CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

by Alberto Ricci Bitti

Flash Innovation 2003
Design Contest Winner

Wireless Monitoring System

Alberto's MC68HC908-based wireless monitoring system is adaptable for use in domestic and industrial settings. The central monitoring station, which consists of a computer-controlled receiver with a relay output and LCD, logs and displays data from up to 20 different sensors. Read on to learn how to build, program, and test your own system.

One of the things I have learned from my everyday engineering practice is that there is always room for ingenuity and improvement. This is particularly true for this project, which applies a couple of inexpensive microcontrollers to an unusual device: a live-catch mousetrap.

Mousetraps of this kind imprison mice instead of killing them with chemicals or bloody mechanisms. They are useful when regulations, laws, or just plain common sense forbid the use of poisons and chemicals. This applies to the food industry, from farms to your preferred restaurant, as well as places like schools and homes (where children and pets can open traps) and even hospitals and pharmaceutical facilities (where contamination should be avoided). As you can see, I'm talking about a worldwide market with millions of customers.

Unfortunately, live-catch traps are extremely expensive to maintain because of the labor costs required to continuously check them. Although a single mouse catch is a rare event in today's hospitals or pharmaceutical depots, the traps must be checked every few days. Making matters worse, the traps are often located in places that are difficult to access.

I designed my system with the objective of drastically cutting the labor costs involved with checking traps (see Figure 1). Now you can leave the traps unattended until the system calls for assistance.

My system consists of a monitoring station—a computer-controlled receiver with an LCD and relay output—and up to 20 mouse sensors. It works by placing a mouse sensor (a small plastic box) inside each trap. When a mouse is captured, the sensor transmits its trap identifier to the monitoring station, which logs and displays it where it's conveniently viewed. Alternatively, the receiver can dispatch the call to an external service, triggering an ordinary automatic phone dialer connected to its relay out. Refer to the "The System at Work" sidebar for more information.

NONTRIVIAL REQUISITES

Although conceptually simple, such a system design is nontrivial. The parts count must be kept to a minimum in order to contain costs. The dimensions are essential to fit even the smallest traps on the market. In addition, mouse sensors must be battery operated, so the batteries must last for years of continuous service. This calls for low-power parts and carefully designed software.

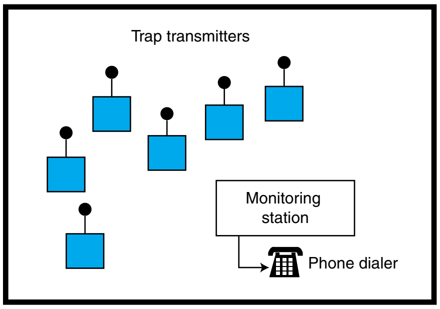

Figure 1—*My goal was to cut labor costs and eliminate the need for weekly trap checks. A transmitter is placed in every trap to signal the presence of mice. If a trap needs to be emptied, the system automatically calls home.*

Another requirement is that the system be able to resolve collisions that result from the simultaneous transmission of two or more mouse sensors. It should also tell you when the batteries need to be replaced and be able to detect when a trap is lost or destroyed, which isn't an unlikely event in industrial environments. And, of course, the system must be simple, resistant to dirt, reliable, and easy to manufacture.

On the receiver side, the monitoring station must be cost-effective, dependable, and easy to set up and use. It must show complete trap information, and the configuration data must be retained after power interruptions. The design should be compact, modular, and flexible. As with all new products, additional requirements are expected to emerge as work on the design progresses; therefore, high-level programming languages are preferable.

MCU SELECTION

Not many years ago, I would have started this design by selecting a specialized remote control encoder/decoder IC pair, looking for ultralow-power parts, and adding glue logic. If a microcontroller were required, I would have selected an MCU with a familiar architecture and instruction

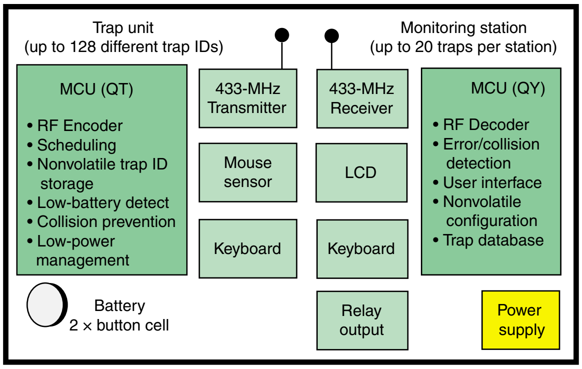

Figure 2—*As you study the transmitter and receiver block diagrams, keep in mind that the microcontrollers contain most of the functions required by the system. This keeps the number of external parts to a minimum. The functions listed in the MCU boxes are implemented with a mixture of software and on-chip peripherals.*

set. The time it takes to learn the development tools (not to mention the cost) would have influenced my choice.

