The QRSS-Rx

An entry into the Circuit Cellar NXP mbed Design Challenge

Clayton G, VK1TKA
clayton@isnotcrazy.com

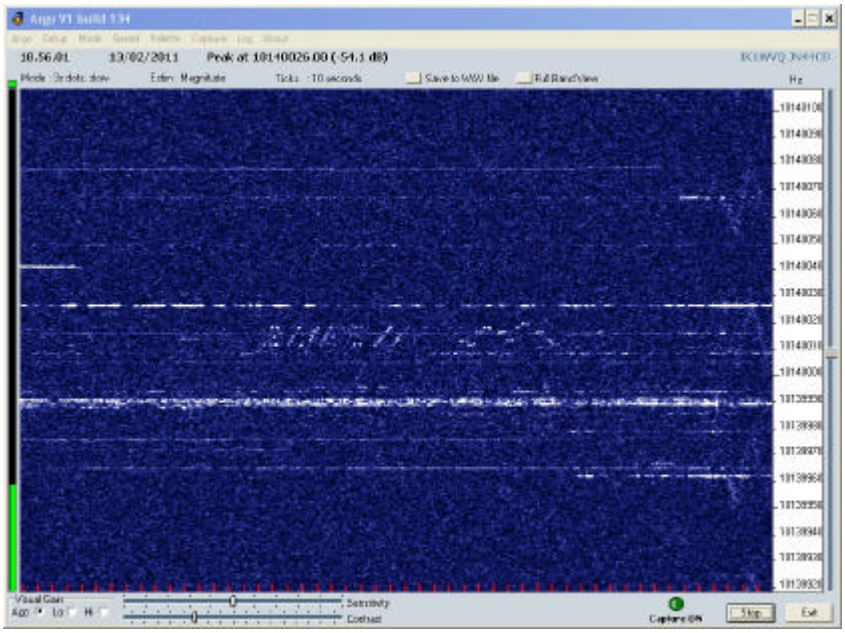
1. Introduction

QRSS is one of many modes of communications and experimentation used by radio amateurs. It involved the transmission of very low speed Morse (3 second long 'dits') from very low power transmitters, and their reception on special 'visually' receivers called 'grabbers'. Because of the slow transmission rates very low level signals can still be received, even if they cannot be hear by ear. QRSS transmitters are called 'Manned Experimental Propagation Transmitters' – or MEPT for short. Even with only 100mW of transmit power, signals can be received on the other side of the world. QRSS signals are mostly on the HF bands where radio propagation varies a lot. Most activity occurs on the 30 metre band in a 100Hz wide frequency band between 10,140,000Hz and 10,140,100Hz.

The QRSS receiver systems use what are called 'grabbers'. These systems receive a small bandwidth of the RF spectrum, convert it to an image (spectrum vs time), and make it available on the internet. Reception of a signal can therefore be viewed in (almost) real-time from receiver stations all over the world. A grabber image will typically show the spectrum of only 100Hz of RF bandwidth, displayed with an update rate of about 1 pixel column per second. A typical image would therefore show over 10 minute of activity. Such images then allow for Morse signals (with the transmission turning on and off or being frequency modulated by a few hertz), or for more graphical modes such as Hellschreiber where the spectrum displays text characters.

The QRSS-RX NXP mbed Design Challenge

An example of a grabber image:



The hardware involved with a grabber receive system is not insignificant. It requires a very stable radio receiver, a PC (with soundcard), and an internet connection. The receiver audio is feed into the PC soundcard. Software then converts the audio to an image. Further software makes these images available on a webpage. There are a variety of different software configurations used with each grabber having its own look and feel.

Each Grabbers is hosted on its own webpage - although there are some website that will aggregate a number of grabbers onto the same page. Finding your signal can therefore require the inspection of a number of websites.

Dealing with such a small RF spectrum requires that the grabber receiver is precisely tuned to the band. Typically receivers will drift, so keeping a grabber on frequency can be challenging.

Because of these receiver challenges there are few grabber stations available – there are only about a dozen grabber commonly operating.

