# App Note 3537: Environmental Monitoring with the MAXQ3210

*The MAXQ3210's capabilities make it unique in both the MAXQ family and the embedded microcontroller market. The MAXQ3210 integrates EEPROM code and data storage, a piezoelectric horn driver, and a 9V regulator into a low-pin-count package. A high-performance 16-bit RISC core makes the device both fast and power-friendly. Based on the MAXQ10 core, the MAXQ3210 differs from other MAXQ microcontrollers as it has 8-bit accumulators rather than 16-bit accumulators. The MAXQ3210 can be used in many applications where a few I/O pins and some smart control are required. This article describes some environmental-monitoring applications that are ideal for the MAXQ3210.*

The new MAXQ3210's capabilities make it unique in both the MAXQ family and the embedded microcontroller market. The MAXQ3210 integrates EEPROM code and data storage, a piezoelectric horn driver, and a 9V regulator into a low-pin-count package. A high-performance 16-bit RISC core makes the device both fast and power-friendly. Based on the MAXQ10 core, the MAXQ3210 differs from other MAXQ microcontrollers as it has 8-bit accumulators rather than 16-bit accumulators. The MAXQ3210 can be used in many applications where a few I/O pins and some smart control are required. This article describes some ideal environmental-monitoring applications.

## MAXQ3210 Features and Monitoring Capabilities

The MAXQ3210 has 2kB of EEPROM code space, 128 bytes of EEPROM data space, and 64 bytes of RAM. An integrated 9V regulator simplifies the circuit in battery-powered applications. The MAXQ3210 also provides a regulated 5V output for other circuit components. A JTAG debug engine allows in-application debugging without an expensive emulator.

The MAXQ3210 integrates unique peripherals that can be used in environmental-monitoring applications. A piezoelectric horn driver and high-current LED driver provide immediate status feedback when an environmental condition is unsafe or changing. These peripheral capabilities are useful in many monitoring applications; simple security systems, smoke alarms, temperature monitors, and motion detectors all have a place for a microcontroller that drives an electric horn.

Additionally, the device provides multiple options for interfacing to environmental-monitoring circuits. The MAX3210's internal analog comparator monitors voltage changes in external circuits, which occur as a result of environmental changes. This external circuit could be something as simple as a thermistor measuring temperature, or something more complex such as a slope analog-to-digital converter (ADC) that measures the amount of time a current takes to charge a capacitor.

Another option for monitoring external circuits is through the MAXQ3210's digital I/O. When an out-of-range condition occurs, for example, the environmental-monitoring circuit generates an external interrupt that awakens the MAXQ3210. The MAXQ3210's I/O pins could also use a serial transmission protocol to communicate data with an external IC that measures distance or lighting conditions.

## Software Architecture for a Monitoring Application

Applications written for the MAXQ3210 are generally small and simple enough to be coded in the MAXQ assembly language. For the example application presented later in this article, the MAX-IDE toolset is used. MAX-IDE is a free development environment from Dallas Semiconductor, providing an assembler and a debugging environment for MAXQ devices. **Figure 1** shows the basic architecture for an environmental-monitoring application.

On startup, the device passes through an initialization period in which registers and configuration bits are set for general application use. If the device was just powered on, extra operations may be required, such as manufacturing test and configuration. After passing through initialization and power-on check, the application enters the main loop where it measures and reacts to its environment. First, environmental readings are taken through the comparator or the digital I/O pins, and then analyzed for out-of-range conditions. Next, the application performs periodic diagnostics, which may include testing external circuits, measuring the battery, or checking for permanent faults recorded in the data EEPROM. Following the diagnostics, the application checks the status, which can range from warnings (low battery) to alert conditions (temperature too high). When the environmental readings require action, the application has several options that we discuss below: sound a horn, flash an LED, use the I/O pins to communicate with another device, or simply record the condition into the data EEPROM for later analysis.

