The IC-901 Tone Unit Clone 5/09/21

Joseph Haas, KE0FF

I recently obtained a couple of IC-901 radios. They came with a fair compliment of accessories (and a few problems), but one of them was missing the “Tone Unit” which is a very basic CTCSS encoder. This optional board wasn’t really an “option” as it came standard with the radio when new. Thus, there are few of them to be had in the after-marketplace. There are many off-the-shelf encoder options available, but some assembly would be required to allow the serial protocol of the IC-901 to control the CTCSS selection. Why not just build the whole thing from scratch?

I’ve done CTCSS (and tone generation in general) on microcontrollers for decades using DDS (Direct Digital Synthesis) techniques. While I wasn’t instrumental in the development of DDS, I have put it to good use on a number of occasions. DDS allows for precision tone generation with relatively simple hardware and software.

My go-to tiny MCU is the SiLabs C8051F531 (the 530 is also pin/feature compatible). I've used it for several of my projects. One of its many advantages is that it can run on 5V power. I’ve recently been introduced to the ATtiny series from Microchip (which can also run on 5V) and have been rather impressed with its capabilities. However, the clock system of the ATtiny3217 (I have some infrastructure to support this particular MCU) didn’t appear to readily support a crystal source for its timer systems. I was all geared up to go with the new processor, but decided to stick with what I knew would work.

To this end, I copied the code from my HM133 microphone interface project as the seed for this one. There, I used the MCU to generate DTMF tones from the serial data codes produced by the ICOM HM133 (and HM151) microphone. This application would be a simplified subset of that project.

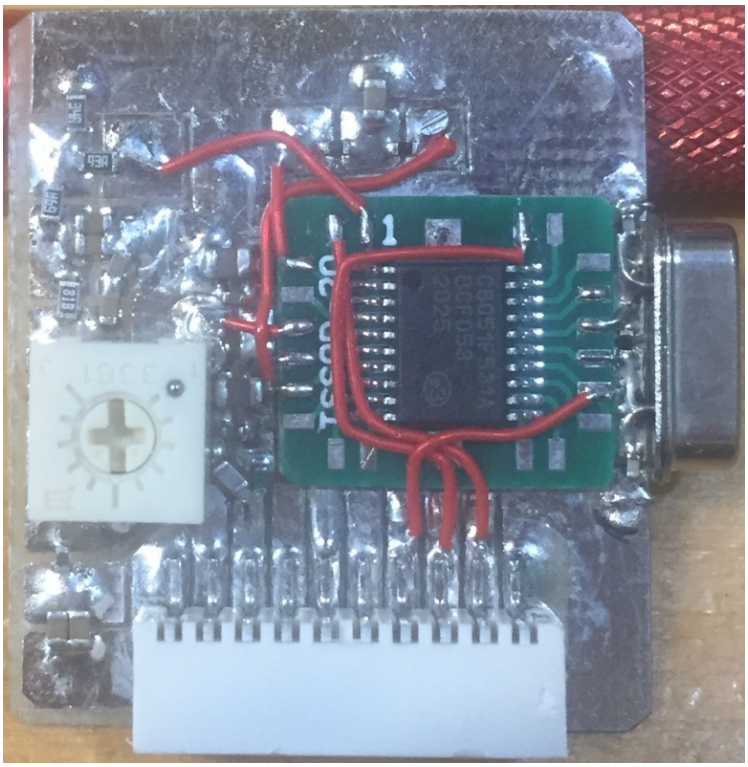
Since the F531 doesn’t have an on-chip DAC, I used one of the PWM features to generate the analog output. The result is a pretty decent pure tone that is adjustable in very small frequency increments (about +/- 0.06Hz in this instance). By over-sampling the DDS clock (the DTMF application runs at about 23KHz, while the CTCSS app will run at 6991 Hz) a 2-pole RC filter does a great job of removing the PWM carrier artifacts.

