FEATURE ARTICLE   by Peter Csaszar & Monica Flores

Message Acquisition

Ring Signal Detection And Interpretation

You'll never miss another voicemail message with Peter and Monica's voicemail monitoring system. Their device monitors the signal from an analog phone line, analyzes it, tracks the number of calls and messages, and displays the number on a VFD.

In some institutions, the internal phone exchange system (aka private branch exchange, or PBX) offers voicemail, but it can't drive the "Message Waiting" indicator (typically a flashing light). In other institutions, the telephones are not equipped with such a feature. The only indication of unchecked messages is a distinctive dial tone that plays when the handset is picked up. In such systems, some messages aren't received on time, which causes a variety of embarrassing situations. The incoming call indicator of certain caller ID boxes could serve as a workaround, but

only if the service is supported by the PBX, which was not the case in our study. We designed a solution to this problem for use at Lawrence Technological University in Southfield, Michigan.

Our design, code named the Phone Call Monitoring Facility (PCMF), monitors the ring signal coming through an analog phone line, analyzes it, and then, based on the number of rings, tracks the total number of calls, and calls with the possibility of a voicemail message (see Photo 1). We designed the system around a Microchip Technology PIC18F4520 microcontroller. Using

Microchip's C18 compiler, we developed the firmware entirely in C language. In this article, we'll describe how we built the system.

VOICEMAIL DETECTION

A classic method used by phone exchanges to notify users about new messages is a stutter dial tone. For a few seconds after the phone is picked up, the dial tone is interrupted by short pauses. Voicemail indicators designed for stutter dial tone detection poll the line to "listen" to the nature of the dial tone. Unless "silent" voicemail is supported
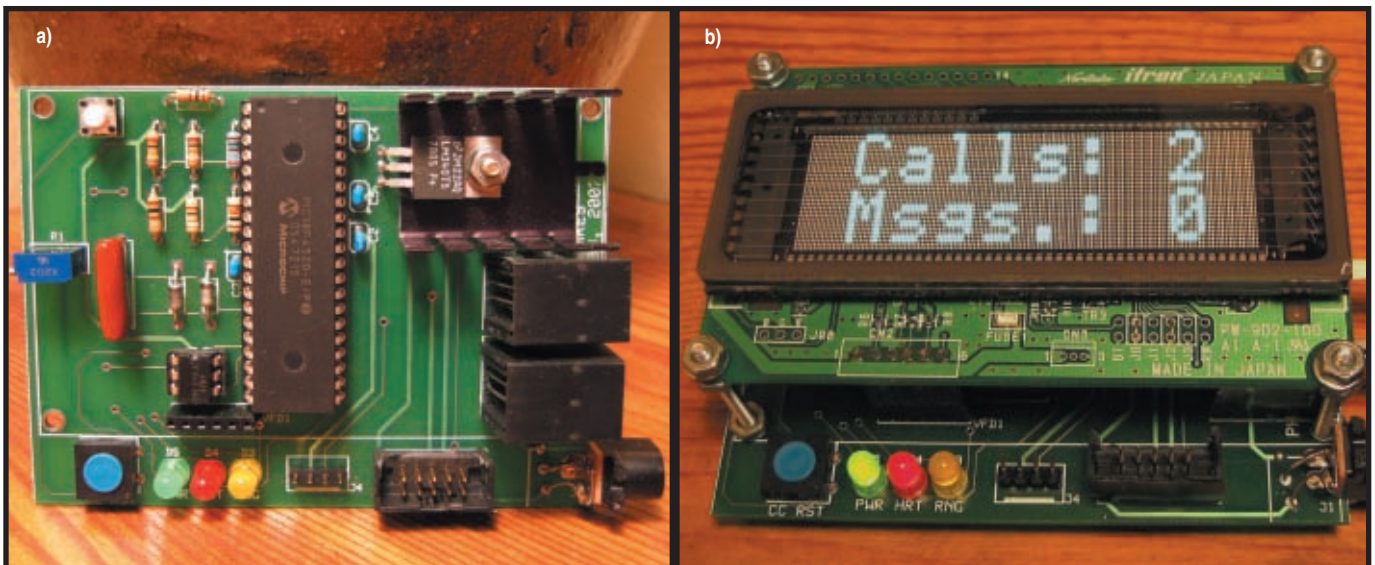

Photo 1a—This is the fully assembled circuit board, without the display. **b**—This is the monitoring system in operation. The end user will interact only with the blue Call Count Reset button.

