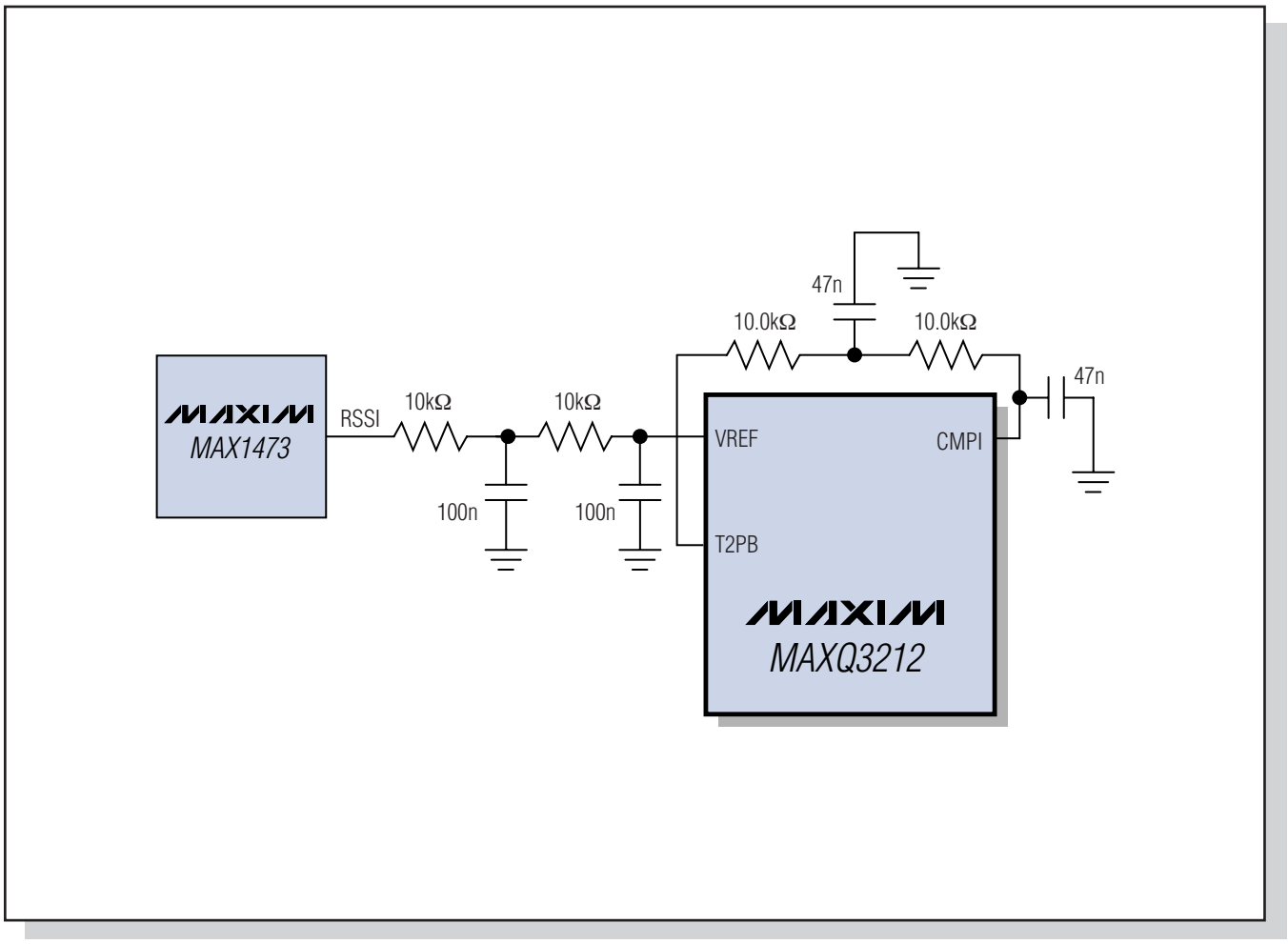
Microcontroller ENGINEERING REVIEW

Volume 6

Table of Contents



The MAXQ3212 comparator is used to measure the analog signal strength. (See page 6.)

Security in Embedded Systems

Key protection requires that the secret key never leave the confines of the embedded system, as this would provide an easy way for an attacker to defeat the device's security.

