Table 2-12 shows the organization of the data in the pages for the low power mode. The data is less than 1 page.

TABLE 2-12: DIAGNOSTIC DEBUG T37 DATA – Low Power MODE

Page Data	Index	Data[]
Page 0	0 – 1	Low power idle delta
Page 0	2 – 3	Low power idle reference level
Page0	4 – 5	Raw low power idle signal
Page0	6	Status: 0 = device in idle mode; 1 = device in active mode
Page0	7 – 8	Raw (unfiltered) low power signal

2.2.2 Configuration

TABLE 2-12: CONFIGURATION FOR DIAGNOSTIC DEBUG T37 (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	Current page							
2 – (2+*n*-1)	DATA[n]	Current page of data							

Note: *n* = total number of bytes per page (32 on the MCU with touch module)

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 "DIAGNOSTIC Field").

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 "DIAGNOSTIC Field").

DATA [] Field

This field contains the current page of data.

3.0 General Objects

3.1 Introduction

General objects provide global configuration, such as receiving commands and transmitting messages. Table 3-1 lists the general objects on the MCU with touch module.

TABLE 3-1: 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 3.2 "Message Processor T5 Object".
Command Processor T6 (GEN_COMMANDPROCESSOR_T6)	Performs a command when written to. Commands include reset, calibrate and backup settings. See Section 3.3 "Command Processor T6 Object".
Power Configuration T7 (GEN_POWERCONFIG_T7)	Controls the sleep mode of the device. Power consumption can be lowered by controlling the acquisition frequency and the sleep time between acquisitions. See Section 3.4 "Power Configuration T7 Object".
Acquisition Configuration T8 (GEN_ACQUISITIONCONFIG_T8)	Controls how the device takes each capacitive measurement. See Section 3.5 "Acquisition Configuration T8 Object".

3.2 Message Processor T5 Object

3.2.1 Introduction

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 status of an object has changed. For this to happen, the object report enable bit must be set.

The system provides an interrupt style mechanism using the $\overline{\text{CHG}}$ line to indicate to the host that there is at least one message to read. This mechanism is available on both the I2C and the SPI interface. See Section 3.2.1.1 "Reading Messages with the CHG Line" for more details.

3.2.1.1 Reading Messages with the $\overline{CHG}$ Line

When using the I^{2}C or SPI interface, if a device has data to send, it asserts the $\overline{\text{CHG}}$ line (active low) to indicate to the host that there is a message (note that the $\overline{\text{CHG}}$ line is also asserted during SPI transactions to show that the slave is ready to send a response). 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 which has the potential for fast response times and reduces the need for wasteful 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 6.4 "Message Count T44 Object") 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 or the start of the Message Processor T5 object (that is, the report ID of the next available message).

NOTE

• 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.

• 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 unnecessary reads of the unused checksum byte.

3.2.2 Configuration

TABLE 3-2: CONFIGURATION FOR MESSAGE PROCESSOR T5 (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*	MESSAGE	Message data from source object							
n+1	CHECKSUM	Checksum							

Note: *n* = size of largest message possible

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 1.6.6 “Report IDs” 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 1.6 “Object Table”).
For information on the contents of the MESSAGE field, see the descriptions for each object elsewhere in this protocol guide.

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.
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 B “Checksum Calculation” for details on how to calculate the checksum.

3.3 Command Processor T6 Object

3.3.1 Introduction

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.

3.3.2 Configuration

TABLE 3-3: CONFIGURATION FOR COMMAND PROCESSOR T6 (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 non-zero value is written. If 0xA5 is written to this field, the device resets itself and enters into bootloader mode.
Write value: Non-zero (normal) or 0xA5 (bootloader)

BACKUPNV Field

This field supports the commands in Table 3-4:

TABLE 3-4: BACKUPNV COMMANDS

Command	Description
0x55	Backs up the settings to the Non-volatile Memory (NVM)
All other values	No effect

Once the device has processed this command it generates a status message containing the new NVM checksum.

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 3.4.1 “Introduction”), a message is generated when the device wakes from sleep.
This field is cleared once the command has been processed.
Write value: Non-zero

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.
If the device is in Deep Sleep mode (see Section 3.4.1 “Introduction”), the REPORTALL command will be processed once the device wakes from sleep.
This field is cleared once the command has been processed.
Write value: Non-zero

DIAGNOSTIC Field

This field allows commands to be written to control the use of the Diagnostic Debug T37 object (see Section 2.2.1 “Introduction”). Specifically, it allows the pages of diagnostic 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 3-5.

TABLE 3-5: DIAGNOSTIC DEBUG COMMANDS

Command	Name	Description
0x01	Page Up	Increment page number.
0x02	Page Down	Decrement page number.
Other Value	Debug data mode	The Diagnostic Debug T37 object holds the debug data for a specific debug mode. See Section 2.2 “Diagnostic Debug T37 Object” for details of the system debug modes and the commands to access them. See individual objects for details of the object-specific debug modes and the commands to access them.

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

3.3.3 Messages

The message data for the Command Processor T6 object is shown in Table 3-6.

TABLE 3-6: MESSAGE DATA FOR COMMAND PROCESSOR T6 (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 further 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 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 by CRC checking of the EEPROM. The message generated at bootup when config crc checking error, and will use default config for running.
See Section 1.7 "Configuration Checks" for more information on configuration checks.

NOTE The device will stop scanning for touches while the error persists.
It is possible to execute a backup command while the device is in this error state.

CAL: The device is calibrating. Note that CAL messages may not be generated if other objects (for example, Acquisition Configuration T8) disable these.
SIGERR: This is meaning the Fuse area is mismatch in packed firmware and current chip.
OFL: The acquisition and processing cycle length (see "IDLEACQINT and ACTVACQINT Fields" and "IDLEACQINT and ACTVACQINT Fields") has overflowed the desired power mode interval. The OFL flag is not updated in Free-run or Deep Sleep modes. This message can be suppressed by setting the OVFRPTSUP bit in the Power Configuration T7 CFG field (see Section 3.4.2 "Configuration").
RESET: The device has reset.

CHECKSUM Field

Reports the checksum of the configuration settings held in the non-volatile memory. See Appendix B "Checksum Calculation" for details on how to calculate the checksum.

3.4 Power Configuration T7 Object

3.4.1 Introduction

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

3.4.2 Configuration

TABLE 3-7: CONFIGURATION FOR POWER CONFIGURATION T7 (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 (the Auto scan trigger period)							
1	ACTVACQINT	Active acquisition interval (Time elapsed since update)							
2	ACTV2IDLETO	Active to idle time out (waiting time to switch to low power after the last touch)							

IDLEACQINT Field - QTM_AUTOSCAN_TRIGGER_PERIOD

This field defines the auto scan trigger period. The Low-power measurement period determine the interval between low-power touch measurement. For `Event system sleep` mode, the value is discrete and only valid with 4^(n) and less than 256, if the value more than 4^(n), it will be set as 4^(n+1). For `Software sleep` mode, this value is a continuous.

Range: NODE_SCAN_4MS(1) to NODE_SCAN_4096MS(11)

ACTVACQINT Field - DEF_TOUCH_MEASUREMENT_PERIOD_MS

The length of active burst cycles is determined by the Active Acquisition Interval (ACTVACQINT) fields. In the MCU this field is the time elapsed since update and the time is used to synchronize the internal time count used by the touch key module. When the internal time count reached the time elapsed since update, the time to measure touch flag is set and touch acquisition cycle will restart. ACTVACQINTFINE field defines the measurement time in milli seconds.

The Active Acquisition interval must not be set to less than the actual acquisition time. The device is also designed to sleep as much as possible in order to conserve power.

Range: 1 to 255
Typical: 20

ACTV2IDLETO Field - DEF_TOUCH_ACTIVE_IDLE_TIMEOUT

In the MCU this field is Waiting time (in millisecond) for the application to switch to low-power measurement after the last touch.

Range: 0 (never timeout), 1 to 255 (unit 200ms)

3.5 Acquisition Configuration T8 Object

3.5.1 Introduction

The Acquisition Configuration T8 object controls how the device takes capacitive measurements. In the MCU, this object is related with the acquisition module. The acquisition module implemented the features as blow:

- Hardware calibration for sensor nodes
  - Calibration of Prescaler/Resistor/Charge share delay to compensate for time constant of sensor electrodes
  - Calibration of internal compensation circuit to match sensor load
- Selfcap and mutual cap sensor touch measurement with normal sequencing
- Low-Power mode of automated scanning using Event System

3.5.2 Configuration

TABLE 3-8: CONFIGURATION FOR ACQUISITION CONFIGURATION T8 (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	ATCHDRIFT	Reserved (Antitouch drift rate)							
2	TCHDRIFT	Touch drift (Touch drift rate)							
3	DRIFTST	Drift suspend time (Drift hold time)							
4	TCHAUTOCAL	Touch automatic calibration (Max on duration time)							
5	SYNC	Reserved							
6	ATCHCALST	Antitouch calibration suspend time							
7	ATCHCALSTHR	Antitouch calibration suspend threshold (Antitouch recalibration suspend threshold)							
8	ATCHFRCCALTHR	Antitouch force calibration threshold							
9	ATCHFRCCALRATIO	Antitouch force calibration ratio							
10	MEASALLOW	Reserved		MUTUALTCH8P	MUTUALTCH4P	SELFPROX	Reserved	SELFTCH	MUTUALTCH
11	MEASTSTDEF	Reserved		Reserved	Reserved	SELFPROX	Reserved	SELFTCH	MUTUALTCH
12	MEASACTVDEF	Reserved		Reserved	Reserved	SELFPROX	Reserved	SELFTCH	MUTUALTCH
13	REFMODE	Reserved (Switch Reference/Signal Order in Firmware version less than v25)							
14	CFG	Reserved							

ATCHDRIFT Field - DEF_ANTI_TCH_DRIFT_RATE

More about drift refer to TCHDRIFT field. The unit of the drift rate is 200ms.