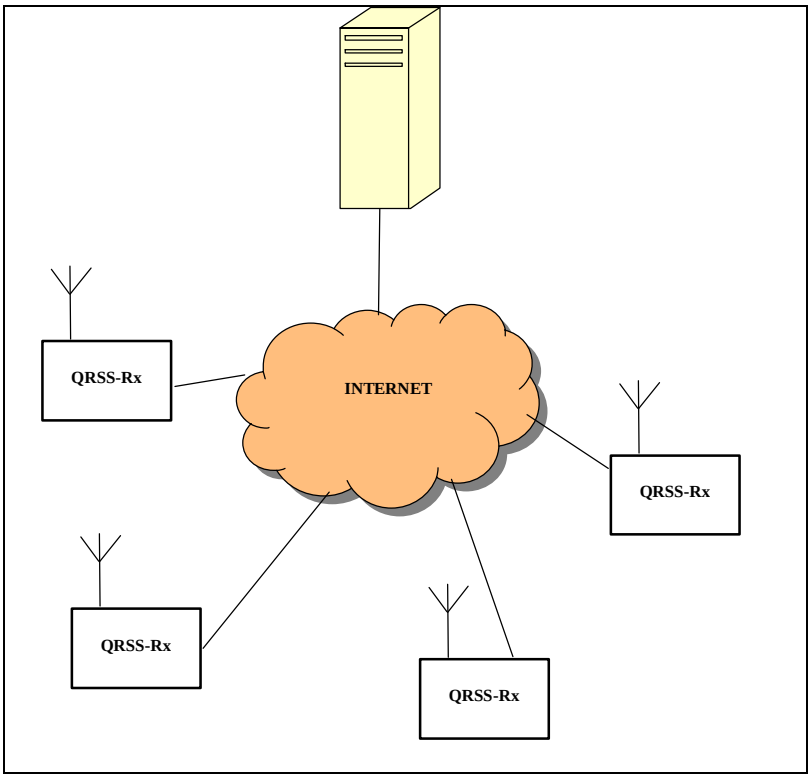
This project. - the ‘QRSS-Rx’ - is designed to improve the QRSS grabbers system. The idea is for a central server to receive spectrum data from all over the world, and then make it available on a single website. The QRSS-Rx is the receiver part of this system. Many of these can be located around the world. They receive and digitize the RF spectrum of interest, and pass the samples data to the server for further processing and display. The QRSS-Rx is designed to be a cheap and simple unit that will allow the setup of a receiver without tieing up expensize receivers and PCs.

The server portion of this QRSS grabber system is beyond the scope of this mbed project documented here – only enough server software to prove the operation of the QRSS-Rx has been developed.

The QRSS-Rx has the following features:

- Uses a ‘softrock’ RF fronted. This is a software defined radio receiver available as a kit from a radio amateur (Tony Parks KB9YIG). It is cheap at only $14. It produces a quadrature audio output which is usually designed to feed into a PC sound-card where software implements a normal radio receiver. It is crystal controlled, so is limited to just a small RF spectrum (typically 50-100kHz, depending on the soundcard used)
- Has a 24-bit stereo ADC to convert the Softrock outputs. This interfaces to the mbed I2S interface.
- Uses the Softrock’s crystal osciallator as the ADC reference clock.
- Interfaces to a GPS receiver which provides location information and a 1PPS (1 pulse per second) pulse to allow the softrock sciallator to be calibrated.
- Performs DSP (digital signal processing) in software to further down-convert the signals from the softrock down to the frequency of interest, and then to filter and decimate (reduce sample rate) the samples.
- Sends out samples over Ethernet in UDP packets.
- Sends GPS and timing information to the server to allow the server to calculate calibrartion parameters
- Accepts tuning information from the server and uses it to acjust the DSP down-converter.

2. The Proposed QRSS Network.



The QRSS-Rx is the key to the construction of a network of QRSS receivers around the world. The QRSS-Rx is small enough and cheap enough that it is likely to be more widely installed than the current grabber installation base. A central server will provide a common interface into these many receivers, allowing all world-wide grabbers to be shown on a single web page and updated in real time.

