



**Atacama
Large
Millimeter
Array**

Correlator Subsystem Software Design

COMP-70.40.00.00-001-F-DSN

Status: Approved

Jim Pisano

Jesus Perez

Prepared By:		
Name(s) and Signature(s)	Organization	Date
Jim Pisano	NRAO	2009-08-12
Jesus Perez	NRAO	2009-08-12
Approved By:		
Name and Signature	Organization	Date
Released By:		
Name and Signature	Organization	Date



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 2 of 86

Change Record

Version	Date	Affected Section(s)	Change Request #	Reason/Initiation/Remarks
A		All		Initial Draft
B		All		Incorporated IDR comments
C	2003-05-16	All		Incorporated PDR comments
D	2003-07-29	All		Incorporated CDR1 comments
E	2004-03-22	All		Initial version for CDR2
F	2005-05-06	All		Update for CDR3
G	2005-06-09	All		Update for CDR3 review comments
A	2006-04-27	Various		Update for CDR 4 Revision 'C'
B	2006-05-30	Various		Incorporation CDR 4 review comments
D				CDR 5
D	2007-06-26	Various		Incorporate CDR 5 review comments
E	2008-06-5	Various		CDR 6
F	2009-05-20	Various		CDR 7
F	2009-08-12	Various		Incorporate CDR 7 review comments


	<p>ALMA Project</p> <p>Correlator Subsystem Software Design</p>	<p>Doc # : COMP-70.40.00.00-001-F-DSN</p> <p>Date: 2009-08-12 Status: Approved</p> <p>Page: 3 of 86</p>
--	---	--

Table of Contents

1	Introduction	7
1.1	Purpose	7
1.2	Scope	8
1.3	Glossary	8
2	Requirements	9
2.1	SSR Requirements	9
2.2	Operations Plan Requirements	9
3	Architecture	9
3.1	Overview	9
3.2	CCC Packages and Functional Overview	11
3.2.1	ACS Interface	11
3.2.2	Monitor	12
3.2.3	Command Dispatcher	12
3.2.4	Tunable Filter	12
3.2.5	LO Offsetting	12
3.2.6	Geometric Delay	12
3.2.7	Digitizer Statistics	12
3.2.8	Array Management	12
3.2.9	Maintenance	13
3.2.10	CAN I/F	13
3.2.11	Array Time Interface	14
3.2.12	TE Handler	14
3.2.13	Correlator Configuration Validation	15
3.3	Detailed CCC Package Descriptions	15
3.3.1	CCC Command Dispatcher	15
3.3.2	Geometric Delay	15
3.3.3	Array Management	18
3.3.4	Monitor	19
3.3.4.1	Correlator Hardware Monitor	19
3.3.4.2	CCC Hardware Monitor	19
3.3.4.3	Correlator Configuration Monitor	19



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 4 of 86


3.3.5	Correlator CAN Commands.....	20
3.3.5.1	LTA CAN Commands	20
3.3.5.2	SCC CAN Commands	21
3.3.5.3	QCC CAN Commands.....	21
3.3.5.4	Final Adder Commands	21
3.3.5.5	DPI CAN Commands	21
3.4	Sub-scan Control Sequence	21
3.5	Preloaded Configurations	22
3.6	CDP Packages and Functional Overview	25
3.6.1	Array Time Interface.....	28
3.6.2	TE Handler	28
3.6.3	CDP Monitor	28
3.6.4	CDP Maintenance	28
3.6.5	Cluster Administration	28
3.6.6	CDP Master Node Interface	29
3.6.7	Master Data Publisher	29
3.6.8	CDP Node	29
3.6.9	Node Data Publisher	29
3.6.10	Spectral Processing	30
3.6.11	Lag Processing	30
3.6.12	HPDI 32	30
3.6.13	Residual Delay	30
3.6.14	Atmospheric Phase Correction.....	30
3.6.15	Sideband Separation.....	30
3.6.16	CDP Configuration.....	32
3.6.17	Array Configuration	32
3.6.18	TE Scheduler	32
3.7	Detailed CDP Package Descriptions	32
3.7.1	Lag Processing	32
3.7.2	Correlator Flagging	35
3.7.3	Spectral Processing	36
3.7.3.1	Lag Normalization	37




ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 5 of 86

3.7.3.2	Quantization Correction.....	37
3.7.3.3	Windowing.....	37
3.7.3.4	FFT.....	37
3.7.3.5	TFB Subband Stitching.....	38
3.7.3.6	TFB Bandpass Calibration.....	38
3.7.3.7	TFB Scaling Calibration	38
3.7.3.8	Residual Delay Adjustment	38
3.7.3.9	Atmospheric Phase Correction	40
3.7.3.10	Integration Averaging	40
3.7.3.11	Spectral Channel Averaging	41
3.7.3.12	Channel Average.....	41
3.7.4	Data Publishing	41
3.7.4.1	Scaling Factors.....	43
3.8	Data Flow Robustness	43
3.8.1	Detail Data Flow	43
3.8.2	Alarms	44
3.9	Physical Architecture	45
3.9.1	Correlator Control Computer	47
3.9.2	Correlator Data Processor	47
3.9.3	Network Infrastructure	47
3.9.4	Correlator Hardware.....	47
3.9.4.1	Basebands	47
3.9.4.2	Correlator Chip Accumulation.....	48
3.9.4.3	Bin switching	48
3.9.5	Physical Computer Racks	48
3.10	Computer Cooling	49
3.11	Dynamic Model	50
3.11.1	CCC Timing Discussion	50
3.11.2	CDP Compute Node Timing.....	51
3.11.3	CDP Master Node Timing	54
3.11.4	Computational Load Resolution.....	55
3.11.5	CDP Processing Pipeline.....	56

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 6 of 86
--	--	---

3.12	System Reliability	57
3.12.1	Single-Point Failures	57
3.12.1.1	Correlator Control Computer	57
3.12.1.2	Correlator Data Processor Computer	57
3.12.2	Sources of Failure	57
3.12.3	High altitude disk server	58
3.12.4	Error Handling	58
3.12.4.1	External interfaces to the CDP Master	58
3.12.4.2	External interfaces to the CCC	59
3.12.4.3	Internal interfaces between CCC and CDP Master	59
3.12.4.4	Internal interfaces between CDP Master and CDP nodes	59
3.12.5	MTBF	59
3.13	System Startup	59
3.13.1	UPS Monitoring	60
3.14	Telescope and Monitor Configuration Database	61
4	Correlator Simulator	61
4.1	Correlator Simulator Software Components	62
4.1.1	CAN Commands	63
4.1.2	Lag Data	63
4.1.3	Configuration Tool	63
4.2	ALMA Observatory Simulator	63
5	Correlator GUI	63
5.1	Correlator configuration and spectral viewing	63
5.2	Correlator monitoring and diagnostics	63
6	Correlator Monitoring GUI	64
6.1.1	Error Diagnostic	64
6.1.2	Monitor Point Viewing	64
6.1.3	Diagnostic Testing	64
7	References	68
8	System Interfaces	70
8.1	Package – Interface Relationship	70
9	Appendices	71

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 7 of 86
--	--	---

9.1	Minimum Integration Duration	71
9.1.1.1	Archive Limitations	72
9.1.1.2	Sideband separation	72
9.1.1.3	Science Requirements	73
9.2	Equations	75
9.2.1	V_s Subtraction	75
9.2.2	Lag Normalization	75
9.2.3	Geometric Delays	75
9.2.4	Atmospheric Phase Correction	75
9.2.5	Digitization Correction	75
9.2.6	Windowing Functions	75
9.2.7	Discrete Fourier Transforms	76
9.2.8	Channel Averaging	76
9.2.9	Sideband Separation	76
9.3	Correlator Hardware CAN Commands	76
9.3.1	Common CAN Commands	76
9.3.2	LTA Specific CAN Commands	77
9.3.3	SCC-Specific CAN Commands	78
9.3.4	QCC-Specific CAN Commands	79
9.4	Correlator Configuration	79


1 Introduction

1.1 Purpose

This document provides a high-level system design for the correlator computer system software which performs the following tasks:

- Configure the correlator hardware and data processing parameters for a sub-scan.
- Process raw lags from the correlator hardware into raw spectral blocks
- Transmit the correlated spectral data to the Archive.
- Monitor and publish status information for the correlator hardware and Correlator subsystem computers.
- Provide an interface for maintenance and diagnostics

The main constraints on this subsystem include the real-time demands placed on it by the high data rates of the correlator hardware output, the synchronization of execution with the array-wide timing events and intensive data processing loads due to correlator hardware output rates.

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 8 of 86
--	--	---

1.2 Scope

The design described is valid for the computer subsystem which controls, configures, monitors and processes the raw data from the NRAO baseline correlator. This design also covers the 2-antenna prototype correlator to be delivered to the Antenna Test Facility (ATF) at the VLA site.


1.3 Glossary

The standard ALMA glossary of terms can be found at:

<http://edm.alma.cl/forums/alma/dispatch.cgi/Glossary/showFile/100004/d20030521120217/Yes/Glossary.pdf>

Common terms used in this document are:

ACC	Array Control Computer, the computer located at the OSF and responsible for coordinating all instrument activities. It is an ordinary Unix workstation.
ACS	ALMA Common Software, a software infrastructure to support a distributed software system utilizing CORBA
APC	Atmospheric Phase Correction, path length correction due to atmospheric fluctuations
ARTM	Array Real Time Machine, a computer at the Array Technical Building which aligns external time to hardware timing events
BDD	Bulk Data Distributor, an ACS module which allows the distribution of high speed data from one publisher to many subscribers.
CAI	Correlator Antenna Input Each set of 4 baseband pairs at an antenna are routed to correlator inputs in the correlator hardware. These inputs are called 'Correlator Antenna Inputs' and are represented as 0-based numbers. A mapping exists between antenna labels and correlator antenna inputs.
CAMB	Correlator & Antenna Master Bus, the correlator and control subsystem's media used for communicating between a master computer and other devices.
CCC	Correlator Control Computer, a rack-mounted PC with Ethernet, CAN, RS-232, and RS-485.
CDP	Correlator Data Processor, a PC-cluster which obtains raw lags from the correlator, and converts them to spectral data blocks.
DPI	Data Port Interface, the high-speed, 32-bit binary interface between the correlator Long Term Accumulator and the CDP.
HPDI	High Performance Data Interface, the interface card in each CDP compute which receives raw lags from the correlator via the 'DPI'.
LTA	Long Term Accumulator, the main computer interface for control and data processing of the correlator hardware.
QCC	Quadrant Control Card, a type of interface card in each correlator quadrant which monitors quadrant voltages and temperatures.
RTAI	Real Time Application Interface [1], a set of Linux kernel modules which allow for deterministic, real-time behavior with the Linux O/S.

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 9 of 86
--	--	---

SCC	Station Controller Card, a type of interface card which monitors and controls the correlator station and the filter cards.
TE	Timing Event - an array-wide, highly-accurate 48 ms timing signal which is used to synchronize distributed hardware and computers.
TFB	Tunable Filter Board - high-resolution digital filter consisting of 32 multiple sub-filters which can be tuned anywhere within the 2 GHz base-band providing higher resolution and multiple spectral windows of similar or different resolutions within a single 2 GHz baseband.
TMCDB	Telescope Monitor Configuration Data Base - a relational database which is used by the Correlator software to manage and track hardware configurations.
WVR	Water Vapor Radiometer - an instrument on the telescope which measures the intensity of water vapor absorption in the atmosphere.
Dump	At the end of a correlator switching cycle, raw lags are transferred to the CDP. This transfer is called a dump.
Lag	The integrated product of two digitized signals, one delayed in time with respect to the other. If the two signals are from the same antenna source, then it is an 'auto-correlation product', else if the signals are from different antennas, then it is a 'cross-correlation product.'

2 Requirements

2.1 SSR Requirements

General scientific software requirements are found in [2]. Refinements of the SSR as they apply to the specific details of the correlator hardware and computer systems are found in [3].

2.2 Operations Plan Requirements

Requirements based on the operations plan for the Correlator system can be found in [4].

3 Architecture

3.1 Overview

Figure 1 shows the architectural overview of the Correlator subsystem. It shows the five subsystems to which the correlator interfaces with data flows highlighted.

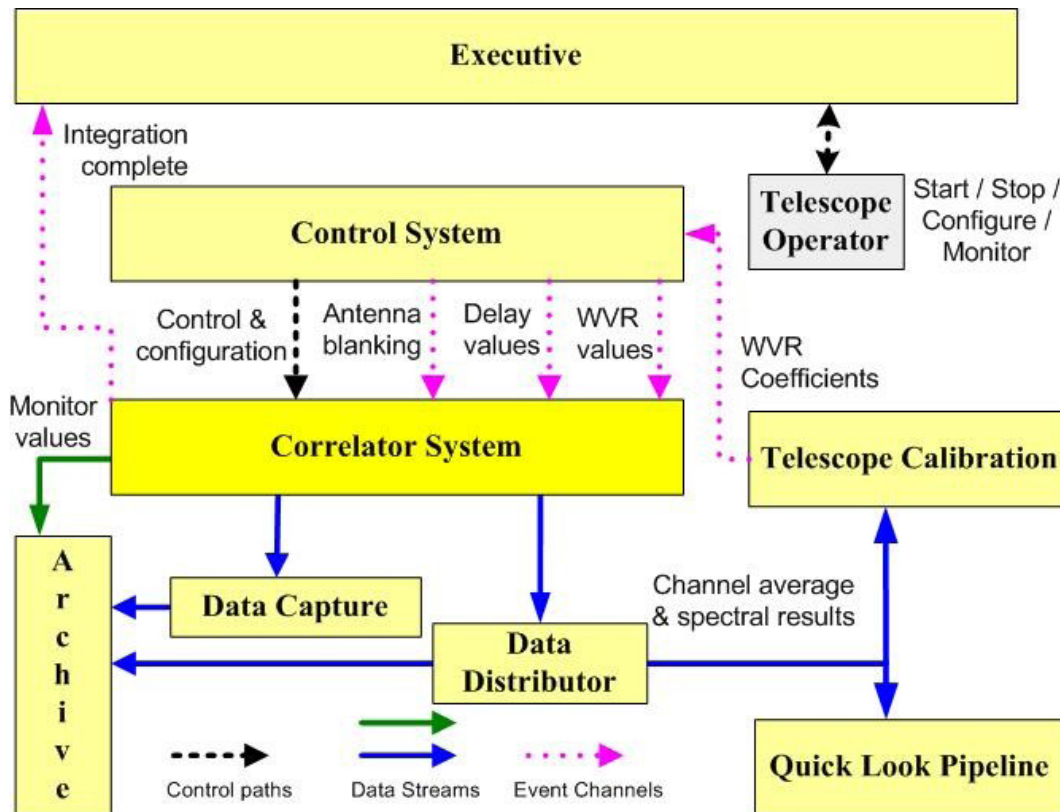


Figure 1 Architecture Overview

The telescope operator interacts with the Correlator system indirectly through the Executive and Control systems. The operator also monitors the status of sub-scans and the correlator via GUI interfaces provided by the Control and Correlator systems. The Control system provides configuration and control information, geometric delay event channels, antenna blanking event channels, and WVR event channels for each antenna. The Executive receives events which signal the completion of integrations assisting it in tracking the progress of sub-scans. The Correlator sends spectral and channel average results to the Bulk Data Distributor, a component of the Archive, which distributes this data to the Archive, Telescope Calibration, and Quick Look. The Telescope Calibration subsystem accepts spectral results and channel average results (the sum of all channels in a bandpass) in order to provide atmospheric phase correction coefficients to the Control system which are passed on to the Correlator subsystem as configuration parameters. The Quick Look pipeline accepts channel average and spectral results to provide real-time images via a real-time filler application which collects the data on an integration basis and produces a sub-scan ASDM for quick look. Finally the Archive accepts spectral results, channel average data and correlator monitor data (via the Control system).

The Correlator subsystem is divided into two computer systems: the CCC and the CDP (see section 3.10 for a description of the physical architecture). The CCC is responsible for controlling and monitoring the correlator hardware and the CDP is responsible for processing raw correlator lag data. ACS [5], a middleware layer which is based on CORBA, provides a variety of services and is used extensively throughout the Correlator subsystem. These services include remote function calls, message logging, error reporting, device property monitoring,



CORBA notification service, and high-speed data transport mechanisms. TAO [6] is the ORB used exclusively for the CCC and CDP computers.

Both the CCC and CDP have ACS Containers which provide interfaces to their contained components and are accessible as distributed nodes in the sense of the CORBA paradigm. They implement the *porous* container design pattern due to their real-time needs, i.e., the ACS Container allows for direct access to the components. It is important to note that only the CCC is a ‘public’ device, i.e., it is the only device accessible to computers outside of the Correlator subsystem. In general, the CDP is not accessed from external computers so as to minimize the interfaces, although there may be exceptions to this rule, for example the Control system directly obtains CDP computer monitor data from the CDP nodes and the CDP directly subscribes to the Control system’s antenna blanking and WVR data channels. The CCC and CDP communicate with one another and the CCC acts as a communication gateway to the CDP for configuration and control.

The correlator device follows a ‘configure-and-run’ model where the hardware is configured for a specific sub-scan and operates in that mode until it is reconfigured or commanded to stop.

3.2 CCC Packages and Functional Overview

Figure 2 shows package diagrams for the CCC. The lines that connect the packages imply a functional interaction between the packages. A brief summary of each package follows.

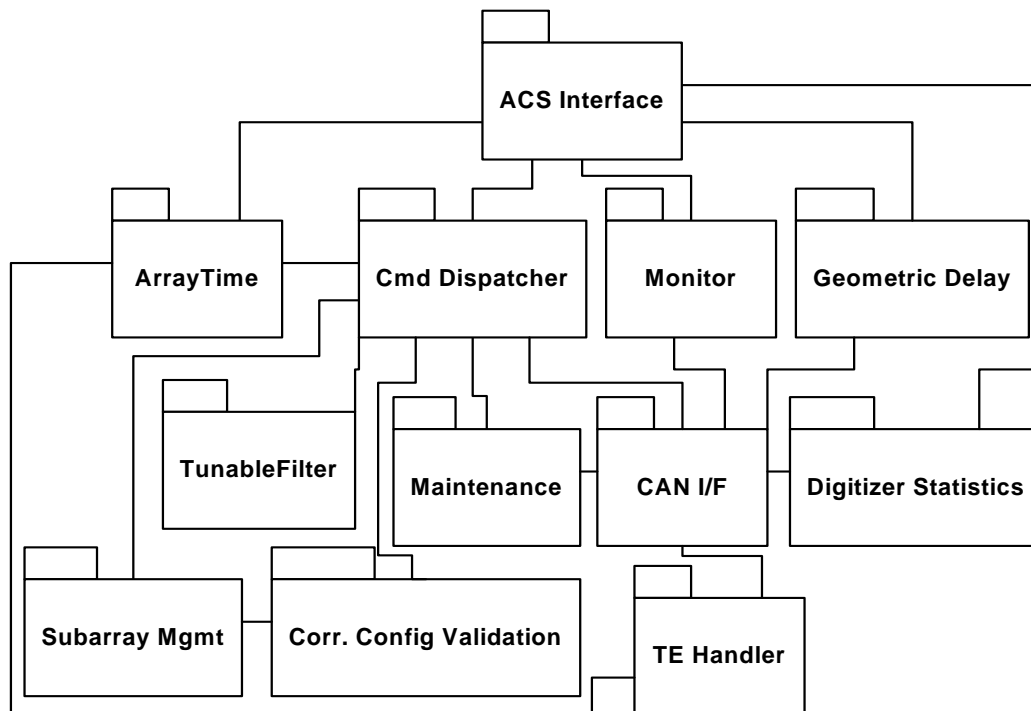


Figure 2 CCC Package Diagram

3.2.1 ACS Interface

This package provides an interface layer needed for common ACS services which provide a wrapper for the ACE Orb (TAO), CORBA remote invocation services, logging services, configuration database, notification services and ACS properties. This package has the following responsibilities:

- instantiates the distributed object component and connects to the ACS Container utilizing initial values from a configuration database



- subscribes to the notification channel that supplies geometric delay events
- instantiates CORBA reference objects for the `alma.Correlator.ObservationControl`, `alma.Correlator.ObservationQuery`, `alma.Correlator.CCC_Monitor`, `alma.Correlator.Maintenance`, and `alma.Correlator.ArrayTime` interfaces.

3.2.2 Monitor

This package is responsible for constructing ACS monitor properties for the correlator hardware and the CCC. As properties are self-contained within ACS, there is actually little to describe. We use CAN and memory properties. This package implements `alma.Correlator.CCC_Monitor` interface. The `alma.Correlator.ObservationQuery` interface can be implemented as a set of properties. It is envisaged that this interface will most often be used in debugging the correlator software during development and commissioning.

3.2.3 Command Dispatcher

This package is responsible for routing external commands to the correlator hardware or the CDP. Commands to the correlator are routed through the CAN I/F package. This package also processes the correlator configuration which can be found in the *CorrConfigIDL.idl* file.

3.2.4 Tunable Filter

This package is responsible for configuring and monitoring the tunable filters for a sub-scan. There are 2 TFBs per antenna per baseband pair. Each TFB contains 32 digital FIR filters which can be tuned anywhere within the 2 GHz baseband with a resolution of 30.5 kHz ($2 \text{ GHz}/2^{17}$). These filters (or sub-bands) can be overlapped and combined, also referred to as stitching, to provide different resolutions and bandwidths. These stitched sub-bands are referred to as *spectral windows*. There can be multiple spectral windows within a baseband.

3.2.5 LO Offsetting

This procedure is fully described in [7]. Briefly, to remove DC offsets introduced in the digitization process, the first LO in one antenna is offset in frequency some small amount relative to another antenna. This offset is removed in the TFB during a subscan configuration as part of the TFB sub-band tuning. The amount of offset is fixed during a sub-scan with the magnitude dependent on an antenna index. The offset direction depends on which sideband (USB or LSB) is used. LO offsetting can also be used to separate sidebands when using double-sideband receivers.


3.2.6 Geometric Delay

This package distributes antenna-based geometric delay parameters to correlator hardware registers. The Control system's geometric delay model server supplies these delay parameters before a sub-scan and as needed throughout the sub-scan. The digitizers at the antennas can accept 3 bits of fractional delay of one sample (at 4×10^9 samples/sec), i.e., 0 – 0.25 ns with a resolution of 32.25 psecs ($0.25 \text{ ns} / 8$). The correlator hardware accepts a 32-bit coarse delay value which represents the remaining delay up to a value of ~1 second at a resolution of 1 sample (250 picoseconds). Any residual delay which shows up as a phase slope across the band is removed in the CDP as discussed in section 3.8.3.8.

3.2.7 Digitizer Statistics

This package is responsible for obtaining the digitizer statistics and delivering them to the Control system in order to set the levels of the downconverter attenuators. The Control subsystem requests the digitizer statistics for a given antenna and for a specific integration interval during a total power detector calibration. The interface between Control and Correlator will be a CORBA function call returning the relevant data.

3.2.8 Array Management

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 13 □ of 86 □
--	--	--

This package manages the current array definitions used within the correlator hardware. There will be up to four array slots which can be assigned lists of antennas. an array in this context means a list of CAIs which are configured and controlled as a unit. Note that for the correlator, an *array* is identical to a *sub-array*. This package:

- Assists with correlator configuration validation.
- Associates array identifiers to a group of CAIs for correlator hardware configuration, etc.

3.2.9 Maintenance

This package is responsible for managing maintenance functions with the correlator hardware and CCC computers. Maintenance implements the *alma.Correlator.Maintenance* interface as specified in the Correlator ICD. Examples of maintenance for the correlator hardware would be the upgrade of FPGA images or microprocessor code both of which are downloadable to on-board flash memory. Maintenance also performs diagnostics and reports their outcome.

Thirdly, this package is responsible for configuration information related to the moving of an antenna to a new pad. The physical antenna label is mapped to an appropriate correlator antenna input. This mapping also involves an optical patch panel at the AOS building which connects optical fibers from a given pad to a correlator antenna input. This information is obtained from the telescope configuration data base once it is updated by the Control subsystem and then notifies the Correlator subsystem of any changes. Following this antenna move, a test procedure will be run to ensure that the baseband fibers are correctly connected to the correlator antenna inputs. In this procedure, the DTS transmitter sends a known pattern to the corresponding DTS receiver and verifies that the pattern is correct. This test is managed by the Control subsystem.

Most maintenance functions occur when the correlator hardware and computers are off-line in a non-operational mode. There will be station-based tests which test the signal path from the output of the filter card through the correlator chips and to the LTA that can run while a given antenna is moving to a new position. This will not require reconfiguration of the correlator.

Obtaining digitizer statistics is another ‘maintenance’ function which can run during on-line mode. The filter cards measure the count of samples at each level which are then transmitted to the Control subsystem as discussed in section 3.2.7.

3.2.10 CAN I/F

The *CAN I/F* package is really a group of packages: *CAN_Cmds*, *CAN_Manager*, and *CAMBServer*. These packages are responsible for the creation, formatting, transmittal, and reception of CAN messages to/from the CCC and to/from the correlator-antenna master bus (CAMB) nodes (LTA, SCC, QCC, final adder, and data port interface). In the Correlator subsystem CAMB consists of 5 individual physical CAN buses referred to as channels that provide the CAN message medium for all control and monitor functions of the correlator. Each node is uniquely identified by its serial number or alternately, channel and address within the specific channel. Because of multicast message format the node addresses are limited to 0 – 47, but this should not be a problem as each bus will have no more than 38 nodes – see [8] for details.

The *CAMBServer* package is responsible for providing a single point of access to all the channels of the CAMB, transmitting CAN messages generated by the *CAN_Manager* and *CAN_Cmds* packages at the specified time-stamps, and providing monitor and other CAMB reading functions. The *CAMBServer* package also provides the CAN drivers, synchronization, and queuing mechanisms necessary for the orderly transmittal/reception of CAN messages. To do so requires that some (hard real-time) software modules of the *CAMBServer* reside in kernel space while others (soft real-time) reside in user space.

The *CAN_Manager* package is responsible for instantiating CAMB nodes that are responsible for handling their node-specific messages and a single multicast message handler (see *Detailed CCC Package Descriptions* sec-



tion). Additionally, the *CAN_Manager* package is responsible for encapsulating the functionality of each CAN channel (e.g. keeping track of the status of each channel, resetting channels as needed, keeping lists of nodes attached to the channel, sending/receiving CAN messages, etc.).

The *CAN_Cmds* package provides classes that encapsulate various commands that are then translated into one or more CAN messages for use by instantiated nodes or the single multicast message handler. These classes have been grouped according to their intended node destination type (i.e. LTA, SCC, QCC, final adder, or data port interface) as shown in Figure 3.

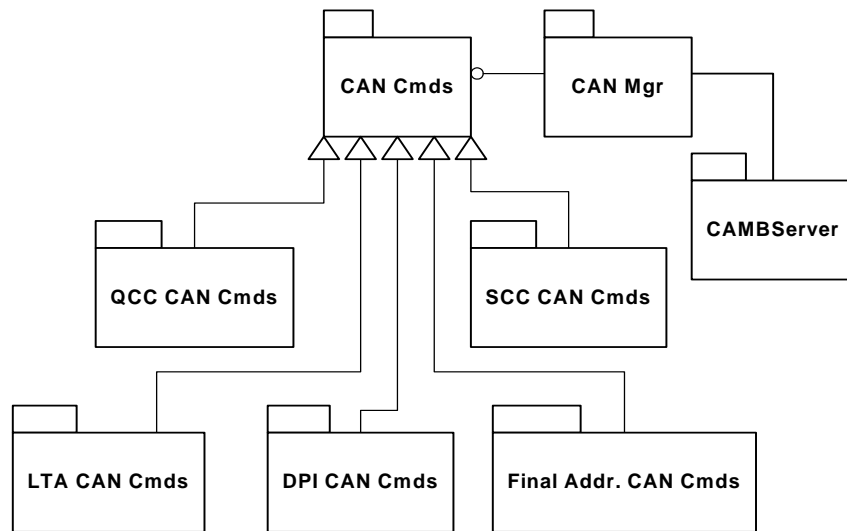


Figure 3 CAN Interface Package Diagram

3.2.11 Array Time Interface


Array time is the fundamental time system of the interferometer utilizing TAI (International Atomic Time) and is maintained centrally by a master clock computer (ARTM). Distributed clocks in the CDP and CCC are slaved to the master clock and kept synchronous by utilizing an array-wide optical fiber network which transmits pulses every 48 milliseconds called *timing events* (TE).

