The CHIP64 digital mode

by Antonino Porcino, IZ8BLY
iz8bly@yahoo.it

Introduction

The selection of the baudrate figure to adopt for a particular HF digital mode, is the result of a compromise between sensitivity and stability. In fact, while low baudrates guarantee low S/N thresholds, they suffer a great phase instability because of the propagation disturbances present on the HF channels. Indeed, at low baudrates, small variations in the ionosphere produce abrupt phase jumps that interfere with good reception. By increasing the baudrate, the effect of the ionosphere is diminished but the noise level increases thus requiring more power to establish the contact at the same conditions.

Spread spectrum allows to overcome this compromise obtaining the noise levels of low speeds and the stability of high baudrates. When it comes to Spread Spectrum, people usually think of bandwidths too wide to be used on the HF, and maybe for that, the field has remained widely unexplored as regards amateur communications.

I propose the design of a Spread Spectrum system reduced to fit to the HF channels that can provide similar or better performances than actual narrowband modes. A brief description of this design as well its software implementation is discussed below.

What is Spread Spectrum

Spread spectrum (SS) is a family of modulation techniques in which the transmitted signal occupies a bandwidth greatly wider than normally required to transmit the bare information. The are several reasons for doing this, ranging from exploiting maximum available bandwidth to making transmission "hidden".

Among the different possible implementations of SS, Chip64 uses the so called *Direct Sequence Spread Spectrum* (DSSS). In a DSSS transmission, the low speed signal containing the data bits to be transmitted is mixed (multiplied) with a greatly higher speed signal called *code*. The result of this mixing operation, named *spreading*, is a high-speed bit stream which is then transmitted as a normal D-BPSK. Indeed, a DSSS signal looks like nothing else than a wideband BPSK.

Considering the fact that data can be get back by the inverse operation, the mix of the data with the code has the purpose of transferring on the data stream some of the properties of the code. These properties are then exploited in the receive process allowing a different and advantageous decoding system than traditional BPSK.

The code used in the spreading process is a sequence of bits which has the properties of *randomness*, *autocorrelation* and *cross-correlation*.

Randomness is the property to look like a random bit sequence, with no recurring pattern. It guarantees that the output power of the signal distributes evenly over all the bandwidth, generating a low density power spectrum. On the contrary, a bit sequence with recurring patterns generates spectrum spikes (for example, in PSK31 the idle signal "0000…" generates a spectrum made of only two main peaks). The advantage of having a power spectrum well distributed over the whole bandwidth is essential to go below (or at least to keep close to) the S/N level, thus passing "unobserved" and causing minimal interference to narrowband transmissions. Indeed, within its receive window, a narrow band mode listener notices only a slight increase of white noise when a spread spectrum station is transmitting.

The property of autocorrelation is the most important as it allows to identify the code within the arriving bit sequence. With a simple operation of multiplication, called *despreading*, it's possible, instant by instant, to determine if the code is presenting on the receive bit stream. Despreading operations are done in a block called *correlator*, whose output indicates the likelihood that the code has arrived in that particular instant. Codes of greater length have better autocorrelation properties with less chance of false guesses.

With the use of the correlator, there is no need to maintain the synchrony with the transmitter and it's unnecessary to recognize the single incoming symbol. Often indeed, symbols are below the noise threshold and it's not possible to distinguish them. The correlator can instead identify the whole group of symbols that form the code (called *chips*) indicating the exact instant they present to the receiver. The position of the single chip within the code can be later derived by offset, assuming that the transmitter has maintained the synchrony at least for all the duration of the code (which is a reasonable hypothesis).

The property of cross-correlation, is the capability of the code to be distinguished from other possible codes employed. The use of different codes that are also *orthogonal* allows, to a certain extent, several Spread Spectrum signals to coexist on the same channel at the same time, if each one uses a different code. Thus, a good cross-correlation guarantees a minimal interference among the different spread spectrum signals that are sharing the same band. In Chip64 the use of codes is exploited in a rather different way: instead of allowing more transmissions on the same channel, all available codes are assigned to only one station, allowing it alone to exist on the channel, but nonetheless giving it a greater data speed. This choice is justified by the fact that HF hamradio bandwidths are too narrow and allow a very limited throughput when spread spectrum is used.

CHIP64 design

The design constraints that have been followed consider several factors: the maximum baudrate speed allowed on HF (300 baud); the average user keyboard typing speed (evaluated around 30 bps, as in PSK31); the common CPU speed in use among radio amateur stations (approximately 1 GHz at the date of this writing).

After evaluating all the various parameters during laboratory and on-air tests, the length of the spreading code was fixed in $n=64$, reserving a further $n=128$ for slower but more reliable transmissions. Every spreading code has been conceived to be long exactly one data bit, so after the spreading operation a data bit will be 64 chips long.

