

QUICKSENSE™ COMMUNICATIONS INTERFACE

1. Introduction

The QuickSense™ Communications Interface (QSCI) is a protocol which allows an MCU running a configurable QSCI Firmware Library to send QuickSense data through one of the MCU's serial peripherals. QuickSense data includes capacitive sensing measurements, infrared proximity measurements, interpreted group values, gestures, and nD pad positions. QuickSense applications, such as the Performance Analysis Tool, use the QSCI protocol to communicate to QuickSense kits like the QuickSense Front Panel and the IRSliderEK. This document covers the capabilities of QSCI and the protocol specifications. It assumes the reader is familiar with the QuickSense Firmware API. For more information about the QuickSense Firmware API, refer to “AN366: QuickSense Firmware API”.

1.1. Terminology

In this application note, the following definitions apply:

- Device—The MCU that is running the QuickSense Firmware API
- Host—The MCU/application that is communicating with the device
- Command—Request sent from the host to the device
- Response—Data/acknowledgement sent from the device to the host
- Class Type—Classification of the elements in a QuickSense system.
- Data Type—Defines the type of data sent in a response
- Data Transfer—The transmission of runtime values from the device to the host
- Transfer Type—Describes the information sets sent in a data transfer
- Transfer Mode—Determines the events that trigger a data transfer
- XDATA memory—On-chip volatile memory mapped to XRAM addresses
- CODE memory—On-chip non-volatile flash memory

1.2. QSCI Overview

QSCI contains an enumeration process where the device describes its capabilities to the host. This allows the host to know what types of data are available on the device. Data is transmitted in a packetized structure which contains a header, payload, and checksum. The host interacts with the device through a specific set of commands; the device responds accordingly with a set of responses. A host could be an application running on a PC, or it could be another MCU. QSCI uses a hardware peripheral, such as the UART, on the MCU to transmit serial data to a host.

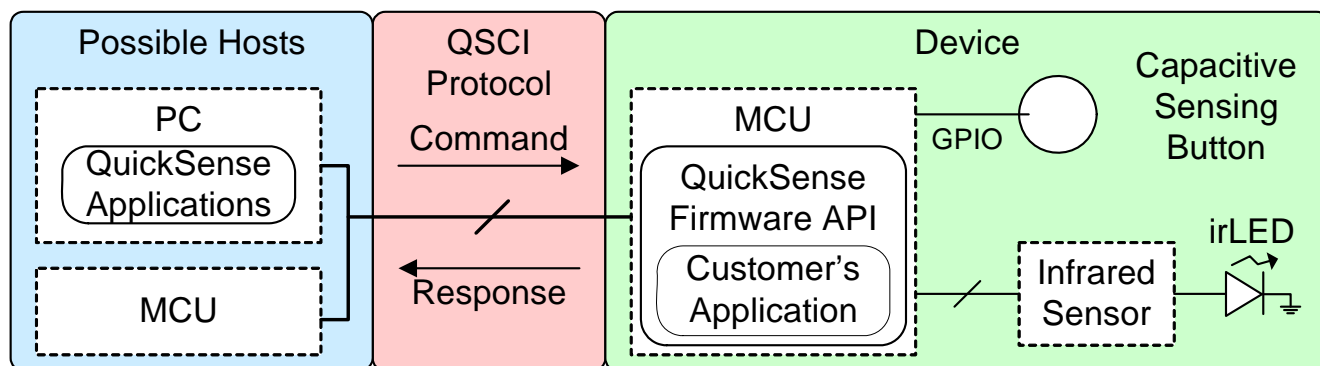


Figure 1. QuickSense System

1.3. QSCI Implementations

There are two implementations of the communications interface. QSCI offers a command based architecture with many features like transfer types and modes. In QSCI the host interacts with the device through an extensive command set. The device parses the command and transmits or modifies the requested QuickSense variables.

QSCI Lite offers a memory-map based architecture which supports a smaller command set than QSCI. The majority of the interaction in QSCI Lite is through address read and write commands. The device provides the host with the addresses of QuickSense variables. The host decides what data is relevant and when to retrieve it. QSCI Lite transfers complexity onto the host, but reduces the memory footprint on the device.

Certain commands such as retrieving the board ID, retrieving the firmware capability arrays, or resetting the device are common across both communication schemes.

The differences in feature sets are described in Table 1.

Table 1. QSCI vs. QSCI Lite Feature Sets

Features	QSCI	QSCI Lite
Device Enumeration	Yes	Yes
Command Interface	Yes	Yes
Packet Header	Yes	Yes
Packet Checksum	Yes	Yes
Firmware Versions	Yes	Yes
Device Calibration	Yes	Yes
Device-based Timing	Yes	No
Transfer Modes	Yes	No
Transfer Types	Yes	No
Direct Device Memory Manipulation	No	Yes
Smaller Code Size	No	Yes

1.4. Initial Connection to a Device

Upon connection, the host may obtain information about the device's capabilities and the board identifier. These steps are useful if the host:

- Supports multiple boards
- Supports multiple firmware versions
- Has the capability to upload a new firmware image to the target
- Wants to test the communication channel (ping the device using either command)

The Firmware Revision and Capabilities array allows the host to know what features are available on the device. The device's board ID allows the host to quickly determine the hardware that is connected. The host could quickly determine the characteristics of a device based on a saved device profile. This would allow the host to skip the enumeration process for particular board IDs.

If the host does not have a device profile for a specific board ID, the host can retrieve enumeration data from the device. The enumeration process for QSCI is explained in "2.2. QSCI Enumeration Process."; the enumeration process for QSCI Lite is explained in "3.2.2. QSCI Lite Enumeration Process" on page 52.

1.5. General Packet Structure

A data packet is comprised of three parts: two header bytes, a data payload and a checksum byte. The two header bytes define the command or response identifier, and the length of the packet. The data payload carries information specific to the command. The checksum byte is the 8-bit sum of the bytes sent in the header and in the data payload.

1.6. Packet Timeouts

The device maintains a 10 ms timeout for each received byte. If a command is being received in a packet, the delay between each byte must not exceed 10 ms. The communications interface's state machine is reset if the timeout is exceeded. This gives both host and device a way to recover from loss of synchronization.

1.7. General Response Packets

All commands elicit a response from the device. Some commands have specific response from the device while other commands only require a General Response. The General Response is also used to transmit error codes if the device cannot parse the command sent.

2. QSCI

QSCI provides access to QuickSense data through a command and response interface. It also provides a data transfer mechanism where the host specifies what data it cares about and when the device transmits the data.

2.1. QSCI Command Set

Table 2 covers the command and response set available to QSCI. Any command ID sent from the host that is not on this list causes the device to send an “Unknown command” error code. Some commands elicit a “Bad Request” warning code if the parameters in the command are invalid. The command ID and response ID values in Table 2 represent Header Byte 0 with the three least significant bits masked out.

Table 2. QSCI Command Set

Command ID	Command Type	Response ID	Response Type
0x00	Get Firmware Revision and Capabilities	0x00	Firmware Revision and Capabilities Response
0x08	Get Board ID	0x08	Board ID Response
0x10	Reset Device	0x10	General Response—No Error
0x18	Start Enumeration	0x10	General Response—No Error
		0x18	Enumeration Response
		0x10	General Response—Bad Request
0x20	Set QSCI Transfer Frequency	0x10	General Response—No Error
		0x10	General Response—Bad Request
0x28	Start Transfer	0x10	General Response—No Error
		0x28	Data Transfer Response
		0x10	General Response—Bad Request
0x30	Stop Transfer	0x10	General Response—No Error
0x38	Set Class Enable	0x10	General Response—No Error
		0x10	General Response—Bad Request
0x40	Get Class Enable Array	0x40	Class Enable Array Response
		0x10	General Response—Bad Request
0x48	Set Flash Key	0x10	General Response—No Error
		0x10	General Response—Bad Request
0x50	Set NVCCA	0x10	General Response—No Error
		0x10	General Response—Bad Request
		0x10	General Response—Bad Threshold
0x58	Get NVCCA	0x58	Get NVCCA Response
		0x10	General Response—Bad Request
0x60	Erase NVCCA	0x10	General Response—Bad Request
		0x10	General Response—Bad Request
0x68	Channel Calibration Check	0x68	Channel Calibration Response
0x70	Set Generic Data	0x10	General Response—No Error
		0x10	General Response—Bad Request

2.2. QSCI Enumeration Process

The enumeration process starts when the host sends the Start Enumeration command. Once the device receives the command, it acknowledges the command with a General Response. The device then sends up its configuration information through multiple Enumeration Responses. In the Enumeration Response there is an enumeration identifier byte which specifies the configuration structure being sent in that packet. If a certain structure does not exist due to device configurations, it is skipped. Structures are sent in ascending enumeration ID value; a list of all enumeration IDs are found in Table 9. Once all configuration structures are sent, the device sends an Enumeration Response with an enumeration finished ID.

The device has a configuration, ENUMERATION_MODE, which includes or excludes the Start Enumeration command in the build. If the device's ENUMERATION_MODE is DISABLED, the build's code size is reduced and it is assumed that the host knows the device's enumeration information. The host could contain many enumeration profiles which are then loaded depending on the device's board ID.

2.3. Data Transfer

QSCI enables the device to stream the following information, in the order shown below:

- Generic Data Class—Generic Data Elements
- 3D Pad Class—3D Pad Points
- 2D Pad Class—2D Pad Points
- 1D Pad Class—1D Pad Points
- Group Class—Interpreted Group Data
- Threshold Class—Threshold States
- Channel Class—Raw Channel Values and Runtime Baselines

Each class has two bit arrays, a TransferEnable array and a ProcessEnable array. The TransferEnable array allows the host to select the indices of a class whose data it wants to receive. An index is selected if its TransferEnable bit is set. A class is selected if any of its indices are selected. The ProcessEnable array allows the host to specify the index that is processed by the QuickSense firmware and MCU. The host has the option to manipulate both of these arrays through the Set Class Enable command.

If transfers are enabled for a class's index, processing is also enabled for the same class's index. If processing is disabled for a class's index, transfers are disabled for the same class's index. This ensures that the host receives accurate values at all times.

When the host sends a Start Transfer command, the QuickSense firmware processes the TransferEnable array for all classes to know what data types to send. For example, if any index in the channel class is enabled for transfers, the firmware sends a Data Transfer Response with a channel value identifier and a Data Transfer Response with a runtime baseline identifier since both data types are apart of the channel class. If none of the indices in the channel class are enabled, the firmware will not send a Data Transfer Response with either a channel value or runtime baseline identifier.

The host cannot manipulate a TransferEnable array while a Start Transfer command is being executed. The host must send a Stop Transfer command, manipulate the TransferEnable array and then send a Start Transfer command.

2.3.1. Transfer Types

There are two different transfer types, Selected transfers and SelectedAndUpdated transfers. Each transfer type has a different data payload structure. Both transfer types only transmit data for classes that have been selected. The Selected transfer type sends data for all selected indices regardless if they have changed since the last transfer. The SelectedAndUpdated transfer type require that the class indices have been selected **and** the data for that index has changed from the last transfer before being sent to the host.

Both transfer types can be supported in firmware at the same time. The TRANSFER_TYPE_SUPPORTED configuration allows the customer to include or exclude certain transfer types. The transfer type for a specific transfer is defined in the transfer configuration byte sent during a Start Transfer command. If a transfer type is not supported by the device, the device sends a General Response—Bad Request Warning.

SelectedAndUpdated transfers do not support the transmission of channel values or runtime baseline values since they change every time a sample occurs due to noise.

2.3.2. Transfer Modes

In addition to the two transfer types, there are three transfer modes: Periodic, On-Update and On-Demand. The transfer modes determine when a data transfer occurs.

In Periodic mode, transfers are synchronous and device driven. The device's timer sets the SI_SendData flag at the QSCI Transfer Frequency. In this mode, SI_Update() checks to see if SI_SendData is set before starting a data transfer. The host will see a data transfer packet at the interval defined by the QSCI Transfer Frequency which can be set via the Set QSCI Transfer Frequency command.

In On-Update mode, transfers are asynchronous and device driven. SI_Update() checks the ClassStatus bit arrays to know if a value for an index has changed. If the ClassStatus bit arrays are nonzero, a data transfer occurs for that class. The host will see a data packet only if data has changed for a selected class. Channel values and runtime baseline values cannot be sent while in this transfer mode.

In On-Demand mode, transfers are host driven. The device sets the SI_HostDemand flag every time a Start Transfer command is sent from the host. The flag forces the firmware to call QS_UpdateChannels() to update the system. The device then starts a data transfer. In this transfer mode, the host is not required to send a Stop Transfer command to stop data transfer. Since transfers are host driven, the timing burden is removed from the device and placed on the host.

All transfer modes can be supported in firmware at the same time. The TRANSFER_MODE_SUPPORTED configuration allows the customer to include or exclude certain transfer modes. The transfer mode is defined in the transfer configuration byte sent during a Start Transfer command. If a transfer mode is not supported by the device, the device sends a General Response—Bad Request Warning.

2.3.3. Transfer Configurations

Table 3 describes the possible transfer configurations. Each transfer configuration defines a transfer type and transfer mode which determines what data and when the data is transferred to the host.

Table 3. Transfer Configurations

Transfer Type	Transfer Mode	Initiated By	Data Transferred	Data not available
Selected	Periodic	Device; Using a timer on the MCU	All enabled indices for all classes	N/A
	On-Update	Device; Value update within a given class	All enabled indices within the class with an updated value	Channel Values and Runtime Baselines
	On-Demand	Host sends a Start Transfer command	All enabled indices for all classes	N/A
SelectedAndUpdated	Periodic*	Device; Using a timer on the MCU	Only enabled indices with a value update for updated classes	Channel Values and Runtime Baselines
	On-Update	Device; Value update within a given class	Only enabled indices with a value update for updated classes	Channel Values and Runtime Baselines
	On-Demand	Host; Sent a Start Transfer Command	Only enabled indices with a value update for updated classes	Channel Values and Runtime Baselines

***Note:** An empty packet is sent if no values have updated.

Selected Periodic Transfers—At the specified QSCI Transfer Frequency, the firmware sends all selected indices to the host for all selected classes.

Selected On-Update Transfers—If a selected index for any of the selected classes has changed, the firmware sends all selected indices to the host for the updated class.

Selected On-Demand Transfers—The host requests an on-demand transfer. The device firmware calls `QS_UpdateChannels()` and sends all selected indices to the host for all selected classes. Stop transfers are not required for this transfer combination.

SelectedAndUpdated Periodic Transfers—At the specified QSCI Transfer Frequency the firmware sends information to the host if the class's index is selected and its value has changed since the last transfer. If no data has changed since the last transfer, the firmware will send an empty packet; a packet that contains no data payload. **Channel and runtime baseline values are not available for this transfer combination.**

SelectedAndUpdated On-Update Transfers—If a selected index for any of the selected classes has changed the firmware sends the updated index and its value to the host. **Channel and runtime baseline values are not available for this transfer combination.**

SelectedAndUpdated On-Demand Transfers—The host requests an on-demand transfer. The device firmware calls `QS_UpdateChannels()` and sends data for a selected class if a selected index's value has changed. **Channel and runtime baseline values are not available for this transfer combination.**

2.4. Header Bytes

The two header bytes of a packet define the command/response identifier and the packet length. Header Byte 0 contains the 5-bit command ID and the most significant 3 bytes of the packet length. Header Byte 1 contains least significant 8-bits of the packet length. This allows for packets up to 2046 bytes long.

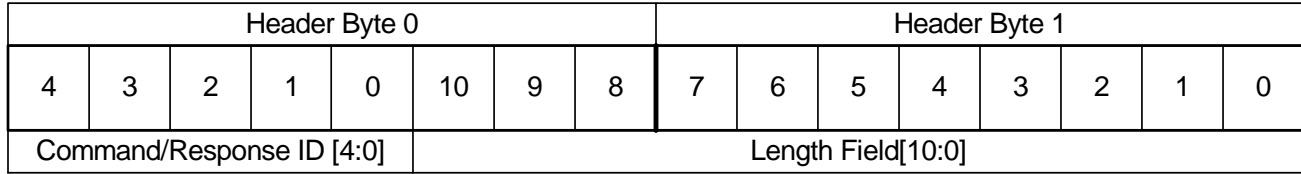


Figure 2. QSCI Header Bytes

2.5. QSCI Commands and Responses

This section defines the properties and payload structure for each command and response.

2.5.1. Get Firmware Revision and Capabilities

Command ID: 0x00

Description: This command asks the device for its firmware revision and capabilities array.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x00							
1	Packet Length	0x01							
2	Checksum	0x01							

Device Response:

Responses	Situation
Firmware Revision and Capabilities Response	Normal Operation

2.5.2. Firmware Revision and Capabilities Response

Response ID: 0x00

Description: This response returns the device's firmware revision and capabilities array. The revision values are BCD encoded. For example if the QSCI Major Revision byte is 0x02 and the QSCI Minor Revision byte is 0x30, the QSCI revision is v02.30. For more information about each byte, refer to AN366.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x00							
1	Packet Length	0x0B							
2	QSCI Major Revision	variable							
3	QSCI Minor Revision	variable							
4	Baselining Method	0x01							
5	Command Availability	0	0	0	0	0	QSCI Mode	Flash Mode	Enum Mode
							Table 4	Table 5	Table 6
6	Transfer Options Supported	0	0	Transfer Types Supported			Transfer Modes Supported		
				Table 7					
7	QuickSense API Major Revision	variable							
8	QuickSense API Minor Revision	variable							
9	Application Major Revision	variable							
10	Application Minor Revision	variable							
11	RX_BUFFER_SIZE	variable							
12	Checksum	variable							

Table 4. QSCI Mode Options

QSCI Mode	Value	Description
QSCI	0	The serial interface used follows the QSCI protocol.
QSCI_LITE	1	The serial interface used follows the QSCI Lite protocol.