Figure 1—The system's circuitry is fairly simple. The PIC18F4520 microcontroller requires very few external components.

(where a message can be left by means other than calling the recipient's own phone number), it suffices to poll the line after the phone rings, or after the subscriber gets off the phone. However, phone companies don't like polling because it keeps their circuits unnecessarily busy.

A technology that is gaining more popularity is frequency shift keyed (FSK) proactive voicemail notification. In this approach, the phone exchange sends a burst of data signals, similar to signals carrying caller ID information, which is recognized by the FSK receiver on the subscriber's side.

In addition to these methods, some PBX systems put a high-voltage DC signal on the line. This drives a neon bulb that is typically built into the phone set.

Our PBX provided only a stutter dial tone for message indication. Designing a circuit that reliably detects the stutter dial tone and picks the right moments for polling is not an easy task. Instead, we chose a more straightforward approach: monitoring ring signals. If the number of rings in a given call reaches the ring limit (i.e., the number after which the call is automatically forwarded to voicemail),

there is a possibility that a message was left. Mathematically speaking, the desired ring count is only a necessary condition for the message. Callers often let the voicemail kick in, but hang up before the outgoing message is over. These false positives are not a major problem. A bigger criticism of this method is its inherent false negatives: messages that are left while the phone was off-hook. However, for a user who doesn't use the phone too often (and for whom this device is intended), the odds of this situation remain negligible.

HARDWARE DESIGN

Figure 1 shows the system's circuitry. The heart and soul of the circuit is U1, a Microchip Technology PIC18F4520 microcontroller. We selected the parts for this design based on a few technical and educational considerations. As part of the computer engineering curriculum at Lawrence Technological University, students gain significant experience with 16-bit Freescale Semiconductor microcontroller devices, such as the 68HCS12 microcontroller. We used a PIC microcontroller for this project so that we could work with this chip family's significantly different

Harvard architecture. In addition, we needed a chip that was widely available, had enough code space to reasonably support firmware development in C language, had plenty of features for experimentation, and had a PDIP packaging that enabled prototyping in a solderless breadboard. In terms of the I/O resources, the PIC18F4520 is more generous than necessary. But due to our aforementioned needs, it was our final choice.

One of our primary design goals was to keep the entire circuit as simple as possible. Power is supplied from a 9-VDC wall adapter. The 5 V for the PIC18F4520 and the display is supplied by a 7805 regulator. Although the fundamental task of the firmware is time measurement performed on the ring signal, the precision of these measurements is not critical. Therefore, it is possible to run the microcontroller from its internal oscillator, which requires no additional hardware components whatsoever.

The microcontroller receives two input signals: the ring pulses from the optocoupler (U2) and the Call Count Reset button. The output peripheral of the PCMF is a dot-matrix vacuum fluorescent display (VFD1) module, which

we'll describe in the next section. The firmware also drives an LED, referred to as Heartbeat, in a flashing pattern. In addition to making the device look more gadgety, the light's purpose is to aid diagnostics. The same can be said about the power LED, the Master Reset button, and the test points. They can be removed from the production version of the circuit, but—as Monica experienced first hand—they provide invaluable help during prototyping. In the current design, the only non-power test point is the output of the optocoupler. The signal was helpful during the debugging process, because it showed whether the ring signal made it through the optocoupler. It also offered an easy way to insert the logic-level output of a signal generator into the circuit to test the firmware without the need to make actual phone calls.

RING MY BELL

The detection of phone rings using a microcontroller requires both hardware and firmware support. In hardware, the ring detector circuitry interfaces the ring signal with one of the microcontroller's digital input ports. The timing of the signal is then analyzed by the ring interpreter portion of the firmware, which can group ring pulses into separate rings and rings into individual phone calls.

The ring signal is a pure sinusoidal AC waveform, with a root mean square (RMS) amplitude of 40 to 100 V and a frequency of 16 to 67 Hz (typically 20 Hz in the U.S.).[1] Our ring detector follows the logic of the widespread ring detection circuitry described in detail in Julian Macassey's article "Understanding Telephones," which appeared in *Ham Radio Magazine* in 1985.[2] The role of C1 is to cut the DC path of the ring detector and serve as a series impedance in addition to R4. Zener diodes D1 and D2 make the circuit unresponsive to low-amplitude audio signals and noise. One half period of the AC ring signal turns on the optocoupler's LED, which with the pull-up resistor R2, provides the digital input for the microcontroller. Aside from being a simple and elegant solution, electric isolation of the optocoupler is required for user safety.