The DDS code requires processing at each interval of the sample rate. On the F531, the PWM cannot initiate an ISR or easily trap the condition of the start of period. Thus, a timer ISR was needed to accomplish the DDS processing. The timer maintains synch with the PWM simply by using the exact rate that the PWM is set to generate.

After removing the un-needed HM133 code from the seed application, the CTCSS code occupies less than 900 bytes, with about 300 of that allocated to the sine table and CTCSS frequency lookup table. Everything happens in interrupts, so the main running loop is empty except for a line of code to put the MCU into IDLE mode (basically, a wait for interrupt condition).

While the F531 has an SPI peripheral, it was not suitable for this application. Here, there is no chip select, only a strobe signal. We need the last 8 bits sent before the strobe went active and the SPI peripheral would not accomplish that task. Thus, it was necessary to design a “bit-bang” SPI. An edge interrupt grabs the serial clock from the radio and left shifts (msb first is the mantra) the data into a memory register while another edge interrupt grabs the strobe line. The strobe interrupt triggers the data register to be processed (lookup the CTCSS tone in the table and update the tone generator and enable or disable the tone output based on the high-bit of the data register).

The Prototype:

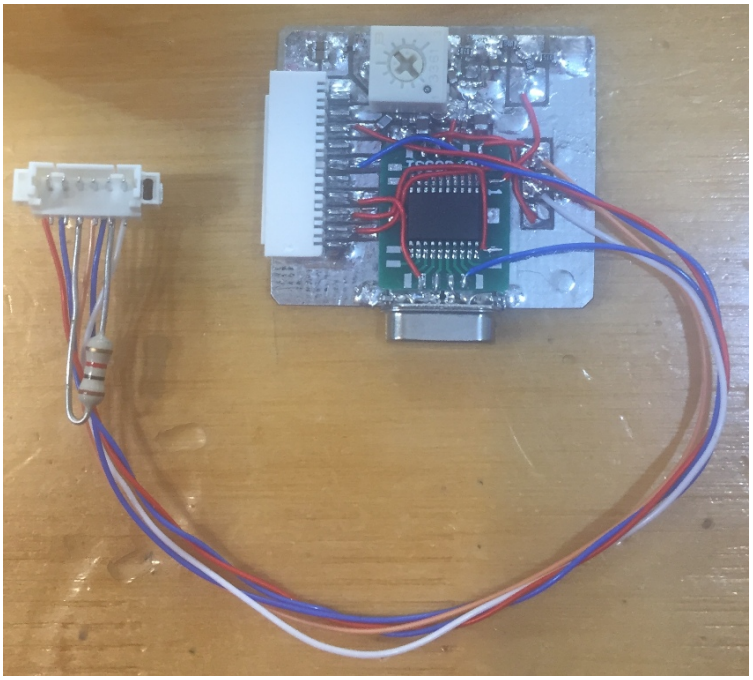


The chicklets (a technical term describing just about any two or three-terminal SMD device) above the pot are the level adjust and filter circuit for the tone signal. The three red-wires attached to the connector are the data, clock, and strobe lines. The colorburst crystal is to the right helps keep the frequencies on in the ppm range as opposed to the “%” range available using the on-chip oscillator. The remaining chicklets are for power supply bypassing and reset.

The back side is bare save for a small patch of copper to which the crystal case is soldered.

The MCU is soldered to a “SCAB” (Supplemental Circuit Addition Board), a term that ends up being appropriate if not savory. Over the years, I have developed several of these for different packages that I can deploy for prototyping. They literally solder to the base PCB (which is bare FR-4, 0.030" thick, 0.5oz Cu) which was etched with an Xacto knife and a particular pair of tweezers that I happen to have which are very good at grabbing copper foil and tugging without letting go (the 0.5oz thickness helps a lot with this... carving 1oz Cu is doable, but with increased difficulty). The breakout pads are small, but just big enough that they can be managed without much effort.

