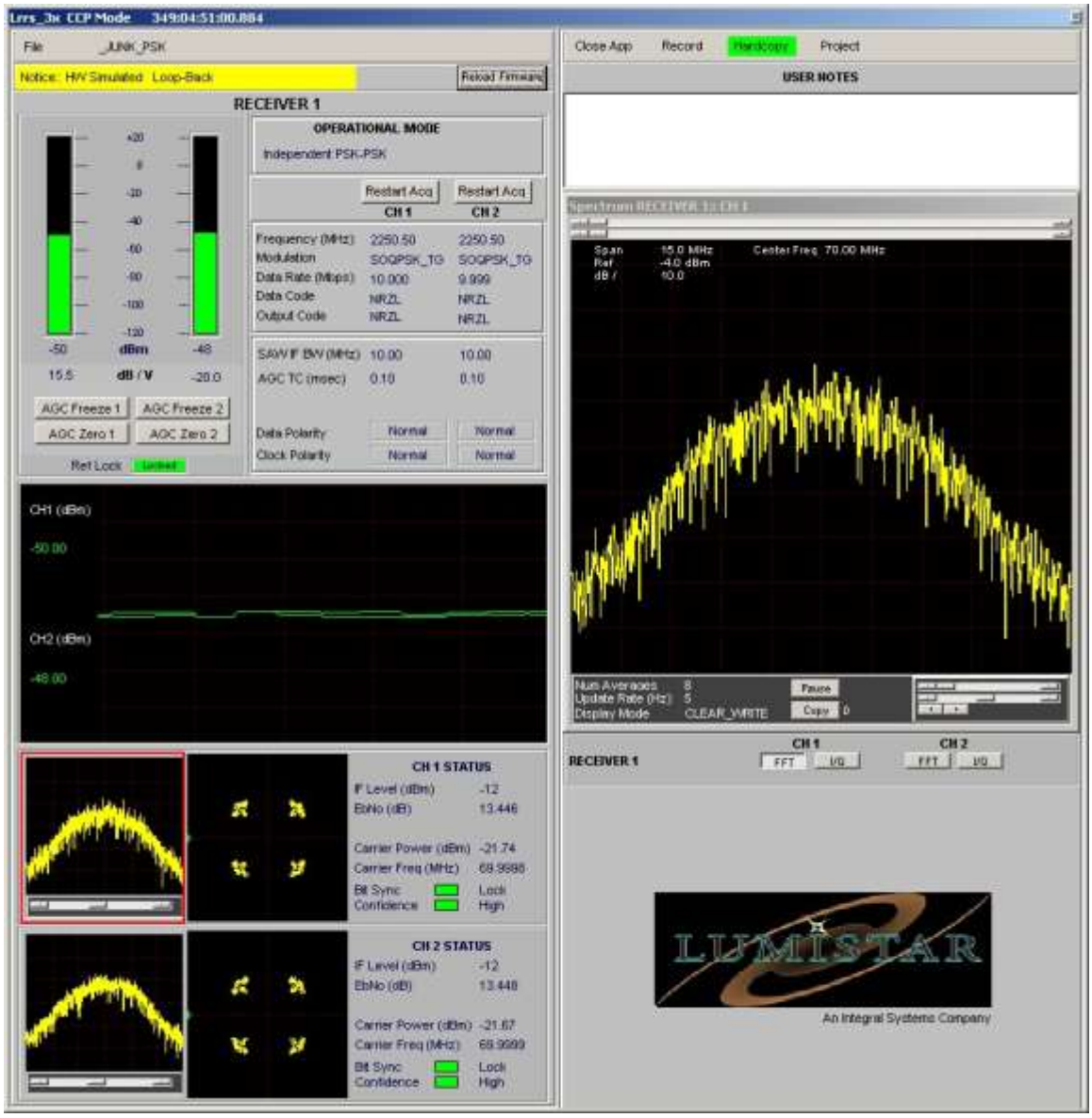
05/29/13

LRRS_3X Network Interface, Control

The LRRS_3x application can broadcast status information on the network, if you so desire. Commanding the LRRS_3x application via the network is limited to primary control commands, and the items not available are automatically set up per IRIG 106.