Also this package implements the interface specified by *alma.Correlator.ArrayTime*. Array time is set by first having the ACC provide the distributed clock with a *TimeSource* object that it can use to request the array time of the next TE from the ARTM. The ACC checks that the distributed clock's array time is synchronized and, if not, commands the distributed clock to retry obtaining the array time from the ARTM. For details on the control subsystem side, see the Control subsystem ICD.

This interface is shared between the Correlator and Control subsystems.

3.2.12 TE Handler

This package encompasses the TE hardware signal handler. An interrupt service routine is triggered by each TE pulse and updates the local clock's internal version of array time by 48ms. The *TE Handler* must ensure that no TE is missed as this obviously corrupts the distributed clock's version of array time. A mailbox distributes all relevant clock data (CPU frequency, time stamp clock at last tick, number of total ticks, absolute time at tick zero, etc.) at every handled tick. If a TE is missed then the distributed clock switches to a degraded SOFT mode in which the CPU frequency is frozen to its latest available estimate, and ticking of events is mimicked by sleeping

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 15 of 86
--	--	--

the handler for 48ms each time. The ACC can query the handler state and, if necessary, cause the distributed clock to reset itself when the hardware signal is available again.

The correlator hardware can provide both a 48ms and a 16ms tick to the CCC and CDP computers. The CCC will use the 48 ms tick via its parallel port. Although the CAN nodes will not explicitly track array time, the CCC will keep track of the CAN nodes' current TE count in order to specify a specific TE on which to execute a command.

3.2.13 Correlator Configuration Validation

There are two types of correlator configuration validation covered by this package: *static* and *dynamic*. Static validation performs a simple check that all of configuration parameters are reasonable. This static checking is written in Java and used by the Observation Preparation Tool (OT) to ensure that all configurations are within the bounds of current correlator hardware capabilities, e.g., number of antennas, number of basebands, BBCs, etc. Current and future correlator hardware capabilities reside in the TMCDB which is accessible to any subsystem which needs this type of information. The static configuration validator can be used as a Java library directly by the OT and as an ACS component by the CCC as there may be cases when Control creates correlator configurations in manual mode that are sent directly to the Correlator.

Dynamic validation determines if the configuration fits into the current running state of the correlator hardware or if two configurations scheduled for the same time conflict. This is a complex process which can only be evaluated at run time and is handled by the CCC Command Dispatcher package.

3.3 Detailed CCC Package Descriptions

This section provides detailed class diagrams and descriptions of the more complex CCC packages.

3.3.1 CCC Command Dispatcher

The command dispatcher provides distribution of configuration and control commands for all internal devices of the Correlator subsystem – the correlator hardware, CCC and CDP master. This division provides a single point of control for external subsystems.

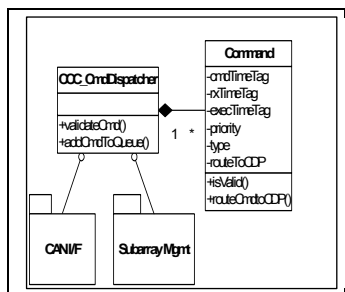


Figure 4 Command Dispatcher Package Class Diagram

The Command class is the base class for all commands that are received via the *ACS CCC Interface* package. *CCC_CmdDispatcher* validates commands or flags and rejects erroneous commands. Time tags detailing when the command was received (*rxTimeTag*), when it should be executed (*cmdTimeTag*), and when the command was actually executed (*execTimeTag*) are assigned as needed. Note that commands are not scheduled to execute, this is done by the *CAMB_Server*. If no time tag accompanies the command, it is put in low-priority queue which transmits the message as soon as possible. It is important to note that no downloads are required for standard correlator observing modes.

3.3.2 Geometric Delay



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 16 of 86

This package is responsible for receiving quantized delays from the geometric delay event channel and distributing them to the correlator hardware.

At the beginning of a subscan and on a periodic basis during a sub-scan, say every 30 – 60 seconds, the geometric delay model server, which runs at the OSF, transmits a set of geometric delay parameters with time stamps which define a valid temporal range, antennas for which they apply, the fine and coarse quantized delay values, and a set of total delay values suitable for polynomial interpolation for a given array. Each delay value includes an offset in milliseconds relative to the overall starting time of the valid interval. From the Control system's IDL, we have the following:

```
struct FineDelayValue
{
    unsigned short relativeStartTime; /// Relative ms till these are valid
    octet fineDelay;                /// Three bits of delay values
};
struct CoarseDelayValue
{
    unsigned short relativeStartTime; /// Relative ms till these are valid
    unsigned long coarseDelay;        /// Quantized delay (250 ps samples)
};
struct TotalDelayValue
{
    unsigned short relativeStartTime; /// Relative ms till these are valid
    double totalDelay;                /// 64 Bits of delay value
};
typedef sequence<FineDelayValue> FineDelayValueSeq;
typedef sequence<CoarseDelayValue> CoarseDelayValueSeq;
typedef sequence<TotalDelayValue> TotalDelayValueSeq;
```

```
struct DelayTable
{
    ACS::Time startTime;
    ACS::Time stopTime;
    TotalDelayValueSeq delayValues;
};
```

```
struct AntennaDelayEvent {
    ACS::Time startTime;
    ACS::Time stopTime;
```




```
string antennaName;  
CoarseDelayValueSeq coarseDelays;  
FineDelayValueSeq fineDelays;  
DelayTableSeq delayTables;  
};
```

Coarse delays are sent to the correlator hardware after conversion to CAN messages. The time at which the messages are sent are deduced from the *startTime* and the *relativeStartTimes* by converting the array time to a TE plus some offset in the range of 0 – 47 milliseconds after the specific TE. The fastest expected rate of update for quantized (whole sample) delays for a given baseline is 172.8 milliseconds [9], but with arrays, the CCC could be sending many delay updates per TE.

If delay events arrive late, i.e., their start time has passed, then an error is logged and correlator data are blanked by the CDP (discussed below in section 3.8.1). The correlator retains the current quantized delay value until it is updated. It is assumed that delay updates for a given antenna will be contiguous in time. Any gaps between the *stopTime* of one delay event and the *startTime* of the subsequent delay event will be an error causing data to be blanked (in the CDP) during that missed interval.

Care must be taken when new delays arrive due to the changing of a source in that old delays are replaced by the new delays for the new source.

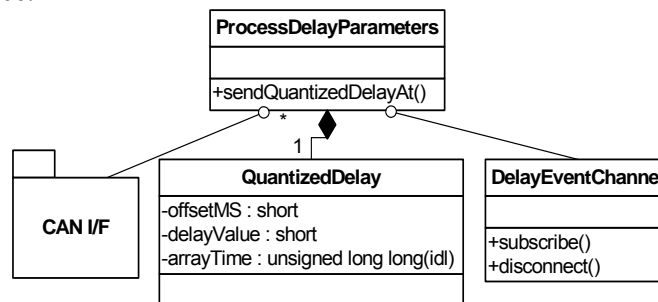


Figure 5 CCC Geometric Delay Class Diagram

ObservationControl::delayEventHandler subscribes to the geometric delay event channel (via *DelayEventChannel*) receiving the *TotalDelay* events and performs two things. First, if the *TotalDelayEvent* is valid, i.e., received in time, it sends it off to the CDP via a notification channel. Secondly, it extracts the relevant information from the *TotalDelayEvent*, constructs *QuantizedDelay* which is then transmitted to the correlator hardware via CAN commands. The sequence diagram for applying the geometric delay is shown in Figure 6. The geometric delay model server supplies initial delay values before the sub-scan starts and as needed during the sub-scan.

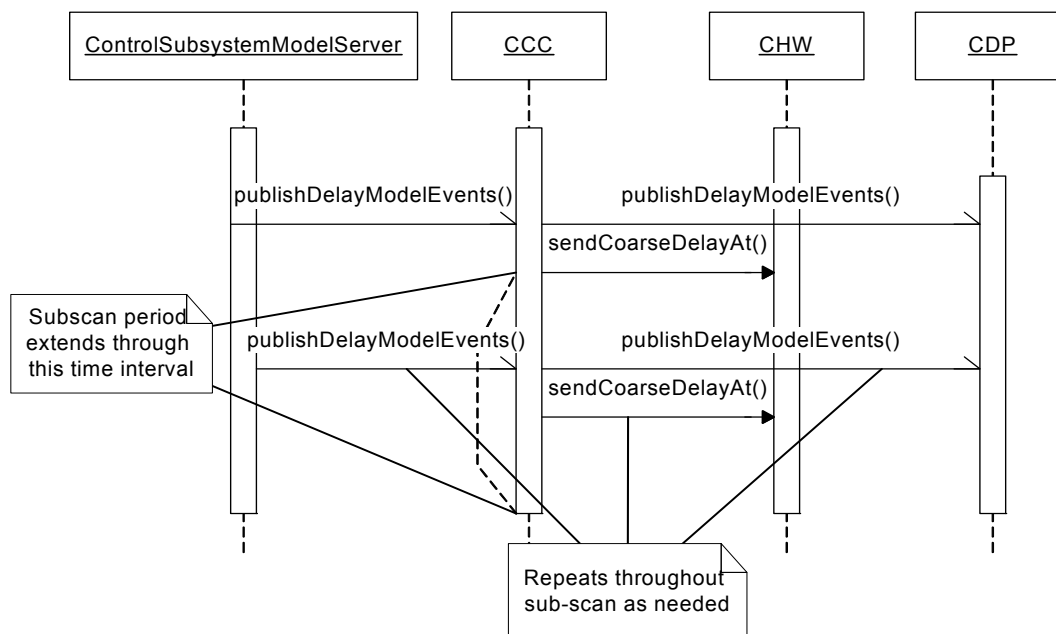


Figure 6 Geometric Delay Sequence Diagram

3.3.3 Array Management

Array management essentially maintains a list of arrays each defined by an identifier, a list of antenna labels with corresponding 0-based correlator antenna inputs. **IMPORTANT: The antenna labels from the Control system must be mapped to correlator antenna inputs (CAIs) which are inputs to the correlator chips.** The CCC and the Control system must coordinate the mapping of fixed antenna labels to pad identifiers which in turn must be mapped to correlator antenna inputs. This mapping also involves an optical patch panel at the AOS building that connects optical fibers from a given pad to a correlator antenna input. This mapping is maintained by the telescope and monitor configuration database (TMCDB) and updates to the mapping will be signaled in the *alma.Correlator.Maintenance* interface.

The *AntennaList* class maintains a sequence of the following structure:

```

struct antennaNumberMap_t
{
    string antennaLabel;      /// ALMA-wide antenna label as a string
    short correlatorAntennaInput;  /// the corresponding CAI
};
  
```

which maps antenna labels to CAIs. Correlator antenna inputs are used to configure the correlator hardware while the corresponding antenna labels are used by the Control system when configuring the correlator for a subscan.

ArrayManager creates and destroys *AntennaList* objects as arrays are formed or deleted. Control notifies the correlator when to create arrays with a name and list of antenna labels and when to destroy the arrays. Array names are subsequently used for correlator configurations. It also assists in dynamic correlator configuration validation by determining the intersection of two *AntennaList* objects. Recall that there are at most six arrays for ALMA, but we can have a subscan which has all antennas in auto-correlation mode only which can be slightly offset from one another. The resulting data sets, which contain all data in a single binary BLOB, can then be processed individually in post-processing software. There is no special requirements for the correlator software to produce this type of data.

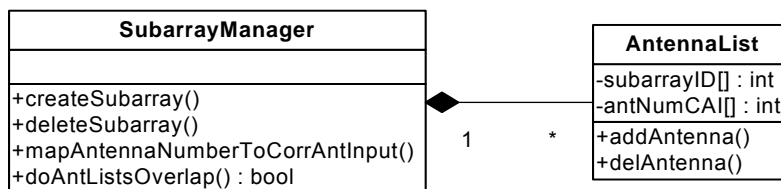


Figure 7 Array Management Package Class Diagram

3.3.4 Monitor

The Monitor package provides status information about the CCC software, hardware, and configuration. This package contains three sub-packages: Correlator hardware monitor, correlator configuration monitor and CCC hardware monitor.

The Correlator subsystem is regarded as a device of the Control system. It is the responsibility of the Control system to retrieve all of the Correlator subsystem's monitor data and save it in the MonitorStore component of the Archive. This applies to CCC and CDP monitor points.

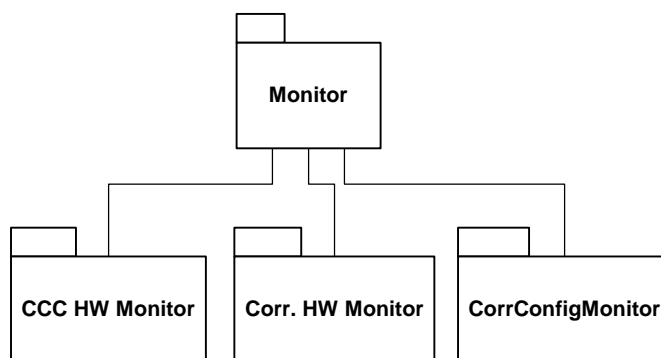


Figure 8 CCC Monitor Packages

3.3.4.1 Correlator Hardware Monitor

These are the “standard” monitor properties for the correlator hardware like voltages, temperatures, etc. They are managed through the QCC for each quadrant consequently providing 4 sets of voltages and temperatures.

It is important to note that the correlator hardware is self-monitoring in the sense that if an out-of-voltage, or over-temperature condition is encountered which may damage the hardware, it shuts itself down. How much of the hardware is powered down is still TBD. For now, the CCC monitoring software detects these error conditions and delivers monitor data to the monitor store which can then be utilized to flag bad data. Data are blanked in this case. This leads to ‘correlator quadrant blanking’ where data are blanked on a quadrant basis. The CCC then notifies the CDP which quadrants are malfunctioning with the CDP ignoring any data from the affected quadrant.

3.3.4.2 CCC Hardware Monitor

Hardware dependent code retrieves monitor information from the CCC computer hardware. The current CCC hardware uses a rack-mounted PC chassis which has voltage, temperature, and fan speed monitor data available as properties.

3.3.4.3 Correlator Configuration Monitor

This monitor keeps track of correlator hardware configuration including but not necessarily limited to:



- card serial numbers – every card in the correlator has a serial number chip to uniquely identify it, but many of these cards are not CAMB nodes, although their identification information can be accessed via LTA or SCC nodes. These cards are: correlator (8 per LTA), tunable filter bank (8 per SCC), station (4 per SCC), correlator interface (8 per LTA), station interface (2 per SCC), power supply (2 per SCC, 1 per LTA).
- FPGA checksums (16 per LTA, 8 per SCC, 3 per final adder, 3 per DPI card).
- firmware checksums (3 per each LTA, SCC, final adder card, DPI card, and QCC).
- quadrature parameters (16 per LTA, 8 per SCC). Quadrature parameters define signal phasing for internal correlator cables which connect station cards to correlator cards.
- QCC timing event tracking status, PLL and DLL status.

3.3.5 Correlator CAN Commands

The CCC utilizes a CAN interface to communicate with the correlator. It is important to note that the protocol used by the correlator hardware as described in [10] is related to the ALMA M&C protocol defined in [11] in that it supports some of the basic ALMA M&C functions like node identification and its master-slave architecture, but not the timing specifications. More importantly, the correlator CAN protocol differs from the ALMA M&C protocol by providing multicasting of CAN messages to a range of CAN nodes and transmittal/reception of multiple packet data structures. Multicasting is a powerful tool for the correlator as there are many nodes which require identical information allowing efficient configuration and control.

The class diagram (Figure 9) shows the layout of basic CAN protocols as a base class and *mix-in* classes from which specific commands inherit. The base class, *CorrCanCmd* contains generic functionality used by the *CAN I/F* package. The leaves of the class hierarchy then define the specific command either as a multicast or singlecast command with *CANMulticast* or *CANSinglecast* providing the necessary addressing capability.

The currently available CAN commands are listed in the Appendix [9.3].

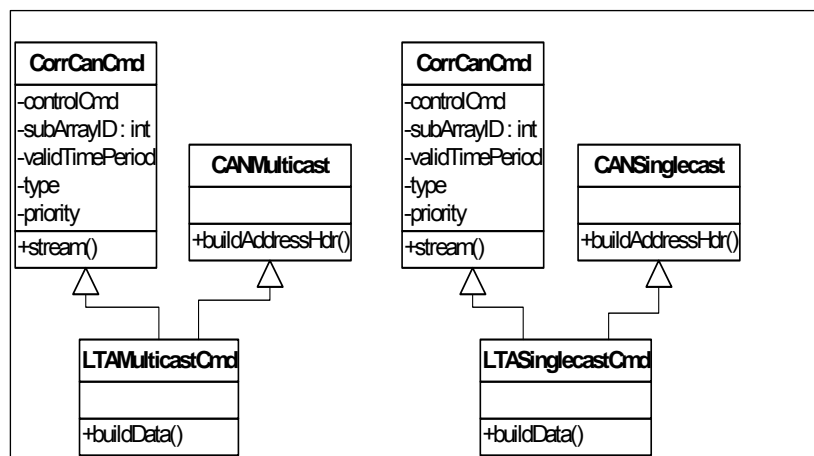



Figure 9 CAN Commands Package Details

3.3.5.1 LTA CAN Commands

Software associated with CAN commands to configure, control and monitor the LTA, correlator, correlator interface, and correlator bin power supply cards (see [12] for details).

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 21 □ of 86 □
--	--	--

3.3.5.2 SCC CAN Commands

Software associated with CAN commands to configure, control, and monitor the Station Control Card, station, station interface, filter cards, and station bin power supply cards.

3.3.5.3 QCC CAN Commands

Software associated with CAN commands to monitor the quadrant control card which provide voltage, current, and temperature monitors for each quadrant.

3.3.5.4 Final Adder Commands

Software associated with CAN commands to monitor and control the final adder card. The only CAN command functions are serial number/checksum verification and FPGA personality/code downloads.

3.3.5.5 DPI CAN Commands

Software associated with CAN commands to monitor and control the data port interface. The only envisioned CAN command functions are serial number/checksum verification and FPGA personality/code downloads.

3.4 Sub-scan Control Sequence

Figure 10 is a sequence diagram showing the time-ordered sequence for a configuration and start of a sub-scan with the Correlator subsystem from the viewpoint of the CCC. Geometric delay configurations were covered earlier and are omitted here. Note that there is some interaction with the CDP here. This will be discussed in further detail.

The ACC uses the CORBA *alma.Correlator.configureSubScan()* function to send a complete correlator configuration with an antenna list and a time stamp as to when the correlator starts using this configuration – this can be viewed as 'start observing with a specified configuration'. The *ArrayManager* class defines an array from the antenna list which subsequent configuration and control commands for the sub-scan use to refer to this array.

Once the configuration is received and validated by the CCC, it relays the *configureSubScan()* to the CDP. The CDP then configures itself at the correct TE in synchronization with the correlator hardware being configured at the same TE so that lags are correctly processed according to the new configuration. At the appropriate TE, the CCC commands the correlator hardware to use the new configuration. The sub-scan continues until the requested number of integrations is reached or the sub-scan is commanded to stop. The correlator hardware needs time to set up internal program words, so the configuring and starting of an integration is two-stepped: first the configuration is downloaded to the correlator followed by a 'use configuration' command delayed by 1.5 seconds. This assumes that reception of the configuration command from the ACC takes place at least 1.5 seconds before expected usage.

This 1.5 second delay between the start of subscans can be avoided by pre-loading configurations. The correlator has enough 'configuration registers' available such that four arrays can be have 2 configurations preloaded for each array. The correlator can then be commanded to toggle from one configuration to another on a give TE boundary for the duration of the subscan. One could also increase the number of configuration registers and proportionally decrease the number of arrays. For example, pre-load 4 configurations into 2 subarrays allowing them to switch sequentially among the configuration registers on TE boundaries. The one caveat is that for each group of correlator configurations to pre-load, one must still allow 1.5 seconds for each configuration, e.g., pre-loading of 2 configurations must be done 3 seconds in advance of their use.

Important Caveat: I have recently (July 2008) found out that the ALMA-B correlator CANNOT switch between 2 FDM configurations without delay. It can switch between a TDM & FDM configuration without delay, but re-



quires up to 1.5 seconds to switch between 2 FDM configurations. This must be brought to the attention of the SSR, CONTROL & ObsPrep.

The total number combinations of arrays with different configurations must be less than 16. If the number of arrays is S and the number of configurations is C , then

$$S^C \leq 16$$

Where S is 1 – 4 and C is 1 – 4. Note that we can have separate arrays which do not toggle the configurations. For example, S is 2 and C is 4, two more arrays can be configured as long as they don't change.

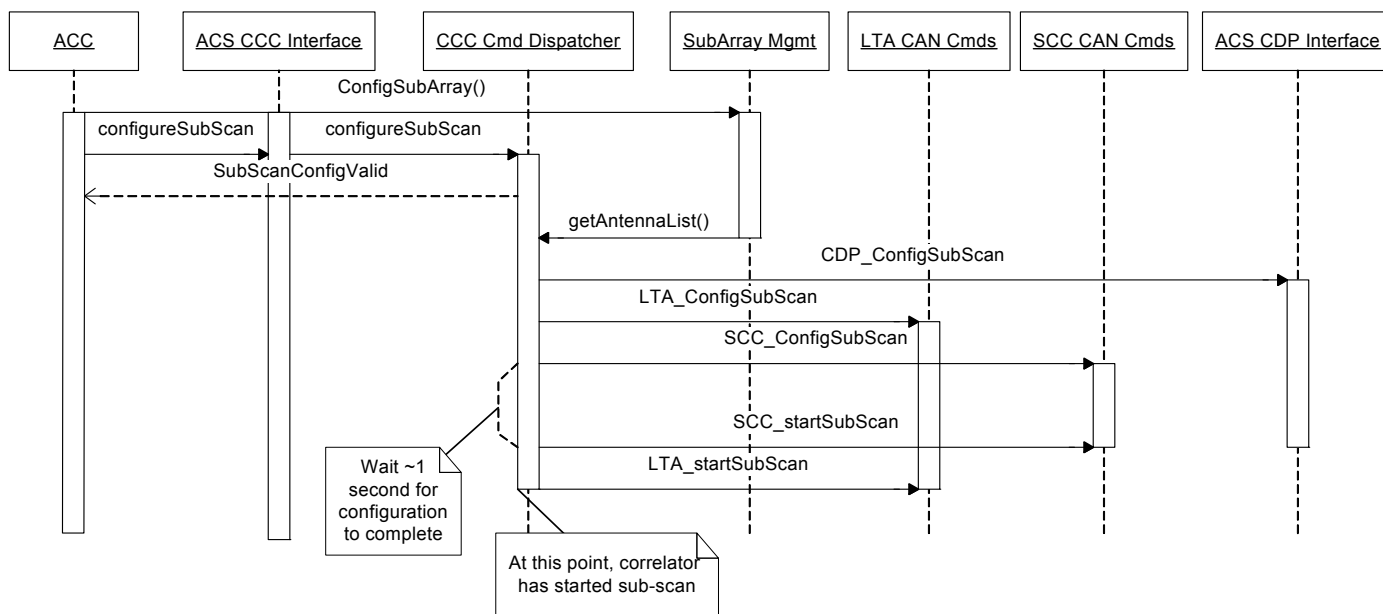


Figure 10 Sub-scan Configuration Sequence Diagram

3.5 Preloaded Configurations

A requirement exists with the baseline correlator and control software to enable fast switching between two or more different correlator configurations and subscans for an array. The correlator hardware and software requires about 1.5 seconds to configure itself for a subscan. There are some scientific cases where it is desired to switch between 2 different configurations on a TE boundary without the 1.5 second overhead. This section describes the approach to preload correlator configurations which can then be rapidly switched. **Figure 11**

shows this switching pattern between 2 configurations.

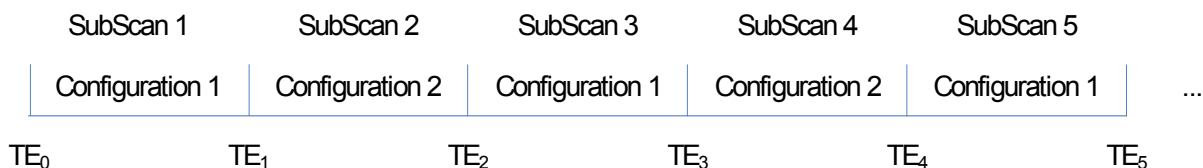


Figure 11

Note that there can be up to 4 configurations to toggle among for a given subscan.



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 23 of 86

The current implementation of correlator configurations is that each subscan is configured for execution in the future. The requirement is for the correlator software to receive a configuration at least 1.5 seconds before it is executed. Many configurations can be queued up for future execution.

This method does not prevent a switching mechanism, but would be tedious for the CCL to implement. It would require a loop like this:

```
for n in range(N_subscans,2):
    define correlator configuration n
    obtain start time of configuration n
    define dataOIDs for configuration n

    define correlator configuration n+1
    obtain start time of configuration n+1
    define dataOIDs for configuration n+1

    configureSubScan(array SA, configuration n, dataOIDs_n, start_time_n,...)
    configureSubScan(array SA, configuration n+1, dataOIDs_n+1, start_time_n+1,...)
```

The idea is to load the correlator configurations and all of the support parameters in advance each with an associated configuration identifier. Then the CCL explicitly commands the correlator to use a given configuration identifier plus subscan durations for a subscan. The goal is to allow as much flexibility to the users as technically feasible. The Control subsystem has also requested this feature as it often reuses the same configuration for multiple subscans and it is wasteful keep reloading configurations when all that is needed is to reuse an existing one. The steps are (thanks to Robert Lucas on this):

- at the beginning of execution, the array is allocated a predefined number of correlator configurations.
- the script defines these configurations from the SB data, and preloads them in the correlator; later they can be referenced by a specialized identifier,
- to start the execution of one or a set of subscans, the CCL passes to the correlator a list of subscan durations and correlator configurations referred by identifier.

```
config1 = configureCorrelator(parameters1)
config2 = configureCorrelator(parameters2)

beginScan()
# do a scan with 4 subscans: 5 sec. on config1, 2 sec on config2, and
# repeat once
listOfConfigs = [config1, config2, config1, config2]
listOfDurations = [2., 5., 2., 5.]
startToggledSubscans(number, listOfConfigs, listOfDurations)
endScan()
```

The following information must be defined for a subscan sequence and provided to the correlator:

- The correlator configurations which include dump and integration durations.
- The subscan duration for each configuration. Computations would be required to ensure that the subscan durations are such that the subscan end on a TE boundary. Error checking in the correlator software would double-check this.
- The start time of the whole sequence. The stop time can be computed by the duration of each subscan and the number of subscans to execute.
- The data OIDs for all of the integrations and sub-integrations in all of the subscans.



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 24 of 86

- Scan, execute block, and subscan identifiers for all subscans.

Note that the data OIDs, scan, subscan and execution block identifiers are not defined in the CCL, but at script execution time. Also, the list of configurations and durations can be of length one which allows for manual mode scripts.

