Adding Multi-Function Mic Capability to an IC-901

by Joseph Haas, KE0FF
joeh-*at*-ke0ff-*dot*-org
03/08/2024

The IC-901 was a big deal when it was first released. Expandable frequency coverage by way of add-on TX/RX band modules and remote control head functionality (see Photo 1) were mostly unheard of prior to the release of the IC-900/901 series. 10m, 6m, 2m, 220, 440, and 1296 all in one radio was also unheard of. It didn't last though – for many reasons the modular form-factor was relatively short-lived. Still, it was around long enough to develop an almost cult-like following among some amateur radio operators which has served to make the radio a still sought after item almost 25 years after it was discontinued.

Photo 1. The essentials of a stock IC-901 system (minus cables).
From left to right: HM-14 mic, Remote Controller, EX-766 Optical A-Unit,
and an IC-901 Base Unit with a full compliment of RF modules.

One of the possible reasons for the radio's demise as a continuing product line was the limited operator interface. Only 12 memories and no remote control capability and no direct frequency entry options made the user interface "clunky" to be fair. Another shortcoming regards a subtlety of using a remote control head in a mobile environment – the fact that placing the control head where it is easily visible while driving generally makes it uncomfortable to operate the controls. I can truthfully state that the IC-900/901 control heads are no pleasure to use while driving. Modern mobile radios have turned to the multi-function microphone to address this issue. Such a device allows the operator to access various radio controls via buttons on the microphone. This is the story of the steps I have taken to bring the IC-901 into the 21st century (if just barely) with the help of some add-on microcontrollers (MCUs).

A Haphazard Course…

I have worked with the IC-901 since shortly after its introduction. These works have run the gamut from repairs to retrofits to full-up redesigns. This particular leg of my IC-901 journey started a number of years ago when I was working on my centralized mobile radio controller. It was intended to simplify mobile operations by presenting a unified user interface regardless of the radio(s) "behind the curtain". Part of that effort was to reverse engineer the control data stream from an ICOM HM-151 microphone to allow it to act as a hand-held keypad interface to the system. The HM-133 data format is essentially the same as the HM-151 (or vis-à-vis, depending on your point of view) and thus, this led to my HM-133 DTMF adapter design (published in QEX, March-April 2020) which produced a modern, DTMF-output

© Joseph M. Haas, 03/08/2024, all rights reserved

microphone that was readily available. Not long after that effort, I got the idea to use a Bluetooth interface (data only) to allow the HM-133 adapter hardware to effect button presses on one of my IC-901 control heads thus taking advantage of the additional buttons present on the HM-133.

The Bluetooth interface worked well (at least, at first) and has made the radio much easier to operate while mobile, but the implementation has been plagued by Bluetooth connection issues and after looking more closely at the cabling of the IC-901 control head, I realized that a wired remote was possible by re-tasking one of the microphone lines (the Up/Dn signal) with no extra cables required. By placing a small MCU inside the IC-901 control head, it was possible to encode the Up/Dn signal as a data message and then re-produce the up/dn signal at the remote controller processor.

About this time, I also decided to upgrade the IC-901 system RAM from 2K to 8K, effectively quadrupling the memory storage of the radio. This is not a perfect solution, because the IC-901 has no notion of expanded memory. What you end up with is 4 different, and logically separate, radio configurations. Also, you need to mechanize some way to switch between the different "radios" by presenting a 2-bit selection address along with a reset signal to the control head processor. The reset is needed because changing the system RAM address while the control head processor is running will cause undetermined behavior (something missing in the previous implementations I found on the internet). Still, these issues can be overcome and the result far outweighs any imperfections.

All of these efforts converged to present a set of modifications that together bring about a profound change in the user interface of the IC-901. The result: expanded memories, remote microphone control, medium-resolution back-light brightness control, and some other interesting features.

The Laundry List

There is a lot going on with this IC-901 control head modification. First off, I whenever I get hold of a radio that is new to me, I generally replace all of the incandescent backlight bulbs with LEDs (usually, there are several that are burned out anyway). I've modified a couple of IC-901 controllers with LEDs, but for this latest swap-out, I deviated from my previous attempts and got a number of really high-brightness red LEDs. I also added several in places where there were no bulbs previously. The result is dramatic – a VERY bright display (almost TOO bright). This then became the first item on the list: some way to control the backlight brightness.