Table 5. Flash Mode Options

Flash Mode	Value	Description
READ_WRITE	0	The host can read from and write to NVCCA flash memory.
READ_ONLY	1	The host can only read from NVCCA flash memory.

Table 6. Enumeration Mode Options

Enumeration Mode	Value	Description
DISABLED	0	The Start Enumeration command is not available.
ENABLED	1	The Start Enumeration command is available.

Table 7. Transfer Options Supported Bits

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
N/A	N/A	SelectedAndUpdated	Selected	N/A	On-Demand	On-Update	Periodic

A value of 0 means the option is not supported.
A value of 1 means the option is supported.

2.5.3. Get Board ID

Command ID: 0x08

Description: This command asks the device for its board ID.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x08							
1	Packet Length	0x01							
2	Checksum	0x09							

Device Response:

Responses	Situation
Board ID Response	Normal Operation

2.5.4. Board ID Response

Response ID: 0x08

Description: In this response, the device sends its board ID.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x08							
1	Packet Length	0x02							
2	Board Identifier	variable							
3	Checksum	variable							

2.5.5. Reset Device

Command ID: 0x10

Description: This command tells the device to perform a soft reset.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x10							
1	Packet Length	0x01							
2	Checksum	0x11							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation

2.5.6. General Response

Response ID: 0x10

Description: This response is sent to acknowledge that a command has executed on the device. This response provides an error code about the command that was processed.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x10							
1	Packet Length	0x02							
2	Error Code	variable							
		Table 8							
3	Checksum	variable							

Table 8. Error and Warning Codes

Error Code	Description
0x00	No error or warning; Command completed successfully
0x01	ERROR: Bad Checksum
0x02	ERROR: Unknown Command
0xFF	WARNING: Bad Address
0xFD	WARNING: Packet Timeout
0xFC	WARNING: Bad Request
0xFB	WARNING: Bad Threshold

AN494

2.5.7. Start Enumeration

Command ID: 0x18

Description: This command tells the device to transmit back information about the device's capabilities. This command also disables all threshold detection on every channel. To determine the enumeration mode, use the Get Firmware Revision and Capabilities command.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	0x01							
2	Checksum	0x19							

Device Response:

Responses	Situation
General Response—No Error	Immediately after receiving the Start Enumeration command
Enumeration Response	When sending the device's capability arrays. Also sent when all enumeration information has been transferred.
General Response—Bad Request	If the device's ENUMERATION_MODE is DISABLED

2.5.8. Enumeration Response

Response ID: 0x18

Description: This response sends all available enumeration data to the host. The enumeration data is preceded by the enumeration type which identifies the enumeration data in the packet. The following subsections define all of the possible enumeration types that can be sent using this response. Table 9 defines all of the valid enumeration identifiers.

Table 9. Enumeration Type Identifiers

Enumeration Type	Identifier
Enumeration Finished	0x00
Channel Information	0x01
Group Type	0x02
Group Channel Lists	0x03
Threshold Percentages	0x04
Reference Baseline Magnitude	0x05
Baseline Update Rate	0x06
QSCI Update Frequency Range	0x07
Channel Calibration Information	0x08
IR Channel Configuration	0x09
1D Pad Capabilities	0x0A
1D Pad Channel List	0x0B
2D Pad Capabilities	0x0C
2D Pad Channel List	0x0D
3D Pad Capabilities	0x0E
3D Pad Channel List	0x0F
Generic Data Elements	0x10

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	variable							
		Table 9							
3	Enumeration Data	variable							
...							
N	Checksum	variable							

AN494

2.5.8.1. Enumeration Finished

Enumeration Type: 0x00

Description: This response tells the host that the device is finished transmitting all available enumeration information.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	0x02							
2	Enumeration Type	0x00							
3	Checksum	0x1A							

2.5.8.2. Channel Information

Enumeration Type: 0x01

Description: This response sends the QS_ChannellInfo[] array which describes the characteristics of each defined channel, including the sampling method and the threshold detection capabilities. The number of channels in system is the length of QS_ChannellInfo[].

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x01							
3	Enumeration Data; QS_ChannellInfo[0]	0	0	0	N/A	Sensing Method	Threshold Detection Supported		
						Table 10	Table 11		
...							
N	Checksum	variable							

Table 10. Sensing Methods

Sensing Method	Value	Description
Capacitive Sensing	0x00	Sampled using the CS0 peripheral on the MCU
IR Proximity	0x01	Sampled using a fixed function IR device; e.g., Si1120
IR Ambient Light	0x02	Sampled using a fixed function IR device; e.g., Si1120

Table 11. Threshold Detection Supported

Threshold Detection	Value	Description
No Threshold Detect	0x00	Threshold detection not enabled for this channel
Threshold not Modifiable	0x01	Threshold detection enabled, but host cannot modify calibration

Table 11. Threshold Detection Supported

Threshold Modifiable	0x02	Threshold detection enabled; host can modify calibration
Group Calibration	0x03	Specific threshold calibration for group algorithms; do not modify

2.5.8.3. Group Type**Enumeration Type:** 0x02

Description: This response transmits the QS_GroupType[] array which describes the type of each defined group in the device. The number of groups in the system is defined by the length of QS_GroupType[].

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x03							
3	Enumeration Data; QS_GroupType [0]	variable							
		Table 12							
...							
N	Checksum	variable							

Table 12. Group Types

Group Type	Value
Slider	0x00
Control Wheel	0x01
1D Gesture Group	0x02
2D Gesture Group	0x03
3D Gesture Group	0x04
Ambient Light Type Group	0x05

AN494

2.5.8.4. Group Channel Lists

Enumeration Type: 0x03

Description: This response transmits the contents of the arrays collected in the QS_GroupChannelAssignmentTable[] array. Each group channel list is transmitted in separate responses.

For gesture groups, the response sends the index of the pad that is creating the gesture.

For all other groups, the response sends the indices of the channels that are used to create the interpreted group value.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x03							
3	Channel index in group channel list	variable							
...							
N+1	Checksum	variable							

2.5.8.5. Threshold Percentages

Enumeration Type: 0x04

Description: This response transmits the QS_Threshold[] array which stores the threshold percentages for all channels. It transmits one channel at a time, starting with the one-byte inactive-to-active threshold percent followed by the one-byte active-to-inactive threshold percent.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x04							
3	Inactive-to-Active threshold; Channel 0	variable							
4	Active-to-Inactive threshold; Channel 0	variable							
...							
N	Checksum	variable							

2.5.8.6. Reference Baseline Magnitude**Enumeration Type:** 0x05

Description: This response transmits the QS_ReferenceBaseline[] array which contains the two-byte Reference Baseline Magnitude for each channel. The Reference Baseline Magnitude defines the difference between the channel's max value and the channel's idle value.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x05							
3	Reference Baseline Magnitude High Byte	variable							
4	Reference Baseline Magnitude Low Byte	variable							
...							
N	Checksum	variable							

2.5.8.7. Baseline Update Rate**Enumeration Type:** 0x06

Description: This response transmits the one-byte Baseline Update Rate for the system.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	0x03							
2	Enumeration Type	0x06							
3	Baseline Update Rate	variable							
4	Checksum	variable							

AN494

2.5.8.8. QSCI Update Frequency Range

Enumeration Type: 0x07

Description: This response transmits the UpdateFrequency[] array which defines, in Hz, the minimum and maximum transfer frequencies for the Periodic transfer mode.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	0x04							
2	Enumeration Type	0x07							
3	Minimum transfer frequency	variable							
4	Maximum transfer frequency	variable							
5	Checksum	variable							

2.5.8.9. Channel Calibration Information

Enumeration Type: 0x08

Description: This response transmits the QS_ChannelCalibration[] bit array that defines which channel indices are calibrated and which are not. The most significant bit signifies the 0, 8th, ... index. A value of 1 means the index is calibrated, a value of 0 means the index is not calibrated.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x08							
3	Channel Calibration	channel 0	channel 1	channel 2	channel 3	channel 4	channel 5	channel 6	channel 7
...							
N	Checksum	variable							

2.5.8.10. IR Channel Configuration

Enumeration Type: 0x09

Description: This response sends the QS_IRConfig[] array. The byte defines the mode that the IR peripheral uses to measure an IR Channel. Depending on the measurement mode, IR channel values could have a different meaning. The first byte sent is the configuration for the first IR Channel. This does not have to be channel 0 if there are CS0 and IR channels in a mixed system. Refer to AN366 for exact configuration values for each IR peripheral.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x09							
3	IR Configuration	variable							
...							
N	Checksum	variable							

2.5.8.11. 1D Pad Capabilities

Enumeration Type: 0x0A

Description: This response sends the 1D pad capability arrays pointed to by QS_1DpadCapabilities[]. The array defines the axes used for the pad, the type of pad, the maximum number of concurrent points, the gesture group associated with the pad, the axis coordinate maximum and the axis coordinate minimum for the pad. All coordinate values are a signed 16-bit value. All 1D capability arrays are sent in one response.

The dimensions byte defines the axes used for the pad. The first bit set correlates to the first axis, etc.

AN494

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x0A							
3	Dimensions	0	0	0	0	0	X	Y	Z
4	Pad Type	variable							
		Table 13							
5	Max number of concurrent points; high byte	variable							
6	Max number of concurrent points; low byte	variable							
7	Reserved	0x00							
8	Reserved	0x00							
9	Axis 1 coordinate maximum; high byte	variable							
10	Axis 1 coordinate maximum; low byte	variable							
11	Axis 1 coordinate minimum; high byte	variable							
12	Axis 1 coordinate minimum; low byte	variable							
...							
N	Checksum	variable							

Table 13. Pad Types

Pad Types	Value	Description
IR Slider	0x01	Defines the pad as an infrared Slider
IR 3LED	0x02	Defines the pad as an infrared 3 LED system

2.5.8.12. 1D Pad Channel List**Enumeration Type:** 0x0B**Description:** This response sends the arrays pointed to by QS_1DpadChannels[]. This defines the channels allocated to each pad. This works similarly to group channel list enumeration data.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x0B							
3	Channel index in 1D pad channel list	variable							
...							
N	Checksum	variable							

2.5.8.13. 2D Pad Capabilities**Enumeration Type:** 0x0C**Description:** This response sends the arrays pointed to in QS_2DpadCapabilities[]. This defines the type of pad, the maximum number of concurrent points, the gestures group associated with the pad, axis 1 coordinate maximum, axis 1 coordinate minimum, axis 2 coordinate maximum and axis 2 coordinate minimum. All coordinate values are signed 16-bits.

The dimensions byte defines the axes used for the pad. The first bit set correlates to the first axis, etc.

AN494

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x0C							
3	Dimension	0	0	0	0	0	X	Y	Z
4	Pad Type	variable							
		Table 13							
5	Max number of concurrent points; high byte	variable							
6	Max number of concurrent points; low byte	variable							
7	Reserved	0x00							
8	Reserved	0x00							
9	Axis 1 coordinate maximum; high byte	variable							
10	Axis 1 coordinate maximum; low byte	variable							
11	Axis 1 coordinate minimum; high byte	variable							
12	Axis 1 coordinate minimum; low byte	variable							
13	Axis 2 coordinate maximum; high byte	variable							
14	Axis 2 coordinate maximum; low byte	variable							
15	Axis 2 coordinate minimum; high byte	variable							
16	Axis 2 coordinate minimum; low byte	variable							
...							
N	Checksum	variable							

2.5.8.14. 2D Pad Channel List**Enumeration Type:** 0x0D**Description:** This response sends the arrays pointed to by QS_2DPadChannels[]. This defines the channels allocated to each pad. This works similarly to group channel list enumeration data.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x0D							
3	Channel index in 2D pad channel list	variable							
...							
N	Checksum	variable							

2.5.8.15. 3D Pad Capabilities**Enumeration Type:** 0x0E**Description:** This response sends the arrays pointed to in QS_3DPadCapabilities[]. This defines the type of pad, the maximum number of concurrent points, the gestures group associated with the pad, axis 1 coordinate maximum, axis 1 coordinate minimum, axis 2 coordinate maximum and axis 2 coordinate minimum, axis 3 coordinate maximum and axis 3 coordinate minimum. All coordinate values are signed 16-bits.

The dimensions byte defines the axes used for the pad. The first bit set correlates to the first axis, etc.

AN494

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x0E							
3	Dimensions	0	0	0	0	0	X	Y	Z
4	Pad Type	variable							
		Table 13							
5	Max number of concurrent points; high byte	variable							
6	Max number of concurrent points; low byte	variable							
7	Reserved	0x00							
8	Reserved	0x00							
9	Axis 1 coordinate maximum; high byte	variable							
10	Axis 1 coordinate maximum; low byte	variable							
11	Axis 1 coordinate minimum; high byte	variable							
12	Axis 1 coordinate minimum; low byte	variable							
13	Axis 2 coordinate maximum; high byte	variable							
14	Axis 2 coordinate maximum; low byte	variable							
15	Axis 2 coordinate minimum; high byte	variable							
16	Axis 2 coordinate minimum; low byte	variable							
17	Axis 3 coordinate maximum; high byte	variable							
18	Axis 3 coordinate maximum; low byte	variable							
19	Axis 3 coordinate minimum; high byte	variable							
20	Axis 3 coordinate minimum; low byte	variable							
...							
N	Checksum	variable							

2.5.8.16. 3D Pad Channel List**Enumeration Type:** 0x0F**Description:** This response sends the arrays pointed to by QS_3DPadChannels[]. This defines the channels allocated to each pad. This works similarly to group channel list enumeration data.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	variable							
2	Enumeration Type	0x0F							
3	Channel Index in 3D pad channel list	variable							
...							
N	Checksum	variable							

2.5.8.17. Generic Data Elements**Enumeration Type:** 0x10**Description:** This response sends the number of generic data elements in the system.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x18							
1	Packet Length	0x03							
2	Enumeration Type	0x10							
3	Number of generic data elements	variable							
4	Checksum	variable							

2.5.9. Set QSCI Transfer Frequency**Command ID:** 0x20**Description:** This command is used to change the frequency at which the device streams QuickSense data in the Periodic transfer mode. The data is in units of Hz.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x20							
1	Packet Length	0x02							
2	Transfer Frequency (Hz)	variable							
3	Checksum	variable							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation

AN494

2.5.10. Start Transfer

Command ID: 0x28

Description: This command tells the device to transfer QuickSense data in a specific transfer configuration. What data is sent is dependent on the settings in each class's TransferEnable array. Only indices with a set TransferEnable bit is sent. The transfer configuration determines how and when the data is transferred.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	0x02							
2	Transfer Configuration	Transfer Type	0	0	0	0	0	Transfer Mode	
		Table 14						Table 15	
3	Checksum								

Table 14. Transfer Type

Transfer Type	Value	Description
Selected	0x00	Only selected indices are sent to the host
SelectedAndUpdated	0x01	Only selected and updated indices are sent to the host

Table 15. Transfer Mode

Transfer Mode	Value	Description
Periodic	0x00	Data is transferred periodically based on the device's timer
On-Update	0x01	Data is transferred only when a value has updated
On-Demand	0x02	Data is transferred only when the host requests it

Device Response:

Responses	Situation
General Response—No Error	When the device receives the Start Transfer command
Data Transfer Response	When the device transfers data to the host
General Response—Bad Request	If the transfer configuration is not supported in firmware
	If no TransferEnable arrays are set

2.5.11. Data Transfer Response

Response ID: 0x28

Description: This response transfers data based on the settings of each class's TransferEnable and Status array. The response structure depends on which transfer type is selected with the Start Transfer command. Each response contains a data type identifier which defines what data is in the response packet. The data type identifiers are defined in Table 16. Data transfer responses are sent in descending data type order. The following subsections describe each packet structure in more detail.

Table 16. Data Type Identifiers

Data Type	Identifier
Channel Values	0x00
Runtime Baseline Values	0x01
Threshold States	0x02
Group Values	0x03
1D Pad Points	0x04
Reserved	0x05
2D Pad Points	0x06
Reserved	0x07
3D Pad Points	0x08
Reserved	0x09
Generic Data	0x0A

Packet Structure and Options: *General Structure*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	variable							
		Table 16							
3	Data Byte	variable							
...							
N	Checksum	variable							

In a SelectedAndUpdated Periodic transfer, if no values has changed since the last transfer the device will send an empty packet.

AN494

Packet Structure and Options: *For a SelectedAndUpdated Periodic transfer with no updates*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	0x02							
2	Data Type	variable							
		Table 16							
N	Checksum	variable							

2.5.11.1. Channel Values

Data Type: 0x00

Description: This response transfers data located in the QS_ChannelValue[] array. Channel data is transferred in ascending channel index order. Only channel indices selected in QS_ChannelTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_ChannelTransferEnable[] and QS_ChannelStatus[] can be sent.

