How FSQCall and WSQCall Work

A view under the hood of these fascinating modes

Introduction

This paper describes how the FSQCall and WSQCall selective calling and command protocol works. The paper mostly refers only to FSQCall, as the details also apply to WSQCall, which uses a sub-set of the FSQCall commands. Where WSQCall is different, this will be mentioned.

FSQ means Fast Simple QSO, and that's what this mode is all about. WSQ means Weak Signal QSO, and is the much slower LF/MF version. At the simplest level, FSQ/WSQ allows you to talk to your radio friends from your computer in a very straightforward manner. Just like Skype™ Chat or phone text messaging, you simply type a sentence, and press Enter.

But that's not all! Oh no! FSQ/WSQ includes a very powerful ***Directed*** mode, which allows you to send your sentences to specific stations, not just everybody. In Directed mode you can send pictures, files (even with error correction), Alerts, ask the other station for a signal report or other station information. This mode can also be used for telemetry. (WSQCall can only send text and a few simple commands).

The other guy need not be in the shack when you send him a sentence or file – he'll see it when he next walks by the computer. This is a great mode for busy folk, and also for those with poor hearing; and if you're a slow typist, nobody will know!

FSQ seems to send very slowly (typically four tones per second), but don't be fooled – because of the way it is designed (efficient alphabet, many tones), it has a typing speed approaching 40 WPM. WSQ goes at 0.5 tones per second, about 5.5 WPM.

These modes operate slowly to gain advantages of narrow bandwidth (interference rejection, high sensitivity) and to defeat serious timing errors induced by ionospheric propagation. There are other tricks too. Performance is so good that you don't need error correction, and so FSQ and WSQ are very slick – faster than you'd expect, and no delays between transmissions.

What makes FSQ different most of all is that it is non-synchronous,[1] which makes it very tolerant of ionospheric timing errors. It doesn't matter what the sending speed is. This means one receiver works with all sending speeds, and is extremely tolerant of ionosphere-induced timing errors. FSQ/WSQ also use a most unusual incremental keying MFSK system,[2] which makes it drift-proof and even more tolerant of propagation errors.

WSQ is similar, but has an advanced synchronous decoder, which uses a cross-correlator to discover sync. Since it retains the non-synchronous decoder operating in parallel, WSQ is still very tolerant of timing errors, but can only receive one speed at a time.

---

[1] This means that each tone is reckoned to be valid after a certain time, not at a constant rate.

[2] MFSK means multi-frequency shift keying. 33 tones are used, one at a time, smoothly changing with no gaps. The incremental tone scheme we use is called IFK+.

Selective Calling

This is what this paper is about. It will explain what Selective Calling is, and how it works in FSQCall/WSQCall.

Selective calling has been around for a long time, way back in the RTTY days (1970s) and later in AMTOR, packet radio and PACTOR, where you could send messages to specific stations. There have been a number of equivalent commercial and military schemes. But FSQ/WSQ are now able to deploy selective calling reliably ***without requiring error correction***, which means that it's practical to use Selective Calling on HF without the delays and overheads of error correcting modes.

The military and diplomatic services have for many years made wide use of selective calling, typically using protocols such as ALE (Automatic Link Establishment) to find and make contact with other stations. The modem used to establish contact is generally not the same one used to subsequently pass traffic. For example, connection is made using ALE, and traffic is passed via SSB.

With FSQ/WSQ we have added a relatively simple text-based protocol on top of the IFK+ modem used by FSQ/WSQ, and we call this FSQCall (or WSQCall). It is considerably simpler to use than the military systems, and requires no special registration of stations or special ID numbers before use, and virtually no training is required. The equipment required is that found in every ham shack – a computer with sound card, and an HF SSB or VHF FM radio.[3]

There's a trendy term being bandied about these days: '*Mesh Networks*', and a Mesh Network is indeed what is formed once a few stations share the same frequency using FSQCall/WSQCall. Any station can contact any other station, several stations, or all stations, even relay to other stations, using a simple set of commands.

What makes FSQCall and WSQCall different?

The defining difference with an FSQ/WSQ network compared to commercial, diplomatic and military radio networks is that it is completely ***ad hoc***. Nodes (stations) can come and go at will, can change call signs at will, and can even use pseudonyms (for example in emergency deployments call signs such as 'townhall' or 'post4'). There is no pre-registration, no distribution of node definitions to other stations, and each node should always knows which nodes are active, via a process we call *Sounding*. The node ID is simply a radio station call sign.

An important factor in the design of FSQCall and WSQCall is that they are designed to operate *infrastructure free*. Performance does not depend on, and is neither enhanced by or limited by connection to the Internet, wifi, cellular phone, mains power or repeater systems. FSQCall can make use of a local network in a multi-seat station, but functions perfectly well stand-alone.

