CCB Configurations & Use Scenarios
v1 04feb03 bsm – 1st draft
v2 24feb03 Brian Mason, Melinda Mello
v3 01mar04 bsm, mm

Contents

1 Introduction **1**

2 Example CCB Configurations **2**
2.1 Ka band Switched with Cal 2
2.2 Ka band Switched no Cal 3
2.3 Ka band Switched with Cal, Fast Sampling 4
2.4 Pre Scan Idle Configuration 5

3 User/Manager/Server Interaction **6**
3.1 Parameter Configuration & Monitoring via CLEO 6
3.2 Parameter Configuration via GO, Observers Interface, & the Configuration Tool 6
3.3 Execution of a Scan 7

4 Rules & Dependencies **8**

5 FITS File **9**

6 Observing Procedures **9**

1 Introduction

The purpose of this document is to outline the configuration of the Caltech Backend under a number of hypothetical use scenarios. Except as noted, I assume that the YGOR device manager for the Caltech Backend provides an essentially transparent interface to the parameters controlled by the following CCB server API functions: `ccb_queue_phase_switch_cnf`, `ccb_queue_cal_diode_cnf`, `ccb_queue_telemetry_cnf`, & `ccb_timing_cnf`. The definition of YGOR scan states & transitions (start, stop, abort; standby; ready; *etc.*), and the relation of these YGOR states to CCB scan states must be enumerated later. *Note that an YGOR scan is a different entity than a CCB scan, and the default meaning of "scan" in Shepherd & Yamasaki's documentation is the latter.* A CCB-scan can more accurately (or less confusingly) be thought of as simply a reconfiguration of the device.

Following are some cases. The function calls/parameter values which must be made to the CCB server to configure each scan of this observation are specified. Comments are parenthetical. Sets (of bits, or elements in an array) are enumerated between [].

2 Example CCB Configurations

2.1 Ka band Switched with Cal

- `ccb_queue_phase_switch_cnf()`
  - `active_switches` = $[1, 1]$
  - `driven_switches` = $[1, 1]$
  - `initial_states` = $[0, 0]$
  - `samp_per_state` = 250 (= 25 μsec per phase switch state ie measurement in Martin's terminology, at 100 nsec per sample)
- `ccb_queue_cal_diode_cnf`
  - `ncal` = 10
  - `driven_diodes` = $[1, 1]$
  - `diode_a` = $[0, 1, 0, 1...0, 1]$ (10 elements, five non-zero)
  - `diode_b` = $[0...0]$ (10 zeroes)
  - `ninteg` = 1 (duration in integrations of each element in the above two arrays)
- `ccb_queue_telemetry_cnf`
  - `integ_period` = 1000 (with 2 phase switches hence 4 measurements per cycle, and 250 100 nsec samples per phase switch state within a cycle (ie per measurement), this results in a 0.1 sec integration period for each of $4(freqchannels) \times 2(feeds)\ 2(polarizations) \times 4(phaseswitchstates) = 16$ ccb output channels.
  - `monitor_interval` = 10 (in integrations)
  - `stream_selection` = CCB_ALL_STREAMS
- `ccb_queue_timing_cnf`
  - `sample_dt` = 1 (100 nsec— in fact, *not* a paramater as it's fixed)
  - `phase_switch_dt` = 10($\times$100 nsec)
  - `diode_rise_dt`= 20 ($\times$100 nsec)
  - `diode_fall_dt` = 20($\times$100 nsec)

2.2 Ka band Switched no Cal

- `ccb_queue_phase_switch_cnf()`
  - `active_switches` $= [1, 1]$
  - `driven_switches` $= [1, 1]$
  - `initial_states` $= [0, 0]$
  - `samp_per_state` = 250 (= 25 μsec, at 100 nsec per sample)
- `ccb_queue_cal_diode_cnf`
  - `ncal` = 10
  - `driven_diodes` $= [1, 1]$
  - `diode_a` = [0...0] (10 zeroes)
  - `diode_b` = [0...0] (10 zeroes)
  - `ninteg` = 1
- `ccb_queue_telemetry_cnf`
  - `integ_period` = 1000 (with 2 phase switches hence 4 measurements per cycle, and 250 100 nsec samples per phase switch state within a cycle (ie per measurement), this results in a 0.1 sec integration period for each of $4(freqchannels) \times 2(feeds)$ $2(polarizations) \times 4(phaseswitchstates) = 16$ channels
  - `monitor_interval` = 10 (in integrations)
  - `stream_selection` = `CCB_ALL_STREAMS`
- `ccb_queue_timing_cnf`
  - `sample_dt` = 1 (100 nsec— in fact, *not* a paramater as it's fixed)
  - `phase_switch_dt` = 10($\times$100 nsec)
  - `diode_rise_dt`= 20 ($\times$100 nsec)
  - `diode_fall_dt` = 20($\times$100 nsec)