The channel value for ALS Channels are encoded using the following formula:

$$[\text{Bit } 14:\text{Bit } 0] \times 4^{[\text{Bit } 15]}$$

Packet Structure and Options: *For the Selected transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x00							
3	Channel x value; high byte	variable							
4	Channel x value; low byte	variable							
...							
N	Checksum	variable							

Packet Structure and Options: *For the SelectedAndUpdated transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x00							
3	Channel x Index	variable							
4	Channel x value; high byte	variable							
5	Channel x value; low byte	variable							
...							
N	Checksum	variable							

2.5.11.2. Runtime Baseline Values**Data Type:** 0x01

Description: This response transfers data located in the QS_RuntimeBaseline[] array. Runtime baseline data is transferred in ascending channel index order. Only channel indices selected in QS_ChannelTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_ChannelTransferEnable[] and QS_ChannelStatus[] can be sent.

Packet Structure and Options: *For the Selected transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x01							
3	Runtime Active Baseline for channel x; high byte	variable							
4	Runtime Active Baseline for channel x; low byte	variable							
5	Runtime Inactive Baseline for channel x; high byte	variable							
6	Runtime Inactive Baseline for channel x; low byte	variable							
...							
N	Checksum	variable							

AN494

Packet Structure and Options: *For the SelectedAndUpdated transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x01							
3	Channel x index	variable							
4	Runtime Active Base-line for channel x; high byte	variable							
5	Runtime Active Base-line for channel x; low byte	variable							
6	Runtime Inactive Baseline for channel x; high byte	variable							
7	Runtime Inactive Baseline for channel x; low byte	variable							
...							
N	Checksum	variable							

2.5.11.3. Threshold State Values

Data Type: 0x02

Description: This response transfers data located in the QS_ThresholdState[] bit array. Threshold state data is transferred in ascending channel index order. Only indices selected in QS_ThresholdTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_ThresholdTransferEnable[] and QS_ThresholdStatus[] can be sent.

A threshold state value of 1 means the channel is active; a value of 0 means the channel is inactive.

Packet Structure and Options: *For the Selected transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x02							
3	Threshold State for channel x	variable							
...							
N	Checksum	variable							

Packet Structure and Options: *For the SelectedAndUpdated transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x02							
3	Channel x Index	variable							
4	Threshold State for channel x	variable							
...							
N	Checksum	variable							

AN494

2.5.11.4. Group Values

Data Type: 0x03

Description: This response transfers data located in the QS_GroupValue[] array. Group data is transferred in ascending group index order. Only indices selected in QS_GroupTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_GroupTransferEnable[] and QS_GroupStatus[] can be sent.

Depending on the group type, the group value is interpreted differently.

Table 17. Group Value High Byte

Group Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Slider	Interpreted Slider Position High Byte							
Control Wheel	Interpreted Control Wheel Position High Byte							
1D Gesture	Application defined value							
2D Gesture	Application defined value							
3D Gesture	Application defined value							
Ambient Light Type	0x00							

Table 18. Group Value Low Byte

Group Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Slider	Interpreted Slider Position Low Byte							
Control Wheel	Interpreted Control Wheel Position Low Byte							
1D Gesture	Application defined gesture symbol							
2D Gesture	Application defined gesture symbol							
3D Gesture	Application defined gesture symbol							
Ambient Light Type	0	0	0	0	0	Fluorescent	Incan- descent	Sunlight

Packet Structure and Options: *For the Selected transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x03							
3	Group x value; high byte	variable							
		Table 17							
4	Group x value, low byte	variable							
		Table 18							
...							
N	Checksum	variable							

Packet Structure and Options: *For the SelectedAndUpdated transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x03							
3	Group x Index	variable							
4	Group x value; high byte	variable							
		Table 17							
5	Group x value, low byte	variable							
		Table 18							
...							
N	Checksum	variable							

2.5.11.5. 1D Pad Points**Data Type:** 0x04

Description: This response transfers data located in the QS_1DPadPoints[] array. Point data is transferred in ascending pad index order. Only indices selected in QS_1DTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_1DTransferEnable[] and QS_1DStatus[] can be sent.

All coordinates are signed 16-bit values. If any coordinate in a point contains the value 0x7FFF, then the point is invalid.

Packet Structure and Options: *For the Selected transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x04							
3	Pad X, Point 0, Axis 1; high byte	variable							
4	Pad X, Point 0, Axis 1; low byte	variable							
...							
m	Pad X, Point Y, Axis 1; high byte	variable							
m+1	Pad X, Point Y, Axis 1; low byte	variable							
...							
N	Checksum	variable							

AN494

Packet Structure and Options: *For the SelectedAndUpdated transfer type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x04							
3	Pad X Index	variable							
4	Pad X, Point 0, Axis 1; high byte	variable							
5	Pad X, Point 0, Axis 1; low byte	variable							
...							
m	Pad X, Point Y, Axis 1; high byte	variable							
m+1	Pad X, Point Y, Axis 1; low byte	variable							
...							
N	Checksum	variable							

2.5.11.6. 2D Pad Points

Data Type: 0x06

Description: This response transfers data located in the QS_2DPadPoints[] array. Point data is transferred in ascending pad index order. Only indices selected in QS_2DTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_2DTransferEnable[] and QS_2DStatus[] can be sent.

All coordinates are signed 16-bit values. If any coordinate in a point contains the value 0x7FFF, then the point is invalid.

Packet Structure and Options: *For the Selected Transfer Type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x06							
3	Pad 0, Point 0, Axis 1; high byte	variable							
4	Pad 0, Point 0, Axis 1; low byte	variable							
5	Pad 0, Point 0, Axis 2; high byte	variable							
6	Pad 0, Point 0, Axis 2; low byte	variable							
...							
m	Pad 0, Point Y, Axis 1; high byte	variable							
m+1	Pad 0, Point Y, Axis 1; low byte	variable							
m+2	Pad 0, Point Y, Axis 2; high byte	variable							
m+3	Pad 0, Point Y, Axis 2; low byte	variable							
...							
N	Checksum	variable							

AN494

Packet Structure and Options: *For the SelectedAndUpdated Transfer Type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x06							
3	Pad X Index	variable							
4	Pad X, Point 0, Axis 1; high byte	variable							
5	Pad X, Point 0, Axis 1; low byte	variable							
6	Pad X, Point 0, Axis 2; high byte	variable							
7	Pad X, Point 0, Axis 2; low byte	variable							
...							
m	Pad X, Point Y, Axis 1; high byte	variable							
m+1	Pad X, Point Y, Axis 1; low byte	variable							
m+2	Pad X, Point Y, Axis 2; high byte	variable							
m+3	Pad X, Point Y, Axis 2; low byte	variable							
...							
N	Checksum	variable							

2.5.11.7. 3D Pad Points

Data Type: 0x08

Description: This response transfers data located in the QS_3DPadPoints[] array. Point data is transferred in ascending pad index order. Only indices selected in QS_3DTransferEnable[] can be sent. In the SelectedAndUpdated transfer type and/or On-Update transfer mode, only channel indices set in QS_3DTransferEnable[] and QS_3DStatus[] can be sent.

All coordinates are signed 16-bit values. If any coordinate in a point contains the value 0x7FFF, then the point is invalid.

Packet Structure and Options: *For the Selected Transfer Type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x08							
3	Pad X, Point 0, Axis 1; high byte	variable							
4	Pad X, Point 0, Axis 1; low byte	variable							
5	Pad X, Point 0, Axis 2; high byte	variable							
6	Pad X, Point 0, Axis 2; low byte	variable							
7	Pad X, Point 0, Axis 3; high byte	variable							
8	Pad X, Point 0, Axis 3; low byte	variable							
...							
m	Pad X, Point Y, Axis 1; high byte	variable							
m+1	Pad X, Point Y, Axis 1; low byte	variable							
m+2	Pad X, Point Y, Axis 2; high byte	variable							
m+3	Pad X, Point Y, Axis 2; low byte	variable							
m+4	Pad X, Point Y, Axis 3; high byte	variable							
m+5	Pad X, Point Y, Axis 3; low byte	variable							
...							
N	Checksum	variable							

AN494

Packet Structure and Options: *For the SelectedAndUpdated Transfer Type*

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x28							
1	Packet Length	variable							
2	Data Type	0x08							
3	Pad X Index	variable							
4	Pad X, Point 0, Axis 1; high byte	variable							
5	Pad X, Point 0, Axis 1; low byte	variable							
6	Pad X, Point 0, Axis 2; high byte	variable							
7	Pad X, Point 0, Axis 2; low byte	variable							
8	Pad X, Point 0, Axis 3; high byte	variable							
9	Pad X, Point 0, Axis 3; low byte	variable							
...							
m	Pad X, Point Y, Axis 1; high byte	variable							
m+1	Pad X, Point Y, Axis 1; low byte	variable							
m+2	Pad X, Point Y, Axis 2; high byte	variable							
m+3	Pad X, Point Y, Axis 2; low byte	variable							
m+4	Pad X, Point Y, Axis 3; high byte	variable							
m+5	Pad X, Point Y, Axis 3; low byte	variable							
...							
N	Checksum	variable							

2.5.12. Stop Transfer

Command ID: 0x30

Description: This command tells the device to stop data transfers.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x30							
1	Packet Length	0x01							
2	Checksum	0x31							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation

2.5.13. Set Class Enable

Command ID: 0x38

Description: This command allows the host to set a class's TransferEnable and ProcessEnable array. The index byte allows the host to choose which index of the enable array to modify. An index value of 0xFF sets the enable state for all indices of the appropriate enable array.

The upper nibble of the enable state byte modifies the class's ProcessEnable array. The lower nibble of the enable state byte modifies the class's TransferEnable array. The possible class types are defined in Table 19.

The generic data class does not have a ProcessEnable array. Attempts to modify its ProcessEnable array will cause the device to return a bad request warning.

Table 19. Class Type Identifiers

Class Type	Identifier
Channel Class	0x00
Reserved	0x01
Threshold Class	0x02
Group Class	0x03
1D Pad Class	0x04
Reserved	0x05
2D Pad Class	0x06
Reserved	0x07
3D Pad Class	0x08
Reserved	0x09
Generic Data Class	0x0A

AN494

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x38							
1	Packet Length	0x04							
2	Class Type	variable							
		Table 19							
3	Index Byte	variable							
4	Enable State Byte	0	0	ProcessEnable		0	0	TransferEnable	
				Table 20				Table 21	
5	Checksum	variable							

Table 20. ProcessEnable Options

Enable State	Value	Description
Disable	0x00	Clears the process enable bit for the selected index
Enable	0x01	Sets the process enable bit for the selected index
Keep Current State	0x02	Does not modify the process enable bit for the selected index

Table 21. TransferEnable Options

Enable State	Value	Description
Disable	0x00	Clears the transfer enable bit for the selected index
Enable	0x01	Sets the transfer enable bit for the selected index
Keep Current State	0x02	Does not modify the transfer enable bit for the selected index

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request Warning	If the class type does not exist
	If modifying the ProcessEnable for generic data class

2.5.14. Get Class Enable Array

Command ID: 0x40

Description: This command requests the device to return a class's enable array. The most significant bit in the class type byte determines whether the device returns a TransferEnable or a ProcessEnable array.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x40							
1	Packet Length	0x02							
2	Class Type	Enable Type	variable						
		Table 22	Table 19						
3	Checksum	variable							

Table 22. Enable Type Option

Enable Type	Value	Description
TransferEnable	0x00	The device sends the transfer enable array for the class
ProcessEnable	0x01	The device sends the process enable array for the class

Device Response:

Responses	Situation
Class Enable Array Response	Normal Operation
General Response—Bad Request Warning	If the class type does not exist
	If requesting the ProcessEnable for generic data class

AN494

2.5.15. Class Enable Array Response

Response ID: 0x40

Description: This response transmits an enable array of the class requested by the host. Each bit in the enable array represents one index of the class. The most significant bit represents the 0, 8th, ..., etc. index. The most significant bit in the class type byte signifies whether a TransferEnable array or ProcessEnable array is sent in the data payload.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x40							
1	Packet Length	variable							
2	Class Type	Enable Type	variable						
		Table 22	Table 19						
3	Enable Array Byte 0	channel 0	channel 1	channel 2	channel 3	channel 4	channel 5	channel 6	channel 7
...							
N	Checksum	variable							

2.5.16. Set Flash Key

Command ID: 0x48

Description: This command writes to the QS_FlashKey[] array. The correct key codes must be written to enable flash writes and erases. Read to the device MCU's datasheet for flash key codes.

Packet Structure and Options:

Byte	Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x48							
1	Packet Length	0x03							
2	First Flash Key	variable							
3	Second Flash Key	variable							
4	Checksum	variable							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request	If FLASH_MODE is READ_ONLY

2.5.17. Set NVCCA

Command ID: 0x50

Description: This command writes to the non-volatile calibration and configuration area (NVCCA). The packet structure is dependent on the NVCCA type described in Table 23. The host must set the correct flash key codes before using this command. The following subsections describe each packet structure in more detail.

Table 23. NVCCA Type Identifiers

NVCCA Type	Identifier
Threshold Percentage	0x00
Reference Baseline Magnitude	0x01
Baseline Update Rate	0x02

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x50							
1	Packet Length	variable							
2	NVCCA Type	variable Table 23							
3	NVCCA Type specific data	variable							
...							
N	Checksum	variable							

2.5.17.1. Set Threshold Percentage

NVCCA Type: 0x00

Description: This command sets a threshold percentage for a specific channel. The host must define which threshold to write to in the threshold type byte.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x50							
1	Packet Length	0x05							
2	NVCCA Type	0x00							
3	Channel Index	variable							
4	Threshold Type	variable Table 24							
5	Threshold Percentage	variable							
6	Checksum	variable							

Table 24. Threshold Type Options

Threshold Type	Value	Description
Inactive-to-Active	0x00	This percentage determines when a channel becomes active.
Active-to-Inactive	0x01	This percentage determines when a channel becomes inactive.

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request	If FLASH_MODE is READ_ONLY
	If NVCCA Type is invalid
	If CHANNEL_THRESHOLD is DISABLED
General Response—Bad Threshold	If threshold percentage is greater than 100

2.5.17.2. Set Reference Baseline Magnitude

NVCCA Type: 0x01

Description: This command sets the Reference Baseline Magnitude for a specific channel. The Reference Baseline Magnitude defines the general number of counts between an idle channel and its max active value at calibration.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x50							
1	Packet Length	0x05							
2	NVCCA Type	0x01							
3	Channel Index	variable							
4	Reference Baseline Magnitude; high byte	variable							
5	Reference Baseline Magnitude; low byte	variable							
6	Checksum	variable							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request	If FLASH_MODE is READ_ONLY
	If NVCCA Type is invalid

2.5.17.3. Set Baseline Update Rate

NVCCA Type: 0x02

Description: This command sets the Baseline Update Rate for the system. The Baseline Update Rate determines how often the runtime baselines update in the system. The Baseline Update Rate value is defined in units of seconds.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x50							
1	Packet Length	0x03							
2	NVCCA Type	0x02							
3	Baseline Update Rate	variable							
4	Checksum	variable							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request	If FLASH_MODE is DISABLED
	If NVCCA Type is invalid

2.5.18. Get NVCCA

Command ID: 0x58

Description: This command asks the device to send information from the non-volatile calibration and configuration area.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x58							
1	Packet Length	0x03							
2	NVCCA Type	variable							
		Table 23							
3	Channel Index	variable							
4	Checksum	variable							

Device Response:

Responses	Situation
Get NVCCA Response	Normal Operation
General Response—Bad Request	If CHANNEL_THRESHOLD is DISABLED and request threshold percentages
	If NVCCA Type is invalid

AN494

2.5.19. Get NVCCA Response

Command ID: 0x58

Description: This response sends NVCCA information to the host. The payload structure changes dependent on NVCCA type. The following subsections describe the individual response packets based on NVCCA type.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x58							
1	Packet Length	variable							
2	NVCCA Type	variable							
		Table 23							
3	NVCCA Type specific data	variable							
...							
N	Checksum	variable							

2.5.19.1. Threshold Percentage Response

NVCCA Type: 0x00

Description: This response sends the Inactive-to-Active and Active-to-Inactive threshold percentages for a specific channel.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x58							
1	Packet Length	0x05							
2	NVCCA Type	0x00							
3	Channel Index	variable							
4	Inactive-to-Active Threshold	variable							
5	Active-to-Inactive Threshold	variable							
6	Checksum	variable							

2.5.19.2. Reference Baseline Magnitude Response**NVCCA Type:** 0x01**Description:** This response sends the Runtime Baseline Magnitude for a specific channel.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x58							
1	Packet Length	0x05							
2	NVCCA Type	0x01							
3	Channel Index	variable							
4	Runtime Baseline Magnitude; high byte	variable							
5	Runtime Baseline Magnitude; low byte	variable							
6	Checksum	variable							

2.5.19.3. Baseline Update Rate Response**NVCCA Type:** 0x02**Description:** This response returns the Baseline Update Rate of the system. The Baseline Update Rate is in units of seconds.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x58							
1	Packet Length	0x03							
2	NVCCA Type	0x02							
3	Baseline Update Rate	variable							
4	Checksum	variable							

AN494

2.5.20. Erase NVCCA

Command ID: 0x60

Description: This command tells the device to erase the page(s) of flash memory that contain the non-volatile calibration and configuration area. The host must set the flash key codes before using this command. After the NVCCA page is erased, the device clears its threshold states and calibration states for each channel.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x60							
1	Packet Length	0x01							
2	Checksum	0x61							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request	If FLASH_MODE is READ_ONLY

2.5.21. Channel Calibration Check

Command ID: 0x68

Description: This command asks the device if a particular channel index is calibrated.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x68							
1	Packet Length	0x02							
2	Channel Index	variable							
3	Checksum	variable							

Device Response:

Responses	Situation
Channel Calibration Response	Normal Operation

2.5.22. Channel Calibration Response**Response ID:** 0x68**Description:** This response tells the host if a specific channel index is calibrated. It returns a value of 1 if the channel is calibrated; it returns a value of 0 if the channel is not calibrated.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x68							
1	Packet Length	0x03							
2	Channel Index	variable							
3	Calibration Byte	variable							
4	Checksum	variable							

2.5.23. Set Generic Data**Command ID:** 0x70**Description:** This command sets a particular value for a generic data element.**Packet Structure and Options:**

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x70							
1	Packet Length	0x04							
2	Generic Data Index	variable							
3	Generic Data Value; high byte	variable							
4	Generic Data Value; low byte	variable							
5	Checksum	variable							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request	If there are no generic data elements in the system

3. QSCI Lite

QSCI Lite provides access to QuickSense data with a reduced memory footprint, compared to QSCI, obtained through a reduced command set. The host is responsible for retrieving the data it requires. The transfer functionality is similar to the On-Demand transfer mode in QSCI where the host sends a command every time it wants data from the device.

