CIRQUE®

Interfacing to Gen 4 Using I2C

Application Note

Document Version 1.3.5

JANUARY 2017
GP-AN-130823

This document provides the information needed to interface to a Cirque Gen 4 trackpad using I2C. Important features that are required to read, write, and process data are included. This document applies to Cirque's Gen 4 firmware version: 1.2 and later.

Note: *Contact Cirque for the proper Firmware ID for your application.*

Documentation Version History

Date	Current Version	Description
NOV 2013	1.0	Document creation
JUNE 2014	1.1	Updated memory table. Removed developer's names from examples.
JUNE 2014	1.2	Updated the copyright, trademark, and patent information.
FEBRUARY 2015	1.3	Updated register information and removed the sample code section.
JUNE 2015	1.3.1	Added information for the SHA-256 Firmware Verification Process manual information.
MAY 2016	1.3.2	Updated extended memory table.
JULY 2016	1.3.3	Updated extended memory table.
OCTOBER 2016	1.3.4	Updated Data Packet information and other sections for better clarity.
JANUARY 2017	1.3.5	Added the steps for enabling Absolute mode.

Notice

Copyright, Trademark, and Patent Information

Table of Contents

1. Introduction

Cirque® GlidePoint® Gen 4 provides a feature rich and innovative product line that has built-in gestures and other high-end features that are normally only found on trackpads after installing a custom driver. The Gen 4 module communicates through the I2C HID interface, which is common in the latest models of trackpads. The Gen 4 solution follows the standard HID interface to report the mouse and keyboard data. Position data for up to five (5) fingers with true multi-point tracking is possible. By default, Gen 4 works as a standard HID mouse. This means that the device is enabled by moving a finger across the sensitive area of the trackpad. The trackpad will then start to produce Relative Mouse Position Data. The data is easily read for embedded applications. The data can be read and passed directly to the USB protocol.

Sample Gen 4 firmware code is provided to enable communication with a trackpad using the I2C protocol.

1.0.1 Document Overview

This document provides the information needed to interface a Gen 4 sensor to a host. Important features that are required to READ, WRITE, and PROCESS data are included, as well as how to select the preferred features and options. Proper communication information is provided in the following sections:

- *Gen 4 Communication*
  - *Gen 4 Low-level I2C Interface*
  - *Gen 4 Data Ready Signal*
  - *HID Protocol*
- *Gen 4 I2C HID Interface*
- *Absolute Position Data*
- *Extended Memory Access*
- *Enabling and Disabling Gestures*
- *Appendix A*

1.0.1.1 Additional Documentation

Other documents that provide information about the GlidePoint Gen 4 platform:

- *Rushmore Electrical Specification*
- *GP130831 GlidePoint Gen 4 Electrical and Mechanical Specification*
- *GlidePoint Gen 4 Boot Loader App Note*
- *GT150602 SHA-256 Firmware Verification Process*

2. Gen 4 Communication

This section describes proper communication with Gen 4, as well as how to select the preferred features and options. This information is provided in the following sections:

- *Gen 4 Low-level I2C Interface*
- *Gen 4 Data Ready Signal*

The code for the tasks described in this document can be found in the sample-code zip file. See *Appendix A on page 39* for more information.

2.1 Gen 4 Low-level I2C Interface

Gen 4 hardware uses the I2C HID protocol layered on top of the low-level I2C protocol. This section of the document describes the low-level I2C interface. A description of the HID interface layer is provided later in *Gen 4 I2C HID Interface on page 10*.

The I2C bus contains a data signal (SDA) and a clock signal (SCL). The clock is provided by the master and the data line is bidirectional.

Table 1. Gen 4 I2C Signals

Signals	Description
I2C	
SDA	Serial Data line for both input and output for master and slave.
SCL	I2C Serial Clock line provided by the master as an input to all slave devices.
Gen 4	
DR	Indicates Gen 4 has data ready to send
GND	Ground
VDD	Power supply

Gen 4 is designed as a slave device on the I2C bus. The default slave address for a Gen 4 is 0x2A. I2C slave addresses are seven bits long to allow an eighth bit to signify a READ or WRITE command. In the first byte sent by the host, the hexadecimal value for the slave address (0x2A) is shifted up one bit, plus the READ/WRITE bit. The final value is either 0x54 for WRITE or 0x55 for READ (see Figure 1 below).

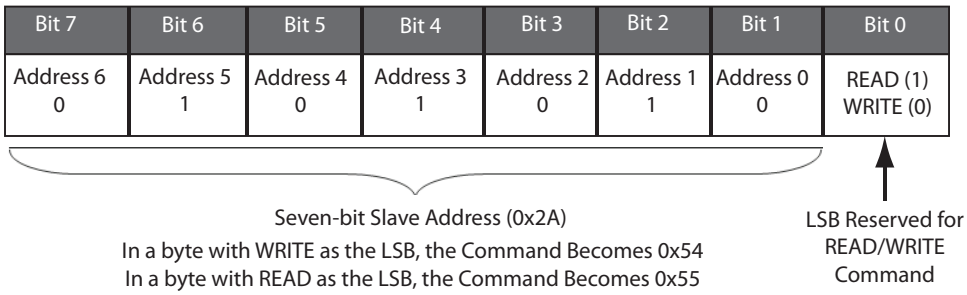

Figure 1. I2C READ or WRITE Command with Gen 4's Address

When the bus is idle, the SDA and SCL lines are high due to pull-up resistors on each line. The master starts communication with a start condition, which is a high to low transition on the SDA line while the SCL line remains high. After the start signal, data is placed on the SDA line when the SCL line is low. The receiving device latches the data when the SCL line is high. All 8-bits are transmitted and a ninth bit is designated as the acknowledge bit (ACK [0] /NACK [1]). The first byte transmitted will always be a 7-bit address of the target slave and a READ/WRITE bit indicating the direction of data flow. Any bytes following the slave address byte are only for the addressed slave.

When reading from the slave, the master must NACK the last byte to indicate to the slave that it was the last byte to be exchanged. After at least one data byte is sent or received, the master can cause a stop condition to free the bus. The stop signal requires SDA to transition from low to high, while the clock is high.

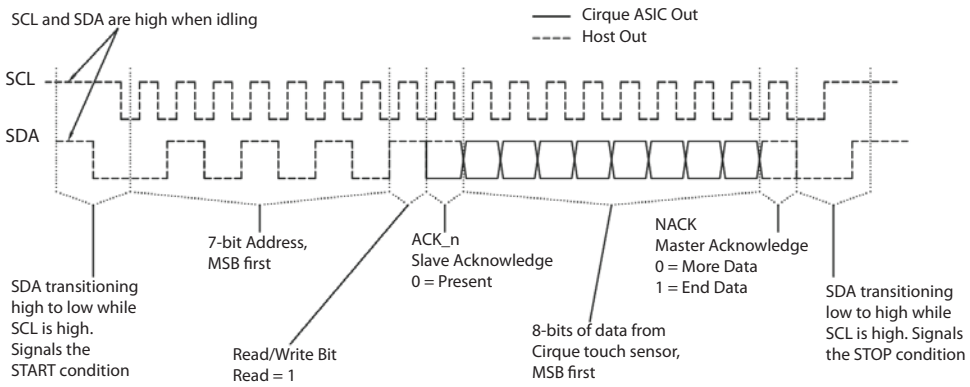

Figure 2. I2C Signals - READ

When writing, the slave is not required to NACK the last byte because the slave does not know how many bytes it is to receive. A Stop Condition occurs when the SDA line rises while the SCL line is high.

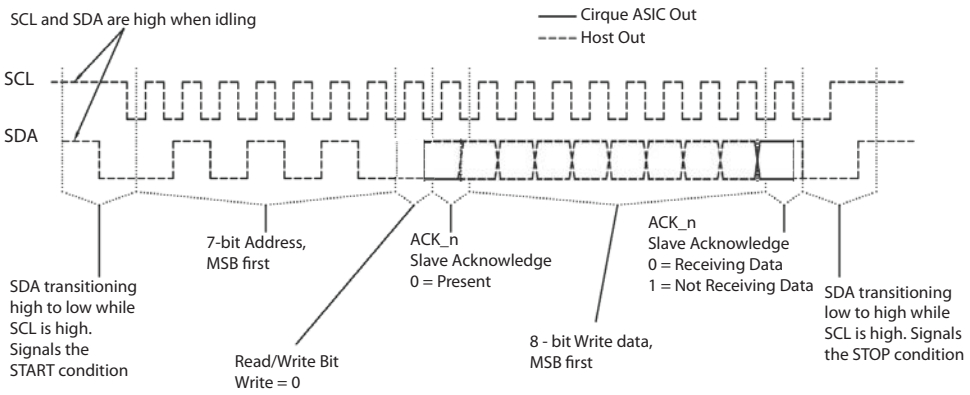

Figure 3. I2C Signals - WRITE

2.2 Gen 4 Data Ready Signal

The Gen 4 solution has a hardware signal that identifies when new data is ready to be sent from the trackpad to the host.

- The Data Ready (DR) signal will go low (assert) any time the device has a new touch data packet.
- The DR signal will go high (de-assert) when the slave device (Gen 4) is addressed. If the host READs the entire data packet, DR will stay high until a new data packet is generated.
  - If the host does not READ the entire data packet, the DR will re-assert (go low) immediately after the I2C stop condition. The same data packet will be sent at the next host READ. In other words, partially read packets are re-queued. You must read the entire data packet before Gen 4 will generate a new data packet.

The DR Signal will not be modified by any extended memory READ or WRITE.

