

SDR TCP/IP INTERFACE

Protocol specifications Version 0.1 (modifications from previous version 0 in red)

Important: the previous version 0 (not used) of this protocol is considered as obsolete and must not be used.

Main modifications:

- Change of the TCP/IP port number,
- Extension of the server (MP) types,
- Removal of the hardware type 8 management,
- Removal of several not indispensable procedures: Set_HWSR Show_GUI Hide_GUI,
- Lists of mandatory and optional procedures modified.

Abbreviations

CHR(): means "Character corresponding to a given number between 0 to 255", for example CHR(65) corresponds to "A"

Client : worth for "client program", i.e. "CP"

CP: worth for "Client Program" (as Multipsk...) which hosts the "client" TCP/IP interface.

GUI: Graphical User Interface

HAM: Amateur radio operator

HW: SDR Hardware (receiver or transceiver)

Note: Transceiver=receiver (RX) + transmitter (TX) on the same HF frequency

LO: SDR hardware's Local Oscillator

MP: worth for "Master Program" (MP), i.e. a SDR program which interfaces the hardware through a socket in one side and the CP in the other side. The MP hosts the "Master" TCP/IP interface. **It is also called the "server"**.

ORD(): means "Number (between 0 to 255) corresponding to a character", for example ORD(A) corresponds to 65.

RX: "reception" or "receiver" or "receives",

Server: worth for "server program", i.e. "MP"

SWL: Short Waves Listener (monitoring all frequencies nowadays)

TX: "transmission" or "transmitter" or "transmits"

1. Introduction and general specifications

At the present time, for programs as coding/decoding programs or programs driving transceivers thanks to the CAT system (called, here, "Client Programs" or "CP"), SDR receivers and transceivers cannot be interfaced directly.

For this, it is necessary to pass through programs interfacing the hardware which are either general programs as SDR # or HDSDR or specific programs attached to one equipment.

There is no practical way to interface these equipments. EXTIO DLL (invented by Alberto I2PHD) exists but it is difficult (if not impossible) to have help to interface these DLL or information about the specific features of equipments or updated specifications. This interface system must be considered, in a practical way, as "closed".

However, with the development of SDR hardware, there is a repeated demand coming from users of these SDR equipments, for an easy exchange between a "SDR" software attached to a hardware and any "client" program (coding/decoding program for example).

Moreover, people developing coding/decoding of HAM programs are not, all, system programmers and consequently, don't have all, capabilities in C++. They are much more people close to electronics, transmissions or digital processing matters. As they don't have necessity to program systems, they can use other languages than C++ as C, Basic, Pascal, Fortran, Python etc..

This to say that it misses an interface easy to use for a non-system programmer, using any language.

Due to the fact that this protocol is simple, one point of interest is that HAM and SWL be able to program an MP interfaced to the CP, for their own homemade SDR. However it will remain the problem of the transmission of control/data from the SdR to the MP, which is outside of the scope of this protocol.

It would be also nice that this interface could be such that it could take profit of Internet, permitting a remote control of the receiver (or the transceiver) through Internet. The only answer is the TCP/IP link.

TCP/IP is a connected reliable protocol which goal is to transmit characters without errors. So the following protocol is based on a TCP/IP link between a "server" hosted by the MP (attached to the SDR equipment, called "HW") and a client hosted by the CP.

Note that the TCP/IP client or server on modern developing platforms (DELPHI, C++...) is just an object that can be programmed (without necessity to know how it works in detail).

Not only programs having a SDR capacity (as Multipsk...) to demodulate I/Q data, must be allowed to use this interface, but all the other programs, having or not having this capacity. So it is expected that the MP (server), if having this demodulation capacity (in USB, baseband), transmits the output of its demodulation in the common form of a real audio signal (let's say "I") within a bandwidth of 22 KHz (without alias) at a 48 KHz sampling frequency with 16 bits samples, in the same form as sound card samples (little Endian, i.e. weakest byte first). In fact, the MP will act exactly as a standard sound card for the CP.

