

CIRCUIT CELLAR

INK®

COMMUNICATIONS

THE COMPUTER APPLICATIONS JOURNAL

July 1994 — Issue #48

In-house Personal PBX

Golay Error Correction Code

The Great Encryption Debate

Journey to the Land of Protected Mode Programming



\$3.95 U.S.
\$4.95 Canada

FEATURES

14

The Personal PBX

24

Using the Golay Error Detection and Correction Code

36

The Great Encryption Debate

40

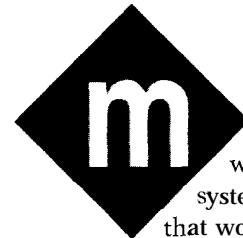
Take a Bus to the Nearest Star

The Personal PBX

While small-scale key system units (KSUs) are available commercially for the home, they usually have extra features and require special telephones. Instead, put your own system together for a fraction of the cost.

FEATURE ARTICLE

Richard Newman



Many years ago, I wanted a telephone system in my home that would let all the devices that use a telephone line share the two I had. Looking at commercial phone systems, I found they were all too expensive and had features meant for a business environment rather than the home. I decided to build one, and the first version worked great, using five relays for every phone. The version of the project I'm presenting here came along about 10 years later and uses only one relay per phone.

The small switching system, or Personal PBX, presented here is actually the prototype of a larger system that is used commercially in the private operator service industry to provide dial tone to hotels, dorms, and institutional environments.

In this article, I will describe how to provide basic telephone service to single-line telephones (called "2500 sets," which are served by POTS—Plain Old Telephone Service). The project will provide dial tone, receive DTMF [Dual Tone, MultiFrequency, often called Touch Tone) signals, ring the phones, and set up conversations between two or more sets.

I'll also leave open the possibility of adding more features, an interface to the public switched network, and an interface to a PC serial port for remote control and voice response.

TELEPHONE BASICS

Your telephones at home connect to the telephone company's central office (CO) by way of two wires called the wire pair, with one wire called tip and the other ring. Curiously, these

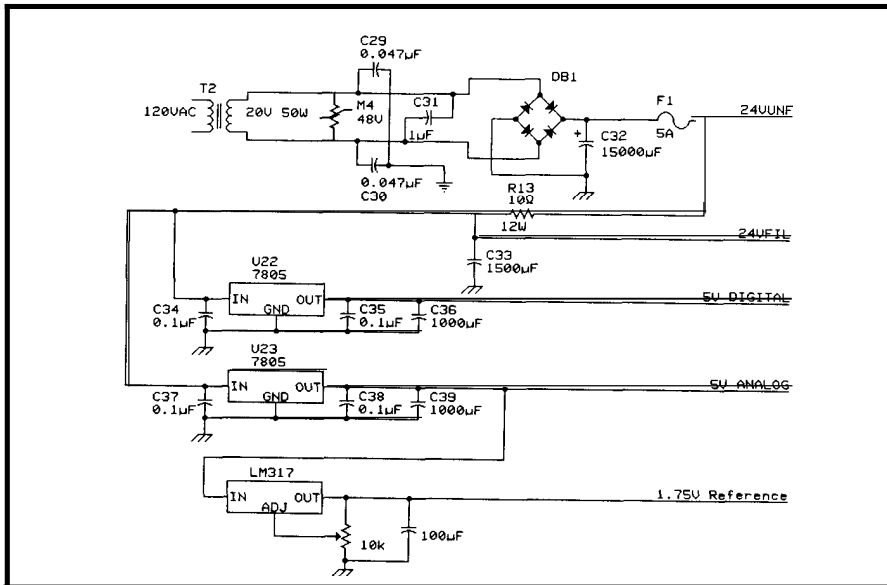


Figure 1b—The power supply is large enough to service 32 phones and all the control circuitry.

need [as long as you don't run out of I/O space or CPU time).