Range: 0(Disabled), 1 to 255, touch key group config
DELAYTIME Typical: 2u = 4 seconds

TCHDRIFT Field - DEF_TCH_DRIFT_RATE

Drift in a general sense means adjusting reference level (of a sensor) to allow compensation for temperature (or other factor) effect on physical sensor characteristics. Decreasing reference level for such compensation is called Negative drift & increasing reference level is called Positive drift. Specifically, the drift compensation should be set to compensate faster for increasing signals than for decreasing signals.

Signals can drift because of changes in physical sensor characteristics over time and temperature. It is crucial that such drift be compensated for; otherwise false detections and sensitivity shifts can occur.

Drift compensation occurs only while there is no detection in effect. Once a finger is sensed, the drift compensation mechanism ceases since the signal is legitimately detecting an object. Drift compensation works only when the signal inquestion has not crossed the 'Detect threshold' level.

The drift compensation mechanism can be asymmetric. It can be made to occur in one direction faster than it does in the other simply by changing the appropriate setup parameters.

Signal values of a sensor tend to increase when an object (touch) is approaching it or a characteristic change of sensor over time and temperature. Increasing signals should not be compensated quickly, as an approaching finger could be compensated for partially or entirely before even touching the channel (towards touch drift).

However, an object over the channel which does not cause detection, and for which the sensor has already made full allowance (over some period of time), could suddenly be removed leaving the sensor with an artificially suppressed reference level and thus become insensitive to touch. In the latter case, the sensor should compensate for the object's removal by raising the reference level relatively quickly (away from touch drift). The unit of the drift rate is 200ms.

Range: 0(Disabled), 1 to 255, touch key group config
DELAYTIME Typical: 2u = 4 seconds

DRIFTST Field - DEF_DRIFT_HOLD_TIME

Drift Hold Time (DHT) is used to restrict drift on all sensors while one or more sensors are activated. It defines the length of time the drift is halted after a key detection. This feature is useful in cases of high density keypads where touching a key or floating a finger over the keypad would cause untouched keys to drift, and therefore create a sensitivity shift, and ultimately inhibit any touch detection. The unit of the drift hold time is 200ms.

Range: 0(Disabled), 1 to 255, touch key group config
DELAYTIME Typical: 2u = 4 seconds

TCHAUTOCAL Field - DEF_MAX_ON_DURATION

If an object unintentionally contacts a sensor resulting in a touch detection for a prolonged interval it is usually desirable to recalibrate the sensor in order to restore its function, after a time delay of a few seconds.
The Maximum ON duration timer monitors such detections; if detection exceeds the timer's settings, the sensor is automatically recalibrated. After a recalibration has taken place, the affected sensor once again functions normally even if it still in contact with the foreign object.
Max-ON duration can be disabled by setting it to zero (infinite timeout) in which case the channel never recalibrates during a continuous detection (but the host could still command it).
The units of the max on duration is 200ms.

Range: 0(Disabled), 1 to 255, touch key group config
DELAYTIME Typical: 0

ATCHCALSTHR Field - DEF_ANTI_TCH_RECAL_THRSHLD

Recalibration threshold is the level beyond which automatic recalibration occurs. Recalibration threshold is expressed as a percentage of the detection threshold setting.
This setting is an enumerated value and its settings are as follows:
• Setting of 0 = 100% of detect threshold (RECAL_100)
• Setting of 1 = 50% of detect threshold (RECAL_50)
• Setting of 2 = 25% of detect threshold (RECAL_25)
• Setting of 3 = 12.5% of detect threshold (RECAL_12_5)
• Setting of 4 = 6.25% of detect threshold (RECAL_6_25)
• Setting of 5 = 0% of detect threshold (MAX_RECAL)
However, an absolute value of 4 is the hard limit for this setting. For example, if the detection threshold is, 40 and the Recalibration threshold value is set to 4. Although this implies an absolute value of 2 (40 * 6.25% = 2.5), it is hard limited to 4.

Range: RECAL_100 (0) to MAX_RECAL (5), touch key group config
DELAYTIME Typical: RECAL_100 (0)

MEASALLOW Field

This field defines which measurement types are allowed in the user's product. One or more types of touches can be set using the bits in this field. A value of 0 means 1; that is, mutual capacitance touches are always allowed.
MUTUALTCH: If this bit is set to 1, mutual capacitance touches are allowed.
SELFTCH: If this bit is set to 1, self capacitance touches are allowed.
HOVER: If this bit is set to 1, hover touches are allowed.
SELFPROX: If this bit is set to 1, self capacitance single-ended measurements are allowed.
MUTUALTCH4P: If this bit is set to 1, mutual capacitance parallel measurement with 4 X sensors are allowed.
MUTUALTCH8P: If this bit is set to 1, mutual capacitance parallel measurement with 8 X sensors are allowed.

Note: This is acquisition node group config and all of the measurement types are allowed as default.

MEASACTVDEF/MEASTSTDEF Field

Measurement definition after test and in test

This field defines the default measurement type that is to be used in active mode; that is, the default mode prescribes which data the Multiple Touch Touchscreen T100 processes during normal operation.
MUTUALTCH: If this bit is set to 1, the default measurement type is mutual capacitance in active mode.
SELFTCH: If this bit is set to 1, the default measurement type is self capacitance in active mode. Note that if the default measurement type is set to SELFTCH, then the Auxiliary Touch Configuration T104 object must be enabled for use. Otherwise a configuration error results.
HOVER: If this bit is set to 1, the default measurement type is hover in active mode.
SELFPROX: If this bit is set to 1, the default measurement type is self capacitance single-ended in active mode.
MUTUALTCH4P: If this bit is set to 1, the default measurement type is mutual capacitance parallel measurement with 4 X sensors in active mode.

MUTUALTCH8P: If this bit is set to 1, the default measurement type is mutual capacitance parallel measurement with 8 X sensors in active mode.

This Field is related with acquisition sensor type NODE_SELFCAP, NODE_SELFCAP_SHIELD, NODE_MUTUAL, NODE_MUTUAL_4P and NODE_MUTUAL_8P.

• NODE_SELFCAP: self capacitance sensor node, consisting one or more PTC Y pins. This type of sensor will set SELFPROX and SELFTCH to 1. • NODE_SELFCAP_SHIELD: Self capacitance sensor node with driven shield. Node consists one or more PTC Y pins, shield driven to one or more PTC X pins. This type of sensor will set SELFPROX to 1. • NODE_MUTUAL: Mutual capacitance sensor node. Node consists one or more PTC X Drive and one or more PTC Y Sense pins for measurement of XY capacitance. This type of sensor will set MUTUALTCH to 1. • NODE_MUTUAL_4P: Mutual capacitance parallel measurement with 4 X sensor node. This type of sensor will set MUTUALTCH4P to 1.

• NODE_MUTUAL_8P: Mutual capacitance parallel measurement with 8 X sensor node. This type of sensor will set MUTUALTCH8P to 1.

Note: This is acquisition node group config and the measurement type is selected according to the sensor node type automatically.

REFMODE Field

For Test purpose only, in platform early than v25, there has a Reference mode switch. When set this byte, the **DIAGNOSTIC Reference** will output Signal value, and **DIAGNOSTIC Reference** will output Reference value in T37 Object. This byte is not used in platform v25 or later.

Range: 0(Disabled), 1 enabled
DELAYTIME Typical: 0

4.0 Touch Objects

4.1 Introduction

Touch objects operate on measured signals from the touch sensor and report touch data. For example, a Multiple Touch Touchscreen T9 object reports XY touch positions. Table 4-1 lists the touch objects on the MCU with touch module.

TABLE 4-1: TOUCH OBJECTS

Object	Description
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 4.2 “Key Array T15 Object”.
Multiple Touch Touchscreen T9 (TOUCH_MULTITOUCHSCREEN_T9)	Creates a Touchscreen that supports the tracking of more than one touch. See Section 4.3 “Multiple Touch Touchscreen T9 Object”.