Initially, a classical approach to DSSS was followed, with the use of *m-sequences* and *gold codes* (m-sequences are the basic codes with which all other codes are built). The result was a system with a data

speed too slow to be usable. Indeed, with 64 chips long gold codes and using D-BPSK at 300 bauds, a modest 4.7 bps throughput is achieved; a speed which is too lower than our compare term of 30 bps.

The next step was to use more gold codes at the same time as to form an alphabet, but discovering too soon that there weren't enough 64 chips long gold codes. Indeed, the number of gold codes, and their goodness, is very limited for n=64.

The solution came with the use of *Walsh-Hadamard* codes, already known to radio amateurs because also used in MT63. These codes are a full set of 64 codes which are mutually orthogonal (each one is easily discernible from all the others) but that have a very poor autocorrelation. Indeed, Walsh-Hadamard codes are highly repetitive and contain a great number of recurring patterns. While in MT63 the bits of the code present in parallel to the receiver (one on every of the 64 tones), in our DSSS system they present serially, therefore good autocorrelation properties are required to make possible the identification of the starting point of the code.

Walsh-Hadamard codes can be improved in autocorrelation by mixing them with an m-sequence. A new set of codes, which I named WHP codes (Walsh-Hadamard-Porcino), can be thus derived by simply multiplying each code with a generating m-sequence. The new codes show features of autocorrelation and orthogonality of their "parent" codes and they can be used to form an alphabet that encodes 6 bits words (2^6=64).

With the purpose of further increasing the total bit rate which was still too low, it was chosen to use two tables of WHP codes, each one obtained by multiplication with a different m-sequence. While this operation lowers the quality of the codes, it permits to double the data speed.

Each of the WHP codes can be also transmitted with positive or negative polarity that can be directly detected by the correlator, providing one additional data bit. In total, we have a WHP codes alphabet which encodes 8 bit words. The total data rate is calculated as:

$$datarate = \frac{chiprate}{codelength} \times nbits = \frac{300}{64} \times 8 = 37.5 bps$$

The parameters of the system can be summed in the following table:

Mode	Chip 64	Chip 128
Modulation	D-BPSK @300 chips/s	D-BPSK @300 chips/s
Code length	64 chips	128 chips
Alphabet length	128 (2x64 WHP codes)	256 (2x128 WHP codes)
Generating m-sequences (zero padded at end)	$M_{63}=x^6 + x^5$ $M_{63}= x^6 + x^5 + x^2 + x^1$	$M_{127}=x^7 + x^3$ $M_{127}= x^7 + x^3 + x^2 + x^1$
Data bits encoded by the alphabet	8 bits	9 bits
Total data rate	37.5 bps	21.09 bps

Rather than fixed 8-bit ASCII, a variable-length encoding (also known as *Varicode*) is used for characters. The Varicode has two functions: first it provides a sort of Huffman compression when lowercase English language is used. Second, it gives a character-to-character synchronization because each character has its ending delimiter, making possible faster recovers without needing an higher level layer of synchronization. The Varicode system featured in Chip64 is the enhanced version that also

appears in MFSK16, which provides a more accurate probability distribution among letters and a reuse of bit sequences containing more than two zeros ("00").

The whole message transmission is also delimited with special non-printable characters allowing both station identification and message framing. The format is the following:

<SOH> *callsign* <STX> *message* <EOT>

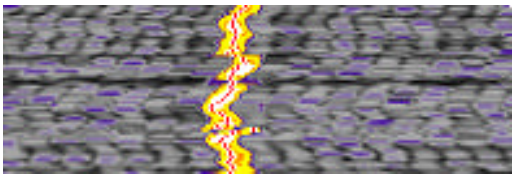
Where <SOH>, <STX> and <EOT> are respectively varicode ASCII character #1, #2 and #4. <NUL> (#0) is used to flush the buffer at start/end of the transmission and as fill-in during keyboard empty buffer times.

The transmitter / receiver software

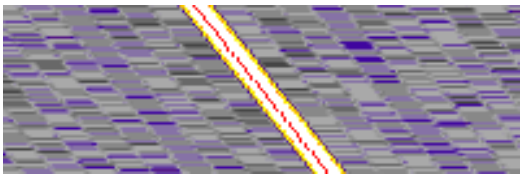
Chip64 signals can be transmitted and received with a specific client software written for Windows and soundcard. It can be freely downloaded from:

http://www.geocities.com/iz8bly/Chip64

The program features a special display called *correloscope*. It shows the presence of the code by displaying its time-offset. The correloscope plots in a graphical way the output of the correlator. On the horizontal axis, the time frame of a code-length is represented (1/300 sec). At each sampling step, one dot is drawn: the more the correlation confidence, the whiter the dot plotted. By plotting the correloscope in a waterfall fashion, the user is able to perceive how the offset of the received codes moves over time. A disturbed signal will plot an unsteady vertical trace (where the shape of this trace depends on the propagation conditions). A weak signal will plot a steady but faint vertical trace. The correloscope can also show possible clock drifts between the transmitter and the receiver, which are seen as slanted traces.



