Features

- maXTouch™ Touchscreen
  - Two touchscreens with true 12-bit multiple touch reporting and real-time XY tracking for up to 16 concurrent touches per touchscreen
  - Screen sizes 8 – 12.1 inches diagonal supported at 5 mm electrode pitch.
- Number of Channels
  - Electrode grid configurations of up to 32 X and 52 Y lines supported
  - Touchscreens up to 1664 channels (subject to other configurations)
  - Up to 32 channels can be allocated as fixed keys (subject to other configurations)
- Signal Processing
  - Advanced digital filtering using both hardware engine and firmware
  - Self-calibration
  - Auto drift compensation
  - Adjacent Key Suppression® (AKS®) technology
  - Noise cancellation algorithms for display noise suppression
  - Grip suppression and suppression of unintentional touches
  - Down-scaling and clipping support to match LCD resolution
  - Ultra-fast start-up and calibration for best user experience
  - Supports axis flipping and axis switch-over for portrait and landscape modes
  - Fast and powerful 32-bit processor core
- Scan Speed
  - Maximum single touch >250Hz, subject to configuration
  - Maximum 16 touches >100Hz, subject to configuration
  - Configurable to allow power/speed optimization
  - Programmable timeout for automatic transition from active to idle states
- Response Times
  - Initial latency <15 ms for first touch from idle, subject to configuration
- Sensors
  - Works with PET or glass sensors, including curved profiles
  - Works with all proprietary sensor patterns recommended by Atmel®
- Stylus Support
  - Supports passive stylus with 2 mm contact diameter, subject to configuration
  - Supports Atmel maXStylus™, subject to configuration
- Environmental Conditions
  - Operating temperature –40°C to +85°C
  - Moisture tolerance good
- Panel Thickness
  - Glass up to 2.5 mm, screen size dependent
  - Plastic up to 1.2 mm, screen size dependent
- Interfaces
  - I^{2}C-compatible slave mode; Standard/Fast Mode: up to 400 kHz, High speed mode: up to 1.7 MHz
  - USB 2.0-compliant composite device, full speed (12 Mbps)
  - HID-I^{2}C interface for Microsoft® Windows® 8
- Power
  - Digital 1.8 V (I^{2}C-compatible mode only) or 2.7 V to 3.3 V nominal
  - Analog 2.7 V to 3.3 V nominal
  - High voltage X line drive up to 6.0 V nominal
- Package
  - 128-ball VFBGA 7 × 7 × 1 mm, 0.5 mm ball pitch

Atmel®

maXTouch™ 1664-channel Touchscreen Controller

mXT1664S Revision 1.0

maXTouch™

1. Overview of the mXT1664S

1.1 Introduction

The Atmel maXTouch family of touch controllers has set a new industry benchmark for capacitive touchscreens with their low current consumption, fast response time and high levels of accuracy. The mXT1664S single-chip solution offers the benefits of the maXTouch enhanced architecture on devices with touchscreens up to 14 in. diagonal:

- **Patented capacitive sensing method** – The mXT1664S uses a unique charge-transfer acquisition engine to implement the Atmel-patented QMatrix® capacitive sensing method. This allows the measurement of up to 1664 mutual capacitance nodes. Coupled with a state-of-the-art CPU, the entire touchscreen sensing solution can measure, classify and track individual finger touches with a high degree of accuracy.
- **Capacitive Touch Engine (CTE)** – The mXT1664S features an acquisition engine, which uses an optimal measurement approach to ensure almost complete immunity from parasitic capacitance on the receiver inputs (Y lines). The engine includes sufficient dynamic range to cope with anticipated touchscreen mutual capacitances, which allows great flexibility for use with the Atmel proprietary ITO pattern designs. One and two layer ITO sensors are possible using glass or PET substrates.
- **Noise filtering** – Hardware noise processing in the capacitive touch engine provides enhanced autonomous filtering and allows a broad range of noise profiles to be handled. The result is good performance in the presence of charger and LCD noise.
- **Processing power** – The main CPU has two powerful, yet low power, microsequencer coprocessors under its control. These combine to allow the signal acquisition, preprocessing, postprocessing and housekeeping to be partitioned in an efficient and flexible way. This gives ample scope for sensing algorithms, touch tracking or advanced shape-based filtering. An in-circuit reflash can be performed over the chip's hardware-driven interface.
- **Interpreting user intention** – The Atmel mutual capacitance method provides unambiguous multitouch performance. Algorithms in the mXT1664S provide optimized touchscreen position filtering for the smooth tracking of touches. Stylus support allows stylus touches to be detected and distinguished from other touches, such as finger touches. The suppression of unintentional touches from the user's gripping fingers, resting palm or touching cheek or ear also help ensure that the user's intentions are correctly interpreted.

1.2 Understanding Unfamiliar Concepts

If some of the concepts mentioned in this datasheet are unfamiliar, see the following sections for more information:

- Appendix C on page 80 for a glossary of terms
- Appendix D on page 82 for QMatrix technology

1.3 Resources

The following datasheet provide essential information on configuring the device:

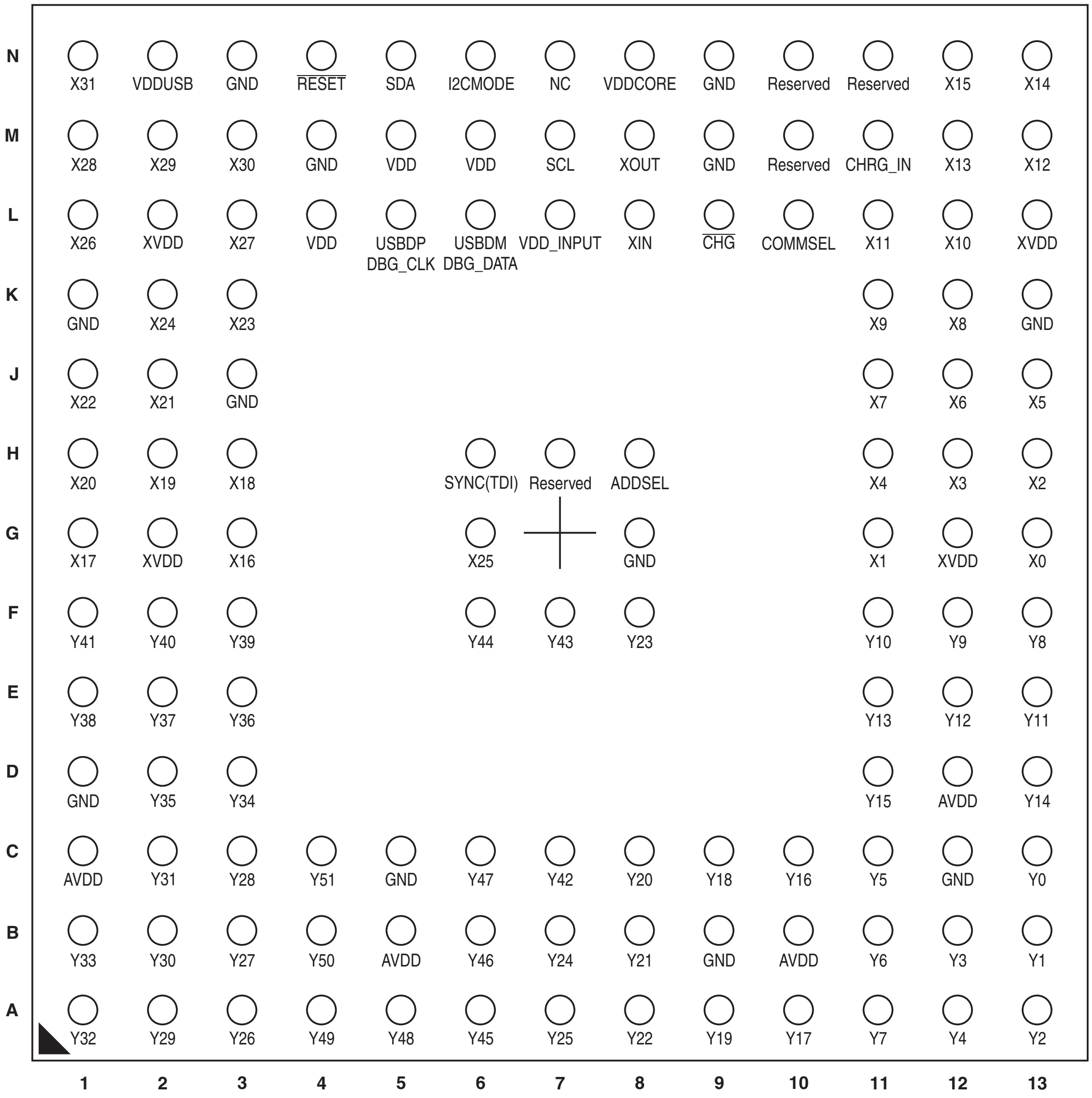
- *mXT1664S Protocol Guide*

The following documents may also be useful (available by contacting Atmel's Touch Technology Division):

- **Configuring the device:**
  - Application Note: QTAN0058 – *Rejecting Unintentional Touches with the maXTouch Touchscreen Controllers*
  - Application Note: QTAN0078 – *maXTouch Stylus Tuning*
- **Miscellaneous:**
  - Application Note QTAN0050 – *Using the maXTouch Debug Port*
  - Application Note QTAN0061 – *maXTouch™ Sensitivity Effects for Mobile Devices*
  - Application Note QTAN0086 – *Touchscreen Design for Gloved Operation*
- **Touchscreen design and PCB/FPCB layout guidelines:**
  - Application Note QTAN0054 – *Getting Started with maXTouch Touchscreen Designs*
  - Application Note QTAN0048 – *maXTouch PCB/FPCB Layout Guidelines*
  - Application Note QTAN0080 – *Touchscreens Sensor Design Guide*
- **Other documents –** The device uses the same core technology as the mXT768E, so the following documents may also be useful (available by contacting the Atmel Touch Technology division):
  - Application Note: QTAN0083 – *mXT768E Power and Speed Considerations*
  - Application Note QTAN0052 – *mXT224 Passive Stylus Support*

2. Pinout and Schematic

2.1 Pinout Configuration



	1	2	3	4	5	6	7	8	9	10	11	12	13
N	X31	VDDUSB	GND	$\overline{\text{RESET}}$	SDA	I2CMODE	NC	VDDCORE	GND	Reserved	Reserved	X15	X14
M	X28	X29	X30	GND	VDD	VDD	SCL	XOUT	GND	Reserved	CHRG_IN	X13	X12
L	X26	XVDD	X27	VDD	USBDP DBG_CLK	USBDM DBG_DATA	VDD_INPUT	XIN	$\overline{\text{CHG}}$	COMMSEL	X11	X10	XVDD
K	GND	X24	X23								X9	X8	GND
J	X22	X21	GND								X7	X6	X5
H	X20	X19	X18			SYNC(TDI)	Reserved	ADDSEL			X4	X3	X2
G	X17	XVDD	X16			X25		GND			X1	XVDD	X0
F	Y41	Y40	Y39			Y44	Y43	Y23			Y10	Y9	Y8
E	Y38	Y37	Y36								Y13	Y12	Y11
D	GND	Y35	Y34								Y15	AVDD	Y14
C	AVDD	Y31	Y28	Y51	GND	Y47	Y42	Y20	Y18	Y16	Y5	GND	Y0
B	Y33	Y30	Y27	Y50	AVDD	Y46	Y24	Y21	GND	AVDD	Y6	Y3	Y1
A	Y32	Y29	Y26	Y49	Y48	Y45	Y25	Y22	Y19	Y17	Y7	Y4	Y2

Bottom View

2.2 Pinout Descriptions

Table 2-1. Pin Listing

Ball	Name	Type	Comments	If Unused, Connect To...
A1	Y32	I	Y line connection	Leave open
A2	Y29	I	Y line connection	Leave open
A3	Y26	I	Y line connection	Leave open
A4	Y49	I	Y line connection	Leave open
A5	Y48	I	Y line connection	Leave open
A6	Y45	I	Y line connection	Leave open
A7	Y25	I	Y line connection	Leave open
A8	Y22	I	Y line connection	Leave open
A9	Y19	I	Y line connection	Leave open
A10	Y17	I	Y line connection	Leave open
A11	Y7	I	Y line connection	Leave open
A12	Y4	I	Y line connection	Leave open
A13	Y2	I	Y line connection	Leave open
B1	Y33	I	Y line connection	Leave open
B2	Y30	I	Y line connection	Leave open
B3	Y27	I	Y line connection	Leave open
B4	Y50	I	Y line connection	Leave open
B5	AVDD	P	Analog power	–
B6	Y46	I	Y line connection	Leave open
B7	Y24	I	Y line connection	Leave open
B8	Y21	I	Y line connection	Leave open
B9	GND	P	Ground	–
B10	AVDD	P	Analog power	–
B11	Y6	I	Y line connection	Leave open
B12	Y3	I	Y line connection	Leave open
B13	Y1	I	Y line connection	Leave open
C1	AVDD	P	Analog power	–
C2	Y31	I	Y line connection	Leave open
C3	Y28	I	Y line connection	Leave open
C4	Y51	I	Y line connection	Leave open
C5	GND	P	Ground	–
C6	Y47	I	Y line connection	Leave open
C7	Y42	I	Y line connection	Leave open
C8	Y20	I	Y line connection	Leave open
C9	Y18	I	Y line connection	Leave open

Table 2-1. Pin Listing

Ball	Name	Type	Comments	If Unused, Connect To...
C10	Y16	I	Y line connection	Leave open
C11	Y5	I	Y line connection	Leave open
C12	GND	P	Ground	–
C13	Y0	I	Y line connection	Leave open
D1	GND	P	Ground	–
D2	Y35	I	Y line connection	Leave open
D3	Y34	I	Y line connection	Leave open
D11	Y15	I	Y line connection	Leave open
D12	AVDD	P	Analog power	–
D13	Y14	I	Y line connection	Leave open
E1	Y38	I	Y line connection	Leave open
E2	Y37	I	Y line connection	Leave open
E3	Y36	I	Y line connection	Leave open
E11	Y13	I	Y line connection	Leave open
E12	Y12	I	Y line connection	Leave open
E13	Y11	I	Y line connection	Leave open
F1	Y41	I	Y line connection	Leave open
F2	Y40	I	Y line connection	Leave open
F3	Y39	I	Y line connection	Leave open
F6	Y44	I	Y line connection	Leave open
F7	Y43	I	Y line connection	Leave open
F8	Y23	I	Y line connection	Leave open
F11	Y10	I	Y line connection	Leave open
F12	Y9	I	Y line connection	Leave open
F13	Y8	I	Y line connection	Leave open
G1	X17	O	X matrix drive line	Leave open
G2	XVDD	P	X line drive voltage – see schematics in Section 2.3 on page 10	–
G3	X16	O	X matrix drive line	Leave open
G6	X25	O	X matrix drive line	Leave open

Table 2-1. Pin Listing

Ball	Name	Type	Comments	If Unused, Connect To...
G8	GND	P	Ground	–
G11	X1	O	X matrix drive line	Leave open
G12	XVDD	P	X line drive voltage – see schematics in Section 2.3 on page 10	–
G13	X0	O	X matrix drive line	Leave open
H1	X20	O	X matrix drive line	Leave open
H2	X19	O	X matrix drive line	Leave open
H3	X18	O	X matrix drive line	Leave open
H6	SYNC	I	External synchronization	Leave open
H7	Reserved	–	Reserved for future use	Leave open
H8	ADDSEL [1]	I	I^2C-compatible address select	Leave open (USB mode)
H11	X4	O	X matrix drive line	Leave open
H12	X3	O	X matrix drive line	Leave open
H13	X2	O	X matrix drive line	Leave open
J1	X22	O	X matrix drive line	Leave open
J2	X21	O	X matrix drive line	Leave open
J3	GND	P	Ground	–
J11	X7	O	X matrix drive line	Leave open
J12	X6	O	X matrix drive line	Leave open
J13	X5	O	X matrix drive line	Leave open
K1	GND	P	Ground	–
K2	X24	O	X matrix drive line	Leave open
K3	X23	O	X matrix drive line	Leave open
K11	X9	O	X matrix drive line	Leave open
K12	X8	O	X matrix drive line	Leave open
K13	GND	P	Ground	–
L1	X26	O	X matrix drive line	Leave open
L2	XVDD	P	X line drive voltage – see schematics in Section 2.3 on page 10	Leave open
L3	X27	O	X matrix drive line	Leave open
L4	VDD	P	Digital power	–

Table 2-1. Pin Listing

Ball	Name	Type	Comments	If Unused, Connect To...
L5	USBDP [1]	USB	USB device port data +	GND
	DBG_CLK	I/O	Debug clock	Input: GND Output: leave open
L6	USBDM [1]	USB	USB device port data -	GND
	DBG_DATA	I/O	Debug data	Input: GND Output: leave open
L7	VDD_INPUT	–	For factory use only	VDD
L8	XIN	I	External 16 MHz oscillator – only needed in USB mode	Leave open
L9	$\overline{\text{CHG}}$ [2]	OD	State change interrupt	Leave open (USB mode)
L10	COMMSEL	I	Selects communications interface: I^{2}C-compatible – connect to GND USB [3] – connect to Vdd	–
L11	X11	O	X matrix drive line	Leave open
L12	X10	O	X matrix drive line	Leave open
L13	XVDD	P	X line drive voltage – see schematics in Section 2.3 on page 10	Leave open
M1	X28	O	X matrix drive line	Leave open
M2	X29	O	X matrix drive line	Leave open
M3	X30	O	X matrix drive line	Leave open
M4	GND	P	Ground	–
M5	VDD	P	Digital power	–
M6	VDD	P	Digital power	–
M7	SCL [1]	OD	Serial Interface Clock	Connect to VDD (USB mode)
M8	XOUT	O	External 16 MHz oscillator – only needed in USB mode	Leave open
M9	GND	P	Ground	–
M10	Reserved	–	Reserved for future use	–
M11	CHRG_IN	I	Charger detect pin	GND via 10 kΩ
M12	X13	O	X matrix drive line	Leave open
M13	X12	O	X matrix drive line	Leave open
N1	X31	O	X matrix drive line	Leave open
N2	VDDUSB	P	Digital USB power	–
N3	GND	P	Ground	–
N4	$\overline{\text{RESET}}$	I	Reset low; has internal 20 kΩ to 60 kΩ pull-up resistor	Vdd [4]
N5	SDA [1]	OD	Serial Interface Data	Connect to VDD (USB mode)
N6	I2CMODE	I	I^{2}C-compatible protocol select [5]	Leave open
N7	Reserved	–	Reserved for future use	–

Table 2-1. Pin Listing

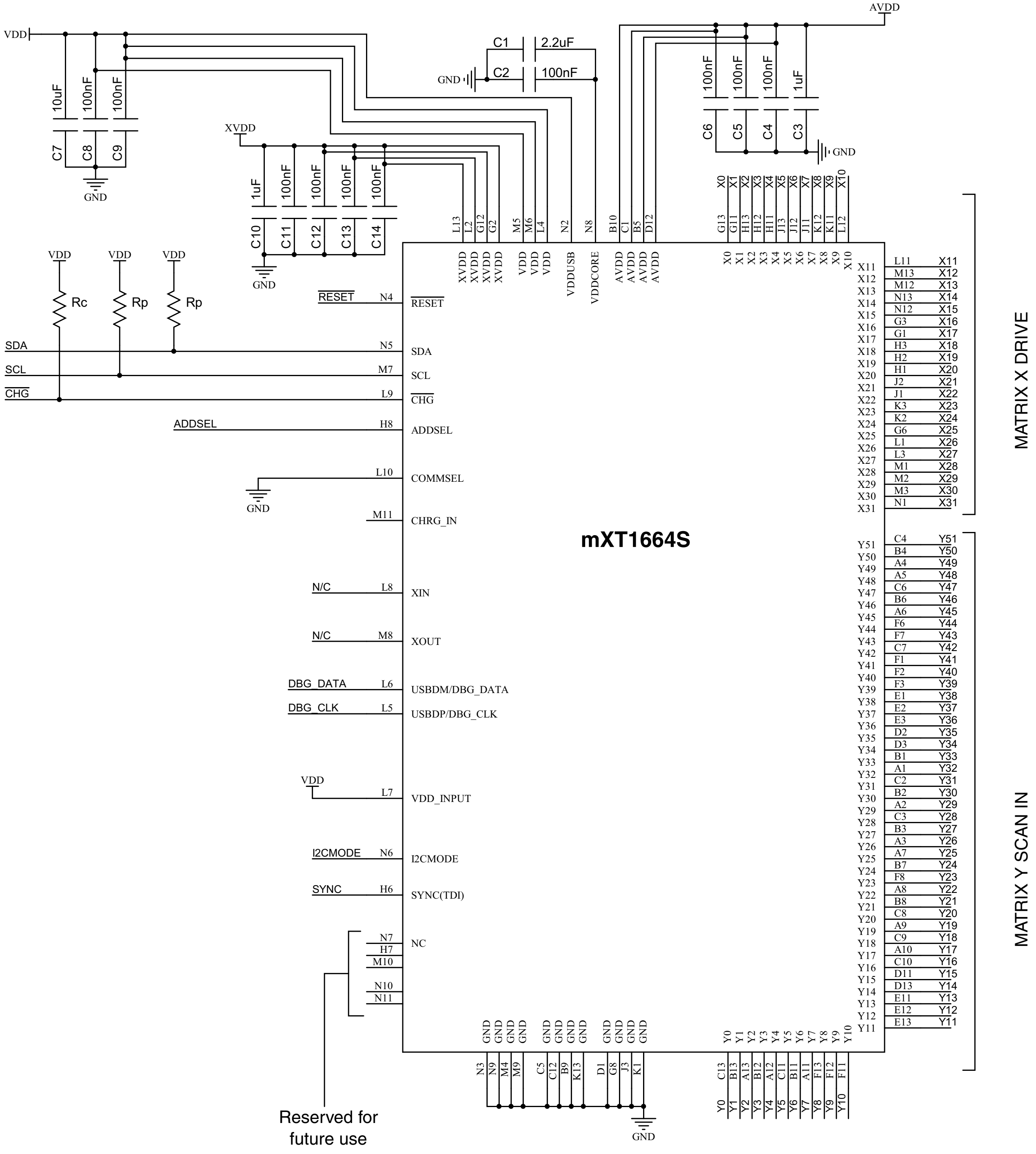
Ball	Name	Type	Comments	If Unused, Connect To...
N8	VDDCORE	P	Digital core power. Must be connected as in schematics (see Section 2.3 on page 10)	–
N9	GND	P	Ground	–
N10	Reserved	–	Reserved for future use	Leave open
N11	Reserved	–	Reserved for future use	Leave open
N12	X15	O	X matrix drive line	Leave open
N13	X14	O	X matrix drive line	Leave open

1. Either I^2C-compatible or USB interface can be used, but only one interface should be used in any one design.
2. $\overline{\text{CHG}}$ is momentarily set (approximately 100 ms) as an input after power-up or reset for diagnostic purposes.
3. If using the self-powered USB configuration (rare), this pin must be connected to VBUS via potential divider of a suitable value for the supply level. For example, with a 5V VBUS and 3.3V Vdd line, a 170 kΩ and 330 kΩ pair will typically be used.
4. It is recommend that $\overline{\text{RESET}}$ is connected to the host system.
5. Leave open for standard Atmel object protocol, or connect to GND to select the HID-I^2C mode.