3. HID Protocol

This section is an abbreviated summary of Human Interface Device (HID) protocol. The details can be found in Microsoft's HID Over I2C Protocol Specification document, which Cirque recommends downloading from Microsoft's MSDN site.

Note: *HID is a protocol layer defined for operating systems such as Windows and Linux. If your application does not use an operating system with HID support, you can skip this section.*

3.1 Using HID for I2C or USB

At the HID layer, the data format is the same for both I2C and USB. For this reason, it is possible to pass the data from an I2C HID device directly to a USB HID device. Three things must be accomplished to use HID data from Gen 4:

1. Identify the HID descriptor address (0x0001).
2. READ the HID descriptor.
3. READ the touch data. See *Mouse Data on page 12*.

3.1.1 Identifying the HID Descriptor Address

The HID descriptor can be stored at any 16-bit address. Cirque stores the HID descriptor at the following address: 0x0001.

3.1.2 Reading the HID Descriptor

Reading the descriptor is completed by issuing an I2C WRITE command to the HID descriptor address and then issuing an I2C READ command afterwards. The response of the I2C that is read from the trackpad will contain the full I2C HID descriptor.

Once this descriptor has been read, it can be used as the USB HID descriptor if the application uses USB. No modifications are required to use the descriptor over USB.

The examples code in the sample code.zip (See *Appendix A on page 39* for more information.), may be used to read the HID descriptor and pass the data over USB.

3.1.3 Reading the Touch Data

Please see the data format descriptions in *Gen 4 I2C HID Interface on page 10*.

3.2 Differences Between Absolute and Relative Mode

Cirque's products can report positions in either an Absolute or a Relative manner.

- In Relative mode, gestures and mouse data are reported natively by the Gen 4 trackpad, with no host calculation required.
- In Absolute mode, five XY positions are reported but gestures and mouse data must be calculated by the host.

Gen 4 devices are in Relative mode by default. When using Relative mode (mouse mode), the finger position is used to move a mouse cursor.

Follow these steps to select either Absolute or Relative mode (as explained in *Extended Memory Write on page 19*):

- Write 0x03 in register 0xC2C4 to enter Absolute mode.
- Write 0x01 to register 0xC2C4 to enter Relative mode.

The data packet format will change to match the selected mode. For more information:

- See *The Absolute Position Data Packet on page 15* for the Absolute Data format.
- See *HID Packet Format on page 10* for the Relative Data format.

4. Gen 4 I2C HID Interface

The Gen 4 solution follows the standard HID interface to report the mouse and keyboard data. I2C to USB conversion is described in *Using HID for I2C or USB on page 8*. By default, Gen 4 works as a standard HID mouse. After a power on event, the trackpad will be enabled as a finger moves across the sensitive area of the trackpad. The trackpad will then start to produce Relative Mouse Position Data.

Gen 4 signals when new data is available by the Data Ready (DR) signal going low. Once the DR has gone low, new data is read from the trackpad by issuing an I2C read command and then allowing the trackpad to send this data in a packet.

A data packet contains both the cursor and gesture data by default. Cursor data is sent using a HID mouse report. The mouse report ID is 6. Gesture data is sent using keyboard key strokes and mouse scroll-wheel reports. The keyboard report ID is 8.

4.1 HID Packet Format

The HID packet format is outlined in Figure 4 below. The data is explained in detail in Table 2, Table 3, and Table 4 on page 11.

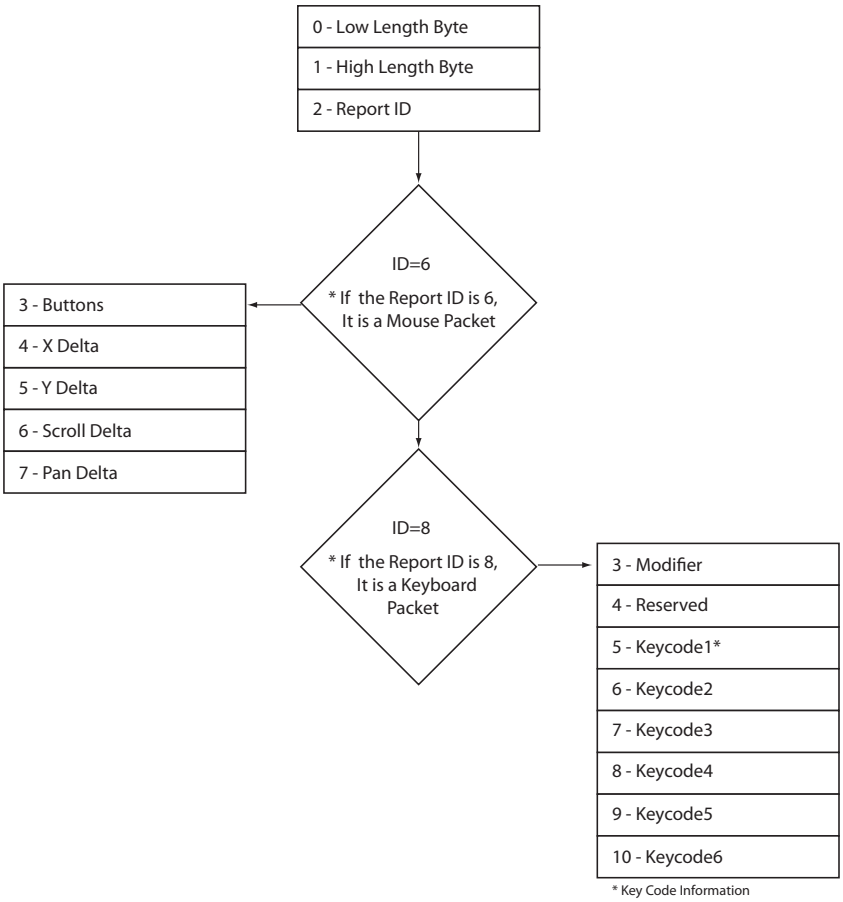

Figure 4. The HID Packet Format

Table 2. Gen 4 Data Format

Byte Number	Name	Description
0	LengthLowByte	The lower byte of the length byte.
1	LengthHighByte	The upper byte of the length byte.
2	ReportID	The unique identification assigned to each report that is used to distinguish between mouse and keyboard data. **Note** *If the report ID is 6, it is a mouse packet. If the report ID is 8, it is a keyboard packet.*

Table 3. Mouse Packet Format - Report ID 6

Byte Number	Name	Description
3	Buttons	The button data.
4	X Delta	The X motion deltas, signed 8-bit value.
5	Y Delta	The Y motion deltas, signed 8-bit value.
6	Scroll Delta	The Vertical scroll deltas, signed 8-bit value.
7	Pan Delta	The Horizontal scroll deltas, signed 8-bit value.

Table 4. Keyboard Packet - Report ID 8

Byte Number	Name	Description							
3	Modifier Keys	Bit 7 Right GUI	Bit 6 Right Alt	Bit 5 Right Shift	Bit 4 Right Ctrl	Bit 3 Left GUI	Bit 2 Left Alt	Bit 1 Left Shift	Bit 0 Left Ctrl
4	Reserved	Reserved							
5	Keycode 1	See Table 5 - *Keycode Table for Keycode 1- of Report ID 8*							
6	Keycode 2	Not used							
7	Keycode 3	Not used							
8	Keycode 4	Not used							
9	Keycode 5	Not used							
10	Keycode 6	Not used							

Table 5. Keycode Table for Keycode 1- of Report ID 8

Value	Gesture	Key	Description
0x07	Minimize/ Maximize all	Letter “D”	Used for the **Windows Logo** key + **D** key command.
0x50	Back command	Left arrow	Used for the **Alt** + **left arrow** function.
0x4F	Forward command	Right arrow	Used for the **Alt** + **right arrow** function.
0x2B	Windows Task View	Tab	Used for the **Windows Logo** key + **Tab** key command.
0x00	Menu	None	Used for the **Windows Logo** key

4.2 Reporting Mouse and Gesture Data

All mouse and gesture data is reported by using the HID mouse and keyboard descriptors. This section will describe the mouse and keyboard data and the different options for that data.

4.2.1 Mouse Data

Mouse data is reported in byte 3, 4, 5, 6, and 7 of the HID mouse packet (Figure 4 on page 10).

4.2.1.1 X and Y Deltas

The relative cursor motion is reported as X and Y deltas in bytes 4 and 5 of the mouse packet as a signed 8-bit value.

- For X motion, moving from left to right results in a positive X delta value being reported, moving right to left results in a negative X delta value being reported.
- For Y motion, moving from bottom to top results in a positive Y delta value. Moving down from top to bottom results in a negative Y delta value.

The built-in acceleration and cursor ballistics in Gen 4's firmware manages the mouse ballistics.

4.2.1.2 Scroll Data

Scroll data is byte 6 of the mouse packet and it is a signed 8-bit value.

- Positive values indicate a scroll up.
- Negative values indicate a scroll down.

4.2.1.3 Pan Data

Pan data is byte 7 of the mouse packet and it is a signed 8-bit value.

- Positive values indicate a pan from left to right.
- Negative values indicate a pan from right to left.

4.2.2 Keyboard Data

Keyboard data is reported directly to the OS and can be used to create keyboard combinations for gestures.

- Each gesture requires a different set of keys, or key combinations, to be reported.
- Each gesture, and its reported combination, is described below.

4.2.2.1 Zoom Gesture

A zoom gesture is reported when two fingers are moved apart or pinched together.

Therefore, when two fingers perform a zoom gesture on the Gen 4 trackpad, a keyboard packet with a control key pressed is reported. The subsequent packets will be mouse packets that only contain scroll data.

Sending the **Ctrl** key + **mouse scroll** data will produce a zoom gesture.

