XAPP356 (v1.0) August 6, 2001

XPATH Module Design with CoolRunner XPLA3 and Handspring

Summary

This application note illustrates the implementation of a Handspring™ Springboard™ module design. The XPATH (Xilinx Pressure Altimeter Temperature Heading) module described here is implemented using the Xilinx CoolRunner™ XPLA3 CPLD and Handspring™ Visor. CoolRunner CPLDs are the lowest power CPLDs available and the ideal target device for portable handheld applications. For more information on developing with Handspring, please see section, "References" on page 30.

This application note describes the hardware (both board and VHDL design) and software (PocketC code) development. The VHDL and C code for this design may be downloaded from the Xilinx website site; more information can be found in the section "HDL and C Code" on page 29.

Introduction

The XPATH board was created for use as a reference design to aid developers creating Handspring add-on modules. The Insight Springboard Development Kit was used for development. For more information on this development kit, please refer to "References" on page 30.

The Insight Springboard Development Kit includes a 12-bit serial Data Acquisition System or ADC (ADS7870 from Burr-Brown), a Toshiba 4-Mbit SRAM, a Toshiba 32-Mbit Flash, and a Xilinx CoolRunner XPLA3 256-macrocell CPLD. The ADS7870 has eight available single-ended or four differential-ended analog input channels. The XPATH board was designed to fit onto the daughter card connectors available on the Insight Springboard Development board.

Figure 1 illustrates a high-level diagram of the XPATH module. Appropriate transducers were used to collect data for obtaining heading, temperature, and pressure; altitude is calculated using the pressure and temperature sensor. The ADC is utilized to gather data from the pressure and magnetic field transducers. The CoolRunner XPLA3 CPLD is responsible for interfacing with the ADC, the SRAM, the temperature sensor, and the Handspring Springboard Expansion Slot.

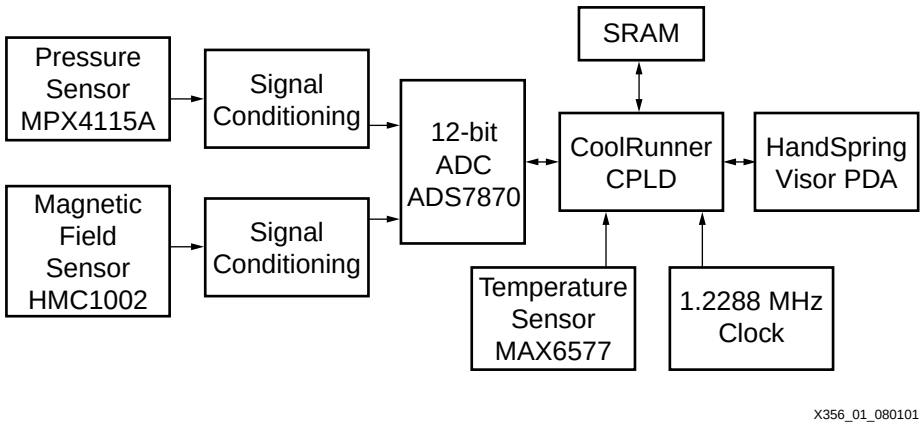

Figure 1: **XPATH Block Diagram**

All Handspring PDA development was done using PocketC. PocketC allows developers to create standalone PRC files to run a desired PDA application. Refer to "References" on page 30 for more information on developing with PocketC.

Analog Design Aspects

The XPATH analog board was designed to fit on the daughter card area of the Insight Springboard Development board. The XPATH board and the Insight Springboard Development board are shown in Figure 2.

Figure 2: **XPATH and Insight Development Board**

The analog components used on the Insight Springboard Development Kit and the XPATH board are described in this section.

Burr-Brown ADS7870

All eight channels of the ADS7870 ADC are configured for single-ended operation. Table 1 shows the functionality assigned to each channel.

Table 1: **ADC Input Configuration**

ADC Input Channel	Label	Function
A/D 0	N/A	N/A
A/D 1	Y AXIS	Magnetic Sensor Y Axis Output
A/D 2	Pressure	Pressure Sensor Output
A/D 3	N/A	N/A
A/D 4	18V	9V Battery Boost Voltage Monitor
A/D 5	X AXIS	Magnetic Sensor X Axis Output
A/D 6	BATT	9V Battery Voltage Monitor
A/D 7	MIC	Dedicated to MIC input from Handspring (Not applicable in this design)

During initialization, the ADC is configured to output the conversion clock, CCLK. CCLK is approximately 2.5 MHz and is used as the system clock in the CPLD. The gain of the ADC is set to unity during initialization. Initialization of the ADC is described in detail in "ADC Interface" on page 9.

Magnetic Field Sensor

The Honeywell HMC1002 magnetic sensor is used to detect the earth's magnetic field. The earth's magnetic field is approximately 0.5 gauss. Utilizing a 2-axis sensor device, a digital compass can be developed. With an X and Y axis output from the magnetic sensor, a bearing can be calculated. The Honeywell HMC1002 magnetic sensor is configured as a 4-element wheatstone bridge. Each resistor in the bridge is a magnetoresistive sensor, which will convert a magnetic field into a differential output voltage. The differential output voltage is measured by the differential amplifier stage and then read by the ADC.

The HMC1002 magnetic sensor requires a SET and RESET current pulse to realign the magnetic domains in the sensor. After applying each current pulse, the output of each bridge is read. The bridge output can be expressed as:

$$Vout = \frac{Vout(set) - Vout(reset)}{2}$$

This technique cancels out offset and temperature effects introduced by the electronics as well as the bridge temperature drift.

For more information on the Honeywell HMC1002 magnetic sensor products, refer to "References" on page 30.

Pressure Sensor

To determine atmospheric pressure, the Motorola MPX4115A pressure transducer is used in this design. The MPX4115A requires a 5V voltage supply and an output amplification stage.

The MPX4115A pressure transducer is able to detect pressure from 15 kPa to 115 kPa. The MPX4115A has a single output voltage as follows, where P is measured in kPa:

$$Vout = Vs * ((0.009 * P) - 0.095) \pm ERROR$$
$$Vs = 5.0\ Vdc$$

For more information on the Motorola MPX4115A pressure transducer, refer to "References" on page 30.

Temperature Sensor

The temperature sensor used in this design is the Maxim 6577 single-wire output. The MAX6577 converts the ambient temperature into a square wave with a frequency proportional to absolute temperature. The frequency range of the output is selectable by hard-wiring two time-select pins. TS1 and TS0 were both pulled to ground to select a 4X frequency output. By doing so, the time necessary to read temperature is reduced by a factor of four. The frequency output of the MAX6577 is read directly by the CPLD. The output frequency of the MAX6577 is measured in degrees Kelvin. The temperature range of the MAX6577 is -40°C to 125°C.

For more information on the Maxim MAX6577 temperature sensor, refer to "References" on page 30.

CPLD Design Aspects

The CoolRunner XPLA3 CPLD is responsible for initializing the ADC for data acquisition, capturing data from the ADC, storing data in SRAM, reading the frequency output from the temperature sensor, and interfacing to the Handspring Springboard Expansion Slot. Each of these functions are described in the sections that follow.

Control Register

The CPLD control register allows the Handspring processor to issue commands to the CPLD and allows the CPLD to provide status back to the Handspring processor. The control register is accessed by writing/reading to address 0x29000000. The control register is defined as shown in Table 2 with the specification for each bit.