Then there was the SRAM expansion. I needed some way to control the A[12:11] expansion address lines for the SRAM, processor /RESET, and some way for the user to select the desired address. The address and reset are easy to do given that there is an MCU nearby but the selection interface also needs a path to the operator (I didn't want to try and re-task or multi-task any of the exiting controller buttons). My first attempt was to use a rotary switch and a 7-segment LED display to serve as the operator interface – effective, but clunky. Ultimately, I wanted to do this using HM-133 microphone button presses.

Since the appliance was going to be intercepting button press events that had nothing to do with the original IC-901 buttons, it became necessary to mimic the beeper output used by the IC-901 to provide "beep" feedback of these "non-standard" button presses. This required that I break out my logic analyzer (easier to setup and use than an oscilloscope) to measure the beeper pulse frequency used to activate the piezoelectric transducer present in the IC-901 Remote Unit so that I could accurately reproduce the tone.

One of the more unique features I implemented with the Bluetooth remote was a "PTT-SUB-Mute" mode which would cause the sub-band to mute when the PTT was pressed. Actually, it toggles the sub-mute status when PTT was pressed, and toggles it again when PTT was released. So, it could mute the sub-band when in TX or RX, depending on how the sub-mute was configured initially.

This turned out to be a really nice feature. I could listen to the sub band but have it mute when I transmitted, greatly reducing the distraction offered by traffic on the sub-band. If I am listening to the full-duplex output of a repeater hoping to hear my signal quality, I can mute while in RX removing the sub-band audio when the traffic there is just an annoying duplication of the main repeater output. With the wired remote, I decided to use the IC-901 TX LED status (on or off) to better mechanize this feature. Bottom line: This also needs a path to the operator to allow for configuration (enable or disable).

© Joseph M. Haas, 03/08/2024, all rights reserved

Lots of Buttons

As illustrated in Photo 2, the HM-133 has 25 buttons, plus a PTT switch. The PTT switch is actually the 26th button as there is no PTT signal output provided on the stock HM-133. I prefer to modify the HM-133 so that it provides the PTT signal (since there is an unused pin for the purpose), but with the adapter software as designed, this is not required. It has been somewhat of a challenge to come up with a utilization map that allows all of the button features to be mechanized (see Table 1), but the end result seems to be useful enough to justify the effort required to memorize the button functions (since most of the buttons on the microphone have meanings that are either not applicable or have a less than optimal location.

Photo 2. HM-133 button layout

Modifying the basic HM-133 DTMF Adapter/Bluetooth-Remote to be a wired remote was not very difficult. This part of the software is relatively simple because the adapter is just a messenger, all of the "thinking" happens at the far end of the connection. The Bluetooth remote already had the basic logic required and most of the work involved simply removing the Bluetooth support features. The first wired remote was actually deployed on my IC-900 controller clone. When I turned to the IC-901, I decided that I wanted the hardware and software of the adapter to be unchanged so that the adapter could be used with either radio.

The "button map" featured in Table 1 lists all of the HM-133 buttons, their printed names, and their functions as implemented here. The studious reader will note that the button assignments appear less than optimal considering the names printed on the HM-133 buttons. This is an unfortunate happenstance steeped in the history of my getting to this point in the design. Originally, I based my DTMF adapter on the HM-151 microphone which has a different printed nomenclature from that of the HM-133. The data streams of the two microphones are essentially identical, so they can be used interchangeably, but having first focused on the HM-151, the map of Table 1 evolved from that point. Having "gotten used to" that map, I have stuck with it. I, for one, generally "feel" my way around the HM-133/151 anyway, so the disparity between printed nomenclature and actual function is not often an issue.