I	Input only	OD	Open drain output	O	Output only, push-pull
USB	USB communications	P	Ground or power		

2.3 Schematics

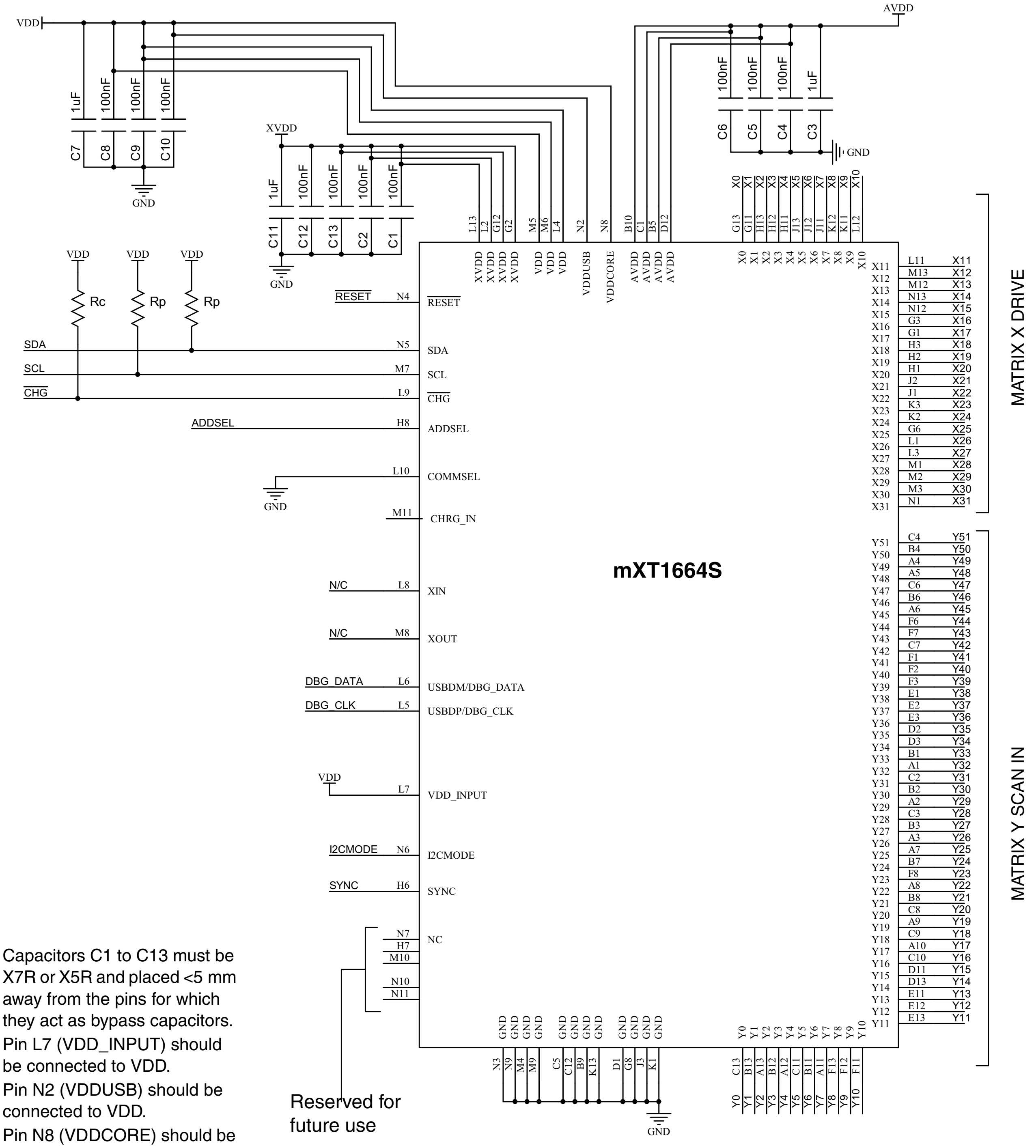
2.3.1 I²C-compatible Mode – Digital Supply 2.7–3.3V



2.3.1.1 Notes on I^{2}C-compatible Mode – Digital Supply 2.7 – 3.3V

1. Capacitors C1 to C14 must be X7R or X5R and placed <5 mm away from the pins for which they act as bypass capacitors
2. Pin L7 (VDD_INPUT) should be connected to VDD.
3. Pin N2 (VDDUSB) should be connected to VDD.
4. Pin N8 (VDDCORE) should be decoupled to GND through capacitors.
5. Either I2C-compatible or USB interface can be used, but only one interface should be used in any one design. Pin L10 (COMMSEL) should be connected accordingly. (GND in I^{2}C-compatible mode or VDD in USB mode).
6. The mXT1664S incorporates an internal regulator that derives the 1.8V VDDCORE supply from VDD. For stability of this regulator, a 10 µF input capacitor (C7) and a 2.2 µF output capacitor (C1) are required. Capacitor C1 should have an ESR of 0.5 Ω to 10 Ω.
7. Connect the I2CMODE pin to GND to select the HID-I^{2}C mode, leave NC to select normal Object based protocol mode.

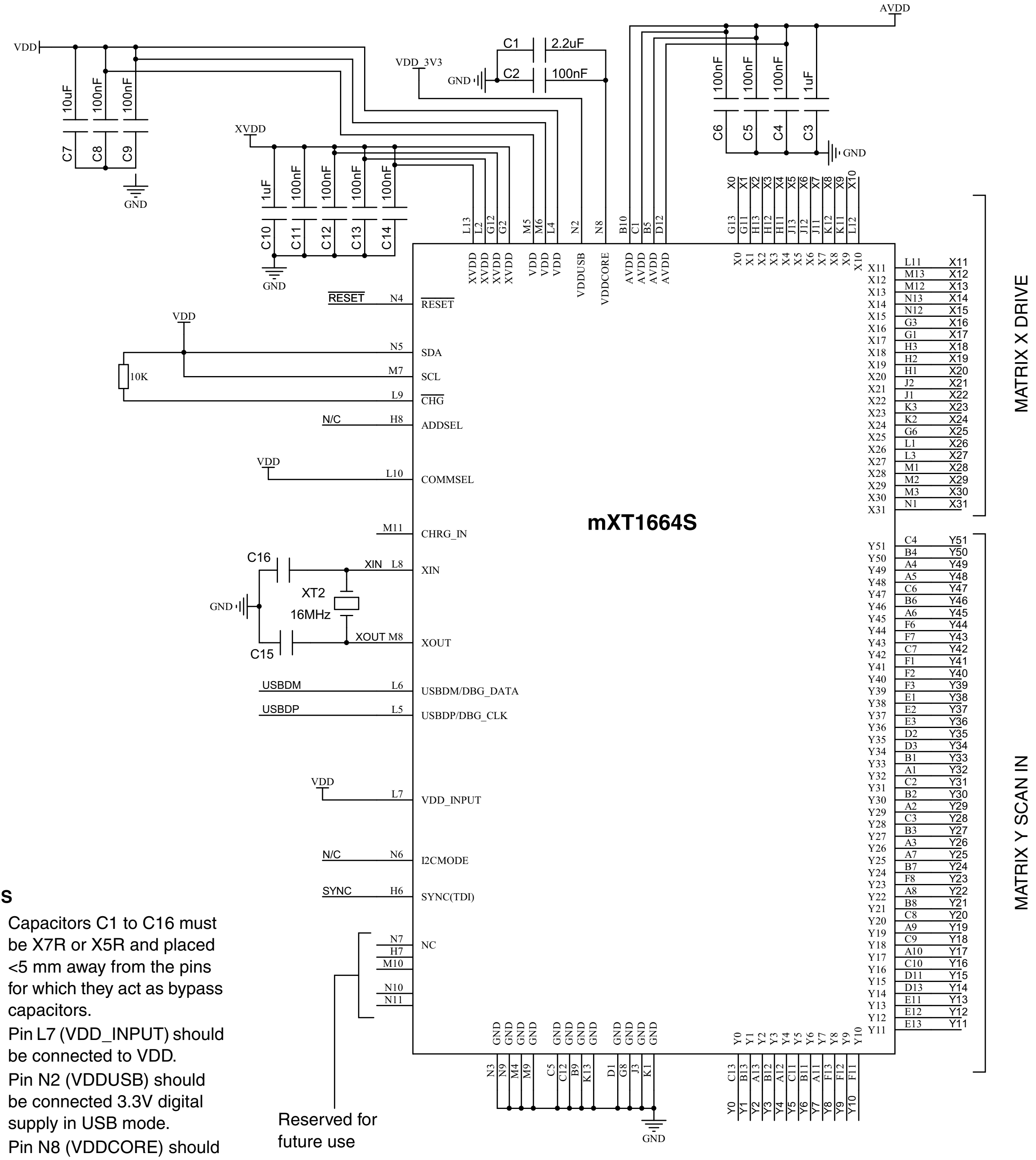
2.3.2 I²C-compatible Mode – Digital Supply 1.8V


mXT1664S
VDD
AVDD
XVDD
GND
1uF
100nF
C7
C8
C9
C10
C11
C12
C13
C2
C1
C6
C5
C4
C3
Rc
Rp
SDA
SCL
CHG
ADDSEL
RESET
N4
N5
M7
L9
H8
L10
COMMSEL
M11
CHRG_IN
N/C
L8
XIN
M8
XOUT
DBG_DATA
L6
USBDM/DBG_DATA
DBG_CLK
L5
USBDP/DBG_CLK
L7
VDD_INPUT
I2CMODE
N6
SYNC
H6
NC
N7
H7
M10
N10
N11
L13 L2 G12 G2
XVDD XVDD XVDD XVDD
M5 M6 L4
VDD VDD VDD
N2
VDDUSB
N8
VDDCORE
B10 C1 B5 D12
AVDD AVDD AVDD AVDD
X0 X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
G13 G11 H13 H12 H11 J13 J12 J11 K13 K12 K11 L12
X11–X31
L11 M13 M12 N13 N12 G3 G1 H3 H2 H1 J2 J1 K3 K2 G6 L1 L3 M1 M2 M3 N1
MATRIX X DRIVE
Y51–Y11
C4 B4 A4 A5 C6 B6 A6 F6 F7 C7 F1 F2 F3 E1 E2 E3 D2 D3 B1 A1 C2 B2 A2 C3 B3 A3 A7 B7 F8 A8 B8 C8 A9 C9 A10 C10 D11 D13 E11 E12 E13
MATRIX Y SCAN IN
GND
N3 N9 M4 M9 C5 C12 B9 K13 D1 G8 J3 K1
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10
C13 B13 A13 B12 A12 C11 B11 A11 F13 F12 F11
Reserved for future use

Notes

1. Capacitors C1 to C13 must be X7R or X5R and placed <5 mm away from the pins for which they act as bypass capacitors.
2. Pin L7 (VDD_INPUT) should be connected to VDD.
3. Pin N2 (VDDUSB) should be connected to VDD.
4. Pin N8 (VDDCORE) should be connected to VDD.
5. Pin L10 (COMMSEL) should be connected to GND.
6. No internal regulators used so C7 can be 1 µF.

2.3.3 USB Mode – Digital Supply 3.3V



NOTES

1. Capacitors C1 to C16 must be X7R or X5R and placed <5 mm away from the pins for which they act as bypass capacitors.
2. Pin L7 (VDD_INPUT) should be connected to VDD.
3. Pin N2 (VDDUSB) should be connected 3.3V digital supply in USB mode.
4. Pin N8 (VDDCORE) should be decoupled to GND via capacitors.
5. Pin L10 (COMMSEL) should be connected to VDD.
6. Capacitors C15 and C16 should be 15 pF

3. Touchscreen Basics

3.1 Sensor Construction

A touchscreen is usually constructed from a number of transparent electrodes. These are typically on a glass or plastic substrate. They can also be made using non-transparent electrodes, such as copper or carbon. Electrodes are normally formed by etching a material called Indium Tin Oxide (ITO). This is a brittle ceramic material, of high optical clarity and varying sheet resistance. Thicker ITO yields lower levels of resistance (perhaps tens to hundreds of Ω/square) at the expense of reduced optical clarity. Lower levels of resistance are generally more compatible with capacitive sensing. Thinner ITO leads to higher levels of resistance (perhaps hundreds to thousands of Ω/square) with some of the best optical characteristics.

Interconnecting tracks formed in ITO can cause problems. The excessive RC time constants formed between the resistance of the track and the capacitance of the electrode to ground can inhibit the capacitive sensing function. In such cases, ITO tracks should be replaced by screen printed conductive inks (non-transparent) outside the touchscreen's viewing area.

A range of trade-offs also exist with regard to the number of layers used for construction. Atmel has pioneered single-layer ITO capacitive touchscreens. For many applications these offer a near-optimum cost/performance balance. With a single layer screen, the electrodes are all connected using ITO out to the edges of the sensor. From there the connection is picked up with printed silver tracks. Sometimes two overprinted silver tracking layers are used to reduce the margins between the edge of the substrate and the active area of the sensor.

Two-layer designs can have a strong technical appeal where ultra-narrow edge margins are required. They are also an advantage where the capacitive sensing function needs to have a very precise cut-off as a touch is moved to just off the active sensor area. With a two-layer design the QMatrix transmitter electrodes are normally placed nearest the bottom and the receiver electrodes nearest the top. The separation between layers can range from hundreds of nanometers to hundreds of microns, with the right electrode design and considerations of the sensing environment.

3.2 Electrode Configuration

The specific electrode designs used in Atmel touchscreens are the subject of various patents and patent applications. Further information is available on request.

The device supports various configurations of electrodes as summarized below:

Touchscreens:	2 Touchscreens allowed 3X x 3Y minimum (depends on screen resolution) 32X x 52Y maximum (subject to other configurations)
Keys:	1 Key Array allowed Up to 32 keys (subject to other configurations)

3.3 Scanning Sequence

All channels are scanned in sequence by the device. There is a full parallelism in the scanning sequence to improve overall response time. The channels are scanned by measuring capacitive changes at the intersections formed between the first X line and all the Y lines. Then the intersections between the next X line and all the Y lines are scanned, and so on, until all X and Y combinations have been measured.

The device can be configured in various ways. It is possible to disable some channels so that they are not scanned at all. This can be used to improve overall scanning time.

3.4 Touchscreen Sensitivity

3.4.1 Adjustment

Sensitivity of touchscreens can vary across the extents of the electrode pattern due to natural differences in the parasitics of the interconnections, control chip, and so on. An important factor in the uniformity of sensitivity is the electrode design itself. It is a natural consequence of a touchscreen pattern that the edges form a discontinuity and hence tend to have a different sensitivity. The electrodes at the far edges do not have a neighboring electrode on one side and this affects the electric field distribution in that region.

A sensitivity adjustment is available for the whole touchscreen. This adjustment is a basic algorithmic threshold that defines when a channel is considered to have enough signal change to qualify as being in detect.

3.4.2 Mechanical Stackup

The mechanical stackup refers to the arrangement of material layers that exist above and below a touchscreen. The arrangement of the touchscreen in relation to other parts of the mechanical stackup has an effect on the overall sensitivity of the screen. QMatrix technology has an excellent ability to operate in the presence of ground planes close to the sensor. QMatrix sensitivity is attributed more to the interaction of the electric fields between the transmitting (X) and receiving (Y) electrodes than to the surface area of these electrodes. For this reason, stray capacitance on the X or Y electrodes does not strongly reduce sensitivity

Front panel dielectric material has a direct bearing on sensitivity. Plastic front panels are usually suitable up to about 1.2 mm, and glass up to about 2.5 mm (dependent upon the screen size and layout). The thicker the front panel, the lower the signal-to-noise ratio of the measured capacitive changes and hence the lower the resolution of the touchscreen. In general, glass front panels are near optimal because they conduct electric fields almost twice as easily as plastic panels.

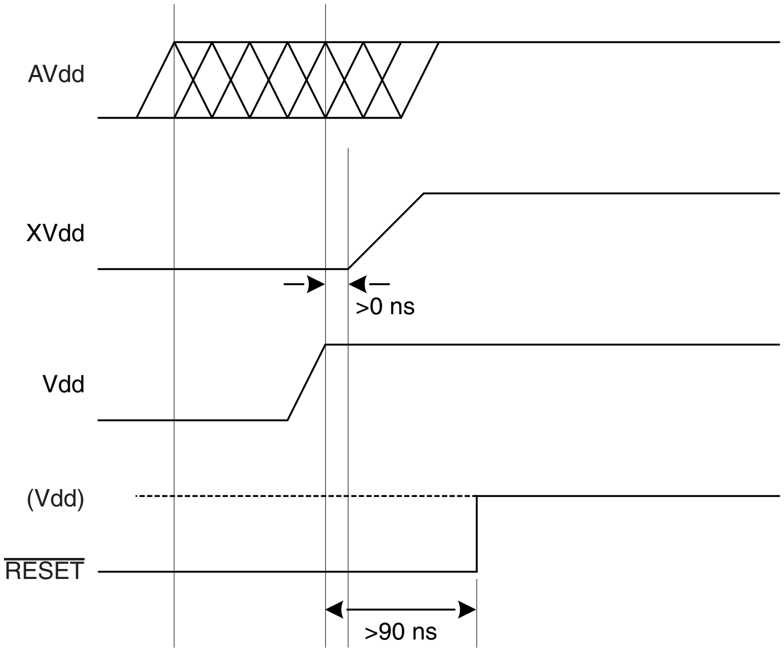
Note: Care should be taken using ultra-thin glass panels as retransmission effects can occur.

4. Detailed Operation

4.1 Power-up/Reset

There is an internal Power-on Reset (POR) in the device.

The device must be held in $\overline{\text{RESET}}$ (active low) while the power supplies (Vdd and AVdd) are powering up. If a slope or slew is applied to the digital or analog supplies (Vdd, AVdd and XVdd) must reach their nominal values before the $\overline{\text{RESET}}$ signal is de-asserted (that is, goes high). This is shown in Figure .**Please note that the XVdd rail has a maximum rate of rise specification** (see Section 9.3.3 on page 58), **that is, a soft-start XVdd supply must be used**..



Note: 1) Vdd and AVdd can be powered up in any order
2) XVdd must not be powered up until after Vdd and must obey the rate-of-rise specification

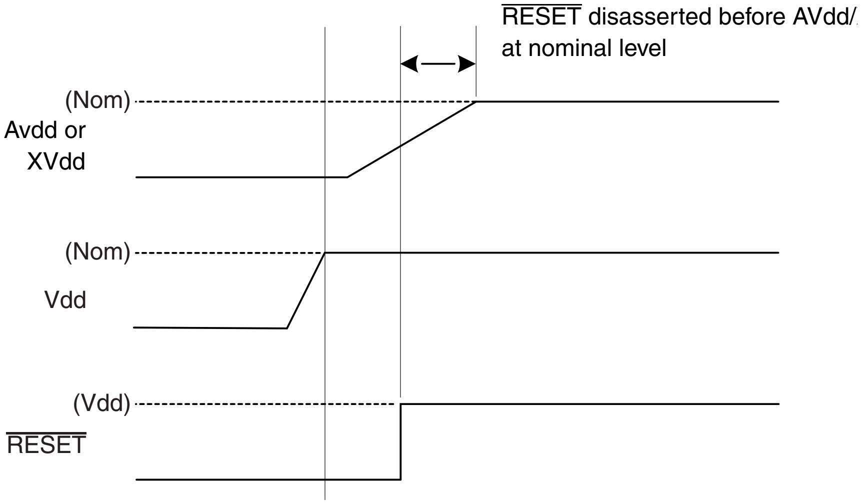
The digital or analog (AVdd) supplies can be applied independently and in any order on the mXT1664S during power-up. Vdd must be applied to the device before XVdd to ensure that the different power domains in the device are initialized correctly. Typically this can be done by connecting the enable pin of the Switched Mode Power Supply (SMPS) supplying XVdd to a 10 kΩ pull-up resistor connected to the Vdd, but the XVdd can be controlled separately by the host, if required.

After power-up, the device takes 65 ms before it is ready to start communications. Vdd must drop to below 1.45 V in order to effect a proper POR. See Section 9 on page 56 for further specifications.

If the RESET line is released before the AVDD and /or XVDD supplies have reached their nominal voltage (see Figure 4-1), then some additional operations need to be carried out by the host. There are two options open to the host controller:

- Start the part in deep sleep mode and then send the command sequence to set the cycle time to wake the part and allow it to run normally. Note that in this case a calibration command is also needed.
- Send a reset command.

Figure 4-1. Power Sequencing on the mXT1664S – Late rise on AVDD



The RESET pin can be used to reset the device whenever necessary. The RESET pin must be asserted low for at least 90 ns to cause a reset. After releasing the RESET pin the device takes ~65 ms before it is ready to start communications. It is recommended to connect the RESET pin to a host controller to allow it to initiate a full hardware reset without requiring a power-down.

Note that the voltage level on the RESET pin of the device must never exceed Vdd (digital supply voltage).

A software reset command can be used to reset the chip (refer to the Command Processor object in the *mXT1664S Protocol Guide*). A software reset takes a maximum of 280 ms. After the chip has finished it asserts the CHG line to signal to the host that a message is available. The reset flag is set in the Message Processor object to indicate to the host that it has just completed a reset cycle. This bit can be used by the host to detect any unexpected brownout events. This allows the host to take any necessary corrective actions, such as reconfiguration.

A checksum check is performed on the configuration settings held in the nonvolatile memory. If the checksum does not match a stored copy of the last checksum, then this indicates that the settings have become corrupted. This is signaled to the host by setting the configuration error bit in the message data for the Command Processor object (refer to the *mXT1664S Protocol Guide* for more information).

Note that the $\overline{\text{CHG}}$ line is momentarily set (approximately 100 ms) as an input after power-up or reset for diagnostic purposes. It is therefore particularly important that the line should be allowed to float high via the $\overline{\text{CHG}}$ line pull-up resistor during this period. It should not be driven by the host.

At power-on, the device performs a self-test routine to check for shorts which might cause damage to the device. Refer to the Self Test T25 section of the *mXT1664S Protocol Guide* for more details about this process.

4.2 Calibration

Calibration is the process by which a sensor chip assesses the background capacitance on each channel. Channels are only calibrated on power-up and when:

- The channel is enabled (that is, activated).

OR

- The channel is already enabled and one of the following applies:
  - The channel is held in detect for longer than the Touch Automatic Calibration setting (refer to the *mXT1664S Protocol Guide* for more information on TCHAUTOCAL setting in the Acquisition Configuration object).
  - The signal delta on a channel is at least the touch threshold (TCHTHR) in the anti-touch direction, while no other touches are present on the channel matrix (refer to the *mXT1664S Protocol Guide* for more information on the TCHTHR field in the Multiple Touch Touchscreen and Key Array objects).
  - The host issues a recalibrate command.
  - Certain configuration settings are changed.

A status message is generated on the start and completion of a calibration.

Note that the device performs a global calibration; that is, all the channels are calibrated together.

4.3 Operational Modes

The device operates in two modes: active (touch detected) and idle (no touches detected). Both modes operate as a series of burst cycles. Each cycle consists of a short burst (during which measurements are taken) followed by an inactive sleep period. The difference between these modes is the length of the cycles. Those in idle mode typically have longer sleep periods. The cycle length is configured using the IDLEACQINT and ACTVACQINT settings in the Power Configuration object. In addition, an Active to Idle timeout (ACTV2IDLETO) setting is provided.

Refer to the *mXT1664S Protocol Guide* for full information on how these modes operate, and how to use the settings provided.

4.4 Touchscreen Layout

The physical matrix can be configured to have one or more touch objects. These are configured using the appropriate touch objects (Multiple Touch Touchscreen and Key Array Key Array). It is not mandatory to have all the allowable touch objects present. The objects are disabled by default so only those that you wish to use need to be enabled. Refer to the *mXT1664S Protocol Guide* for more information on configuring the touch objects.

When designing the physical layout of the touch panel, obey the following rules:

– Each touch object should be a regular rectangular shape in terms of the lines it uses.

– Touch objects can share X and Y lines, as necessary. Note, however, that the first instance (instance 0) of the Multiple Touch Touchscreen T9 object cannot share Y lines if the SlimSensor T56 object is enabled.

– The design of the touch objects does not physically need to be on a strict XY grid pattern.

4.5 Signal Processing

4.5.1 Adjacent Key Suppression Technology

Adjacent Key Suppression (AKS) technology is a patented method used to detect which touch object is touched when objects are located close together. A touch in a group of AKS objects is only indicated on the object in that group that is touched first. This is assumed to be the intended object. Once an object in an AKS group is in detect, there can be no further detections within that group until the object is released. Objects can be in more than one AKS group.

Note that AKS technology works best when it operates in conjunction with a detect integration setting of several acquisition cycles.

The device has two levels of AKS. The first level works between the touch objects (Multiple Touch Touchscreen T9 and Key Array T15). The touch objects are assigned to AKS groups. If a touch occurs within one of the touch objects in a group, then touches within other objects inside that group are suppressed. For example, if a Touchscreen and a Key Array are placed in the same AKS group, then a touch in the Touchscreen will suppress touches in the Key Array, and vice versa.

The second level of AKS is internal AKS within an individual Key Array object (note that internal AKS is not present on other types of touch objects, only a Key Array). If internal AKS is enabled, then when one key is touched, touches on all the other keys within the Key Array are suppressed.

AKS is configured using the touch objects (Multiple Touch Touchscreen T9 or Key Array T15). Refer to the *mXT1664S Protocol Guide* for more information.

Note: If a touch is in detect and then AKS is enabled, that touch will not be forced out of detect. It will not go out of detect until the touch is released. AKS will then operate normally. This applies to both levels of AKS.

4.5.2 Detection Integrator

The device features a touch detection integration mechanism. This acts to confirm a detection in a robust fashion. A counter is incremented each time a touch has exceeded its threshold and has remained above the threshold for the current acquisition. When this counter reaches a preset limit the sensor is finally declared to be touched. If, on any acquisition, the signal is not seen to exceed the threshold level, the counter is cleared and the process has to start from the beginning.

The detection integrator is configured using the appropriate touch objects (Multiple Touch Touchscreen T9, Key Array T15). Refer to the *mXT1664S Protocol Guide* for more information.

4.5.3 Digital Filtering and Noise Suppression

The mXT1664S supports the on-chip filtering of the acquisition data received from the sensor. Specifically, the maXCharger T62 object provides an algorithm to suppress the effects of noise (for example, from a noisy charger plugged into the user's product). This algorithm can automatically adjust some of the acquisition parameters on-the-fly to filter the analog-to-digital conversions (ADCs) received from the sensor. The algorithm can make use of a Grass Cutter (which rejects any samples outside a predetermined limit).

Noise suppression is triggered when a noise source is detected (typically when a charger is turned on). A hardware trigger can be implemented using the CHRG_IN pin. Alternatively, the host's driver code can indicate when a noise source is present.

An alternative burst mode on the X lines, known as Dual X Drive, is provided. This improves the signal-to-noise ratio (SNR) on a closely spaced X sensor matrix (when finger touches are likely to cover more than one X line).

Refer to the *mXT1664S Protocol Guide* for more information on the maXCharger T62 object.

