



APPLICATION NOTE 3885

Using the MAXQ3210 As A Water Sensor/Alarm System

Abstract: The MAXQ3210 is a high-performance, low-power 16-bit RISC microcontroller ideally suited to environmental monitoring and alarm systems. Its internal 5V to 9V voltage regulator, wake-up timer, Stop mode, and ring oscillator allow very low-power operation. The integrated analog comparator, piezoelectric horn driver, and precision voltage reference minimize component count. This application note describes how the MAXQ3210 can be used to build a system that detects water and then sounds an alarm. The complete assembly language application software is provided.

Overview

The [MAXQ3210](#) is a powerful RISC microcontroller with features and capabilities that make it ideally suited for battery-powered applications that detect a condition and sound an alarm. The microcontroller's integrated 5V to 9V regulator, piezoelectric horn driver, and analog voltage comparator support a minimal component count system. Additionally, various low-power modes, including Stop mode and wake-up timer, allow this microcontroller to operate for significant periods when powered by a 9V battery.

This application note describes a system that uses the MAXQ3210 microcontroller to detect water by using its inherent conductivity caused by mineral content, and then sounds an alarm. While the simple system described here is not intended to be production ready, it illustrates how the MAXQ3210's features and capabilities can be used effectively. The water-detection mechanism used for this example *has not been scrutinized* for reliability under extreme environmental conditions or evaluated for effectiveness over time. It is a simple implementation created solely for demonstration in this application note, and it functioned successfully using tap water.

The code for this application note was written and tested specifically for the MAXQ3210, but can also run on other [MAXQ® devices](#) containing similar resources, the MAXQ3212, for example. The complete development environment for this example code is version 1.0 of the MAX-IDE and revision B of the MAXQ3210 evaluation kit board. Further information on this evaluation kit is available at [MAXQ3210EVKIT](#).

Implementation Details

This application note will demonstrate as many features of the MAXQ3210 as practical. The MAXQ3210's on-board comparator allows a simple water sensor to be implemented with minimal external components. By using the processor's low-power sleep mode and wake-up timer, this battery powered system will remain in Stop mode most of the time, waking the system periodically to test the water sensor to see if an alarm should be sounded.

The following sections describe several MAXQ3210 features and how they are used in this application. Descriptions of the configuration and setup of these features are provided.

Water Sensor Probe

The water sensor probe is shown in **Figure 1**. As mentioned above, this sensor is not a production-ready design. It is fabricated using a piece of scrap plastic material drilled to snugly hold the leads fashioned from a paperclip.

The spacing between these leads is arbitrary and subjectively based on the available materials. To connect the sensor, a 4-pin connector with 0.1 spacing allows it to be plugged directly into connector J4 (pins 9-P0.4, 11-P0.5/CMPI, and 13-P0.6) of the MAXQ3210 evaluation board. A 1.0M Ω resistor (visible in the picture inline with one of the wires near the connector) acts as a pull-up for one lead of the sensor, and is soldered directly to one of the connector pins.