Here is the required information in IDL form:


```
// One of these for each configuration in the sequence of subscans to
// be preloaded before the subscan begins
struct SubScanConfig
{
    long calibrationSubscanId;
    Correlator::CorrelatorConfiguration corrConfig;
    CorrelatorCalibrationMod::CorrelatorCalibration acaSubscanType;
};
/// One entry per subscan
typedef sequence<SubScanConfig> SubScanConfigSeq;

/// One of these for each configuration in the sequence to be used at run-time
struct SubScanInfo
{
    long subScanNumber;
    long subScanNumber;    /// the subscan number (1-based)
    long executeBlockNumber;    /// the exec block number (1-based)
    asdmIDLTypes::IDLEntityRef executeBlockID;
    /** One data OID for each integration & sub-integration in the sub-scan
    ** The uidRange object provides the UIDs for this subscan
    */
    xmlstore::IdentifierRange uidRange;
    /// APC parameters
    Correlator::PathCorrResult_t pathCorrCoefficients;
    /// Configuration identifier to use returned from configureSubScanSequence()
    long corrConfigIdentifier;

    /// Type of subscan
    CorrelatorCalibrationMod::CorrelatorCalibration subscanType;
    /// ACA correlator needs these, ALMA-B doesn't set to 0 & ""
    ACS::TimeInterval relativeStartTime;
    /// ACA-specific calibration information
    boolean refreshHFSCdata;
};
/// The length of this sequence defines the number of subscans in
/// the scan. This can be 1 for manual mode observations
typedef sequence<SubScanInfo> SubScanInfoSeq;

// The function loads the sequenced configurations.
void configureSubScanSequence( in string arrayID, in CorrConfigSeq corrConfigs,
    out ACS::longSeq corrConfigIdentifiers)
    raises(CorrErr::InvalidConfigEx,
        CorrErr::InvalidArrayEx,
        CorrErr::NoAvailableCorrelatorSlotsEx);

/// Sequences of subscans are started (activated) using this function:
void startSubscanSequence(in asdmIDLTypes::IDLEntityRef executeBlockID,
    in ACS::Time subScanSequenceStartTime,
    in SubScanInfoSeq subScans)
    raises(CorrErr::InvalidStartTimeEx,
        CorrErr::InvalidSubScanInfoEx);
```


	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 25 of 86
--	--	--

Note that the `startSubscanSequence()` must be invoke a few (TBD) seconds before `subScanSequenceStartTime` to allow for configuration of the CDP. There is an alternative mechanism in which the correlator starts the sub-scan as soon as possible and returns the sub-scan start time to the caller, referred to as `startSubscanSequenceASAP()`.

Aborting these subscans, that is, stopping them explicitly before their time runs out can be done with calls to `stopSubScanSequence(in ACS::stringSeq corrConfigIds)`, where `corrConfigId` corresponds to the preloaded configurations. Once a subscan sequence is no longer needed, it is removed from the correlator hardware and software memories via `clearConfigurationIds()`.

There are a limited number of ‘configuration slots’ in the correlator which allow for this switching. Internally the correlator has 16 slots available to switch among in this fashion. Configurations are loaded into the correlator which it then builds ‘program words’ for the correlator chips in the array. Once these program word sets are ready, the correlator can be commanded to use a given correlator configuration on a specific timing event. In R6, we plan to support switching between 2 configurations per array.

Configuration of the correlator takes about 1.5 seconds per configuration and is multiplicative factor for all pre-loaded configurations. Also the lead time for `startSubscanSequence()` depends on the number of pre-loaded configurations to use. At this time, I believe that 1 second per configuration is sufficient.

IMPORTANT NOTE: Recently (April 2008) we discovered that there are limitations in the correlator hardware which prevent switching between pre-loaded configurations with a delay. Apparently, part of the correlator firmware can perform this switching on a TE boundary without delay, but other parts do not have this pre-loaded capability in place preventing a configuration switch without a 1.5 second delay. The correlator hardware requirements are still for the 1.5 second delay and will need to be changed in order to fully support the subscan sequence mechanism developed for the correlator software.

3.6 Correlator Configuration XML

In February, a discussion of the representation of the correlator configuration was held. Participants included members of Control, ObsPrep, ACA correlator, and Correlator subsystems. There has always been a problem where Control receives a correlator configuration as an XML entity and then translates it to an IDL structure. Control, rightly so, feels that there is no need for it to be a translator. Consequently, we agreed to the following.

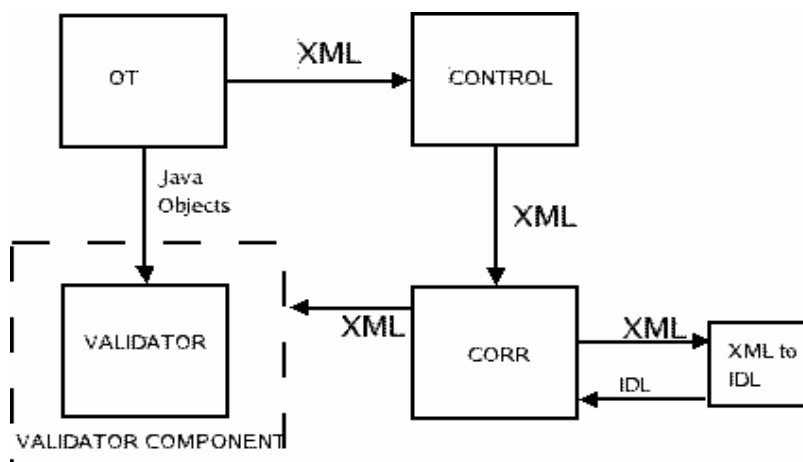



Figure 12 Correlator Configuration Data Flow

- OT

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 26 of 86
--	--	--

OT creates a correlator configuration using its internal representation, denoted here as 'Java Objects'. It validates the configuration directly using the VALIDATOR in a Java library. This Java validator library will reside in ICD/SharedCode and will be jointly maintained by ObsPrep & Correlator. An initial version will be written by ObsPrep based on the currently implemented validator in ICD/CORR. The validator will require a *correlator type* which provides information specific to a given correlator, e.g., ACA or BL and a 'version' of BL correlator, e.g., OSF 2-antenna, first quadrant, etc. This is necessary because of slight differences among the different correlators. OT serializes the configuration within an SB which Control will extract as XML.

- **Control**

Control obtains the XML entity representing the correlator configuration to Correlator via the **configureSubScan()** or **configureSubscanSequence()** commands. It is the responsibility of Control to extract the appropriate configurations from the SB for a given subscan. Note that this requires an API change in *ObservationControl.idl*, the primary interface between Control & Correlator. There is no translation from XML to IDL done by Control

- **Correlator**

Correlator receives the correlator configuration(s) as XML entities directly from Control. The configurations are first sent to the *Validator Component*, a Java container running on the ACC. This Validator Component creates *Java Objects* from XML and then uses the Java Validator to validate *Java Objects*. The Validator Component must accept an array of XML entity structs as Correlator may receive an array of configurations from Control in a **configureSubscanSequence()** call so all the configurations can be validated at one time.

Once the configurations are valid, they are then translated to IDL structures for internal use by the Correlator. This translator will be written in Java and run in a Java container along with the Validator Component. The initial version of this XML-to-IDL translator will be based on the implementation already in place in Control. The responsibility for maintenance of this translator will reside with Correlator with assistance from Control and possibly ObsPrep.

3.7 CDP Packages and Functional Overview

Figure 13 and Figure 14 show packages for the CDP master and compute nodes respectively. A brief summary of each package follows. Table 1 shows deployment details as to which packages exist on the (single) master or (multiple) compute nodes. There are many packages which have functionality on both the master and the compute nodes.

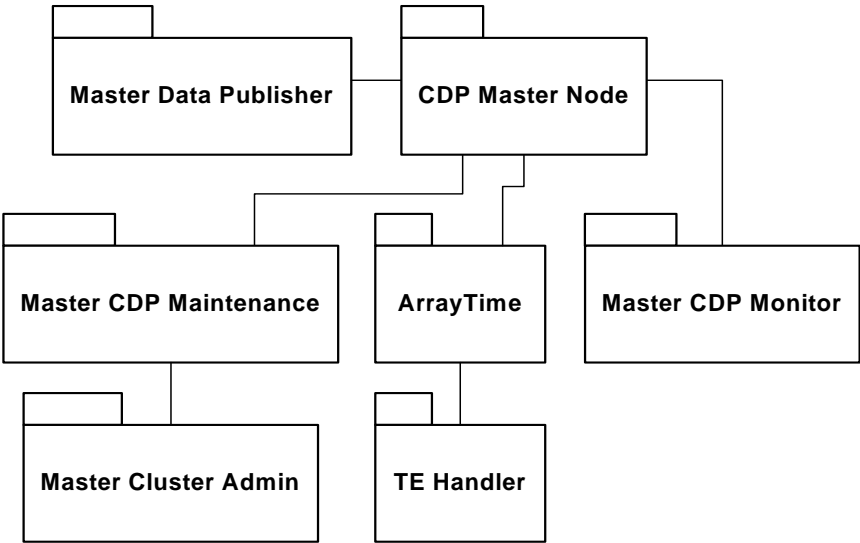
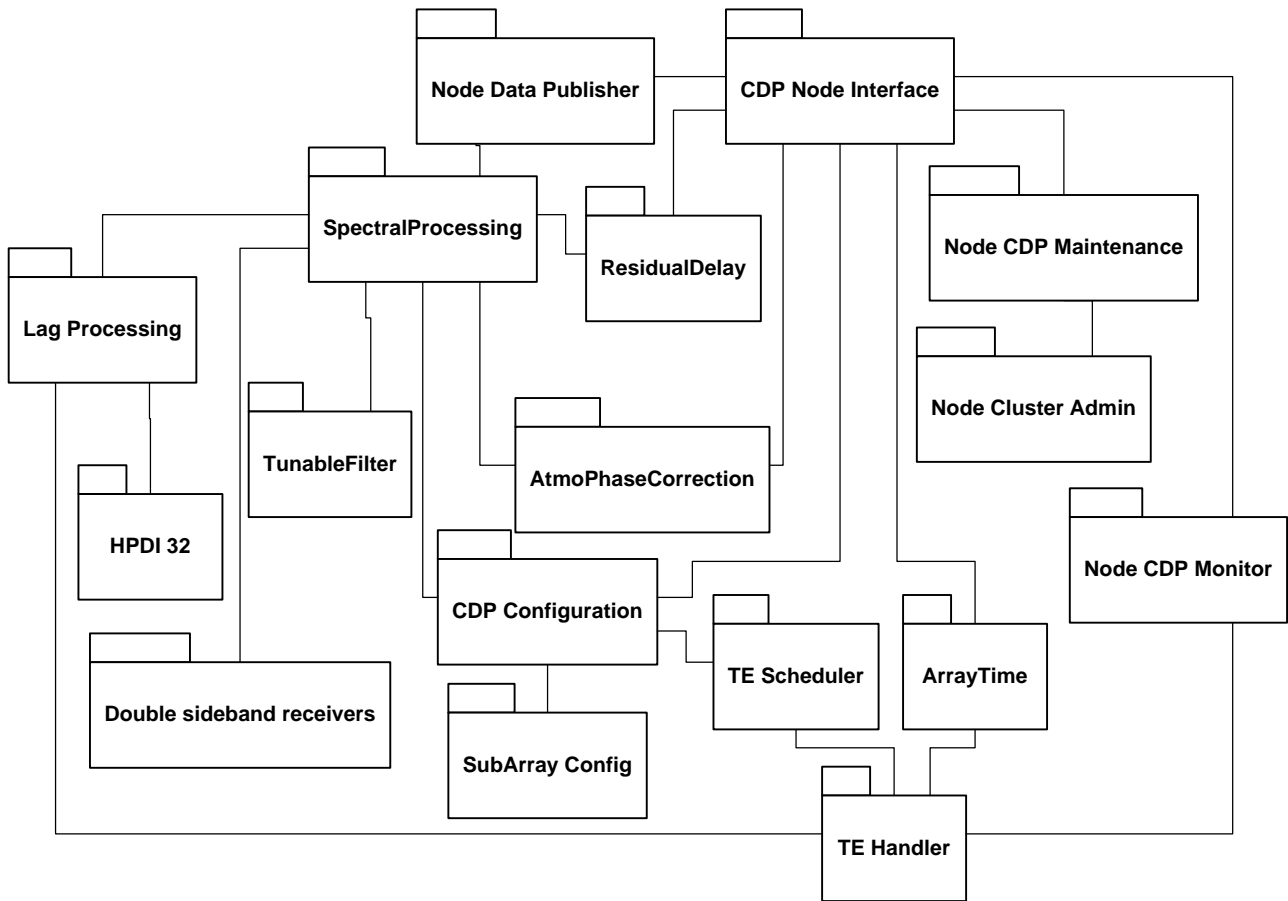


Figure 13 CDP Master Node Package Diagram




	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 28 of 86
--	--	--

Figure 14 CDP Compute Node Package Diagram

Package	Master Node	Compute Nodes	Notes
Array Time	√	√	Identical code in both Node & Master
CDP Monitor	√	√	Similar code in both Node & Master
CDP Maintenance	√	√	Similar code in both Node & Master
Cluster Admin	√	√	Similar code in both Node & Master
TE Handler	√	√	Identical code in both Node & Master
CDP Master Node IF	√		
Master Data Publisher	√		
CDP Node Interface		√	
Node Data Publisher		√	
Spectral Processing		√	
Lag Processing		√	
HPDI 32		√	
Residual Delay		√	
Atmospheric Phase Correction		√	
Sideband separation		√	
CDP Configuration		√	
Array Configuration		√	
TE Scheduler	√	√	

Table 1 CDP Node Package Deployment

3.7.1 Array Time Interface

This package is responsible for interfacing between the TE hardware signal and internal software packages that require synchronization and time stamps. The software functionality duplicates the CCC array time interface discussed previously in section 3.2.11. If a compute node misses a TE, it must signal an error and blank its data while it resynchronizes with array time from the ARTM.

3.7.2 TE Handler

This package is identical to what is used in the CCC, refer to section 3.2.12 for details. The CDP nodes obtain their TE ticks from the correlator via the PCs' parallel port.

3.7.3 CDP Monitor


This package is responsible for constructing ACS properties for the CDP. As properties are self-contained within ACS, there is actually little to describe. The same comments as in section 3.2.2 apply here.

3.7.4 CDP Maintenance

This package is responsible for managing maintenance functions CDP computers. The maintenance package is responsible for performing diagnostics and reporting their outcome. This package implements *alma.Correlator.Maintenance*.

3.7.5 Cluster Administration

This package contains tools and procedures to monitor and administer the CDP master and compute nodes.

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 29 of 86
--	--	--

The master node monitors each compute node to check its health, i.e., ‘pinging’. If the master detects that a compute node has rebooted, it logs this error and reconfigures the affected node.

3.7.6 CDP Master Node Interface

This package provides an ACS interface layer needed between the CCC and CDP. This is a ‘private’ interface in that it should only be used by the CCC. It uses the namespace, *alma.CorrelatorPrivate*, to differentiate it from the ‘public’ ACS interface that external software subsystems can use. *CDP Master Node* determines what target compute nodes need the current command and routes it to them. Although a history of commands is kept here, no queuing is performed, i.e., commands are immediately sent out. Also CDP configurations are saved here in case a compute node needs its configuration parameters reloaded in case the compute node reboots.

This package is responsible for interfacing functional code with the CORBA interface and has the following responsibilities:

- constructs and manages the distributed object component and connects to the ACS container.
- obtains references to the MACI manager.
- routes commands from the CCC to the appropriate compute nodes.
- creates the notification channel which publishes integration events.

3.7.7 Master Data Publisher

This package publishes spectral and channel average data to the BDD using the ACS BulkData package. As each compute node completes the processing of spectral data, it transmits its results to the master node. The master node then assembles all of the baseline results for a given array and integration and publishes it to the BDD. The bulk data format is XML headers plus binary tables. The format for the binary data output is found in [13] and becomes part of the ALMA Science Data Model [14].

The Master Data Publisher also sends summary data of each integration and sub-integration for each sub-scan to the DataCapture component of the Offline subsystem. This interface is defined by the *SubScanCorrelatorData* IDL structure in ICD/CORR.

3.7.8 CDP Node

This package provides an ACS interface between the CDP master and CDP nodes. It exposes the *alma.CorrelatorPrivate.Node* interface. Some of the functionality here includes:

- routes commands to the appropriate CDP node packages
- subscribes to notification channels that supply antenna blanking, geometric delays and WVR receiver values.
- constructs Bulk Data streams and flows to the master node for delivery of spectral and channel average data.

Each CDP node implements all of these packages.

3.7.9 Node Data Publisher

This package is responsible for delivering spectral results for each integration and channel average results for each sub-integration. Each binary block of data contains a header which uniquely identifies it by baseline, polarization product, APC corrected/uncorrected, correlator bin, integration ID integration duration and integration time. This header information is required by the CDP Master node to assemble the final output it sends to the BDD. These data are transmitted from the CDP Node to the CDP Master via the ACS BulkData system due to the high data rates. Each CDP node computer creates one stream which can hold up to 12 flows; one for spectral data and one channel average data for up to 6 arrays.



3.7.10 Spectral Processing

This package manages processing of lag sets into raw spectra. Lag sets are received from the *Lag Processing* package and converted to spectra. Spectral processing details are discussed in section 3.8.3. Note that for the tunable filters, extra processing steps are required as found in [15] which are:

- 4- or 16-level quantization correction
- Smoothing or windowing function applied
- $\frac{1}{2}$ channel frequency shift
- Bandpass calibration
- Scaling sub-band for relative total power
- Sub-band stitching

3.7.11 Lag Processing

This package is responsible for buffering raw lags from correlator hardware. It receives the raw lags from the *HPDI32* package after each correlator dump and queues the lags in the order received and by correlator bin number. These lags remain in this fashion for a certain period (nominally one second) allowing for antenna blanking which can remove lags as needed (see section 3.8.1 for details). The output of this package is groups of lags called 'lag sets' which are processed into spectra with the *Spectral Processing* package.

3.7.12 HPDI 32

Raw lags are sent from the correlator LTA via 32-bit data port interfaces to each compute node. Low-level drivers (running as real-time tasks within RTAI) transfer the raw lag data to a shared memory area via DMA transfers and signal other tasks to begin the lag processing once the DMA transfer completes.

3.7.13 Residual Delay

This package is responsible for receiving *TotalDelayEvents* from the CCC and removing any phase slope across the band pass due to any residual delay not removed by the digitizer or correlator hardware.

3.7.14 Atmospheric Phase Correction

This package performs atmospheric phase correction. The Control subsystem provides path length correction coefficients from TelCal along with information to transform the frequencies from sky frequencies to intermediate frequencies as part of a sub-scan configuration. The CDP subscribes to an event channel where each antenna provides its WVR receiver values. The CDP applies the path length correction to each channel of the spectral results with the receipt of each new set of WVR receiver values.

3.7.15 Sideband Separation

The use of double sideband receivers requires that separate phase states must be binned and summed separately. Also sideband separation can sometimes be used on single sideband receivers to measure the image sideband rejection. Four bins are required for the 0° , $+90^\circ$, and -90° combinations. As the correlator hardware is unable to calculate the -90° combinations, the CDP is responsible for calculating and summing them.

An interface is required to determine the phase of each antenna for each 16 msec sample, (e.g. the start time and 64 patterns, each 64 states long). The baseline phase is computed from the antenna phase (90° , 0° , -90°) with the results summed into two bins (90° , 0°) and normalized correctly. The -90° is a subtraction in the 90° bin. Once the integration is complete, the upper and lower sideband visibilities are formed with:

$$V_{\text{lower}} = (V_{0^\circ} + i \cdot V_{90^\circ}) / 2 \qquad V_{\text{upper}} = (V_{0^\circ} - i \cdot V_{90^\circ}) / 2$$

The signs may depend on the conventions of the LO system.

From Ed Fomalont:



Let's start a double side-band data streams in the signal to two antennas 0 and 1. The correlator multiplies the two streams and integrates them with various delay offsets i to obtain the two lag functions $C_0[01]$ and $C_0[10]$. These are related to the Fourier transform of the complex spectrum $R_1(i) + jR_2(i)$:

$$FT(C_0[01](i) + C_0[10](-i)) = R_1(i) + jR_2(i)$$

Now, you can put a 90 degree phase shift in the local oscillator used for the second telescope (data stream) and correlate this data to get the set of lag data $C_{+90}[01]$ and $C_{+90}[10]$

$$FT(C_{+90}[01](i) + C_{+90}[10](-i)) = R_3(i) + jR_4(i)$$

Now, you can put a -90 deg phase shift in the local oscillator used for the second telescope (data stream) and correlate this data to get the set of lag data $C_{-90}[01]$ and $C_{-90}[10]$,

$$FT(C_{-90}[01](i) + C_{-90}[10](-i)) = R_5(i) - jR_6(i)$$

$$R_5 = -R_3; R_6 = -R_4$$

The upper (U), lower (L) side-band response is then (see Thompson et. al. (6.26, 6.27)[17]:

$$\begin{aligned} \text{U:} & \quad 0.5[(R_1 - R_4) + j(R_2 + R_3)] \\ \text{L:} & \quad 0.5[(R_1 + R_4) + j(R_2 - R_3)] \end{aligned}$$

Fourier transform of both sides give

$$\begin{aligned} C_{upper}[01](i) &= 0.5[C_0[01](i) - C_{+90}[10](-i)] \\ C_{upper}[10](-i) &= 0.5[C_{+90}[01](i) + C_0[10](-i)] \\ C_{lower}[01](i) &= 0.5[C_0[01](i) + C_{+90}[10](-i)] \\ C_{lower}[10](i) &= 0.5[-C_{+90}[01](i) + C_0[10](-i)] \end{aligned}$$

The bottom line procedure is:

Correlate with phase difference = 0 and get lags between antenna 0 and 1: $C_0[01](i)$ and $C_0[10](i)$. Note that phase difference of 0 and 180 are identical and are produced automatically. Thus the lags for C_0 will be the average of the two: $C_0 = (C_0 + C_{180}) / 2$



Correlate with phase difference = +90 and get lags: $C_{+90}[01](i)$ and $C_{+90}[10](i)$

Correlate with phase difference = -90 and get lags: $C_{-90}[01](i)$ and $C_{-90}[10](i)$

Assuming that C_0 lags are done for two cycles, but the C_{+90} and C_{-90} lags are done in one cycle, then upper sideband lags then should be:

$$C_{upper}[01](i) = 0.5[C_0[01](i) - C_{+90}[10](i) + C_{-90}[10](i)] \quad (1)$$

$$C_{upper}[10](i) = 0.5[C_{+90}[01](i) - C_{-90}[01](i) + C_0[10](i)] \quad (2)$$

Lower sideband lags then should be:

$$C_{lower}[01](i) = 0.5[C_0[01](i) + C_{+90}[10](i) - C_{-90}[10](i)] \quad (3)$$

$$C_{lower}[10](i) = 0.5[-C_{+90}[01](i) + C_{-90}[01](i) + C_0[10](i)] \quad (4)$$

3.7.16 CDP Configuration

This package is responsible for receiving, tracking and managing the configuration of the CDP. Configurations arrive at the master node as an IDL structure which is derived from the HLA ALMA Project Data Model and is sent off to the appropriate compute nodes where they are queued and applied at the correct array time.

Each compute node handles $\frac{1}{4}$ of the baselines for a given correlator quadrant. The master node tracks which compute nodes have which configurations to facilitate the ‘assembly’ of spectral data packets from individual compute nodes into spectral data blocks for an array to be sent to the BDD. This assembly process is part of the *Master Data Publisher* package.

3.7.17 Array Configuration

Sub-scan configurations are specific to an array and a correlator quadrant. Much of the discussion of arrays in section 3.3.3 applies here. It is important to note here that the correlator antenna inputs must be mapped to the antenna labels in the output data in order to recover the original array information.

3.7.18 TE Scheduler

The *TE Scheduler* package receives scheduling requests and executes them at the correct time based on the array time being managed by the *TE Handler*. The scheduler provides an interface for requesting semaphores that are given (signaled) in a periodic or one-shot fashion at a programmable offset in milliseconds after every time event. In this way user tasks can synchronize independently to the same time base. When the TE Handler switches to SOFT mode the scheduler will actually keep giving the already requested semaphores, causing no disruption to the user tasks. It is the responsibility of the ACC, or some higher level entity, to detect the degraded state and stop gracefully before forcing the distributed clock to re-synchronize.

3.8 Detailed CDP Package Descriptions

This section provides detailed class diagrams and descriptions of the more complex CDP packages.

3.8.1 Lag Processing

At each dump, a set of raw lags are transferred via DMA to one of two memory buffers. Once the DMA transfer is complete, processing of the memory buffer begins.



Each raw lag set from the correlator hardware delivers data in a preset order depending on the baseline using CAIs, the polarization product, bin number, number of lags and other information. This information is carried through to the spectral processing phase to determine how spectra are summed and becomes part of the header information when the spectral data are published.

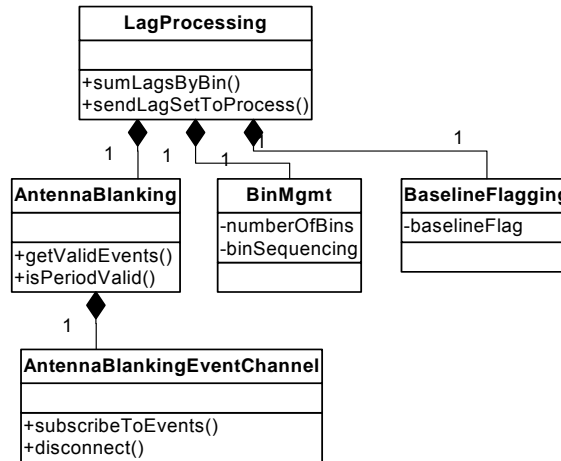


Figure 15 Lag Processing Package Details

Each antenna computer (ABM) publishes status information for each 48 ms period to an antenna blanking event channel to which the CDP subscribes. If a negative status or no status information is received for a given time period, the CDP discards the affected lag sets. Due to the potential latency in the notification channel, the *AntennaBlanking* class will wait up to one second in order to not miss any blanking events. This introduces a one second latency in the data throughput, but it does not extend the integration duration. Note that this wait interval is a programmable value which can be **decreased** from one second as we better understand the throughput and latency issues with the ALMA Ethernet networks. Also, this delay may affect the TelCal subsystem.

The blanking event structure is defined as follows:

```

/** \struct antenna blanking structure sent by each antenna for a period of
** 21 TEs. Each antenna will accumulate the blanking flags for every 21 TEs
** which is 1.008 seconds. Then it will set the teOffsetBitMask to a '1' or
** '0' signifying that TE interval (relative to the start time) is valid or
** not.
** Each bit represents one of the 21 TE intervals w/ the LSB being TE[0] to
** bit 20 being TE[20]:
**
** TE Interval: TE[20]|TE[19]| ... |TE[2]|TE[1]|TE[0]
** Bit Value:   1 | 0 | ... | 0 | 1 | 1 | some blanking here
** Bit position: 20 19      2  1  0
**/
struct antennaBlankingEvent_t
{
    ACS::Time timeStamp;    ///< Time stamp of the beginning of the event interval
    string antennaID;      ///< Antenna ID as a string, e.g., 'ALMA001'
    long teOffsetBitMask;   ///< The bitmask for each TE interval LSB is the 1-st TE
};
  
```



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 34 of 86

This structure extends for a duration of 21 TEs (1.008 seconds). Each bit in *teOffsetBitmask* represents one TE interval after the *timestamp* for a given antenna. A bit value of 1 means to blank whereas 0 means to not blank data.

Blanking introduces some subtleties regarding the integration times and durations which must be recorded for each integration:

- Requested integration duration – the observer sets this integration duration
- Actual integration duration – the requested duration minus any blanked dumps
- Integration start time – the time stamp that the integration is supposed to start
- Integration centroid time – the average time that non-blanked data was integrated. The timestamp for that data record needs to indicate the exact centroid of the data.

Also there is a provision to blank an entire array if, for example, a local oscillator malfunctions. In this case the following structure would be used:

```
struct arrayBlankingEvent_t
{
    ACS::Time timeStamp;    ///< Time stamp of the beginning of the event interval
    StringSeq antennalIDs;  ///< Antenna IDs is array
    long teOffsetBitMask;   ///< The bitmask for each TE interval LSB is the 1-st TE
};
```

This is similar to the previous event, but the individual antenna is replaced by a sequence of antenna labels on which to perform blanking.

