A DTMF to RS-232-C Converter and HF-Link Controller

If you're interested in experimenting with DTMF control signals or building an HF-link controller, you'll find this circuit fills most of your needs.

By Nigel Thompson, KG7SG

1009 317th Ave NE

Carnation, WA 98014

Having recently read several articles describing HF-link controllers, I decided it was about time to try some experiments myself. I had all the radios I needed and a computer that I could use to control the link, but I had no way to get DTMF control signals from the uplink into the computer. So, I built a simple DTMF to RS-232-C converter that converts DTMF tones into ASCII characters and sends them to my computer over a serial line. I added a few components to the DTMF-to-serial converter to allow the same computer port to control my IC-735 transceiver. A block diagram of my system is shown in Figure 1. Note: According to FCC rules, all auxiliary operation like that described in this article must occur on frequencies of 222.15 MHz and above.

Not only is this unit simple to build, *it requires no adjustments*. If you want the DTMF converter and not the control functions, merely omit the relays, all of the transistors and the few components that support them.

DTMF to RS-232-C Conversion

Figure 2 is the diagram of the entire converter. From the uplink receiver, the audio—which contains the DTMF control tones—is fed to an SSI202P/75T202 DTMF decoder (U1). The decoder output is a 4-bit code representing 1 of 16 possible DTMF codes. The DV output is a strobe signal that goes high when a valid DTMF code is present on the data lines (D1, D2, D4 and D8). The 4-bit code is fed to a pair of parallel-load shift registers (U4 and U5). On reception of a valid DTMF code, it's loaded into the shift register. When the tone ends, the shift registers clock out the ASCII character code in a serial bit stream.

U6, a MAX232, converts the TTL-level signal to RS-232-C levels. The data-rate clock signal is generated by an oscillator (part of U2) and U3, which divides the crystal frequency down to the required data rate. My system runs at 1200 bauds, but you can use a different output of U3 to set the data rate you desire. Figure 2's setup uses PC-board jumpers to allow selection of 1200 or 9600 bauds.

The DTMF codes are converted to ASCII characters by adding 30H (hexadecimal) to the code produced by the SSI202P. This converts some of the DTMF codes to their correct ASCII equivalent, but other characters must be translated by the computer software. Table 1 shows the DTMF codes output by the SSI202P and the ASCII codes sent by the converter.

ICOM Radio Control

Only the input lines of the computer's serial port are used in receiving the DTMF signals. I use the serial port's output lines to control my IC-735 HF radio. Q1 supplies the level conversion needed for the radio's remote-control line. Running the serial interface at 1200 bauds matches the factory setting for the radio's remote-control interface. With this arrangement, you can only *send* data to the radio, *not receive* data from it, so there's no way to validate the radio's settings. This isn't a problem for my needs. If you require better control of the radio, use another RS-232-C port on the computer and an interface to the radio that allows two-way communication. Documentation for the interface protocol used by ICOM radios can be obtained from ICOM's office in Bellevue, Washington.[1]

Controlling the Transmitters

I used the RTS (request to send) and DTR (data terminal ready) serial lines to switch two relays. The first relay controls the downlink. When one of the relays (K1) is energized, the downlink transmitter is active. Another relay (K2) enables the squelch on the uplink radio to activate the PTT line on the HF transmitter. So, by setting DTR low, I can disable the HF rig while keeping the link active for testing.

The squelch output of the uplink radio is level-converted by R11, D5 and U6C and connected to the RS-232-C DCD (data carrier detect) line. The computer can test the DCD signal to see if the uplink receiver squelch is open. I use this signal to determine when a transmission on the uplink has ended. Then, any DTMF command sequence is processed. This signal is also used to control an inactivity timer in the software that shuts down the link after a specific delay if, for instance, the remote unit gets out of range. Of course, you can use the two relay outputs for whatever you like, and the input signal connected to the DCD line can be used to sense dc voltages above about 3 V.

