QEX

$5

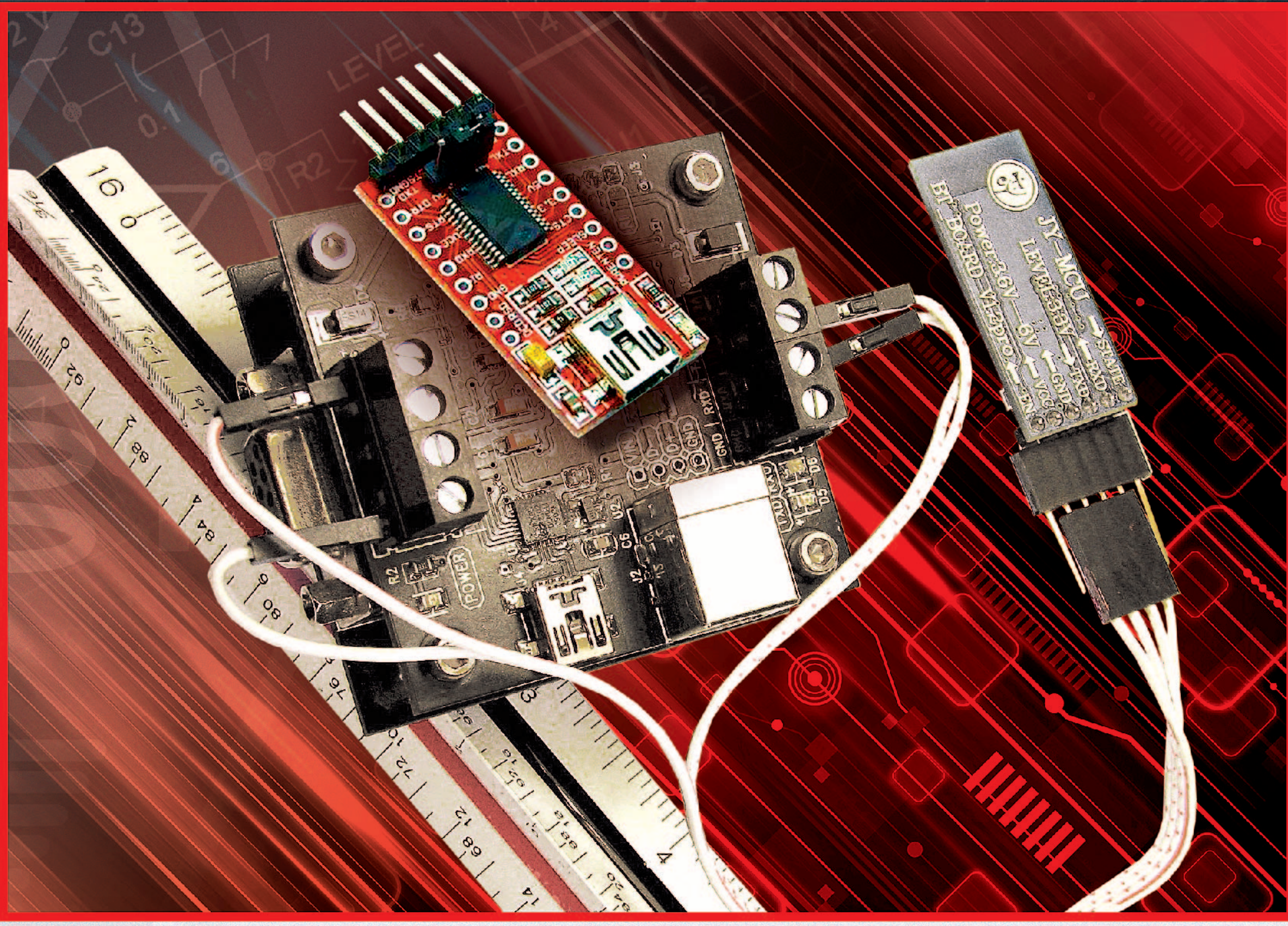
September/October 2014

www.arrl.org

ARRL 100 YEARS

A Forum for Communications Experimenters

Issue No. 286



VA7TA wraps up his "Android Wireless Project Control" article by describing a Bluetooth Wireless Link, which can be used to connect your Android device to the hardware being controlled.

ARRL The national association for AMATEUR RADIO™
225 Main Street
Newington, CT USA 06111-1494

QEX (ISSN: 0886-8093) is published bimonthly in January, March, May, July, September, and November by the American Radio Relay League, 225 Main Street, Newington, CT 06111-1494. Periodicals postage paid at Hartford, CT and at additional mailing offices.

POSTMASTER: Send address changes to: QEX, 225 Main St, Newington, CT 06111-1494
Issue No 286

Harold Kramer, WJ1B
Publisher

Larry Wolfgang, WR1B
Editor

Lori Weinberg, KB1EIB
Assistant Editor

Zack Lau, W1VT
Ray Mack, W5IFS
Contributing Editors

Production Department

Steve Ford, WB8IMY
Publications Manager

Michelle Bloom, WB1ENT
Production Supervisor

Sue Fagan, KB1OKW
Graphic Design Supervisor

David Pingree, N1NAS
Senior Technical Illustrator

Brian Washing
Technical Illustrator

Advertising Information Contact:

Janet L. Rocco, W1JLR
Business Services
860-594-0203 – Direct
800-243-7768 – ARRL
860-594-4285 – Fax

Circulation Department

Cathy Stepina, *QEX Circulation*

Offices

225 Main St, Newington, CT 06111-1494 USA
Telephone: 860-594-0200
Fax: 860-594-0259 (24 hour direct line)
e-mail: **qex@arrl.org**

Subscription rate for 6 issues:

In the US: ARRL Member $24, nonmember $36;

US by First Class Mail:
ARRL member $37, nonmember $49;

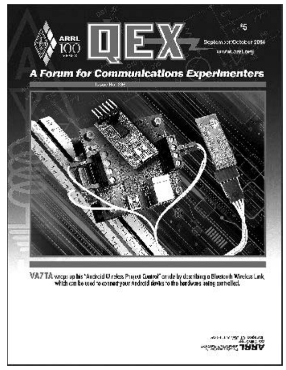
International and Canada by Airmail: ARRL member $31, nonmember $43;

Members are asked to include their membership control number or a label from their QST when applying.

In order to ensure prompt delivery, we ask that you periodically check the address information on your mailing label. If you find any inaccuracies, please contact the Circulation Department immediately. Thank you for your assistance.

About the Cover

Thomas Alldread, VA7TA, describes a Bluetooth Wireless Link in Part 3 of his "Android Wireless Control Project." Tom uses the Bluetooth link to connect his Android tablet to his NimbleSig III RF Signal Generator. His tablet can then control the signal generator to perform tests and collect measurements.



In This Issue

Features

Index of Advertisers

Larry Wolfgang, WR1B

Empirical Outlook

ARRL National Convention and More

I have only been to 2 or 3 previous National Conventions in my many years as an Amateur Radio operator. With this year's Centennial National Convention being held in Hartford, Connecticut, I wanted to take advantage of the opportunity. Many ARRL Staff members had spent countless hours over the course of the previous year or longer in the planning and organizing of this year's special convention. I did not play a role in any of the planning, but I would like to start off with congratulations to all of the staff and volunteers who made it happen! I saw first hand some of the work and dedication involved, and recognize that all of this was in addition to keeping up with their normal work responsibilities. My hat is off to every one of them.

I signed up for Contest University during the Thursday all-day seminar tracks. I had thought about taking this course a few times previously, and this seemed like a perfect opportunity. I am not a real competitive contester, nor to I dream of becoming one. I do enjoy spending some time on the air during a contest, and challenging myself to do better from one year to the next. I also believe that if I can learn some techniques to make my station and my operating practices more effective, this will be a benefit in other aspects of my enjoyment of the hobby.

CTU was a very interesting program, and I enjoyed hearing from various experts in particular aspects of station design and operation. I gained a few tips for improvements I want to make to my station, as well as some operating tips that may help improve my scores in some of my favorite contests. If you have even a passing interest in this aspect of Amateur Radio, I highly recommend signing up for the course when you have an opportunity. The Dayton Hamvention seems to be the most regular place to find it.

I was not able to attend the Convention on Friday. Some of us had to stay at the office (and appear to be working) for those who were visiting Headquarters during the day. I had lots to do for this issue of *QEX*, so it was best that I spent the day working, anyway! My wife Jean, WB3IOS, and I attended the Convention Banquet on Friday evening, and we had a wonderful time. We met and talked with quite a few other attendees, enjoyed a wonderful meal, and heard an excellent keynote speech from Craig Fugate, KK4INZ. Later, we also attended the Royal Order of the Wouff Hong Ceremony. This was my second time to go through the ceremony, but Jean's first. It is always fun. If you have not gone through a Wouff Hong Ceremony, try to attend a Convention where it is offered, and do it.

On Saturday I got to be a "tourist" and check out the exhibit hall and take in a couple of the forums. It was great fun to browse the various booths, drool over a few radios, and meet up with old friends and new. The setting at the Connecticut Convention Center was one of the best that I've been to. I have gone to several other programs at the Connecticut Convention Center, and it is always nice, but it really seemed to suit the needs of our program very well. The forum rooms upstairs were comfortable, although when I arrived at the room for one of the programs I wanted to attend, I found only an empty space. I guess all those people heading the other way should have been a clue, but I just guessed they were leaving a previous forum. Because of the large number of people who showed up, the forum was moved to a larger room. I found the new location, and ARRL Rocky Mountain Division Director Brian Mileshosky's presentation on the Raspberry Pi was quite informative.

A highlight for me was going to hear Joe Taylor, K1JT, talk about some of his work on JT65 and other software. It was also quite an honor to greet him and shake his hand after his talk. There are not that many opportunities to meet and greet an Nobel Prize winning Physicist!

Introducing a New Column with This Issue

When you look at the Table of Contents for this issue you will notice an article by Scotty Cowling, WA2DFI. Actually, this is more than *just* an article. This is the first in what we hope will become a regular column in *QEX*. As the title indicates, Hands On SDR is intended to guide you through the process of trying out some hardware and software that will let you experience the wonders of software defined radio. I know that many of you are old hands at this stuff, but I believe others have been hearing about it, and wanting to experience it, but just haven't figured out where to start. Scotty will take us through that process. In this issue he introduces three different bits of hardware and one piece of software that can control any (or all) of those three. There are several price point options that I hope you will find attractive. Even if you don't have the hardware right now, Scotty will show you some of the things you can do with it over the next several columns. We probably won't run Hands On SDR in every issue, especially if Ray Mack, W5IFS is able to get back to writing about some of the theoretical aspects of SDR in his SDR: Simplified column.

I hope you find this new column fun and interesting. Let Scotty and me know what you think, and ask him any questions you might have.

Thomas M. Alldread, VA7TA
7056 Railway Ave, Courtenay, BC V9J 1N4, Canada; **VA7TA@telus.net**

Android Wireless Project Control Part 3 — Android Bluetooth Wireless Link

The final installment of this series shows how to provide a wireless connection between the tablet and the NibleSig III sweep generator.

This final part of this series will describe the implementation of an economical Bluetooth serial universal asynchronous receiver/transmitter (UART) data transceiver module for the wireless control of Amateur Radio projects. The information here is intended to provide a guideline for those that wish to adopt one of these Bluetooth data transceivers to form the basis for a wireless control link. As described earlier, a Bluetooth connection paves the road for the use of a mobile device such as a Bluetooth equipped tablet that can be used to provide a user-friendly graphics user interface (GUI) wireless controller.

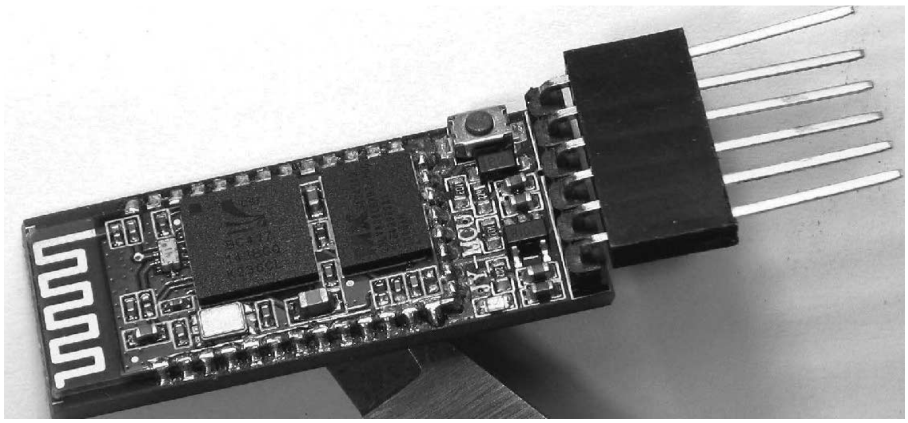


Figure 1 — This is a component-side view of the HC-05 Bluetooth transceiver.

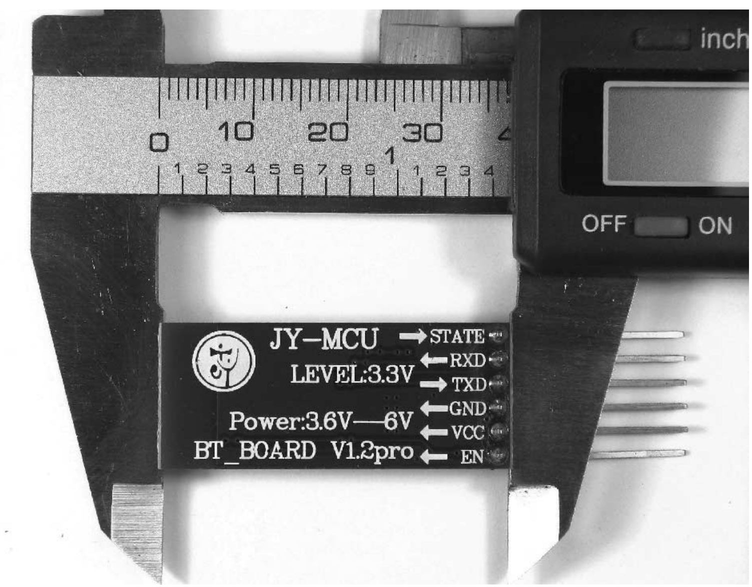


Figure 2 — Here is a printed-side view of the HC-05 Bluetooth transceiver. The calipers provide a size reference.

Bluetooth Transceiver Modules

As mentioned in Parts 1 and 2 of this series (Mar/Apr and May/Jun 2014 *QEX*) the Bluetooth (BT) link makes use of a Bluetooth transceiver module assembly, which is shown in Figures 1 and 2. The overall assembly is quite small, roughly the size of two small postage stamps stacked end-to-end (1.5 in × 0.5 in). The transceiver module consists of two circuit boards; a tiny postage stamp sized motherboard (sometimes referred to in eBay item listings as a back board) and a wireless transceiver daughter board. The wireless transceiver daughter board has 34 solder contacts around three sides of the perimeter, which attach/connect it to the motherboard. The motherboard also provides mounting space for a V_{CC} polarity reversal protection diode, a 3.3 V regulator, male header pins for external connections, a push button switch for implementing configuration setup mode, an LED status indicator and a few passive components. The fourth side of the daughter board provides real estate for a circuit-board-etched 2.4 GHz antenna.

The push button switch is sometimes referred to as the "Clear "switch, but within this article it will be called the "PIN34HI" switch because when operated it places 3.3 V on pin 34 of the Bluetooth daughter board. For those that happen to have a motherboard without the PIN34HI switch, placing 3.3 V on the motherboard "Key" pin usually provides the same function.

The transceiver requires 5 V power, which is reduced to 3.3 V by the regulator on the motherboard. It is important to remember that internally this device runs from a 3.3 V power bus, thus it uses 3.3 V for both the logic levels and the receive and transmit UART input/output RXD/TXD unipolar serial data streams.

There are several models of look-alike Bluetooth transceivers within this family of modules. The most commonly available models are the HC-05 and HC-06. The HC-05 uses much more advanced, more recent firmware and can be switched to either master or slave mode. In contrast the HC-06 is factory programmed as either a slave or master device and in general is not nearly as versatile as the HC-05. Note that the HC-06 with the firmware named "Linvor1.5" behaves differently than the HC-05 described here. It uses a different command protocol and provides different status indications. This article will focus on the use of the HC-05 model.[1, 2] Should there be a need, detailed HC-06 information can be found with a Google Internet search.

The transceivers are available from various online vendors including many on eBay. The "Bluetooth Transceiver Procurement Criteria" Sidebar provides a list of factors that should be considered when selecting the particular module to be purchased.

Interface Hardware for USB Virtual Port Parameter Setup

Various hardware interface approaches for configuring the HC-05 transceiver can be found on the Internet. One approach that I tried was to program an Arduino AVR microcontroller board, which I then used to program the Bluetooth module indirectly. After experiencing success with that method it occurred to me that an easier approach would be to simply use a USB to TTL serial data adapter module interfaced directly to a computer USB port, which eliminates the intermediate microcontroller hardware. Inexpensive adapters of this type that use the well-established FTDI USB interface chip, and that support both 3.3 V and 5 V logic levels were, at the time of this writing, available for purchase online for about $5.00. When connected, they provide a virtual serial port on a computer, which can be used by a terminal program (such as *Windows* HyperTerminal) to communicate directly with the Bluetooth transceiver.

Besides the cost savings, another advantage of using an adapter on a computer for this task is that convenient support is built into the operating systems for changing the serial port parameters (for example, data rate, word length, parity and stop bits). This makes it relatively easy to set up and test a Bluetooth module for different data rates and interface options.

Although there is little doubt that the use of the basic USB to TTL converter described above and as shown in Figure 3 would work for configuring the Bluetooth transceivers, my personal favorite USB adapter for a task like this is the dual function, double-decker design by Sure Electronics that is shown in Figure 4.[3] The top deck converts USB

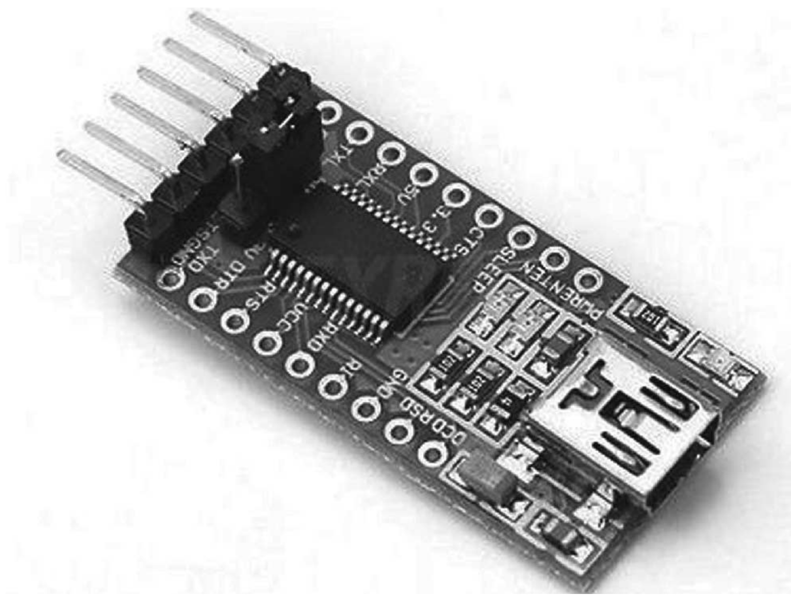


Figure 3 — One example of a typical low cost FTDI USB to TTL converter module.

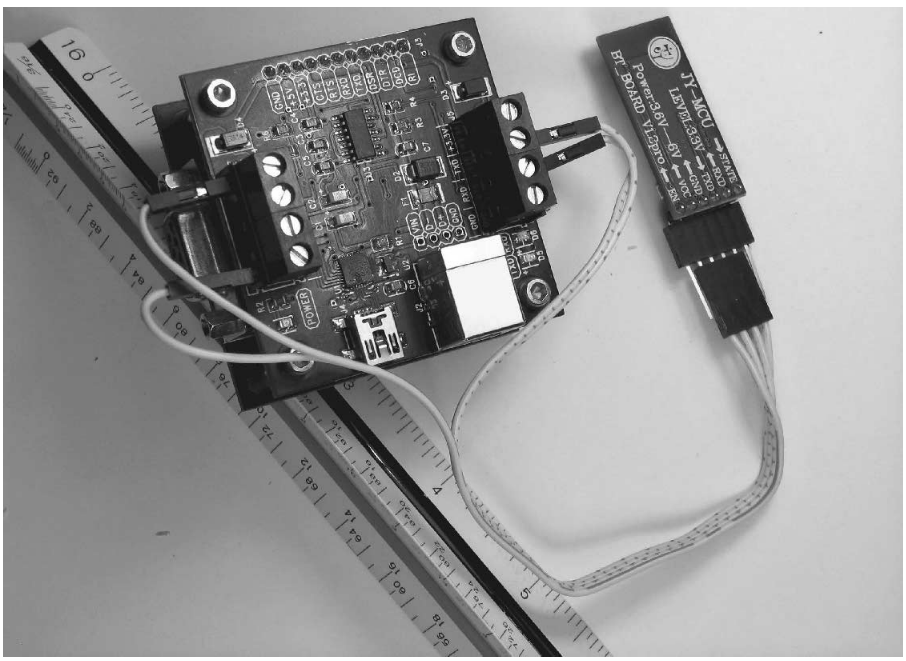


Figure 4 — Here is my Bluetooth configuration setup, using the Sure Electronics double-deck USB adapter.

[1]Notes appear on page 12.

to TTL serial data and the bottom deck converts the TTL unipolar data to RS232 bipolar levels, which appear on the two DE-9 connectors. [Note that these connectors are often incorrectly designated as DB-9 connectors. See **en.wikipedia.org/wiki/D-subminiature** for more information about the naming standards. — *Ed*.]

The two decks of this adapter can be separated to form two separate adapters, one for each of the two conversions. The bottom deck on which the female and male DE-9 connectors are mounted for the RS232 interface is not needed here for Bluetooth module communications, but could be used for an alternate Bluetooth application as will be described later.

On the upper deck both 5 V and 3.3 V RXD/TXD data streams appear, along with both 5 V and 3.3 V regulated power, on screw terminal blocks. This makes Bluetooth transceiver module interconnection easy because both the 5 V power and 3.3 V data streams are physically easy to access. As shown in Figure 4, the Bluetooth module V_{CC} and ground pins are connected to the 5 V terminal block on the right hand side of the upper deck and the TXD/RXD data leads are connected to the 3.3 V block on the left hand side. The computer USB port supplies the 5 V power for both the adapter and Bluetooth module assemblies.

The installation of wire-wrap pins into the +5 V and GND positions of the 5 V terminal block, and into the RXD and TXD positions of the 3.3 V terminal block makes connection of the USB/TTL adapter to the Bluetooth transceiver an easy task. Singleton header socket connectors can then be easily slid onto the corresponding wire-wrap pins. Once connected, the computer operating system should establish a virtual Com port communications link to the adapter. If the *Windows* operating system requires a driver for the CP2102 chip, which is used in this particular adapter, the driver install files can be downloaded from the Silicon Labs website.[4]

Upon connecting to a USB port, the initial power up of the Bluetooth transceiver should be confirmed by the status LED quickly flashing about two flashes per second. This quick flash is indication that the HC-05 transceiver is ready and waiting for a connection request. Various status LED indicator states are listed and described in Table 1.

At this point the *Com* port assignment for the adapter needs to be determined. With the *Windows* XP operating system, we can determine this from the Control_Panel>System>Device Manager > Ports(COM&LPT) list. The Sure Electronics adapter that I use is identified within the ports list as "Silicon Labs CP210x USB to UART Bridge." In my case it was assigned to Com-3.

uCon Terminal Emulator Software

I used a *Windows* computer terminal emulator freeware application named *uCon*. The GUI interface is shown in Figure 5. This terminal emulator offers customizable hot key style mouse click buttons. You can download a self-extracting installation file for *uCon* directly from the *uCon* website.[5] The *uCon* program is a good choice for setting up the Bluetooth transceiver options because the 24 hot keys can be pre-programmed with the AT commands that are needed to set up an HC-05 module.

I have pre-programmed and saved the commands that I consider relative into a *uCon* setup file named "HC05_Bluetooth.ct". For details regarding the numerous other commands refer to the HC-05 AT Commands documentation. (See Note 1.) This customized-for-the-job *uCon* configuration file is available for download on the ARRL website.[6]

Once saved, this HC-05 settings file can be loaded via the *File>Open* menu item when *uCon* is first started. If your computer assigned a different port than Com3 for the USB/TTL converter virtual port it will be necessary to change the *uCon* port number to match your computer configuration. This setting can be found via the Config>Backend>COM port menu item. After the installation is complete it will be necessary to copy the custom "HC05_Bluetooth.ct" setup file into the *uCon* configuration folder. The path for this folder is usually *C:\Program Files\ucon\config*. When the "HC05_Bluetooth.ct" setup file is opened, the *uCon* GUI should show the various HC-05 AT commands pre-loaded

Table 1
HC-05 Bluetooth Transceiver Status LED Indication Definitions

HC-05 Status LED Indication	*Transceiver State*
READY	
Quick Flash – About 2 Flashes per Second	Default Power Up – Pin 34 at logic low level: The transceiver is ready and is waiting for a connection request. It will not accept configuration commands. Daughter board pin 34 toggled high after power up (switch depressed momentarily): The transceiver switched into AT command mode awaiting configuration commands to be sent using the current UART settings. The transceiver still monitors for a connection request and if received it will automatically switch back into ready mode and connect. (See Note 1)
PAIRED	
Slow Repeating Quick Single Blink	Displays this LED indication when first paired with a master device.
CONNECTED	
Slow Repeating Quick Double Blink	Connected and in data transmission mode.
38.4 K BAUD AT COMMAND MODE	
Slow Flash, about 2 seconds on and 2 seconds off	This AT command mode is entered by holding pin 34 at logic high during power up (See Note 2) and uses fixed 38,400 baud, 8N1 UART settings for communications. Connection requests are ignored during this mode. This mode provides a known port setting and speed to re-establish communications should the port settings for regular data transmission mode become unknown.

Note 1: Some AT commands will not respond unless pin 34 is held at logic high. AT+NAME and AT+CLASS are two that are known to require pin 34 to be held high.
Note 2: To enter this mode the PIN34HI switch can be held operated during power up, which places a logic high on pin 34. Once the slow flash starts the switch can then be released.

into the hot key buttons as shown along the top of the screen shot shown in Figure 5.