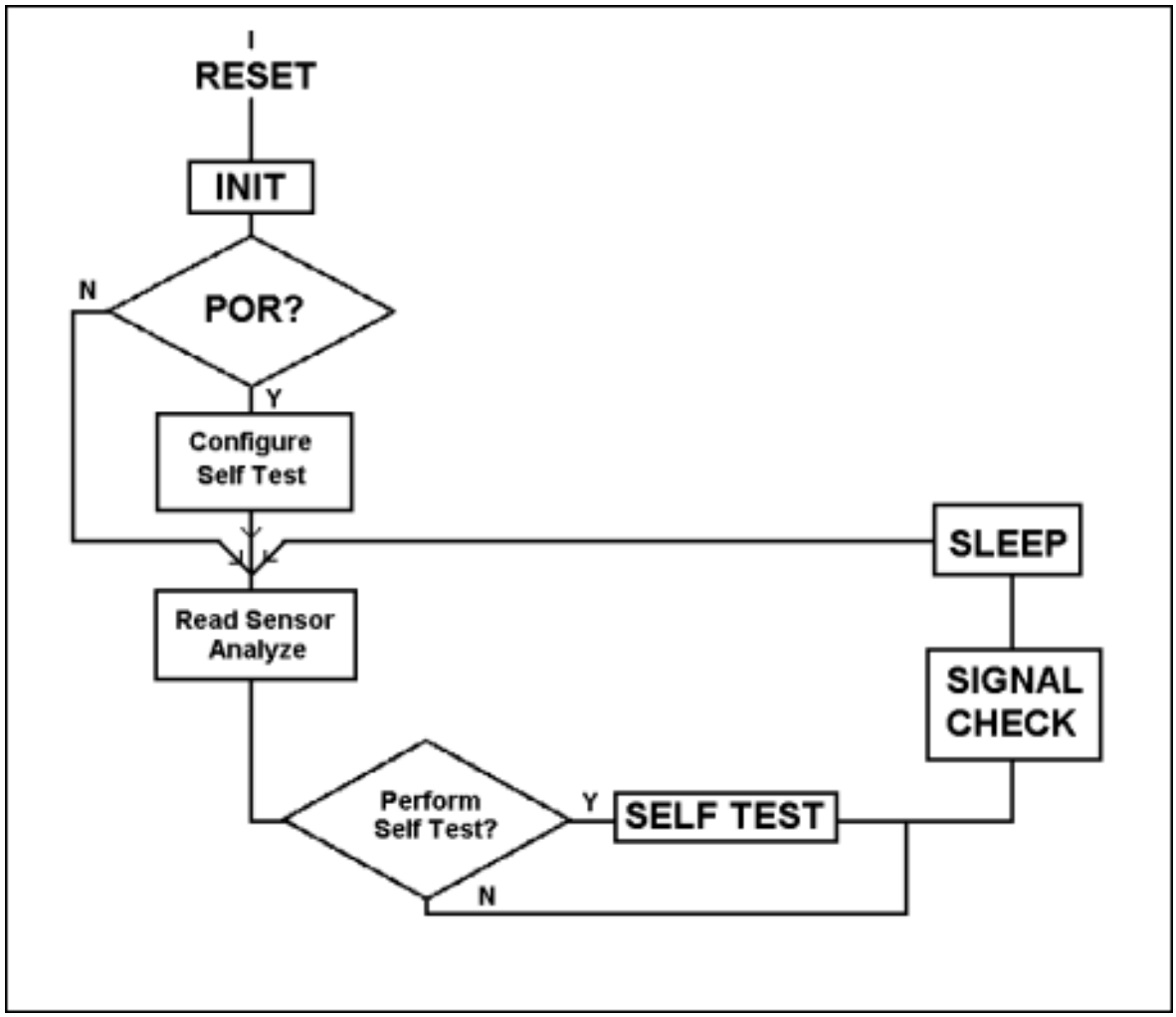


*Figure 1. The MAX3210's main program loop in an environmental-sensing application spends most of its time in sleep mode, waking up periodically to read a sensor and analyze the results.*

## Software for a Simple Monitoring Application

A simple application that models an environmental monitor is available for download. It was built and tested on the MAXQ3210 evaluation kit. A pushbutton toggles between alarming and normal conditions. The horn sounds to indicate an alarm.

The main loop of the environmental-monitoring application appears in the following paragraph. Notice that the state machine for an environmental monitor is very simple; it takes sensor readings and analyzes them to see if the system has exceeded some threshold (temperature too hot, too much smoke in the air, etc.). If the condition is out of bounds, an alarm signals.

```
MainLoop:
    move  DP[0], #CONDITION_FLAG  ; see if we are alarming
    move  ACC, @DP[0]             ; read the alarm flag
    jump  z, MainLoop_NoSignal    ; skip next code if not alarming

    ;
    ; If our condition is above threshold, see if it is
    ; time to sound the horn
    ;
```

```
    call  CheckSignalTime         ; see if it is time to sound the horn
    jump  nz, ReadAndSleep        ; back to sleep if no signal
    call  SignalCondition         ; sound horn, light LEDs, etc.
    jump  ReadAndSleep            ; let's go to sleep now
    ;
    ; In a real sensor, we still want to take readings even if we are
    ; signaling.  We need to check to see if environmental conditions
    ; have returned to normal.
    ;
MainLoop_NoSignal:
    call  CheckForSelfTest        ; time to run periodic diagnostics?
    jump  z, ReadAndSleep         ; skip if not time yet
    call  SelfTest                ; perform self diagnostics

ReadAndSleep:
    call  ReadSensor              ; get a 'sensor reading'
    call  AnalyzeSensor           ; see if condition out of threshold
    jump  Sleep                   ; put the device into low power mode
```