	VFO/LOCK: MHz {un-shift}			***MR/CALL***: CALL {PTTsub}		***BAND/OPT***: M/S {LOCK/FACINIT}	
	***UP*:** MIC-UP {DIAL-UP}			***F-1***: V/M {sram bank}		***F-2***: HI/LO {mw}	
	***DN*:** MIC-DN {DIAL-DN}			***DTMF***: n/a		***FUNC***: button-shift {n/a}	
1:	BRT {1}	***2***:	DIAL-UP {2}	***3***:	Squ-UP {3}	***A***:	Vol-UP {SET}
4:	DIM {4}	***5***:	DIAL-DN {5}	***6***:	Squ-DN {6}	***B***:	Vol-DN {TS}
7:	n/a {7}	***8***:	n/a {8}	***9***:	n/a {9}	***C***:	BAND {MODE}
*******:	SUB {n/a}	***0***:	T {0}	***#***:	CHECK {CHECK}	***D***:	SMUTE {SMUTE}

Table 1. HM-133 Button/function map for the IC-901. HM133 labels are in ***BOLD***, {Braces} indicate button-shift functions. "n/a" indicates button function is not used. All caps indicate IC-901 button nomenlcature.

© Joseph M. Haas, 03/08/2024, all rights reserved

Most of the button-press events simply translate to IC-901 button presses. Press a button on the HM-133, and the corresponding button is "as-good-as-pressed" on the IC-901 Remote Controller. However, there are a few buttons that are interpreted by the Multi-Function Mic MCU. For example, if PTT is activated, the 16 keys that correspond to the standard DTMF keypad are tasked with generating DTMF tones (the HM-133 Adapter does not send any press-event messages when it is generating DTMF tones). Others pertain to backlight brightness control, expansion RAM bank selection, and PTTsub mode selection. The following describes these events.

The ***FUNC*** button ("button-shift") is an HM-133 context-switch button and it doesn't result in any press-event messages to the system. Pressing it causes the next key pressed to be sent in its "shifted" form. The ATTINY recognizes the shifted buttons and executes the "shifted function". In addition, a timer is set and any further button press-events will be recognized as their shifted counterpart until either the timer runs out (about 5 sec) or the ***VFO/LOCK*** button is pressed. This allows the shifted buttons to be executed without having to press ***FUNC*** before each one. A 4-beep tone sequence is issued when the ***FUNC***-shift mode is canceled.

*Note: The HM-151 doesn't have a **FUNC** button, but rather a F-2 button which does send a button-code to the remote controller. The ATTINY is designed to interpret this button press as a **FUNC** button and modifies the context for the next button presses in a seamless fashion.*

Backlight brightness is adjusted in 10 steps by the "BRT" (***1***) and "DIM" (***4***) buttons. The PWM resource has 100 steps of resolution, but that many steps are much more than is needed. Holding one of the "BRT/DIM" buttons down for more than about 1 sec will cause the brightness value that was in effect before the press-and-hold to be stored to EEPROM on the ATTINY MCU, to be recalled on the next power-on cycle. "BRT" button-hold programs the bright EEPROM value, and the "DIM" button-hold programs the dim EEPROM value. The light sensor output is read by an MCU analog to digital converter pin and that value is used to determine if the "bright" or "dim" setting is to be used.

"PTTsub" (***FUNC***, ***MR/CALL*** *– which means press the **FUNC** button followed by the **MR/CALL** button*) and "SRAM Bank" (***FUNC***, ***F-1***) are prefix commands that don't immediately do anything when pressed, but rather set the system up for the following parameter (a digit button). The "SRAM Bank button" controls the SRAM bank select and supports numbers from "***0***" to "***3***" (any other number interrogates the setting with the beeper) and the "PTTsub" button supports mode numbers from "***0***" to "***2***" (off, mode 1, and mode 2). Mode 1 is Toggle SMUTE anytime the TX state changes. Mode 2 is toggle SUB-CALL whereby the sub-band CALL feature is toggled anytime the TX mode changes. "9" stores the PTTsub mode setting to MCU EEPROM and is indexed to the bank setting so that each SRAM bank can have an independent PTTsub mode setting. Any other digit interrogates the setting using the beeper.

The beeper interrogate represents the best one can hope for without adding on an external display, although I've done that with my mobile rig. Invalid mode or bank values produce the "error beep" which is an 800 Hz beep. Other values of mode or bank produce 1, 2, or 3 (as appropriate) beeps of a 1600 Hz tone.