Before connecting the Bluetooth module it is wise to confirm that the computer port, USB/TTL adapter and the *uCon* app are all functional. This can be done with a simple loop back test, which can be set up by temporarily connecting the RXD/TXD TTL data terminals together on the USB/TTL converter. Once the RXD/TXD terminals are looped, any typed characters should be double echoed back to the *uCon* screen. Note that with the "HC05_Bluetooth.ct" setup file parameters loaded, a double echo of the characters is normal because the HC-05 custom setup sets a *uCon* option to internally echo transmitted characters to the screen. This software loop-back option is set purposely in order to provide a view of the AT commands sent to the module during configuration as shown in Figure 5. It is important to remember that with the "HC05_Bluetooth.ct" settings in force, a single character echo would be indicative of a loop-back failure from the USB/TTL adapter — a double character echo is required before proceeding.

Bluetooth Module Configuration Settings

Once the configuration tools are tested okay, the loop-back jumper from the RXD/TXD adapter terminals must be removed to allow connection of the data streams to the Bluetooth module. Ensure 5 V is used for the power and 3.3 V data is used for the TXD/RXD leads. Ensure the converter TXD output is connected to the Bluetooth module RXD terminal and vice versa.

The HC-05 firmware provides two command modes for configuring the Bluetooth module. The first mode, which is usually used for initial setup, has fixed UART settings that cannot be altered. The second mode, where the UART settings can vary, is relatively easy to access for making most configuration changes after the module has been placed into service. The first mode will be referred to here as the *AT Command Fixed UART Mode* and the second as the *AT Command Variable UART Mode.*

The UART settings for the *AT Command Fixed UART Mode* are fixed at 38,400 baud, 8N1. To enter this mode the PIN34HI switch must be steadily pressed for a few seconds during Bluetooth module power up. Confirmation of entry into this mode is provided by the indicator LED cycling with a slow two-second-on /two-second-off sequence. While running within this mode the Bluetooth module is offline and thus not available for connection. Advantages of this mode are that for performing the initial setup or for doing extensive setup changes there is no chance for interruption by an outside connection request, the UART settings can be altered without interrupting the computer *uCon* terminal emulator communications to the Bluetooth module and should the UART settings for the current application become unknown one has a restoration means to re-establish connection to the Bluetooth module with known proper settings. Possible disadvantages for the *AT Command Fixed UART Mode* are:

• The power to the Bluetooth module must be interrupted for both *AT Command Fixed UART Mode* entry and exit.

• The computer terminal emulator port settings, which may be set to match the application at hand, may need to be altered to match the 38,400, 8N1 settings of the *AT Command Fixed UART Mode.*

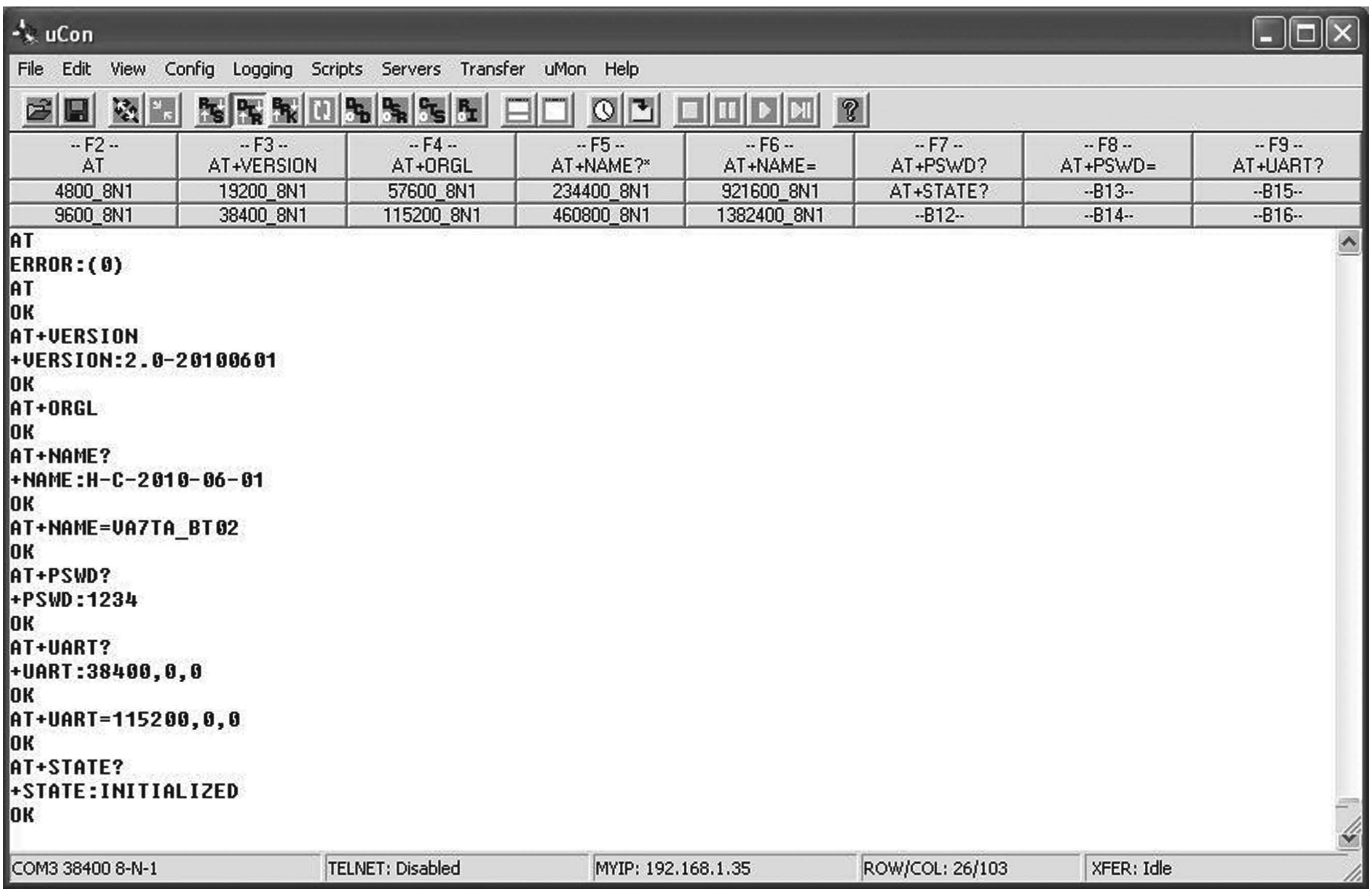


Figure 5 — This screen shot shows the *uCon* terminal emulator. You can see the Hot Keys across the top of the window, which have been customized for the HC-05 configuration.

The UART settings for the *AT Command Variable UART Mode* (which by factory default are similarly set to 38,400, 8N1) will vary according to the settings used for the target application. This can be a convenience in the event that the computer *uCon* port settings have been previously set to match the target application. This mode is relatively simple to enter after normal power up, while the Bluetooth module is idling in *Ready Mode* awaiting connection. While in this state, with the LED indicator flashing at a quick half-second rate, the *AT Command Variable UART Mode* can be entered with a simple momentary press of the PIN34HI switch. In this case the LED flash rate indication does not change because the Bluetooth module remains available for connection.

This mode is particularly convenient for making any in-service configuration changes that don't affect the UART settings. Possible disadvantages of the *AT Command Variable UART Mode* mode are:

• The UART settings cannot be changed without loosing communications to the computer terminal emulation console.

• The configuration setup change session would be interrupted if a legitimate outside connection request is received during the session. In this event the Bluetooth module would automatically switch to *Connected Mode*, at which time the LED indicator would commence a slow, double blink cycle, confirming the connection. Once in this state it is not possible to return to the *AT Command Variable UART Mode* until the connection is dropped and the Bluetooth module returns to the *Ready Mode*. At that time another press of the PIN34HI switch would be needed to re-initiate a return.

To initially configure an HC-05 module, first enter the *AT Command Fixed UART Mode*. Once entry is made evident by the indicator LED, establishment of communications to the Bluetooth module must be confirmed before proceeding. Confirm communications by mouse clicking the F2 button on the *uCon* GUI screen. This will send the "AT" command, and as shown in Figure 5, the module should respond with an "OK" confirmation. If, alternately, it responds with an error message, possibly due to random characters that ended up in the buffers during startup, repeat the F2 "AT" command. Keep in mind that a need to repeat commands may reoccur whenever the module is re-started or after the PIN34HI switch is operated.

Once communications has been confirmed "OK" the *uCon* GUI shortcut buttons may be used to complete the configuration. The commands have been assigned to the shortcuts in a typical sequential order for doing the configuration. Refer to Table 2 for a description of the individual commands.

Figure 5 was captured from an actual module setup sequence. The commands with the responses that occurred during to the following setup sequence are listed in Figure 5:

• F2 sends "AT" to confirm Bluetooth module communications — confirmed with the "OK" acknowledgement.

• F3 reports the firmware version "+VERSION: 2.0-20100601."

• F4 resets the module to factory defaults, which includes setting it to slave mode.

• F5 will report the current name of the module only if the PIN34HI switch is held pressed during this command. Pressing the PIN34HI switch maintains a logic high on pin 34.

• F6 prepares to set a new name, which

Table 2
HC-05 Bluetooth Transceiver Partial Command List with *uCon* ShortCut Key ID

Command	*Response*	*Comments*
AT (*uCon* Key F2)	OK	Used at onset for checking the connection and data rate. If you can't get the OK response nothing else will work. See "Additional Notes" below.*
AT+VERSION (*uCon* Key F3)	+VERSION: 2.0-20100601	Prints module firmware version. This confirms the firmware is HC-05 compliant.
AT+ORGL (*uCon* Key F4)	OK	This command resets the module settings to original factory default settings.
AT+NAME? (*uCon* Key F5)	+NAME:H-C-2010-06-01	Prints existing Name that is used to connect to the device.*** Factory Default: Probably the firmware release date. Note: Pin 34 must be held high to get a response (hold PIN34HI button pressed).
AT+NAME=*name* (*uCon* Key F6)	OK	Prefix string for changing the name. Where *name* is any text string. This is the device name that the Bluetooth connect sequence will look for.**
AT+PSWD? (*uCon* Key F7)	+PSWD:*pswd*	Prints existing password needed to connect to the device. Default password: 1234
AT+PSWD=*pswd* (*uCon* Key F8)	OK	Prefix string for changing the password. Enter desired password where *pswd* is any four numbers.**
AT+UART? (*uCon* Key F9)	AT+UART =38400,0,0	Prints existing UART settings used for data transmission.
AT+UART= *baud*, *stop*, *parity* (*uCon* Mouse Click Buttons: B1..B10; — all settings 8N1)	OK	Set serial port data transfer mode parameters where: *baud*=desired data rate, *stop* is 0 for one stop bit or 1 for two stop bits, *parity* is 0 for no parity, 1 for odd or 2 for even. Default setting is 38400 baud, 8N1 which would be set with the script command (without the quotes): "AT+UART=38400,0,0\r\n" bauds accepted are: 4800, 9600, 19200, 38,400, 57,600, 115200, 234400, 460800, 921600, 1382400. Note: 8N1 stands for **8** data bits, **N**o parity, **1** stop bit.
AT+STATE? (*uCon* Mouse Click Button AT+STATE?)	+STATE: PAIRABLE	After the above settings are completed the transceiver state should be *pairable*. It should now be possible to connect to this device after pairing with a master device (Android Tablet) using the name and passwords defined above.

Additional Notes:
* After a PIN34HI button press or mode change it may be necessary to repeat the first command as it is not uncommon for an initial "ERROR:(0)" AT command error message due to the emptying of erroneous characters from the buffer. It should respond OK when the command is repeated.
** HC-05 firmware commands must be terminated with CR and LF. It is important that the terminal emulation software options be set so that a CRLF sequence is sent when the ENTER key is pressed (terminate commands with the "\r\n" escape characters in scripts).
*** Command will not respond with confirmation unless pin 34 is held at logic high. AT+NAME and AT+CLASS are two commands that are known to require pin 34 to be held high.

must be entered by the user. As shown in Figure 5 the name was changed to "VA7TA_BT02." One could now repeat the previous F5 step to confirm the name change (this is not shown in Figure 5).

• F7 reports the existing password. I chose to leave "1234" for my password. Thus the F8 step was skipped. F8 could be have been used to change the password.

• F9 reports the current data rate and port parameters settings.

• I chose to set the new data rate to 115,200 by pressing the 115200_8N1 button.

• AT+STATE? reported the state as "INITIALIZED." The setup is complete.

After completing the above setup sequence the module should be ready for operation and after reset power cycling it should come back up with the LED quick flashing again. At this point it should now be possible to pair the Bluetooth module with a master such as a tablet and connect to it using the newly assigned name.

Bluetooth Link Initial Testing

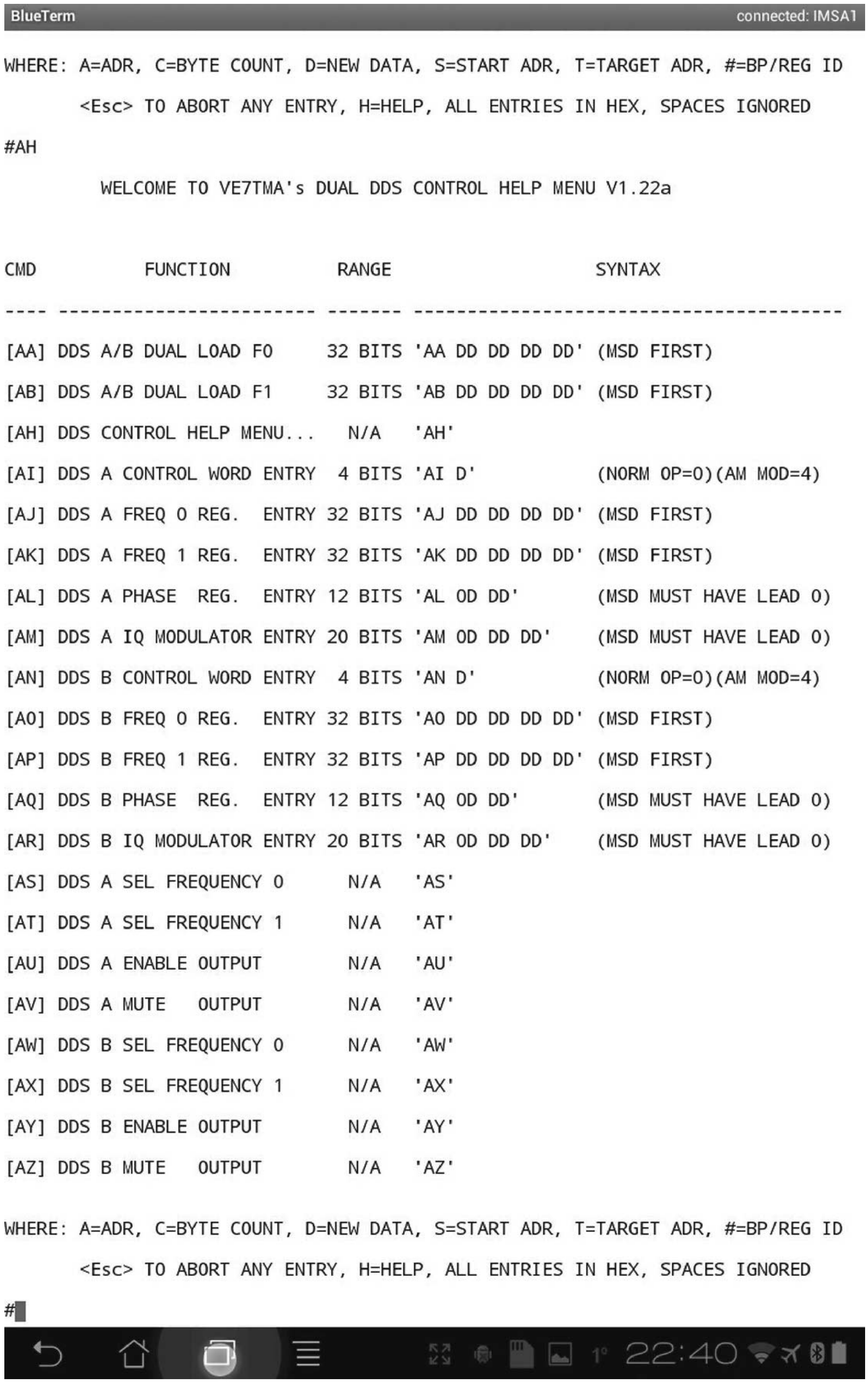
Bluetooth Link Establishment and Preliminary Testing

While the Bluetooth transceiver is still set up for parameter programming and still linked to the computer running *uCon* it is a good time to test the Bluetooth link with the tablet. This test sets up a wireless link to support keyboard communications between *uCon* running on the computer and *BlueTerm* running on the tablet.

The first step in establishing a Bluetooth link to the tablet is to enter the Android operating system *Settings* application and ensure that the Bluetooth service is supported (unfortunately not all Android tablets support Bluetooth) and if supported confirm that it is turned ON (Note that it is usually OFF by default to improve battery life). The Bluetooth service item is usually listed directly below the WiFi item in the top left hand side of the *Settings* screen. As the settings menu listing is quite long and extends off screen it may be necessary to scroll to the top of the list to find the Bluetooth item. The presence of an ON/OFF touch screen button for Bluetooth directly under the WiFi button confirms that the tablet is equipped with Bluetooth capability.

Once the Bluetooth service is confirmed available and turned ON it is necessary to pair the new Bluetooth module with the tablet. While still in the Android *Settings* menu ensure the *Wireless & Networks>Bluetooth* item is selected. A list of Bluetooth devices, which should include the name of the newly activated Bluetooth module (for the example above: "VA7TA_BT02"), should then be on the right under *AVAILABLE DEVICES*. Select the desired Bluetooth module and enter the PIN (for this example it was left at the factory setting of "*1234"*), select *OK* and the new device name should almost immediately be shifted into the *PAIRED* devices list. The Bluetooth module status LED sequence should switch from the quick flash to a slow repeating single quick blink indicating it has been paired but not connected. This tablet/Bluetooth transceiver pairing must be established to allow Android applications access to any new Bluetooth device. Without pairing even those applications that have been authorized to use the Bluetooth service would not be permitted to connect to the new Bluetooth transceiver.

Once the Bluetooth service is confirmed and turned ON, start the *BlueTerm* terminal



```
BlueTerm                                                    connected: IMSA1

WHERE: A=ADR, C=BYTE COUNT, D=NEW DATA, S=START ADR, T=TARGET ADR, #=BP/REG ID
       <Esc> TO ABORT ANY ENTRY, H=HELP, ALL ENTRIES IN HEX, SPACES IGNORED
#AH
         WELCOME TO VE7TMA's DUAL DDS CONTROL HELP MENU V1.22a

CMD          FUNCTION          RANGE                  SYNTAX
---- ------------------------ ------- ---------------------------------------
[AA] DDS A/B DUAL LOAD F0     32 BITS 'AA DD DD DD DD' (MSD FIRST)
[AB] DDS A/B DUAL LOAD F1     32 BITS 'AB DD DD DD DD' (MSD FIRST)
[AH] DDS CONTROL HELP MENU...   N/A   'AH'
[AI] DDS A CONTROL WORD ENTRY  4 BITS 'AI D'           (NORM OP=0)(AM MOD=4)
[AJ] DDS A FREQ 0 REG.  ENTRY 32 BITS 'AJ DD DD DD DD' (MSD FIRST)
[AK] DDS A FREQ 1 REG.  ENTRY 32 BITS 'AK DD DD DD DD' (MSD FIRST)
[AL] DDS A PHASE  REG.  ENTRY 12 BITS 'AL OD DD'       (MSD MUST HAVE LEAD 0)
[AM] DDS A IQ MODULATOR ENTRY 20 BITS 'AM OD DD DD'    (MSD MUST HAVE LEAD 0)
[AN] DDS B CONTROL WORD ENTRY  4 BITS 'AN D'           (NORM OP=0)(AM MOD=4)
[AO] DDS B FREQ 0 REG.  ENTRY 32 BITS 'AO DD DD DD DD' (MSD FIRST)
[AP] DDS B FREQ 1 REG.  ENTRY 32 BITS 'AP DD DD DD DD' (MSD FIRST)
[AQ] DDS B PHASE  REG.  ENTRY 12 BITS 'AQ OD DD'       (MSD MUST HAVE LEAD 0)
[AR] DDS B IQ MODULATOR ENTRY 20 BITS 'AR OD DD DD'    (MSD MUST HAVE LEAD 0)
[AS] DDS A SEL FREQUENCY 0      N/A   'AS'
[AT] DDS A SEL FREQUENCY 1      N/A   'AT'
[AU] DDS A ENABLE OUTPUT        N/A   'AU'
[AV] DDS A MUTE   OUTPUT        N/A   'AV'
[AW] DDS B SEL FREQUENCY 0      N/A   'AW'
[AX] DDS B SEL FREQUENCY 1      N/A   'AX'
[AY] DDS B ENABLE OUTPUT        N/A   'AY'
[AZ] DDS B MUTE   OUTPUT        N/A   'AZ'

WHERE: A=ADR, C=BYTE COUNT, D=NEW DATA, S=START ADR, T=TARGET ADR, #=BP/REG ID
       <Esc> TO ABORT ANY ENTRY, H=HELP, ALL ENTRIES IN HEX, SPACES IGNORED
#
                                                          22:40
```

Figure 6 — This screen capture from my Android tablet shows the *BlueTerm* screen displaying the help menu from a dual DDS control program.

program. Then touch the menu list icon (four horizontal line button symbol) located on the *BlueTerm* bottom task bar. Then touch the *Connect Device* menu item icon that appears. A list of in-range Bluetooth devices that have been previously paired to the tablet will appear. This list should contain the name of the newly activated Bluetooth module (for the example described here: "VA7TA_BT02"). Touching this name should initiate the Bluetooth connection, which will be confirmed by a *connected:NAME* notice in the top right of the *BlueTerm* title bar. It should also be confirmed by the Bluetooth module status LED switching to the slow repeating double blink sequence as an indication that it is connected.

Once connected, the link can be tested by sending text back and forth between the computer and the tablet via the *uCon* and *BlueTerm* applications respectively. Figure 6 is a screen capture of the *BlueTerm* screen on my Android tablet, as it displays the help menu for a DDS control program. Once this test is successfully completed the Bluetooth link will be ready for other uses once the test link is released. To release the link press the *BlueTerm MENU>DISCONNECT DEVICE* icon. This will release the Bluetooth connection, which will be confirmed by a *not connected* status notification in the *BlueTerm* title bar and the status LED on the Bluetooth module returning to the *Ready Mode* quick flash sequence. At this point the Bluetooth link is available for use by other apps including any Bluetooth based Amateur Radio GUI controller applications.

ASCII Interface Advantage

As mentioned in the earlier parts of this article a significant argument for adopting a standard ASCII character command line interface for a project is that the project can be tested from the tablet using a known good Bluetooth terminal application. This provides an independent reference test method to confirm that the project can be controlled properly via the wireless link. If a problem is encountered while one is debugging a new Android application, an ASCII interface provides the developer the option to employ a known good terminal program to confirm that the link and project are still functioning properly. I found that from time to time, when I ran into GUI development glitches, that this technique was useful to fall back on for confirming the command responses from my NimbleSig generator module. In all cases, I found the Bluetooth link to be error free, which was reassuring.

Bluetooth Transceiver Enclosure Options

Once the parameters are set up and the wireless link is successfully tested, the Bluetooth transceiver assembly needs to be suitably mounted and connected to the target equipment. In many typical cases it might be most convenient to just simply mount the HC-05 module within the project enclosure, while ensuring the 2.4 GHz circuit-board-etched antenna is not shielded from the surroundings. In the case of my NimbleSig III project, this was not an option because

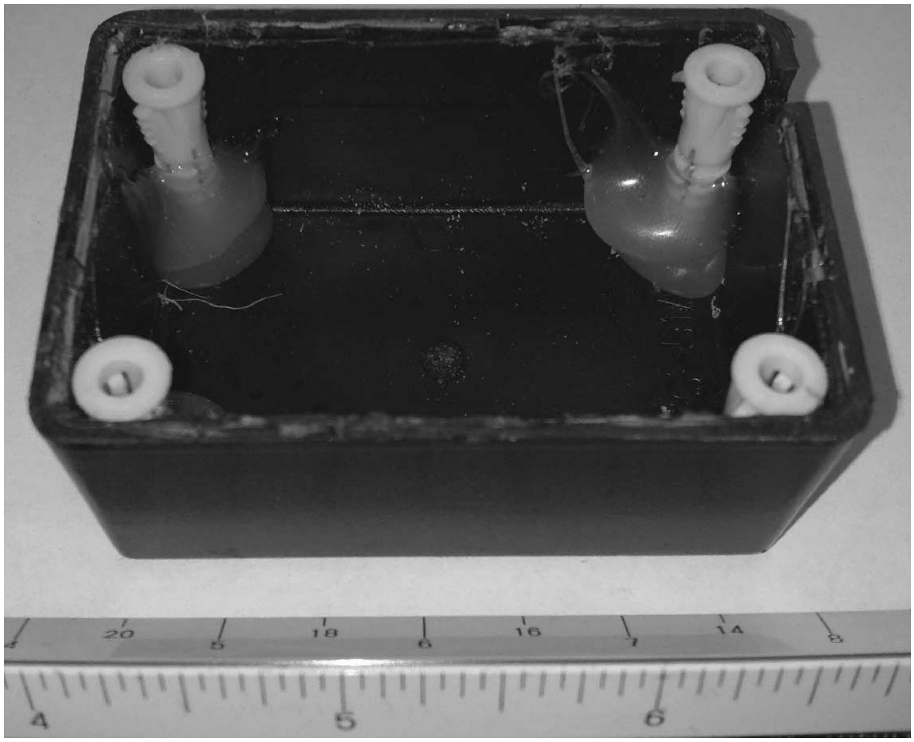


Figure 7 — Here is the empty "RF transparent" wall wart shell that I used to house my Bluetooth module. Note the plastic screw anchors hot glued into the corners to hold the cover screws.