Table 2: **Control Register Definition**

Control Register Bit	Identifier	Description
CNTR7	N/A	N/A
CNTR6	N/A	N/A
CNTR5	OPMODE	CNTR5, the OPMODE control bit, is used to initiate an execute command. The OPMODE bit is = "1" when the processor wishes to send an execute command to the CPLD. Refer to "XPATH Module Design" on page 11 for more information on the execute command function.
CNTR4	DONE	The DONE bit is asserted by the CPLD and read by the processor. The DONE bit will be set (DONE = "1") once the CPLD is done processing the current execute cycle. The processor must poll for this bit to be set by the CPLD before valid data stored in SRAM can be read. While the DONE bit = "0" the CPLD has control of the data, address, and control lines of the SRAM. The SRAM bus is released to the Handspring processor when the DONE bit = "1".
CNTR3	N/A	N/A

Table 2: **Control Register Definition *(Continued)***

Control Register Bit	Identifier	Description
CNTR2	BATTON	BATTON is the 9V battery enable control bit. Whenever an execute command is sent, the battery enable bit should also be asserted. The Handspring can reduce power consumption by writing a "0" to the bit and disabling the battery power supply to the analog components.
CNTR1	PRES1	It is the responsibility of the processor to assign the CNTR1 and CNTR0 bits. CNTR1 and CNTR0, PRES1 and PRES0 control bits are set by the Handspring processor to alert the CPLD of the operating range of the pressure sensor. The definition of these bits is shown in Table 3.
CNTR0	PRES0	

Table 3: **Pressure Range Definition**

PRES1	PRES0	Definition
0	0	Range 0 - At lowest elevation
0	1	Range 1
1	0	Range 2
1	1	Range 3 - At highest elevation

Memory Space Definition

The CPLD is responsible for reading data from the ADC and temperature sensor. Upon reading the sensor data, the data needs to be stored in SRAM. This allows the Handspring processor to read SRAM and display the data. See "References" on page 30 for more information on the Handspring address space. The Handspring SRAM address space for this module design is shown in Table 4.

Table 4: **SRAM Memory Space**

SRAM Address	Definition	
0x29000000	Used to specify CPLD control register	
0x29000002		
0x29000004	Magnetic X Axis Zero Value	
0x29000006		
0x29000008	Magnetic Y Axis Zero Value	
0x2900000A		
0x2900000C	9V Battery Value	
0x2900000E		
0x29000010		
0x29000012	X Axis Set Sample 1	Magnetic Sensor Data
0x29000014	X Axis Set Sample 2	
0x29000016	X Axis Set Sample 3	

Table 4: **SRAM Memory Space *(Continued)***

SRAM Address	Definition	
0x29000018		
0x2900001A	Y Axis Set Sample 1	Magnetic Sensor Data
0x2900001C	Y Axis Set Sample 2	
0x2900001E	Y Axis Set Sample 3	
0x29000020		
0x29000022	X Axis Reset Sample 1	
0x29000024	X Axis Reset Sample 2	
0x29000026	X Axis Reset Sample 3	
0x29000028		
0x2900002A	Y Axis Reset Sample 1	
0x2900002C	Y Axis Reset Sample 2	
0x2900002E	Y Axis Reset Sample 3	
0x29000030		
0x29000032	Temp Data Sample 1	Temperature Sensor Data
0x29000034	Temp Data Sample 2	
0x29000036	Temp Data Sample 3	
0x29000038		
0x2900003A		
0x2900003C		
0x2900003E		
0x29000040		
0x29000042	Pressure Data Sample 1	Pressure Sensor Data
0x29000044	Pressure Data Sample 2	
0x29000046	Pressure Data Sample 3	

Overview

The CoolRunner CPLD is responsible for controlling the analog components on the XPATH board. Table 5 highlights the main XPATH analog control signals controlled by the CoolRunner CPLD.

Table 5: **Analog Control Signals**

Signal Name	Function
sr_sdn	Active Low shutdown switch on 9V boost regulator
five_v_enn	Active Low enable for 5V switching regulator
mag_set	Active High magnetic sensor SET pulse enable
mag_reset	Active High magnetic sensor RESET pulse enable

Table 5: **Analog Control Signals *(Continued)***

Signal Name	Function
batt_on	Active High 9V battery enable signal (Asserted when the BATT_ON = "1" from Control Register)
temp_out	Output from MAX6577 temperature sensor
pres_range_io0[1]	Active High enable to pressure circuit for Range 1 operation (Asserted when PRES1 and PRES0 = "01" from Control Register)
pres_range_io3[1]	Active High enable to pressure circuit for Range 2 operation (Asserted when PRES1 and PRES0 = "10" from Control Register)
pres_range_io5[1]	Active High enable to pressure circuit for Range 3 operation (Asserted when PRES1 and PRES0 = "11" from Control Register)

Notes:

1. When pres_range_io0 = pres_range_io3 = pres_range_io5 = 0 then the pressure circuit is enabled to operate in Range 0 (sea level pressure range).

Figure 3 illustrates the high-level design of the CPLD for the XPATH add-on module. The CPLD interfaces with the Burr Brown ADS7870 data acquisition system, the Toshiba 4-Mbit SRAM,

the Handspring Springboard expansion slot, the MAX6577 temperature sensor, and the XPATH board.

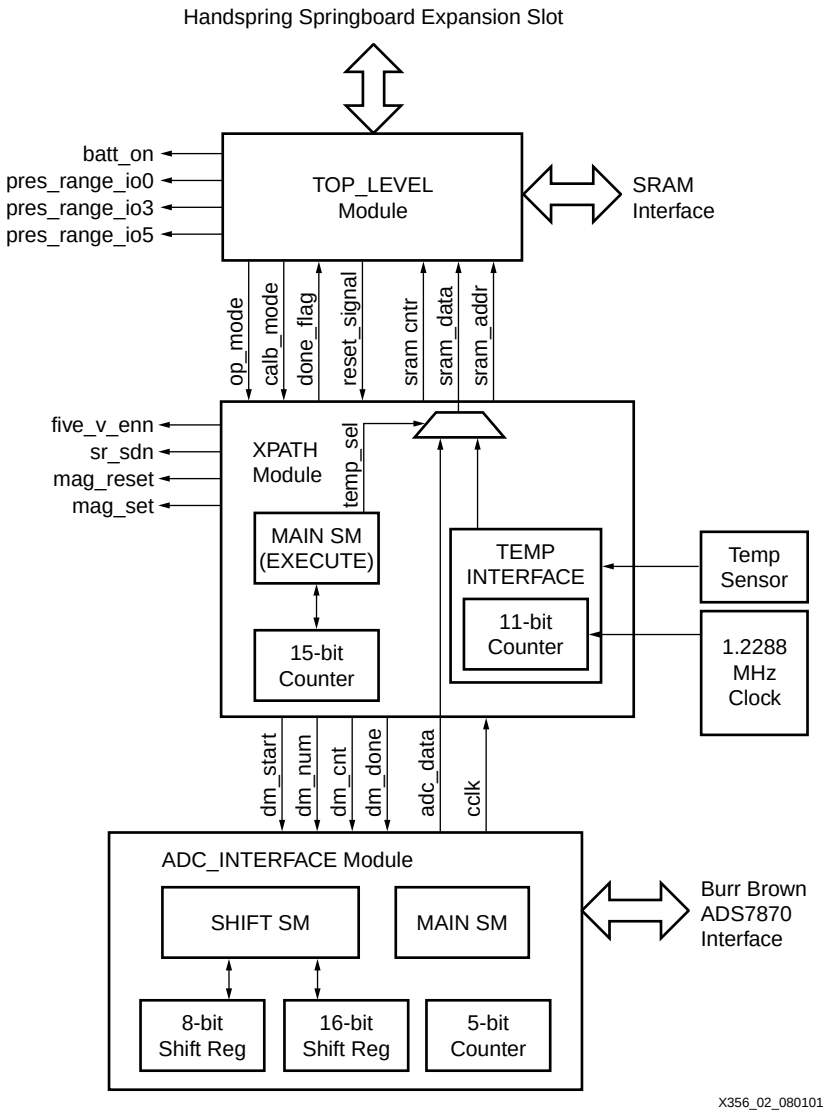

Figure 3: **CPLD Top Level Block Diagram**

TOP_LEVEL Module

The TOP_LEVEL module is responsible for interfacing the XPLA3 CPLD to the Handspring Springboard Expansion Slot. The TOP_LEVEL module is responsible for decoding a read/write to the control register, reset address, or any SRAM address. This module will assert the pressure range signals and the battery enable, batt_on signal. It is also responsible for interfacing with the Toshiba 4-Mbit SRAM. The XPATH logic is also instantiated within the TOP_LEVEL module.