QSCI Lite support for the SMBUS and SPI protocols was added in QSCI Version 2.32.

3.1. QSCI Lite Command Set

Table 25 covers the command and response set available to QSCI Lite. A command identifier sent from the host that is not in this list causes the device to send an “unknown command” error code. Some commands send a “bad request” warning code if the parameters in the command are invalid or if the command is not available due to device configurations.

Table 25. QSCI Lite Command Set

Command ID	Command Type	Response ID	Response Type
0x00	Get Firmware Revision and Capabilities	0x00	Firmware Revision and Capabilities Response
0x08	Get Board ID	0x08	Board ID Response
0x10	Reset Device	0x10	General Response—No Error
0x60	Erase NVCCA	0x10	General Response—No Error
		0x10	General Response—Bad Request
0x78	Read Address	0x78	Read Address Response
		0x10	General Response—Bad Address
0x80	Write Address	0x10	General Response—No Error
		0x10	General Response—Bad Address

3.2. QSCI Lite Host Interaction

This section provides an overview of how a host interacts with a QuickSense device using QSCI Lite.

When a host first connects to a device, it should send the Get Firmware Revision and Capabilities command to know if the device is using QSCI Lite and what firmware versions are supported on the device. This enables the host to know how the device can behave.

The second task is to get the Board ID of the device. This can be used to identify what type of board is connected to the host. If the host and device agree to a specific board ID, then the host can load specific device profiles based on the Board ID and skip the enumeration process. If the board ID is not known by the host, then the host can follow the enumeration process defined in Section “3.2.2. QSCI Lite Enumeration Process.”

Figure 3 displays the initial connection process.

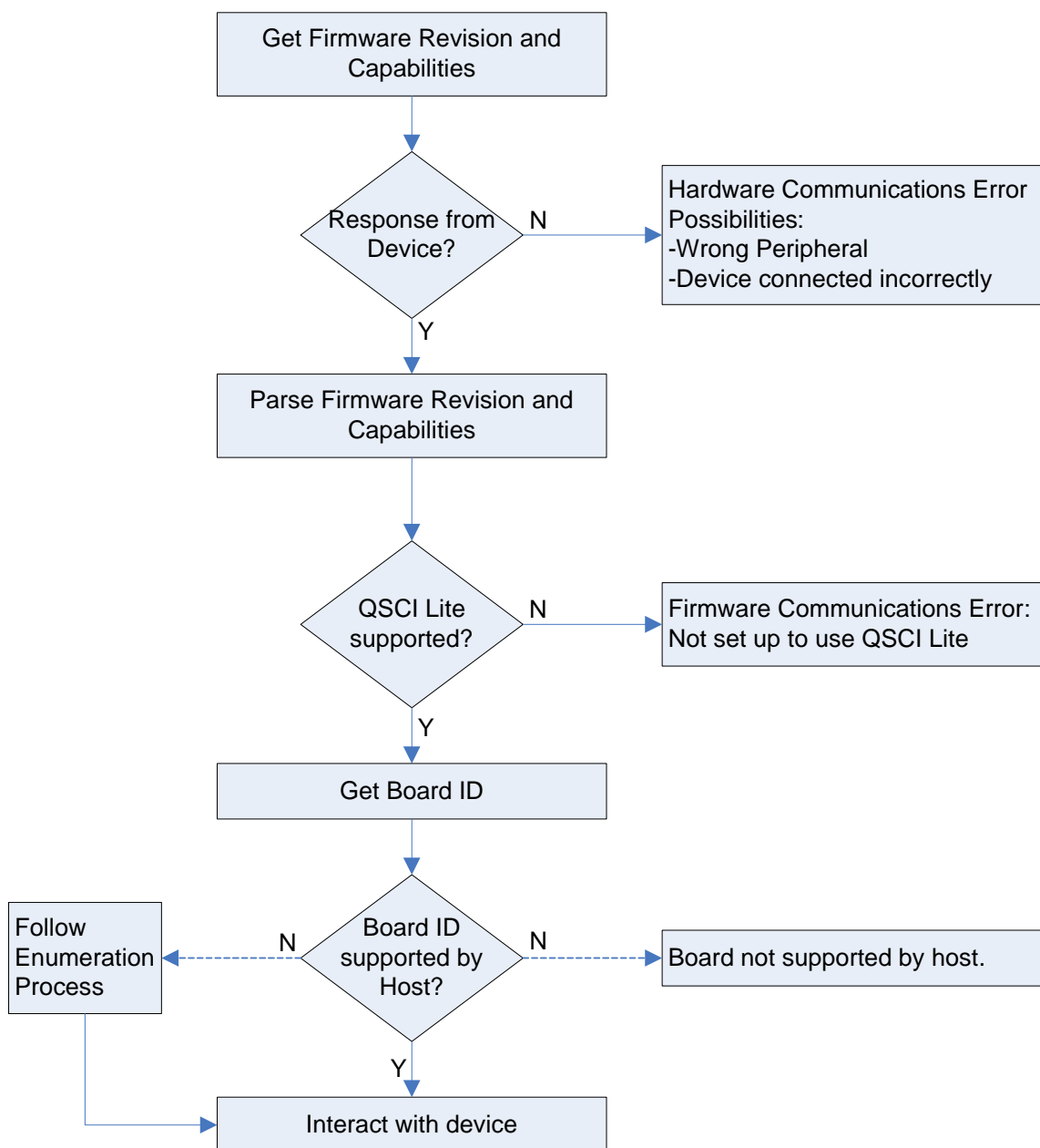


Figure 3. Host Initial Connection

3.2.1. Data Transfer

The host retrieves data from the device through the Read Address command (see "3.4.8. Read Address" on page 65). The host is allowed to read from two memory spaces, XDATA and CODE memory. Reads from CODE memory are limited to the configuration enumeration data and the calibration constants located on the NVCCA page. The device allows the host to read from all XDATA addresses.

The host sends data to be stored in the device's memory through the Write Address command (see "3.4.10. Write Address" on page 68). The host can write to two memory spaces, XDATA and CODE memory. Writes to CODE memory are restricted to the NVCCA page of flash. The device allows the host to write to all XDATA addresses. The host is responsible for not corrupting overlaid memory and to only write to the addresses supplied in the enumeration process.

3.2.2. QSCI Lite Enumeration Process

This section describes how to retrieve enumeration information from a device running QSCI Lite. The only way for the host to read QuickSense data from the device is through the Read Address command (see "3.4.8. Read Address" on page 65) where the host supplies an address and a memory space to read from. This means that QuickSense data must be located at specific addresses on the device and these addresses must be communicated to the host.

3.2.2.1. QuickSense Variable Addresses

Devices running QSCI Lite contain three arrays contiguously located in CODE memory starting at address 0x0100 which contain two-byte addresses to QuickSense variables. The variables in each table are located contiguously with each other so the length of each listed variable is the difference between the next valid address. The host reads these arrays through the Read Address command (see "3.4.8. Read Address" on page 65).

The first array is the CODE_CONTENTS[] array located at 0x0100. The array contains 11 two-byte addresses to configuration variables located in CODE memory. If a variable is not available due to device configurations, the entry for that array contains the address 0x0000.

Table 26. CODE_CONTENTS[] Array

Index	Variable Name	Description
0	QS_GroupChannelAssignmentTable[] ¹	Contains a pointer to a group's channel list
1	QS_1DPadCapabilities[] ²	Contains a pointer to each 1D Pad's capability array
2	QS_1DPadChannels[] ²	Contains a pointer to each 1D Pad's channel list
3	QS_2DPadCapabilities[] ³	Contains a pointer to each 2D Pad's capability array
4	QS_2DPadChannels[] ³	Contains a pointer to each 2D Pad's channel list
5	QS_3DPadCapabilities[] ⁴	Contains a pointer to each 3D Pad's capability array
6	QS_3DPadChannels[] ⁴	Contains a pointer to each 3D Pad's channel list
7	QS_ChannelInfo[]	Contains the properties for each channel
8	QS_IRConfig[] ⁵	Contains the IR mode for each IR channel
9	QS_GroupType[] ¹	Contains the type of each group
10	CODE_END_ADDRESS	Contains the end address of the configuration space

Notes:

1. Not available if GROUP_VALUE_SAVE is DISABLED or if there are zero groups in the system
2. Not available if there are zero 1D pads in the system
3. Not available if there are zero 2D pads in the system
4. Not available if there are zero 3D pads in the system
5. Not available if there are zero IR channels in the system

The second array is the NVCCA_CONTENTS[] array located at 0x0116. It contains four 2-byte addresses to calibration variables located on the NVCCA page. If a variable is not available due to device configurations, the entry for that array is 0x0000.

Table 27. NVCCA Contents[] Array

Index	Variable Name	Description
0	QS_Threshold[] ¹	Contains threshold percentages for each channel
1	QS_RuntimeBaselineMagnitude[]	Contains the Runtime Baseline Magnitude for each channel
2	QS_BaselineUpdateRate	The Baseline Update Rate for the system
3	NVCCA_END_ADDRESS	The end address of the NVCCA variables
Notes:		
1. Not available if CHANNEL_THRESHOLD is DISABLED		

The final array is the XDATA_CONTENTS[] array located at 0x11E. It contains 17 two-byte addresses to runtime variables located in XDATA memory. If a variable is not available due to device configurations, the address for that variable is the same as the address in the next entry. For instance, in a system with no groups but contains a 1D pad, the entry for QS_GroupValue[] will be the same as the entry for QS_1DPadPoints[].

Table 28. XDATA_CONTENTS[] Array

Index	Variable Name	Description
0	QS_ChannelValue[]	Contains the measured values for each channel
1	QS_RuntimeBaseline[]	Contains runtime active and inactive baseline for each channel
2	QS_GroupValue[] ^{1, 2}	Contains the interpreted value for each group
3	QS_1DPadPoint[] ³	Contains 1D location(s) for each 1D pad
4	QS_2DPadPoints[] ⁴	Contains 2D location(s) for each 2D pad
5	QS_3DPadPoints[] ⁵	Contains 3D location(s) for each 3D pad
6	QS_GenericData[] ⁶	Contains a value for each generic data element
7	QS_ThresholdState[] ⁷	Bit array for the current threshold state for each channel
8	QS_ChannelCalibration[]	Bit array for the calibration state of each channel
9	QS_ChannelProcessEnable[]	Bit array for the process enable of each channel
10	QS_ThresholdProcessEnable[] ⁷	Bit array for the process enable for thresholds of each channel
11	QS_GroupProcessEnable[] ²	Bit array for the process enable of each group
12	QS_1DProcessEnable[] ³	Bit array for the process enable of each 1D pad
13	QS_2DProcessEnable[] ⁴	Bit array for the process enable of each 2D pad
14	QS_3DProcessEnable ⁵	Bit array for the process enable of each 3D pad
15	QS_FlashKey[] ⁸	Contains the two flash keys to write to/erase flash pages
16	XDATA_END_ADDRESS	The end address for XDATA variables
Notes:		
1. Not available if GROUP_VALUE_SAVE is DISABLED.		
2. Not available if there are zero groups in the system.		
3. Not available if there are zero 1D pads in the system.		
4. Not available if there are zero 2D pads in the system.		
5. Not available if there are zero 3D pads in the system.		
6. Not available if there are zero generic data elements in the system.		
7. Not available if CHANNEL_THRESHOLD is DISABLED.		
8. Not available if FLASH_MODE is READ_ONLY.		

3.2.2.2. Obtaining System Elements

Once these three arrays are read, the host knows the location of all relevant QuickSense variables. The host can then figure out the number of system elements through the size of some arrays.

The number of channels in the system is the same as the length of `QS_ChannelInfo[]`. The number of groups in the system is the length of `QS_GroupType[]`. The number of nD Pads is the size of `QS_nDPadCapabilities[]` divided by two, since each entry in the array is a two-byte pointer. The number of generic data elements in the system is the length of `QS_GenericData[]` divided by two since each generic data element is a two-byte value.

3.2.2.3. Reading Channel List Information

`QS_GroupChannelAssignmentTable[]` and `QS_nDPadChannels[]` are arrays of pointers which point to a channel list array for each group and nD pad. Reading the channel list for an element is a three step process. For more information about channel list arrays, please refer to AN366.

First, the host must get the address of the channel list to read from `QS_GroupChannelAssignmentTable[]` or `QS_nDPadChannels[]`. Second, the host must get the length of the channel list by reading the byte at that address. Finally, the host can read the rest of channel list.

Figure 4 describes an example of reading group 0's channel list.

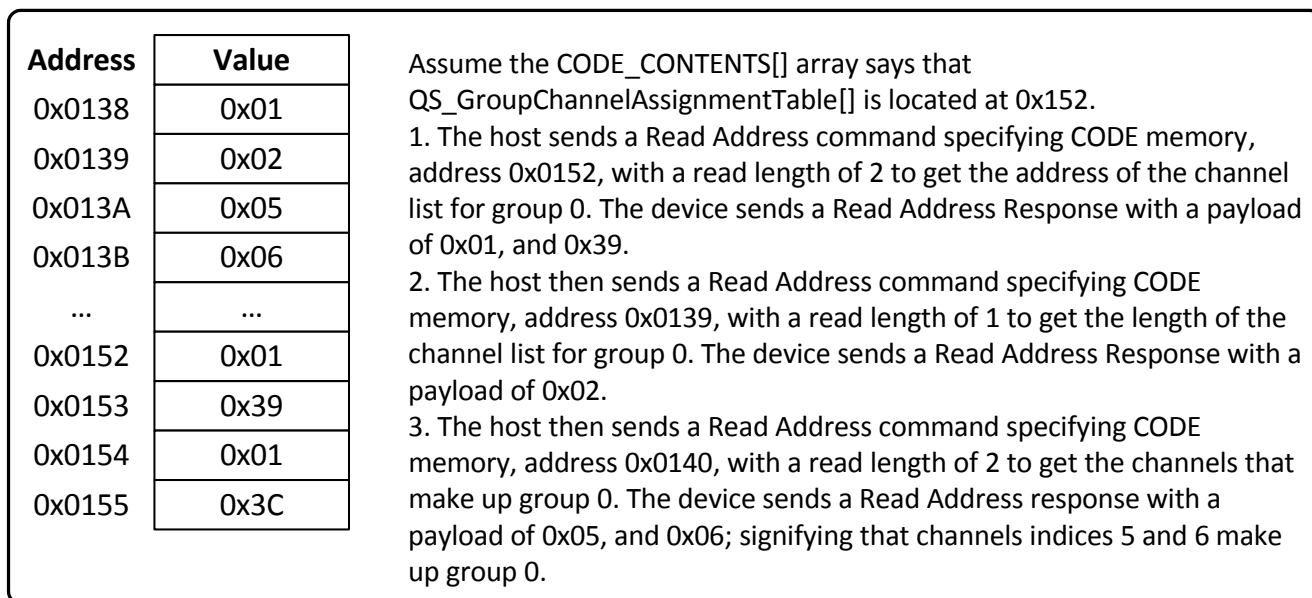


Figure 4. Reading a Channel List

3.2.2.4. Reading QS_nDPadCapabilities

`QS_nDPadCapabilities[]` are arrays of pointers which point to a nD pad capability array. The capability array size for each n-dimension is different. For 1D pads, there are 10 bytes per array. For 2D pads, there are 14 bytes per array. For 3D pads there are 18 bytes per array. Reading the capability array for a nD pad is a two step process.

First, the host must get the address for the capability array from `QS_nDPadCapabilities[]`. The host then uses that address to read the capability array. The steps are similar to reading a channel list described in Figure 4 but the host can skip step 2 in those directions because the length is fixed for each nD pad capability array.

3.2.3. QSCI Lite Hardware Protocols

The following section describes QSCI Lite communication with the UART, SPI and SMBUS protocols. Each section explains the process to send commands and receive responses from the device. A Get Board ID transaction is also provided in each section.

3.2.3.1. UART

Figure 5 describes how a host communicates to a device using the UART protocol. UART is unique in that it is not master driven like SPI or SMBUS. This allows the device to drive information to the host. In its simplest form, the UART transmits the command and then waits until the device responds.