The *BinMgmt* class is responsible for identifying raw lag sets by the correlator bins. There are two types of binning envisaged. *Antenna-based* switching where, e.g., the nutator switches positions and lags are placed into separate bins in the correlator hardware and consequently the CDP for each nutator position. Note that there should be a bin which collects the data while the nutator is moving. This bin's data can be discarded. *Baseline-based* switching is used where sideband separation is desired and different correlation products of 90° phases for a baseline are kept in separate bins. All bin switching schemes must be synchronized with the correlator bin switching timing boundaries of 16 ms.

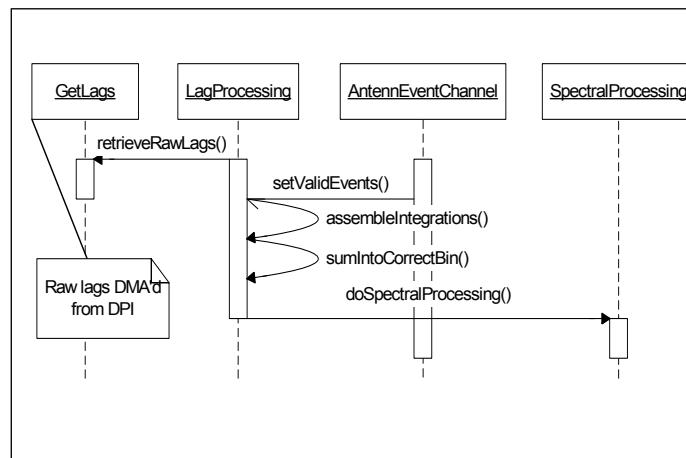



Figure 16 Lag Processing Sequence Diagram

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 35 of 86
--	--	--

BaselineFlagging allows for flagging a dump. If a certain part of the correlator hardware, e.g., a station card or correlator chip, fails a test, then the lags corresponding to the affected baselines can be flagged as ‘bad’ or ‘questionable’. The lags are still converted to spectra, but these flags are provided to the DataCapture system of the Offline system which can then discard the data if desired.

3.8.2 Correlator Flagging

Here we present the various baseline flags which are sent to the ALMA archive via the binary data at each integration or sub-integration or to Data Capture at the end of a subscan. The Antenna/Baseline column represents how the flags affect the data, i.e., on an antenna or baseline basis. Many of the flags are antenna-based, but affect all baselines associated with a given antenna.

Flag name	Action	Antenna/ Baseline	Description
WVR Receiver Values	Flag	Baseline	WVR data not received by an antenna's WVR receiver. This affects the APC corrected data for all baselines associated with the antenna
TFB Scaling Factor	Flag	Baseline	TFB scaling factors for a given antenna were not received. This affects all baselines associated with the antenna. It also affects the spectral window for the given scaling factor.
Blank Integration	Blank	Baseline	Control delivered blanking signal(s) for a specific antenna for an entire integration (or sub-integration) leading to no data for the integration interval.
Correlator Chip Error	Blank	Baseline	One or more antenna inputs in a correlator chip are bad. All baselines using the affected antennas are blanked.
Missed blanking event	Blank	Baseline	Blanking signals which are expected for a given time interval and antennas are not received in the blanking channel. Data are blanked for all baselines of the affected antennas in the missed interval .
Missed delay event	Blank	Baseline	Antenna delay data which is expected for a given time interval is not received in the delay channel. Data are blanked for the missed interval.
Antenna Flag	Flag	Antenna	Control delivers antenna flagging info which is passed onto DataCapture
Polarization Flag	Flag	Antenna	Control delivers polarization flagging info which is passed onto DataCapture
Baseband Flag	Flag	Antenna	Control delivers baseband flagging info which is passed onto DataCapture
Bulk Delay Flag	Flag	Baseline	Correlator antenna-based station card increments bulk delay (64 samples) in a sub-scan causing a phase error in an integration. All baselines using the affected antenna are flagged.



Table 2 Correlator Flags

3.8.3 Spectral Processing

Once a lag set is processed, it is ready to be converted to a spectral data block. With the advent of the TFB, the specific steps for processing lags from the NRAO FIR filter are a little different then when processing lags from the TFB and are noted in the following subsections.

Figure 17 shows the classes which are involved with the steps of this processing. The order of the steps is: lag normalization, quantization correction, windowing, FFT, atmospheric phase correction, fine (or residual) delay adjustment, spectral averaging and channel averaging. Figure 18 is a sequence diagram for the spectral processing which shows the order of processing steps.

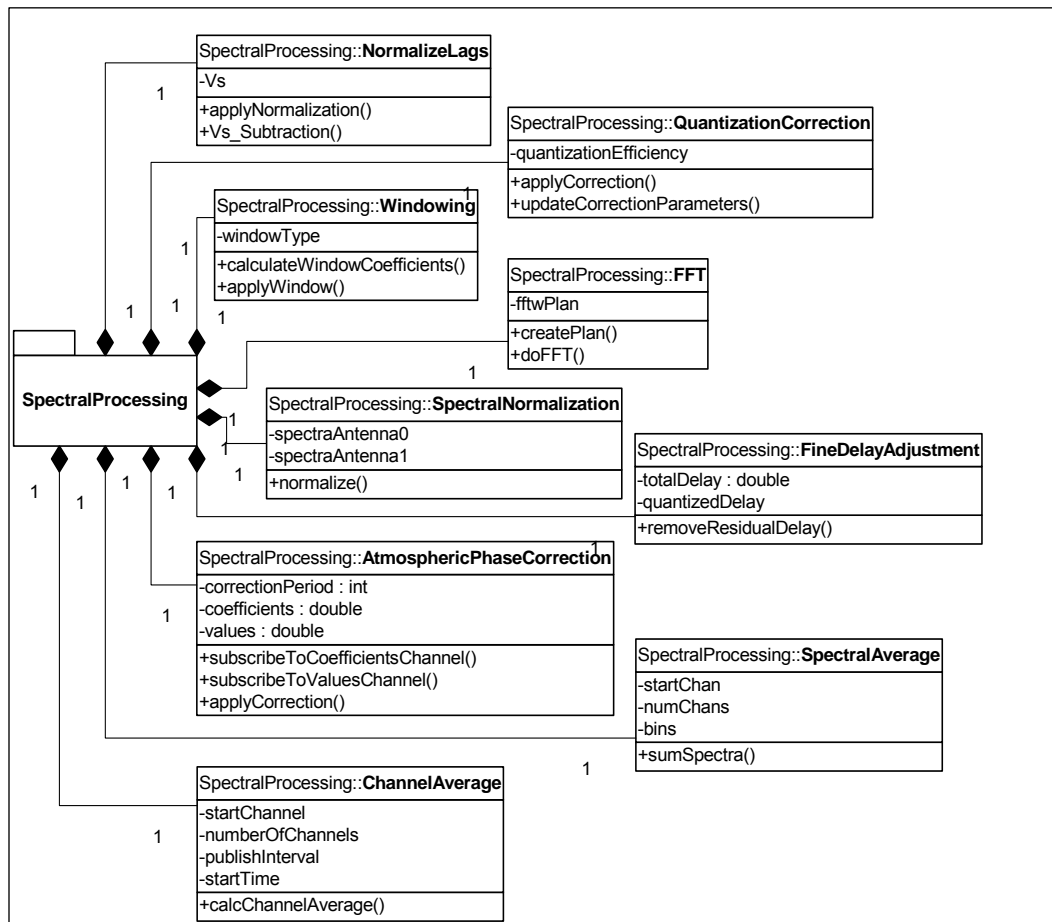


Figure 17 Spectral Processing Package Details

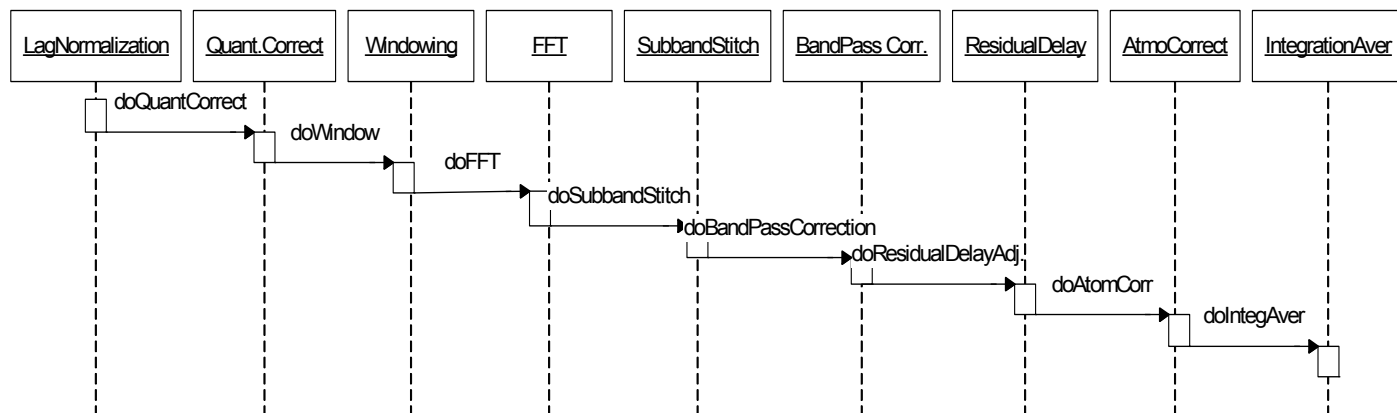


Figure 18 Spectral Processing Sequence Diagram

3.8.3.1 Lag Normalization

Lag normalization is a multi-step process:

The cross spectra are normalized to unity by dividing each lag value by the square root of the product of the zero lag values for the corresponding antennas, while the auto spectra are normalized by just dividing by the zero lag value. This is discussed in "Spectral Normalization" of [16], and would produce lags where 100% correlation has a value of unity, except for the presence of multiplication table bias which is dealt with in the second step.

In this second step, a multiplication bias introduced at the correlator chip level in its multiplication table when calculating the correlation function is removed. This bias removal is sometimes called ' V_s subtraction'. See section 9.2.1 for this equation.

A final step is the gain correction using the digitizer statistics. The digitizer statistics values are obtained via the CCC and delivered to the appropriate CDP compute node. The basic process is discussed in section 8.5 of [17].

For the TFB, there is no normalization as this is done by the quantization correction.

3.8.3.2 Quantization Correction

This step performs quantization correction on the lags. The algorithm for quantization correction is to first perform a 2-bit, four-level correction followed by a 3-bit, eight-level correction. For the TFB, currently a 2-bit, four-level correction will be done as the eight-level correction is small relative to the four-level correction. See F. Schwab's GBT memo [18] which discusses his correction methods.

The TFB has a 3 x 3 bit mode with 9 levels and 4 x 4 bit mode which will require a 16-level correction. We are working with F. Schwab on these corrections.

3.8.3.3 Windowing

Apply a smoothing function to the lags. Normally a Hann (or Hanning) window is used, but the following windows are also available: Uniform (no window), Hamming, Welch, Bartlett, Blackman and Blackman-Harris. See section 9.2.6 for the equations for each of these windowing functions.

3.8.3.4 FFT

The correlation functions are Fourier transformed to give spectra. The FFTW algorithm from MIT [19] is used due to its optimizations and fast calculations. It has an important efficiency which creates a 'plan' for a given size



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 38 of 86

of FFT. This plan is relatively slow to calculate which is done once and when complete, the actual FFT calculations are very fast. In the future, we will be investigating the AMD Core Math Library (ACML) for performance which has an optimized FFT [20].

For the TFB, a $\frac{1}{2}$ channel shift is performed on the lag channels before the FFT. This moves the center frequency of each channel from the leading edge to its center. This $\frac{1}{2}$ channel phase shift can be optional, i.e., if the number of overlapping channels is even when stitching together the sub-bands, then the shift is performed otherwise the shift is not done for an odd number of channels.

3.8.3.5 TFB Subband Stitching

For the TFB, adjacent subbands must be stitched together (also called ‘re-gridding’). A concept is introduced here, the ‘spectral window’ which defines a set of TFB subbands which share a common resolution and form a spectral window within the total 2 GHz bandpass. There can be up to 32 spectral windows which are defined by a bandwidth, resolution, center frequency and number of polarization products. The 62.5 MHz TFB subbands which comprise the given TFB configuration are then overlapped with some channels discarded and stitched together to form a contiguous bandpass. See [15] for details.

3.8.3.6 TFB Bandpass Calibration

For the TFB, a bandpass calibration is performed which corrects the bandpass for the filter characteristics. These are constant values for a given TFB mode and are calculated when a new configuration is received. See [15] for details.

3.8.3.7 TFB Scaling Calibration

For the TFB, each subband must be scaled by the total power measurement. This total power value is determined from registers on the TFB by the CCC and provided to the CDP node processing software. This scaling removes ‘platforming’ differences between the subbands in the presence of strong spectral lines.. These scaling measurements will be done via a specific calibration command in the command control language (CCL). The scheduling block script, based upon the observation type, will insert this command as needed.

3.8.3.8 Residual Delay Adjustment

As discussed in section 3.2.6, the majority of the geometric delay adjustment is performed in hardware at the digitizers and correlator. Nevertheless, small errors in the geometric delay resulting from the discrete nature of the digital delay are accumulated over the course of the integration. These residual delay errors are removed in the CDP.

The spectrum is corrected for the average delay error over the integration by removing the residual delay which is defined as the difference between the unquantized delay and the quantized delay integrated over the fine delay interval, i.e., the integration duration.

The quantized delay includes the quantized delay provided by the model server which has a resolution of 31.25 picoseconds (1/8 of 1 sample period of 0.25 ns) plus the delay offset for a given antenna due to its cable distance from the array center point (the geometric center) to the correlator.

The residual delay is simply:

$$D(n)_{res} = D(n) - D(n)_{quant}$$

(1)



where $D(n)$ is the total, unquantized delay provided by the geometric delay model server and $D(n)_{quant}$ is the total quantized delay over the integration interval. The delay model provides delay information for each antenna based upon its distance from the array's geometric center.

As delays are baseline-based, the actual residual delays are the difference of two antenna values (antenna n and antenna m) for a baseline, leading to:

$$D_{baseline} = D(n)_{res} - D(m)_{res} \quad (2)$$

$D_{baseline}$ can be used to apply the residual delay correction.

Delay events are transmitted by the delay model server that covers a certain period of time, nominally 10 seconds. Delay events are received by the CCC and routed to the CDP. The events are contiguous in that there is no temporal gap between one delay event and the next. After the end of a delay event's stop time, if the CDP does not receive a subsequent delay event whose start time immediately follows the previous event's stop time, then the CDP blanks the lag data. The CDP continues to blank data until a valid delay event is received.

This procedure ensures that no bad data due to are delivered the archive. Also delay events are solely received from the CCC and applied to the hardware if the delay event timestamps are valid. If they are invalid, e.g., the events arrive too late, the CCC does **not** apply the quantized delay to the hardware and does **not** deliver the delay event thus generating a temporal gap for the CDP which prompts it to blank correlator data.

Figure 19 shows a stylized representation of an array of antennas and relationship of each antenna to the 'geometric center'. The actual geometric center may actually be one of the antennas. The line between the geometric center and the antennas represents the cable offset, a fixed length for each antenna.

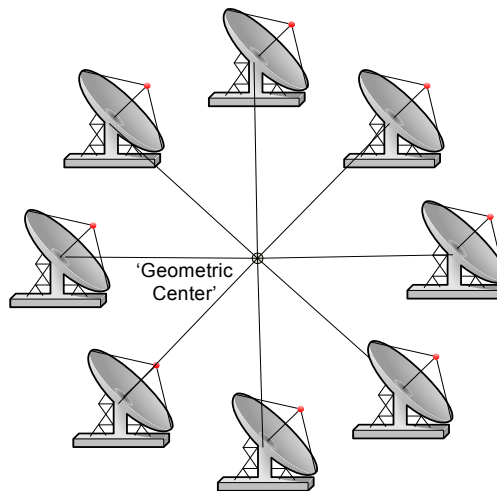


Figure 19 Geometric Center

There is also a baseband and antenna dependency to the fixed delay. As there is only one digitizer for all basebands, there are differences for each antenna through each baseband. The TMCDB will contain a table which provides the fixed delay for each antenna and baseband combination as such:



ALMA Project

Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
 Date: 2009-08-12 Status: Approved
 Page: 40 of 86

	A0	A1	A2	A3	A4	A5	A6	...
BB_1	0	0	0	0	0	0	0	
BB_2	10	50	30	240	20	30	18	
BB_3	14	45	25	300	30	40	20	
BB_3	10	35	20	310	20	35	25	

For values less than 250 ps, this correction is handled solely by the CDP nodes in its residual delay correction. If some values are more than 250 ps, then the CCC will have to add this fixed delay to that antenna/baseband with the remainder being applied to the residual delay. For example, BB_3 & A3 have a fixed delay of 300ps. Thus the CCC would add one extra sample (250ps) to its delay and the CDP would add 50ps (300-250) to its correction.

3.8.3.9 Atmospheric Phase Correction

The path length correction due to the atmosphere as a function of the baseband frequency is done in the CDP nodes. For each antenna and each frequency band, correction coefficients are obtained from telescope calibration subsystem via the control subsystem. WVR channel values are published from each antenna's WVR receiver. The CDP utilizes the coefficients and data stream and applies the atmospheric phase correction to the spectra.

The path length correction coefficients are published before a sub-scan starts, when a sub-scan starts and at periodic intervals during which they are valid. The WVR channel values are published about every 1/2 second which is the frequency that the corrections are applied.

The correction is of the form (note that these are preliminary forms which will be revised in future versions of this document):

$$\Phi_{ni} = \varphi_i \{ C_{n0} V_{n0} + C_{n1} V_{n1} + C_{n2} V_{n2} + C_{n3} V_{n3} + C_{n4} V_{n4} \} \quad (3)$$

$$\Phi_{mi} = \varphi_i \{ C_{m0} V_{m0} + C_{m1} V_{m1} + C_{m2} V_{m2} + C_{m3} V_{m3} + C_{m4} V_{m4} \} \quad (4)$$

$$\Phi_i = \Phi_{ni} \Phi_{mi}^* \quad (5)$$

Where:

Φ_{ai} is the atmosphere – corrected phase for a specified antenna in a baseline n-m for the i^{th} channel

φ_i is the phase at the i-th channel for baseline n-m.


C_{0-4} are the atmospheric path length correction coefficients (converted to phase) for a specified antenna (n or m) supplied by the telescope calibration system V_{0-4} are the WVR values from a specified antenna's (n or m) WVR receiver

Equations (3) and (4) provide phase corrections for antenna n and m . Equation (5) shows the correction for antenna n multiplied by the complex conjugate of the correction for antenna m producing the final phase correction for each channel in the spectral window.

It is important to note that APC and non-APC data are integrated separately. At the end of each subscan, the CDP aligns the WVR values to integration and sub-integration time boundaries and sends them to the data capturer for the array which becomes part of the ASDM for the subscan. TelCal uses this information to produce a new set of coefficients for the next subscan.

3.8.3.10 Integration Averaging

This step simply adds spectra for each dump into an integration accumulator with APC and non-APC data summed separately. The integration duration is an integer multiple of correlator dumps and once the integration duration is complete, the resulting spectral data are passed on to the *DataWriter* component (see section 3.8.4).

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 41 □ of 86 □
--	--	--

Integration spectral averaging allows one to decrease the final data output rate of the correlator to meet the 60 MB/sec. specification.

3.8.3.11 Spectral Channel Averaging

Adjacent spectral channels can be averaged together to reduce the number of spectral channels. The number of spectral channels to average is specified in the correlator configuration and must be a power of 2 such that it evenly divides the total channels. Spectral channel averaging is done as a final step before delivering the spectral data to the archive.

3.8.3.12 Channel Average

This class performs periodic channel averaging on the spectral data sets. A complete discussion of channel averaging is provided in section “Channel Average” of [16], but is paraphrased here:

The channel average for the cross correlation spectra is the vector average of the complex visibilities across the spectrum. For a continuum source the channel average has the same value as the channels, but with reduced noise. To remove any phase slope across the band, residual delays are first removed before the channel average is calculated.

The channel averaging configuration defines a rate at which the channel averaging is performed and a set of spectral channels over which to calculate the average (up to 32 separate ranges can be defined within a 2 GHz base-band). The spectral channels are chosen to avoid the edges of the bandpass and sometimes selected to match a spectral line (maser). There is one channel average value per defined channel average region. The integration duration ranges between 0.5 – 1.0 seconds such that there are an integral number of channel average integrations within a spectral integration duration. As the APC correction interval is generally larger than the channel average interval, only the APC-corrected results are used.

3.8.4 Data Publishing

Finally, the spectral and channel average data are published separately in two stages. First the compute nodes send spectral and channel average data to the master node and then the master node publishes data to the Archive. The compute nodes’ job is straight-forward, they just publish the data blocks with identifying headers to the master node using the Bulk Data module of ACS [21].

The *DataWriter*, in the master node, assembles the baselines of an integration for a specific array. The XML header and binary data are joined together in the appropriate data structure. The spectral data blocks and channel average data are then published using the *DataPublisher* class.

A multicasting mechanism is proposed where the CDP publishes to the BDD which in turn multicasts the data to three subscribers: (1) the Archive, (2) the telescope calibration subsystem and (3) the quick look pipeline via a real-time ASDM filler. The Bulk Data module of ACS [21] which is based on the CORBA Audio/Video streaming service.

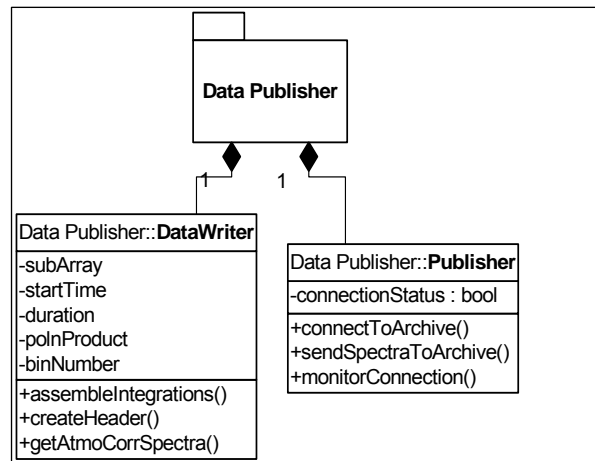


Figure 20 Data Publisher Package Details

The autocorrelation spectra are always taken and archived. These passbands may be used in the calibration process and are certainly important for debugging the front ends, filters, digitizers and data transmission system. They are a free spectrum analyzer.

As discussed earlier, the output consists of XML headers with binary data. The binary portion of the data is stored in a fashion that avoids rotation of the data by pipeline processing computers to preserve compatibility with the ALMA Science Data Model (ASDM) [14] for an efficient quick-look operation. The byte order of the binary data will nominally be Little Endian (Intel), but the header will explicitly specify the byte ordering. The following streams are produced by the Correlator subsystem for each array:


- Spectral Data: XML meta-data + actual integration durations and times + binary data go to the BDD
- Channel Average Data: XML meta-data + actual sub-integration durations and times + binary data go to the BDD
- ASDM Main table containing spectral and channel average information go to the DataCapture component.

The purpose of DataCapture is to bridge the information regarding the telescope operation to the science output for each subscan. The Control subsystem creates a DataCapture component for each array notifying all interested subsystems of the new component via a notification channel. The CDP Master subscribes to the channel and, when a new DataCapture component is created, obtains a reference to the CORBA object to send sub-scan information to it.

Most of the information needed by the DataCapture is sent by the Control subsystem at the beginning of a sub-scan. At the end of the subscan, Correlator provides status information about the subscan via a CORBA function call which include:

- Whether or not there were any flags for integrations or sub-integrations
- The total number of un-blanked integrations executed
- The number of sub-integrations per integration.

The format of this main table data is one row for all integrations and one row for all sub-integrations in the sub-scan.

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 43 □ of 86 □
--	--	--

3.8.4.1 Scaling Factors

The final spectral results must be scaled for efficient storage in the archive as specified in SSR 2.3-R5. A scaling factor is computed based on bandwidth of a spectral window and integration duration which is used to determine if the spectral results can fit into a 16- or 32-bit signed integer. This scale factor accompanies the data results and is specific to each integration, although it may be the same for many integrations. Only cross correlation data will be scaled integers while auto-correlations will always be floating point values. A detailed procedure for determining scale factors can be found in ‘Data Scaling’ of [16]. Note that channel average data is not scaled as it is a single floating-point value.

3.9 Data Flow Robustness

Data flow through the CDP processing software must be resistant to failures. If hardware or software errors prevent spectral results to arrive at the archive, the software must notify the operator via alarms and continue to run in a degraded state.

Data flow through the CDP processing software must be resistant to failures. If hardware or software errors prevent spectral results to arrive at the archive, the software must notify the operator via alarms and continue to run in a degraded state or to fail gracefully.

3.9.1 Detail Data Flow

The correlator data flows as a synchronous pipeline with timed waits at many steps of this pipeline as shown in the following diagram.



Figure 21

Each box represents a data producer and/or consumer with the correlator software components in the gray boxes. The arrows which connect boxes show direction of the data and those which run below the boxes are timed waits, where a given thread waits on a timed semaphore for data to arrive. The last arrow between the CDP Master and the BDD is a simple data publish.

The Lag Processor (LP) waits on correlator dumps. LP delivers its data to the Spectral Processor (SP) which converts the lags to raw spectra (or channel averages). The CDP Node publishes these raw spectra to the CDP Master which in turn assembles all the integrations into an ASDM binary blob which is then published to the BDD.

The first three components are part of the CDP Node and that there are multiple CDP Nodes all sending their data to a single CDP Master.

During R5.1, we have encountered problems with the BulkData transmission between multiple senders in different computers and multiple receivers in the same computer. These are different streams but have caused implementation problems. I wanted to drop the BulkData system and replace it with a simple TCP/IP socket implementation for this internal communication, but CIPT management required that we continue to use the BulkData mechanism.

If any delay occurs in this pipeline an alarm is raised denoting data failure.



3.9.1.1 Data Flow Failure Handling

The important receiver is the CDP Master and here is where the data flow failures are handled. The CDP Master receives CDP Node data blocks for each baseline, polarization product, bin, APC, etc. and knows when to expect each data block and how many blocks to receive for each integration. If data delivery failure occurs anywhere in the pipeline, a timeout will be sensed by the CDP Master. In general, the CDP Master expects to receive all of the data blocks shortly after each integration boundary beginning with the subscan start time. A data block is considered 'missing' if it doesn't arrive in this time frame.

Once a data block is missed, an alarm is triggered with appropriate information to identify which block is missed. The following exception algorithm is executed:

```
numberOfTries = 0
do
    wait for data block within an integration interval
    if missing data block arrives
        add missing data block to received_data_block_list
        break
while( numberOfTries < MAX_RETRY_COUNT)
if missing_data_block no received in MAX_RETRY_COUNT
    set flag error indicating cause of failure
    add dummy data block to received_data_block_list
publish_data_to_BDD()
```

A dummy data block for now will hold all 0's, but there is no special significance of the values. The dummy data block is added to keep the binary attachment size at the expected size for the integration.

The concept of flagging exists in the ASDM binary code in general, but specific flags have not been defined but will be worked on in R6.

3.9.2 Alarms

In general, alarms are separated into 4 categories:

- Data corrupted
- Data might be corrupted
- Data may be suspect
- System offline

In order to avoid an 'alarm overload' situation where there could be a cascade of alarms repeatedly occurring throughout a subscan, 4 alarms with optional information describing the source of the alarm have been defined. Using these alarms with textual source information, one could look into the logs to find the source of the problem. The 4 alarms are:

- CDP Master fails to receive data from a CDP node
Here the CDP Master will have information regarding the expected data including channel average or spectral data, the antennas, array, baseband, etc. This will guide the debugger to find the correct CDP Node and source of the alarm.
- CDP Master fails to send data to the Bulk Data Distributor



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 45 of 86

Here the CDP Master can add additional information regarding the type of data, channel average or spectral, the BDD flow number, array id, etc. and when in the subscan, i.e., at the beginning of the subscan, for an integration or sub-integration, or at the end of the subscan.