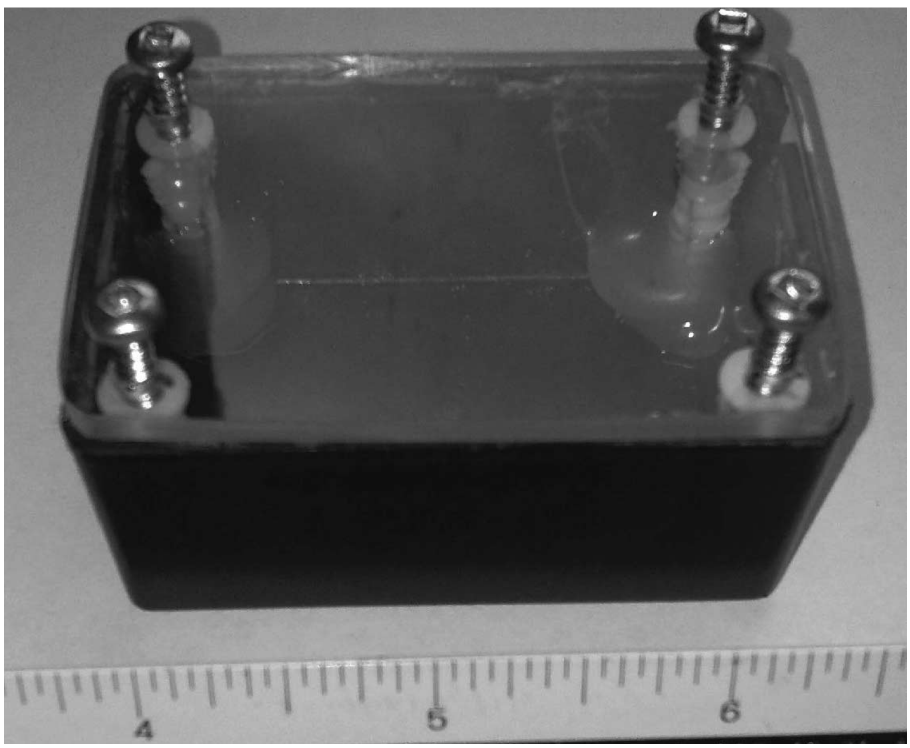


Figure 8 — With a clear plastic cover fastened in place on the wall wart shell, I am able to view the Bluetooth module status LED.

the NimbleSig III RF generator had been previously built and was already mounted within a small RF tight housing. Additionally, I wanted convenient experimentation access to the Bluetooth transceiver.

For the NimbleSig III interface I decided to use a plastic housing that would not shield the 2.4 GHz circuit-board-etched antenna nor block the view of the HC-05 status LED. I wanted convenient access to the TXD/RXD data streams so that I could easily connect my logic analyzer to view the serial data. Finally, to allow powering the whole project from a single regulated 5 V supply, I wanted to provide a 5 V output source for the NimbleSig III RF generator module. With these considerations in mind I decided to use the shell from a surplus, junk box wall wart power supply, which just happened to be about the right size. As shown in Figure 7, common plastic screw anchors hot glued into the shell corners worked nicely to

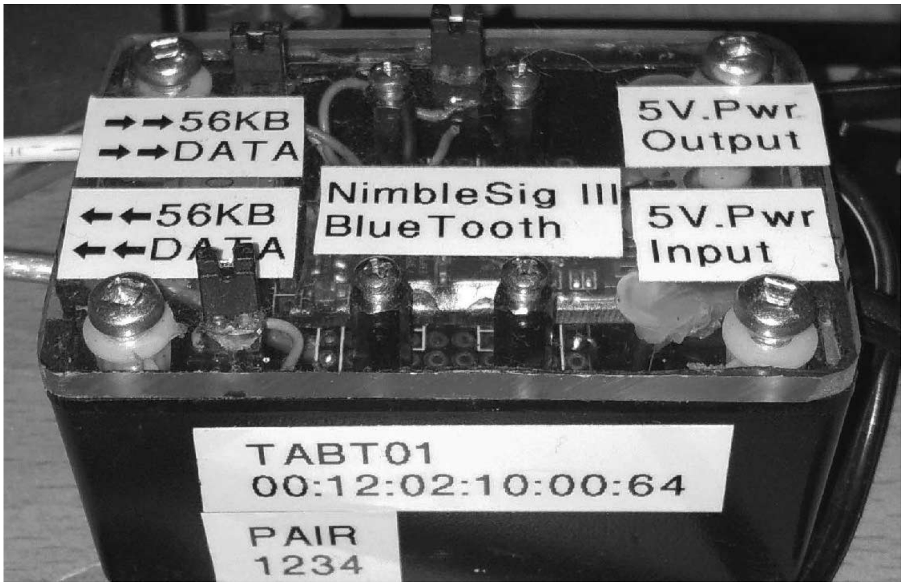


Figure 9 — Here is my completed Bluetooth adapter secured into the wall wart shell.

```
59.5K CP/M VERS 2.21

A>b:
B>dir
B: SYSGEN   COM : BOOT     ASM : DUMP     ASM : DDT      COM
B: SUBMIT   COM : LINKASM  COM : VDIAG    COM : MON      HEX
B: DUMP     COM : LOAD     COM : STAT     COM : XSUB     COM
B: ED       COM : MON      COM : MOVCPM   COM : MON      ASM
B: ED80     COM : CBIOS60  PRN : DUMP     HEX : PIP      COM
B: BIOS     ASM : DEBLOCK  ASM : DISKDEF  LIB : XDIR     COM
B: CBIOS    BAK : CBIOS    HEX : CPM60    COM : CBIOS60  HEX
B: LOADER60 HEX : LOADER60 ASM : CPMDVLOP SUB : CBIOS    ASM
B: CBIOS60  BAK : CBIOS60  ASM : TEST         : CPMDVLOP BAK
B: SYSGEN   SUB : LOADER60 PRN : SYSGEN   BAK : MODESET  ASM
B: COPY     ASM : DEBLOCK  OT$ : MAST     CAT : COPY     PRN
B: COPY     HEX : COPY     COM : -CPM2    BAK : -CPM2    2
B: TEST     DOC
B>type cbios.hex
:10000000F20000F20000F25F00F26200F20000F283
:100010000000F20000F26200F27F00F27E00F28F38
:1000200000F20000F20000F20000F20000F25F00B7
:10003000F20000CDD800CD7E00CD7F00CD5800CDA0
:100040005800CD8F00CD5800CD5800CD0000F200F3
:10005000007656CD00005676C9CDD800F25C00F28D
:1000600006F0F209F0F20CF0C956C9C9F40400C94F
:10007000565656565676CD1BF076C9CD0000C9CDE2
:100080001EF0C9CD0000C9CD21F0C9CD0000C9CDF9
:1000900024F0C9CD0000C9CD27F0C9CD0000C9CDDD
:1000A000D800CD0000CDD800CDD800F40300CD7E1F
:1000B00000CDD800CDCE00CDD800CDD800CD0C00DD
:1000C000CDD800C9CDD800CD0900F40300C95656DB
:1000D000C9F42400C9CD0C00C5CDDD00C187278F30
:0500E00027CD0C00C952
:0000000000

B>
```

Figure 10 — This screen shot from my Android tablet is the *BlueTerm* screen that shows the CP/M ASCII directory/hex file dump from my legacy IMSAI 8080 computer.

Bluetooth Transceiver Procurement Criteria

When selecting the transceiver to purchase, besides taking the usual online purchasing precautions, you should be aware of the following Bluetooth module specific selection criteria:

- Make certain that the item listing specifies the model HC-05 transceiver. It is possible to use a model HC-06 board as long as it is a "slave" version but the setup commands are limited, the firmware is not nearly as complete and as I found out there is at least one setup pitfall that needs to be avoided.
- The mother and daughter boards can be purchased separately or as a completed assembly with the daughter board pre-mounted. As there is little or no cost saving with purchasing the boards separately I prefer the pre-mounted version. Mounting the daughter board is not very difficult for those with surface mount construction skills, however.
- There are some backboards (motherboards) that do not have the push button (PIN34HI) switch, which is sometimes referred to as the "Clear" switch in online vendor listings. Try to purchase the version with the PIN34HI switch because this switch offers a nice convenience during the configuration of the module. As an alternative some of the backboards without the switch provide an external header pin connection sometimes labelled "Key." Placement of +3.3 V on this pin usually will provide the same function as pressing the PIN34HI switch. The switch is more convenient to use, however, especially if a +3.3 V voltage source isn't easy to access externally.
- The transceivers can be purchased either with or without desirable external connection leads. The nicest external lead option I have seen offered was in the form of a 4 conductor ribbon cable harness terminated with a 4 pin socket on one end and singleton wire-wrap pin socket connectors on the opposite end. I found the connection leads useful during the configuration setup procedure and then later for connecting to the target hardware. If you happen to already have a selection of header terminal jumper leads they could be used as an alternative.

provide a way to attach the cover. The clear polycarbonate cover for the shell, shown in Figure 8, provides an unobstructed view of the Bluetooth transceiver status LED.

I mounted the Bluetooth module face up via its 4 pin header connector to a small 100 mil grid prototype circuit board, which in turn was mounted with standoffs to the polycarbonate cover. The completed Bluetooth transceiver package is shown in Figure 9. The Bluetooth transceiver is powered from a commonly available, regulated 5 V dc wall wart power supply. The power is looped through the transceiver box for supplying power to the NimbleSig III module. I placed U-links on the TXD/RXD leads to provide the logic analyzer test points.

Legacy Equipment Control

Figure 10 is a tablet screen-shot of a *BlueTerm* window that shows a directory listing and a hex file dump from my 1979 vintage IMSAI 8080 computer. As shown, my IMSAI will still run the legendary CP/M operating system, which was well respected in its day. Like much legacy gear, the IMSAI 8080 provides an ASCII text command RS232C interface. When I built my IMSAI from a kit some 35 years ago, keyboard control was from a Vucom terminal that seemed to weigh about 100 pounds and occupy about 8 cubic feet of volume on my

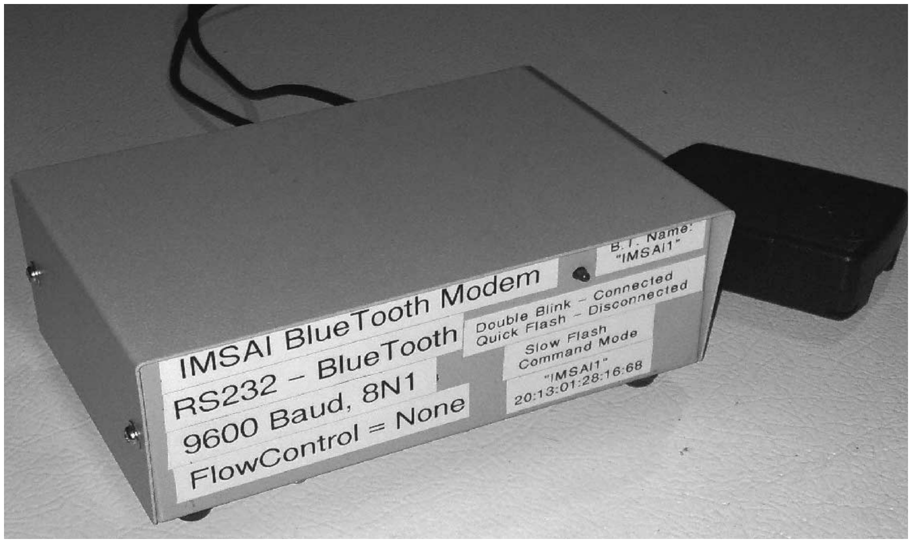


Figure 11 — I built an RS232 to Bluetooth converter into a metal enclosure to use with my legacy IMSAI 8080 computer and other applications. I added an LED indicator on the front panel to provide a Bluetooth status indicator.

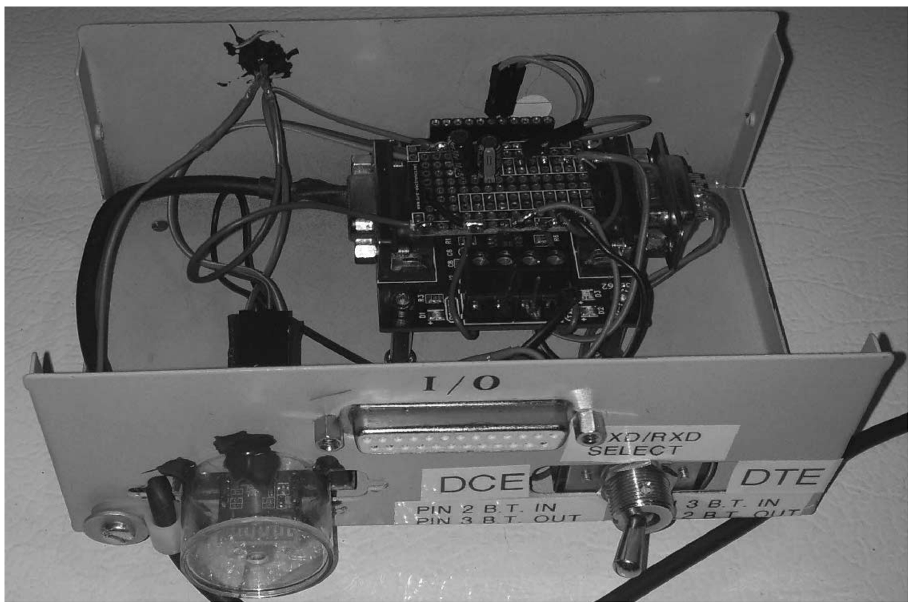


Figure 12 — I used the bottom of a plastic pill-bottle-bottle style container to form a radome for the 2.4 GHz etched-circuit-board antenna on the Bluetooth module

desktop. I certainly never imagined at the time that my then introductory microcontroller project would still be working three and a half decades later, and that I would be able to control it wirelessly with a portable hand held device. The IMSAI, which provides an RS232C ASCII console port interface, is a good example of how easy it is now to wirelessly control legacy equipment with an Android tablet. All that is needed is a simple-to-build RS232 adapter, an example of which is shown in Figures 11 and 12. For those that don't wish to build a Bluetooth to RS232 adapter from two separate modules there now are pre-assembled versions available.[7]

In many cases it should be inexpensive to adapt equipment with RS232 serial port ASCII character interfaces to Bluetooth for linking wirelessly to modern day mobile devices for user friendly, portable touch screen control.

The RS232 adapter project permits Bluetooth control of any ASCII controlled project that uses an RS232 9600 baud 8N1 interface. As outlined above, my main use for this interface is for nostalgic fun communicating with my 1979 IMSAI 8080 S100 CP/M computer using my modern day Android tablet. This adapter provides a Bluetooth-to-RS232 conversion. (In the day, the 25 pin D connector —a DB-25 — was commonly used for the RS232 interface.) It uses an HC-05 Bluetooth module connected to the RS232 lower deck half of the previously described Sure Electronics adapter (The USB upper deck of the adapter remains available for configuring Bluetooth modules or other possible uses.). A surplus printer A/B switch metal enclosure was used for the housing. The Bluetooth module is mounted so that the 2.4 GHz circuit-board-etched antenna pokes through one of the unused connector openings. A transparent plastic shell radome made from the bottom of a pill bottle style glue container provides protection for the HC-05 circuit board while allowing the 2.4 GHz etched antenna freedom to radiate into the surrounding area. A front panel mounted LED is extended from the Bluetooth module to provide a visible connection status indicator.

Conclusions

Although I am still climbing the steep portion of the Android development learning curve my initial wireless project control experiences have been rewarding and show future promise. I think development of Android applications will become easier as new development tools and more software examples become available. I believe the use of Bluetooth equipped hand held devices with user friendly touch screen graphics will become common place in the future for wirelessly controlling a variety of equipment within amateur radio stations.

Tom Alldread, VA7TA, became interested in electronics while still in grade school, repaired radio and television sets in his teens, obtained his Amateur Radio ticket at the age of 19, and went on to obtain commercial radio operator certification a year later. He subsequently graduated from the Capitol Radio Engineering Institute Engineering Technology program. His career was in the telecommunications industry where he worked in microwave and VHF radio equipment maintenance, training, engineering standards and design, and long distance network operations management.

Tom is now retired and lives on Vancouver Island with his wife Sylvia, VA7SA. Tom has a history in repeater building and emergency communications, and he enjoys operating CW, building equipment and is currently the net manager for the 20 m Trans Canada Net. He is interested in microcontroller development projects associated with RF technology. He had the good fortune of winning second place in the 2006 Design Stellaris and first place in the 2011 Renesas RX international microcontroller design contests. Other hobbies include computing, hobby farming and bicycling.

Notes

[1]For a tutorial on the Bluetooth HC-05 command set you can visit **http://robopoly.epfl.ch/files/content/sites/robopoly/files/Tutoriels/Bluetooth/HC05-at_command_set.pdf**.

[2]You can find a data sheet for the HC Bluetooth transceiver modules at **www.exp-tech.de/service/datasheet/HC-Serial-Bluetooth-Products.pdf**.

[3]There is more information about the Sure Electronics USB adapter at: **http://store.sure-electronics.com/mb-cm13112.** At publication time this converter could also be found on ebay under "CP2102 USB/TTL/RS232 Serial Port Converter Transceiver.

[4]The driver files and documentation for the Silicon Labs CP210 chipset used on the Sure Electronics USB adapter are available for download from: **www.silabs.com/Support%20Documents/Software/CP210x_VCP_Windows.zip**

[5]You can learn more about the *uCon* terminal emulator software and download the program file at the *uCon* home site:
www.umonfw.com/ucon/
There is also a self-extracting *uCon* installation file available for direct download at:
www.umonfw.com/releases/ucon_install.exe

[6]The program files for this project are all open source code files. They are available for download from the ARRL *QEX* files website. Go to **www.arrl.org/qexfiles** and look for the file **9x14_Alldread.zip**.

[7]One such adapter, online priced at about $15-20, that appears to use a Bluetooth daughter board similar to the HC-05 has the part number "BLK-MD-BC04-B_DEMO" silk-screened onto the circuit board. A Google search will turn up several vendors.

Jim Satterwhite, K4HJU

15006 Sawyer Ranch Rd, Austin, TX 78737; **k4hju@arrl.net**

Calibration and Monitoring of Frequency Standards — Phase Method

That frequency standard is only as good as your calibration technique allows. The author describes a monitoring system he uses to keep his under control.

In this article I am speaking primarily to the engineer, technician or experimenter who has an interest in precision frequency standards and their calibration and monitoring. This article will deal with the calibration and monitoring of the various 10 MHz frequency standards. I wanted to measure frequency sources dynamically to an accuracy in the vicinity of one part in 10^{12}. This resulted in designing and building a high performance test set for this purpose. The purpose of the design is to detect a phase difference as quickly as possible to (A) detect short-term drift and (B) minimize measurement time. The performance achieved is an error display noise floor of approximately of 2 parts per trillion.

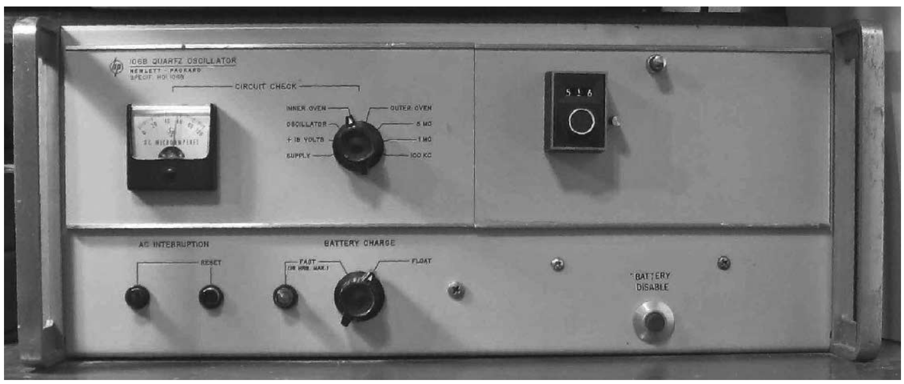


Figure 1 — My HP-106B quartz oscillator.

Years ago, I acquired an HP-106B frequency standard, an early 1960s frequency standard intended as a reference for calibrating other standards at remote locations. It would be calibrated, put on an airplane and taken to the remote location. Then the remote standard would be calibrated using the HP-106B as the reference. The HP-106B contains a 5 MHz crystal oscillator in a double oven. See Figure 1. Frequency standard technology has come a long way since then.

I searched for ways to verify the calibration of this instrument. What is involved and how do I go about doing it? I had designed and built what I consider to be some quite good and versatile frequency counters, and was familiar with frequency measurement; at least to parts in 10^8, but the HP-106B specification is a stability of $\pm 5 \times 10^{-11}$ over 24 hours. When the objective becomes accuracy in the vicinity of one part in 10^{11} or 10^{12}, the challenge increases considerably.

Low-tech ways of calibrating the HP-106B would involve comparing the device under test (DUT) with a standard reference using one of the following methods:

Audio Zero Beat: Listening on a receiver for a zero-beat signal when the output of the device under test is mixed with the signal from an external reference such as WWV.

- Accuracy is poor, because the ear can only detect errors of a few parts per hundred.
- Accuracy is also determined by the low-frequency response of the audio portions of the receiver because phase errors of one or two hertz must be audible to the observer.

Frequency Counter: Directly measuring the frequency of the device under test with a frequency counter.

- Accuracy is limited because most counters have internal references that are accurate to a few parts per million for low end references and tens of parts per billion for high end references.
- Since the HP-106B is capable of precision approaching $\pm 5 \times 10^{-11}$, the reference to which it will be compared must be at least as accurate, and by preference it will be at least an order of magnitude better than this.

Lissajous Pattern: By applying the reference and device under test signals to the X and Y channels of an oscilloscope, a Lissajous pattern can be created. The pattern goes from an ellipse to a 45° line and back to an ellipse as the phase between the reference and device under test varies through 360°. This is quite effective for tuning over a limited range; it is poor for measurement, however.

Table 1
Common Accuracy Terms

Description Common Term	*Scientific Notation Abbreviation*
One part in 10^6	1×10^{-6}
Parts per million	ppm
One part in 10^9	1×10^{-9}
Parts per billion	ppb
One part in 10^{12}	1×10^{-12}
Parts per trillion	ppt

Table 2
Relative Accuracy of Various Frequency References

Reference	*Modification*	*Accuracy Range*
Crystal	—	1 to 100 ppm
Crystal	Temperature Compensated TCXO	~ 0.1 ppm
Crystal	Ovenized OXCO	0.001 - 0.1 ppm
Crystal	Double Oven	~ 50 ppt
Crystal	GPS supervised OXCO	~ 5 ppt
Rubidium	—	~ 50 ppt
Rubidium	GPS supervised	~ 5 ppt
Cesium	—	0.01 to 0.1 ppt
Hydrogen Maser	Passive	1 ppt
Hydrogen Maser	Active	0.0007 ppt

Reference Signal Sources

In discussing accuracy several terminologies are commonly used. Table 1 summarizes this terminology.

The first part of frequency calibration is to compare the unknown frequency to a frequency known to an accuracy and precision greater than the desired calibration level. That means at the outset you have to have access to a source more accurate than the desired calibration level.

References for frequency measurement cover the gambit from stand-alone crystal oscillators on the low end to Cesium standards and beyond on the high end. Somewhat of an overview is given below for the various reference technologies in Table 2. For the budget concerned experimenter, there are two good choices.

1) Rubidium standards are appearing as surplus items because they are being retired from service in cellular telephone systems around the world. Frequency-stable outputs can be accurate at the 10s of parts per trillion (ppt) level.

2) Crystal oscillators can be "disciplined" — phase-locked — to US Global Positioning System (GPS) satellites.

- The GPS system requires that individual satellite clocks be maintained at accuracies exceeding a few parts per trillion.
- Earth-bound oscillators can be synchronized with the GPS satellite clocks, using timing information derived from the GPS signal.

Calibration

In order to calibrate a standard or device under test, it must first be measured relative to a known reference. Then the task is to adjust the standard or device under test into agreement with the reference.

Measurement

Given the availability of such references, the *metrologist* performing a verification or calibration must answer two important questions:

1) Is a single, brief frequency comparison sufficient to ensure an accurate measurement?

2) In making the measurement, how does one compare two frequency sources and estimate the difference/error between them?

The answer to the first question is No.

- Separate measurements an hour, a day, or a week apart do not confirm that the difference between a reference and a device under test is consistent.
- At best, they can only confirm that the measured error was the same at two points in time. They could not show that the device under test frequency remained constant through the measurement period.
- Many measurements at short intervals over a long time are needed to establish the variation of the device under test with time.
  - The results can be plotted to show, graphically, how the reference and device under test frequencies differed over time.
  - The results can be "averaged" to arrive at a crude figure of merit for the device under test.

A number of answers can be given to the second question:

- A frequency counter with a sufficiently accurate time base can be programmed to accumulate measurements over a period of tens, hundreds, or even thousands of sec-

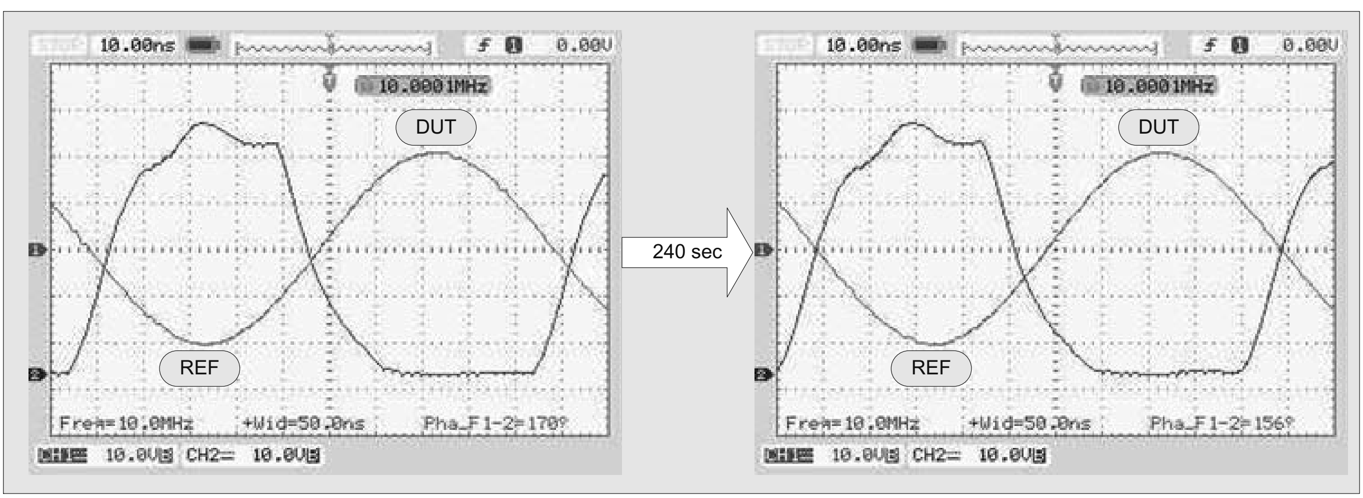


Figure 2 — These screen shots show an example of relative phase measurement with an oscilloscope.

onds. The long measurement time base improves overall accuracy by integrating errors over time; transient deviations are averaged, however.

