REQUIREMENTS AND FUNCTIONAL SPECIFICATION

EVLA Correlator Backend

Project Document: A25252N0000

Revision 1.0

Tom Morgan, September 23, 2003

National Radio Astronomy Observatory

Array Operations Center

P.O. Box O

Socorro, NM 87801-0387

Revision History

Date	Version	Description	Author
23-Sept-2003	1.0	Initial draft. Used Requirements document as a starting point and added functional specifications	Tom Morgan

Table of Contents

1 Introduction

1.1 Purpose

The primary goal of this document is to provide a complete and accurate list of requirements for the EVLA Correlator Backend System. Upon completion, the document will act as a binding contract between developers and users and will provide a common point of reference for system expectations.

The primary audience of this document includes, but is not limited to, project leaders, the designers and developers of the system and the end user. The document may also be of interest to EVLA project scientists and engineers or as a reference for individuals involved in similar projects with similar requirements.

The requirements contained in this document are numbered based on the section/subsection in which they appear.

Note: Text found between "<" and ">" indicates questions or comments to myself and/or readers. And In most cases, the phrase "The user" can be replaced with "An authorized user".

1.2 Scope

The Correlator Backend System lies between the Correlator and the End-to-End (e2e) System. It is the primary component of the real-time astronomical data processing capability (the processing pipeline) of the EVLA. Its primary responsibility is to perform basic data assembly, formatting and processing services and to support the desire for real-time inspection of the astronomical data stream.

The major functions the Correlator Backend System must perform are as follows:

1) Receive data from the Correlator in real-time.
2) Assemble time-series from the Correlator lag output.
3) Perform Fourier Transforms of the assembled time series.
4) Perform a limited number of additional processes upon user request.
5) Deliver suitably formatted results to the End-to-End System.

This document will define those requirements that must be fulfilled by the Correlator Backend System, plus requirements imposed on other systems by the Backend System.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Definitions

Administrator – An individual with unrestricted access to all aspects of the system.
Auxiliary Data – All other (non-astronomical) data.
Buffer – A data storage area defined by software and usually in main memory.
Cluster – A group of computers connected by a network and generally running identical operating systems and message passing middle ware.
Correlator Dump – The emptying of Long Term Accumulator (LTA) storage areas on the WIDAR Correlator. Generally all lags for all polarization products for all sub-bands and baselines are output in a short (compared to the time to accumulate the LTA contents) time.
Data – Astronomical observational data.
Lag Frame – The basic unit of data output from the WIDAR Correlator. For a complete definition of its contents at the bit level see NRC-EVLA Memo #014, Refined EVLA WIDAR Correlator Architecture page 70.
Lag Set – A complete, properly ordered series of lag values that can be submitted to the Fourier Transform function. The lag frames received from the Correlator will contain up to 128 lag values, so lag sets longer than 128 values will span multiple lag frames and require proper ordering and assembly into complete lag sets.
Metadata – All data about the astronomical data.
NaN – Literally, "Not a Number". For floating point data types, a bit string that does not translate into a valid floating-point number.
Node – A single computer in a Cluster. The computer may have more the one CPU.
Non-real-time – Offline operations with data input from some external storage device or generated internal (e.g., for testing).
Processing Pipeline – The series of BE functions performed on the astronomical data, i.e., that set of functions that the data passes directly through.
Processor – A physical computation device (hardware).
Process – A data processing procedure (software) realized as separate and distinct executable.
Real-time – Online operations with active astronomical data streaming from the Correlator.
Requantizers – Resamplers (used in the WIDAR Correlator).
State Counts – Sampling statistics produced by quantizers.

1.3.2 Acronyms

AOC –Array Operations Center
CMIB – Correlator Monitor Interface Board
CMCS – Correlator Monitor and Control System
COTS – Commodity-off-the-shelf (i.e., generic products)
e2e – End-to-End System (archive)
EVLA – The VLA Expansion Project
LTA – WIDAR Correlator Long Term Accumulator
M&C – Monitor and Control System
MPI – Message Passing Interface (message passing standard)
PVM – Parallel Virtual Machine (message passing middleware)
RFI – Radio Frequency Interference
SyRS – Refers to the *System Requirements* document.
SRS – Refers to the *Software Requirements Specification* document.