POWER SUPPLY

The power supply for the system is large enough to service 32 phones and all of the common control providing service for them (Figure 1b). It consists of a transformer and rectifier that provides 24-40 volts unfiltered DC, a filter arrangement that provides 24-40 volts of filtered DC (called the "talk supply"), a positive 5-volt regulator (or two) that powers all of the digital logic, and another 5-volt regulator that powers all of the analog parts including the switch matrix and common controls. A variable floating regulator provides a virtual ground for all of the op-amps and SLICs that actually sits at about 1.75 volts.

The analog portions of the system require very quiet DC supplies or the noise will be coupled into the conversation. Even though

people talking on the telephone sets cannot hear the noise, the DTMF receivers can, and a noisy power supply will increase the signal detection time considerably. To this end, I

provide a separate analog 5-volt supply that powers anything relating to audio, including op-amps, dial tone generators, DTMF receivers, and the virtual ground regulator.

The virtual ground is necessary because I am using a single supply for all of the audio paths between the system's subcomponents. Without a virtual ground sitting somewhere between real ground and the supply voltage, the AC waveforms would go below ground on the negative swing of the signal, causing some components to misbehave. Since all of the waveforms are referenced to the 1.75-volt virtual ground, they end up swinging from about 0.25 V to about 3.25 V.

The 24 volts filtered DC is used to provide power to all of the telephone sets, which need 2030 mA in the off-hook condition. A filtered supply is required to prevent the AC line hum from being coupled into the conversation. I chose a filtering arrangement

with a power resistor because it is less costly than a large inductive filter even though it dissipates lots of heat.

The unfiltered 24 V is used to power anything that is noisy, including the relays and the ring generator. When the ring generator is used, a large DC reserve from the filter cap is necessary to create a field of sufficient strength in the ring generator coil to make an AC waveform ring the bell in a telephone set.

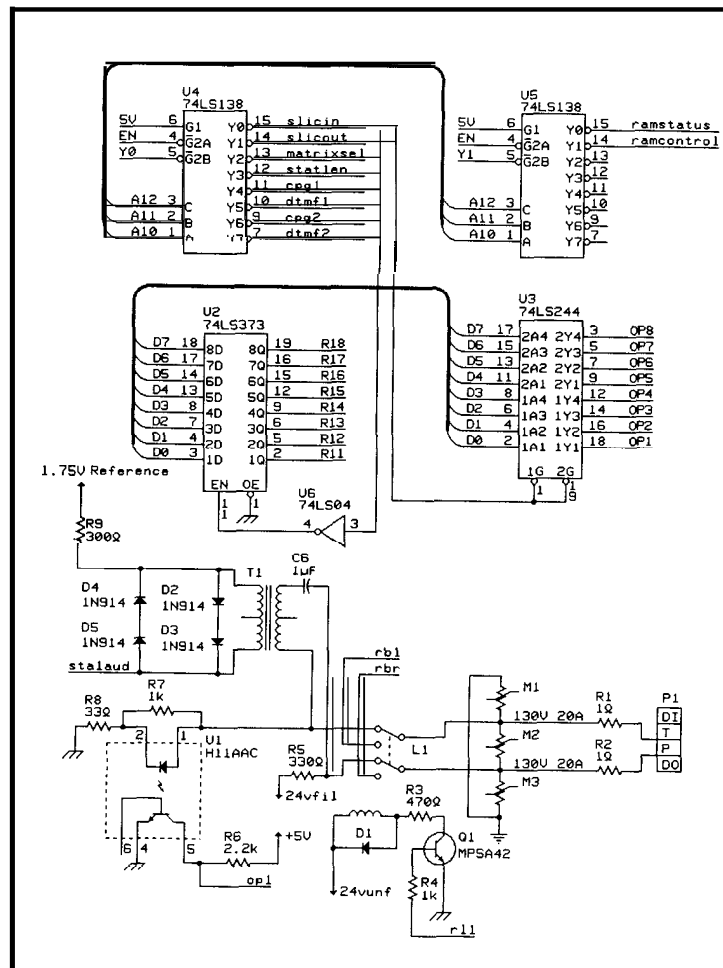


Figure 1c—It is the responsibility of the SLIC to electrically isolate the high-voltage telephone loop from the low-voltage switching matrix.

