Keywords: touch screen driver example code

APPLICATION NOTE 4125

Getting Started with the MAX1233/MAX1234 Touch-Screen Controllers

Nov 14, 2007

Abstract: This application note shows how to exercise the features of the MAX1233/MAX1234 touch-screen controllers. A simplified console menu system is provided that allows direct, low-level access to the MAX1233/MAX1234 device registers. Each register read or write operation uses 32 SPI™ clock cycles. The software uses short, mnemonic names for each register. When used with the MAX1234 evaluation kit (EV Kit) board and the MINIQUSB+ command module, this software allows maximum low-level control. Full source code is provided in the accompanying zip file.

MAX1233 operation is identical to MAX1234 operation, except that the MAX1233 operates from a 3.3V supply voltage instead of 5.0V. Jumper JU1 on the MAX1234 EV Kit board emulates the MAX1233 by operating a MAX1234 at 3.3V.

Note: The suffix "-bar" (e.g. CS-bar) indicates the active-low functionality of the CS, PENIRQ, KEYIRQ, and BUSY pins.

Table of Contents

1.1) Required Hardware

- Maxim MAX1234 EV Kit (MAX1234EVKIT)
- Maxim MINIQUSB+ (includes USB A-B cable and MINIQUSB-X+ expander board)
- Windows® 2000/XP PC with USB
- Four-wire resistive touch screen (such as a replacement PDA digitizer/glasstop)
- Optional: DMM for measuring the DAC output voltage
- Optional: Voltage sources to drive the AUX and BAT inputs
- Optional: Oscilloscope to observe autoscan interrupt pulses on the PENIRQ-bar and KEYIRQ-bar pins

1.2) MINIQUSB+ Firmware Update Note

The MAX1233/MAX1234 require the CS-bar pin to deassert high before the end of the first conversion; otherwise, the ADC will be unable to store conversion results. Prior to use with this application note, the standard MINIQUSB+ module firmware must be updated to allow the SPI interface's CS-bar pin to deassert within 1.4µs of the 32nd SCLK. At 2MHz, the 32-bit auto-CS-bar controlled mode holds CS-bar low for 21.70µs. The MINIQUSB+ firmware in the MAXQ2000 microcontroller's nonvolatile flash memory only needs to be updated once. This new firmware is backwards-compatible with the standard 01.05.39 baseline firmware.

In addition to improving the SPI interface's CS-bar timing, the firmware upgrade includes an interrupt-driven pulse accumulator to allow verifying that PENIRQ-bar and KEYIRQ-bar have issued their self-clearing interrupt pulses when the MAX1233/MAX1234 are configured for autoscan modes. The duration of PENIRQ-bar depends on the configured ADC conversion rate, and the duration of KEYIRQ-bar depends on the configured switch debounce time.

1.3) Setup

Download and unzip the application note files (ZIP, 2.4MB).

Assemble the hardware according to **Figure 1**.

1. Connect wires from MAX1234 EV Kit connector J1 to MINIQUSB-X+ expander board (included with MINIQUSB+), in accordance with **Table 1**. As an alternative to soldering wires to the MAX1234 EV Kit, a 3M® intra-connector 922576-40 can be plugged into J1, providing convenient connection points. Leave terminal block TB1 unconnected.

   Table 1. Connection Between the MAX1234 EV Kit and MINIQUSB+ Board Set

MAX1234 Signal	MAX1234 EV Kit	MINIQUSB-X+	MINIQUSB Signal
GND	J1-1	H2-8	GND
V_{CC}	J1-7	H2-1	3.3V supply from MINIQUSB+
BUSY-Bar	J1-27	H2-7	GPIO-K7 (MAXQ2000-INT2)
PENIRQ-Bar	J1-29	H1-3	GPIO-K6 (MAXQ2000-INT1)
KEYIRQ-Bar	J1-31	H1-8	GPIO-K5 (MAXQ2000-INT0)
DOUT	J1-35*	H2-2	MISO (SPI master in, slave out)
DIN	J1-36*	H2-5	MOSI (SPI master out, slave in)
SCLK	J1-37*	H2-3	SCLK (SPI clock)
CS-Bar	J1-38	H2-4	CS-bar (SPI chip select)
USB+5V	J1-5	J4-7	USB+5V supply from PC

* Note: Digital inputs to the MAX1234 EV Kit must be driven through connector J1, they cannot be driven directly to the test points surrounding U1. The on-board MAX1841 level translators must be used to drive the MAX1234 EV Kit digital signals.

2. Plug the MINIQUSB+ on top of its expander board.
3. Connect the MINIQUSB+ to the PC's USB port. If this is the first time a MINIQUSB+ has been attached to the PC, the plug-and-play wizard will appear. Guide Windows to the installed location of the device driver (which is included in the accompanying zip file).
4. Update the MINIQUSB+ firmware by launching the firmware updater batch file FWUPDATE.BAT.
5. After the firmware update is complete, disconnect the MINIQUSB+ from the PC's USB port.