Nowadays, the design path is somewhat reversed: MCU selection is one of the first steps in the design process. You can use general-purpose, low-cost microcontrollers for tasks previously done with dedicated ICs, with the extra advantage of adding functionalities in the software. Tools are inexpensive and sometimes free. In addition, it's no longer necessary to know assembly language because high-level language compilers are available for programming (including the smallest possible controllers).

I selected the 8-bit MC68HC908 microcontroller for this design. The same MCU core comes in a tiny eight- (QT suffix) or 16-pin (QY) package, with 128 bytes of RAM and 1 to 4 KB of flash memory. This IC includes unique features that make it perfect for this application. It does not require external reset, and its flash memory-calibrated internal oscillator is suitable for battery-operated devices because it keeps steady despite varying power voltages. Therefore, all of the pins—except the power supply—are available for input and output, which makes the eight-pin packaging effective for the trap transmitter. The 16-pin version nicely fits the receiver's requirements.

At an aggressive price, I'm talking about a truly classic MCU, with a true stack, capable of running true ANSI C code. Note that there is also hardware support for one-pin in-circuit debugging (ICD), so an emulator isn't required. Furthermore, flash memory can conveniently emulate EEPROM, a feature I used to store trap IDs.

As for the tools, the Metrowerks Codewarrior IDE includes an assembler, ANSI C compiler, simulator, programmer, in-circuit emulator/debugger, and even a light version of Processor Expert, which is an automatic C code generator. Although the tools are professional-grade, they are available for free.

SYSTEM BUILDING BLOCKS

The block diagrams here show how most of the building blocks have moved from external hardware to internal software modules. The transmitter's block diagram counts only three blocks outside the chip, as opposed to six internal function blocks supported by the MCU through a combination of software and on-chip peripherals (see Figure 2). The parts outside the microcontroller are the sensor mechanism and reed relay, two keys for setting up the trap ID and arming the trap, and a 433-MHz low-power transmitter with a rod antenna. The power comes from a couple of button batteries.

Inside the microcontroller, software modules implement a digital generator for encoding data to be transmitted, a scheduler for cyclic "keep alive" transmissions, storage of user-programmable nonvolatile ID data, a detector for the battery charge state, a manager for a simple mechanism to prevent overlapping transmissions, and a module for administration of the low-power modes.

The monitoring station includes a 433-MHz receiver module and its antenna, a relay stage for driving an external alarm or phone dialer, a 2 × 16 LCD module, and a five-button keyboard. I powered the unit with a wall-cube mains adapter through a linear regulator stage. As for the transmitter, many of the functional blocks are implemented by the MCU that's

THE SYSTEM AT WORK

The system is in Automatic mode at power-up. Photo 1a is the message for normal operation when no traps are triggered.

Photo 1b depicts a situation when a trap trigger is signaled immediately. The system supports up to 20 trap memories. Trap IDs can range from 0 to 127. If more than one trap triggers, their respective messages scroll automatically every few seconds. Press Clear to reset the message.

The system checks if a trap gets lost (no signal is received for more than 6 h), as well as if its battery is exhausted (see Photo 1c). To select Manual mode, press the Forward or Back keys in order to browse trap messages (see Photo 1d). In Manual mode, you can check data for all of the 20 trap memories, including those still free (see Photo 1e).

The Setup key enters Setup mode. To add a new trap, search for free memory and press Rearm on the trap itself. Its ID will appear immediately on the screen. Press Store to assign it to the memory currently selected (see Photo 1f).

Press the Clear button to delete a trap and free its nonvolatile memory. A trap's Rearm key also serves as reset after a trigger is detected and the trap is emptied. The Setup key on the transmitter changes the trap ID, scrolling IDs from zero to 127. You can check the new ID on the screen because it is immediately transmitted with each key press.

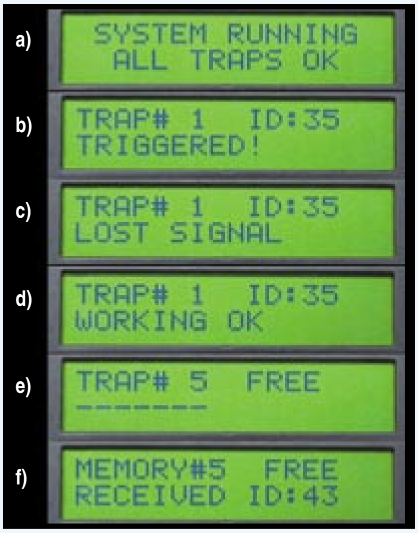
a)
```
 SYSTEM RUNNING
  ALL TRAPS OK
```
b)
```
TRAP# 1   ID:35
TRIGGERED!
```
c)
```
TRAP# 1   ID:35
LOST SIGNAL
```
d)
```
TRAP# 1   ID:35
WORKING OK
```
e)
```
TRAP# 5   FREE
-------
```
f)
```
MEMORY#5   FREE
RECEIVED  ID:43
```