The CP will have only to specify a LO (local oscillator) frequency to receive the demodulated audio data (real signal). **This possibility is not implemented in this version, but could be in the future if someone is interested.**

The first version of this protocol will only manage SDR receivers (RX).

In a next version, the protocol could include SDR transceivers (RX/TX) if a need appears.

2. General working

The server of the MP (attached to the SDR equipment) is waiting for TCP/IP connection (**port 50001 by default**) when the CP takes the initiative to connect the server.

The range of TCP/IP ports from 50001 to 50020 is associated to this protocol. Each port number can be associated with one instance of the CP (Multipsk...). So 20 instances of the CP, working in different frequencies and modes, could be connected to the SdR (if this one is able to manage multi-instances).

The server (MP) and the client (CP) can be in the same PC (linked through the 127.0.0.1 IP adress) or can be located in distant QRAs (linked through Internet using the IP or DNS of the PC hosting the server).

In version 0.1 of this protocol:

- it is mandatory for the CP to manage an MP in the same PC (as described in §2.3),
- it is optional to manage a distant server. In that case, this server will manage only one client.

2.1 Diagram of working

RX

SDR Receiver (IQ data)-->Server (MP)-->TCP/IP link-->Client (CP) for RX operation (decoding...)

SDR control

Client (CP) (frequency) -->TCP/IP link-->Server (MP) --> SDR XCVR

2.2 Tasks of the server and the client

The tasks of the server (MP) are:

- 1) at any time, to interface the SDR receiver through a socket as the USB one, extracting the IQ data and controlling the SDR receiver (the mean to do this is outside of the scope of this protocol),
- 2) as a server, to open a TCP/IP link so as to permit the connection from a client,
- 3) after the success of the connection, to transmit the IQ data to the client and receive control orders from the same client, through the TCP/IP link (in local loop of the PC or through Internet), using the protocol described here.

The tasks of the client (CP) are:

- 1) possibly, to start up the internal MP (server) if the TCP/IP link is done in local loop (in the same PC and in the same directory),
 - to connect the MP server through a TCP/IP link (in local loop of the PC or through Internet),
- 2) after the success of the connection, the most rapidly possible, to receive the samples from the server through the TCP/IP link and to control the HW, using the protocol described here,
- 3) to decode the samples (RX operation) or other task (outside of the scope of this protocol).

2.3 Starting up of the MP program (server) if TCP/IP link in local loop

The MP program, as an executable, can be of three main forms:

- an “external” (relatively to the CP directory) MP program being a sole server or a program integrating a MP. It is the default option.
- an MP (server) “internal” (called also “local”) to the CP directory which name is on the form SDR_TCP_IP_xyz.EXE. “SDR_TCP_IP_” and “.EXE” are compulsory, whereas “xyz” is the name defining the hardware and the author (for example: “RTL_FZ9XYZ”).
It must be located on the root of the CP program. The only present local (test) server is SDR_TCP_IP_TEST.EXE.
- if the MP is located on a distant PC, it is considered as a “remote” server and will be connected with an IP or a DNS address.

The CP must be able to give choice to the user between an external MP and different local MP (if any). Once the user has selected the server associated to the hardware, the CP starts up the MP (as TCP/IP server), if the MP is local. The MP configures, itself, in listening position

for a client connection. Then the CP, as TCP/IP client, connects immediately the TCP/IP server.

Once the connection is established, the dialog can begin.

In case of external (or remote) MP, the user must, first, starts the MP before the CP (Multiplik..).

2.4 Normal working of the CP and the MP

Once the TCP/IP link established, the MP and the CP exchange procedures and manage these procedures, according to this protocol. When it has finished, the CP disconnects from the MP and, possibly, close it (if the MP is **internal**). Reversely the server MP can disconnect the CP in case of problem on the HW (the CP returning to a standard working on the sound card, for example). In all cases, a Bug_on_P procedure can, possibly, be sent for information.