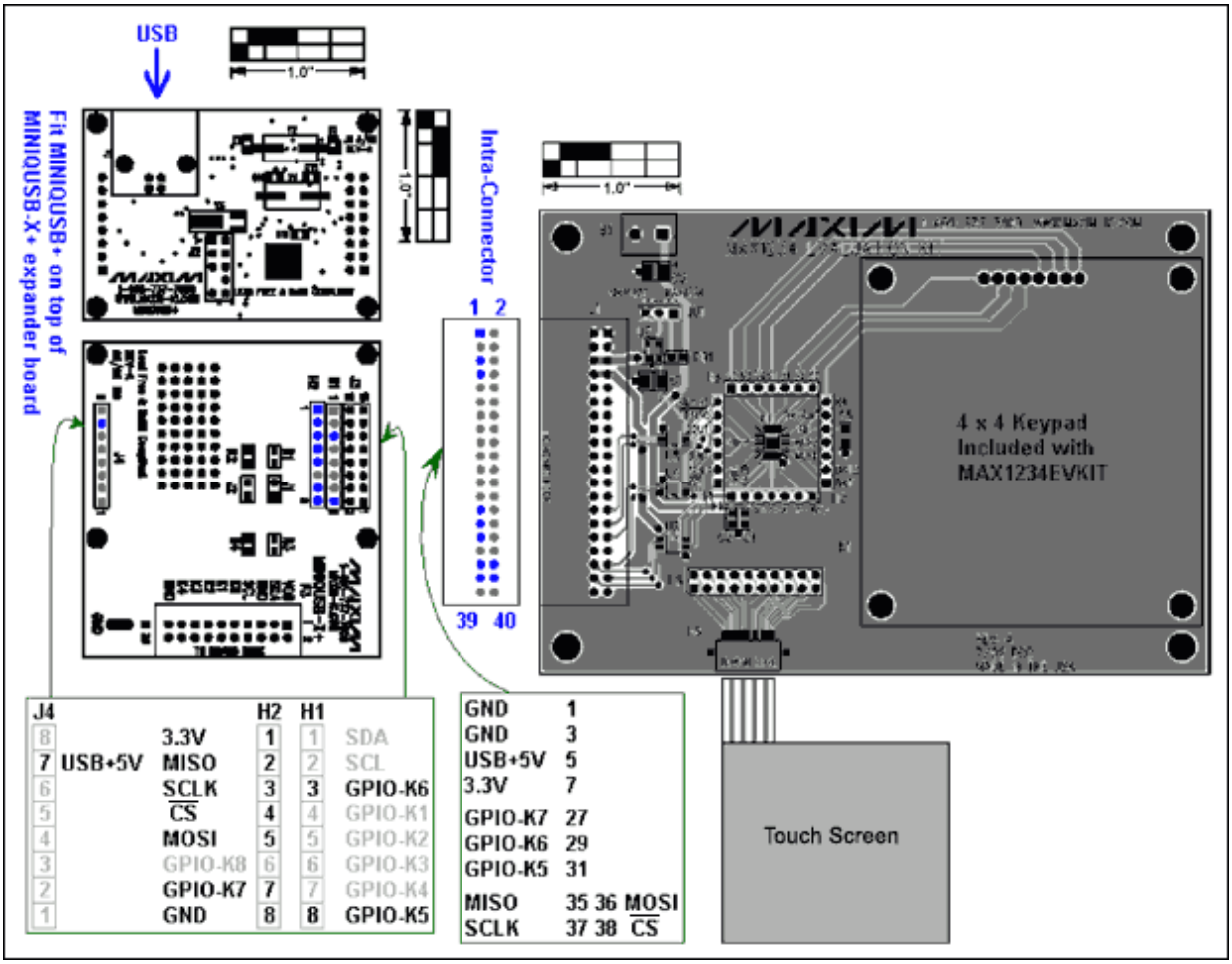

Figure 1. Hardware configuration. (The touch screen will be connected in a later section.)

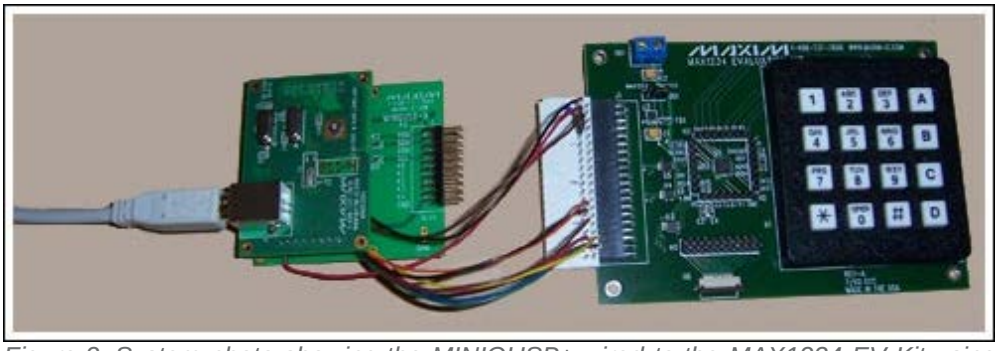

Figure 2. System photo showing the MINIQUSB+ wired to the MAX1234 EV Kit using a 3M intra-connector.

1.4) Procedure

1. Set MAX1234 EV Kit jumper JU1 to the "MAX1234" position.
2. Connect the MINIQUSB+ to the PC's USB port. Verify that the DACOUT voltage = mid-scale (2.2V).
3. Start the DEMO1234.EXE program. A console will appear on the screen.
4. Enter the following series of commands at the console.

Table 2. Connect and Verify Command Sequence

DEMO1234 Command*	Expected Program Output	SPI data in	Verification**
C	Board connected. Got board banner: Maxim MINIQUSB V01.05.41 > **Firmware version is OK.** (configured for SPI auto-CS 4-byte mode) (SCLK=2MHz) ...		
T W DD FF	Write_Register(regAddr=0x000b wr_DAC_data, data=0x00ff {(no bits defined for this register)}) **result = 1**	0x000b 0x00ff	DACOUT = full-scale (4.5V)
T R DD	Read_Register(regAddr=0x800b wr_DAC_data) result = 1, **buffer = 0x00ff** = 255 {(no bits defined for this register)}	0x800b 0x0000	Data buffer = 0x00ff
T W DD 80	Write_Register(regAddr=0x000b wr_DAC_data, data=0x0080 {(no bits defined for this register)}) result = 1	0x000b 0x0080	DACOUT = mid-scale (2.2V)
T R DD	Read_Register(regAddr=0x800b wr_DAC_data) result = 1, **buffer = 0x0080** = 128 {(no bits defined for this register)}	0x800b 0x0000	data buffer = 0x0080

* **DEMO1234 Command** column lists the command to type into the DEMO1234.exe program.
** **Verification** column lists any physical tests that can be performed to verify that the command was performed.

1.5) Explanation of SPI data in Example Format

The **SPI data in** column lists the SPI data driven into the MAX1233/MAX1234 DIN pin, in hexadecimal, most-significant-byte first. For example, the SPI data in the sequence 0x000b 0x00ff means the 32-bit sequence clocked into DIN is 0000 0000 0000 1011 0000 0000 1111 1111. The first bit is 0 for register write operations and 1 for register read operations.

Register write operations are 32-bit SPI transfers of the form 0000 0000 a7-a0 d15-d0.
Register read operations are 32-bit SPI transfers of the form 1000 0000 a7-a0 0000 0000, with received data clocked in from DOUT during the final 16 bits.

2) Analog I/O Examples

The following examples show how to use the DEMO1234.EXE program to control the DAC output, configure the reference voltage, measure the AUX1/AUX2/BAT1/BAT2 voltage inputs, and measure the internal MAX1234 temperature.

2.1) Controlling DAC Output Voltage

The DAC is controlled by two registers. Write the DAC Data register to set the output voltage. Write the DAC Control register to shut down or power the DAC. The default power-on state is that the DAC is powered and the DAC output is mid-scale. DAC full-scale voltage is typically 90% of AV_{DD} (85% min, 95% max).

For AV_{DD} = 3.3V ±5%, the DACOUT full-scale range is between 2.65V and 3.27V, typically 2.96V.
For AV_{DD} = 5.0V ±5%, the DACOUT full-scale range is between 4.02V and 4.97V, typically 4.48V.