- CDP Master fails to send end-of-subscan information to the DataCapturer

Although this only happens in one spot, it is very important in the data flow as it finalizes the ASDM for a subscan. The DC name and array will be the extra information. Also, bind/release DC function calls can generate this alarm.

- CDP Node or Master computer hardware problem

An example of this failure would be a temperature over-limit value. This would be tied to a BACI property.

3.10 Physical Architecture

Figure 22 shows a block diagram of the correlator computer hardware and interconnections for the full 64-antenna correlator.

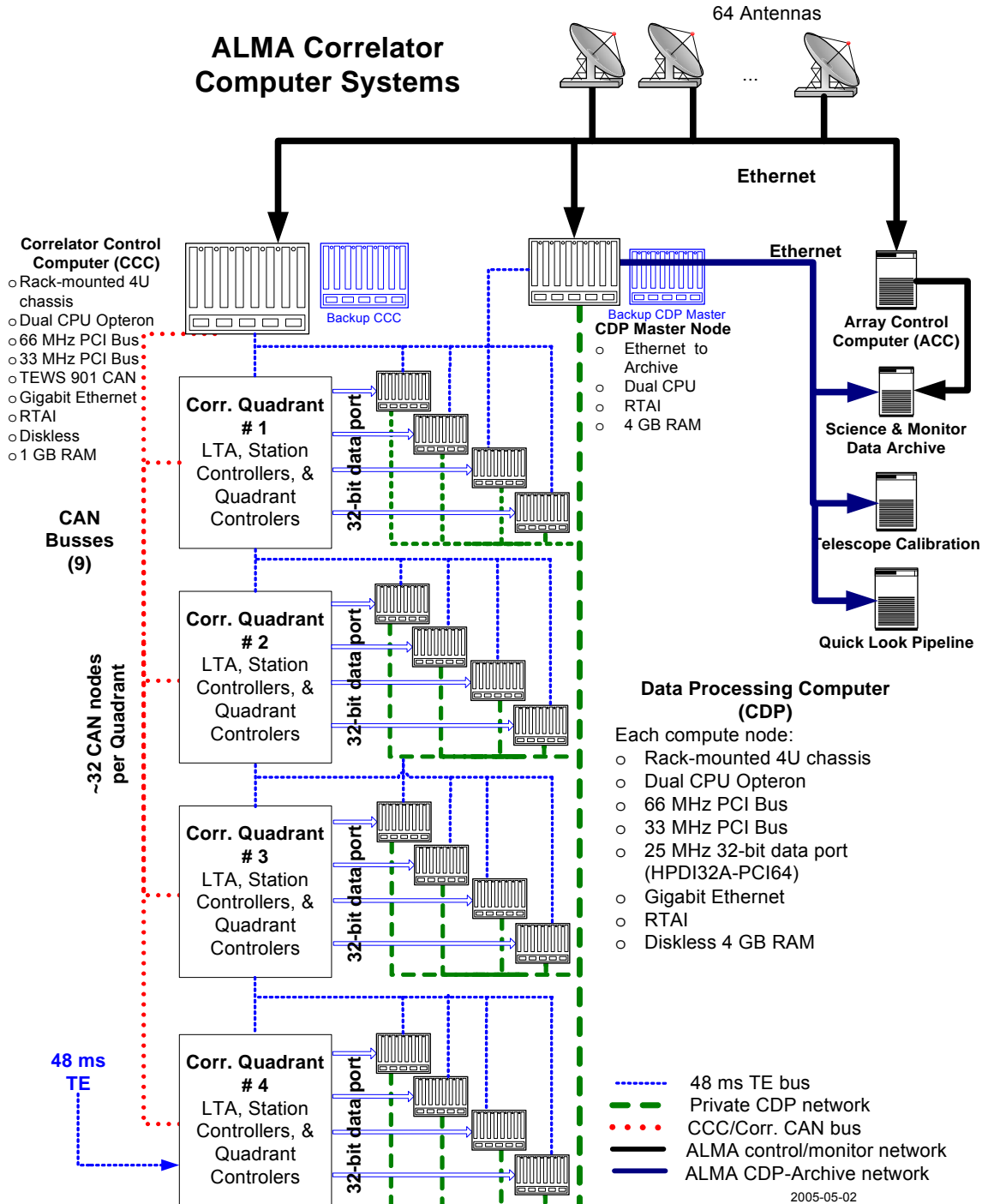



Figure 22 Correlator Computer Layout

The correlator control computer is a dual Opteron CPU rack mounted PC running RedHat Linux and RTAI. Current versions run the CPU at ~2 GHz, but in general we will purchase the fastest CPUs available. Due to the altitude, all computers will be diskless. For file systems which hold kernel logging and temporary run-time data, e.g.,

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 47 of 86
--	--	--

/tmp and /var, either a RAM disk or a tuned NFS and kernel configuration can be used. This will assist in debugging serious software failures. We will be evaluating this behavior in the R6 interval.

3.10.1 Correlator Control Computer

The CCC connects to the correlator hardware via 9 CAN busses using two Tews TPMC901 CAN (www.tews.com). The CCC accesses the array-wide timing events via its parallel port. The CCC also has an RS-232 serial interface to the correlator for low-level bootstrapping functionality. The CCC interfaces to the external computer systems via Ethernet.

3.10.2 Correlator Data Processor

The correlator data processor computer (CDP) is a cluster of dual Opteron CPU rack mounted PCs. The computers connected to the correlator hardware via 32-bit data ports are ‘compute nodes’ and connect to a ‘master node’ which acts as a bridge between the internal cluster network and the external telescope network.

The correlator hardware distributes the array-wide TEs to the CDP compute and master nodes via their parallel ports.

Each compute node essentially performs the same function of extracting raw lags from the correlator hardware and converting them to spectral data blocks. There is no sharing of lags among compute nodes and all the lags for a given baseline end up in one compute node. An exception to this is when the quantization correction is made – the channel 0 lags of the auto-correlation are necessary to compute the correction. This is a single value which should not impose a huge burden on inter-node communication.

3.10.3 Network Infrastructure

There are 3 networks associated with the correlator computers:

- CAN – a deterministic serial protocol used to control and monitor the correlator hardware
- Gigabit Ethernet – used within the CDP cluster for communication between the master and slave nodes.
- Gigabit Ethernet – used to interface the CCC and CDP to external computer systems on the ALMA AOS network which includes the ARTM and ABM computers and the ALMA OSF network which includes the ACC, Archive, Telescope Calibration, and Quick-Look pipeline.

The present plan has a 10Gbit fiber link from the AOS to the OSF to support correlator spectral data and a separate 10Gbit fiber link for monitor data and remote disk access.


3.10.4 Correlator Hardware

While complete descriptions of the correlator hardware can be found at [22], [23], and [24], a brief overview is provided here.

3.10.4.1 Basebands

Each quadrant is connected to a “baseband pair”. Each fiber of the pair carries digitized samples at 4 gigasamples/sec of 3-bits for a given intermediate frequency in ‘0’ and ‘1’ linear polarizations with a 2 GHz bandwidth. Depending on the correlator configuration, 1, 2 or 4 of the polarization products are used in calculating the cross-correlation function. Each antenna and baseband can be individually configured for all parameters, e.g., bandwidth, polarization products, dump durations, etc., although this is not normally done.

The Tunable Filter Board provides 8192 lag channels distributed among sub-filters with varying resolutions. The sub-filters can be tuned to any frequency within the 2 GHz band and multiple filters of different resolutions can be placed anywhere in the 2 GHz band. The TFB has 2-bit and 4-bit capabilities for added sensitivity. Each correlator chip handles 4 x 4 matrix of antenna inputs (CAIs) with a “correlator card” holding 64 chips (for 32 x 32

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 48 □ of 86 □
--	--	--

CAIs) and a “correlator plane” holding 4 correlator cards (for 64 x 64 CAIs). There are 32 planes to a quadrant and the entire correlator consists of 4 quadrants.

Auto- and cross-correlation products are accumulated in the LTA in units of 16 milliseconds (or 1 millisecond for special auto-correlation-only modes) which then dumps the raw lags via the DPI to the CDP for processing. The correlator hardware is synchronized with other array devices via the timing event bus. For the TFB the 16 millisecond dump duration could extend to a maximum of 512 milliseconds as the number of antennas increases in an array – for details on this point, see section 9.1.

3.10.4.2 Correlator Chip Accumulation

Correlator chip accumulation duration is measured in 1 millisecond for auto-correlations or 16 millisecond units for cross-correlations. This is specified as the correlator accumulation mode (CAM). Although 1 millisecond accumulations can be programmed (for autocorrelations only), they are dumped at most every 16 milliseconds, with 16 1- millisecond accumulations.

For the TFB, the minimum dump duration is 512 milliseconds for the 8192 channels for an array containing more than 5 antennas. The TFB has a ‘by-pass’ mode which provides a low resolution of 256 channels and short dump periods of 16 milliseconds.

3.10.4.3 Bin switching

The LTA has up to 4 separate ‘bins’ or memory accumulation registers into which separate correlator chip accumulations can be added. The sequence of switching among bins can be programmed on 16ms boundaries. Bin switching can be utilized for phase or other types of ‘antenna’ switching or it can be used for sideband separation. This is in addition to separate binning provided by the CDP.

3.10.5 Physical Computer Racks

Due to the propensity of earthquakes at the AOS site, an idea of the physical computer rack layout is needed for structural engineering analysis.

There will be at least four racks at least 48 RUs in height.



ALMA Project

Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN

Date: 2009-08-12 Status: Approved

Page: 49 of 86

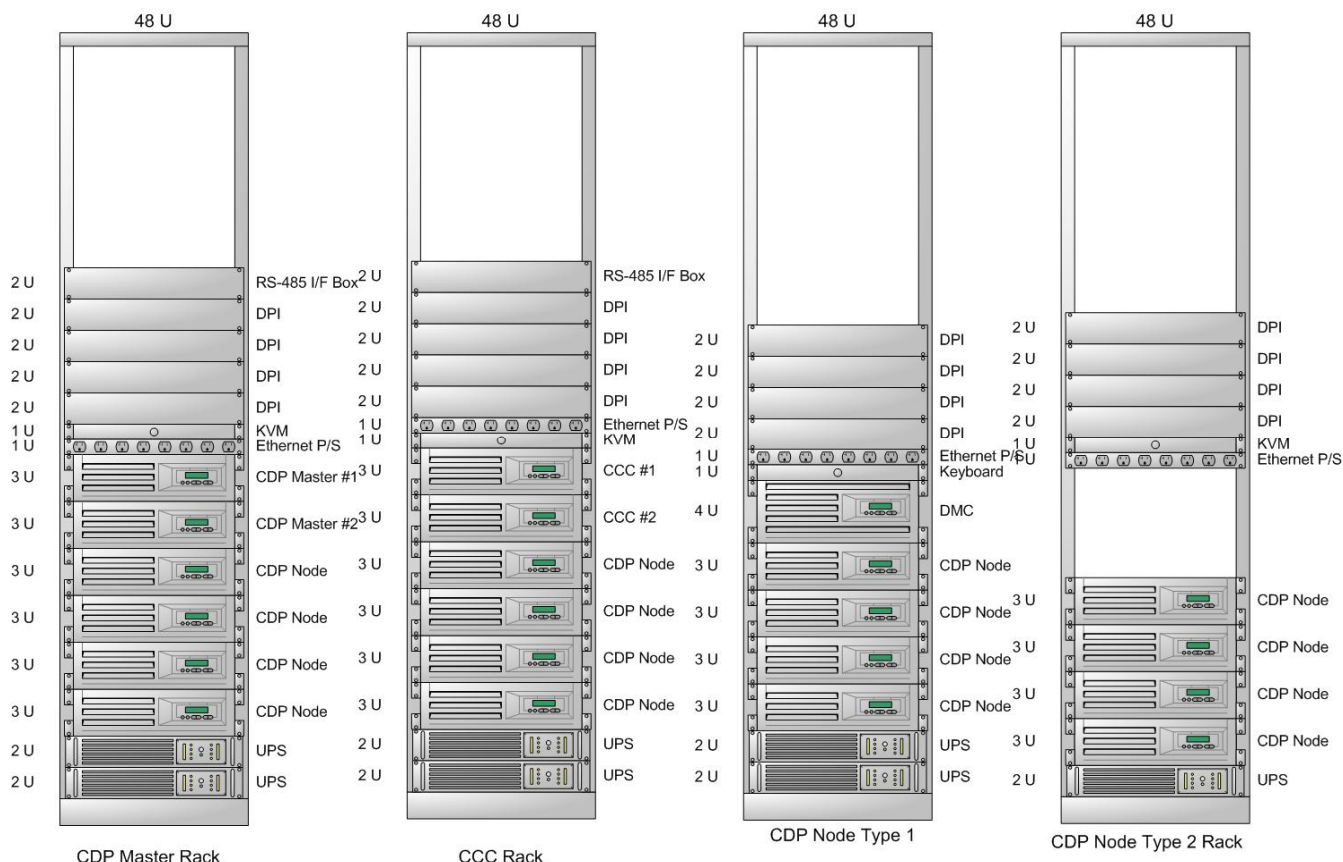


Figure 23 Correlator Computer Rack Layout

The following spreadsheet provides estimates of height and weight information.

Rack #	CDP Master	CDP Node	DPIs	RS-485 Interface	CCC	DMC	Monitor	UPS	Power Switch	Height Us	Height Inches	Rack Height	Contents Weight (kg)
1	2	4	4	1			1	2	1	38	66.5	40	405
2		4	4	1	2		1	2	1	38	66.5	40	405
3		4	4	1		1	1	2	1	36	63	40	379
4		4	4	1			1	1	1	29	50.75	40	296
Totals	2	16	16	4	2	1	4	7	4				1485
Height in Us	3	3	2	2	3	4	2	3	2			40	
Weight kg	26	26	4.5	4.5	26	26	4.5	57	23			85	

Table 3 Correlator Computer Rack Heights and Weights

These racks will be attached to the ends of the correlator racks and sit on a reinforced framework to comply with zone 4 earthquake standards. Also, the racks will be 48 U high to accommodate any extra computers.

3.11 Computer Cooling

Cooling in the correlator room at the AOS technical building uses forced air through a raised floor with an expected ambient temperature of 15°- 20°C. Air duct openings will be located in front of each computer rack which does not have front or rear doors. Each computer chassis draws in cool air via 2 9cm fans with 2 6cm exhaust fans drawing warm air out of the rear of the chassis. At this time, CPU fans are attached to large heat sinks to provide



better cooling of the CPUs. The possibility of using ‘low-power’ CPUs which do not require a CPU fan, but only a heat sink is currently under investigation.

3.12 Dynamic Model

The operational flow of commands and data to and from CCC and CDP has been previously discussed. Here we attempt to evaluate the real-time constraints on these systems.

3.12.1 CCC Timing Discussion

The main timing constraint, i.e., hard deadline, for the CCC is the 48 ms TE. Most CCC time critical processes are synchronized to the TE:

- Start observing command. This command instructs the correlator hardware to begin correlating with a given configuration on a specific TE.
- Distributing the geometric delay to the station cards at a specific TE. Assuming all 64 antennas require delay updates with one update every ~ 3.5 TEs (one update every ~ 172 ms) and assuming the CAN bus can transmit 2.4KB/TE with 256 bytes of delay data for all 64 antennas, then $\sim 3\%$ of the CAN bandwidth is required.
- Besides distributing the geometric delays to the correlator hardware, the CCC must not miss geometric delay events published on the delay notification channel. This can be avoided with the model server transmitting delay values in blocks in advance. Although it is the intention to provide shorter setup times, lead times of 5 to 60 seconds would ensure that network latency problems would be avoided. Prototype testing will allow better definitions of these lead values.
- The *TE Handler* package cannot miss a TE. A semaphore with a timeout of 50 milliseconds is used. If the TE is missed the timeout call-back function is called and recovery routines are executed to re-establish array time.
- Lead time of 1.5 seconds for configuration of the correlator hardware. This restriction is linked to a fast switching interval requirement of 1.5 seconds (see sections 3.1.6.3.1.1 – 3.1.6.3.1.1 of [25]). This is shown schematically in Figure 24.

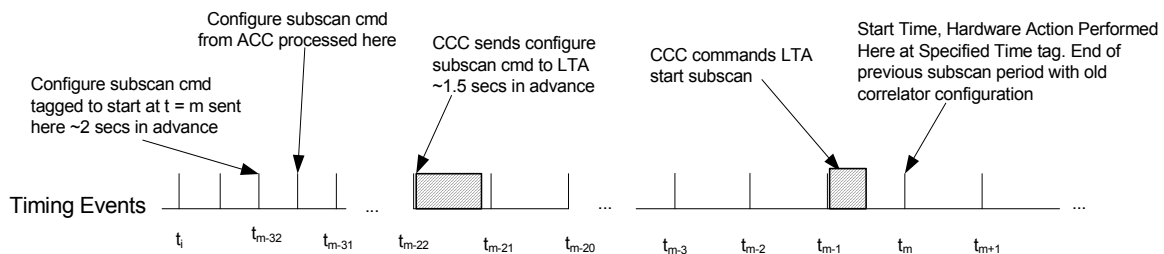


Figure 24 Sub-scan Configuration Timing Diagram

- Configuration for the CDP must be transmitted from the CCC with enough lead time to configure the CDP by a given TE. This is discussed in further detail in section 3.12.1.

In light of this timing information, Table 4 describes the beginnings of a rate monotonic analysis (RMA) done for the CCC. A simple procedure of RMA as described in [26] is followed where Utilization = (execution time / period) * 100%, i.e., the percentage of CPU time taken by a task. As long as the cumulative utilization does not exceed 100%, all tasks can be scheduled. Note that RTOS overhead is not accounted for which is usually a few percent of total CPU time. Each task is listed in decreasing priority order. Benchmarking was done a 1 GHz Pentium III CPU for all timing tests.



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 51 of 86

Task	Purpose	Execution Time (C _i) ms	Period (T _i) ms	Utilization %	Cumulative Utilization
1. Array Time	Process 48ms TE	0.5	48	1.0	1.0
2. Start Sub-Scan	Send start subscan cmd. to LTA & CDP	10	1500	0.7	1.7
3. Send delay	Distribute delay to SCC	10	48	20.8	22.5
4. Receive delays	Receive delay values from event channel	500	10000	5.0	27.5
5. Configure Sub-scan	Receive & schedule sub-scan configuration	100	1500	6.7	34.2
6. Monitor	Monitor corr. hw. & CCC	10	10000	0.1	34.3

Table 4 CCC Rate Monotonic Analysis

All execution times are currently estimates. As development progress continues, actual execution times based on prototypes can be developed to better define this analysis.

3.12.2 CDP Compute Node Timing

For the compute nodes in the CDP, the important deadlines are the receiving of raw lags from the 32-bit data port and processing the previous set of raw lags before the subsequent set arrives. Specific time-critical events include:

- Transfer raw lags from 32-bit data port to a memory buffer via DMA. Every 16ms the LTA can transmit 1024 sets of 256 4-byte lags (1 MB). $1 \text{ MB}/16\text{ms} = 64\text{MB/sec}$ per CDP Node.
- Process 1024 sets of 256 raw lags into spectra each 32 ms. This processing is twice the dump duration as we are using a double buffering scheme discussed in section 3.12.2. Note that this data set size is an example. There are many size lag sets which affect processing times linearly. This processing includes these steps:
 - Non-DMA transfers and copies of lag sets involved with lag processing
 - Antenna blanking including overhead of real-time event channel
 - Sideband separation binning and calculations
 - Lag normalization, windowing, quantization correction and FFT. Tests show that on a 1.0 GHz Athlon this takes approximately ~250 milliseconds for 1024 256-point data sets (256 complex spectral channels). Assuming 2.4 GHz and a linear increase in processing, this value becomes ~120 ms.
 - Fine geometric delay adjustment
- Channel Averaging on 500 ms period
- Transmit spectra to master node for an integration on 500 ms period. Note that this depends strongly on the integration duration which can often extend to 10 seconds
- Array time cannot miss a tick. A short, period task is triggered each 48ms with a watchdog timer to expire if the tick is missed. The watchdog timer can execute recovery routines to re-establish array time. If a timing tick is missed, the CDP must log an error and blank its data until it can resynchronize itself to array time.
- Start observing command on a specific TE
- Configure data processing parameters for a sub-scan with a lead time of 1.5 seconds

The 32-bit data port interface on the correlator actually transfers the 1 MB at an effective burst rate of 125MB/sec. The PC's 32-bit data port card can buffer 128K of 4-byte words which is 1/4 of the total data for the 16ms period. The PC's 32-bit data port card then DMA-transfers from the 32-bit data port FIFO buffer to the PC's main memory.

Figure 25 shows the timing details of the 32-bit data port transfer from the correlator hardware to the PC's RAM and lag processing. Note that the DMA transfer is not sequential, but interspersed between the 32-bit data port transfers to the PC's 32-bit data port FIFO. The steps are:

Each 16ms the correlator hardware dumps 1024 x 256 x 4 bytes (1 MB) to the 32-bit data port card in the compute node PC. This takes 8 ms since the transfer is effectively 125 MB/sec.

The PC's 32-bit data port card has a 128K x 4-byte FIFO buffer to store intermediate results. When this buffer fills up, a DMA transfer occurs between the 32-bit data port FIFO buffer and the PC's RAM. It takes 4ms for a 66 MHz PCI bus for the 1 MB of data.

28 ms remain to process the lags of the previous dump.

This is a pipeline process where processing occurs on a previous dump.

In Table 5, the DMA transfer (task #2) is shown as CPU utilization. Although this is not completely correct, the CPU is effectively idle as no memory accesses can occur while the DMA controller is mastering the bus.

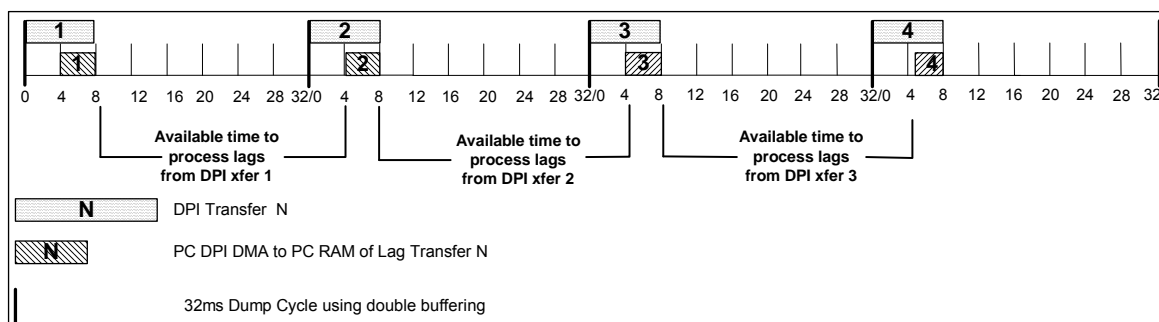



Figure 25 32-bit data port timing diagram

Task	Purpose	Execution Time (C _i) ms	Period (T _i) ms	Utilization %	Cumulative Utilization
1. Array Time	Process 48ms TE	0.500	48	1.0	1.0
2. Transfer Lags	DMA transfer of raw lags from FPDP to RAM	8.0	32	25.0	26.0
3. Sum Lags	Lag processing	1.555	28	5.6	31.6
4. Antenna Blanking	Discard invalid lag sets	3.159	28	11.3	42.9
5. Quantization Correction	2-bit 4-level VanVleck	18.665	28	66.7	108.6
6. Smoothing	Apply windowing function	3.669	28	13.1	121.7

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 53 □ of 86 □
--	--	--

Task	Purpose	Execution Time (C _i) ms	Period (T _i) ms	Utilization %	Cumulative Utilization
7. FFT	FFTW	11.333	28	40.5	162.1
8. Spectral Averaging	Time average spectra	2.150	28	7.7	169.8
9. Channel Average	Perform channel averaging	8.500	500	1.7	171.5
10. Transmit Spectra	Transmit spectral blocks to master	0.195	1000	10.0	181.5

Table 5 CDP Compute Node Rate Monotonic Analysis

For the TFB, we must add in the extra processing and the multiple FFTs – one for each up to 32 sub-bands. At this point, we do not have sufficient timing analysis, but it will undoubtedly take more processing time than shown in Table 5.

Obviously, these CPUs are over-scheduled. There are some issues that need addressing:

- These examples are based on calculations of simulated data for 2 antennas with 256 lags scaled to 32 antennas running on a 1 GHz Pentium III.
- Quad CPU boards are available. This would extend the available processing time to 64 ms by utilizing 4 buffers to which DMA could transfer – see section 3.12.5 for the double buffering discussion. This would further divide these numbers by roughly a factor of 2. Current technology allows for dual core CPUs to directly replace the single core CPUs in our existing nodes which we can then use for further timing tests. We now utilize quad-CPU boards with 2 dual-core CPUs.
- The use of specialized DSP processors is another possibility. This has not been addressed and most likely will not be done in the future.
- The Transmit Spectra task assumes that 1 Gigabit Ethernet is used, i.e., ~50 MB/sec see [27] with ~50% CPU utilization. The percent utilization is determined as such:
 - Amount of data to send every second: 10 MB which represents 10 100 ms integrations
 - Time to send out 1 second of data: 10 MB/(50 MB/sec) = 200 ms.
 - From **[Error! Bookmark not defined.]**, using a CPU utilization of 50%, the total CPU time used is: 200 ms * 0.5 = 100 ms. Thus every 1000 ms the CPU is utilized for 100 ms – Utilization = 100 ms/1000 ms = 10%
 - A backup plan could be to create a ‘network of compute nodes’ to handle the loads. In this scenario, one primary compute node would connect to the 32-bit data port and have 2 or more secondary compute nodes connected to it. The primary compute node distributes lags to the secondary compute nodes for processing. This solution lowers the computation load, but may increase bandwidth loads and definitely increases system complexity. To this end, I have specified taller racks which would allow for extra computers. Also the correlator floor is being fitted with an extra four rack subframes which can accommodate many more computers in case this is needed.
 - gcc compiler options produced an improvement of ~10 - 30%.
- Other questions which need to be evaluated
 - How much overhead exists in master-slave communication?



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 54 of 86

During R5, performance tests were done on a quad-Opteron CPU. The tests used 8K lags and included all processing steps except for residual delay correction and atmospheric phase corrections. Preliminary results look good.

Number Antennas	Processing time in milliseconds
2	0.15139162
4	0.60194354
8	2.40402212
10	3.70195242
12	5.04011172
16	8.35184091
20	13.1476833
32	33.1499615

Table 6: CDP Node processing times

32 antennas are a complete test as this is the total number of antennas handled by one CDP node. By using multiple CPUs, we can easily meet the timing specifications.


3.12.3 CDP Master Node Timing

The main function of the master compute node is a router. It sends commands from external computers to the compute nodes and routes data to the archive.

The time-critical processes include:

- Array time cannot miss a TE tick. A short, period task is triggered each 48ms with a watchdog timer to expire if the tick is missed. The watchdog timer can execute recovery routines to re-establish array time. This is the only hard, real-time task.
- Route configuration and control commands from the CCC to the appropriate compute nodes.
- Receive (from the compute nodes) and assemble all integrations for all baselines lines in a given array.
- Transmit the spectral data integrations plus channel average data sub-integrations for all arrays to the BDD at 60 MB/sec

Task	Purpose	Execution Time (C _i) ms	Period (T _i) ms	Utilization %	Cumulative Utilization
1. Array Time	Process 48ms TE	0.5	48	1.0	1.0
2. Route CCC cmds	Route config. & control cmds. from CCC	2	48	4.2	5.2
3. Spectral data collection	Collect spectral integrations from compute nodes	100	10000	1.0	6.2
4. Build XML	Build XML data for an	1550	10000	15.5	21.7

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 55 □ of 86 □
--	--	--

Task	Purpose	Execution Time (C _i) ms	Period (T _i) ms	Utilization %	Cumulative Utilization
data set	array & integration				
5. Transmit Spectra	Transmit spectral data blocks to BDD	250	10000	2.5	24.2
6. Transmit Channel Aver	Transmit channel average data blocks to BDD	100	500	20.0	44.2
7. Monitor	Monitor CDP master node	10	10000	0.1	44.3
8. Administer cluster	General cluster administration	1000	6.00E+05	0.2	44.5

Table 7 CDP Master Node Rate Monotonic Analysis

Building the binary data is the most expensive task. These execution times are derived from tests and may require revision. In R6, we will be able to evaluate the timings for large data sets.