SUBSCRIBER LINE INTERFACE CIRCUIT

It is the responsibility of the SLIC (Figure 1c) to electrically isolate the high-voltage telephone loop that extends from the telephone equipment to the telephone set from the low-voltage and fragile switching matrix. Additional duties include indicating when the telephone is off-hook, and

RING GENERATOR

The ring generator (Figure 1d) is simply a small inverter that is used to step up the 24-volt DC unfiltered supply to 90 volts AC with a frequency that is controlled by the CPU. The frequency is normally 20 Hz to ring the telephone sets, however some telephone sets have message waiting lights that respond to a 40- or 80-Hz signal without ringing the bell, so the computer has the option of generating these alternate frequencies.

The ring generator circuit also has a loop current detector so if the computer is ringing the telephone set and the user picks up the receiver during a ring, the computer is notified to immediately stop the ring and connect the party to the caller. This "in-cycle" answer of a telephone is called ring trip.

COMMON CONTROL

The system's common control (Figure 1e) includes two dial tone generators and DTMF receivers that can be connected to any of the telephone sets through the switch matrix under the CPU's control whenever a telephone set demands service.

The dial tone generator chip is available from only one source: Teltone. Requiring just a 3.579-MHz crystal to operate, it generates all the industry-standard call progress tones including dial tone, ringback, busy, reorder, several intercept tones, a DTMF "#" (but no other DTMF signals), and others. Its downside is it's pricey, single sourced, and is not directly CPU bus compatible. If you care to, you may replace this device with a dedicated dial-tone circuit consisting of a few op-amps generating a 400-Hz sine wave and gate that into the point where the Teltone generator would have gone.

The dial tone generators are buffered through an op-amp and then coupled to the matrix through a 10- μ F nonpolarized capacitor. The input to the DTMF receiver is connected to a rail in the switch matrix along with the output of the dial tone generator.

The SSI20C90 DTMF receiver also has call progress detection and a built-in DTMF generator. This device is

very similar to the SSI202CP which has been used in several projects in the *Computer Applications Journal*. Requiring only a 3.579-MHz color burst crystal to operate and a single 5-volt supply, it is directly compatible with the CPU bus, but requires a separate I/O port to sense the "digit valid" and "call progress" bits. In some applications, these might instead go to the CPU interrupt structure.

The dial tone generator/DTMF receiver combinations are assigned on an as-needed basis. Once used, they are disconnected and may be used by another party. A conversation in progress between two telephone sets does not need a generator/receiver. Only two persons may be dialing at any given time, but there can be up to eight separate conversations happening at once.

SWITCH MATRIX

The switch matrix is responsible for getting the analog signal from one place to another (Figure 1f). The device used in the Personal PBX project has 96 analog switches in an array of 12 x 8. Called the M093, it is available from a variety of manufacturers including Mite1 and SGS-Thompson. The M093 is powered from the 5-volt analog supply and will pass signals from any input to any output in either direction with only 75 ohms of resistance through any one crosspoint.

The switch matrix is controlled by the microprocessor and has all of the telephone sets connected to a different X rail in the device. The Y rails in the device are the speech paths and may be daisy-chained to multiple switch matrix chips to form a larger switch platform. To connect two telephones together, you close the X rail of the telephone sets on the same Y rail and the sets are then electrically connected in a conversation (see Figure 2).

One oddity of the M093 device is that in addressing the X side of the crosspoints, it skips addresses 6 and 7. As a result, when referring to X0, you use address 0, but when referring to X6, you use address 8. This oddity is handled by the low-level library routines and is transparent when writing software for the switch.