When the fingers have lifted and the zoom gesture has ended, a keyboard report of all zeros will be sent indicating that the key has been lifted.

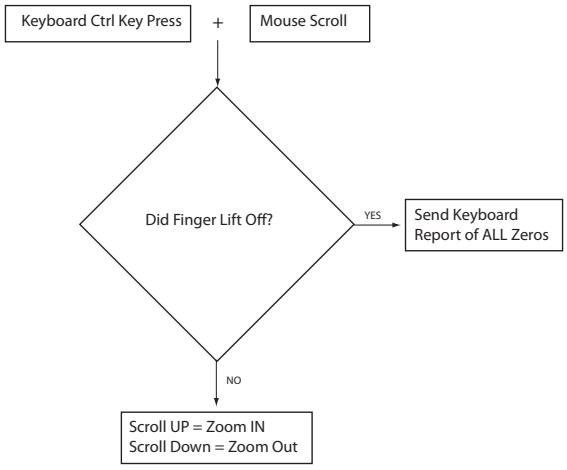

Figure 5. Zoom Gesture

4.2.2.2 Three Fingers Left/Right (Back/Forward) Gesture

A back or forward command in a Web browser is made when three fingers are moved left or right. This gesture sends the **Alt** key + the **Left** or **Right Arrow** key,

- Left for back
- Right for forward

The **ALT** key is cleared once the gesture has been sent. The fingers do not have to lift to clear **ALT**. They only need to lift in order to begin a new gesture.

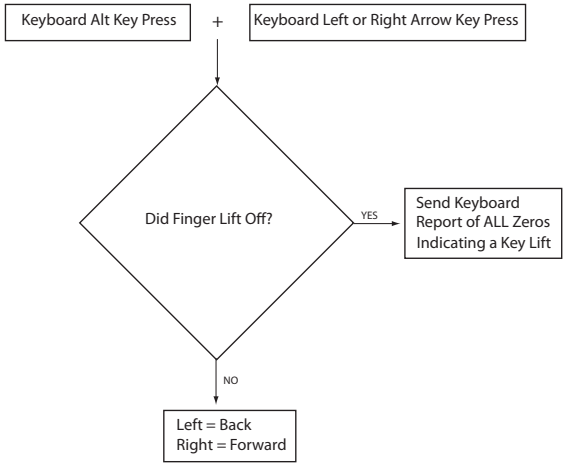

Figure 6. Left or Right Gesture

4.2.2.3 Three Fingers Down

Moving three fingers down will minimize all open Windows in a Windows OS environment. This gesture sends a **Windows Logo** key plus the letter **D.** The **Windows Logo** key press must be sent first.

The keyboard report is cleared by sending a keyboard report of all zeros once the gesture has been sent. The fingers do not have to lift to clear keyboard report. They only need to lift in order to begin a new gesture.

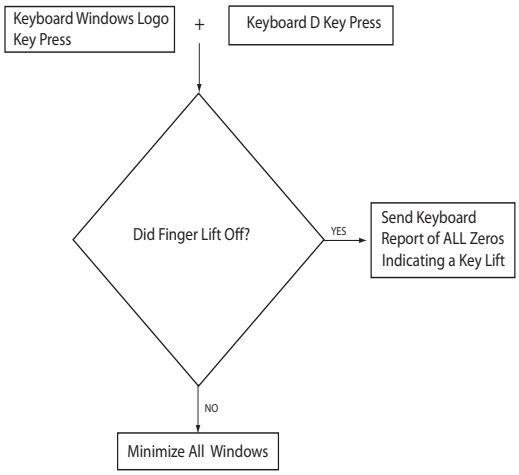

Figure 7. Three Fingers Down Windows Gesture

4.2.2.4 Three Fingers Up

Moving three fingers up will cause the Windows menu to appear. This gesture sends a **Windows Logo** key press.

The keyboard report is cleared by sending a keyboard report of all zeros once the gesture has been sent. The fingers do not have to lift to clear keyboard report. They only need to lift in order to begin a new gesture.

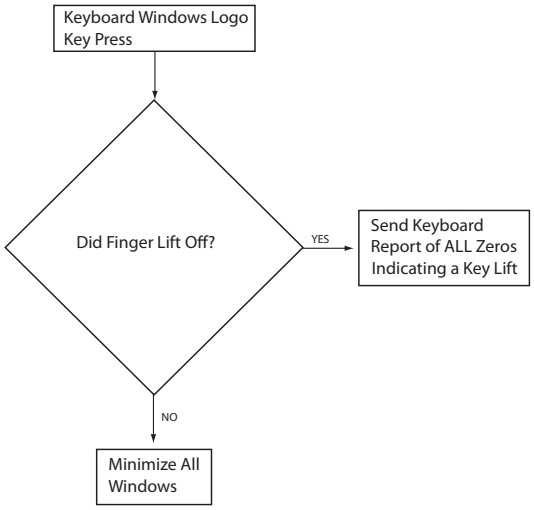

Figure 8. Three Fingers Up Windows Gesture

5. Absolute Position Data

Gen 4 devices use a standard HID interface to provide absolute X and Y-finger position every time a finger is on the sensor. Every time a finger is placed on the sensor, the DR signal will go low. This change indicates that the Gen 4 device has new data. When the DR signal has asserted, no new position data will be sent until the DR signal is cleared by reading the data. The DR signal will automatically clear after the X and Y data has been read. All 53 bytes of position data must be read with every DR event.

Note: *If all 53 bytes are not read with a new DR event, the trackpad will re-assert DR and continue trying to re-send the original data until all 53 bytes are read.*

This section will describe the following:

- *The Absolute Position Data Packet*.
- *Palm Data* and its format.

5.1 The Absolute Position Data Packet

The Absolute Position Data packet contains the position data for all five fingers and the button data, and has 53 bytes of data. The format of this packet is described below in the next section and in Table 6.

5.1.1 Reporting Absolute Data

This section provides the information needed to enable Absolute Data reporting. See Table 8 on page 22 in the *Extended Memory Table* section, for details on the C2C4 register contents.

Follow these steps to report Absolute Data:

- For I2C trackpads, one must set bit #1 (0x02) in the C2C4 register.
- For USB trackpads, one must set bit #4 (0x10) in the C2C4 register.

Note: *When setting a bit, care should be taken not to modify the other bits in the register. To set a bit, the master/host must READ the register's contents, MODIFY the local copy, and then WRITE the new value to the register.*

For example, with a USB trackpad, the host reads the C2C4 register and gets 0x48. The host then ORs 0x10 with 0x48 (to set bit #4), which produces 0x58. The value 0x58 should then be written to C2C4 and this will enable absolute data reporting.

5.1.2 Absolute Position Data Format

The first three bytes in the data packet are the length of the packet and the Report ID. Absolute mode uses a Report ID of 9. Byte 3, NumContacts, is a bit field that indicates which of the five possible fingers are touching.

The sequential order of the fingers is specified in NumContacts. As fingers are placed on the pad, they are ordered sequentially, starting at finger 0 and ending with finger 4. If multiple fingers are

on the pad, and one of those fingers is removed, the fingers order is preserved. The fingers are not re-ordered. For example:

- If fingers 0, 1, and 2 are on the pad, the NumContacts byte will report 0x07
- If finger 1 is removed, but fingers 0 and 2 remain, the NumContacts byte will report 0x05, indicating the bits for fingers 0 and 2.
- If no fingers are present on the pad, the NumContacts byte will be zero, and all XY-position bytes should be considered invalid.

5.1.2.1 Finger Data

Data for all five fingers is sent every time so it is the responsibility of the host to determine which fingers are valid and which are not, based on the position data. As shown in Table 6, the same format is used for each of the five fingers. The first byte for each of the finger data specifies Palm data.

Table 6. Absolute Position Data Format

Byte Number	Name	Description	Finger Data
0	Length Low	Low byte of the length	
1	Length High	High byte of the length	
2	Report ID	A unique identification assigned to each report For Absolute mode, the ID will be 9.	
3	NumContacts	Finger touch bit field	
4	Finger 0 Palm	Finger 0 Palm data (see Table 7 below	Finger 0 Data
5	Finger 0 X Low	Bottom 8 bits of the 16-bit X value	
6	Finger 0 X High	Top 8 bits of the 16-bit X value	
7	Finger 0 Y Low	Bottom 8 bits of the 16-bit Y value	
8	Finger 0 Y High	Top 8 bits of the 16-bit Y value	
9	Finger 1 Palm	Finger 1 Palm data (see Table 7 below	Finger 1 Data
10	Finger 1 X Low	Bottom 8 bits of the 16-bit X value	
11	Finger 1 X High	Top 8 bits of the 16-bit X value	
12	Finger 1 Y Low	Bottom 8 bits of the 16-bit Y value	
13	Finger 1 Y High	Top 8 bits of the 16-bit Y value	
14	Finger 2 Palm	Finger 2 Palm data (see Table 7 below	Finger 2 Data
15	Finger 2 X Low	Bottom 8 bits of the 16-bit X value	
16	Finger 2 X High	Top 8 bits of the 16-bit X value	
17	Finger 2 Y Low	Bottom 8 bits of the 16-bit Y value	
18	Finger 2 Y High	Top 8 bits of the 16-bit Y value	
19	Finger 3 Palm	Finger 3 Palm data (see Table 7 below	Finger 3 Data
20	Finger 3 X Low	Bottom 8 bits of the 16-bit X value	
21	Finger 3 X High	Top 8 bits of the 16-bit X value	
22	Finger 3 Y Low	Bottom 8 bits of the 16-bit Y value	
23	Finger 3 Y High	Top 8 bits of the 16-bit Y value	

Table 6. Absolute Position Data Format - (continued)

Byte Number	Name	Description	Finger Data
24	Finger 4 Palm	Finger 4 Palm data (see Table 7 below	Finger 4 Data
25	Finger 4 X Low	Bottom 8 bits of the 16-bit X value	
26	Finger 4 X High	Top 8 bits of the 16-bit X value	
27	Finger 4 Y Low	Bottom 8 bits of the 16-bit Y value	
28	Finger 4 Y High	Top 8 bits of the 16-bit Y value	
29	Buttons Data	Data for which buttons are pressed	
30 - 52	**Reserved bytes**	**These are all "Reserved" bytes, but they must be read to complete each transaction.**	

5.2 Palm Data

The Palm bit indicates that the object touching the pad is an object large enough to be determined as a palm. Typically, if Bit 7 is set (1), then the X and Y data for that object should be ignored since tracking with large objects is not the intended operation.