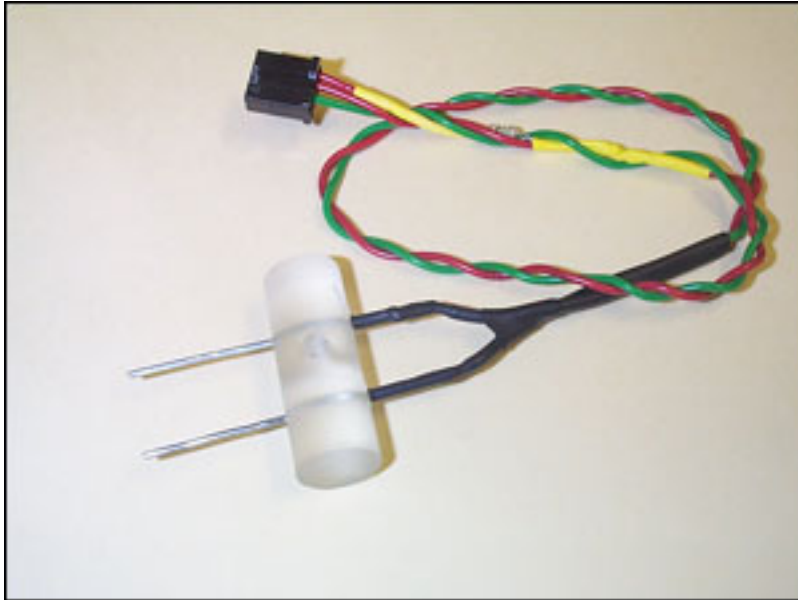


Figure 1. The water sensor probe.

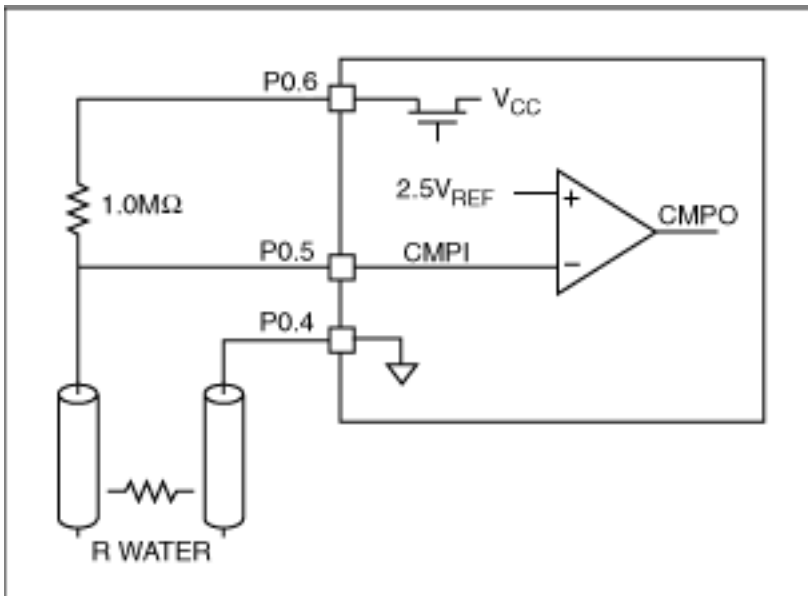


Figure 2. Water sensor schematic.

A schematic of the water sensor probe's connections to the processor is shown in **Figure 2**. One side of the sensor is connected to the MAXQ3210's built-in analog comparator's input CMPI, P0.5. This input is also connected to the 1.0M Ω resistor, which is, in turn, connected to the processor's port pin, P0.6. The software configures P0.6 as an output and sets it high as part of the system initialization. Because of the comparator's high input impedance, this configuration pulls CMPI up close to V_{CC} under normal conditions (i.e., sensor leads not immersed). The other side of the sensor is connected to port pin P0.4 which is configured as an output pin and set low. When the two sensor leads are immersed, the water's conductivity forces the comparator's input to be pulled down toward ground. When this occurs, the comparator's output CMPO changes states. A more thorough description of the analog comparator and its operation is provided in a later section of this document.

Importance of Stop Mode

Except for a power-off state, Stop mode is the lowest power consumption configuration possible for the MAXQ3210. Stop mode disables all circuits in the processor except the ring oscillator and wake-up timer (if enabled). All other on-chip clocks, timers, and peripherals are halted, and no code execution occurs. Once in Stop mode, the MAXQ3210 is mostly in a static state, with power consumption determined primarily by leakage currents. By using Stop mode in conjunction with the wake-up timer, very low-power operation is achieved.

In a real-world environment, one normally expects a change in water level to occur relatively slowly. Therefore, the processor can spend most of its time in Stop mode and wake up infrequently for brief periods of time to check the water sensor. In this example, one minute is chosen as the period between sensor samples. This interval maximizes battery life by spending as much time as possible in Stop mode without sacrificing a sufficient warning for rising water. If this period is too long or short for a specific application, the software's constant wake-up delay (WUDel) can be changed to the desired value, and the program then reassembled. The equation for calculating this time interval is given by the Wake-Up Timer Period equation in the section below.