The SRAM and Handspring interface logic in the TOP_LEVEL module is described in XAPP147 (refer to "References" on page 30 for more information). The TOP_LEVEL VHDL design is described in more detail in "Top Level Module Design" on page 15.

XPATH Module

The XPATH module is responsible for executing a calibrate or execute command which is requested by the Handspring processor. The XPATH module utilizes a MAIN state machine to process the execute command. The XPATH module instantiates the TEMP_INTERFACE logic,

that reads the temperature value from the MAX6577 temperature sensor. The XPATH module instantiates the ADC_INTERFACE logic.

The XPATH VHDL logic design is described in more detail in "XPATH Module Design" on page 11.

ADC_INTERFACE Module

The ADC_INTERFACE is responsible for interfacing with the Burr Brown ADS7870. The ADC_INTERFACE initializes the registers of the ADC upon a reset. The ADC_INTERFACE executes direct mode commands for a specific input channel when specified. The ADC_INTERFACE logic design is described in more detail in "ADC Interface" on page 9.

ADC Interface

The Burr Brown ADS7870 ADC interface is described in detail in XAPP355, "Serial ADC Interface Using a CoolRunner CPLD". Refer to "References" on page 30 for more information. The interface described in XAPP355 has been slightly modified for this application. The ADC interface for the XPATH design, allows a high level module to request a direct mode conversion. The direct mode conversion request is initiated by asserting the dm_start signal and the dm_done signal is asserted by the ADC interface when the request is complete. The high level module requesting a direct mode conversion must specify both the input channel to read from and the number of conversions to perform.

With this type of ADC interface, a high level module can specify a starting SRAM address and increment the address counter for each direct mode conversion. The ADC interface described here can easily be modified for use in any application.

The main similarity to the ADC interface described here and the ADC interface described in XAPP355 is the initialization of the ADC internal registers. Once a valid reset signal is asserted on the system, the ADC interface will initialize all ADC registers.

Only two registers are written during initialization in this design and are shown in Table 6.

Table 6: **ADC Register Initialization**

Address	Data	Function
ADDR3	00000100	Specifies ADC Read Back mode 1. Sets the division factor for CCLK = 1 (DCLK = CCLK).
ADDR7	00111110	Specifies clocking source from internal oscillator. CCLK outputs a 2.5 MHz signal Sets V_{REF} = 2.048V

Once the ADC is initialized, the CPLD ADC interface will wait for a dm_start signal to be asserted. The higher level of logic is responsible for the assertion of the dm_start signal and then will then wait for the assertion of dm_done. The ADC interface in the XPATH design is

shown in Figure 4. Refer to XAPP355 (see "References" on page 30) for more information on the SHIFT control logic that is utilized in the ADC interface.

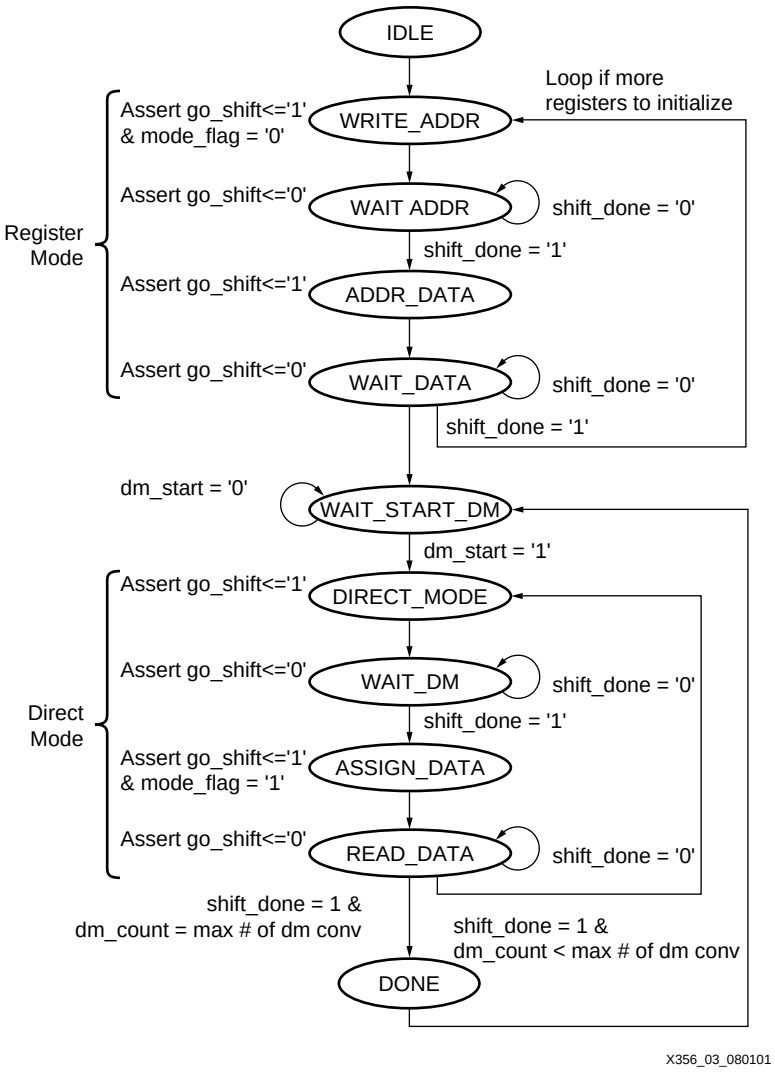

Figure 4: **ADC Interface State Machine**

Note that for each dm_start signal assertion, the ADC interface will execute the number of direct mode conversions that are specified. The number of direct mode conversions are specified by the high-level logic and stored in the dm_conv_cnt signal.

At the end of each direct mode cycle, the count_flag signal is asserted. This enables the high-level logic to increment the SRAM address pointer and write the ADC conversion data to SRAM. The count_flag is asserted in the READ_DATA state before checking to see if the specified number of conversions have been performed. Once the specified number of direct mode conversions are completed, the state machine will progress to the DONE state. In the DONE state, the dm_done signal is asserted.

Anytime the ADC interface logic receives a valid reset, the state machine will go back to the IDLE state. After a valid reset, the CPLD will initialize the ADC registers. Once done initializing all registers, the state machine will wait in the WAIT_START_DM state searching for a valid dm_start signal assertion.

XPATH Module Design

The XPATH module is instantiated by the TOP_LEVEL module as described in "Top Level Module Design" on page 15. The XPATH module as shown in Figure 3 on page 8 is responsible for the following functions:

- Assertion of XPATH analog signals: mag_set, mag_reset, sr_sdn, five_v_enn
- Utilizing the ADC interface described in "ADC Interface" on page 9
- Interfacing with the MAX6577 temperature sensor as described in "Temperature Sensor Interface" on page 14
- Interfacing with the TOP_LEVEL control logic

Figure 5 illustrates the execute operational flow in the MAIN XPATH control logic. The execute command is decoded from a Handspring processor write to the control register. After the XPATH control logic recognizes a valid command, the first operation is to reset the control register, so the execute command is processed only once. Figure 5 illustrates the operational flow for one execute command.