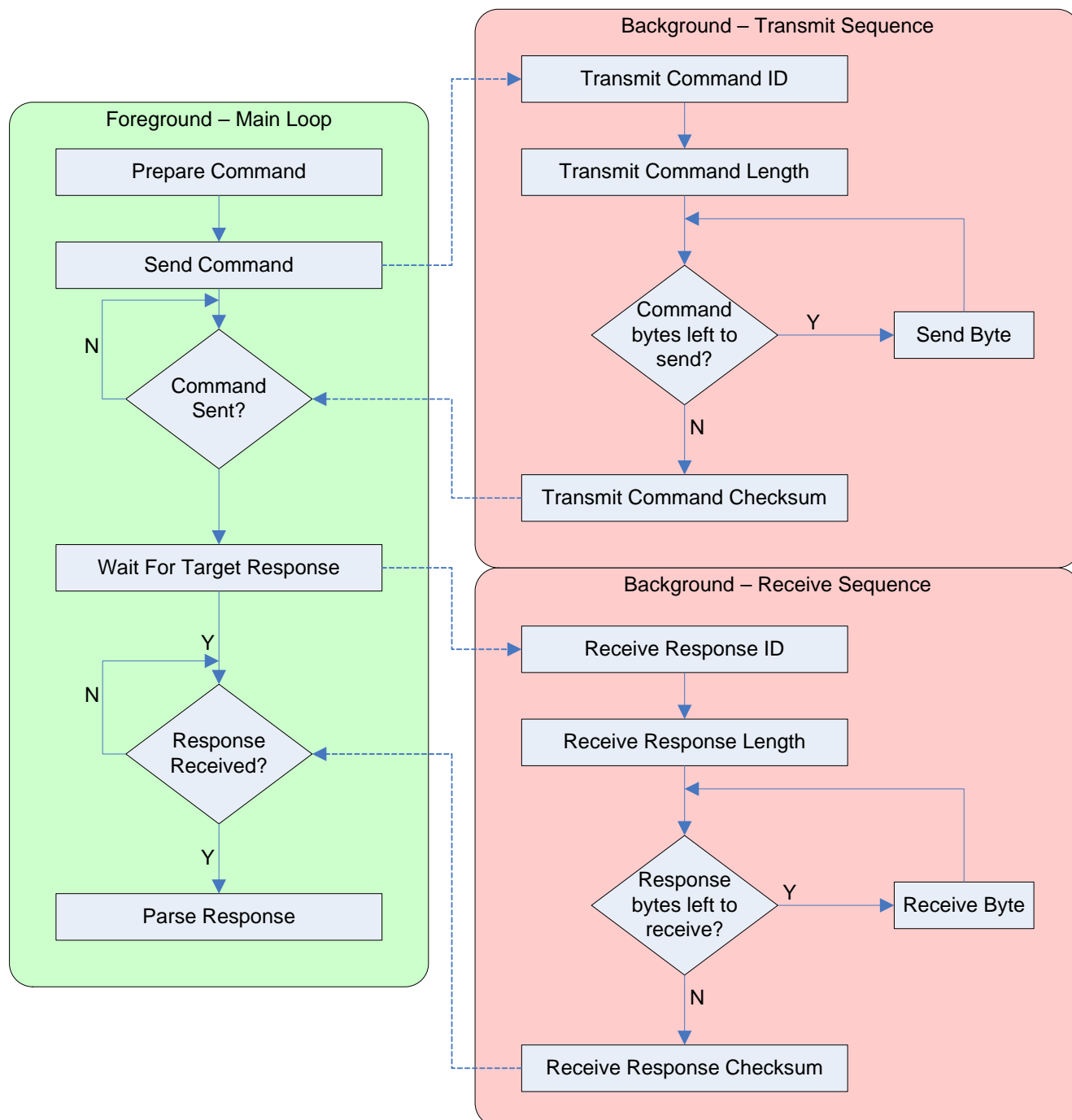


Figure 5. UART Host Sequence

AN494

A UART Get Board ID command is described in the Figure 6. The host prepares the Get Board ID command described in “3.4.3. Get Board ID.” The host sends each byte through the interface. Once the command is received by the device, it parses the command and prepares an appropriate response. The host waits (or handles other activities) while the device sends the response back. Once the response is sent from the device, the host can parse the response and the command transaction is complete.

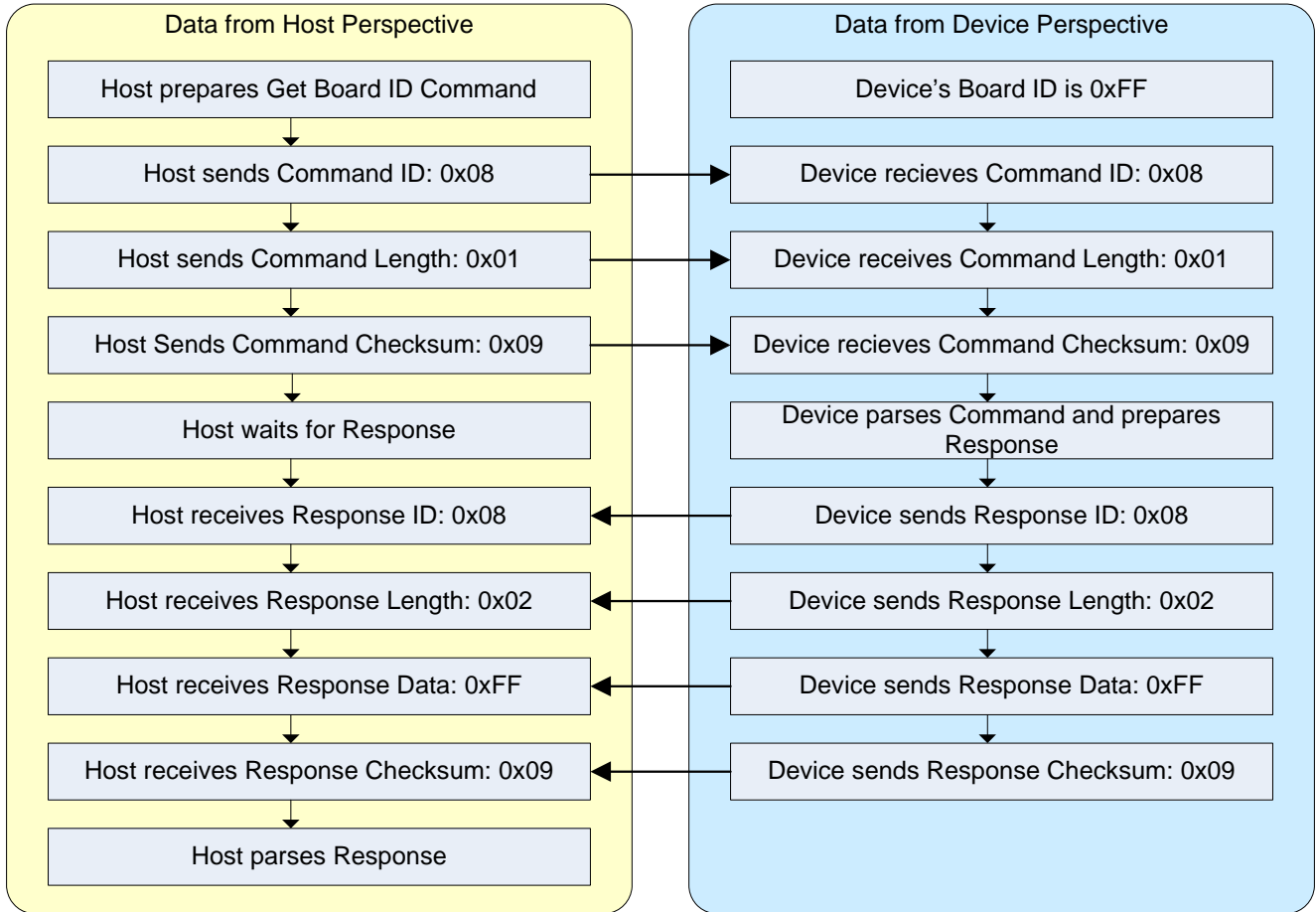


Figure 6. UART Get Board ID

3.2.3.2. SPI

SPI behaves differently compared to the other protocols because whenever the host shifts data out, it receives data. For the host to read the response from the device, the host must send dummy bytes to get that data. This causes some complications where the device could be processing the command while the host is trying to get the response. Therefore, the device sends an invalid byte(0xA5) while the response is being prepared. The host should continue probing the device until it receives the Response ID and from there retrieves the rest of the response.

For Flash Write and Erase NVCCA commands, the host should wait, 2 ms and 50 ms respectively, while the device is operating on its flash. During this period of time, the device has disabled its interrupts and does not receive the dummy bytes sent by the host.

In the following figure, the dummy bytes sent while waiting for the response ID is done in the foreground because the host could be doing other activities while waiting for the device to finish preparing its response.

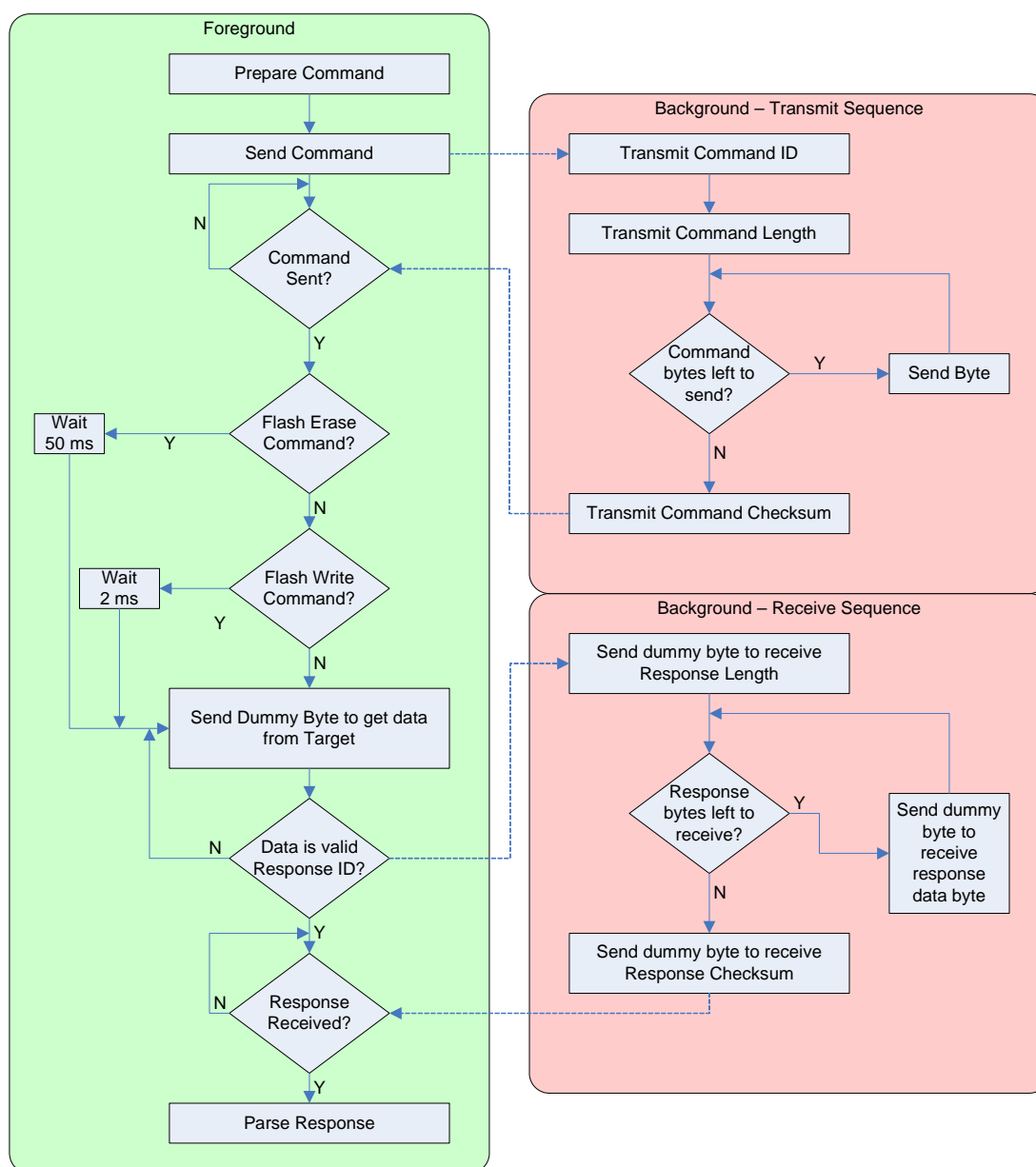


Figure 7. SPI Host Sequence

AN494

An SPI Get Board ID command is described in the following figure. The host prepares the Get Board ID command described in “3.4.3. Get Board ID.” The host sends each byte through the interface. Once the command is received by the device, it parses the command and prepares an appropriate response. The host ignores the bytes shifted back in through the MISO.

The host sends dummy bytes to see if the device has prepared the response yet. Once the host receives a valid response ID, the host sends enough dummy bytes to receive the rest of the response.

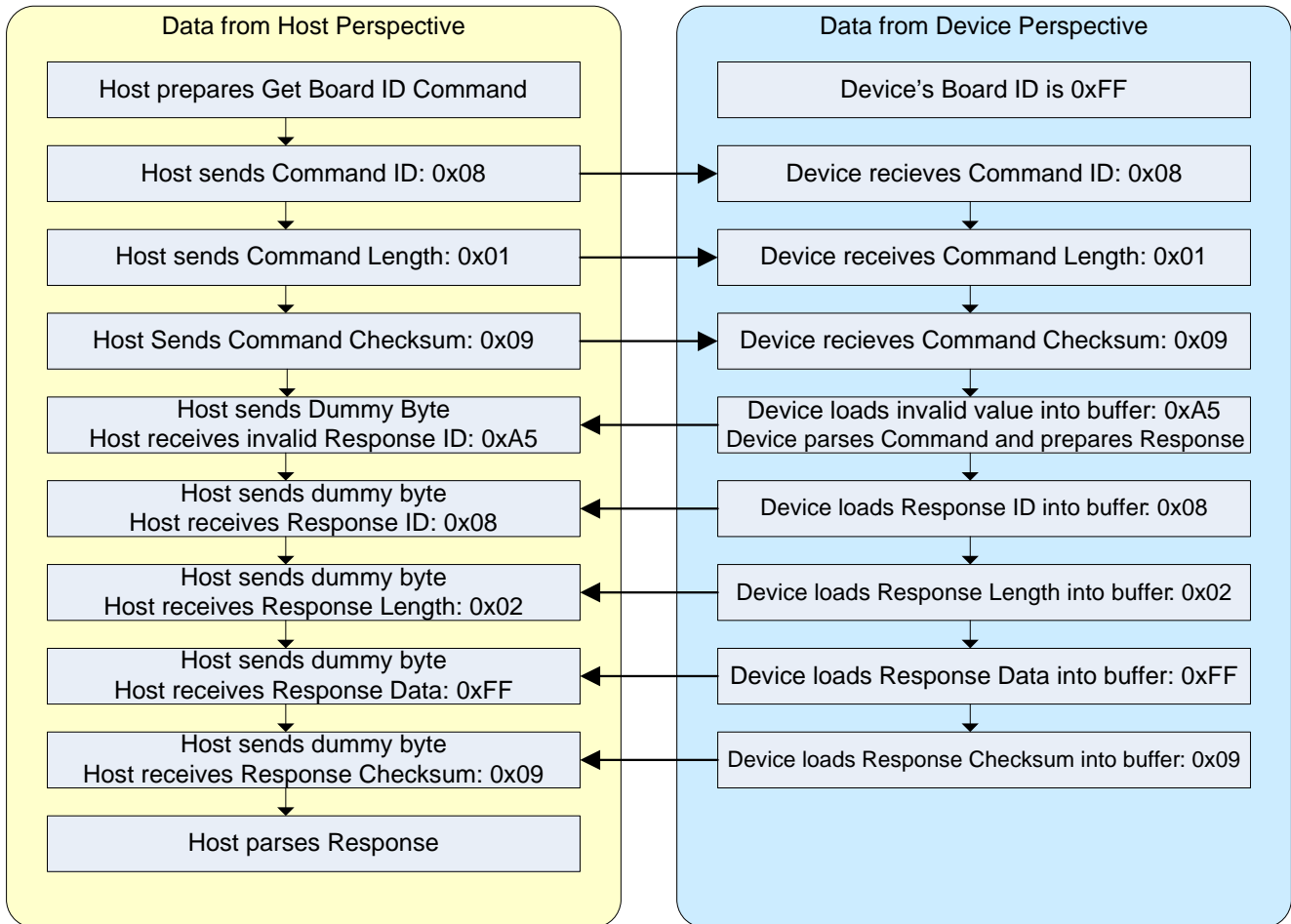


Figure 8. SPI GetBoard ID

3.2.3.3. SMBUS

SMBUS is another master driven protocol where the host specifies if the following transaction is a read sequence or a write sequence. The host first sends the SMBUS address of the device; the device should ACK or the device is not connected to the bus. The host starts a write sequence and sends the command.

Once the command is sent to the device, the host initiates a read transaction to read the response. If the command is either a Flash Write or Erase NVCCA, the host should wait, 2 ms and 50 ms respectively, before starting the read transaction. This is because interrupts are disabled on the device while it writes to flash or erases flash pages which could cause the device to miss the read transaction sent from the host.

Once the host starts the read transaction, the device sends values while the host ACKs. The device could be processing the command still and it sends an invalid value (0xA5) while the response is being prepared. The host should ACK when receiving the invalid values and wait until it receives the Response ID. Once the host receives the Response ID, it should ACK until it receives the Response Checksum and then NACK which signifies to the device that the read transaction is complete. The Figure 9 describes this process.

