VIRGO DDS

Web User Interface Software Reference Manual

INFN Padova

By Lorenzo Castellani

The interface consists of a web page that implements a series of forms for setting the operating parameters, access to all the registers of the various chips and a WebSoket Server (RFC 6455) which responds on port 4444. For the modification of the registers requires authentication while for read-only it is not necessary. To change some registers, need authentication with a privileged user.
LabView library to communicate with the server is available https://www2.pd.infn.it/~caste/pub/WebSockets.zip), the authentication procedure is implemented in the example.

Preliminary commands implemented.

TEXT commands accepted by server are:

ID command: This command is used to obtain the id of the board

Syntax:	**Id?**	
Returns:	**<string>**	an identification string

AppLog command: This command is used to obtain log file of the application

Syntax:	**Log?**	
Returns:	**<text>**	the application log file

OpLog command: This command is used to obtain log file of the operation on DDS registers

Syntax:	O**pLog?**	
Returns:	**<text>**	the opration log file

Authenticate command: This command is used to obtain the information necessary for the

authentication handshake. The nonce value expires after 60 seconds.

Syntax:	**Authenticate?**
Returns:	**{ "realm": "authorized only", "nonce": "bb7a2bc19db7495606c57750f90ba775"}**

Authorization command: This command in conjunction with Authenticate command must be

used to enable the connection to accept write commands. User and password can be added or changed using console with linux command htdigest. Example:

```
~$ htdigest /etc/wspasswd “authorized only” operator
```

/etc/wspasswd is the password file, see configuration file, “authorized only” is realm string obtained by Authenticate command and operator is the username. By default, users can authenticate with user “operator” and password “virgo”.

Syntax:	**Authorization:<user>:<realm>:<nonce>:<response>**
Arguments:	<user> specified the username.
	<realm> specified the realm string obtained by Authenticate command.
	<nonce> specified the nonce string obtained by Authenticate command.
	<response> must be calculate using MD5 hash by the following string:
	ha1=MD5(“<user>:<realm>:<password>”);
	response=MD5(“<ha1>:<nonce>”);
Returns:	**OK** or **ERROR:104,Not authorized**

Authentication handshake example with user “operator” and password “icarus”:

Client -> Authenticate?

Server -> {"realm": "authorized only", "nonce": "93482f2f0719e2b8ed2b5ad54f7e9150"}

Client -> Authorization:operator:authorized only:93482f2f0719e2b8ed2b5ad54f7e9150:d6995fa640f1ad4dafd009199422490a

Server -> OK

Returns Error message Syntax: **ERROR:<number>,<message>**

Binary commands accepted by server are:

Commands implemented with websoket optcode = binary frame.
The format of the frames: [cmd] [arg1] .... [argn].
The first byte represents the command followed by the arguments.
The following structures describe the arguments for each command, UINT32 represents an integer 32bit (4 bytes) in little-Endian format, FLOAT64 an 8byte floating point in little-Endian format.

Write DDS registers, code 0x01,0x02,0x03,0x04: see AD9910 datasheet

```
BYTE cmd;        // 0x01 for dds1, 0x02 for dds2 ..., 0x04 for dds4
```

```
UINT32 cfr1;
UINT32 cfr2;
UINT32 cfr3;
UINT32 auxdac;
UINT32 ioupd;
UINT32 ftw;
UINT32 pow;
UINT32 asf;
UINT32 multc;
UINT32 dig_rampl[2]; //first array element LSB, and second MSB
UINT32 dig_ramps[2];
UINT32 dig_rampr;
UINT32 sin_tonep0[2];
UINT32 sin_tonep1[2];
UINT32 sin_tonep2[2];
UINT32 sin_tonep3[2];
UINT32 sin_tonep4[2];
UINT32 sin_tonep5[2];
UINT32 sin_tonep6[2];
UINT32 sin_tonep7[2];
UINT32 hc4094;          // only the bits related to the DDS are considered the others are ignored
FLOAT64 ref_frequency; //Reference frequency (common to all DDS)
```

Return a frame with the same data type as a response.

HC4094 bit format: DDS1 bit0-4 AD9385 Gain, bit5-7 Profile, DDS2 bit8-12 AD9385 Gain bit13-15 Profile…….