The main limitation of the system at present is the need for a server, however it is likely that someone in the QRSS community would be able to help develop and run such a system.

The data-rate from a single QRSS receiver is relatively low (about 35kbps), such that it is unlikely to be noticeable in a users broadband capacity. There is still plenty of scope in compressing the data sent, and so greatly reducing this rate and the overall amount of data that needs to be transferred and possibly stored on the server.

No longer will a grabber need to tie up expensive radio and PC equipment and consume power. The main restriction now is possibly the need for a reasonable antenna.

The prototype QRSS receiver is designed for the 30 metres band, however it only required a few changes to the Softrock receiver to convert it to any other HF band.

3. Data/Signal Flow through the QRSS-RX

The RF input is to the Softrock receiver. This has a central frequency of 10.125MHz. The Softrock quadrature oscillator (derived by dividing its main oscillator by four) operates at

10.125/3 = 3.375MHz. Using under-sampling the 10.125 centre RF signals are converted to audio frequencies.

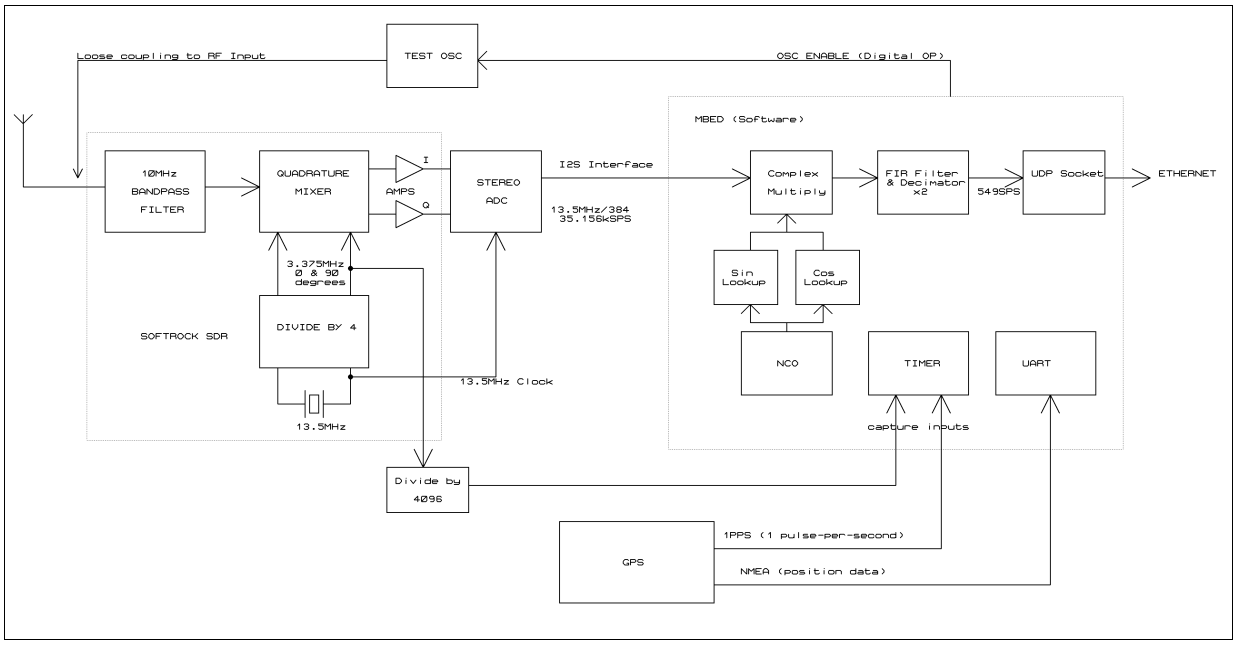
The QRSS band of interest at 10.140MHz will be at 15kHz in the Softrock audio output (10.140MHz – 10.125MHz = 15kHz). The ADC device samples the I and Q signals at 35.156kHz (derived from 13.5MHz/384). Due to Nyquist rules, we need a frequency of over 30kHz to sample our 15kHz signal of interest.