The Control Program

A Visual Basic program running on my PC controls the entire link. A sound card in the PC sends the station ID on the downlink and speaks the link status to me. FCC regulations require that the downlink be identified at least once every 10 minutes. A timer I built into my control software plays a .WAV file—a recording of my speaking my station ID. The software allows you to send this .WAV file as often as you like. While testing the link, I found it useful to send the ID every 60 seconds. Once the link was working, I set the software to send the ID every 9 minutes.

Another software function monitors the uplink. If there's no activity on the uplink for a set period, say, 10 minutes, the entire link is shut down.

On demand, the control program speaks the current frequency and mode of the HF rig. By having the computer do this, I avoid the need for an HF rig that is equipped for speech synthesis. Simple DTMF codes set the operating frequency and the mode of the HF rig, and also shift the operating frequency up or down in small increments. Other commands are used to enable the link and the HF transmitter.

Speaking the frequency (and so on) might sound a bit fancy, but it's quite easy to do and very useful when you're using the link. I recorded myself speaking all the digits, a few letters and so on. The software assembles the individual digits to say a number. (I had a lot of fun with this part, constantly trying to impress my wife and kids with the latest addition to the computer's vocabulary!) The software source code is provided "as is"[2] you're welcome to do whatever you like with it. My spare time is very limited, so I cannot offer any software support.

The Link in Use

I built and tested my entire setup using a solderless breadboard.[3] This part of the project went very well—I wish I could say the same for the uplink and downlink, but I can't! After many experiments, I found that the downlink transmitter was seriously desensitizing the uplink receiver. Consequently, although I could hear my downlink over 10 miles away, I couldn't control it via the uplink more than about 2 miles away—even when using full power from my rig! I'm now working on some filters for the uplink receiver, which I hope will cure this problem.

I also discovered that when the signal levels are a bit weak, the DTMF decoder becomes quite unreliable. Sometimes it turns one tone into two characters. At other times, the decoder misses one character in the middle of a sequence. To combat this, I made sure that all my commands were validated as well as possible by the software before I took any action based on them. When the software doesn't understand a command, it repeats what it heard on the downlink and follows the repeat with *invalid command* and the station ID. So as I drive around, I can tell when the link is getting marginal and shut it down before I completely lose control.

Antennas

My downlink uses a simple dipole. The uplink uses an ICOM AH-7000 discone for no other reason than it was already sitting nice and high above my house and was easily pressed into service.

Summary

If you want to experiment using DTMF tones for computer-controlling a device, this circuit provides all you need to get going. With a little effort and some software, you can use it to control a complete HF link.

Notes

(1) ICOM America Inc, 2380-116th Ave NE, Bellevue, WA 98004; technical support tel: 206-454-7619. Ask for the *CI-V Reference Manual*.

(2) The software (*DTMFCONV.ZIP*) is available on the ARRL BBS (203-666-0578) and the ARRL FTP site at oak.oakland.edu.

(3) PC boards for this project are available from FAR Circuits, 18N640 Field Ct, Dundee, IL 60118-9269; price $6, plus $1.50 shipping.

In 1990, Nigel Thompson got his Novice license, KB7JSA. In May 1991, he went for his Technician Plus ticket, and in August of that same year, received his Advanced class license, KG7SG, after taking the General and Advanced class exams the same night. By March of the following year, Nigel had his Extra Class license. In addition to Amateur Radio, Nigel has "too many hobbies!"

Nigel received an upper second class honors degree in electronics from the University of Southampton, England. He graduated in 1977, specializing in semiconductor physics.

For many years, Nigel worked as an electronics engineer designing and building microprocessor-based equipment. He owned and operated his own electronics business for a few years.

For the last six years, Nigel has been working for Microsoft in Redmond, Washington, as a software design engineer, and more recently as a writer of Windows programming articles. Nigel recently published his first book Animation Techniques in Win32 (Microsoft-Press).

Nigel and his wife, Tammy, are blessed with three children: Ron, Nell and Mark.