Write UG104 register, code 0x05**:** see ADF4360 datasheet

```
BYTE cmd;       // 0x05
```

UINT32 control;

UINT32 rcounter;

UINT32 ncounter;

Return a frame with the same data type as a response.

Write HC4094 register, code 0x06:

BYTE cmd;   // 0x06

UINT32 reg;   // HC4094 data reg

Return a frame with the same data type as a response or error frame.

HC4094 bit format: DDS1 bit0-4 AD9385 Gain, bit5-7 Profile, DDS2 bit8-12 AD9385 Gain bit13-15 Profile…….

Active DDS Output, code 0x08:

BYTE cmd;   // 0x08

BYTE dds;   //select dds= 0-3

BYTE out;   // out on/off = 1/0

Return a frame with the same data type as a response or error frame.

Save Config, code 0x09:

BYTE cmd;   //0x09

BYTE cfg;   //config number: 0 default,1 congig1

Return a frame with the same data type as a response or error frame.

Update DDS, code 0x0A: generate a pulse on the WRAD9910 line

BYTE cmd;   //0x0A

Return a frame with the same data type as a response or error frame.

Set Async Notify, code 0x09:   Enable/disable the connection to receive the register changed by other client to monitor the value

BYTE cmd;   //0x0B

BYTE on;   //notify value

Return a frame with the same data type as a response or error frame.

Read DDS register , code 0x81, 0x82,0x83, 0x84:

```
BYTE cmd;        // 0x81 for dds1, 0x82 for dds2...., 0x84 for dds4
```

Returns a frame with the same data type as the write command in response or error frame

Read UG104 register, code 0x85:

```
BYTE cmd;        // 0x85
```

Returns a frame with the same data type as the write command in response or error frame

Read HC4094 register, code 0x86**:**

```
BYTE cmd;        // 0x86
```

Returns a frame with the same data type as the write command in response or error frame

Read Status, code 0x87:

```
BYTE cmd;        // 0x87
```

Return the following frame

```
BYTE cmd;        //0x07
UINT32 status;  //bit0-3=DDS sync error, bit4-7=DDS power down, bit8=UG104 MuxOut
FLOAT64 Tempeature1;          //temperature sensor1
FLOAT64 Tempeature2;          //temperature sensor2
FLOAT64 Tempeature3;          //temperature sensor3
FLOAT64 Voltage;              //Board Voltage
BYTE autoriz;                 //1 if authorized.
UINT32 time;                  //seconds from cpu boot
```

Read DDS output, code 0x88:

```
BYTE cmd;        // 0x88
BYTE dds;        // select dds
```

Return the same type of data as the 0x08 command or error frame.

Write AD9910, code 0x11, 0x12, 0x13, 0x14**:**

```
BYTE cmd;        // 0x11 for dds1, 0x12 for dds2 ..., 0x14 for dds4

UINT32 cfr1;

UINT32 cfr2;

UINT32 cfr3;

UINT32 auxdac;

UINT32 ioupd;

UINT32 ftw;

UINT32 pow;

UINT32 asf;

UINT32 multc;

UINT32 dig_rampl[2]; //first array element LSB, and second MSB

UINT32 dig_ramps[2];

UINT32 dig_rampr;

UINT32 sin_tonep0[2];

UINT32 sin_tonep1[2];

UINT32 sin_tonep2[2];

UINT32 sin_tonep3[2];

UINT32 sin_tonep4[2];

UINT32 sin_tonep5[2];

UINT32 sin_tonep6[2];

UINT32 sin_tonep7[2];
```

Return a frame with the same data type as a response.

Load Config, code 0x89:

```
BYTE cmd;        //0x89

BYTE cfg;        //config number: 0 default,1 congig1
```

Return a frame with the same data type as a response or error frame.

Read AD9910 register, code 0x91,0x92,0x93,0x94**:**

BYTE cmd; // 0x91 for dds1, 0x92 for dds2...., 0x94 for dds4

Returns a frame with the same data type as the write command in response or error frame

Return data on error , code 0xFF:

BYTE cmd; //0xFF

INT32 errorcode; // Error code

Error code:

-1	Not Authorized
-22	Invalid Value
-5	I/O Error
-9	Unknow command (connection will be closed)
-2	no such file
-13	Permission denied

See linux c/c++ error base for undefined number (errno-base.h)