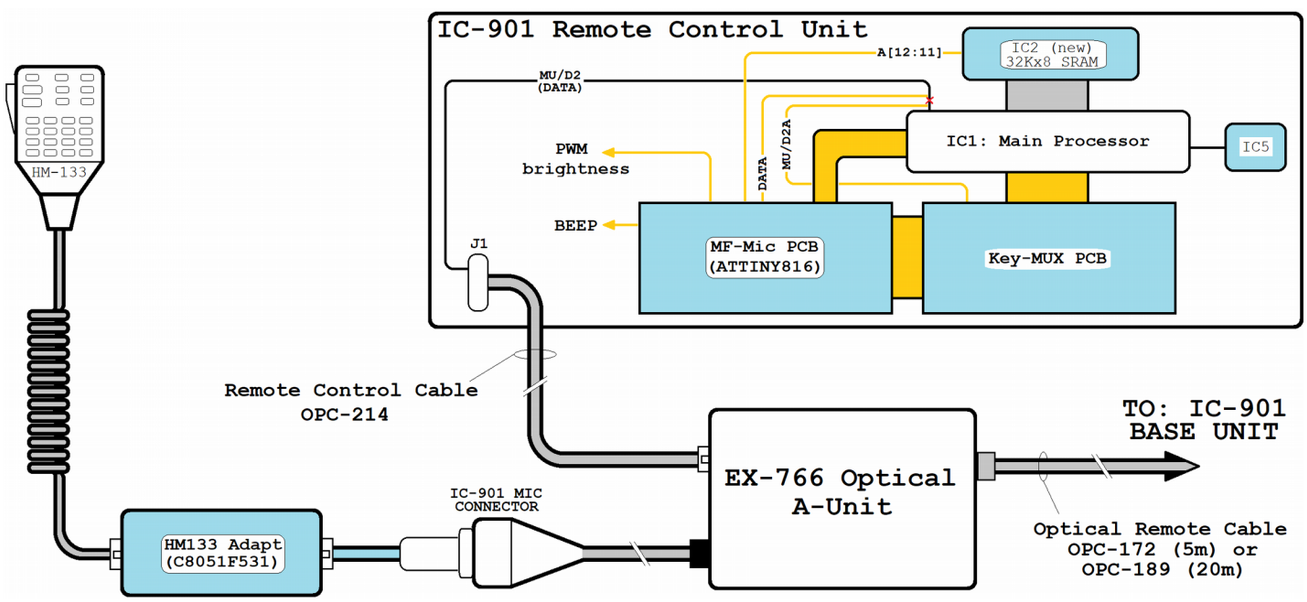

Figure 1. IC-901 Multi-Function Mic modifications when using the EX-766

© Joseph M. Haas, 03/08/2024, all rights reserved

The “LOCK/FACINIT” event (***FUNC, BANK/OPT***) is also a prefix command. Following with a “***9***” will initiate a factory-init sequence by the MCU. This sequence resets the IC-901 four separate times, once for each SRAM bank setting. The MCU also activates the MW button on the IC-901 controller. In this configuration, nothing will happen except for the IC-901 resets. However, if the operator presses and holds the “CHECK” button on the IC-901 remote controller before pressing ***FUNC, BANK/OPT*** (“CHECK” must be held down for the duration of the 4-reset sequence, which lasts about 5 sec), the SRAM will be initialized to the factory-state for all SRAM banks.

Following ***FUNC, BANK/OPT*** with a “***1***” will issue the LOCK press-event which locks the IC-901 controller buttons and dial. LOCK is a toggle button, so simply pressing the button sequence again will unlock the IC-901 buttons. This does not affect the ATTINY MCU operations (such as “BRT/DIM, SRAM Bank, etc...) and it will still respond to these “internal” press-events listed above. It only locks