1.4 References

1) ANSI/IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
2) ANSI/IEEE Std 1233-1996, IEEE Guide for Developing System Requirements Specifications
3) EVLA Memo No. 15, Scientific Requirements for the EVLA Real-Time System
4) EVLA Project Book
5) EVLA System Requirements (SyRS)
6) EVLA Architecture and Design
7) The Very Large Array Observing Log (J. Nieri, February 1994)
8) Refined EVLA WIDAR Correlator Architecture, NRC-EVLA Memo# 014, Brent Carlson, Oct. 2, 2001.
9) EVLA Correlator Monitor and Control System, Test Software, and Backend Software Requirements and Design Concepts, NRC-EVLA Memo # 015, Brent Carlson, Jan. 23, 2002.
10) System Requirements Specification, EVLA Correlator Backend, Project Document A25251N0000, Revision 2.1, Tom Morgan, Dec. 20, 2002.

1.5 Overview

The remainder of this document contains a more detailed description of the Correlator Backend System as well as the requirements necessary to design and build the system. Section 2 provides a general description of the Correlator Backend System. Section 3 details the requirements of the product and the requirements impose on extemal systems. Section 4 presents the functional specification of the major software and hardware components. Sections 3 and 4 are the core of this document.

The format of the document follows that outlined in the IEEE STD 830 document, IEEE Recommended Practice for Software Requirements Specifications.

2 Overall Description

2.1 Product Perspective

The EVLA WIDAR Correlator Backend is a major component of the primary, real time astronomical data path. It sits directly between the Correlator and the Archive. All correlation results that are eventually placed in the Archive or otherwise used by offline systems must pass through the Backend. The Backend will be responsible for the generation of spectral channel results from correlation lags and other related processing activities. It must be designed and implemented as a real-time data processing system. That is, it must be capable of consuming and processing correlation lags at a rate greater than or equal to the rate they are generated by the Correlator.

The Correlator Backend will also sit alongside the EVLA Monitor and Control (M&C) System. M&C will provide all external command and control to the Backend and will receive all error, warning, performance and other reports from the Backend. All data, aside from correlation lags, needed by the Backend during the course of its activities must be provide by M&C.

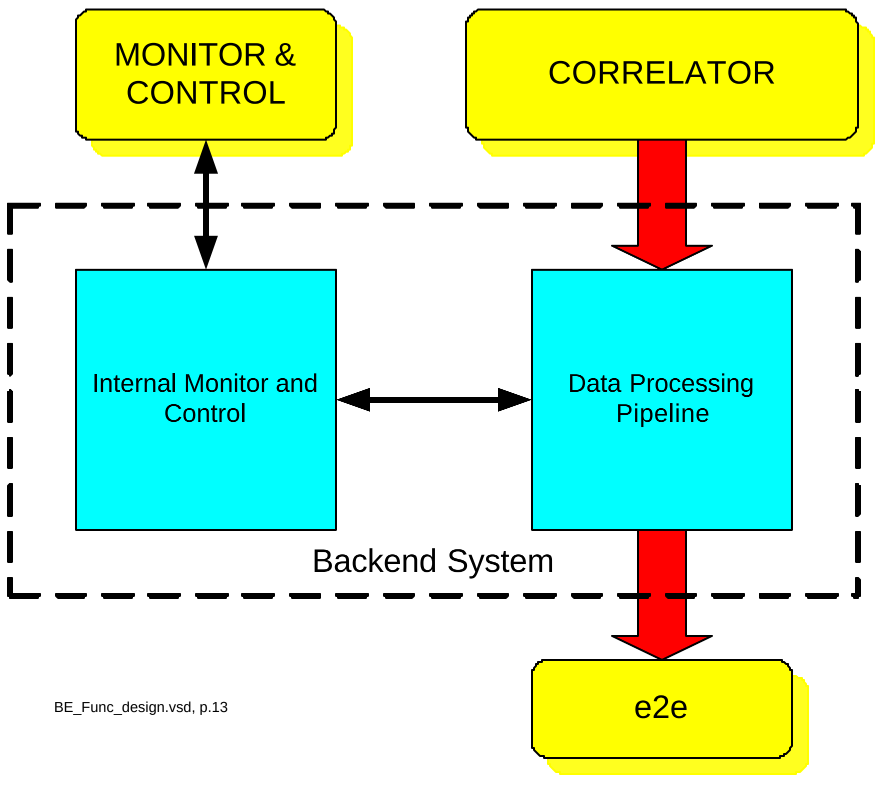


Figure 1. High-level block diagram of the two major components of the Correlator Backend System with interfaces to external systems. Thin arrows represent monitor, control and non-correlator lag frame data flows. The broad arrows represent Correlator lag products flowing into the Backend and processed spectra flowing into e2e.