2.5 Maximum delay before disconnection (due to a program bug)

- The MP server must consider that after 15 seconds waiting for a CP event (end of transmission of a command or beginning of transmission of a new command after having transmitted Pversion), the CP has some problem (often as a freezing) and the MP will disconnect the CP. Just before the disconnection, it will be transmitted the procedure Bug_on_P with the parameter "Disconnection due to a too long waiting time".
- Reversely, the CP client must consider that after 10 seconds (different from 15 seconds to avoid a simultaneous disconnection) waiting for a MP event (end of transmission of a command), the MP has some problem (often as a freezing) and the CP will disconnect from the MP (and will return to a standard working on the sound card, for example). Just before the disconnection, it will be transmitted the procedure Bug_on_P with the parameter "Disconnection due to a too long waiting time"

To manage this delay, the CP and the MP set a timer to estimate the waiting time. Once the timer triggers (10 times/second is sufficient), the MP and the CP analyze the elapsed waiting time, compare it to 10 (or 15) seconds and, possibly, disconnect the other party, as explained above.

2.6 Behavior in case of protocol bug

The TCP/IP is not supposed to transmit wrong characters. **The sole possible errors can come from the applied protocol.** Consequently, on an error in the **applied** protocol, the link will be immediately stopped by a disconnection. Just before the disconnection, it will be transmitted a procedure Bug_on_P with, as parameter, "Incorrect command: " followed by the command concerned by the bug with its own parameters, in the total limit of 255 characters.

3. Description of the protocol

3.1 Generalities

3.1.1 Main functions

The main functions, for this type of interface, have been defined by Alberto I2PHD in his specifications for EXTIO DLL (www.winrad.org/bin/Winrad_Extio.pdf). A subset of these specifications will be the basis of these specifications (with authorization of Alberto). They are in general modified according to the general goals exposed above. Several new functions are added.

Some EXTIO functions are not used as StartHW and OpenHW which role is taken by InitHW. It's the same with StopHW and CloseHW which role is taken into account by the disconnection. SetCallback is replaced by TXHWdata (with all the status and the IQoffs variable ignored).

Set_HWSR, Show_GUI, Hide_GUI, IFLimitsChanged, Getfilters, GetMode, GetTune, Tunechanged, RawDataReady and GetStatus, GetHWLO, GetHWSR don't seem indispensable, at least in version 0.1.

Note: about GetMode: the digital mode to decode by the CP imposes a given demodulation mode (it's not an option), so GetMode has no interest.

GetTune and Tunechanged are not much interest either because the transmission must be exactly tuned on the CP spectrum and not in the MP spectrum if any. If needed, in a next protocol version, it will be re-incorporated these 3 previous functions (with an additive function SetMode).

Moreover, the hardware types 5, 6, 7, 8 are also ignored.

3.1.2 Type of data used

- A character (on 8 bits) is called "CHAR", for example "A".
- A byte (on 8 bits, 0 to 255) is a number called "BYTE", for example "65".
- A short string of characters is called "SHORTSTRING" and is used to transfer text. In this protocol, the SHORTSTRING is preceded by its length in characters expressed by a positive number on 8 bits (1 byte), i.e. from 0 to 255. For example, to transfer "HELLO", it will be sent: CHR(5) then "HELLO".
- A long string of characters is called "LONGSTRING". In this protocol, the LONGSTRING is preceded by its length in characters expressed by a positive number on 32 bits (4 bytes), i.e. from 0 to 4 294 967 295, and transmitted in little Endian (weakest byte first).
LONGSTRING are used to transfer data. For example, to transfer an IQ data, I and Q of 16 bits wide with a value of IQ data equal to "01 02 03 04" (4 bytes), it will be sent: CHR(4) CHR(0) CHR(0) CHR(0) then CHR(1) CHR(2) CHR(3) CHR(4).
Note: an empty LONGSTRING would be CHR(0) CHR(0) CHR(0) CHR(0).
- A WORD is a positive number on 16 bits (2 bytes), i.e. from 0 to 65 535, transmitted in little Endian (weakest byte first).
- A LONGWORD is a positive number on 32 bits (4 bytes), i.e. from 0 to 4 294 967 295, transmitted in little Endian (weakest byte first).