4.2 Key Array T15 Object

4.2.1 Introduction

The Key Array T15 object is used to configure an array of XY nodes on the mutual or self capacitance sensor for use as keys. There can be as many as instances as the number of the sensor groups. Even if it's slider, wheel or surface, it can be recognized as one instance of key array. Users can define the key array based on the actual sensors number and type.

4.2.2 Configuration

TABLE 4-2: CONFIGURATION FOR KEY ARRAY T15 (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		IGNGLOVEMODE	TSCONTSUP	Reserved	RPTEN	ENABLE
1	XORIGIN	X line start position of object							
2	YORIGIN	Y line start position of object (node origin)							
3	XSIZE	Number of X lines the object occupies							
4	YSIZE	Number of Y lines the object occupies (node size)							
5	AKSCFG	Group8	Group7	Group6	Group5	Group4	Group3	Group2	Group1
6	BLEN	GAIN							
		Analog gain				Digital gain			
7	TCHTHR	Touch threshold							
8	TCHDI	Touch detect integration							
9	TCHHYST	Touch hysteresis							
10	CFG	Reserved							

CTRL Field

ENABLE: Enables the use of this instance of the Key Array T15 object. The instance 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.
Only YORIGIN and YSIZE field is used to configure the sensor node.

YORIGIN Field

The YORIGIN field equals the node origin. All the node sensors are configured through Atmel Start. Users can reconfigure the node sensors to the node origin.

YSIZE Field

The YSIZE field equals the node size. It's the number of the touch key channels of one instance. All the node sensors are configured through Atmel Start. Users can reconfigure the node sensors to the node size.

AKSCFG Field - KEY_PARAMS_AKS_GROUP

It provides information about the sensors that belong to specific AKS group. NO_AKS_GROUP indicates that the sensor does not belong to any AKS group and cannot be suppressed. AKS_GROUP_x indicates that the sensor belongs to the AKS group x.
This setting is an enumerated value and its settings are as follows:

- setting of 0 = NO_AKS_GROUP
- setting of 1 = AKS_GROUP_1
- setting of 2 = AKS_GROUP_2
- setting of 3 = AKS_GROUP_3
- setting of 4 = AKS_GROUP_4
- setting of 5 = AKS_GROUP_5
- setting of 6 = AKS_GROUP_6
- setting of 7 = AKS_GROUP_7
- setting of 8 = MAX_AKS_GROUP (Max value of enum type for testing)

Range: NO_AKS_GROUP (0) to MAX_AKS_GROUP (8), touch key config
DELAYTIME Typical: RECAL_100 (0)

BLEN Field - NODE_PARAMS_GAIN

The BLEN filed contains analog gain setting (upper nibble) and digital gain setting (lower nibble). Analog gain is the integration capacitor adjusted to control integrator gain. Accumulated sum is scaled to Digital gain. The BLEN field is applied on all the channels to allow a scaling-up of the touch sensitivity on contact.
This setting is an enumerated value and its settings are as follows:

• setting of 0 = GAIN_1 (no scaling)
• setting of 1 = GAIN_2
• setting of 2 = GAIN_4
• setting of 3 = GAIN_8
• setting of 4 = GAIN_16
• setting of 5 = GAIN_32 (scale-up by 32)

Range: GAIN_1 (0) to GAIN_32(5), acquisition node config
DELAYTIME Typical: GAIN_1 (0)

TCHTHR Field - KEY_PARAMS_THRESHOLD

A sensor's detect threshold defines how much its signal must increase above its reference level to qualify as a potential touch detect. However, the final detection confirmation must satisfy the Detect Integrator (DI) limit. Larger threshold values desensitize sensors since the signal must change more (i.e. requires larger touch) to exceed the threshold level. Conversely, lower threshold levels make sensors more sensitive.
Threshold setting depends on the amount of signal swing when a sensor is touched. Usually, thicker front panels or smaller electrodes have smaller signal swing on touch, thus require lower threshold levels. Typically, detect threshold is set to 50% of touch delta.
The TCHTHR field is used to configure the threshold of all the configurated sensors.

Range: 0 to 255, touch key config
DELAYTIME Typical: 20-50

TCHDI Field - DEF_TOUCH_DET_INT / DEF_ANTI_TCH_DET_INT

The Touch Library features a detect integration mechanism, which confirm detection in a robust environment. The detect integrator (DI) acts as a simple signal filter to suppress false detections caused by spurious events such as electrical noise.
A counter is incremented each time the sensor delta has exceeded its threshold and stayed there for a specific number of acquisitions, without going below the threshold levels. When this counter reaches a preset limit (the DI value) the sensor is finally declared to be touched. If on any acquisition the delta is below the threshold level, the counter is cleared and the process has to start from the beginning. The DI process is applicable to a 'release' (going out of detect) event as well.
For example, if the DI value is 10, the device has to exceed its threshold and stay there for (10 + 2) successive acquisitions without going below the threshold level, before the sensor is declared to be touched. The TCHDI field defines the number in cycles.

Range: 0 to 255, touch key group config
DELAYTIME Typical: 4

TCHHYST Field - KEY_PARAMS_HYSTERESIS

This setting is sensor detection hysteresis value. It is expressed as a percentage of the sensor detection threshold setting. Once a sensor goes into detect its threshold level is reduced (by the hysteresis value) in order to avoid the sensor dither in and out of detect if the signal level is close to original threshold level.
• Setting of 0 = 50% of detect threshold value (HYST_50)
• Setting of 1 = 25% of detect threshold value (HYST_25)
• Setting of 2 = 12.5% of detect threshold value (HYST_12_5)
• Setting of 3 = 6.25% of detect threshold value (HYST_6_25)
• Setting of 4 = 3.125% of detect threshold value (MAX_HYST)
The XTCHHYST field is used to configure the hysteresis of all the configurated sensors.

Range: HYST_50 (0) to MAX_HYST (4), touch key config
DELAYTIME Typical: HYST_25 (1)

4.2.3 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 4-3.

TABLE 4-3: MESSAGE DATA FOR KEY ARRAY T15 (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 Field

Reports the state of each key, one bit per key; 0 = key is untouched, 1 = key is touched.

4.3 Multiple Touch Touchscreen T9 Object

4.3.1 Introduction

The Multiple Touch Touchscreen T9 object is used to configure a Multiple Touch Touchscreen on the sensor matrix. The MCU with touch module may have 2 instances of the Multiple Touch Touchscreen T9 object. Only slider and surface can be configured as a touchscreen instance.

4.3.2 Configuration

TABLE 4-4: CONFIGURATION FOR MULTIPLE TOUCH TOUCHSCREEN T9 (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	GROUP 2	GROUP1
6	BLEN	Gain							
		Analog gain				Digital gain			
7	TCHTHR	Touch threshold							
8	TCHDI	Touch detect integration for first touch (De-bounce counter for additional measurements to confirm touch detection)							
9	ORIENT	Reserved					INVERTY	INVERT X	SWITCHXY
10	MEGTIMEOUT	Merge timeout							
11	MOVHYSTI	Movement hysteresis, initial(distance threshold for initial move or direction change)							
12	MOVHYSTN	Movement hysteresis, next							
13	MOVFILTER	DISABLE	FILTERLIMIT (Enable Median Filter)			ADAPTTHR (IIR)			
14	NUMTCH	Number of reported touches							
15	MRGHYST	Merge hysteresis							
16	MRGTHR	Merge threshold							
17	AMPHYST	Amplitude hysteresis							
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	Jump limit		
31	TCHHYST	Touch threshold hysteresis		
32	XPITCH	X line pitch		
33	YPITCH	Y line pitch		
34	NEXTTCHDI	Touch detect integration for subsequent touches (De-bounce counter for additional measurements to confirm away from touch signal)		

CTRL Field

ENABLE: Enables the use of the 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.

XORIGIN Field

The XORIGIN field equals the X axis node origin. If it's slider, XORIGIN will be set to 0. If it's surface, XORIGIN will be set to be the start key of vertical axis.

YORIGIN Field

The YORIGIN field equals the Y axis node origin. If it's slider, YORIGIN will be set to the start key of slider. If it's surface, YORIGIN will be set to be the start key of horizontal axis.

XSIZE Field

The XSIZE field equals the X axis node size. If it's slider, XSIZE will be set to 0. If it's surface, XSIZE will be set to be the number of keys in vertical axis.

YSIZE Field

The YSIZE field equals the Y axis node size. If it's slider, YSIZE will be set to the number of keys in slider. If it's surface, YSIZE will be set to be the number of keys in horizontal axis.