Photo 1—*Here's how it works step by step.*

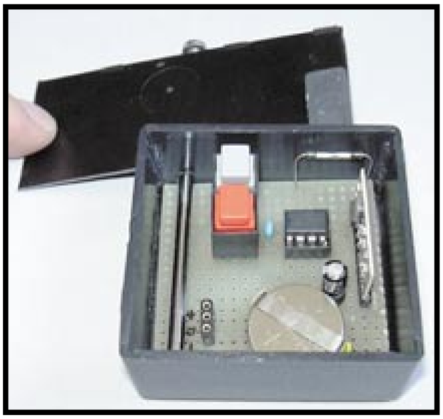

Photo 1—*The trap sensor/transmitter unit prototype is housed in a small plastic box. Putting your fingertip on the balance brings the magnet in front of the reed switch. Note how the antenna doubles as a pivot for the balance.*

in charge of decoding the receiver output and discriminating errors that result from bad reception, interference, or the simultaneous transmission of two or more mousetraps.

The MCU also stores trap information. It registers trap IDs in (internal) nonvolatile memory and programmatically updates trap status records in RAM. The user interface block consists of a trap data browser and a configuration mode for learning trap IDs. A monitor station can handle up to 20 transmitters—although a transmitter ID can range among 128 different IDs—to allow more than one monitor to operate on the same or overlapping areas.

MOUSE SENSOR

The first problem I had to solve was how to sense the presence of mice. I discarded electronics-only methods (e.g., photocells and capacitive sensing) one after another. Some were too sensitive to dirt, expensive, and difficult to clean; others were too accessible to munching rodents and consumed too much power. In this context, drawing a continuous 5 µA (equivalent to a 1-MΩ resistor at 5 V) represents significant power!

I was about to give up, when I saw a program on the National Geographic Channel showing the natural curiosity and vitality of mice. Realizing this, I added a little balance to the trap transmitter (see Photo 1). Sooner or later, an unwary mouse—frantically searching the trap for an escape route—will reach the balance. The balance freely pivots on the transmitter's aerial. A small magnet glued on one side counterweights it. I placed a reed-relay inside the sensor body to detect magnet movements and to trigger in turn the eight-pin processor. Bingo! It's like having the mouse switch on the transmitter for you!

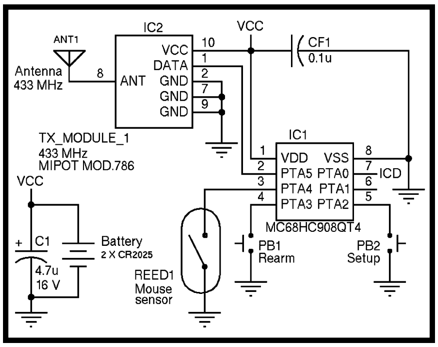

Figure 3—*The 68HC908QT4 is the heart of the transmitter. It embeds a brownout reset as well as a calibrated oscillator that makes reliable data transmission possible without crystals or resonators. The TX module can be changed in order to suit national regulations and frequencies. The reed switch closes when the mouse moves the sensor balance.*