3.1.3 Transmission of numbers (WORD or LONGWORD)

For characters (8 bits) transmission or for bytes (8 bits) transmission, there is no need to consider if the transmission is done in little or big endian, as the unity of transfer of the TCP/IP is block of 8 bits.

However, for transmission of numbers on 16 or 32 bits, it must be considered if the transfer is done in little endian (weakest byte first) or in big endian (strongest byte first). In these specifications, implicitly, all transfers are made in little endian, by compatibility with I2PHD specifications.

3.1.4 Transmission through pseudo-procedures

The most simple for programmers is to exchange subroutines, through the TCP/IP link (a bit as a DLL). For this, it is chosen pseudo-procedures (close to Pascal procedures, or C/C++ functions with void return value). So the MP and the CP will exchange pseudo-procedures through the TCP/IP link. These ones will be simply called "procedures" in the following text.

3.1.5 About the presentation (display) of the procedures

A procedure, in the displayed form, is composed of a name, always 8 characters long (to simplify search of a command among a flow of characters), followed by a list of parameters: Name_of_the_procedure(first_parameter:type of data; second_parameter:type of data,...)

The list of parameters can be void (no parameters at all).

Note: with 8 characters long (i.e. 64 bits), the probability to find a procedure name by hazard in a flow of random characters is negligible.

As it can be seen, the separation character is the “;” character. The name of the parameter is followed by “:” and then by the type of data (as in Pascal).
Between parameters, spaces are ignored.

Parameters, if any, are always outputs. So, the dialog between CP and MP, is a suite of procedures, which can be information and/or requests.

First example (CP to MP):

PROCEDURE Pversion(Protocol_versions_managed_by_the_CP: SHORTSTRING)
The CP sends a list of its managed versions in the form of a SHORTSTRING. It's an information but also a request to answer, because the MP must answer to the CP (in the next example).

Second example (MP to CP):

PROCEDURE Pversion(Protocol_version_selected_by_the_MP: BYTE)
The MP answers with its selected version (knowing the list of the managed CP versions).

Third example (CP to MP)

PROCEDURE HWMPcapa
There is no parameter. It's a request to answer.

Fourth example (CP to MP)

PROCEDURE InitHWMP(type_selected:BYTE;
Initial_sampling_rate_selected:LONGWORD;
Initial_LO_frequency_selected:LONGWORD)
The CP asks the MP to configure itself according to the transmitted parameters.

3.1.6 About the transmission of the procedures through the TCP/IP link

The name of the transmitted procedure (case sensitive, **for example “InitHWMP” is a procedure whereas “INITHWMP” is not a procedure**) is sent followed immediately by the parameters, without “(” or “)” nor “,” nor spaces between parameters, this because the length is defined by the known format length of the parameters. Nothing must alter the global length. TCP/IP transports only characters (8 bits long), so numbers are set in form of characters. For example, for a BYTE, the number “1” will be transformed in CHR(1).

In the previous first example, the transmission from CP to MP could be: PversionCHR(5)0/2/4
Note that it is transmitted the character CHR(5) because the text “0/2/4” is composed of 5 characters.

In the previous second example, the transmission from MP to CP could be: PversionCHR(2)
Note that it is transmitted the character CHR(2) because the answer is version 2.