Table 3. DAC Output Commands

DEMO1234 Command	Action	SPI data in	MAX1233 (3.3V)	MAX1234 (5.0V)
T W DD FF	DACOUT = full-scale	0x000b 0x00ff	DACOUT = 2.96V	DACOUT = 4.48V
T W DD 00	DACOUT = 0V	0x000b 0x0000	DACOUT = 0.0V	DACOUT = 0.0V
T W DD 80	DACOUT = mid-scale	0x000b 0x0080	DACOUT = 1.485V	DACOUT = 2.25V
T W DC 8000	Disable DAC	0x0042 0x8000	DACOUT = 0.0V	DACOUT = 0.0V
T W DC 0	Enable DAC	0x0042 0x0000	DACOUT = 1.485V	DACOUT = 2.25V

2.2) Selecting ADC Reference Power Mode

The ADC requires a reference voltage. For typical embedded-systems operation, the default setting is fine. In auto-power-up mode (ADC3210 = 0000, RES10 = 00), the MAX1233/MAX1234 provides its own internal reference voltage. This internal reference automatically powers up prior to each measurement and then powers down after measurement is complete.

For initial diagnostics, the always-powered-up mode (ADC3210 = 0000, RES10 = 01) allows external verification of the reference voltage using a handheld DVM.

The ADC reference power mode is set by writing into the ADC control register (0x40) with the ADC Scan Select bits set to 0000. The RES1/RES0 bits select the reference power mode, and the reference control bit RFV selects either the internal 1.0V or 2.5V reference (refer to Table 13 in the MAX1233/MAX1234 data sheet).

ADC control word: x x 0 0 0 0 RES1 RES0 x x x x x x RFV

Table 4. Internal Reference Commands

DEMO1234 Command	Action	SPI data in	Verification
T W AC 0100	Internal 1V reference always powered; write ADC control word with ADC3210 = 0000, RES10 = 01, RFV = 0	0x0040 0x0100	Voltage at pin 12 REF is between 0.98V and 1.02V
T W AC 0101	Internal 2.5V reference always powered; write ADC control word with ADC3210 = 0000, RES10 = 01, RFV = 1	0x0040 0x0101	Voltage at pin 12 REF is between 2.47V and 2.53V
T W AC 0001	Internal 2.5V reference powered when needed; write ADC control word with ADC3210 = 0000, RES10 = 00, RFV = 1	0x0040 0x0001	Voltage at pin 12 REF will be powered only briefly as necessary

Table 5. External Reference Command

DEMO1234 Command	Action	SPI data in
	External reference must be provided;	

T W AC 0300	ADC_control_wr_demand_scan:(write)demand scan ADC_control_AD0000:configure reference ADC_control_RES11:external reference	0x0040 0x0300

2.3) Measuring External Voltage Inputs AUX1, AUX2

Table 6. ADC Measurement Command Sequences

DEMO1234 Command	Action (Triggered by A/D3210 Bits)	SPI data in
T M8	Measure AUX1 with 12-bit resolution and 3.5µs conversion rate	0x0040 0x2301 0x8007 0x0000
T W AC 2301	Trigger ADC scan of AUX1; ADC control word 0x2301 means: ADC_control_wr_demand_scan ADC_control_AD1000 /* measure AUX1 */ ADC_control_RES11 /* 12-bit resolution */ ADC_control_AVG00 /* no averaging */ ADC_control_CNR00 /* conversion rate 3.5µs */ ADC_control_RFV /* RFV=1: VREF=2.5V */	0x0040 0x2301
T R A1	Read AUX1 result AUX1_code	0x8007 0x0000
T M9	Measure AUX2 with 12-bit resolution and 3.5µs conversion rate	0x0040 0x2701 0x8008 0x0000

2.4) Translating AUX1 and AUX2 Conversion Results to a Physical Value

The following snippet of C/C++ pseudocode summarizes how the AUX1 and AUX2 conversion results are interpreted by the DEMO1234 program.

```
    /* ADC control resolution value selects num_codes 4096 (12-bit), 1024 (10-
bit), or 256 (8-bit) */
    int num_codes = 4096; /* ADC_control_RES11: 12-bit resolution */

    /* Voltage that corresponds to the full-scale ADC code; may be internal 1V
or 2.5V ref, or ext ref. */
    double ADC_fullscale_voltage = 2.5; /* ADC_control_RFV=1: VREF=2.5V.
RFV=0: VREF=1.0V. */

    /* AUX1_code is the 16-bit result read by SPI command 0x8007 */
    double AUX1_Voltage = (AUX1_code * ADC_fullscale_voltage) / num_codes;

    /* AUX2_code is the 16-bit result read by SPI command 0x8008 */
    double AUX2_Voltage = (AUX2_code * ADC_fullscale_voltage) / num_codes;
```

2.5) Measuring External Voltage Inputs BAT1, BAT2

Table 7. ADC Measurement Command Sequences

DEMO1234 Command	Action (Triggered by A/D3210 Bits)	SPI data in
T M6	Measure BAT1 with 12-bit resolution and 3.5µs conversion rate	0x0040 0x1b01 0x8005 0x0000
	Trigger ADC scan of BAT1;	

T W AC 1b01	ADC control word 0x1b01 means: ADC_control_wr_demand_scan ADC_control_AD0110 /* measure BAT1 */ ADC_control_RES11 /* 12-bit resolution */ ADC_control_AVG00 /* no averaging */ ADC_control_CNR00 /* conversion rate 3.5µs */ ADC_control_RFV /* RFV=1: VREF=2.5V */	0x0040 0x1b01
T R B1	Read BAT1 result BAT1_code	0x8005 0x0000
T W AC 1b21	Trigger ADC scan of BAT1; ADC control word 0x1b21 means: ADC_control_wr_demand_scan ADC_control_AD0110 /* measure BAT1 */ ADC_control_RES11 /* 12-bit resolution */ ADC_control_AVG00 /* no averaging */ ADC_control_CNR10 /* conversion rate 10µs */ ADC_control_RFV /* RFV=1: VREF=2.5V */	0x0040 0x1b21
T R B1	Read BAT1 result BAT1_code	0x8005 0x0000
T M7	Measure BAT2 with 12-bit resolution and 3.5µs conversion rate	0x0040 0x1f01 0x8006 0x0000

2.6) Translating BAT1 and BAT2 Conversion Results to a Physical Value

The following snippet of C/C++ pseudocode summarizes how the BAT1 and BAT2 conversion results are interpreted by the DEMO1234 program. Note: BAT1 and BAT2 measure through a 4:1 input divider.

```
    /* ADC control resolution value selects num_codes 4096 (12-bit), 1024 (10-
bit), or 256 (8-bit) */
    int num_codes = 4096; /* ADC_control_RES11: 12-bit resolution */

    /* Voltage that corresponds to the full-scale ADC code; may be internal 1V
or 2.5V ref, or ext ref. */
    double ADC_fullscale_voltage = 2.5; /* ADC_control_RFV=1: VREF=2.5V.
RFV=0: VREF=1.0V. */

    /* Note: BAT1 and BAT2 measure through a 4:1 input divider. */

    /* BAT1_code is the 16-bit result read by SPI command 0x8005 */
    double BAT1_Voltage = 4 * (BAT1_code * ADC_fullscale_voltage) / num_codes;

    /* BAT2_code is the 16-bit result read by SPI command 0x8006 */
    double BAT2_Voltage = 4 * (BAT2_code * ADC_fullscale_voltage) / num_codes;
```

