# Application Note: read_RCB_LVDS_file.m Modifications

## RCB-LVDS UDP Data

The RCB-LVDS uses UDP (user data protocol) packets to stream RHD2000 sample data over a wireless network. There are several advantages for this over other protocols, but it comes with some caveats.

Overall data throughput can be substantially higher than TCP because acknowledgements are unnecessary. In this case, throughput is as much as 10% higher than TCP.

Another advantage is that the module can potentially multicast the data to several devices simultaneously without increasing network load.

A third advantage is that the data source does not become congested or jammed due to a problem with the path to the destination; if TCP were used, sockets can become stuck in half-open states, resulting in poor performance or outright failure.

One caveat, though, is that packet delivery is not guaranteed – if the network becomes congested, routers or switches along the way are free to drop packets without notifying either the sender or receiver. Consequently, the receiver must be robust enough to handle this situation of missing packets.

A second caveat is that packets may arrive at the destination out of order. This second situation is not nearly as bad as the first, since no data has actually been lost.

A third situation is that sometimes data packets can be received more than once, due to network nodes attempting to be more robust. All of these situations can contribute to a corrupt data file if not handled correctly.

This application note addresses a solution to maximize the value of collected data with the RCB-LVDS system.

## Maximizing Data Integrity

Prior to addressing data files that have already been captured with potential aberrations, it is instructive to review system setup practices that will produce greater likelihood of success. The integrity of the data captured by the RCB-LVDS system can be enhanced by following a few practices:

### Minimize Sample Rate

Reducing the amplifier sample rate will reduce the overall load on the wireless network, and consequently will increase the robustness of the system. Data rate on the network is linearly proportional to channel sample rate.

### Disable Unneeded Channels

Disabling unneeded channels will reduce the overall data rate on the wireless network, and will also increase the robustness of the system. Data rate on the network is nearly proportional to the number of enabled amplifier channels.

### Locate Wi-Fi Router Nearby

Locating the Wi-Fi access point near the RCB-LVDS, e.g. within about ten feet, will improve performance dramatically. Care should be taken not to enclose the RCB-LVDS within a Faraday-cage type of structure that will block 2.4 GHz signals to the access point. In some situations where this has been unavoidable, success has been demonstrated with locating the access point antenna within the same structure.

### Minimize Other 2.4 GHz Traffic

Other equipment that generates RF energy in the 2.4 GHz spectrum includes Bluetooth, some ISM, and other Wi-Fi traffic. If possible, relocate such equipment farther from the RCB-LVDS system, and reassign other Wi-Fi traffic to 5GHz access points.

For example, connect the GUI PC to either the 5 GHz SSID on the router, or with wired LAN, rather than assigning both the PC and the RCB-LVDS to the 2.4 GHz Wi-Fi.

## Maximized Data Integrity

The RCB-LVDS GUI software has been designed to store all data that arrives, whether there are gaps or disordered packets. Since this software is designed to be maximally compatible with Intan RHD2000 GUI data files, it produces binary capture data files in the same format.

As a result, Intan's read_Intan_RHD2000_file.m script can be used directly to read these files. However, the script will fail abruptly if there are any gaps or redundancies in the data.

We have provided a minor modification to the Intan Matlab® script (read_RCB_LVDS_file.m) to reorder the data, if necessary, and remove any redundant packets.

```
    % Check for gaps in timestamps.
    num_gaps = sum(diff(t_amplifier) ~= 1);
    if (num_gaps == 0)
        fprintf(1, 'No missing timestamps in data.\n');
    else
        fprintf(1, 'Warning: %d gaps in timestamp data found.  Time scale
will not be uniform!\n', ...
            num_gaps);

        num_pre = length(t_amplifier);
        [t_amplifier, index_sorted] = unique(t_amplifier);
        amplifier_data = amplifier_data(:, index_sorted);
        num_post = length(t_amplifier);
        fprintf(1, 'Removed %d duplicate timestamps.\n', num_pre-
num_post);

        num_gaps = sum(diff(t_amplifier) ~= 1);
        if (num_gaps == 0)
            fprintf(1, 'Corrected: No missing timestamps in data.\n');
        else
            fprintf(1, 'Corrected: Warning: %d gaps in timestamp data
found.  Time scale will not be uniform!\n', ...
                num_gaps);
        end

    end
```

Fig. 1. Matlab® script modification to maximize available data, starting at line 422.

The script first looks to see if any aberrations exist in the data file. If not, then "No missing timestamps in data." is reported. If there are gaps or redundancy, then the data are both sorted and redundancies removed. The results are then checked again to see if there are still aberrations, and reported to the user. At this point, the maximum amount of data has been recovered.

The script is available on the DSP Wireless website.