Security in embedded systems is usually an afterthought. Engineers design products to get to market quickly, saving a security upgrade for a future revision. This is not illogical behavior, because products with a higher level of security can be more expensive and later to market.

Many systems, however, need a high level of security at the outset. Sometimes the requirement for security derives from the government or a trade organization. The PCI requirements drawn up by credit card companies Visa® and MasterCard®, for example, provide a detailed description of the security required in a point-of-sale terminal or a PIN pad. In other cases a design incorporates security to protect revenue flow. A secure application can impede reverse engineering, prevent a device from being copied, or provide true tamper detection.

But what is it that a secure microcontroller really does, and why is a secure microcontroller so crucial to sensitive applications?

A System Is Only as Secure as Its Key

Security is not accomplished by encryption alone. While the choice of encryption algorithms and key management routines are critical, they are usually not the weak link in a secure application. Imagine that Alice and Bob each have a secure phone that can only communicate with one another. The encryption implemented on the phone is practically unbreakable, and would take a century to break with all the computational power in the world. What is the weak link? The phones. If an attacker gains control of one of the phones, he or she can pose as Alice or Bob and immediately gain access to their secret information. The attacker would not even need to steal the phone, but simply to install a listening device without Alice and Bob's knowledge.

In this scenario, the encryption was not defeated, but rather the encrypting device's security, or "key." In embedded systems, the key is almost always a large, secret number that can be used by a cryptographic routine to encrypt information or authenticate data. A secure embedded system's most important job is, therefore, to protect that secret key. If the system comes under attack, the key must be erased to prevent it from falling into the hands of an attacker. The destruction of the key renders the device inoperative, and prevents an attacker from gaining access to sensitive information such as bank account numbers and passwords.

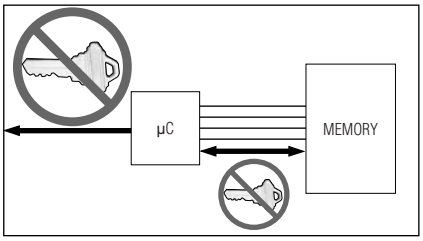


Figure 1. *Secret key data should not leave the device or even be transferred between ICs.*

Key protection requires that the secret key never leave the confines of the embedded system, as this would provide an easy way for an attacker to defeat the device's security (**Figure 1**). A dedicated memory inside the design does not work, however, for key storage because transactions between the microcontroller and the memory could be watched. The best security is one that requires the key to stay inside the processor using it to encrypt or authenticate data. This means that the system's microcontroller needs internal, nonvolatile memory.

Trip Wires and Plastic Protection

Even with key data stored only on the microcontroller, attackers can still discover secret information. For example, if an attacker can access the microcontroller's address and data buses, he or she could insert instructions to dump the key data to an external I/O port. A more sophisticated attacker could actually remove the plastic packaging from the microcontroller and use a microprobe to read the internal memory contents. A secure system, therefore, needs some way to impede this access, and even signal the microcontroller to erase its memory contents.

One simple approach to this security challenge is to seal the entire "sensitive area" (i.e., microcontroller, clocks, memories) in a tamper-evident material, perhaps by filling an area of the PC board with plastic or covering it with a metal box. Trip-wire devices can be used to detect high temperatures or the enclosure's removal. That detection would be useless, however, if the microcontroller is in a low-power state and cannot take action.

DS5250: Real Security for Key Protection

The DS5250's internal nonvolatile SRAM (NV SRAM) provides the perfect storage for sensitive information and encryption keys.

The DS5250, Dallas Semiconductor's state-of-the-art secure microcontroller, solves these problems and helps systems achieve a high level of security, sufficient for use in government and financial applications. It all starts with memory.

NV SRAM

The DS5250's internal nonvolatile SRAM (NV SRAM) provides the perfect storage for sensitive information and encryption keys. This custom, low-leakage SRAM meets two critical requirements:

1) *The data must be nonvolatile*. Using a small, inexpensive battery, key data is maintained for several years.
2) *The data must erase quickly*. The DS5250's SRAM instantly erases when any of the chip's tamper-detection circuits is activated.

Battery-Powered Tamper Detection and Reaction

In addition to NV SRAM, a secure system requires sensors to detect an attack. The DS5250 has multiple battery-powered tamper sensors. The microcontroller does not need to be in an active state to react to a tamper event.

The DS5250 can detect fault-injection attacks with its on-chip temperature and voltage sensors.