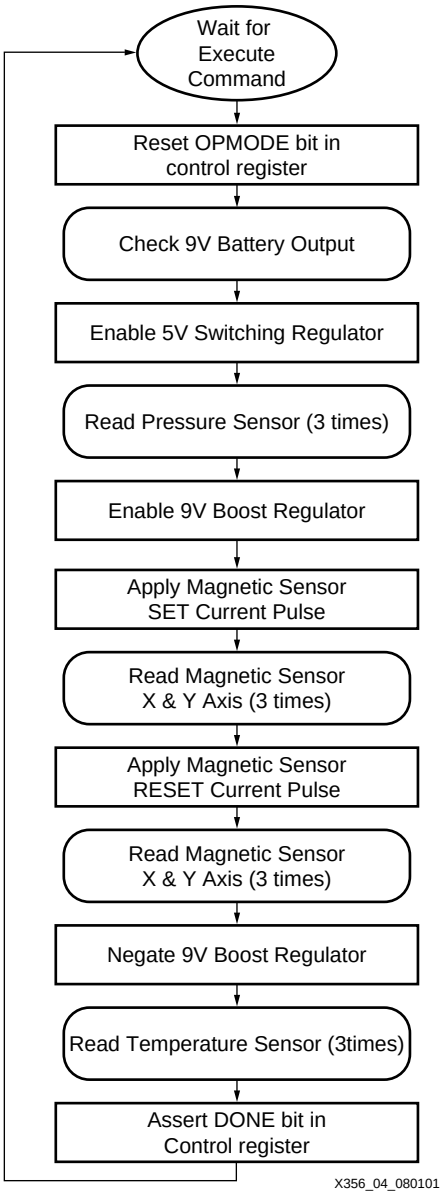

Figure 5: **Execute Operational Flows**

As shown in Figure 6, the MAIN XPATH state machine will process the execute sequence. The execute sequence is started by the assertion of the op_mode control signal.

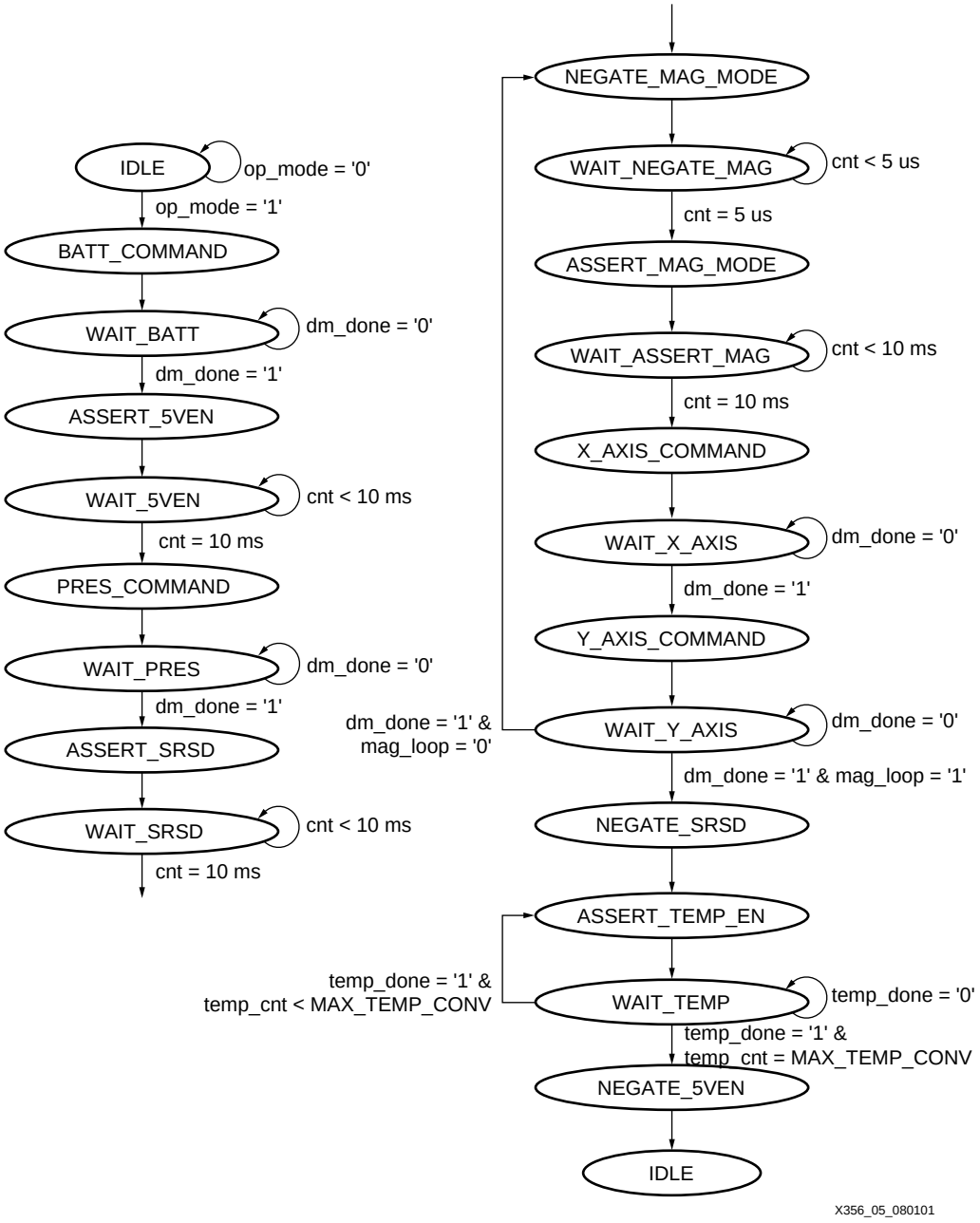

Figure 6: **MAIN XPATH State Machine**

Table 7 describes the functionality of each state in the MAIN XPATH state machine.

Table 7: **Main XPATH State Machine Description**

State	Functionality
IDLE	Initialize specific combinatorial signals. Check the value of op_mode signal to determine next state.
BATT_COMMAND	Reset OPMODE bit in control register. Read 9V battery voltage output. Set direct mode input channel number of ADS7870 to LN6 (dm_num = "110"). Specify the number of direct mode conversions to perform (dm_conv_cnt = "01"). Assert dm_start signal. Assign SRAM address pointer.
WAIT_BATT	Search for assertion of count_flag to increment SRAM address pointer and write battery data to SRAM. Wait for assertion of dm_done signal.
ASSERT_5VEN	Assert five_v_enn active low signal to 5V switching regulator. Clear 15-bit counter.
WAIT_5VEN	Increment 15-bit counter to wait 10 ms after assertion of five_v_enn.
PRES_COMMAND	Read from pressure sensor. Set direct mode input channel number of ADS7870 to LN2 (dm_num = "010"). Specify the number of direct mode conversions to perform (dm_conv_cnt = "11"). Assert dm_start signal. Assign SRAM address pointer.
WAIT_PRES	Search for assertion of count_flag to increment SRAM address pointer and write pressure sensor data to SRAM. Wait for assertion of dm_done signal.
ASSERT_SRSD	Assert sr_sdn active high signal to 9V boost regulator. Clear the 15-bit counter.
WAIT_SRSD	Increment 15-bit counter to wait 10 ms after assertion of sr_sdn.
NEGATE_MAG_MODE	Negate opposite signal for magnetic sensor SET or RESET loop. With first loop, mag_loop = "0" (represents a SET sequence for the magnetic sensor). If mag_loop = "0", deassert mag_reset signal. If mag_loop = "1", de-assert mag_set signal. Clear 15-bit counter.
WAIT_NEGATE_MAG	Increment 15-bit counter to wait 5 µs after deassertion of mag_reset or mag_set.
ASSERT_MAG_MODE	Assert SET/RESET signal for magnetic sensor. If mag_loop = "0", assert mag_set signal. If mag_loop = "1", assert mag_reset signal. Clear 15-bit counter.
WAIT_ASSERT_MAG	Increment 15-bit counter to wait 10 ms after assertion of mag_set or mag_reset.
X_AXIS_COMMAND	Read magnetic sensor X axis value. Set direct mode input channel number of ADS7870 to LN5 (dm_num = "101"). Specify the number of direct mode conversions to perform (dm_conv_cnt = "11"). Assert dm_start signal. Assign SRAM address pointer.