(408) 245-8268

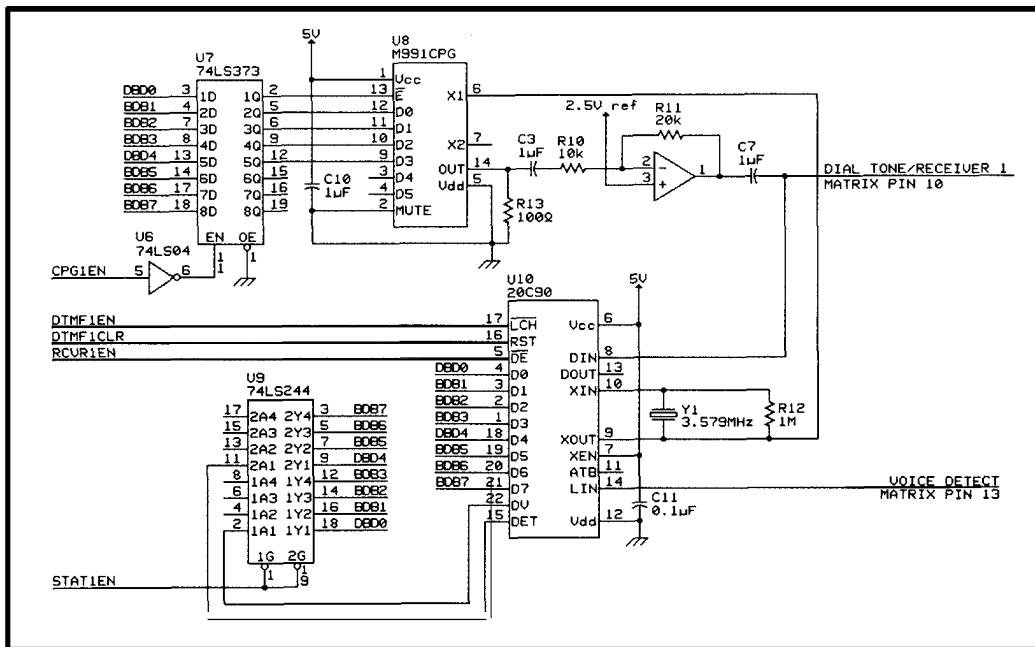


Figure 1e—The system's main control includes two dial tone generators and DTMF receivers that can be connected to any of the telephone sets.

In this project, I will use only one MO93 for now. Each of the rails X0–X7 connects to the SLICs 1-8, respectively. X8 and X9 are connected to dial tone generator/DTMF receiver combinations. X10 is connected to a speech playback device for recorded announcements, and X11 is left available for experimentation.

RECORDEDANNOUNCEMENT

An optional extra feature I haven't mentioned up to this point is a recorded announcement. In the full-blown commercial version of this switch, I use this feature to prompt the caller for a calling card number and keep them informed of the progress of the call. I selected the easy-to-use (though currently impossible to get) DAST device manufactured by Integrated Storage Devices.

The ISD2590 features 90 seconds of record/playback that, when accessed sequentially, requires only five control lines and two status lines. One nice feature of using the sequential message queuing mode is that very little CPU overhead is required to keep track of what this device is doing, so we can take our time in setting up

the message to be played without unexpected results.

There is only one DAST in the system, but like all the other common peripherals here, it is assigned to the call only when it's being used and then is released to service other callers.

THE SOFTWARE

The code that controls the Personal PBX is a large state machine that has been broken down into smaller C functions. The major functions include:

`f_exhm()`: The hook switch monitor. Waits for a user to lift a

phone, then sets up the phone to be serviced by other routines.

`f_exdm()`: The dialing monitor. Everything having to do with dialing and call setup is handled here.

`f_exsm()`: The extension signaling monitor. Ringing the telephone sets is this routine's job.