- Relative phase method: when the frequencies of the reference and the device under test are nearly the same, measuring the difference in the phases of the two signals versus time may also be possible.
    - Oscilloscope — With both reference and device under test signals applied to the inputs of an oscilloscope, the scope can be triggered by the reference signal and the relative phase can be compared at a time A and at a time B. Then with the difference in time between A and B and the difference in the phase readings, either in degrees or in time, the difference in frequency can be calculated. An example is shown in Figure 2. Here the time is 240 seconds and the phase is 156° to 170°, or a shift of 14°, to yield a difference in frequency of 16 ppt. This example is rather straightforward in that the oscilloscope being used here computes phase for us. An eyeball estimation of the phase is somewhat tedious. Also, take note that variations in the phase difference during that time are practically impossible to observe.
    - Strip Chart — My first attempt at verifying the accuracy of my HP-106B standard involved comparing its output to WWVB at 60 kHz and recording the relative phase of the WWVB reference and the device under test on a strip chart recorder.
        - This method can be time consuming because resolving a difference of only 10° at an accuracy of 1×10^{-11} would require 12.9 hours. Increasing the resolution accuracy to 1×10^{-12} would require extending the measurement time to 5.33 days!
        - The measurement time can be reduced when comparisons are done at higher frequencies.

Adjustment

Having measured the frequency difference, the next task in calibration is to adjust the device under test to some given limit, or as close as possible to the reference frequency. When long measurement times are encountered, the adjustment process becomes problematic since the operator cannot see the effect of the adjustment until another measurement is made. It takes time and patience to make this process converge. If the operator can see the effect in real time, the process becomes much easier and straightforward.

Automated Relative Phase Method

Automating the Relative Phase Method for comparing the frequencies of two sources can be extremely accurate if an accurate reference is available.

1) The method determines the rate of change of phase between two sources. A computer monitors the phase difference between them and presents the error graphically.

2) For example, a one-hertz phase error (one 360° phase change per second or 1 Hz frequency difference) represents a phase change of 360° per second. At 10 MHz, the time required to complete one hertz (one RF cycle) is 100 ns, so the error due to phase change can be described as 100 ns per second.

3) We can elaborate the idea by stating the change in degrees of phase error per second for a given fraction of a cycle. Thus for a 10 MHz comparison, Table 3 shows some useful relations.

4) Clearly, small phase errors approaching 0.0036 degrees/second or 0.001 nanosecond/second become visible only after many seconds if they are to be observed on an oscilloscope. A 1 ns error, for example, completes a full 360° phase shift in 1×10^9 seconds, which is nearly 280 hours, or 11½ days.

4) An important human factor involved with these types of measurements on an oscilloscope is keeping track of the variations over long periods of time.

From Table 3, some useful conversion constants for calculations emerge.

- degrees / s = (1/0.0036) ppt
- degrees /s = (1/360) ppm
- ns / s = 1000 ppt
- ns /s = 0.001 ppm

This method must be automated if accuracies approaching 1 ppt are to be realized.

Monitoring

Something I have found interesting in working with precision oscillators is that rarely are they what they seem at first glance. Since I have designed and built the instrument described in this article, I have found the true performance of several standards I have in my lab to be less ideal than I had expected. Because you calibrate an oscillator to 1.1×10^{-10} and you come back tomorrow and it reads 1.2×10^{-10} doesn't necessarily mean that you can depend on it having 2×10^{-10} accuracy for general measurements. Now that I have the capability of monitoring various sources at the 10^{-11} or 10^{-12} level, I have greater reservations about the performance I can expect from a given source with a certain calibration level. If you want to know how a source will truly perform for a given application, I believe it is essential to monitor the error performance over a period of time, and from that determine what you can count on for the performance of that source.

The Test Set

I chose to accurately measure the relative error between a reference standard of known accuracy and an unknown device under test (DUT), dynamically, and in real time. I determined that the best candidate for this

Phase Comparison at 60 kHz

At 60 kHz, a frequency error of 10 ppt represents a phase change difference between the reference and the device under test of 0.0002156° / s. Thus, to accumulate a phase change difference of 10° would require 46382 seconds or 12.9 hours. Achieving a frequency error of 1 ppt would require 463820 seconds or 5.33 days.

For 10 MHz the times are much shorter. For the same error levels as for 60 kHz, the times would be 278 s (10 ppt) and 2783 s (1ppt).

Of course, resolving the frequency difference change accurately depends on being able to actually see the phase difference change on the medium presented, such as the strip chart or oscilloscope, accurately. That is another matter entirely.

Table 3
10 MHz Error Values Versus Frequency Error

Error (Hz)	*ppm / ppt*	*Degrees /s*	*ns / s*
1	0.1 ppm	360	100
0.1	0.01 ppm	36	10
0.01	0.001 ppm	3.6	1
0.001	100 ppt	0.36	0.1
0.0001	10 ppt	0.036	0.01
0.00001	1 ppt	0.0036	0.001

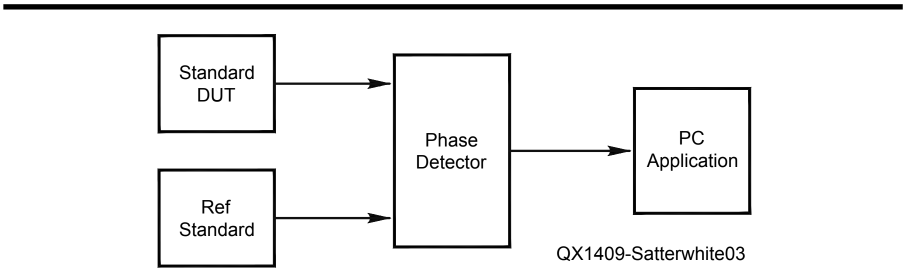


Figure 3 — This block diagram illustrates the basic concept of the phase method of calibrating the oscillator.

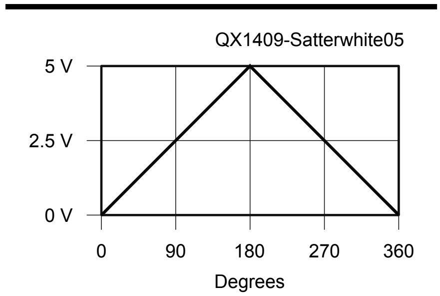


Figure 5 — This graph shows a single XOR gate phase detector output.

measurement was an accurate phase measurement and automation of the method with as much precision as I could bring to bear on the implementation. Further, I wanted to present the results in real time on a computer screen in strip chart form for observation. In this way I could observe the real time performance of various standards. What follows is the result of the evolution of that development. This design limits the input frequencies to 10 MHz for both the reference and the device under test with one exception. The device under test input can accept a 5 MHz input by moving a jumper.

Overview

As shown in Figure 3, the basic concept is simple. The specifics get a little more interesting.

The figure shows that the phase of the reference and device under test are compared and a value relating to the phase difference is passed to the computer. The computer software initiates a fixed time interval trigger for each phase measurement. Calculations that result in the computed error being displayed on the computer screen are made from the fixed time interval and the measured phase difference for each interval.

Figure 4 shows the detailed block diagram of the test set. To achieve high accuracy, the signals must be conditioned significantly. Each channel is first detected and phase-locked to a higher frequency crystal VCO to obtain a precise square wave for presentation to the phase detector. This is necessary because, in simply squaring the input signals, it is difficult to maintain the necessary 50% duty cycle required for precise phase determination. In addition, for the quadrature phase detector chosen, the reference frequency needs to be four times the reference frequency. This will be discussed below.

The reference and device under test channels are almost identical. The exceptions are that the reference channel operates at 40 MHz and its output supplies clocks for the quadrature phase detector. I will discuss the quadrature phase detector below.

The reference input signal is converted from analog to digital form by the input

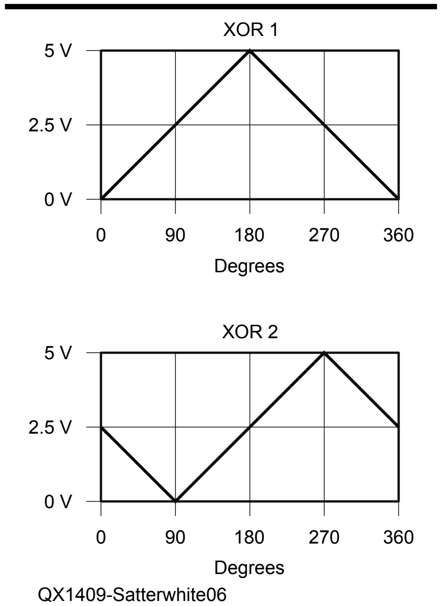


Figure 6 — Here is a graph of the Quadrature XOR phase detector outputs.

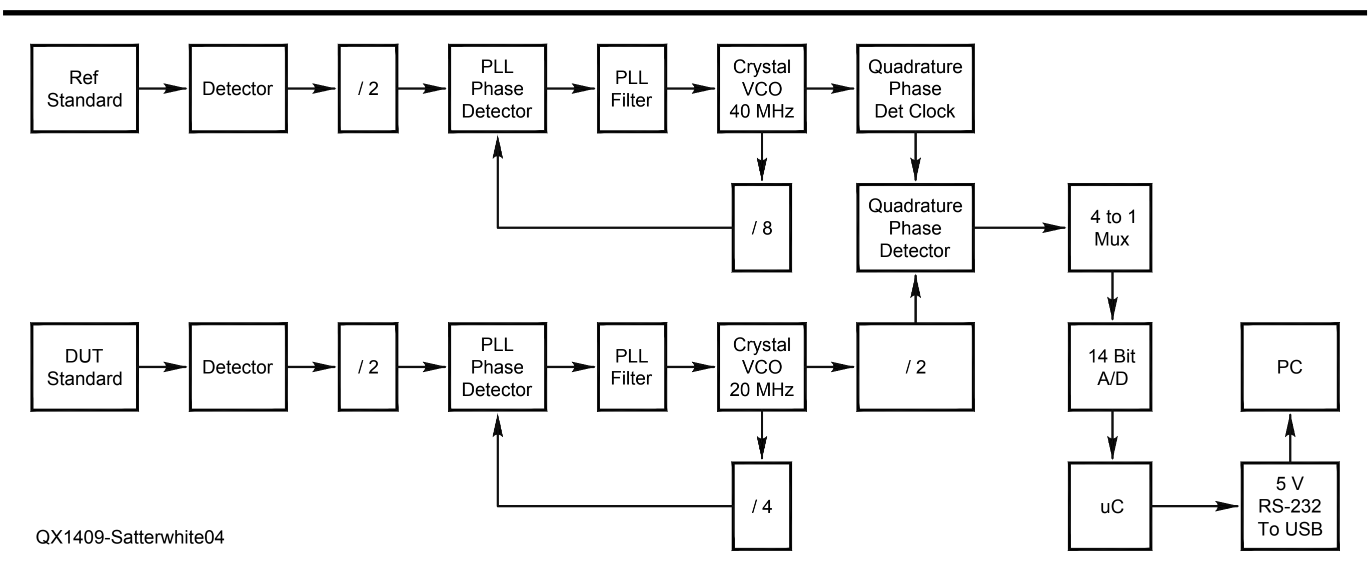


Figure 4 — Here is a detailed block diagram of the system.

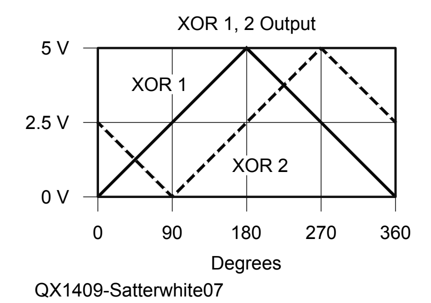


Figure 7 — This graph is an overlay of the Quadrature XOR phase detector outputs.

detector and is then divided by two by a digital counter. The 5 MHz output feeds the 74HC4046 PLL phase detector. I prefer to operate this PLL phase detector at somewhat less than 10 MHz. The output of the PLL phase detector feeds the crystal VCO via the PLL filter. The 40 MHz VCO output feeds a divide by eight circuit, which supplies a 5 MHz square wave to the PLL phase detector to close the 40 MHz PLL loop. It also drives a circuit to derive the reference clocks for the quadrature phase detector.

The device under test channel is identical to the reference channel, with two exceptions. The crystal VCO is operated at 20 MHz and the output is divided by two to feed the quadrature phase detector at 10 MHz and divided by four to feed the PLL phase detector at 5 MHz, closing the 5 MHz PLL loop. The 74HC4046 PLL phase detector is fed from the input divided by two for a 10 MHz device under test input or moving a jumper bypass the divide by two for a 5 MHz device under test input.

The outputs of the reference and device under test channels are compared in the quadrature phase detector (QPD) and fed to a 14 bit A/D that is coupled to the microcontroller via the Serial Peripheral Interface (SPI). The microcontroller takes in the digitized phase values and routes them to the computer via a 5 V RS-232C-to-USB adapter. The computer, running a *Windows* application written in National Instruments *Lab Windows/CVI*, processes the data and presents the analyzed results on the computer monitor.

Quadrature Phase Detector

The phase detector chosen for this implementation is the Exclusive-OR (XOR) logic gate. For a single XOR, the output waveform for a phase difference from 0° to 360° is shown in Figure 5. The output, when filtered, is a triangular wave whose amplitude varies with the phase difference between the two inputs. Note that the output is multi-valued, that is there are two phase values for each voltage value, therefore the unique phase cannot be determined from the voltage reading without additional information.

This can be remedied by the addition of a second XOR gate driven by the reference signal, which is 90° out of phase and which forms the quadrature phase detector. The quadrature phase detector reference signals are derived from the input reference clock multiplied by four. The result is shown in Figure 6 and overlaid on the same trace in Figure 7.

From the combination of the two outputs, the in-phase component (I) and the quadrature component (Q), the unique phase can be determined, since there is a unique value pair for each phase point.

The circuit I use for the generation of the quadrature clocks is shown in Figure 13. It uses two additional XORs to help remove a very slight nonlinearity of phase versus amplitude at the peaks and valleys of the output. For this, I generate 10 MHz clocks at 0° (I), 90° (Q), 180° (–I) and 270° (–Q). Figure 8 shows the XOR outputs using the four clocks. I and Q are the in-phase and quadrature phase components.

The I and –I and the Q and –Q components are combined in software to give the resultant I and Q from which the phase calculations are made. From the I and Q, the phase value is determined. At a sample rate of 10 samples/second and given the calculated phase values, the phase change rate is determined. The phase change rate multiplied by a conversion factor is the basis for the short-term error. The long-term error is

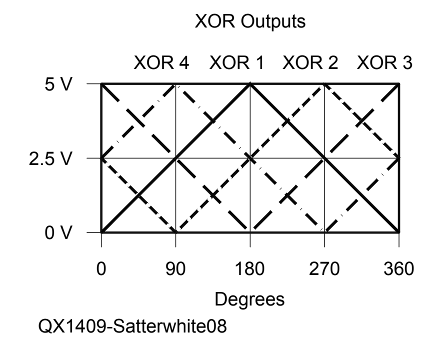


Figure 8 — Outputs of the four XORs versus device under test phase.

GUI Trace Description

The GUI is in color and the Figures are presented without color. On the GUI, the red/pink trace is the short-term error and the blue/light blue trace is the total accumulated averaged error since the test began. Since negative values are problematic on a log chart, the error values are differentiated for sign by color shading, with the dark colors representing positive error and the light shading being the negative error. The green trace is the accumulated phase error in degrees. For this display the phase display is limited to values between –360° and +360°, so that we see the trend in phase with time.

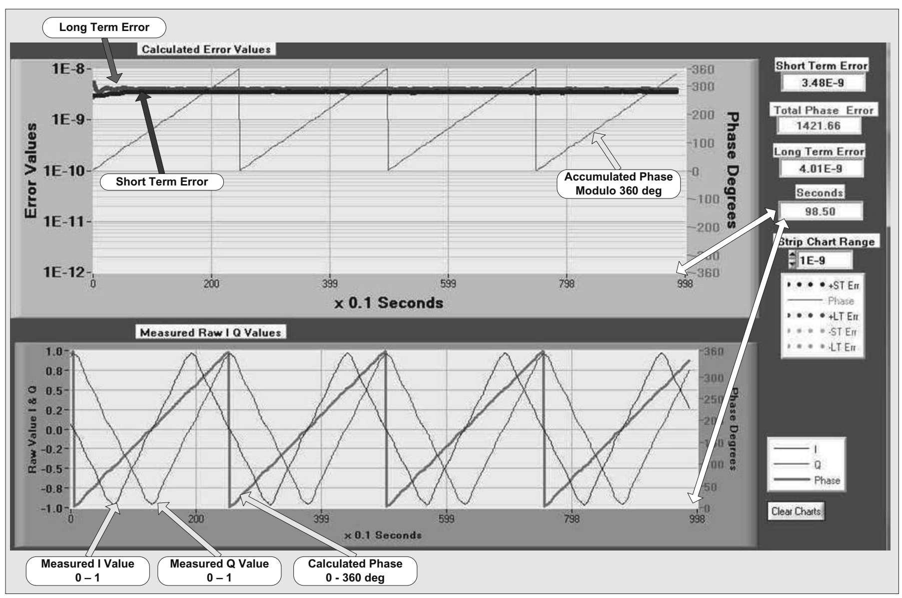


Figure 9 — Here is an example of the Quadrature phase detector output.

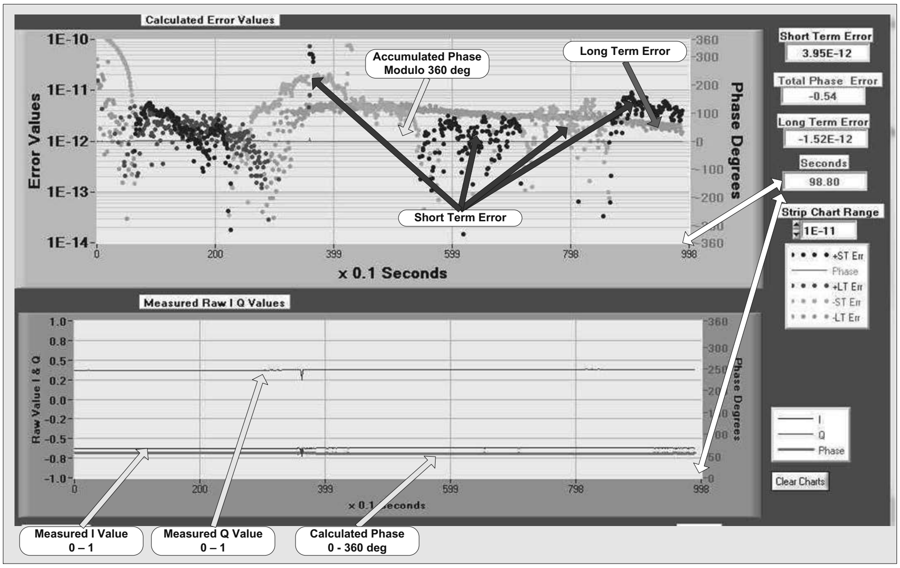


Figure 10 — This is an example of the output data.

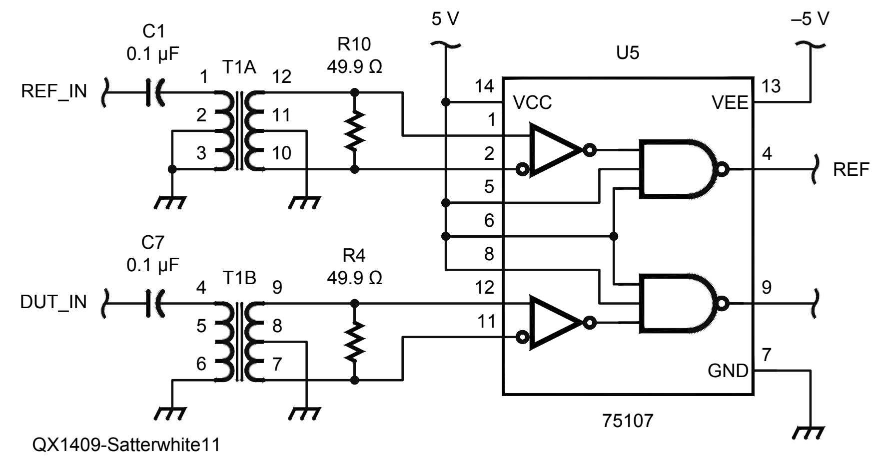


Figure 11 — This is a schematic diagram of the input detector.

calculated by taking the first phase reading in a given run as the start reference. Then the present phase difference minus the start reference divided by the total time since start is the long-term degrees/second that times the conversion factor is the long term error.

Figure 9 shows a screen presentation from the GUI for a period of 98.5 s. A Trimble Thunderbolt GPS-supervised source is used as the reference input and an ovenized crystal oscillator as the device under test input to the test set. There are two strip chart presentations. The lower strip chart shows the measured I and Q values changing with time and the phase calculated from the two. From the calculated phase, the short-term (incremental phase change versus time increment) and the long-term (total phase change versus total time) errors along with the accumulated phase are calculated and presented in the upper strip chart in the figure. In the GUI, colors are used to identify the various traces. Here, the long-term error is 4.01 ppb over a period of 98.5 s, and the last increment of short-term error is 3.48 ppb.

Figure 10 shows an annotated screen presentation of measurements using a Trimble Thunderbolt GPS-supervised source as the reference and a rubidium oscillator as the device under test. Here the phase movement is so slow that the quadrature phase detector outputs look almost stationary (lower strip chart). The top strip chart shows the short-term and long-term error traces and the accumulated phase difference trace.

Note that the scale on the left of the upper strip chart has been adjusted to better show the excursions of the traces. If we study the traces closely we can determine the extent of the long-term error trace as it stabilizes and remains rather convergent while the short-term trace varies considerably. This test is approaching the limit of the test set, as can be seen below in the section on performance. Note that the total phase error for the 98.8 s is only –0.54°, giving a long-term error of –1.52 ppt.

Processing

High Level Circuit Description

The input detector (Figure 11) is designed to accept a variety of signal input levels and types. The transformer provides ground isolation between the two sources. An attempt has been made to isolate the reference and device under test inputs as much as possible to reduce the possibility of entraining of either of the two sources. Some oscillators exhibit the possibility of entraining with minute cross-coupling from a source close to their natural frequency and these two sources are quite close in frequency. The 75107 line receiver provides a reliable transition detector function.

The VCO does not necessarily give a symmetric waveform output, so a divide by two is necessary to guarantee essential symmetry. Symmetry is particularly important in driving the phase detectors to derive phase as precisely as possible.

The quadrature phase detector shown in Figure 12 offers the ability to uniquely identify the phase difference from 0° to 360°. Only two XORs are necessary to accomplish this; however the second two allow a bit more resolution, as will be discussed below.

Figure 12 shows the quadrature phase detector. The device under test signal is compared to the reference signal at 0°, 90°, 180°, and 270° to give the Q, I, –Q and –I signals respectively. Each XOR detector is followed by a pull-down resistor to –5 V and an RC low pass filter. The detector outputs go through an analog multiplexer to the A/D converter. The pull-down resistor to –5 V helps increase the sharpness of the minimum value of the detector output.

To ensure symmetry, the circuit must be driven by four times the output frequency. Figure 13 shows the circuit for the generation of the quadrature phase detector clocks. Starting with the clock output of the 40 MHz crystal VCO that is phase locked to the 10 MHz reference input, the 40 MHz clock drives the four stage synchronous counter to derive the 5 MHz PLL feedback and the D flip-flops to derive the four quadrature phase detector reference clocks.

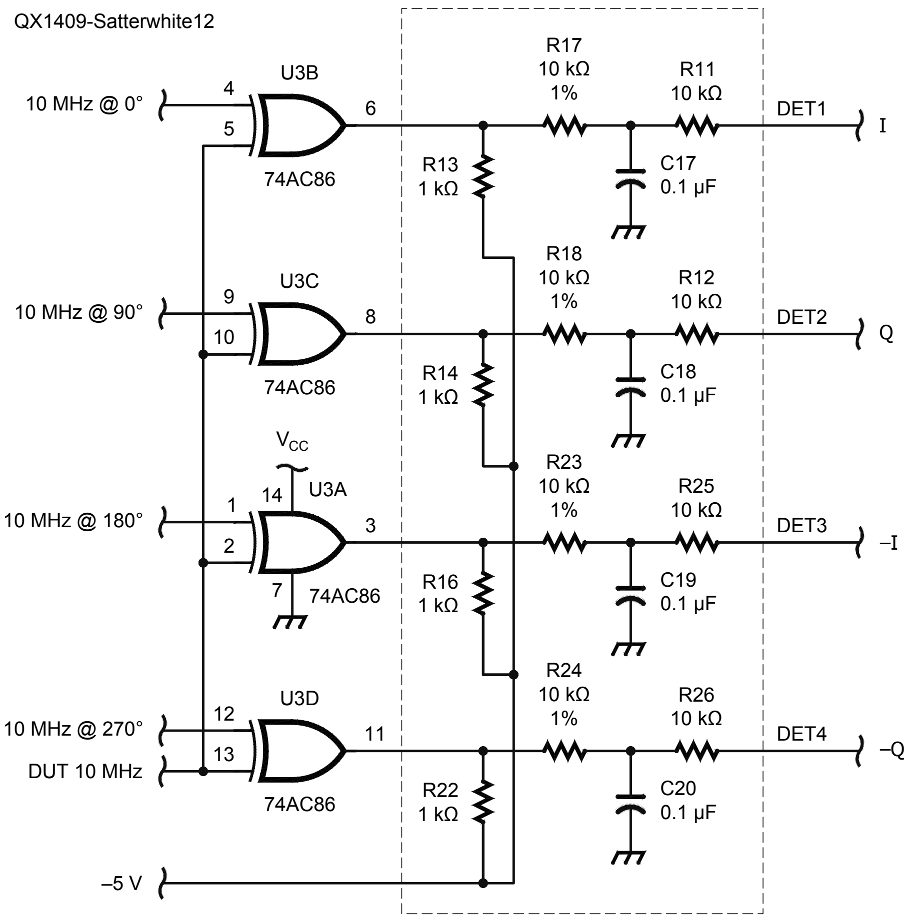


Figure 12 — Here is the schematic of the Quadrature phase detector.

Practical Considerations Limiting Performance

A number of practical considerations limit the performance of the over all system. It is interesting to look at the parameters that limit the performance of the test set. Of course the primary limit is the accuracy and stability of the reference source. Beyond that is the accuracy and stability of the PLLs. Further the short-term accuracy is particularly dependent on the resolution of the A/D.

Reference Signal Sources

It is important to note that for a meaningful reading, the reference accuracy must be at least an order of magnitude better than the desired level of measurement. High-accuracy reference sources are available, but most are expensive and for the average experimenter, cost is important. The Trimble Thunderbolt GPS supervised source and various Rubidium sources are available from surplus sources.