Table 7: **Main XPATH State Machine Description *(Continued)***

State	Functionality
WAIT_X_AXIS	Search for assertion of count_flag to increment SRAM address pointer and write magnetic sensor X axis data to SRAM. Wait for assertion of dm_done signal.
Y_AXIS_COMMAND	Read magnetic sensor Y axis value. Set direct mode input channel number of ADS7870 to LN1 (dm_num = "001"). Specify the number of direct mode conversions to perform (dm_conv_cnt = "11"). Assert dm_start signal. Assign SRAM address pointer.
WAIT_Y_AXIS	Search for assertion of count_flag to increment SRAM address pointer and write magnetic sensor Y axis data to SRAM. Wait for assertion of dm_done signal. If mag_loop = "0" (complete SET sequence), set mag_loop = "1" and go to NEGATE_MAG_MODE state to complete RESET sequence. If mag_loop = "1" (complete both SET and RESET sequences) go to NEGATE_SRSD state.
NEGATE_SRSD	Negate sr_sdn, mag_set, and mag_reset signals. Set SRAM address pointer to write temperature data.
ASSERT_TEMP_EN	Assert temp_go signal to temperature interface logic. Assert temp_sel mux select signal. Increment SRAM address pointer.
WAIT_TEMP	Deassert temp_go signal. Wait for temp_done signal to be asserted. Write temperature data to SRAM. Check if the specified number of temperature data captures have been performed.
NEGATE_5VEN	Negate five_v_enn signal. Assert done_flag to assign DONE bit in control register.

Temperature Sensor Interface

Refer to "References" on page 30 for more information on the Maxim MAX6577 temperature sensor implemented in this design. The MAX6577 can be configured such that the frequency output is 4X the normal frequency output. The output frequency of the sensor correlates to the temperature in degrees Kelvin. By using a clock source to count the frequency output of the MAX6577, the temperature can be determined.

In this design an external oscillator with a frequency of 1.2288 MHz is used. This provides a precise reference when counting the number of clock cycles in one temperature output period.

After the temp_go signal is asserted, the temperature interface searches for a falling edge on the output of the MAX6577. A counter is then enabled until the next falling edge of the MAX6577 output. The temperature range of the MAX6577 is shown in Table 8 with the corresponding period of the output signal.

Table 8: **MAX6577 Temperature Range**

Output Temp (K)	Output Temp (°C)	Output Frequency	Period (1X)	Period (4X)
233K	-40°C	233 Hz	4.3 ms	1.07 ms
398K	125°C	398 Hz	2.5 ms	0.623 ms

The MAX6577 temperature output can be measured by counting the number of clock cycles of the 1.2288 MHz clock. The number of clock cycles are counted for one full period of the

MAX6577 output. An 11-bit counter will allow the count values to represent the full range of temperature values as shown in Table 9.

Table 9: **Temperature Counter Values**

Period of Temp Signal	Counter Value (Stored in SRAM)	Frequency = 1/Period	Frequency / 4
1.07 ms	1315	934.6 Hz	233.6 Hz
0.623 ms	766	1605.14 Hz	401.2 Hz

Figure 7 illustrates the temperature sensor interface state machine logic. Once the temp_go signal is asserted, the WAIT_ZERO and WAIT_EDGE states search for a valid falling edge on the temperature sensor output, temp_out. Once a valid falling edge is detected, an 11-bit counter is enabled. The counter keeps counting until the next falling edge of temp_out. In the DONE state, the temp_done signal is asserted to the higher level control logic.

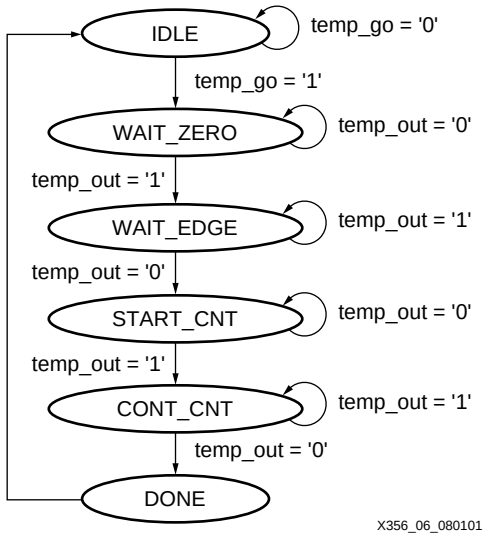

Figure 7: **Temperature Interface State Machine**

Top Level Module Design

The top level module is responsible for the following functions:

- Interface to the Handspring Springboard Expansion Slot (more information is available from XAPP147, see "References" on page 30)
- Interface to the Toshiba 4-Mbit SRAM (more information is available from XAPP147, see "References" on page 30)
- Decode a Handspring read/write with the control register and an SRAM address
- Assert the 9V battery enable, batt_on
- Assert the pressure range control bits
- Assert the calibrate and execute mode flags to the XPATH control logic (op_mode and calb_mode signals)
- Set and clear the DONE bit in the control register
- Reset the OPMODE and CALBMODE bits in the control register
- Decode and assert the system reset signal

Figure 8 illustrates the top level module interface signals.

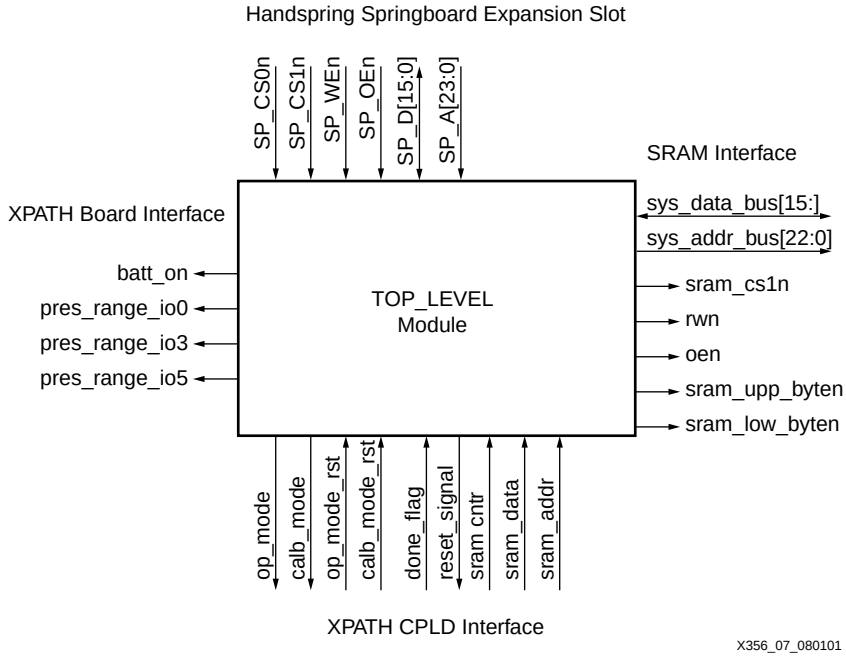

Figure 8: **Top Level Interface Signals**

CoolRunner Implementation

The XPATH design was implemented in VHDL as described above. Xilinx Project Navigator was used for compilation, fitting, and simulation of the design in a CoolRunner CPLD. Xilinx Project Navigator, which includes the ModelTech simulation tool, is available free-of-charge from the Xilinx website at www.xilinx.com/products/software/webpowered.htm. The design was targeted for a 3.3V, 256 macrocell CoolRunner XPLA3 CPLD (XCR3256XL-CS280). The XPATH design utilization is shown in Table 10. These utilizations were achieved using certain fitting parameters, other implementation results may vary.

Table 10: **XPATH XPLA3 256-Macrocell Utilization**

Resource	Available	Used	Utilization
Function Blocks	16	16	100%
Macrocells	256	233	91.02%
Product Terms	768	541	70.45%
I/O Pins	160	108	67.50%

The XPATH VHDL design has been verified through simulation using ModelSim XE in Project Navigator and actual implementation in hardware. The Insight Springboard Kit was used for design verification and functionality testing.

Handspring PocketC Design

The Handspring code development was accomplished using PocketC. PocketC is the hobbyist platform for PDA development. PocketC allows the developer to create a standalone PRC file that can be used on a Handspring PDA.

The software development for XPATH includes the following functions:

- Allow user to read information on the XPATH board from software menus
- Read data values from SRAM

- Display data from SRAM
- Initialize a hardware calibration or execute sequence
- Read/write to/from the CPLD control register
- Calculate altitude
- Convert data to display multiple data units
- North calibrate the digital compass

Figure 9 illustrates the initialization software operational flow. A database is utilized in this application to retain user settings on successive application launches. When the XPATH application is launched, a splash screen will appear to the user during initialization. The XPATH information is presented to the user in the message_loop() function in Figure 10.