The control commands are only available if LRRS_3x **is only available if the option 'Allow Network Control' is checked in the system-options-operations menu**. See the CCP Mode display page for a receiver for a quick reference. Everything you see for status is broadcast via the status sockets.



The CCP Mode divides things up into two groups of controls. Each group contains a dual channel LS-27P3 and a dual channel LS-35. Between the two groups is a common card control, the LS-69S. All controls to the LS-69S are performed automatically for you. You can have either one or two groups.

Each group has a channel 1 and a channel 2, and an operational mode control. The operational mode determines the mode the group is going to work in, and therefore the FPGA load required. Changing the operational mode takes some time (about 5-6 seconds), so you'll have to take that into account when you are commanding that. Along those lines, there is a "Reload Firmware" control. This takes the same amount of time as the operational mode.

Within each group are two channels. Channel 1 is meaningless for control, depending on the operational mode. When you send a control command, the command will consist of the group ID, and the channel ID, followed by the command and the value and/or value string. You will receive an acknowledgement on whether the command was completed successfully or not. For each channel, you will be able to control the following:

- AGC Freeze
- AGC Zero
- AGC Slope Mode
- Restart Acquisition
- Tuned Frequency
- Modulation Format
- Data Rate
- Data Code
- B/S Acquisition Loop Bandwidth (pct)
- B/S Track Loop Bandwidth (pct)
- Output Code
- Saw IF BW
- AGC TC
- Data Polarity
- Clock Polarity
- Combiner Polarity Pre-D
- Combiner Polarity Post-D
- Get Saw IF BW Freqs Available
- Get AGC TC Times Available
- Set Recombine Location (Router, PSK mode) – STRING VAR
- Set Stream Swap Location (Router, PSK mode) – STRING VAR
- Set Invert Location (Router, PSK mode) – STRING VAR
- Viterbi Rate Mode - STRING VAR
- Viterbi Symbol Order - STRING VAR
- Viterbi Enable
- Viterbi Invert G1
- Viterbi Invert G2

There are a few Test Modulator commands.
SETS

- Test Mod Carrier Enable
- Test Mod Modulation Enable
- Test Mod Noise Enable
- Test Mod Carrier Amplitude
- Test Mod Noise Level
- Test Mod Modulation Mode– STRING VAR
- Test Mod Modulation Type– STRING VAR
- Test Mod Pcm Encoder 1 Data Source– STRING VAR

- Test Mod Pcm Encoder 1 Data Source– STRING VAR
- Test Mod Pcm Encoder 1 Data Source– STRING VAR
- Test Mod Pcm Encoder Output Code– STRING VAR
- Test Mod Pcm Encoder 2 Data Source– STRING VAR
- Test Mod Pcm Encoder 2 Output Code– STRING VAR
- Test Mod Rx Bert Reset Statistics
- Test Mod Rx Bert Prn Pattern– STRING VAR
- Test Mod Tx Bert Data Rate
- Test Mod Tx Bert PRN Pattern– STRING VAR
- Test Mod Tx Bert Clock Polarity
- Test Mod Tx Bert Data Polarity
- Test Mod Tx Bert Xmit BER
- Test Mod PM/PSK Modulation Index
- Test Mod I Data Source– STRING VAR
- Test Mod Q Data Source– STRING VAR
- Test Mod IQ Routing– STRING VAR
- Test Mod Symbol Routing– STRING VAR
- Test Mod Input Preprocessing– STRING VAR

GETS

- All the above SETS, with the exception of Reset Bert Stats

Group commands will have the same format, but the channel ID is meaningless. Those commands are:

- Reload Firmware
- Operational Mode
- Save Setup
- Recall Setup
- Recall Project
- Test Mod Calls

The operational mode has seven modes:

- Independent FM-FM
- Independent PSK-PSK
- Independent PSK-FM
- Pre-D Combined PSK
- Pre-D Combined FM
- Post-D Combined PSK
- Post-D Combined FM
- Independent PM-PSK

Lrrs_3x – Network Setup

Setup of the network for Windows is **your** responsibility. If the Windows network is not setup, then see your network administrator.
Ensure you have privileges to broadcast and receive. This is broadcast protocol, but you set the mask to be to a single client.
Ensure you have your firewalls and anti-virus software set up to allow data flow.
You can fight with your network administrator about those issues.

