The diagnostic programs of the CCB.

`[Document number: ???, revision 1]`

Martin Shepherd
California Institute of Technology

November 2, 2005

This page intentionally left blank.

Abstract

This document describes a suite of programs that can be used to test and diagnose problems in the CCB. These programs include passive programs that display dump-mode samples, but don't control the CCB, and active diagnostic programs which both control the CCB, and read back data.

Contents

Chapter 1

Introduction

The CCB manager is optimized for astronomical observervations, where all that the observer wants is for the manager to acquire data and archive it to disk, for subsequent analysis off-line. When debugging problems with the hardware, it would be tedious and error-prone to have to set up an observation scan, arrange for the resulting data to be written to a file, and then run another program on that file. This would preclude tests that involved taking hundreds of scans, each with different configuration parameters. It would also preclude procedures that involved eyeballing the data while interactively tweaking something in the hardware.

For these reasons two classes of test programs have been provided that are optimized for performing diagnostic tests and performing real-time visualization.

1. **Passive dump-mode visualization programs**

   Passive dump-mode visualization programs allow one to syphon-off and display dump-mode data, while the CCB is being separately controlled by either the CCB manager, or by the ccb_demo_client program.

2. **Diagnostic test programs**

   Diagnostic test programs not only collect data and analyse it, but also configure the CCB, and tell it when to start diagnostic scans. They thus handle all of the steps that would otherwise have to be performed by hand with the manager and off-line analysis programs, and can be run quickly and repeatedly.

Chapter 2

Dump-mode visualization programs

The following is a short summary of the dump-mode visualization programs:

ccb_display_dump	-	Continuously display dump-mode samples, as they arrive. This program requires that a separate manager or the `ccb_demo_client` program be used to start a dump scan.

2.1 ccb_display_dump

This program displays dump-mode data, whenever the CCB manager or the ccb_demo_client programs execute dump-mode scans. When no dump-mode scan is in progress, it displays nothing.

Note that when displaying dump-mode samples on a terminal, it usually makes sense to reduce the number of samples that are acquired per dump-frame to less than the number of lines in the terminal window. With the ccb_demo_client program, this is done by finding the entry box that says *samples/frame* and changing the value shown there to (for example) 15. You will then see the first 15 samples of each integration period.

The program accepts the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact. If this option isn't used, then the default host-name, "ccb1.gb.nrao.edu" is assumed.

- -base *n*
  Specify which numeric-base should be used to display the numbers. Legal forms of this option are:

-base 2	-	Display in binary.
-base 8	-	Display in octal (base 8).
-base 10	-	Display in decimal (the default).
-base 16	-	Display in hexadecimal (base 16).

If this option is not used, then the numbers are displayed in decimal.

Chapter 3

Diagnostic test programs

The following is a short summary of the diagnostic test programs:

ccb_test_phase_timing	-	Acquire phase-switch response curves.
ccb_test_adc_delay	-	Measure noise versus the ADC-clock delay.
ccb_test_dc_response	-	Acquire the response to a linear DC ramp.
ccb_test_fake_samples	-	Verify that test-mode dump samples match the expected values of pseudo-random test samples.
ccb_test_fake_integ	-	Verify that test-mode integrations match the expected sums of pseudo-random test samples.
ccb_test_sample_stats	-	Measure the mean and RMS noise of one dump-mode frame of 16383 ADC samples per input-port and phase-switch state.

3.1 Commonly used command-line arguments

The following command-line options are common to many of the test programs. Which programs support which arguments will be described separately for each program, along with any program-specific arguments.

- -host *hostname*

  By default each of the test programs assumes that the CCB server is running on the computer at ccb1.gb.nrao.edu. The -host argument allows one to tell a test program to talk to a CCB server on a different computer.

  For example:

```
        ccb_test_program -host ccb2
```

would run a test program called ccb_test_program, and tell it to talk to a local computer called ccb2, instead of ccb1.