`f_excm()`: The conversation monitor. During the call, this routine is in control, but its biggest job is cleanup when the call is finished.

The main procedure contains the startup code. When the system is

powered up or the processor is reset, the startup code immediately turns off the ringing generator coils, resets all relays to the off state, and clears the state machine variables to a known state.

When the dispatcher detects that it is time for a phone to get a share of the CPU time, it sets a global variable called `ext_check` that equals the port being serviced and is also the index to the arrays that hold the state the telephone set is in.

Each phone in the system has memory variables in an array assigned to it. If there are eight phones in the system, then the array contains eight elements. The arrays `ext_process[x]`, `ext_substate[x]`, and all others beginning with `ext_` are permanently assigned to the port. Normally the `ext_` variables are indexed with a global variable called `dispatch-process`. When port 3, for example, has a share of the CPU time, `dispatch-process` equals 3.

There are other variables that are also arrays, but they are not permanently assigned to any one phone. Instead, they are assigned to a phone when that phone goes off-hook and they stay assigned during the

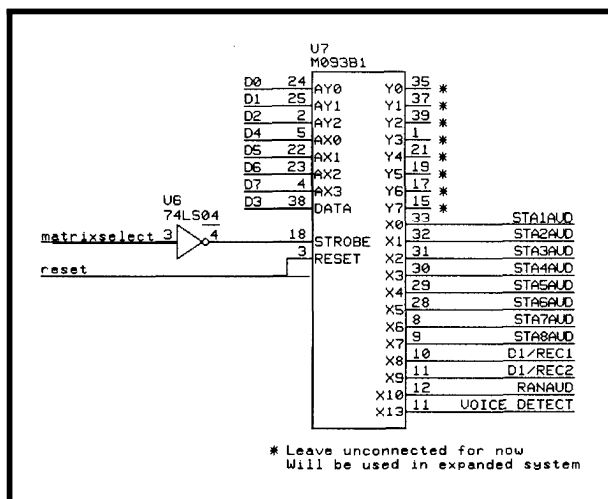


Figure 1f—The switch matrix is responsible for getting the analog signal from one place to another.

entire call. These are called *call control blocks* and are sets of arrays indicated by `ccb_` before their name (e.g., `ccb_digits[x]`, `ccb_acp[x]`, etc.). Call control blocks are assigned to the phones by a port variable called `ext_status[]`. If CCB 1 is assigned to port 3, then `ext_status[3]` will equal 1.

Because each phone has a different index to the same set of variables, this state machine could be said to be servicing different *contexts* each time it is dispatched. Each context has very different concurrently executing functions. One phone might be dialing while another is ringing, but they are being serviced by the same software. You can expand the number of ports being serviced by making the array that holds the common variables larger.

SYSTEM OPERATION

The software attempts to emulate a central office, so it is transparent to a user dialing from a telephone connected to it. It might be helpful to follow a call through the state machine.

First, the source and destination telephone ports are idle. They are both executing state 0, which is looking at the current detector and waiting for it to detect off-hook.

When the person on port 0 picks up the phone, HookMonitor first locates a free call control block that will be used to store information about the call in progress. If HookMonitor cannot find a free call control block, it simply does nothing and falls through to the end of the code block. The dispatcher then goes on to service all the other phones in the system. When it gets back around to port 0 again, it starts the same process of looking for a call control block. Ideally, it will always find a free call control block. In the worst case, it would find one when someone else hangs up and their block is freed.