CCIR Poor



+0.1% clock drift

Fig. 1 the correloscope trace under two different conditions

The output of the correlator also controls the squelch of the receiver which is turned off when the confidence falls below a certain threshold.

The software has also the option to turn off the correlator and demodulate the signal as a traditional BPSK transmission (referred in the program as "clocked decoding"). Its purpose is to show the differences between the two decoding systems under different conditions.

By averaging chips phase errors, a very smooth frequency tuning is possible. This is indeed another advantage of Spread Spectrum. The averaged phase errors, constitute a stable frequency error correction signal that can be put in feedback to the main NCO allowing an automatic and precise frequency tuning.

The correlator

In the decoding process, each receiving sequence has to be matched with the WHP codes from the alphabet, that is, the despreading process has to be repeated for every WHP code. This is necessary because in this system we haven't just a single code as in plain DSSS, but multiple codes present serially and randomly at the receiver.

Thus, each chip from the arriving sequence is compared with the corresponding chip of the reference WHP code in the table. If the chip sign matches, confidence is augmented; otherwise if the sign doesn't match, confidence is decreased. At the end of the chips, the resulting confidence for that code is obtained and the matching work is repeated for all the other WHP codes in the table.

The time-offset of the arriving code sequence is calculated as the point with best confidence over a code duration, and the corresponding code is used for decoding into text.

For better performances, *soft* chips are processed. A soft chip is a chip that does not represents discrete states (+1 or -1) but represents the likelihood of the chip of being "+1". This likelihood system provides a better accuracy giving a continuous range in the confidence values, rather than discrete steps of 1/64. The soft bit information is derived directly from the phase error output of the BPSK detector. Soft bit values range linearly from 0.0 (0° angle) to 1.0 (180° angle).

To avoid in between chip synchronization, the despreading work has to be done in time steps smaller than one chip, thus picking with greater precision the offset point of the code. This is done by working with an oversampled signal respect to the chiprate. Depending on the CPU speed, three oversampling figures are provided: 21x, 7x and 3x; these apparently odd figures, are all divisors of 44100 Hz, the samplerate used by the soundcard. An higher oversample gives to the correlator a better resolution, but has an heavier load on the CPU. The CPU load is an actual limitation, as it can be too high to let the program run on some old PC workstations. The load is proportional to the number of elementary chip matching operations that can be executed in the time unit. Since these elementary operations translate into simple ADD and SUB register machine code instructions, they can be executed in just one CPU-clock cycle, therefore the total CPU load (lower bound) can be approximately calculated as:

$$nop = chiprate \times oversample \times chiplength \times ncodes$$

For Chip64: $nop = 300 \times 21 \times 64 \times 128 = 51{,}609{,}600 \approx 50$ *Mips*
For Chip128: $nop = 300 \times 21 \times 128 \times 256 = 206{,}438{,}400 \approx 200$ *Mips*

With current CPU clocks in the range of few GHz, it's necessary that the correlator has the maximum possible performance. For this reason, integer math was used as it guarantees faster execution speeds than floating point math. In particular, the correlation *routine* was written in pure assembler doing all operations in CPU registers and interlacing the machine language instructions to take advantage of the processor's execution pipeline.

Performance of the system

Chip64 and Chip128 have been extensively tested on air during the year 2004 by me and a restricted group of people: Murray Greenman ZL1BPU (who also contributed in the design of the system), Chris Gerber HB9BDM and Manfred Salzwedel OH/DK4ZC.

The system proved to be efficient and we found it comparable to the other modern digital modes. Being totally different in its architecture, it shows better performance during certain circumstances, while in others it shows no actual gain. In particular, it performs better under multipath where normal BPSK can't track arriving symbols, but in quiet environments it doesn't show any improvement over plain BPSK. This is expected because of the losses that occur due to the imperfect autocorrelation of the codes.

Conclusions

The design of this new digital mode served to introduce the Spread Spectrum technology among radio amateurs by providing a communication tool to experiment with. Its purpose was to prove that it's possible to take advantage of the Spread Spectrum techniques even on the HF channels, making the communication possible under conditions where traditional narrowband modes fail.

All accurate performance tests and possible fields of application are left for future studies.

If you are interested in the Chip64 project you might want to join the Chip64 mailing list at the address: chip64@yahoogroups.com.