- -repeat *n*

  By default, with the exception of ccb_display_dump, the test programs run their tests once, then exit. If you wish to run a given test multiple times, use the -repeat argument to specify how many times. The special value of 0 (ie. zero) indicates that the program should repeat its test indefinitely.

  For example:

  ```
  ccb_test_program -repeat 2
  ```

  would cause a program called ccb_test_program to run its test 2 times, instead of just once.

- -conf *assignments*

  This parameter can be used to override a test program's default CCB configuration parameters. It's argument should be a single text string that contains one or more assignments to named configuration parameters, separated by spaces. Beware that because the configuration string must be a single string, the space-separated list of assignments should be enclosed within quotes.

  ```
  ccb_test_program -conf "samp_per_state=500"
  ```

  would tell a test program called ccb_test_program to have the CCB advance from one phase-switch state to the next every 500 × 100ns ADC samples, instead of whatever default number the program normally chooses.

  Note that programs that provide this option, also display the final configuration to the terminal, before starting to run their tests.

- `-verbose`

  By default, most test programs only display their conclusions of analysing data that they obtain. To see intermediate details, such as the values of samples obtained, then include the -verbose flag.

  ```
  ccb_test_program -verbose
  ```

- `-port` *1-16*

  Test programs that operate the CCB in dump-mode, and only read from one front-panel input-port, default to acquiring data that arrive at the J1 input port on the front-panel of the CCB. To use a different port, locate the port on the front panel of the CCB, and enter the number that follows the J of its name.

  For example,

  ```
  ccb_test_program -port 2
  ```

  would tell a program called ccb_test_program to acquire data from the J2 input port of the CCB.

Subsequent sections describe these programs in more detail.

3.2 ccb_test_phase_timing

This program obtains low-noise averages of the time evolution of the CCB's response to phase-switch transitions. From these averages, it attempts to automatically figure out the roundtrip delay and settling times of the phase-switching signals. It can also optionally be told to display the phase-switch response curves to the standard output of the program, in the form of a table with 3 columns of numbers, the first column specifying the delay of each row of the table, with respect to the originating transition of the phase-switch control signal, the second column showing the average sample at this time for a switch-off transition, and the third showing the average sample at this time, for a switch-on transition.

If the numbers in the displayed table are imported into a program, such as IDL or Gnuplot, and the numbers in the 2nd and 3rd columns are plotted versus the numbers in the 1st column, then the time evolution of the response of the CCB to off and on phase-switch transitions can be examined by eye.

3.2.1 How the program works

Looked at from a high-level, what this program does is the following:

1. It starts a dump-mode frame, with both of the phase-switches held continuously in the off state, and after discarding data from one integration-period, to ensure that the phase-switches have had time to settle into the off state, it takes a single dump-mode frame. It then measures the mean value of the samples in this dump-frame, along with the scatter about that mean. In verbose mode, it displays these numbers.

2. Next it does the same thing, but this time with both phase-switches turned continuously on.

3. The program now has good estimates of the noise levels of data from both the on and the off states of the phase-switches. It subsequently uses this information to anticipate the noise levels of averages of multiple frames of data.

4. Now the program configures the chosen phase-switch to toggle, while holding the other phase-switch turned off. This results in a phase-switch cycle of two states. The first state of each cycle is configured to hold the phase-switch off, while the second holds the phase-switch on. Thus at the start of each integration period the phase-switch should be transitioning from on to off, and the average of multiple dump-mode frames of data from the start of each integration period should reveal a low-noise picture of the response in the data to a single phase-switch turning off.

5. The program now starts a dump-mode scan with the above configuration, and acquires and averages together a specified number of frames of data. It then keeps this average "switch-off" frame for later analysis and possible display.

6. The program then repeats the above two steps, except that instead of configuring the chosen phase-switch to be turning on at the start of each integration period, it configures it to be turning off. Thus the program also acquires an average frame of data containing a switch-on transient.

7. If the -verbose command-line flag has been specified, the program now displays the contents of the two averaged frames of the switch-off and switch-on transients side by side. If the standard output of the program has been redirected to a file, then these two columns of numbers can be read into a program, such as IDL or Gnuplot, and plotted, to show the response curves of the receiver to phase-switch transitions.