4.5.4 Shieldless Support

The mXT1664S can support shieldless sensor design even with a noisy LCD. The SlimSensor T56 object provides a number of algorithms to suppress the effect of noise emitted by the display.

The T56 display noise suppression operates on a completely different mechanism to the maXCharger T62 object. This allows the device to overcome display noise simultaneously with charger noise.

The device can make use of the following mechanisms to overcome display noise:

- Optimal Integration is not filtering per say, instead it is a feature that enables the user to use a shorter integration window. The integration window optimizes the amount of charge collected against the amount of noise collected, to ensure an optimal SNR. This feature also benefits the system in the presence of an external noise source such as charger.
- The main noise suppression method for display noise is the noise canceller. The noise canceller measures the noise generated by the display and subtracts it from the noise cancellation feature measurement. When the noise canceller is enabled the maXCharger T62 object cannot use the Grass Cut filter.

Refer to the *mXT1664S Protocol Guide* Protocol Guide for more information on the SlimSensor T56 object.

4.5.5 Stylus Support

The mXT1664S allows for the particular characteristics of stylus touches, whilst still allowing conventional finger touches to be detected. Stylus touches are configured by the Stylus T47 object. There is one instance of the Stylus T47 object for each Multiple Touch Touchscreen T9 object present on the device.

For example, stylus support ensures that the small touch area of a stylus registers as a touch, as this would otherwise by considered too small for the touchscreen. Additionally, there are controls to distinguish a stylus touch from an unwanted approaching finger (such as on the hand holding the stylus).

The touch sensitivity and threshold controls for stylus touches are configured separately from those for conventional finger touches so that both types of touches can be accommodated.

4.5.6 Grip Suppression

The mXT1664S has a grip suppression mechanism to suppress false detections when the user grips a handheld device.

Grip suppression works by specifying a boundary around a touchscreen, within which touches can be suppressed whilst still allowing touches in the center of the touchscreen. This ensures that a "rolling" hand touch (such as when a user grips a mobile device) is suppressed. A "real" (finger) touch towards the center of the screen is allowed.

Grip suppression is configured using the Grip Suppression T40 object. There is one instance of the Grip Suppression T40 object for each Multiple Touch Touchscreen T9 object present on the device. Refer to the *mXT1664S Protocol Guide* for more information.

4.5.7 Unintentional Touch Suppression

The Touch Suppression T42 object provides a mechanism to suppress false detections from unintentional touches from a large body area, such as from a face, ear or palm. The Touch Suppression T42 object also provides Maximum Touch Suppression to suppress all touches if more than a specified number of touches has been detected. There is one instance of the Touch Suppression T42 object for each Multiple Touch Touchscreen T9 object present on the device. Refer to the *mXT1664S Protocol Guide* for more information.

4.6 Circuit Components

4.6.1 XVdd Power Supply

The X line driver power supply XVdd can be used in two different modes:

- XVdd connected to AVdd. This mode limits the range of XVdd to 2.7 V to 3.3 V.
- XVdd connected to an external supply. In this configuration the external supply should be in the range 2.7 V to 6.0 V. The higher voltages improve the SNR of the system.
- If XVdd < 4.75 V, please note restriction on minimum Cx in Section 9.2 on page 57.

4.6.2 Bypass Capacitors

Each power supply (Vdd, XVdd and AVdd) requires a 1 µF bypass capacitor. If the internal 1.8V VDDCORE regulator is used, the Vdd 1 µF should be replaced with a 10 µF capacitor and a 2.2 µF capacitor should be added on the VDDCORE pin. In addition, there should be a 100 nF bypass capacitor on each power trace. The capacitors should be ceramic X7R or X5R. See the schematics in Section 2.3 on page 10 for more details.

The PCB traces connecting the bypass capacitors to the pins of the device must not exceed 5 mm in length. This limits any stray inductance that would reduce filtering effectiveness. See also Section 9.14 on page 67.

4.6.3 Supply Quality

While the device has good Power Supply Rejection Ratio properties, poorly regulated and/or noisy power can significantly reduce performance. See Section 9.14 on page 67.

Always operate the device with a well-regulated and clean AVdd (and XVdd, if used) supply. It supplies the sensitive analog stages in the device. See Figure 9-1 on page 73 for an example XVdd supply.

4.6.4 Supply Sequencing

Vdd and AVdd can be powered independently of each other without damage to the device. Vdd must be applied to the device before XVdd to ensure proper initialization of the device. All voltage ranges should be used with in the limits specified in Section 9.2 on page 57.

Make sure that any lines connected to the device are below or equal to Vdd during power-up. For example, if $\overline{\text{RESET}}$ is supplied from a different power domain to the mXT1664S VDD pin, make sure that it is held low when Vdd is off. If this is not done, the $\overline{\text{RESET}}$ signal could parasitically couple power via the mXT1664S $\overline{\text{RESET}}$ pin into the Vdd supply.

4.6.5 Oscillator

A 16 MHz crystal oscillator must be connected to the device when the device is operating in USB mode. A crystal oscillator with a minimum accuracy of 100 ppm must be used.

An external oscillator is not needed in I^{2}C-compatible mode.

4.6.6 Decoupling Requirements

Certain pins have specific decoupling requirements:

- Pin L7 (VDD_INPUT) should be connected to VDD.
- Pin N8 (VDDCORE) is a decoupling connection for an internal LDO (Low Drop-Out regulator) circuit and should not be powered by an external supply unless in 1.8 V I^{2}C-compatible mode.

See also the schematics in Section 2.3 on page 10.

4.6.7 PCB Cleanliness

Modern no-clean-flux is generally compatible with capacitive sensing circuits.

CAUTION: If a PCB is reworked to correct soldering faults relating to any of the device devices, or to any associated traces or components, be sure that you fully understand the nature of the flux used during the rework process. Leakage currents from hygroscopic ionic residues can stop capacitive sensors from functioning. If you have any doubts, a thorough cleaning after rework may be the only safe option.

4.7 PCB Layout

See Appendix A on page 71 for general advice on PCB layout.

4.8 Debugging

The device provides a mechanism for obtaining raw data for development and testing purposes by reading data from the Diagnostic Debug T37 object. Refer to the *mXT1664S Protocol Guide* for more information on this object.

A second mechanism is provided that allows the host to read the real-time raw data using the low-level debug port. This can be accessed via the SPI interface or the USB interface. Note that if both the I^{2}C-compatible and USB interfaces are used for normal communications, the debug data is output on the USB interface. Refer to QTAN0050, *Using the maXTouch Debug Port*, for more information on the debug port.

There is also a Self Test T25 object that runs self-test routines in the mXT1664S to find hardware faults on the sense lines and the electrodes. This object also performs an initial pin fault test on power-up to ensure that there is no X-to-Y short before the high-voltage supply is enabled inside the chip. A high-voltage short into the analog circuitry would break the device.

Refer to the *mXT1664S Protocol Guide* and QTAN0059, *Using the maXTouch Self Test Feature*, for more information on the Self Test T25 object.

4.9 Communications

Communication with the host is achieved using either the I^{2}C-compatible interface (see Section 5 on page 24), the HID-I^{2}C-compatible interface (see Section 7 on page 43), or the USB interface (see Section 6 on page 31). Any interface can be used, depending on the needs of the user's project, but only one interface should be used in any one design. The selection of the I^{2}C-compatible or the HID-I^{2}C-compatible interface is determined by the I2CMODE pin.

The interface is selected using the COMMSEL pin. Connect COMMSEL to Vdd to select the USB interface, or to GND to select one of the two I^{2}C-compatible interfaces.

Note that you only need to connect those pins that are actually required for use with the chosen communications interface. This ensures optimal power consumption and correct functioning. See Section 2.2 on page 5 for details on what should be done with the unconnected pins.

4.10 Configuring the Device

The device has an object-based protocol that organizes the features of the device into objects that can be controlled individually. This is configured using the Object Protocol common to many of Atmel's touch sensor devices. For more information on the Object Protocol and its implementation on the device, refer to the *mXT1664S Protocol Guide*.

5. I²C-compatible Communications

5.1 Communications Protocol

The device can use an I²C-compatible interface for communication. See Appendix E on page 84 for details of the I²C-compatible protocol.

The I²C-compatible interface is used in conjunction with the $\overline{\text{CHG}}$ line. The $\overline{\text{CHG}}$ line going active signifies that a new data packet is available. This provides an interrupt-style interface and allows the device to present data packets when internal changes have occurred.

5.2 I²C-compatible Addresses

5.2.1 I²C-compatible Addresses

The mXT1664S supports two I²C-compatible device addresses that are selected using the ADDSEL line at start-up. The two internal I²C-compatible device addresses are 0x4A (ADDSEL low) and 0x4B (ADDSEL floating or high). These are shifted left to form the SLA+W or SLA+R address when transmitted over the I²C-compatible interface, as shown in Figure 5-1.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Address: 0x4A or 0x4B							Read/write

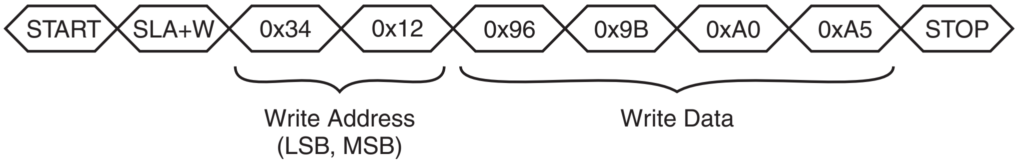
5.3 Writing To the Device

A WRITE cycle to the device consists of a START condition followed by the I²C-compatible address of the device (SLA+W). The next two bytes are the address of the location into which the writing starts. The first byte is the Least Significant Byte (LSByte) of the address, and the second byte is the Most Significant Byte (MSByte). This address is then stored as the address pointer.

Subsequent bytes in a multi-byte transfer form the actual data. These are written to the location of the address pointer, location of the address pointer +1, location of the address pointer + 2, and so on. The address pointer returns to its starting value when the WRITE cycle's STOP condition is detected.

Figure 5-1 shows an example of writing four bytes of data to contiguous addresses starting at 0x1234.

Figure 5-1. Example of a Four-byte Write Starting at Address 0x1234

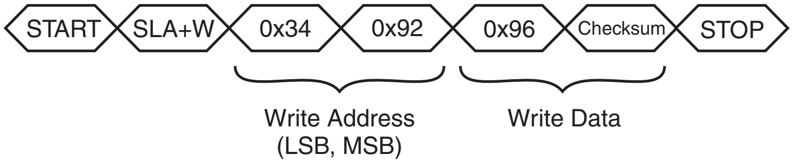


5.4 I^{2}C-compatible Writes in Checksum Mode

In I^{2}C-compatible checksum mode an 8-bit CRC is added to all I^{2}C-compatible writes. The CRC is sent at the end of the data write as the last byte before the STOP condition. All the bytes sent are included in the CRC, including the two address bytes. Any command or data sent to the device is processed even if the CRC fails.

To indicate that a checksum is to be sent in the write, the most significant bit of the MSByte of the address is set to 1. For example, the I^{2}C-compatible command shown in Figure 5-2 writes a value of 150 (0x96) to address 0x1234 with a checksum. The address is changed to 0x9234 to indicate checksum mode.

Figure 5-2. Example of a Write To Address 0x1234 With a Checksum



5.5 Reading From the Device

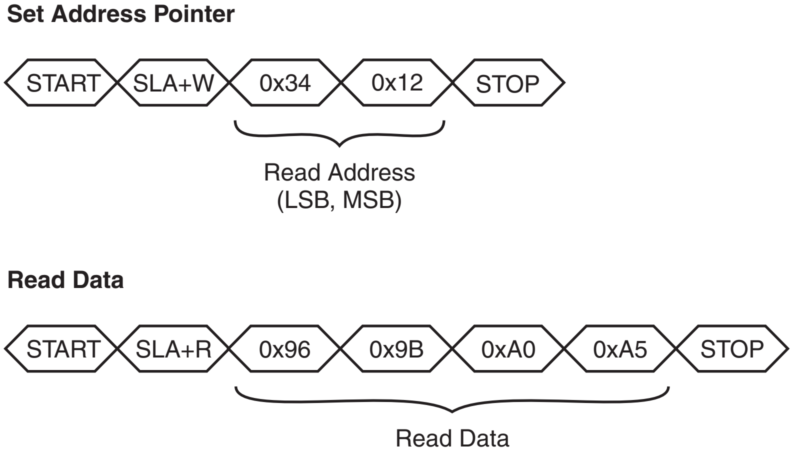
Two I^{2}C-compatible bus activities must take place to read from the device. The first activity is an I^{2}C-compatible write to set the address pointer (LSByte then MSByte). The second activity is the actual I^{2}C-compatible read to receive the data. The address pointer returns to its starting value when the read cycle's NACK is detected.

It is not necessary to set the address pointer before every read. The address pointer is updated automatically after every read operation. The address pointer will be correct if the reads occur in order. In particular, when reading multiple messages from the Message Processor T5 object, the address pointer is automatically reset to allow continuous reads (see Section 5.6).

The WRITE and READ cycles consist of a START condition followed by the I^{2}C-compatible address of the device (SLA+W or SLA+R respectively).

Figure 6-10 shows the I^{2}C-compatible commands to read four bytes starting at address 0x1234.

Figure 5-3. Example of a Four-byte Read Starting at Address 0x1234



5.6 Reading Status Messages with DMA

The device facilitates the easy reading of multiple messages using a single continuous read operation. This allows the host hardware to use a direct memory access (DMA) controller for the fast reading of messages, as follows:

1. The host uses a write operation to set the address pointer to the start of the Message Count T44 object, if necessary. [1] If a checksum is required on each message, the most significant bit of the MSByte of the read address must be set to 1.
2. The host starts the read operation of the message by sending a START condition.
3. The host reads the Message Count T44 object (one byte) to retrieve a count of the pending messages (refer to the *mXT1664S Protocol Guide* for details).
4. The host calculates the number of bytes to read by multiplying the message count by the size of the Message Processor T5 object. [2]

   Note that the size of the Message Processor T5 object as recorded in the Object Table includes a checksum byte. If a checksum has not been requested, one byte should be deducted from the size of the object. That is: number of bytes = count x (size-1).
5. The host reads the calculated number of message bytes. It is important that the host does *not* send a STOP condition during the message reads, as this will terminate the continuous read operation and reset the address pointer. No START and STOP conditions must be sent between the messages.
6. The host sends a STOP condition at the end of the read operation after the last message has been read. The NACK condition immediately before the STOP condition resets the address pointer to the start of Message Count T44 object.

Figure 5-4 on page 27 shows an example of using a continuous read operation to read three messages from the device without a checksum. Figure 5-5 on page 28 shows the same example with a checksum.

[1] The STOP condition at the end of the read resets the address pointer to its initial location, so it may already be pointing at the Message Count T44 object following a previous message read.

[2] The host should have already read the size of the Message Processor T5 object in its initialization code.

Figure 5-4. Continuous Message Read Example – No Checksum

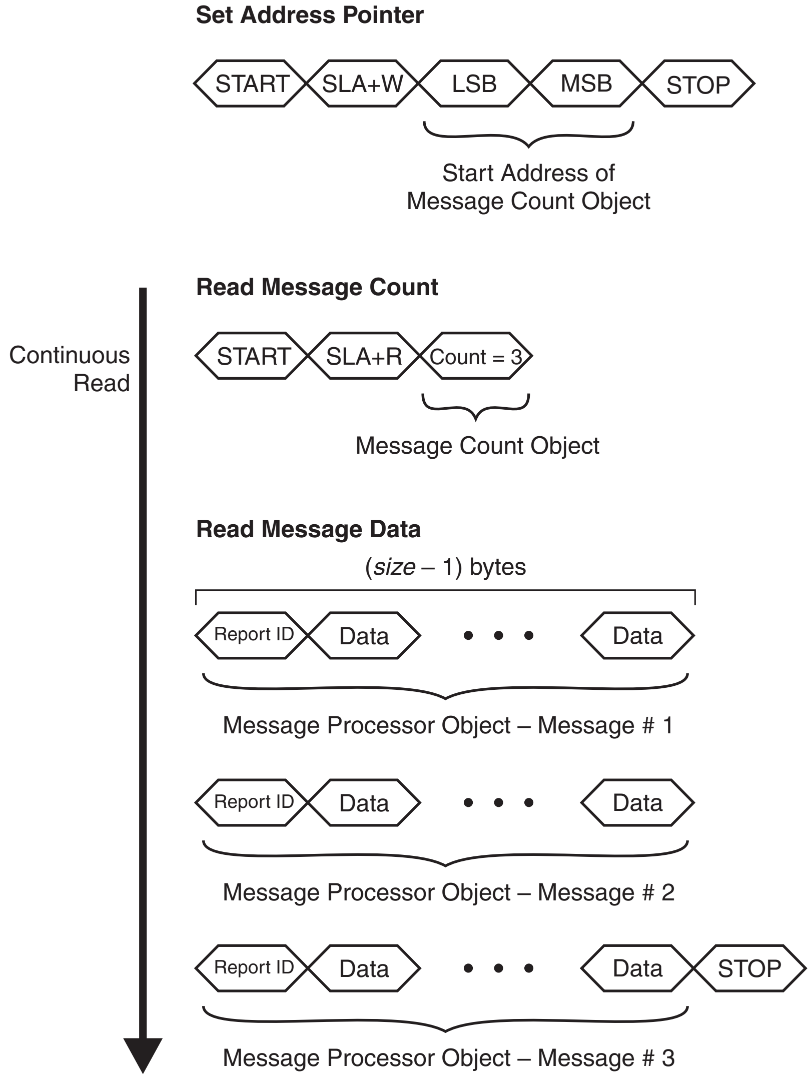
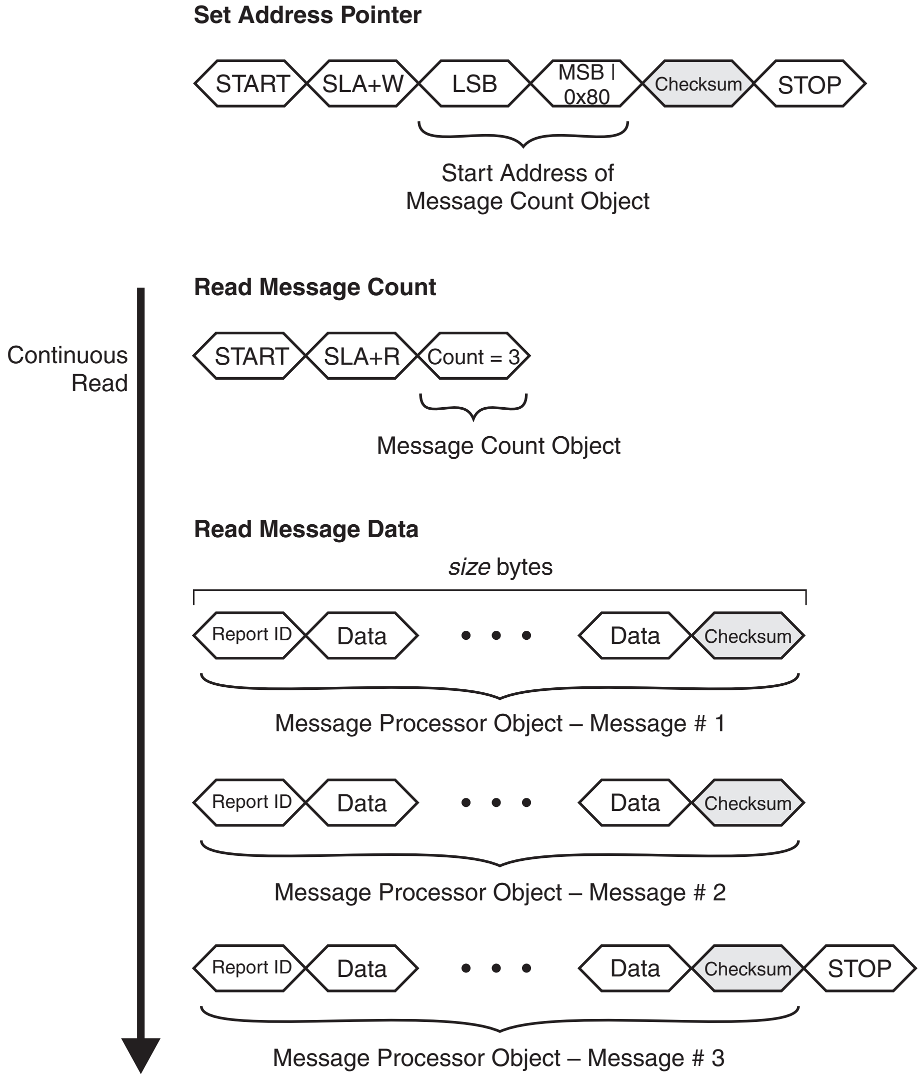


Figure 5-5. Continuous Message Read Example – I²C-compatible Checksum Mode



There are no checksums added on any other I²C-compatible reads. An 8-bit CRC can be added, however, to all I²C-compatible writes, as described in Section 5.4 on page 25.

An alternative method of reading messages using the $\overline{\text{CHG}}$ line is given in Section 5.7.

5.7 $\overline{\text{CHG}}$ Line

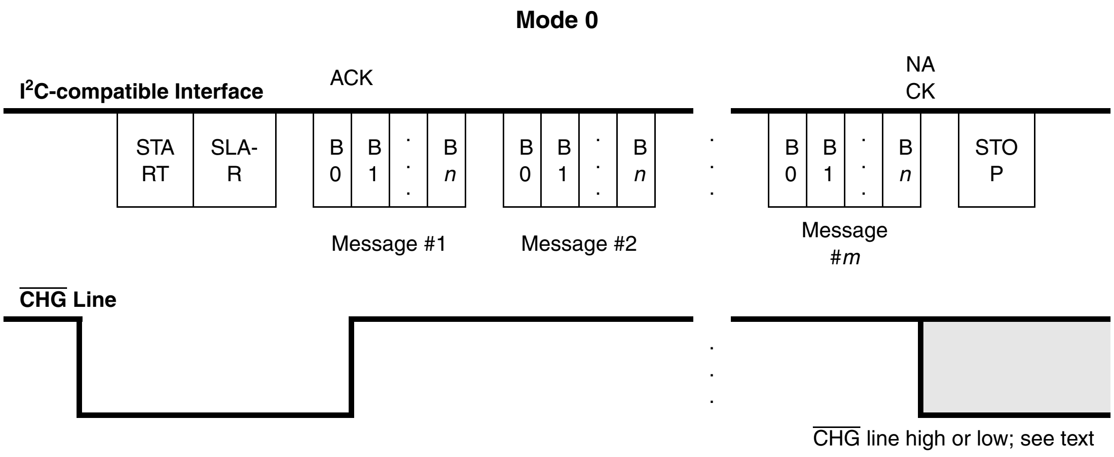
The $\overline{\text{CHG}}$ line is an active-low, open-drain output that is used to alert the host that a new message is available in the Message Processor T5 object. This provides the host with an interrupt-style interface with the potential for fast response times. It reduces the need for wasteful I²C-compatible communications.

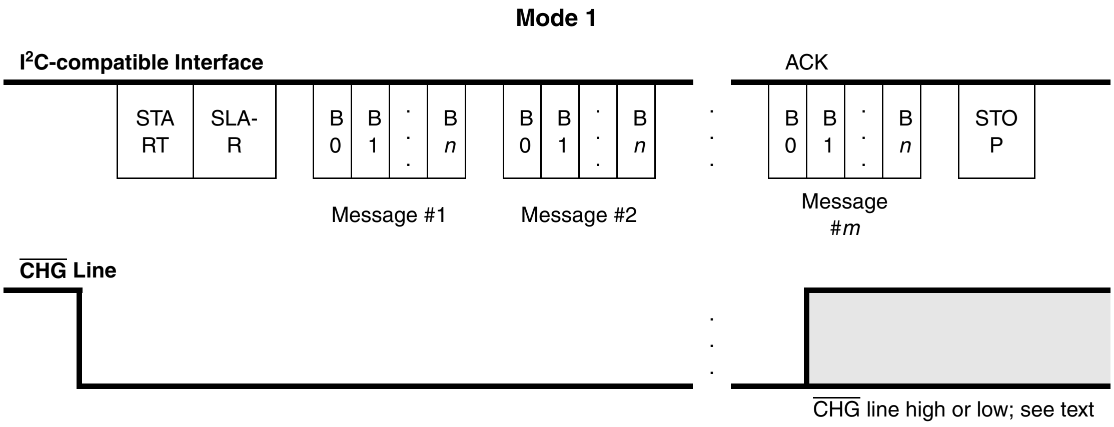
The $\overline{\text{CHG}}$ line remains low as long as there are messages to be read. The host should be configured so that the $\overline{\text{CHG}}$ line is connected to an interrupt line that is level-triggered. The host should not use an edge-triggered interrupt as this means adding extra software precautions.

The $\overline{\text{CHG}}$ line should be allowed to float during normal usage. This is particularly important after power-up or reset (see Section 4.1 on page 16).

A pull-up resistor is required, typically 10 kΩ to Vdd.

The $\overline{\text{CHG}}$ line operates in two modes, as defined by the Communications Configuration T18 object (refer to the *mXT1664S Protocol Guide*).


