Using Your Micro Controller for Phone Work With Inexpensive Modems and Other Devices

Jan 2004- this project was developed using the BasicBoard (AKA BBoard) from EL Products; written in the compiler that comes with it, Atom BASIC, using an inexpensive modem. The modem cost$18, with free shipping. It was found on Pricewatch.com. The program presented here is in Atom BASIC only. I leave it to you to rewrite this to a language you are familiar with. Download, or study, the Atom manual at the BasicMicro.com site for explanations about usage of these commands.

There are no schematics supplied, nor will I supply any, as the BasicBoard is a copyrighted commercial product. The web oozes simple micro controller layouts with serial ports that are suitable for this project. Happy hunting/building/buying.

I want to thank the many people who took the time to advise me in the threads when I was trying to get this working, and for their enthusiastic support.

First, a few notes:
- there are dangerous voltages on the phone line. Stay with the interface solution here, i.e. using a modem. Another, costlier solution can be found for audio work during this article.
- There are severe fines for connecting ANY device to the phone lines, even in a private home, that has not received FCC registration and licensing per part 68. We're talking jail and a $10,000 a day fine, folks.
- I submit this document for the enjoyment and education of hobbyists and enthusiasts interested in embedded processor work. I MAKE NO PROMISES OR REPRESENTATIONS OF COMMERCIAL FITNESS OR USABILITY, and all that other "just leave me out of this" legalese.
- Pictures are supplied at the end of this article.

****

There have been a few forum requests recently for ideas on how to do jobs that require interfacing over the phone. The first: automating the process of leaving a phone number at a pager or cell service. The second request, call a number and play a tune, will be touched on.

Two criteria came out of the threads:
1. A circuit allowing connection to the phone lines good enough to send tones out without destroying the micro controller. Several methods were mentioned, but they were hard to build, illegal, or failed to pass clean dialup tones. It was suggested that people use an FCC approved $40 circuit (discussed later) to interface to the phone lines. This is unnecessary for just leaving a number, but critical for audio work.
2. A way to generate usable DTMF (Dual Tone Modulated Frequencies). People were complaining that the DTMF(2) commands in their compilers were not producing usable tones.

I say, use a modem. It has the safe, approved phone line interface; UL approval (in case you want to go commercial, here!); a good DTMF tone generator; and a command set for device control. Best of all, you use a serial port to talk to it, something that is well documented.
I tested this against four different modems. Three were recent 56k units. The fourth was an older 14.4k Supra FAX/ modem. All worked, with small software changes.

First, we need to discuss serial communications on a PC, as this plays a critical part in setting up the modem initially. The three 56k units wouldn't work without the proper set up string.
A PC uses a serial connector configuration called DTE (male plug). This can connect to a modem (DCE) (female socket) with a straight thru cable. The BBoard is RS232 DCE (female), as is the modem. A simple null-modem cable would have sufficed to connect them together, but I wanted to experiment. I developed a simple patch board with DB9 connectors at both ends. I used this to reconfigure the signals for compatibility. Using this helped a lot. At first I

experimented with looping back handshake signals that the modem might need to operate properly. I pulled wires off one at a time, and discovered the modems I tested against needed only two lines: the controller's TxD to modem's RxD, and Ground. No handshaking lines at all!

You have to prep the modem with the proper setup codes. For this you need a computer and serial communications software. I used a PC running Windows 98SE. The communications software was HyperTerminal, which is supplied with this version of Windows. You should find it at START/Programs/Accessories/Communications/HyperTerminal (a folder). If it isn't there, you must install it from the Windows OS CD.
Go into the folder and start HYPERTRM.EXE. Click 'cancel' in the first box. Select 'File", then 'Properties'. A new setup box appears. Find the 'Connect using:' box and select whatever COM port you are using. I used COM1. Now, select the 'Configure' button. Set the bit rate to 2400, and set the flow control to none. My modems worked fine with noting more than 'Receive Data' and ground connected. However, it seems from what I've read that there are differences between modems, so yours may demand some form of flow control. Click 'OK' twice.
You should be at the terminal entry window with a blinking cursor. You should have the appropriate serial cable installed from PC to modem, and the modem's AC adapter connected at both the power jack and AC outlet. Power up the modem. When the lights settle down, go back to the terminal screen. Type in the following string, all on one line:

at e0 q1 s0=0 &w0 &y0

Now press Enter/Return key to send this to the modem. You can now turn off the modem and disconnect it from the PC. You're ready to use this modem with the program below.
Be warned: after pressing Enter, nothing you type from HyperTerminal will appear on the screen. Everything you typed originally went to the modem, which then 'echoed' it back for you to see. This changed when you hit Enter. We have turned off 'echo', so nothing is there now.