BLEN Field - NODE_PARAMS_GAIN

The BLEN filed contains analog gain setting (upper nibble) and digital gain setting (lower nibble). Analog gain is the integration capacitor adjusted to control integrator gain. Accumulated sum is scaled to Digital gain. The BLEN field is applied on all the channels to allow a scaling-up of the touch sensitivity on contact.
This setting is an enumerated value and its settings are as follows:
• setting of 0 = GAIN_1 (no scaling)
• setting of 1 = GAIN_2
• setting of 2 = GAIN_4
• setting of 3 = GAIN_8
• setting of 4 = GAIN_16
• setting of 5 = GAIN_32 (scale-up by 32)

Range: GAIN_1 (0) to GAIN_32(5), acquisition node config
DELAYTIME Typical: GAIN_1 (0)

TCHTHR Field - KEY_PARAMS_THRESHOLD

A sensor's detect threshold defines how much its signal must increase above its reference level to qualify as a potential touch detect. However, the final detection confirmation must satisfy the Detect Integrator (DI) limit. Larger threshold values desensitize sensors since the signal must change more (i.e. requires larger touch) to exceed the threshold level. Conversely, lower threshold levels make sensors more sensitive.
Threshold setting depends on the amount of signal swing when a sensor is touched. Usually, thicker front panels or smaller electrodes have smaller signal swing on touch, thus require lower threshold levels. Typically, detect threshold is set to 50% of touch delta.
The TCHTHR field is used to configure the threshold of all the configurated sensors.

Range: 0 to 255, touch key config
DELAYTIME Typical: 20-50

TCHDI Field - DEF_TOUCH_DET_INT

The Touch Library features a detect integration mechanism, which confirm detection in a robust environment. The detect integrator (DI) acts as a simple signal filter to suppress false detections caused by spurious events such as electrical noise.
A counter is incremented each time the sensor delta has exceeded its threshold and stayed there for a specific number of acquisitions, without going below the threshold levels. When this counter reaches a preset limit (the DI value) the sensor is finally declared to be

touched. If on any acquisition the delta is below the threshold level, the counter is cleared and the process has to start from the beginning. The DI process is applicable to a 'release' (going out of detect) event as well.
For example, if the DI value is 10, the device has to exceed its threshold and stay there for (10 + 2) successive acquisitions without going below the threshold level, before the sensor is declared to be touched. The TCHDI field defines the number in cycles.

Range: 0 to 255, touch key group config
DELAYTIME Typical: 4

ORIENT Field

SWITCHXY: Switches the X and Y positions; that is, the screen is flipped about the diagonal from (X0, Y0) to (Xmax, Ymax).
INVERTY: Inverts Y coordinates; that is: Ynewval = (Ymax – Y).
INVERTX: Inverts X coordinates; that is: Xnewval = (Xmax – X).
Note that an INVERTX and/or INVERTY operation takes place before a SWITCHXY.

MOVHYSTI Field - SURFACE_CS_POS_HYST

The MOVHYSTI field defines the minimum travel distance to be reported after contact or direction change. It applies to horizontal and vertical. This field is 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 MOVHYSTI setting is applied to the reported position. Once the user's finger starts to move, the MOVHYSTI must be exceeded in a positive or negative X or Y direction (that is, in one of four directions) for position updates to be reported. This allows the host to differentiate between an intended drag and a simple press.

Range: 0 to 255, surface config
DELAYTIME Typical: 0

MOVFILTER Field - SURFACE_CS_FILT_CFG

This field controls the position filter that is used for filtering the reported touch position.
ADAPTTHR: IIR config, 0 = 0%, 1 = 25%, 2 = 50%, 3 = 75%.
FILTERLIMIT: Bit 4 = Enable Median Filter (3-point)

XRANGE/YRANGE Field

These two fields control the resolution, and thus the reported position, of the touchscreen. This two fields get the value from the upper nibble of resol_deadband in the surface config.

TCHHYST Field - KEY_PARAMS_HYSTERESIS

This setting is sensor detection hysteresis value. It is expressed as a percentage of the sensor detection threshold setting. Once a sensor goes into detect its threshold level is reduced (by the hysteresis value) in order to avoid the sensor dither in and out of detect if the signal level is close to original threshold level.
This setting is an enumerated value and its settings are as follows:
• Setting of 0 = 50% of detect threshold value (HYST_50)
• Setting of 1 = 25% of detect threshold value (HYST_25)
• Setting of 2 = 12.5% of detect threshold value (HYST_12_5)
• Setting of 3 = 6.25% of detect threshold value (HYST_6_25)
• Setting of 4 = 3.125% of detect threshold value (MAX_HYST)
The XTCHHYST field is used to configure the hysteresis of all the configurated sensors.

Range: HYST_50 (0) to MAX_HYST (4), touch key config
DELAYTIME Typical: HYST_25 (1)

NEXTTCHDI Field - DEF_ANTI_TCH_DET_INT

De-bounce counter for additional measurements to confirm away from touch signal to initiate Away from touch re-calibration. The NEXTTCHDI field defines the number in cycles.

Range: 0 to 255, touch key group config
DELAYTIME Typical: 4

4.3.3 Messages

Table 4-5 shows the message data for a Multiple Touch Touchscreen T9 object for when it is read via the Message Processor T5 object. This message is reported for a single touch on the touch pad.

TABLE 4-6: MESSAGE DATA FOR MULTIPLE TOUCH TOUCHSCREEN T9 (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

STATUS Field

Reports the current status of the touch. The touch is active if the DETECT bit is set. The MOVE, 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.

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, XYPOSLSB Fields

This 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 1 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 format for the X and Y positions are shown in Table 4-7 and Table 4-8.

TABLE 4-7. X Position Formats

XPOSMSB								XYPOSLSB							
Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
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 4-8. Y Position Formats

YPOSMSB								XYPOSLSB							
Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
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

5.0 Signal Processing Objects

5.1 Introduction

5.2 Noise Suppression T72 Object

5.2.1 Introduction

5.2.2 Configuration

5.2.3 Messages

6.0 Support Objects

6.1 Introduction

Support objects provide additional functionality on the device. Table 6-1 lists the support objects on the MCU with touch module.

TABLE 6-1: SUPPORT OBJECTS

Object	Description
Communications Configuration T18 (SPT_COMMSCONFIG_T18)	Configures additional communications behavior for the device. See Section 6.2 "Communications Configuration T18 Object".
Self Test T25 (SPT_SELFTEST_T25)	Configures and performs self-test routines to find faults on a touch sensor. See Section 6.3 "Self Test T25 Object".
Message Count T44 (SPT_MESSAGECOUNT_T44)	Provides a count of pending messages. See Section 6.4 "Message Count T44 Object".
Auxiliary Touch Configuration T104 (SPT_AUXTOUCHCONFIG_T104)	Allows the setting of self capacitance gain and thresholds for a particular measurement to generate auxiliary touch data for use by other objects. See Section 6.5 "Auxiliary Touch Configuration T104 Object".
Self Capacitance Global Configuration T109 (SPT_SELFCAPGLOBALCONFIG_T109)	Provides configuration for a self capacitance measurements employed on the device. See Section 6.6 "Self Capacitance Global Configuration T109 Object".
Self Capacitance Configuration T111 (SPT_SELFCAPCONFIG_T111)	Provides configuration for self capacitance measurements employed on the device. See Section 6.7 "Self Capacitance Configuration T111 Object".
Low Power Idle Configuration T126 (SPT_LOWPOWERIDLECONFIG_T126)	Configures the overall behavior of the low power self capacitance features. See Section 6.8 "Low Power Idle Configuration T126 Object".

6.2 Communication Configuration T18 Object

6.2.1 Introduction

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

6.2.2 Configuration

TABLE 6-2: CONFIGURATION FOR COMMUNICATIONS CONFIGURATION T18 (SPT_COMMSCONFIG_T18)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	DISMNTR	RETRIGEN	Reserved		HISPEED SPI	MODE	Reserved	
1	COMMAND	CHG line command code							

CTRL Field

RETRIGEN: Enables CHG line host retriggering mode. This provides extra edge-based interrupts to the host on the CHG line. If this mode is enabled, the CHG line is deasserted once every acquisition cycle for 50 µs and then re-asserted. This creates edges on the CHG line for the host in case it missed the CHG line being asserted. CHG line host retriggering mode is enabled if this bit is set to 1 and disabled if set to 0.

6.3 Self Test T25 Object

6.3.1 Introduction

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.

6.3.2 Configuration

TABLE 6-3: CONFIGURATION FOR SELF TEST T25 (SPT_SELFTEST_T25)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved		DISPERTSTDET	PERTSTMODE		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							
		...							
(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							
(4*n*+2)	PINDWELLUS	Pin fault test pin dwell time							
(4*n*+3) – (4*n*+4)	SIGRANGELIM[0]	Signal range limit LSByte for touch object 0							
		Signal range limit MSByte for touch object 0							
		...							
(6*n*+1) – (6*n*+2)	SIGRANGELIM[n–1]	Signal range limit LSByte for touch object *n*–1							
		Signal range limit MSByte for touch object *n*–1							
(6*n*+3)	PINTHR	Pin fault test threshold							

Note: *n* = number of touch objects, assigned in the following order:
All Multiple Touch Touchscreen T9 objects
All Key Array T15 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. A message is sent on completion of a test.

CMD Field

This field is used to send test commands to the device. Valid test commands are listed in Table 6-4.