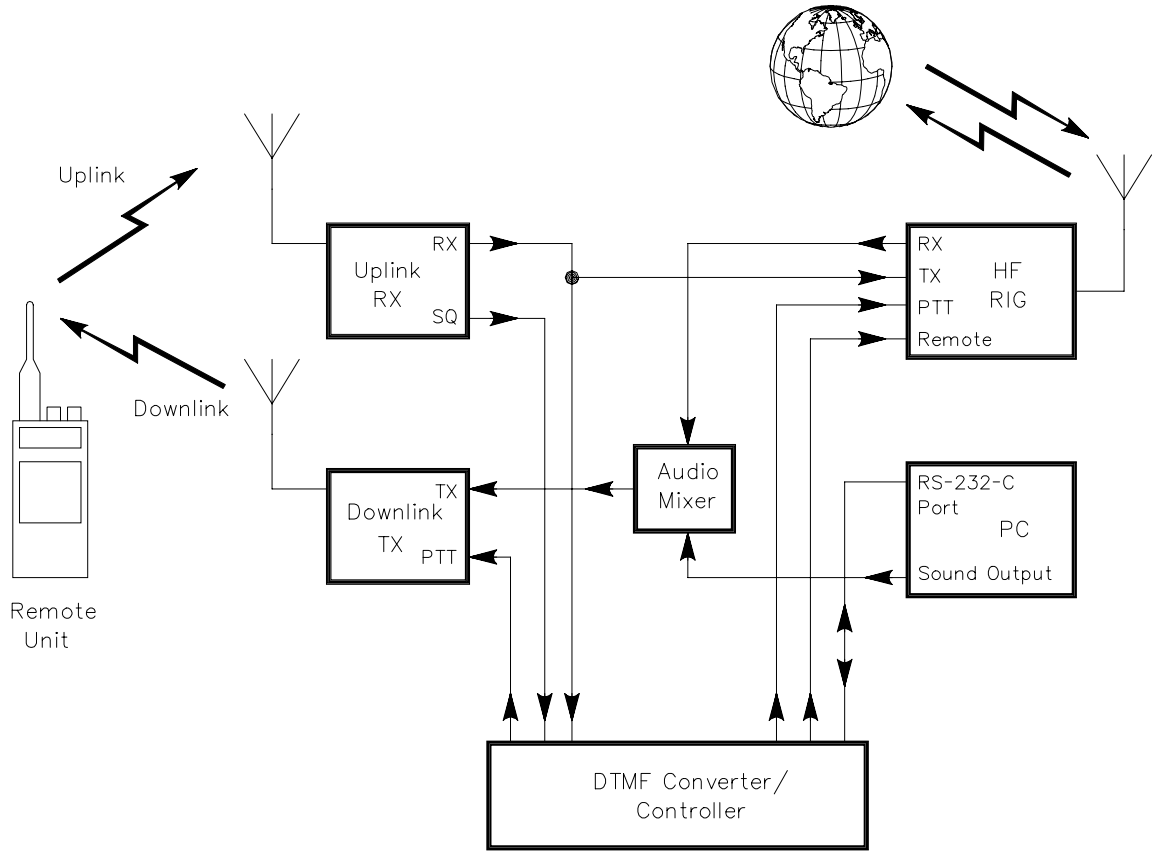

Figure 1—Block diagram of the DTMF converter and HF link controller. All uplinks and downlinks must use frequencies of 222.15 MHz and above.