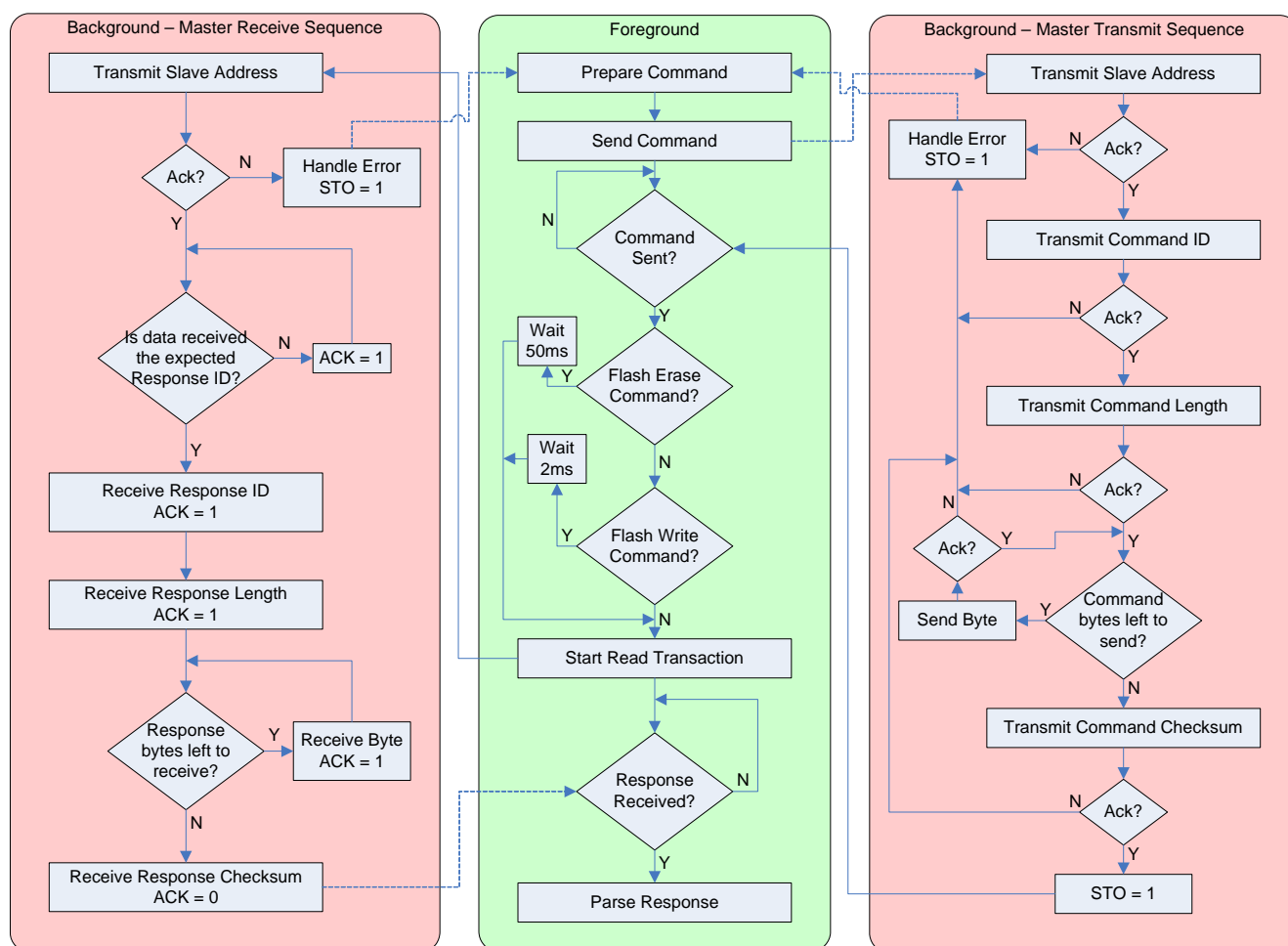


Figure 9. SMBus Host Sequence

AN494

Figure 10 describes an SMBUS Get Board ID command sequence. The host prepares the Get Board ID command described in “3.5.3. System Configuration.” The host initiates a write transaction by sending the SMBUS device address with the R/W bit set to Write. The host then sends each command byte to the device and sends the STOP bit after the checksum to signal the end of the write transaction.

The host then initiates a read transaction by sending the SMBUS device address with the R/W bit set to Read. The host then waits until the device sends data, and ACKs until the Response ID is received. The host then ACKs and receives the response data. Once the Response Checksum is received, the host NACKs, therefore signaling the end of the read transaction.

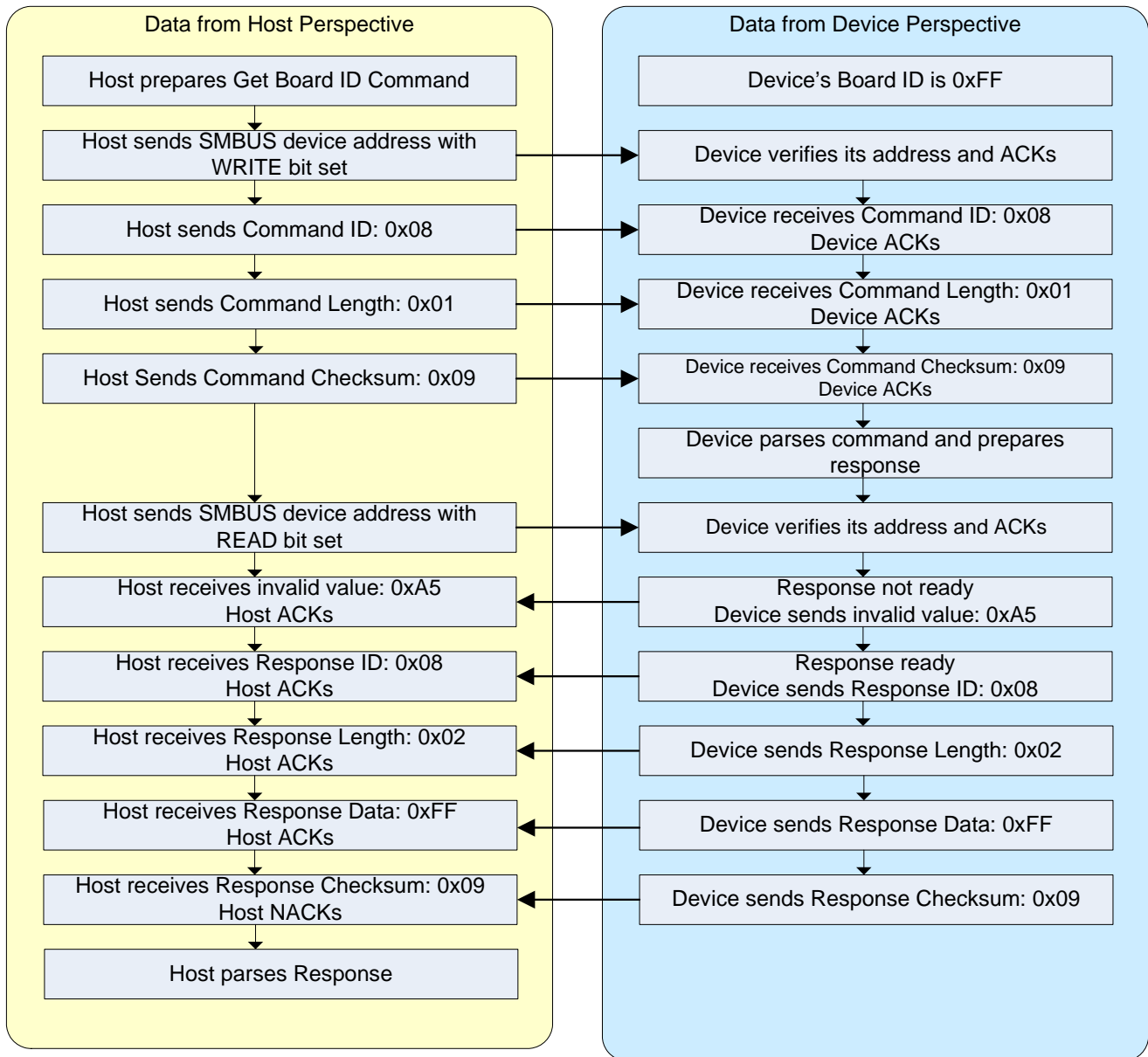


Figure 10. SMBUS GetBoard ID

3.3. QSCI Lite Header Bytes

The two header bytes for QSCI Lite differ from the ones defined in QSCI. Header Byte 0 contains the 8-bit command ID and Header Byte 1 contains the 8-bit packet length. This allows the processing of the bytes to be less complicated compared to QSCI. This limits packet lengths to 254 bytes.

Header Byte 0								Header Byte 1							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Command/Response ID [7:0]								Length Field[7:0]							

Figure 11. QSCI Lite Header Bytes

3.4. QSCI Lite Commands and Responses

This section describes the properties and payload structure for each command and response in QSCI Lite.

3.4.1. Get Firmware Revision and Capabilities

Command ID: 0x00

Description: This command asks the device for its firmware revision and capabilities array.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x00							
1	Packet Length	0x01							
2	Checksum	0x01							

Device Response:

Responses	Situation
Firmware Revision and Capabilities Response	Normal Operation

AN494

3.4.2. Firmware Revision and Capabilities Response

Response ID: 0x00

Description: This response returns the device's firmware revision and capabilities array. The revision values are BCD encoded. For example, if the QSCI Major Revision byte is 0x02 and the QSCI Minor Revision byte is 0x30, the QSCI revision is v02.30. For more information about each byte, refer to AN366.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x00							
1	Packet Length	0x0B							
2	QSCI Major Revision	variable							
3	QSCI Minor Revision	variable							
4	Baselining Method	0x01							
5	Command Availability	0	0	0	0	0	QSCI Mode	Flash Mode	Enum Mode
							Table 29	Table 30	Table 31
6	Transfer Options Supported	0	0	Transfer Types Supported		Transfer Modes Supported			
				Table 32					
7	QuickSense API Major Revision	variable							
8	QuickSense API Minor Revision	variable							
9	Application Major Revision	variable							
10	Application Minor Revision	variable							
11	RX_BUFFER_SIZE	variable							
12	Checksum	variable							

Table 29. QSCI Mode Options

QSCI Mode	Value	Description
QSCI	0	The serial interface used follows the QSCI protocol
QSCI_LITE	1	The serial interface used follows the QSCI Lite protocol

Table 30. Flash Mode Options

Flash Mode	Value	Description
READ_WRITE	0	The host can read and write into NVCCA flash memory
READ_ONLY	1	The host can only read from NVCCA flash memory

Table 31. Enumeration Mode Options

Enumeration Mode	Value	Description
DISABLED	0	The Start Enumeration command is not available
ENABLED	1	The Start Enumeration command is available

Table 32. Transfer Options Supported Bits

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
N/A	N/A	SelectedAndUpdated	Selected	N/A	On-Demand	On-Update	Periodic
A value of 0 means the option is not supported. A value of 1 means the option is supported.							

3.4.3. Get Board ID

Command ID: 0x08

Description: This command asks the device for its board ID.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x08							
1	Packet Length	0x01							
2	Checksum	0x09							

Device Response:

Responses	Situation
Board ID Response	Normal Operation

AN494

3.4.4. Board ID Response

Response ID: 0x08

Description: In this response, the device sends its board ID.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x08							
1	Packet Length	0x02							
2	Board Identifier	variable							
3	Checksum	variable							

3.4.5. Reset Device

Command ID: 0x10

Description: This command tells the device to perform a soft reset.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x10							
1	Packet Length	0x01							
2	Checksum	0x11							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation

3.4.6. General Response

Response ID: 0x10

Description: This response is sent to acknowledge that a command has executed on the device. This response provides an error code about the command that was processed.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x10							
1	Packet Length	0x02							
2	Error Code	variable							
		Table 33							
3	Checksum	variable							

Table 33. Error and Warning Codes

Error Code	Description
0x00	No error or warning; Command completed successfully.
0x01	ERROR: Bad Checksum
0x02	ERROR: Unknown Command
0xFF	WARNING: Bad Address
0xFD	WARNING: Packet Timeout
0xFC	WARNING: Bad Request
0xFB	WARNING: Bad Threshold

3.4.7. Erase NVCCA

Command ID: 0x60

Description: This command tells the device to erase the NVCCA page(s) of flash. The host must set QS_FlashKey[] before using this command.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x60							
1	Packet Length	0x01							
2	Checksum	0x61							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Request Warning	If the device's FLASH_MODE is READ_ONLY

3.4.8. Read Address

Command ID: 0x78

Description: This command tells the device to transmit X bytes from either XDATA or CODE memory starting at a particular location. From CODE memory, the host can only read from the configuration area or NVCCA memory. The host has full read access to XDATA. The host cannot request more than 252 bytes in one packet.

AN494

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x78							
1	Packet Length	0x05							
2	Memory Space	0	0	0	0	0	0	0	Memory Space Table 34
3	Address High Byte	variable							
4	Address Low Byte	variable							
5	Num bytes to read	variable							
6	Checksum	variable							

Table 34. Memory Space Options

Memory Space	Value	Description
XDATA	0	Address is a location in XRAM memory
CODE	1	Address is a location in flash memory

Device Response:

Responses	Situation
Read Address Response	Normal Operation
General Response—Bad Address Warning	Address requested is outside of readable CODE memory

3.4.9. Read Address Response

Response ID: 0x78

Description: This response transmits X requested bytes from a specific memory space and address. This response echoes the memory space and address requested.

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x78							
1	Packet Length	variable							
2	Memory Space	0	0	0	0	0	0	0	Memory Space Table 34
3	Address High Byte	variable							
4	Address Low Byte	variable							
5	Byte at Address + 0	variable							
6	Byte at Address + 1	variable							
...							
N	Byte at Address + X	variable							
N+1	Checksum	variable							

AN494

3.4.10. Write Address

Command ID: 0x80

Description: This command tells the device to write the following X bytes starting at a specific address in a memory space. The size of this command must not exceed the RX_BUFFER_SIZE of the device. If the host is writing to CODE memory, the host must set QS_FlashKey[].

Packet Structure and Options:

Byte	Byte Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0x80							
1	Packet Length	variable							
2	Memory Space	0	0	0	0	0	0	0	Memory Space Table 34
3	Address High Byte	variable							
4	Address Low Byte	variable							
5	Write to Address + 0	variable							
6	Write to Address + 1	variable							
...							
N	Write to Address + X	variable							
N+1	Checksum	variable							

Device Response:

Responses	Situation
General Response—No Error	Normal Operation
General Response—Bad Address Warning	Address requested is not on NVCCA page
	If device's FLASH_MODE is READ_ONLY and memory space is CODE.

3.5. QSCI Lite Host Example User Guide

This section describes the QSCI Tester example firmware. The tester firmware is installed with QuickSense Studio, which can be downloaded from www.silabs.com/QuickSense. The default installation path for the tester firmware is C:\SiLabs\MCU\QuickSense_Studio\Firmware\QSCI_Host_Example. The firmware provides an example of how a host connects to a device, enumerates the device to find its properties, and uses the data retrievable through the QSCI Lite interface. In this case, the data retrieved by the host is displayed via the HyperTerminal program.

This firmware code example uses the F340TB. All of the jumpers on J12 should be connected. The example uses the following pins to communicate to a device.

UART:

P0.0 - UART1 TX

P0.1 - UART RX

SPI:

P0.0 - SCK

P0.1 - MISO

P0.2 -MOSI

P0.3 - NSS

SMBUS:

P0.0 - SDA

P0.1 - SCL

P2.0 is used as a hardware switch and is used to exit any of the data streaming menu options.

The terminal should be set up for 230400 baud, 8 bits, 1 stop bit, no parity bit.

3.5.1. Initial Connection

When the QSCI Tester connects to the device, it follows the process outlined in “3.2. QSCI Lite Host Interaction.”. It retrieves the Firmware Revision and Capabilities array, the board ID and enumerates the device as described in Section 3.2.2 QSCI Lite Enumeration Process.

Once successfully connected, the terminal should look like “Figure 12. Tester Startup Prompt.” It prompts the user to verify the board connected to the host. This is to verify that the device sent the correct Board ID.

