

POCSAG

The following summary describes the coding used on POCSAG pager signals and may be of interest to those curious about what those ear-splitting beeps and buzzes mean and how they encode data. This summary is based on a very old text of the standard from my files; the current text of the POCSAG standard is available as CCIR Radiopaging Format 1.

Before you read further, you may want to hear what POCSAG sounds like:

Note that some current POCSAG signals (so called Super-POCSAG) transmit paging at 1200 or 2400 baud instead of the 512 baud I refer to here, but use essentially a similar coding standard.

The interested USA reader is reminded that willfully intercepting other than tone only paging is a violation of the ECPA with similar penalties and criminal status to willfully intercepting cellular phone calls.

The interested reader is advised that at least two of Universal Shortwave's RTTY reading devices (the M8000 and the new C-400) are capable of reading at least the older 512 baud version of POCSAG paging, so commercial devices for this purpose are currently being sold in the USA.

And finally, much alphanumeric paging - particularly that installed some time ago, uses a proprietary Motorola encoding format called GOLAY which is quite different from POCSAG. The two can be told apart by their baud rates - GOLAY is 600 baud.

POCSAG

First POCSAG stands for Post Office Code Standardization Advisory Group. Post office in this context is the British Post Office which used to be the supplier of all telecommunications services in England before privatization.

POCSAG as defined in the standard, (original POCSAG) is 512 bits per second direct FSK (not AFSK) of the carrier wave with ± 4.5 kHz shift (less deviation than that is used in some US systems). Data is NRZ coded with the higher frequency representing 0 (space) and the lower one representing 1 (mark).

The basic unit of data in a POCSAG message is the codeword which is always a 32 bit long entity. The most significant bit of a codeword is transmitted first followed immediately by the next most significant bit and so forth. The data is NRZ, so that mark and space values (plus and minus voltages) as sampled on the output of the receiver discriminator at a 512 Hz rate correspond directly to bits in the codeword starting with the MSB. (Note that the audio output circuitry following the discriminator in a typical voice scanner may considerably distort this square wave pattern of bits, so it is best to take the signal directly off the discriminator before the audio filtering).

The first (msb) bit of every POCSAG codeword (bit 31) indicates whether the codeword is an address codeword (pager address) (bit 31 = 0) or a message codeword (bit 31 = 1). The two codeword types have different internal structure.

Message codewords (bit 31 = 1) use the 20 bits starting at bit 30 (bit 30-11) as message data. Address codewords (bit 31 = 0) use 18 bits starting at bit 30 as address (bits 30-13) and bits 12 and 11 as function bits which indicate the type and format of the page. Bits 10 through 1 of both types of codewords are the

bits of a BCH (31,21) block ECC code computed over the first 31 bits of the codeword, and bit 0 of both codeword types is an even parity bit.

The BCH ECC code used provides a 6 bit hamming distance between all valid codewords in the possible set (that is every valid 32 bit codeword differs from every other one in at least 6 bits). This makes one or two bit error correction of codewords possible, and provides a robust error detection capability (very low chance of false pages). The generating polynomial for the (31,21) BCH code is $x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1$.

Codewords are transmitted in groups of 16 (called batches), and each batch is preceded by a special 17th codeword which contains a fixed frame synchronization pattern. At least as of the date of the spec I have, this sync magic word was 0x7CD215D8.

Batches of codewords in a transmission are preceded by a start of transmission preamble of reversals (101010101 pattern) which must be at least 576 bits long. Thus a transmission (paging burst) consists of carrier turnon during which it is modulated with 512 baud reversals (the preamble pattern) followed by at least 576/512 seconds worth of actual preamble, and then a sync codeword (0x7CD215D8), followed by 16 data/address codewords, another sync codeword, 16 more data/address codewords and so forth until the traffic is completely transmitted. As far as I am aware there is no specified postamble. I believe that all 16 of the last codewords of a transmission are always sent before the carrier is shut off, and if there is no message to be sent in them the idle codeword (0x7A89C197) is sent. Later versions of the standard may have modified this however.