2.7) Measuring Internal Temperature TEMP1, TEMP2

Table 8. ADC Measurement Command Sequences

DEMO1234 Command	Action (Triggered by A/D3210 Bits)	SPI data in
T MA	Measure TEMP1 with 12-bit resolution and 3.5µs conversion rate	0x0040 0x2b01 0x8009 0x0000
	Trigger ADC scan of TEMP1;	

T W AC 2b01	ADC control word 0x2b01 means: ADC_control_wr_demand_scan ADC_control_ AD1010 /* measure TEMP1 */ ADC_control_RES11 /* 12-bit resolution */ ADC_control_AVG00 /* no averaging */ ADC_control_CNR00 /* conversion rate 3.5µs */ ADC_control_RFV /* RFV=1: VREF=2.5V */	0x0040 0x2b01
T R T1	Read TEMP1 result TEMP1 _code	0x8009 0x0000
T MC	Measure TEMP1, TEMP2 with 12-bit resolution and 3.5µs conversion rate	0x0040 0x3301 0x8009 0x0000 0x800a 0x0000
T W AC 3301	Trigger ADC scan of TEMP1 and TEMP2; ADC control word 0x3301 means: ADC_control_wr_demand_scan ADC_control_ AD1100 /* measure TEMP1,TEMP2 */ ADC_control_RES11 /* 12-bit resolution */ ADC_control_AVG00 /* no averaging */ ADC_control_CNR00 /* conversion rate 3.5µs */ ADC_control_RFV /* RFV=1: VREF=2.5V */	0x0040 0x3301
T R T1	Read TEMP1 result TEMP1 _code	0x8009 0x0000
T R T2	Read TEMP2 result TEMP2 _code	0x800a 0x0000

2.8) Translating TEMP1 Conversion Results to a Physical Value

The following snippet of C/C++ pseudocode summarizes how the TEMP1 conversion results are interpreted by the DEMO1234 program.

```
    /* ADC control resolution value selects num_codes 4096 (12-bit), 1024 (10-
bit), or 256 (8-bit) */
    int num_codes = 4096; /* ADC_control_RES11: 12-bit resolution */

    /* Voltage that corresponds to the full-scale ADC code; may be internal 1V
or 2.5V ref, or ext ref. */
    double ADC_fullscale_voltage = 2.5; /* ADC_control_RFV=1: VREF=2.5V.
RFV=0: VREF=1.0V. */

    /* TEMP1_code is the 16-bit result read by SPI command 0x8009 */
    double TEMP1_Voltage = (TEMP1_code * ADC_fullscale_voltage) / num_codes;

    /* Calibration values */
    const double Temp1V_Room = 0.590; // temp1 voltage at room temperature 25C
    const double Temp1K_Room = 298.15;  // Room temperature Kelvins
(298.15K=25C)
    const double Temp1V_Per_K = -0.002; // TempCo -2mV per degree C

    /* Convert to absolute temperature */
    double Kelvin = (TEMP1_Voltage - Temp1V_Room) / Temp1V_Per_K + Temp1K_Room;

    /* Optional conversion to commonly used temperature units */
```

```
    double Centigrade = Kelvin - 273.15;
    double Fahrenheit = (Centigrade * 9.0 / 5.0) + 32;
```

2.9) Translating TEMP1 and TEMP2 Conversion Results to a Physical Value

The following snippet of C/C++ pseudocode summarizes how the TEMP1 and TEMP2 conversion results are interpreted by the DEMO1234 program. TEMP2 is only meaningful when compared to TEMP1.

```
    /* ADC control resolution value selects num_codes 4096 (12-bit), 1024 (10-
bit), or 256 (8-bit) */
    int num_codes = 4096; /* ADC_control_RES11: 12-bit resolution */

    /* Voltage that corresponds to the full-scale ADC code; may be internal 1V
or 2.5V ref, or ext ref. */
    double ADC_fullscale_voltage = 2.5; /* ADC_control_RFV=1: VREF=2.5V.
RFV=0: VREF=1.0V. */

    /* TEMP1_code is the 16-bit result read by SPI command 0x8009 */
    double TEMP1_Voltage = (TEMP1_code * ADC_fullscale_voltage) / num_codes;

    /* TEMP2_code is the 16-bit result read by SPI command 0x800a */
    double TEMP2_Voltage = (TEMP2_code * ADC_fullscale_voltage) / num_codes;

    /* Calibration values */
    const double K_Per_Temp21_Delta_V = 2680.0; // nominal 2680 5/27/2002

    /* Convert to absolute temperature */
    double Kelvin = (TEMP2_Voltage - TEMP1_Voltage) * K_Per_Temp21_Delta_V;

    /* Optional conversion to commonly used temperature units */
    double Centigrade = Kelvin - 273.15;
    double Fahrenheit = (Centigrade * 9.0 / 5.0) + 32;
```

2.10) Measuring External Voltage Inputs AUX1, AUX2, BAT1, BAT2, and Temperature

Table 9. ADC Measurement Command Sequences

DEMO1234 Command	Action (Triggered by A/D3210 Bits)	SPI data in
		0x0040 0x2f01
		0x8005 0x0000
		0x8006 0x0000
T MB	Measure BAT1/4, BAT2/4, AUX1, AUX2, TEMP1, TEMP2 with 12-bit resolution and 3.5µs conversion rate	0x8007 0x0000
		0x8008 0x0000
		0x8009 0x0000

		0x800a 0x0000
T W AC 2f01	Trigger ADC scan of BAT1-2, AUX1-2, TEMP1-2; ADC control word 0x2f01 means: ADC_control_wr_demand_scan ADC_control_ AD1011 /* measure AUX1 etc. */ ADC_control_RES11 /* 12-bit resolution */ ADC_control_AVG00 /* no averaging */ ADC_control_CNR00 /* conversion rate 3.5µs */ ADC_control_RFV /* RFV=1: VREF=2.5V */	0x0040 0x2f01
T R B1	Read BAT1 result BAT1 _code	0x8005 0x0000
T R B2	Read BAT2 result BAT2_code	0x8006 0x0000
T R A1	Read AUX1 result AUX1 _code	0x8007 0x0000
T R A2	Read AUX2 result AUX2 _code	0x8008 0x0000
T R T1	Read TEMP1 result TEMP1 _code	0x8009 0x0000
T R T2	Read TEMP2 result TEMP2 _code	0x800a 0x0000

3) Touch-Screen Examples

The following examples show how to use the DEMO1234.EXE program to acquire touch-screen data.

3.1) Obtaining a Low-Cost, Off-the-Shelf Touch Screen

Run an internet search for the term "PDA Digitizer/Glasstop" for suitable replacement touch screens. Prices range from $50 to $10 for the clear touch-sensitive glass, depending on the model and whether the glass is affixed to a complete display.