Trimble Thunderbolt

There are at least two kinds of GPS-supervised sources. The one commonly seen in experimenter circles uses the GPS one-pulse-per-second (1PPS) output, that is compared with an ovenized crystal oscillator or a Rubidium source and used to derive a signal to correct the source. The second, which is used by Trimble in the Thunderbolt derives the GPS receiver reference clock from an ovenized crystal oscillator. The timing error from the GPS position solution corrects the

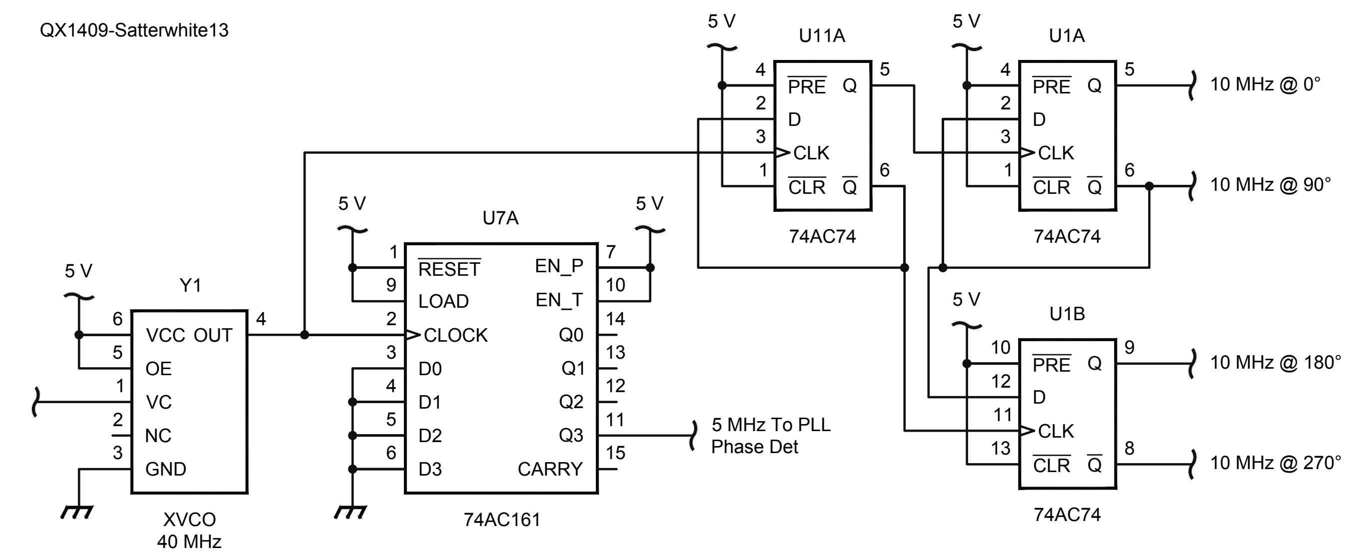


Figure 13 — This is the Quadrature clock circuit.

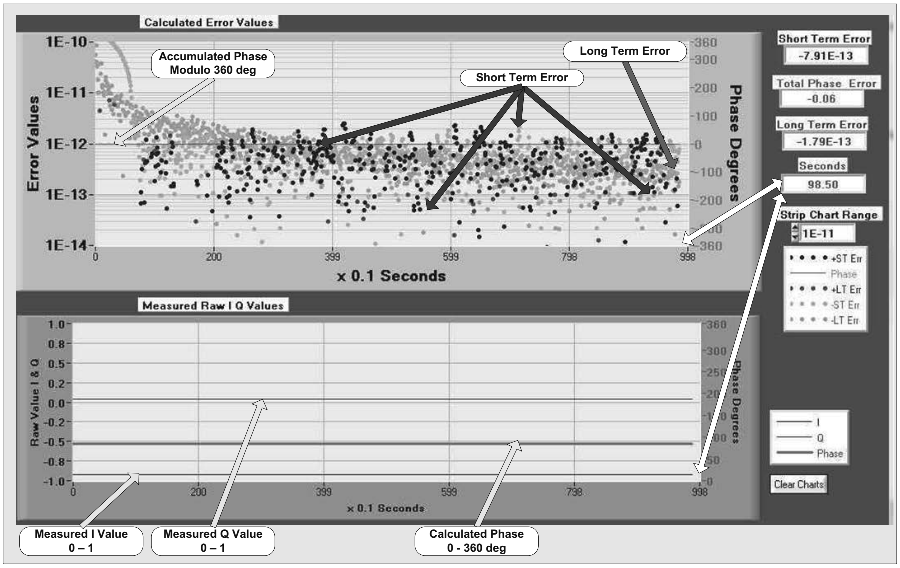


Figure 14 — Here is the data from a Trimble Thunderbolt to Reference and device under test Inputs.

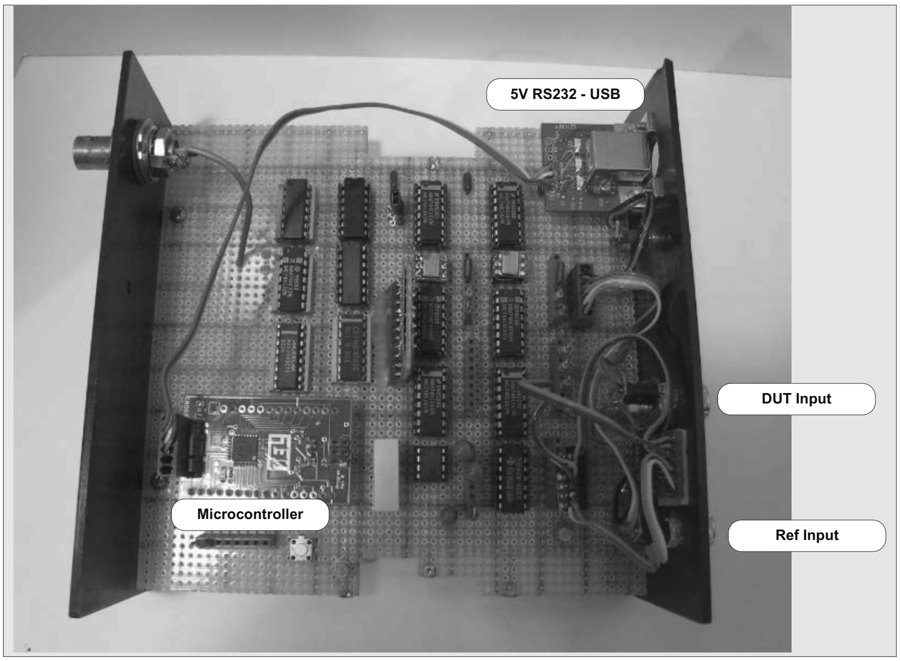


Figure 15 — This photo is a component view of the test set.

ovenized crystal oscillator. The problem with the 1 pulse per second technique is that it is difficult to determine the difference between the supervised source and the 1 pps signal accurately, so making accurate corrections is problematic.

It is interesting that Trimble does not specify the accuracy of the Thunderbolt 10 MHz output, but they do specify the phase noise. I suspect the lack of an accuracy statement is due to the various potential user-related situations where a solid GPS solution is not possible or is intermittent. Such situations might include the setup or the antenna placement. The *Windows* utility they supply to set up and monitor the unit in operation does give a dynamic error report and with the unit I have that error varies around parts in 10^{-12}. Some thought about the stability of these types of sources reveals that at some level they must jitter around. Consider that the source is an ovenized crystal oscillator (VCXO) and with a very few exceptions it is very difficult to make them perform better than 1×10^{-10}. (The exceptions mentioned refer to ovenized crystal oscillators built as instruments such as the HP-106B that can perform to a higher level.) To me that means that in this application, with the VCXO controlled to $X \times 10^{-12}$, it is constantly moving away from that point and must be constantly brought back by the controller. This is variation in frequency about the desired frequency.

This approach works, although I can't verify it because the Thunderbolt is the most accurate source that I possess. For my work, I consider the Thunderbolt to be able to hold something like $< 5 \times 10^{-12}$.

Phase-Locked Loops

The PLLs must very closely follow the reference and device under test sources for an accurate short-term measurement. In a PLL, the VCO output is constantly compared to the reference frequency and an error correction signal constantly controls the VCO. The error correction signal will have some noise and there will also be some error. If the PLL is locked, this only means the VCO phase stays in a given region relative to the reference. There is always some jitter or variation. The objective here is to keep that jitter very, very small. As discussed below in the Performance section, the floor of the test set is approximately less than 2×10^{-12}.

Long-term measurements are less critical because, although the PLL VCO may jitter about the input frequency, it is still locked and the longer-term phase difference is still valid even though the short-term phase difference may have significant error.

A/D Converter

The resolution of the A/D converter is very important. The A/D employed resolves 14 bits, thus allowing the least significant bit to be one part in 16384. Since the range of one slope of the quadrature phase detector output is 90°, one least significant bit represents 0.0055° of resolution. From Table 3, we see that 1 ppt results in a phase difference of 0.0036° / s. Thus, this A/D cannot resolve a one second error of 1 ppt. To improve this resolution, a number of readings are taken for each point and averaged, to yield the value used in the calculations. Resolution could be improved by employing a 16-bit A/D. Resolution is much more important than absolute accuracy, because the calculations are based on difference measurements.

Performance

As is sometimes said, the proof of the

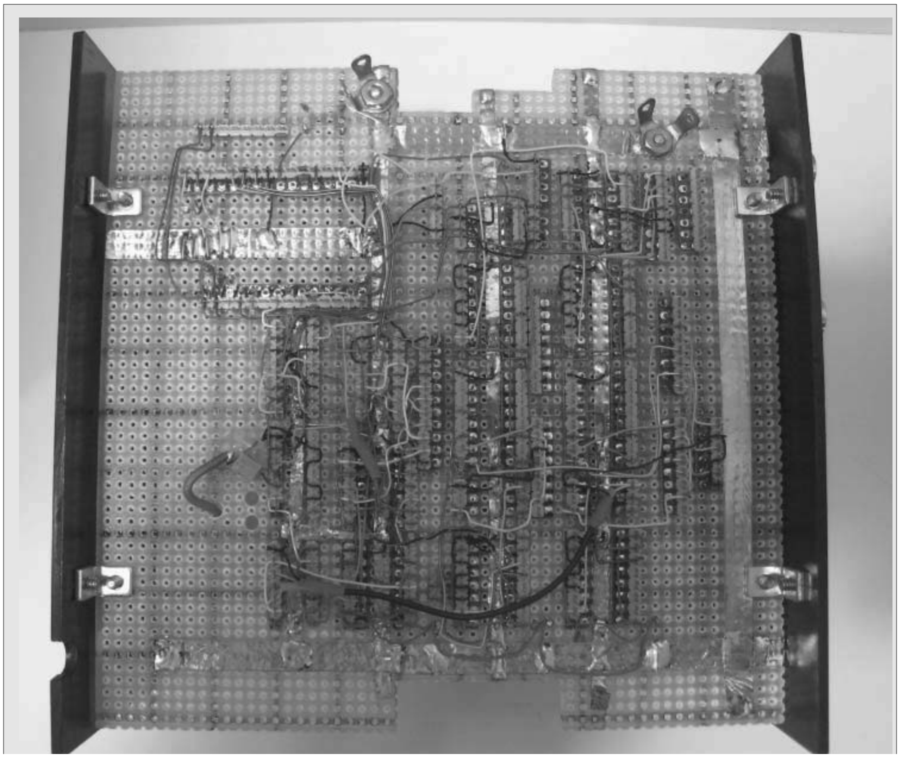


Figure 16 —Test set wiring view.

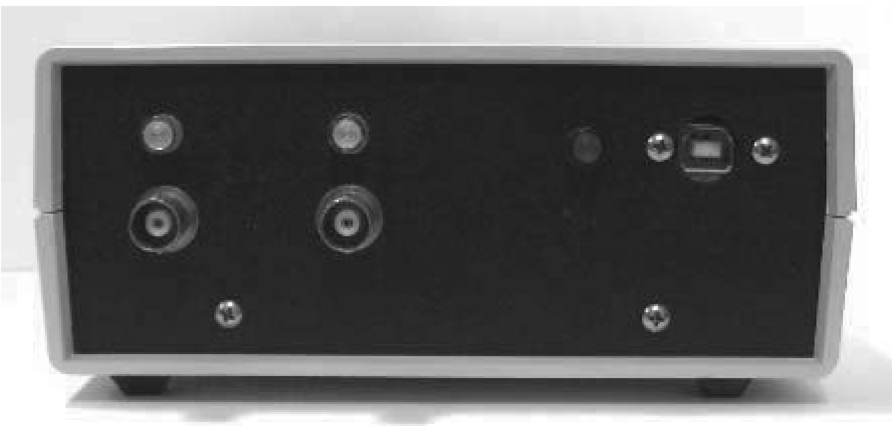


Figure 17 —Test set front panel view.

pudding is in the eating. All of the factors limiting the range of the test set come into play when both the reference and device under test inputs are connected to the same high-performance source and establish the noise floor of the test set. Figure 14 shows the result of feeding the test set with the same source on both the reference and device under test inputs. This means the two PLLs are operating at the same frequency independently and that the control dynamics of each crystal VCO PLL loop are independent. For the result shown, the two PLLs have to be operating independently with accuracies on the order of 1×10^{-12}. Otherwise, the phase difference between the two channels would be much greater. One factor that does not come into consideration here is that the two are operating from the same frequency and therefore are not moving relative to each other. I have observed in earlier layouts, some interaction between the two channels when the zero crossings of the two channels are in the same region in time. Notice in this case, that the long-term error is predominately positive and that it begins to settle out as time progresses, while the short-term error continues to vary. Again, I beg your indulgence in interpreting the presented data as it is difficult to discern the traces from the black and white image.

From this, it can be seen that the noise floor of the test set is approximately 2×10^{-12}.

Hardware and Software

The hardware is constructed using breadboard techniques and thru-hole DIP ICs, as shown in Figures 15, 16, and 17. The interconnections are made with insulation displacement wiring. The microcontroller is an Atmel ATMEGA328P in a custom module.

The microcontroller code is written in *C* for the Atmel ATMEGA 328P and the *Windows* application is written in *C* using National Instruments *Lab Windows/CVI*.

I created the schematic using *Eagle* software. Figures 18 and 19 show the schematic diagram.

Conclusions

The Test Set has met or exceeded my design goals. This project has given me a much deeper understanding of the performance and also the “care and feeding” of high accuracy frequency reference sources. I have found it interesting to discover just how these high accuracy sources vary with time. Visibility into the operation in real or near real time of a high accuracy standard has given me insight that allows more understanding. One point that has emerged related to the control circuits that allow external adjustment of the sources is that the control circuits need to be of high precision and stability as well.

I have included links to several technical papers about frequency source stability and analysis in the References section.

Questions remain, and in getting here ideas have come forth to improve the test and to include more analysis such as Allan Variance.

Acknowledgement

I would like to offer a special note of thanks to Ken Burch and especially to Brad Rehm, KV5V, for their in-depth reviews, patience and suggestions.

Jim Satterwhite, K4HJU, an ARRL Member, was first licensed in 1956 as KN4HJU. He holds an FCC General Radiotelephone Operator’s License, and is a registered Professional Engineer in the state of North Carolina. He is an iNarte Certified EMC Engineer and is a Life Senior Member of IEEE. He received a BEE degree from the University of Florida in 1965 and an MSEE degree from Purdue University in 1966. Jim has been in electronics development and research for the better part of 60 years, including 12 years as a Member of Technical Staff at Bell Labs and

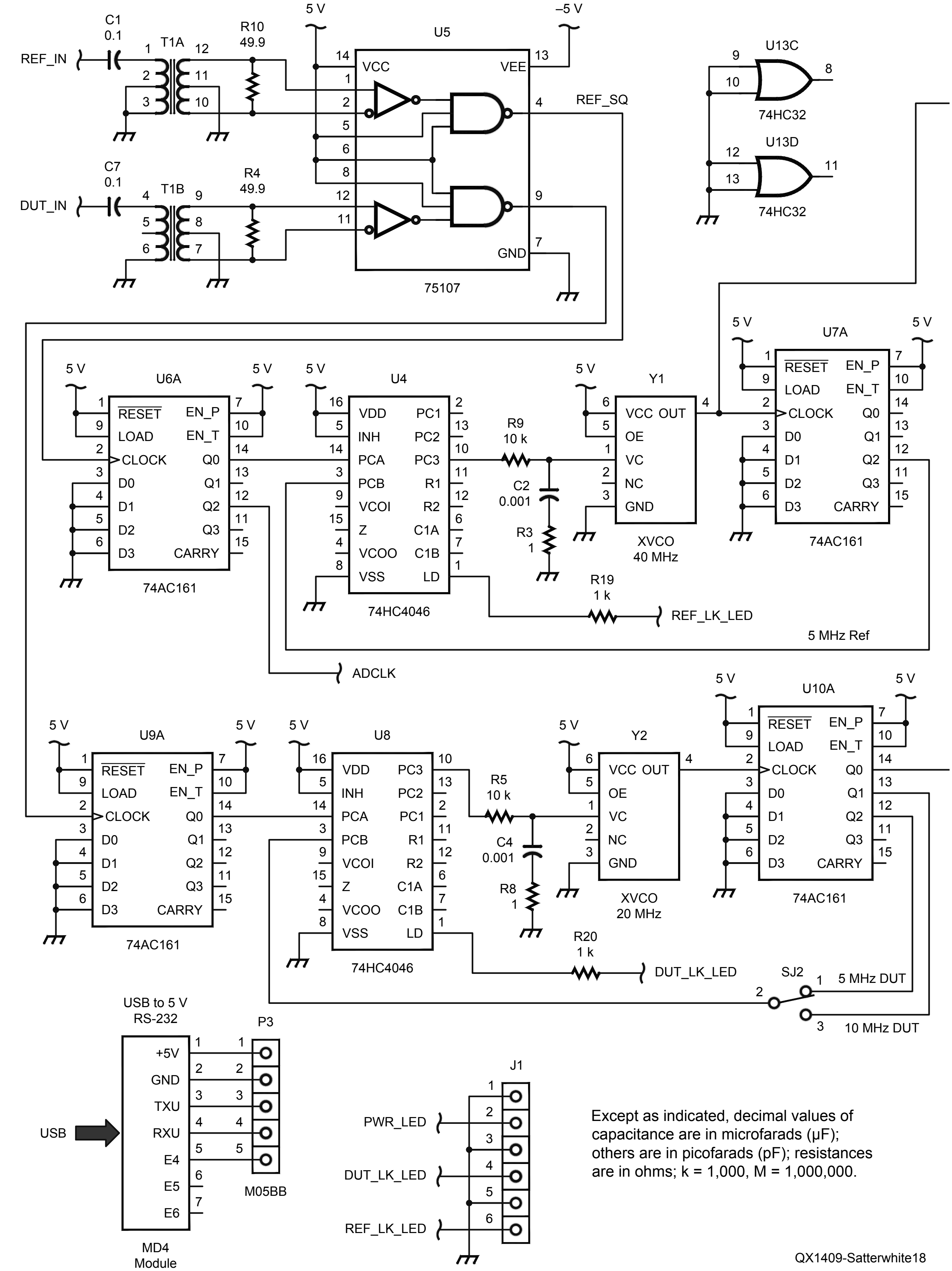
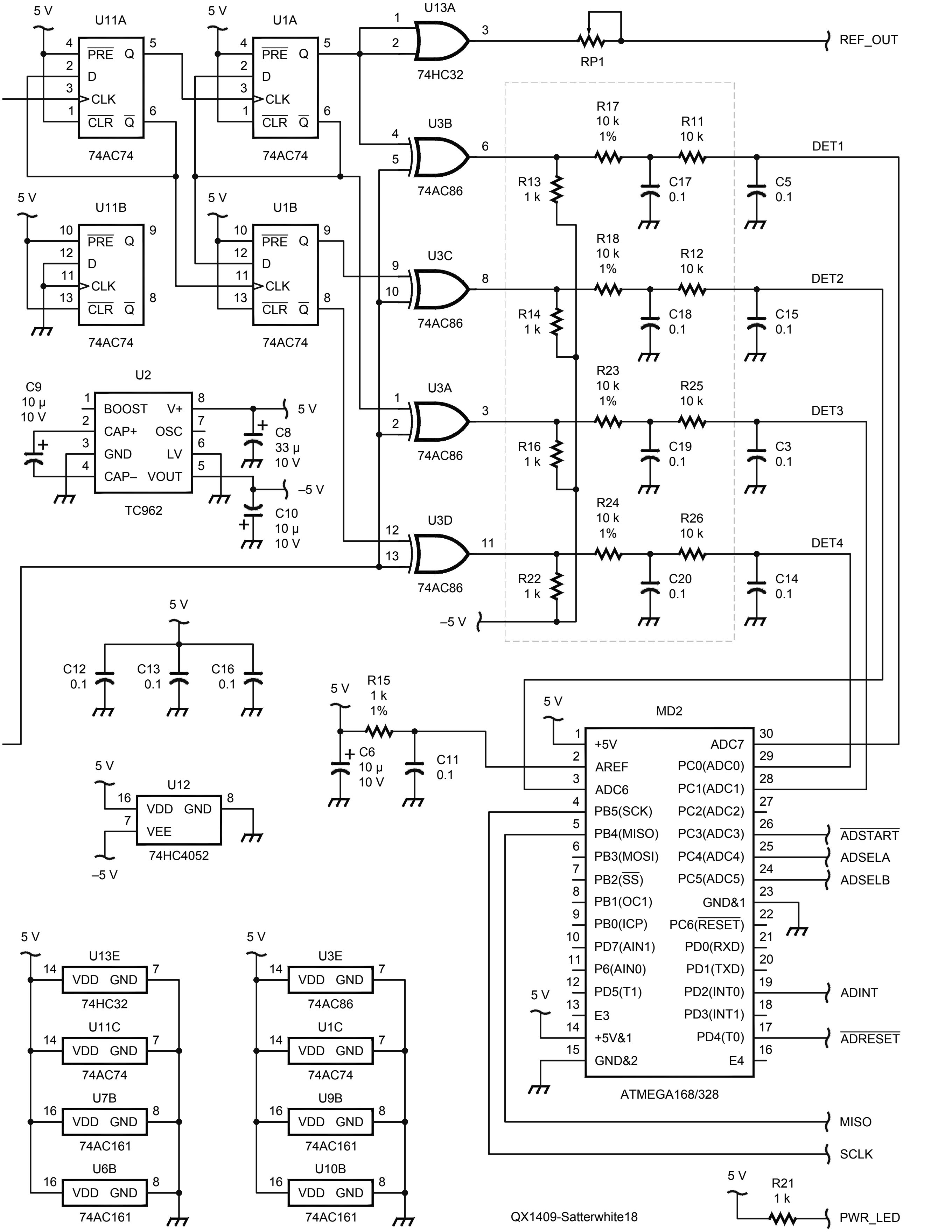


Figure 18 — Here is the schematic diagram of the test set, Part A.



QX1409-Satterwhite18

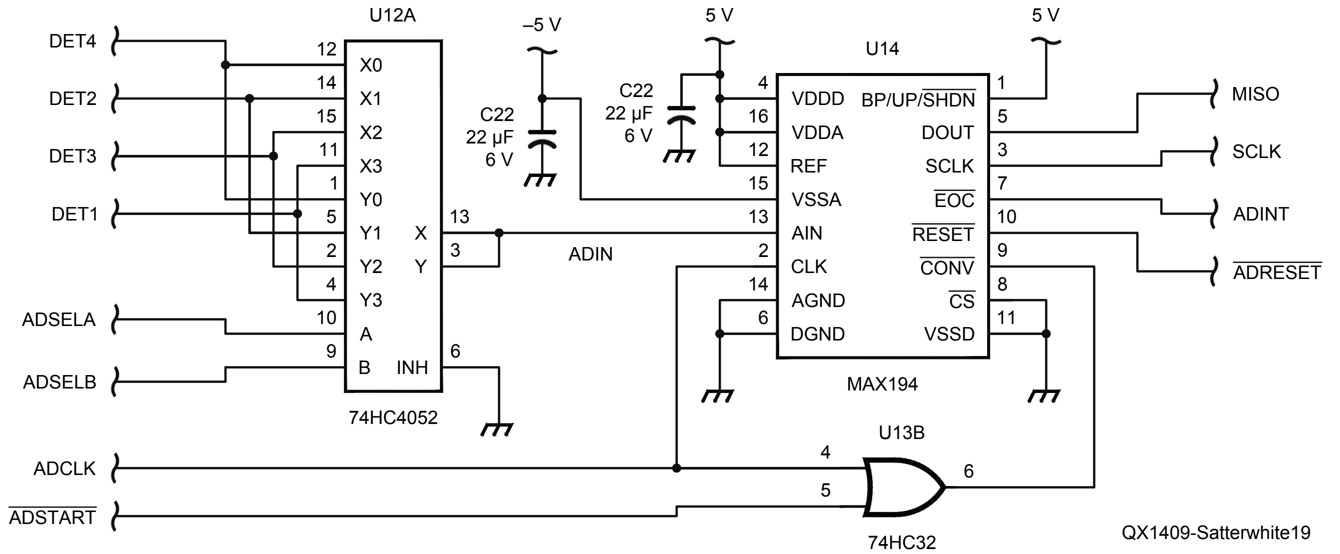


Figure 19 — This is the test set schematic, Part B.

32 years as a development engineer with Teltest Electronics, the company he founded in 1982. He holds a number of patents and patent applications. Jim enjoys developing electronic systems and is more comfortable with a track-ball or soldering iron than a microphone.

References

"A Note about Rubidium Oscillators", *Datum*, 12/10/1998, **www.amtestpl.com/DATUM/GmbH/Rubidium-Note-e.pdf**

D. B. Sullivan, D.W. Allan, D.A. Howe, and F. L. Walls, "Characterization of Clocks and Oscillators", *NIST Technical Note 1337*, (March 1990), **http://tf.nist.gov/timefreq/general/pdf/868.pdf**

D. A. Howe, D.W. Allan, and J.A. Barnes, "Properties of Signal Sources and Measurement Methods," *Proceedings of the 35th Symposium on Frequency Control*, **http://tf.nist.gov/timefreq/general/pdf/554.pdf**

W. J. Riley, *Handbook of Frequency Stability Analysis*, NIST Special Publication 1065, July, 2008, **http://tf.nist.gov/timefreq/general/pdf/2220.pdf**

David A. Howe, "Frequency Domain Stability Measurements: A Tutorial Introduction", *NBS Tech Note 679*, March, 1976, **www.adretelectronique.fr/anciens/photos/jacques_rutman/NBS_technical_note_679.pdf**

John Petrich, W7FU

1629 E. Lake Sammamish Pkwy NE, Sammamish, WA 98074; **petrich@u.washington.edu**

Tom McDermott, N5EG

3950 Southview Ter, Medford, OR 97504; **n5eg@arrl.net**

Digital Signal Processing (DSP) Projects: Examples of GNU Radio and GRC Functionality

After introducing us to GNU Radio Companion and leading us through the software installation in Part 1, the authors will lead us through several design examples and show us some of the things we can do with the software.

