Ghpsdr3 protocols

server HPSDR hardware server

The HPSDR server handles communications with the HPSDR hardware and is designed to work with multiple receivers either on a single Mercury card or multiple Mercury cards. It demultiplexes the I/Q samples into separate streams for each receiver. The hardware server listens on port 11000 for TCP connections from clients. The client (dspserver) makes a connection request through port 11000, and sets up a TCP socket on on port 12000+***rx***, on which it sends commands to the server. IQ data is sent over UDP.

Commands to server

The server recognizes a number of commands, shown below. It sends a response to the client after parsing each command. The possible responses are also shown.

-
  - **attach *rx***

    attach client to receiver ***rx***
    The receiver number *rx* is an integer from 0 to 3
    example: attach 0
    response:
    "Error: Client is already attached to receiver" (Receiver *rx* is already attached.)
    "Error: Invalid Receiver" (*rx* is higher than number of receivers specified on the command line.)
    "Error: Receiver in use"
    "OK" <sample Rate>

  - **detach *rx***

    detach client from receiver ***rx***
    example: detach 0
    response:
    "Error: Client is not attached to receiver"
    "Error: Invalid Receiver" (*rx* is higher than number of receivers specified on the command line.)
    "Error: Not owner of receiver"

  - **frequency *f***

    Set attached receiver frequency to the integer frequency ***f*** (Hz)
    example: frequency 7056000
    response:
    "Error: Client is not attached to receiver"
    "Error: Invalid Receiver" (*rx* is higher than number of receivers specified on the command line.)

  - **start iq *port***

    start sending I & Q samples for the attached receiver to the UDP ***port*** specified
    example: start iq 13000
    response:
    "Error: Invalid Command"

  - **start bandscope *port***

    start sending bandscope samples to the UDP ***port*** specified
    example start bandscope 12500
    response:
    "Error: Invalid Command"

  - **stop iq**

    stop sending I & Q data to the attached receiver
    response:
    "Error: Invalid Command"

  - **stop bandscope**

    stop sending bandscope data
    response:
    "Error: Invalid Command"

A typical exchange would be:

Client	Server
attach 0	OK 96000
set frequency 7056000	OK
start iq 13000	OK
set frequency 7056100	OK
stop iq	OK
detach 0	OK

Note that there are currently no commands to control setting the number of receivers, preamp, dither, random, samplerate or clock sources. As this was developed to support multiple receivers, I decided to not allow clients to change these as they would be changed for all receivers. They are all controlled by command line arguments when running the server.

I / Q data stream

The I/Q samples are accumulated in the server until there are 1024 I/Q samples to send. The samples are sent in 512 byte UDP packets, to the port specified in the 'start iq' message, with 12 byte header and up to 500 bytes of I/Q samples.

The UDP packet header contains:

- 8 byte sequence number (At present, the Windows port uses only a 4 byte sequence

number. Until this is changed, both server and dspserver must be running on Windows or Linux, not one on Windows and one on Linux.)
- 2 byte offset within this set of samples
- 2 byte length of samples

Following the header, the I and Q samples are sent, first 1024 float I samples, followed by 1024 float Q samples.

server command line options (*default value*)

- --receivers <***1***..4>
- --samplerate [48000|***96000***|192000]
- --dither [***off***|on]
- --random [***off***|on]
- --preamp [***off***|on]
- --10mhzsource [atlas|penelope|***mercury***]
- --122.88mhzsource [penelope|***mercury***]
- --micsource [janus|***penelope***]
- --class [E|***other***]

Example:

./server --receivers 4 --samplerate 96000 --dither on --10mhzsource atlas

Note: As an experiment, there is also a softrockserver that supports the same protocol with the restriction that there is only one receiver and it uses the USB/si570 interface to control the frequency. The I/Q samples are taken from an Audio device specified on the command line.

dspserver

dspserver is a client to the hardware server. In turn, it is the server for another client, typically a GUI, such as jmonitor. It provides a dsp processing service using DttSP. Typically one dspserver is started for each receiver. The dspserver listens for client TCP connections on port 8000+rx. i.e. rx0 listens on 8000, rx1 listens on 8001, etc. The client sends commands to the dspserver and dspserver sends data packets to its client on this same port. The first byte of each packet sent to the client identifies the type of data in the packet.

- 0 spectrum data
- 1 audio data

Commands to dspserver

-
    - **getSpectrum *points***

      Request ***points*** of spectrum data. The value sent for each point is an 8 bit unsigned integer. If the calculated spectrum contains more points than the client requests, adjacent points are combined, and the highest value in the group is sent.
      Example: getSpectrum 480

    - **setFrequency *f***

Set the receiver frequency to ***f***, in Hz
example: setFrequency 7056000

- **setMode *mode***

  mode is an integer specifying the type of demodulation

-
  - 0 - LSB
  - 1 – USB
  - 2 – DSB
  - 3 – CWL
  - 4 – CWH
  - 7 - DIGU
  - 8 - SPEC
  - 9 - DIGL
  - 10 - SAM
  - 11 – DRM

  example: setMode 0

- **setFilter *low high***

  Set filter cutoff frequencies
  For LSB, both frequencies are negative
  For USB, both frequencies are positive
  For AM or DSP, typically **low** is negative and **high** is positive
  example: setFilter 150 3450

- **setAGC *n***

  Select AGC time

-
  - 1 - LONG
  - 2 - SLOW
  - 3 - MEDIUM
  - 4 - FAST

  example: setAGC 3

- **setNR *n***

  Set noise reduction

-
  - 0 - off
- **setNB *n***

  Set noise blanker

-
  - 0 - off
- **setANF *n***

  Set automatic noise filter

- - 0 - off
  - **setRXOutputGain *gain***

    Set receiver audio output level to the value ***gain***
    The value is a floating point number in the range 0 to 1.
    example: setRXOutputGain 0.5

  - **startAudioStream *buffer size***

    start sending audio data.
    buffer size is optional, and is an integer, typically 480.
    If omitted, a value of 480 is used.

  - **stopAudioStream**

    stop sending the audio data.

data packet format Each packet has a 48 byte header.

Byte	Function
0	Packet type: 0=spectrum data; 1=audio data
1-31	Reserved
32-39	Sample rate
40-47	Meter reading
48-end	spectrum or audio data

Sample rate and Meter reading numbers are in ASCII characters.

Audio data is sent as one channel of 8-bit aLaw samples at 8000 samples per second.

dspserver command line options (*default value*)

- --soundcard SANTA_CRUZ, AUDIGY_2_ZS, MP3_PLUS, EXTIGY,

  DELTA_44, FIREBOX, EDIROL_FA_66, ***HPSDR***

- --receiver ***0***, 1, 2, or 3
- --server IP address ***127.0.0.1*** (this is "localhost")
- --offset ???

summary of port usage

sender	type	port	purpose
dspserver	TCP	11000	connection request to server
dspserver	TCP	12000 + 2*rx	commands to server
server	TCP	12000 + 2*rx	responses to commands

server	UDP	13000 + 2*rx	I & Q data to dspserver
dspserver	UDP	15000 + 2*rx	demodulated audio to server for Mercury headphone/speaker output
dspserver	TCP	8000 + rx	spectrum & alaw-encoded audio to monitor
monitor	TCP	8000 + rx	commands to dspserver