TABLE 6-4: TEST COMMANDS

Command	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

0xFE	Run all the tests

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 0xFE to the CMD field causes the device to run all the tests in the order above.

UPSIGLIM[] and LOSIGLIM[] Fields Field

The UPSIGLIM and LOSIGLIM fields specify the higher and lower signal limits for the signal tests. UPSIGLIM must be greater than LOSIGLIM. The signal uses the compensation capacitor for limit.
When the test is run, the mutual capacitance signals must fall between LOSIGLIM and UPSIGLIM for the test to pass. Any signal values outside this range will generate a signal limit error in the test. The limits are specified on a per touch object basis.
The values chosen for UPSIGLIM and LOSIGLIM should differ depending on whether the unit is a design prototype or a production item.

For production use, LOSIGLIM and UPSIGLIM should in the range of the MCU. The exact values, however, are determined by analyzing the data from actual sensor samples. Using the full range can cause sensors that have too much signal to pass the test.
Range: See Table 6-4

TABLE 6-4: RECOMMENDED VALUE FOR UPSIGLIM[] AND LOSIGLIM[]

Touch Object	Range		Recommended Design Values		Recommended Production Values	
	LOSIGLIM	UPSIGLIM	LOSIGLIM	UPSIGLIM	LOSIGLIM	UPSIGLIM
Key Array T15	0	-	-	-	Determined on a case-by-case basis	
Multiple Touch Touchscreen T9	0	-	-	-	Determined on a case-by-case basis	

Note: UPSIGLIM must be ≥ LOSIGLIM, the unit is pf / 1000

PINDWELLUS Field

This field configures the pin dwell time (in µs) for the pin fault test. PINDWELLUS is also used by the initial pin fault test on power-up, irrespective of whether the Self Test T25 object is enabled or not. The value of this field should be set so as to allow enough time for the X lines to slew to the correct level before testing. The PINDWELLUS field should always be set to correct value for each design, regardless of whether Self Test T25 feature is used or not by the application.

Range: 0 (200), 1 to 254

SIGRANGELIM[] Field

This field specifies the signal range limit for the signal tests. When the test is run, the derivation with the standard reference base(512) * Digital Gain, with be checked by SIGRANGELIM setting here
Range: 1 to 13500

PINTHR Field

This field specifies the Pin Thresholds(ADC count) used to detect shorts in the pin fault tests. Increasing the value of this threshold increases the resistance that will be detected as a short.

Range: 0 (225 = 200 kΩ), 1 to 255

6.3.3 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 6-5.

TABLE 6-5: MESSAGE DATA FOR SELF TEST T25 (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 – 7	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 6-6.

TABLE 6-6: 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.
0x01	AVdd is not present. This failure is reported to the host each time checked. Note that if the POR initial AVDD test fails, then the Self Test

	T25 object will generate a message with this result code on reset then stop to scanning until recovered. For 3.3v system, the default AVDD checking range is 2.95v to 3.84v: If we use 1.5v VREF as measure target(Vmeasured), and VDD is set as Vref. **Vref = Vmeasured(1.5) *256 /adcval** Note: Valid adcval value is between (100, 130)
0x12	The test failed because of a pin fault. The INFO fields indicate the first pin fault that was detected (see Table 6-7). Note that if the POR initial pin fault test fails, then the Self Test T25 object will generate a message with this result code on reset then stop to scanning until recovered
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 – Analog Power Fault

Current 8bit ADC value measure of AVDD.

NOTE If the Analog Power Test fails, it will be necessary to perform a power cycle on the product otherwise the device may still report a pin fault issue.

INFO Field – Pin Fault

If the result was a pin fault, the INFO field has the data format shown in Table 6-7.

TABLE 6-7: 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	PIN	Failing pin indication							
4	VAL	Failing pin ADC value Measured							

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 6-8.

TABLE 6-8: SEQUENCE NUMBERS FOR PIN FAULT TEST

Code	Test Result
0x01	**Driven Ground** This test detects shorts to a power rail (that is, they fail high). All pins are driven low to Ground. Note that the detection of X/Y shorts is dependent on the supply voltage levels.
0x02	**Driven High** This test detects shorts to Ground (that is, they fail low). This test can detect resistive shorts, but only up to ~200 k for X/Y sense pins or ~30.k for the driven shield pin. All pins are pulled high.
0x03	**Walking 1** This test is similar to test sequence 01, but uses the pull-up resistors to detect resistive shorts (but only up to ~200 k for X/Y sense pins or ~30.k for the driven shield pin). 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.
0x04	**Walking 0** This test is similar to test sequence 02, but uses the pull-up resistors to detect resistive shorts (but only up to ~200 k for X/Y sense pins or ~30.k for the driven shield pin). 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.

0x07	**Initial High Voltage** This test detects an initial high voltage pin fault. Specifically, it detects shorts between high voltage drive lines and GND, X or Y lines that would damage the device if a high voltage acquisition were performed. All pins are driven low and then the high voltage drive lines are all pulled high one after another.

PIN Fields

This is the Physical Sensor channel defined by Touch.h. These fields indicate which pins have failed. A non zero value in X_PIN indicates the X line number plus 1 of a failed X sense pin. A non zero value in Y_PIN indicates the Y line number plus 1 of a failed Y sense pin. For example, if X5 has failed, X_PIN will read 6. Similarly, if Y3 has failed, Y_PIN will read 4.

If both fields are zero, the driven shield has failed.

If more than one pin fails, the test reports the first failed pin detected.

See Table 6-9 for a summary.

TABLE 6-9: PIN FAILURE RESULTS

PIN Value	Pin Failure
Physical Pin used	A sense pin has failed. PIN will be set to the number of the pin plus 1. For example, if XY3 has failed, XY_PIN will read 4.

INFO Field – Signal Limit Error

If the result was a signal limit error, the INFO field has the data format shown in Table 6-10.

TABLE 6-10: 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							
4	CHANNEL	Touch channel with signal limit error							
5-6	CAPICITANCE	Touch object 16bit capacitance value (LSB)							
7-8	REFERENCE	Touch object 16bit reference value (LSB)							

TYPE_NUM Field

The type number of the touch object for which a signal limit error was found (see Section 1.6 “Object Table”).

TYPE_INSTANCE Field

The instance number of the touch object for which a signal limit error was found.

CHANNEL Field

The channel of the touch object for which a signal limit error was found. This is the Physical Sensor channel defined by Touch.h.

CAPICITANCE Field

The channel capacitance of the touch object for which a signal limit error was found.

REFERENCE Field

The channel reference of the touch object for which a signal limit error was found.

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.

6.4 Message Count T44 Object

6.4.1 Introduction

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 3.2.1 “Introduction” for more information.

6.4.2 Configuration

TABLE 6-6: CONFIGURATION FOR MESSAGE COUNT T44 (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.

6.5 Auxiliary Touch Configuration T104 Object

6.5.1 Introduction

An Auxiliary Touch Configuration T104 object allows the setting of gain and thresholds for self capacitance measurements to generate auxiliary touch data for use by other objects. This object is used to set the whole self capacitance key module configuration in MCU with touch module.

6.5.2 Configuration

TABLE 6-7: CONFIGURATION FOR AUXILIARY TOUCH CONFIGURATION T104 (SPT_AUXTOUCHCONFIG_T104)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	Reserved							ENABLE
1	XGAIN	X axis Analog gain							
		X axis Analog gain				X axis Digital gain			
2	XTCHTHR	X axis touch threshold							
3	XTCHHYST	X axis touch hysteresis							
4	XINTTHR	X axis internal threshold							
5	XINTHYST	X axis internal hysteresis							
6	YGAIN	Y axis gain							
		Y axis Analog gain				Y axis Digital gain			
7	YTCHTHR	Y axis touch threshold							
8	YTCHHYST	Y axis touch hysteresis							
9	YINTTHR	Y axis internal threshold							
10	YINTHYST	Y axis internal hysteresis							

XGAIN Field - NODE_PARAMS_GAIN

The GAIN filed contains analog gain setting (upper nibble) and digital gain setting (lower nibble). Analog gain is the integration capacitor adjusted to control integrator gain. Accumulated sum is scaled to Digital gain. The XGAIN field is applied on all the X axis channels to allow a scaling-up of the touch sensitivity on contact.
This setting is an enumerated value and its settings are as follows:

- setting of 0 = GAIN_1 (no scaling)
- setting of 1 = GAIN_2
- setting of 2 = GAIN_4
- setting of 3 = GAIN_8
- setting of 4 = GAIN_16
- setting of 5 = GAIN_32 (scale-up by 32)

Range: GAIN_1 (0) to GAIN_32(5), acquisition node config
DELAYTIME Typical: GAIN_1 (0)

XTCHTHR Field - KEY_PARAMS_THRESHOLD

A sensor's detect threshold defines how much its signal must increase above its reference level to qualify as a potential touch detect. However, the final detection confirmation must satisfy the Detect Integrator (DI) limit. Larger threshold values desensitize sensors since the signal must change more (i.e. requires larger touch) to exceed the threshold level. Conversely, lower threshold levels make sensors more sensitive.
Threshold setting depends on the amount of signal swing when a sensor is touched. Usually, thicker front panels or smaller electrodes have smaller signal swing on touch, thus require lower threshold levels. Typically, detect threshold is set to 50% of touch delta. Desired touch delta for a button is ~30 to 80 counts and for wheels or sliders is ~50 to 120 counts.
The XTCHTHR field is used to configure the threshold of all the X axis sensors.