In the previous third example, the transmission from CP to MP will be: HWMPcapa

In the previous fourth example, the transmission from CP to MP could be:
InitHWMPCHR(3)CHR(128)CHR(187)CHR(0)CHR(0)CHR(216)CHR(180)CHR(214)CHR(0)

Note that it is transmitted successively:

- CHR(3) for the type 3,
- CHR(128)CHR(187)CHR(0)CHR(0) for an initial sampling rate of 48000 Hz
- CHR(216)CHR(180)CHR(214)CHR(0) for a frequency of 14071000 Hz (i.e. 14.071 MHz)

3.1.7 Prefix of each procedure transmission through the TCP/IP link

Before each procedure transmission, it will be transmitted the length of the procedure, in the form of a LONGWORD. This prefix is not indispensable for the decoding, but this redundant information can be useful to simplify data reception, above all if the transmission is split.

For example, it will be transmitted, for the previous example:

CHR(17)CHR(0)CHR(0)CHR(0)InitHWMPCHR(3)CHR(128)CHR(187)CHR(0)CHR(0)CHR(216)
CHR(180)CHR(214)CHR(0)

3.2 List of the procedures and compatibility

Introduction

In version 0.1 of this protocol, the number of procedures must be considered as the minimum, without “bells and whistles”.

The successive versions of this protocol (if any) will, probably, extend the number of procedures and increase the length of the present procedures by adding new parameters or extending the length of the present parameters. So from now on, it will be considered that versions will not be upward compatible.

List of CP procedures

The management of all the procedures for which an action of the CP is required are mandatory, this because the MP could not know which procedures are managed and which are not.

List of the procedures which must be implemented by the CP

Pversion (RX/TX) HWMPcapa (RX/TX) InitHWMP (TX) TxHWdata (RX) Set_HWLO (RX/TX)
Received (TX) Bug_on_P (RX/TX) HW_Pause (RX)

List of MP procedures

List of the mandatory procedures which must be implemented by the MP

Pversion (RX/TX) HWMPcapa (RX/TX) TxHWdata (TX) Received (RX) Bug_on_P (RX/TX)
Set_HWLO (RX)

List of the optional procedures which can be, possibly, implemented by the MP

HW_Pause (TX) Set_HWLO (TX)

Procedures only managed by the MP

HW_Pause TXHWdata

The MP, of course, must be able:

- to answer to all procedures,
- to send data, when available, through the TXHWdata procedure,
- to pause the process (HW_Pause),
- to control the protocol (Bug_on_P).

It is reminded (see §2.6) that in case of incorrect received procedure, the other party will send a Bug_on_P procedure with some message as for example: “Incorrect command: HELLO!”.
A disconnection will follow this procedure.

3.3 Description of the procedures

For each procedure, it is given:

- the case of application (CP->MP or MP->CP or both)
- the presentation of the procedure (see above for this presentation)
- the description
- an explication

- the role of each parameter

Pversion CP->MP (Information+request) then MP->CP (answer)

- CP->MP (Information+request)
PROCEDURE Pversion(Protocol_versions_managed_by_the_CP: SHORTSTRING)
- MP->CP (answer)
PROCEDURE Pversion(Protocol_version_selected_by_the_MP: BYTE)

Description

It is the first procedure to send by the CP and only used once. Before all, the CP and the MP must use the same version of the protocol to use. So an agreement is necessary.

Important: the CP and the MP must be able to manage at least the version 0.1 of the protocol.

The CP gives its managed protocol versions. The MP will return the maximum version of the CP compatible with its own managed versions, which can be, as a minimum, the version 0.1.

Parameters

- Protocol versions managed by the CP: it is supplied the list of the versions managed by the CP, in a growing direction, each one followed by "/". For example: "0/2/4", which means that the managed versions are 0, 2 and 4 (but not 1 or 3). The CP must, at least, manage the version 0.1 (so "0" must be found as the first character of the list).
- Protocol version selected by the MP: according of the CP capacities in terms of protocol versions, the MP will answer with the protocol version that both the CP and the MP can follow, i.e. equal to the maximum version managed between the MP and the CP versions: 1 in the example, if the MP can manage the version 1. The MP must, at least, manage the version 0.1, which will be the minimum version.
Note: of course, if a version is used, all the following commands must be done and interpreted according to this version.