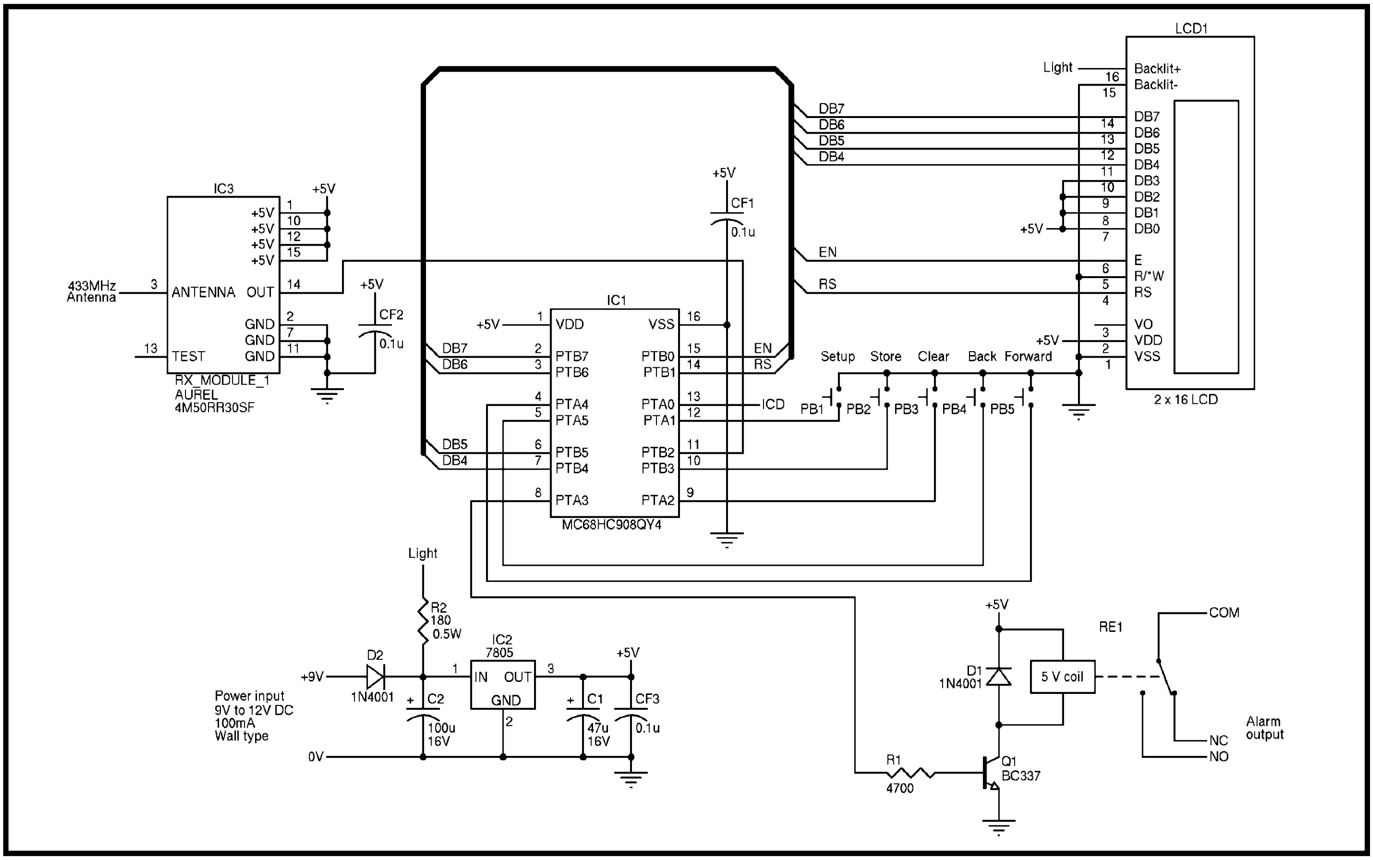

Figure 4—*The receiver has few parts. Pull-ups, an oscillator, reset generation, and EEPROM are all included in the MCU. Connections to LCD pins 15 and 16 (backlight) and R2 can vary to suit your LCD's specifications. Older LCDs need a contrast control voltage to be set on pin 3. The relay can trigger a phone dialer when a trap triggers.*

TRANSMITTER CIRCUIT

The transmitter's schematic diagram accounts for few parts other than the eight-pin processor (see Figure 3). The more that's inside the MCU, the less that's outside. This means not only a leaner bill of materials, but also a small circuit footprint, which is important in this application.

I connected two of the six available I/O pins to push buttons for rearming the trap after trigger detection and to set up the trap ID. Another pin goes to the reed-switch trap trigger. These pins are pulled up internally by the MCU.

A fourth pin drives the 433.92-MHz 2-5000-786 thick film transmitter module. It is compliant with European frequency standards and measures only 25.5 mm × 12.5 mm. The modulation method is on/off keying (OOK) according to the status of the PTA5 pin. When it isn't transmitting, the module consumes as little as 0.1 µA (more on this later). Check this figure when replacing it with similar parts, because it is vital for battery life.

The only remaining components on the board are a bypass capacitor (CF1) and an electrolytic capacitor (C1) required to lower the output impedance of the two button-type batteries that power the circuit. The circuit is compatible with user-monitor mode in-circuit programming. Connect the ICD interface to pin 7 to watch the code run.

MONITORING STATION

Figure 4 reveals the receiver's internals. I used modules for the LCD and the radio receiver; therefore, the design is noticeably neat and contains few parts. The MC68HC908QY feature set contributes to the circuit's tidiness. It includes input pull-ups, a steady internal clock oscillator, a brownout detector and reset generation, and flash memory that can be used as a replacement for EEPROM.

The 433.92-MHz receiver module takes the signal from the antenna and converts it to a more manageable digital-level pulse train. The receiver can be replaced with similar models to suit national regulations and frequencies.

The microcontroller processes the pulse train in order to distill meaningful signals from the inevitable RF noise background. It then displays the results on the 2 × 16 LCD module, which is connected in 4-bit mode, requiring six out of the 14 available MCU I/O pins. The LCD software driver assumes pins DB0 through DB3 to be at logic level 1 or unconnected. Depending on your LCD module, you may need to apply a contrast control voltage from 0 to 5 V to pin 3. Recent modules usually work well with this pin unconnected.

The keyboard, which consists of five push buttons, takes another five pins, which have their internal pull-ups enabled. I kept ICD pin (port A0) free for in-circuit debugging, as I did for the transmitter, making the board user monitor-debug compatible.

The last available pin is used to drive the relay output, which is realized with a classic transistor stage and a diode for protection against coil's over-voltages. The circuit is mains powered through a wall adapter supplying 9 to 12 VDC, regulated to 5 V by IC2, a classic 7805 stage. Diode D2 protects the circuit from power reversals. Resistor R2 limits the current for the LED backlight to a safe 40 mA. The backlight works from an unregulated power supply.

CONFLICT PREVENTION

To ensure reliable data circulation, the system must cope with the possibility of a simultaneous transmission from two or more traps. Another cause of trouble is interference from the cluttered license-exempt RF band. To protect the integrity of the data, the system acts on both the transmitter and receiver ends.