Next, HookMonitor clears the port's `ext_route[]` variable. This

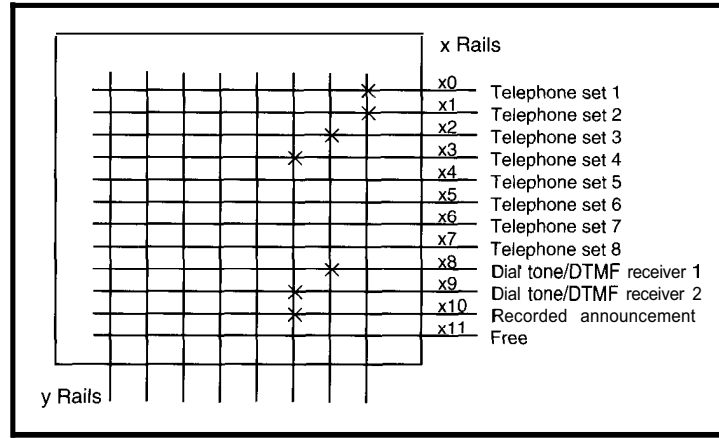


Figure 2—The switch matrix has *all* the telephone sets connected to the X rails while the Y rails are connected to speech paths.

allows any process to see that the call has yet to go somewhere.

The first steps HookMonitor takes are to locate a free dial tone sender, DTMF receiver, and speech path for the conversation to take place on. Once all of the above is done, the user hears dial tone from the earpiece of his telephone set. HookMonitor now sets the extension state variable `ext_process[]` to 33, which is the entry state of DialingMonitor.

DialingMonitor decrements the extension timer, which gives the user a finite amount of time to dial the first digit. If nothing is dialed within this

predetermined period of time, a reorder tone [fast busy] is sent to the port. When the reorder tone is sent, the same timer is loaded with a finite amount of time again. When it expires a second time, all of the resources assigned to the call are freed, including the call control block and speech path. There is now a phone off-hook, but no hardware assigned to it, so a *lockout condition*, or

state 36, is assigned to it. This state does nothing more than wait for the phone to go on-hook, which will start the off-hook monitor all over again.

When DialingMonitor does detect a digit, it is stored in the `ccb_digits` area and the routing tables are searched for a match. In my example, let's say the user dials a 4 first. After the 4 button is pressed and released, DialingMonitor will store the digit and then call `search()`, which will look in a single-digit table. In this case, `search()` won't find a match for the 4 because there is nothing in the single-digit table to find. When the user then