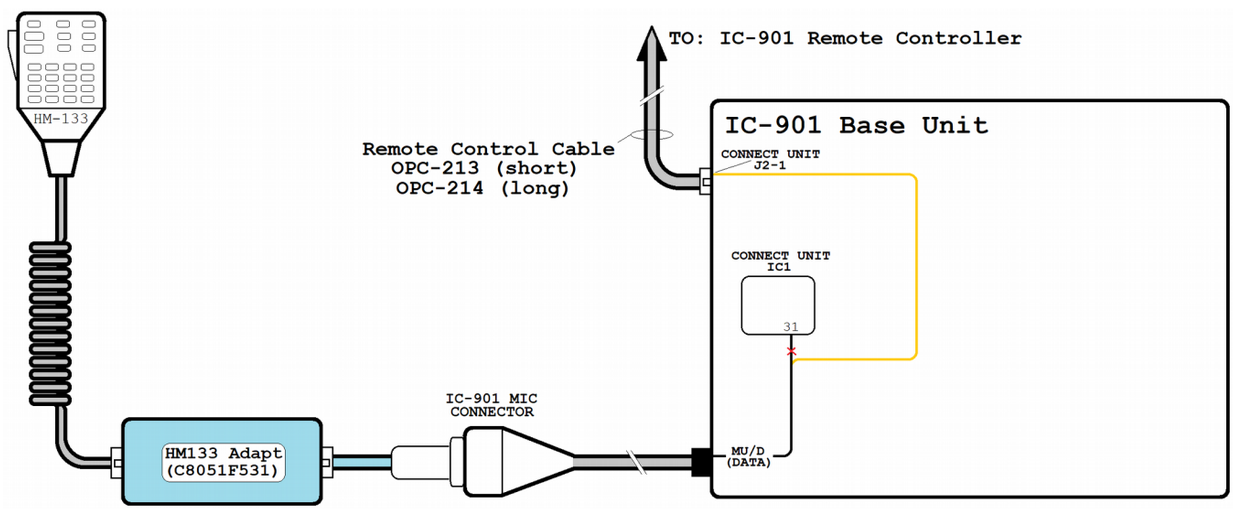

Figure 2. IC-901 Multi-Function Mic modifications for a non-optical installation.

the IC-901 buttons and dial, not the HM-133 buttons. *Note: the LOCK state does not affect the Volume, Squelch, SMUTE, or CHECK buttons.*

A side note about the difference between “MIC-UP/DN” and “DIAL-UP/DN” operations. These may appear to be the same thing, and generally they do accomplish the same end result, but there is a subtle difference. When in the IC-901 is in SET mode, the DIAL is what must be used to adjust parameters. Using the MIC-UP/DN buttons in SET mode will cancel SET mode. So, both of these adjustments are needed to fully support the IC-901 user interface. Also, when the “MHz” mode is active, the dial adjusts the frequency in MHz increments, but the mic buttons still adjust in the current “TS” frequency step setting. This actually becomes a really neat “feature” allowing for quick frequency changes by jumping between the DIAL-U/D and MIC-U/D buttons.

The Picturesque Landscape…

The IC-901 hardware has many options resulting in many possible unique configurations. However, of all the possibilities, the two basic differences that concern this modification are those that either have the EX-766 Optical Remote, or don’t. Figure 1 illustrates the add-ons for an IC-901 with the EX-766 optical cable option. This actually represents the central modification scenario and all of these changes are needed even if the optical interface is not part of the radio configuration. In addition, this modification is scale-able: the SRAM expansion, PWM brightness control, and bulb replacement need not be included in the fray if those features are not desired.

The ATTINY816 MCU located in the IC-901 Control Unit is the heart of the multi-function mic system modification. It receives the button-press event messages from the HM-133 Adapter via the MU/D connection that routes from the microphone connector to the control head processor (via the EX-766) as part of the stock cabling configuration (no changes are required to the EX-766 or any of the stock cabling). The processor pin for MU/D2 (IC1 pin 31) is lifted to interrupt the signal path (which will not be compatible with the original up/dn format, a 3-state voltage level). The IC1 pin is connected to the Key-MUX board which re-establishes the up/dn signal into the main processor under the control of the ATTINY MCU. The ATTINY can detect a standard microphone and replace the original functionality allowing one to hot-swap an original microphone and still use the up/dn buttons.

© Joseph M. Haas, 03/08/2024, all rights reserved

The MU/D pad that was vacated when lifting IC1-31 is then connected to the ATTINY UART RXD pin which completes the control path from the microphone. Additionally, the processor TX-LED and RESET signals are connected to the ATTINY. The MUX board is attached to the top of the processor with metal-safe RTV and the pads that connect to the processor are aligned to be right above the pins they connect to. This makes the installation easy with minimal "blue-wire" jumpers routed across the board. Other connections to the MCU board are the PWM output, the SRAM A[12:11] signals, the BEEP output, and the light sensor input (all optional, more or less).