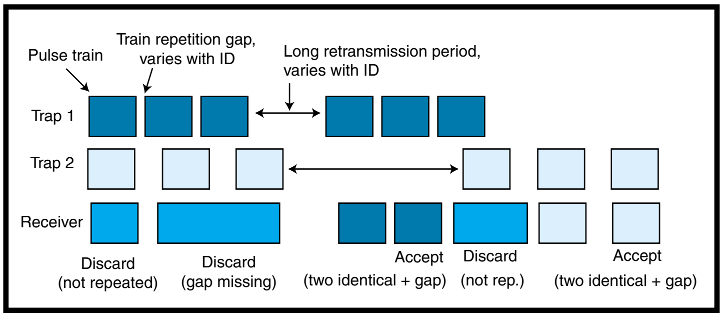
Pulse train
Train repetition gap, varies with ID
Long retransmission period, varies with ID
Trap 1
Trap 2
Receiver
Discard (not repeated)
Discard (gap missing)
Accept (two identical + gap)
Discard (not rep.)
Accept (two identical + gap)

Figure 5—*To protect data integrity, actions are taken on the transmitter and receiver ends. The transmitter repeats its data after a short pause, whose length varies in accord with ID. The receiver discards any data that is not repeated. ID-dependent gaps are also used for longer periodic retransmissions in order to avoid the parasitic synchronization of two or more transmitters.*

With a period determined by its heartbeat function, the transmitter sends the trap status, encoding it as a packet of 11 width-modulated pulses. It repeats the pulse train numerous times to add redundancy.

In order to distinguish RF noise from real data, the receiver checks the shape of the pulses before accepting them. Pulses too long or too short are discarded. As an additional requirement, it rejects any data that isn't preceded by a silence gap. Lastly, the receiver must get two identical data packets consecutively in order to validate them.

Simple redundancy like this does not protect against the unwanted synchronization of two or more transmitters, which, for example, can happen as a consequence of clock frequency drifts over long periods. To minimize this problem, I altered the data repetition rate and heartbeat period in accordance with trap IDs, as shown in Figure 5.

If two traps start transmitting at the same time, their repeated packets overlap at different points at each repetition because of the variable spacing between them. The receiver discards this variable interference pattern because the redundancy check requires two identical packets to accept data.

There is always the possibility that a transmission will get lost as a result of complete overlapping. In that case, you must wait the long retransmission period (heartbeat) for another chance at receiving the information. There will be no collisions on the next heartbeat, because the heartbeat period differs from one transmitter to another as it varies according to trap ID. Therefore, the system occasionally allows heartbeat signals to be missed as a consequence of external interference, poor reception, and (although not particularly likely) trap-to-trap overlaps. The receiver knows that a trap is still alive from its heartbeat. If more than 6 h slip away without receiving a single heartbeat, the receiver deems the trap lost and sends you a signal.

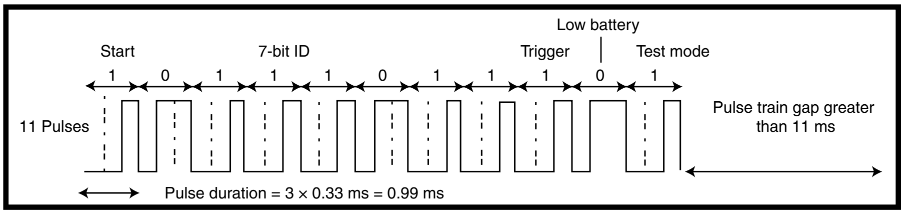
Low battery
Start
7-bit ID
Trigger
Test mode
1
0
1
1
1
0
1
1
1
0
1
11 Pulses
Pulse train gap greater than 11 ms
Pulse duration = 3 × 0.33 ms = 0.99 ms

Figure 6—*Transmitter data is packed in 11-bit code words. Pulse-width modulation is used for transmission, with 66% and 33% duty cycle pulses representing zeros and ones respectively. To add redundancy, code words are repeated and spaced apart at least 11 ms. (The exact value varies according to the trap ID.)*

SOFTWARE BASICS

I wrote the C code for the transmitter first. I didn't have a prototype at the time, so I ran the software on the Motorola QT demo board, using an oscilloscope to verify the signals. The demo board runs in User Monitor mode. It is preloaded with a small monitor program that fits a handful of unused flash memory bytes and uses interrupt table relocation to offer single-pin, in-circuit emulation and programming. I changed the value of `InitConfig1` to `%00100111` to allow for the use of the `STOP` instruction.

The demo board circuit and user-monitor program are detailed in Jim Sibigtroth's application note titled "User Mode Monitor Access for MC68HC908QY/QT Series MCUs." You must use the user monitor to load and run the transmitter and receiver software. To do so, follow the procedure described in John Suchyta's application note, "Reprogramming the M68DEMO908QT4." The transmitter code structure is straightforward, although it requires special programming to optimize power consumption.