Table 7. Palm Data Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Palm reject	Not Used	Not Used	Not Used	Pen *(If supported)*	Not Used	Touch confidence	Touch, single sample

Note: *In versions where the stylus (pen) operation is supported, bit 3 of this byte is also used to indicate that the stylus is touching the surface.*

For the 53-byte packets, use bit 3 to denote the presence of a stylus in Gen 4 versions that support a stylus (or pen).

6. Extended Memory Access

Extended memory access is used to modify settings on a Gen 4 device. It has unique READ and WRITE extended memory access commands. This section will cover:

- *Extended Memory Read*
- *Extended Memory Write*
- *Example Code*

The Gen 4 Register set is provided in *Extended Memory Table on page 22*.

6.1 Extended Memory Read

An extended memory READ is accomplished by creating a packet. It has a 16-bit Extended Access Register value set to 0x0901. The lower byte is first, followed by the upper byte. The 32-bit address that is to be read is next, starting with the lowest 8-bits and following sequentially up to the top 32-bits. After the 32-bit address the 16-bit length value, the lower byte first followed by the upper byte. Once the packet has been created, it is sent to the device using an I2C WRITE followed by an I2C READ. The response from the Gen 4 device is:

- The first two bytes are the length bytes
- The data bytes containing the values at the extended memory locations (if the length is greater than one it is a multi-byte READ and the Gen 4 device will respond with multiple bytes).
- The last byte is an 8-bit CheckSum calculated over the entire packet.

The CheckSum is used to validate the integrity of the values read. The data sequence for an extended memory read is shown in the example below.

6.1.0.1 Legend

S	=	Start (I2C 2-wire start condition)
W	=	Write
R	=	Read
A	=	ACK (Acknowledge)
N	=	NAK (NOT-Acknowledge)
P	=	STOP (I2C stop condition)

S	Slave Addr	W	A	Ext Access - L (01)	A	Ext Access - H (09)	A	Byte 0 (xx)	A	Byte 1 (xx)	A	Byte 2 (xx)	A	Byte 3 (xx)	A	Length -L (xx)	A	Length -H (xx)	A
I2C WRITE				Ext Access Register READ				Extended Address								Number of Bytes			

S	Slave Addr	R	A	Length - L (xx)	A	Length - H (xx)	A	Byte 0 (xx)	A	...	A	Byte n-1 (xx)	A	Checksum (xx)	N	P
I2C READ				Number of Bytes				Byte Data								

6.2 Extended Memory Write

An extended memory write is performed by creating a packet. It has a 16-bit Extended Access Register set to value 0x0900. The lower byte is first, followed by the upper byte. The 32-bit address that is to be read is next, starting with the lowest 8-bits and following sequentially up to the top 32-bits. After the address is the 16-bit length value, lower byte first followed by the upper byte. Then the Data bytes are sent (the number of bytes must match the length). The final byte is a CheckSum of the entire packet. Once the packet has been created, it is sent to the device using an I2C WRITE.

The data sequence for an extended memory Write (see *Extended Memory Table on page 22* for the available register map and its values) is shown in tables below.

S	Slave Addr	W	A	Ext Access - L (00)	A	Ext Access - H (09)	A	Byte 0 (xx)	A	Byte 1 (xx)	A	Byte 2 (xx)	A	Byte 3 (xx)	A
				Ext Access Register WRITE				**Extended Address**							

Length - L (xx)	A	Length - H (xx)	A	Byte 0 (xx)	A	...	A	Byte n-1 (xx)	A	Checksum (xx)	P
Number of Bytes				**Byte Data**							

6.3 Example Code

Sample code for extended memory reads and writes is provided below.

```
#define SUCCESS 0
#define WRITE_FAIL 1
#define READ_FAIL 2
#define CHECKSUM_ERROR 3

typedef struct _BulkAccessAddresses
{
    uint16_t Write;
    uint16_t Read;
} BulkAccessAddresses;

BulkAccessAddresses.Write = 0x0900;
BulkAccessAddresses.Read = 0x0901;


uint8_t Read(uint32_t Address, uint16_t length, uint8_t* Report)
{
 uint8_t Checksum;
 uint8_t OutBuf[8] = { (uint8_t)(BulkAccessAddresses.Read & 0xFF),
                  (uint8_t)((BulkAccessAddresses.Read >> 8) & 0xFF),
                  (uint8_t)(Address & 0xFF),
                  (uint8_t)((Address >> 8) & 0xFF),
                  (uint8_t)((Address >> 16) & 0xFF),
                  (uint8_t)((Address >> 24) & 0xFF),
                  (uint8_t)(length & 0xFF),
                  (uint8_t)((length >> 8) & 0xFF) };

if (I2CInterface1.I2CWrite(OutBuf, sizeof(OutBuf), I2CInterface.Flags.NoStop) !=
I2CInterface.RetValue.Success)
{
    return (WRITE_FAIL);
}

if (I2CInterface1.I2CRead(Report, sizeof(Report), I2CInterface.Flags.NoFlags) !=
I2CInterface.RetValue.Success)
{
    return (READ_FAIL);
}

 Checksum = ChecksumBuffer(Report, Report.Length - 1);

if (Checksum != Report[Report.Length - 1])
{
    return (CHECKSUM_ERROR);
}

return (SUCCESS);
}


// private helper function-checksum calculation
uint8_t ChecksumBuffer(uint8_t* Buffer, uint16_t Length)
{
    uint16_t temp;
    uint8_t Checksum = 0;
    for (temp = 0; temp < Length; temp++)
    {
        Checksum += *(Buffer+temp);
    }
```

```
    return (Checksum);
}

uint8_t RetValue Write(uint16_t Address, uint8_t* Report, uint16_t numBytes)
{

//This is sample code only. Of course, arrays cannot be dynamically sized like this
in C
uint8_t OutBuf[2 + 4 + 2 + numBytes + 1];

OutBuf[0] = (uint8_t)(BulkAccessAddresses.Write & 0xFF);         // 2 +
OutBuf[1] = (uint8_t)((BulkAccessAddresses.Write >> 8) & 0xFF);
OutBuf[2] = (uint8_t)(Address & 0xFF);                           // 4 +
OutBuf[3] = (uint8_t)((Address >> 8) & 0xFF);
OutBuf[4] = (uint8_t)((Address >> 16) & 0xFF);
OutBuf[5] = (uint8_t)((Address >> 24) & 0xFF);
OutBuf[6] = (uint8_t)(numBytes & 0xFF);                     // 2 +
OutBuf[7] = (uint8_t)((numBytes >> 8) & 0xFF);

Report.CopyTo(OutBuf, 8);                                 // Report.Length +

OutBuf[OutBuf.Length - 1] = ChecksumBuffer(OutBuf, OutBuf.Length - 1);

if (I2CInterface1.I2CWrite(OutBuf, OutBuf.Length, I2CInterface.Flags.NoFlags) !=
I2CInterface.RetValue.Success)
{
   return (WRITE_FAIL);
}

return (SUCCESS);
}
```

7. Extended Memory Table

This section provides the Gen 4 register set.

Table 8. Gen 4 Command Register

Logical Address	Name	Description	Size (Bytes)	Bit Number
C2C0	Chip ID	Identifies the version of the chip used.	1	
C2C1	Firmware Version	Major firmware version. **Note** *A firmware sub-version is also located at address C2DC.*	1	
C2C2	SysConfig1	Main configuration register for basic system functions.	1	
	SysConfig1.PS2Mode_Single_0_Dual_1	Controls the PS2 configuration.		0
	SysConfig1.TrackingEnable	Enables finger touch points extraction. **Note** *Must be set to occur.*		1
	SysConfig1.AnyMeasEnable	Enables the Cirque “Any Meas” interface. **Note** *Used for making arbitrary sensor measurements.*		2
	SysConfig1.Assert_DR_At_POR	Enables the assertion of the Data Ready (DR) signal at POR.		3
	SysConfig1.InterfacePriorityPrimary	Allows locking data feed on primary. **Note** *Must be set to occur.*		4
	SysConfig1.InterfacePrioritySecondary	Allows locking data feed on secondary. **Note** *Must be set to occur.*		5
	SysConfig1.InterfacePriorityTertiary	**RESERVED**		6
	SysConfig1.RandomizeDrivePatterns	Randomize the ordering of the drive patters with each frame. **Note** *Must be set to occur.*		7
C2C3	SysConfig2		1	
	SysConfig2.NoIntellimouseAllowed	Prevents the Intellimouse sequence from activating Intellimouse mode.		0
	SysConfig2.UpsetRecoveryEnable	Enables the ESD upset recovery algorithm.		1
	SysConfig2.DoubleMeasResults	Doubles the values of the measurement results from the AFE.		2
	SysConfig2.ResultScalerEnable	Enables the measurement result scaler algorithm.		3
	SysConfig2.GeneratePeakData	Enables the calculation and reporting of sensor signal peak data. See registers C384 and C386.		4
	SysConfig2.DisableRushmoreWatchdog	Disable monitoring of ASIC state for possible reset needed condition.		5
	SysConfig2.TristateUnmuxedSensetrodes	Float unused sense electrodes.		6
	SysConfig2.ExecuteEFAutotune	Scan for quiet electrode frequency combination.		7
C2C4	FeedConfig1		1	
	FeedConfig1.PrimaryFeedEnable	Enables touch data reporting.		0
	FeedConfig1.PrimaryFeedType	0=Relative (mouse) data reporting. 1=Absolute (multi-finger) data reporting.		1