GNU Radio is the open source DSP software library that uses the GNU Radio Companion (GRC) graphical interface to implement DSP systems depicted as a visual flow graph on a computer screen. The examples of DSP projects begin with a visual orientation to the GNU Radio Companion graphical user interface (GUI) and detailed explanation of basic GNU Radio Companion terminology. Following the orientation to the GNU Radio Companion graphical user interface are illustrations of typical DSP flow graph applications that highlight some of the capabilities of GNU Radio Companion: a basic DSP that generates a 1 kHz tone internally and outputs the tone to the computer sound card, and more complex DSP back ends for FM and AM SDR receivers.

Example 1: Orientation to the GNU Radio Companion GUI

GNU Radio Companion opens on the computer screen with the graphical user interface depicted in Figure 1. The relatively empty center of the screen is the "active" work area. The active work area is the location where flow graphs are constructed and executed. The far right hand margin of the screen is the location of the DSP library which contains the DSP blocks. The library is organized into broad functional categories arranged vertically. Across the top of the screen is a menu bar that contains the buttons that reflect the available tools and operational commands for GNU Radio Companion. Of particular importance are the colorful "generate" button in the center of the menu bar, and the grayed out "execute" button just to its right. The generate button compiles the graphical flow graph into *Python* code and the execute button activates the DSP for signal processing. In the upper left hand corner of the active area is a small "Variable" block where the user can enter the DSP sample rate to reflect the desired digital bandwidth for the signal being processed.

To use this graphical user interface, you author a DSP flow graph in the active work area in a stepwise and logical manner. The first task is to enter the desired DSP sample rate variable for the bandwidth of the signal you want to process. The Variable block contains a variable named "samp_rate" that can be used within the DSP flow graph wherever a numeric value might be entered (rather than hard coding the sample rate in all block locations). Other variables can be defined and even linked to additional GUI blocks, GUI sliders or GUI choosers, which the user can change from the GUI while the DSP is operational. The next step is to place the desired DSP block from the right hand library into the active work area. A double click on the desired DSP library block is all that is needed to cause the desired block to automatically appear in the active work area. The DSP block can be moved within the active work area with a simple drag and drop technique. To open the parameter menu for any block in the active area, the user simply double clicks on the block.

Once double clicked, the parameter menu for that block appears in the center of the screen. With the parameter menu open, the user must enter the "Type" (data stream format — complex, float, or integer), and other desired block parameters into the appropriate spaces in the menu. Clicking "ok" or <Enter> closes that block parameter menu and allows the user to continue authoring the DSP flow diagram. The process of moving desired library blocks into the active work area, followed by entering the needed numeric values or variable names into the block parameter menu is repeated until all the needed blocks for the desired DSP flow graph are placed and configured within the active area.

When all the necessary blocks are placed and configured in the work space, the block ports must be connected in a series fashion with arrow links. Check to make sure there are no errors in your intended flow diagram. GNU Radio Companion has an automatic error message system that identifies errors in real time. If there is an error in the block parameters, or a missing input or output connection, GNU Radio Companion colors

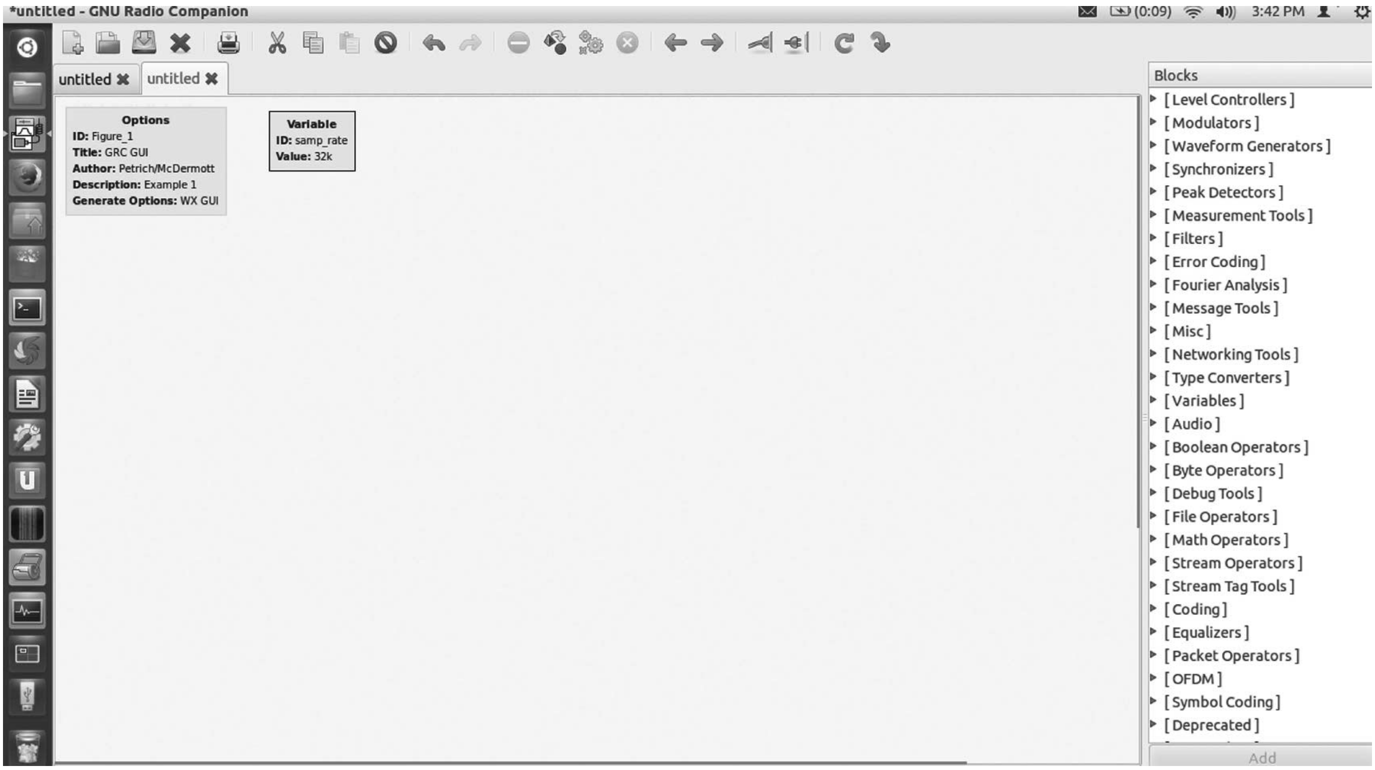


Figure 1 — This screen shot is the opening screen of the GNU Radio Companion graphical user interface.

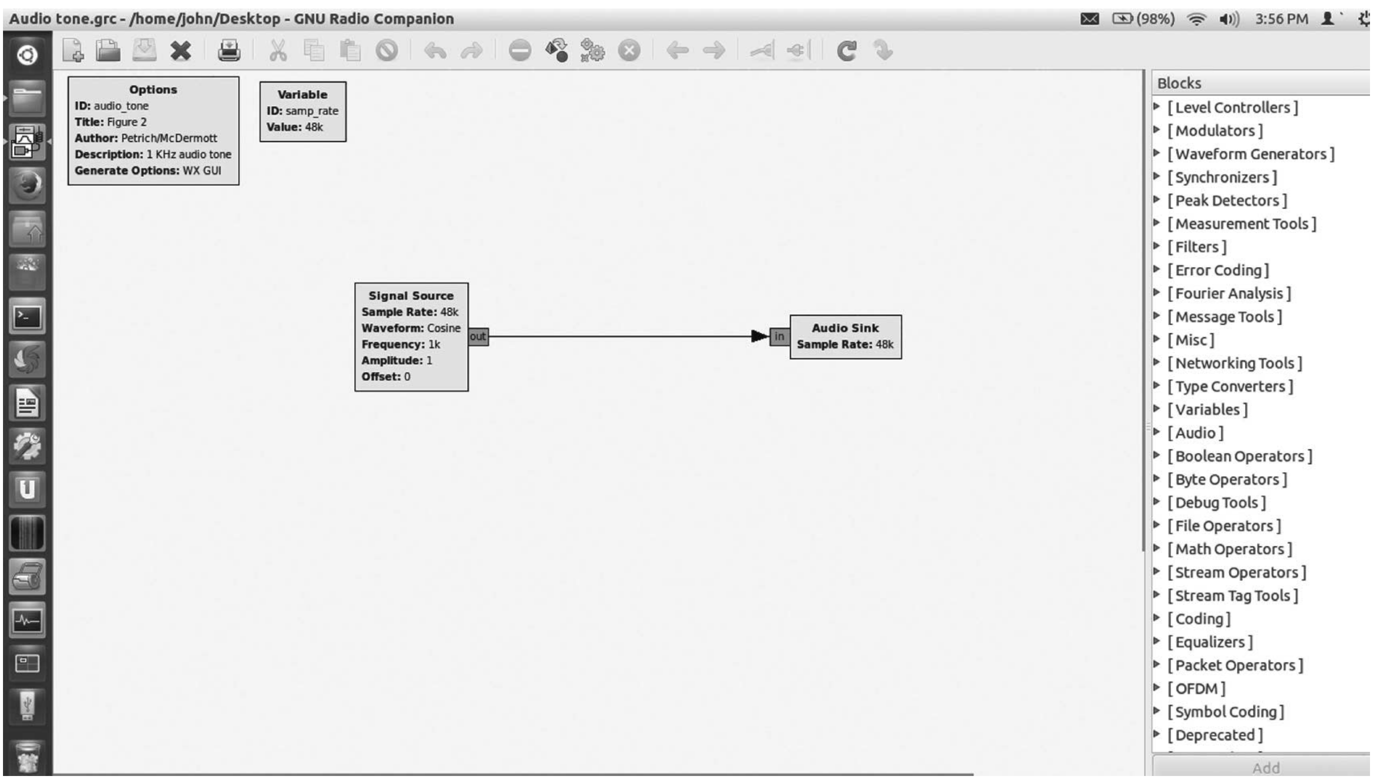


Figure 2 — This GNU Radio Companion screen shot shows a basic Source and Sink flow graph.

the title of the block red. Blocks can also write error messages to the GNU Radio Companion status window located in a panel below the active work area. Once errors are eliminated, compile the underlying code using the "Generate" button from the menu bar at the top of the screen. To make the DSP flow graph operational, the "Execute" button to the right of the generate button executes the flow graph. An excellent video series by Balint Seeber depicts the GUI and the steps necessary to create functional DSP flow graphs.[1]

Example 2: A Basic "Source" and "Sink" Flow Graph

Begin using GNU Radio Companion with a simple DSP project such as an audio waveform generator driving a speaker. Figure 2 illustrates such a DSP flow graph. The exact same GNU Radio Companion graphical user interface is rendered as was shown in Figure 1, but now, the work area contains the DSP blocks necessary to generate and output the tone. The sample rate in the Variable box at the upper left of the active area was set to 48000 Hz to match the sample rate of the sound card in the test computer. The Waveform generator is found in the intuitively named "Waveform Generators" category in the DSP library. This will create the Signal Source block in our flow graph.[2] You can find the Audio Sink block in the "Audio" category of the DSP library, and it is connected internally to the computer sound card.[3]

The source parameters are set for a "float" digital format "Type," a cosine waveform, a frequency of 1 kHz and a sample rate to the same value as is set in the sample rate "Variable" box, 48 kHz. The parameters for the "Audio Sink" are set complementary to the source parameters in this example. The DSP is activated with the execute button in the menu bar. If you are following along with this example on your computer, you will hear a 1 kHz tone from your computer speaker. You can adjust the output volume using your computer sound card volume control.

Example 3: A VHF FM Receiver DSP Using a TV Dongle Front End

Figure 3 differs from the previous figures, and only shows the active work area of GNU Radio Companion, without the library on the right side of the screen. This example presents a more complex DSP flow graph: the DSP and control panel component blocks of an FM receiver that uses the inexpensive TV Dongle as the SDR front end.

This DSP flow graph demodulates the received signals and adds provisions for a visual display of the RF and IF passband, as well as a receiver control panel. These features are visible when the flow graph is executed and the receiver is operational.

The flow graph was constructed using the process described in the previous example. The TV Dongle block is found in the "Sources" library category and named "RTL-SDR" block. The parameter menu is set to tune the receiver to the FM broadcast band. Note the frequency parameters visible in the RTL-SDR Source block: Ch0 Frequency = 88.5 MHz. The RF and IF gain parameters for the RTL-SDR Source block are set by the user for optimum performance.

Following the digital stream flow graph from left to right, the digital output from the source goes through a "Low Pass Filter" block, and then the information from the signal is extracted using a wide band FM demodulator ("WBFM Receive" block). The demodulated digital information continues through another "Low Pass Filter" block and a "Rational Resampler" block before going to a volume control ("Multiply Constant" block) and finally to an "Audio Sink" that completes the audio chain.

Fast Fourier Transform (FFT) display blocks are positioned before and after the

[1]Notes appear on page 30

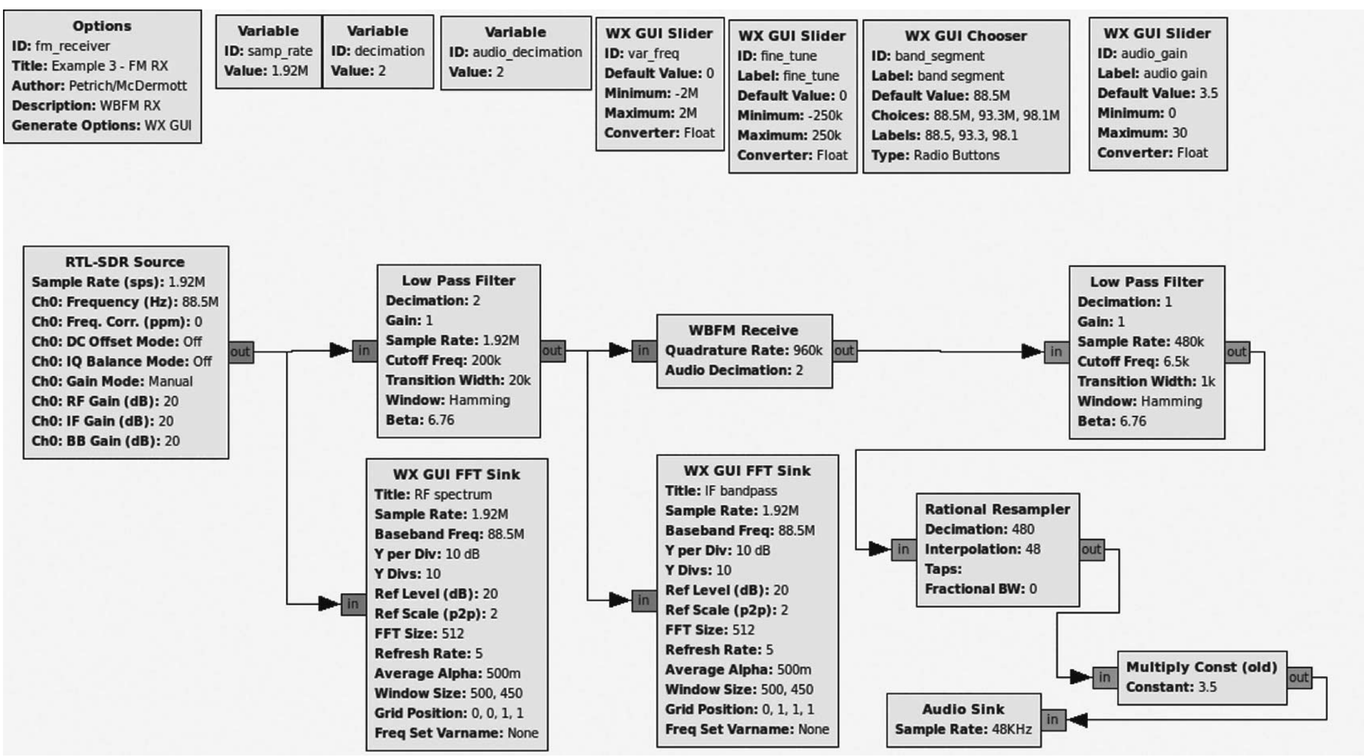


Figure 3 — Here is the DSP flow graph of a simple VHF FM Receiver.

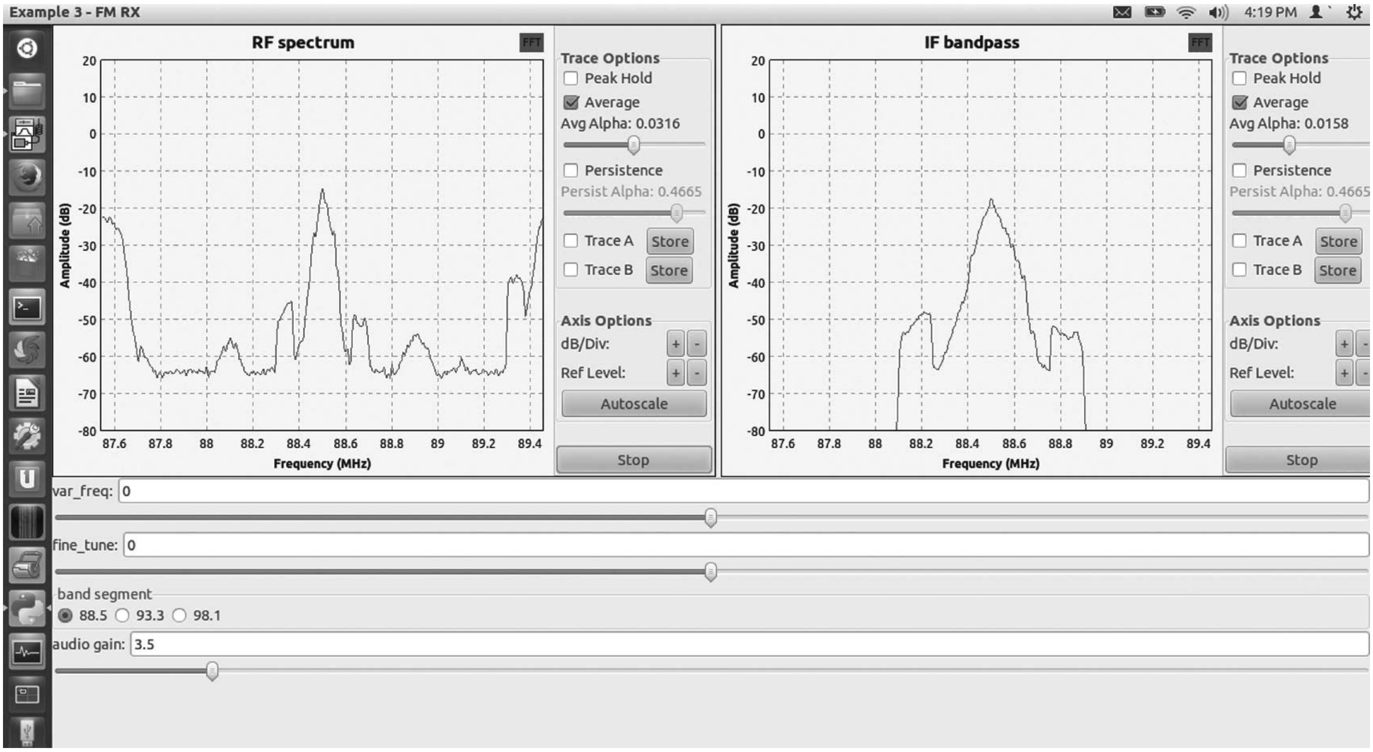


Figure 4 — This screen shot is of the Display and Control Panel for the VHF FM receiver while it is operating at 88.5 MHz.

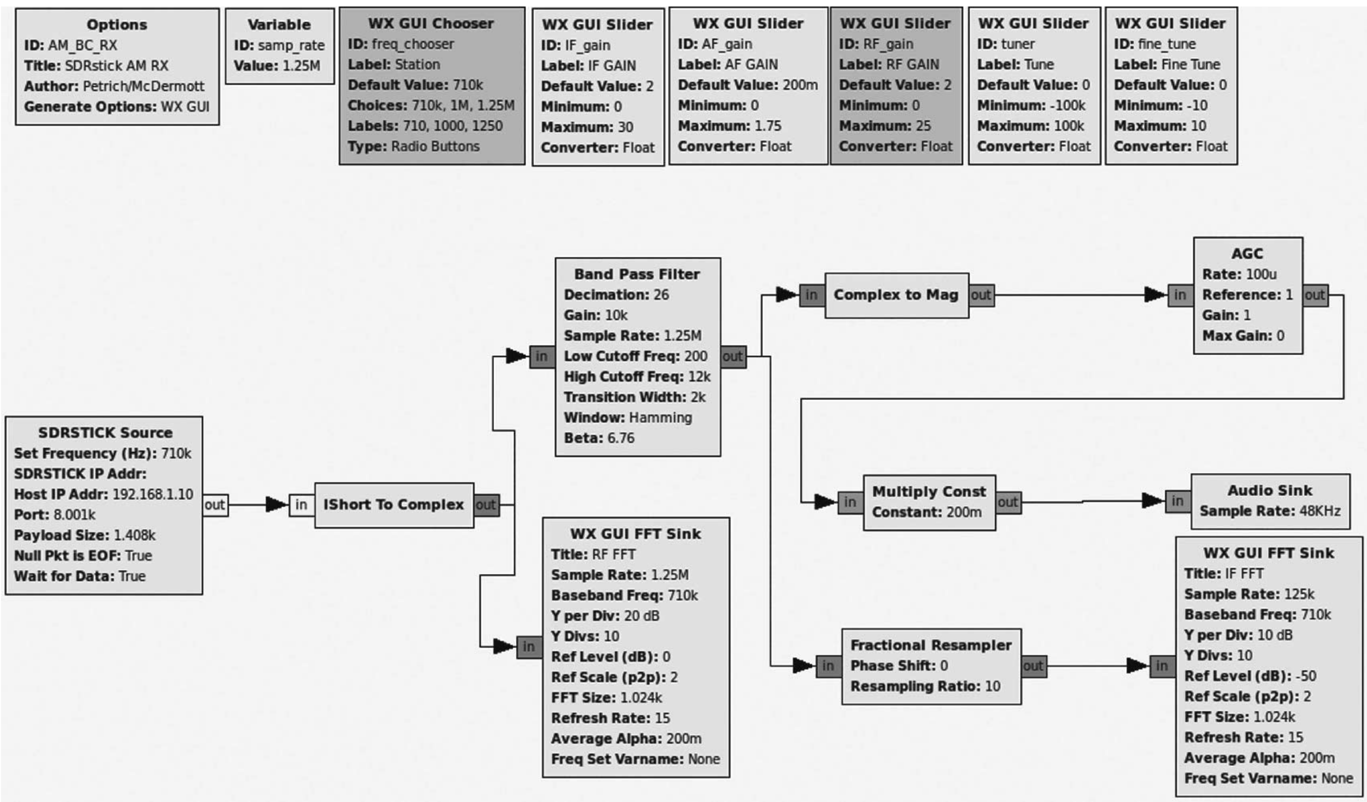


Figure 5 — Here is a screen shot of the DSP flow graph of an AM Broadcast Band receiver using the SDRstick™ HF1 SDR hardware.

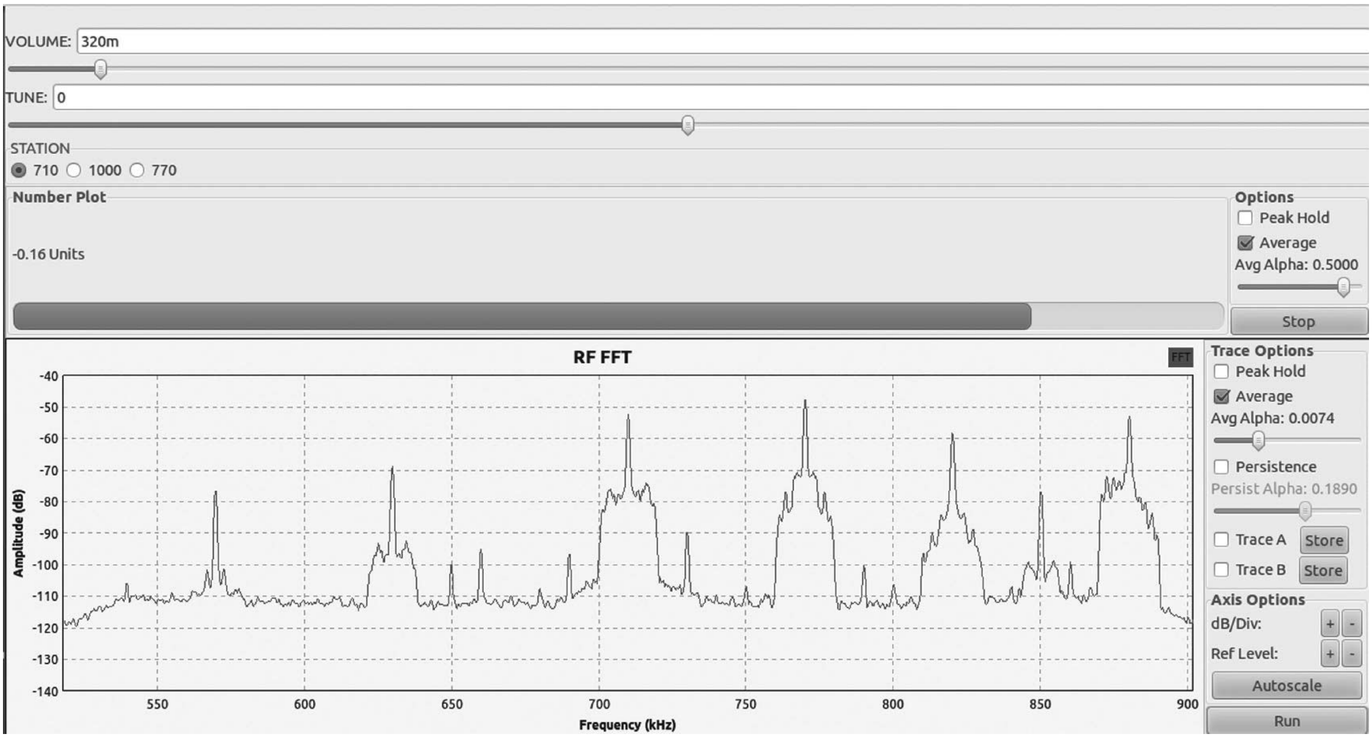


Figure 6 — Here is the RF FFT Display and Control panel for an SDRstick™ HF1 AM Broadcast Band receiver while it is operating at 710 kHz.

first low pass filter to display the RF and IF passbands. Upon executing the DSP flow graph, the FFT and control screen shown in Figure 4 is displayed, overlaying the flow graph.