2.2 *Product Functionality*

2.2.1 Data Input

Correlator lag data will be received directly from the Correlator Baseline Boards in the form of Lag Frames. The lag frames contain correlation lag values and all auxiliary parameters needed to assemble the lags into complete lag sets (properly ordered time series). It is currently assumed that all observational modes yielding correlator results that are transmitted to the Backend will be in the form of lag frames.

Additional auxiliary data and meta-data needed for processing prior to output to the e2e System will arrive via the Monitor and Control System, whether produced by the Correlator or some other part of the EVLA System.

The BE will receive and act upon status requests and control commands originating in or via the M&C System.

2.2.2 Data Processing

The Correlator lag frames will be assembled into time ordered series, normalized, and when necessary time stamp adjusted. The time series will be Fourier Transformed and integrated over a user-determined period of time. User selectable time and/or frequency domain processes may be applied to the data before and/or after Fourier Transform. Prior to output, the end results will be formatted to meet the internal needs of the e2e.

2.2.3 Data Output

Formatted spectra will be transferred to the end-to-end System. Pertinent meta-data available to the BE System will be associated with the output. The fundamental unit of output is a sub-band cross-power spectrum produced by the Correlator. No “stitching” operations that combine spectra from different sub-bands will be performed.

The BE will produce a variety of error, warning, status and other reports and messages that will be transferred to M&C for final disposition.

2.2.4 Monitoring

The Correlator Backend System will conduct a number of self-monitoring activities on application and system software as well as hardware systems to detect system failure and out of spec conditions.

2.2.5 Recovery

The ability to attempt recovery from failure and out of spec performance conditions will be built into the system.

2.2.6 Control

The system will provide control and auxiliary parameters to internal input, output, processing, monitor, recovery, and other functions and receive status and performance data from them. It will also communicate with the external Monitor and Control System.

2.3 User characteristics

All use of the Correlator Backend System will be indirect via the Monitor and Control System. The BE system will not directly produce user interface screens.

2.3.1 Array Operator

The primary contact with array operations will be via status and error messages channeled through the Monitor and Control System.

2.3.2 Engineers and Technicians

The ability of the Backend System to achieve and maintain real-time processing will be vitally dependent upon reliable operation and rapid diagnosis and repair of faults in the hardware and software systems. These individuals will be responsible for performing corrective and preventive maintenance along with periodic performance tests and upgrades. Engineers and technicians will need tools to inspect individual devices from remote locations.

2.3.3 Astronomer/Scientist

These individuals are primarily interested in the science that is obtained from the instrument. Their main interaction will be to provide the basic parameters of the observation and to select and provide parameters for any additional data processing beyond the Fourier transforms.

2.3.4 Software Developer

These individuals are responsible for developing the software and will interact with the system to ensure that it is functioning properly. They will need access to error and status data on an interactive basis and need to conduct debugging as well as recompilation and linking. The software developer requires remote access to the system so that troubleshooting can be accomplished away from the EVLA and during non-working hours.

2.3.5 Web User

A few authorized individuals may be allowed access to parts of the system that are usually considered restricted.

2.4 Constraints

2.4.1 Criticality of the Application

The Correlator Backend System is a critical component in the Astronomical data path. If it is unavailable, incoming astronomical data will be lost.

2.4.2 Computer Hardware Limitations

The ultimate throughput capability of the real-time data processing pipeline of the Backend System will be constrained by the computational performance limits of available computer hardware and the practical ability to configure and maintain large numbers of processors.

2.4.3 Computer Software Limitations

The ultimate throughput capability of the real-time data processing pipeline of the Backend System will be constrained by the efficiency of supporting software systems, data processing code and our ability to configure and tune them for maximum performance.
.

2.4.4 Communications Limitations

The ability to realize and maintain real-time operations is critically dependent upon the performance levels of available network systems.