This is an image with the temporary programming cable attached:



The programming cable is removed once the code is loaded and checked out. A bit inconvenient, but the application is simple enough that this should only need to be done once or twice to meet the basic project requirements. Future-feature creep is a dilemma for which there are few successful countermeasures.

The following table summarizes the tone frequency error from the CTCSS specified frequencies and compares truncation math vs. rounding math. The table assumes a perfect crystal source, so the error of the crystal would add to the error shown below. The “radix” is the representation of the division needed to get the integer part of the phase accumulator while “N” is the length of the sine table (see the appnote at https://www.nxp.com/docs/en/application-note/AN1771.pdf for more detail on the care and feeding of a DDS subsystem using small form-factor MCUs).

Fxtal =>	3579545		Fsamp =	6991		
			radix =>	256	"N" =>	256
CTCSS	**raw delF**	**Fgen(trunc)**	**Error (trunc)**	**rounded delF**	**Fgen**	**Error (ppm)**
67.0	628.05	66.994	85.71	628	66.994	85.71
71.9	673.99	71.795	1463.12	674	71.901	-20.59
74.4	697.42	74.355	603.61	697	74.355	603.61
77.0	721.79	76.915	1098.94	722	77.022	-286.50
79.7	747.10	79.689	137.65	747	79.689	137.65
82.5	773.35	82.463	452.40	773	82.463	452.40
85.4	800.53	85.343	667.41	801	85.450	-581.76
88.5	829.59	88.437	715.40	830	88.543	-490.01
91.5	857.72	91.424	833.96	858	91.530	-331.92
94.8	888.65	94.731	730.66	889	94.837	-394.65
97.4	913.02	97.398	23.59	913	97.398	23.59
100.0	937.39	99.958	420.07	937	99.958	420.07
103.5	970.20	103.478	208.78	970	103.478	208.78
107.2	1004.89	107.105	881.82	1005	107.212	-113.32
110.9	1039.57	110.839	548.01	1040	110.946	-413.92
114.8	1076.13	114.786	118.99	1076	114.786	118.99
118.8	1113.62	118.733	560.16	1114	118.840	-337.81
123.0	1152.99	122.894	862.40	1153	123.001	-4.91
127.3	1193.30	127.268	253.31	1193	127.268	253.31
131.8	1235.48	131.748	392.55	1235	131.748	392.55
136.5	1279.54	136.442	423.98	1280	136.549	-357.55
141.3	1324.54	141.243	405.73	1325	141.349	-349.25
146.2	1370.47	146.150	342.73	1370	146.150	342.73
151.4	1419.21	151.377	150.91	1419	151.377	150.91
156.7	1468.90	156.604	610.01	1469	156.711	-70.77
162.2	1520.45	162.152	297.74	1520	162.152	297.74
167.9	1573.88	167.806	561.76	1574	167.912	-73.61
173.8	1629.19	173.780	116.86	1629	173.780	116.86
179.9	1686.37	179.860	220.24	1686	179.860	220.24
186.2	1745.43	186.154	244.76	1745	186.154	244.76
192.8	1807.30	192.769	163.34	1807	192.769	163.34
203.5	1907.60	203.436	312.61	1908	203.543	-211.61
210.7	1975.09	210.691	44.90	1975	210.691	44.90
218.1	2044.46	218.051	222.96	2044	218.051	222.96
225.7	2115.70	225.626	329.80	2116	225.732	-142.86
233.6	2189.75	233.520	343.35	2190	233.626	-113.32
241.8	2266.62	241.734	272.72	2267	241.841	-168.47
250.3	2346.30	250.268	126.42	2346	250.268	126.42

The source code #defines for the CTCSS frequencies result in slightly increased errors due to truncation performed in the calculations (they are done in the compiler with integer math). There is a slight improvement to be had by using the spreadsheet values which were rounded. Still, the overall improvement to be had is only about 0.14% at 71.9Hz. At 77 Hz, the improvement is only about 0.08%. Any other improvements are below that.