Mode 0
I^{2}C-compatible Interface
ACK
NA
CK
STA
RT
SLA-
R
B
0
B
1
B
n
STO
P
Message #1
Message #2
Message
#*m*
$\overline{\text{CHG}}$ Line
$\overline{\text{CHG}}$ line high or low; see text


Mode 1
I^{2}C-compatible Interface
ACK
STA
RT
SLA-
R
B
0
B
1
B
n
STO
P
Message #1
Message #2
Message
#*m*
$\overline{\text{CHG}}$ Line
$\overline{\text{CHG}}$ line high or low; see text

In Mode 0:

1. The $\overline{\text{CHG}}$ line goes low to indicate that a message is present.
2. The $\overline{\text{CHG}}$ line goes high when the first byte of the first message (that is, its report ID) has been sent and acknowledged (ACK sent) and the next byte has been prepared in the buffer.
3. The STOP condition at the end of an I^{2}C-compatible transfer causes the $\overline{\text{CHG}}$ line to stay high if there are no more messages. Otherwise the $\overline{\text{CHG}}$ line goes low to indicate a further message.

Mode 0 allows the host to continually read messages. Messaging reading ends when a report ID of 255 (“invalid message”) is received. Alternatively the host ends the transfer by sending a NACK after receiving the last byte of a message, followed by a STOP condition. If and when there is another message present, the $\overline{\text{CHG}}$ line goes low, as in step 1. In this mode the state of the $\overline{\text{CHG}}$ line does not need to be checked during the I^{2}C-compatible read.

In Mode 1:

1. The $\overline{\text{CHG}}$ line goes low to indicate that a message is present.
2. The $\overline{\text{CHG}}$ line remains low while there are further messages to be sent after the current message.
3. The $\overline{\text{CHG}}$ line goes high again only once the first byte of the last message (that is, its report ID) has been sent and acknowledged (ACK sent) and the next byte has been prepared in the output buffer.

Mode 1 allows the host to continually read the messages until the $\overline{\text{CHG}}$ line goes high, and the state of the $\overline{\text{CHG}}$ line determines whether or not the host should continue receiving messages from the device.

Note: The state of the $\overline{\text{CHG}}$ line should be checked only between messages and not between the bytes of a message. The precise point at which the $\overline{\text{CHG}}$ line changes state cannot be predicted and so the state of the $\overline{\text{CHG}}$ line cannot be guaranteed between bytes.

The Communications Configuration T18 object can be used to configure the behavior of the $\overline{\text{CHG}}$ line. In addition to the $\overline{\text{CHG}}$ line operation modes described above, this object allows the use of edge-based interrupts, as well as direct control over the state of the $\overline{\text{CHG}}$ line. Refer to the *mXT1664S Protocol Guide* for more information.

5.8 SDA, SCL

The I^{2}C-compatible bus transmits data and clock with SDA and SCL, respectively. These are open-drain. The device can only drive these lines low or leave them open. The termination resistors (Rp) pull the line up to Vdd if no I^{2}C-compatible device is pulling it down.

The termination resistors should be chosen so that the rise times on SDA and SCL meet the I^{2}C-compatible specifications for the interface speed being used, bearing in mind other loads on the bus, (see Section 9.10 on page 66).

5.9 Clock Stretching

The device supports clock stretching in accordance with the I^{2}C specification. It may also instigate a clock stretch if a communications event happens during a period when the device is busy internally. The maximum clock stretch is approximately 10 – 15 ms.

The device has an internal bus monitor that can reset the internal I^{2}C-compatible hardware if SDA or SCL is stuck low for more than 200 ms. This means that if a prolonged clock stretch of more than 200 ms is seen by the device, then any ongoing transfers with the device may be corrupted. The bus monitor is enabled or disabled using the Communications Configuration T18 object. Refer to the *mXT1664S Protocol Guide* for more information.

6. USB Communications

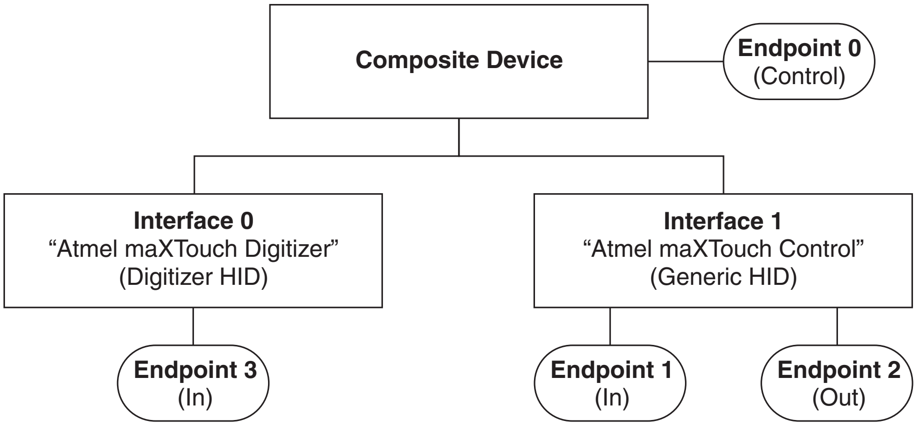
6.1 Communications Protocol

The device is a composite USB device with two Human Interface Device (HID) interfaces:

- **Interface 0** – This interface supports two Digitizer usage, one for the touchscreen and one for the Pen (Active maXStylus). This interface is supported by Microsoft® Windows® 7 without the need for additional software. The HID identifier string is "Atmel maXTouch Digitizer".
- **Interface 1** – This interface provides a Generic HID that allows the host to communicate with the device using the Object Protocol. The HID identifier string is "Atmel maXTouch Control".

The topography of the USB device is shown in Figure 6-1.

Figure 6-1. USB Topography



Communication takes place using Full-speed USB at 12 Mbps.

For more information on the USB HID specifications visit www.usb.org.

6.2 Endpoint Addresses

The endpoint addresses are listed in Table 6-1.

Table 6-1. Endpoint Addresses

Endpoint	Direction	Address
Endpoint 0	Bidirectional (control)	–
Endpoint 1	In	0x81
Endpoint 2	Out	0x02
Endpoint 3	In	0x83

6.3 Composite Device

The composite device is a USB 2.0-compliant USB composite device running at full speed (12 Mbps). It has the following specification:

Vendor ID:	0x03EB (Atmel)
Product ID:	0x212C (mXT1664S)

Version:	16-bit Version & Build Identifier in the form 0xVVBB, where:
	VV = Version Major (Upper 4 bits) / Minor (Lower 4 bits)
	BB = Build number

The composite device has one bidirectional endpoint: the Control Endpoint (Endpoint 0). It is used by the USB Host to interrogate the USB device for details on its configurations, interfaces and report structures. It is also used to apply general device settings relating to USB Implementation.

6.4 Interface 0 (Digitizer HID)

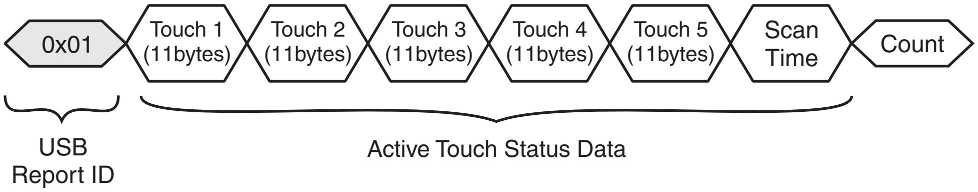
6.4.1 Normal Touch Report

Interface 0 is a Digitizer-class HID, compliant with HID specification 1.11 with amendments. [1]

This interface consists of a single interrupt-In endpoint (Endpoint 3).

The format of an input report is shown in Figure 6-2. Each input report start with a USB Report ID [2] (value 0x01). This is followed by 5 sets of data (11 bytes each) that describe the status of up to 5 active touches. The input report is terminated by a single byte that contains the number of active touches.

Figure 6-2. Input Report Packet



Any unused touch data bytes are set to zero (for example, the data for one active touch would be followed by 44 zeroed bytes). If there are more than five active touches to be reported, further input reports are sent with the remaining touch data. In this case, the count (for all touches) is sent in the first count byte and the count byte in the following reports is zero. An example of the input report packets for 7 active touches is shown in Figure 6-3 on page 33.

[1] This is an implementation of Microsoft's USB HID specification for Multitouch digitizers.

[2] The term USB Report ID should not be confused with the term Report Id as used in the Object Protocol; the two are entirely different concepts.

Figure 6-3. Example Input Report Packets for 7 Active Touches

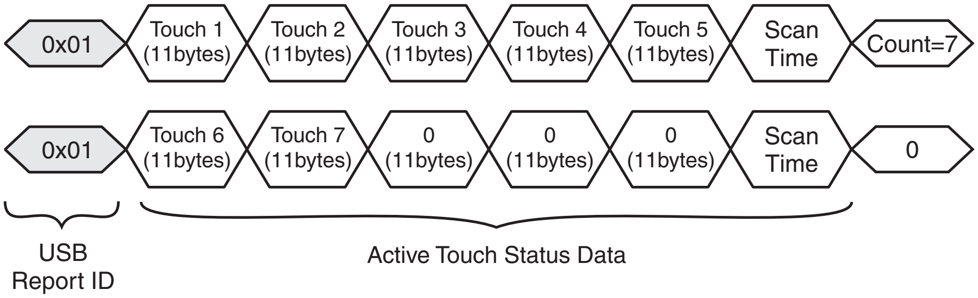


Table 6-2 gives the detailed format of an input report packet.

Table 6-2. Input Report Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	USB Report ID							
1	Touch ID (first touch)					Reserved		Status
2	X Position LSByte (first touch)							
3	0	0	0	0	X Position MSBits (first touch)			
4	X Position LSByte (first touch)							
5	0	0	0	0	X Position MSBits (first touch)			
6	Y Position LSByte (first touch)							
7	0	0	0	0	Y Position MSBits (first touch)			
8	Y Position LSByte (first touch)							
9	0	0	0	0	Y Position MSBits (first touch)			
10	Touch Width							
11	Touch Height							
12 – 22	Touch data for second touch – same format as bytes 1 – 11							
23 – 33	Touch data for third touch – same format as bytes 1 – 11							
34 – 44	Touch data for fourth touch – same format as bytes 1 – 11							
45 – 55	Touch data for fifth touch – same format as bytes 1 – 11							
56 – 57	Scan Time							
58	Contact Count							

In Table 6-2:

- **Byte 1:**

  Status: 1 = In detect, 0 = Not in detect.

  Touch ID: Identifies the touch for which this is a status report (starting from 1).

- **Bytes 2 to 9:**

  X and Y positions: These are scaled to 12-bit resolution. This means that the upper four bits of the MSByte will always be zero.

- **Byte 10:**

  Touch Width: Reports the width of the detected touch.

- **Byte 11:**

  Touch Height: Reports the height of the detected touch.

- **Byte 56 to 57:**

  Scan Time: Timestamp associated with the current report packet with a 10 kHz resolution.

- **Byte 58:**

  Contact Count: Number of active touches.

6.4.2 Active maXStylus Report

The format of an active maXStylus report is shown in Figure 6-4. Each input report start with a USB Report ID [1] (value 0x03).

Figure 6-4. Active maXStylus Report Packet

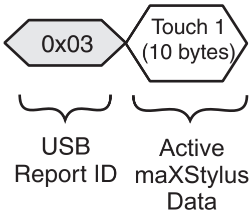


Table 6-3 gives the detailed format of an active stylus report packet.

Table 6-3. Active maXStylus Report Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	USB Report ID							
1	Reserved			In Range	Reserved	Eraser	Barrel	Tip
2	X Position							
3								
4	X Center Position							
5								
6	Y Position							
7								
8	Y Center Position							
9								
10	Tip Pressure							

- **Byte 1:**

Tip: 1 = Contact of the stylus with the touch screen surface, 0 = No contact

Barrel: 1 = Barrel button on, 0 = Barrel button off

Eraser: 1 = Eraser function on, 0 = Eraser function off

[1] The term USB Report ID should not be confused with the term Report Id as used in the Object Protocol; the two are entirely different concepts.

In Range: 1 = Stylus approaching the touchscreen detected, 0 = No stylus approaching the touchscreen detected.

- **Byte 2 to 3:**

  X position
- **Byte 4 to 5:**

  X center position
- **Byte 6 to7:**

  Y position
- **Byte 8 to 9:**

  Y center position
- **Byte 10:**

  Tip Pressure: Force exerted against the touch screen surface by the stylus.

There are two update conditions:

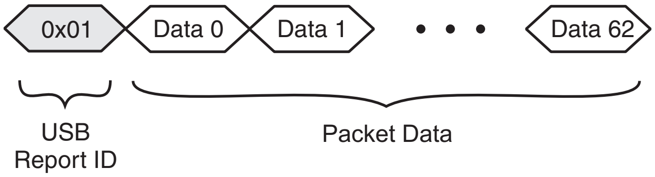
- **Change:** A change in status of any contact (touch) triggers a touch update message to be sent to the host.
- **Idle:** The idle delay of the Digitizer Interface may be controlled via the Control Endpoint as per the HID 1.11 specification (Set Idle command). By default this is set to a delay of 2 (8 ms).

6.5 Interface 1 (Generic HID)

Interface 1 is a Generic Human Interface Device, compliant with HID specification 1.11 with amendments. [1]

It consists of two endpoints: an interrupt-In endpoint (Endpoint 1) and an interrupt-out endpoint (Endpoint 2). The data packet in each case contains a 1-byte USB Report ID followed by 63 bytes of data, totalling 64 bytes (see Figure 6-5).

Figure 6-5. Data Packet for Interface 1



Commands are sent by the application software over the Interrupt-out endpoint, Endpoint 2. The command is sent as the first data byte of the packet data (data byte 0), followed by conditions and/or data.

The supported commands are as follows:

- Read/write Memory Map
- Send Auto-return messages
- Start debug monitoring
- End debug monitoring

[1] This is an implementation of Microsoft's USB HID specification for Multitouch digitizers.

Responses from the device are sent via the interrupt-In endpoint, Endpoint 1.

6.5.1 Read/Write Memory Map

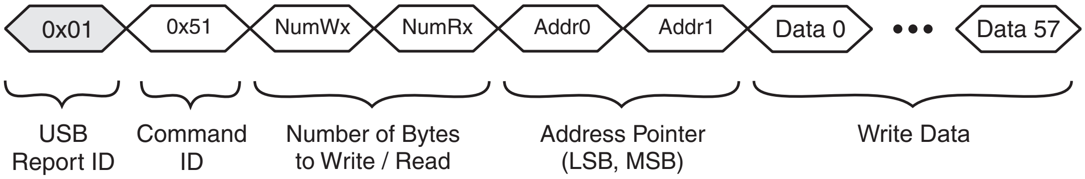
6.5.1.1 Introduction

This command is used to carry out a write/read operation on the memory map of the device.

The USB Report ID is 0x01.

The command packet has the generic format given in Figure 6-6. The following sections give examples on using the command to write to the memory map and to read from the memory map.

Figure 6-6. Generic Command Packet Form

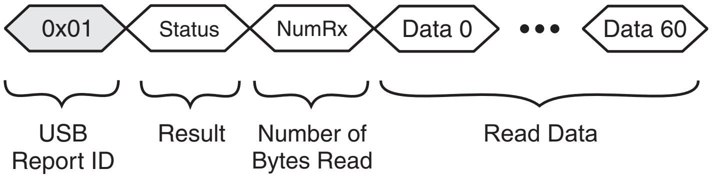


In Figure 6-6:

- **NumWx** is the number of data bytes to write to the memory map (may be zero). If the address pointer is being sent, this must include the size of the address pointer.
- **NumRx** is the number of data bytes to read from the memory map (may be zero).
- **Addr 0** and **Addr 1** form the address pointer to the memory map (where necessary; may be zero if not needed).
- **Data 0** to **Data 57** are the bytes of data to be written (in the case of a write). Note that data locations beyond the number specified by NumWx will be ignored.

The response packet has the generic format given in Figure 6-7.

Figure 6-7. Response Packet Format



In Figure 6-7:

- **Status** indicates the result of the command:

  0x00 = read and write completed; read data returned

  0x04 = write completed; no read data requested
- **NumRx** is the number of bytes following that have been read from the memory map (in the case of a read). This will be the same value as NumRx in the command packet.
- **Data 0** to **Data 60** are the data bytes read from the memory map.

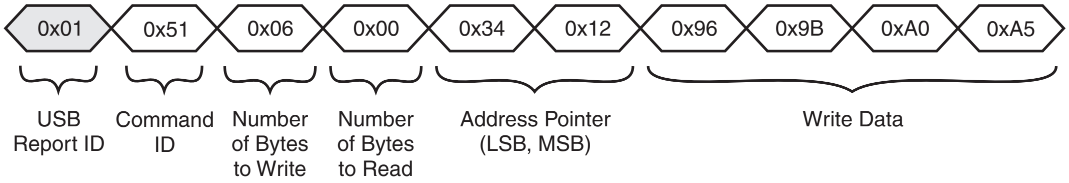
6.5.1.2 *Writing To the Device*

A write operation cycle to the device consists of sending a packet that contains six header bytes. These specify the USB report ID, the Command ID, the number of bytes to read, the number of bytes to write, and the 16-bit address pointer.

Subsequent bytes in a multibyte transfer form the actual data. These are written to the location of the address pointer, location of the address pointer +1, location of the address pointer + 2, and so on.

Figure 6-8 shows an example command packet to write four bytes of data to contiguous addresses starting at 0x1234.

Figure 6-8. Example of a Four-byte Write Starting at Address 0x1234

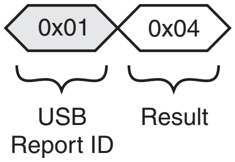


In Figure 6-8:

- The number of bytes to read is set to zero as this is a write-only operation.
- The number of bytes to write is six: that is, four data bytes plus the two address pointer bytes.

Figure 6-9 shows the response to this command. Note that the result status returned is 0x04 (that is, the write operation was completed but no read data was requested).

Figure 6-9. Response to Example Four-byte Write

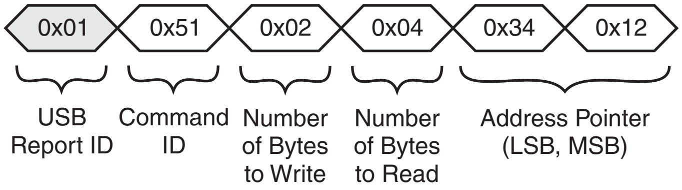


6.5.1.3 *Reading From the Device*

A read operation consists of sending a packet that contains the six header bytes only and no write data.

Figure 6-10 shows an example command packet to read four bytes starting at address 0x1234. Note that the address pointer is included in the number of bytes to write, so the number of bytes to write is set to 2 as there are no other data bytes to be written.

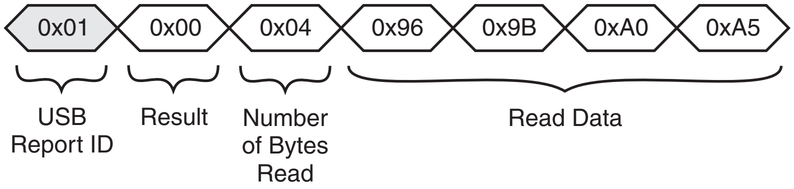
Figure 6-10. Example of a Four-byte Read Starting at Address 0x1234



It is not necessary to set the address pointer before every read. The address pointer is updated automatically after every read operation, so the address pointer will be correct if the reads occur in order.

Figure 6-11 shows the response to this command. The result status returned is 0x00 (that is the write operation was completed and the data was returned). The number of bytes returned will be the same as the number requested (4 in this case).

Figure 6-11. Response to Example Four-byte Read

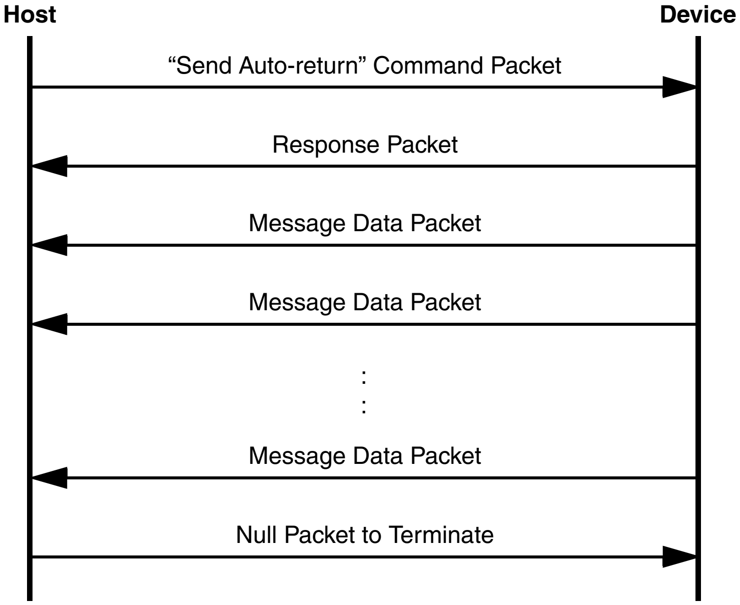


6.5.2 Send Auto-return Messages

6.5.2.1 Introduction

With this command the device can be configured to return new messages from the Message Processor T5 object autonomously. The packet sequence to do this is shown in Figure 6-12.

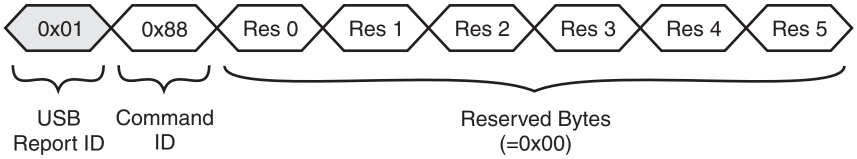
Figure 6-12. Packet Sequence for “Send Auto-return” Command.



The USB Report ID is 0x01.

The command packet has the format given in Figure 6-13.

Figure 6-13. Command Packet Format

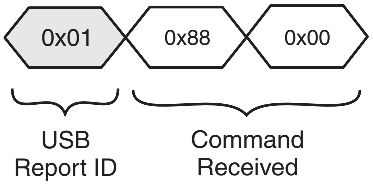


In Figure 6-13:

- **Res 0** to **Res 5** are reserved bytes with a value of 0x00.

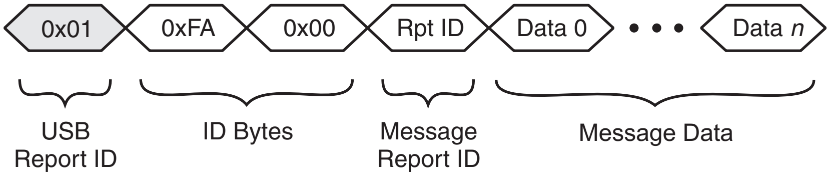
The response packet has the format given in Figure 6-14. Note that with this command, the command packet does not include an address pointer as the device already knows the address of the Message Processor T5 object.

Figure 6-14. Response Packet Format



Once the device has responded to the command, it starts sending message data. Each time a message is generated in the Message Processor T5 object, the device automatically sends a message packet to the host with the data. The message packets have the format given in Figure 6-15.

Figure 6-15. Message Packet Format



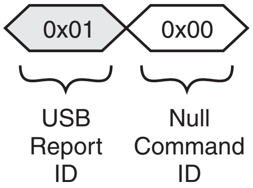
In Figure 6-15:

- **ID Bytes** identify the packet as an auto-return message packet.
- **Rpt ID** is the Report ID returned by the Message Processor T5 object. [1]
- **Message Data** bytes are the bytes of data returned by the Message Processor. The size of the data depends on the source object for which this is the message data. Refer to the *mXT1664S Protocol Guide* for more information.

To stop the sending of the messages, the host can send a null command packet. This consists of two bytes: a report ID of 0x01 and a command byte of 0x00 (see Figure 6-16).

[1] This is the Report ID used in the Object Protocol and should not be confused with the USB Report ID. Refer to the *mXT1664S Protocol Guide* for more information on the use of Report IDs in the Object Protocol.