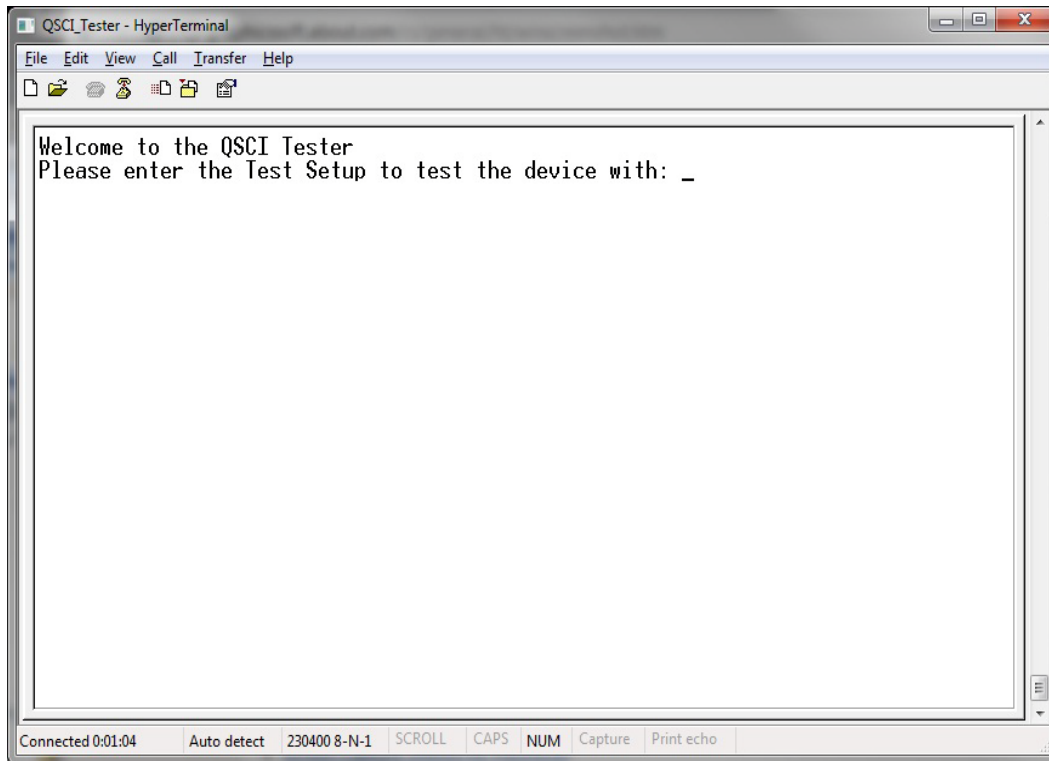


Figure 12. Tester Startup Prompt

3.5.2. QSCI Tester Menu

Once the Board ID is verified to be correct, the main menu of the tester is displayed on the terminal.

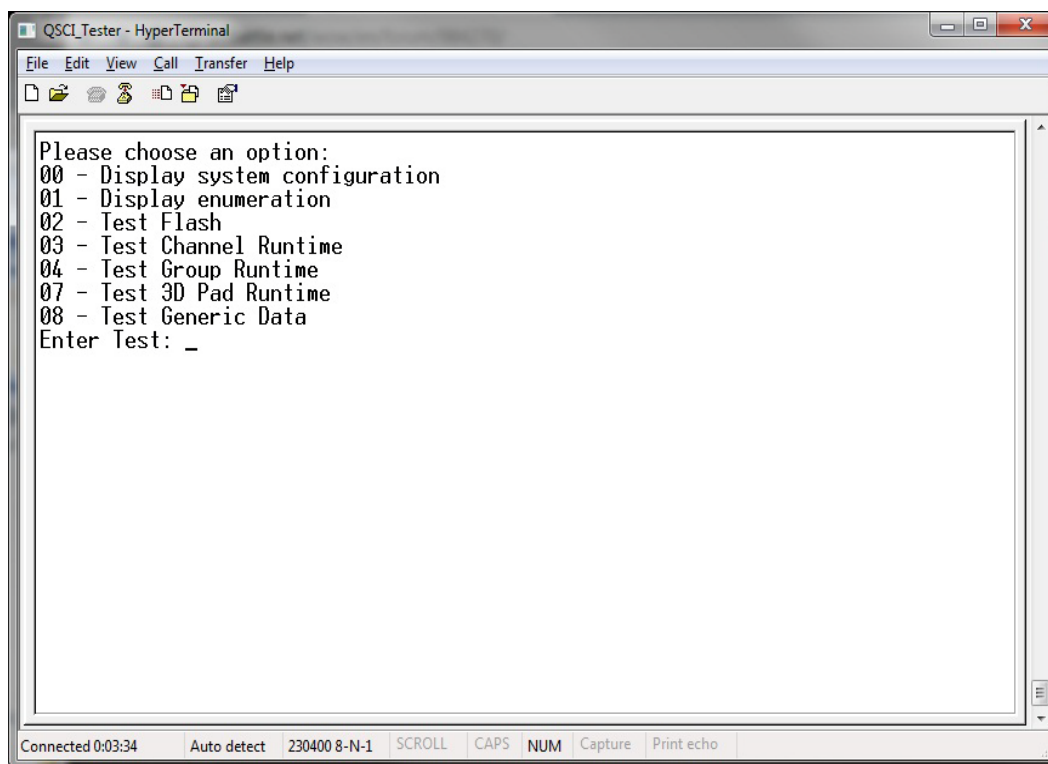


Figure 13. Tester Main Menu

The main menu allows the user to interact with the device in many ways. First, the host can display the configuration of the device in text via option "00 - Display system configuration" and "01 - Display enumeration". In "02 - Test Flash", the host reads, writes, and erases flash on the device in an automated test. The final options stream data from the device and display it for the user to see. The user can then interact with the device and see how the data changes.

3.5.3. System Configuration

The system configuration menu option allows the user to see the contents of the device's Firmware Revision and Capabilities array in human readable format. It displays the version numbers of the firmware on the device, the device configurations, and the system elements found on the device as derived in Section 3.2.2.2 Obtaining System Elements.

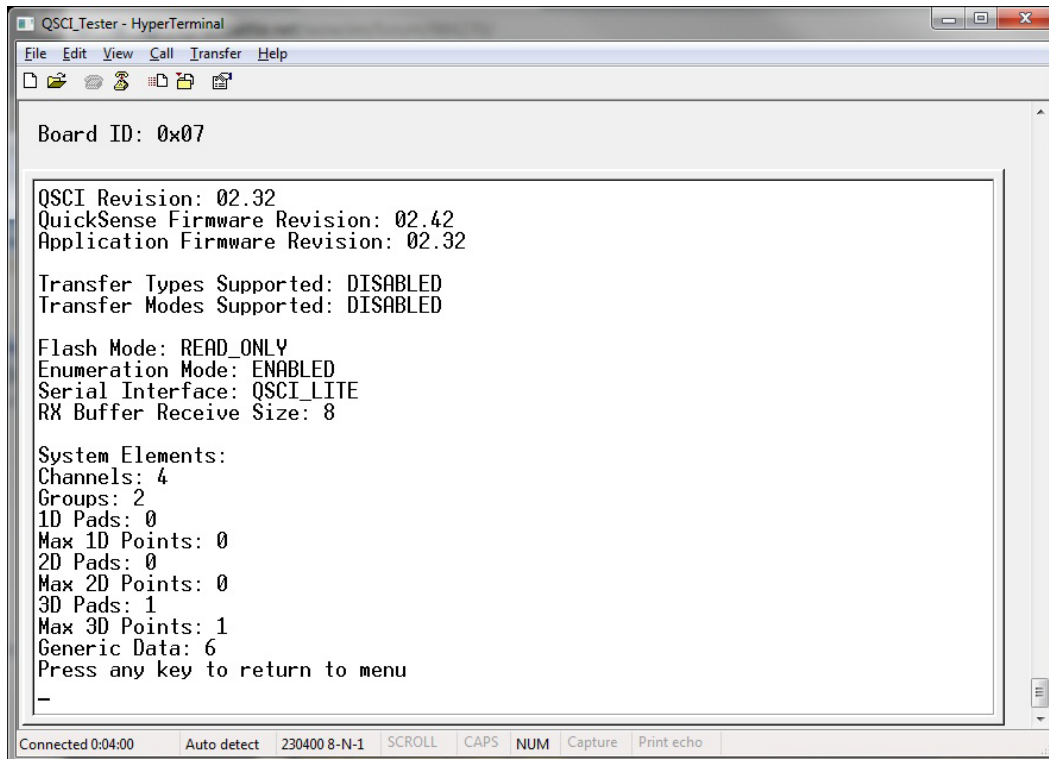
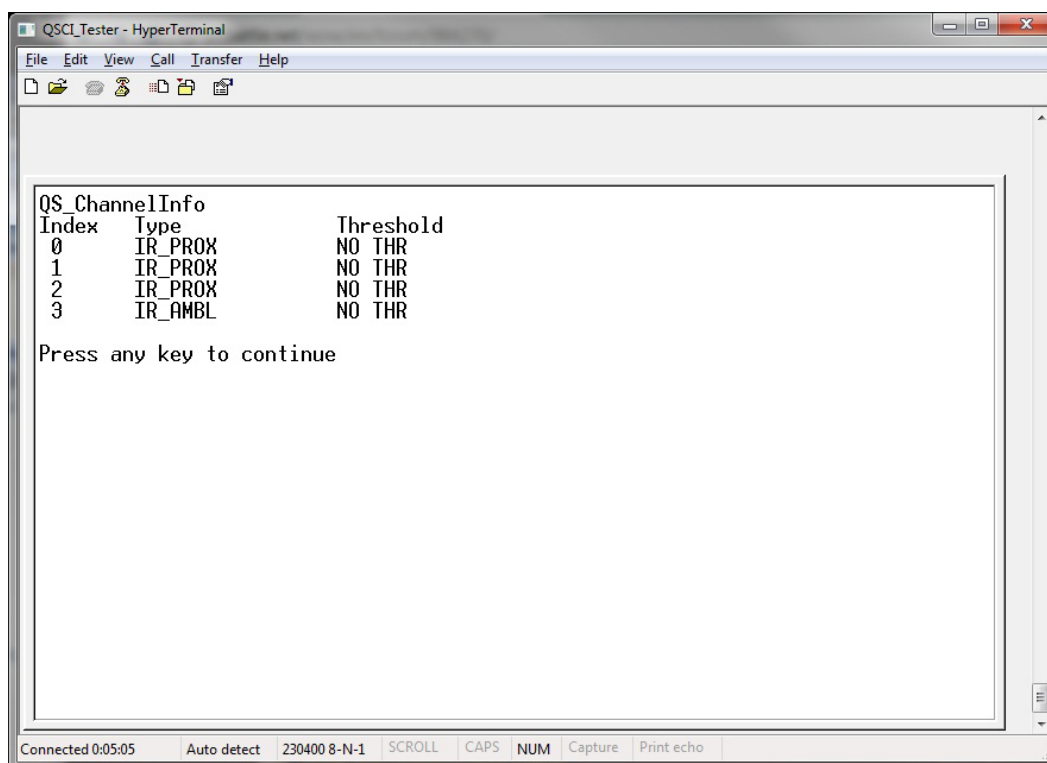


Figure 14. Tester Showing Retrieved Device Configuration

3.5.4. Device Enumeration

The display enumeration menu option displays all of the configuration arrays located on the device. If a configuration array does not exist, then the host checks to see that the device prompts the correct error. The host translates the enumeration data into text for the user to verify.



```
QS_ChannelInfo
Index  Type          Threshold
0      IR_PROX         NO THR
1      IR_PROX         NO THR
2      IR_PROX         NO THR
3      IR_AMBL         NO THR

Press any key to continue
```

Figure 15. Retrieved Enumeration Information

3.5.5. Flash Test

In the test flash menu option, the host tries to read, write, and erase the NVCCA page on the device. If the device's FLASH_MODE is READ_ONLY, then the host verifies that the device sends the correct warnings.

If the device's FLASH_MODE is READ_WRITE, then the host saves the device's NVCCA data. It then erases the NVCCA page on the device and verifies that the page has been erased. The host writes in arbitrary values to the NVCCA page and verifies that the data is written. Once the host verifies the operation writing to flash, the host restores the initial NVCCA values back to the device.

A successful test looks like the following figure.

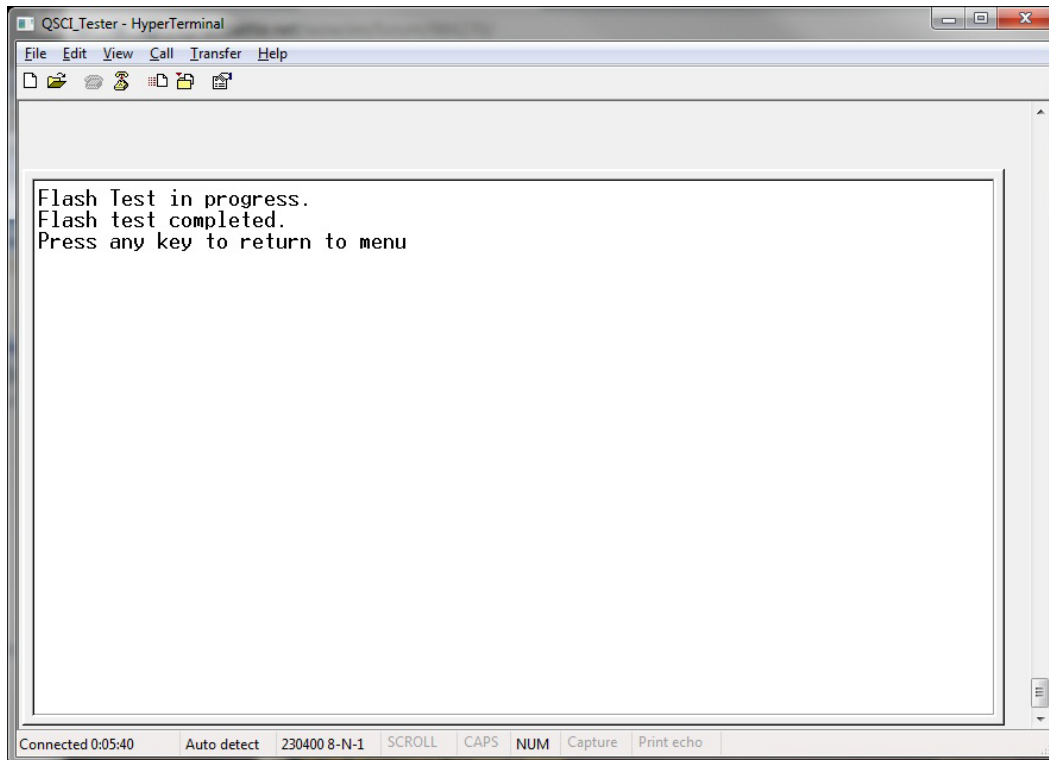
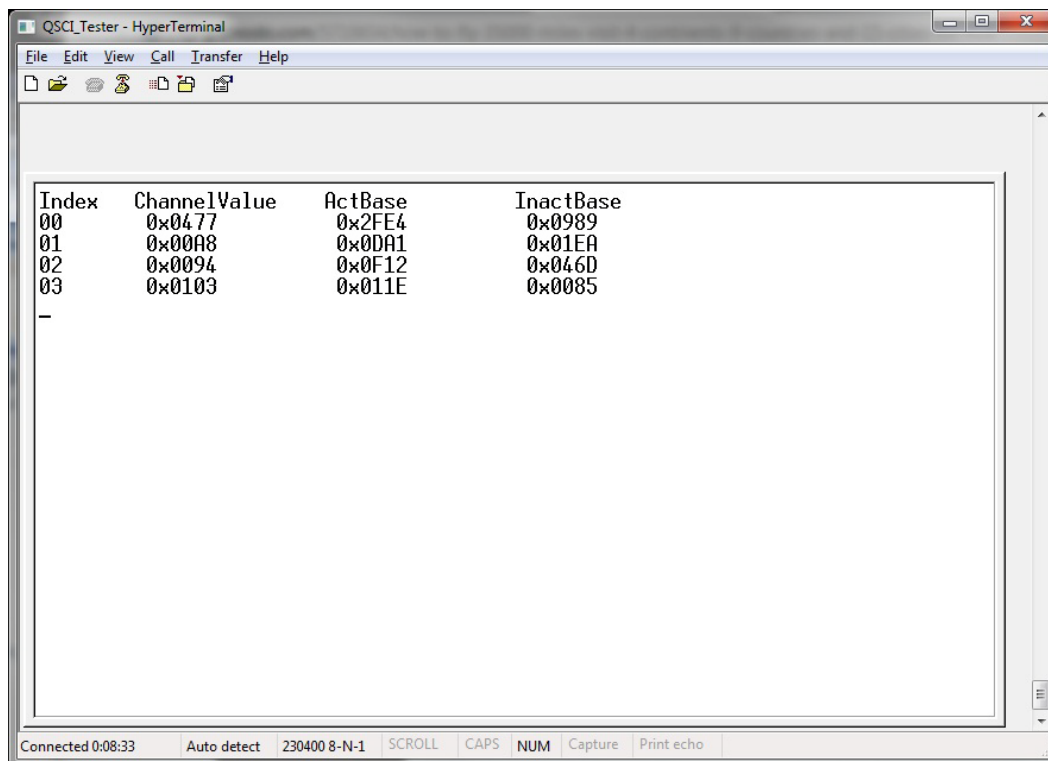


Figure 16. Flash Test Output

3.5.6. Channel Data

In channel data menu option, the host streams channel data from the device and displays it on the terminal every 500 ms. This includes raw channel values, runtime inactive baseline, runtime active baseline, and the threshold state for each channel. To stop streaming, press and hold the P2.0 hardware button until streaming stops and a prompt appears.



The screenshot shows a HyperTerminal window titled "QSCI_Tester - HyperTerminal". The window contains a table of channel data with four columns: Index, ChannelValue, ActBase, and InactBase. The data is as follows:

Index	ChannelValue	ActBase	InactBase
00	0x0477	0x2FE4	0x0989
01	0x00A8	0x0DA1	0x01EA
02	0x0094	0x0F12	0x046D
03	0x0103	0x011E	0x0085

The status bar at the bottom of the window shows "Connected 0:08:33", "Auto detect", "230400 8-N-1", "SCROLL", "CAPS", "NUM", "Capture", and "Print echo".

Figure 17. Retrieved Channel Information

3.5.7. Group Data

In the group data menu option, the host streams group data from the device and displays it onto the terminal every 500 ms. To stop streaming, press and hold the P2.0 hardware button until streaming stops and a prompt appears.