Stop mode is entered immediately when the STOP bit of the processor's Clock Control register, CKCN.4, is set to 1. Stop mode will be exited if any of the following conditions occur:

- External active-low reset on P1.1/active-low RESET (if not disabled)
- Power-on reset (if not disabled)
- External interrupt on P0.6/INT (if enabled)
- The wake-up timer reaches zero (if enabled) and its interrupt is acknowledged

Exiting Stop mode after the wake-up timer interrupt does not affect the processor's configuration, including the settings of the clock control bits, for example. This is, however, not the case if an external reset initiates an exit from Stop mode. In this latter case the processor will return to its default reset condition. Therefore, before entering Stop mode, the processor should be initialized to its normal operating condition; it will return to this configuration when Stop mode is exited. Besides bringing the processor out of Stop mode, the wake-up timer interrupt and its Interrupt Service Routine (ISR) initiates all other system functions (e.g., test the water sensor, sound the horn, test for low-battery voltage, etc.).

Wake-Up Timer

The MAXQ3210's wake-up timer is a 20-bit timer that can be configured to count down using system clocks or, alternatively, cycles of the processor's internal ring oscillator. The application software loads the initial value into the wake-up timer register (WUT), and the timer counts down from this value. When zero is reached, the time-out period is complete, and the interrupt flag bit (WTF) of the Wake-Up Timer Control (WTCN.1) register is set. If enabled, this flag causes an interrupt, which causes an exit from Stop mode. If the interrupt is disabled, no exit from Stop mode can occur.

The upper 16 bits of the 20-bit timer are accessible to the processor and its software through the WUT register, while the lower 4 bits are only accessible to the timer hardware. When, however, the WUT register is written by software, these lower 4 bits are cleared to zero. The wake-up timer period is given by the equation:

Wake-up timer period = (source clock period) x WUT[19:4] x 16

where WUT[19:4] are the upper 16 bits of the 20-bit timer. Note that the period is multiplied by sixteen to account for the lower 4 bits of the timer not contained in the WUT register. Using this equation, it can be seen that for a typical ring oscillator frequency of 8kHz, a maximum wake-up period of approximately 131 seconds is possible. As discussed above, a time-out period of one minute was selected. Loading a countdown value of 30,000 (07530h) into WUT results in a time-out value of one minute. This action presumes that the wake-up timer is set to run from the ring oscillator.

Configuring the wake-up timer for this application requires a single write to the timer's control register. The timer is enabled by setting the Wake-Up Timer Enable (WTE) bit, WUTC.0. The Wake-Up Timer Clock Select bit

(WTCS), WUTC.2, must also be set to 1 to cause the timer to run from the processor's ring oscillator. Therefore, the application software must write the value 05 hex to the Wake-Up Timer Control register (WUTC) to initialize the timer. The Wake-Up Timer Flag (WTF) bit, WUTC.1, is set by the timer's hardware, but must be cleared by the interrupt service routine to prevent further acknowledgements of the same interrupt.

Analog Comparator

The MAXQ3210's internal 1-bit, analog-to-digital comparator and its related 2.5V voltage reference are key to this application. The comparator has two inputs, + and -, as shown in Figure 2 above. The comparator's output is a function of the difference of the analog voltages applied to these two inputs. In this application, the on-board 2.5V reference is connected to the '+' input and the '-' input is connected to one side of the water sensor. As the figure shows, the '-' input is also pulled high through a 1.0M Ω resistor by port pin P0.5 which is set to 1. Therefore, in its normal state, the '-' input will be close to the 5V supply making it higher than the 2.5V reference on the '+' input. The comparator's polarity select (CPOL) bit, CMPC.1, is set to 0 in this application. The resulting output CMO follows:

CMO = 0 when ($V_{REF} < CMPI$)

CMO = 1 when ($V_{REF} > CMPI$)

Therefore, it can be seen that the normal state of the comparator's output CMPO is 0. When water sensor leads become submerged, the conduction between the two leads pulls the comparator's input toward ground. In this state, the reference voltage is greater than the CMPI voltage, and the output CMO goes high. Due to its high input impedance, very little current flows into the comparator's input under normal (inactive) conditions. When the water sensor leads are submerged, the water's conductance and the 1.0M Ω resistor limit the current flow between the sensor leads.