Figure 6-16. Null Command Packet Format

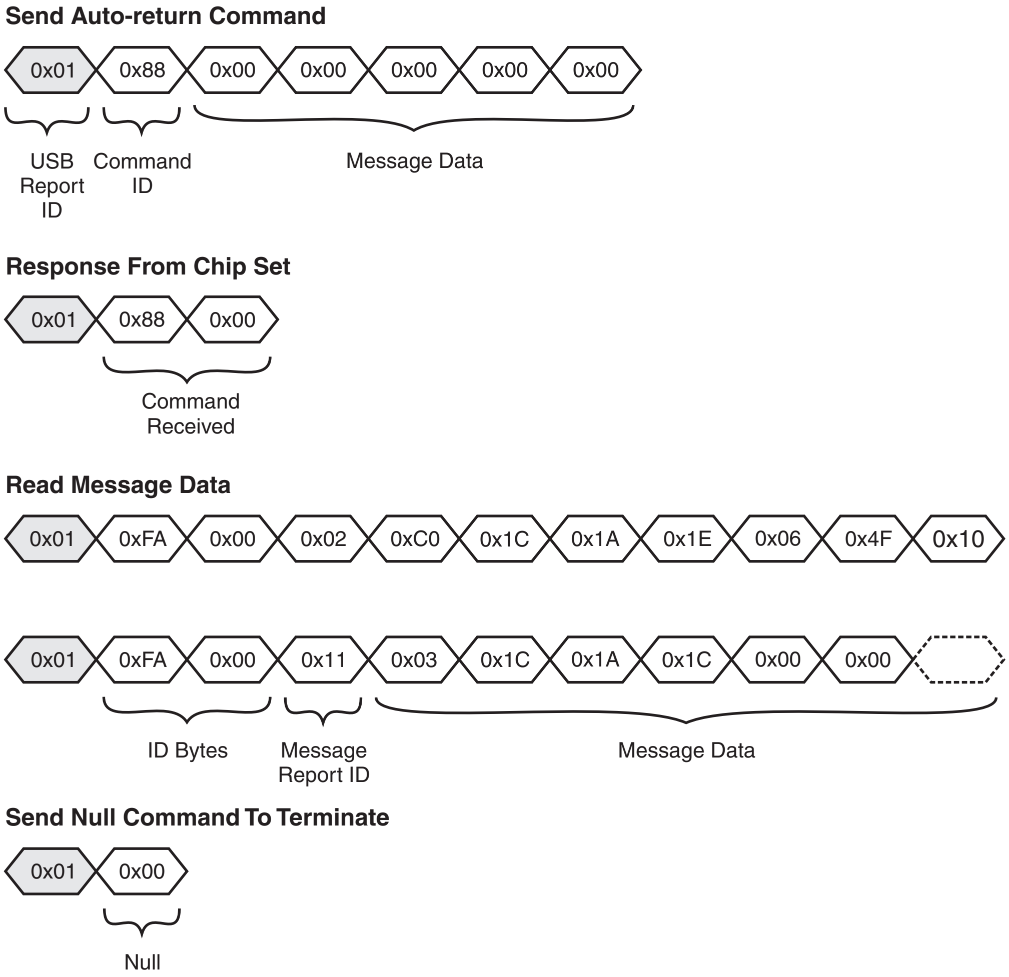


Note that the "Start Debug Monitoring" command may also terminate any currently enabled auto-return mode (see Section 6.5.3).

6.5.2.2 Reading Status Messages

Figure 6-10 shows an example sequence of packets to receive messages from the Message Processor T5 object using the "Send Auto-return" command.

Figure 6-17. Example Auto-return Command Packet



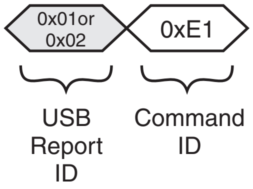
6.5.3 Start Debug Monitoring

This command instructs the device to return debug-monitoring data packets using the debug port, if this feature has been enabled in the Command Processor T6 object.

The USB Report ID can be either 0x01 or 0x02. This allows the source of the request to be identified. The main difference is that a USB Report ID of 0x01 will terminate any currently enabled auto-return mode (see Section 6.5.2 on page 38).

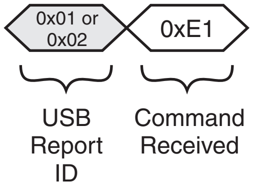
The command packet has the format given in Figure 6-18.

Figure 6-18. Command Packet Format



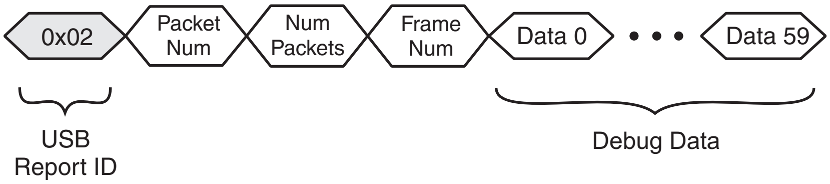
The response packet has the format given in Figure 6-19. Note that the USB Report ID will be the same as that used in the command packet.

Figure 6-19. Response Packet Format



The debug data packet has the format given in Figure 6-20.

Figure 6-20. Debug Data Packet Format



In Figure 6-20:

- **PacketNum** is the number of this USB packet in the debug data frame (full set of debug data). Refer to QTAN0050, *Using the maXTouch Debug Port*, for more information on the format of the debug data.
- **NumPackets** is the total number of USB packets that make up a debug data frame.
- **FrameNum** is the ID number of this frame.
- **Data 0** to **Data 59** are 59 bytes of debug data.

6.5.4 Stop Debug monitoring

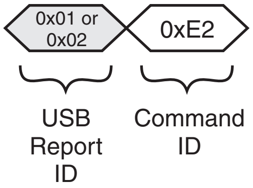
This command instructs the device to cease returning debug-monitoring data packets.

The command packet has the following format:

The USB Report ID is either 0x01 or 0x02.

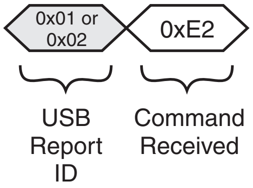
The command packet has the format given in Figure 6-21.

Figure 6-21. Command Packet Format



The response packet has the format given in Figure 6-22.

Figure 6-22. Response Packet Format



6.6 USB Suspend Mode

When the mXT1664S is used in USB configuration, the USB “System Suspend” event can be used to minimize current consumption. Note that it is possible to put the mXT1664S into deep sleep mode without also sending a “System Suspend” event on the USB bus, but the current consumption is not as low. The USB controller must send a USB “System Wake Up” event on the bus to bring the mXT1664S out of suspend mode.

The mXT1664S can also be configured to respond to USB “Remote Wakeup” requests. In this case, if the operating system enables remote wakeup and the mXT1664S is suspended, the device will continue to scan at a preset sensor refresh rate. Use of the remote wake up feature and the sensor refresh rate are configured using the Digitizer HID Configuration T43 object (refer to the *mXT1664S Protocol Guide* for more information).

7. HID-I²C-compatible Communications

7.1 Communications Protocol

The chipset is an HID-I²C DEVICE presenting two top-level collections (TLCs):

Interface 0 (Digitizer HID-I²C) – supplies touch information to the host. This interface is supported by Microsoft Windows 8 without the need for additional software.

Interface 1 (Generic HID-I²C) – This interface provides a generic HID-I²C interface that allows the host to communicate with the chipset using the object protocol.

To use the device in HID-I²C mode, the I2CMODE pin should be pulled low. Other features are identical to standard I²C-compatible communication described in Section 5.2 on page 24.

7.2 I²C-compatible Addresses

See Section 5.2 on page 24.

7.3 Device

The device is compliant with HID-I²C specification V0.9. It has the following specification:

Vendor ID:	0x03EB (Atmel)
Product ID:	0x212C

Version:	16-bit Version & Build Identifier in the form 0xVVBB, where:
	VV = Version Major (Upper 4 bits) / Minor (Lower 4 bits)
	BB = Build number

HID descriptor address:	0x0000.

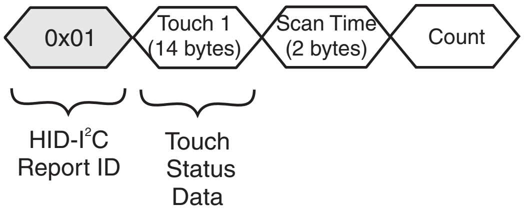
7.4 Interface 0 (Digitizer HID-I²C)

Interface 0 is a digitizer class HID.

7.4.1 Normal Touch Report

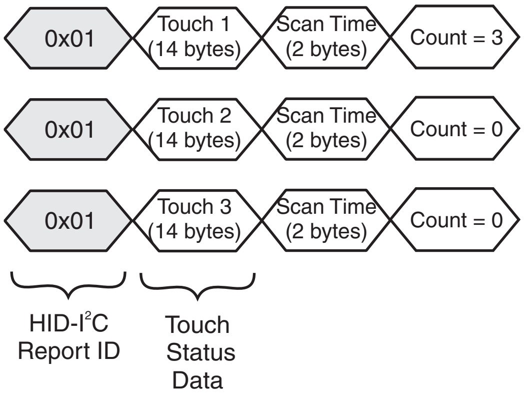
The format of an input report is shown in Figure 7-1. Each input report starts with a report ID and each input report message report contains data of one touch.

Figure 7-1. Input Report Packet



An example of the input report packets for 3 active touches is shown in Figure 6-3 on page 33.

Figure 7-2. Example Input Report Packets for 3 Active Touches



Each input report consists of a HID-I^{2}C report ID followed by 17 bytes of that describe the status of one active touch. See Table 7-1.

Table 7-1. Input Report Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	HID-I2C Report ID							
1	Reserved							Status
2	Touch ID							
3 – 4	X Position							
5 – 6	X' Position							
7 – 8	Y Position							
9 – 10	Y' Position							
11	Touch Width							
12	Reserved							
13	Touch Height							
14	Reserved							
15 – 16	Scan Time							
17	Number of active touches to be sent in one package							

- **Byte 2:**

  Touch ID: Identifies the touch for which this is a status report (starting from 0).

- **Bytes 3 to 10:**

  X and Y positions: These are scaled to 12-bit resolution. This means that the upper four bits of the MSByte will always be zero.

- **Byte 11:**

  Touch Width: Reports the width of the detected touch.

- **Byte 13:**

  Touch Height: Reports the height of the detected touch.

- **Byte 15 to 16:**

Scan Time: Timestamp associated with the current report packet with a 10 kHz resolution.

7.4.2 Active maXStylus Report

The format of an active maXStylus report packet is shown in Figure 7-3.

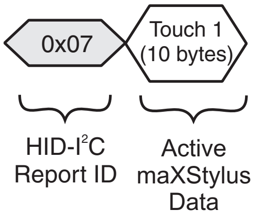


Figure 7-3. Example Active maXStylus Report

Each active maXStylus report start with a HID-I^{2}C Report ID (value 0x06). Table 7-2 gives the detailed format of an active stylus report packet.

Table 7-2. Active maXStylus Report

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	HID-I^{2}C Report ID = 0x07							
1	Reserved			In Range	Reserved	Eraser	Barrel	Tip
2	X Position							
3								
4	X Center Position							
5								
6	Y Position							
7								
8	Y Center Position							
9								
10	Tip Pressure							

- **Byte 1:**

  Tip: 1 = Contact of the stylus with the touch screen surface, 0 = No contact.

  Barrel: 1 = Barrel button on, 0 = Barrel button off

  Eraser: 1 = Eraser function on, 0 = Eraser function off.

  In Range: 1 = Stylus approaching the touchscreen detected, 0 = No stylus approaching the touchscreen detected.

- **Byte 2 to 3:**

  X position

- **Byte 4 to 5:**

  X center position

- **Byte 6 t o7:**

  Y position

- **Byte 8 to 9:**

Y center position

- **Byte 10:**

  Tip Pressure: Force exerted against the touch screen surface by the stylus.

7.5 Interface 1 (Generic HID-I^{2}C)

Interface 1 is a generic human interface device. It supports an input report for receiving data from the device and an output report for sending data to the device.

Commands are sent by the host using the output reports. Responses from the device are sent using input reports.

Supported commands are:

- Read/Write Memory Map
- Send Auto-return Messages

7.5.1 Read/Write Memory Map

7.5.1.1 Introduction

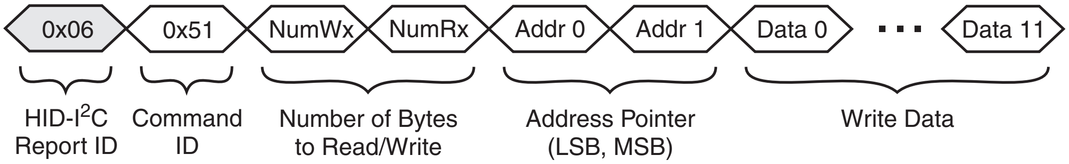
This command is used to carry out a write/read operation on the memory map of the device.

The HID-I^{2}C Report ID is 0x06.

Note: This value may change.

The command packet has the generic format given in Figure 7-4. The following sections give examples on using the command to write to the memory map and to read from the memory map.

Figure 7-4. Generic Command Packet Format

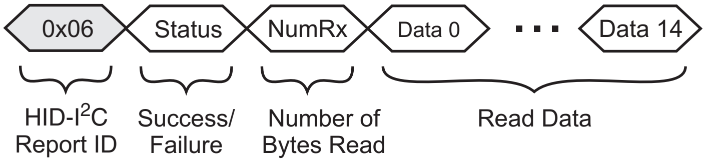


In Figure 7-4:

- **NumWx** is the number of data bytes to write to the memory map (may be zero). If the address pointer is being sent, this must include the size of the address pointer.
- **NumRx** is the number of data bytes to read from the memory map (may be zero).
- **Addr 0** and **Addr 1** form the address pointer to the memory map (where necessary; may be zero if not needed).
- **Data 0** to **Data 11** are the bytes of data to be written (in the case of a write). Note that data locations beyond the number specified by NumWx will be ignored.

The response packet has the generic format given in Figure 7-5.

Figure 7-5. Response Packet Format



In Figure 7-5 on page 47:

- **Status** indicates the result of the command:

  0x00 = read and write completed; read data returned

  0x04 = write completed; no read data requested
- **NumRx** is the number of bytes following that have been read from the memory map (in the case of a read). This will be the same value as NumRx in the command packet.
- **Data 0** to **Data 14** are the data bytes read from the memory map.

7.5.1.2 *Writing To the Device*

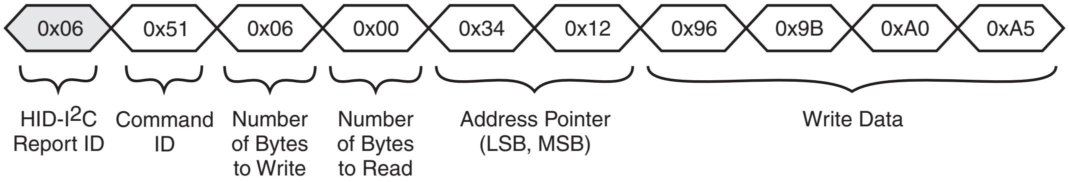
A write operation cycle to the device consists of sending a packet that contains six header bytes. These specify the HID-I^{2}C report ID, the Command ID, the number of bytes to read, the number of bytes to write, and the 16-bit address pointer.

Subsequent bytes in a multibyte transfer form the actual data. These are written to the location of the address pointer, location o

If the address pointer +1, location of the address pointer + 2, and so on.

Figure 7-6 shows an example command packet to write four bytes of data to contiguous addresses starting at 0x1234.

Figure 7-6. Example of a Four-byte Write Starting at Address 0x1234

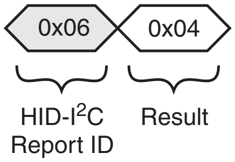


In Figure 7-6:

- The number of bytes to read is set to zero as this is a write-only operation.
- The number of bytes to write is six: that is, four data bytes plus the two address pointer bytes.

Figure 7-7 shows the response to this command. Note that the result status returned is 0x04 (that is, the write operation was completed but no read data was requested).

Figure 7-7. Response to Example Four-byte Write

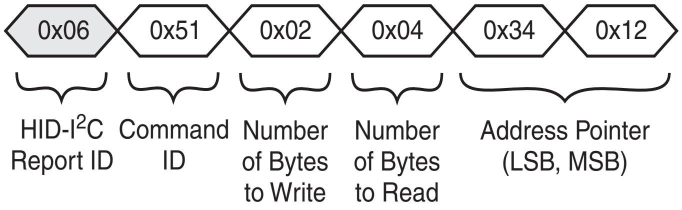


7.5.1.3 *Reading From the Device*

A read operation consists of sending a packet that contains the six header bytes only and no write data.

Figure 7-8 shows an example command packet to read four bytes starting at address 0x1234. Note that the address pointer is included in the number of bytes to write, so the number of bytes to write is set to 2 as there are no other data bytes to be written.

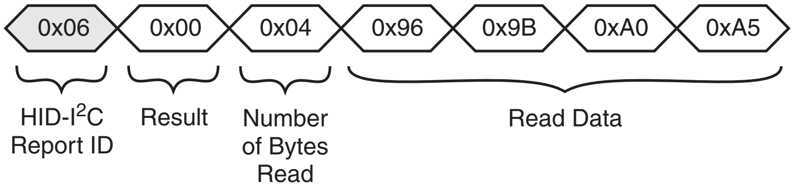
Figure 7-8. Example of a Four-byte Read Starting at Address 0x1234



It is not necessary to set the address pointer before every read. The address pointer is updated automatically after every read operation, so the address pointer will be correct if the reads occur in order.

Figure 7-9 shows the response to this command. The result status returned is 0x00 (that is the write operation was completed and the data was returned). The number of bytes returned will be the same as the number requested (4 in this case).

Figure 7-9. Response to Example Four-byte Read

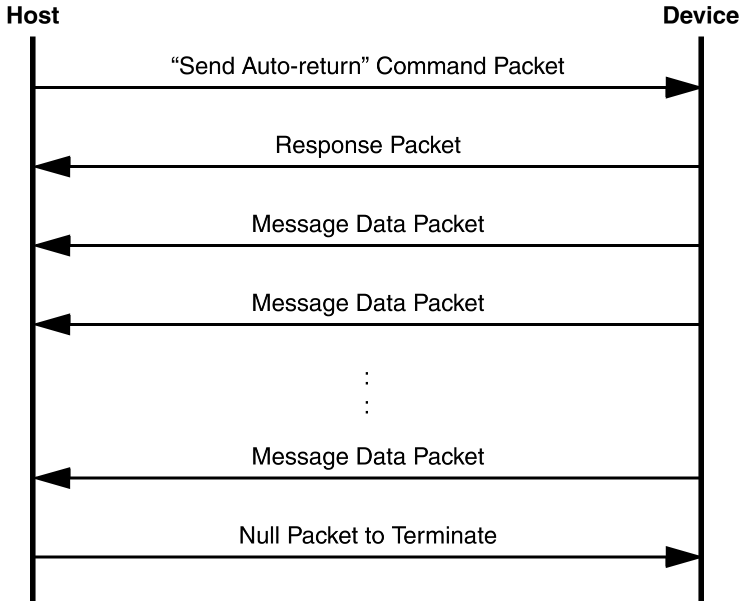


7.5.2 Send Auto-return Messages

7.5.2.1 Introduction

With this command the device can be configured to return new messages from the Message Processor object autonomously. The packet sequence to do this is shown in Figure 7-10.

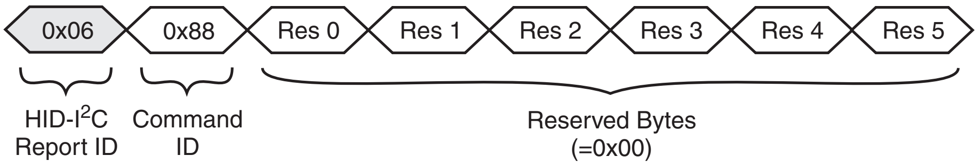
Figure 7-10. Packet Sequence for “Send Auto-return” Command.



The HID-I^{2}C Report ID is 0x06.

The command packet has the format given in Figure 7-11 on page 49.

Figure 7-11. Command Packet Format

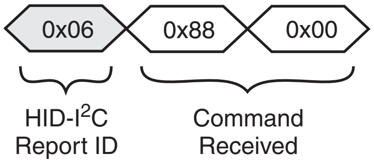


In Figure 7-11:

- **Res 0** to **Res 5** are reserved bytes with a value of 0x00.

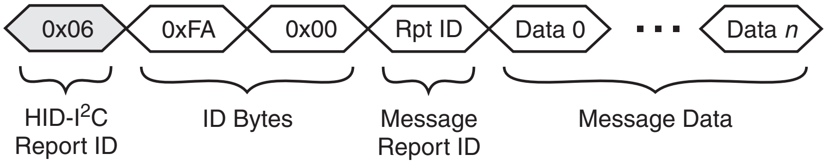
The response packet has the format given in Figure 7-12. Note that with this command, the command packet does not include an address pointer as the device already knows the address of the Message Processor object.

Figure 7-12. Response Packet Format



Once the device has responded to the command, it starts sending message data. Each time a message is generated in the Message Processor object, the device automatically sends a message packet to the host with the data. The message packets have the format given in Figure 7-13.

Figure 7-13. Message Packet Format

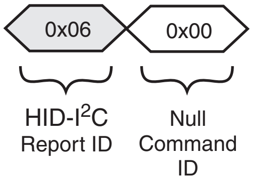


In Figure 7-13:

- **ID Bytes** identify the packet as an auto-return message packet.
- **Rpt ID** is the Report ID returned by the Message Processor object. [1]
- **Message Data** bytes are the bytes of data returned by the Message Processor. The size of the data depends on the source object for which this is the message data. Refer to the *mXT1664S Protocol Guide* for more information.

To stop the sending of the messages, the host can send a null command packet. This consists of two bytes: a report ID of 0x01 and a command byte of 0x00 (see Figure 7-13).

Figure 7-14. Null Command Packet Format



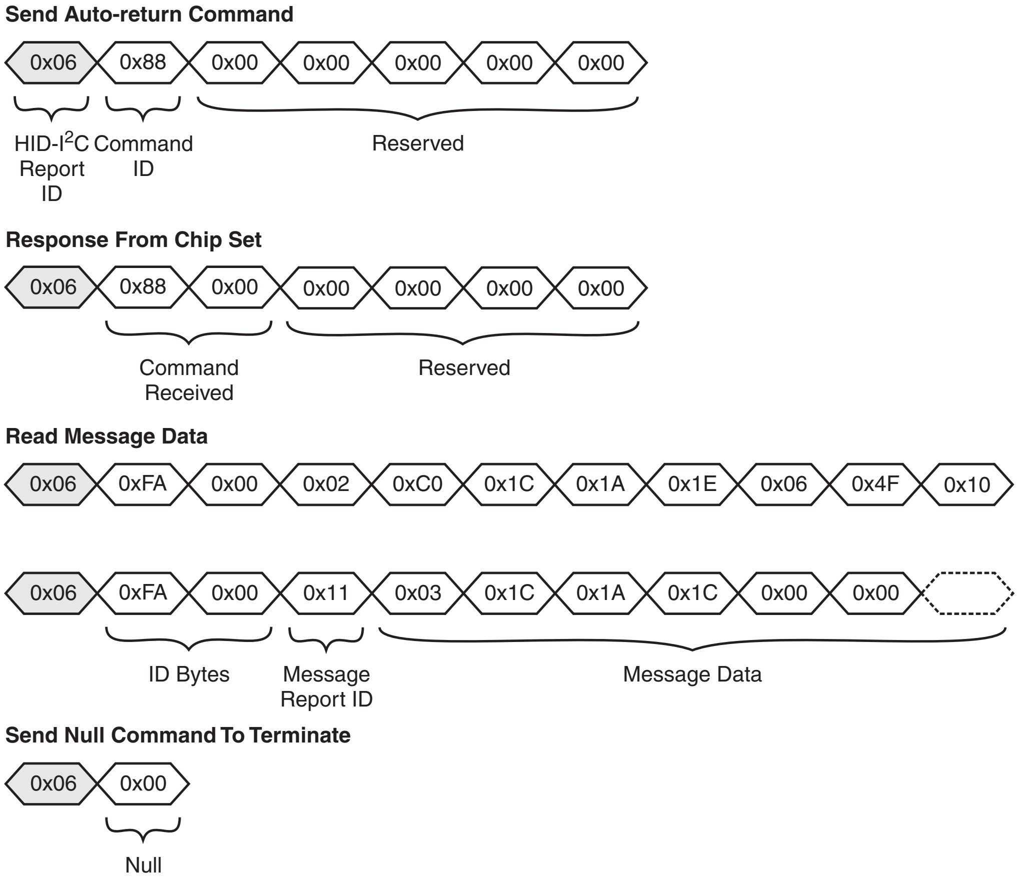
Note that any read or write will also terminate any currently enabled auto-return mode (see "Start Debug Monitoring" on page 40).

[1] This is the Report ID used in the Object Protocol and should not be confused with the USB Report ID. Refer to the *mXT1664S Protocol Guide* for more information on the use of Report IDs in the Object Protocol.

7.5.2.2 *Reading Status Messages*

Figure 7-15 shows an example sequence of packets to receive messages from the Message Processor object using the “Send Auto-return” command.

Figure 7-15. Example Auto-return Command Packet



7.6 $\overline{\text{CHG}}$ Line

The $\overline{\text{CHG}}$ line is an active-low, open-drain output that is used to alert the host that a new message is available in the Input Buffer. This provides the host with an interrupt-style interface with the potential for fast response times. It reduces the need for wasteful I^{2}C-compatible communications.

Further information on the $\overline{\text{CHG}}$ line is given in Section 5.7 on page 28.

7.7 SDA, SCL

Identical to standard I^{2}C-compatible operation. Refer to Section 5.8 on page 30.

7.8 Clock Stretching

Identical to standard I^{2}C-compatible operation. Refer to Section 5.9 on page 30.

7.9 Microsoft Windows 8 Compliance

The mXT1664S has algorithms within the Digitizer HID Configuration T43 and Multiple Touch Touchscreen T9 specifically to ensure Microsoft Windows 8 compliance. These, and other device features, may need specific tuning.

8. Getting Started with mXT1664S

8.1 Establishing Contact

8.1.1 Communication with the Host

The host can use either the I^{2}C-compatible interface (see Section 5.1 on page 24) or the USB interface (see Section 6.1 on page 31) to communicate with the device.