Figure 2 illustrates the changes needed for an IC-901 without the optical cable option. These changes are in addition to the Remote Control Unit changes highlighted above. Here, all that is needed is to break the connection to the Base Unit processor, and re-route it to the Control Unit connector. Since the IC-901 Remote Control Unit doesn't ever "know" if the EX-766 is connected, it's MU/D2 signal is always interpreted. The Figure 2 modifications are not needed for the optical remote installations, but they do not interfere, so the best-practice is to go ahead and modify the base unit in case the radio might be used without the optical interface.

Nurse, Scalpel!

Note: The specific steps involved in this modification are far too involved for this article. See my web-article for more detail. Also, readers are cautioned… those with a weak stomach when it comes to viewing the innards of an electronic device may find the following to be a bit graphic.

The PCBs for this project were designed to minimize the effort required to install into the IC-901 units. Once the PCBs are placed onto the IC-901, the resulting connections are generally just a few mm in length. An adhesive is recommended to secure the boards, but this is not strictly required. I keep metal-safe RTV on hand, but this is not a cheap item, so most hams may not have this readily available. The first step is to remove some components from the IC-901 remote controller. The removals are organized into four areas: 1) lift pin 31 of IC1; 2) The brightness control circuit (R5, R6, R9, Q6, & Q7); 3) The reset controller, IC5; and 4) The SRAM, IC2, and the LiMn battery (used to retain the SRAM data, this is repalced later, usually with a new battery). If the brightness modification is not required, item (2) is eliminated. If the SRAM change is not desired, items (3) and (4) are eliminated.

Once the removals are complete, new components are added to each section and the new PCBs (pre-assembled) are also added. Photo 3 shows a close-up of the PCBs added around IC1 illustrating the short wire connections. The long connections come last, followed by inspection and pre-power testing. Of course, care and attention to detail are the standing orders when modifying older gear.

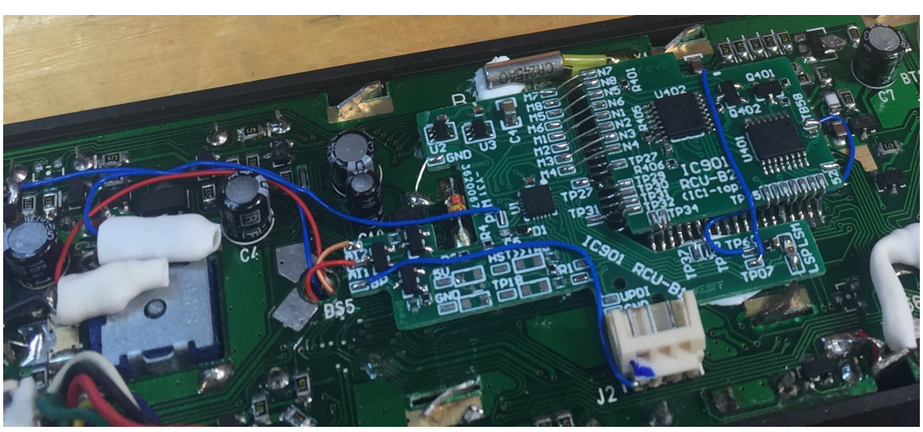

Photo 3. An image of the core of the IC-901 MF-mic modification. The IC-901 Processor (IC1) is located under the "RCU-B2) board.

Once the modification is determined to be free of shorts or other catastrophes, it is time for the final test and reassembly. If the SRAM is replaced, one of the first steps is to perform a factory reset. The easiest way to accomplish this is with ***FUNC***, ***9***, ***F-1*** while holding the CHECK button on the IC-901 remote controller (as mentioned above) .

For the Base Unit PCB, the modifications are less involved. One must dislodge the Logic and Connector Units (located at the very front of the Base Unit), lift IC1-31, remove R17 & C27, and run a jumper-wire from the pad for IC1-31 to J2-1. Testing mostly consists of connecting an HM-14 (or

© Joseph M. Haas, 03/08/2024, all rights reserved

similar) microphone and verifying that the up/dn buttons are working correctly. The final test is to connect the HM-133 Adapter and verify that its button press-events are recognized.