Once sampled, the I/Q data is treated as complex numbers (i.e. as a real and imaginary component). This allows the DSP frequency shifting to be performed by doing a complex multiply between the signal samples and the software generated local oscillator. This oscillator operates at 15kHz (its exact frequency is tuned by the server) to bring the frequencies of interest down to 0 hertz.

The samples are then filtered with a software Low-pass FIR filter. This has a cut-off frequency of 200Hz. Samples can then be decimates by 1:64 (i.e. we only keep every 64th sample). Since we have filtered the samples, the extra samples do not provide any extra information and so can be discarded. The sample rate is then at 559SPS, as complex data (two values per sample). This is the data that is then sent to the server for further processing into grabber images or stored for future reference.

4. Block Diagram

This diagram shows the structure of the QRSS-Rx hardware and software. It also shows how the signal flows from the antenna, through the Softrock (hardware), and then (after being converter by the ADC) through the DSP (software) before being sent out as Ethernet data. The frequency calibration system, GPS module, and test oscillator are also shown.



4.1. Softrock

This is the initial RF down converter. RF goes in; audio comes out. The down-converter is a quadrature converter. There are two mixers happening – one with the local oscillator; the other with a phase shifted (by 90 degrees) local oscillator. By generating both of these signals further processing can be used to distinguish one sideband from the other (i.e. signals above the local oscillator in frequency between signal below the local oscillator in frequency). Without the two outputs the audio would be a mixture of both sidebands with no way to separate them.

4.2. ADC Converter

A stereo ADC is used to translate the Softrock output to digital values. This converter has an I2S interface, so is compatible with the mbed I2S port. A 24 bit ADC is used so that the signals have a large dynamic range. The converter uses the Softrock's 13.5MHz clock. This is at a convenient frequency to generate 35.156kSPS - the ADC is configured for a clock to sample ratio of 384.

4.3. Frequency Divider

A counter is used to divide the oscillator frequency from the Softrock. The Softrock divides its oscillator by four. The counter further divides this by 4096. The result in a 830Hz frequency passed to the MBed for frequency measurement.

4.4. MBed

The mbed I2C interface is used for the ADC interface.

A UART is used to receive serial data from the GPS module. This data is NMEA format. It contains GPS status and position information. Software parses this data so that it can pass it on to the server.

A timer is used in capture mode for frequency calibration purposes. The timer is set to run at the full CPU clock rate (96MHz). The 1PPS signal from the GPS is used to capture the value of this timer at the 1PPS instance. By observing these values, the exact frequency of the 96MHz CPU clock can be calculated.

At every 1PPS pulse the software also records the last capture value for the 830Hz input. This will allow the period of approximately 830 cycles of this signal to be determined in CPU clock rates. The actual number of 830Hz cycles will not be known because of the clock synchronization – it may be 829, 830 or 831; but because the clock has a reasonable accuracy (it is crystal controlled) the true frequency can be easily determined anyway. Because the CPU clock rate has been accurately calculated using the 1PPS captures, the 13.5MHz reference clock frequency can also be accurately calculated from the 830MHz capture readings.