HWMPcapa CP->MP (request) then MP->CP (answers)

- CP->MP (request)
PROCEDURE HWMPcapa
- MP->CP (answer)
PROCEDURE HWMPcapa (Name:SHORTSTRING; Model:SHORTSTRING;
Message:SHORTSTRING; HW_Types_managed_by_the_MP:SHORTSTRING;
Sampling_rates_managed_by_the_MP:SHORTSTRING;
LO_Frequencies_managed_by_the_MP:SHORTSTRING;
GUI_managed_by_the_MP:BYTE)

Description

This procedure is the second called by the CP (and only used once). It is used to get back pieces of information about the HW and the MP capacities. It is requested a descriptive name and model (or Serial Number) of the HW, together with the HW working types, sampling rates, and frequencies managed by the HW and the MP. Afterwards, the CP, knowing all the capacities of the HW/MP, will be able to decide if it will use these HW/MP or not:

- if it is not possible for the CP to work with these HW/MP, the CP will send the procedure Bug_on_P, with an explanation depending of the incompatibility, for example "Server not compatible with Multipsk which only manages type: 3 (I/Q 16 bits) and sampling rate: 48000 Hz". Then the CP will end the TCP/IP link and will stop the MP if it has been started at the beginning,
- if it is possible, the CP will continue with the InitHWMP procedure.

Parameters

- Name: descriptive name of the hardware (in ANSI characters, without control characters). Preferably not longer than about 16 characters. It must be long of at least of one character.
- Model: model code of the hardware (in ANSI characters, without control characters), or its Serial Number. Preferably not longer than about 16 characters. It must be long of at least of one character.
- Message: the HW, through the CP, could, possibly, send a short message (reception one) that will be displayed by the MP. Preferably not longer than about 16 characters. It can be void (no message). Let's say that name+model+message must not pass 48 characters.
- HW Types managed by the MP: it is supplied the HW list of the types of hardware and MP capacities for about demodulation, each one followed by "/". For example: "3/4" if the types 3/4 are managed by the HW/MP. For the HW types, look at InitHWMP, below (type 3 being the standard one).
- Sampling rates managed by the MP: it is supplied the list of the sampling rates (in Hz), in growing order, managed by the MP, each one followed by "/" (without any spaces). An interval is represented with "minimum..maximum". This string must be only composed of figures, "." and "/". For example: "11025..22050/48000/96000/192000..292000", which means that the managed sampling rates are 11025 to 22050, 48000, 96000 and 192000 to 292000 Hz.

The minimum sampling rate is 2000 Hz. The maximum is 999 999 999 Hz.

Note 1: 48000 Hz is the standard value for a sampling rate. The CP and the MP must be able to work at 48000 Hz.

Note 2: a hardware type 4 will send an interval in the form 2000..maximum rate (for example, "2000..96000"), this because the CP which takes samples from the sound card can sample at any rate from 2000 to the maximum (96000 in the example). See also the first note 1, downwards.

- LO Frequencies managed by the MP: it is supplied the list of the LO (Local Oscillator) frequencies (in KHz) managed by the MP, in growing order, each one followed by "/" (without any spaces). An interval is represented with "minimum..maximum". This string must be only composed of figures, "." and "/". For example: "10000..40000/100000..600000", which means that the managed frequencies are 10000 to 40000 KHz and 100000 to 600000 KHz. This is an information for the CP, so as to know what frequencies can be monitored.

The minimum LO frequency is 1 KHz (minimum sampling rate/2). The maximum LO frequency is 4 100 000 KHz (4.1 GHz, i.e. maximum LONGWORD – margin).