Range: 0 to 255, touch key config
DELAYTIME Typical: 20-50 (For buttons), 30-80 (For sliders and wheels)

XTCHHYST Field - KEY_PARAMS_HYSTERESIS

This setting is sensor detection hysteresis value. It is expressed as a percentage of the sensor detection threshold setting. Once a sensor goes into detect its threshold level is reduced (by the hysteresis value) in order to avoid the sensor dither in and out of detect if the signal level is close to original threshold level.

- Setting of 0 = 50% of detect threshold value (HYST_50)
- Setting of 1 = 25% of detect threshold value (HYST_25)
- Setting of 2 = 12.5% of detect threshold value (HYST_12_5)
- Setting of 3 = 6.25% of detect threshold value (HYST_6_25)
- Setting of 4 = 3.125% of detect threshold value (MAX_HYST)

The XTCHHYST field is used to configure the hysteresis of all the X axis sensors.

Range: HYST_50 (0) to MAX_HYST (4), touch key config
DELAYTIME Typical: HYST_25 (1)

YGAIN Field - NODE_PARAMS_GAIN

The GAIN filed contains analog gain setting (upper nibble) and digital gain setting (lower nibble). Analog gain is the integration capacitor adjusted to control integrator gain. Accumulated sum is scaled to Digital gain.
The YGAIN field is applied on all the Y axis channels to allow a scaling-up of the touch sensitivity on contact.
This setting is an enumerated value and its settings are as follows:

- setting of 0 = GAIN_1 (no scaling)
- setting of 1 = GAIN_2
- setting of 2 = GAIN_4
- setting of 3 = GAIN_8
- setting of 4 = GAIN_16
- setting of 5 = GAIN_32 (scale-up by 32)

Range: GAIN_1 (0) to GAIN_32(5), touch key config
DELAYTIME Typical: GAIN_1 (0)

YTCHTHR Field - KEY_PARAMS_THRESHOLD

The YTCHTHR field is used to configure the threshold of all the Y axis sensors. More about threshold refer to XTCHTHR Filed.

Range: 0 to 255, touch key config
DELAYTIME Typical: 20-50 (For buttons), 30-80 (For sliders and wheels)

YTCHHYST Field - KEY_PARAMS_HYSTERESIS

The YTCHHYST field is used to configure the hysteresis of all the Y axis sensors. More about hysteresis refer to XTCHHYST Filed.

Range: HYST_50 (0) to MAX_HYST (4), touch key config
DELAYTIME Typical: HYST_25 (1)

6.6 Self Capacitance Global Configuration T109 Object

6.6.1 Introduction

The Self Capacitance Global Configuration T109 object provides configuration for self capacitance measurements employed on the device.

6.6.2 Configuration

TABLE 6-8: CONFIGURATION FOR SELF CAPACITANCE GLOBAL CONFIGURATION T109 (SPT_SELFCAPGLOBALCONFIG_T109)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	DISMEASY	DISMEASX	Reserved		DISPRECHARGE	SNGLENDEN	RPTEN	Reserved
1	Reserved	Reserved							
2	CMDONRESET	Command performed on reset							
3	CMD	Command							
4	LFCOMPSFX	Low frequency compensation scaling factor for X							
5		Low frequency compensation scaling factor for X							
6	LFCOMPSFY	Low frequency compensation scaling factor for Y							
7		Low frequency compensation scaling factor for Y							
8	TUNECFG	Reserved				EXTRATUNITER			

CTRL Field

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.
SNGLENDEN: Enables Single Ended Measurement.

CMD Field

This field schedules the given command.
The command field is cleared after the command has been actioned.
Range: See Table 6-9

TABLE 6-9: COMMANDS

Value	Command	Description
0	None	None.
1	Tune	Send a calibration command to the system and clear all the status.

6.6.3 Messages

TABLE 6-10: MESSAGE DATA FOR SELF CAPACITANCE GLOBAL CONFIGURATION T109 (SPT_SELFCAPGLOBALCONFIG_T109)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	CMD	Command processed							
2	ERRORCODE	Error code							

CMD Field

This field reports the command that was processed and has caused this message (see "CMD Field").
0x01 – Tune command had been excuted.

ERRORCODE Field

This field reports the success/error code. An error code of 0 means that there is no error (success). Otherwise, a non-zero indicates the error resulting from the command that caused this message. There's no error code now.

6.7 Self Capacitance Configuration T111 Object

6.7.1 Introduction

The Self Capacitance Configuration T111 object provides configuration for self capacitance measurements.

6.7.2 Configuration

TABLE 6-11: CONFIGURATION FOR SELF CAPACITANCE CONFIGURATION T111 (SPT_SELFCAPCONFIG_T111)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	DISDCCORR	Reserved						
1	DBGCTRL	Reserved			DCEN	RAWDATAEN	SIGSEN	REFSEN	DELTASEN
2	INTTIME	Integration Time							
		Reserved				Prescaler			
3	DELAYTIME	Delay time (CSD, Charge share delay)							
4	IDLESYNCSPERL	Number of ADCs per sensor line when idle							
5	ACTVSYNCSPERL	Number of ADCs per sensor line when active (Samples to Accumulate)							
6	DRIFT	Drift interval (Drift rate)							
7	DRIFTST	Drift suspend time (Drift hold time)							
8	Reserved	Reserved							
9	CALRECSTR	Calibration recovery strength							
10	Reserved	Reserved							
11	Reserved	Reserved							
12	Reserved	Reserved							
13	Reserved	Reserved							
14	Reserved	Reserved							
15	INRUSHCFG	Reserved	XRESISTOR			Reserved	YRESISTOR		
16	ALTINTTIMEX	Alternative X axis measurement Integration Time							
		Reserved				Alternative X axis Prescaler			
17	ALTDELAYTIMEX	Alternative X axis measurement delay time (Alternative X axis CSD, Charge share delay)							
18	DCDRIFT	DC drift interval							
19	Reserved	Reserved							
20	DCFILTER	Reserved		SLEWACC	SLEWEN	IIRCOEF			IIREN
21	DCCALRECSTR	DC Calibration recovery strength							
22	DCCALERRRATIO	DC Calibration recovery error ratio							
23	SYNCDELAY	Delay from PSYNC falling edge LSByte							
24		Delay from PSYNC falling edge MSByte							
25	ADCSPERSYNC	Reserved		ADCSPERSYNC					
26	DCGAINSF	DC gain scaling factor							
27	DCTHRX	X axis DC Self capacitance threshold							
28	DCTHRY	Y axis DC Self capacitance threshold							
29	DCIDLESLEWMIN	Idle Mode Minimum DC Slew Rate							

INTTIME Field - NODE_PARAMS_RESISTOR_PRESCALER

The prescaler parameter denotes the clock divider for the particular channel. It can be set on per channel basis and is independent to each sensor node/channel. This parameter is auto tuned based on the auto tune settings. Tuning this parameter allows for improved noise performance.

INTTIME is used to configure the prescaler setting in the Y direction, and if ALTINITIMEX is not configured, INITIME is also used to configure the Prescaler setting in the X direction.
This setting is an enumerated value and its settings are as follows: (For example, if Generic clock input to PTC = 4MHz)
• setting of 0 = PRSC_DIV_SEL_1 (For example, if Generic clock input to PTC = 4MHz)
• setting of 1 = PRSC_DIV_SEL_2 (For example, PRSC_DIV_SEL_2 sets PTC Clock to 2MHz)
• setting of 2 = PRSC_DIV_SEL_4 (For example, PRSC_DIV_SEL_4 sets PTC Clock to 1MHz)
• setting of 3 = PRSC_DIV_SEL_8 (For example, PRSC_DIV_SEL_8 sets PTC Clock to 500KHz)
• setting of 4 = PRSC_DIV_SEL_16
• setting of 5 = PRSC_DIV_SEL_32
• setting of 6 = PRSC_DIV_SEL_64
• setting of 7 = PRSC_DIV_SEL_128
Only the lower nibble are used to configure the Prescaler parameter.

Range: PRSC_DIV_SEL_1 (0) to PRSC_DIV_SEL_128 (7), acquisition node config
DELAYTIME Typical: PRSC_DIV_SEL_1 (0)

DELAYTIME Field - NODE_PARAMS_CSD

Charge Share Delay
Charge share delay indicates the number of additional charge cycles that are inserted within a capacitance measurement cycle to ensure that the touch sensor is fully charged. The CSD value is dependent on the sensor capacitance along with the series resistor on the Y line.
Note: Any increase in the charge share delay also increases the measurement time for a specific configuration.
When manual tuning is performed, the CSD value for the sensor with largest combination of capacitance along with series resistance should be considered.