8.1.2 I^{2}C-compatible Interface

On power-up, the $\overline{\text{CHG}}$ line goes low to indicate that there is new data to be read from the Message Processor T5 object. If the $\overline{\text{CHG}}$ line does not go low, there is a problem with the device.

The host should attempt to read any available messages to establish that the device is present and running following power-up or a reset. Examples of messages include reset or calibration messages. The host should also check that there are no configuration errors reported.

8.1.3 USB Interface

The host can establish contact with the device as specified in the USB 2.0 specification and the USB HID specification (both available from www.usb.org).

8.2 Using the Object Protocol

The device has an object-based protocol that is used to communicate with the device. Typical communication includes configuring the device, sending commands to the device, and receiving messages from the device. Refer to the *mXT1664S Protocol Guide* for more information.

The host must perform the following initialization so that it can communicate with the device:

1. Read the start positions of all the objects in the device from the Object Table and build up a list of these addresses.
2. Use the Object Table to calculate the report IDs so that messages from the device can be correctly interpreted.

8.3 Writing to the Device

There are three mechanisms for writing to the device:

- Using an I^{2}C-compatible write operation (see Section 5.3 on page 24).
- Using the USB Generic HID write operation (see Section 6.5.1.2 on page 37).
- Using the Generic HID-I^{2}C write operation (see Section 7.5.1.2 on page 47).

To communicate with the device, you write to the appropriate object:

- To send a command to the device, you write the appropriate command to the Command Processor T6 object (refer to the *mXT1664S Protocol Guide*).
- To configure the device, you write to an object. For example, to configure the device's power consumption you write to the global Power Configuration T7 object, and to set up a touchscreen you write to a Multiple Touch Touchscreen T9 object. Some objects are optional and need to be enabled before use. Refer to the *mXT1664S Protocol Guide* for more information on the objects.

8.4 Reading from the Device

Status information is stored in the Message Processor T5 object. This object can be read to receive any status information from the device. There are two mechanisms that provide an interrupt-style interface for reading messages in the Message Processor T5 object:

- When using the I^{2}C-compatible interface, the $\overline{\text{CHG}}$ line is asserted whenever a new message is available in the Message Processor T5 object (see Section 5.7 on page 28). See Section 6.5.1.3 on page 37 for information on the format of the I^{2}C-compatible read operation.
- When using the USB interface, the Generic HID interface provides an interrupt-driven interface that sends the messages automatically (see Section 6.5.2 on page 38).
- When using the HID-I^{2}C interface, the Generic HID-I^{2}C interface provides an interrupt-driven interface that sends the messages automatically (see Section 7.5.1.3 on page 48)

Note that in both cases the host should always wait to be notified of messages. The host should not poll the device for messages.

The USB Digitizer HID provides a third alternative interrupt-style mechanism for reading a subset of the touch data. See Section 6.4 on page 32 for more information.

8.5 Configuring the Device

The objects are designed such that a default value of zero in their fields is a “safe” value that typically disables functionality. The objects must be configured before use and the settings written to the nonvolatile memory using the Command Processor T6 object.

Perform the following actions for each object:

1. Enable the object, if the object requires it.
2. Configure the fields in the object, as required.
3. Enable reporting, if the object supports messages, to receive messages from the object.

Refer to the *mXT1664S Protocol Guide* for more information on configuring the objects.

The following objects are read-only and require no configuration:

- Debug Objects
  - Diagnostic Debug T37
- General objects:
  - Message Processor T5
- Support objects:
  - User Data T38
  - Message Count T44

The following objects must be configured before use:

- General objects
  - Power Configuration
  - Acquisition Configuration

The following objects should be checked and configured as necessary:

- General objects:
  - Command Processor T6
- Support objects:
  - Communications Configuration T18
  - CTE Configuration T46

The following objects should also be enabled and configured, as required:

- Touch objects:
  - Multiple Touch Touchscreen T9
  - Key Array T15
- Signal processing objects:
  - Grip Suppression T40
  - Stylus T47
  - One-touch Gesture Processor T24
  - Two-touch Gesture Processor T27
  - Touch Suppression T42
  - Adaptive Threshold T55
  - SlimSensor T56
  - Extra Touchscreen Data T57
  - maXCharger T62
- Support objects:
  - Digitizer HID Configuration T43
  - Self Test T25
  - Timer T61

9. Specifications

See Appendix B on page 75 for the reference configuration.

9.1 Absolute Maximum Specifications

Vdd	3.6 V
XVdd	12 V
AVdd	3.6 V
Vdd Core (for 1.8V I^{2}C-compatible mode only)	1.98 V
Max continuous pin current, any control or drive pin	20 mA
Voltage forced onto any pin	–0.3 V to (Vdd or AVdd) + 0.3 V
Configuration parameters maximum writes	10,000

⚠ **CAUTION:** Stresses beyond those listed under *Absolute Maximum Specifications* may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum specification conditions for extended periods may affect device reliability.

9.2 Recommended Operating Conditions

Operating temp	–40°C to +85°C
Storage temp	–60°C to +150°C
Vdd External	1.8 V –5% +10%(I^{2}C-compatible 1.8 V mode)
	2.7 to 3.3 V ±10% (I^{2}C-compatible mode)
	3.0 to 3.3 V ±10% (USB mode)
AVdd	2.7 to 3.3 V ±10%
XVdd	2.7 to 6.0 V ± 10%
Vdd vs AVdd power sequencing	No sequencing required
Vdd vs XVdd power sequencing	XVdd must not be powered before Vdd
Vdd supply ripple	±50 mV 1 Hz to 1 MHz
XVdd supply ripple	±25 mV 1 Hz to 1MHz
AVdd supply ripple (Noise suppression disabled)	±25 mV 1 Hz to 1 MHz
Cx transverse load capacitance per channel	0.95 pF to 4.8 pF, when XVdd = 2.7 V to 4.75 V 0.5 pF to 4.8 pF, when XVdd = 4.75 V to 6.0 V
Temperature slew rate	10°C/min

9.3 DC Characteristics

9.3.1 Digital Voltage Supply

Parameter	Description	Min	Typ	Max	Units	Notes
Vdd	Operating limits	1.71		1.98	V	I^{2}C-compatible 1.8 V
		2.43		3.6	V	I^{2}C-compatible mode
		2.7		3.6	V	USB
Rate of rise	Rate of rise	0.002		2.5	V/µs	

9.3.2 Analog Voltage Supply

Parameter	Description	Min	Typ	Max	Units	Notes
AVdd	Operating limits	2.5		3.6	V	See Section 4.6.3 on page 21
Rate of rise	Rate of rise	0.002		2.5	V/µs	

9.3.3 XVdd Supply

Parameter	Description	Min	Typ	Max	Units	Notes
XVdd	Operating limits	2.5	6.0	6.3	V	
Rate of rise	Rate of rise	2		30	V/ms	

Note: The rate of rise values must be followed to avoid permanent damage to the device.

9.3.4 Input/Output Characteristics

Parameter	Description	Min	Typ	Max	Units	Notes
Input ($\overline{\text{RESET}}$, SDA, SCL)						
Vil	Low input logic level	–0.3		0.3 × Vdd	V	Vdd = 1.8 V to 3.3 V
Vih	High input logic level	0.7 × Vdd		Vdd + 0.3	V	Vdd = 1.8 V to 3.3 V
Iil	Input leakage current			1	µA	Pull-up resistors disabled
Output ($\overline{\text{CHG}}$)						
Vol	Low output voltage			0.2 × Vdd	V	Vdd = 3 V, I_{OL} = 2.7 mA, High Drive Enabled Vdd = 3 V, I_{OL} = 1.35 mA
Voh	High output voltage	0.8 × Vdd			V	Vdd = 3 V, I_{OH} = 2.7 mA, High Drive Enabled Vdd = 3 V, I_{OH} = 1.35 mA

9.4 ESD Information

Parameter	Value	Reference standard
Electrostatic Discharge HBM	±2000 V	MIL– STD883 Method 3015.7

9.5 Supply Current

9.5.1 Analog Supply – I^2C-compatible Interface

Parameter	Description	Min	Typ	Max	Units	Notes
AIdd	Active average supply current		21.51		mA	100 Hz, 1 Touch
	Idle average supply current		3.16		mA	16 Hz, no touches
	Sleep average supply current		0.003		mA	

9.5.2 Analog Supply – USB Bus

Parameter	Description	Min	Typ	Max	Units	Notes
AIdd	Active average supply current		20.93		mA	100 Hz, 1 Touch
	Idle average supply current		3.09		mA	16 Hz, no touches
	Sleep average supply current		0.01		mA	

9.5.3 Analog Supply – HID-I^{2}C Bus

Parameter	Description	Min	Typ	Max	Units	Notes
AIdd	Active average supply current		21.36		mA	100 Hz, 1 Touch
	Idle average supply current		3.16		mA	16 Hz, no touches
	Sleep		0.003		mA	

9.5.4 Digital Supply – I^{2}C-compatible Interface

Parameter	Description	Min	Typ	Max	Units	Notes
DIdd	Active average supply current		10.18		mA	100 Hz, 1 Touch
	Idle average supply current		0.92		mA	16 Hz, no touches
	Sleep average supply current		0.11		mA	

9.5.5 Digital Supply – USB Bus

Parameter	Description	Min	Typ	Max	Units	Notes
DIdd	Active average supply current		15.09		mA	100 Hz, 1 Touch
	Idle average supply current		12.16		mA	16 Hz, no touches
	Sleep		0.21		mA	

9.5.6 Digital Supply – HID-I^{2}C

Parameter	Description	Min	Typ	Max	Units	Notes
DIdd	Active average supply current		8.27		mA	100 Hz, 1 Touch
	Idle average supply current		0.92		mA	16 Hz, no touches
	Sleep		0.11		mA	

9.5.7 X Drive Supply – I^{2}C-compatible Interface

Parameter	Description	Min	Typ	Max	Units	Notes
XIdd	Active average supply current		0.25		mA	100 Hz, 1 Touch
	Idle average supply current		0.04		mA	16 Hz, no touches
	Sleep average supply current		0.00		mA	

9.5.8 X Drive Supply – USB Bus

Parameter	Description	Min	Typ	Max	Units	Notes
XIdd	Active average supply current		0.25		mA	100 Hz, 1 Touch
	Idle average supply current		0.04		mA	16 Hz, no touches
	Sleep		0.00		mA	

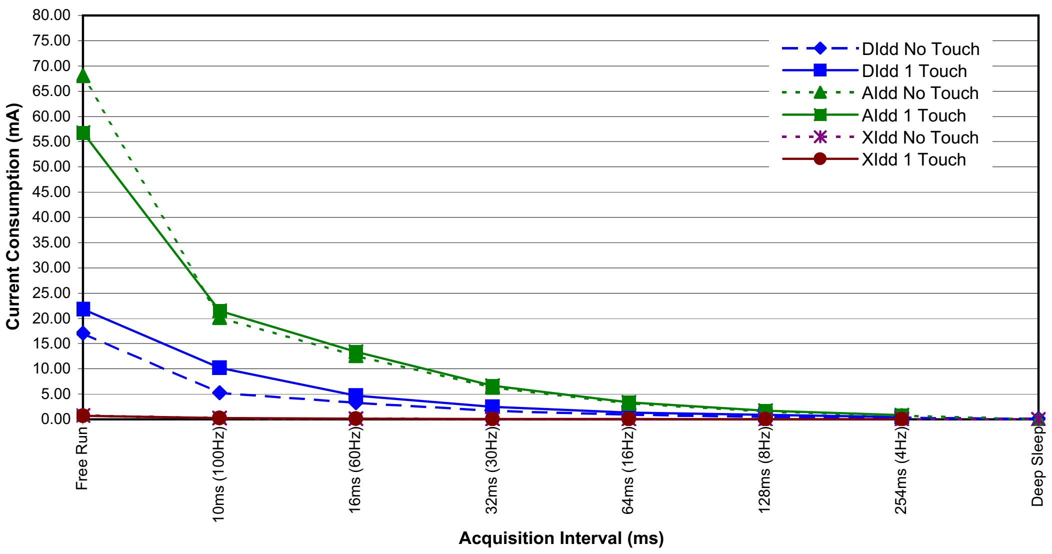
9.5.9 X Drive Supply – HID-I^{2}C Bus

Parameter	Description	Min	Typ	Max	Units	Notes
XIdd	Active average supply current		0.27		mA	100 Hz, 1 Touch
	Idle average supply current		0.04		mA	16 Hz, no touches
	Sleep		0.00		mA	

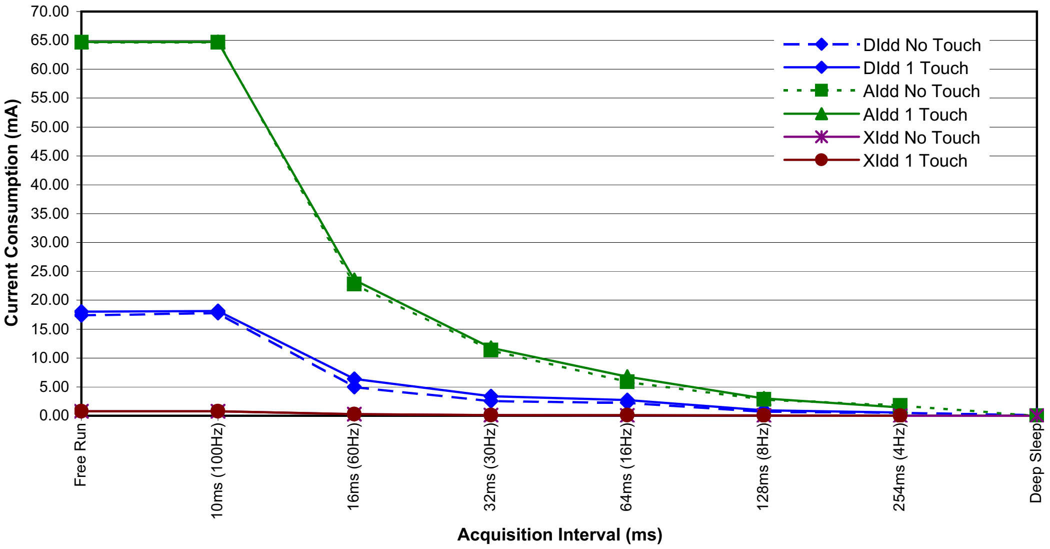
9.6 Power Consumption

9.6.1 I^{2}C-compatible 3.3 V Non-Shieldless

XSIZE = 32, YSIZE = 52, CHRGTIME = 120(2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8,
SlimSensor T56 disabled, Pipelining = on

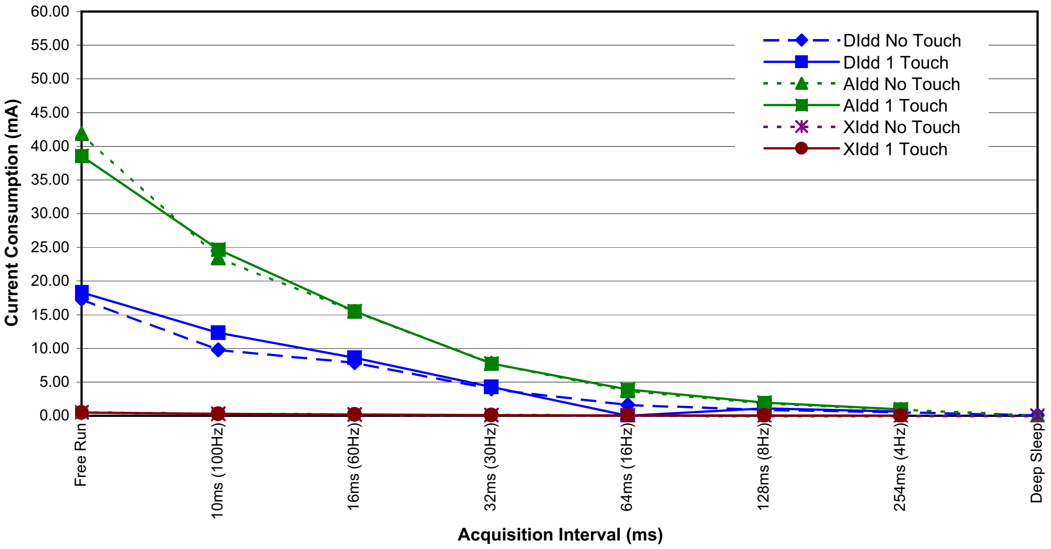


XSIZE = 32, YSIZE = 52, CHRGTIME = 120(2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 16
SlimSensor T56 disabled, Pipelining = on

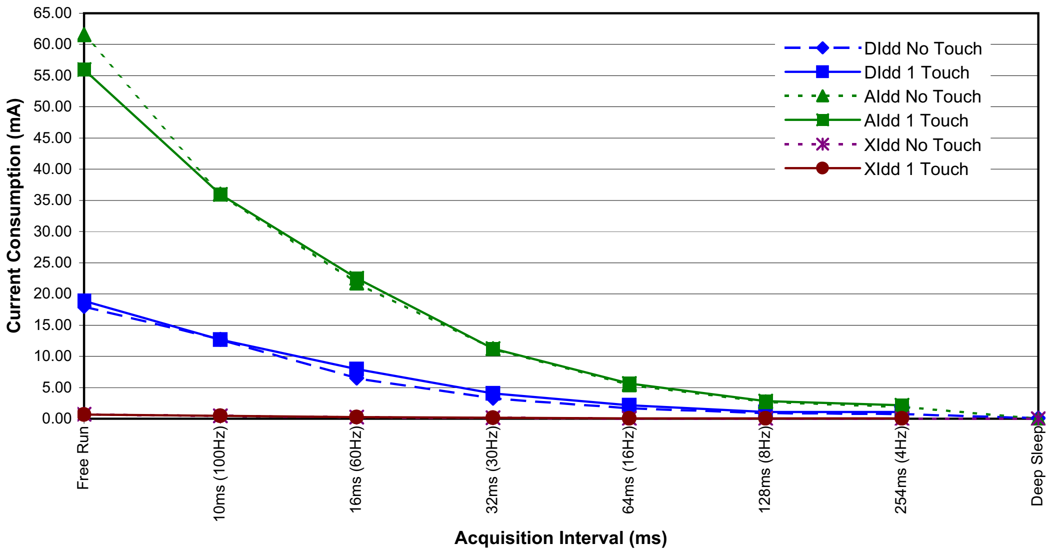


9.6.2 I^{2}C-compatible 3.3 V Shieldless

XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8,
SlimSensor T56 enabled: Multicut GC Noise Cancellation and Optimal Integration= on, Pipelining = on

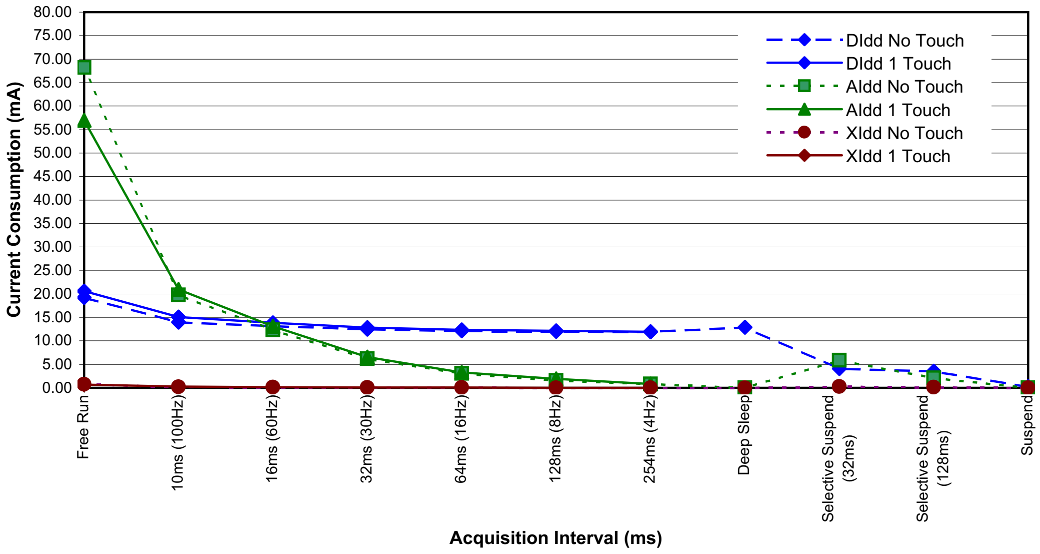


XSIZE = 32, YSIZE = 52, CHRGTIME = 120(2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 16, SlimSensor T56 enabled: Multicut GC Noise Cancellation and Optimal Integration= on, Pipelining = on

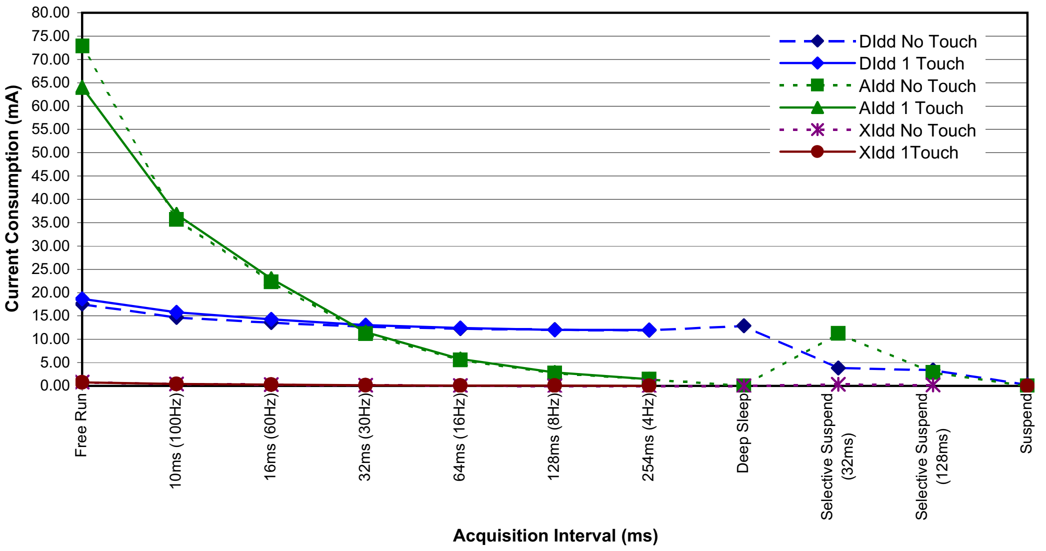


9.6.3 USB 3.3 V Non-shieldess

XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8, SlimSensor T56 disabled, Pipelining = on

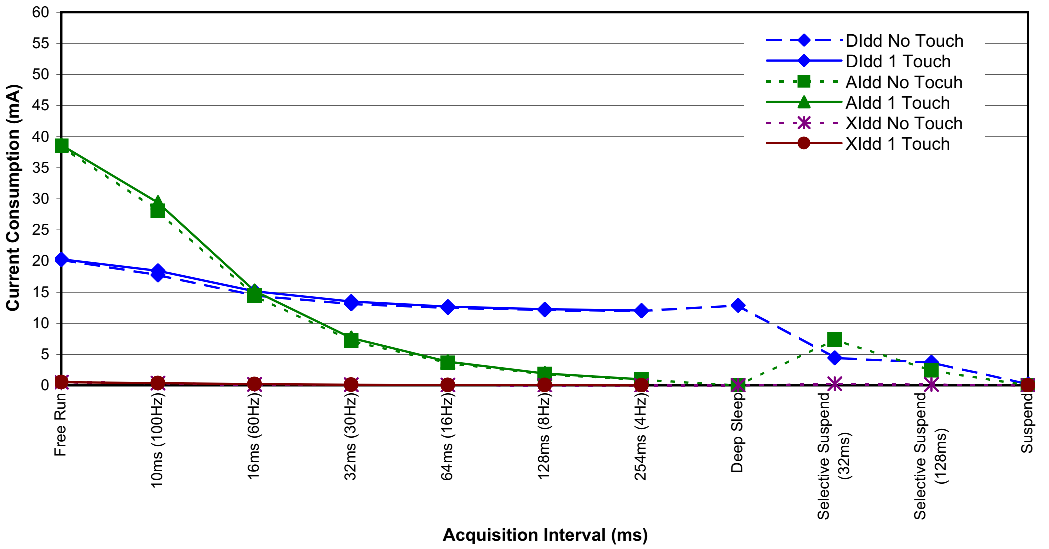


XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 16, SlimSensor T56 disabled, Pipelining = on