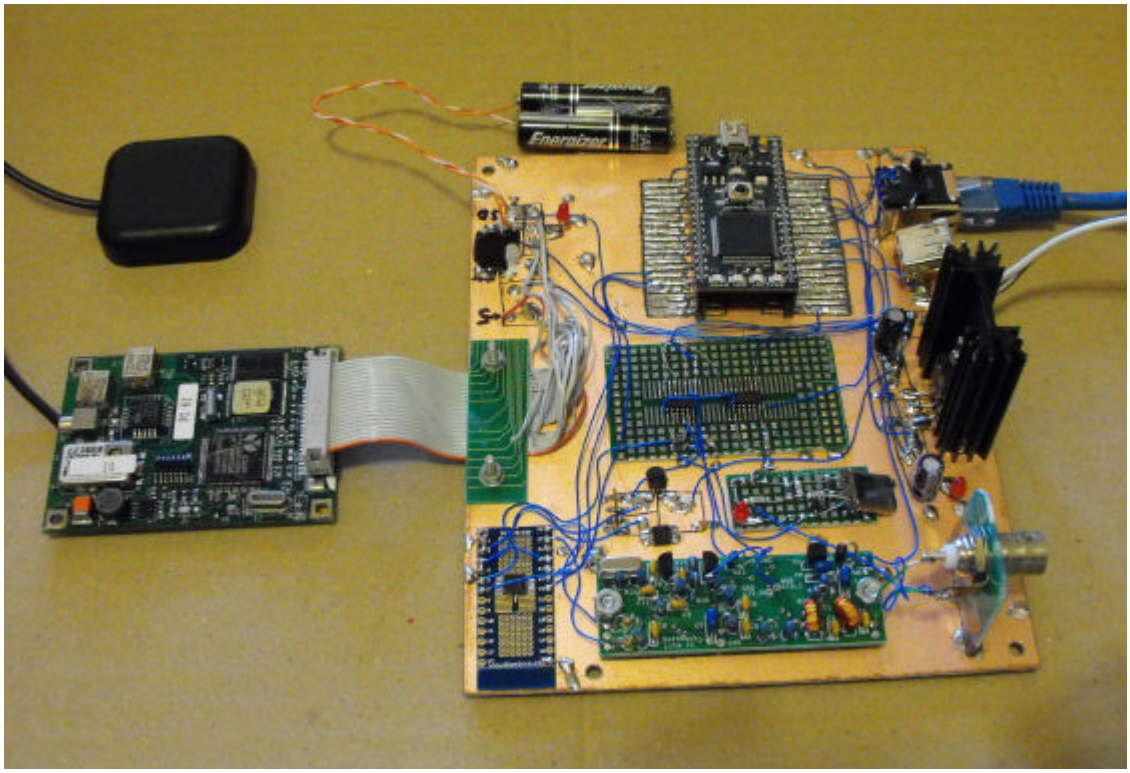
The mbed Ethernet port connects to an RJ-45 MagJack. This provides the Ethernet magnetics as well as a standard Ethernet RJ-45 connector.

4.5. Test Oscillator

For test purposes a 10.14MHz test oscillator is included. The mbed can turn this on and off. Its output is loosely coupled to the RF input of the Softrock (a wire is loosely wrapped around the RF input wires). This is included to provide a signal to test the system performance.

5. Hardware

A picture of the QRSS-Rx:



The QRSS-Rx is assembled on a piece of blank double-sided PCB. The PCB is hand routed using a rotary tool and a dental bit to cut islands of copper in the board. Islands are cut for the mbed socket – a small pad for each mbed pin. Further island are cut for some of the power supply circuits and the GPS connections. Other parts of the circuit use various prototype PCB. The ADC uses a small TSOP14 break-out adaptor PCB. The logic chips mount on a similar piece of prototype PCB. All interconnections are then made using thin wire-wrap wire.