3.2) Connecting the Touch Screen to the EV Kit

The MAX1234 EV Kit provides breakout headers H5/H6 to connect to flex cable 10mm or less in width. The H6 connector's pitch is 0.5mm, which may be finer than the actual touch-screen flex cable pitch. Simply fit the flex cable into H6, secure the latch, and choose pins on H5 that are approximately centered on each of the four flex cable traces. Jumper wires from H5 to the labeled U1 test points for X+, Y+, X-, and Y-.

3.3) Verifying Touch-Screen Connections

When connecting a touch screen for the first time, use the following procedure to verify that the X and Y connections are correct. Several permutations of touch-screen connections are possible—most will not work right. In these examples we assume X- = left, X+ = right, Y- = top, Y+ = bottom.

Table 10. Touch-Screen Physical Connection Verification Command Sequence

DEMO1234 Command	Action	SPI data in	Verification
	Connect DVM to X+/GND		
T MD	No measurement; drive Y+,Y-	0x0040 0x3701	
	Touch top left		X+ = approx. Y-

	Touch top right		X+ = approx. Y-
	Touch bottom left		X+ = approx. Y+
	Touch bottom right		X+ = approx. Y+
	Connect DVM to Y+/GND		
T ME	No measurement; drive X+,X-	0x0040 0x3b01	
	Touch top left		Y+ = approx. X-
	Touch top right		Y+ = approx. X+
	Touch bottom left		Y+ = approx. X-
	Touch bottom right		Y+ = approx. X+

Table 11. Correcting Touch-Screen Connection Problems

Symptom	Correction
Touch coordinates are mirrored top-to-bottom	Swap the Y+ and Y- connections
Touch coordinates are mirrored left-to-right	Swap the X+ and X- connections
Touch coordinates are rotated 180 degrees	Swap the X+ and X- connections, and swap the Y+ and Y- connections
Touch coordinates are mirrored diagonally	Swap the X+ and Y+ connections, and swap the X- and Y- connections
Touch coordinates do not seem to track, and the distortion is not a simple flip/rotate/mirror transformation	Swap the X+ and Y+ connections; if distortion persists, swap the X+ and Y- connections; if distortion still persists, disconnect touch screen and use DVM to verify X+ to X- resistance and Y+ to Y- resistance; verify with no touch X+ and X- are isolated from Y+ and Y-

3.4) Detecting Touch: Demand Scan

To configure the MAX1234 to detect touch and digitize touch location on demand, write register 0x40 (ADC Control) with PENSTS=0 and ADSTS=0 (refer to Table 6 in the MAX1233/MAX1234 data sheet). After reading register 0x00 (X coordinate), the PENIRQ-bar signal latches low when subsequent touch is detected, and remains low until ADC Control is written to measure the X,Y coordinate.

Table 12. Touch-Screen Measurement Command Sequence: Demand Scan

DEMO1234 Command	Action	SPI data in	Verification
T W AC 0b01	Demand scan	0x0040 0x0b01	
T R AX	Read conversion result register X	0x8000 0x0000	
P R 6	Read PENIRQ-bar pin status		PENIRQ = 1
	Touch the touch screen		
P R 6	Read PENIRQ-bar pin status		PENIRQ = 0
		0x0040 0x0b01 0x8000 0x0000	

T M2	Measure X,Y,Z1,Z2	0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	
P R 6	Read PENIRQ-bar pin status		PENIRQ = 1
	Touch and hold the touch screen		
P R 6	Read PENIRQ-bar pin status		PENIRQ = 0
T M2	Measure X,Y,Z1,Z2	0x0040 0x0b01 0x8000 0x0000 0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	
P R 6	Read PENIRQ-bar pin status		PENIRQ = 0
T M2	Measure X,Y,Z1,Z2	0x0040 0x0b01 0x8000 0x0000 0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	
P R 6	Read PENIRQ-bar pin status		PENIRQ = 0
	Release the touch screen		
P R 6	Read PENIRQ-bar pin status		PENIRQ = 0
		0x0040 0x0b01 0x8000 0x0000	

T M2	Measure X,Y,Z1,Z2	0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	
P R 6	Read PENIRQ-bar pin status		PENIRQ = 1

3.5) Detecting Touch: Autoscan

To configure the MAX1234 to digitize touch location automatically when screen touch is detected, write register 0x40 (ADC Control) with PENSTS=1 and ADSTS=0 (refer to Table 6 in the MAX1233/MAX1234 data sheet). The PENIRQ-bar signal briefly pulses low when the screen is first touched, and does not pulse again until after the X register is read.

Table 13. Touch-Screen Measurement Command Sequence: Autoscan

DEMO1234 Command	Action	SPI data in	Verification
	Optional: connect oscilloscope to PENIRQ-bar		
I C 1 3	Configure PENIRQ-bar pulse accumulator: falling-edge trigger		
I 0 1	Reset the pulse accumulator		
I R 1	Read the number of times PENIRQ-bar has pulsed low		count = 0
T W AC 8bff	Wait for touch, then scan X,Y,Z1,Z2	0x0040 0x8bff	
	Touch the touch screen		PENIRQ pulse
I R 1	Read the number of times PENIRQ-bar has pulsed low		count has increased
T R P	Read X,Y,Z1,Z2 conversion results	0x8000 0x0000 0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	
	Touch the touch screen		PENIRQ pulse
I R 1	Read the number of times PENIRQ-bar has pulsed low		count has increased
		0x8000 0x0000	

T R P	Read X,Y,Z1,Z2 conversion results	0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	
	Touch the touch screen		PENIRQ pulse
I R 1	Read the number of times PENIRQ-bar has pulsed low		count has increased
T R P	Read X,Y,Z1,Z2 conversion results	0x8000 0x0000 0x8001 0x0000 0x8002 0x0000 0x8003 0x0000	

4) Keypad and General-Purpose Input/Output Pins

The following examples show how to use the DEMO1234.EXE program to scan a keypad, and how to use the key-scanning pins for GPIO.

4.1) Configuring Keypad and GPIO pins

The GPIO Control register configures each of the C1–C4 and R1–R4 pins as either an input, an output, or as part of the keypad (refer to Tables 26 and 27 in the MAX1233/MAX1234 data sheet). Additionally, output pins can be configured as open-drain outputs by writing the GPIO Pullup Disable register.