FYI: 'at' is the command notifier; &f0 = use factory defaults; e0 turns off echoes (an absolute must!); q1 = quiet, no result reporting (another must); s0=0 means never answer (the modem now does phone output only, a must); &w0 and &y0 store all this in a startup register and use this register for power up configuration.

Let's get to the code:

```
'*****************************************************************
'Modem Dialer: dial out, leave number, hang up. Use for simple pager services.
'*****************************************************************
PAUSE 500 ' Optional: let BBoard stabilize. If No bootloader, drop this line
LED con P0 ' Pin for indicator LED. Put any pin here you want.
sw1 var In18 ' Assign start up duties to switch1. Switch is on P18 for me
's_out con (pin#) ' If you have an Atom chip, just drop this line. If you're
' breadboarding, you'll have to say which pin your serial stream is transmitting from.

pace var byte 'optional millisecond value for delay between transmitted characters.
pace = 100 ' A full tenth of a second. Try something smaller if you want.

HIGH LED  '  'signs-of-life' indicator
GOSUB WAIT  ' wait for switch press at this subroutine

MAIN:
HIGH LED  ' Here we go!
```

```
' This is the SEND LINE
SEROUT s_out, i2400, pace, ["atdt 7672676 ,,,,, 01234567890 ,,", 13]

' You have to do this all in one line. Any data sent to the modem after it dials out causes it to
' hang up. The 13 is carriage return, and tells the modem to process the command line. Each
' comma inserts a small pause, in seconds. This 'seconds value' is stored in a register
' in the modem. Mine is 2 seconds. Experiment with your comma settings to get it right.

PAUSE 16000 ' Spin the tires 16 seconds, waits for modem to do its job.
SEROUT s_out, i2400, pace, [13] ' the carriage return is needed to get the modem to hang up.

' End of job. Wait for switch again
GOSUB WAIT
GOTO MAIN ' return from WAIT:, do it again

'****** SUBROUTINES ******

WAIT:
' Monitor switch1. Stay here until switch pressed. Just a switch to ground, with weak
' pullup resistor at input pin, say 10k Ohms or so. Use any event to trigger the call out. This
' switch is just one of many possibilities.
IF sw1 = 1 THEN
PAUSE 300 ' Small pause.
TOGGLE P0 ' change voltage to opposite of what's there. Makes LED blink.
GOTO WAIT
ENDIF
RETURN

' An 'end' instruction for safety's sake...
END

********** END PROGRAM **********
```

The switch and LED are just a convenience. If you want to leave them out, drop all lines referring to them. Put any triggering event you want in the WAIT subroutine to start things.

Operation: after reset or power up, the LED starts to blink. Press the switch until the LED lights up solid. Release the switch. The modem should indicate it is off hook. If the speaker is operational, you'll hear the dialup tones after a few seconds. When the line is answered at the other end, the speaker may stop, or not. This program starts a 16 second delay after the string ends. At the end of the delay, the phone is forced to hang up with a carriage return (13) and the LED starts to blink again. The controller is waiting for another press of the switch.

This program calls the phone company's time-of-day service, POP CORN. Use it for checking out your circuit initially. Insert the service number you want when ready.
The modem delays 10 seconds before sending out the numeric string. Replace this string with the number you want to leave. The delay is programmed into the modem using commas, five in this case, two seconds per comma. This comma-delay value can be set in a register. For this exercise, just add or subtract commas until you get a delay that works for you.
In fact, you are going to have to experiment a bit here for the right values. You will have to determine how many commas to insert to wait until the service answers, and then add one for

safety. This can be done just calling up the service and measuring the time it takes from the last digit dialed in, until the phone is answered and the tone or voice message completes. Do this two or three time. Use the longest time measured to set by.
You'll want to add some commas after the string number to ensure the modem remains online until the controller kills it. The total comma delays must be longer than the program delay.
I have found sending anything between the two carriage returns is bad news. This causes the modem to try to "handshake" (screams like a Banshee) next time you call out. If it does this, you'll have to power cycle the modem. EVERYTHING has to go out in the one line.