Figure 4 shows the receiver control panel and both the RF FFT and IF FFT display for the received FM. Looking at the RF FFT display on the left, the main audio channel for the selected station is located in the center of the FFT display. The transmitted information sidebands, visible to the right and left of the main audio channel, contain the stereo and radio data system (RDS) information that is broadcast by the station. Adjacent FM stations are partially visible on the extreme margins of the display. The IF FFT display is on the right. The FM signal is centered in the IF passband.

The sliders and buttons below the FFT displays are user configured controls that permit changing the received frequency and adjustment of the audio volume. Imagine viewing the FFT display, tuning the receiver, and simultaneously hearing the demodulated audio in your headphones or speaker using a custom DSP that you authored!

Example 4: An HF AM Receiver DSP Using the SDRstick™ HF-1 Front End

Figure 5 depicts the flow graph in the active work area for the final example: a DSP authored to receive and demodulate AM broadcast band signals using the SDRstick™ HF1 as the SDR front end.[4] The SDRstick™ HF1 is a recently available HF SDR front end that was designed for use with a GNU Radio Companion DSP back end. The arrangement of this flow graph is similar to the arrangement of the flow graph for the FM receiver. In this flow graph, the AM demodulator, "Complex to Mag" block, replaces the FM demodulator, along with adding refinements such as AGC and a custom calibrated IF FFT display. Note the numeric variables visible in the "Band Pass Filter" block, which define the bandwidth, stop band, and filter shape. Variables are visible in other blocks as well. The digital stream "Type" is also different than Example 3. In this example the "Type" is "Interleaved Shorts" (IShort), which is the digital stream "Type" from the SDRstick™ source block to the remainder of the DSP. This digital stream Type is changed from Interleaved Shorts to complex using the "IShort To Complex" block.

Figure 6 illustrates the RF spectra of an AM broadcast station centered in the display at 710 KHz. Adjacent AM Broadcast Band signals are visible at lower and higher frequencies. The GUI sliders permit control of audio volume and the receiver tuning frequency as in the previous example of the FM receiver.

Conclusion

Flow graph authoring is easy but not totally intuitive. Developing DSP projects with GNU Radio Companion flow graphs leverages the user's foundational knowledge of signal flow and demodulation theory. GNU Radio Companion is a tool that builds on that knowledge and allows the user to easily implement practical DSP systems. To really learn digital signal processing, you must study DSP concepts and have a reasonable understanding of complex numbers.[5, 6, 7, 8] GNU Radio with GNU Radio Companion makes it easier to learn DSP by allowing you to focus on blocks rather than arcane source code and to develop functional DSP systems with a hands on approach.

More complex DSP flow graphs can be constructed to implement full featured communications oriented DSP systems, receivers and transmitters, for every imaginable modulation type and frequency. The potential is constrained only by the SDR front end hardware and the author's imagination.

There is a lot to learn and we urge you to embrace the journey, because it is very rewarding!

Note: One of the inevitable consequences of writing a "How To" article about dynamic and rapidly changing Open Source software is that changes to the software and the interface often come in rapid succession. Since the publication of Part 1 of this article in the Jul/Aug 2014 issue of *QEX*, there have been at least five version revisions to GNU Radio Companion, and some aspects of GRC have changed. In an effort to provide the latest information and installation instructions for GNU Radio and GNU Radio Corporation, the authors have created a website. Go to **www.w7fu.com** for the latest information.

ARRL member, Official Observer, and Amateur Extra class licensee John Petrich, W7FU, was first licensed as K6OJV in 1955 and then as W7HQJ after moving to Seattle. He is a practicing physician, and Clinical Associate Professor of Psychiatry, School of Medicine, University of Washington. John is active in community affairs, enjoys family life, sea kayaking and cycling. John's radio experience parallels the evolution of radio communications technology over the past century. He started with a homebrew crystal receiver followed by a much loved and modified single tube regenerative receiver in a cardboard box. Upon earning his license, he graduated to operating QRP using a crystal controlled 6V6 tube transmitter constructed on a wooden chassis. Subsequently his rigs evolved from modified surplus gear to home constructed to full featured analog receivers and transmitters. John's first love is CW, the prototypical digital QRP mode. He credits radio with endless opportunities for engaging learning opportunities and long lasting friendships. At present John's rig is an experimenter's station built around the HPSDR Atlas bus system. The station is supplemented with back-up rigs using Ettus and SDRstick™ SDR "front ends" and DSP "back ends" built using GNU Radio Companion. John is interested in communicating with others who have similar interests.

ARRL Life Member, and Amateur Extra class licensee Tom McDermott, N5EG, has been licensed 45 years. He is a member of TAPR, IEEE, and Internet2, and has been involved in the development of fiber optic transmission and switching systems since the initial deployment of single-mode fiber in positions ranging from ASIC designer to CTO. He currently is a participant in the IEEE 802.3 Ethernet 100GE and 400GE standards projects. Tom has a BSEE degree from the University of California, Berkeley, and has written one textbook on wireless digital communications. He's been involved in many computer-related ham projects, from the TEXNET layer 3 packet radio system, to a VNA project, network simulation, and other efforts. His current interest is using a HPSDR Hermes SDR transceiver and GNU Radio to experiment with DSP algorithms.

Notes

[1]Balint Seeber has posted a YouTube video tutorial about how to set up some DSP functions: **www.youtube.com/watch?v=N9SLAnGlGQs**.

[2]A "source block" is defined as having only output ports and is the origin of the digital stream for the DSP.

[3]The term "sink block" is reserved for DSP blocks that only have input ports. The digital stream does not extend out of the sink, only into the sink. An arrow connects the output port of the source to the input port of the sink.

[4]You can learn more about the SDRstick™ HF1 on the SDRstick website: **http://sdrstick.com/**. Scotty Cowling, WA2DFI, also described the SDRstick HF1 and HF2 boards in his article "Hardware Building Blocks for High Performance Software Defined Radios," in the Jul/Aug 2014 issue of *QEX*.

[5]There is a free DSP textbook available on the Internet: *The Scientist's and Engineer's Guide to Digital Signal Processing* by Dr. Steven W. Smith. **www.dspguide.com/**.

[6]*The ARRL Handbook For Radio Communications*, includes a chapter on "DSP and Software Radio Design. H. Ward Silver, NØAX, *Ed*, *The ARRL Handbook For Radio Communications*, ARRL, 2013, ISBN: 978-1-62595-001-7; ARRL Publication Order No. 0007, $49.95. ARRL publications are available from your local ARRL dealer or from the ARRL Bookstore. Telephone toll free in the US: 888-277-5289, or call 860-594-0355, fax 860-594-0303; pubsales@arrl.org. **www.arrl.org/shop/2014-Handbook-Softcover-Centennial-Edition/**.

[7]Wes Hayward, W7ZOI, Rick Campbell, KK7B, Bob Larkin, W7PUA, *Experimental Methods in RF Design*, ARRL, 2003, ISBN: 978-087259-923-9; ARRL Publication Order No. 9239, $49.95. **www.arrl.org/shop/Experimental-Methods-in-RF-Design/**.

[8]Doug Smith, KF6DX, *Digital Signal Processing Technology*, ARRL, 2001, ISBN: 978-0-87259-819-5; ARRL Order No. 8195, $34.95. **www.arrl.org/shop/Digital-Signal-Processing-Technology/**.

Scotty Cowling, WA2DFI

PO Box 26843, Tempe, AZ 85285; **scotty@tonks.com**

Hands-On-SDR

You've been hearing about it, you've even read about DSP techniques, now it's time to actually put your hands on some hardware and install some software!

I often find that the fascination with something that works comes first, and the understanding of exactly how it works comes later. "I can actually pull music from out of the air with that little glass bead! How does that crystal set work, anyhow?"

In this new *QEX* column, I will show you how to combine off-the-shelf hardware and software components to make a working software defined radio (SDR). Our focus won't be as much on *how it works* as *how do I make it work*. The digital signal processing (DSP) heavy lifting, complex mathematical functions and data transport (the Ethernet interface, for example) have already been implemented by others. We will hook up existing hardware and software blocks in order to build a working SDR. While I won't delve into the internal workings of each block (at least not *too* much), I will show you how to get on the air with as little pain and confusion as possible.

The main focus will be on the software, or user interface (UI). The UI acts as a "virtual" front panel (in other words, generated by software) of our SDR. We will control the operation of the radio from this virtual front panel. The UI that we will be using is called *HDSDR*, short for ***H****igh* ***D****efinition* ***S****oftware* ***D****efined* ***R****adio*. Even though we will focus on the software, we still need *some* hardware to pull signals from the ether.

In this first installment of the column, I will describe how to set up and use three different sets of commonly available hardware with the *HDSDR* software. You will want to select one of these radios to follow along.

The first option I will describe uses the SDRstick™ HF1 boards that I described in the July/August 2014 issue of *QEX*.[1, 2] The second option will be to set up an SDR-IQ™ receiver from RFSpace.[3] Last, we will install and configure a Softrock RX Ensemble II receiver, available from Five Dash Inc.[4] The end result of each lash-up will be a working SDR receiver.

You might ask, "Why set up three different radios, when I only have two ears?" There are several reasons. Each of these receivers connects to the computer in a different way (Ethernet, USB, or PC sound card); each has its own capabilities (display bandwidth, frequency range); and the price range is wide (from a $67 kit to a $525 assembled unit). By showing different connection methods, you will be more prepared to hook up *any* SDR than if we just showed one example. Taken together, these three radios are representative of the majority of SDRs that you are likely to encounter in the amateur marketplace. By looking at different capabilities, you will become familiar with the features of more than one SDR and learn the relative importance of these features. Lastly, you are more likely to have access to the hardware if we show three examples; if SDR hardware is not in your budget right now, you might be able to borrow some from one of your buddies in the local club. Beware! This is a fun part of amateur radio. After playing with your SDR Elmer's borrowed hardware, you might just have to go out and get your own!

Is this for me?

With the limited space available within each column, I must assume some technical knowledge on the reader's part. In general, what do you, as the reader, need to know? What equipment do you need to have available? Keep in mind that many of the procedures are the same or similar for all SDRs. Even if you do not have the hardware, you may pick up some tricks by following along with the text. Note that things like programming knowledge, advanced math skills and RF design experience are conspicuously absent. We are *assembling* an SDR, not *designing* one.

A "Good" *Windows* PC

While the Marines need a few good men, we need one good PC. If you are a *Linux* or a MAC guy, you can still play with SDRs, but the experiments here require a *Windows* PC, running XP, 7, 8 or later. What is a "good" PC? I am not trying to be evasive, but the answer is, "it depends." Generally any mid-grade PC built within the last few years will do, but here are some guidelines:

- At least two cores at 2.0 GHz or faster.
- 8 GB of memory.
- At least 10 GB of free disk space.
- Gigabit Ethernet port (minimum 100 Mb Ethernet is required).
- Minimum display resolution of 1024 × 768 (the bigger the better).
- One free USB 2.0 port.
- Sound system with line input and speakers.

The ultimate test of a computer's suitability is simply to try it. Just be prepared to upgrade if you are using an old workhorse PC; it is likely not up to the task.

[1]Notes appear on page 37.

Operating System Knowledge

I assume that you know your way around the operating system, and can perform the following tasks, as a minimum. There are many on-line references to help you if you need to learn or just to brush up on the finer points of your particular operating system.

- Install applications.
- Check system information and update hardware drivers.
- Enable/disable both wired and wireless network ports.
- Change PC IP address and network mask.
- Enable and disable PC DHCP client.
- Use *Windows* Command Shell to check IP addresses and ping network devices.
- Check/set environment variables using the system properties dialog box.
- Navigate the PC file system to copy, move and unzip files.

You will need a signal source. This could be a resonant amateur band antenna or non-resonant wire (to gather interesting signals) or a signal generator (not so interesting, but good for testing). Attenuators and high-pass filters are helpful too, especially if you live next door to a high-powered MW broadcast station. (More on this later.) We will list the specific hardware you need for each project before we show you how to put it all together to make an SDR.

Today's Project: Let's Make a Receiver!

The SDRs we will put together today are receivers that cover some or all of the medium wave band (300 kHz to 3 MHz) and all of the shortwave band (3 MHz to 30 MHz). We will use *HDSDR* software for our user interface for each receiver, and connect each SDR to the PC in a different way. In spite of the simplicity of some of the hardware, all of these receivers perform well.

What You Need: Hardware and Software

We will work with three different receiver hardware configurations. We will show you how to set up the software for all three of these and become an expert at assembling different SDR systems. When you run the *HDSDR* software, you will select the hardware that you want to use. Note that the three configurations were not chosen at random; each setup connects to the PC in a different way. The three different connection methods are the three most commonly used by modern SDRs. If you can make these three different setups work, you can hook up just about any SDR! See table 1 for a quick comparison of the three receivers we will set up.

In addition to your *Windows* PC, here is the hardware you will need for the SDRstick™ receiver:

- BeMicroSDK development board and a UDPSDR-HF1 receiver board (Figure 1).
- 5 V DC, 500 mA power supply.
- Short Cat5 Ethernet cable.
- SMA adapter to your antenna feed.

Here is the hardware you will need for the SDR-IQ™ receiver:

- RFSpace SDR-IQ™ receiver (Figure 2).
- USB type A to type B cable (typical USB printer cable).
- BNC adapter to your antenna feed.

Here is the hardware you will need for the Softrock RX Ensemble II receiver:

- Softrock RX Ensemble II receiver (Figure 3).
- USB type A to type B cable (typical USB printer cable).
- Audio patch cord, 3.5 mm stereo male-to-male.
- 12 V DC, 500 mA power supply.
- Enclosure (optional).
- BNC adapter to your antenna feed.

There are six pieces of software you need to make these receivers operate with *HDSDR*. We will cover software downloading and installation in the next section.

- *HDSDR* application for all receivers.
- FTDI USB driver for SDR-IQ™.
- LibUSB driver for Softrock receiver.
- ExtIO_SDRSTICK.dll file for SDRstick™ receiver.
- ExtIO_SDRIQ.dll file for SDR-IQ™ receiver.
- ExtIO_Si570.dll file for Softrock receiver.

Software Download and Installation

Installing the software is by far the most complicated and time-consuming task of the project. Here we will show you where to get each piece of the software and show you some tips for download and installation that will save you some time.

For all receivers, we need the HDSDR User Interface

Download the *HDSDR* installer by going to **www.hdsdr.de** and clicking the **Download** button at the bottom of the page.

Table 1
Feature Comparison of the Three SDR Receivers

	SDRstick™ HF1	*SDR-IQ™*	*Softrock RX Ensemble II*
Receiver type	DDC	DDC	QSD mixer
PC Connectivity	Ethernet	USB	Soundcard + USB
Freq Coverage	100 kHz to 30 MHz	100 Hz to 30 MHz	1.8 MHz to 30 MHz
Display Bandwidth	1.25 MHz	190 kHz	96 kHz
A/D Converter	14 bit @ 80 Msps	14 bit @ 67 Msps	soundcard
Case	no	included	$20 option
Power	USB or external 5V	USB only	external 12V
Cost	$243 assembled	$525 assembled	$67 kit/$92 assembled

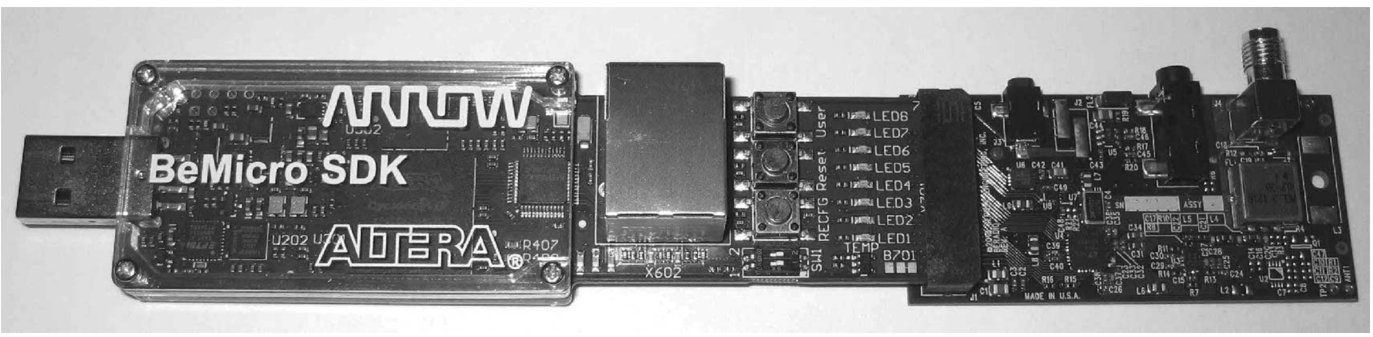


Figure 1 — The BeMicroSDK data engine together with the HF1 RF front-end board.

Figure 2 — The SDR-IQ receiver from RFSpace.

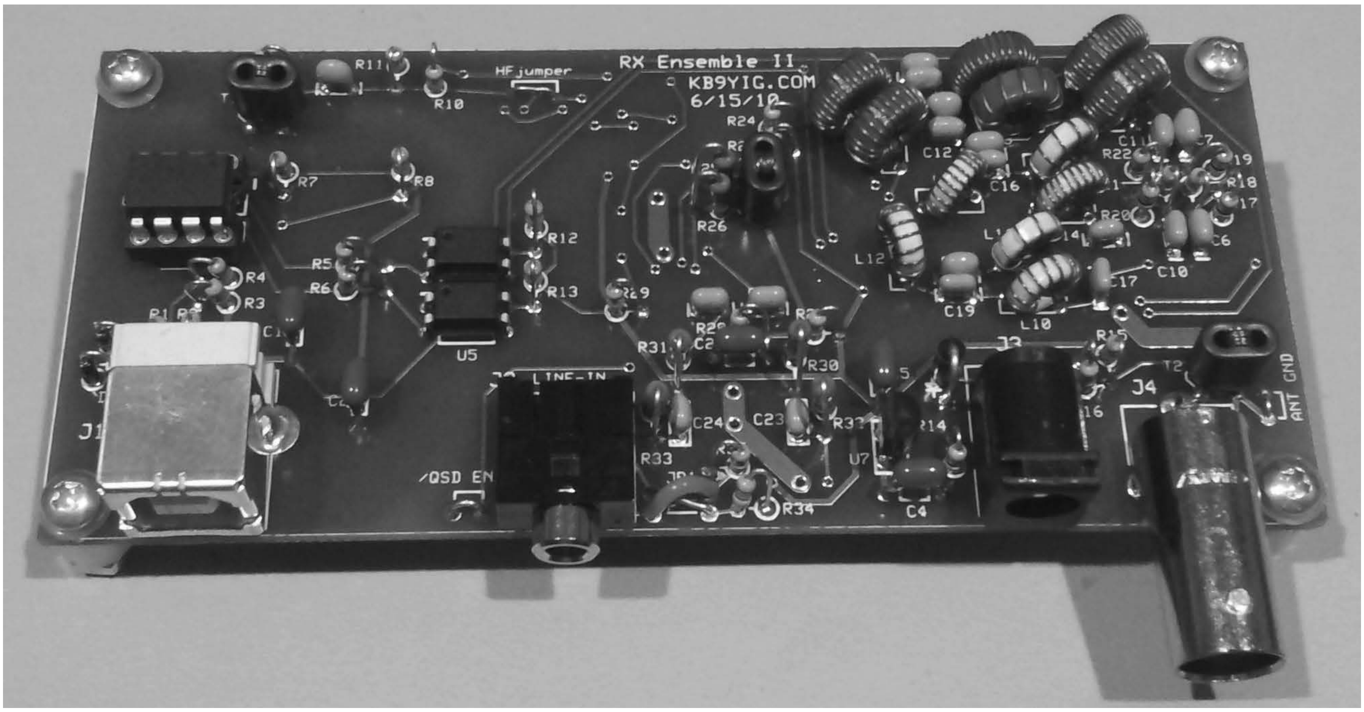


Figure 3 — The Softrock RX Ensemble II Receiver from Five Dash Inc.

Save the software to a folder of your choice. Next, run the installer by using *Windows Explorer* to navigate to the folder where you saved the program, and double-click on **HDSDR_install.exe**. You will have to accept the license agreement (click on the "I accept the agreement" button). You will also have to select the Destination Location, the Start Menu Folder and the icons that you want. (I suggest that you accept the defaults on all of these choices.) You will then be shown a summary of all of the selections that you have made. At this step, write down the Destination Location, because you will need to know it later.

Finally, click the **Install** button to install the *HDSDR* user interface software. The final dialog box gives you the option to Launch *HDSDR*. Uncheck this box, since we have a few more things to do before we are ready to run the software. Click the **Finish** button to complete the installation. We will use this software to operate all three radios, but we are not quite there yet.

In order for *HDSDR* to recognize and "talk" to our receiver hardware, we need one more file. This file is generically called an ExtIO.dll file, and we need a different one for each kind of hardware that we want to use. This means that there are three of these ExtIO.dll files that we will need, one for each receiver. (Take a look at the list of software, above, for the three file names.) Think of the ExtIO.dll files as "translators" that sit between the *HDSDR* software and our hardware. The "translators" convert the data from our hardware (in three different formats, a different one for each receiver) into the single native format that *HDSDR* understands.

For the SDRstick™ HF1 receiver, download the file **ExtIO_SDRSTICK.dll** from **svn.sdrstick.com** in the **/sdrstick-release/software/extio** section. Place this file in the *HDSDR* install directory. (You did write down the Destination Location during the install, didn't you? Just in case you didn't, the default destination location for *Windows*

XP is **C:\Program Files\HDSDR**, and for *Windows* 7 it is **C:\Program Files (x86)\HDSDR**).

For the SDR-IQ™ receiver, download the USB driver from **rfspace.com/support_files/SDR14IQUSBdriver.zip** and save it to a folder of your choice. Unzip the archive to either the same folder or a subfolder. Note the location of this folder, because we will need it to install the USB driver when we first plug in the SDR-IQ™ receiver. Next, download the zip archive from **www.hdsdr.de/download/ExtIO/ExtIO_SDRIQ14_Bonito.zip** and save it to a folder of your choice. The archive contains two files; unzip it and copy the **ExtIO_SDRIQ.dll** file into the *HDSDR* install directory.

The Softrock procedure is a bit different. We need the **LibUSB** driver to talk to the Softrock USB port. Download the driver installer from **pe0fko.nl/SR-V9-Si570/PE0FKO-USB-Driver-installer.exe**. Save this installer to a folder of your choice and run it. The Driver Setup wizard will guide you through the steps necessary to install the **LibUSB** driver. Use the defaults for Destination Location and Start Menu Folder and click **Next** after each selection. If you are prompted to install the certificate, click **Install Cert**. Click **Install** to run the Device Driver wizard, then **Next** and **Finish** to complete the **LibUSB** driver installation. Finally, click **Finish** to exit the Driver Setup wizard. If it seems like you are stepping through two installation wizards, it is because you are! The first (Driver Setup) wizard checks for the ROOT certificate and prompts you to install it if needed. This certificate is used to validate the driver. After the certificate is installed, the second (Device Driver) wizard installs the **LibUSB** driver itself.

The **ExtIO_Si570.dll** file that we need is part of an install package that includes the CFGSR.exe configuration utility (which we do not actually need). To get both of these, download the setup program from **pe0fko.nl/CFGSR/setup.exe** and save it to a folder of your choice. Run this program, and it will automatically download and run the correct *Windows* installer version for your operating system. You will need to click the **Next** button to get to the Select Installation Folder dialog box. Accept the defaults and click **Next**, then confirm that you want to install by clicking **Next** yet again before installation will commence. After installation is complete, click **Close** to complete the setup. Now navigate to the CFGSR installation folder and find the subfolder **ExtIO_Si570**. (If you didn't make a note of it, the default installation folder for *Windows* XP is **C:\Program Files\CFGSR**, and for *Windows* 7 it is **C:\Program Files (x86)\CFGSR**). Look in this subfolder to find the file **ExtIO_Si570.dll**; copy this file into the *HDSDR* install directory.

Now that the software and its three **ExtIO.dll** "translators" are installed, we are ready to hook up the hardware.

Hardware Programming

If you purchased your SDRstick™ BeMicroSDK and HF1 from iQuadLabs, the BeMicroSDK is already programmed, and no hardware programming is needed. If you purchased your boards from Arrow (**www.arrownac.com/solutions/bemicro-sdk/**), you must program the BeMicroSDK yourself before it will work. The procedure is outlined in the HF1 user's manual, so we will not cover it here.[5] No hardware programming is required for the SDR-IQ™ or Softrock RX Ensemble II receivers.