Software

There are a couple of software applications at work here. One is the HM-133 Adapter while the other is the ATTINY816 Control Unit adapter. The pedigree for the HM-133 is pretty straightforward flowing from the original application to the Bluetooth application and then to the wired-remote application. The ATTINY code to receive and decode the HM-133 data stream traces back to an ARM application, so there are a couple of turns in that path. My IC-900 Remote Controller "clone" project implemented all of the multi-function microphone features described herein. When I went to code the ATTINY portion of the IC-901 MFmic, I ended up copying much of it from the IC-900 ARM code. With all this code copying, there was a fair amount of debug to get the pieces all working together.

There are several interrupt-driven processes which handle much of the low-level tasks such as capturing serial button-even messages, application timers, and the beep functionality. State-machine processes drive the higher level routines to process completed button-event messages and other tasks. Most of the state-machine steps are regulated by application timers and other boundary variables. This structure serves to keep the code reasonably modular (which hopefully makes it easier to maintain).

This application produces a rudimentary user interface using the IC-901 beeper. Unfortunately, there is no way to independently affect the LCD screen of the IC-901. This leaves the beeper as the only other available resource. Of course, other display solutions could be added to allow non-standard status information to be displayed. I've done this on one of my Remote Control Unit mods by adding a 7-segment display to show the current bank selection. However, this requires some manner of mechanical interface to support the non-standard display. Personal preference is most of what might be involved in such a decision.

Conclusion

Most who are unfamiliar with the IC-901 might react to this task with a quizzical look. "WHY???" becomes the common question. I enjoy working with older gear and learning how it works and how to improve it. That is why enough for me. However, these changes have given me a radio that I would be challenged to find from a commercial source. The features are useful (to me, at least) and mesh whith how I use the radio in my mobile install. Being able to exercise all of the radio features from the microphone greatly improves the mobile experience. Making the modified IC-901 an easy choice for my mobile installation whereas using the original IC-901 would be cumbersome and much less enjoyable.

There are a couple of on-line locations which house information about this project. One is my web page, http://www.ke0ff.org (this is a reflector address to my main web page – once at the main page, navigate to the "IC901 MFmic" link form there) and the other is my github repository at https://github.com/ke0ff/mfmic. The web page has a writeup that covers the IC-901 changes in greater detail while the github repo has the detailed hardware and software design data (schematics, PCB layouts, and source code) for the various project sub-sections.

Because the design is open source, it may be customized by the experimenter, just adhere to the applicable license indicated within the github repo. In addition, this system could be deployed to modify other radios that utilize a scanning-matrix button input scheme allowing it to reach a broader radio audience.

The modification is in some ways rather extreme, but I've done my best to minimize the "hacking" required to implement the changes described. For those willing to take the plunge, it might be a good excuse to dig that old IC-901 out of the closet and see about getting it back on the air with a new remote user interface.

© Joseph M. Haas, 03/08/2024, all rights reserved

Bibliography/Reference:

HM-133 Microphone DTMF Adapter Project, QEX (published by the ARRL), MAR-APR 2020

ICOM Service Manual, IC-901A/E, DOC# A5078H-S, 1990, Icom, Inc., 6-9-16, Kamihigashi, Hirano-ku, Osaka 547, Japan

Datasheet, ATTINY816, DOC# DS40002288A, via web at: https://www.microchip.com/

Datasheet, C8051F52x/F52xA/F53x/F53xA, via web at: https://www.silabs.com/

The following are also navigable via my reflector domain address, http://www.ke0ff.org/ :

SiLabs/Tiva/ATtiny Programming Guide, Rev 3.2 (or later), 6/21/22, by Joseph Haas, via web at: https://ke0ff.github.io/Orion/silabspgm.pdf

IC-901 Multi-Function Mic (and other mods), 10/17/22, by Joseph Haas, via web at: https://ke0ff.github.io/mfmic/index.html

IC-901 Multi-Function Mic Design repo, 03/08/24, by Joseph Haas: https://github.com/ke0ff/mfmic

HM-133 Microphone Modifications, 02/27/24. A compilation of mods for the HM-133 created by the author and others: https://ke0ff.github.io/hm133a/hm133mods.pdf

Revision History

03/08/2024: Text updates (various pages). Updated email address.

02/08/2023: Typo corrections (various pages).

10/17/2021: Initial release.

© Joseph M. Haas, 03/08/2024, all rights reserved