The SelfTest function allows periodic system diagnostics, in which applications could monitor their battery condition or check for misbehaving circuits. SelfTest is also a good place to increment an internal timer to track how long the MAXQ3210 has been active, thereby allowing the external systems with sensors that degrade over time to have a planned end-of-life.

The application code demonstrates how MAXQ peripherals are easy to use, and how they conserve code space and execution cycles. For instance, the horn driver only requires a single bit to activate or deactivate the horn output.

```
SoundTheHorn:
    move  HORN_DRIVER, #1
    move  LC[0], #10
    call  DelayMilliseconds
    move  HORN_DRIVER, #0
    ret
```

## Power Management

Power consumption is one of the most important factors in environmental-monitoring applications, which typically run off a battery. The MAXQ3210 provides a low-power stop mode and a low-voltage battery monitor.

When the application is periodically measuring an environmental condition, the MAXQ3210's low-power stop mode has two options for wakeup: an external interrupt or a wakeup timer that can bring the device out of sleep mode and begin code execution. The external interrupt is a good option when the application is waiting for an external circuit to trigger a condition. Typical examples are waiting for a door to open or the voltage across a thermistor to cross a threshold for the external interrupt.

The wakeup timer is another option for bringing the MAXQ3210 out of stop mode. Wakeup is the function discussed earlier in the demo application: the external monitoring circuit wakes up the MAX3210, which measures the environment, reacts if necessary, and then returns to sleep. **Figure 2** shows the typical current-consumption model for such an application. Most of the microcontroller's time is spent in low-power sleep mode. When the device does wake up, the current consumption is much higher. This is where the high performance of the MAXQ core is useful. The MAXQ3210 performs its computations quickly, spending less time in the high-power consumption state and more time in low-power sleep mode.

Because battery life is a crucial component of most monitoring applications, it is useful to detect when a battery is nearing end of life. The MAXQ3210 determines if battery voltage has fallen below a threshold by simply checking a status bit in a register. This voltage threshold is fixed at 7.7V, which is where 9V batteries begin to break down. At this voltage level, there is ample power left in the battery for the MAXQ3210 to continue to run. A power-conscious application can run for days or weeks on a low battery and issue periodic warning signals, as is commonly done in smoke alarms.

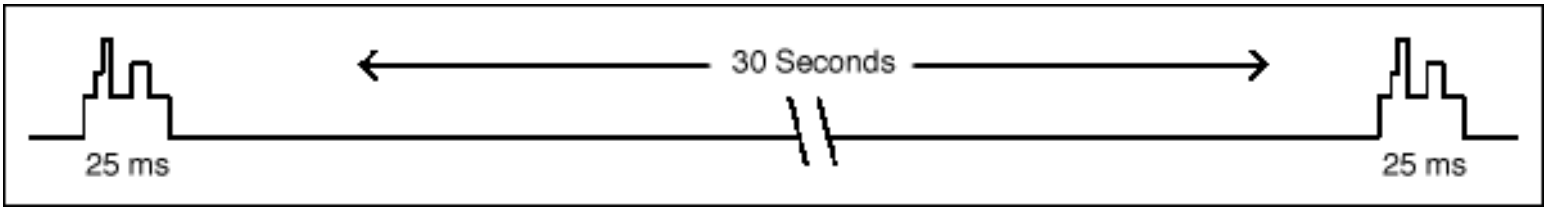


*Figure 2. Monitoring applications sleep most of the time to conserve power, waking up periodically for very short runtimes.*

## Data EEPROM

The MAXQ3210's 128-byte data EEPROM makes an application smarter. It allows applications to keep permanent configuration and status data, even across power failures or battery removals. Permanent data storage is useful for several purposes.

1. Yield improvements. Devices that behave slightly outside specification (for instance, a distance detector that measures a little short) can store permanent configuration information, allowing software to compensate for an external circuit's variations. This allows end devices to be activated or sold that might have previously been discarded.
2. Behavioral configuration and customization. MAXQ3210 applications can be customized for their specific target environments or end users. For example, an environmental-monitoring application might be configured as part of a larger network. When the device's measurement is triggered at some threshold, not only would the microcontroller sound its horn, but it could also toggle port pins to alert other devices about the condition. Factory configuration can enable or disable this network notification.
3. End of life. In an environmental sensor, the circuit measuring the environment might degrade with use. By updating the EEPROM of the MAX3210 as time passes, an application has control over how long it runs before it must be replaced. A sensor, for example, after five years of running can automatically disable itself, signaling with a horn or flashing LED that it is no longer functional.