Horn Driver

The MAXQ3210 provides an on-board, three-pin, piezoelectric horn driver interface that can directly drive a piezoelectric horn. The three-pin interface consists of the following pins:

- HORN B (Horn Brass): This output is connected to the piezoelectric horn's metal support electrode.
- HORN S (Horn Silver): This output is connected to the piezoelectric horn's ceramic electrode. It provides an output complementary to HORN B when the piezoelectric horn drive is enabled.
- FEED: This input pin is connected to the feedback electrode of the piezoelectric horn.

The piezoelectric horn driver is self-driving and enabled or disabled using the Horn Enable (HRNE) bit in the horn control register, HRNC.0. When this bit is set to 1, the horn driver is activated and a horn will sound. When this bit is cleared to 0, the horn will not sound. In this example application, the horn sounds at regularly spaced intervals of five beeps as long as the sensor leads are submerged. If a low-battery voltage is detected, the horn will sound with eight regularly spaced beeps with a one-minute silence between the groups of eight. This pattern will continue until the battery voltage falls low enough to cause a power-down reset, or until an external reset is applied. (This happens when the external reset is not disabled.)

Ring Oscillator

The MAXQ3210 contains an internal ring oscillator used as the default source for the system clock following any power-on-reset or exit from Stop mode. This ring oscillator begins oscillating almost immediately when enabled, unlike the crystal oscillator which requires a minimum of 65,536 clock cycles to warm up and stabilize. If a function is delayed 65,536 clock cycles when waking up from Stop mode, a significant amount of power is wasted waiting for this period to expire while no instructions are executed (i.e., no work is performed). By using the ring oscillator when exiting Stop mode, all that wasted power is conserved. Actually, the ring oscillator does require four clock cycles to warm up and stabilize when exiting Stop mode, but that is a considerably shorter interval than waiting on the crystal oscillator.

To minimize the power consumed, the initialization software configures the ring oscillator to be the processor's system clock. This is achieved by setting the Ring Select (RGSL) bit, CKCN.6, to 1. When this bit is set while the processor is already running from the crystal oscillator (at system initialization), the clock source is switched to the ring immediately, and there is no four-clock-cycle delay.

The ring oscillator's target operating frequency is 8kHz, but the actual frequency can vary from one device to another. The frequency varies over temperature and supply voltage as well, so this variability should be considered in applications where precise timing is required. In this application, the exact frequency is not important.

Because the crystal oscillator does not run when the processor is in Stop mode, the wake-up timer must be set to run from the ring oscillator (WTCS = WUTC.2 = 1) in this application. It is possible to run the processor from the crystal oscillator while the wake-up timer runs from the ring oscillator. However, this approach was not used in this application because of timing differences during read/write to/from the wake-up timer registers.

Low-Battery Detection

The MAXQ3210 is equipped with low-battery detection circuitry. By setting the low-battery detection enable bit (LBDE), PWCN.1, to 1, the low-battery interrupt flag (LBF), PWCN.3, will be set by the processor's hardware whenever the power supply input, V_{DD} , drops below the low-battery threshold, V_{BF} . This flag can cause an interrupt if enabled, but the interrupt is not used in this application. The flag bit is polled every time the processor exits Stop mode and tests the water sensor. If the battery is low, the horn beeps eight times with a one-minute interval between each series of eight beeps.