2.4.5 Processing Limitations

Data processing operations performed shall be reversible. That is, uncorrected input data to a processing operation must be recoverable from the processing output. This will hold only up to the point another data processing operation is performed. Data integration is not considered to be a data processing operation. It is fundamentally irreversible once done, and no intermediate results will be retained to even partially back out an integration.

2.4.6 Reliability

The ability to maintain real-time operations over realistic extended periods of time is dependent on the mean time to failure of the hardware and software components of the computing and communications systems.

2.5 Assumptions

2.5.1 Incoming Data Stream

It is assumed that the Correlator will deliver suitably formatted network data packets to the input network of the Backend System. Lag frames will not necessarily arrive in Lag Set order. All lag frames for the same baseline will be directed to the same Backend processor. It is further assumed that the number of lags per Lag Set will always be a power of two.

The initial (for earliest science data in 2007) aggregate data volume delivered from the Correlator to the Correlator Backend shall not exceed 1.6 Gbytes/sec. This represents a 0.1 sec dump rate of 160 baseline boards of 64 correlator chips each. Each chip will produce 1024 complex lags or 2048 lag values of 8 bytes each. (160 boards) (64 chips/board) (2048 lags/chip) (8 bytes/lag) (10 /sec) = 1.6 Gbytes/sec.

An ultimate upper bound on Correlator output rate and aggregate volume is set by the fastest Correlator dump rate. This rate is currently anticipated to be 320 microseconds or one third of a millisecond. This is 300 times faster than the initial rate and will produce 300 times the aggregate data volume or about 500 Gbytes/sec.

2.5.2 Auxiliary Data

It is assumed that all auxiliary data needed for processing and formatting operations will be provided directly by the correlator or indirectly by the Monitor and Control System in a timely manner. Much of this data will originate from the Station Board CMIBS.

2.5.3 Outgoing Data Stream

The initial (for earliest science data in 2007) aggregate data volume delivered from the Correlator Backend to the e2e archive shall not exceed 25 Mbytes/sec. The present VLA system produces about 0.25 Tbytes of archived data per year, which is roughly 10% of its total capability if it were run constantly at the highest rate. Applying the same 10% factor to the EVLA will result in an archive filling rate of 75 Tbytes per year or 300 times the current VLA rate. It is assumed that the e2e System will be capable of accepting these output data rates and volumes.

Visibility data from different baselines could be processed by different Backend processors. It is expected that final assembly of all visibility data will to be performed by the e2e System.

3 Requirements

3.1 External Interface Requirements (Including Requirements imposed on External Systems)

3.1.1 Correlator to Backend Interface

.

Req. ID	Description
3.1.1.1	Lag Frames – *The BE shall receive LTA or Speed Dump Lag Frames from the Correlator.* For a detailed description of the two dump formats see Reference 8, pages 69 to 71. This will most likely be in the form of one or more frames per UDP/IP packet
3.1.1.2	Transfers – *The transfer shall take place in such a manner that at a minimum all data needed to perform any Fourier Transform shows-up on a single processor.* It is preferred that all data for the same baseline be directed to a single BE IP address.
3.1.1.3	Interface Medium – *The interface network shall be capable of a sustained aggregate transfer rate of 1.6 Gbytes/sec*. It is expected that at minimum a 1 Gbit/sec switched fiber optic network will be required.
3.1.1.4	Lag Frame Delivery – *A lag frame shall be considered to be lost if in is not received within two Correlator dump periods of its expected arrival time.* Note: It is currently expected that each UDP/IP packet will deliver one lag frame.

3.1.2 Backend to/from Monitor and Control Interface

Req. ID	Description
3.1.2.1	External Auxiliary Data (e.g., Quantizer Power Measurement Data - State Counts) - *The BE shall receive, via M&C, auxiliary data needed for real time processing such as quantizer power levels from the digitizers at the antennas and the requantizers inside the Correlator. This data shall be delivered no later than 5.0 seconds and preferably within 200 ms after the arrival on the BE of the lag data to which it applies. (See 3.3.2.4 Memory)*
3.1.2.2	Observational Mode – *The BE shall receive, via M&C, data and parameters specific to the current EVLA Observational Mode needed for processing the Correlator Lag values. New or changes to existing Observational Mode data shall be provided to the BE at least 60 seconds prior to the time it is to take effect.* This time will be needed to assure that all necessary parameters have been propagated to the DP processes and all data structure changes have been made.