LEDs are infamous for having a low tolerance for reverse bias. The LED

inside the optocoupler is no exception. To protect it from the negative half-cycles of the ring signal, D3 is connected across to it in the opposite direction. By making D3 an LED, it can also serve as a debugging aid that visually indicates the presence of ring signals in the circuit. In this arrangement, the two LEDs are "watching each other's back." The values of C1 and R4 are chosen so ring signals, with extreme amplitude and frequency values, register with the microcontroller, but do not damage the optocoupler. Trimmer potentiometer R1 can be used to fine-tune the sensitivity of the circuit. (Texas Instruments used to manufacture a chip specifically designed for ring detection, the TCM1520A. However, it's no longer available.)

VFD MATTERS

The display is a Noritake GU140X32F-7003, which is a 140 × 32 pixel graphic VFD module. It is an elegant, pricey display intended primarily for high-end applications such as vending machines, office appliances, casino equipment, and more. If designed for mass-marketing, a simple appliance such as the PCMF with this kind of display would only make financial sense in the gold-plated, diamond-encrusted edition. The standard version would probably employ an LCD module with comparable features. However, due to a generous donation from Noritake to Lawrence Technological University, a couple of these displays were available for faculty and students to use in design projects, so it became a natural choice for our project.

The VFD module is an intelligent peripheral that includes driver electronics responsible for the low-level operation of the display panel. Towards a host, the module offers both asynchronous and synchronous serial interfaces. The former is semi-compatible with the RS-232 standard (the same signal timing, but TTL logic levels). The latter is a particular implementation of the SPI bus standard. This project uses only the display in Character mode, with its native size of 20 × 4 characters. The interface supports a long list of commands in addition to displaying ASCII characters, such as cursor positioning,

font magnification, font selection, scrolling, virtual display area, flashing display, and more.

For ease of implementation and debugging, we decided to communicate with the display using its 8-bit SPI connected to the PIC18F4520's on-chip SPI module. Considering that the display always acts like a slave with certain signal timing requirements, the microcontroller's SPI is configured as a master, clocked by 1/64 of the 8-MHz CPU frequency, and follows the SPI standard Master Mode 11 for data clocking.

Interfacing this VFD module posed a couple of challenges, adding tremendous value to the educational content of this project. (Monica appreciated it a lot, but only in retrospect.) You may easily refer to the first problem as "the mother of all ironies": the SPI modules on the microcontroller and the display use opposite bit orders! Another complication is that, in certain display commands, the value 0 can occur as a parameter, which is not a legitimate character in a C-style zero-terminated string, per definition. These issues are addressed in our firmware by the `putsVFD` function. It is dedicated to sending a string character by character to the VFD using the `WriteSPI` function from Microchip Technology's MPLAB C18 compiler library, replacing the API's generic `putsSPI` string function.

One well-known property of a VFD is its tendency for the infamous "burn-in" phenomenon. Because the individual display elements gradually lose brightness the longer they are illuminated, a predominant static pattern can reduce the brightness of its forming elements (in this case, pixels) more than that of the surrounding ones. Multiple "screen saver" schemes have been developed throughout the decades to address a similar issue with cathode ray tube (CRT) displays.