As a backup plan, there can be multiple master nodes in a hierarchical arrangement. A primary master node can handle configurations from the CCC and there can be multiple node bridges between the internal CDP network and the external network. This plan increases the software complexity, but may be necessary to handle the data loads. These issues will be investigated during development and testing phases.

In the R6 development cycle, we will be testing end-to-end throughput from the correlator hardware to the archive in Charlottesville. This will provide definitive timing results for the CDP Master to Archive data transfers. Note that these performance tests will also include CDP node processing tests.

3.12.4 Computational Load Resolution

To provide a clear path as to how I intend to approach a solution of the computational load on the CDP, I will investigate the following:

- Compiler optimizations specific to the CPU we're using.
- BIOS/RTOS optimizations to eliminate unnecessary hardware interrupts, e.g., power management and USB support
- Dual-core CPUs resulting in 4 CPUs.
- 'Divided computer architecture' where multiple computers process data in parallel.

In R2, we have looked into the first two items which provided minor, on the order of a few percent, improvements. In R3, the first item provided an improvement of ~30% in floating point operations. The second item was mostly done as part of RTAI kernel configurations. For R4, the CDP node dual-core CPUs were purchased in order to run in a quad-CPU environment. During R5, we will determine if these quad-CPU systems are sufficient in combination with 64-bit Linux which better exploits the underlying 64-bit architecture of the Opteron and motherboard. Some reports indicate a factor of 1.5 – 2 speed improvement under 64-bit Linux. We did not complete a 64-bit version of real-time Linux during R5, so this test was not done. But as noted above, I believe that we can meet the processing requirements in 32-bit Linux. Nevertheless, moving to a 64-bit implementation should continue to be investigated.

For R6, we will continue with processing tests and further itemize the exact durations for each processing step.

During the R7.0/7.1 cycle (Nov. 2009) completed tests will be performed itemizing the durations for each step with real hardware, interrupts, etc. with a single CDP Node. A cluster scalability test utilizing 8 CDP nodes is scheduled for April 2010.



3.12.5 CDP Processing Pipeline

Figure 26 shows the data processing pipeline in the CDP compute nodes. There are three important aspects of the data processing pipeline: 1) the use of double buffering, 2) the use of real-time tasks for the various processing stages and the inter-process communication mechanisms between tasks, and, 2) the concept of arrays, that is, subsets of antennas which work together as independent units.

By having two memory buffers to which the DMA transfers can write, we can extend the time available to the CPU to process data results. The amount of available time essentially doubles by having one buffer per data pipeline per CPU.

Although the processing pipeline can be performed by a large monolithic task, it is broken up into stages by functionality – DMA transfer of raw lags to shared memory, raw lag processing, spectral processing, and data delivery. The first three are closely synchronized where one task does not start until it signaled to use the output from the previous stage. The final stage, data publishing, is asynchronous because the data receiver in the CDP master node may not consume bursts of data as fast as the CDP nodes publish it due to network latency and master node load. Therefore the data publisher task queues up blocks of spectral integrations (or channel average sub-integrations) and delivers them when it can, decoupling the synchronous processing tasks from the slower data publishing task.

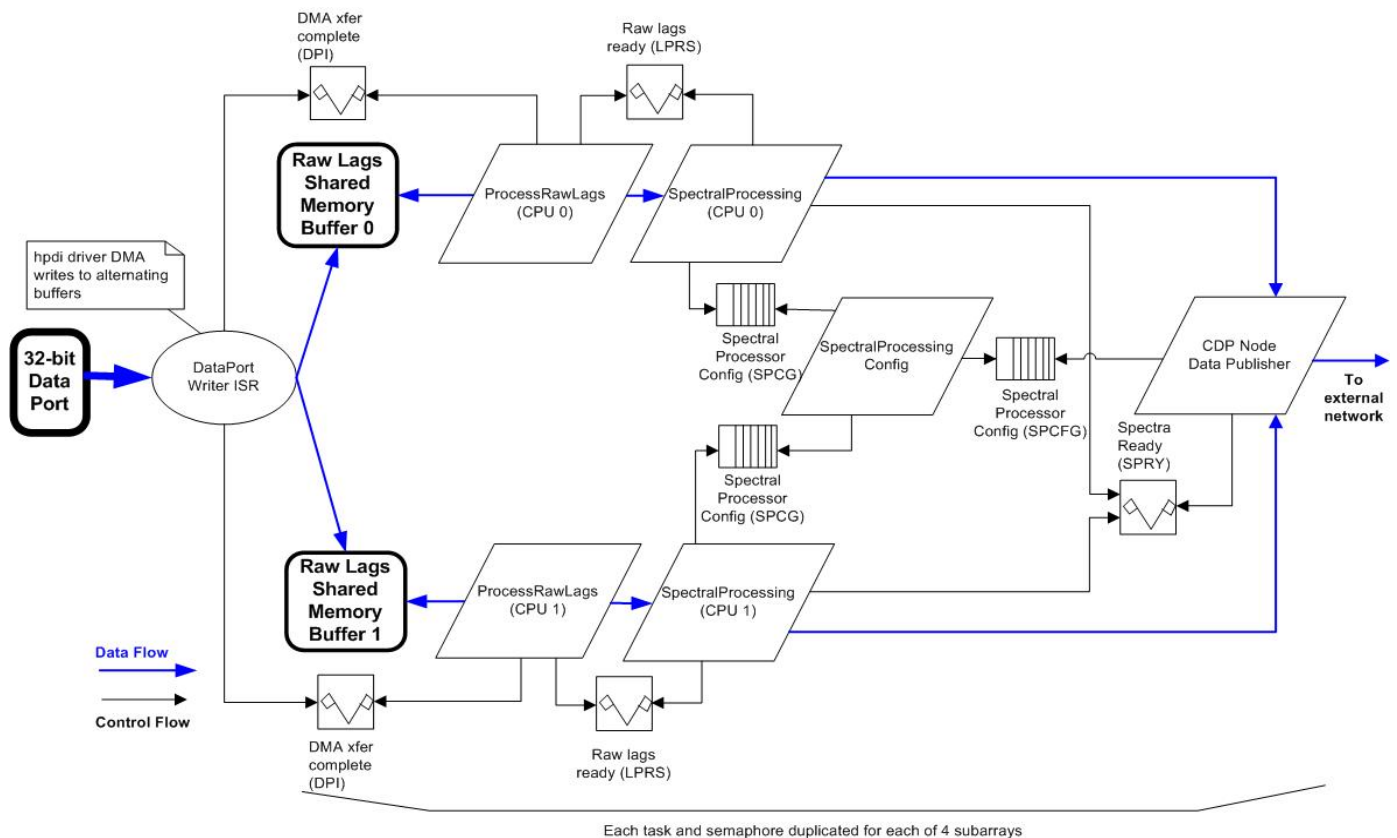



Figure 26 – CDP Processing Pipeline

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 57 of 86
--	--	--

3.13 System Reliability

Here we attempt to identify reliability issues with the Correlator subsystem computers and software. Currently there are no technical requirements specified nevertheless, this is an important issue which needs addressing.

3.13.1 Single-Point Failures

It is crucial to identify single points of failure in the Correlator subsystem. Single points of failure are computers which, if they fail, stop the flow of all data from the correlator. There are two computers which fall into this category – the CCC and the CDP master node.

3.13.1.1 Correlator Control Computer

There will be two CCCs - a primary and secondary computer. Each will be connected to the ALMA AOS network, the correlator CAN bus, and a remote power switch. Only the primary computer will be powered and running its application software. If there is a fault with this computer, the operators can reboot the primary CCC by cycling power via a remote power switch.

If the primary computer does not correctly restart, the operators will then power-down the primary CCC and then power up the secondary CCC. When it powers up it will load the same application software and become the primary computer. If a scheduling block was being executed, then it will have to be restarted or discarded as the operator sees fit. The failed CCC can then be removed and repaired by technicians at a convenient time. The expected downtime in this situation is ~15 minutes.

3.13.1.2 Correlator Data Processor Computer

The CDP master node will have an identical redundant computer as in the case of the CCC. For the CDP compute nodes, a single node failure only affects 1/16 of the correlator output, i.e., other nodes continue to produce data so the Correlator subsystem operates in a degraded mode. Because of this and because we can have only one computer connected to each correlator's DPI interfaces, we do not have plan to have redundant CDP compute nodes.

3.13.2 Sources of Failure

As the CCC and CDP nodes are identical computers, we treat them identically. There are several areas of potential hardware failure with the CDP nodes.

- Cooling – The computers have high-performance AMD CPUs which run very hot. The computer enclosures will have many fans (with filters) and the CPUs also have a cooling fan. The correlator computers will share the temperature-controlled environment of the correlator hardware which should prevent overheating as long as all of the fans are functioning. These fans are the most susceptible components to failure and can be monitored in real time. CPU and enclosure temperatures can be tracked to assist in scheduling maintenance.
- Note that low-power Opteron CPUs are now available which run at much cooler temperatures. This type of CPU should be used to minimize the overheating risk.
- Networking – Cabling and routers used by the correlator computers are hardware-replaceable items which should involve a down time of about an hour.
- Software Failures – Failures due to software fall into 3 levels of severity:
 - Low – These are warnings for which software errors have been identified in the code and are programmatically handled. These warnings are logged and software execution continues uninterrupted.
 - Medium – These are software errors which allow for execution to continue, but if enough occur, then software failure can ensue. An example of this is dynamic memory consumption increases



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 58 of 86

until there is no available memory for the application to successfully execute. These errors may or may not be logged.

- Severe – These are software errors which cause one or more major software modules to cease execution. Examples of these are null or invalid pointer references, dynamic memory exhaustion, stack overflows, etc. Often these errors are not logged before execution halts and the fact that the system has failed may be difficult to determine.

Once software execution fails, the CCC must be rebooted which can take several minutes to come up. Subsequent re-configurations of the CCC and correlator hardware will take more time. We plan to have remotely controlled power strips for remote reboots.

Spare computers and I/O boards will be available for cold-swapping broken CDP Node computers. Replacement of computers or boards should be straight-forward with modular I/O connectors facilitating the cable connections. Replacements should take approximately 1 hour. Failed computers can then be repaired at the OSF.

3.13.3 High altitude disk server

During commissioning one may expect that the fiber connection between the AOS and OSF could fail at which point operations would be halted. It is possible to create a disk server at the AOS technical building from which all of the diskless computers in Correlator and Control could utilize as boot servers in this situation. This high-altitude server would contain hard disks rated at 5000 meters. In 2007, these 73 GB disks cost about \$500US and 5-10 drives would be required to serve all of the diskless computers.

3.13.4 Error Handling


This section discusses the various scenarios of software failures and how they are handled. Each recovery scenarios must be dealt individually due to how different parts of the correlator software interfaces with others both internally and externally.

3.13.4.1 External interfaces to the CDP Master

The CDP Master has two external subsystem clients, the bulk data distributor and the Data Capturer component. Recall that for each integration and sub-integration sent to the bulk data distributor, corresponding meta data is sent to the Data Capturer (actually an array of structures with one element for each integration or sub-integration for the entire sub-scan). If the CDP Master fails to send data to the bulk data distributor, a CORBA exception is thrown which the CDP Master catches and flags that block of data as bad in the corresponding Data Capturer structure. Thus post-processing software can determine that there is missing binary data.

If the Data Capturer component has failed at the end of a subscan, it finalizes its bdf blob with an abortObservation element and sends to DC (sendSubScanCorrelatorData) meta-data including only the number of completed integrations and bytes in the blob. That is, at this level of interaction CORR has provided to DC all available meta-data at the moment the error has occurred. This also means that the operation of the correlator sub-system cannot affect the collection of data beyond a sub-scan boundary. On the other hand, CONTROL will know of the aborted/stopped sub-scan by means of a callback mechanism which should be ready to use by September 23th 2009.

The CDP Master also publishes and subscribes to events on notification channels. As stated earlier, if antenna blanking or delay event data are not received in a timely fashion, then spectral data for the affected integrations are blanked with blanking flags set accordingly along with the spectral data. At this time, there is no way to know if the notification channel on which integration events are published fails. An external agent should monitor the CORBA notification service for failures and restart any notification channels as needed. Of course this may not be transparent requiring some reinitializations, but at this time, I'm not sure how to handle this.

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 59 □ of 86 □
--	--	--

3.13.4.2 External interfaces to the CCC

The CCC software only receives data and CORBA function calls from external subsystems. The CCC subscribes to notification channels whose failures were previously discussed.

3.13.4.3 Internal interfaces between CCC and CDP Master

The CCC invokes methods on the CDP Master to start and stop subscans. If the CCC fails invoking these methods, then it should throw an exception. The Correlator Master component should then recognize that the CDP Master has failed and will require restarting. Currently the restarting technique will require all components to shut down and restart placing all of the correlator software in its initial ready state.

3.13.4.4 Internal interfaces between CDP Master and CDP nodes

3.13.5 MTBF

Exact mean time between failures numbers are difficult to determine as complete analysis requires MTBF for all components. [28] discusses the MTBF for the CCC and CDP systems. In summary, the MTBFs for the correlator computers are:

- CDP compute node cluster: approximately 1.2 years
- CDP master node: approximately 19.3 years
- CCC : approximately 13.3 years

This means that the CDP cluster will have approximately one failure per year.

3.14 System Startup

The Correlator subsystem must start in a predefined fashion. As the Correlator subsystem is a device of the Control subsystem, startup of the Correlator is controlled by the Control subsystem and similar to other devices. While details of the start up sequence are in the Control subsystem design document, an outline is provided here.

Control's MasterComponent is responsible for starting the correlator software. The MasterComponent starts the Correlator's components – these are the ACS LifeCycle methods. Most of the components of the Correlator will derive from *ControllerComponent* as defined by the Control subsystem design thus inheriting the *ComponentLifecycle* interface, an *ErrorAnalyzer* class, and contain one or more resources.

Once the software components are operational, the MasterComponent instructs the Correlator (via the *Controller* interface) to start its hardware dependent startup sequence. This is a two-pass system which allows for explicit execution of hardware dependent software, e.g., RTAI kernel modules, CAN node notification and verification of hardware, etc. In the second pass, CORBA notification channel connections.

Some external subsystems must be already started while other subsystems will wait on the Correlator subsystem for services. The sequence diagram in Figure 27 shows the startup procedure.

A bootserver computer (currently gns) must be running first to provide remote disks for the correlator computers to booting from and NFS-mount their file systems. The ARTM must also be operational in order to synchronize the CCC and CDP computers to Array Time. Once the computers have their time set, they subscribe to the various data notification channels. Lastly, the CDP creates the data streaming channels to the BDD and the integration event notification channel.

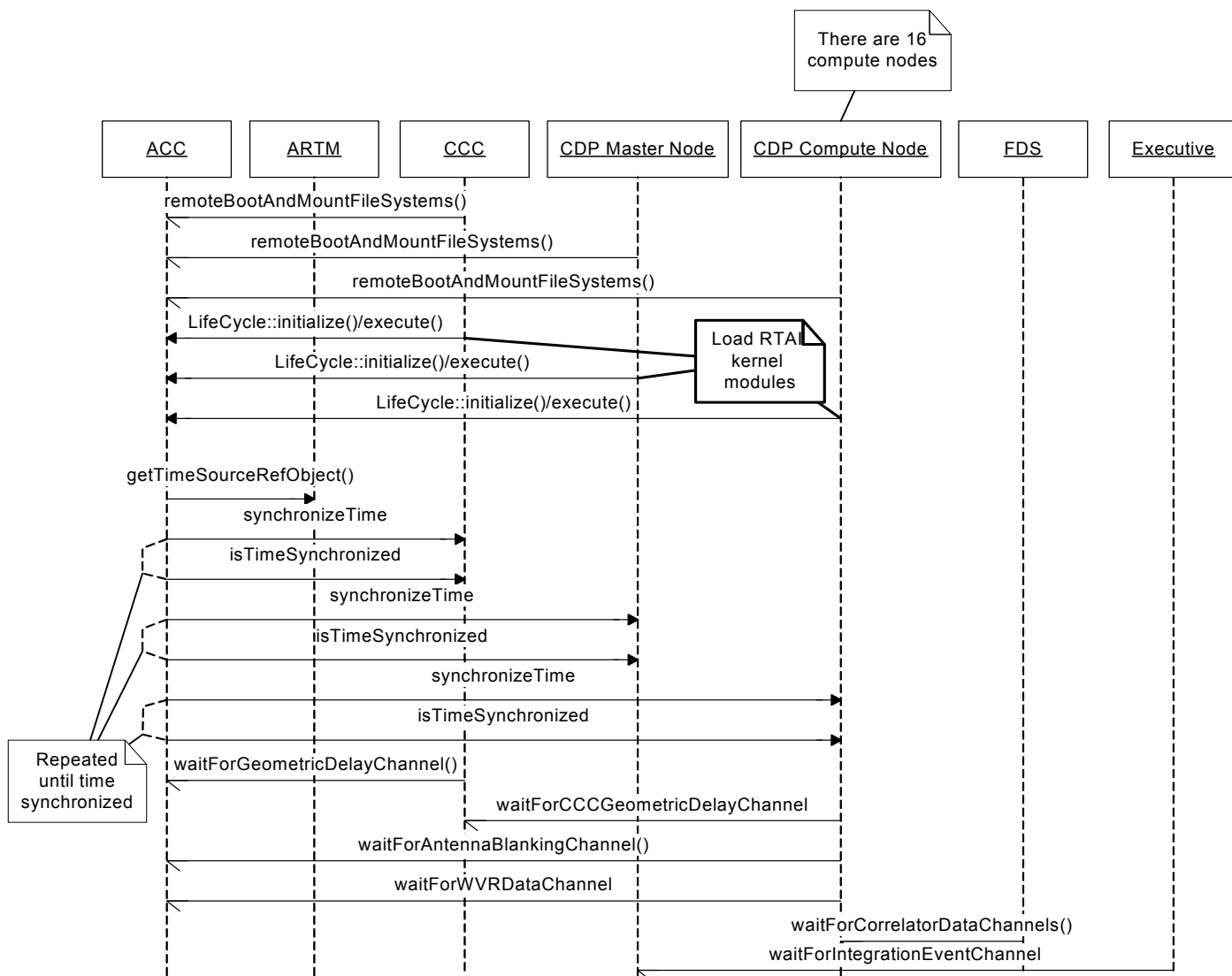


Figure 27 – Correlator startup sequence

3.14.1 UPS Monitoring

Each rack of computers (there will be 4 racks each with 4-7 computers each) will have a uninterrupted power supply providing power line filtering against voltage ‘glitches’ and short-term battery backup (approximately 5 minutes). The correlator computers will share the same power of the correlator, so if the power fails for the correlator, it also fails for the computers. There is no battery backup for the entire correlator – only minimal monitoring cards are battery-backed up, so once the power fails, there is no need to run the correlator computers until the power returns. But the computers should shutdown gracefully and inform external subsystems of the impending shutdown. No hardware will be damaged if this shutdown sequence does execute completely.

The sequence diagram in Figure 28 shows the series of events in the UPS shutdown process. Each UPS will have a serial connection to a single computer in the rack (the ‘main rack computer’) to which it sends a signal notifying that the power has failed.



The main rack computer issues a standard Linux shutdown command on each slave rack computer. Each slave rack computer notifies the Control system's master component that it is shutting down due to a power failure and then executes a shutdown script which unloads all of the ACS containers in an orderly fashion. The main rack computer then notifies Control's master component that it is shutting down and then executes a shutdown script.

It is the responsibility of the operator to power on the computer rack (via a remote Ethernet power switch) once the power is restored to the correlator.

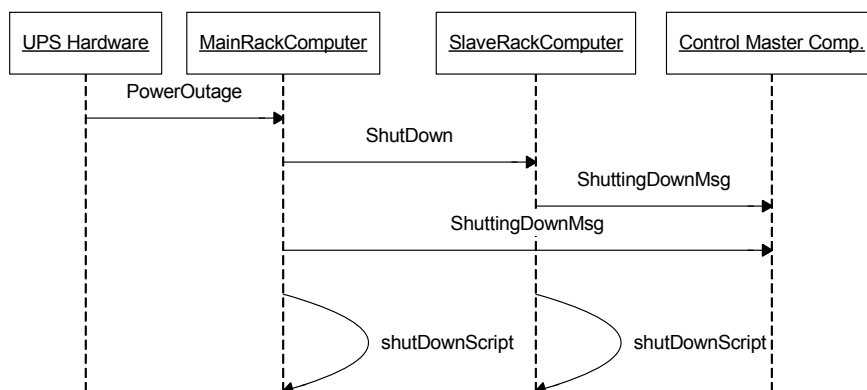


Figure 28 UPS shutdown sequence diagram

3.15 Telescope and Monitor Configuration Database

The TMCDB provides much information defining hardware configuration parameters needed by the correlator software at a given point in time. These items do not change frequently, but when they do, the Correlator subsystem will be notified to reload the TMCDB using the Maintenance interface previously discussed. These items include:

- Mapping of antenna inputs to CAIs
- Mapping of baseband pairs to correlator quadrants.
- LO offsetting parameters for each antenna
- A software description of the correlator software monitor properties
- Version and type of correlator hardware
- Number and type of CAN nodes and LRUs (Line Replaceable Units) in the correlator hardware
- Computer hardware descriptions
- Antenna/Baseband fixed delay offsets
- Walsh function switching sequences for each antennas

4 Correlator Simulator

The purpose of the correlator simulator is to provide a test bed for the CCC and CDP software without relying on the correlator hardware. The correlator simulator presents software interfaces to the CCC and CDP components simulating the behavior of the correlator. What cannot be simulated is the speed of the correlator.

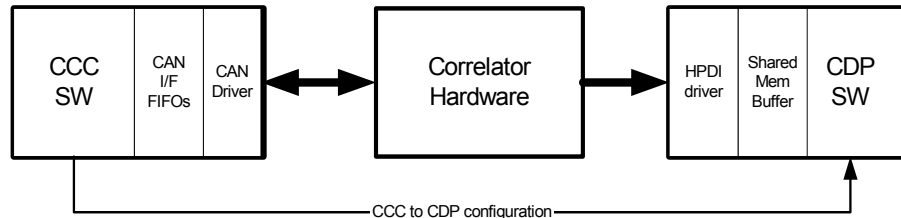


Figure 29 Hardware view of correlator, CCC, and CDP

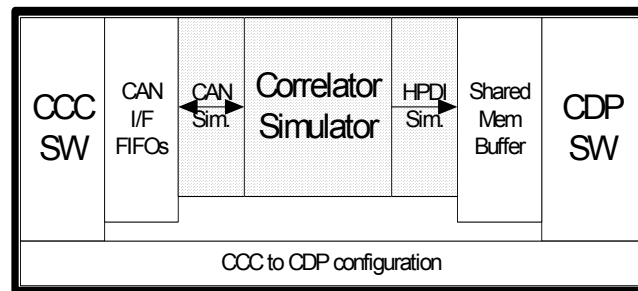


Figure 30 Simulator View

Each box in bold represents a separate computer (or piece of hardware in the case of the correlator) with software components identified within. The differences between Figure 29 and Figure 30 are:

- In Figure 29, we see that
 - the CCC and CDP have special hardware interface drivers
 - there are physical connections between the CCC, CDP and correlator
 - configuration flows from the CCC to the CDP via an external (Ethernet) connection
- In Figure 30,
 - the hardware interface drivers are replaced by software interfaces
 - all software components including the correlator simulator reside on a single computer with inter-process communication (IPC) connections replacing the physical connections


From the viewpoint of the CCC and CDP software, we see that the CCC pushes CAN commands to the correlator via a real-time FIFO (**TxFIFO**) interface and accepts response from the correlator via a separate real-time FIFO (**RxFIFO**). The CDP receives raw data from the correlator via a shared memory buffer. Here we see that the interfaces are abstracted and allow for the correlator simulator to ‘plug in’ an adapter which simulates the data for these interfaces.

The correlator simulator accepts CAN commands through the **TxFIFO** and returns appropriate responses via the **RxFIFO**. In turn, it writes raw data to a shared memory buffer which is read by the CDP software and processes the raw data (called ‘lags’) into raw spectra (as a result of a start subscan command).

The simulator requires the use of RTAI and that CIPT Management at CDR6 agreed that a non-RTOS mode of simulation is not required.

4.1 Correlator Simulator Software Components

Here we define the major packages that are necessary for the correlator simulator

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 63 □ of 86 □
--	--	--

4.1.1 CAN Commands

This package is responsible for accepting CAN commands, interpreting them, and returning ‘meaningful’ responses. A meaningful response is one which provides an expected value within a reasonable period of time. What cannot be expected is to perform this functionality on the same time basis as the physical CAN interface.

4.1.2 Lag Data

This package responds to a ‘start subscan command and begins delivering simulated raw lag data to the CDP via its shared memory interface. The format and duration of the data must correspond to the configuration received via a CAN command.

4.1.3 Configuration Tool

As there are many phases of the correlator throughout the lifetime of the ALMA construction phase, the simulator needs to adapt to these changes. This allows for the correlator software to be tested against a specific version of the correlator hardware. For example, the prototype correlator is a two-antenna system with limited functionality and commands. Also the CAN command protocols are specific to the prototype correlator and are different (or non-existent) in the final correlator. The final correlator will be built in stages – it would be beneficial to determine how these different configurations affect the correlator software.

4.2 ALMA Observatory Simulator

In R3, an end-to-end observatory simulator was developed. This shared simulator produces raw lag data independently of the existing correlator simulator. The shared simulator generates lags taking into account other hardware simulators, e.g., antenna position and focus parameters, geometric delays, etc. These lag sets are used by the correlator simulator which then routes them to the CDP as described above.

5 Correlator GUI


5.1 Correlator configuration and spectral viewing

A graphical user interface (*CorrGUI*) has been developed for engineering tests and has matured to become a valuable component in testing the correlator software and will be used during ALMA commissioning. This allowed us to integrate it into the Executive's subsystem Operator Master Client (OMC) application as part of the GUI function based team. A user manual can be found on the Twiki [29] which provides a detailed view of its functionality.

CorrGUI allows a user to configure the correlator hardware and software and collect and view the spectral results. It utilizes the software interface that the Control subsystem uses for configuration and taps into the data path between the CDPNode and CDPMaster software components for the spectral results. As development of the correlator software progresses, we add functionality to take advantage of the new features.

In the coming year, we will work more on incorporating it into the OMC and to implement a 'view-only' option which allows the user to view spectra and monitor properties, but not to configure the correlator.

5.2 Correlator monitoring and diagnostics

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 64 of 86
--	--	--

6 Correlator Monitoring GUI

This section discusses the correlator monitoring GUI. This GUI will be manifold and include the following capabilities:

- Provide a high-level view of the “correlator” status. This status includes correlator hardware and correlator computer hardware
- The ability to quickly identify that the status of correlator hardware either as fully functional, partially functional, or fully failed
- The ability to “drill down” to the LRU (Line-Replaceable Unit) level to determine the specific LRU which is faulty
- The ability to view specific monitor data for a given LRU
- Provide a common monitoring GUI for both the ACA and ALMA-B correlators.
- Provide an interface to run hardware diagnostic tests on the correlator.

6.1.1 Error Diagnostic

The diagram in Figure 31 presents the hierarchical view of the correlator. There are 5 levels applicable to the ALMA-B correlator, (there are fewer levels for the ACA correlator). The user is initially presented with level 1, a block labeled “ALMA-B Correlator”. If its color is green, then all is okay. If not, then the user clicks this box to begin the drill-down search to find the failed LRU. The user successively clicks on blocks until s/he reaches the failed LRU guided by the color codes of each block. In Figure 31, we see that the failed LRU is Quadrant 1, Station Rack 1, Bin 1, Card 1 by following the non-green color sequence from yellow to orange to red. Finally, we see that the monitor point PS2 is over-voltage causing the failure.