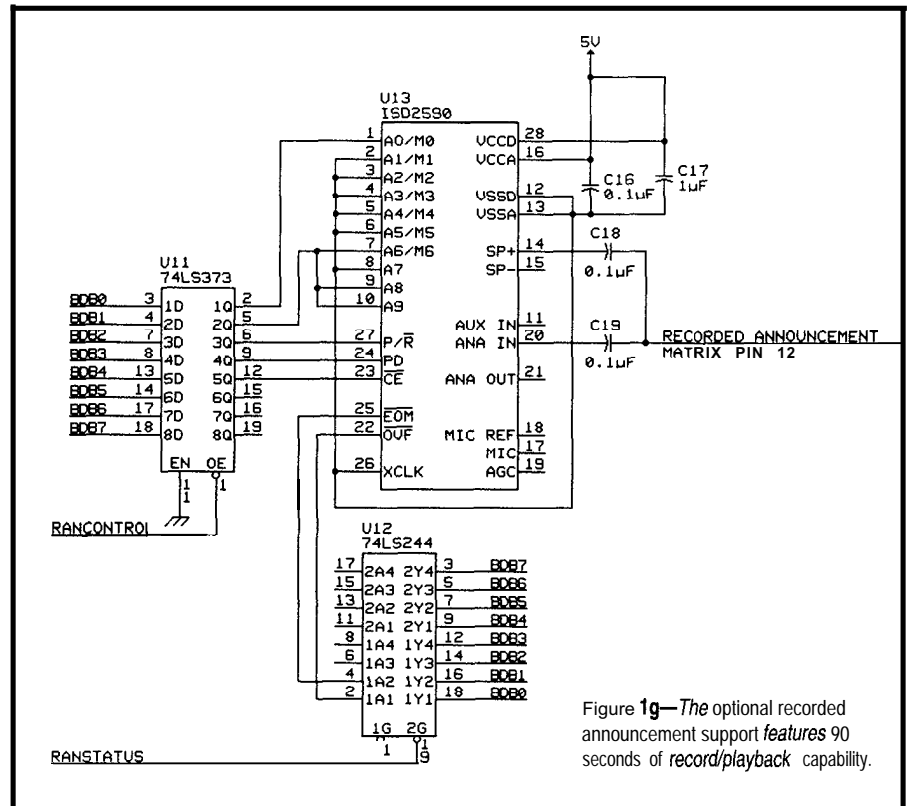


Figure 1g—The optional recorded announcement support features 90 seconds of record/playback capability.

dials a 3 as the second digit, search() will now look in a double-digit table, and will find a match for the 4 as the first digit and the 3 as the second digit. When this match is found, search() will update ext_route[] to be the action stored with the found digits. In our case, the action is to send the call to another port on the system.

When DialingMonitor sees that the time for a digit to be dialed has expired, it will look to see if a route has been chosen and, if so, whether it is a call to another port on this system.

There is a reason I wait for the interdigit time to expire rather than just place the call as soon as I find a match. One of the original system design goals was to make this small system work transparently to its users, but still provide all of the features of a private branch exchange. In order to do this, I wanted to allow extension numbers to be able to conflict with local exchange numbers. For example, you might have a local telephone number like 423-5555 and at the same time support extension number 42. When the software sees that the dialed digits match 42, it selects the route to the other extension. If the user keeps dialing, the call would be sent to the local exchange via a trunk rather than to extension 42.

Since a decision has now been made where to send the call, DialingMonitor now looks to the extension variables for the selected destination extension. Since the process state of the destination extension is 0, we know the destination is idle and can change its state without interrupting a call. Changing its state to 66, which is the extension signaling monitor, we also change the station's ext_status[] variable to be the same as our own, which points to the call control block number we were assigned along with dial tone. Next, we change the state of our calling phone to be 37, which provides the caller with ringback tone.

Now the destination is ringing and the caller is listening to ringback. When the destination answers, the extension signaling monitor debounces the line by letting it sit for a moment without any ringing current on it. Then it connects the SLIC to the audio

path found in `ext_status[]` and changes the state to the conversation monitor, which is state 98.

Meanwhile, the calling station is still hearing ringback. While doing that, `DialingMonitor` is looking at the destination station's data, waiting for the state to be 98. When it sees that state, it knows the called party has answered. It then releases all the common controls, freeing them for others to use, and connects the calling extension to the same audio path. Finally, the calling station's state is also changed to 98.

At this point, we have a conversation and are both in state 98. The only thing the system has to do is wait for either caller to go on-hook and complete the call. When that happens, each station's state changes back to 0 to ready the ports for another call.

THE SERIAL PORT

A serial port is provided for doing "OA&M" (Operations Administration and Maintenance). The code provided has modest requirements for adminis-

tration. To change extension numbers, you must recompile the code.

Typing "N" from the serial port will load the RAM with the default extension numbers 40-45. It also loads an end-of-dialing digit so you can dial a "#" as the last digit and the call will be placed immediately rather than wait for the interdigit timer to expire.

THE FUTURE

This system demonstrates basic telephone switching. The hardware could be used with most any computer controlling it. The software is very basic since, right now, it only supports internal extension calling.

We can expand this project to include external calling to the telephone company so our calls can reach the real world. To do so will require another interface between the switch matrix and the central office line pair.

The serial port is also awfully lonely. It could be attached to a PC to use the Personal PBX as a front end for voice response and remote control in a home. Stay tuned. □

Richard Newman lives in Dallas, Texas where he is an independent engineer designing equipment for the telephone industry. He may be reached at (214) 522-8935.

SOURCE

A printed circuit board is available from the author. Contact him at (214) 522-8935 for more information.

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful