EVLA Correlator Monitor and Control System, Test Software, and Backend Software Requirements and Design Concepts

NRC-EVLA Memo# 015

Brent Carlson, January 23, 2002

ABSTRACT

Much work has been done on studying, developing, defining, and documenting the EVLA WIDAR correlator hardware architecture. However, as yet, there has not been much work on defining and documenting correlator monitor and control software and correlator "backend" output data handling software. The purpose of this document is to start this process by defining some straw-man requirements and providing design concepts that aim to address the important aspects of the requirements. This should be considered a "genesis" document to be used for discussion and further refinement and formalization into more detailed specifications and, ultimately, implementation.

1 Introduction

Some initial design concepts for correlator software have been developed. This includes concepts for the on-line Correlator Monitor and Control System (CMCS), concepts for the layer of CMCS test software used to test the CMCS, and concepts for the so-called correlator "backend". The CMCS is responsible for correlator configuration and real-time status monitoring. It is part of the overall EVLA monitor and control (M&C) system, but provides a level of abstraction to the rest of the EVLA system via the Virtual Correlator Interface (VCI). The VCI allows the EVLA M&C system to control and monitor the correlator, but without having to know about the intimate details of the correlator. The CMCS test software resides above the VCI and allows testing of the correlator through the VCI during development and before other EVLA M&C software is available. The test software is not throw-away software and may be used after the correlator is operational for technician and/or engineer access and troubleshooting.

The correlator backend is the layer of hardware and software between the correlator and the "end-to-end" (e2e) System [1]. The e2e System[1] is responsible for calibration, archiving, and image processing and is not part of correlator data processing. The correlator backend processes the raw data from the correlator and "throttles" the data output rate before the data is handed over to the e2e System. Processing that occurs in the backend includes any *requested* data corrections, windowing, Fast Fourier Transforms (FFTs), interference excision, and possibly visibility interference cancellation. The "quanta" that is output from the correlator backend is the minimum possible sub-band cross-power spectrum that is produced by the correlator. That is, any "stitching" operations that combine spectra from different sub-bands are not performed—

[1] The e2e System is responsible for MUCH more, but what is described is the component of the e2e System that directly uses the correlator output data products.

this operation is left to the e2e System since some operations that could be performed cannot be reversed and therefore violate archive data requirements[2].

A simplified block diagram of correlator software and interfaces described above is shown in Figure 1-1. This figure shows the major interfaces and software components of the correlator and surrounding infrastructure.

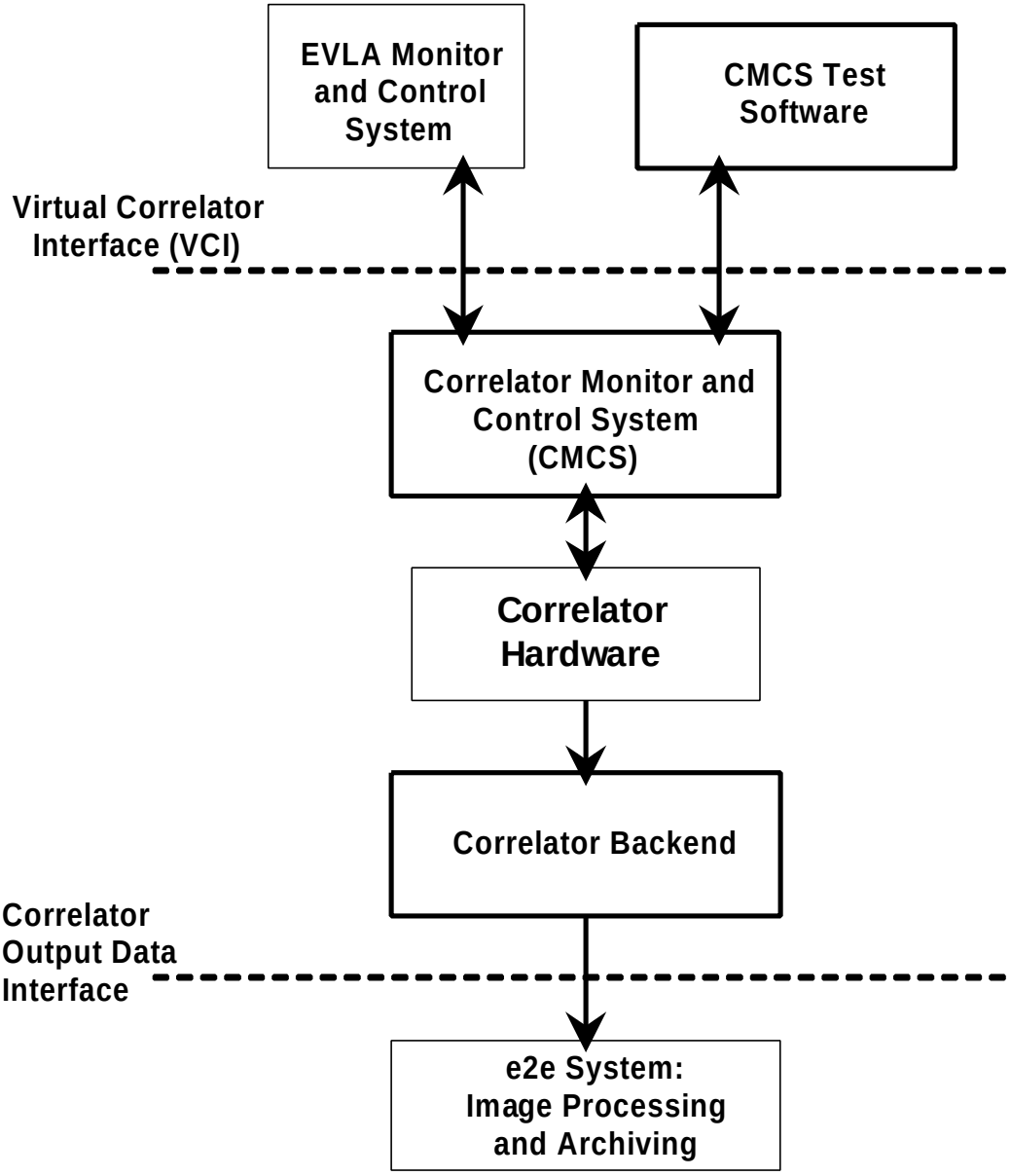

Figure 1-1 Simplified correlator interface and software layer diagram. The blocks outlined in bold are described in this document.

[2] i.e. to always be able to recover the raw, uncorrected data.

This document presents CMCS, CMCS test software, and correlator backend software requirements and design concepts. It is a "genesis" document for further discussion, specification, and development. This document includes the results of several discussions that were held in Socorro on December 6 and 7, 2001.

2 Requirements

This section outlines the requirements for the CMCS, CMCS test software, and correlator backend hardware and software. These requirements are developed from the correlator architecture defined in [2] and the data output topology described in [3].

2.1 CMCS and VCI Requirements

The following are the requirements of the CMCS, VCI, and the CMCS hardware:

1. Connection to the CMCS will be via a 100 Mbit/sec Ethernet to a **Main Correlator Control Computer** (**MCCC**), and via a separate 10/100 Mbit Ethernet to a separate **Correlator Power Control Computer** (**CPCC**).

2. Communication to the CMCS MCCC will be via RPC (Remote Procedure Calls) [4], using TCP/IP as the underlying protocol. CMCS services will be provided to the external world with a number of RPC servers. [Caveat: This document will specify the use of RPC using a "client-server" model for communications. However, these are subject to revision if a better mechanism is proposed.] Note: As a natural consequence of RPC, all data passing across the VCI will be XDR-encoded and is thus platform independent.

3. The MCCC will contain its own disk and will run the Linux operating system. All run-time libraries, binaries etc. will reside on this disk. It shall be possible to rlogin to the MCCC via the Ethernet interface to perform system administration functions.

4. The MCCC will be a high-reliability PC-type platform. This probably means that it will be a Compact PCI board in a rack-mount chassis. It may be worth considering using a redundant (hot-standby) processor for increased reliability.

5. It must be possible for monitor and control clients above the VCI to obtain real-time, station-based data such as power measurements and state counts from the Station Boards via the MCCC. Any acquisition of baseline data (visibilities) will be through an e2e System interface, and not through the VCI.

6. The CPCC will use similar communication mechanisms and will have similar reliability as the MCCC, but software residing on the CPCC is relatively simple and robust. (i.e. power control is on the separate CPCC so that control remains even if the MCCC crashes.)

7. The CPCC will contain special hardware interface boards to monitor and control the hundreds of DC-DC power supplies and cooling fans in the system. The CPCC should monitor and control the 48 VDC power plant (depending on power plant capabilities), and thus will have to be on its own high reliability UPS. Additional requirements for fail-safe "power dump" control of the 48 VDC plant will be defined in a separate correlator room requirements document.

8. The CPCC may monitor and control the MCCC, depending on suitability. At the very least, the CPCC should be able to reboot the MCCC.

9. The CPCC must contain a watchdog timer so that it can self-reboot in the event that it crashes. This self-reboot must in no way affect the current operation of the correlator.

10. Each correlator board (Station Board, Baseline Board, Phasing Board, TIMECODE Generator Box) will contain a mezzanine card with an embedded processor on it. This mezzanine card will be a PC/104+ compliant CPU module with a 10/100 Mbit/sec Ethernet interface, a serial RS232 interface, and the 16-bit ISA bus and 32-bit PCI bus for communication with devices on the carrier board. The CPU module will have interrupt capability, and synchronous interrupts will occur every 10 milliseconds. This embedded processor mezzanine card will be referred to as the **CMIB** (**Correlator Monitor Interface Board**).

11. The CMIB will contain a RTOS (real-time operating system). This could be VxWorks or real-time Linux depending on suitability. It is believed that a VxWorks target license for a PC/104+ module is very inexpensive, and it is a known RTOS that will do the job. Whether real-time Linux is suitable for this purpose requires further detailed investigation and testing.

12. DRAM and Flash memory requirements on the CMIB are TBD, but normally these boards have the capacity for up to ~64 Mbytes of DRAM and up to a few hundred megabytes of Flash "Disk on a chip" memory. The Flash memory looks like a disk to the operating system and user.

13. The CMIB will read a 16-bit identifier from the carrier board[3] that has its source from the backplane that the carrier card plugs into. This 16-bit identifier will be obtained during boot-up to form part of the IP address for the CMIB. Presumably, the 16-bit identifier will form the least-significant 16 bits of the IP address. This way carrier boards can be swapped out without having to change configurations in the MCCC.

14. The CMIB will boot from the on-board Flash memory disk, rather than over the network. Run-time binaries will reside on the Flash memory disk and, if the

[3] i.e. Station Board, Baseline Board, Phasing Board, or TGB.

CMIB OS is Linux, will be started using standard multi-processing facilities[4]. FPGA "personalities" will probably reside on the Flash memory disk, but may reside on the MCCC if there are excessive cost and memory requirements to do so.

15. Conceptually there may be 4 flavours of CMIBs in the correlator, one for each type of carrier board it plugs into. However, CMIBs are all the same type of hardware, only the software (i.e. run-time binaries) in the Flash memory disk is different. Depending on memory disk requirements and capacity, all binaries for all types of carrier boards could reside on every CMIB's memory disk. If this is the case, then each CMIB is identical—only the software that gets invoked is different.

16. It must be possible to download CMIB software upgrades and FPGA personality upgrades to the CMIB over the Ethernet (i.e. without having to physically remove the carrier board or the CMIB). It is desirable to perform this upgrade while the CMIB is on-line. Rebooting the CMIB/carrier board or killing and restarting a process will cause the new software to take effect.

17. The CMIB must have a remote reboot capability. A reboot request from the MCCC will reboot the CMIB processor and the carrier board electronics.

18. The CMIB must have a local reboot capability that is governed by a watchdog timer. This is so that if the CMIB crashes and the MCCC can no longer communicate with it, it will reboot itself and eventually reconnect to the MCCC to resume normal operation. "Resuming normal operation" means that the MCCC must start any programs and provide the CMIB with most recent configuration data so that it can resume processing.

19. Any error messages or debug messages that are generated by the CMIB must be transferred for handling to the MCCC. That is, it should never be necessary to rlogin to the CMIB to see low-level debug or error messages.

20. It shall be possible to sustain a crash or reboot of the MCCC without having to disrupt the operation or reboot any CMIBs. This could result in a momentary disruption of correlator operation (e.g. loss of delay and phase tracking), but once the MCCC is rebooted, it re-establishes connections to all of the CMIBs and continues processing.

21. It is a requirement to electrically isolate communications between the correlator hardware and any external hardware (COTS output/control computers). (This is because the correlator will run on 48 VDC, and front and backend computers will be COTS equipment running on 110 VAC. While these systems will have a

[4] i.e. if Linux is used on the CMIB, the CMIB will appear just as any other Unix-like computer on the network. All code developed on desk-top Linux machines can simply be saved on the CMIB Flash-disk, and started/run over the network.

common ground, it is necessary to eliminate any other coupling between them for noise reasons.) 100 Mbit Ethernet is transformer coupled and it is expected that this will provide the level of isolation required, negating the need for additional fiber-optic isolation.

2.2 CMCS Test Software Requirements

The following are the requirements for the CMCS test software that resides outside of the CMCS and on top of the VCI. This software is used for testing the CMCS and VCI during development, but must be available after the system is operational for troubleshooting. This software essentially provides a manual method for controlling and monitoring the correlator. This software will reside on a computer platform independent of the MCCC and the CPCC.

1. A user interface that allows parameters for all RPC calls that cross the VCI to be generated and invoked (i.e. actually cause an RPC call) must be available. This should be a combination of a graphical user interface and a command line interface. For example, the graphical user interface (GUI) could allow parameters to be "ticked off", stations to be selected etc. This user interface must be able to be started anywhere on the Internet, connect to the CMCS via the VCI (by qualified and password-protected personnel), and start generating traffic across the VCI.

2. A window in the test software must flexibly allow the display of all Ethernet/RPC traffic crossing the VCI. It must be possible to set various display levels that range from displaying all the details of the RPC traffic (including all parameters associated with the function calls and returns), to displaying basic information (such as just the name of the RPC function). All RPC traffic must be time-stamped using wall clock time.

3. The test software user interface must allow the invocation of low-level hardware device driver function calls executed by the CMIBs using RPC as a wrapper. Any error or status messages returned by the calls must be displayed in the test software user interface.

4. Additional command-line functions that are not direct RPC calls or device driver calls should be provided as appropriate. This could include things like checking disk space, memory, or other system status functions that would not normally be invoked or requested by the EVLA M&C system.

5. It must be possible to rlogin to any CMIB or other correlator control or backend computer. This function need not formally be part of the test software interface, but the test software could provide an easy-to-use front-end for performing this function.

6. It must be possible to invoke the CMCS test software while the correlator is otherwise under control of the EVLA monitor and control system. That is, once

the test software is "connected" to the CMCS via the VCI, both it and the EVLA monitor and control system will be able to simultaneously issue commands and receive unsolicited messages that cross the VCI. It should be noted that this simultaneous access could have disruptive effects on normal observing and thus the test software should only be used and accessible by qualified personnel.

2.3 Correlator Backend Requirements

This section defines the requirements of the correlator backend software and hardware. The backend is where the primary data products of the correlator (cross-correlation results) are processed.

1. The Baseline Boards will produce UDP/IP packets on one (or more) fiber-isolated, Gigabit Ethernet outputs. Each packet contains one frame or "quanta" of lag data (a preliminary definition of which is in [2]) though generally, lags from multiple frames may have to concatenated to produce one complete set of lags. These packets must be routed to backend processors using Gigabit Ethernet switches. The backend network configuration and the destination IP addresses used by the Baseline Boards will be such that all data needed to perform any FFT (to convert lag-domain data to the frequency domain) shows up on a single processor. Thus, it is not necessary for the backend to use distributed FFT algorithms or provide a high bandwidth network fabric between processors. However, visibility data from different baselines could be produced/processed by different computers in the backend and final assembly of all visibility data will be performed by the downstream e2e System.

2. The backend processors will perform the following processing functions on the data, but only if configured to do so.

   - Coarse quantization (Van Vleck) corrections. These corrections require state count and/or quantizer power measurement data from the Station Boards. These corrections must be performed on lag-domain data. This includes corrections for the requantizer and the initial 3 or 8-bit quantizer.

   - Windowing. Lag-domain data will normally have to be windowed before the FFT, especially in the presence of strong narrowband RFI (N.B. Alternatively, it may be necessary or even desirable to window data after the FFT by performing a convolution function). The list of window functions and parameters that can be selected is TBD.

   - FFT of lag domain data to the frequency domain. Normally, negative frequencies are discarded after this operation, but stitching of sub-bands in the e2e System may require some negative frequency spectral channels and so all data points should be retained. Further detailed investigation is required.

- Real-time RFI excision. In this function, spectral channels with narrowband signals in them are set to zero before being accumulated into a longer-term accumulation buffer. The software must keep track of the number of samples/dumps integrated in each spectral channel and provide this information with the data when it travels to the e2e System. This method is used to minimize the effects of intermittent or burst-like RFI on the final data and is one of the performance-driving factors of the backend. The integration of RFI-excised data for longer periods of time is what throttles the output of the correlator to a rate manageable by the e2e System.

- Real-time RFI *cancellation*. This is the process [5] of using one antenna as an interference detection antenna (i.e. that is sensitive to the interference and not the radio source), and correlating it with all other antennas. RFI in the radio astronomy cross-power spectrum is cancelled using the RFI-only sensitive correlations. Interference-cancelled cross-power spectra are then integrated for longer periods of time before producing output data to the e2e System. Suitability and integration of this capability with the EVLA requires further study. It may be possible to provide this capability without having to build extra hardware by using a VLA antenna (with appropriate front-end attenuation) as an interference detection antenna. In this case, the astronomer would have to "give up" an antenna in the array if RFI cancellation for a specific interference source is required. A better method that would not require the use of a VLA antenna, would be to use a dedicated RFI detection antenna, although this implies more cost and infrastructure overhead. Perhaps a low-cost 6.1 m ATA antenna, with its broadband cooled receiver would be appropriate.

3. The backend could be programmed to perform no functions on the data. In this case the output to the e2e System will be raw lag data (but with data valid count normalization performed).

4. All other data corrections are performed in the e2e System. This includes calculated bandshape corrections, stitching, sub-band total power corrections etc. These additional corrections require state counts, power measurements, filter parameters, frequency shift parameters, and windowing parameters that originate in the correlator hardware and CMCS, but are funneled to the e2e System via the correlator backend. That is, the output of the correlator backend produces all of the data (produced or analyzed by the correlator) needed by the e2e System for archiving and further processing.

5. Spectral data output to the backend will be complex and in single precision format in order to minimize data rates but still provide adequate dynamic range.

6. When the system first becomes operational, the backend must have *sustained* performance on all required functions so that *all* data from the correlator can be dumped every 100 msec. The backend must have *burst* performance on all required functions so that all data from the correlator can be dumped every 20

msec. The duration of the burst that can sustain the highest rate is dependent on the size of the disks in each backend computer, but a burst duration of 5 minutes seems to be well within reach. Smaller dump times must be possible if proportionately less data is dumped.

7. When the system first becomes operational, the backend must have a total sustained output data rate to the e2e System of 25 Mvis/sec (roughly 200 Mbytes/sec when negative frequencies are retained).

8. The output format of the data from the correlator backend is TBD. It could be in the form of FITS fragments, or the "AIPS++ measurement set".

9. It is a requirement to easily port backend software to new, more powerful computers as technology advances. This, and the need to maximize performance and minimize cost, requires that the computers are generic COTS machines running a generic operating system. This probably means that the computers will be vanilla PC boxes running the Linux operating system and arranged into a Beowulf cluster.

10. The backend must be self-healing and tolerate a computer crash or failure with only momentary loss of data. This requires some interaction with the CMCS so that Baseline Boards can be programmed to re-direct IP packets to hot-standby computers.

11. Backend processing at full required speed must be available 99.9% of the time that the correlator is generating data.

12. It must be possible to remotely control the power of each backend computer via the CPCC. Since these are COTS boxes presumably running off AC power, this probably means that a signal from the CPCC will control AC power for each computer.

13. It must be possible to remotely control the power of each backend Gigabit Ethernet switch via the CPCC as described above.

14. A failure, crash, reboot, or power cycle of any backend computer must not affect the processing of any other backend computer. This is a natural consequence of the self-healing requirement.

This document does not specify or have any specific requirements for the administration of the backend computer network. This is a separate issue that must be given careful consideration in order for the system to be robust and easy to maintain.

3 Design Concepts

This section contains a number of design concepts that can meet the requirements outlined in section 2. This includes network connectivity concepts as well as data flow diagrams and simplified task descriptions for all processing functions.

3.1 Network Architecture

Figure 3-1 is a network architecture diagram of the correlator. All computing elements defined in the previous section are shown in this diagram. The MCCC is the main entry point for correlator monitor and control. For reliability, the CPCC is a separate box that handles power monitor and control, but conceptually it is accessed through the VCI as well. The MCCC performs top-level monitor and control functions for all of the downstream modules. It communicates with the embedded CMIBs via 100 Mbps Ethernet and the CMIBs are where "hard real-time" control functions for the Station, Baseline, and Phasing Boards are performed. The MCCC also performs high-level control functions for the TIMECODE Generator Box via its CMIB. The Baseline Boards package lag-domain correlation coefficients into UDP/IP packets that are switched to Backend Computers with COTS Gigabit Ethernet switches. The switches are arranged so that data from Baseline Boards that process the same baselines (but different sub-bands) go into the same switch. Thus, there is no need for Backend Computers to communicate with each other via an additional wideband network. The Backend Computers take the data, and data acquired from Station Boards, to produce output data that goes to the e2e System. The monitor and control network connections to the Backend Computers are for administration purposes and to allow the Backend Computers to acquire data from the Station Board CMIBs (possibly via the MCCC).

3.2 Data Flow Diagrams and Task Descriptions

This section contains data flow diagrams and task descriptions that describe operations within the context of the network architecture defined in Figure 3-1. Station, Baseline, and Phasing Board control operations on the MCCC are described in separate diagrams. These operations could be on the same physical CPU, or on separate CPUs within the same box (e.g. CPU cards plugged into a CompactPCI backplane). (If performance capabilities of one MCCC CPU are exceeded, functionality could be split onto different CPUs. Thus, it may be advantageous to design IPC mechanisms on the MCCC up front to allow for this possibility.)

3.2.1 Station Control

A complete station monitor and control data flow diagram is shown in Figure 3-2. Much of this functionality has been used in the system described in [6]. In this diagram, high-level station control functions are performed on the MCCC ("**MCCC: Station Control Functions**" in the diagram) and this computer passes off lower-level hard-real-time control functions to embedded CMIBs on each Station Board. All access to hardware functions is abstracted via a layer of device driver software (not shown). A description of each task shown in the diagram is contained in the following sections.

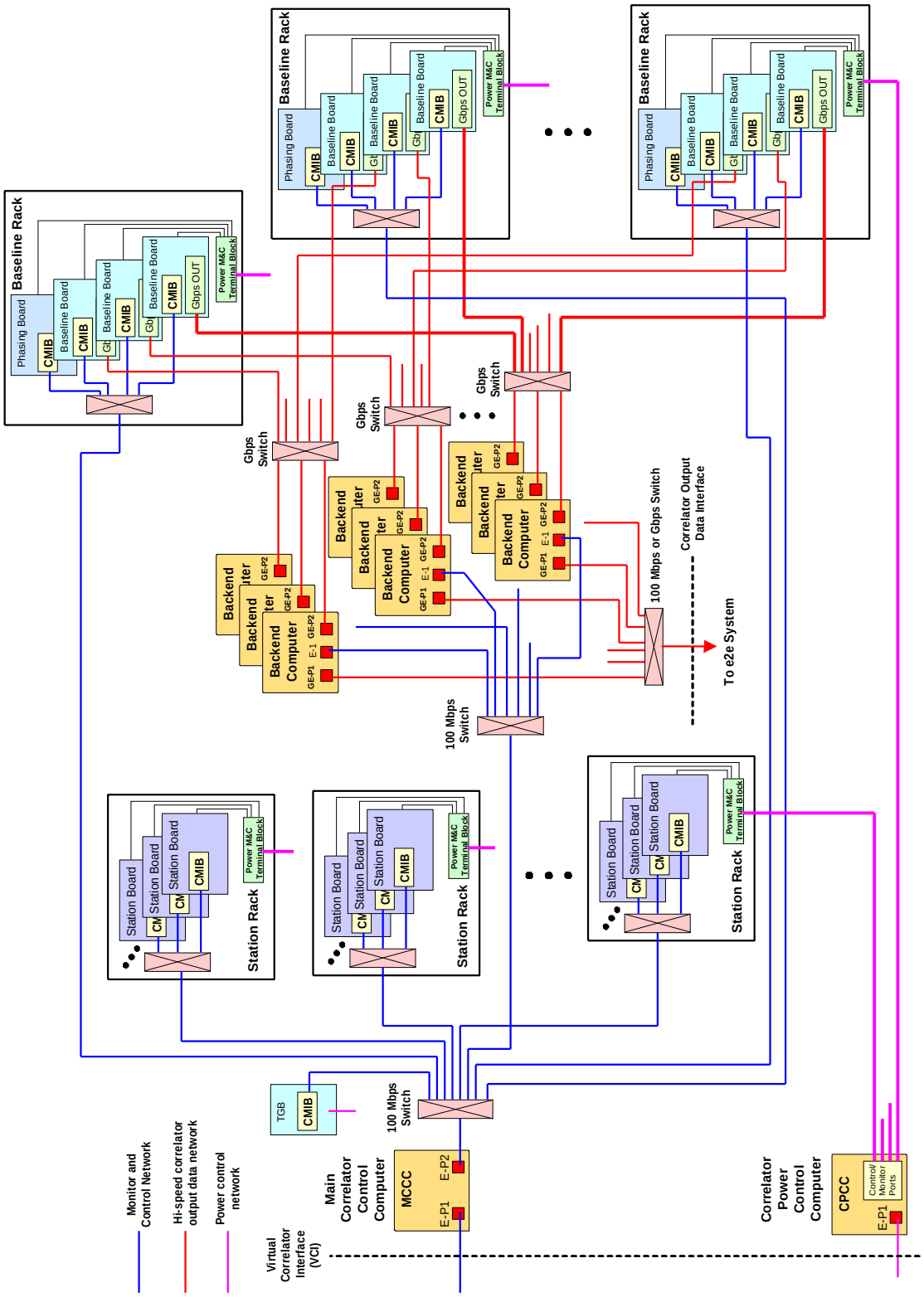

Figure 3-1 Network diagram of the correlator. The MCCC is the entry port for monitor and control. The Backend Computers handle all output data including data (indirectly) from the Station Boards.

3.2.1.1 Station Board CMIB Tasks

ISR: Interrupt Service Routine

The Interrupt Service Routine is where the lowest-level hard real-time code execution occurs. The 10 millisecond time ticks embedded in TIMECODE [2] are used to synchronously (and thus predictably and deterministically) interrupt the CMIB. The functions performed by the ISR every interrupt are as follows:

- Load point-slope delay and phase models into Station Board hardware. Also, capture delay models from the output of the Model queue to provide to the Delay and Phase Control Task. Phase models are calculated directly from final delay models given the Local Oscillator frequency, frequency offset, and time.
- Trigger data dumps on the next 10 millisecond time tick. The information required to do this is contained in the "Configuration Table".
- Check for data ready to readout and give a semaphore to the Data Readout Task if it is.
- Send TIMECODE to the Time Client Task. This is the mechanism by which the CMIB, the MCCC, and eventually higher-level software knows what time it is.
- Provide and receive necessary triggers to/from the DUMPTRIG Control Task.
- Send any error messages to the Error Message Task.

Delay and Phase Control Task

This task checks on a regular basis to ensure that the Station Board is properly tracking delay (and phase, although tracking phase is probably a null function because of the way phase is handled in the system [2]). Nominally, it does this by periodically checking the actual inserted delay with the delay model captured by the ISR from the Model queue. Additionally, this task could be responsible for updating FIR filter coefficients in some input baseband modes where very fine delay tracking is performed by changing FIR filter tap coefficients.

DUMPTRIG Control Task

This task is responsible for synchronization of DUMPTRIG [2]. Nominally this means exchanging trigger and status information with the ISR, although it is unclear at this point exactly what happens since a mechanism for generating DUMPTRIG has not yet been defined. What is clear is that in the case where DUMPTRIG is synchronized to a pulsar timer, careful synchronization across all Station Boards must be performed (i.e. DUMPTRIG generated by each set of four Station Boards must act "in concert" across the entire system so that binning and timing are synchronized).

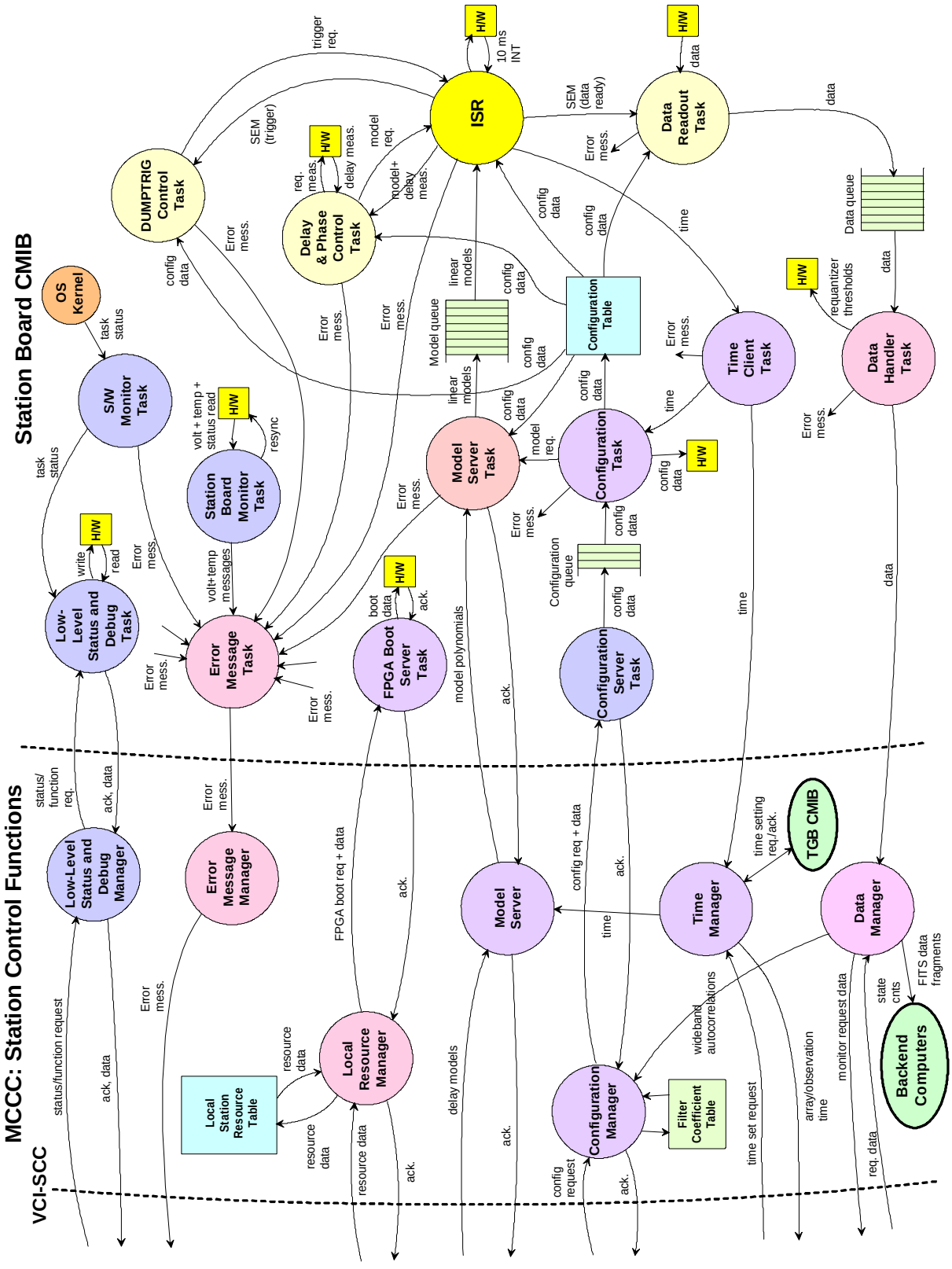

Figure 3-2 Complete data flow diagram for correlator station control.

Data Readout Task

The ISR gives a semaphore to this task any time there is data ready to read out. This task reads the data out, timestamps it with TIMECODE, and sends it to the Data queue for further processing. This task must readout the data before the next hardware dump occurs, and must properly handle overrun conditions.

Model Server Task

This task is responsible for performing low-level point-slope fits to delay model polynomials[5] received from the MCCC-level Model Server. This server in turn receives its models from a model generator task running on a computer residing above the VCI. In the case where sub-band multi-beaming is employed, this server calculates the difference between the baseband beam delay and the sub-band beam delay, and uses the result for the sub-band beam delay. Thus, in this case, the model generator task above the VCI calculate a polynomial for every sub-band beam.

Configuration Task

The Configuration Task is responsible for taking configuration data in the (short) Configuration queue and configuring the hardware at the time requested. While configuration is occurring, it will be necessary to temporarily suspend current operation as well as ensure proper Model Server Task synchronization. This task writes configuration data into the Configuration Table.

Time Client Task

This task is responsible for packaging TIMECODE from the ISR and sending it to the Time Manager on the MCCC. It is referred to as a client task since it is a network client for the Time Manager. This task also provides the Configuration Task with time (probably via shared memory). Since TIMECODE contains the complete absolute time, it is not necessary for there to be any "time adjustment feedback mechanism" from the Time Manager on the MCCC.

Data Handler Task

The function of this task is to simply package data from the Data queue for transmission to the MCCC's Data Manager. If the Data queue is reasonably large, then this task can suffer substantial latency and can run at a low priority.

Configuration Server Task

The Configuration Server Task is a network (RPC) server that accepts packaged configuration data from the MCCC, unwraps it, checks it for consistency, and then writes it into the Configuration queue for eventual execution at the requested time.

[5] This task should have the capability of merging multiple delay polynomials to form one polynomial before point-slope fitting occurs. This allows things like station clock models (in real-time VLBI) to be merged with delay models.

FPGA Boot Server Task

This task is normally only active on start-up or reset when the Station Board FPGA hardware must be initialized[6]. FPGA "personalities" (or more correctly, configuration bitstreams) will be resident on the CMIB's local Flash Disk, and so there is very little data that must be transferred to this task to start the boot sequence. While FPGAs are being booted, no interrupts are active and thus all other task functions should be suspended.

Station Board Monitor Task

This task is activated every so often (few seconds?) to read and check on the status of on-board voltage and temperature monitors. It also may check on the status of on-line synchronization and signal integrity monitors, although it is not yet clear if this kind of function should be performed by this task directly or via the ISR.

Software Monitor Task

The job of the Software Monitor Task is to monitor the health of the operating system and the health and existence of all of the other tasks. This is to provide more fault coverage (although it can never be 100%) than would be possible without it, since it is possible for a task to crash and not be noticed by the MCCC. Every time status is checked and found to be ok (or contain a fault), a message is sent via the Error Message Task to the MCCC. If the MCCC does not receive these messages, then it can assume that the CMIB has crashed and attempt to reboot it (this function is also provided by a h/w watchdog timer, so maybe this is redundant and unnecessary?). If this task detects a task crash, it could also try to clean up and restart the task, although this could lead to further problematic operation.

Error Message Task

The function of this task is to simply take error messages from all of the other tasks (via a message queue not shown) and the ISR and package them for transmission to the MCCC. The format and numbering of these error messages must meet the overall EVLA monitor and control system requirements [7][8].

Low-level Status and Debug Task

The purpose of this task is to allow "back-door" access to low-level device driver calls and status information. This is to facilitate troubleshooting without having to rlogin to the CMIB (although rlogin is entirely possible). In addition, this task should be able to enable a global switch that turns on all debug output, funnels it to this task, that then transmits it to the MCCC and eventually the CMCS Test Software (see section 2.2) for display. That is, any debug statements installed by a programmer during development should be able to be invoked and accessed by this task. To be effective, this implies that programmers should employ a consistent debug methodology to facilitate this operation.

[6] That is, if configuration changes do not require rebooting the FPGAs—the current design goal.

3.2.1.2 MCCC: Station Control Functions

The MCCC contains a number of high-level processes/tasks that are the interface between the low-level CMIB tasks and the monitor and control system software lying above the VCI. These tasks operate in "quasi" real-time, and thus some arbitrary (but reasonable) latency in their operation must be accounted for in the design.

Model Server

This process responds to delay model polynomial messages coming from a separate model generation computer lying above the VCI. It relays (and perhaps repackages) these models to the appropriate low-level CMIBs. Thus, the external model generator has one point of contact for all models. This server may have to handle additional models coming from things like WVRs (Water Vapour Radiometers) that, for speed, may want to by-pass the external model generation computer. The exact mechanism and real-time performance requirement for handling models from the WVRs is TBD.

Time Manager

The Time Manager process is responsible for setting time on the TIMECODE Generator Box (TGB) in response to a request from the higher-level monitor and control system lying above the VCI. For the EVLA this is normally the real-time UTC, but if part of the correlator is used for non-real-time VLBI, then this could be an arbitrary setting (the TGB can generate four independently programmable TIMECODES, one for real-time, and three for VLBI). The exact mechanism for synchronization of the TIMECODE setting is TBD, but some issues to consider are as follows:

- Latency through the network (above the VCI) and to the TGB CMIB must be considered. The TGB will be provided a reference clock and time tick from an external source, and the TGB will be commanded to set an epoch on the next occurrence of the time tick. Network latency must be much less than the time tick period for this setting to be reliable. (The TGB could be provided with a small counter value that increments each time tick to alleviate this problem.).

- The TIMECODE that TGB generates must be delayed from real-time UTC by an amount equivalent to one-half the Station Board delay buffer depth. If a 64 M deep buffer is used, and the data is clocked at 256 MHz, then a delay of 0.25 seconds is required. This delay is needed so that the time-tagged data from an antenna at the array delay center shows up at the output of the buffer at the same time as TIMECODE. Positive and negative delays relative to the array phase center can thus be realized.

This process also collects time messages from CMIBs (probably including CMIBs on Baseline Boards, although this requirement is less clear) and feeds time messages to the monitor and control software above the VCI.

Data Manager

This process collects all incoming data from Station Board CMIBs and packages it for transmission/distribution to the Backend Computers. The Backend Computers need this data to perform some visibility data corrections and also to package it with the visibility data (from the Baseline Boards) for transmission to the e2e System. The Data Manager also sends select data to the higher-level monitor and control system on request (i.e. for operator monitoring purposes of things like state counts, power levels, and autocorrelation spectra), to meet the requirements of section 2.1.

Configuration Manager

This process is the configuration entry point for correlator station configuration requests from the monitor and control system. A straw-man concept of the configuration information required is defined in section 3.3.1. This configuration manager is shown as separate from any Baseline or Phasing Board configuration—not necessarily the case, although there may be some advantages in keeping them separate (e.g. one could imagine changing baseline and/or phasing configurations without changing station configuration).

Local Resource Manager

This process is told by the higher-level monitor control system information about hardware resources in the system (such as CMIB IP addresses), and facilitates mapping between logical requests for configuration and physical devices. This manager is also responsible for telling CMIBs to boot hardware.

Error Message Manager

This process's simplest function is to collate error/warning/status messages coming from the CMIBs for transmission to the monitor and control system across the VCI. Additionally, this process may provide some "throttling" capability and/or intelligent interpretation of messages to provide a higher level of error flagging to the monitor and control system. The exact functionality of this process is TBD.

Low-level Status and Debug Manager

This process provides a single point of entry for low-level access to the similar process on all correlator CMIBs. There is probably one process to provide access to all CMIBs installed in all hardware modules. Exact functionality is TBD.

3.2.2 Baseline Control

A complete baseline monitor and control diagram is shown in Figure 3-3. Many of the tasks/functions on the MCCC parallel those for station monitor and control (section 3.2.1)—these could be incorporated into the same tasks on the same CPU, or could be separate tasks to facilitate operation on a different physical CPU.

In this diagram, high-level baseline control functions are performed on the MCCC ("**MCCC: Baseline Control Functions**" in the diagram) and this computer passes off lower-level hard real-time control functions to embedded CMIBs on each Baseline

Board. All access to hardware functions is abstracted via a layer of device driver software (not shown). A description of each task shown in Figure 3-3 is contained in the following sections.

3.2.2.1 Baseline Board CMIB Tasks

ISR: Interrupt Service Routine

The ISR on the baseline monitor and control CMIB is relatively simple compared to that on the Station Board. This is because all "hard real-time" processing is performed in hardware by signals driven from the Station Boards. The ISR is simply used to acquire TIMECODE on 10 millisecond interrupts, supply time to the Time Client Task, and (synchronous to the 10 millisecond time tick) wake up other tasks for processing.

Configuration Task

This task is used to configure Baseline Board hardware and operates in a similar fashion to the analogous task on the Station Board CMIB. Some of the configuration functions it must perform are as follows:

- Set correlator chip configuration including defining data path switches and use of VLBI-mode vernier delay tracking and phase modifiers.

- Set Recirculation Controller configurations including data path switches, recirculation parameters etc. Also, load coefficients into the delay-to-phase lookup tables. Each lookup table nominally has 256 coefficients.

- Set destination IP addresses for the output serial transmission chip.

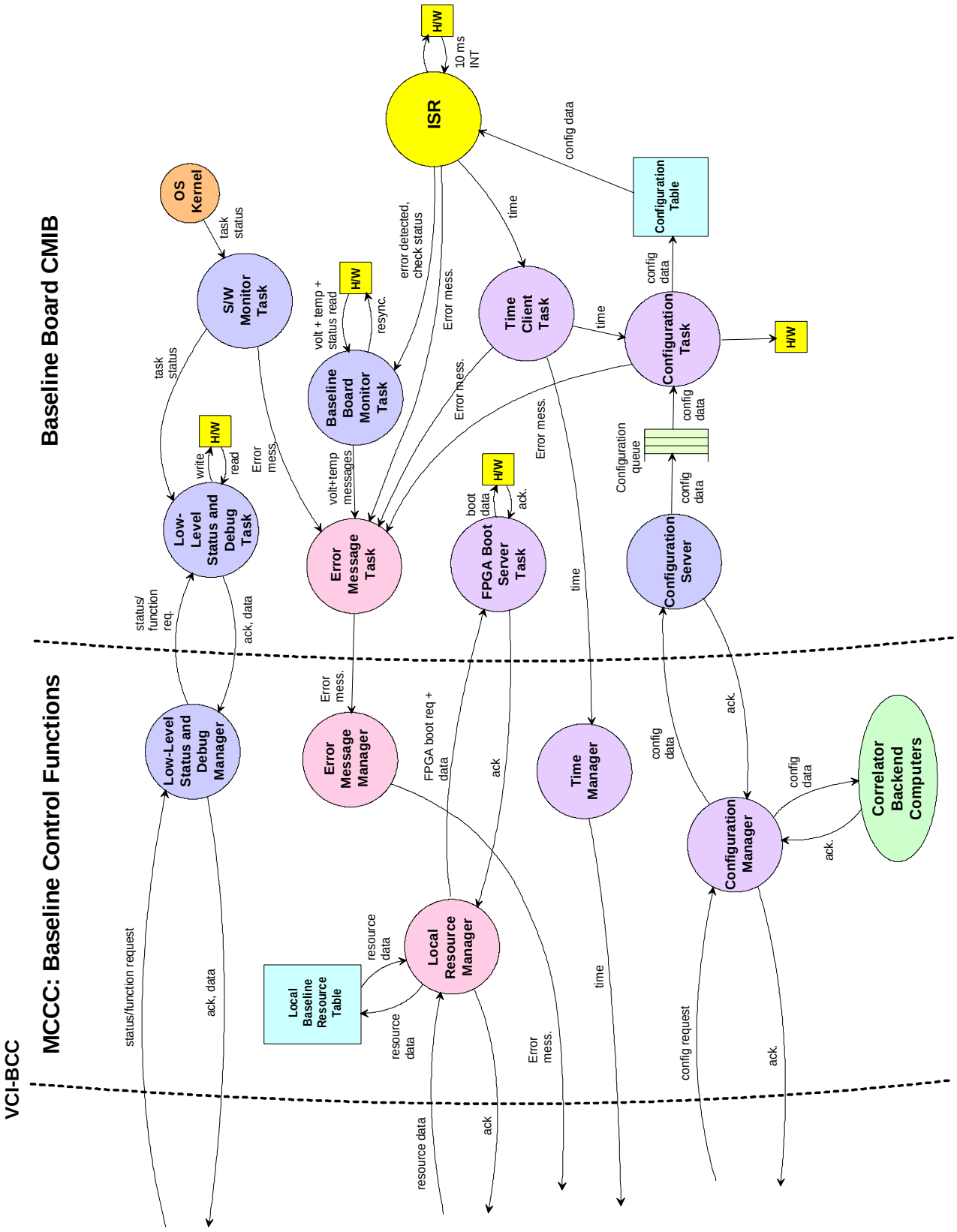

Figure 3-3 Baseline monitor and control data flow diagram.

Time Client Task

This task performs the same functions as on the Station Board CMIB. It supplies other CMIB tasks with time, and sends time messages to the Time Manager on the MCCC.

Baseline Board Monitor Task

The task reads on-board voltage and temperature monitors, and the status of other devices (such as Recirculation Controller and correlator chip synchronization) and sends the results to the Error Manager Task. This task will probably initialize a resynchronization command in the case where a chip has lost synch or is reporting errors.

Configuration Server

This function is analogous to the same function on the Station Board CMIB. It accepts incoming configuration requests from the MCCC, performs consistency checking on the data, and puts the configuration data into the short Configuration queue for eventual execution by the Configuration Task.

FPGA Boot Server Task

This function is identical to the same function on the Station Board CMIB.

Error Message Task

This function is identical to the same function on the Station Board CMIB.

S/W Monitor Task

This function is identical to the same function on the Station Board CMIB.

Low-level Status and Debug Task

This function is identical to the same function on the Station Board CMIB.

3.2.2.2 MCCC: Baseline Control Functions

Configuration Manager

This process is responsible for sending configuration requests to the Configuration Server on the Baseline Board CMIB. This process must also send configuration data to the correlator Backend Computers so they know what to do with the data, how to label it etc.

Time Manager

This process acquires time messages from all Baseline Board CMIBs and reports it, on a periodic basis, to the monitor and control system above the VCI. This process, and the same process defined in section 3.2.1.2, could probably be one and the same.

Error Message Manager

This process is identical to that for MCCC station control. It can probably be one and the same process, unless performance requirements and the desire to have it run on a different CPU dictate that it be a different process.

Local Resource Manager

This process provides the same functionality as the process defined in section 3.2.1.2.

Low-Level Status and Debug Manager

This process provides the same functionality as the process defined in section 3.2.1.2.

3.2.3 Phasing Control

A data flow diagram for monitor and control of the Phasing Board has not been developed since it is the simplest of all functions and contains similar tasks to the baseline monitor and control data flow diagram of Figure 3-3. Functions that must be performed by the Phasing CMIB are as follows:

- Enable or disable each antenna for phasing within the first stage adders.
- Load delay-to-phase lookup table coefficients in the first stage adders.
- Set second stage adder switches according to sub-array requirements.
- Set output requantizer, filter, and switch functions.
- Read and report data (power measurements and/or state counts) from various test points to monitor the health of each node of the adder tree for each sub-array. Also, read and report data synchronization status and on-board voltage and temperature monitors.

Phasing control functions in the MCCC are almost identical to those shown for baseline control in Figure 3-3. The only exception is that there should be no need for a Time Manager function. TIMECODE on the Phasing Board only functions to synchronize data streams before phasing, and to provide an output time reference for further processing.

3.2.4 Correlator Backend Processing

Correlator Backend computers process lag visibility data from Baseline Boards, merge it with data from Station Boards, and provide it to the e2e System for further processing (archiving, calibration, and image processing). For maximum performance at minimum cost, the current plan is to use commodity PC boxes that are fed data from Baseline Boards via Gigabit Ethernet links through commodity switches. This plan was first proposed in [3] and has the advantage that in all correlator modes all lag data needed for the FFT can be routed to one CPU box, avoiding the need for an additional wideband

network fabric between the computers[7]. Other advantages are that it facilitates dynamic load sharing amongst the PCs, is self-healing in the event of a PC crash, is easily scaleable in performance, and the PCs can be located some distance from the correlator boards.

As shown in Figure 3-1 (network architecture), each Backend Computer contains three Ethernet ports:

1. Gigabit Ethernet port for high-speed UDP/IP packet data from the network switch, originating at Baseline Boards.

2. Gigabit Ethernet (or 100 Mbit Ethernet?) port for transmitting data to the e2e System.

3. 100 Mbit Ethernet port for communication to the MCCC. This port would also be used for system administration.

The requirements of the Backend Computers are listed in section 2.3 and a data flow diagram is shown in Figure 3-4. Each Backend Computer could be a dual-processor machine with an operating system automatically handling task load sharing. A description of each of the tasks that run on the Backend Computers follows.

UDP Packet Receiver Task

This process should run at the highest user priority and is responsible for receiving incoming UDP/IP packets, reformatting them, merging them with configuration data and required station data, and writing them to the Data queue. Since the PC is a virtual memory machine with (probably) several Gigabytes of memory, the data queue can be quite large—supporting the requirement for high dump rate burst operation. Lag data from the UDP packets as well as configuration and real-time station data is written to the Data queue to maintain synchronization integrity in the case where a configuration change occurs with Data still in the queue[8].

FITS Fragment Generator

This process performs the bulk of the processing on the data. The processing that is performed is outlined in section 2.3. This process writes or transmits data to the e2e System for further processing. The current format of the output data is undefined, but it is shown in the diagram as being FITS file fragments.

[7] i.e. to support a distributed FFT.
[8] i.e. the queue contains all of the necessary data so that the "FITS Fragment Generator" process does not need to worry about any data/configuration synchronization issues.

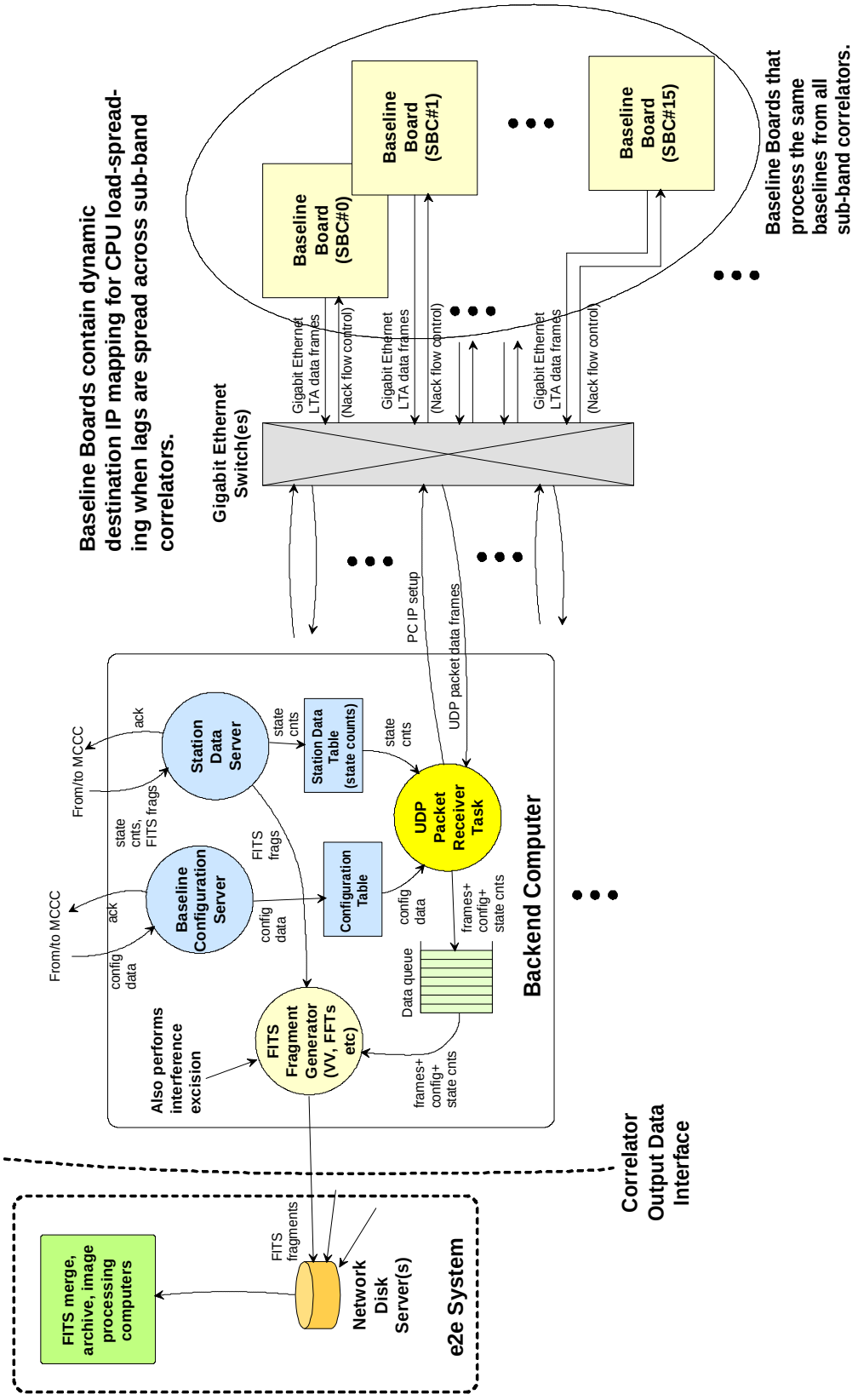

Figure 3-4 Correlator Backend processing data flow diagram.

Station Data Server

This server acquires data products from the MCCC (coming from the Station Boards) that are required for visibility data processing as defined in section 2.3. These are written into a "Station Data Table" for use by the UDP Packet Receiver Task. Additionally, this server may acquire station data FITS fragments for merging with the data that travels to the e2e System. Although this process is defined here as a server, it is not yet known whether a client on the MCCC will initiate data distribution to all Backend Computer servers, or whether this process should be a client that initiates requests to the MCCC (or the CMIBs on the Station Boards for that matter).

Baseline Configuration Server

The simple job of this server is to accept configuration data from the MCCC, and place it in the local Configuration Table for use by the UDP Packet Receiver Task. This information is required so that visibility data from Baseline Boards can be properly labeled (with local oscillator frequencies etc.) before going to the e2e System. The exact content of this data requires further definition.

3.3 Data Structure/Object Diagrams

This section contains straw-man concepts for *configuration* data structures or objects that cross the VCI. These objects form a baseline plan for what the EVLA monitor and control system must provide to the correlator via the VCI, and ultimately what a "correlator configuration builder" in the e2e System must generate. Many objects are defined in logical terms and will eventually, in many cases, contain more detail than specified here. Also, some housekeeping elements such as "observation code" and "sub-array" code are ommitted.

3.3.1 Station Configuration

The structure/object shown in Figure 3-5 define the bulk of how the correlator is configured. The structure defines what each sub-band of each baseband of each antenna of a sub-array is to do, including correlator parameters like number of spectral channels and number of polarizations. Exactly how the spectral channels and polarizations are implemented is defined in section 3.3.2 on Baseline Configuration. A description of the baseband and sub-band structure shown in Figure 3-5 follows.

3.3.1.1 Baseband Control Parameters

Each "station input" to the correlator consists of four wideband baseband pairs. A Station Board consists of a pair of basebands and each baseband of a pair can be independently programmable in delay. Although for the EVLA each antenna normally uses four Station Boards, it is possible for a separate antenna to connect to each Station Board and the control parameters defined here allow for this possibility. The following

list is a straw-man description of "physical baseband" ("**PhBB**") control parameters identified in Figure 3-5.

- ***antenna_ID*** – This is an identifier for the physical antenna that this baseband connects to. This ID is embedded in the data stream (from the antenna) on the FOTS and can be read from the fiber-optic receiver module that plugs into the Station Board.

- ***pad_ID*** – This is an identifier for the physical antenna pad that the antenna is currently mounted on. This ID is embedded in the data stream (from the pad site) on the FOTS and can be read from the fiber-optic receiver module that plugs into the Station Board.

- ***antenna_coords*** – This is the coordinates of the antenna. This parameter may or may not be required depending on the operation of the delay model generator and the requirements of data output from the correlator backend.

- ***SRC_coord***, ***SRC_name*** – Source information that may or may not be required depending on the operation of the delay model generator and the requirements of data output from the correlator backend.

- ***array_center*** – Phase center of the array required by the delay model generator (may or may not be needed by the correlator...).

- ***BBstr[16]{group,bits,L|M,coding,LO,fshift,BW,sideband,pol}*** – This describes the content of each of the 16, 4-bit data paths for the baseband. The ***group*** defines what sampler this came from, and normally for the EVLA this will be '0' for each of the 16 entries. However, there could be up to 16 groups, each one sampling a different baseband with a different sampler. The ***bits*** defines how many bits are used of the four available. The ***L|M*** parameter defines whether this 4-bit path constitutes the *L*east-significant or *M*ost-significant bits of a word wider than 4-bits (i.e. for 8-bit sampling). The ***coding*** parameter defines the type of coding used (2's complement, signed-binary etc.). The ***LO*** parameter is the sum of all of the LO stages for this baseband including the last down-conversion from quasi-baseband provided by the sampler. It **does** include the small frequency offset (***fshift***), since that is the LO that is actually used. ***fshift*** is the small frequency offset introduced in the LO. It must be known in addition to the actual LO used so that the correct fringe rotation phase can be calculated in the correlator. ***BW*** is the bandwidth of the group—the sampling frequency is always taken as double this value. The ***sideband*** parameter indicates whether this is upper or lower sideband. If lower sideband, then an inverted frequency sense is assumed. Finally, ***pol*** indicates the polarization of this group.

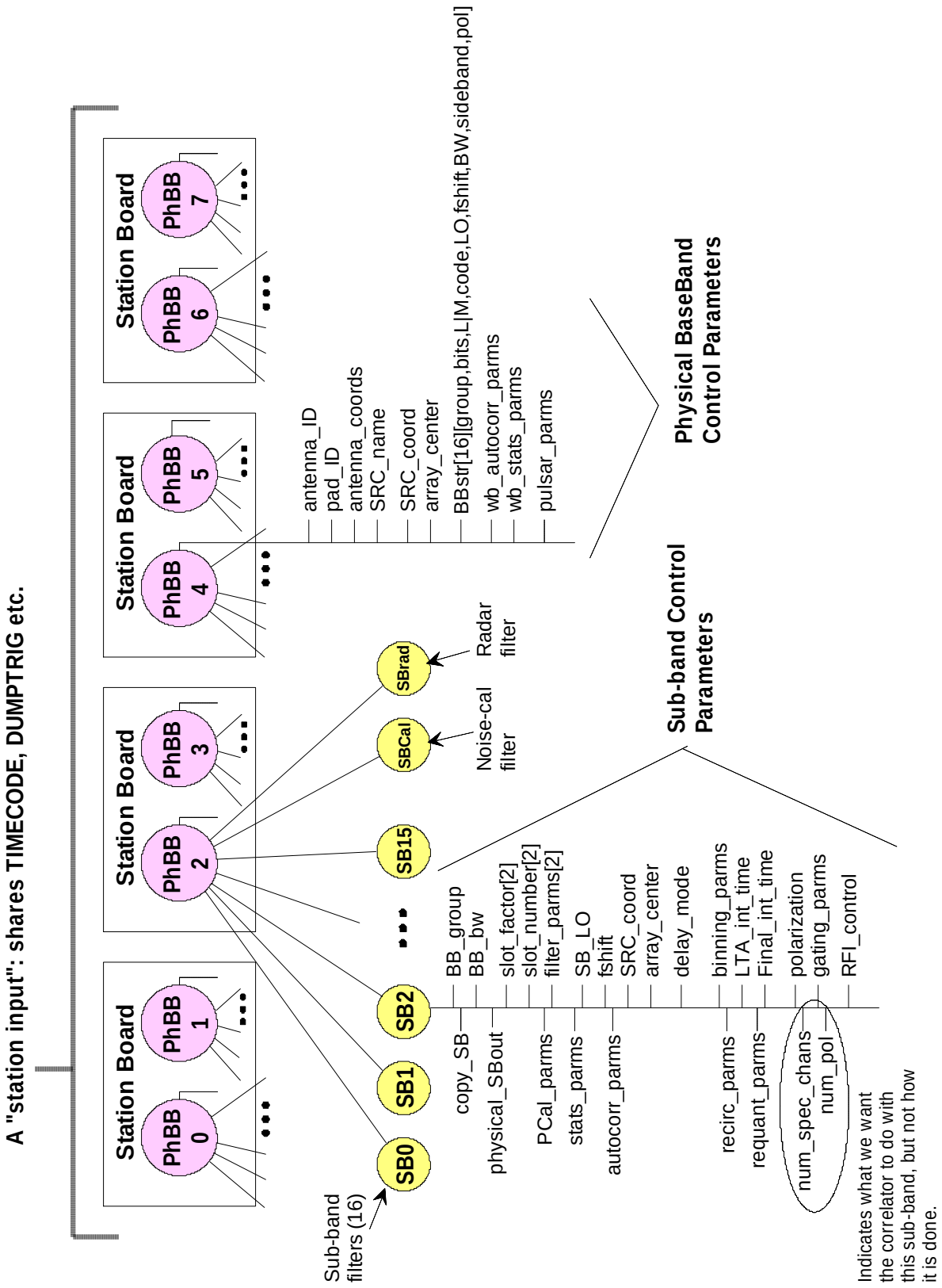

Figure 3-5 Station control data structure/object. With this structure, a station's basebands and sub-bands are completely specified.

- ***wb_autocorr_parms*** – This describes the wideband autocorrelator requirements. Exact definition of these parameters requires further development.
- ***wb_stats_parms*** – This describes wideband quantizer statistics acquisition requirements. Exact definition of these parameters requires further development.
- ***pulsar_parms*** – This defines the pulsar timer parameters for this baseband such as epoch, period and "p-dot". There is one independent pulsar timer for each baseband. It may be necessary to update these parameters from time to time without having to perform a full configuration.

3.3.1.2 Sub-band Control Parameters

Within each "PhBB" there are 18 sub-band filters. Sixteen of these filters are "general purpose", one filter is used for interference-free noise-diode power acquisition[9], and one has additional taps[10] for narrowband "radar-mode". A description of each of the sub-band parameters is as follows:

- ***BB_group*** – This indicates which group of the baseband (PhBB) is used as inputs for the filter. Normally for the EVLA this is '0' (i.e. 1 group, all 16 streams), but for other baseband configurations or antennas, other groups could be used.
- ***BB_bw*** – This is the bandwidth of the group, and mirrors the information contained in PhBB[m].BBstr[].***BW*** parameter. Since this is redundant information, it could be omitted.
- ***slot_factor[2]*** – These parameters indicate filter slot factors for each of two possible stages. That is, a filter stage will be one slot number in this number of slots (e.g. 16, 32, 64, 128, etc.)[11]. If only one stage filtering is used, then the second stage parameter is set to some null value.
- ***slot_number[2]*** – These parameters indicate filter slot numbers for each of the two possible stages. These are integers modulo *slot_factor* (i.e. 0...*slot_factor*-1). If only one stage filtering is used, then the second stage parameter is set to some null value.
- ***filter_parms[2]*** – This parameter(s) defines desired filter characteristics including perhaps some filter notches. The contents of this parameter requires further definition because it depends on how filter coefficients are generated (i.e. selection from a pre-defined set of coefficients, on-the-fly coefficient generation, or some hybrid of the two).

[9] This filter may need some dynamic configuration properties in case RFI suddenly shows up in it for some reason.
[10] If the general purpose filters have enough taps, then the need for a separate radar-mode filter may be eliminated.
[11] Note that *slot_factor* does not have to be a power of two. For example, if 3 input streams define a baseband, then *slot_factor* could be 3, 6, 12 etc.

- ***SB_LO*** – This is the final effective LO frequency for the sub-band. This could be derived from PhBB[m].BBstr[].***LO*** and sub-band filter parameters and so may be redundant information. This LO **does** include the frequency offset (*fshift*).

- ***fshift*** – This is the small frequency offset or shift that is actually in the LO. This is used directly in the fringe phase model, and is the same as the "parent" PhBB's fshift.

- ***SRC_coord***, ***array_center*** – These parameters are used to define a sub-band beam that may be different than the baseband beam (*array_center* may be redundant and unnecessary information). They are null values if the sub-band beam pointing is not different than the baseband beam.

- ***delay_mode*** – This defines the very fine delay tracking mode. Very fine delay tracking can use a different method and have different precision depending on the sub-banding factor of a particular baseband. Nominally, this is either "normal" or "VLBI-mode", but could be "FIR-tap-mode" if very fine delay is implemented by changing FIR filter tap coefficients.

- ***binning_parms*** – Defines how output data is binned. This, along with *LTA_int_time* and *pulsar_parms* defines the operation of DUMPTRIG. Note that if dumping is synchronized to a pulsar timer, it can only be synchronized to one timer on the "master" Station Board (PhBB0).

- ***LTA_int_time*** – The station-based cross-correlation H/W LTA integration time for this sub-band. This parameter primarily determines the output data rate from the Baseline Board to the backend computers.

- ***Final_int_time*** – The final integration time that occurs in the backend computers. This determines the output data rate to the e2e System.

- ***polarization*** – The polarization of this sub-band.

- ***gating_parms*** – Gating parameters (if used) for this sub-band. Parameters include offset time from the baseband pulsar timer epoch, and gate width.

- ***RFI_control*** – This defines what sort of post-correlation RFI mitigation is to be used for this sub-band. This could be "none", "excision", or "cancellation" (and there may be others).

- ***copy_SB*** – This parameter allows multiple Station Board sub-band outputs (of which there are 18) to contain copies of data from the same sub-band filter. If this is set to a non-null value, then the physical filter that this object refers to is idle and *physical_SBout* (i.e. the output of the Station Board cross-bar switch) is connected to the *copy_SB* filter.

- ***physical_SBout*** – This defines which physical output (0...17) of the Station Board (for this baseband) this sub-band output is connected to. <u>This essentially defines which sub-band correlator the output of this filter is routed to</u>.

- ***PCal_parms*** – Phase-cal parameters for this sub-band (frequencies, integration time). Note that *fshift* is not removed yet, and so the phase-cal frequencies must take this into account.
- ***stats_parms*** – Requantizer statistics' (power, state counts) acquisition parameters for this sub-band.
- ***autocorr_parms*** – Sub-band autocorrelation parameters.
- ***recirc_parms*** – Recirculation parameters for this sub-band, as it affects DUMPTRIG. Requires further definition.
- ***requant_parms*** – Parameters that define the operation of the requantizer, including number of bits (4 or 7), and software AGC loop parameters.
- ***num_spec_chans***, ***num_pol*** – Parameters that define what we want to do with this sub-band, but not "how" it is done. How it is done is defined with baseline control parameters.

3.3.2 Baseline Configuration

The station configuration objects of the previous section define how basebands and sub-bands are configured. The baseline configuration objects of this section define what resources are used for the correlation. The structure/object required for baseline configuration is shown in Figure 3-6. Each structure defines the configuration for a particular sub-array, and the default is that all baselines within a sub-band correlator within the same sub-array are identical. This restriction is somewhat enforced by the fact that the output of a Recirculation Controller on a Baseline Board connects to all correlator chips in a row or column in an identical fashion. In the case where different baselines are being correlated differently within the same sub-array (such as when stations are being correlated with different total bandwidths) it will be necessary to specify the X and Y station types. One mechanism for handling this exception is described in the following sub-section.

3.3.2.1 Baseline Configuration Parameters

The elements of Figure 3-6 are defined in the following list:

- ***SBC0, SBC1, ..., SBC15, ...*** These elements represent sub-band correlators. Nominally, there are 16 sub-band correlators in the system, but there is provision for up to 18 [2] (and, if desired, there could be more, although with non-unique data inputs). Each sub-band correlator correlates all baselines for one sub-band from all basebands. For example, SBC0 could correlate all baselines, for sub-band 0 (i.e. physical output 0 [*physical_SBout*]) from all basebands.
- ***Baseline Correlator Unit***. This represents a physical correlator chip and contains 16 independent cross-correlators (CCCs). These cross-correlators can be concatenated in various ways to form longer "lag chains" and various combinations of polarization products. Correlator chips can be concatenated across sub-band correlators, and recirculation can be used to increase the number

of lags a correlator chip can generate. There are some restrictions on data connectivity in the chip [2].

- ***CCC0, CCC1, ... CCC15***. These are "Correlator Chip Cells" and each one is a separate 128-lag indivisible correlator unit. Four CCCs make up a CCQ (Correlator Chip Quad) and each CCQ has all of the routing resources necessary for all four polarization products from a PhBB pair (i.e. PhBB(0,1), PhBB(2,3) etc.)

Data structures/objects required for configuration define the properties of each CCC since a CCC is the smallest indivisible correlator unit. These properties could indicate that a CCC is just one part of a larger correlation unit comprised of multiple CCCs. A CCC structure/object is defined below:

- ***X-input***, ***Y-input***: These elements independently define the X and Y inputs to each CCC. Each one is of the form **PhBB[m][SBn]**, where "**m**" (0,1, ..., 7) refers to the baseband the data came from, and "**n**" (0,1, ..., 17) refers to the sub-band within the baseband. In more physical terms, **n** is the physical sub-band output number from the Station Board (Figure 3-5, section 3.3.1, "*physical_SBout*") independent of what sub-band filter it is actually connected to and **m** is the physical baseband (PhBB) number (Figure 3-5). **m** and **n** must be consistent with station baseband/sub-band definitions (section 3.3.1), fixed cable routing, Recirculation Controller switching capabilities, and correlator chip routing limitations [2]. If the X and/or Y input connects to an adjacent CCC, as would be done if CCCs are concatenated, then X-input and/or Y-input is "**adjacent**".

- ***X-type***, ***Y-type***. These elements define what type of antenna is connected to the X and Y inputs. The type primarily refers to the bandwidth that the antenna is producing and whether it uses 1, 2, or 4 Station Boards. This provides one mechanism (there could be others that are better) to allow correlations between antennas with different bandwidths, but without having to define an object for each and every correlator chip in the system. For example if all correlations for a sub-array are to be performed with mixed EVLA (16 GHz) and VLBA (4 GHz)[12] antennas, then 3 of the objects shown in Figure 3-6 would have to be defined. Namely (X-type=EVLA, Y-type=EVLA), (X-type=EVLA, Y-type=VLBA), and (X-type=VLBA, Y-type=EVLA). If only EVLA-EVLA antenna correlations are performed, then only one structure/object is required (noting here that a structure/object contains a description for all sub-band correlators—i.e. Figure 3-6).

- ***CCgroup***. This number defines a unique and indivisible cross-correlation group or identifier for all "***type***-matched" baselines for this sub-array. It is a unique number across all sub-band correlators. Any CCC with the same CCgroup that is part of the same sub-array is part of the same "lag chain" and, after FFT, will produce one cross-power spectrum.

[12] Just as an example...this not meant to put future bandwidth restrictions on the VLBA antennas!

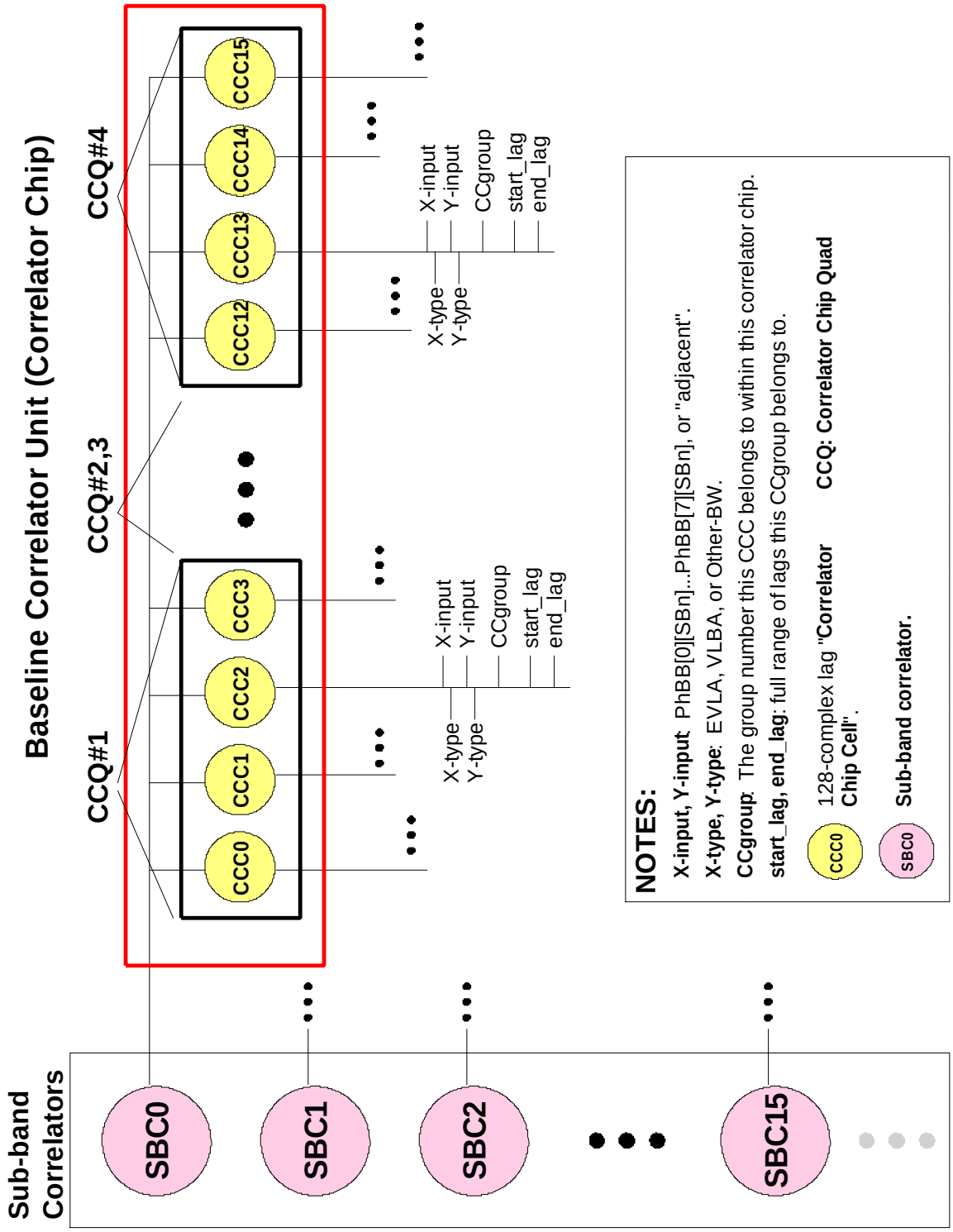

Figure 3-6 Baseline configuration structure/object. This indicates what the correlator is supposed to do with each active sub-band of each active baseband of a sub-array, using cross-correlator resources.

- ***start_lag***, ***end_lag***. These parameters define the range of lags that are acquired for this CCgroup that this CCC is part of. Thus, the CCC object of Figure 3-6 does not define precisely how a particular CCC is used, only that it is a part of a CCgroup, and what the requirements of that group are. In the special case where (start_lag-end_lag+1) is less than 128 and a CCgroup is *only* one CCC, the LTA controller will strip off unneeded lags (keeping the center lags) before shipping them off to the backend computers for further processing. This is normally only required at high dump rates to keep within the dump performance capabilities of the correlator and backend computers.

The above definitions provide a great deal of flexibility to allow (for example) multiple redundant cross-correlations, using different numbers of lags, different sub-band correlators, and different (or same) sub-band outputs from Station Boards.

In the correlator hardware, each CCC can be programmed to dump data on either the X or Y-station DUMPTRIG and thus use the X or Y-station *LTA_int_time* and *Final_int_time*. It is asserted that there is no need to specify the "dump controlling" station for each baseline as a configuration parameter that crosses the VCI since the setting can be derived from the station configuration data's dump time, antenna coordinates, and array phase-center coordinates as follows:

1. The station with the smallest dump time in the baseline is used.

2. The rule in 1. may be refined by calculating the baseline length, and choosing the X or Y dump time that most closely matches antenna dump times with antenna-to-phase-center baselines of similar length.

If the above algorithm is unacceptable, then a list specifying X or Y dumping for each baseline for each possible EVLA configuration could be provided as separate configuration data.

3.3.3 Phasing Configuration

This section contains a description of a structure/object that can be used to set Phasing Board configurations. Section 3.3.1 on station configurations already defines how sub-bands and basebands are configured and organized. The elements of a phasing configuration structure/object define how those sub-bands are to be phased, and what outputs are to be produced. There is one structure/object for each sub-array, using one sub-band of one baseband[13]. This approach provides flexibility in that it allows each Phasing Board to phase several sub-arrays, and each Phasing Board can be configured differently. The elements of the structure/object are as follows:

- ***phased_antennas[N]***. This is a list of antennas, each defined by a *pad_ID* (section 3.3.1) that are to be phased together.

[13] i.e. for one sub-band chunk.

- ***PhBB(m)SB(n)***. ***m*** and ***n*** define the physical sub-band chunk that is to be phased. Note that since the correlator will probably not be populated with Phasing Boards for all of the bandwidth, and since wiring to the Phasing Boards is fixed in the correlator, this specification must be based on apriori knowledge of the available wiring connections.

- ***output[k]{BW,slot,bits}***. This array defines output parameters for each of the ***k*** possible outputs of the Phasing Board that this sub-array will use. ***k*** is currently undefined, but it will probably be about 13 or 14 (8 to accommodate secondary filter outputs, 5 for primary outputs, and 1 for an 8-bit expansion output). ***BW*** is the bandwidth of the output. If ***BW*** is 0, then this k^{th} output is not used for this sub-array, but may be used for another sub-array. If ***BW*** is equal to the ***PhBB(m)SB(n)*** bandwidth (section 3.3.1), then no secondary filtering is performed. If ***BW*** is less than that bandwidth, then secondary filtering is performed, with the bandpass set for the desired ***slot*** (the "slot_factor" is derived from the bandwidth ratio). ***bits*** defines how many bits are used in the final quantizer before output. Nominally, ***bits*** can be one of 8, 4, 3, 2, or 1, but some outputs may not have (for example) 8-bit capability, although all probably will have at least 4-bit capability. Limitations will become more concrete once a more detailed hardware definition is available.

3.4 Miscellaneous Design Concepts

3.4.1 Slot-based IP Address Format

In section 2.1, it was stated that a 16-bit slot number will be used to form part of the IP address of the CMIB. This is a requirement that facilitates hot-swapping of modules without having to change the IP address for the replacement module in the MCCC. This 16-bit number is set on the entry module or backplane of the slot that the module plugs into and is unique in the system. A possible composition of the CMIB IP address is shown in Figure 3-7.

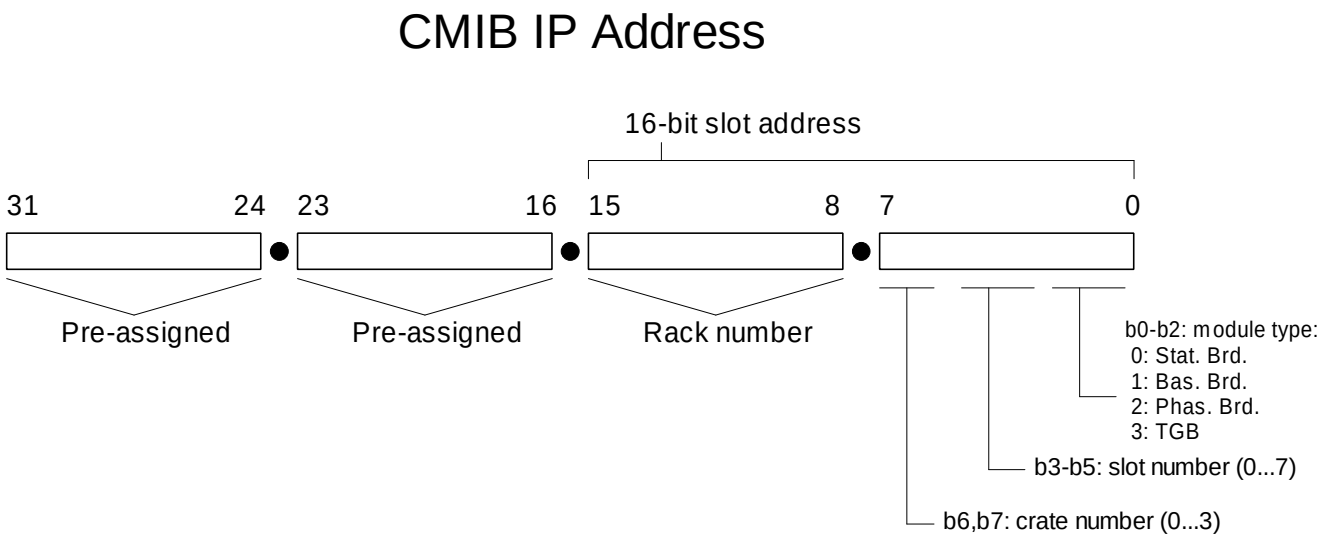

Figure 3-7 Possible composition of the CMIB IP address. The least-significant 16 bits uniquely identify the module and its location, and are provided by a 16-bit number from the slot's backplane.

3.4.2 Power Control Identification

Each module in the system can have its power independently controlled by the Correlator Power Control Computer (CPCC) shown in Figure 3-1. Since there could be several hundred modules in the EVLA correlator, it is necessary to establish a scheme to ensure that the CPCC can properly identify, and thus control, each module. Monitor and control of each module's power supply (independent of on-board CMIB monitoring) is achieved by using one sense/control line per module. Additionally, it is desired to be able to monitor fan failures within a crate (a.k.a. sub-rack). Thus, for each crate there will be 9 monitor/control lines (8 for modules, and one for fans) and one reference ground.

The following list indicates how all of these monitor and control points could get to the CPCC.

- Each crate contains a 15-pin terminal block of which 10 are used (9 m&c points and one ground). This terminal block routes all of these lines to (for example) a DB-15 pin connector.

- Each crate's 15-pin connector connects to a cable that goes to the CPCC. The CPCC contains enough m&c interface cards to accept all of the 15-pin connectors from all of the crates (48 for a 32-station correlator). All cables and routing must be labeled and programmed in the CPCC since there is no self-identification on each 15-pin connector.

- Multiple 15-pin connectors may be merged into a larger connector depending on the requirement of the monitor and control card that plugs into the CPCC.

In addition to “back-door” monitor and control of modules as described above, there will a need to monitor and control the correlator’s 48 VDC power plant and 110 VAC UPS(s). The mechanism to control these power sources is dependent on what the manufacturer supplies, but it should be able to be done via the CPCC. Thus, the CPCC should have its own dedicated high-reliability UPS.

4 References

[1] Cornwell, T., E2E Project Book Version: 0.5, NRAO, 11 October, 2001.

[2] Carlson, B., Refined EVLA WIDAR Correlator Architecture, NRC-EVLA Memo# 014, October 2, 2001.

[3] Rowen, B., WIDAR Correlator Backend Processing Options, NRAO, Socorro, 2001, November 19.

[4] Corbin, J.R., 1991, The Art of Distributed Applications (New York: Springer).

[5] Briggs, F.H., Bell, J.F., Kesteven, M.J., Removing Radio Interference from Contaminated Astronomical Spectra Using an Independent Reference Signal and Closure Relations, AJ, December 2000.

[6] Carlson, B.R., Dewdney, P.E., Burgess, T.A., Casorso, R.V., Petrachenko, W.T., Cannon, W.H., The S2 VLBI Correlator: A Correlator for Space VLBI and Geodetic Signal Processing, Publications of the Astronomical Society of the Pacific, 1999, 111, 1025-1047.

[7] Koski, W., Peck, G., Sahr, W., EVLA Monitor and Control System (EVLA Project Book, Chapter 9).

[8] Moeser, R., EVLA Operation’s System Software Requirements Specification, V1.0, Nov. 9, 2001