The DS5250 can detect fault-injection attacks with its on-chip temperature and voltage sensors. When the operating voltage or temperature passes outside the microcontroller's operation range, the DS5250 instantly erases its internal NV SRAM. This action eliminates the possibility of a hacker recovering any key data. To prevent microprobing of the NV SRAM cells, the top layer of the DS5250's silicon implements an ultra-fine mesh. If traces of that mesh are shorted, the DS5250 triggers a self-destruct and the key data is erased.

The DS5250 also has inputs that enable external circuits to trigger a self-destruct. This allows a system to implement multiple layers of security. The types of external circuits that can trigger a self-destruct are limitless. Some of the more common external sensors include:

- Switches on an enclosure to detect entry.
- PC board traces that are broken when a covering is removed.
- Light sensors to detect when a case has been opened or is being examined.

Encrypted Code Space

During initial system loading, the DS5250 uses a random 3DES key to encrypt its instruction code before the code is stored in an external memory (**Figure 2**). This prevents an attacker from inserting malicious code into the DS5250 for execution, and also resists any attempt to reverse-engineer the application. Integrity checks can also be inserted in the code, thus detecting an attacker's attempt to alter the program code.

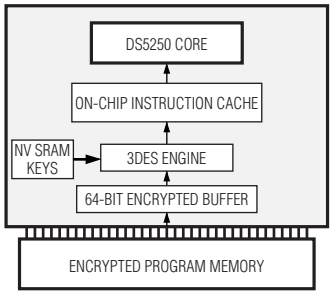


Figure 2. *Encrypted code space safeguards the algorithms and data in an external memory space.*

Encrypted code space not only prevents an attacker from reverse-engineering an application, but it also prevents someone from copying the device. Because the encryption keys are randomly generated on each DS5250, no two systems have the same data stored in their external flashes. The external flash would only be useful if an attacker knew the encryption key, but we have already seen that the DS5250 does not give up its secrets easily.

Engineering for Security

Designing secure systems is a challenging task. Trying to enhance existing designs is even more challenging. Key protection is the most critical part of a secure systems design. The DS5250 is designed specifically to safeguard the key, and thus provides the highest level of security for protecting any sensitive data. For more information about the DS5250 and our secure microcontrollers, go to www.maxim-ic.com/securemicro.

Remote Keyless Entry with the MAXQ3212

The MAXQ family of microcontrollers is designed to be electrically quiet for best integration with analog circuitry, including RF receivers.

Most automobiles today ship with a factory-installed remote keyless entry (RKE) system. But what if you wanted to add one to your older, hard-to-find-parts, "classic" automobile?

The MAXQ family of microcontrollers is designed to be electrically quiet for the best integration with analog circuitry, including RF receivers. This article discusses the components needed to make an RKE receiver using the MAXQ3212 microcontroller and the MAX1473 receiver.

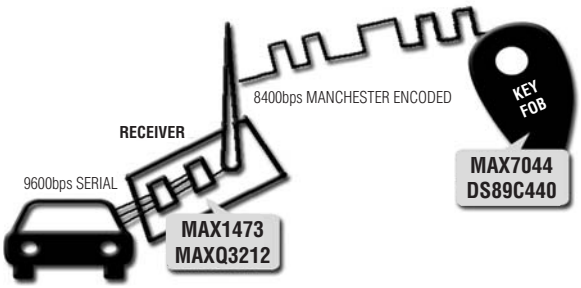


Figure 1. *An RKE system needs both key fob and receiver.*

System Overview

An RKE needs a key-fob transmitter and a receiver mounted somewhere in the car. **Figure 1** shows an overview of the system. (Note that the MAXQ3212 is a variant of the MAXQ3210, which could also be used for this project.)

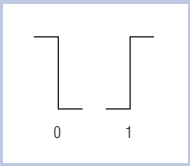
Protocol

Keyless entry protocols differ vastly among car manufacturers, models, and model years. Using a programmable microcontroller is therefore a good idea for this after-market project. For this article, we arbitrarily pick an 8400bps Manchester encoded digital data stream (see *Manchester Encoding* sidebar), transmitted at 433MHz using ASK. To use FSK or a different frequency, you must substitute a different receiver chip for the MAX1473. Visit www.maxim-ic.com/wireless for more information about Maxim receivers.