2.3 Ka band Switched with Cal, Fast Sampling

- `ccb_queue_phase_switch_cnf()`
  - `active_switches` = $[1, 1]$
  - `driven_switches` = $[1, 1]$
  - `initial_states` = $[0, 0]$
  - `samp_per_state` = 250 (= 25 μsec, at 100 nsec per sample)
- `ccb_queue_cal_diode_cnf`
  - `ncal` = 10
  - `driven_diodes` = $[1, 1]$
  - `diode_a` = [0...0] (10 zeroes)
  - `diode_b` = $[0, 0, 1, 0...0]$ (10 elements, one non-zero)
  - `ninteg` = 1
- `ccb_queue_telemetry_cnf`
  - `integ_period` = 10 (with 2 phase switches hence 4 measurements per cycle, and 250 100 nsec samples per phase switch state within a cycle (ie per measurement), this results in a 1 msec integration period for each of $4(freqchannels) \times 2(feeds)\ 2(polarizations) \times 4(phaseswitchstates) = 16$ channels
  - `monitor_interval` = 1000 (in integrations)
  - `stream_selection` = CCB_ALL_STREAMS
- `ccb_queue_timing_cnf`
  - `sample_dt` = 1 (100 nsec— in fact, *not* a paramater as it's fixed)
  - `phase_switch_dt` = $10(\times 100$ nsec)
  - `diode_rise_dt`= 20 ($\times$100 nsec)
  - `diode_fall_dt` = $20(\times 100$ nsec)

2.4 Pre Scan Idle Configuration

The purpose of this configuration is to provide a well-defined pre-scan idle state. This will allow an actual observing scan to be started (via `ccb_queue_start_scan_cmd()`) with a minimal and predictable overhead of 1ms.

- `ccb_queue_phase_switch_cnf()`
  - `active_switches` $= [0, 0]$
  - `driven_switches` $= [1, 1]$
  - `initial_states` $= [0, 0]$
  - `samp_per_state` $= 1000$ $(1000 \times 100ns = 100us)$
- `ccb_queue_cal_diode_cnf`
  - `ncal` $= 1$
  - `driven_diodes` $= [1, 1]$
  - `diode_a` $= [0]$
  - `diode_b` $= [0]$
  - `ninteg` $= 1$
- `ccb_queue_telemetry_cnf`
  - `integ_period` $= 10$
  - `monitor_interval` $= 1000$
  - `stream_selection` = CCB_LOG_STREAM
- `ccb_queue_timing_cnf`
  - `sample_dt` $= 1$
  - `phase_switch_dt` $= 10$
  - `analog_reset_dt` $= 10$
  - `diode_rise_dt`$= 0$
  - `diode_fall_dt` $= 0$

3 User/Manager/Server Interaction

3.1 Parameter Configuration & Monitoring via CLEO

1. **User**: sets a parameter in a CLEO widget.
2. **CLEO**: sends the parameter data to the manager via set_param() (a CLEO function call). If the parameter change results in an illegal state, the parameter is not changed.
3. **User**: sets another parameter in a CLEO widget.
4. **CLEO**: sends the parameter data to the manager via set_param(). If the parameter change results in an illegal state, the parameter is not changed.
5. **User**: Presses the "Prepare" button in CLEO.
6. **CLEO**: conveys this to the manager.
7. **Manager**: Changes the configuration of the hardware by
   - a series of ccb_queue_*_cnf() commands
   - ccb_queue_stop_scan_cmd()
   - ccb_send_control_msgs()

This assumes Auto-prepare is off; if that's not the case, the sequence is 1–2–7 for *each* parameter.

Note that due to the interdependencies of some parameters getting to some states may require an indirect route. The worst of this can be avoided by making the CLEO interface for the CCB calibration cycle (those parameters controlled by ccb_queue_cal_diode_cnf) analogous to the CLEO interface for the phase table. This would then be an exception to what is specified above in that the set of parameters controlled by the cal table would be set all at once.

3.2 Parameter Configuration via GO, Observers Interface, & the Configuration Tool

Since GO, the Configuration Tool, & Observers Interface provide a higher level interface to the backend, this is simpler.

1. **somebody**: As part of the observing setup, slave the Ka band receiver to the CCB by settting bits 1 and 2 of RA49h, and bits 1 and 2 of RA4Ah, all to 1 (see §§2.2 and 2.3 of the 1cm MCB interface Specification Document—is there a mapping of these addresses & settings to parameter names??)
2. **Observer** (or observer-generated script) specifies a complete configuration for the backend.