3.1.2.3	Observation Parameters – *The observational Mode parameters shall include at a minimum: the number of subbands, polarization products, basebands, subarrays, lags per lag set, and antennae per subarray; the start and end time of the observation, the Correlator dump rate, the lag sample interval, the BE integration interval, the frequency ranges of the subbands, and the frequency ranges of the basebands*.
3.1.2.4	Optional Processes Parameters - *The BE shall receive, via M&C, parameters specific to any optional processes that will be applied to the lag sets before and/or after FFT*. It is expected that this information will accompany the Observation Mode information.
3.1.2.5	Meta-data – *The BE shall receive, via M&C, all meta-data necessary to format BE results for delivery to the e2e*.
3.1.2.6	Operational Status and Control – *The BE shall provide operational status data to and receive control data from the M&C System*. This includes Lag Frame destination addresses and address changes.
3.1.2.7	Error and Warning – *The BE shall provide error and warning reports to M&C as operating conditions warrant*.
3.1.2.8	Debug/Test Messages – *The BE shall provide several optionally selectable levels of printed messages detailing operational parameters at critical locations in the system*.
3.1.2.9	Interface Medium – *The interface network shall be Ethernet (IEEE 802.3 compliant) of 100 Mbits/sec or better data rate*.
3.1.2.10	Availability – *All required M&C services shall be available no later than one year prior to first science*. Earliest first science is currently anticipated in Q2, 2007.

3.1.3 Backend to e2e Interface

Req. ID	Description
3.1.3.1	Formatted Output – *The BE shall deliver formatted final results to the e2e System. The BE shall produce all data needed by the e2e System for archiving and further processing*. The output is currently expected to be in a form compatible with AIPS++ Measurement Sets.
3.1.3.2	Interface Medium – *Interface communication media shall be capable of sustaining a minimum aggregate transfer rate of 25 Mbytes /sec*.
3.1.3.3	Reliability – *The e2e Archive shall not be offline for more than 12 continuous observing hours or a total of 18 observing hours during any 48-hour period*. These times are based upon planned local BE node storage capacity. (See 3.3.2.5 Storage)
3.1.3.4	Availability - *All required e2e services shall be available no later than one year prior to first science*. Earliest first science is currently anticipated in Q2, 2007.

3.2 *Functional Requirements*

3.2.1 Information and data flows

Req. ID	Description
3.2.1.1	Monitor and Control System – *The BE shall acknowledge receipt of all data received from M&C.*
3.2.1.2	Correlator System – *The BE shall notify M&C of any detected interruptions of data delivery from the Correlator.*
3.2.1.3	e2e – *The BE shall verify successful delivery of output to the e2e.*
3.2.1.4	Internal Data – *The BE shall guarantee safe delivery of all internal messages.*
3.2.1.5	Lag Frames – *The BE shall be able to handle lag frames of less than 128 values.*
3.2.1.6	Lag Sets - *The BE shall be able to handle lag sets up to a maximum size of 262,144 values (256K lags).*
3.2.1.7	Correlator dump rate – *The BE shall be able to handle different dump rates for different basebands.* It is understood that the dump rates will always vary harmonically.
3.2.1.8	Basebands – *The BE shall be able to handle up to eight basebands in up to four pairs.*
3.2.1.9	Baseband Size – *The BE shall be able to handle independent (i.e., non-correlated) basebands of differing numbers of sub-bands.*
3.2.1.10	Sub-bands – *The Be shall be able to handle up to sixteen sub-bands per baseband.*
3.2.1.11	Sub-band Size – *The BE shall be able to handle independent (i.e., non-correlated) sub-bands of differing numbers of lags in the same baseband.*
3.2.1.12	Sub-arrays – *The BE shall be able to handle multiple sub-arrays up to a maximum equal to the number of antennas in the array.*
3.2.1.13	Sub-array size – *The BE shall be able to handle sub-arrays of from one to the maximum number of antennas in the array.*
3.2.1.14	Baselines – *The BE shall be able to handle a total number of baselines across all sub-arrays equal to the maximum number of cross-correlations and autocorrelations produced by the array.*
3.2.1.15	Polarization Products – *The BE shall be able to handle 1, 2 or 4 polarization products independently per baseband pair.*

3.2.2 Process Descriptions

Req. ID	Description
3.2.2.1	Data Receive – *The BE shall receive incoming data packets from the Correlator to Backend network interface.* This network is a part of the BE System.
3.2.2.2	Verify Receive – *The BE shall verify the successful receipt of incoming data from the Correlator.* This includes checking for receive errors and determining that all expected data was received, accumulation of error statistics and comparison against tolerances, and reporting of all out of tolerance conditions.