8. The program now attempts to measure the roundtrip phase-switching control-delay and the phase-switch settling time from the averaged frames.

   The roundtrip delay is the amount of time that elapses between the CCB telling the receiver to change the state of a phase-switch, and the initial effects of this being seen in the data that are received by the CCB. This includes rise-time and cabling delays in the RFI-filtered digital control-signals that switch the phase-switches, the response time of the post-detector Bessel filters, and the pipeline delay of the ADCs in the CCB. When the CCB's roundtrip_dt parameter is set to zero, the visible effect in the received data, of phase-switch transients, is to see a few samples at the start of each integration that don't change, to within the noise level, followed by the signal starting to rise or fall, in response to the phase-switch transition.

   The program uses a simplistic algorithm to determine the roundtrip time, and applies this separately to the mean switch-off and switch-on frames. It uses the noise level that was previously computed, for the phase-switch state that preceded the switch transition, multiplies this by a parameter nsigma, which can be set by the -nsigma argument, and looks for the first sample of the averaged frame, that differs by at least this amount from the value of the first sample of the frame. No assumptions are made about which direction this deviation should be in, since this depends on the RF inputs to the two horns of the receiver.

The roundtrip_dt CCB configuration-parameter, which is specified as a number of 100ns sample intervals, needs to be set to one greater than the number of quiescent samples that are found at the start of the integration period. Thus the program outputs this number.

Given that both the phase-switch control signals, and the cal-diode control signals traverse equivalent circuitry to their respective switches, and transients in the data have to go through the same signal-path, regardless of what caused them, the CCB assumes that a single common roundtrip-delay parameter is sufficient to accomodate all of these switching signals. This program, however, outputs two estimates of the roundtrip_dt parameter; one for switching on phase-switches and one for switching them off. If these numbers are significantly different, and looking at the data by eye confirms this discrepancy, then the *smaller* of the two numbers should be used, since using a number that is too large for one of the transitions, would result in data from the transition being moved into the end of the previous phase-switch state, where it wouldn't be blanked.

9. Next the program attempts to deduce the settling time of the phase-switch transitions. It does this by searching the samples that follow the roundtrip quiescent period, for the first sample that differs from its predecessor by less than nsgima times the anticipated noise level of the post-switch signal level.

   It then displays this number minus the roundtrip delay, to indicate the corresponding value of the CCB phase_switch_dt configuration parameter.

In practice, it may make more sense to plot the switch-on and switch-off responses, and deduce roundtrip and settling times from them by eye, than to rely on the simplistic algorithm that is described above. This can be done by using the -verbose command-line option, to tell the program to output the average phase-switch responses as columns of numbers, and then read these numbers into a plotting program, such as Gnuplot or IDL.

If you do this, be sure to zoom in on the response curves, in order to be able to see the asymptotic parts of the long tails of these curves, where the deviation may still be significant in integrated data, even if they look very small.

3.2.2 Optional command-line arguments

The ccb_test_phase_timing program has the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact (See section 3.1).

- -repeat *n*
  Specify how many times to repeat (See section 3.1).

- -conf *assignments*
  Modify the CCB configuration parameters (See section 3.1)

  Note that in this program, the only configuration parameter that can reasonably be changed, is the samp_per_state parameter, which configures the length of each phase switch state. It makes sense to modify this parameter, because it also sets the length of the dump frames that the program reads, and thus determines the duration of the time after each phase-switch transition, that is acquired. For example

  ```
  ccb_test_phase_timing -conf "samp_per_state=500"
  ```

  would tell the program to acquire dump-mode frames of 500 samples, as well as causing it to toggle the chosen phase-switch every 500×100ns. This would allow the phase-switch responses to be examined out to 500×100ns. Beware that the run-time of the program scales directly with this number. This is because the speed of the program is limited by the rate at which data can be received from the firmware over a USB1.1 link.

  Note that if the value of the samp_per_state parameter isn't changed in this way, it defaults to 1000.