I grouped hardware-specific details in the hardware.c file, which you may download from the *Circuit Cellar* ftp site. It hides port and register initializations, aliases for all of the pins used, macros for manipulating them (to disable trigger input pull-ups), functions and macros to go in Stop and Wait modes, and interrupts. Where possible, I prefer to handle port pins at the bit level, setting bits individually instead of setting the entire port at the same time. This makes the code more flexible, allowing painless pin swapping when routing a PCB for production.

The main.c file, which is the application's body, implements the usual big endless loop found on most embedded systems. At each loop, the MCU awakens from Stop mode and checks if it has been called by the keyboard interrupt (buttons pressed, sensor triggered) or the wake-up timer. It also checks the battery level. If the sensor detects a mouse or key press (or at almost every hour), the MCU formats and transmits the trap status.

The trap status is formatted for transmission, assembling a start bit, the 7-bit ID, and the trap-triggered and low-battery flags, in addition to a special test flag (set when the Rearm button is pressed) that's used when configuring the system. As you can see in Listing 1, the code word is handed to the `rf_encoder()` routine, which uses the 16-bit timer combined with the processor's Wait mode to generate a train of 33% or 66% duty cycle pulses (see Figure 6).

The monitoring station software is more complex. Refer to Figure 7 for help understanding its main structure. The receiver gets the most recent data from the radio decoder. Should the data match any of the traps stored in its internal database, the respective record is updated to reflect the new status. The receiver presents trap information according to the current user-interface mode (Automatic or Manual Scroll mode), which determines how to layout LCD data and react to keyboard clicks.

In order to execute time-scheduled housekeeping routines, the receiver checks the timeflags structure. Its different bits are set by the timer interrupt routine to indicate if twentieths,

Listing 1—*The routine in charge of transmitting the code word invokes `WAIT_MODE` to wait one-third bit time in Low Power mode. `TX_ON` and `TX_OFF` macros drive the RF module. The hardware.h file hides implementation details for the macros, making the code easier to read.*

```
static void rf_encoder(unsigned int codeword)
{
  unsigned int mask;
  //Prepare for timer overflow every 0.33 ms (a bit third)
  //Overflow will wake up from Wait mode
  TMOD = CLOCK_FREQ * BIT_THIRD_DURATION;

  //Prepare mask for filtering leftmost bit
  mask = 1 << (CODE_BITS - 1);
  while( mask != 0 )
  {
      TSC_TSTOP = 0;  //Restart timer
      WAIT_MODE;      //Go in low-power mode until a 0.33-ms
                      //period expires.
      if ( (codeword & mask) == 0 )
                      //Are you transmitting a zero?
          TX_ON;      //Yes, force a 66% duty cycle.
      WAIT_MODE;      //Go in low-power mode until a 0.33-ms
                      //period expires.
      TX_ON;          //Set output to ensure a duty cycle no
                      //less than 33%.
      WAIT_MODE;      //Go in low-power mode until a 0.33-ms
                      //period expires.
      TX_OFF;         //Turn off transmitter
      mask >>= 1;     //Prepare mask for filtering next bit.
    };
    TSC_TSTOP = 1;    //Stop timer before exit to reduce power
                      //consumption.
}
```

seconds, minutes, or hours have elapsed. Every second, the receiver updates a timeout that serves to restore Automatic mode after a short period of nonuse. Every hour, the receiver updates the trap heartbeat counter. Finally, if a trap needs to signal an alarm condition, the receiver activates the relay output and repeats the entire cycle from the beginning.

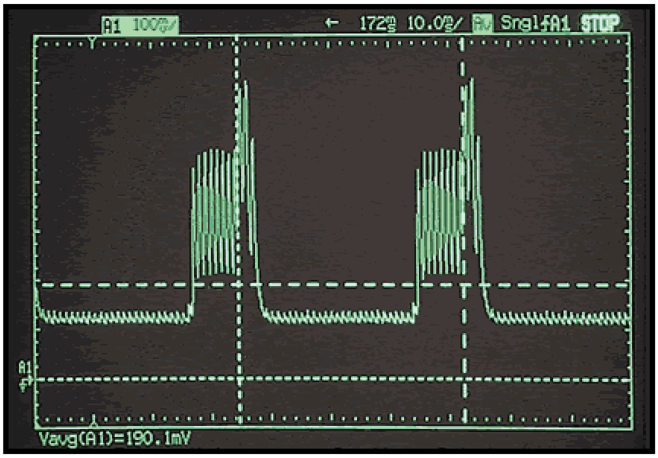

Photo 2—*The batteries last for years. Using Wait mode limits the average current during transmission to 1.9 mA. The overall average consumption is just 3.1 µA because data is transmitted only one time per hour (voltage drop on a 100-Ω series resistor: vertical = 100 mV/div, horizontal = 10 ms/div).*

According to good programming style, I encapsulated the various functions (radio decoder, trap database manager, flash memory-based EEPROM emulation, keyboard and LCD drivers, delay routines, and hardware-related primitives) in separate files orchestrated by main.c. Note that trap ID codes are stored in flash memory, which is used as a sort of EEPROM.[1]