How to tune the CSD setting manually?
1. Initially, use an arbitrarily large value such as 64 and note the signal value. A large value ensures that the charge time is enough for full charge transfer.
2. Reduce the CSD and verify the signal value drop, until signal is approximately 97-98% of the value used initially. This ensures a good charge transfer without any major loss in the signal.
3. Continue the same procedure [Step 1 and 2] for all the sensors available in the system. Use the largest value of the CSD used in the system for the global setting.
Note: For the same CSD setting, Mutual capacitance has a lower burst time than self-capacitance. A unit increase in mutual capacitance CSD consumes around 12 PTC cycles. Whereas for the self capacitance an increase in CSD consumes approximately twice the mutual capacitance CSD time with the same setting.

DELAYTIME is used to configure the CSD setting in the Y direction, and if ALTDELAYTIMEX is not configured, DELAYTIME is also used to configure the CSD setting in the X direction. The CSD setting defined the charge share delay in PTC cycles.

Range: 0 to 250, acquisition node config
DELAYTIME Typical: 0

ACTVSYNCSPERL Field - NODE_PARAMS_ADC_OVERSAMPLING

This filed defines the number of samples to accumulate for each measurement. Oversampling must be configured to be greater than or equal to digital gain for correct operation. The oversampling setting controls the number of samples acquired to resolve each acquisition. A higher oversampling setting provides improved signal to noise ratio under noisy conditions, while increasing the total time
for measurement which results in increased power consumption.
This setting is an enumerated value and its settings are as follows:
• setting of 0 = FILTER_LEVEL_1
• setting of 1 = FILTER_LEVEL_2
• setting of 2 = FILTER_LEVEL_4
• setting of 3= FILTER_LEVEL_8
• setting of 4 = FILTER_LEVEL_16
• setting of 5 = FILTER_LEVEL_32

Range: FILTER_LEVEL_1 (0) to FILTER_LEVEL_64 (6), acquisition node config
DELAYTIME Typical: FILTER_LEVEL_16 (4)

DRIFT Field - DEF_TCH_DRIFT_RATE

Drift in a general sense means adjusting reference level (of a sensor) to allow compensation for temperature (or other factor) effect on physical sensor characteristics. Decreasing reference level for such compensation is called Negative drift & increasing reference level is called Positive drift. Specifically, the drift compensation should be set to compensate faster for increasing signals than for decreasing signals.

Signals can drift because of changes in physical sensor characteristics over time and temperature. It is crucial that such drift be compensated for; otherwise false detections and sensitivity shifts can occur.

Drift compensation occurs only while there is no detection in effect. Once a finger is sensed, the drift compensation mechanism ceases since the signal is legitimately detecting an object. Drift compensation works only when the signal inquestion has not crossed the 'Detect threshold' level.

The drift compensation mechanism can be asymmetric. It can be made to occur in one direction faster than it does in the other simply by changing the appropriate setup parameters.

Signal values of a sensor tend to increase when an object (touch) is approaching it or a characteristic change of sensor over time and temperature. Increasing signals should not be compensated quickly, as an approaching finger could be compensated for partially or entirely before even touching the channel (towards touch drift).

However, an object over the channel which does not cause detection, and for which the sensor has already made full allowance (over some period of time), could suddenly be removed leaving the sensor with an artificially suppressed reference level and thus become insensitive to touch. In the latter case, the sensor should compensate for the object's removal by raising the reference level relatively quickly (away from touch drift). The unit of the drift rate is 200ms.

Range: 0 to 255, key group config
DELAYTIME Typical: 2u = 4 seconds

DRIFTST Field - DEF_DRIFT_HOLD_TIME

Drift Hold Time (DHT) is used to restrict drift on all sensors while one or more sensors are activated. It defines the length of time the drift is halted after a key detection.This feature is useful in cases of high density keypads where touching a key or floating a finger over the keypad would cause untouched keys to drift, and therefore create a sensitivity shift, and ultimately inhibit any touch detection. The unit of the drift hold time is 200ms.

Range: 0 to 255, key group config
DELAYTIME Typical: 2u = 4 seconds

INRUSHCFG Field - NODE_PARAMS_RESISTOR_PRESCALER

The series resistor denotes the resistor used on the particular channel for the acquisition. The value is tunable and allows both auto and manual tuning options. Tuning this parameter allows for improved noise performance. For Mutual cap mode, this series resistor is switched internally on the Y-pin. For Self cap mode, the series resistor is switched internally on the Sensor pin.

The least 3 bits is used to configure the series resistor setting in the X direction, and if the bit 4-6 is set to zero, the least 3 bits is also used to configure the series resistor setting in the X direction, otherwise, the bit 4-6 is used to configure the series resistor setting in the X direction.

This setting is an enumerated value and its settings are as follows:

- setting of 0 = RSEL_VAL_0 (For example, RSEL_VAL_0 sets internal series resistor to 0ohms)
- setting of 1 = RSEL_VAL_20 (For example, RSEL_VAL_20 sets internal series resistor to 20Kohms)
- setting of 2 = RSEL_VAL_50 (For example, RSEL_VAL_50 sets internal series resistor to 50Kohms)
- setting of 3 = RSEL_VAL_70 (For example, RSEL_VAL_70 sets internal series resistor to 70Kohms)
- setting of 4 = RSEL_VAL_100 (For example, RSEL_VAL_100 sets internal series resistor to 100Kohms)

Range: RSEL_VAL_0 (0) to RSEL_VAL_100 (4), acquisition node config
DELAYTIME Typical: RSEL_VAL_100 (3)

ALTINTTIMEX Field - NODE_PARAMS_RESISTOR_PRESCALER

ALTINTTIMEX is used to configure the Prescaler setting in the X direction, and if ALTINTTIMEX is set to zero, INTTIME is used to configure the Prescaler setting in the X direction. The Prescaler setting defined the charge share delay in PTC cycles. More about Prescaler setting refer to INITIME filed.

Range: PRSC_DIV_SEL_1 (0) to PRSC_DIV_SEL_128 (7), acquisition node config
DELAYTIME Typical: PRSC_DIV_SEL_1 (0)

ALTDELAYTIMEX Field - NODE_PARAMS_CSD

ALTDELAYTIMEX is used to configure the CSD setting in the X direction, and if ALTDELAYTIMEX is set to zero, DELAYTIME is used to configure the CSD setting in the X direction. The CSD setting defined the charge share delay in PTC cycles. More about CSD setting refer to DELAYTIME filed.

Range: 0 to 250, acquisition node config
DELAYTIME Typical: 0

6.8 Low Power Idle Configuration T126 Object

6.8.1 Introduction

The Low Power Idle Configuration T126 object configures the overall behavior of the Low Power Idle mode features.

6.8.2 Configuration

TABLE 6-12: CONFIGURATION FOR LOW POWER IDLE CONFIGURATION T126 (SPT_LOWPOWERIDLECONFIG_T126)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	CTRL	DBGEN	Reserved		AUTOEN	RPTTCHEN	RPTAUTOEN	RPTEN	ENABLE
1	NODE	Node selection for Low-power scan							
2	GAIN	Measurement Gain							
3	THRESHOLD	Touch Threshold							
4	SYNCSPERL	Number of Sync Groups per Line							
5	DRIFTCOEF	Reserved			DRIFTCOEF (drift measurement period)				
6	SIGFILTCOEF	Reserved			SIGFILTCOEF				
7	TUNPARAM0	Fine and Accuracy Compensation value							
8	TUNPARAM1	Rough and Coarse Compensation value							
9	CSD	The charging share delay							
10	RESPRES	Node Resistor and Prescale selection							
11	DRIFTNODE	Target node for drifting							

CTRL Field

ENABLE: Enables the use of this Low Power Idle Configuration T126 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.

RPTAUTOEN: Enable T15 message for Non-Low power button in diel mode.

RPTTCHEN: Enables T15 message reporting for Low power button in idle mode.

AUTOEN: Enables T15 message reporting for Low Power Button in active mode.

NODE Field - QTM_AUTOSCAN_NODE/ DEF_LOWPOWER_KEYS

This field sets node selection for low-power scan. When the system is in low power mode, only the selected node will be scanned. (For Event system sleep, this is always ZERO; for software sleep, this is node channels mask, which supports multi channels wakeup)

Range: 0 to (DEF_NUM_CHANNELS-1)

THRESHOLD Field - QTM_AUTOSCAN_THRESHOLD

This field sets the detect threshold (in delta values) to use for the low power idle measurements.

Range: 10 to 255

DRIFTCOEF Field - DEF_TOUCH_DRIFT_PERIOD_MS

DRIFTCOEF: Specifies the Drift measurement period. During low-power measurement, it is recommended to perform periodic active measurement to perform drifting. This parameter defines the measurement interval to perform drifting.