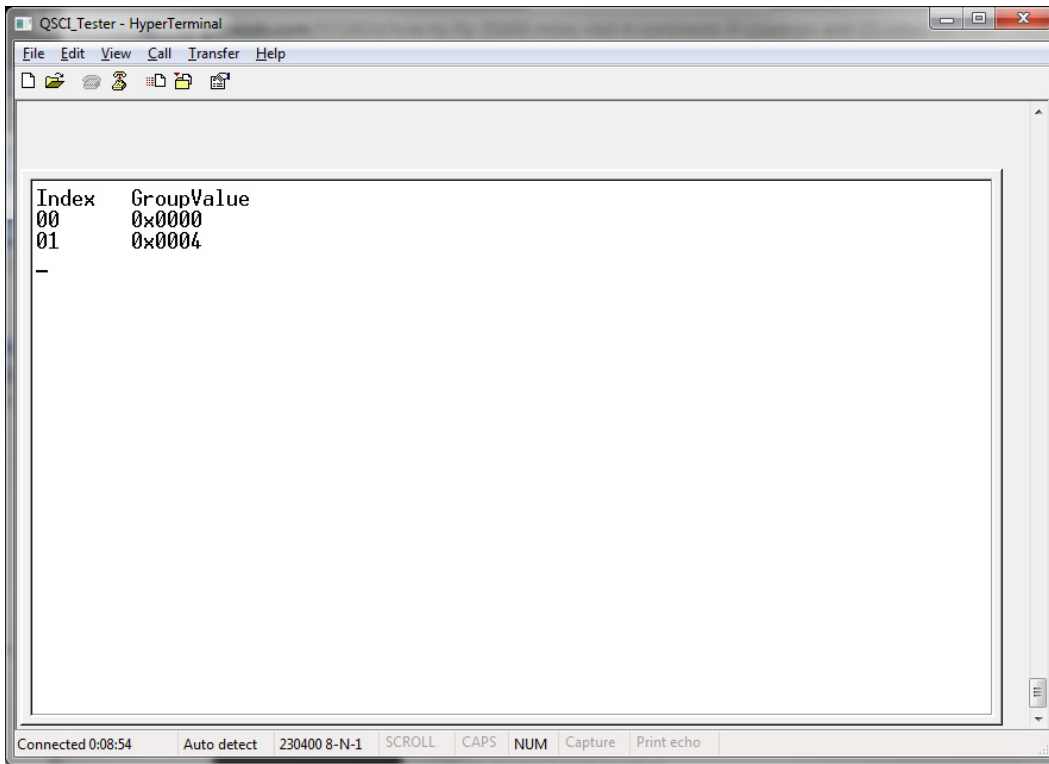


Figure 18. Retrieved Group Information

3.5.8. nD Pad Data

The nD pad menu options are only shown if the device supports a specific dimension. In this example, the host is connected to the Si1120EK, which has one 3D pad. For this option, the user is prompted to identify which pad index to stream data from.

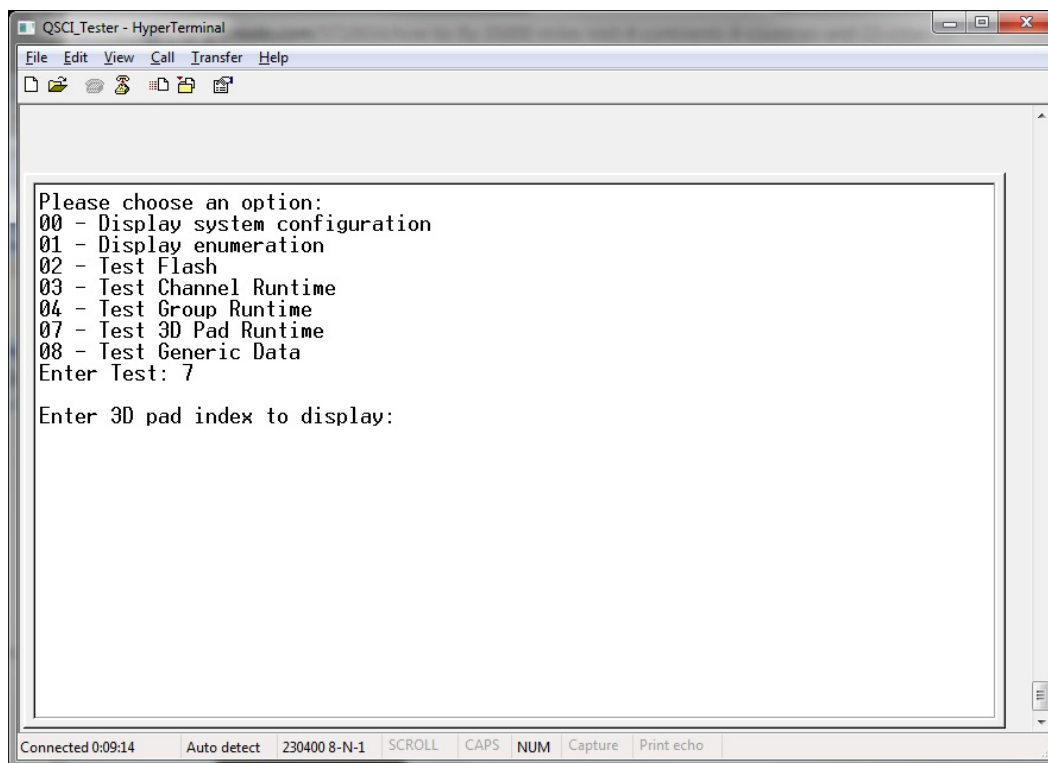


Figure 19. nD Pad Test Menu

Once the user chooses which pad index to display, the host retrieves coordinate data about that specific pad and displays it to the terminal every 500ms. It also displays multiple points if the pad supports more than one point (i.e., Multi-touch situations). To stop streaming, press and hold the P2.0 hardware button until streaming stops and a prompt appears.

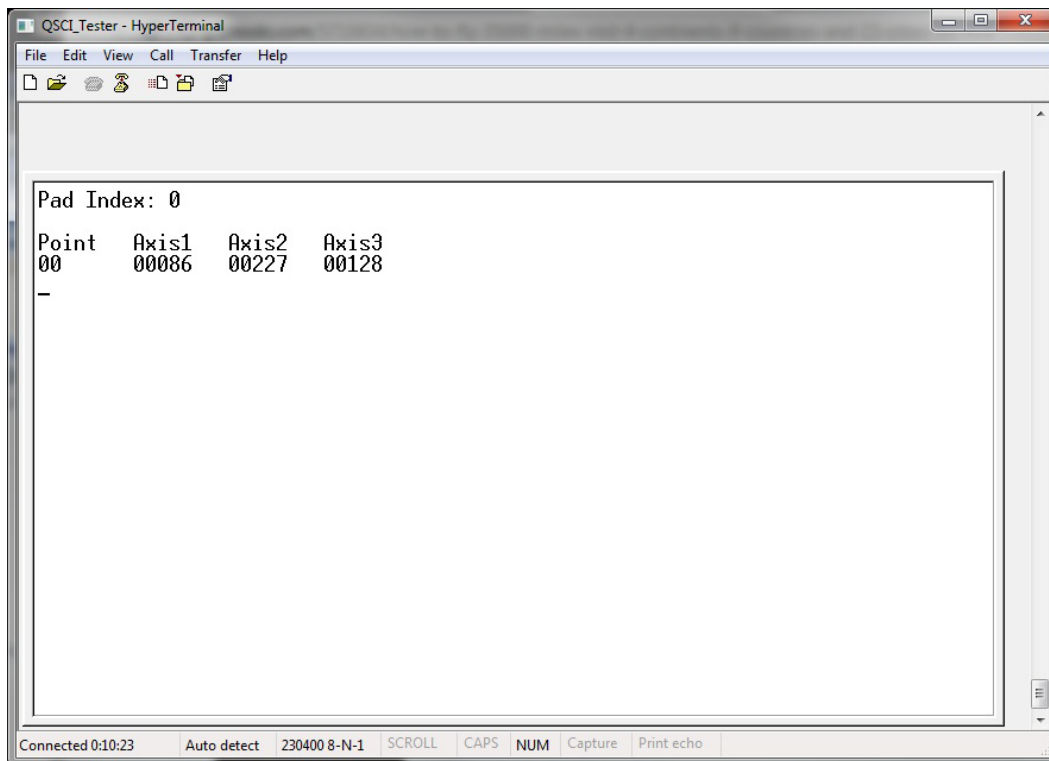


Figure 20. Example Retrieved nD Pad Output

3.5.9. Generic Data

The generic data menu option allows the user to stream generic data from the device and also write values to generic data elements on the device. This causes it to have its own menu.

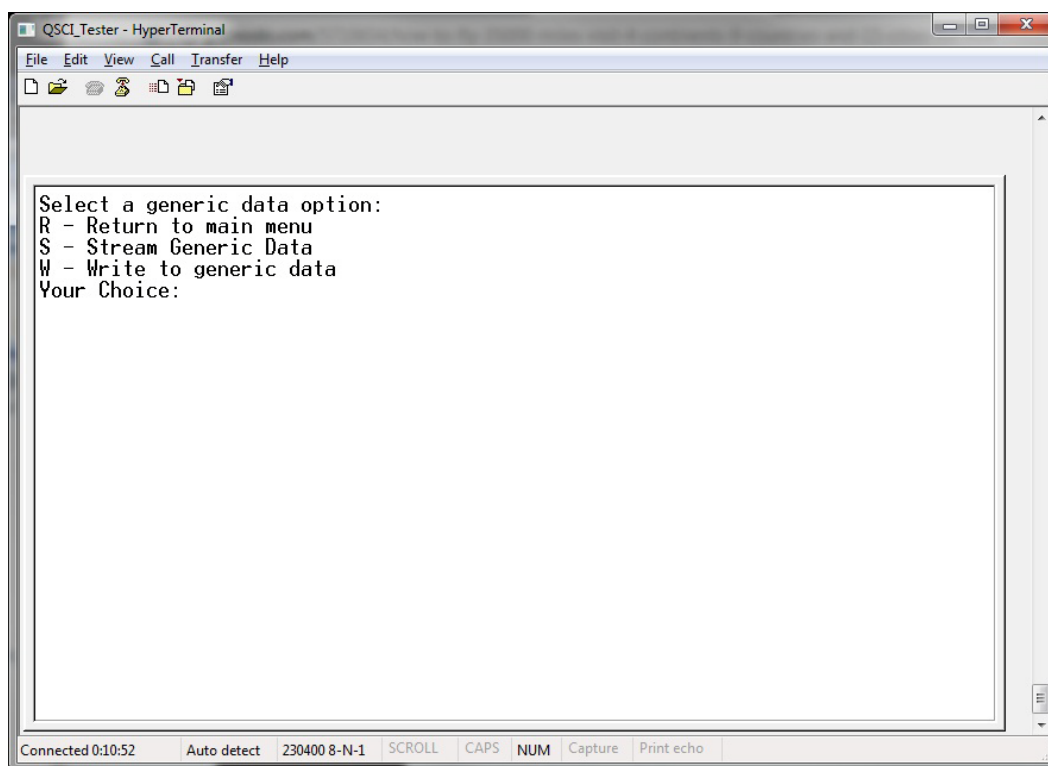


Figure 21. Generic Data Channel Menu

Press 'W' or 'w' to bring up the write generic data prompts. First, it asks the user to define which generic data element to write. Finally, it asks what data the user wants to write to that generic data index in hex format. Once the user fills in the prompts, the host writes the data to the device and prompts the user to return to the generic data menu.

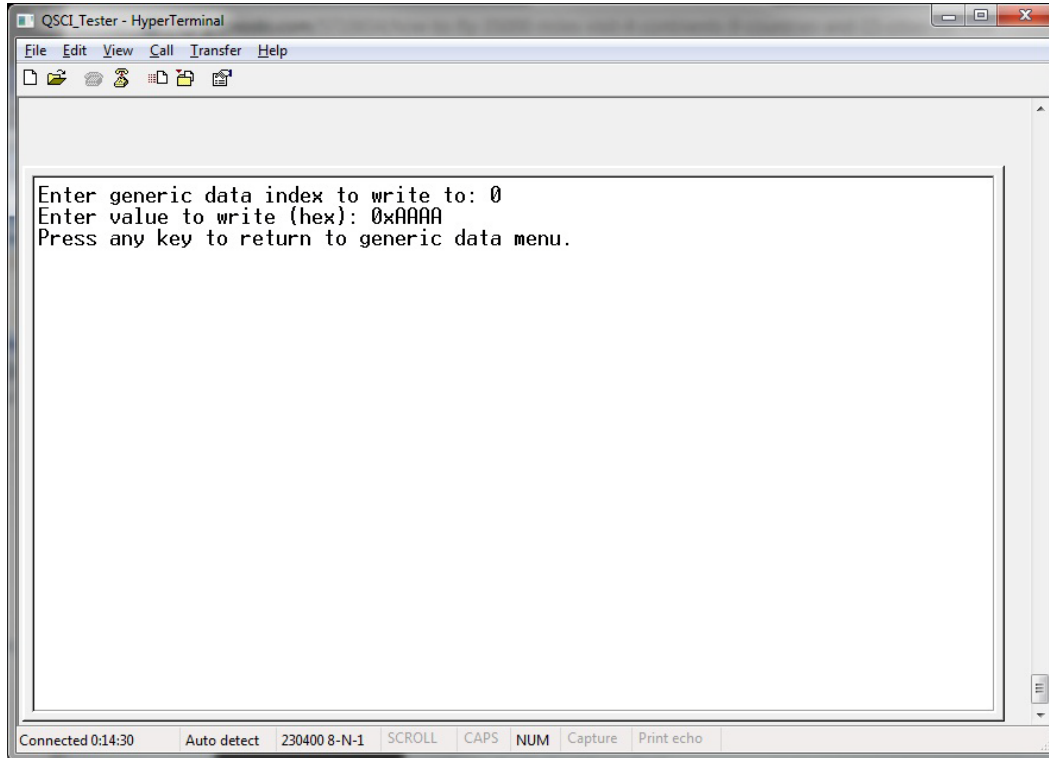
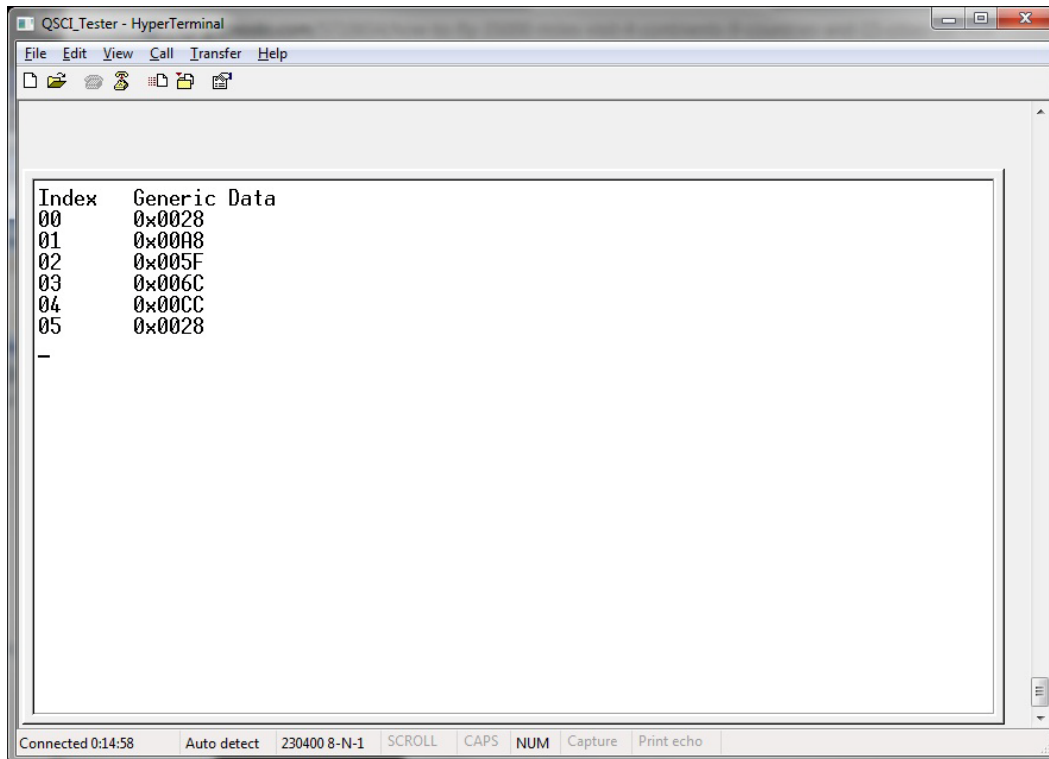


Figure 22. Writing to a Generic Data Channel

If the user presses 'S' or 's', the host retrieves generic data elements from the device and streams them to the terminal every 500 ms. To stop streaming, press and hold the P2.0 hardware button until streaming stops and a prompt appears.



```
QSCI_Tester - HyperTerminal
File Edit View Call Transfer Help
Index  Generic Data
00     0x0028
01     0x00A8
02     0x005F
03     0x006C
04     0x00CC
05     0x0028
-
Connected 0:14:58  Auto detect  230400 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

Figure 23. Reading from Generic Data Channels

CONTACT INFORMATION

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
<https://www.silabs.com/support/pages/contacttechnicalsupport.aspx>
and register to submit a technical support request.

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories, Silicon Labs, and QuickSense are trademarks of Silicon Laboratories Inc.
Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.