Every station received is logged, and a complete log of messages is also kept. A list is maintained on screen of all stations heard on the working channel. You can also use a third-party tool[4] to plot signal strengths of multiple stations over 24 hours.

[3] It's not quite that easy on LF/MF, where specialised transverters and amplifiers are needed.
[4] FSQplot or WSQplot, designed by the author.

In FSQ/WSQ, call signs and text are preferably sent in lower case, as this is faster and has half the error rate of upper case, since lower case involves only one symbol (as opposed to two) per character.

The Selective Calling Sentence

FSQCall (or WSQCall) is a simple text protocol, which sits on top of the basic FSQ/WSQ modem. Without this selective calling option, every stations sees every transmission, there is no logging (since call signs are not sent in the preamble), and no automated commands are possible.

Each transmission in the selective calling *Directed* (addressed mode) is called a *Sentence*, which must at least contain a header, and may contain a *Direction* (destination address), and a command, and may be followed by a payload (message).

There are hidden characters embedded in the sentence, at the beginning and the end, which are used to inform the modem what is happening. At the start, the sentence has this sequence:

`{SP}{CRLF}source_call:checksum`

That is, a space, a new line command, and the source information which will be described in a moment. The convention {...} is used to indicate a non-printing character.

The first space is not always decoded at the receiver. Its purpose is to provide a reference frequency for the incremental frequency keying (IFK+) decoder. The CRLF is important, as it provides a new line for the receiver and also acts as a start marker for the received sentence. It is the first of several markers that define the sentence structure. In FSQCall and WSQCall, CRLF is a single character, which performs the functions of both CR (ASCII 13) and LF (ASCII 10).

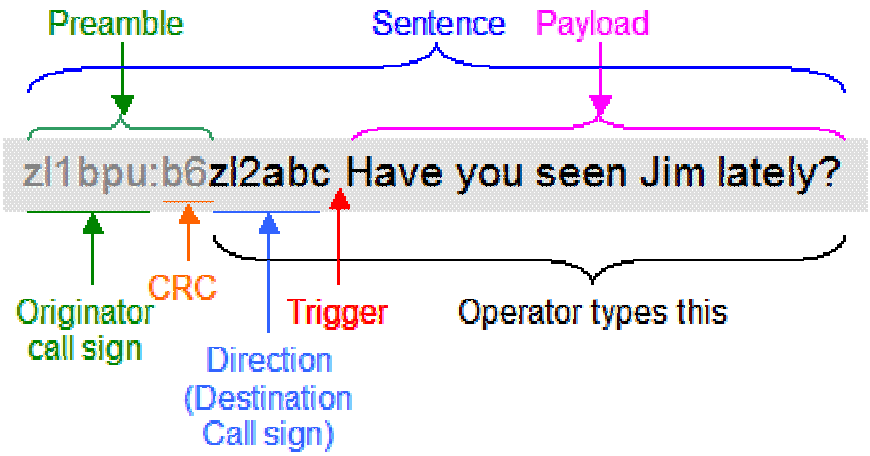

Fig.1. The FSQCall/WSQCall sentence structure

Figure 1 shows the complete sentence structure, but omits the ‘non printing’ characters before and after the sentence which were just discussed.

The Header

The header starts with the originating station callsign, then a second marker (we use a colon ‘:’, and then a checksum, which is calculated from the originator call sign. This allows receiving stations to determine and verify who is transmitting. If the verification

fails, the sentence may be ignored, as the station cannot be sure whom the message is from, or who to reply to, if a reply is required.

The reason for this second (colon) marker is that the originator call sign can be any length, from one to as many characters as needed, so using a fixed number of characters is impractical. The software knows that the source call sign consists of all the characters between the CRLF and the colon. Clearly a callsign cannot include a colon.

After the colon (Figure 1), we have two characters that define the checksum. It is a 16-bit hexadecimal number calculated from the call sign, and is expressed in two numbers or letters, '00' to 'ff'. It's not terribly secure, but quick and adequate for the purpose.

The next item in the header is an optional *Direction*, or address, a destination call sign. This is the station to which the message is addressed. Again, the call sign can be any length, so again markers delimit it. The first marker is the above-mentioned colon: it's obvious that the destination call sign starts after the checksum, so three characters after the colon. The second marker is the *Trigger* character, which follows the Direction, and will be described in the next but one section.

There are quite a few different Triggers, which define different commands, so the program needs to search for them all. There is no need for a check sum on the destination call sign, as each station knows its own call sign, and can simply match what comes in for an exact match. To exclude sub-sets (zl1ee and zl1ee/2 for example), a length check is also made. If the address and the station's defined call sign don't match, the message is ignored. Either it is intended for someone else, or there's an error in the address.