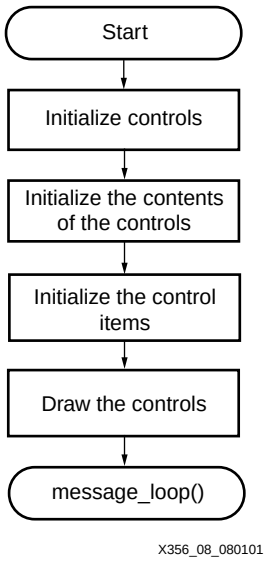

Figure 9: **Initialization Operational Flow**

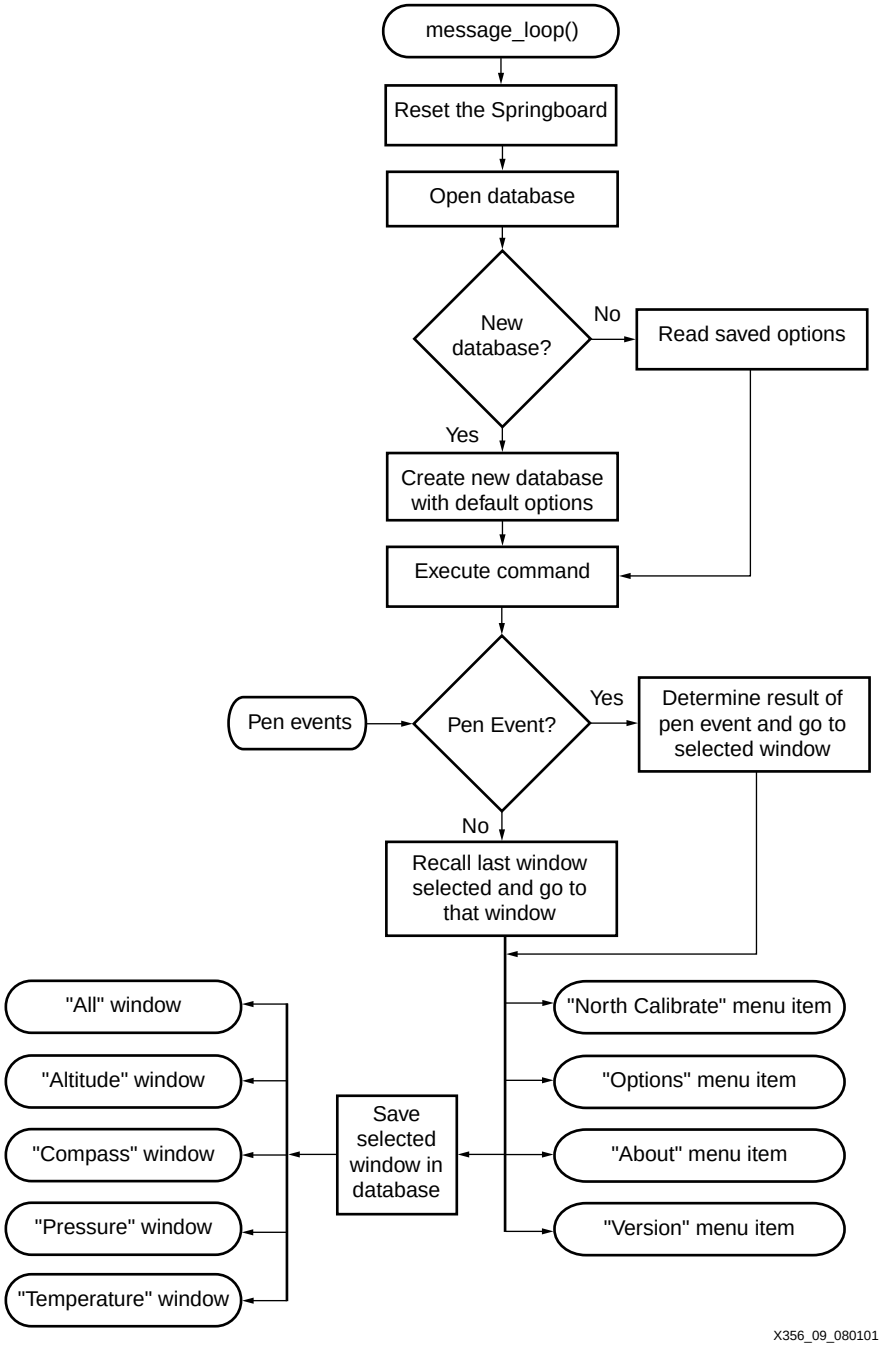

Figure 10: **Message_loop() Operational Flow**

Figure 10 shows the menu capabilities available to the user. The user can select under the View menu to view: "All", "Altitude", "Compass", "Pressure", or "Temperature". The user can also select to perform a "North Calibration" or set "Options" for the application. Information about the XPATH application is available under the "About" or "Version" menu tabs.

If no pen events were detected, the flow will continue to display the current window. This is implemented by the appropriate function call. If a pen event occurs, the flow will change to the corresponding function. All functions will return to "Pen events" in the message_loop() function to again check for a new pen event.

When the user selects to view all XPATH data, the "All" window is selected. The "All" window displays data for the digital compass, temperature, pressure, and altitude as shown in Figure 11. The "All" window also displays the voltage level of the system battery. In the "All"

window graphical user interface (GUI), the user can select to view data from each sensor. This is accomplished by recognizing a "pen event" from the user in the desired area of the GUI. This will take the user to a GUI dedicated to display that data; transferring control to the altitude, temperature, compass, or pressure window.

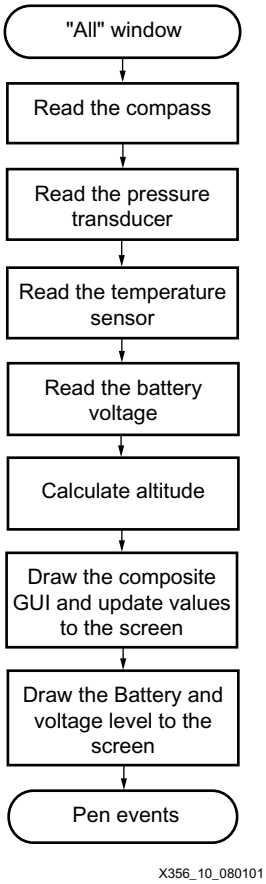

Figure 11: **"All" Window Description**

The Handspring Visor shown in Figure 12, is running the XPATH application and the "All" window view is selected by the user.

Figure 12: **Handspring "All" Window GUI**

When the "Altitude" window is selected, the calculated altitude is displayed. The Handspring Visor reads data from the pressure and temperature sensors. The Handspring then calculates the altitude from the pressure and temperature data values as shown in Figure 13.

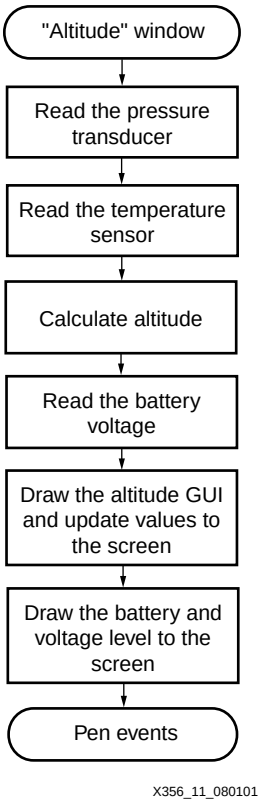

Figure 13: **"Altitude" Window Description**

When the "Compass" window is selected, the magnetic sensor data is captured in an Execute command by the Handspring Visor. The Execute command will capture three sets of data to be averaged by the Handspring. When reading the magnetic sensor, both X axis and Y axis set and reset data are captured. Refer to "Magnetic Field Sensor" on page 3 for more information

on the magnetic sensor data algorithm. The operational flow for displaying the compass data is shown in Figure 14.