Range: 0(never drift), 1 to 255 (unit 200ms), the maximum value will be limited by WDTDOG setting when using event system

TUNPARAM0/TUNPARAM0 field

Specifies the compensation value for the low power node. The compensation value is made up by 14bit value:

Accuracy: bit[3:0], give 0.00675pf each count

Fine: bit[7:4], give 0.0675pf each count

Coase: bit[8:11], give 0.675pf each count

Rought: bit[13:12], give 6.75pf each count

```
cc value formula:
        (val & 0x0F)*0.00675 + ((val >> 4) & 0x0F)*0.0675 + ((val >> 8) & 0x0F)*0.675 + ((val >> 12) & 0x7) * 6.75
```

CSD Field

Charging Share Delay / Sampling Delay Selection

These bits define the delay between consecutive ADC samples. The programmable sampling delay allows modifying the sampling frequency during hardware accumulation to suppress periodic noise sources that may otherwise disturb the sampling. The SAMPDLY field can also be modified automatically from one sampling cycle to another, by setting the ASDV bit. The delay is expressed as CLK_ADC cycles and is given directly by the bit field setting. The sampling cap is kept open during the delay.

RESPRES Field

High 4bits, resistor selection, low 4bits prescales selection

```
/* Combine Resistor / Prescaler */
#define NODE_RSEL_PRSC(r, p) (uint8_t)(((r) << 4u) | (p))
```

Bits 2:0 – PRESC[2:0] Prescaler

These bits define the division factor from the peripheral clock (CLK_PER) to the ADC clock (CLK_ADC).

Value	Name	Description
0x0	DIV2	CLK_PER divided by 2
0x1	DIV4	CLK_PER divided by 4
0x2	DIV8	CLK_PER divided by 8
0x3	DIV16	CLK_PER divided by 16
0x4	DIV32	CLK_PER divided by 32
0x5	DIV64	CLK_PER divided by 64
0x6	DIV128	CLK_PER divided by 128
0x7	DIV256	CLK_PER divided by 256

DRIFTNODE Field

The target sensor node for autoscan drift followed. In Evsys lowpower mode, the the autoscan node is not scanning in normal, so there we dedicate a normal to follow drifting. Note if autoscan node's digital gain is not matched with the target node's digital gain, that means you set the faster or slower drift rate than target node(2^n).

6.8.3 Messages

The message data for the Low Power Idle Configuration T126 object is shown in Table 6-13.

TABLE 6-13: MESSAGE DATA FOR LOW POWER IDLE CONFIGURATION T126 (SPT_LOWPOWERIDLECONFIG_T126)

Byte	Field	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	STATUS	Mode	Method	Reserved			TYPE		
2	DELTA	Delta value LSByte							
3		Delta value MSByte							

STATUS Field

This field reports the current state of the Low Power Idle Configuration T126 object.

Mode: Indicates whether the system is in low power (idle) mode; 1 – low power mode, 0 – active mode.

Mode: Indicates the method used for low power; 1 – software (periodical), 0 – event system.

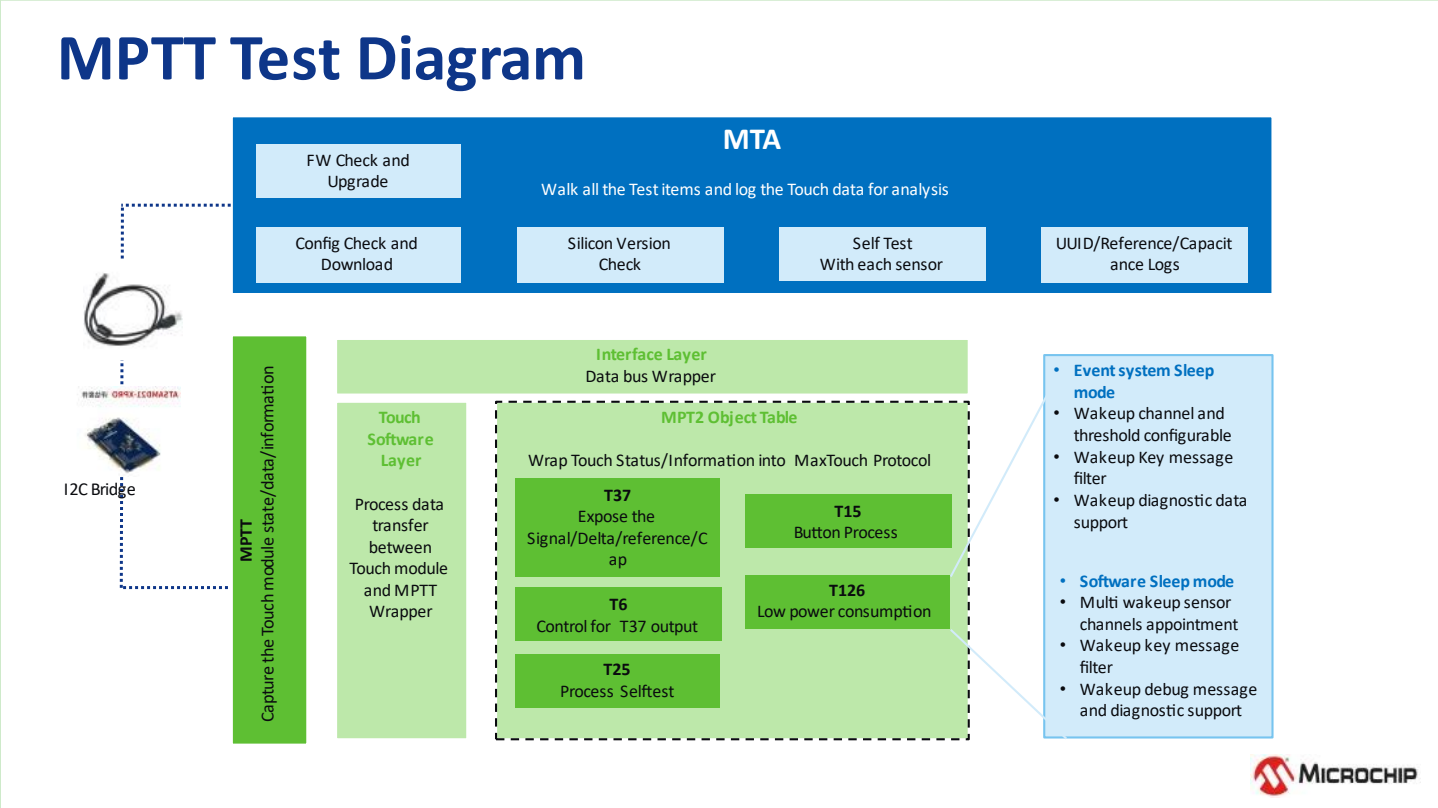
TYPE: Indicates the Message type (see Table 6-69).

TABLE 6-69:MESSAGE TYPE

Value	Meaning
0	Reserved for Command Processor T6 Report All
1	Positive threshold breached (touch detect)
2	Negative threshold breached (antitouch detect)
3	Autonomous mode started
4	Autonomous mode ended (other than touch or antitouch detect)
5	Force to wake up
6	Entering sleep mode
All other values	Reserved

Appendix A: Measurement processing on QTouch

Please refer to:
https://github.com/PitterL/mpt2/blob/EVK_3217_Xpro/doc/MPTT%20architecture%20update(v25)%2020210716.pdf



Appendix B: CHECKSUM CALCULATION

Please refer to:

https://github.com/PitterL/mpt2/blob/EVK_3217_Xpro/mpt2/crc.c

```
/*
calculate one byte input value with CRC 8 Bit
    @crc: last crc value
    @data: data input
    @returns calculated crc value
*/
u8 crc8(u8 crc, u8 data)
{
    static const u8 crcpoly = 0x8C;
    u8 index;
    u8 fb;
    index = 8;

    do
    {
        fb = (crc ^ data) & 0x01;
        data >>= 1;
        crc >>= 1;
        if (fb)
            crc ^= crcpoly;
    } while (--index);

    return crc;
}


/*
calculate two byte input value with CRC 24 Bit
    @crc: last crc value
    @firstbyte: byte 1
    @secondbyte: byte 2
    @returns calculated crc value
*/
u32 crc24(u32 crc, u8 firstbyte, u8 secondbyte)
{
    const u32 crcpoly = 0x80001B;
    u32 data_word;

    data_word = ((u16)secondbyte << 8) | firstbyte;
    crc = ((crc << 1) ^ data_word);

    if (crc & 0x1000000)
        crc ^= crcpoly;

    return crc;
}
```

Appendix C: Revision History

Revision A (July 1st 2019)

Initial edition for basic function parameter – Draft

Revision B (July 29th 2019)

Initial edition for full function parameter – V1.00

Revision C (June 18th 2021)

Add T126 and modify T7 for low power – V1.01

Add T25 message format – V1.02

Add T37/T6/T8/T25 specifiation

Checksum calculation

Measurement processing diagram – V1.03