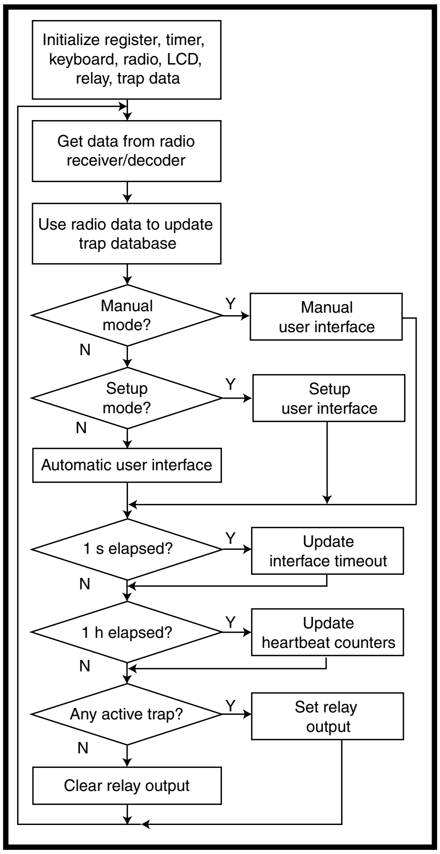

Figure 7—*The software is in charge of polling the RF module, updating the trap database, running the user interface, and performing time-scheduled checks. A different interface is presented according to the operating mode: Automatic, Set Up, or Manual. The timeout flag and RF module data is updated by interrupt service rou-*

Contrary to popular belief, C language code and data encapsulation are not flash memory-hungry ogres. In fact, the complete application requires only 3 KB. This suits these little 4-KB devices nicely, leaving plenty of space for future expansions.

CUT THAT POWER BILL

Getting minimal power consumption requires careful design and programming—no detail can be ignored. The CR2025 lithium battery can supply 170 mAh. This translates to an average of 19 µA over a one-year period. The single sensor internal pull-up can easily draw 10 times that current (see Figure 8)! Therefore, it's wise to disable the pull-up after a trap trigger is detected: simply switch its data-direction bit, making the pin an output, and set the output to zero. (Refer to the `REED_ENABLE` and `REED_DISABLE` macros in hardware.h.)

Another trick is to enable pull-ups for port B. Although not bonded out on the eight-pin QT part, they exist on the silicon. This is why I included QY in place of QT header files in the transmitter project.

Most of the time, the MCU is in Stop mode, relying on an automatic wake-up timer and keyboard interrupts to get back on from time to time. The timer interrupt replaces the usual software-based wait loops, making the MCU rest in Wait mode during transmission in order to reduce power. To save additional current, I disabled the ADC and stopped the 16-bit timer when it wasn't in use. Refer to Donnie Garcia's "MC68HC908QT4 Low Power Application" for tips about low power.

Photo 2 shows the current drawn by the prototype during transmission. I took the measurement from the voltage drop through a 100-Ω resistor in series with the positive supply. A 5-V bench supply powered the circuit.

The average current over a full transmission cycle (see cursors) is just 1.9 mA. The trap transmits for up to 200 ms every hour during operation (i.e., 1/18,000 of the time), requiring 0.1 µA (1.9/18,000) on average. This contributes to the current required by the circuit in Stop mode, which can be anywhere from 0.1 to 5 µA, according to the datasheets. My prototype required approximately 3 µA, which means that it can theoretically run for about 55,000 h from a 170-mAh charge. That's six years! In practice, the actual battery life might be noticeably shorter than this because of environmental conditions, tolerances, and discharge curves.

SYSTEM TEST

I built a couple of prototypes to perform preliminary tests. I used dual-in-

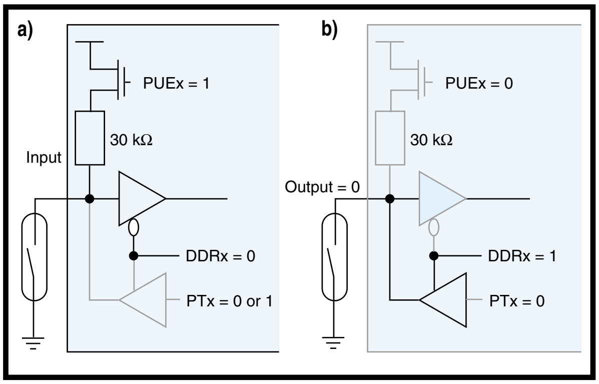

Figure 8a—*When the switch is closed, the input pull-up (ranging from 16 to 36 kΩ) can eat up 10 times the average current required by the entire circuit.* ***b—****To avoid unnecessary current leaks, after a switch closure is detected, the pull-up is disabled and the pin direction is changed to an output, whose value is set to zero.*