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
	FeedConfig1.AbsXYZZEnable	Enables Cirque proprietary XYZZ packet type for absolute data.		2
	FeedConfig1.SecondaryFeedEnable	**RESERVED**		**3**
	Feedconfig1.SecondaryFeedType	0 = Relative (mouse) data reporting. 1 = Absolute (multi-finger) data reporting.		4
	FeedConfig1.LockDataPort	Locks feed to data port determined by bit 6.		5
	FeedConfig1.LockDataPort_pri0_sec1	Port locking selection: 0 = Primary 1 = Secondary		6
	FeedConfig1.Calibrate	Forces the system to perform a sensor calibration.		7
C2C5	FeedConfig2		1	
	FeedConfig2.InvertXData	X data reported as MAX - 0 (inverted).		0
	FeedConfig2.InvertYData	Y data reported as MAX - 0 (inverted).		1
	FeedConfig2.SwapXYData	Swap X and Y coordinate reporting.		2
	FeedConfig2.YPanInvert	Inverts Vertical Scroll Gesture Reporting.		3
	FeedConfig2.DisableCoordinateData	Disables X and Y-touch data reporting.		4
	FeedConfig2.DisableButtonData	Disables button data reporting.		5
	FeedConfig2.KeypadAvailable	Makes keypad function available for user switching. **Note** *Only available on some versions.*		6
	FeedConfig2.KeypadEnable	Activates keypad mode. **Note** *Only available on some versions.*		7
C2C6	FeedConfig3		1	
	FeedConfig3.LinearCorrectionEnable	Enables the Coordinate Linear Correction Algorithm.		0
	FeedConfig3.DontStabilizeEdges	Disables edge coordinate stabilization. **Note** *Only available on some versions.*		1
	FeedConfig3.DeltaAccelerationEnable	Enables relative mode mouse acceleration. See registers *C346* through *C355*.		2
	FeedConfig3.NoiseAvoidanceEnable	Enables the noise avoidance frequency selection algorithm.		3
	FeedConfig3.MultiFingerPerFrameEnable	Disabled - only 1 finger added per frame. Enabled - all fingers reported from first frame.		4
	FeedConfig3.ClipFingersAtScalerBoundaries	Finger liftoffs are forced at logical scaler boundaries when set.		5
	FeedConfig3.DiscardNoisyDataSets	Enabled - Allows tracking algorithm to throw away noisy data sets.		6
	FeedConfig3.IntellimouseMode	Enabled - Intellimouse Packets are reported. Disabled - Standard mouse packets are reported.		7
C2C7	CompConfig1.0:	Initiates build-in self-test. Clears when test is complete.	1	
	CompConfig1.BITSTRun	R/W 1 = Run Built-in Self-Test. **Note** *Only available on some versions.*		0
	CompConfig1.OffsetCorrectionEnable	Enable Offset Correction Compensation Algorithm.		1
	CompConfig1.BackgroundCompEnable	Enables the Background Compensation Algorithm.		2
	CompConfig1.SignalSignCompEnable	Enables the Signal Sign Compensation Algorithm.		3

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
	CompConfig1.CongruenceCompEnable	Enables the Congruence Compensation Algorithm.		4
	CompConfig1.AllSignalPosCompEnable	Enables the All Signs Positive Compensation Algorithm. **Note** *Only available in some versions.*		5
	CompConfig1.BIST_PASS1_FAIL0	Indicates result of BIST: 1 = PASS 0 = FAIL **Note** *Only available in some versions.*		6
	NFC Active Notify	Notifies system of NFC RF active, enable countermeasures.		7
C2C8	PS2AuxControl		1	
	PS2AuxControl.CommandPassThruEnabled	PS/2 commands are passed through to auxiliary device when set.		0
	PS2AuxControl.ProcessAuxExtendedData	Auxiliary device data is assumed to be extended data when set.		1
	PS2AuxControl.GlidepointDataDisable	Disables all data from trackpad.		2
	PS2AuxControl.AuxDataDisable	Disabled all data from auxiliary device.		3
	PS2AuxControl.GlidepointCoordinateDisable	Disables position data from trackpad.		4
	PS2AuxControl.AuxCoordinateDisable	Disables position data from auxiliary device.		5
	PS2AuxControl.AuxAA00DetectDisable	Disables PS/2 AA, 00 detection from auxiliary device.		6
	PS2AuxControl.AuxDevicePresent	Set if system detects the presence of an auxiliary device.		7
C2C9	SampleRate	Sets the data-reporting rate. **Note** *Only available in some versions and then only some modes.*	1	
C2CA	ZIdle	Sets the number of zero-idle packets that are to be sent after touch release.	1	
C2CB	FilterControl		1	
	FilterControl.JitterEnable	Enables the Coordinate Jitter Filter Algorithm.		0
	FilterControl.LeastChangeEnable	Enables the Least Change Filter Algorithm.		1
	FilterControl.SmoothingEnable	Enables the Coordinate Smoothing Filter Algorithm.		2
	FilterControl.AdaptiveExpEnable	Enables the Adaptive Exponential Algorithm.		3
	FilterControl.OnlyFilterMultiFingerPTP	Filtering is only applied when multiple fingers are present.		4
	FilterControl.HoverFilterEnable	Enable Stylus/finger Filter.		5
	FilterControl.SwarmFilterEnable	Enable Stylus/resting Hand Filter.		6
	FilterControl.7	**RESERVED**		**7**
C2CC	GPIOControl	Set states for extra GPIO pins if applicable.	1	
C2CD	Button ZIdle	Sets the number of Z-idle packets that are sent after a button release.	1	
C2CE	PowerControl		1	
	PowerControl.DeepSleepEnable	Enables the deep sleep power control.		0
	PowerControl.ButtonWakeupEnabled	Enables the wake from deep sleep via button push.		1

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
	PowerControl.SleepWhenDisabled	Enables the activation of ultra-low power mode. **Note** *Can only be awakened via command or button.*		2
	PowerControl.CompOnEnable	Enables the causes system to compensate after wake from disabled sleep on PS/2.		3
	PowerControl.DisableStage2Sleep	Disables power control for stage 2.		4
	PowerControl.ShutdownPS2InI2CSleep	Causes system to shut off PS2 module when in I2C sleep when enabled.		5
	PowerControl.6	**RESERVED**		**6**
	PowerControl.7	**RESERVED**		**7**
C2CF	Reset Control/Status		1	
	Reset Control/Status.Cause0	System reset cause code bit 0.		0
	Reset Control/Status.Cause1	System reset cause code bit 1.		1
	Reset Control/Status.Cause2	System reset cause code bit 2.		2
	Reset Control/Status.Cause3	System reset cause code bit 3.		3
	Reset Control/Status.Cause4	System reset cause code bit 4.		4
	Reset Control/Status.Cause5	System reset cause code bit 5.		5
	Reset Control/Status.Cause6	System reset cause code bit 6.		6
	Reset Control/Status.ForceSystemReset	Set - Interface resets and will force entire system to reset. Clear - Interface resets are virtual.		7
C2D0	PTP Filter Control	Precision Touch Pad (PTP) versions only.	1	
	FilterControl.JitterEnable	Enables the Coordinate Jitter Filter Algorithm.		0
	FilterControl.1	Enables the Least Change Filter Algorithm.		1
	FilterControl.2	Enables the Coordinate Smoothing Filter Algorithm.		2
	FilterControl.AdaptiveExpEnable	Enables the Adaptive Exponential Algorithm.		3
	FilterControl.OnlyFilterMultiFingerPTP	Enables the Only Filter Multi Finger PTP Algorithm.		4
	FilterControl.5	**RESERVED**		**5**
	FilterControl.6	**RESERVED**		**6**
	FilterControl.7	**RESERVED**		**7**
C2D1	I2C Address	Sets the I2C slave address.	1	
C2D2	HID Descriptor Address	Sets or reads the HID descriptor address.	2	
C2D3				
C2D4	HID Descriptor - wVendorID	HID VID.	2	
C2D5				
C2D6	HID Descriptor - wProductID	HID PID.	2	
C2D7				
C2D8	HID Descriptor - wVersionID	HID version ID.	2	
C2D9				
C2DA	FeedConfig4		1	
	FeedConfig4.AdvancedABSEnable	Enables advanced absolute packet type. **Note** *Only available in some versions.*		0
	FeedConfig4.RightTapZoneEnable	Enable legacy Right-tap Zone.		1