## Environmental-Monitoring Applications

Some of the more obvious environmental-monitoring applications for the MAXQ3210 are home-safety applications: fire alarms and gas alarms. The MAXQ3210 has all the tools to implement these applications integrated on-chip. The MAXQ3210 is, however, far more versatile than a dedicated smoke-alarm microcontroller. A variety of applications can be created using the simple environment-monitoring software architecture previously discussed. Some of the following examples target safety applications that prevent or minimize damage to businesses or homes. Other applications provide convenience to the consumer.

To prevent damage to the home or office, one application is a water-level monitor for a basement, where a build-up of water might not be noticed for some time. In this case, water is detected with a humidity sensor or a tank apparatus similar to what is used in a toilet. When the water causes the float to rise above a certain point, the float triggers an external interrupt, and the MAXQ3210 sounds an electric horn to alert residents. In addition, the MAXQ3210 communicates the situation to a larger home or business network, which in turn notifies the business or homeowner about the condition.

Temperature monitoring is another potential application. The contents of a supermarket freezer or a refrigerated car on a delivery truck are monitored for excessive heat. A simple thermistor is used along with the analog comparator; when the temperature of a food cooler exceeds safe limits, the MAXQ3210 indicates the condition to a clerk in the grocery store. This local temperature monitoring has endless useful applications, such as network equipment, beverages, film, laboratory equipment, art supplies, and virtually any perishable product.

Applications can also be about convenience. The MAXQ3210 in a smart motion detector alerts a homeowner when a pet, child, or intruder enters an area of the house that is off-limits. Pushbuttons are used to configure the sensor.

The MAXQ3210 is a natural fit as a parking assistant. Using a simple distance-detection circuit, the MAXQ3210 sounds its horn for different durations depending on the distance measured. This application requires configuration and intelligence in the microcontroller. When placed in the garage, this circuit helps owners park their cars without bumping into the walls. An end user might not want their automated parking assistant sounding the electric horn each time they walk in front of the circuit. Consequently, the device is programmed with an initial delay—when motion is first detected, the system waits two seconds to see if any additional motion is detected. If not, the motion was probably someone walking in front of the sensor. Also, the device could be disabled through the use of pushbuttons; it would be inconvenient if the device constantly beeped while the end user worked in the garage.

## Evaluation kit

The MAXQ3210 evaluation kit (EV kit) is an excellent platform to begin prototyping any MAXQ3210 application. It runs off a 9V supply or a 9V battery. Two pushbuttons control the reset and external interrupt signals. A 10-pin JTAG header provides access to hardware debugging routines, thereby allowing viewing and modification of registers, memory, and stack. The I/O pins are connected to a convenient 2 x 20 header, close to a prototyping area for testing external circuits.

An on-board piezoelectric horn and LED can be used to test the sights and sounds of the application. By default, the horn outputs a damped sound—loud, but not painful. Jumpers can be added to the board to short the dampening circuit, allowing the horn to be driven at its full 85dB volume.

The MAXQ3210 EV kit can be used with MAX-IDE. It supports the hardware debug engine of the MAXQ3210, providing source-code-level debugging and memory monitoring.