Hardware setup

To set up the HF1 receiver, first plug the HF1 front-end board onto the BeMicroSDK, making sure to line up the pin 1 marks. See the HF1 users' manual for details. Plug the 5 V DC power supply into the J3 connector on the HF1 board.[6] Alternatively, you can plug the BeMicroSDK/HF1 combination into a

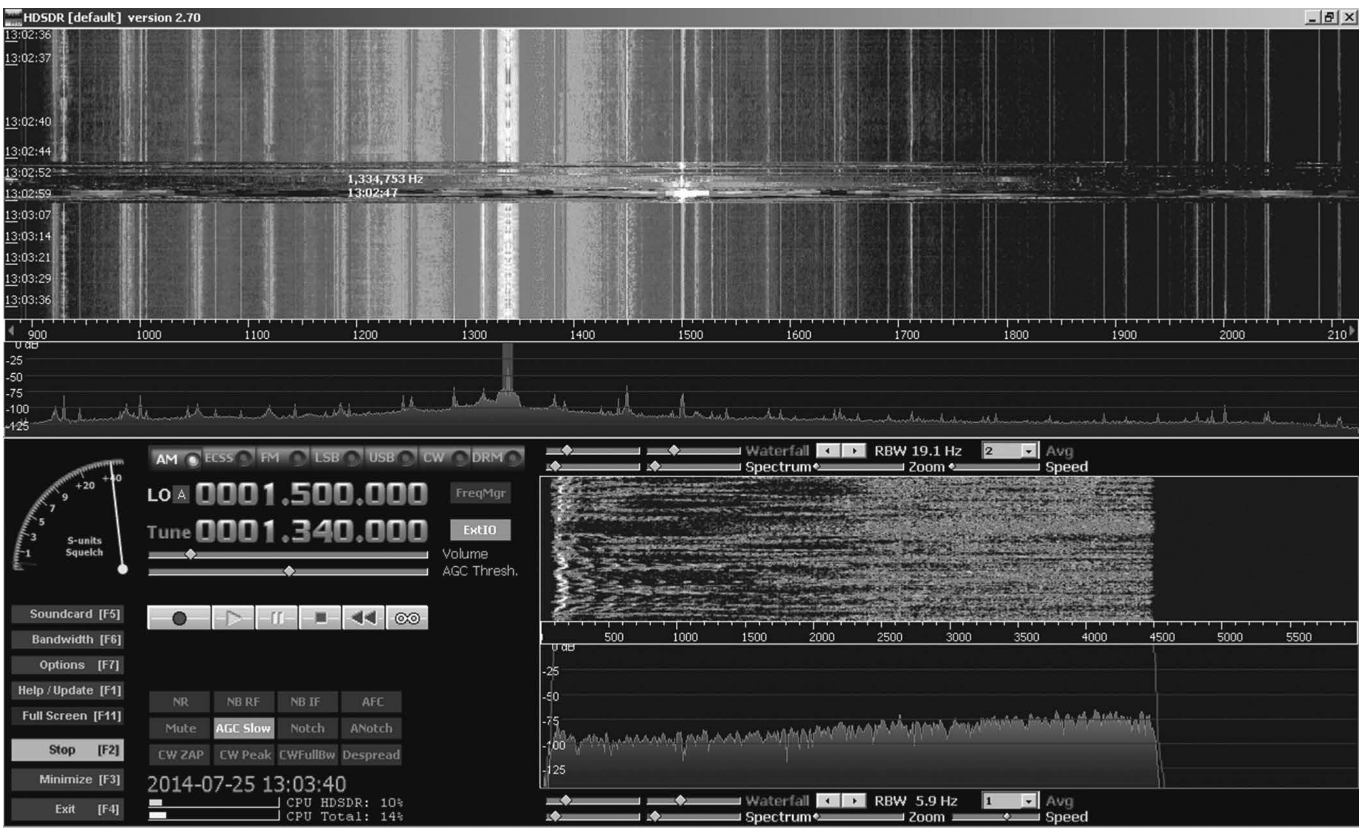


Figure 4 — Screen shot of the *HDSDR* user interface application.

USB port on your PC. *Windows* will not be able to power the receiver without the Altera USB Blaster driver, however. Again refer to the HF1 users' manual for instructions if you want to power your receiver this way. When the receiver is correctly powered up, LED V701 lights on the BeMicroSDK board and LED1 and LED2 flash at about 1 Hz. Connect the BeMicroSDK Ethernet port either directly to your PC or to a port on an Ethernet switch that is on the same subnet as your PC. (Most home networks have only one subnet, so a free port on your router's internal Ethernet switch will work fine.) Connect your antenna or signal generator to the SMA antenna connector (J4) and your hardware setup is complete.

To set up the SDR-IQ™ receiver, connect it to the PC with a USB type A to USB type B cable. These are commonly used to connect printers to your PC. The USB driver may install automatically (if a suitable FTDI driver is already on your system), but if it doesn't, point the installer to the USB driver folder to where you unzipped the driver that you downloaded. As an alternative, you can go to the Device Manager and update the driver that way, again pointing to the unzipped driver folder. Connect your antenna or signal generator to the BNC antenna connector and your hardware setup is complete. When the receiver is correctly powered up, the green power light should be on, and the yellow LED next to the antenna icon should be slowly pulsing.

To set up the Softrock RX Ensemble II receiver, connect it to the PC with a USB type A to USB type B cable. The USB driver will install automatically, since you have already installed the **LibUSB** software. Connect the **Line in** connector of the Softrock to the **Line in** connector of your PC sound card using a 3.5 mm male-to-male audio patch cord. Note that you must use the **Line in** connection; typically the **Microphone in** connection will not work because it is a monaural input. Connect 12 V power to the front panel power connector (the pin is positive). Connect your antenna or signal generator to the BNC antenna connector and your hardware setup is complete.

Operating and Troubleshooting

We are now ready to "fire" up our SDR receivers. Each receiver communicates with the PC differently. If this communication channel is not working, our SDR will not work either. Let's tackle the most difficult interface first: the Ethernet interface on HF1.

HF1 Operation

HDSDR operation with HF1 is normally quite simple, but there are a few network hurdles that you might face the first time you start the software. If you have a DHCP server running on your network and use it to assign addresses to both your PC and SDR, you will have the least problems, since the PC and radio will be on the same subnet by design. If you use a fixed address for either the PC or the radio, you must make sure that the addresses are unique and that they are on the same subnet. The default IP address for HF1 is 192.168.1.25 on the 192.168.1.x subnet.

Go ahead and start the *HDSDR* application (see Figure 4). Since you have three **ExtIO.dll** files in the *HDSDR* working directory, you will have to choose the one to use. Select **ExtIO_SDRSTICK.dll** from the list and click OK. *HDSDR* will perform a network discovery and, with any luck, find your HF1 SDR (Figure 5A). The network discovery dialog box (see Figure 5B and 5C) will warn you if your HF1 cannot be found or if your HF1 and PC are on different subnets. If you get one of these warnings, you must fix the problem before your receiver will work properly.

If you start *HDSDR* and see no dialog box of any kind, check to make sure that you placed the three **ExtIO.dll** files in the correct *HDSDR* installation directory. Also check to make sure that your PC and radio IP addresses are on the same subnet. Open a *Windows* Command Shell and ping the BeMicroSDK IP address; it should respond. If not, check your network addresses and connections. Another thing to be aware of is that multiple available networks on the PC confuse *HDSDR* network discovery. The most common occurrence of this problem is when you have your wireless network turned on. Disable your wireless network connection and it will work just fine. (You can even turn the wireless network back on after successful network discovery.) Speaking of wireless networks, the HF1 can operate over a wireless connection, but it uses a large chunk of bandwidth on the network at the higher bandwidth setting.

HF1 is capable of two display bandwidth settings, 384 kilo samples per second (Ksps) and 1.25 million samples per second (Msps). At 384 Ksps, it uses about 15 million bits per second (Mbps) of network bandwidth, and at 1.25 Msps it uses about 50 Mbps. Some wireless networks cannot handle the lower rate, and most wireless networks cannot handle the higher rate. Wired Ethernet is recommended.

Once *HDSDR* is running, click the **Start** button to turn the radio on. To select the display bandwidth, click on the **Bandwidth (F6)** button in the lower left of the display (or just press **F6**). At the bottom of the dialog box that pops up, there is a selection box; click on the down arrow to see the two sampling rate choices: 384 kHz or 1.25 MHz. Choose the higher sampling rate to see more of the band at once or the lower sampling rate if you have a slower PC.

SDR-IQ™ Operation

Start the *HDSDR* application. Again, you will have to choose which **ExtIO.dll** to use. For SDR-IQ™, select **ExtIO_SDRIQ.dll** from the list and click **OK**. In this case, there is no network discovery since the SDR-IQ™ is connected to a USB port. Just click the **Start** button to turn the radio on. To select the display bandwidth, click on the **Bandwidth (F6)** button or press **F6**. Click on the down arrow in the selection box to see the three sampling rate choices: 55.555 kHz,

(A) (B) (C)

Figure 5 — *HDSDR* Network Discovery Dialog Boxes. (A) indicates that everything is okay, (B) is seen when no SDRstick™ can be found and (C) tells you that the PC and radio are not on the same subnet.

111.111 kHz or 196.078 kHz. Just like before, choose the higher sampling rate to see more of the band at once or the lower sampling rate if you have a slower PC.

Two other settings that you can choose are only found in the SDR-IQ™, but not in the HF1 or Softrock: the RF attenuator and IF gain settings. Click on the **ExtIO** button to see the controls and settings that are implemented within the **ExtIO_SDRIQ.dll** file. Note that you can set the IF gain, RF attenuation level and the sample rate within the **ExtIO** dialog box. While you can set the sample rate elsewhere (as we have shown), the other two settings can only be made from the **ExtIO** dialog box.

Softrock RX Ensemble II Operation

Start the *HDSDR* application. As usual, you will have to choose which **ExtIO.dll** to use. For the Softrock, select **ExtIO_Si570.dll** from the list and click **OK**. No network discovery is performed and you can click the **Start** button to turn the radio on. To select the display bandwidth, click on the **Bandwidth (F6)** button or press **F6**. This time there is no selection box, but you can choose from the sample rates in the **Input** column. The highest rates may not be supported by your sound card; try them to see how they work. Typical rates will be: 48 kHz, 96 kHz or 192 kHz. Just like before, choose the higher sampling rate to see more of the band at once or the lower sampling rate if you have a slower PC.

Click on the **ExtIO** button to see controls and settings that are implemented within the **ExtIO_Si570.dll** file. Wait, what is all this? There are dozens of settings here, and things are complicated! It is time for a more detailed explanation of the Softrock RX Ensemble II architecture so we know what we are doing (at least more than we know now). The Softrock uses what is called a Quadrature Sampling Detector or QSD. This detector is similar to two conventional mixers, one operating in-phase (the I channel) and one operating in quadrature, or shifted by 90º (the Q channel). The Softrock uses a Silicon Labs Si570 IC hardware frequency synthesizer for its local oscillator. The center frequency of the display bandwidth is determined by the frequency of this local oscillator that feeds the two mixers.

The Softrock uses the USB port to control the Si570 chip. The Si570 is very versatile, and many settings are needed to configure it. This is why you see so many tabs when you click on the **ExtIO** button. Note that the I and Q data from the mixer is sent to the PC soundcard line-in, and not over the USB connection. This is the analog receive data that the PC sound card processes to form the digital I/Q data stream that *HDSDR* demodulates and filters. The term *analog* should give away the reason that the sound card is used to transport the data instead of the USB port. The Softrock uses separate control (USB) and data (soundcard) ports, while the HF1 and SDR-IQ™ each use only one port for both control and data (Ethernet and USB, respectively). The HF1 and SDR-IQ™ have on-board Analog-to-Digital converters, so their receiver data is *digital* and can be easily sent over the digital interfaces (USB and Ethernet).

It is beyond the scope of this column to cover all of the Softrock ExtIO options here, but there are a few settings that you need to make before you can use the receiver. These settings only need to be made once, the first time you run *HDSDR* with new hardware. In the **USB** tab, click on the USB device from the list in the bottom window (there will likely only be one device in the list) and then click the **Open** button at the bottom of the window. In the **Init** tab, click the button next to the radio type, in our case **Ensemble RX II LF/HF(HF)**. You can now click on the **Hide** button to dismiss the ExtIO window and get back to using the radio. To open the ExtIO window again, just click on the **ExtIO** button on the main *HDSDR* screen at any time. There is a wealth of information on the Softrock RX Ensemble II on line, and you can research and dig into the finer nuances of Si570 settings to your heart's content.[7]

HDSDR Operation for All Receivers

You can adjust the frequency by mousing over the fields in the **Tune** section and using your mouse wheel. If your mouse has no wheel, use the left mouse button for up and the right mouse button for down. As already described, use the **Bandwidth** (or **F6**) button to choose to see either more or less bandwidth on the panadapter and waterfall displays. Choose narrower bandwidths if you are running a less powerful PC and experience audio dropouts. The lower waterfall and panadapter windows are just zoomed-in versions of the pass band that is highlighted in the upper window. Slowing the screen update rates with the **Speed** sliders below each panadapter window can also ease the load on your creaky old PC if you experience audio problems.

Go ahead and play around with the settings in the user interface. You can grab either edge of the pass band in the lower panadapter window and drag it to perform low-cut or high-cut on the audio. You can click in the upper panadapter window to instantly change frequency. You can record and playback using the controls just below the **Volume** and **AGC** sliders. Keep in mind that when you record, you are recording the entire spectrum, not just your selected pass band. This uses a lot of disk space at the wider bandwidth settings. *HDSDR* even has a scheduler for recording at predetermined times (it is under the **Options** menu). The advantage of recording the entire spectrum is that while playing back your recording, you can tune within the recorded spectrum, just as if you were listening to the airwaves "live."

One other thing to be aware of is ADC overload. As we have already described, the SDR-IQ™ has an internal 20 dB attenuator that you can use to prevent overload. The HF1 front-end uses a 20 dB preamp that provides sufficient sensitivity for reasonable size antennas, but has no way to adjust any gain settings, and can be overloaded when using a large antenna. The Softrock can also be overloaded with RF from a large antenna. If you have a Yagi at 150 feet or live next door to a 50 kW broadcast station, you may overload the receiver front-end. For HF1, LED8 on the BeMicroSDK indicates ADC overload. On the SDR-IQ™ front panel, the red LED above the clipped sine wave icon is the ADC overload indicator. You should *never* see these light up for more than an occasional flicker. If they do, you must reduce the level of the offending signal. Remember, the front-end has a broadband frequency response.

The offending signal could be almost anywhere! You can just reduce all signals by adding an in-line attenuator (or using the SDR-IQ™ built-in attenuator), or you can selectively remove the culprit with an external notch-, low-, or high-pass filter if you can identify it. The Softrock does have four internally switched low-pass filters, but unfortunately it has no overload indicator. Overload indications are distortion in the audio accompanied by a general rise in the noise floor (as seen on the panadapter display) and/or appearance of many spurious signals across the viewable spectrum.

If you get stuck and just can't make your receiver work, there is help available. For the HF1 receiver, the **SDRstick** Yahoo group has about 250 members and is a friendly group.[8] For the SDR-IQ™, the **SDR-IQ** group is your resource.[9] The Softrock Yahoo group is named **Softrock-40** after the first radio in the series.[10] Just ask a question on any of these forums and you will very likely get help quickly.

Next Installment: Interesting SDR Applications

Now that we have a working SDR, what can we use it for? Get ready to do some broadcast-band DXing, where you can *see* the signals, listen to digital shortwave broadcasts or copy one of the new digital modes.

Scotty Cowling, WA2DFI, was first licensed in 1967 as WN2DFI, and has been continuously active since that time. An Extra Class licensee and ARRL Life Member, Scotty is active while mobile on HF CW and on APRS. He is an advisor for Explorer Post 599, a BSA affiliated ham club for teens in the Phoenix, Arizona area. He also enjoys minimalist QRP operating. He has participated in every ARRL Field Day since 1968!

Scotty has been involved in the openHPSDR project for the last 8 years, and has served on the TAPR Board of Directors (2006-2012) and as TAPR Vice President (2011-2012). He is active in the production of openHPSDR components and with other TAPR projects. He is a co-founder of iQuadLabs, LLC, a supplier of openHPSDR systems and other Software Defined Radio components, and is President of Zephyr Engineering, Inc, an engineering consulting firm.

Scotty's professional specialty is FPGA and embedded systems hardware design. He designed his first project with a microprocessor in 1975 and his first FPGA project was in 1987. He holds a BSEE from Rensselaer Polytechnic Institute and an MSEE from Arizona State University.

Notes

[1] BeMicroSDK and UDPSDR-HF1 are available for $243 from either iQuadLabs (**www.iQuadLabs.com**) or Arrow Electronics (**www.arrow.com**).

[2] Scotty Cowling, WA2DFI, "Hardware Building Blocks for High Performance SDRs", *QEX*, Jul/Aug 2014, pp 28-40.

[3] SDR-IQ™ receivers are available for $525 from RFSPACE, Inc. **www.rfspace.com/RFSPACE/SDR-IQ**.

[4] Softrock RX Ensemble II kits are available for $67 from Five Dash Inc **www.fivedash.com/index.php?main_page=product_info&cPath=1&products_id=7**. Assembled boards are available for $92 and metal enclosures for $20.

[5] The HF1 *User's Manual* can be downloaded at **www.sdrstick.com/manuals/UDPSDR_HF1_users_manual_V1_1.pdf**.

[6] The power connector for the HF1 is a 2.35 mm × 0.7 mm barrel, CUI part number PP-012, DigiKey CP-012-ND.

[7] Softrock RX Ensemble II information is available at these links: **www.wb5rvz.org/ensemble_rx_ii/index?projectId=16**. **www.wb5rvz.org/common/softwareinstallation**.

[8] The SDRstick Yahoo group is at: **www.groups.yahoo.com/groups/sdrstick**.

[9] The SDR-IQ Yahoo group is at: **www.groups.yahoo.com/groups/SDR-IQ**.

[10] The Softrock Yahoo group is at: **www.groups.yahoo.com/groups/softrock40**.

Letters to the Editor

An Automated Method for Measuring Quartz Crystals (Nov/Dec 2013)

Hi Larry,

Readers may like to note that in the article "An Automated Method for Measuring Quartz Crystals" by Richard J. Harris, G3OTK, *QEX* Nov/Dec 2013 pp 3-8, an exact formula can in fact be written for C_h. The formidable looking expressions in the text hide the fact that the algebra contains numerous cancellations. Using the valid approximation:

$$\left(\frac{f_n}{f_0}\right)^2 - 1 \approx 2\left(\frac{f_n}{f_0} - 1\right)$$ we can derive:

$$C_h = \frac{C_2^{\parallel}\left(f_2 - f_0\right)\left[C_{osc} - \left(\frac{f_1}{f_0}\right)^2 C_1^{\parallel}\right] - C_1^{\parallel}\left(f_1 - f_0\right)\left[C_{osc} - \left(\frac{f_2}{f_0}\right)^2 C_2^{\parallel}\right]}{\left(f_1 - f_0\right)\left[C_{osc} - \left(\frac{f_2}{f_0}\right)^2 C_2^{\parallel}\right] - \left(f_2 - f_0\right)\left[C_{osc} - \left(\frac{f_1}{f_0}\right)^2 C_1^{\parallel}\right]} \quad \text{[Eq 1]}$$

where:

$$C_n^{\parallel} = \frac{C_n C_{osc}}{C_n + C_{osc}} \quad \text{[Eq 2]}$$

(n = 1, 2) are the series capacitances as given in the text. Similar to Equation 12 in the text, which can be rewritten as Equation 3:

$$C_m = 2\left(\frac{f_0 - f_s}{f_s}\right)\left(C_h + C_{osc}\right) \quad \text{[Eq 3]}$$

we can also easily show that:

$$C_m = 2\left(\frac{f_n - f_s}{f_s}\right)\left(C_h + C_n^{\parallel}\right) \quad \text{[Eq 4]}$$

Note that in the limit C_{osc} going to infinity then $C_n^{\parallel} \approx C_n$ and we recover the familiar Equation 2 from the August 2007 *QST* Technical Correspondence letter by Jack Smith, K8ZOA, "Measuring Motional Parameters of a Quartz Crystal," but it is not identical to it:[1]

$$C_m = 2\left(\frac{f_n - f_s}{f_s}\right)\left(C_h + C_n\right) \quad \text{[Eq 5]}$$

From the above results we can also obtain Equation 6:

$$C_m \approx 2\left(\frac{f_n - f_0}{f_0}\right)\left[\frac{\left(C_h + C_n^{\parallel}\right)\left(C_h + C_{osc}\right)}{\left(C_{osc} - C_n^{\parallel}\right)}\right] \quad \text{[Eq 6]}$$

This approximation is good for $C_m << (C_h + C_{osc})$, which in the limit of C_{osc} going to infinity is identical with Equation 2 from Jack Smith's Technical Correspondence:

$$C_m \approx 2\left(\frac{f_n - f_0}{f_0}\right)\left(C_h + C_n\right) \quad \text{[Eq 7]}$$

Using values from Figure 1 of Jack Smith's Technical Correspondence, we find a discrepancy of just under 7% too low from the formula of Equation 6, in agreement with Jack's remarks of a – 5% further discrepancy in his Equation 2 from vector network analyzer measurements. Unfortunately no simple fudge factor formula as he suggested for a capacitance can be added to C_n in Equation 7 to obtain agreement with Equation 6. Using my Equation 6, we can also derive a simpler formula for C_h than my Equation 1:

$$C_h \approx \frac{C_2^{\parallel}\left(f_2 - f_0\right)\left[C_{osc} - C_1^{\parallel}\right] - C_1^{\parallel}\left(f_1 - f_0\right)\left[C_{osc} - C_2^{\parallel}\right]}{\left(f_1 - f_0\right)\left[C_{osc} - C_2^{\parallel}\right] - \left(f_2 - f_0\right)\left[C_{osc} - C_1^{\parallel}\right]} \quad \text{[Eq 8]}$$

In view of the simpler formulas given in Equations 6 and 8, which can easily be coded into a program or spreadsheet, they can be used for finding C_h and C_m with an accuracy of at least 2%, without having to perform a numerical solution.

Note that all the formulas are subject to 1 / Q^2 corrections (See the *QEX* Letters to the Editor by Alan, Bloom N1AL.[2]), which are insignificant for high *Q* HF crystals but may be important for VHF and higher frequency overtone crystals. This is a subject recently investigated further by Jim Koehler, VE5FP.[3]

— 73, Tuck Choy, MØTCC; **m0tcc@arrl.net**

Hi Larry,

I must congratulate Tuck, MØTCC, for deriving an equation for the holder capacitance of crystals. I had tried to derive such a formula myself but gave up when I realized that I could achieve the same result using a numerical method. I have used Tuck's formula (Equation 1) with data that I measured and I can confirm that it yields the same values for holder capacitance within ± 0.1 pF. Such small differences may well be due to measurement uncertainty.

In my article I mentioned that I had not read of a method of automatically measuring the motional parameters and holder capacitance of crystals. I have subsequently learned of another method that was described by Jim Koehler, VE5FP, in an article entitled "Some Thoughts on Crystal Parameter Measurement," and which was published in the July/August 2008 edition of *QEX*, as noted by Tuck. (See Note 3.)

—73, Richard Harris, G3OTK, 10 South St, South Petherton, Somerset TA13 5AD, United Kingdom; **r.j.harris.g3otk@gmail.com**

Notes

[1]Jack Smith, K8ZOA, "Measuring Motional Parameters of a Quartz Crystal (Technical Correspondence)," *QST* Aug 2007, pp 74-75. A straightforward derivation in a Colpitts oscillator configuration is also given by Andrew Smith, G4OEP on his website: **http://g4oep.atspace.com/crystal%20parameters/deriving_g3uur.htm** but the Colpitts oscillator configuration with C_{osc} assumed infinite is in fact not necessary for its derivation.

[2]Alan Bloom, N1AL, "Letters to the Editor," *QEX* Sep/Oct 2008, pp 41-42 and *QEX* Jan/Feb 2010, pp 37-38.

[3]Jim Koehler, VE5FP, "The Shunt Method for Crystal Parameter Measurement," *QEX* Jul/Aug 2010, pp 16-25. Jim performed a quite thorough analysis of the discrepancies in various formulas for overtone crystals.

Upcoming Conferences

AMSAT Symposium

October 10-12, 2014
Double Tree by Hilton Baltimore — BWI Airport
890 Elkridge Landing Rd
Linthicum, MD 21090
Hotel Reservation Phone:
410-859-8400

The 2014 AMSAT Symposium and General Membership Meeting will be held in the Baltimore/Washington DC area, at the Double Tree by Hilton Baltimore — BWI Airport, on the weekend of October 10-12 2014.

Whether you are a seasoned satellite operator or think you might like to give it a try, the Symposium will have plenty of presentations of interest. Catch up on the latest news about AMSAT's various satellite projects as well as presentations about equipment and operating practices. The Call For Papers has not been posted to the AMSAT website at the time this column was being prepared, but now is the time to start thinking about any paper you might be interested in writing, for presentation at the Symposium, or for inclusion in the *Proceedings* book.

Check the AMSAT website (**ww3.amsat.org)** for updated information. Mark those dates on your calendar now, and plan to attend.

Microwave Update

October 24-25, 2014
Rochester Marriott Airport
1890 Ridge Road West
Rochester, NY 14615
Hotel Reservation Phone: 585-248-8640 or 800-228-9290

The Rochester VHF Group (RVHFG) is hosting MUD 2014 at the Rochester Marriott Airport Hotel. Microwave Update is an annual technical conference and includes presentations by leading Amateur Radio microwave experimenters. Attendees from all over the world have the opportunity to discuss the latest technical developments and operating achievements taking place in the 902 MHz-and-up amateur radio frequencies. There will be test equipment for measurements, including noise figure up to 47 GHz.

The conference will conclude with the Saturday evening dinner banquet. For early arrivals on Thursday, there will be a tour at the nearby Antique Wireless Association (AWA) Museum.

Microwave Update 2014 — Call For Papers

The Microwave Update 2014 program committee is calling for papers and presentations on the technical and operational aspects of microwave Amateur Radio communications. Papers will be published in the conference proceedings (print and CD). Many will also be selected for presentation at the conference.

The deadline for proceedings paper submissions is August 15, 2014. The Microsoft Word file format (text) is preferred for these papers. The deadline for the presentation version of selected papers is September 1, 2014; Microsoft PowerPoint (slides) file format is preferred for presentations.

Even if you will be doing a presentation at the conference, please try to make your proceedings paper submission more than just the outline slides from the presentation. We would like the published proceedings to provide full content for people who are not able to attend the conference presentations.

Detailed formatting information for authors (margins, photos, other files) is provided on the website.

Please e-mail your papers, as well as questions or comments regarding the technical program, to Bill Rogers K2TER: **k2ter@rochester.rr.com**.

Solicited topic areas include:

- Centimeter, millimeter, submillimeter and light wavelengths.
- Antenna design, simulation, construction, measurement, application.
- Microwave building blocks (LNAs, PAs, LO chains, Mixers, Synthesizers, Filters, and so on).
- Transverters (single and multiband).
- Fixed station, Rover and Beacon design, packaging and operation.
- Operating techniques, software and other aids.
- Weak signal propagation modes and enhancements.
- New or unusual emission modes (ATV, digital modulation, wide area packet networks, and similar topics).
- Practical effects and limits of phase noise, antennas, path characteristics on various emission modes.
- Microwave components (affordable and available modern commercial components; homebrewed; surplus).
- Repeaters (microwave bands and/or unusual modes like ATV, packet WAN).
- Construction techniques (SMT, wirebond, microstrip, waveguide, substrates, homebrew).
- Measurement equipment and techniques (tuning amplifiers or filters, optimizing noise figure, measuring phase noise, antenna patterns and gain; professional results on homebrew/shoestring budgets).
- CAD (preferably free or low cost) for circuit, antenna, path and system simulation and design.
- Conversion of surplus microwave equipment.
- Or — suggest your own topics.

Submissions may range from short notes to full length technical papers, original research to hints and tips, new designs to surplus conversions, professionally engineered to hacked on a shoestring budget.

Survey papers that summarize current know-how and tutorials that help and encourage newcomers are also welcome. Some topics may be organized and presented as workshops (for example, construction and measurement techniques).

We are looking forward to seeing you and your presentation at MUD 2014.

Bill Rogers K2TER: **k2ter@rochester.rr.com**.

MUD 2014 Technical Program Chairman

For further details and registration, go to the Microwave Update 2014 website **www.microwaveupdate.org**.