6.1.2 Monitor Point Viewing

We can also identify specific monitor points to view. The user drills down to any LRU. Once the monitor points are displayed for a given LRU, s/he may select any monitor point widgets and drag them to a special panel which allows a real-time display of monitor data similar to what is shown in Figure 32.

6.1.3 Diagnostic Testing

The ALMA-B correlator has many low-level tests developed by the correlator engineers to validate internal correlator signal paths. It is envisioned that during idle, non-observation times, these tests can be initiated with the results checked at one or more points along these signal paths. These tests would be started by specific interfaces to the CCC which in turn, would send specific CAN commands to the various control microprocessors in the correlator.

Figure 33 shows diagrammatically how data would flow for a given test from each set of cards for a quadrant. The data would flow between cards, bins and racks with results being checked at appropriate points. The intermediate and final results would be available to the CCC allowing indication of any malfunctioning hardware.



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 65 of 86

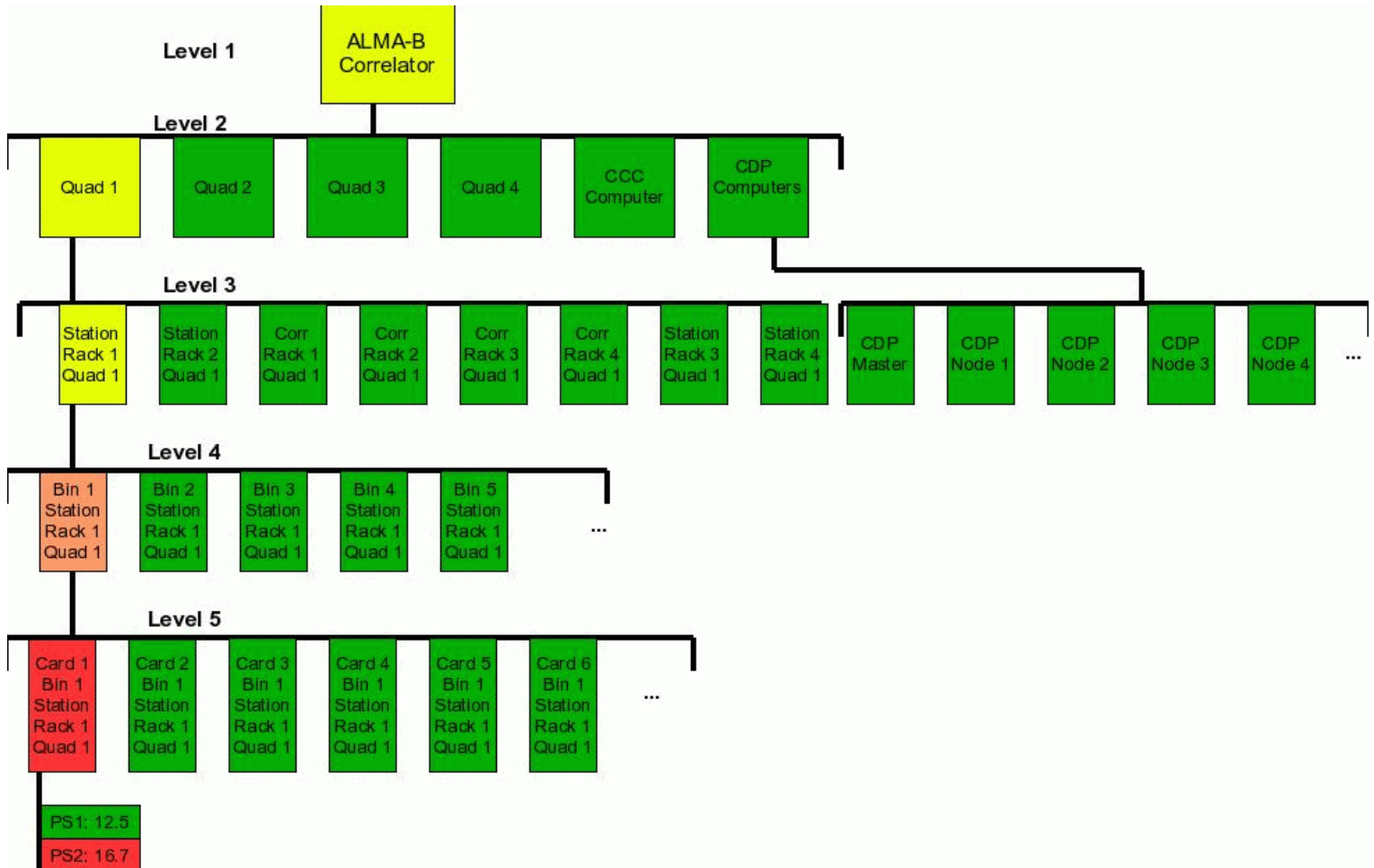


Figure 31 Correlator Monitoring Heirachical View

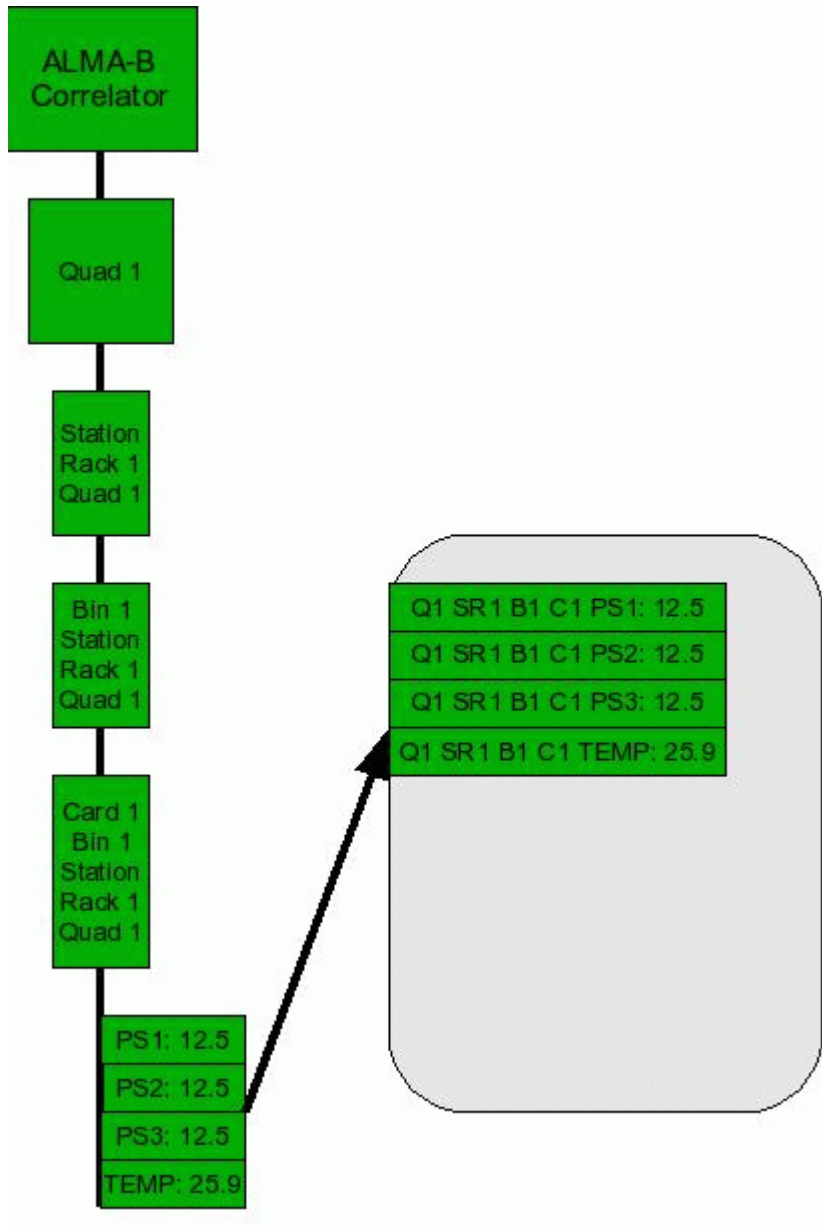


Figure 32: Correlator Monitoring View



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 67 of 86

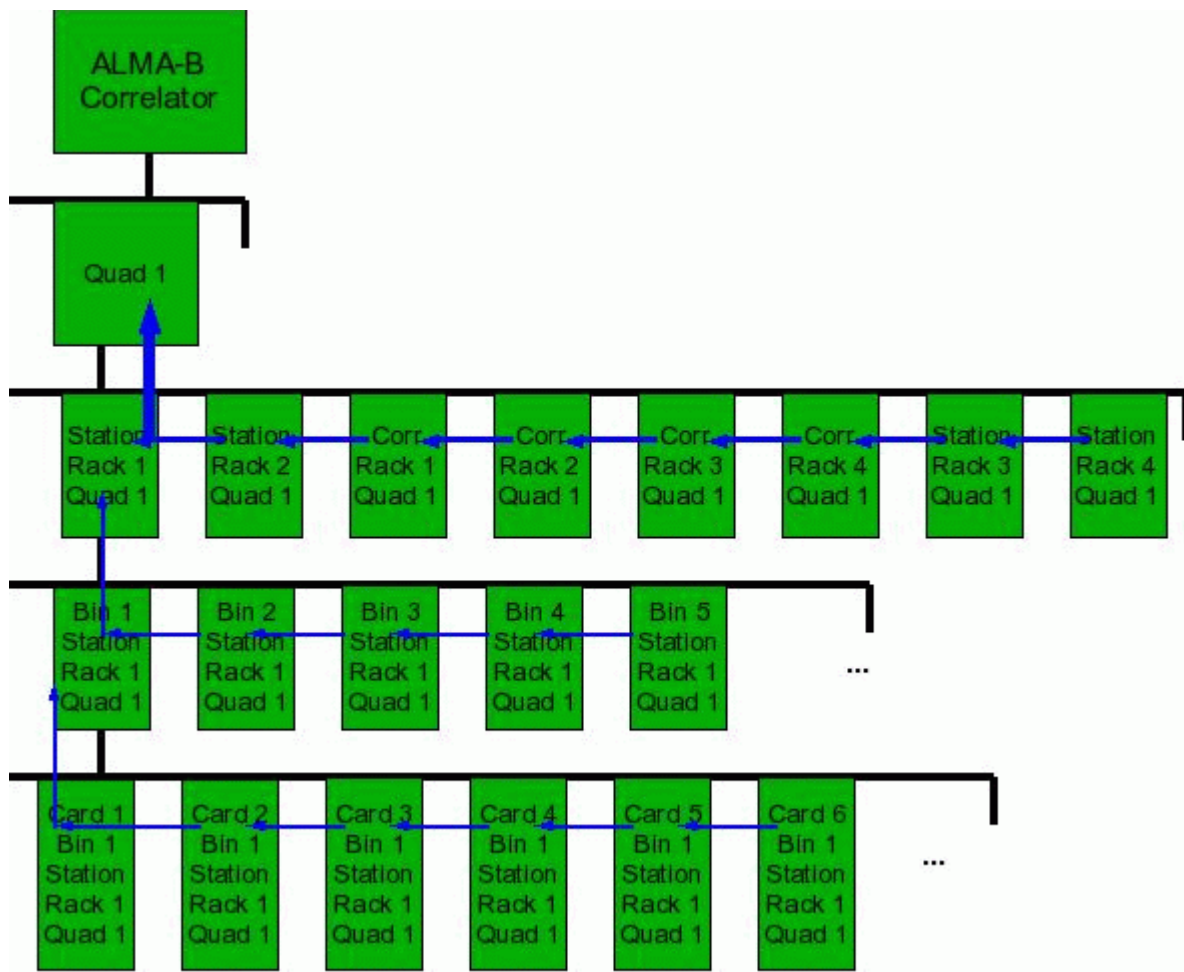




Figure 33: Correlator Diagnostic View


	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 68 □ of 86 □
--	--	--

7 References

- 1 *Real Time Application Interface*, <http://mail.aero.polimi.it/~rtai/>.
- 2 Lucas, R. et al. *ALMA Science Software Requirements and Use Cases*, ALMA Computing Memo 11, 2001-May-03, <http://www.mma.nrao.edu/development/computing/docs/joint/0011/ssranduc.pdf>
- 3 Pisano, J., Hale, A., Scott, S., *ALMA Correlator Software Granular Requirements*, Vers. 1.4 (2005), <http://almasw.hq.eso.org/almasw/bin/view/SSR/SoftwareSubsystemsDetailedRequirements> .
- 4 Glendenning, B., *Operations Requirements and Specification on the Computing IPT COMP-70.00.00.00-005-A-SPE Version: A*
- 5 Chiozzi, G., Gustafson, B., Jeram, B., ALMA Common Software, <http://www.eso.org/~gchiozzi/AlmaAcs/>
- 6 The Ace ORB, <http://www.cs.wustl.edu/~schmidt/TAO.html>
- 7 Napier, P. *ALMA Use of LO Offsetting for Spurious Signal Suppression and Sideband Rejection*, SYSE-80.04.00.00-018-A-DSN, 2007.
- 8 Broadwell, C.M., Pisano, J. A., *Interface Control Document From Correlator to Computing Correlator Software*, ALMA-60.00.00.00-70.40.00.00-A-ICD (2004).
- 9 D'Addario, L., *Notes on Delay Tracking for ALMA Resolution & Tolerance*, 2003-Feb-08
- 10 Broadwell, C. M., *Long Term Accumulator CAN Protocol*, http://www.mma.nrao.edu/development/correlator/lta/LTACCC_IPD.pdf (2002).
- 11 Brooks, M., D'Addario, L., *ALMA Monitor and Control Bus Interface Specification*, ALMA-SW-0007, (2001).
- 12 Broadwell, C., *The Long Term Accumulator Subsystem Manual*, CORL-60.02.03.00-002-A-MAN, (2003).
- 13 Pokorny, M., Pisano, J., *Science Data Model Binary Data Format*, Ver. 1.0 (2008).
- 14 Viallefond, F., *ALMA Science Data Model*, <http://aramis.obspm.fr/~alma/AEDF/ver2.0/allhtm/>, (2005).
- 15 Comoretto, G., *Algorithms and formulae for hybrid correlator data correction*, (2004).
- 16 Scott, S. *SSR Specifications and Clarifications of ALMA Correlator Details*, <http://almasw.hq.eso.org/almasw/pub/CORR/CorrelatorDocuments/AlmaCorrel.pdf>, (2003).
- 17 Thompson, A.R., et. al., *Interferometry and Synthesis in Radio Astronomy*, Krieger, (1998).
- 18 Schwab, F. *Van Vleck Correction for the GBT Correlator*, (Draft, 2002).
- 19 The Fastest Fourier Transform in the West, <http://www.fftw.org>
- 20 AMD Core Math Library (AMCL), ver. 2.5.0, http://www.developwithamd.com/apppartnerprog/acml/docs/acml_userguide.pdf (2004).
- 21 Bulk Data Transfer Discussion, <http://almasw.hq.eso.org/almasw/bin/view/ACS/BulkDataTransfer> .
- 22 Escoffier, R. *The ALMA Correlator Block Diagram*, <http://www.mma.nrao.edu/development/correlator/BLOCK.pdf>

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 69 □ of 86 □
--	--	--

-
- 23 Escoffier, R., Broadwell, C., *The ALMA Correlator Long Term Accumulator*, ALMA Memo No. 294 (2000).
 - 24 Quertier, B., et. al, *Enhancing the Baseline ALMA Correlator Performances with the Second Generation Correlator Digital Filter System*, ALMA Memo 476 (2003).
 - 25 Baars, J. ed. *ALMA Construction Project Book*, Ver. 5.50,
<http://www.mma.nrao.edu/projectbk/construction/> (2002).
 - 26 Briand, L.P., Roy, D.M., *Meeting Deadlines in Hard Real-Time Systems – The Rate Monotonic Approach*, IEEE Computer Society, (1999).
 - 27 Hasegaw, Y., et. al., *DAQ/EF-1 Event Builder system on Linux/Gigabit Ethernet*,
<http://rd13doc.cern.ch/Atlas/Notes/147/Note147-8.html> (2000).
 - 28 Zivick, J. *ALMA Computing Reliability Analysis Final Report*, SYSE-80.11.00.00-005-B-REP (2005).
 - 29 Pisano, J. *Correlator GUI Manual*, <http://almasw.hq.eso.org/almasw/bin/view/CORR/CorrGuiManual>

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 70 of 86
--	--	--


8 System Interfaces

This section discusses all external interfaces provide by and used by the Correlator subsystem.

8.1 Package – Interface Relationship

This section shows the relationship of interfaces, data streams and events defined in the Correlator IDL files to packages which have been defined in this document. As the following table shows, many interface functions are divided among many packages.

Interface Item	Packages
alma.Correlator.ObservationControl	ACS CCC Interface CCC Command Dispatcher Array Management CAN I/F CCC_CDP_IF CDP Configuration TE Scheduler
alma.Correlator.CCC_Monitor	CCC Monitor ACS CCC Interface CAN I/F
alma.Correlator.CDP_Monitor	CDP Monitor
alma.Correlator.ObservationQuery	CCC Monitor
alma.Correlator.Maintenance.Diagnostic alma.Correlator.Maintenance.Quadrature alma.Correlator.Maintenance.CorrCanMgr	CCC Maintenance CAN I/F CCC Command Dispatch TE Scheduler
alma.Correlator.ConfigurationValidator	Corr. Config Validation
alma.Correlator.ArrayTime	Array Time Interface (CCC & CDP)
PublishIntegrationEvent	Master Data Publisher
WVRValueEvents	CDP Node WVR Correction
AntennaBlankingEvents	CDP Node Lag Processing
GeometricDelayModelEvents	CDP Node Residual Delay Spectral Processing ACS CCC Interface Geometric Delay CAN/IF
Correlator Data Stream	HPDI32 Lag Processing Spectral Processing Atmospheric Phase Correction sideband separation Node Data Publisher

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 71 □ of 86 □
--	--	--

Interface Item	Packages
Channel Average Stream	Lag Processing Data Processing Data Publisher
Monitor Stream	ACS CDP Interface CCC Monitor CDP Node Node CDP Monitor Master CDP Monitor

Table 8 Interface-Package cross reference

9 Appendices

9.1 Minimum Integration Duration

Determining the minimum integration duration for visibilities has many constraints. I outline the technical constraints and then conclude with request that the SSR provide specific scientific use cases to set a minimum integration duration. Note that while the technical details are well-understood, the scientific use cases need further input.

Determining the minimum integration duration for the ALMA-B correlator is not straightforward due to the parallel processing of groups of CAIs. The correlator hardware is divided into 4 sets of 32x32 CAIs (called a 'correlator card') with each correlator card's output going to a single CDP node. Thus, the data that flows to each CDP node from a correlator card is an independent pipeline so the minimum data rate is specified by the number of CAIs in use for a given correlator card.

For this reason, the table below, Table 9, shows the minimum dump time for 1 – 32 CAIs and 256 – 8192 lags. The maximum data rate for a correlator card is:


$32 \times 32 \times 256$ lags every 16 ms or $32 \times 32 \times 256 = 262,144$ lags

For a data rate of $262,144 / 0.016$ secs = 16.384×10^6 lags/second.

Due to the parallel nature of the 4 correlator cards, the minimum dump duration for 33 – 64 CAIs is identical to the first 32 for a given resolution.

The user *can* play some games by knowing which antennas are connected to which CAIs and consequently which correlator cards to squeeze out more antennas at a lower dump duration. This is discouraged as the mapping of antennas to CAIs is a run-time issue and would require specific planning. But if one tries to do this, then s/he can get up to 16 times the number of antennas processed for a given dump duration.

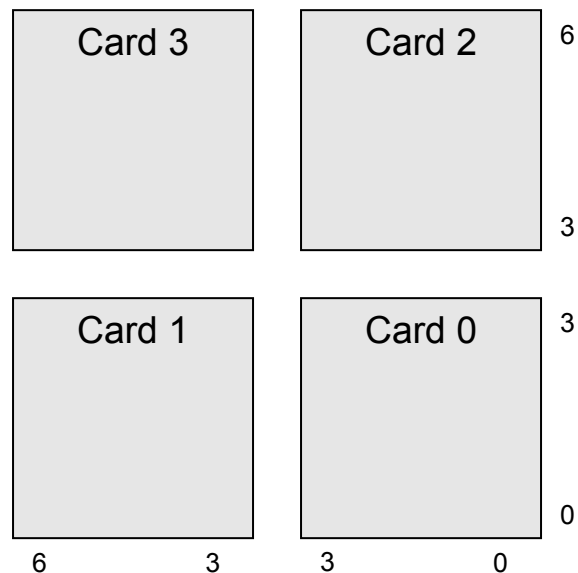
Each correlator card is physically connected to a CDP node computer, so we can also view the correlator cards as CDP nodes. If the antennas are connected to the CAIs such that for an array of dimension N the maximum number of antennas associated to one single CDP node is $f(N) = n$ then 'n' is the number of antennas listed in column 1 of Table 9. If the arrays follow in general a certain pattern then the properties of the function $f()$ could be optimized by cleverly connecting the correlator inputs.

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 72 of 86
--	--	--

The best scenario happens when $f(N)=N/4$. In this case the user could see a minimum dump duration improved by a factor of 16, at most (not always, though). The worst scenario happens when $f(N)=\min\{N,32\}$, this scenario is shown by inspecting Table 1. These considerations bound what the user will actually see as a minimum dump duration in real life.

Note also that for configurations prepared off-line, without access to information about the current assignment of correlator inputs, it is possible that at the moment of actually executing the sub-scan, the integration time becomes to be impossible to attain! Thus, we provide the table, but planning (if possible at all) for an optimum $f()$ function should be kept in mind.

The correlator cards are divided into 4 simple grids:



9.1.1.1 Archive Limitations


Andreas Wicenec, notes that there currently are technical limitations on the frequency of data transmission to the archive. In the following Wiki page: <http://websqa.hq.eso.org/almasw/bin/view//Archive/CreateFiles>, and from private communications with Andreas, the archive team is working on tuning the archive system to achieve an average minimum time to write a file is ~100 milliseconds.

There is some possibility to group together multiple short integrations within the CDP Master computer over a longer interval to meet this technical specification, but this would require a substantial amount of coding effort.

Also, there currently is a maximum data rate into the archive of 60 MB/sec which places an overall limitation on data rates.

9.1.1.2 Sideband separation

Sideband separation using 90° phase switching via Walsh function sequences require that a correlator dump must complete a full sequence of Walsh switching states. The length of this sequence depends on the number of antennas in an array. For the ACA correlator with 16 antennas, the minimum duration to complete a full Walsh function sequence is 512 ms or 32 16-ms states. The ALMA-B correlator with 64 antennas requires a minimum dura-

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 73 □ of 86 □
--	--	--

tion of 2.048 seconds or 128 16-ms states. Note that there has been discussions of dumping out one half of the sequence independently of the second half in order to obtain 1 second sub-integrations. Also for the ALMA-B correlator, this imposes the use of TDM or 256-lag FDM modes only. See *Walsh Function Definition for ALMA*, D. Emerson, ALMA Memo 565, 2006, for more details.

9.1.1.3 Science Requirements

Of course, the major driver for these minimum times are scientific. We believe that we need some solid use cases from the SSR to justify the work to overcome the technical hurdles outlined in this note. These use cases should include the following information:

- Number of antennas in an array
- Minimum integration duration
- Spectral resolution
- Sub-scan and scan durations which define the overall duration of these high data rates.
- Auto-correlations only versus visibility data.
- OTF imaging, pulsar gating, and ALMA VLBI

At this time, (CDR6), I have not pursued these specifications with the Science IPT, but will do so during R6.



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 74 of 86

	256	Cross Min Dump Time (secs)	512	Cross Min Dump Time (secs)	1,024	Cross Min Dump Time (secs)	2,048	Cross Min Dump Time (secs)	4096	Cross Min Dump Time (secs)	8,192	Cross Min Dump Time (secs)
# Ants												
1	256	0.016	512	0.016	1,024	0.016	2,048	0.016	4,096	0.016	8,192	0.016
2	1,024	0.016	2,048	0.016	4,096	0.016	8,192	0.016	16,384	0.016	32,768	0.016
3	2,304	0.016	4,608	0.016	9,216	0.016	18,432	0.016	36,864	0.016	73,728	0.016
4	4,096	0.016	8,192	0.016	16,384	0.016	32,768	0.016	65,536	0.016	131,072	0.016
5	6,400	0.016	12,800	0.016	25,600	0.016	51,200	0.016	102,400	0.016	204,800	0.016
6	9,216	0.016	18,432	0.016	36,864	0.016	73,728	0.016	147,456	0.016	294,912	0.032
7	12,544	0.016	25,088	0.016	50,176	0.016	100,352	0.016	200,704	0.016	401,408	0.032
8	16,384	0.016	32,768	0.016	65,536	0.016	131,072	0.016	262,144	0.016	524,288	0.048
9	20,736	0.016	41,472	0.016	82,944	0.016	165,888	0.016	331,776	0.032	663,552	0.048
10	25,600	0.016	51,200	0.016	102,400	0.016	204,800	0.016	409,600	0.032	819,200	0.064
11	30,976	0.016	61,952	0.016	123,904	0.016	247,808	0.016	495,616	0.032	991,232	0.064
12	36,864	0.016	73,728	0.016	147,456	0.016	294,912	0.032	589,824	0.048	1,179,648	0.080
13	43,264	0.016	86,528	0.016	173,056	0.016	346,112	0.032	692,224	0.048	1,384,448	0.096
14	50,176	0.016	100,352	0.016	200,704	0.016	401,408	0.032	802,816	0.064	1,605,632	0.112
15	57,600	0.016	115,200	0.016	230,400	0.016	460,800	0.032	921,600	0.064	1,843,200	0.128
16	65,536	0.016	131,072	0.016	262,144	0.016	524,288	0.048	1,048,576	0.080	2,097,152	0.144
17	73,984	0.016	147,968	0.016	295,936	0.032	591,872	0.048	1,183,744	0.080	2,367,488	0.160
18	82,944	0.016	165,888	0.016	331,776	0.032	663,552	0.048	1,327,104	0.096	2,654,208	0.176
19	92,416	0.016	184,832	0.016	369,664	0.032	739,328	0.048	1,478,656	0.096	2,957,312	0.192
20	102,400	0.016	204,800	0.016	409,600	0.032	819,200	0.064	1,638,400	0.112	3,276,800	0.208
21	112,896	0.016	225,792	0.016	451,584	0.032	903,168	0.064	1,806,336	0.112	3,612,672	0.224
22	123,904	0.016	247,808	0.016	495,616	0.032	991,232	0.064	1,982,464	0.128	3,964,928	0.256
23	135,424	0.016	270,848	0.032	541,696	0.048	1,083,392	0.080	2,166,784	0.144	4,333,568	0.272
24	147,456	0.016	294,912	0.032	589,824	0.048	1,179,648	0.080	2,359,296	0.160	4,718,592	0.304
25	160,000	0.016	320,000	0.032	640,000	0.048	1,280,000	0.080	2,560,000	0.160	5,120,000	0.320
26	173,056	0.016	346,112	0.032	692,224	0.048	1,384,448	0.096	2,768,896	0.176	5,537,792	0.352
27	186,624	0.016	373,248	0.032	746,496	0.048	1,492,992	0.096	2,985,984	0.192	5,971,968	0.368
28	200,704	0.016	401,408	0.032	802,816	0.064	1,605,632	0.112	3,211,264	0.208	6,422,528	0.400
29	215,296	0.016	430,592	0.032	861,184	0.064	1,722,368	0.112	3,444,736	0.224	6,889,472	0.432
30	230,400	0.016	460,800	0.032	921,600	0.064	1,843,200	0.128	3,686,400	0.240	7,372,800	0.464
31	246,016	0.016	492,032	0.032	984,064	0.064	1,968,128	0.128	3,936,256	0.256	7,872,512	0.496
32	262,144	0.016	524,288	0.032	1,048,576	0.080	2,097,152	0.144	4,194,304	0.272	8,388,608	0.528

Table 9 Minimum Integration Durations



9.2 Equations

Equations used in this design are summarized here.