3.2.2.3	Input Data Management - *The BE shall store input data records in a memory buffer and track buffer locations of all input data until data processing is complete*. Report any buffer overflow conditions.
3.2.2.4	Processing Management – *The BE shall respond to incoming correlator mode changes, user optional processing sequence and/or parameter changes, and other external inputs that affect the data processing pipeline*. Update internal parameter tables and synchronize data processing pipeline with new operational conditions.
3.2.2.5	Time Series Assembly – *The BE shall assemble the received input data into continuous time series (lag sets)*.
3.2.2.6	Data Integrity Verification – *The BE shall ensure that time series data is correctly ordered and contains valid data values along its entire extent*. Compare against tolerances and report all out of tolerance conditions.
3.2.2.7	Data Invalid – *The BE shall replace all invalid data with zero values*.
3.2.2.8	Data Invalid Count – *The BE shall keep track of data invalids*.
3.2.2.9	Normalization – *The BE shall be able to apply normalizations based on reported (from the Correlator via the lag frames) data invalid counts*.
3.2.2.10	Coarse Quantization Correction - *The BE shall be able to apply corrections based on state count and/or quantizer power measurement data*. This is the VanVleck correction
3.2.2.11	Time Stamp Adjustment – *The BE shall be able to make time stamp adjustments as required by the observational mode and correlator output parameters*. This may arise when recirculation is used.
3.2.2.12	Windowing – *The BE shall be able to perform windowing operations prior and subsequent to Fourier Transform*. This will be needed for narrow band RFI mitigation. Post Fourier Transform windowing will be applied as a convolution.
3.2.2.13	Time Domain Processing – *The BE shall be able to apply user selected time domain processes*. These processes should be constructed to be chainable (output of any time domain process can be piped to input of any other, including replica of self and Fourier Transform) and repeatable in the chain. No Optional time domain processes have as yet been proposed.
3.2.2.14	Fourier Transform Processing – *The BE shall be able to Fourier Transform the lag set time series*. A power-of-two complex-to-complex Fast Fourier Transform with retention of all output positive and negative frequencies will be used. This process must be able to accept as input the output of any of the time domain processes.
3.2.2.15	Frequency Domain Processing – *The BE shall be able to apply user selected frequency domain processes*. These processes should be constructed to be chainable (output of Fourier Transform and any frequency domain process can be piped to input of any frequency domain process including replica of self) and repeatable in the chain. No frequency domain processes have as yet been proposed.
3.2.2.16	Integration – *The BE shall be able to sum the frequency domain, spectral results*. The amount (time duration) of summation will be controlled by an observational mode parameter obtained via M&C. *The BE shall keep track of the number of samples/dumps integrated in each spectral channel.* The summation will occur after all optional frequency domain processing, or if none, after the Fourier Transform. Integration for long periods of time is what will throttle the output of the Correlator to a rate manageable by the e2e.
3.2.2.17	Output Formatting – *The BE shall combine the finished spectra with meta- and auxiliary data to form suitably formatted output data sets*. AIPS++ Measurement Sets or compatible subsets and/or fragments thereof are the expected entities.