The Trailer

The end of each sentence is marked by a few characters, typically spaces and a Backspace character {BS}.

`{SP}{SP}{BS}{SP}{SP}`	(FSQCall)
`{SP}{BS}`	(WSQCall)

The {BS} cannot be sent from the keyboard, and is a reserved character. Its function is to close the receiver squelch quickly to prevent printing of junk text after the sentence.

Trigger Commands

There is no reason to describe all the commands here, as they are well described in the Help information provided with the program. They are listed later, but just two will be described in detail: the simple text sentence and the query.

The Triggers are always just one letter. There are 16 in FSQCall, and only seven in WSQCall. Each one elicits a specific automated response at the receiver of the addressed station. If the trigger character is a space, the whole payload will be copied and placed on the receive pane.

If the trigger is a '?' character, a 'query signal quality' command is performed. The software will read the station's Signal to Noise Ratio (SNR), will assemble a reply sentence, append the signal report, and then send it. So for example:

zl1ee-2:31zl2ee?

Will see this returned from the station zl2ee:

zl2ee:41zl1ee-2 snr =-21

Notice that the returned message is using the space as its trigger, so simply places the text on the screen. Couldn't be simpler. Here's a real example, captured by a WSQCall debugging program:

Figure 2. A complete captured sentence

It is identical to what was just described. The 'z' on the end is an extraneous character that occurred when the decoder met noise at the end of the transmission. It wasn't printed.

The Parser

This very complex bit of logic has the task of understanding each incoming sentence. While what has just been described seems simple enough, the coding involved is far from simple. The main reason for the complexity is that the Parser must reliably handle the myriad exceptions to the syntax just described.

What should it do if the originator call sign is correct, but the check sum is not? How is it to know? What if it recognises more than one trigger or call sign in the payload? What happens if the Trigger character is unrecognised? Every permutation needs to be explicitly handled if the program is not to give unexpected or incorrect results.

One interesting factor that the Parser is able to handle is that of multiple addressees. For example the following sentence:

```
zl2ee:41zl1ee zl1qm Murray and Graham, are you about?
```

Will be correctly handled at both addressed stations, the print at each station starting at their own call signs. This usage with other trigger commands is limited, as there could be undetermined results or multiple replies triggered.

If the originator wishes to address all stations on the channel, he can address 'allcall' and all stations will print, since 'allcall' acts as an alias address at all stations. None of the commands with automated responses operate with 'allcall' as the destination, in order to prevent mayhem on the channel!

The Parser identifies and verifies the originator call sign, the destination call sign, indicates whether it is for the receiving station; it identifies the Trigger and the payload.

The Command Processor

Once the Trigger character in an arriving sentence has been identified, the Command Processor takes over and performs one of several tasks within its repertoire. These can include placing the payload on the receive screen, saving the payload as a file, or retrieving one of several stored messages for transmission.

The simplest of these is the signal report query, where the SNR of the last station is recovered, and included in a reply sentence.

The relay command (!) and repeat command (+) (WSQCall only) retrieve the last received message, or the last transmitted message, and send it out again.

Reply sentences have their headers assembled from stored information (originator's call sign) and information recovered by the Parser, the originator call sign.

Here's a brief list of the commands offered:

SP	F,W	Receive and display this sentence
?	F,W	Query signal quality (SNR)
$	F,W	Query Heard List
@	F,W	Query QTH message
&	F,W	Query QTC message
^	F	Query version number
_	F	Query squelch setting
<	F	Decrease sending speed
>	F	Increase sending speed
*	F	Activate Call mode
#	F	Send file
+	F	Request file
+	W	Repeat last message
|	F	Send Alert (message in big box on screen)
!	F,W	Relay message
~	F	Send delayed relay
%	F	Send or request image

In this list 'F' signifies FSQCall; 'W' signifies WSQCall.

These one-letter commands (left column) are of course the trigger characters for the Command Processor.

The simplest message has no Trigger command and usually no payload. It consists of only the Header – originator call sign, colon and checksum. This is what we call a 'Sounding'. It can be transmitted by simply pressing ENTER while the cursor is in the TX pane of the software, and can be sent periodically by a timer in the software. The purpose of the Sounding is to provide an entry in the Heard List and Log at all stations on the channel. The Heard Log is pivotal to monitoring propagation.

These Soundings do not usually print on the main RX pane.

Some other commands can carry unexpected payloads. For example, you can combine the Sounding with a query (?) and even add a text payload. It will print at the recipient station, and of course if the command was a query, the station will reply with a signal report.