Evaluation Board Considerations

The software for this application note was written and tested using the MAXQ3210 evaluation kit board. When working in this development environment, several factors should be considered. The first consideration is running the processor from the ring oscillator. Communication with the evaluation board is performed through a serial-to-JTAG board by the debug/visibility JTAG bus on the processor. The JTAG clock cannot be faster than 1/8th the processor's clock. Because running the processor from the ring oscillator violates this rule, the JTAG board cannot communicate with the evaluation board. When the JTAG board does not receive the appropriate information from the evaluation board, the PC software assumes that the communication failed. When this occurs, the PC displays an error message, and the debugger hangs-up. This situation can be eliminated by inserting a long delay loop in the code prior to switching the clock source to the ring oscillator. The ring select bit, RGSL, is cleared to 0 on a power-up reset. By inserting this delay, the debugger has the time to gain control of the board before the ring oscillator engages. The code to provide this delay was 'commented out' in the source file, but was left in the file as an example of this solution.

Another consideration is the MAX5160LEUA digital potentiometer included on the board. This device is connected to the voltage comparator's input, CMPI, when jumper J11 is installed; it is intended to provide a convenient mechanism for applying a variable voltage to this input. One end of the potentiometer's internal resistor chain, H, is connected to the board's V_{CC5} supply; the other end, L, is connected to ground, and the wiper, W, is connected to CMPI. This digital potentiometer has an end-to-end resistance of 50k Ω from H to L, and therefore represents a significantly lower impedance than the normally very high impedance (FET input) of CMPI. Therefore, jumper J11 should be removed from the board in this application. This removes the digital potentiometer from CMPI, and allows the high impedance of this input to dominate.

Although not critical for this demonstration application, in a real-world environment the lifetime performance of different battery brands and types can vary significantly. The MAXQ3210's low-battery detection voltage is designed to be approximately 7.2V, which is well suited for most alkaline batteries and environments. A new, heavy-duty, 9V alkaline battery will sound the alarm for an acceptable period of time after low-battery voltage is detected; other battery types and extreme environments can shorten the alarm period significantly.

The combination of battery type and anticipated extreme environments should also be considered and verified for any production-ready design. Using the provided application software, the board's horn will sound indicating

a low battery until the battery falls low enough to cause a power-down reset. At this point, the battery will be depleted, and will need to be replaced for continued operation.

For testing this application note, resistors R1 and R2 were removed from the board to conserve battery power. These resistors are used as current-limiting resistors for LEDs D1 and D2 respectively. These LEDs are not needed for this application.

Example Code

The software accompanying this application note is contained in the [an3885_sw.zip](#) file on the Maxim ftp web site at ftp://ftp.dalsemi.com/pub/microcontroller/app_note_software/. This file contains the source code file `Alarm.asm` and the file `maxq3210.inc` that contains the MAXQ3210's register address definitions. Also included in this .zip file is the MAX-IDE project file `Alarm.prj` and the 'load-able' hex file `Alarm.hex`. By extracting these files to a common directory, the software can be assembled and executed on the MAXQ3210 evaluation kit.

Conclusion

The MAXQ3210 microcontroller contains a number of features that make it ideally suited to cost-conscious, battery-powered applications such as chemical detectors, alarm systems, and white goods, but it can also be used in any application that requires high-performance and low-power operation. Internal features such as a 5V to 9V regulator, piezoelectric horn driver, and analog voltage comparator support an implementation with a minimal component count. Additionally, features such as its internal 8kHz ring oscillator, low-battery detection circuitry, 20-bit wake-up timer, and low-power Stop mode support low-power operation and permit significant run times when powered from a single 9V battery.

To discuss your experiences and additional applications for the MAXQ microcontrollers with colleagues and experts, visit the Maxim Discussion Board for microcontrollers: <http://discuss.dalsemi.com/>.

Application note 3885: www.maxim-ic.com/an3885

More Information

For technical support: www.maxim-ic.com/support

For samples: www.maxim-ic.com/samples

Other questions and comments: www.maxim-ic.com/contact

Automatic Updates

Would you like to be automatically notified when new application notes are published in your areas of interest? [Sign up for EE-Mail™](#).

Related Parts

MAXQ3210: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)

MAXQ3212: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)

AN3885, AN 3885, APP3885, Appnote3885, Appnote 3885

Copyright © by Maxim Integrated Products

Additional legal notices: www.maxim-ic.com/legal