9.2.1 V_s Subtraction

For each lag value the following correction is performed to remove the correlator chip multiplication bias:

$$\text{lag}(k) = \left(\frac{(\text{lag}(k) - \text{dumpIntegrationCounts})}{\text{lag}(0)} \right)$$

where *dumpIntegrationCounts* is a value dependent on internal ALMA-B correlator hardware aspects.

9.2.2 Lag Normalization

See ‘Spectral Normalization’ [16] for details.

9.2.3 Geometric Delays

The geometric delays are discussed in section 3.8.3.8.

9.2.4 Atmospheric Phase Correction

See section 3.8.3.9.

9.2.5 Digitization Correction

See [18] for details.

9.2.6 Windowing Functions

Bartlett:

$$w(k+1) = \begin{cases} \frac{2(k)}{n-1}, & 0 \leq k \leq \frac{n}{2} - 1 \\ \frac{2(n-k-1)}{n-1}, & \frac{n}{2} \leq k \leq n-1 \end{cases}$$

Blackman:

$$w(k+1) = 0.42 - 0.5 \cos\left(2\pi \frac{k}{n-1}\right) + 0.08 \cos\left(4\pi \frac{k}{n-1}\right) \quad 0 \leq k \leq (n-1)$$

Blackman-Harris:

$$w(k+1) = 0.34875 - 0.48829 \cos\left(2\pi \frac{k}{n-1}\right) + 0.48829 \cos\left(4\pi \frac{k}{n-1}\right) - 0.01168 \cos\left(6\pi \frac{k}{n-1}\right) \quad 0 \leq k \leq (n-1)$$

Hamming:

$$w(k+1) = 0.54 - 0.46 \cos\left(2\pi \frac{k}{n-1}\right) \quad 0 \leq k \leq (n-1)$$

Hann (Hanning):

$$w(k+1) = 0.5 \left(1 - \cos\left(2\pi \frac{k}{n-1}\right) \right) \quad 0 \leq k \leq (n-1)$$



$$w(k) = 1 - \left(\frac{k - \frac{n}{2}}{\frac{n}{2}} \right)^2 \quad 0 \leq k \leq (n-1)$$

Welch:

9.2.7 Discrete Fourier Transforms

See [19] for details.

9.2.8 Channel Averaging

See 'Channel Average' in [16].

9.2.9 Sideband Separation

See section 3.7.15.

9.3 Correlator Hardware CAN Commands

Here we list the currently available commands which the CCC can use. The source for these commands comes from sections 8 – 11 in [10].

9.3.1 Common CAN Commands

The following commands are common to all correlator cards directly on the CAMB bus:

Command Name	Description
Node Identification Request	Request node ID
Block Transfer to Absolute Memory Address	Send data to absolute target memory address
Set 48ms Tick Count	Set a specific TE to a given value
Bootstrap Operations	Download new microprocessor code & reboot
Erase Flash Sectors	Erase microprocessor flash memory
Program Data Flash	Program microprocessor data flash sector(s)
Program Code Flash	Program microprocessor code flash sector(s)
Block Read back from Absolute Memory	Read data from absolute target memory address
Read CAN Debug or History Queue	Read CAN debug data or CAN history data
Read Status Blocks	Read TBD status blocks
Set/Get 48ms Tick Count	M&C set or get 48 ms tick count
Get Data Flash Status	Get status information regarding flash memory erase & program operations
Get Bootstrap Status	Get status of bootstrap operations
Get Checksum Status	Get microprocessor code checksum status
Get Code Checksum	Get microprocessor code checksum
Get Code Flash Status	Get microprocessor code flash status
Get Data Checksums	Get microprocessor data checksums
Set/Get CAN History Queue Number of En-	Set/Get size of CAN history queue



ALMA Project
Correlator Subsystem Software Design


Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 77 of 86

Command Name	Description
tries	
Command CAN Debug Capture	Begin capture of CAN commands for debugging

Table 10 - Common CCC CAN Commands

9.3.2 LTA Specific CAN Commands

Command Name	Description
Download Configuration Parameters	Send correlator configuration data
Download All Swap Control Bytes	Parameters used by LTA to reduce the number of errors in data transmittal between correlator and station interface cards.
Download Phased array parameters	Configure phased array mode
Generate New State	Command embedded CPU to generate new FPGA control words for a correlator configuration
Apply New State	Apply new correlator configuration at the next TE
Set Blanking Duration	Set the duration of the blanking interval that occurs once every msec in the correlator chips for multiple LTAs.
Setup Correlator Card Tests	Run microprocessor tests
CI Test Ctrl	Set parameters that control "cable training" tests.
Read CAN Debug or History Queue	Read CAN debug data or CAN history data
Read Status Blocks	Read TBD status blocks
Read Correlator Card Lags	Read raw lags
Read Cable Training Error Counts	Get error count in "cable training" tests
Read Cable Training Alive Status	Get the "alive" status of each bit stream in cables interfacing correlator and station interface cards.
Set/Get My Accumulation Planes	M&C Specifies which 8 correlator planes for an LTA microprocessor
Set/Get My Control Planes	M&C Specifies a pair of correlator planes for an LTA microprocessor
Set/Get Blanking Duration	Set/get the duration of the blanking interval that occurs once every msec in the correlator chips for one LTA.
Get CPLD2 Status	Mask indicating which target cards are in the correlator bin.
Get FPGA Init Status	Mask indicating which target cards succeeded in the INIT phase of an FPGA download.
Get FPGA Done Status	Mask indicating which target cards succeeded in the data transfer phase of an FPGA download.
Get FPGA Flash Storage Status	Mask indicating which FPGA in flash is valid.


	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 78 of 86
--	--	--

Command Name	Description
Get State Status Low	Status resulting from the GENERATE NEW STATE and APPLY NEW STATE commands described above. This monitor covers STATE numbers 0 – 7.
Get State Status High	Status resulting from the GENERATE NEW STATE and APPLY NEW STATE commands described above. This monitor covers STATE numbers 8 – 15.
Set/Get Test Accumulation Number of Ticks	Set/Get number of 16 ms ticks used for test accumulations
Get List of Cards Present	Get list of correlator and interface cards present
Get Cable Training Test Iterations	Get iteration count in “cable training” tests.

Table 11 – LTA CAN Commands

9.3.3 SCC-Specific CAN Commands

Command Name	Description
Download All Fractional Phases	Download the fractional phase setting for every data stream between the station interface cards and the correlator interface cards.
Download BBC	Set BBC for 4 CAIs
Apply BBC	Apply the downloaded BBC at the next TE
Download TFB Tap Weights	Set the TFB tap weights
Select PN Data	Select pseudo number generators for in the station interface cards, station cards, or TFB cards as the source of output data.
Set All Fractional Phases	Set all the fractional phase control bit fields to the specified settings.
Set Filter Card Input	Select the source of input data for either a single TFB card in a bin or for all eight TFB cards in a bin.
Download Delay Model	Set quantized delay model parameters
Read Digitizer Statistics	Read digitizer statistic measurements in TFB cards. Each TFB card measures the duty cycle of a given input and output state each msec
Get List of Cards Present	Get list of station, TFB, and interface cards present in bin
Set/Get My Antennas	Specifies the lowest numbered antenna (out of 4 possible) handled by this SCC.
Get CPLD2 Status	Return status of target cards in station bin
Get FPGA Init Status	Get status of FPGAs and initialization
Get FPGA Done Status	Get status of FPGAs after download
Get FPGA Flash Storage Status	Mask indicating which FPGA personalities were found to be stored with a proper SYNC word in flash memory. We assume a valid personality is present if the SYNC word is correct.
Set One Fractional	Set a single fractional transmission phase control bit field to the

	ALMA Project Correlator Subsystem Software Design	Doc # : COMP-70.40.00.00-001-F-DSN Date: 2009-08-12 Status: Approved Page: 79 of 86
--	--	--

Command Name	Description
Phase	specified setting.

Table 12 - SCC CAN Commands

9.3.4 QCC-Specific CAN Commands

Command Name	Description
Read Warn Counts	Request the warning counts for each type of warning
Get Rack On	Get rack on/off status
Read Monitor Data Blocks	Request all monitor points for a specified card type

Table 13 -QCC CAN Command

9.4 Correlator Configuration

The correlator configuration is part of the ALMA Project Data Model defined in UML. It contains the correlator configuration which generates an IDL file that is used by various subsystems to configure the correlator. This is the current IDL structure follows.

```

/* @(#) $Id: CorrConfig.idl,v 1.9 2008/11/25 16:03:19 ramestic Exp $
*/

#include <acstime.idl>
#include <almaEnumerations_IF.idl>

#pragma prefix "alma"

/** This IDL file defines all the parameters for correlator configuration. It
** is a hierarchical structure of containers:
** 1. Correlator configuraiton:
**    2. 1 - 4 Baseband configurations
**      3. 1 - 8 Spectral window configurations
**
** A note about sidebands.
** There are 3 types of receivers:
** Band      Type
** 1,2       SSB w/ USB & LSB
** 3 - 8     2 sidebands, with either sideband suppression removing either USB
**           or LSB or sideband separation which routes the USB to one baseband
**           pair & the LSB to another baseband pair.
** 9,10      Double sidebands, with 90d phase switching to do sideband separation
**           or no separation & the 2 sidebands are overlaid.
**
** The relevant parameters to define this are:
** NetSideBand defines if the spectral window is USB or LSB
** SideBandSeparationMode defines 90d phase switching or sideband rejection
** ReceiverSideband defines which type of receiver is in use: SSB, 2SB or DSB
**
** The order of frequency channels are implied that for the USB, the lowest sky
** frequency is mapped to the lowest IF (or baseband) frequency and for the LSB,
** the lowest sky frequency is mapped to the highest IF frequency, i.e., the
** channel order is 'reversed'.
**
** A note about frequencies.
** The correlator knows only about baseband frequency. All frequencies in this IDL
** are <b>baseband</b> frequencies, i.e., 2 - 4 GHz.
*/

module Correlator

```



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 80 of 86

```
{  
    /// Sequence of stokes parameters to define single, dual, or full polarization  
    typedef sequence <StokesParameterMod::StokesParameter> StokesParameterSeq;  
    /// Sequence of APC to allow: (CORRECTED &/or UNCORRECTED) or MIXED  
    typedef sequence <AtmPhaseCorrectionMod::AtmPhaseCorrection> AtmPhaseCorrectionSeq;  
  
    /** The maximum number of spectral windows per baseband.  
    ** For the BLC there can only be 4 independent spectral windows per baseband.  
    ** We allow for 8 because there can be 4 pairs of spectral windows with DSB receivers.  
    **/  
    const short MAXIMUM_NUMBER_SPECTRAL_WINDOWS = 64;  
  
    /// The maximum number of channel average regions per spectral window  
    const short MAXIMUM_NUMBER_CHANNEL_AVERAGE_REGIONS = 10;  
  
    /// The maximum number of basebands supported in a configuration  
    const short MAXIMUM_NUMBER_BASEBANDS = 4;  
  
    /** Nutator or frequency switching definition. This structure defines the  
    ** type of switching, the number of switching positions, the dwell time,  
    ** i.e., the time in a given position and the 'dead time', i.e., the time  
    ** that the data should be ignored due to mechanical delays, e.g., moving time  
    ** and settling time. Some caveats:  
    ** -# Frequency switching usually has no dead time.  
    ** -# dwellTime + deadTime must be multiple of 48ms  
    ** -# The size of the dwellTime & deadTime sequences must equal numberOfPositions.  
    ** -# If SwitchingType == NO_SWITCHING, then numberOfPositions = 0.  
    ** -# Frequency switching is per baseband while nutator switching applies to all  
    ** basebands.  
    **/  
    struct BinSwitching_t  
    {  
        /// NO_SWITCHING | LOAD_SWITCHING | POSITION_SWITCHING | PHASE_SWITCHING |  
        /// FREQUENCY_SWITCHING | NUTATOR_SWITCHING | CHOPPER_WHEEL  
        SwitchingModeMod::SwitchingMode SwitchingType;  
        /// 2 | 3 | 4  
        long numberOfPositions;  
        /// The duration at a given position  
        ACS::TimeIntervalSeq dwellTime;  
        /// The duration of data to ignore for each position  
        ACS::TimeIntervalSeq deadTime;  
    };  
  
    /** This structure defines the phase switching configuration 180 degrees  
    ** which is performed in the ACA correlator.  
    **/  
    struct ACAPhaseSwitchingConfigurations  
    {  
        /** If doD180modulation = TRUE then '180-degree anti-demodulation' is  
        ** performed in the DTS receiver, else no '180-degree anti-demodulation'  
        ** is performed.  
        /** The '180-degree anti-demodulation' cancels the 180 degrees demodulation  
        ** in the DTS transmitter to detect potential spurious signals mixed at the  
        ** digitizer. The '180 degrees anti-demodulation' is also used to reduce the  
        ** bias generated in the FFT calculation.  
        **/  
        boolean doD180modulation;  
  
        /** If doD180demodulation = TRUE then '180-degree demodulation' is  
        ** performed in the CIP module, else, no '180-degree demodulation'  
        ** is performed.  
        /** The '180-degree demodulation' cancels the '180 degrees anti-demodulation'  
        ** in the DTS receiver if doD180modulation = TRUE. Or the '180-degree  
        ** demodulation' cancels the 180 degrees modulation performed in the LO, if  
        ** neither the 180 degrees demodulation in the DTS transmitter nor '180-
```




ALMA Project
Correlator Subsystem Software Design

Doc #: COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 81 of 86

```
** degree anti-demodulation' in the DTS receiver are performed.
** The doD180demodulation must be TRUE when doD180modulation = TRUE,
** because '180-degree anti-demodulation' must be canceled if the '180-
** degree anti-demodulation' is performed.
*/
boolean doD180demodulation;

/** The minimum duration of an element of the 180-degree modulation.
** This value is also used for the 180-degree demodulation.
** d180Duration must be 250us, 500us, 1ms, 2ms, 4ms or 8ms, and this value
** must be the same as the minimum 180 degree phase switching duration
** both in the LO and the DTS transmitter. So this value must be coordinated
** with CONTROL.
*/
ACS::TimeInterval d180Duration;
};

/** \struct Defines a region for channel average data. Note that these
** parameters are based on effective channels. That is, startChannel
** must be a zero based index within the effective range of spectral
** channels and the number of channels is interpreted based on the
** spectral resolution (band-width per channel) before any spectral
** averaging (spectralAveragingFactor) has been applied.
*/
struct ChannelAverageRegion
{
    long startChannel;    ///< the first channel of this channel average region,
    long numberChannels;  ///< the number of channels to average together
};
typedef sequence<ChannelAverageRegion,MAXIMUM_NUMBER_CHANNEL_AVERAGE_REGIONS> ChannelAverageRegion-
Seq;

/** \struct Defines a spectral window for the Tunable filter.
*/
struct SpectralWindow
{
    /** The center frequency of each spectral window in MHz <b>relative to the
    ** baseband</b>, i.e., 2 - 4 GHz. The step size of the band is 2 GHz/8192
    ** = 244.141 KHz.
    */
    double centerFrequencyMHz;

    /** the effective bandwidth of each spectral window in MHz
    */
    double effectiveBandwidthMHz;

    /** the effective number of channels of the spectral window.
    */
    long effectiveNumberOfChannels;

    /** Specify the number of spectral channels to average together for
    ** each spectral window. This factor is the number of adjacent channels to
    ** average together and must be a power of 2, e.g., 1, 2, 4, 8, etc.
    ** The spectral averaging factor can be used to reduce the amount of
    ** spectral data sent to the Archive.
    */
    long spectralAveragingFactor;

    /** The sideband defines which sideband to be output: USB or LSB.
    ** Note that if this spectral window is LSB, then its frequency (or channel)
    ** order is reversed from the sky frequency, i.e., lowest sky frequency is
    ** highest baseband frequency.
    ** The SideBandMode should be common to all spectral windows in a baseband
    ** except for the DSB of ReceiverSideBand.
    */
}
```



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 82 of 86

```
NetSidebandMod::NetSideband sideBand;

/** For double sideband receivers (DSB), spectral windows are defined as
** pair due to the nature of sideband separation. If SideBandSeparationMode
** is PHASE_SWITCHING_SEPARATION or FREQUENCY_OFFSET_SEPARATION then we
** must pay attention to which spectral windows comprise a pair. The OT
** has 2 options, either associate a pair of spectral windows,
** e.g., 1 & 2, or suppress one of the spectral windows in the pair.
** Currently it's TBD how this suppression is specified, either by not
** including the 'other' spectral window or by setting a flag.
** TBD: How to convey the allocation of sub-bands to USB and LSB in
** DSB receivers.
** If SideBandSeparationMode is not PHASE_SWITCHING_SEPARATION or
** FREQUENCY_OFFSET_SEPARATION, then this value is 0 (ignored).
** Note: FREQUENCY_OFFSET_SEPARATION applies in this context only to
** the ALMA-B correlator, which can perform this separation by sharing
** TFB sub-bands between upper and lower side-bands.
**/
long associatedSpectralWindowNumberInPair;

/** For DSB receivers, the user could suppress an image spectral window of a
** given pair. This would allow the user to improve data rate by discarding
** unwanted data. This flag is always TRUE for SSB & 2SB receivers and can
** be TRUE or FALSE only for DSB receivers.
**/
boolean useThisSpectralWindow;

/** An enumerated value representing the data smoothing function to use.
**/
WindowFunctionMod::WindowFunction windowFunction;

/** The observer can select up to 10 sets of contiguous channels (regions)
** which produces a channel average result for each channel average region
** for a spectral window.
** Each band is selected by the following parameters:
** -# startChannel The starting (0-based) channel used to calculate the
** spectral channel average.
** startChannel < effective number of channels
** -# numberChannels The region width as the number of spectral channels
** used to calculate the spectral channel average for the region.
** startChannel + numberChannels < effective number of channels
**/
ChannelAverageRegionSeq channelAverageRegions;

/** <ACA correlator specific>
** The ACA correlator outputs the spectral results with a frequency
** channel profile compatible with that of the baseline correlator
** when frqChProfReproduction = TRUE. The ACA correlator outputs the
** spectral results with the FX correlator specific frequency channel
** profile when frqChProfReproduction = FALSE.
**/
boolean frqChProfReproduction;

/** <BL correlator specific>
** Options: 2x2, 3x3, 4x4
**/
CorrelationBitMod::CorrelationBit correlationBits;

/** <BL correlator specific>
** Nyquist sampling/oversampling True means do Nyquist oversampling, False
** means non-oversampling.
**/
boolean correlationNyquistOversampling;

/** <BL correlator specific>
```



ALMA Project

Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 83 of 86

```
** Options: XX | YY (single pol'n) or XX,YY, (dual pol'n) or XX, YY, XY, YX (full pol'n)
** The ACA correlator always outputs all 4 polarization products. The
** ACA correlator will share the polnProductsSeq if the polnProductsSeq is
** common to all spectral windows in a baseband. For the ACA correlator, the
** polarization products can not be changed by the spectral windows.
** AUTO_ONLY:
** "XX"          for autoData: {XX}
** "YY"          for autoData: {YY}
** "XX YY"       for autoData: {XX,YY}
** "XX XY YX YY" for autoData: {XX,XY,YY}
**
** CROSS_AND_AUTO:
** "XX"          for autoData: {XX}          if for crossData: {XX}
** "YY"          for autoData: {YY}          if for crossData: {YY}
** "XX YY"       for autoData: {XX,YY}       if for crossData: {XX,YY}
** "XX XY YX YY" for autoData: {XX,XY,YY}    if for crossData: {XX,XY,YX,YY}
*/
Correlator::StokesParameterSeq polnProductsSeq;

/** <BL correlator specific>
** Boolean selection to perform quantization ('VanVleck') correction on
** the lag data. Normally this is true, but for diagnostic purposes, we
** may wish to view the uncorrected lags.
** <ACA correlator specific comments>
** The ACA correlator does not use this parameter, because the ACA-CDP always
** does the quantization correction on the spectral results.
*/
boolean quantizationCorrection;
};

/** up to 8 spectral windows per baseband either as 8 separately tunable spectral
** windows for SSB & 2SB receivers or 4 pairs of separately tunable spectral
** windows for DSB receivers.
*/
typedef sequence<SpectralWindow,MAXIMUM_NUMBER_SPECTRAL_WINDOWS> SpectralWindowSeq;

/** \anchor BaseBandConfig. Each baseband can be configured independently
** (except for integration & sub-integration durations) with up to four
** baseband configurations within a correlator configuration.
*/
struct BaseBandConfig
{
    /** The baseband name "BB_1" to "BB_4" to be configured. The TMCDB must
    ** define the mapping between baseband names and correlator quadrants.
    **/
    BasebandNameMod::BasebandName basebandName;

    /** The Correlator Accumulation Mode. If CAM is ALMA_FAST, i.e., 1-ms dumps,
    ** only auto-correlation products are sent to the CDP. If CAM = ALMA_NORMAL,
    ** then both auto- and cross-correlation products are sent to the
    ** CDP w/ a minimum dump time of 16ms.
    **/
    AccumModeMod::AccumMode CAM;

    /** Allows the user to select both auto & cross correlation products or
    ** auto only or cross only when CAM = ALMA_NORMAL. When CAM = ALMA_FAST,
    ** only the auto products are available.
    */
    CorrelationModeMod::CorrelationMode dataProducts;

    /** Defines sideband separation mode as 90d phase switching (PHASE_SWITCHING_SEPARATION),
    ** TFB offsetting (FREQUENCY_OFFSET_REJECTION) or NONE
    ** plus mechanisms to suppress a sideband (PHASE_SWITCH_REJECTION).
    ** FREQUENCY_OFFSET_REJECTION and FREQUENCY_OFFSET_SEPARATION are achieved through
    ** LO offsetting, PHASE_SWITCHING_SEPARATION and PHASE_SWITCH_REJECTION are performed

```



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 84 of 86

** with bin switching.
**
** From Rodrigo Amestica (as viewed from P. Napier's note regarding
** the Use of LO Offsetting for Spurious Signal Suppression and Sideband
** Rejection) sidebands and TFB frequency offsets and ACA frequency
** shifts we have the following table (more details available here
** <http://almasw.hq.eso.org/almasw/bin/view/CORR/SidebandSeparationAndLOoffsetting>):

<pre>

| Mode# | Receiver Type | SB | 180d Walsh | 90d Walsh | LO1 | LO2 | TFB Offset | ACA Offset | SideBandSeparationMode |
|-------|---------------|-------|------------|-----------|------|-----|------------|------------|-----------------------------|
| [1] | 2SB | upper | ON | OFF | ND | -ND | 0 | 0 | NONE |
| [2] | 2SB | lower | ON | OFF | ND | ND | 0 | 0 | NONE |
| [3] | 2SB | upper | OFF | OFF | 2*ND | -ND | -ND | -AC | FREQUENCY_OFFSET_REJECTION |
| [4] | 2SB | lower | OFF | OFF | 2*ND | ND | ND | AC | FREQUENCY_OFFSET_REJECTION |
| [5] | DSB | both | ON | ON | 0 | 0 | 0 | 0 | PHASE_SWITCHING_SEPARATION |
| [6] | DSB | upper | OFF | OFF | 2*ND | -ND | -ND | -AC | FREQUENCY_OFFSET_REJECTION |
| [7] | DSB | lower | OFF | OFF | 2*ND | -ND | 3*ND | 3*AC | FREQUENCY_OFFSET_REJECTION |
| [8] | DSB | both | | | | | | | |
| | | upper | OFF | OFF | 2*ND | -ND | -ND | N/A | FREQUENCY_OFFSET_SEPARATION |
| | | lower | OFF | OFF | 2*ND | -ND | 3*ND | N/A | FREQUENCY_OFFSET_SEPARATION |

</pre>

Note: the difference between modes {1,2} and {3,4} is the introduction of an LO2 offset, which basically replace the 180d Walsh function cycling for removal of spurious signals along the data path. As explained in Peter's document a 180d Walsh function mechanism is not suitable for 'high' resolution modes, therefore, modes {3,4} are introduce to cope with those observing modes. The importance of specifying FREQUENCY_OFFSET_REJECTION for modes {3,4} is that this is the only way we have to tell the BL correlator that TFB mixer LO frequencies require offsetting from the their default frequencies. Modes {1,2} are meant for low frequency resolution modes and, therefore, use Walsh functions for spurious rejection and no offset is involved.

And from M. Watanabe, we have a similar view:

<pre>

| LO (1st and/or 2nd) | | TFB | ACA | quadrant | Sideband of the |
|---------------------|-----------|-----------|-----------|-----------|---------------------|
| | | / | / | /ACA | CDP spectral window |
| | | | | | |
| | | | | | |
| | | | | | |
| 90d Walsh | LO offset | LO offset | LO offset | 90d Walsh | 90d Walsh |
| ON | OFF | - | - | ON | OFF |
| | | | | | |
| ON | OFF | - | - | OFF | ON |
| | | | | | |
| ON | OFF | - | - | OFF | OFF |
| | | | | | |
| OFF | ON | ON | N/A | - | - |
| | | | | | |
| OFF | ON | N/A | ON | - | - |
| | | | | | |

</pre>

*/
SidebandProcessingModeMod::SidebandProcessingMode sideBandSeparationMode;

/** 8 spectral windows can be defined per baseband. In the case of SSB or 2SB
** receivers, there can be up to 4 independently tuned spectral windows. For
** DSB receivers, there can be up to 4 independently tuned pairs of spectral
** windows.



ALMA Project
Correlator Subsystem Software Design

Doc # : COMP-70.40.00.00-001-F-DSN
Date: 2009-08-12 Status: Approved
Page: 85 of 86

```
*/
SpectralWindowSeq spectralWindows;

/** Nutator switching is common to all basebands, but frequency switching
** occurs at the 2-nd LO so it can be different for each baseband.
**
*/
BinSwitching_t binSwitchingMode;

/** <ACA correlator specific>
** The polarizationMode defines which and how polarization products
** to be processed and output.
**
*/
ACAPolarizationMod::ACAPolarization polarizationMode;

/** <ACA correlator specific>
** The center frequency of the fringe rotation.
** This value is used in the residual delay compensation to calculate
** the phase error.
**
*/
double centerFreqOfResidualDelayMHz;
};

typedef sequence<BaseBandConfig,MAXIMUM_NUMBER_BASEBANDS> BaseBandConfigSeq;

/** \anchor CorrelatorConfiguration. 2 The correlator configuration container. */
struct CorrelatorConfiguration
{
    /** <BL Correlator specific> The dump interval must be a multiple of 16ms for
    ** cross & auto mode, or 1ms for auto-only mode.
    **
    */
    ACS::TimeInterval dumpDuration;

    /** the integration duration must be a multiple of the dump durations
    **
    */
    ACS::TimeInterval integrationDuration;

    /** The duration of the channel averaging is between 0.5 and 1.024
    ** seconds, such that the smallest number of channel average durations
    ** fits into the spectral integration duration. The units of ChannelAverageDuration
    ** is ACS::TimeInterval, i.e., 100ns. If the integration duration is
    ** less than 0.5 seconds, then no channel averages are calculated.
    **
    */
    ACS::TimeInterval channelAverageDuration;


    /** the subscan duration must be a multiple of of integration duration.
    **
    */
    ACS::TimeInterval subScanDuration;

    /** Type of receiver used for all basebands */
    ReceiverSidebandMod::ReceiverSideband receiverType;

    /** Up to 4 basebands can be configured in a single correlator configuration */
    BaseBandConfigSeq baseBands;

    /** This flag determines which APC results are published. Sequence allows for
    ** CORRECTED or UNCORRECTED or CORRECTED + UNCORRECTED or MIXED
    **
    */
    AtmPhaseCorrectionSeq APCDataSets;

    /** <ACA correlator specific>
    ** This parameter is common among all of four sets of the ACA correlator.
    **
    */
    ACAPhaseSwitchingConfigurations ACAPhaseSwConfig;
};
```

	<p>ALMA Project</p> <p>Correlator Subsystem Software Design</p>	<p>Doc # : COMP-70.40.00.00-001-F-DSN</p> <p>Date: 2009-08-12 Status: Approved</p> <p>Page: 86 of 86</p>
--	---	---

};