Manchester Encoding

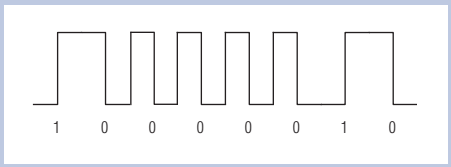
Each data bit is signified by at least one signal transition, making the data stream self-clocking. **Figure 2** shows the symbols for 0 and 1, if we choose a falling edge for 0 and a rising edge for 1.

Figure 2. *A falling edge encodes a 0; a rising edge encodes a 1.*



Serial data is commonly transmitted LSB first. The ASCII character "A" (41h, 0100.0001b) is transmitted as 1000.0010b, as shown in **Figure 3**. The encoding can be derived by concatenating the symbols for 0 and 1 bits.

Figure 3. *ASCII "A" is encoded by concatenating the symbols for 0 and 1.*



Key Fob

Since we are concentrating on the receiver for this RKE, we use two evaluation kits (EV kits) for our transmitter: DS89C450-KIT and MAX7044EVKIT. These kits can be fitted into a case side by side with rechargeable batteries underneath (**Figure 4**). The key fob is somewhat oversized, but serves well as a demonstration transmitter.

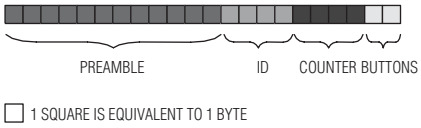
With an antenna, the range can be orders of magnitude higher than that of a standard key fob. *Note: Use this transmitter in a shielded lab environment only and follow standard procedures such as limiting the output power.*

Figure 4. *The key-fob transmitter uses two EV kits side by side.*

Data Stream

When a button is pressed on the key fob, it sends a synchronization preamble, followed by a transmitter ID, a counter, and button data (**Figure 5**). The transmitter repeats this sequence until the button is released, requiring a software debounce routine. In our example code, this is achieved by simply disabling the receiver for a short period.

Real-world systems also encrypt parts of the data to prevent vehicle theft. Decryption is usually handled by the car's body control module (BCM).



1 SQUARE IS EQUIVALENT TO 1 BYTE

Figure 5. *The key fob transmits a preamble, ID, counter, and the key code.*

Receiver

The receiver consists of a MAXQ3212 16-bit microcontroller and a MAX1473 receiver mounted side by side. **Figure 6** shows the populated PC board. The wires on the side connect to the car's BCM. For this demonstration, we dedicated a port pin on the MAXQ3212 to transmit asynchronous serial data at 9600bps. **Figure 7** shows the MAXQ code for a simple bit-banged serial port.

Software

The receiver software measures the receive signal strength, waits for and synchronizes on the preamble, decodes the data stream, and transmits the values through the serial port.

Signal Strength Measurement

Measuring the signal strength is independent of our main task, the Manchester decoding. The MAX1473 receiver has an analog received signal-strength indicator (RSSI), which we measure. The MAXQ3212 features an analog comparator (comparing VREF and CMPI inputs), and can generate pulse-width modulation (PWM) on the timer output pins.

Figure 8 shows how to construct an ADC from the comparator and PWM. We feed the RSSI signal into the MAXQ3212 comparator's VREF. We then program the timer to PWM that, when suitably filtered, yields a DAC output. This DAC is connected to the other comparator input, CMPI. The comparator then compares the signal levels; if they match, we have a successful analog-to-digital conversion without a dedicated hardware ADC.