Table 14. Keypad and GPIO Configuration Examples

DEMO1234 Command	Action	SPI data in
T W GC FFFF	Keypad: none; GPIO outputs: C4,C3,C2,C1,R4,R3,R2,R1; GPIO inputs: none	0x004f 0xffff
T W GC FF00	Keypad: none; GPIO outputs: none; GPIO inputs: C4,C3,C2,C1,R4,R3,R2,R1	0x004f 0xff00
T W GC 0000	Keypad: (C4,C3,C2,C1) x (R4,R3,R2,R1); GPIO outputs: none;	0x004f 0x0000

	GPIO inputs: none	
T W GC C8C0	Keypad: (C2,C1) x (R3,R2,R1); GPIO outputs: C4,C3; GPIO input: R4	0x004f 0xc8c0
T W GP 4000	GPIO pullup disable: C3	0x004e 0x4000

4.2) Reading and Writing GPIO Pins

The GPIO Data register reads GPIO input pins and writes GPIO output pins. Note: in these examples, C3, C4, and R4 are pin names, not component names.

Table 15. GPIO Examples

DEMO1234 Command	Action	SPI data in	Verification
T W GC C8C0	Keypad: (C2,C1) x (R3,R2,R1); GPIO outputs: C4,C3; GPIO input: R4	0x004f 0xc8c0	
T W GP 4000	GPIO pullup disable: C3	0x004e 0x4000	
	Connect external resistor between C3 pin and DVDD		
	Connect DVM to C4 pin		
T W GD 8000	GPIO write C4 = 1	0x000f 0x8000	C4 pin = high
T W GD 0000	GPIO write C4 = 0	0x000f 0x0000	C4 pin = low
T W GD 8000	GPIO write C4 = 1	0x000f 0x8000	C4 pin = high
T W GD 0000	GPIO write C4 = 0	0x000f 0x0000	C4 pin = low
	Connect DVM to C3 pin		
T W GD 4000	GPIO write C3 = 1	0x000f 0x4000	C3 pin = high
T W GD 0000	GPIO write C3 = 0	0x000f 0x0000	C3 pin = low
T W GD 4000	GPIO write C3 = 1	0x000f 0x4000	C3 pin = high
T W GD 0000	GPIO write C3 = 0	0x000f 0x0000	C3 pin = low
	Connect R4 pin to DVDD		
T R GD	GPIO read	0x800f 0x0000	Buffer = 0x0800
	Connect R4 pin to GND		
T R GD	GPIO read	0x800f 0x0000	Buffer = 0x0000

4.3) Detecting Keypress: Autoscan

The Keypad Control register can be configured to scan the keypad automatically when a keypress is detected.

Table 16. Keypress Command Sequence: Autoscan

DEMO1234 Command	Action	SPI data in	Verification
	Optional: connect oscilloscope to KEYIRQ-bar		
I C 0 3	Configure KEYIRQ-bar pulse accumulator: falling-edge trigger		
I 0 0	Reset the pulse accumulator		
I R 0	Read the number of times KEYIRQ-bar has pulsed low		count = 0
T W GC 0000	Keypad: (C4,C3,C2,C1) x (R4,R3,R2,R1); GPIO outputs: none; GPIO inputs: none	0x004f 0x0000	
T W KC bf00	Wait for keypress; maximum debounce and hold times	0x0041 0xbf00	
	Press and release R1C1 (key "1")		KEYIRQ pulse
I R 0	Read the number of times KEYIRQ-bar has pulsed low		count has increased
T R KB	Read raw keypad result	0x8004 0x0000	0x0001 = R1C1 key
	Press and release R2C2 (key "5")		KEYIRQ pulse
I R 0	Read the number of times KEYIRQ-bar has pulsed low		count has increased
T R KB	Read raw keypad result	0x8004 0x0000	0x0020 = R2C2 key
	Press and release R3C2 (key "8")		KEYIRQ pulse
I R 0	Read the number of times KEYIRQ-bar has pulsed low		count has increased
T R KB	Read raw keypad result	0x8004 0x0000	0x0040 = R3C2 key

4.4) Masking Individual Keys off the Keypad

Mask off individual keys using the Key Mask register and the Keypad 2 result register. Masked keys are scanned into the KPD register, but are not reported in the Keypad 2 result register.

Table 17. Keypress Command Sequence: Mask off Individual Keys

DEMO1234 Command	Action	SPI data in	Verification
T W GC 0000	Keypad: (C4,C3,C2,C1) x (R4,R3,R2,R1); GPIO outputs: none; GPIO inputs: none	0x004f 0x0000	
T W KC bf00	Wait for keypress; maximum debounce and hold times	0x0041 0xbf00	
T W KM 0020	Mask only R2C2 key	0x0050 0x0020	
	Press and release R1C1 (key "1")		
T R KB	Read raw keypad result	0x8004 0x0000	0x0001 = R1C1 key

T R K2	Read masked keypad result	0x8011 0x0000	0x0001 = R1C1 key
	Press and release R2C2 (key "5")		
T R KB	Read raw keypad result	0x8004 0x0000	0x0020 = R2C2 key
T R K2	Read masked keypad result	0x8011 0x0000	0x0000 = no key
	Press and release R3C2 (key "8")		
T R KB	Read raw keypad result	0x8004 0x0000	0x0040 = R3C2 key
T R K2	Read masked keypad result	0x8011 0x0000	0x0040 = R3C2 key

4.5) Masking a Column off the Keypad

Mask off entire columns using the Key Column register. A masked column does not get scanned, so the KPD register never detects keys in that column.

Table 18. Keypress Command Sequence: Mask off an Entire Column of Keys

DEMO1234 Command	Action	SPI data in	Verification
T W GC 0000	Keypad: (C4,C3,C2,C1) x (R4,R3,R2,R1); GPIO outputs: none; GPIO inputs: none	0x004f 0x0000	
T W KC bf00	Wait for keypress; maximum debounce and hold times	0x0041 0xbf00	
T W KK 2000	Mask entire C2 column	0x0051 0x2000	
	Press and release R1C1 (key "1")		
T R KB	Read raw keypad result	0x8004 0x0000	0x0001 = R1C1 key
	Press and release R2C2 (key "5")		
T R KB	Read raw keypad result	0x8004 0x0000	(previous value)
	Press and release R3C2 (key "8")		
T R KB	Read raw keypad result	0x8004 0x0000	(previous value)
	Press and release R2C3 (key "6")		
T R KB	Read raw keypad result	0x8004 0x0000	0x0200 = R2C3 key

5) Managing Power Consumption

Table 19. Power-Off Commands

DEMO1234 Command	Action	SPI data in	Verification
T W AC C000	Power off ADC	0x0040 0xc000	—
T W AC 0300	Power off internal reference	0x0040 0x0300	REF = not driven
T W DC 8000	Disable DAC	0x0042 0x8000	DACOUT = 0.0V
T W KC C000	Power off keypad	0x0041 0xc000	—

6) Menu System

The complete source code implements the following console menu system which connects to the MINIQUSB+ module.

CmodComm test program main menu—prior to connect
A) adjust timing parameters
L) CmodLog... functions
C) connect
D) Debug Messages
X) exit

Response to C (Connect) Command
C
Hardware supports optimized native SMBus commands.

Board connected.