- `-verbose`
  Display intermediate details and data (See section 3.1)

- `-port` *1-16*
  Select which input port is sampled (See section 3.1)

- `-switch` *a* or *b*

  By default the program toggles phase-switch A, while holding phase-switch B in a steady state. Thus the acquired phase-switching response is that caused by toggling phase-switch A. To have phase-switch B toggled instead of A, one would type:

  ```
  ccb_test_phase_timing -switch b
  ```

- `-average` *number-of-frames*

  Unless told otherwise, the program averages 10000 frames of dump-mode samples, separately for the switch-on and switch-off transients, to acquire low-noise averages of the corresponding phase-switch response curves. This number can be changed with the -average command-line argument. Thus to reduce the number of frames that are averaged to 2000, one would type:

```
ccb_test_phase_timing -average 2000
```

Since the run-time of the program scales linearly with this number, the above example would reduce the run-time by a factor of 5. The drawback of this would be that the noise level would be increased by a factor of $\sqrt{5}$.

- `-nsigma` *`factor`*

  When the program attempts to locate the first sample that changes by a significant amount from the quiescent value at the start of a phase-switch transition, and when it then attempts to figure out when the transition has stabilized, it interprets deviations in the signal as being significant if they exceed the measured noise level, multiplied by a factor. This factor, which defaults to 1.0, can be changed with the -nsigma command-line argument.

  Thus, to make the program notice deviations in the signal, of half the measured noise level, one would type:

  ```
  ccb_test_phase_timing -nsigma 0.5
  ```

3.3 ccb_test_adc_delay

The purpose of the ccb_test_adc_delay program is to measure the noise levels of all of the CCB inputs, as a function of the phase-shift-delay between the FPGA and ADC clock signals. It optionally displays all of the measured noise levels, and then outputs the delay that results in the lowest noise levels across all inputs.

3.3.1 How the program works

Looked at from a high-level, what this program does is the following:

1. Unless the default configuration is changed on the command-line, the program arranges that there will be no phase-switching or cal-diode switching in any of the scans that it starts.
2. For each of the input ports and each of the 10 supported clock delays, the program then does the following:
   (a) It starts a dump-mode scan for the specified input-port, and reads one dump-mode frame of samples.

(b) It then works out the mean of the samples, along with the standard deviation of this mean, and stores these numbers for later analysis.

(c) If the -verbose command-line flag was specified, the program then displays the mean ADC count and standard deviation, along with the name of the originating port, the current ADC delay, and the number of unsaturated ADC-samples that went into the mean.

3. Next, if the -verbose command-line flag was specified, the program displays a table of noise levels versus ADC delay and input port.
4. It then finds the ADC delays that resulted in the lowest noise levels of each of the input ports, and prints these to the terminal.
5. Finally, it computes the noise levels as a function of ADC delay, averaged across the measurements from all ports, and prints these numbers out, followed by a line that indicates which ADC delay resulted in the lowest average noise level across all ports.

3.3.2 Optional command-line arguments

The ccb_test_adc_delay program has the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact (See section 3.1).
- -repeat *n*
  Specify how many times to repeat (See section 3.1).
- -conf *assignments*
  Modify the CCB configuration parameters (See section 3.1)
- `-verbose`
  Display intermediate details and data (See section 3.1)

3.4 ccb_test_dc_response

This program explores the response of the CCB to a staircase ramp of DC input signals. To do this it requires that the ribbon cable that connects the CCB computer to the GPIO card be replaced with the test ribbon cable that breaks out one of analog output channels of the GPIO card to a coaxial cable that can be plugged into the test video-amplifier module. In addition to optionally printing a row of statistics for each DC input-level, it fits a straight line to the graph of input voltage versus measured ADC count, and reports both the gradient and zero-offset of this line.

3.4.1 How the program works

What the program actually does is the following:

1. The program starts a loop over the range of GPIO-card DAC counts that generate differential voltages covering the input-range of the CCB inputs.