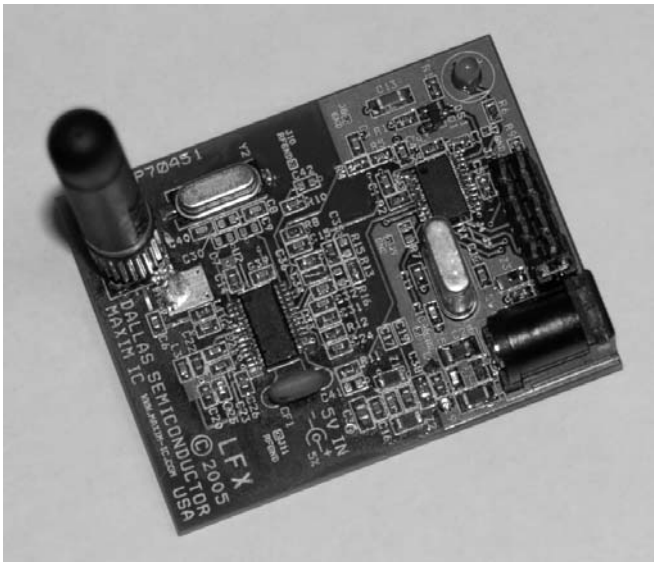


Figure 6. *The RKE receiver board is populated with a MAXQ3212 and a MAX1473.*

Rather than implementing successive approximation in software (which causes repeated large swings of the DAC signal, thus requiring longer settling times), we choose a slope-ADC. Starting at a reasonable minimum, the DAC output is ramped up until the comparator indicates a match.

```
;
*****************************************************************
; Transmit a byte over the bit-bang serial port on P0.0.
; Baud rate 9600, 3.2768 MHz ==> 341 clock cycles per bit
;
SerialPortOutput:
     move LC[0], #9              ; Start bit + 8 data bits
     move C, #0                  ; First output bit to 0
serport_nextbit:
     move LC[1], #335            ; 6 cycles + delay loop == 335
     sjump C, serport_onebit     ; Is this a one bit?
     move P00.0, #1              ; Set a zero bit
     sjump serport_delay         ; Jump to the delay
serport_onebit:
     move P00.0, #0              ; Set a one bit
     nop                         ; Even out the timing
serport_delay:
     djnz LC[1], $               ; Single bit delay
     sr                          ; Shift for next bit, current in c
     djnz LC[0], serport_nextbit ; Next bit
     move P00.0, #0              ; Send a stop bit
     move LC[1], #600            ; Extra long stop time
     djnz LC[1], $
     ret
```

Figure 7. *Serial port output can be generated using a simple port pin.*

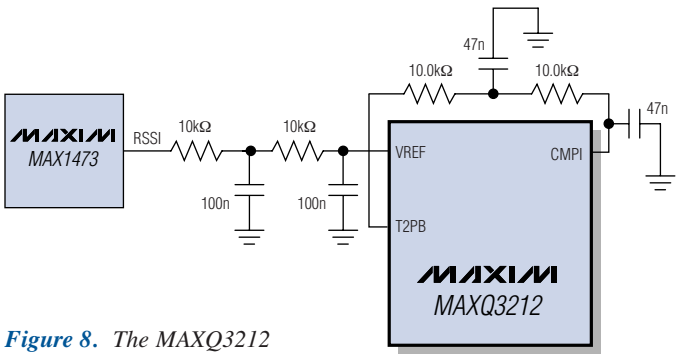


Figure 8. *The MAXQ3212 comparator is used to measure the analog signal strength.*

RF Signal Decoder

The MAX1473 supplies a digital signal output (DATAOUT). Due to always-present RF noise, this pin is continuously toggling, regardless of whether the key fob is actually transmitting or not. To distinguish this noise from a signal, the MAXQ microcontroller must implement a small state machine to measure the times between rising and falling signal edges, and to recognize the synchronization preamble.

The most efficient way to measure edge distances is by using interrupts. The MAXQ can be programmed to trigger an interrupt on a rising or a falling edge. To start a measurement, we set the interrupt to "rising edge." Once that edge is detected, we reset and start the timer and change the interrupt edge to "falling." On the falling edge, the interrupt handler reads the timer value. **Figure 9** shows a code fragment that reads and resets the timer and then toggles the interrupt sense.

If the edge distances match the data rate of 8400bps (plus/minus a reasonable tolerance), and the protocol-specific number of synchronization pulses is detected, the microcontroller's software state machine switches to receive mode and starts interpreting the rest of the packet data.

Conclusion

MAXQ microcontrollers are designed to be electrically quiet and integrate very well with Maxim RF components, without significantly degrading the RF signal. Code and schematics for the demonstration transmitter and receiver described in this article can be requested by email at micro.software@dalsemi.com. When emailing, be sure to tell us about your project.

```
;
; We have an edge on P0.0. Grab timer value and toggle interrupt sense.
;
   move  a[T2_CURR], T2H        ; Read shifted by ~ 4 cycles
   move  T2H, #0                ; Reset timer
   move  acc, EI0
   xor   #5                     ; Flip edge phase and clear interrupt
   move  EI0, acc
```

Figure 9. *Edge detection and timing can be entirely interrupt driven.*