Got board banner: Maxim MINIQUSB V01.05.41 >
Firmware version is OK.
(configured for SPI auto-CS 4-byte mode) (SCLK=2MHz) ...

Main menu—valid after connect
T) Test the device
8) CmodP8Bus... functions
A) adjust timing parameters
L) CmodLog... functions
P) CmodPin... functions
S) CmodSpi... functions
M) CmodSMBus... functions
$) CmodCommStringWrite list of hex codes

R) CmodBoardReset
D) Disconnect

Test menu commands— valid after connect
R) Read register
W) Write register
M0) measure no measurement; configure reference
M1) measure X,Y
M2) measure X,Y,Z1,Z2
M3) measure X
M4) measure Y
M5) measure Z1,Z2
M6) measure BAT1/4
M7) measure BAT2/4
M8) measure AUX1
M9) measure AUX2
MA) measure TEMP1
MB) measure BAT1/4,BAT2/4,AUX1,AUX2,TEMP1,TEMP2
MC) measure TEMP1,TEMP2
MD) no measurement; drive Y+,Y-
ME) no measurement; drive X+,X-
MF) no measurement; drive Y+,X-
.) Exit this menu

6.1) Register Read/Write Commands

Table 20. Read Register Mnemonics

DEMO1234 Command	Mnemonic	SPI data in
T R A1	Test Read AUX1 register	0x8007 0x0000
T R A2	Test Read AUX2 register	0x8008 0x0000
T R AC	Test Read ADC_control register	0x8040 0x0000
T R AX	Test Read X register	0x8000 0x0000
T R AY	Test Read Y register	0x8001 0x0000
T R AZ1	Test Read Z1 register	0x8002 0x0000
T R AZ2	Test Read Z2 register	0x8003 0x0000
T R B1	Test Read BAT1 register	0x8005 0x0000
T R B2	Test Read BAT2 register	0x8006 0x0000
T R DC	Test Read DAC_control register	0x8042 0x0000
T R DD	Test Read DAC_data register	0x800b 0x0000
T R GC	Test Read GPIO_control register	0x804f 0x0000
T R GD	Test Read GPIO_data register	0x800f 0x0000
T R GP	Test Read GPIO_pullup register	0x804e 0x0000
T R K1	Test Read KPDATA1 register	0x8010 0x0000
T R K2	Test Read KPDATA2 register	0x8011 0x0000
T R KB	Test Read KPD register	0x8004 0x0000
T R KC	Test Read KEY_control register	0x8041 0x0000
T R KK	Test Read KPCOLMASK register	0x8051 0x0000
T R KM	Test Read KPKEYMASK register	0x8050 0x0000
T R T1	Test Read TEMP1 register	0x8009 0x0000
T R T2	Test Read TEMP2 register	0x800a 0x0000

Table 21. Write Register Mnemonics

DEMO1234 Command	Mnemonic	SPI data in
T W AC *hexValue*	Test Write ADC_control register	0x0040 *hexValue*
T W DC *hexValue*	Test Write DAC_control register	0x0042 *hexValue*
T W DD *hexValue*	Test Write DAC_data register	0x000b *hexValue*
T W GC *hexValue*	Test Write GPIO_control register	0x004f *hexValue*
T W GD *hexValue*	Test Write GPIO_data register	0x000f *hexValue*
T W GP *hexValue*	Test Write GPIO_pullup register	0x004e *hexValue*
T W KC *hexValue*	Test Write KEY_control register	0x0041 *hexValue*
T W KK *hexValue*	Test Write KPCOLMASK register	0x0051 *hexValue*
T W KM *hexValue*	Test Write KPKEYMASK register	0x0050 *hexValue*

Table 22. Touch-Screen Measurement Command Sequences

DEMO1234 Command	Action (Triggered by A/D3210 Bits)	SPI data in Sequence
T M1	Measure X,Y	0x0040 0x0701 0x8000 0x0000

		0x8001 0x0000
T M2	Measure X,Y,Z1,Z2	0x0040 0x0b01 0x8000 0x0000 0x8001 0x0000 0x8002 0x0000 0x8003 0x0000
T M3	Measure X	0x0040 0x0f01 0x8000 0x0000
T M4	Measure Y	0x0040 0x1301 0x8001 0x0000
T M5	Measure Z1,Z2	0x0040 0x1701 0x8002 0x0000 0x8003 0x0000
T MD	No measurement; drive Y+,Y-	0x0040 0x3701
T ME	No measurement; drive X+,X-	0x0040 0x3b01
T MF	No measurement; drive Y+,X-	0x0040 0x3f01

6.2) Interrupt and Status Pin Commands

Table 23. Pin Status Read Commands

DEMO1234 Command	Action	SPI data in
P R 5	Read KEYIRQ-bar pin status	N/A
I C 0 3	Enable KEYIRQ-bar falling-edge trigger pulse accumulator	N/A
I C 0 1	Enable KEYIRQ-bar rising-edge trigger pulse accumulator	N/A
I C 0 0	Disable KEYIRQ-bar pulse accumulator	N/A
I R 0	Read the number of times KEYIRQ-bar has pulsed low	N/A
I 0 0	Clear the KEYIRQ-bar pulse accumulator	N/A
P R 6	Read PENIRQ-bar pin status	N/A
I C 1 3	Enable PENIRQ-bar falling-edge trigger pulse accumulator	N/A
I C 1 1	Enable PENIRQ-bar rising-edge trigger pulse accumulator	N/A
I C 1 0	Disable PENIRQ-bar pulse accumulator	N/A
I R 1	Read the number of times PENIRQ-bar has pulsed low	N/A
I 0 1	Clear the PENIRQ-bar pulse accumulator	N/A
P R 7	Read BUSY-bar pin status	N/A

6.3) Commands Added to Upgraded MINIQUSB+ Firmware

Table 24. SPI Commands Available in Upgraded MINIQUSB+ Firmware 01.05.40

DEMO1234 Command	Action	CPOL	CPHA	CS-Bar Control	AF Length
S C L0	Configure SPI for CPOL=0	0	—	GPIO-K9	1 byte
S C L1	Configure SPI for CPOL=1	1	—	GPIO-K9	1 byte