- GUI managed by the MP: it is possible that the MP manages a GUI (Graphical User Interface). If so, the CP will be able to choose or not to show the GUI (according of its own GUI). The answer will be 0 for "No" and 1 for "Yes". Note that for type managed 4, it is better (but not compulsory) to show a GUI.

InitHWMP CP->MP (commands)

```
PROCEDURE InitHWMP(type_selected:BYTE;
Initial_sampling_rate_selected:LONGWORD;
Initial_LO_frequency_selected:LONGWORD;
GUI_displayed:BYTE)
```

Description

This procedure is the third called by the CP (and only used once), and it is used both to tell to the MP that it is time to initialize the hardware, taking into account the type of work, the initial sampling rate and the initial Local frequency, together with a possible reception message

from the MP. Once done, audio data are immediately exported from the MP to the CP, via the TXHWdata procedure. So the next procedure to listen by the CP could be normally TXHWdata or possibly HW_Pause or Bug_on_P.

Important: if the MP HW is not compatible with the CP possibility of HW management, the CP will send a Bug_on_P procedure with some message as, for example: "Server not compatible with Multipsk which only manages type: 3 (I/Q 16 bits) or 4 and sampling rate: 48000 Hz". A disconnection will follow this procedure.

Parameters

- CP_type: based on the MP types information, the CP asks the MP/HW to work on this type. This type is an index code that the MP uses to identify the hardware type(s) supported by the CP and the possibility of demodulation from the MP.

Important: the standard type is the 3 one, so it is recommended that the CP be able to work in type 3, as a minimum (but it is not compulsory).

For the type, consider the following values:

Important about the samples sent: all the 16 bits samples **for type 3** are sent by the MP as if were samples sent by a sound card, so in little endian, two's complement. The initial signed signal is supposed symmetrical between -32767 à 32767.

3: the hardware is a receiver. It does its own digitization. The audio data (I/Q data) are sent by the MP to the CP via the TXHWdata procedure. Data must be in 16-bit format (WORD), little endian and two's complement, in an interleaved format (I-Q-I-Q-I-Q etc). It must send each I-Q together not separated (so a whole, so as a "two times 16 bits" packet).

4: the hardware is a receiver. The audio data (I/Q data) are returned via the sound card managed by the CP. The CP has just to control the remainder (LO mainly) through the MP.

- Initial sampling rate selected: among the sampling rates managed by the MP, the CP selects a sampling rate. It is transmitted in little Endian (weakest byte first). 48000 Hz is the standard value for a sampling rate.

Note 1: for a Hardware Type 4, the selected sampling rate is without importance for the CP, because the MP can sample at the required speed on the sound card, and the CP sends the samples at its maximum possible sampling rate. The CP must simply check that the MP maximum sampling rate is superior to its (CP) own sampling rate requirement. See the first note 2, upwards, for the way to specify the MP interval.

- Initial LO frequency selected: among the LO frequencies managed by the MP, the CP selects a LO frequency. It is transmitted in little Endian (weakest byte first). The LO frequency can be changed with SetHWLO.
- GUI displayed: if the MP is able to manage a GUI, the CP can ask the MP to display its GUI, with GUI_displayed=1 ("Yes"). In the other cases GUI_displayed takes the value 0 ("No").

Bug_on_P CP->MP (information before disconnection) or MP->CP (information before disconnection)

PROCEDURE Bug_on_P (explanation_of_the_bug:SHORTSTRING)

Description

According to §2.5 and 2.6, on a bug in the Protocol (or the MP or CP Program), the link is immediately stopped by a disconnection. Just before the disconnection, it is transmitted this

procedure with, as parameter, an explanation of the bug (the text could be void so the parameter sent would only be CHR(0)). It is used also in case of incompatibility between the CP and the MP (see InitHWMP above).

Note: this procedure can be used for any information, even if it is not strictly a bug on the program or the protocol. For example, if the MP disconnect the CP for HW problem, a word of explanation could be nice for the CP.