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
	FeedConfig4.EdgeScrollZoneEnable	Enable legacy Edge Scroll Zone.		2
	FeedConfig4.LogicalScalingEnable	Scale reported coordinates according to table.		3
	FeedConfig4.PacketMultiplierEnable	Enable high-speed ABS packet output.		4
	FeedConfig4.5	**RESERVED**		**5**
	FeedConfig4.6	**RESERVED**		**6**
	FeedConfig4.LockDownToOneFinger	Limit to single point coordinate reporting. **Note** *Only available on some versions.*		7
C2DB	DR Glitch Count	Number of errant Rushmore DR's.	1	
C2DC	SubVersion ID	Firmware minor version.	1	
C2DD	TestConfig1		1	
	TestConfig1.SerialDataDumpEnable	Enables serial data dump.		0
	TestConfig1.ExtendedMemAccessEnable	Enables access to extended memory.		1
	TestConfig1.QuickMeasDumpEnable	Enable a quick measure data dump.		2
	TestConfig1.SerialMeasTrack	Serial meas track.		3
	TestConfig1.4	**RESERVED**		**4**
	TestConfig1.5	**RESERVED**		**5**
	TestConfig1.QuickMeasViewRawDataEnable	Enable QM raw data.		6
	TestConfig1.QuickMeasViewTrackCompDataEnable	Enable QM tracking compensation data.		7
C2DE	RushmoreMisc		1	
	RushmoreMisc.EXTClkOutEnable	Output the ext Clk.		0
	RushmoreMisc.ClientPullupEnable	Enables Rushmore client PS/2 interface pull ups.		1
	RushmoreMisc.HostPullupEnable	Enables Rushmore host PS/2 interface pull ups.		2
	RushmoreMisc.TP4ResetPulldownDisable	Enabled Pull down on TP4 reset pin. **Note** *Must be implemented.*		3
	RushmoreMisc.4	**RESERVED**		**4**
	RushmoreMisc.5	**RESERVED**		**5**
	RushmoreMisc.6	**RESERVED**		**6**
	RushmoreMisc.7	**RESERVED**		**7**
C2DF	PersistentDataControl		1	
	PersistentDataControl.SaveConfig	Saves current configuration to NV RAM.		0
	PersistentDataControl.IncludeFactoryCompWithConfigSave	Saves factory compensation with a configuration save when set.		1
	PersistentDataControl.RestoreDefaultConfigAndFactoryComp	Restores the default FW configuration and zero the factory compensation.		2
	PersistentDataControl.3	**RESERVED**		**3**
	PersistentDataControl.4	**RESERVED**		**4**
	PersistentDataControl.5	**RESERVED**		**5**
	PersistentDataControl.6	**RESERVED**		**6**
	PersistentDataControl.7	**RESERVED**		**7**

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C2E0	ADCTESTConfig	The ASIC ADC tests register value.	1	
C2E1	ADCSpare	The ASIC ADC spare register value.	1	
C2E2	ADC AW Control	The ASIC ADC aperture window controls register value.	1	
C2E3	ADC ETogControl	The ASIC ADC electrode toggles register value.	1	
C2E4	ADC RC LFSR Seed	The ASIC ADC RC LFSR seed register value.	2	
C2E5				
C2E6	ADC EF LFSR Seed	The ASIC ADC EF LFSR seed register value.	2	
C2E7				
C2E8	ADC Gain	The ASIC ADC Gain registry value.	1	
C2E9	ADC Offset	The ASIC ADC Offset registry value.	1	
C2EA	Structure Padding	Structure Padding	2	
C2EB				
C2EC	Sense_En	ASIC sense enable configuration.	4	
C2ED				
C2EE				
C2EF				
C2F0	E_Sel1	ASIC electrode selection 1 register.	4	
C2F1				
C2F2				
C2F3				
C2F4	E_Sel0	ASIC electrode selection 0 register.	4	
C2F5				
C2F6				
C2F7				
C2F8	ADCControl	ASIC ADC control.	1	
C2F9	Structure Padding	Structure Padding		
C2FA	Structure Padding	Structure Padding		
C2FB	Structure Padding	Structure Padding		
C2FC	DetectFingerSeed		2	
C2FD				
C2FE	DownStep		2	
C2FF				
C300	DownRange		2	
C301				
C302	**Structure Padding**	**PS/2 FF to AA00 delay time in 2 ms increments**	**2**	
C303				
C304	Access Offset Address	The read/write address offset. **Note** *This is applied to convert from logical address to physical address.*	4	
C305				
C306				

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C307				
C308	BG_Trim (WARNING: CHANGE WITH CARE)	ASIC BG trim setting.	1	
C309	Clock_Trim (WARNING: CHANGE WITH CARE)	ASIC clock trim setting.	1	
C30A	Z_Trim (WARNING: CHANGE WITH CARE)	ASIC Z trim setting.	1	
C30B	ButtonsMask	Allows button data to be masked. One bit per button.	1	
C30C	X Delta Threshold	X motion delta threshold for Filtering Algorithms.	1	
C30D	Y Delta Threshold	Y motion delta threshold for Filtering Algorithms.	1	
C30E	PalmCornerTimeout	Frame count for timeout on corner palm rejection.	1	
C30F	Jitter Same Max	Same maximum setting for the Jitter Filter.	1	
C310	Jitter Opposite Max	Opposite maximum setting for Jitter Filter.	1	
C311	Jitter Engage Threshold	Engagement threshold for Jitter Filter.	1	
C312	EF Channel 1	Electrode toggle frequency channel 1.	1	
C313	EF Channel 2	Electrode toggle frequency channel 2.	1	
C314	Max Consecutive Noisy Dataset Discards	Maximum number of noisy data sets that can be discarded.	1	
C315	TrackEF	The current tracking electrode frequency.	1	
C316	ApplyBackgroundCompLim	The signal limit allowed for background compensation.	1	
C317	**RESERVED**	**RESERVED**	**1**	
C318	MeasConfig2.ADTestConfig	ASIC ADC test register value.	1	
C319	MeasConfig2.ADSpare	ASIC ADC spare register value.	1	
C31A	MeasConfig2.AWControl	ASIC ADC aperture window control register value.	1	
C31B	MeasConfig2.EtogControl	ASIC ADC electrode toggle register value.	1	
C31C	AnyMeasData1.RC_LFSR_Seed	ASIC ADC RC LFSR seed register value.	2	
C31D				
C31E	AnyMeasData1.EF_LFSR_Seed	ASIC EF LFSR seed register value.	2	
C31F				
C320	MeasConfig2.ADGain	ASIC ADC Gain register value.	1	
C321	MeasConfig2.ADOffset	ASIC ADC offset register value.	1	
C322	Structure Padding	Structure Padding	1	
C323				
C324	MeasConfig2.SenseEN	ASIC sense enable configuration.	4	
C325				
C326				
C327				
C328	MeasConfig2.ESel1	ASIC electrode selection 1 register.	4	
C329				
C32A				
C32B				
C32C	MeasConfig2.ESel0	ASIC electrode selection 0 register.	4	
C32D				
C32E				
C32F				

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C330	MeasConfig2.ADControl	ASIC ADC control.	1	
C331	Structure Padding	Structure Padding		
C332	Structure Padding	Structure Padding		
C333	Structure Padding	Structure Padding		
C334	FZScaler	Scale reported Z based on 255 full-scale.	2	
C335				
C336	TetheredPenFilterControl	Enable tethered pen smoothing filter.	1	
C337	TetheredPenConfig		1	
	TetheredPenConfig.PenOnly	Disable finger tracking on tethered pen systems.		
	TetheredPenConfig.FingerOnly	Only track fingers on tethered pen systems.		
	TetheredPenConfig.PenPriority	Track both, but only report pen when it's present.		
	TetheredPenConfig.LinearCorrectionEnable	Enable tethered pen linear correction.		
	TetheredPenConfig.ZCorrectionEnable	Enable tethered pen signal normalization.		
	TetheredPenConfig.AverageZXZY	Average the signal between the two axes.		
	TetheredPenConfig.Unused6	Not used.		
	TetheredPenConfig.Unused7	Not used.		
C338	TetheredPenZContactThreshold	Signal threshold for tethered pen tracking.	1	
C339	TetheredPenZDownShift	Signal attenuation of tethered pen.	1	
C33A	TetheredPenNumSmoothingElements	Sample depth of Tethered Pen Smoothing Filter.	1	
C33B	MinOffsetElementsForCorrection	Minimum valid offset elements to allow offset correction.	1	
C33C	OffsetCorrectionUpdateInterval	Interval at which offset correction is applied.	1	
C33D	PenFingerMode		1	
C33E	OffsetCorrectionMaxNewOffset	Maximum offset that can be applied per correction.	2	
C33F				
C340	OffsetCorrectionPosThreshold	Maximum positive value allowed for offset correction element.	2	
C341				
C342	OffsetCorrectionNegThreshold	Minimum positive value allowed for offset correction element.	2	
C343				
C344	BckGndCompCounterReload	Interval reload for the background compensation counter.	2	
C345				
C346	Delta -> 0	Delta acceleration table value 0. **Note** *For each 1/15th of full-scale actual movement, this is the amount of augmentation to apply for an acceleration effect (relative mode only).*	1	
C347	Delta -> 1	Delta acceleration table value 1.	1	
C348	Delta -> 2	Delta acceleration table value 2.	1	
C349	Delta -> 3	Delta acceleration table value 3.	1	