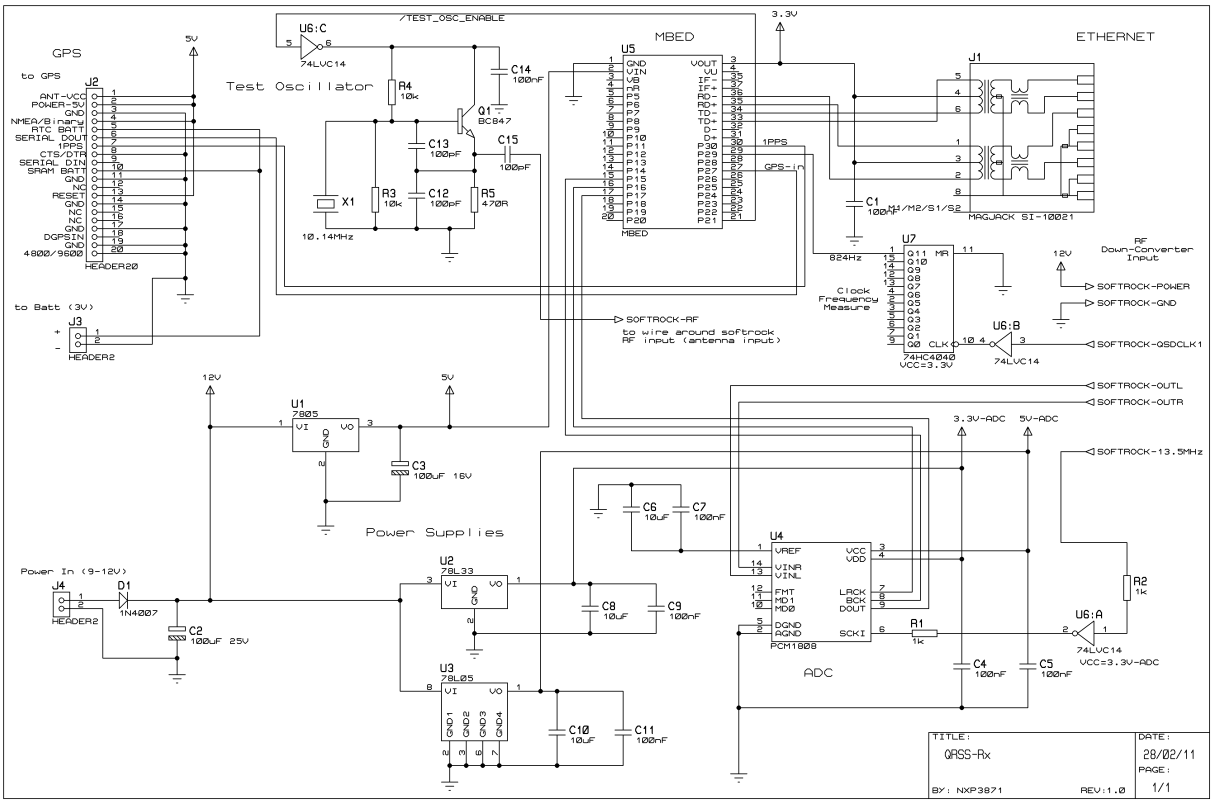
The Softrock RF PCB is mounted on the PCB and direct wired as necessary into the circuit.

The main power supply (for the mbed and the GPS unit) requires a heatsink. This is bolted down to the PCB and the regulator mounted to it. A separate power supply (5V and 3.3V) is implemented for the ADC device to minimize noise into this device for the best possible digital samples. The clock buffer for this device is also powered from this supply to minimize sampling clock noise.

The GPS module has a 20-pin pin-header. The connection to this is made with a length of ribbon cable and a header plug. The ribbon cable wires are split, stripped and connected to PCB islands which are then connected to the signals required. A GPS antenna connected directly to the GPS module.

The Ethernet connection is secured to the PCB by soldering it down. A USB connector is also added to the PCB and connected to 5V only. It is there to power a Ethernet-to-WiFi adapter that was used during testing.

The schematic of the QRSS-Rx is shown below:



6. Software

The software is written in C++. Extensive use is made of the libraries from the mbed online compiler. The ‘mbed’ library provides access to the basic ports (serial ports, digital IO) and runs software timers. The ‘NetServices’ library handles the Ethernet and UDP communications. A ‘DNSResolver’ library is used for DNS lookup, to determine where to send the poll message to.

The following software modules are then added:

6.1. BufferSys

This is the buffer system. It handles data samples. Each sample consists of a pair of ADC values – the I value and the Q value. The buffer system defines a structure to hold a set of sample data. It has a buffer-handler class, which can 'hold' a sample set and perform operations on it (add or remove data). Also implemented is a buffer queue that can be used to queue up sets of sample data. Finally a buffer pool is defined. This is a place for used sample sets to be returned to ready for reuse.

The buffer system is set up for a sample-set size of 512 samples. Since each sample is eight bytes (two 32-bit words) each buffer is 4K long. There is only room for four buffers in the system, but this seems to be enough and can cope with the various delays in the system processing.