The PCMF firmware employs two strategies. When there are no phone calls to report, the display is switched to Power Save mode after 4 s of inactivity. In this mode, the VFD's filaments are also turned off, which, in addition to saving the pixels, also spares the filaments from extended use and reduces the power consumption of the entire device considerably. When phone calls are captured, the display is switched
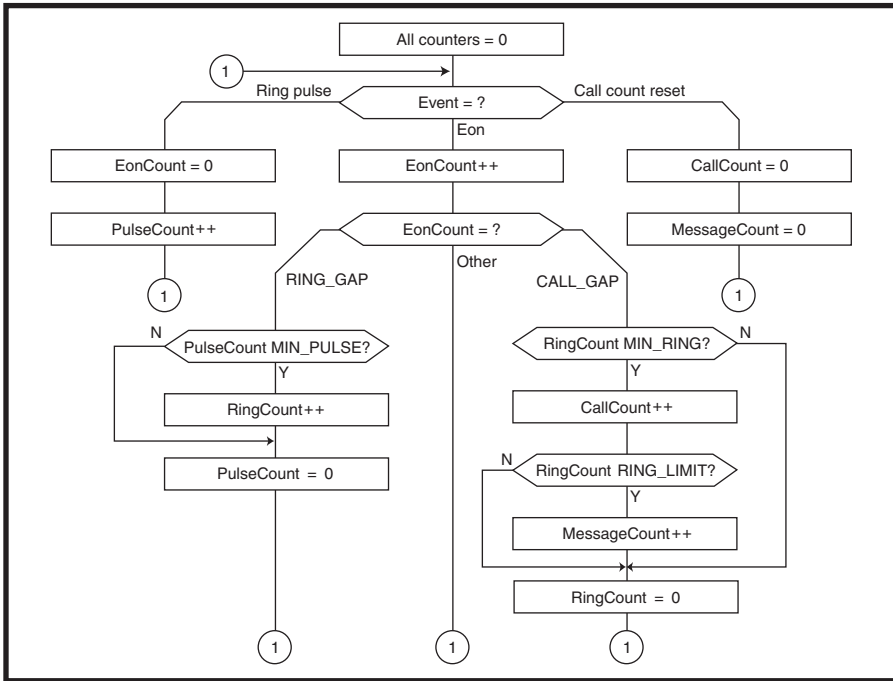
Figure 2—This is the ring interpreter algorithm and the role of the Call Count Reset button. In the firmware, this functionality is implemented by the `Main` loop.

between normal and reverse with exactly 50% duty cycle. When potential messages have been detected, the flashing is done rapidly, so the device draws immediate attention.

FIRMWARE DESIGN

The most challenging part of the firmware is the interpretation of the received ring signal, which eventually provides the number of rings in the phone calls. Our algorithm is based on the measurement of time elapsed between the ring pulses (more precisely, the half periods that turn on the optocoupler's LED).

In the U.S., the ring cadence is defined as 2 s of ringing, separated by 4 s of silence; however, the different distinctive ring patterns use periods of silence shorter than 4 s within the same ring. The documentation of cadence for distinctive rings and rings in different countries is somewhat sketchy. Based on the tidbits of information we collected, our firmware assumes that pulses separated by less than 1.3 s (called Ring Gap Timeout) belong to the same ring. If no pulses arrive for more than or equal to 4.5 s (called Call Gap Timeout), the current call is considered over. All other gaps are interpreted as silence between rings within

the same call. The duration of the rings is not measured; however, the compile-time constant `RING_MIN_PULSE` is used to define the minimum number of pulses an admissible ring must contain. This feature is used to reject fake pulses if the sensitivity of the ring detector needs to be increased due to an anemic ring signal. As a result, the circuit detects the transients caused by the phone going on- or off-hook (referred as "bell tap" in the literature). This may also be caused by the ticks of pulse dialing from a vintage phone.

The microcontroller's Timer1 is used in 16-bit mode to measure the time periods between ring pulses; it is clocked from the internal clock source through 1:8 prescaling. Because the accuracy of the measurements is not highly critical, we used the microcontroller's 8-MHz internal oscillator. This brings the clock of Timer1 to 250 kHz. With an overflow cycle time of 262 ms, this period is referred to as an "eon." We also employed the microcontroller's Input Capture 1 module as the Timer1-based "stop watch," although its readings ended up serving only experimental and debugging purposes. For the ring interpreter algorithm, the resolution of time intervals measured in eons is sufficient to properly identify the

exact nature of the ring pulse gaps.

To provide the most elegant solution, we decided to take full advantage of the PIC18F4520's interrupt system for event handling. The high-priority interrupt is associated with the Input Capture module, signifying the arrival of a ring pulse. The responsibility of the interrupt service routine (ISR) is to reset Timer1 immediately, so time measurements will be precise even for short time periods. In this application, this is not absolutely crucial, but it may be required in future enhancements. Considering that the VFD module is a textbook case of a slow peripheral (certain commands, such as scrolling, can take up to 100 ms to execute), clearing the timer in the `Main` loop would generally be a bad design. On the other hand, the response to a Timer1 overflow has no such particular urgency. Therefore, this event is implemented as a "silent interrupt," where the interrupt flag is set by the timer module, but the invocation of the corresponding ISR is not enabled.