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C34A	Delta -> 4	Delta acceleration table value 4.	1	
C34B	Delta -> 5	Delta acceleration table value 5.	1	
C34C	Delta -> 6	Delta acceleration table value 6.	1	
C34D	Delta -> 7	Delta acceleration table value 7.	1	
C34E	Delta -> 8	Delta acceleration table value 8.	1	
C34F	Delta -> 9	Delta acceleration table value 9.	1	
C350	Delta -> 10	Delta acceleration table value 10.	1	
C351	Delta -> 11	Delta acceleration table value 11.	1	
C352	Delta ->12	Delta acceleration table value 12.	1	
C353	Delta -> 13	Delta acceleration table value 13.	1	
C354	Delta -> 14	Delta acceleration table value 14.	1	
C355	Delta -> 15	Delta acceleration table value 15.	1	
C356	X Physical Size	The X-physical size reported in the report descriptor.	2	
C357				
C358	Y Physical Size	The Y-physical size reported in the report descriptor.	2	
C359				
C35A	Signal Sign Pos Threshold	Positive threshold for the Signal Sign Compensation Algorithm.	2	
C35B				
C35C	Signal Sign Neg Threshold	Negative threshold for the Signal Sign Compensation Algorithm.	2	
C35D				
C35E	Signal Sign Accum Neg Counter Max	Maximum signal sign accumulator counts before compensation.	1	
C35F	PalmResCorner	Proprietary Tracking Algorithm Setting.		
C360	Bad Congruence Count Max	Maximum bad congruence counts before compensation.	2	
C361				
C362	Min Non-Congruence	Minimum amount of non-congruence allowed.	2	
C363				
C364	PwrCtrl_PostButtonDeepSleepDelay	Delay before sleep after a button event.	2	
C365				
C366	Average Noise Max	Maximum noise before noise avoidance activates.	2	
C367				
C368	Average Noise	Current noise value.	2	
C369				
C36A	X Axis Scaler	X-axis relative motion scaler (256 nominal).	2	
C36B				
C36C	Y Axis Scaler	Y-axis relative motion scaler (256 nominal).	2	
C36D				
C36E	CompCount	Current compensation count.	1	
C36F	BckGndCompMinElements	Minimum required elements to allow background compensation.	1	

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C370	Memory Access Key	Key required to access to protected memory.	2	
C371				
C372	Meas Result Scale Multiplier	Multiplier used when scaling measurements results.	2	
C373				
C374	PalmRejectionTimeout	Proprietary Tracking Algorithm Setting.	1	
C375	PalmResMag	Proprietary Tracking Algorithm Setting.	1	
C376	MaxUpSeek	Proprietary Tracking Algorithm Setting.	1	
C377	PalmResEdgeEW	Proprietary Tracking Algorithm Setting.	1	
C378	PalmResEdgeN	Proprietary Tracking Algorithm Setting.	1	
C379	FingerControl	Proprietary Tracking Algorithm Setting.	1	
C37A	DownBottom	Proprietary Tracking Algorithm Setting.	2	
C37B				
C37C	ButtonRegionLimitYIn	Coordinate in boundary for button region.	2	
C37D				
C37E	ButtonRegionLimitYOut	Coordinate out boundary for button region.	2	
C37F				
C37A	DownBottom	Down Bottom.	2	
C37B				
C37C	ButtonRegionLimitYIn	Coordinate in boundary for button region.	2	
C37D				
C37E	ButtonRegionLimitYOut	Coordinate out boundary for button region.	2	
C37F				
C380	PowerControl Deep-Sleep Stage 1 Time (ms)	Milliseconds to sleep in stage 1 deep sleep.	2	
C381				
C382	PwrCtrl_PostTrackDeepSleepDelay	Delay before sleep after a tracking event.	2	
C383				
C384	XSensitivityPeak	Reported value for peak signal on X-axis.	2	
C385				
C386	YSensitivityPeak	Reported value for peak signal on Y-axis.	2	
C387				
C388	GestureSettle	Proprietary Gesture Algorithm Setting.	1	
C389	GestureTouchToCursor	Proprietary Gesture Algorithm Setting.	1	
C38A	GestureBetweenToIdle	Proprietary Gesture Algorithm Setting.	1	
C38B	GestureResettle	Proprietary Gesture Algorithm Setting.	1	
C38C	GestureRestouchToDrag	Proprietary Gesture Algorithm Setting.	1	
C38D	GestureControl	Proprietary Gesture Algorithm Setting.	1	
C38E	TapMaxPosDiff	Proprietary Gesture Algorithm Setting.	1	
C38F	RawMeasGainShift	Proprietary Gesture Algorithm Setting.	1	
C390	SuppressDeltasOnButtonChange	Proprietary Tracking Algorithm Setting.	1	
C391	ExponentialDXY	Proprietary Tracking Algorithm Setting.	1	

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C392	PalmResOnset	Proprietary Tracking Algorithm Setting.	1	
C393	MaskScalerX	Proprietary Tracking Algorithm Setting.	1	
C394	UpCheck	Proprietary Tracking Algorithm Setting.	2	
C395				
C396	ALPS IDConfig Byte 1	ALPS IDConfig Byte 1.	1	
C397	ALPS IDConfig Byte 2	ALPS IDConfig Byte 2.	1	
C398	ALPS IDConfig Byte 3	ALPS IDConfig Byte 3.	1	
C399	ALPS IDConfig Byte 4	ALPS IDConfig Byte 4.	1	
C39A	ALPS IDConfig Byte 5	ALPS IDConfig Byte 5.	1	
C39B	ALPS IDConfig Byte 6	ALPS IDConfig Byte 6.	1	
C39C	ALPS IDConfig Byte 7	ALPS IDConfig Byte 7.	1	
C39D	ALPS IDConfig Byte 8	ALPS IDConfig Byte 8.	1	
C39E	ALPS IDConfig Byte 9	ALPS IDConfig Byte 9.	1	
C39F	ALPS IDConfig Byte 10	ALPS IDConfig Byte 10.	1	
C3A0	TapTwoMaxPosDiff	Proprietary gesture algorithm setting.	1	
C3A1	MaskScalerY	Proprietary Tracking Algorithm Setting.	1	
C3A2	PalmDetect_GPSuppressTap	Proprietary Tracking Algorithm Setting.	1	
C3A3	PalmControl	Proprietary Tracking Algorithm Setting.	1	
C3A4	EFOffset_Ch2	EF offset applied for EF channel 2.	2	
C3A5				
C3A6	XPanDeltaThreshold	Horizontal scroll granularity/movement threshold per tick.	1	
C3A7	YPanDeltaThreshold	Vertical scroll granularity/movement threshold per tick.	1	
C3A8	PalmMaskX0		2	
C3A9				
C3AA	PalmResEnd		1	
C3AB	GestureRecovery		1	
C3AC	GestureReject		1	
C3AD	GestureReleaseTwo		1	
C3AE	PowerControl Deep-Sleep Stage 2 Time (ms)	Milliseconds to sleep in stage 2 deep sleep.	2	
C3AF				
C3B0	PwrCtrl_DeepSleepStageChangeDelay	Delay between deep sleep stages.	2	
C3B1				
C3B2	X Logical Scaler.Actual Min	Logical scaler actual minimum for X-axis.	2	
C3B3				
C3B4	X Logical Scaler.Actual Max	Logical scaler actual maximum for X -axis.	2	
C3B5				
C3B6	X Logical Scaler.Logical Min	Logical scaler logical minimum for X-axis.	2	
C3B7				
C3B8	X Logical Scaler.Logical Max	Logical scaler logical maximum for X-axis.	2	
C3B9				
C3BA	Y Logical Scaler.Actual Min	Logical scaler actual minimum for Y-axis.	2	

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C3BB				
C3BC	Y Logical Scaler.Actual Max	Logical scaler actual maximum for Y-axis.	2	
C3BD				
C3BE	Y Logical Scaler.Logical Min	Logical scaler logical minimum for Y-axis.	2	
C3BF				
C3C0	Y Logical Scaler.Logical Max	Logical scaler logical maximum for Y-axis.	2	
C3C1				
C3C2	PalmMaskX1	Proprietary Palm Detection Algorithm Setting.	2	
C3C3				
C3C4	PalmMaskX2	Proprietary Palm Detection Algorithm Setting.	2	
C3C5				
C3C6	PalmMaskX3	Proprietary Palm Detection Algorithm Setting.	2	
C3C7				
C3C8	PalmMaskX4	Proprietary Palm Detection Algorithm Setting.	2	
C3C9				
C3CA	PwrCtrl_WakeupCount	Number of stage 2 sleep cycles with touch before wake up.	1	
C3CB	StablilzerStickDistance	Proprietary Tracking Algorithm Setting.	1	
C3CC	StablilzerSlipInDistance	Proprietary Tracking Algorithm Setting.	1	
C3CD	ZXPeakScale	Scaler value for reported X sensitivity.	1	
C3CE	ZYPeakScale	Scaler value for reported Y sensitivity.	1	
C3CF	Structure Padding	Structure Padding		
C3D0	XEdgeClipMin	Proprietary Tracking Algorithm Setting.	2	
C3D1				
C3D2	XEdgeClipMax	Proprietary Tracking Algorithm Setting.	2	
C3D3				
C3D4	YEdgeClipMin	Proprietary Tracking Algorithm Setting.	2	
C3D5				
C3D6	YEdgeClipMax	Proprietary Tracking Algorithm Setting.	2	
C3D7				
C3D8	XEdgeCorrectionStart0	Proprietary Tracking Algorithm Setting.	1	
C3D9	XEdgeCorrectionDiv0	Proprietary Tracking Algorithm Setting.	1	
C3DA	XEdgeCorrectionStart1	Proprietary Tracking Algorithm Setting.	1	
C3DB	XEdgeCorrectionDiv1	Proprietary Tracking Algorithm Setting.	1	
C3DC	YEdgeCorrectionStart0	Proprietary Tracking Algorithm Setting.	1	
C3DD	YEdgeCorrectionDiv0	Proprietary Tracking Algorithm Setting.	1	
C3DE	YEdgeCorrectionStart1	Proprietary Tracking Algorithm Setting.	1	
C3DF	YEdgeCorrectionDiv1	Proprietary Tracking Algorithm Setting.	1	
C3E0	XEdgeStabilizerMin	Minimum value for edge stabilization of X axis.	2	
C3E1				
C3E2	XEdgeStabilizerMax	Maximum value for edge stabilization of X axis.	2	