S C A0	Configure SPI for CPHA=0	—	0	GPIO-K9	1 byte
S C A1	Configure SPI for CPHA=1	—	1	GPIO-K9	1 byte
S C C0	Configure SPI for 8-bit	—	—	GPIO-K9	1 byte
S C C1	Configure SPI for 8-bit auto-CS-bar	—	—	Automatic	1 byte
S C C2	Configure SPI for 16-bit auto-CS-bar	—	—	Automatic	2 bytes
S C C3	Configure SPI for 24-bit auto-CS-bar	—	—	Automatic	3 bytes
S C C4	Configure SPI for 32-bit auto-CS-bar	—	—	Automatic	4 bytes
$ 2 AE 00	Configure SPI for 8-bit	0	0	GPIO-K9	1 byte
$ 2 AE 01	Configure SPI for 8-bit	0	1	GPIO-K9	1 byte
$ 2 AE 02	Configure SPI for 8-bit	1	0	GPIO-K9	1 byte
$ 2 AE 03	Configure SPI for 8-bit	1	1	GPIO-K9	1 byte
$ 2 AE 08	Configure SPI for 8-bit auto-CS-bar	0	0	Automatic	1 byte
$ 2 AE 09	Configure SPI for 8-bit auto-CS-bar	0	1	Automatic	1 byte
$ 2 AE 0A	Configure SPI for 8-bit auto-CS-bar	1	0	Automatic	1 byte
$ 2 AE 0B	Configure SPI for 8-bit auto-CS-bar	1	1	Automatic	1 byte
$ 2 AE 18	Configure SPI for 16-bit auto-CS-bar	0	0	Automatic	2 bytes
$ 2 AE 19	Configure SPI for 16-bit auto-CS-bar	0	1	Automatic	2 bytes
$ 2 AE 1A	Configure SPI for 16-bit auto-CS-bar	1	0	Automatic	2 bytes
$ 2 AE 1B	Configure SPI for 16-bit auto-CS-bar	1	1	Automatic	2 bytes
$ 2 AE 28	Configure SPI for 24-bit auto-CS-bar	0	0	Automatic	3 bytes
$ 2 AE 29	Configure SPI for 24-bit auto-CS-bar	0	1	Automatic	3 bytes
$ 2 AE 2A	Configure SPI for 24-bit auto-CS-bar	1	0	Automatic	3 bytes
$ 2 AE 2B	Configure SPI for 24-bit auto-CS-bar	1	1	Automatic	3 bytes
$ 2 AE 38	Configure SPI for 32-bit auto-CS-bar	0	0	Automatic	4 bytes
$ 2 AE 39	Configure SPI for 32-bit auto-CS-bar	0	1	Automatic	4 bytes
$ 2 AE 3A	Configure SPI for 32-bit auto-CS-bar	1	0	Automatic	4 bytes
$ 2 AE 3B	Configure SPI for 32-bit auto-CS-bar	1	1	Automatic	4 bytes
$ 2 AF *xx*	Perform an 8-bit SPI transfer (CS-bar = GPIO or auto-CS-bar = 1-byte)	—	—	—	1 byte
$ 3 AF *xx xx*	Perform a 16-bit SPI transfer (requires auto-CS-bar = 2-byte mode)	—	—	—	2 bytes
$ 4 AF *xx xx xx*	Perform a 24-bit SPI transfer (requires auto-CS-bar = 3-byte mode)	—	—	—	3 bytes
$ 5 AF *xx xx xx xx*	Perform a 32-bit SPI transfer (requires auto-CS-bar = 4-byte mode)	—	—	—	4 bytes
$ 2 F9 0	Drive CS-bar pin low	—	—	GPIO-K9	—
$ 2 F9 1	Drive CS-bar pin high	—	—	GPIO-K9	—

Table 25. Interrupt Pulse Accumulator Commands in Upgraded MINIQUSB+ Firmware 01.05.41

DEMO1234 Command	Action	Int	GPIO Input	Firmware Command
$ 2 C3	Query which of the C3 commands are supported; the return value			

00	is a 2-byte bitmap of commands C300 to C30F, msb first	—	—	C3 00
I Q 0	Query configuration of pulse accumulator	INT0	GPIO-K5	C3 01 00
I Q 1	Query configuration of pulse accumulator	INT1	GPIO-K6	C3 01 01
I Q 2	Query configuration of pulse accumulator	INT2	GPIO-K7	C3 01 02
I Q 3	Query configuration of pulse accumulator	INT3	GPIO-K8	C3 01 03
I C 0 0	Configure pulse accumulator: disable interrupt	INT0	GPIO-K5	C3 02 00 00
I C 1 0	Configure pulse accumulator: disable interrupt	INT1	GPIO-K6	C3 02 01 00
I C 2 0	Configure pulse accumulator: disable interrupt	INT2	GPIO-K7	C3 02 02 00
I C 3 0	Configure pulse accumulator: disable interrupt	INT3	GPIO-K8	C3 02 03 00
I C 0 1	Configure pulse accumulator: rising-edge trigger	INT0	GPIO-K5	C3 02 00 01
I C 1 1	Configure pulse accumulator: rising-edge trigger	INT1	GPIO-K6	C3 02 01 01
I C 2 1	Configure pulse accumulator: rising-edge trigger	INT2	GPIO-K7	C3 02 02 01
I C 3 1	Configure pulse accumulator: rising-edge trigger	INT3	GPIO-K8	C3 02 03 01
I C 0 3	Configure pulse accumulator: falling-edge trigger	INT0	GPIO-K5	C3 02 00 03
I C 1 3	Configure pulse accumulator: falling-edge trigger	INT1	GPIO-K6	C3 02 01 03
I C 2 3	Configure pulse accumulator: falling-edge trigger	INT2	GPIO-K7	C3 02 02 03
I C 3 3	Configure pulse accumulator: falling-edge trigger	INT3	GPIO-K8	C3 02 03 03
I R 0	Read pulse accumulator	INT0	GPIO-K5	C3 03 00
I R 1	Read pulse accumulator	INT1	GPIO-K6	C3 03 01
I R 2	Read pulse accumulator	INT2	GPIO-K7	C3 03 02
I R 3	Read pulse accumulator	INT3	GPIO-K8	C3 03 03
I 0 0	Clear pulse accumulator	INT0	GPIO-K5	C3 04 00
I 0 1	Clear pulse accumulator	INT1	GPIO-K6	C3 04 01

`I 0 2`	Clear pulse accumulator	INT2	GPIO-K7	`C3 04 02`
`I 0 3`	Clear pulse accumulator	INT3	GPIO-K8	`C3 04 03`
`I S 0 xx`	Set pulse accumulator count xx = 0 to 255	INT0	GPIO-K5	`C3 05 00 xx`
`I S 1 xx`	Set pulse accumulator count xx = 0 to 255	INT1	GPIO-K6	`C3 05 01 xx`
`I S 2 xx`	Set pulse accumulator count xx = 0 to 255	INT2	GPIO-K7	`C3 05 02 xx`
`I S 3 xx`	Set pulse accumulator count xx = 0 to 255	INT3	GPIO-K8	`C3 05 03 xx`

7) Conclusion

These examples have briefly shown how to use each of the major functional blocks of the MAX1233/MAX1234 to measure and control hardware, using a simplified console-based C++ program. Consult the MAX1233/MAX1234 data sheet for in-depth details.

Related Parts		
MAX1233	±15kV ESD-Protected Touch-Screen Controllers Include DAC and Keypad Controller	Free Samples
MAX1234	±15kV ESD-Protected Touch-Screen Controllers Include DAC and Keypad Controller	Free Samples