3. **GO/OI/CfgTool**: Validates this configuration internally, and passes it as a single entity, effectively, to the manager (via appropriate use of the register change functions— an YGOR-interals detail).

4. **Manager**: Changes the configuration of the hardware by

   - a series of ccb_queue_*_cnf() commands
   - ccb_queue_stop_scan_cmd()
   - ccb_send_control_msgs()

3.3 Execution of a Scan

This section describes how an *individual M&C scan* is executed. Here is the sequence of events

1. **User**: requests a backend configuration (manually or with a script; see cases in § 3.1 and 3.2) which is implemented as described in § **??**.

2. **User**: sends scan start and scan length to the ScanCoordinator (possibly in a hands-off fashion via GO/Observer's Interface).

3. **ScanCoordinator**: requests the earliest start time from the backend manager.

4. **Manager**: returns a value of 1 second.

5. **Manager** during its activated state[1]: invokes the pre-scan idle state by: queueing configuration commands corresponding to the configuration in § 2.4, followed by `ccb_queue_stop_scan_cmd(), ccb_send_control_msgs()`.

6. **Manager** during its committed state: wait until $1 - \delta$ seconds before the scheduled start-of-scan, at which time issue: `ccb_queue_start_scan_cmd(), ccb_send_control_msgs()`.

7. **Manager**: after the requested duration has elapsed (or the requested stop time is reached) the FITS file is closed out and the scan is complete. The hardware need not know this has happened.

Note that the pre-scan idle state is internal to the manager. The switch to this configuration and the subsequent reversion to the requested observing configuration require no user interaction other than asking for a scan.

The observer's interface allows three ways to specify a scan:

- start as soon as possible and take data for N seconds;
- start at time x and take data for N seconds;
- start as soon as possible and stop taking data at time y.

The observer's interface can use the above sequence for all three of these methods (in which case "as soon as possible" will always be 1 second).

[1] "activated" and "committed" are states of the YGOR scan state machine; they are invoked by the ScanCoordinator by standard mechanisms.

4 Rules & Dependencies

Following are the rules for CCB parameter configurations.

- The number of elements of the arrays ccb_queue_cal_diode_cnf.diode_{a,b} is equal to ccb_queue_cal_diode_cnf.ncal.
- The value of ccb_queue_phase_switch_cnf.samp_per_state cannot exceed $32/2^N$, where N is the number of active phase switches (0, 1, or 2) specified by ccb_queue_phase_switch_cnf.active_switches.
- No entry in ccb_queue_phase_switch_cnf.active_switches can be nonzero if the corresponding entry of ccb_queue_phase_switch_cnf.driven_switches is zero.
- The two entries of ccb_queue_cal_diode_cnf.driven_diodes correspond to the two arrays ccb_queue_cal_diode_cnf.diode_{a,b}. If one of the entries of ccb_queue_cal_diode_cnf.driven_diodes is zero, the corresponding array (diode_a or diode_b) must contain all zeroes.
- For the CCB associated with the 3mm receiver:
  - ccb_queue_phase_switch_cnf.active_switches = $[*, 0]$
  - ccb_queue_cal_diode_cnf.driven_diodes = $[*, 0]$ and then that implies
  - ccb_queue_cal_diode_cnf.diode_b = $[0, 0...0]$
- The duration of a scan should not be less than the duration of a calibration cycle. The duration of a calibration cycle can be calcuated from a number (that number is ccb_queue_cal_diode_cnf.ncal $\times$ ccb_queue_cal_diode_cnf.ninteg) invocations of the equation Martin gives for the physical duration of an integration

  $$\begin{aligned} \mathrm{t} &= \mathrm{ccb\_cal\_diode\_delay(ccb, integration, ncal, diode\_a, diode\_b, ninteg)} \\ & \quad + \mathrm{integ\_period} * \mathrm{ccb\_cycle\_length(ccb, active\_switches)} * \\ & \quad \mathrm{samp\_per\_state} * (\mathrm{sample\_dt} + \mathrm{analog\_reset\_dt}) \end{aligned}$$

  This is not really a rule but rather is a good idea, and the user should be warned if it's violated.
- Except as above, all parameters have legal ranges specified by the datatypes in Martin's document "The Communications Interface between the Ygor Manager and the CCB Server".

Note that as pertains the individual receivers the correspondences between phase switches, cal diodes, and parameters, is clearly TBD (in particular as regards whether the 3mm rx is missing phase switch 1, or phase switch 2; cal diode a or cal diode b).

5 FITS File

6 Observing Procedures