Lrrs_3x – Control Protocol

The control protocol is similar to the SNMP protocol. It has a command port and an acknowledgement port. You send the command, then wait for the ack. The ack structure will contain the result of your command (complied/failed), and an error string if it failed. If you command was one of the 'Get' commands, the ack structure will have the data you requested to get in it.

See the "NetLrx3xCmn_DefinesUnit.h". It contains the structures for the command and ack buffers.

netls3xsendcmdstruct
netls3xackcmdstruct

These two structures are made up of other structures, so just feed your way back on them. They are fairly simple.

NOTE. You'll note there are several "Reserved_xxx" type variables in the structures. They are there for growth. It is best you don't use them. If any of those variables have other than 0 in them, then you probably need to contact Lumistar to get an updated version of the "NetLrx3xCmn_DefinesUnit.h".

See the "Lrs3xRemoteNetUdpCmdUnit.h" (and cpp) files. This is how the protocol works. You'll also need the "NetLrx3xCmn_UtilsUnit.h" (and cpp) files. These handle the UDP sockets, the command functions, and the wait for ack functions.

The ls35clientcmdudpclass has an ack thread listening for the server to ack, and a mutex to keep from sending another command before the ack was received (or timed out). Probably a good idea to maintain that logic.

The SendTheCmd() function has a call it it that says to wait for the ack for x milliseconds. "result = WaitForAck(maxwaitmsecs);".
Most calls you make are returned in just a couple milliseconds. A few calls take seconds to return. You'll have to program accordingly. See "IsALongCommand(cmd)" to see which commands take a long time to return the ack.

Control Commands

See the enum type def “netls3xtypecommand”. First, you need to ensure your compiler is set to treat enums as ints.

Again, you’ll see several “RESERVED” type enums. For growth. It is best you use the enum, just in case the growth gets too large and I run out of RESERVED items.

Some commands require a string to be sent, others require a double to be sent. See above for the ones that require a string. You must fill in the appropriate type value. The string, if that is what is needed, is case sensitive.

Some Get commands return a string value vs. a double. Look at the NumStrings var in the netls3xackdatastruct. If it is greater than 0, then the value you want on return from the Get call is in the ReturnedCharAry. Otherwise, it is in the ResultAry.

Client Example

The Lrs3xClientUdp.exe example controls everything available, and lets you know if you can control the LRRS_3X application. When you send a command, the memo fills with the command you sent and the results of that command (OK or the fail reason).

You'll note in the enums for the commands, that there are two "Get" commands. These two are so you can see what the IF Filter BW and AGC TC values are, so you can set your GUI up to display them instead of the index for the item to select. Just press the "Get Avail IF/TC" button at the top and it will request the values and fill in the GUI and popups for those items. Otherwise the GUI will just display the index number.

For the Test Modulator, there are several Get commands. See the example code in the 2 Hz timer where all the Test Modulator Get commands are run at once. Some of these are doubles, others are Booleans, but most of them are strings.

When you send a command, you'll see a yellow "Waiting" bar at the top right. Most of the time, this will just flash on and then off. If you send one of the commands that take longer, it will be displayed until the server has finished doing your request.

There is no real logic in the GUI to keep the user from sending a command to receiver 1 if the operational mode doesn't allow it. The server will handle that anyway and send the command to both receivers as needed, so the command will actually go through. Sending the wrong modulation type will result in a fail from the server. You'll have to incorporate that kind of logic yourself. Making things invisible for that kind of logic is up to you and how your powers that be want it to work.