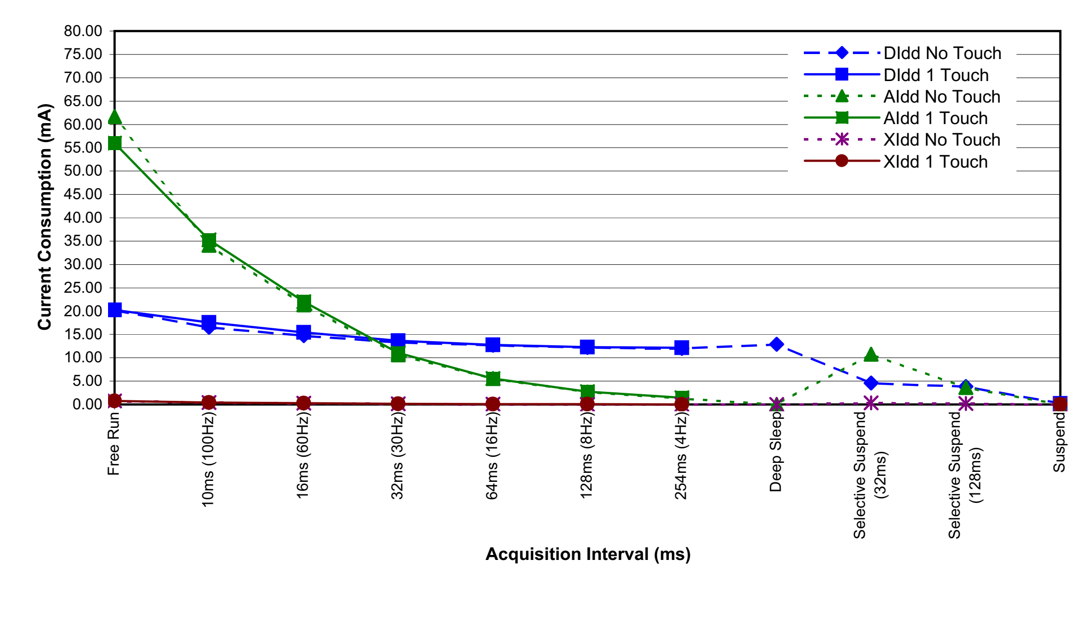


9.6.4 USB 3.3 V Shieldless

XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8, SlimSensor T56 enabled: Multicut GC Noise Cancellation and Optimal Integration= on, Pipelining = on

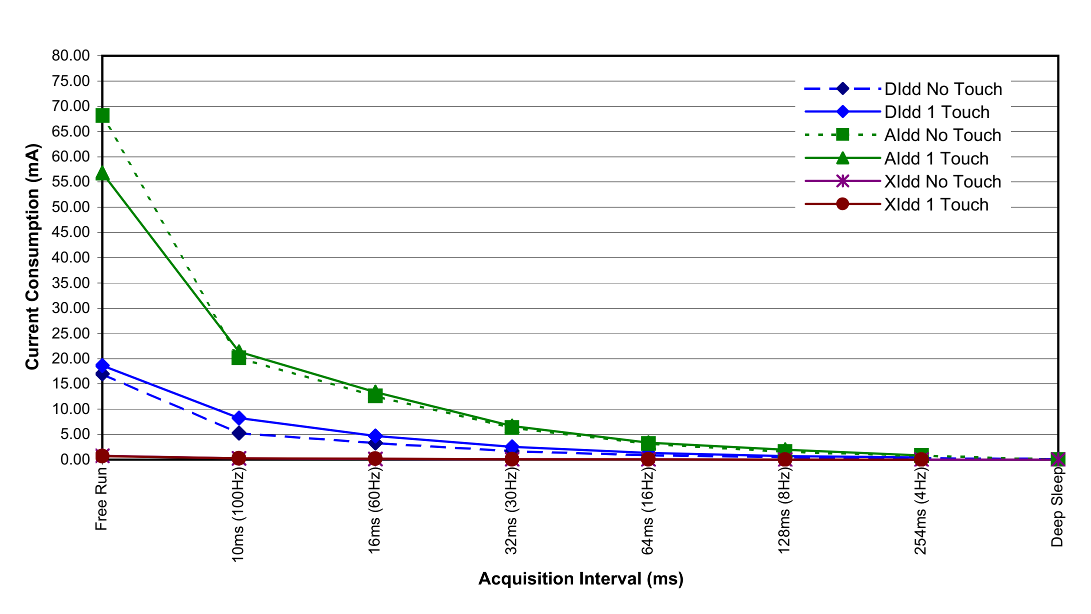


XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 16, SlimSensor T56 enabled: Multicut GC Noise Cancellation and Optimal Integration= on, Pipelining = on



9.6.5 HID-I^{2}C Non-shieldless

XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8, SlimSensor T56 = Off, Pipelining = on

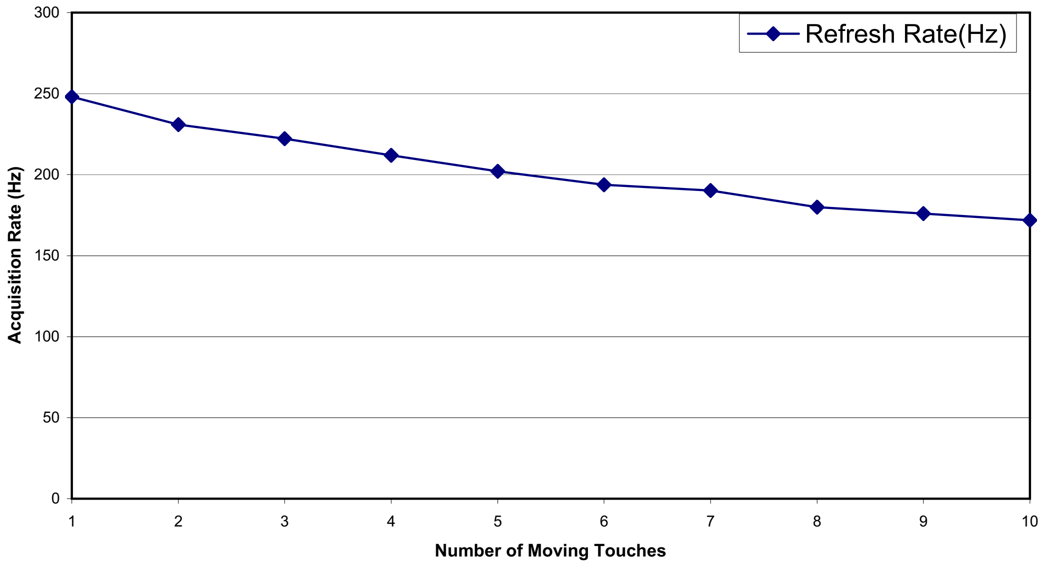


9.7 Timing Specifications

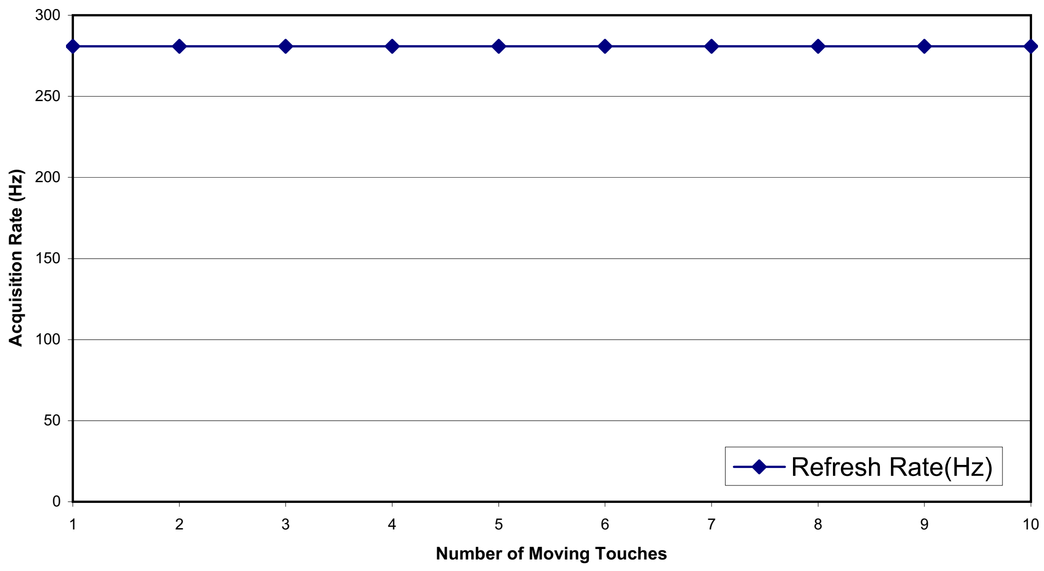
Parameter	Description	Min	Typ	Max	Units	Notes
Tlatency	80 Hz	11	19.5	29	ms	TCHDI = 0 IDLESYNCSPERX/ ACTSYNCSPERX = 8
	100 Hz	11	16	24	ms	
	200 Hz	11	12.5	14	ms	

9.8 Speed

XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8, SlimSensor T56 = Off, Pipelining = off, maXCharger T62 = off



XSIZE = 32, YSIZE = 52, CHRGTIME = 120 (2.5 µs), IDLESYNCSPERX = ACTVSYNCSPERX = 8, SlimSensor T56 = Off, Pipelining = on, maXCharger T62 = off



9.9 Reset Timings

Parameter	Min	Typ	Max	Units	Notes
Power on to $\overline{\text{CHG}}$ line low		65	70	ms	
Hardware reset to $\overline{\text{CHG}}$ line low		65	70	ms	
Software reset to $\overline{\text{CHG}}$ line low		105	280	ms	

9.10 I^{2}C-compatible Specifications

Parameter	Value
Addresses	0x4A or 0x4B
Maximum bus speed (SCL)	1.7 MHz
I^{2}C specification	Version 2.1
Required pull-up resistance for standard mode (100 kHz)	1 kΩ to 10 kΩ
Required pull-up resistance for fast mode (400 kHz)	1 kΩ to 3 kΩ
Required pull-up resistance for high-speed mode (1.7 MHz)	0.5 kΩ to 0.75 kΩ

9.11 USB Specification

Parameter	Operation
Endpoint Addresses	0x81 (Endpoint 1) 0x02 (Endpoint 2) 0x83 (Endpoint 3)
Maximum bus speed	12 Mbps
Vendor ID	0x03EB (Atmel)
Product ID	0x212C (mXT1664S)
USB specification	USB 2.0 HID specification 1.11 with amendments for multitouch digitizers

9.12 HID-I^{2}C Specification

Parameter	Operation
Vendor ID	0x03EB (Atmel)
Product ID	0x212C (mXT1664S)
HID-I^{2}C specification	0.9

9.13 Touch Accuracy and Repeatability

Parameter	Min	Typ	Max	Units	Notes
Linearity		±0.5		mm	
Accuracy		±1		mm	
Accuracy at edge		±2		mm	
Repeatability		±0.25		%	X axis with 12-bit resolution

9.14 Power Supply and Ripple Noise

Parameter	Min	Typ	Max	Units	Notes
Vdd			±50	mV	Across frequency range 1 Hz to 1 MHz
XVdd and AVdd			±25	mV	Across frequency range 1 Hz to 1 MHz
XVdd and AVdd			± 40	mV	Across frequency range 1 Hz to 1 MHz, with Noise Suppression enabled

9.15 Thermal Packaging

9.15.1 Thermal Data

Parameter	Typ	Unit	Condition	Package
Junction to ambient thermal resistance	33.7	°C/W	Still air	VFBGA 128, 7 X 7 mm
Junction to case thermal resistance	5.0	°C/W		VFBGA 128, 7 X 7 mm

9.15.2 Junction Temperature

The average chip junction temperature, T_J in °C can be obtained from the following:

$$T_J = T_A + (P_D \times \theta_{JA})$$

If a cooling device is required, use this equation:

$$T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$$

where:

- θ_{JA}= package thermal resistance, Junction to ambient (°C/W).
- θ_{JC} = package thermal resistance, Junction to case thermal resistance (°C/W).
- $\theta_{HEATSINK}$ = cooling device thermal resistance (°C/W), provided in the cooling device datasheet.
- P_D = device power consumption (W).
- T_A is the ambient temperature (°C).

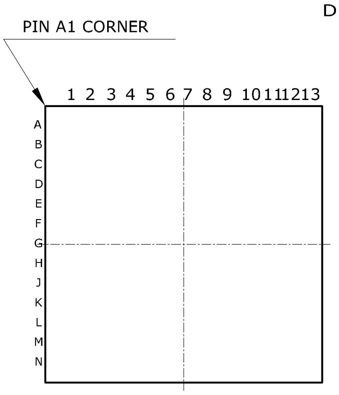
9.16 Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/s max
Preheat Temperature 175°C ±25°C	150 – 200°C
Time Maintained Above 217°C	60 – 150 s
Time within 5°C of Actual Peak Temperature	30 s
Peak Temperature Range	260°C
Ramp down Rate	6°C/s max
Time 25°C to Peak Temperature	8 minutes max

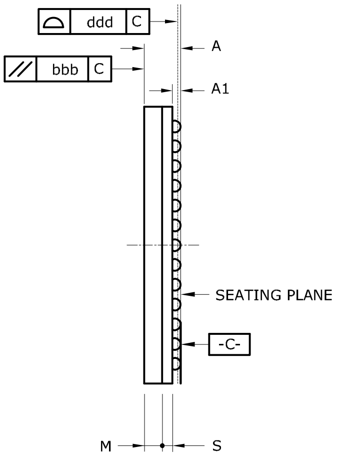
9.17 Mechanical Dimensions

9.17.1 128-ball 7 x 7 x 1 mm

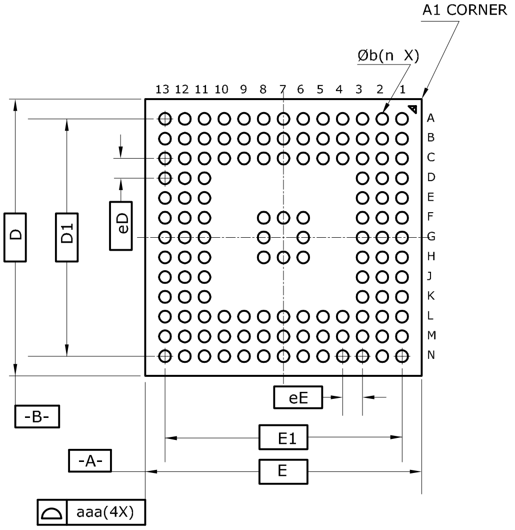
DRAWINGS NOT SCALED



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS

(Unit of Measure = mm)

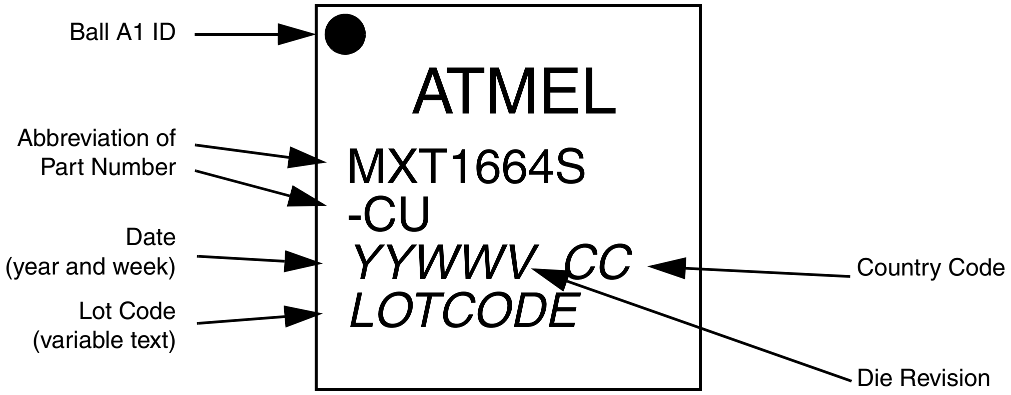
SYMBOL	MIN	NOM	MAX	NOTE
A	-----	-----	1.00	3
A1	0.160	-----	0.260	
E/D	7.00 / 7.00			
E1/D1	6.00 / 6.00			
ball width	0.270	-----	0.370	4
eD/eE	Ball pitch : 0.500			
M	Mold thickness : 0.450 ref			
S	Subst thickness : 0.260 ref			
aaa	Pack edge tolerance : 0.100			
bbb	Mold flatness : 0.100			
ddd	Coplanarity : 0.080			
ball diam	0.300			
n	128			

Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MO-225, Variation CF for proper dimensions, tolerances, datums, etc.
2. Array as seen from the bottom of the package.
3. Dimension A includes stand-off height A1, package body thickness, and lid height, but does not include attached features.
4. Dimension b is measured at the maximum ball diameter, parallel to primary datum C.

09/16/2011

	TITLE	GPC	DRAWING NO.	REV.
Atmel Package Drawing Contact: packagedrawings@atmel.com	7E, 128-ball (13x13 array, 3 rings + 3x3, 1 ring), 0.5mm pitch, 7x7x1mm Very Thin Fine-Pitch Ball Grid Array Package (VFBGA)	CGB	7E	A

9.18 Part Marking



9.19 Part Numbers

Part Number	QS Number	Description
ATMXT1664S-CU	QS620	128-ball 7 x 7 mm VFBGA RoHS compliant Supplied in trays
ATMXT1664S-CUR	QS620	128-ball 7 x 7 mm VFBGA RoHS compliant Supplied in tape and reels

9.20 Moisture Sensitivity Level (MSL)

MSL Rating	Peak Body Temperature	Specifications
MSL3	260°C	IPC/JEDEC J-STD-020

Appendix A. PCB Design Considerations

A.1 Introduction

The following sections give the design considerations that should be adhered to when designing a PCB layout for use with the mXT1664S. Of these, power supply and ground tracking considerations are the most critical.

By observing the following design rules, and with careful preparation for the PCB layout exercise, designers will be assured of a far better chance of success and a correctly functioning product.

A.2 Printed Circuit Board

Atmel recommends the use of a four layer printed circuit board for mXT1664S applications. This, together with careful layout, will ensure that the board meets relevant EMC requirements for both noise radiation and susceptibility, as laid down by the various national and international standards agencies.

A.3 Supply Rails and Ground Tracking

Power supply and clock distribution are the most critical parts of any board layout. Because of this, it is advisable that these be completed before any other tracking is undertaken. After these, supply decoupling, and analog and high speed digital signals should be addressed. Track widths for all signals, especially power rails should be kept as wide as possible in order to reduce inductance.

The Power and Ground planes themselves can form a useful capacitor. Flood filling for either or both of these supply rails, therefore, should be used where possible. It is important to ensure that there are no floating copper areas remaining on the board: all such areas should be connected to the 0 V plane. The flood filling should be done on the outside layers of the board.

In applications where the USB bus supplies power to the board, care should be taken to ensure that suitable capacitive decoupling is provided close to the USB connector. The tracking to the on-board regulators should also be kept as short as possible.

It should also be remembered that the screen of the USB cable is not intended to be connected to the ground or 0V supply of a remote device. It should either be left open circuit (being connected only at the host computer end) or decoupled with a suitable high voltage capacitor (typically 4.7 nF – 250 V) and a parallel resistor (typically 1 MΩ). Note that these components may not be required when the USB cabling is internal and permanently wired, and is routed away from the noisier parts of the system.

A.4 Power Supply Decoupling

As a rule, a suitable decoupling capacitor should be placed on each and every supply pin on all digital devices. It is important that these capacitors are placed as close to the chip’s supply pins as possible (less than 5mm away). The ground connection of these capacitors should be tracked to 0V by the shortest, heaviest traces possible.

Capacitors with a Type II dielectric, such as X5R or X7R and with a value of at least 100nF, should be used for this purpose.

In addition, at least one ‘bulk’ tantalum decoupling capacitor, with a minimum value of 4.7 µF should be placed on each power rail, close to where the supply enters the board.

Surface mounting capacitors are preferred to wire leaded types due to their lower ESR and ESL. It is often possible to fit these decoupling capacitors underneath and on the opposite side of the PCB to the digital ICs. This will provide the shortest tracking, and most effective decoupling possible.

Refer to the application note *Selecting Decoupling Capacitors for Atmel’s PLDs* (doc0484.pdf; available on Atmel’s website) for further general information on decoupling capacitors.

A.5 Suggested Voltage Regulator Manufacturers

The AVdd supply stability is critical for the mXT1664S because this supply interacts directly with the analog front end. Atmel therefore recommends that the supply for the analog section of the board be supplied by a regulator that is separate from the logic supply regulator. This reduces the amount of noise injected into the sensitive, low signal level parts of the design.

A single low value series resistor (around 1Ω) is required from the regulator output to the analog supply input on the mXT1664S device. This, together with the regulator output capacitor, and the capacitors at the DC input to the device, forms a simple filter on the supply rail.

A low noise device should be chosen for the regulator. If possible this should have provision for adding a capacitor across the internal reference for further noise reduction. Reference should be made to the manufacturer’s datasheet.

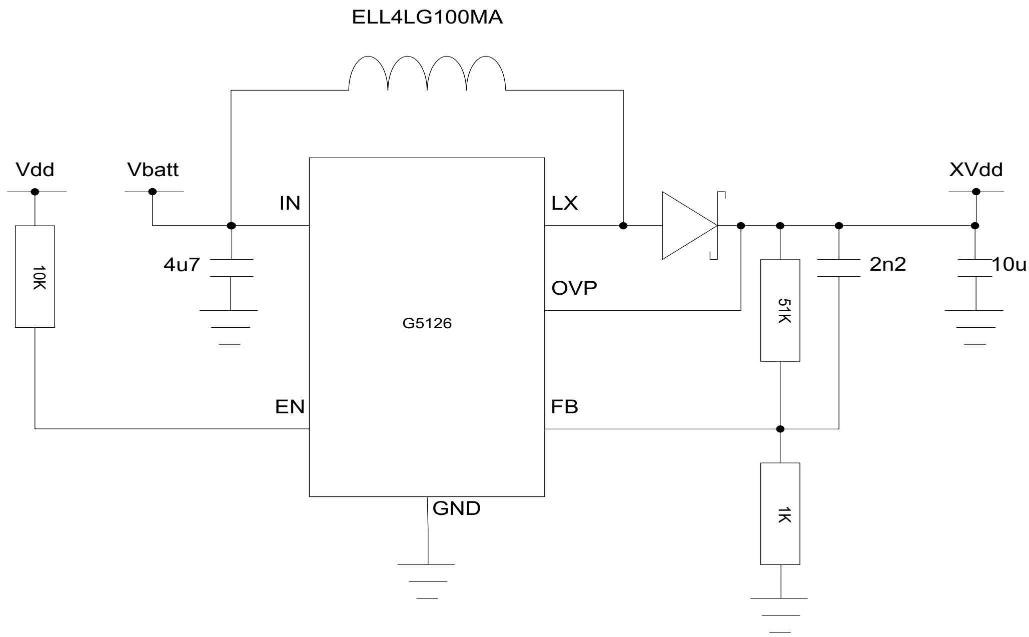
The voltage regulators listed in Table 9-1 have been tested and found to work well with the mXT1664S. They have compatible footprints and pin-out specifications, and are available in the SOT-23 package.

Table 9-1. Recommended Voltage Regulators

Manufacturer	Pin	Part Number
Texas Instruments	AVdd	TLV70028
Linear Technology	Vdd	LT1761
Micrel	Vdd	MIC5255
National Semiconductor	Vdd	LP2981
Torex	Vdd	XC6204
AMS	XVdd	AS1340
GMMT	XVdd	G5126

Note some manufacturers claim that minimal or no capacitance is required for correct regulator operation. However, in all cases, a minimum of a 1.0 µF ceramic, low ESR capacitor at the input and output of these devices should be used. The manufacturers’ datasheets should always be referred to when selecting capacitors for these devices and the typical recommended values, types and dielectrics adhered to.

Figure 9-1. Example Regulator Circuit



Note that a "soft-start" regulator with excellent noise and load step regulation will be needed to satisfy the XVdd supply requirements. 1% resistors should be used to define the nominal output voltage. If 5% resistors are used, the nominal XVdd voltage must be reduced accordingly to ensure that the recommended voltage range is adhered to. Figure 9-1 provides an example circuit for the XVdd supply.

A.6 Crystal Oscillator

If a crystal oscillator is used, its placement is critical to the performance of the design. The connecting leads between the mXT1664S and the crystal should be as short as possible. These tracks, together with the crystal itself, should be placed above a suitable ground plane. It is also important that no other signal tracks are placed close to, or under, these tracks. The crystal input pins are at a relatively high impedance and cross-talk from other signals will seriously affect oscillator stability and accuracy. The crystal's case should also be connected to ground if possible.

If an oscillator module is used, care still needs to be taken when tracking to the mXT1664S. The clock signal should be kept as short as possible, with a solid ground return underneath the clock output.

A.7 Analog I/O

In general, tracking for the analog I/O signals from the mXT1664S device should be kept as short as possible. These normally go to a connector which interfaces directly to the touchscreen.

Ensure that adequate ground-planes are used. An analog ground plane should be used in addition to a digital one. Care should be taken to ensure that both ground planes are kept separate and are connected together only at the point of entry for the power to the PCB. This is usually at the input connector.

A.8 Component Placement

It is important to orient all devices so that the tracking for important signals (such as power and clocks) are kept as short as possible. This simple point is often overlooked when initially planning a PCB layout and can save hours of work at a later stage.

A.9 Digital Signals

In general, when tracking digital signals, it is advisable to avoid sharp directional changes, sensitive signal tracks (such as analog I/O) and any clock or crystal tracking.

A good ground return path for all signals should be provided, where possible, to ensure that there are no discontinuities in the ground return path.