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C3E3				
C3E4	YEdgeStabilizerMin	Minimum value for edge stabilization of Y axis.	2	
C3E5				
C3E6	YEdgeStabilizerMax	Maximum value for edge stabilization of Y axis.	2	
C3E7				
C3E8	PwrCtrl_PostSPDeepSleepDelay	Delay before sleep after a PS2 aux event.	2	
C3E9				
C3EA	PwrCtrl_PostTrackDeepSleepDelayPS2	Delay before sleep after a PS2 event.	2	
C3EB				
C3EC	GestureSuiteEnable	Per-gesture enable/disable. **Note** *Only available on some versions.*	2	
	ScrollEnable	Enable Scroll Gesture.		
	PanEnable	Enable Pan Gesture.		
	ZoomEnable	Enable Zoom Gesture.		
	RotateEnable	Enable Rotate Gesture.		
	Vertical3FingerSwipeEnable	Enable Vertical, Three-finger Swipe Gesture.		
	Horizontal3FingerSwipeEnable	Enable Horizontal, Three-finger Swipe Gesture.		
	XEdgeSwipeEnable	Enable Windows8, Left and Right Edge Gestures.		
	YEdgeSwipeEnable	Enable Windows8, Top Edge Gesture.		
	GlideExtendEnable	Enables the Cirque GlideExtendEnable Feature.		
	GestureSuite9	**RESERVED**		
	GestureSuite10	**RESERVED**		
	GestureSuite11	**RESERVED**		
	GestureSuite12	**RESERVED**		
	RotateControlsVolume	Rotate issues data for HID consumer device volume control.		
	GestureSuite14	**RESERVED**		
	GestureSuite15	**RESERVED**		
C3ED				
C3EE	GlideExtendMarginSize	Margin around edge of sensor to activate GlideExtend. **Note** *Set to 0 for drag lock function.*	2	
C3EF				
C3F0	GlideExtendTimeoutSamples	Number of frames to hold GlideExtend function before timeout.	1	
C3F1	Structure Padding	Structure Padding		
C3F2	RightTapeZoneXEngage	Coordinate boundary for Tap Zone.	2	
C3F3				
C3F4	ScrollZoneXEngage	Coordinate boundary for Scroll Zone.	2	
C3F5				
C3F6	FingerDetectControl		1	
	FingerDetectControl.Enable	Enable low-power object detection for wakeup.		
	FingerDetectControl.reserved1			

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
	FingerDetectControl.forcePeriodicTrackCycle	Force a full-image scan periodically to detect object presence.		
	FingerDetectControl.forceBackgroundCompCheck	Force background compensation during Low-power Mode.		
	FingerDetectControl.reserved4			
	FingerDetectControl.reserved5			
	FingerDetectControl.reserved6			
	FingerDetectControl.reserved7			
C3F7	FingerDetectWakeupDelayCountMax	Number of cycles between object detection scans.	1	
C3F8	FingerDetectThreshold	Signal threshold to exit low-power object detection mode.	2	
C3F9				
C3FA	FingerDetectMaxCycleCountStage1	Number of idle cycles before entering finger detect mode stage 1.	2	
C3FB				
C3FC	FingerDetectMaxCycleCountStage2	Number of idle cycles before entering finger detect mode stage 2.		
C3FD			2	
C3FE	FingerDetectCompQuantumOld	(READ) dynamic detection signal value previous.		
C3FF	FingerDetectCompQuantumNEW	(READ) dynamic detection signal value new.	1	
C400	FingerDetectCurrentDetectValue	(READ) current signal value for low power object detection.	2	
C401				
C402	PalmStickyNormalTime	Proprietary Tracking Algorithm Setting.	1	
C403	DownStepDivisor	Proprietary Tracking Algorithm Setting.	1	
C404	PenThreshold	Stylus tracking threshold.	2	
C405				
C406	TetheredPenWidePresenceThreshold	Tethered pen tracking threshold stage 1.	2	
C407				
C408	TetheredPenL1L2Threshold	Tethered pen tracking threshold stage 2.	2	
C409				
C40A	TetheredPenDetectThreshold	Tethered pen disconnect detection threshold.	2	
C40B				
C40C	TetheredPenNoDetectCycleInterval	How often to check for tethered pen disconnection.	2	
C40D				
C40E	TetheredPenNoWideToDetectCycleInterval	Timeout after tethered pen idle.	2	
C40F				
C410	TetheredPenWideButNoNarrowToDetectCycleInterval		2	
C411				
C412	SecurityCongruenceTheshold	Accumulated signal thresh at boot to trigger por congruence trip.	2	
C413				
C414	SecurityDisconnectThreshold	Accumulated signal min per frame before disconnect trip.	2	
C415				
C416	SecurityHotRowThreshold	Accumulate signal max per row before “hot row” trip.	2	

Table 8. Gen 4 Command Register - (continued)

Logical Address	Name	Description	Size (Bytes)	Bit Number
C417				
C418	SecurityStatus		1	
	SecurityStatus.0	Security status indicator. 0 = System nominal. 1 = Sensor disconnect. 2 = Bug detected. 3 = Overlay detected. 4 = Pen disconnect.		
	SecurityStatus.1			
	SecurityStatus.2			
	SecurityStatus.ForcePORIntrusionRecheck	Force a recheck of the POR intrusion test.		
	SecurityStatus.DefeatSensorGone	Disable sensor disconnect test.		
	SecurityStatus.DefeatHotRow	Disable bug detection test.		
	SecurityStatus.DefeatPORIntrusion	Disable POR intrusion test.		
	SecurityStatus.DefeatTPenDisconnect	Disable tethered pen disconnect test.		
C419	Vacant1		1	
C41A	Vacant2		1	
C41B	Vacant3		1	
C41C	Vacant4		1	
C41D	Vacant5		1	
C41E	ActiveConfigPageKey		2	
C41F				

8. Enabling and Disabling Gestures

The *GestureSuiteEnable* register is located at address 0x*C3EC* (see *Extended Memory Table on page 22*). Following the convention described in *Extended Memory Access on page 18*, this section will provide examples to enable specific gestures. See the following examples:

- *Example 1: Enable Scroll and Zoom*
- *Example 2: Enable Multiple Gestures*

8.1 Example 1: Enable Scroll and Zoom

8.1.0.1 Legend

S	=	Start (I2C 2-wire start condition)
W	=	Write
R	=	Read
A	=	ACK (Acknowledge)
N	=	NAK (NOT-Acknowledge)
P	=	STOP (I2C stop condition)

In this example, the Scroll and Zoom gestures will be enabled and all others will be disabled. Scroll is controlled with bit 0, and Zoom with bit 2. The output data will be:

Output = 0x0005

The entire data exchange is described in the table below

Note: *The I2C slave address is 0x2A. As explained in Gen 4 Low-level I2C Interface on page 5, the value used for a write command is 0x54.*

S	Slave Addr (54)	W	A	Ext Access - L (00)	A	Ext Access - H (09)	A	Ext Addr 0 (EC)	A	Ext Addr 1 (C3)	A	Ext Addr 2 (00)	A	Ext Addr 3 (00)	A
				Ext Access Register WRITE				Extended Address							

Length - L (02)	A	Length - H (00)	A	Byte 0 (05)	A	Byte 1 (00)	A	Checksum (BF)	P
Number of Bytes				Byte Data					

No further action is required to enable either the Scroll or Zoom gesture, but disable all other gestures.

8.2 Example 2: Enable Multiple Gestures

This example describes how to enable multiple gestures, avoiding the RESERVED and non-supported bits. The supported gestures are:

- SCROLL: 0x01
- PAN: 0x02
- ZOOM: 0x04
- 3-FINGER VERTICAL SWIPE: 0x10
- 3-FINGER HORIZONTAL SWIPE: 0x20

Our output word is: 0x0037

The I2C transaction is then:

S	Slave Addr (54)	W	A	Ext Access - L (00)	A	Ext Access - H (09)	A	Ext Addr 0 (EC)	A	Ext Addr 1 (C3)	A	Ext Addr 2 (00)	A	Ext Addr 3 (00)	A
				Ext Access Register WRITE				**Extended Address**							

Length - L (02)	A	Length - H (00)	A	Byte 0 (37)	A	Byte 1 (00)	A	Checksum (F1)	P
Number of Bytes				**Byte Data**					

The gestures will be enabled immediately upon the completion of this command.

9. Appendix A

The supplied USB-I2C Bridge 1p3.zip file provides code examples for integrating the Gen 4 sensor. Two file types are used, C file (code) and H file (H source code). The C files contain the variable declarations, instructions, functions, loops, and other statements that define how to interface with the Gen 4 platform. The H file contains the C declarations and definitions that will be shared between several source files.

10. Contact Information

Contact a Cirque sales representative for a complete list of Cirque's OEM products.

In United States & Canada	(800) GLIDE-75 (454-3375)
Outside US & Canada	(801) 467-1100
Fax	(801) 467-0208
Web site	http://www.cirque.com

THIS INFORMATION IS PROVIDED “AS IS.” CIRQUE SPECIFICALLY DISCLAIMS ALL WARRANTIES EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENTATION AND USE OF THE DOCUMENTATION IN DESIGN OF ANY PRODUCTS OR FOR ANY PARTICULAR APPLICATION.