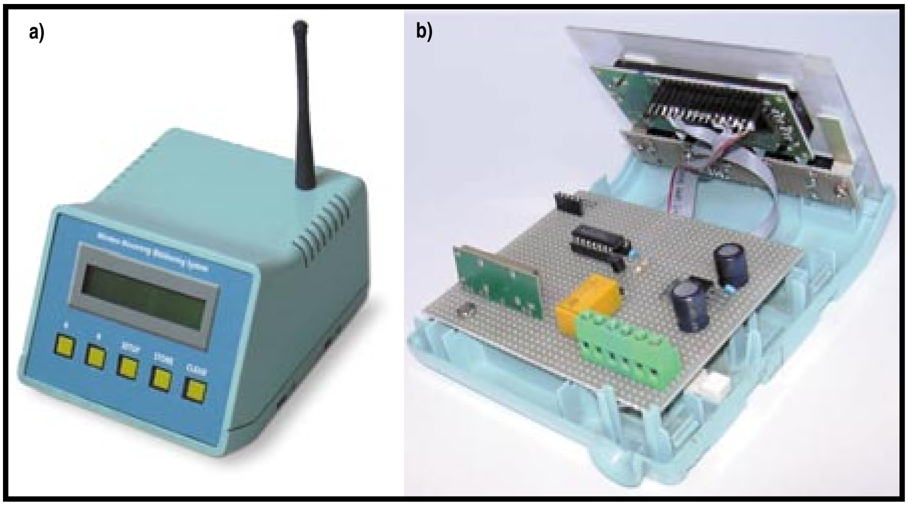

Photo 3a—*The receiver box is recycled from an old soldering station.* ***b—****The receiver is simple enough to be assembled on a prototype board. The display and keyboard are fixed to the front panel with thick double-adhesive tape.*

line MCU samples that are suitable for prototype boards and manual soldering. I placed the transmitter inside an off-the-shelf plastic box measuring only 54 mm × 58 mm × 28 mm, which looks spacious (see Photo 1). Production units can be much smaller than this. An older solder station case, refurbished for the occasion and completed with a few Dremel tool touches, provided an excellent enclosure for the receiver board (see Photo 3).

You can place the transmitter inside most commercial traps without difficulty. However, the transmitter range is greatly reduced for all-metal traps because of the shielding effect. A future release should provide an external aerial. Special hardened plastic must be used for the transmitter box because it's likely that some rodents will try to bite it. I am also considering reducing the number of possible trap codes from 128 to 64, or even 32, to make the ID set up less tedious.

The system works well, with neither false nor missed triggers. At last, I can monitor traps in the attic and basement from my desktop!

PLEASURE TO DESIGN

This project was a pleasure to design. The result is a simple and viable solution with excellent overall features. I hope you acknowledge its technical merit, too!

The system is highly optimized, with only a few components on the transmitter and receiver boards. Therefore, the system is definitely cost-effective, particularly the trap unit. Because there are up to 20 traps per receiver, even a $0.05 saving grows to be a dollar for a complete system.

Before starting this design, I was skeptical (to say the least) about the possibility of writing true C code for an eight-pin processor with only 128 bytes of RAM. Now I'm glad that I tried it. The compiler does a wonderful job of optimizing every single bit, and I had no trouble porting portions of code that were originally written for larger processors.

In the race for a better mousetrap there is no shortage of competition. I hope to see this unit in production some day. In my opinion, the design suits series-production technologies because it is easy to manufacture and test, and it doesn't require calibration.

Nevertheless, being modular by design and flash memory microcontroller-based, you can easily adapt the system and add new features. System variants can range from translation to languages other than English and integration in a home control system, to the use of different radio frequencies. For use in your home, you can replace the LCD with a few LEDs. Furthermore, you can use the same basic design for other tasks, like checking if all of your windows and doors are closed. But that's another story.

Alberto Ricci Bitti holds a degree in Computer Science from the University of Milan. He has more than 10 years of experience designing and writing software for embedded systems. Alberto currently designs industrial controllers and instrumentation for Eptar. In his free time, he enjoys competing in design contests. Alberto has been awarded several prizes. You may visit his web site, www.riccibitti.com, or write him at a.riccibitti@iname.com.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2004/167.

REFERENCE

[1] P. Topping, "EEPROM Emulation Using FLASH in MC68HC908QY/QT MCUs," Motorola, AN2346/D, September 2002.

RESOURCES

D. Garcia, "MC68HC908QT4 Low Power Application," Motorola, AN2310/D, August 2002.

J. Sibigtroth, "User Mode Monitor Access for MC68HC908QY/QT Series MCUs," Motorola, AN2305, July 2002.

J. Suchyta, "Reprogramming the M68DEMO908QT4," AN2322/D, August 2002.

SOURCES

4M50RR30SF Receiver module
Aurel S.p.A.
+39 0546 941124
www.aurel.it

Codewarrior Development Studio for HC08
Metrowerks
(800) 377-5416
www.metrowerks.com

2-5000786 Transmitter module
Mipot
+39 0481 630200
www.mipot.com

68HC908QT4 and 68HC908QY4 Microcontrollers
Motorola, Inc.
(800) 521-6274
www.motorola.com