One additional event is the push of the Call Count Reset button, which is used to acknowledge that the reported call numbers are understood and can be reset. This event could also be handled with a silent interrupt triggered by the microcontroller's INT1 input. However, to become familiar with the concept of multi-level interrupts and its implementation in PIC devices, Monica configured INT1 as a low-priority interrupt.

The operation of the firmware's primary functionality is shown in Figure 2. The corresponding code is implemented by the familiar infinite `Main` loop, which serves as the "background process" in low-complexity firmware environments. The device also supports the setting and nonvolatile storage of the Ring Limit. This mode is invoked if the Call Count Reset button is pressed during boot-up.

CONSTRUCTION

Due to the large footprint of the display module, we could have crammed the entire electronics on a PCB of equal size. However, we designed our PCB slightly larger, so the components that need access or visibility are not covered by the display's circuit board. The ExpressPCB
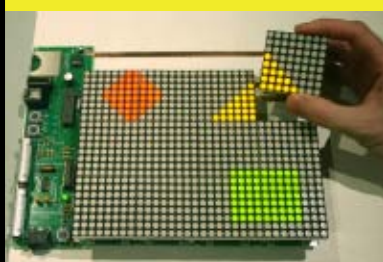
(www.expresspcb.com) board layout file is on the *Circuit Cellar* FTP site.

The two PCBs are held together by four #4 screws, utilizing the mounting holes provided by the VFD module (see Photo 1b). The distance between them is defined by the height of the phone jacks, the tallest components on the PCMF board. A plastic or metal casing can also take advantage of the screws. It is recommended that the VFD be covered with green translucent acrylic. This improves display visibility in ambient light.

One interesting challenge was the fact that the geometry of the VFD module (except the connector's header pins) follows the metric scale (i.e., the different positions are round values of millimeters). Fortunately, the ExpressPCB board design software supports switching between the two scales and does an impressive job.

FUTURE DEVELOPMENTS

Although working on this project was an excellent learning opportunity, the idea itself is not new. Voicemail indicator devices (albeit less fancy than this one) are available for less than $50. However, due to its relative simplicity, this design was a feasible task for one student to complete while demonstrating the entire cycle of product development: problem identification, idea inception, design specification, research of possible solutions, research of prior art, electronic and mechanical design, parts selection, parts procurement, firmware development, prototyping/testing, debugging, circuit board design, production, and final verification. This project proves that the devil is indeed in the details, and that on top of the expected hurdles, there are also plenty of unexpected surprises lurking around (e.g., the strange behavior of the VFD after a hardware reset).

Our voicemail monitoring system works as expected. It has put an end to missed voicemail messages.

If you want to build on this idea, true stutter dial tone detection and the implementation of call logging with time stamps would be two great opportunities for improvement. Both of these features were beyond the intended complexity of this project, therefore we dropped them from the initial specifications.

Peter Csaszar (peterc@ieee.org) is a senior firmware engineer at Synaptics in Santa Clara, CA. Previously, he worked as an assistant professor in the Electrical and Computer Engineering departments at Lawrence Technological University in Southfield, MI. Peter also worked as a senior software engineer at Motorola in Schaumburg, IL. His technical interests include hardware and software design for embedded systems.

Monica Flores (mfloresky@gmail.com) earned a degree in Electrical Engineering at Lawrence Technological University in Southfield, MI. She is currently working as an electrical engineer at Johnson Controls Hybrid Battery Group in Milwaukee, WI. She has worked as a co-op electrical engineer at DTE Energy in Detroit, MI. Her main areas of interest are electronics and embedded systems.

PROJECT FILES

To download code and board layout, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/215.

REFERENCES

[1] J. Macassey, "Understanding Telephones," *Ham Radio Magazine*, September 1985.

[2] J. Darwood, "Telephone Ring Detector Circuit," U.S. Patent 4066848, 1978.

RESOURCES

Mike Sandman Enterprises, Inc., "Message Waiting Products," www.sandman.com/messwait.html.

Noritake Electronics, www.noritake-elec.com.

SOURCES

C18 Compiler, MPLAB C18 compiler library, and PIC18F4520 microcontroller
Microchip Technology, Inc.
www.microchip.com

GU140X32F-7003 VFD
Noritake, Inc.
www.noritake-elec.com