All this works well enough for pager-only services. If you have Sprint, like me, you can leave a numeric message but the process is quite daunting. The phone you want has to be turned off, and you have to respond to canned messages before you can leave the number. It can probably be done, but…eeesh!

Good news: you can have several INDIVIDUAL lines of code with atdt commands in it. Create as many standalone callout strings as you want. Have calls to one, or several, services, leaving the same, or different, numeric strings. A string could just be "911" to indicate an emergency. Go off hook, send out JUST ONE LINE, and then hang up. Do several in a row: after hanging up, have a short delay, and then send another string. Keep doing this way until everything has gone out. Or, have multiple inputs trigger one of a selection of multiple atdt strings. Pressing button one sends out string one; a door switch activates string two; an overflow alarm input sends out string three; etc.

Notice the slow baud rate. I found in my experiments that my device didn't handle 9600 baud well. I eventually went to 2400 baud. This worked fine. Then, to be on the safe side, I included the 'pace' command. This tells the RS232 handler to insert small delays between each character. If I used the hardware UART, I could have probably gone much faster. But, considering the job, who cares? It still dials faster than you or I. Consider experimenting with these values, if you wish. No harm done.

Now, as for playing a tune: a modem falls down here. You can get modems with jacks for a speaker and microphone. I doubt the discount modems have this feature. The one I had with jacks didn't readily take an outside source and play it over the phone. I tried it, but nothing came over the speaker or phone. You might be thinking at this moment about getting inside and adding a jack to tie into the impedance transformer…
NEHHH! Says Mr. Buzzer! I ADVISE AGAINST THIS, AS YOU MAY VIOLATE FCC COMPLIANCE WITH THIS MODIFICATION!

There is a solution: the CircuitWerkes MPC-2 DAA, $40, not including S&H. This circuit board is an FCC approved device that allows interfacing to the phone line. This immediately allows you to do legal audio work on the phone, without the hit-or-miss (and possibly illegal) modem modification(s). It also has several useful features above and beyond simple phone interfacing, one of which is a signal that tells the controller when the connection is made (Sweeet!). There is one limitation: it doesn't generate DTMF tones, so it won't dial out on its own. For this, you'll need a DTMF generator IC, and some way to control it. Or, a micro controller like the BBoard, which has a compiler with DTMF commands. Of course, you could use the MPC in tandem with the modem. Have both the modem and MPC go off hook, use the modem to dial out, kill the modem, wait for the MPC's connection signal to go true, put your audio out over the MPC, hang up the MPC. DONE!
The following site will let you jump to the MPC's data sheet and a document about FCC approval and a stern warning about failure to meet part 68 approvals:

http://www.broadcastboxes.com/mpc-2.html

You will also find their address and phone number here. You have to order directly from CircuitWerkes. No credit cards or PayPal taken. It's COD, check, or money order.

CAUTION: THIS IS AN OPEN BOARD! Therefore, you will be exposed to phone line voltages unless you install this device in a protective enclosure!

Your audio signal could come from a tape player with the motor controlled with a relay, a musical card from a crafts store, a voice recorder, or a microphone listening to your living area! Be careful to isolate the source from the MPC with a non-polarized capacitor around one uF.

One last thing: the Hayes AT commands play a large part in this. They can have an important part in your future modem work. F'rinstance, there are commands that allow you to turn the modem's speaker on all the time, and another sets the volume. This allows you to monitor what's happening after going online. (Then again, you could just pick up the phone and listen!)