6.2. I2S_RX

This is the driver for the I2S interface to the ADC chip. It uses DMA transfers to read I2S data into a buffer sample data set. As each buffer is filled, the DMA interrupt occurs. The buffer is then queued and a new one obtained from the pool for more data.

6.3. GPS Module

The GPS module handles the data from the GPS serial port, and performs the 1PPS capture operation. Data obtained from the GPS is passed to the comms module for sending to the server as required.

GPS data that is recorded is:

- GPS quality (quality of the fix) – this indicates if the position and 1PPS are accurate or not.
- GPS Satellites count – the number of satellites being tracked. This is for diagnostic purposes only – it provides an indication of the GPS and GPS antenna performance
- Time – the current time (UTC) to the second
- Latitude/Longatude/Altitude – the GPS position information
- Timer Capture registers for the 1PPS capture and the 830Hz capture at this time.

6.4. Comms Module

This is the module that handles all communication over the Ethernet. It sends out UDP poll messages to a names server. It accepts commands from this server, along with parameters, to start and stop the receiver. When started it handles the transfer of sample sets to the server using UDP.

6.5. DSP Module

This module is actually inherited from a buffer handle. It adds functions to this handle to perform DSP operations on the sample set held.

One function is to mix a software local oscillator with the samples. The local oscillator is an NCO (numerically controlled oscillator) which with each sample adds a phase value (this determines the NCO frequency) into a phase accumulator. The accumulator is then used to look up a table and obtain a cosine and sin value. This value pair is treated as a complex number, and is multiplied (as a complex multiply operation) with the data sample. This has the effect of shifting in frequency the samples. The server sets the frequency of the NCO so that the band of interest is shifted close to zero frequency.

The second DSP operation is a LPF (low pass filter) and decimator (reduction in samples). A FIR filter is used. It sums up 511 scaled samples (each one scaled based on the filter co-efficients) to produce one output. As the samples are then decimated 1:64, this operation only needs to be performed on every 64 samples – that sample being the result of 511 other samples. The co-efficients for this filter were obtained by an online calculator (http://www-users.cs.york.ac.uk/~fisher/cgi-bin/mkfscript).

The DSP module output is an array of 8 sample sets for each data buffer processed. This size is derived from the buffer size – 512 samples with 64:1 decimation, giving 8 output samples. The DSP module maintains the current NCO phase accumulator, and also contains interim FIR filter totals between one sample set and the next.

6.6. Main

This is the main routine. It runs the main processing loop which handles the transfer of data between the ADC receiver, through the DSP processing, and to the communications module. It also handles the debug console (the serial port) – accepting keystrokes and executing the appropriate test routine.

7. Communications Protocol

Three messages are transferred over the Ethernet – a poll message, which the QRSS-Rx uses to report its status (GPS data etc) and existence to the server. A Command message sent by the server to control the QRSS-Rx, and a data message used to transfer samples.

7.1. Poll Message

This is sent every 6 seconds from the QRSS-Rx to the server.

All fields are 32-bit words sent in network byte order (high byte first)

Poll Message ID (10)
Message Protovol Version (1)
Hardware ID – identified the hardware structure type (1)
Software Version – the version of QRSS-Rx software (1)
GPS quality identified – 0 = offline 1 = online
GPS satellite count
GPS Time – seconds since midnight UTC
Latitude – millionths of a degree units. Negative = south
Longitude – millionths of a degree units. Negative = west
Altitude in metres
CPU reference clock – 96,000,000
RF Mixer Clock – 10,120,000
Sample Clockrate – 13,500,000
Output sample rate divider (from Sample clockrate) – 384 * 64
1PPS capture values. Eight capture samples from the newest down.
Divided RF mixer/sample clocks capture values. Eight capture samples from the newest down.
QRSS-Rx unique name – 10 characters

7.2. Command Message

This is sent from the server to the QRSS-RX in response to a poll message.