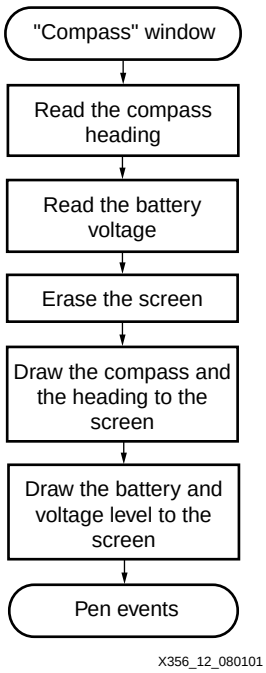

Figure 14: **"Compass" Window Description**

Figure 15 illustrates the software GUI on the Handspring Visor when selecting the "Compass" window view.

Figure 15: **Handspring "Compass" Window GUI**

When the "Pressure" window is selected, the pressure is displayed. The operational flow for displaying the pressure value is shown in Figure 16. The units for displayed pressure can be modified to in Hg, kPa, and mbar. The Execute command will capture three pressure values

from the pressure sensor that are averaged by the Handspring. For more information on the pressure sensor, refer to "Pressure Sensor" on page 3.

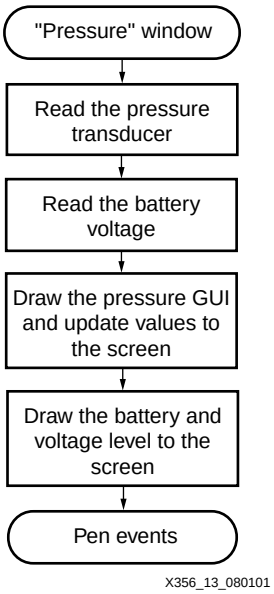

Figure 16: **"Pressure" Window Description**

When the user selects to view only temperature, the "Temperature" window is displayed. The temperature units displayed include °C, °F, and K and can be modified under the Options menu tab. For more information on the temperature sensor, refer to "Temperature Sensor" on page 4. Figure 17 shows the Handspring operation for displaying temperature values to the user.

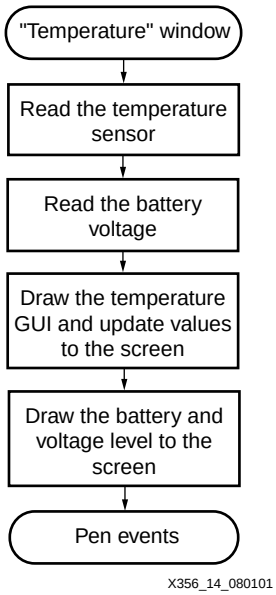

Figure 17: **"Temperature" Window Description**

Figure 18 illustrates the Handspring GUI when the user selects to display the "Temperature" window.

Figure 18: **Handspring "Temperature" Window GUI**

Under the "Options" tab in the XPATH tool, "North Calibrate" can be selected by the user. The "North Calibrate" option will set the North compass bearing as a reference point. This option is used only at the factory, prior to customer shipment. To calculate the compass direction, a reference point is needed and the operations required for this operation are shown in Figure 19.

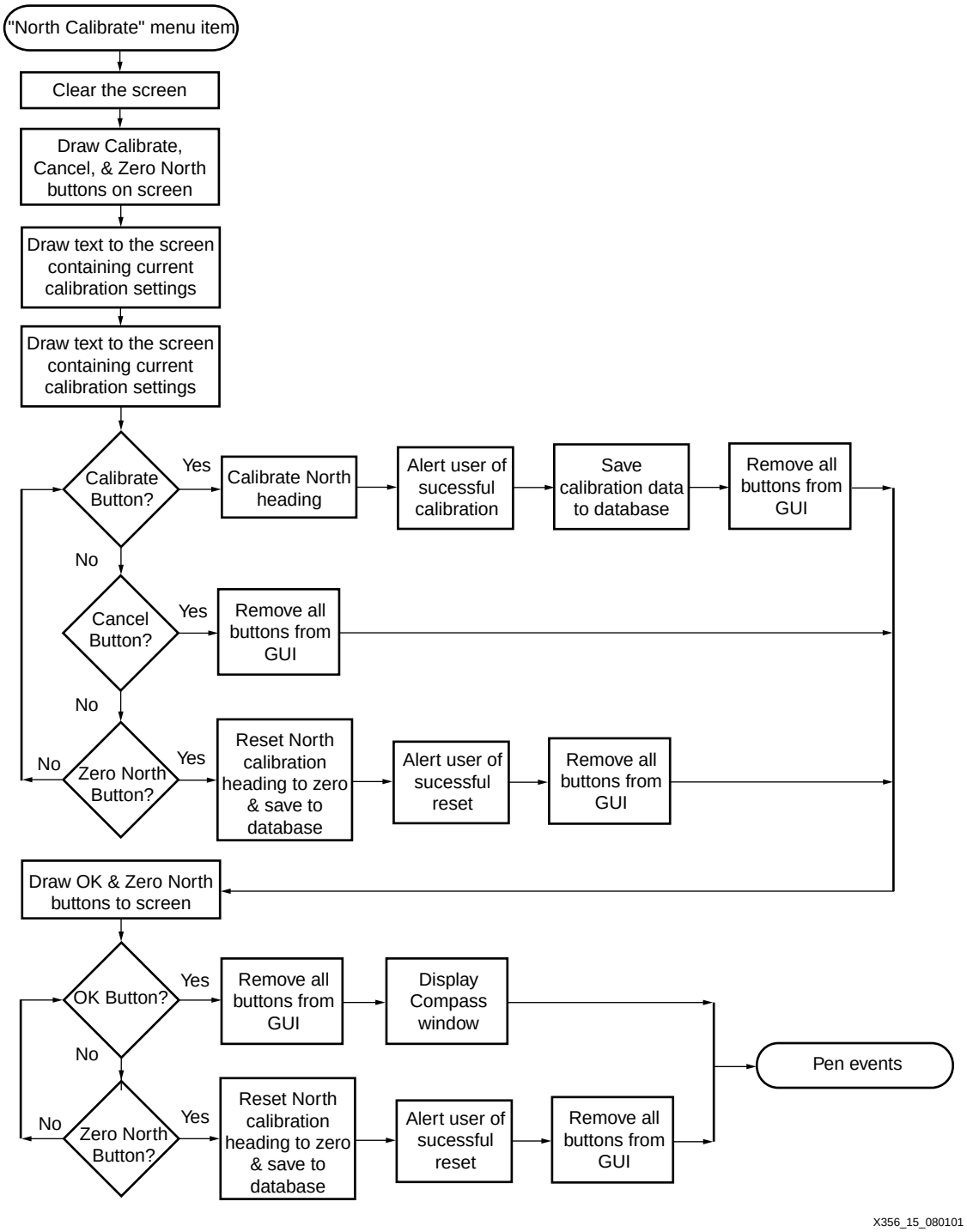

Figure 19: **"North Calibrate" Menu Description**

The operational flow to execute the "Options" menu is shown in Figure 20. Under the "Options" menu tab, the user can change the temperature units, pressure units, and altitude units that are displayed in all windows selected. Once the units are selected by the user, these options are saved in the current application database.

