

Win500 Remote Control & Monitoring Protocol

1. Data types

- a. **char**: 8-bit unsigned value, 0x00-0xFF
- b. **short**: 16-bit unsigned value, 0x0000-0xFFFF
- c. **long**: 32-bit unsigned value, 0x00000000-0xFFFFFFFF
- d. All multi-byte values are sent "little-endian" (least-significant byte sent first)

2. Connection to server

- a. TCP connection to IP : Port specified in Win500 server
- b. Data sent as "stream of bytes" (packetization is above the transport layer)

3. Packet structure

- a. **char**: **Command**: value from list of commands below
- b. **short**: **Length**: *total* size of packet, in bytes
- c. **long**: **Timestamp**: sender's "millisecond timer", for syncing received packets to real time at receiver
- d. **char[]**: **Payload**: <Command>-dependent bytes
- e. **short**: **Checksum**: sum of all previous bytes (starting with <Command>)

4. Commands

a. 'S' (0x53) : Status

Payload length: 16

Payload[0] = current scanner mode:

- 0: Tune
- 1: Scan
- 2: Manual
- 3: Program
- 4: Search
- 5: Weather
- 9: Sweeper/Stalker

Payload[1] = Flags (bitmap)

- bit 0: squelch open
- bit 1: audio unmuted

Payload[2,3] = **short** indicating battery level A/D value

Payload[4,5] = **short** indicating RSSI value

Payload[6,7] = **short** indicating "zeromatic" value

Payload[8] = RED LED value

Payload[9] = GRN LED value

Payload[10] = BLU LED value

Payload[11,14] = currently-tuned frequency, in Hz

Payload[15] = current rx mode: AM=0, FM=1, NFM=2

b. 'L' (0x4C) : LCD Data – full

Payload length : 66

Payload[0,63] = LCD text, top-left to bottom-right

Payload[64] = LCD Icons (bitmap)

bits 0-2: signal bars, 0-5

bit 3: 'S'

bits 4-5: battery

Payload[65] = LCD Icons (bitmap)

bit 0: 'F'

bit 1: 'G'

bit 2: 'A'

bit 3: 'T'

bit 4: <up arrow>

bit 5: <down arrow>

bit 6: backlight on

c. 'd' (0x64) : LCD Data – compressed

Payload length : variable, from 10 through 74

Payload[0] = LCD icons, as above for Payload[64]

Payload[1] = LCD icons, as above for Payload[65]

Payload[2..9] = bitmap of 64 LCD text positions in remainder of Payload[.].

Bit 0 of Payload[2] is 1st character of 1st row, bit 1 is 2nd character of 1st row, ..., bit 7 of Payload[9] is 15th character of 4th row.

Payload[10..<variable>] = one byte for each '1' bit in Payload[2..9]

representing the character at a particular position. For example, if there were ten '1' bits in Payload[2..9], then this field would be ten bytes long (total payload length would be 20 bytes).

Example: Payload[2..9] == [0x05, 0, 0x80, 0, 0, 0, 0, 0]. Payload[10] contains a character for LCD row 0, column 0; Payload[11] contains a character for LCD row 0, column 2. Payload[12] contains a character for LCD row 1, column 7. Total payload length is 13 bytes.

d. 'A' (0x41) : Audio samples, compressed

Payload length : variable, must be deduced from packet's **Length** value

Payload[.] : audio sample data

e. 'a' (0x61) : Audio samples, uncompressed

Payload length : variable, must be deduced from packet's **Length** value

Payload[.] : audio sample data

5. Audio sample format

All audio samples correspond to Microsoft's WAV format, 8000 samples per second, 8 bits per sample, monaural. Uncompressed data is plain PCM, while compressed data is GSM6.10. Each Payload[.] starts with a [Microsoft WAVEHDR structure](#); the remainder of the payload is the sample data. The "pointer" members of the **WAVEHDR** structure are not valid.

It is presumed that the reader is familiar with whatever coding is required to “play audio” on his target platform / environment, so details are not specified here. However, on the Windows Mobile (“Pocket PC”) platform, I used the various *waveOut* APIs, such as *waveOutOpen*, *waveOutSetVolume*, *waveOutPrepareHeader*, *waveOutWrite*, *waveOutUnprepareHeader*, *waveOutReset*, and *waveOutClose*.

6. Keystrokes

Keystrokes are sent to the Win500 server using the same packet format as the above-described incoming data. The **Command** field is ‘k’ (0x6B) and there is one byte of Payload[]: the key value. The key values can be found in the PSR-500 manual, in Appendix A “Remote Control Protocol”.