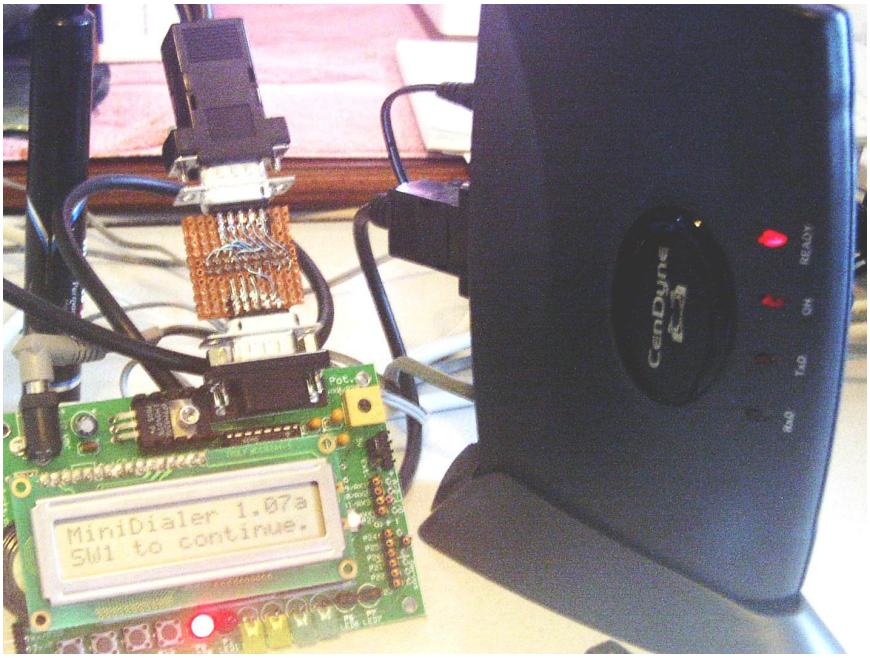
- Does the number string get output in its entirety before the modem is forced offline?
- Is it going out AFTER OR DURING the answering tone or audible voice-query?
- What is the volume of the outgoing tune/message?

Then, when you're done testing, disable the speaker, if you want.
Also, the modem can output status messages that signal connection, busy lines and other errors. There are several sites online that show, and discuss, these commands. Do a Google search on "hayes AT" to find some. Study the possibilities.

****

Feel free to email me (kjennejohn@yahoo.com) with requests to expand on topics touched on here. If I receive enough requests on a subject, I may do an article**. Possibilities:** using the BasicBoard as a 'snoop' device; the controller as an alarm system that also dials out; using an audio device over the phone in detail; using a keypad to edit or add numbers to be called and to be left; etc. **However, I cannot answer individual requests for specific design questions!** Please post your questions in the forums online.

Let the creative juices flow!
Ken Jennejohn



** The Working Units **

The BasicBoard and inexpensive modem are the heart of the basic service dialer. The BBoard can be wired to a number of different external devices. Connected to the serial port at the top of the BBoard is the patch board I made. Amazingly, it took only two wires to communicate to the modem!

** The Complete Ensemble **

Pictured along with the BBoard and modem are the four-connector cable and RS232 patch/monitor box I used during test and troubleshooting. The serial cable included with the modem was a pleasant surprise: It had a DB-9 connector at one end, with both a DB-9 and DB-25 connector at the other. This allowed me to connect the modem to the BBoard AND the patch box at the same time. This simplified trouble shooting considerably.

The cable is a foot long piece of 9-conductor flat cable with four IDC (Insulation Displacement Connector) connectors, two females and two males. This cable and the DB-9 patch board allowed me to use the BBoard as a 'snoop' device. I used the three to watch serial traffic between the modem and PC during early troubleshooting. This is how I discovered the modem's echo problem, where the BBoard and modem were talking at the same time.

The patch/monitor box is typical for this application. It is a commercial product, easily obtained , for $25 - $30. Radio Shack and DigiKey are possible sources. This is one of those simple, but invaluable, tools that aid any technician immensely in determining the states of the serial signals. There are LEDs for the most commonly used signals. Their colors tell the user if signals are 'true' or 'false'. These give warning if both devices are DTE or DCE, and show which handshake lines are in use. There is a switch for every line, so signal isolation is possible. And, there are sockets and patch wires for cross connections and signal sharing.