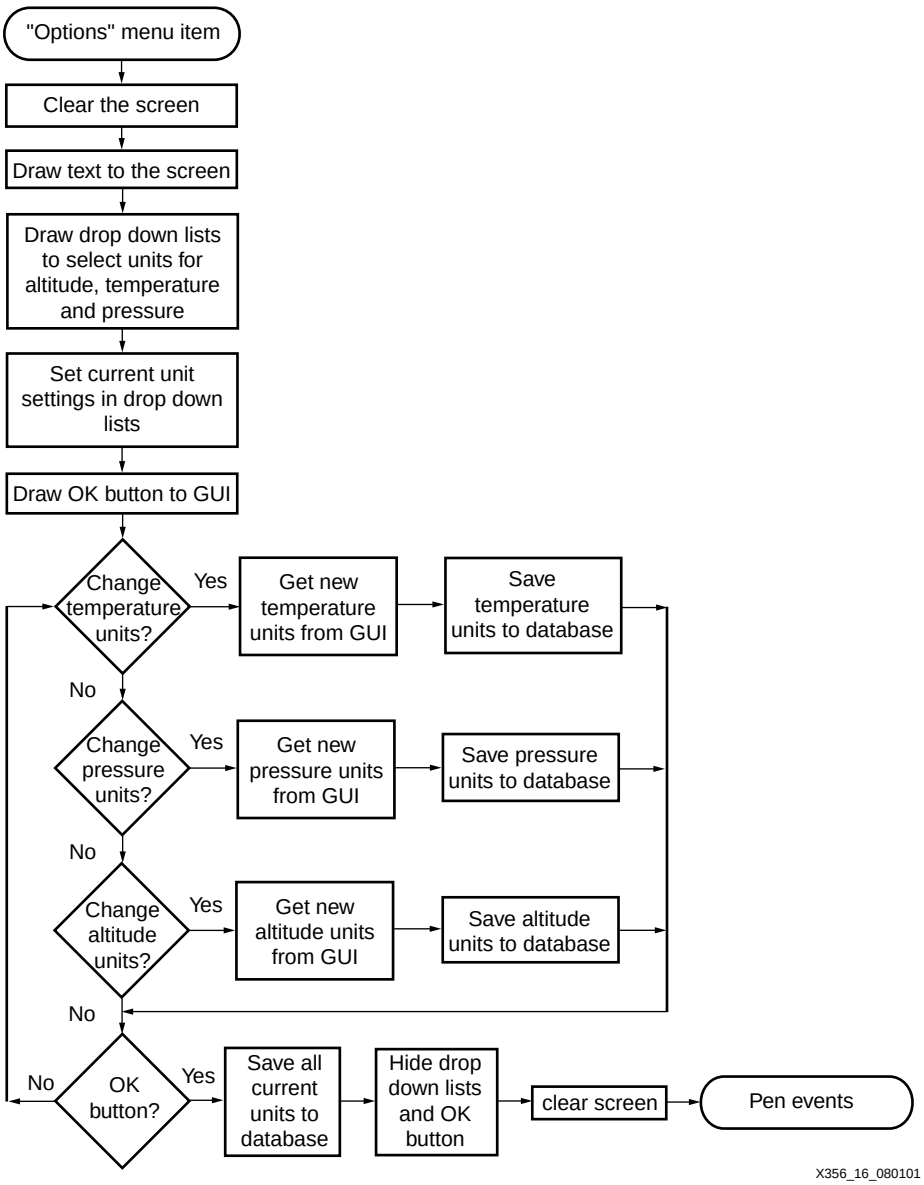

Figure 20: **"Options" Menu Description**

The "About" menu tab will display pertinent information about the XPATH application board. This tab has two text screens that are displayed with navigational buttons. Figure 21 shows the operational flow when the user selects this menu tab.

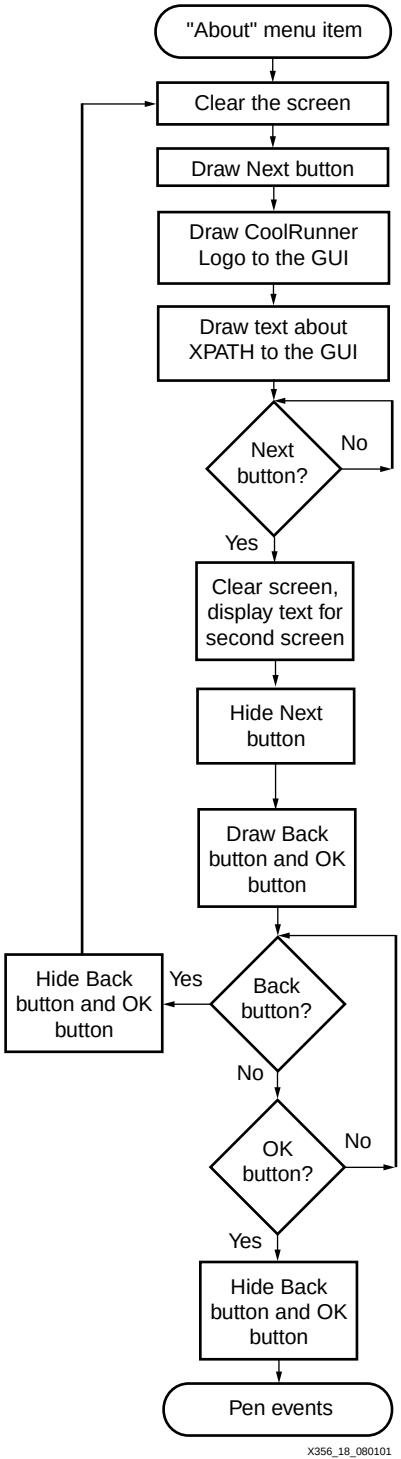

Figure 21: **"About" Menu Description**

The "Version" menu tab will display revision information for the XPATH board. To exit from the test screen, the application waits for the user to select the OK button as shown in Figure 22.

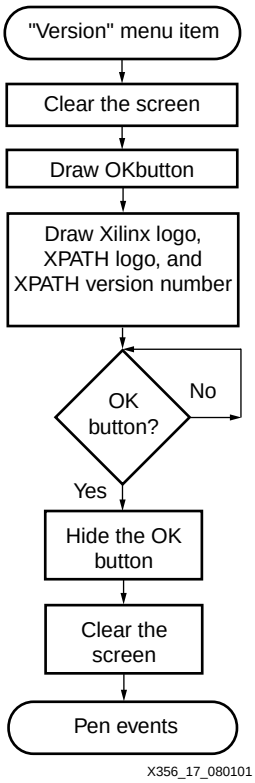

Figure 22: **"Version" Menu Description**

HDL and C Code

THIRD PARTIES MAY HAVE PATENTS ON THE CODE PROVIDED. BY PROVIDING THIS CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS DESIGN, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THIS DESIGN IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGN "AS IS" AS A COURTESY TO YOU.

XAPP356 - **http://www.xilinx.com/products/xaw/coolvhdlq.htm**

Conclusion

The XPATH Handspring add-on module design is provided as a reference design for PDA module development. The XPATH board provides digital compassing, barometric pressure reading, temperature sensing, and altitude display.

References

The web sites shown here are valid as of the publication date of this note.

1. XAPP147: Low Power Springboard Design Using CoolRunner CPLDs (http://www.xilinx.com/contest/reference.htm)
2. XAPP355: Serial ADC Interface Using a CoolRunner CPLD (http://www.xilinx.com/contest/reference.htm)
3. Xilinx and Handspring Cool Module Reference Design website (http://www.xilinx.com/contest/reference.htm)
4. Handspring Developers (http://www.handspring.com/developers/index.jhtml)
5. Orbworks PocketC (http://www.viaweb.com/pilotgearsw/orbworks.html)
6. Insight Springboard Development Board (http://www.insight-electronics.com/solutions/kits/xilinx/springboard.html)
7. Texas Instruments Burr Brown ADS7870 Data Acquisition System Data Sheet (http://focus.ti.com/docs/prod/productfolder.jhtml?genericPartNumber=ADS7870)
8. Honeywell HMC1002 Magnetic Field Sensor Data Sheet (http://www.ssec.honeywell.com/magnetic/index.html)
9. Motorola MPX4115A Pressure Sensor Data Sheet (http://e-www.motorola.com/brdata/PDFDB/SENSORS/PRESSURE/MPX4115A.pdf)
10. Maxim MAX6577 Temperature Sensor Data Sheet (http://pdfserv.maxim-ic.com/arpdf/MAX6576-MAX6577.pdf)
11. Circuit Cellar Article. "An Altimeter for the Traveling Man". Radek Vaclavik. Issue 127. Febuary 2001. (http://www.circuitcellar.com/)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/06/01	1.0	Initial Xilinx release.