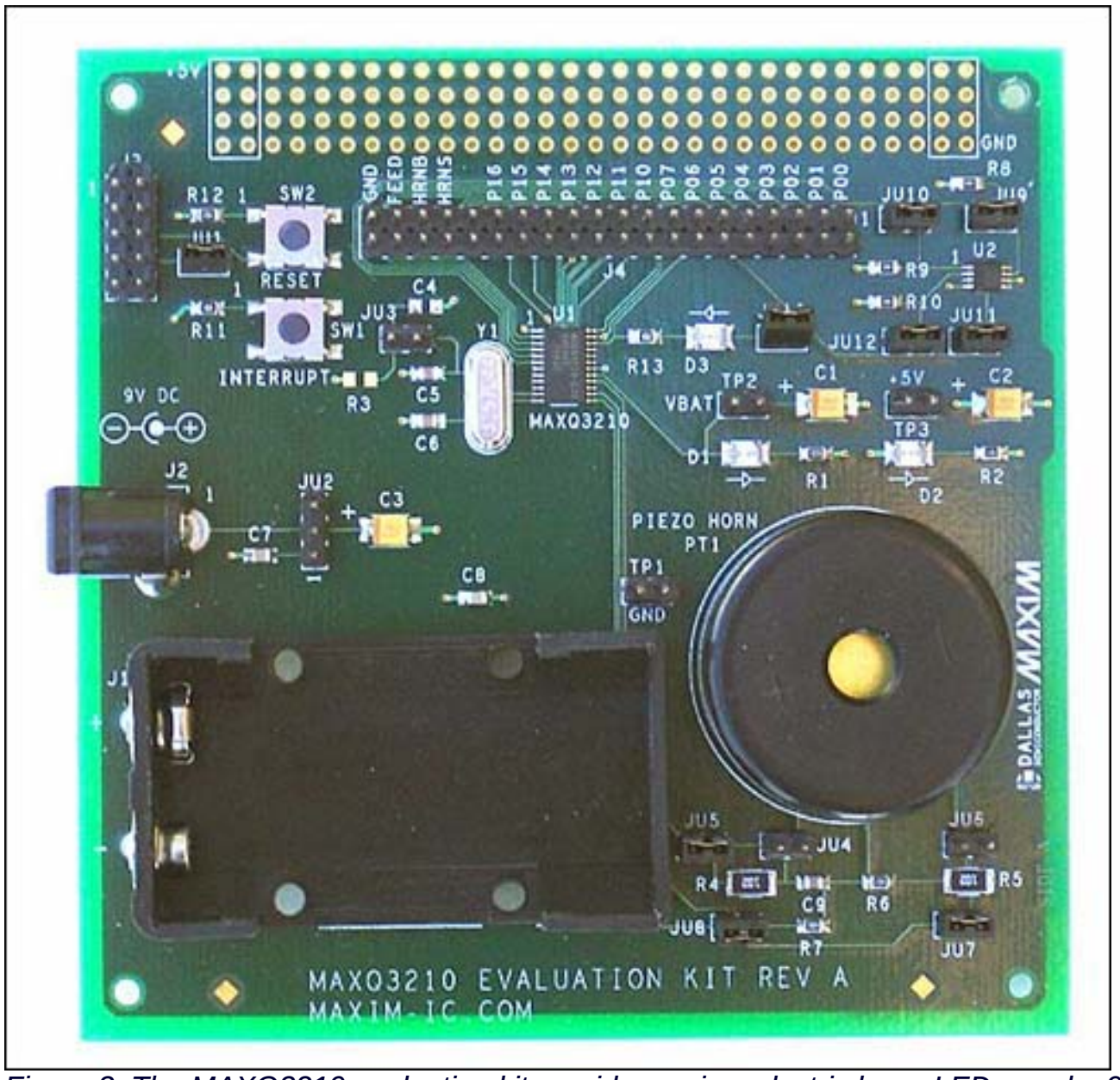


*Figure 3. The MAXQ3210 evaluation kit provides a piezoelectric horn, LEDs, and a 9V battery holder for complete application development.*

## MAXQ3210 Benefits Summary

As we have seen, the MAXQ3210 has several advantages for environmental-monitoring applications. The primary advantage is integration—components required by monitoring applications (comparator, horn, and LED driver) are integrated onto the chip, eliminating the need for external chips to drive these functions. Integration lowers the overall system cost and improves reliability by

reducing the number of components that need to be tested. Plus, a single chip requires fewer connections, lowering the test time for the end circuit board. The single-chip solution also means smaller and less expensive PC boards.

Additional microcontroller benefits are high performance and low-power consumption. The single-cycle MAXQ core and large register space allow applications to store data efficiently and perform computations quickly. The MAXQ3210 spends more time in low-power sleep mode and less time executing code.

Finally, the MAXQ3210's battery monitor and data EEPROM allow smart, self-monitoring applications. Devices can warn the user when their battery is nearing depletion. In addition, applications can track the life of their components and implement a planned end-of-life.

## Conclusion

The MAXQ3210 is a low-pin-count implementation of a MAXQ microcontroller, designed for applications that do not require the peripheral support provided by more expensive microcontrollers. While the MAXQ3210 is an excellent fit for environmental sensors, it is truly a general-purpose, high-performance, power-saving microcontroller, capable of adding intelligence and interaction to many applications.

It is important to note that while this article discusses environmental monitoring, the MAXQ3210's applications are far broader. With its data EEPROM, a 16-bit timer that supports capture, compare, and PWM operations, and high-performance MAXQ microcontroller core, the MAXQ3210 is useful for a wide range of microcontroller applications.