Parameter: it can be, according to the §2.5 “Disconnection due to a too long waiting time” or according to the §2.6, “Incorrect command: ” followed by the command concerned by the bug with its own parameters, in the total limit of 255 characters (maximum length of SHORTINT).

HW_Pause MP->CP (command)
PROCEDURE HW_Pause(Pause:BYTE);

Description

If for any reason, the HW/MP must pause (and will not send any data during while). The MP sends this procedure with the expected maximum pause, in seconds, between 1 to 5 seconds. It's a warning for the CP which will possibly warn the user, with some message (“Pause requested: 5 s”, for example). Each time this procedure is transmitted, it reinitializes the time counter relatively to the risk described in §2.5 (disconnection after 10 seconds without activity). Note that if, by error, the server sends a value equal to 0 or superior to 5, the server must not be disconnected by the CP. Simply the CP warns the user in a fuzzy way (‘Pause requested’ for example).

Parameter:

The pause is in seconds (1 to 5). For example “2” will be interpreted as “HW pauses less than two seconds”.

Note: of course, this function must not be used by a hardware type 4.

Set_HWLO CP->MP (command) or MP->CP (command)
PROCEDURE Set_HWLO(LO_frequency:LONGWORD)

Description

The “LO” frequency represents the SDR hardware's local oscillator frequency. This frequency is expressed in units of Hz.

It can be used either by the CP or the MP.

- If used by the CP, this procedure is used to communicate the desired LO frequency, among the LO frequencies managed by the MP (and known by the CP, so no error is possible). The frequency is expressed in Hz. The procedure is called at each change of the LO value in the CP main screen.
- If used by the MP, it indicates that a change of the LO frequency has taken place, either by a hardware action, or by an interaction of the user with the MP GUI.

Parameters :

LO frequency: the LO frequency to be set, expressed in Hz.

TXHWdata MP->CP (data)
PROCEDURE TXHWdata(data:LONGSTRING)

Description

This procedure is used by the MP to communicate to the CP the new buffer of audio data (I/Q data). Of course a new buffer of audio data are only sent by MP to the CP if the HW/CP type is not 4 (see InitHWMP). In case of type 4 (I/Q data sent to the sound card), this procedure is not used.

Note: this flow of data is permanent. So the CP client must consider that after 10 seconds without data, the MP has some problem and the CP will disconnect from the CP (and will return to standard working on the sound card, for example).

Note about hardware type 4

This function would be useless for a hardware type 4, as data passed through the sound card. However, it is necessary to control that the link is always effective. So each second

a TXHXdata without data (i.e. an empty LONGSTRING) will be sent. To this TXHXdata procedure, a "Received" procedure will be returned. So each side (MP and CP) will know if the link is effective or not.

Parameter

data : this is a string of samples where the CP places the digitized audio data:

- For type 3, the I/Q data are sent in interleaved format (I-Q-I-Q-I-Q etc.). Samples being in 16 bits, they are transmitted in little endian ordering. The number of bytes sent must be equal to $IQpairs * 2 * N$, where IQpairs is the number of IQ samples, and N is the size in bytes of the type of data, as specified by the "type" parameter of the InitHWMP procedure (i.e 2 for a WORD).

Received CP->MP (information)

PROCEDURE Received

Description

The CP must send this procedure after complete reception of data through the TXHWdata procedure. So the MP will know that the CP is still listening and managing data. Indeed, in case of erroneous working of the CP without disconnection, it would be possible to block the MP (possibly distant). So this presence control permits to be avoid this problem.

Note 1: it is reminded that the MP server must consider that after 10 seconds waiting for a CP event (Received command as a flow of data is permanently transmitted), the CP has some problem and the MP will disconnect the CP.

Note 2: if an important procedure must be transmitted from the CP to the MP, as for example Set_HWLO or SetHWSR, this one can replace the "Received" procedure.