A.10 EMC and Other Observations

The following recommendations are not mandatory, but may help in situations where particularly difficult EMC or other problems are present:

- A small common mode choke is recommended on the differential USB data pair. This should be placed directly at the USB connector, between the connector and the relevant mXT1664S pins. Tracking lengths for the USB data pair should be kept as short as possible.
- Try to keep as many signals as possible on the inside layers of the board. If suitable ground flood fills are used on the top and bottom layers, these will provide a good level of screening for noisy signals, both into and out of the PCB.
- Ensure that the on-board regulators have sufficient tracking around and underneath the devices to act as a heatsink. This heatsink will normally be connected to the 0V or ground supply pin. Increasing the width of the copper tracking to any of the device pins will aid in removing heat. There should be no solder mask over the copper track underneath the body of the regulators.
- Ensure that the decoupling capacitors, especially tantalum, or high capacity ceramic types, have the requisite low ESR, ESL and good stability/temperature properties. Refer to the regulator manufacturer’s datasheet for more information.

Appendix B. Reference Unit and Configuration

B.1 Reference Configuration

This appendix lists the configuration values that were used in the reference unit used to validate the interfaces on the device and to derive performance charts.

Note that only those fields that have a non-default value are listed. All other fields have the default value of zero and are not listed.

See *mXT1664S Protocol Guide* for information about the individual objects and their fields..

Field	Value
Power Configuration T7 – GEN_POWERCONFIG_T7 Instance 0	
IDLEACQINT	255
ACTVACQINT	255
ACTV2IDLETO	5
CFG	3
Acquisition Configuration T8 – GEN_ACQUISITIONCONFIG_T8 Instance 0	
CHRGTIME	120
TCHDRIFT	20
DRIFTST	20
ATCHCALSTHR	1
Multiple Touch Touchscreen T9 – TOUCH_MULTITOUCHSCREEN_T9 Instance 0	
CTRL	139
XSIZE	32
YSIZE	52
BLEN	123
TCHTHR	60
TCHDI	2
ORIENT	5
MOVFILTER	142
NUMTOUCH	16
MRGHYST	20
MRGTHR	20
AMPHYST	10
XRANGE	799
YRANGE	1279
XLOCLIP	7
XHICLIP	7
YLOCLIP	5
YHICLIP	5

Field	Value
XEDGECTRL	148
XEDGEDIST	28
YEDGECTRL	146
YEDGEDIST	20
JUMPLIMIT	15
TCHHYST	25
XPITCH	43
YPITCH	42
NEXTTCHDI	2
Key Array T15 – TOUCH_KEYARRAY_T15 Instance 0	
XORIGIN	8
YORIGIN	51
XSIZE	16
YSIZE	1
AKSCFG	1
BLEN	123
TCHTHR	40
TCHDI	2
Communications Configuration T18 – SPT_COMMSCONFIG_T18 Instance 0	
CTRL	4
Grip Suppression T40 – PROCI_GRIPSUPPRESSION_T40 Instance 0	
XLOGRIP	20
XHIGRIP	20
YLOGRIP	20
YHIGRIP	20
Touch Suppression T42 – PROCI_TOUCHSUPPRESSION_T42 Instance 0	
CTRL	32
APPRTHR	20
SUPEXTTO	20
MAXNUMTCHS	17
SUPDIST	3
Digitizer HID Configuration T43 – SPT_DIGITIZER_T43 Instance 0	
HIDIDLERATE	128
XLENGTH	137
YLENGTH	221
RWKRATE	128
CTE Configuration T46 – SPT_CTECONFIG_T46 Instance 0	

Field	Value
IDLESYNCSPERX	8
ACTVSYNCSPERX	8
XSLEW	1
XVOLTAGE	15
ADCCTRL	15
Stylus T47 – PROCI_STYLUS_T47 Instance 0	
CONTMIN	20
CONTMAX	35
STABILITY	2
MAXTCHAREA	5
AMPLTHR	30
STYSHAPE	1
HOVERSUP	120
CONFTHR	3
SYNCSPERX	16
CFG	12
ACTIVEPWR	32
ACTIVERATIO	230
SlimSensor T56 – PROCI_SHIELDLESS_T56 Instance 0	
CTRL	2
OPTINT	1
INTTIME	48
INTDELAY[0]	19
INTDELAY[1]	20
INTDELAY[2]	20
INTDELAY[3]	20
INTDELAY[4]	21
INTDELAY[5]	21
INTDELAY[6]	21
INTDELAY[7]	21
INTDELAY[8]	21
INTDELAY[9]	21
INTDELAY[10]	21
INTDELAY[11]	22
INTDELAY[12]	22
INTDELAY[13]	22
INTDELAY[14]	22

Field	Value
INTDELAY[15]	22
INTDELAY[16]	22
INTDELAY[17]	22
INTDELAY[18]	22
INTDELAY[19]	22
INTDELAY[20]	22
INTDELAY[21]	21
INTDELAY[22]	20
INTDELAY[23]	20
INTDELAY[24]	20
INTDELAY[25]	20
INTDELAY[26]	21
INTDELAY[27]	20
INTDELAY[28]	20
INTDELAY[29]	20
INTDELAY[30]	20
INTDELAY[31]	19
NCNCL	1
TOUCHBIAS	2
BASESCALE	5
SHIFTLIMIT	5
maXCharger T62 – PROCG_NOISESUPPRESSION_T62 Instance 0	
CTRL	127
CALCFG1	11
CALCFG3	23
RESERVED2	1
FREQ[1]	3
FREQ[2]	9
FREQ[3]	12
FREQ[4]	23
HOPCNT	5
HOPCNTPER	10
HOPEVALTO	5
HOPST	5
NLGAIN	80
MINNLTHR	44
INCNLTHR	26

Field	Value
ADCSPERXTHR	26
NLTHRMARGIN	26
MAXADCSPERX	63
MINGCLIMIT	4
MAXGCLIMIT	64
BLEN[0]	80
TCHTHR[0]	88
TCHDI[0]	2
MOVHYSTN[0]	1
NUMTOUCH[0]	5
TCHHYST[0]	22
NEXTTCHDI[0]	5

Appendix C. Glossary of Terms

Channel

One of the capacitive measurement points at which the sensor controller can detect capacitive change.

Jitter

The peak-to-peak variance in the reported location for an axis when a fixed touch is applied. Typically jitter is random in nature and has a Gaussian [1] distribution, therefore measurement of peak-to-peak jitter must be conducted over some period of time, typically a few seconds. Jitter is typically measured as a percentage of the axis in question.

For example a 100 x 100 mm touchscreen that shows ±0.5% jitter in X and ±1% jitter in Y would show a peak deviation from the average reported coordinate of ±0.5 mm in X and ±1 mm in Y. Note that by defining the jitter relative to the average reported coordinate, the effects of linearity are ignored.

Linearity

The measurement of the peak-to-peak deviation of the reported touch coordinate in one axis relative to the absolute position of touch on that axis. This is often referred to as the nonlinearity. Non-linearities in either X or Y axes manifest themselves as regions where the perceived touch motion along that axis (alone) is not reflected correctly in the reported coordinate giving the sense of moving too fast or too slow. Linearity is measured as a percentage of the axis in question.

For each axis, a plot of the true coordinate versus the reported coordinate should be a perfect straight line at 45°. A non-linearity makes this plot deviate from this ideal line. It is possible to correct modest non-linearities using on-chip linearization tables, but this correction trades linearity for resolution in regions where stronger corrections are needed (because there is a stretching or compressing effect to correct the nonlinearity, so altering the resolution in these regions). Linearity is typically measured using data that has been sufficiently filtered to remove the effects of jitter. For example, a 100 mm slider with a nonlinearity of ±1% reports a position that is, at most, 1 mm away in either direction from the true position.

One-touch Gesture

A touch gesture that consists of a single touch. The combination of the duration of the touch and any change in position (that is, movement) of the touch characterizes a specific gesture. For example, a tap gesture is characterized by a short-duration touch followed by a release, and no significant movement.

Resolution

The measure of the smallest movement on a slider or touchscreen in an axis that causes a change in the reported coordinate for that axis. Resolution is normally expressed in bits and tends to refer to resolution across the whole axis in question. For example, a resolution of 10 bits can resolve a movement of 0.0977 mm on a slider 100 mm long. Jitter in the reported position degrades usable resolution.

[1] Sometimes called Bell-shaped or Normal distribution.

Touchscreen

A two-dimensional arrangement of electrodes whose capacitance changes when touched, allowing the location of touch to be computed in both X and Y axes. The output from the XY computation is a pair of numbers, typically 12-bits each, ranging from 0 to 4095, representing the extents of the touchscreen active region.

Two-touch Gesture

A touch gesture that consists of two simultaneous touches. The change in position of the two touches in relation to each other characterizes a specific gesture. For example, a pinch gesture is characterized by two long-duration touches that have a decreasing distance between them (that is, they are moving closer together).

Appendix D. QMatrix Primer

D.1 Acquisition Technique

QMatrix capacitive acquisition uses a series of pulses to deposit charge into a sampling capacitor, Cs. The pulses are driven on X lines from the controller. The rising edge of the pulse causes current to flow in the mutual capacitance, Cx, formed between the X line and a neighboring receiver electrode or Y line. While one X line is being pulsed, all others are grounded. This leads to excellent isolation of the particular mutual capacitances being measured [1], a feature that makes for good inherent touchscreen performance.

After a fixed number of pulses (known as the burst length) the sampling capacitor's voltage is measured to determine how much charge has accumulated. This charge is directly proportional to Cx and therefore changes if Cx [2] changes. The transmit-receive charge transfer process between the X lines and Y lines causes an electric field to form that loops from X to Y. The field itself emanates from X and terminates on Y. If the X and Y electrodes are fixed directly [3] to a dielectric material like plastic or glass, then this field tends to channel through the dielectric with very little leakage of the field out into free-space (that is, above the panel). Some proportion of the field does escape the surface of the dielectric, however, and so can be influenced during a touch.

When a finger is placed in close proximity (a few millimeters) or directly onto the dielectric's surface, some of this stray field and some of the field that would otherwise have propagated via the dielectric and terminated onto the Y electrode, is diverted into the finger and is conducted back to the controller chip via the human body rather than via the Y line.

This means that less charge is accumulated in Cs, and hence the terminal voltage present on Cs, after all the charge transfer pulses are complete, becomes less. In this way, the controller can measure changes in Cx during touch. This means that the measured capacitance Cx goes down during touch, because the coupled field is partly diverted by the touching object.

The spatial separation between the X and Y electrodes is significant to make the electric field to propagate well in relation to the thickness of the dielectric panel.

D.2 Moisture Resistance

A useful side effect of the QMatrix acquisition method is that placing a floating conductive element between the X and Y lines tends to increase the field coupling and so increases the capacitance Cx. This is the opposite change direction to normal touch, and so can be quite easily ignored or compensated for by the controller. An example of such floating conductive elements is the water droplets caused by condensation.

As a result, QMatrix-based touchscreens tend not to go into false detect when they are covered in small non-coalesced water droplets. Once the droplets start to merge, however, they can become large enough to bridge the field across to nearby ground return paths (for example, other X lines not currently driven, or ground paths in mechanical chassis components). When this happens, the screen's behavior can become erratic.

[1] A common problem with other types of capacitive acquisition technique when used for touchscreens, is that this isolation is not so pronounced. This means that when touching one region of the screen, the capacitive signals also tend to change slightly in nearby channels too, causing small but often significant errors in the reported touch position.

[2] To a first approximation.

[3] Air gaps in front of QMatrix sensors massively reduce this field propagation and kill sensitivity. Normal optically clear adhesives work well to attach QMatrix touchscreens to their dielectric front panel.

There are some measures used in these controllers to help with this situation, but in general there comes a point where the screen is so contaminated by moisture that false detections become inevitable. It should also be noted that uniform condensation soon becomes non-uniform once a finger has spread it around. Finger grease renders the water highly conductive, making the situation worse overall.

In general, QMatrix has industry-leading moisture tolerance but there comes a point when even the best capacitive touchscreen suffers due to moisture on the dielectric surface.

D.3 Interference Sources

D.3.1 Power Supply

The device can tolerate short-term power supply fluctuations. If the power supply fluctuates slowly with temperature, the device tracks and compensate for these changes automatically with only minor changes in sensitivity. If the supply voltage drifts or shifts quickly, the drift compensation mechanism is not able to keep up, causing sensitivity anomalies or false detections.

The device itself uses the AVdd power supply as an analog reference, so the power should be very clean and come from a separate regulator. A standard inexpensive Low Dropout (LDO) type regulator should be used that is not also used to power other loads, such as LEDs, relays, or other high current devices. Load shifts on the output of the LDO can cause AVdd to fluctuate enough to cause false detection or sensitivity shifts. The digital Vdd supply is far more tolerant to noise.

⚠ **CAUTION:** A regulator IC shared with other logic can result in erratic operation and is not advised.

Noise on AVdd can appear directly in the measurement results. Vdd should be checked to ensure that it stays within specification in terms of noise, across a whole range of product operating conditions.

Ceramic bypass capacitors on AVdd and Vdd, placed very close (<5 mm) to the chip are recommended. A bulk capacitor of at least 1 µF and a higher frequency capacitor of around 10 nF to 100 nF in parallel are recommended; both must be X7R or X5R dielectric capacitors.

D.3.2 Other Noise Sources

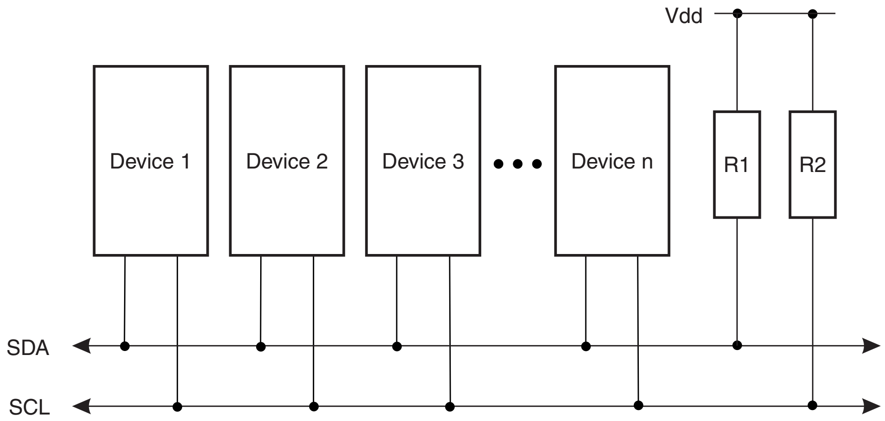
Refer to QTAN0079, *Buttons, Sliders and Wheels Sensor Design Guide*, for information (downloadable from the Touch Technology area of the Atmel website).

Appendix E. I²C Basics (I²C-compatible Operation)

E.1 Interface Bus

The device communicates with the host over an I²C-compatible bus. The following sections give an overview of the bus; more detailed information is available from www.i2C-bus.org. Devices are connected to the I²C-compatible bus as shown in Figure E-1. Both bus lines are connected to Vdd via pull-up resistors. The bus drivers of all I²C-compatible devices must be open-drain type. This implements a wired AND function that allows any and all devices to drive the bus, one at a time. A low level on the bus is generated when a device outputs a zero.

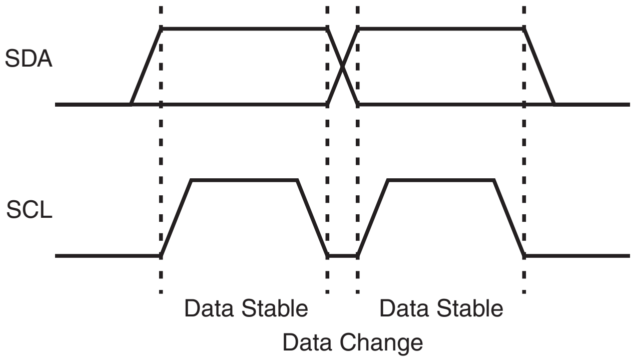
Figure E-1. I²C-compatible Interface Bus



E.2 Transferring Data Bits

Each data bit transferred on the bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high; the only exception to this rule is for generating START and STOP conditions.

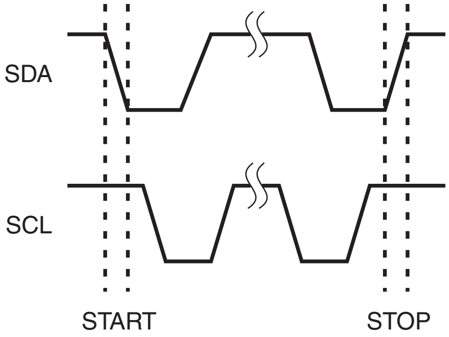
Figure E-2. Data Transfer



E.3 START and STOP Conditions

The host initiates and terminates a data transmission. The transmission is initiated when the host issues a START condition on the bus, and is terminated when the host issues a STOP condition. Between the START and STOP conditions, the bus is considered busy. As shown in Figure E-3, START and STOP conditions are signaled by changing the level of the SDA line when the SCL line is high.

Figure E-3. START and STOP Conditions

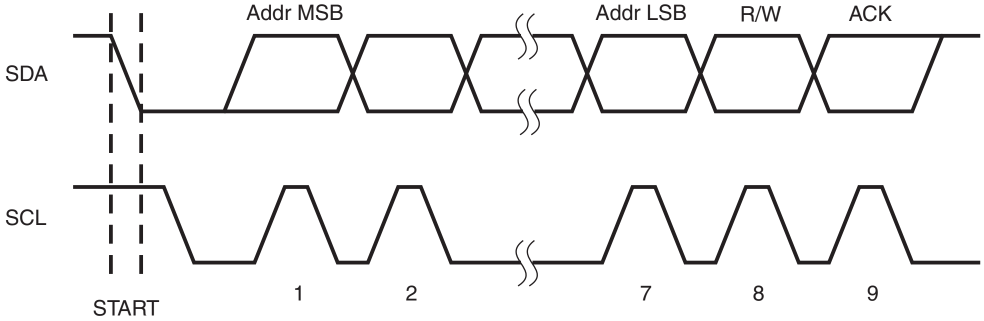


E.4 Address Byte Format

All address bytes are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is performed, otherwise a write operation is performed. When the device recognizes that it is being addressed, it will acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. An address byte consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The most significant bit of the address byte is transmitted first. The address sent by the host must be consistent with that selected with the option jumpers.

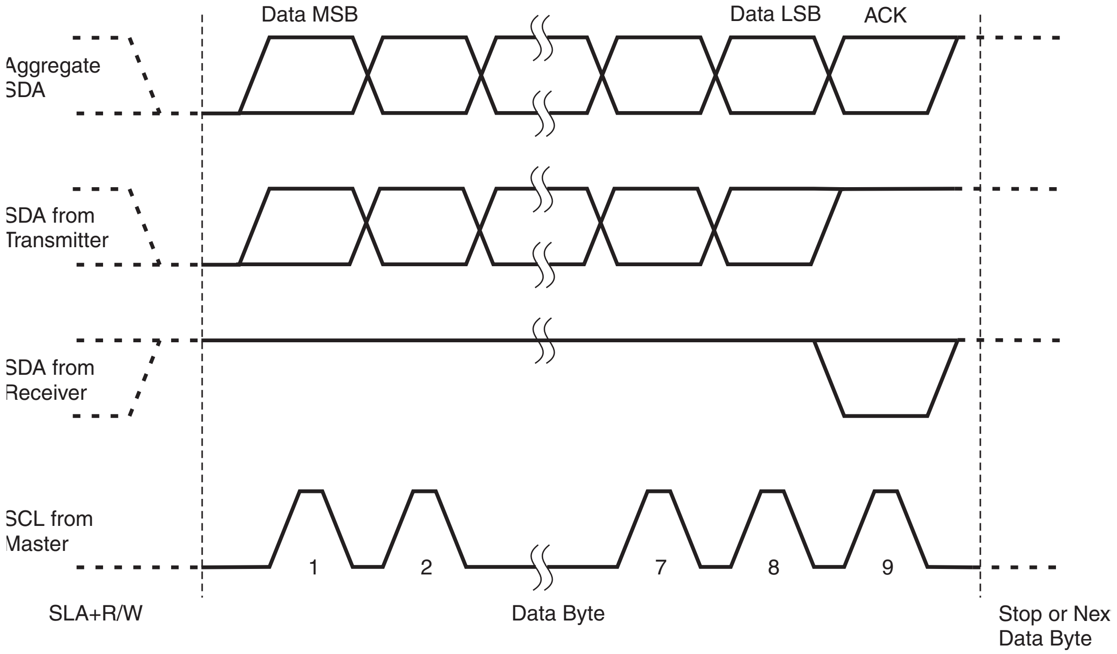
Figure E-4. Address Byte Format



E.5 Data Byte Format

All data bytes are 9 bits long, consisting of 8 data bits and an acknowledge bit. During a data transfer, the host generates the clock and the START and STOP conditions, while the receiver is responsible for acknowledging the reception. An acknowledge (ACK) is signaled by the receiver pulling the SDA line low during the ninth SCL cycle. If the receiver leaves the SDA line high, a NACK is signaled.

Figure E-5. Data Byte Format



E.6 Combining Address and Data Bytes into a Transmission

A transmission consists of a START condition, an SLA+R/W, one or more data bytes and a STOP condition. The wired “ANDing” of the SCL line is used to implement handshaking between the host and the device. The device extends the SCL low period by pulling the SCL line low whenever it needs extra time for processing between the data transmissions.

Note: Each write or read cycle must end with a stop condition. The device may not respond correctly if a cycle is terminated by a new start condition.

Figure E-6 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP.

Figure E-6. Byte Transmission

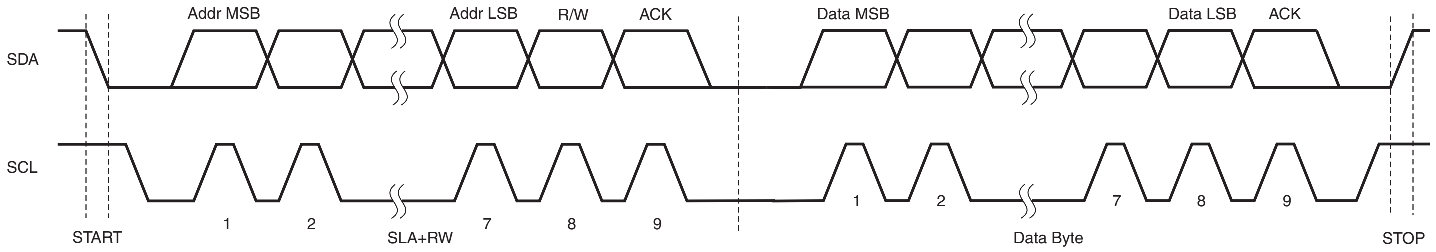


Table of Contents

Revision History

Revision Number	History
Revision AX – December 2011	• Initial release for firmware version 0.5
Revision BX – January 2012	• Minor updates for firmware version 0.5
Revision CX – January 2012	• Section 2.2 Pin listing correction (A13, E13 and H13).
Revision DX – March 2012	• Front page: Added support for maXStylus information and additional information for I^{2}C-compatible High speed bus. • Section 4.1: Added maximum software reset time. • Section 4.5.4: Updated name for T56 object. • Section 4.6.1: Added XVdd supply information including power sequencing description. • Section 5.2.1: Corrected ADDSEL line levels to select the I^{2}C-compatible mode. • Section 7.1: Added mode selection information for HID-I^{2}C. • Section 8.5: Added maXCharger T62 information. • Section 9.3.3: Updated slew rate value. • Section 9.5: Updated supply current values. • Section 9.6: Added power consumption charts. • Section 9.8: Added refresh rate vs number of touches chart. • Section 9.10: Added required pull-up resistance values for different I^{2}C-compatible modes. • Section A.5: Added XVdd specific regulator information with an example circuit.
Revision EX – May 2012	• Minor updates for firmware version 1.0 • Section 2.2 on page 5: Updated description (H6 and M11). • Section 9.2 on page 57: Added recommended Cx values for different XVdd ranges. • Section 9.6 on page 60: Updated power consumption values. • Section 9.8 on page 65: Updated values. • Section 9.9 on page 66: Updated values. • Section 9.15.2 on page 68: Added junction temperature information.
Revision FX – May 2012	• Minor updates for firmware version 1.0 • Section 9.7 on page 65: Updated Timing Specifications. • Section 9.19 on page 70: Added QS numbers.
Revision GX – June 2012	• Section 9.2 on page 57 Updated max value for Cx.
Revision HX – July 2012	• Minor updates to aid clarity.
Revision IX – June 2013	• Updated recommended max voltage for XVdd • Other minor updates to aid clarity.
Revision JX – June 2013	• Updated Vdd specifications

Headquarters

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 436-4314

International

Atmel Asia
Unit 01-05 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong
Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Touch Technology Division
1560 Parkway
Solent Business Park
Whiteley
Fareham
Hampshire
PO15 7AG
UNITED KINGDOM
Tel: (+44) (0) 844 894 1920
Fax: (+44) (0) 1489 557 066

Atmel Munich GmbH
Business Campus
Parkring 4
D- 85748 Garching b.
MUNICH
Tel: (+49) 89-31970-111
Fax: (+49) 89-3194621

Atmel Japan
16F Shin-Osaki Kangyo Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
Tel: (+81) (3) 6417-0300
Fax: (+81) (3) 6417-0370

Product Contact

Web Site
www.atmel.com

Technical Support
touch@atmel.com

Sales Contact
www.atmel.com/contacts

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.