   For each of these voltages, the program does the following:

   (a) It sets the analog output of the GPIO card to the voltage that is dictated by the loop-variable.

   (b) It then starts a dump-mode frame for the selected input port, with the phase-switches and cal-diodes held in the off position.

   (c) It then reads one dump-mode frame, computes the mean value of its samples, and the standard deviation of the samples from the mean.

   (d) It internally records the mean and standard deviation, for later analysis, and in verbose mode, displays a row of numbers, showing the input voltage at the selected input-port, the aformentioned mean value and standard deviation, and the number of unsaturated samples that went into the mean.

2. Now that the program has a list of mean ADC counts and variances, as a function of voltage, it fits a straight line to the mean counts versus voltage, weighted by the inverse of the variances, and reports the best fit gradient (gain), and zero-offset of the ADC counts.

3.4.2 Optional command-line arguments

The ccb_test_dc_response program has the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact (See section 3.1).
- -repeat *n*
  Specify how many times to repeat (See section 3.1).
- -conf *assignments*
  Modify the CCB configuration parameters (See section 3.1)
- `-verbose`
  Display intermediate details and data (See section 3.1)
- `-port` *`1-16`*
  Select which input port is sampled (See section 3.1)

3.5 ccb_test_fake_samples

This program uses dump-mode to verify that when the CCB is told to replace the real ADC samples with fake samples, that the resulting sequence of fake samples matches the expected pseudo-random sequence.

3.5.1 How the program works

The ccb_test_fake_samples program configures the CCB to substitute fake samples for real ADC samples, and then takes a dump-mode frame of data. Since the pseudo-random fake sample generator resets itself at the start of each integration period, the value that each sample in the frame should have, is known. Thus the program simply compares the acquired samples with the expected sequence. In verbose mode it also displays the acquired samples and their expected values, as a table.

3.5.2 Optional command-line arguments

The ccb_test_fake_samples program has the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact (See section 3.1).
- -repeat *n*
  Specify how many times to repeat (See section 3.1).
- -conf *assignments*
  Modify the CCB configuration parameters (See section 3.1)
- `-verbose`
  Display intermediate details and data (See section 3.1)
- `-port` *`1-16`*
  Select which input port is sampled (See section 3.1).

  Note that changing the port number may be used to test the different daughter-cards, since each card handles a different set of 4 input ports.

3.6 ccb_test_fake_integ

This program uses integration-mode to verify that when the CCB is told to replace the real ADC samples with fake samples, that the integrated values in each phase-switch bin of each input port, match the corresponding expected sums of the known sequence of pseudo-random fake samples.

3.6.1 How the program works

The ccb_test_fake_integ program configures the CCB to substitute fake samples for real ADC samples, and then takes an integration-mode frame of data. Since the pseudo-random fake sample generator resets itself at the start of each integration period, the fake numbers that are summed into each phase-switch bin can be calculated. Thus these numbers are compared to what are actually acquired, and any differences are reported. In verbose mode, both the expected and the acquired integrated values are also displayed.

3.6.2 Optional command-line arguments

The ccb_test_fake_integ program has the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact (See section 3.1).
- -repeat *n*
  Specify how many times to repeat (See section 3.1).
- -conf *assignments*
  Modify the CCB configuration parameters (See section 3.1)
- `-verbose`
  Display intermediate details and data (See section 3.1)

3.7 ccb_test_sample_stats

The ccb_test_sample_stats program takes one dump-mode frame of samples for each combination of CCB input-port and phase-switch state, with the phase-switches not switching. It then computes the mean of the samples in each dump-frame, along with the RMS of the deviations of these samples from the mean. As it does this, it reports these numbers to the terminal.

3.7.1 Optional command-line arguments

The ccb_test_sample_stats program has the following optional arguments:

- -host *hostname*
  Specify which CCB computer to contact (See section 3.1).
- -repeat *n*
  Specify how many times to repeat (See section 3.1).
- -conf *assignments*
  Modify the CCB configuration parameters (See section 3.1)
- `-verbose`
  Display intermediate details and data (See section 3.1)