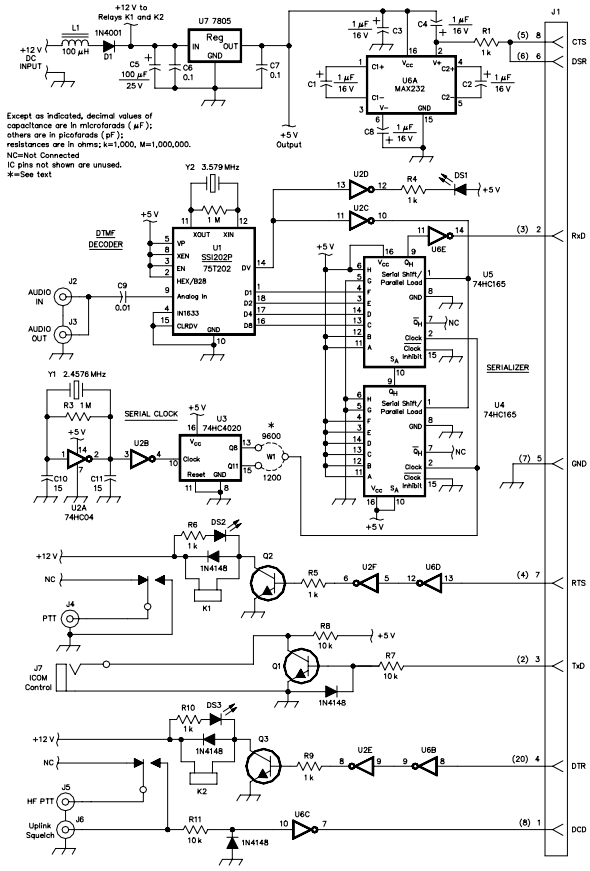

Figure 2—Schematic of the DTMF converter. Equivalent parts can be substituted. Unless otherwise specified, resistors are 1/4-W, 5%-tolerance carbon-composition or film units. PC-board jumper wires are shown as W1. Standard CMOS ICs can be substituted for the 74HC series. Pin numbers at J1 are shown for the DB9 and DB25 connectors; the DB25-pin numbers are in parentheses.RS numbers in parentheses are Radio Shack.

D1—1N4001, 50-V, 1-A power diode

D2-D5—1N4148, 1N914 silicon diode (RS 276-1122)

DS1-DS3—LED

J1—DB9 or DB25 series connector

J2-J6—Phono jacks

J7—Two-conductor phone jack

K1, K2—1-A, SPDT contact relay, 12-V dc coil (RS 275-241)

L1—100-µH RF choke (RS 273-102)

Q1-Q3—2N2222, PN2222, MPS2222, NPN general purpose switch (RS 276-2009)

U1—SSI202P/75T202/CD22202E DTMF decoder (the SSI202P is available from B. G. Micro, Inc, PO Box 280298, Dallas, TX 75228, tel 800-276-2206, 214-271-5546; fax 214-271-2462. The 75T202 and CD22202E are available from JDR Microdevices, 1850 S 10th St, San Jose, CA 95112-4108, tel 800-538-5000, 404-494-1420; fax 800-538-5005.)

U2—74HC04, hex inverter

U3—74HC4020, 14-stage binary ripple counter

U4, U5—74HC165, 8-bit serial or parallel-input/serial-output shift register

U6—MAX232, 5-V RS-232 transceiver (available from Digi-Key Corp, 701 Brooks Ave S, PO Box 677, Thief River Falls, MN 56701-0677, tel 800-344-4539, 218-681-6674, fax 218-681-3880).

U7—7805, 1-A , 5-V regulator (RS 276-1770)

Y1—2.4576 MHz

Y2—3.579 MHz

Table 1

Relationship of the DTMF-Pad Keys, Decoder Data-Line Output and Corresponding ASCII Character†

DTMF Key	*Output from SSI202P D8, D4, D2, D1*	*ASCII Character Sent*
1	0001 (01H)	1 (31H)
2	0010 (02H)	2 (32H)
3	0011 (03H)	3 (33H)
4	0100 (04H)	4 (34H)
5	0101 (05H)	5 (35H)
6	0110 (06H)	6 (36H)
7	0111 (07H)	7 (37H)
8	1000 (08H)	8 (38H)
9	1001 (09H)	9 (39H)
0	1010 (0AH)	: (3AH)
*	1011 (0BH)	; (3BH)
#	1100 (0CH)	< (3CH)
A	1101 (0DH)	= (3DH)
B	1110 (0EH)	> (3EH)
C	1111 (0FH)	? (3FH)
D	0000 (00H)	0 (30H)

†Numbers in parentheses with H suffix are hexadecimal.