In order to save on battery power and not require that a pager receive all the bits of an entire transmission (allowing the receiver to be turned off most of the time, even when a message is being transmitted on the channel) a convention for addressing has been incorporated which splits the pager population into 8 groups. Members of each group only pay attention to the two address code words following the synch codeword of a block that correspond to their group. This means that as far as addressing is concerned, the 16 codewords in a batch are divided into 8 frames of two codewords apiece and any given pager pays attention only to the two in the frame to which it is assigned.

A message to a pager consists of an address codeword in the proper two codeword frame within the batch to match the recipient's frame assignment (based on the low three bits of the recipient's 21 bit effective address), and between 0 and n of the immediately following code words which contain the message text. A message is terminated by either another address code word or an idle codeword. Idle codewords have the special hex value of 0x7A89C197. A message with a long text may potentially spill over between two or more 17 codeword batches.

Space in a batch between the end of a message in a transmission and either the end of the batch or the start of the next message (which of course can only start in the two correct two codeword frame assigned to the recipient) is filled with idle codewords. A long message which spills between two or more batches does not disrupt the batch structure (sync codeword and 16 data/address code words - sync code word and 16 data/address codewords and so forth) so it is possible for a pager not being addressed to predict when to next listen for its address and only turn on its receiver then.

The early standard text I have available to me specifies a 21 bit address format for a pager (I believe this has been extended since) with the upper 18 bits of a pager's address mapping into bits 30-13 of the address codeword and the lower 3 bits specifying which codewords within a 17 codeword batch to look at for possible messages. The address space is further subdivided into 4 different message classes as specified by the function bits (bits 12 and 11 of a codeword). These address classes correspond to different message types (for example bits 12 and 11 both zero indicate a numeric message encoded in 4 bit BCD, whilst bits 12 and 11 both set to 1 indicate an alpha message encoded in 7 bit ASCII). It was apparently envisioned that a given pager could have different addresses for different message types.

There are two message coding formats defined for the text of messages, BCD and 7 bit ASCII. BCD encoding packs 4 bit BCD symbols 5 to a codeword into bits 30-11. The most significant nibble (bits 30,29,28,27) is the leftmost (or most significant) of a BCD coded numeric datum. Values beyond 9 in each nibble (ie 0xA through 0xF) are encoded as follows:

0xA	Reserved (probably used for something now - address extension ?)
0xB	Character U (for urgency)
0xC	" ", Space (blank)
0xD	"-", Hyphen (or dash)
0xE	")", Left bracket
0xF	"(", Right bracket

BCD messages are space padded with trailing 0xC's to fill the codeword. As far as I know there is no POCSAG specified restriction on length but particular pagers of course have a fixed number of characters in their display.

Alphanumeric messages are encoded in 7 bit ASCII characters packed into the 20 bit data area of a message codeword (bits 30-11). Since four seven bit characters are 21 rather than 20 bits and the designers of the standard did not want to waste transmission time, they chose to pack the first 20 bits of an ASCII message into the first code word, the next 20 bits of a message into the next codeword and so forth. This means that a 7 bit ASCII character of a message that falls on a boundary can and will be split between two code words, and that the alignment of character boundaries in a particular alpha message code word depends on which code word it is of a message. Within a codeword 7 bit characters are packed from left to right (MSB to LSB). The LSB of an ASCII character is sent first (is the MSB in the codeword) as per standard ASCII transmission conventions, so viewed as bits inside a codeword the characters are bit reversed.

ASCII messages are terminated with ETX, or EOT (my documentation on this is vague) and the remainder of the last message codeword is padded to the right with zeros (which looks like some number of NULL characters with the last one possibly partial (less than 7 bits)).

The documentation I have does not specify how long a ASCII message may be, but I suspect that subsequent standards have probably addressed the issue and perhaps defined a higher level message protocol for partitioning messages into pieces. The POCSAG standard clearly does seem to allow for the notion of encoding messages as mixed strings of 7 bit alpha encoded text and 4 bit BCD numerics, and it is at least possible that some pagers and paging systems use this to reduce message transmission time (probably by using some character other than ETX to delimit a partial ASCII message fragment).

Brett Miller N7OLQ

Intel Corp.
American Fork, UT
DoD#1461

[<- Back](#)[Forward ->](#)[Remote](#)[Mail Me](#)[Search](#)

Back To Decoder Page | [Mail Me](#)