All fields are 32-bit words sent in network byte order (high byte first)

Command Message ID (11)
Message Protovol Version (1)
Command – 0=stop 1=start
NCO increment value
Test Osc state – 0=off 1=on

7.3. Data Message

This is sent from the QRSS-RX to the receiver when it is running (started)

All fields (except samples) are 32-bit words sent in network byte order (high byte first)

Sample fields are in native format which is little-endian words, Q-data followed by I data. Each word is 32 bits, signed.

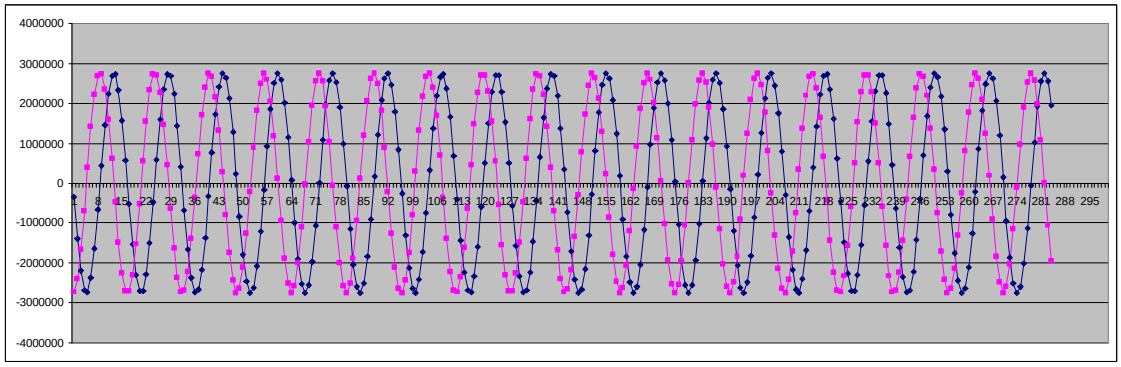
Data Message ID (12)
Timestamp – the contents of the timer at the start of the first sample
64 sets of samples

8. Results

A very simple server test application was built up. It simply displayed received messages, and sent a start command message which operated the QRSS-RX close to the test oscillator frequency, with the oscillator on.

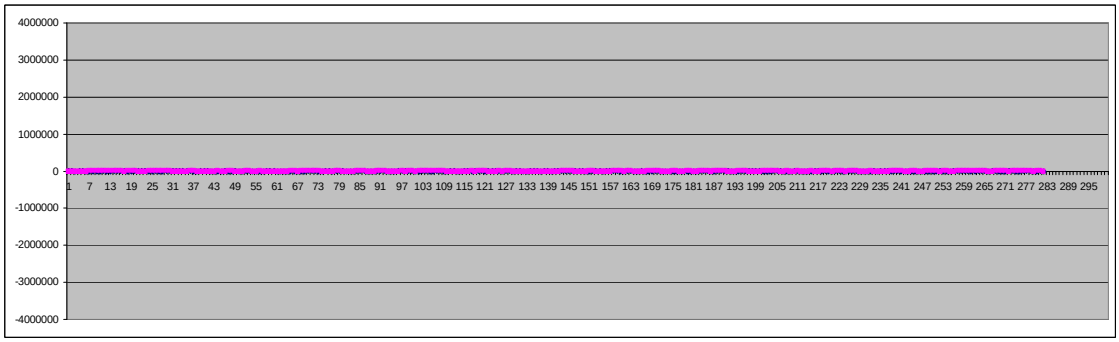
This application was written in C and compiled on a PC using gcc under cygwin.

Data output from this application was captured and copied into Excel where it was then displayed. The resulting graphic is:



Clearly seen here is the sine wave output which is the down converted test oscillator signal.

With the test oscillator off, this is the result:



The QRSS-Rx is clearing producing the expected data. Further processing of the samples would allow the generation of a grabber spectrum image.

9. Conclusion

The mbed environment proved to be an excellent platform. The available libraries made the task a lot simpler – effort could be put into the core task rather than needing to code Ethernet stacks and the like. The processing power of the CPU proved to be useful, allowing the FIR filters to be operated easily which still performing the necessary comms and GPS handling.