3.2.2.18	Output Data Management – *The BE shall store formatted output data records in a memory buffer with backup disk buffering*. Store data ready for transmission to the e2e System until successful transfer has occurred. Report any errors and buffer overflow conditions that occur.
3.2.2.19	Data Send – *The BE shall send output data to the e2e System.*
3.2.2.20	Send Verify – *The BE shall verify that all sent data was successfully received*. Report all errors.
3.2.2.21	Monitor I/O Performance – *The BE shall monitor data transfer rates from the Correlator and to the e2e*. Accumulate data transfer statistics and compare against tolerances. Report all out of tolerance conditions.
3.2.2.22	Monitor Compute Performance – *The BE shall monitor the overall data processing rate*. Compare against tolerances and report all out of tolerance conditions.
3.2.2.23	Monitor Compute Errors – *The BE shall trap, flag and repair inf's, NaN's, underflows, overflows and other computation errors*. Accumulate computation error statistics and compare against tolerances. Report all out of tolerance conditions.
3.2.2.24	Monitor Processes – *The BE shall periodically or upon request check PID's and assure that all started tasks are alive and running*. Report missing, stopped, defunct and other damaged processes.
3.2.2.25	Monitor Processors – *The BE shall periodically or upon request check Backend physical processors and assure that all needed processors are alive and responding.* Report all crashed, stopped, or unresponsive processors.
3.2.2.26	Monitor Networks – *The BE shall periodically or upon request check all Backend internal networks and assure that all communication connections are intact and functioning*. Report all non-functioning components.
3.2.2.27	Start Process – *The BE shall be able to initiate a processing task on any Backend processor.*
3.2.2.28	Stop Process – *The BE shall be able to signal a kill for any Backend process*.
3.2.2.29	Alter Priority – *The BE shall be able to alter the priority of any of the BE tasks.*
3.2.2.30	Reboot Processor – *The BE shall be able to initiate a reboot of any Backend physical processor.*
3.2.2.31	Reboot network – *The BE shall be able to initiate a reboot of any internal network*.
3.2.2.32	Offload – *The BE shall be able to redistribute internal workload among its processors.* This may involve change of destination IP address(es) for the Correlator network.
3.2.2.33	General – *BE processes shall not violate archive data requirements. All processes shall be reversible*; the raw unconverted input always being recoverable from the output.

3.2.3 Data Construct Specifications

Req. ID	Description
3.2.3.1	Input Data Queue – a memory buffer of lag frames. Data entry status queue to track each record in the buffer. The lag frames will contain all information necessary to properly assemble complete lag sets.
3.2.3.2	Output Data Queue – a memory buffer plus backup disk storage of all

	processed spectra. These will be suitably formatted prior to transfer to the e2e. Data entry status queue to track each record in the buffer.
3.2.3.3	Processing Parameters – names, position(s) in sequence, and adjustable parameters for all fixed and user selectable processing pipeline applications.
3.2.3.4	Processing flags – a table of flags needed to identify various internal conditions relating to error response and processing state.
3.2.3.5	Metadata – All internally and externally generated data about the processed time series and spectra including invalid data flags, processes applied, coordinates, etc.
3.2.3.6	Error Report – error number (translatable into text error message), error source, error rates (as applicable), and time stamp.
3.2.3.7	Warning Report – warning number (translatable into text warning message), warning source, warning rates (as applicable), and time stamp.
3.2.3.8	Failure Report – internal system component (e.g., disk drive, processor, process, and network) failure number (translatable into text error message) and time stamp.
3.2.3.9	Recovery Report – internal system component (process, processor, network) recovery action result.
3.2.3.10	Status Report – internal system component (process, processor, network) functional state.

3.3 Performance Requirements

3.3.1 General

Req. ID	Description
3.3.1.1	Data Integrity – *the Backend System shall maintain input data fidelity and dynamic range across all processing, manipulation and I/O functions*.
3.3.1.2	Error Handling – *the system shall be capable of flagging and marking corrupted data segments and proceeding without interruption or effect on other data.* This includes, but is not limited to, partial data, zero data, underflows, overflows, infinities, and NaN's whether obtained on input or arising during processing.

3.3.2 Hardware

Req. ID	Description
3.3.2.1	Input – *The BE System shall be capable of accepting an aggregate data input stream from the Correlator of a minimum of 1.6 Gbytes/sec.* This must be done simultaneously with the output stream, but not necessarily over the same interconnects. This is an initial deployment specification and will be increased over time.
3.3.2.2	Output – *The BE System shall be capable of delivering an aggregate output data stream to the e2e System of a minimum of 25 MBytes/sec.* This includes resends and simultaneous transfer of data stored due to a previous e2e connection outage. This must be done simultaneously with the output stream, but not necessarily over the same interconnects. This is an initial deployment specification and will be increased over time.
3.3.2.3	CPU – *The total processor capability of the BE System shall be*

	(combination of numbers of processors and individual processor speed) sufficient to accomplish all processing tasks while avoiding loss or delay on the input and output data streams.
3.3.2.4	Memory – *The BE System shall have sufficient memory with sufficient access speed to buffer 60 seconds of input data and accomplish all processing tasks while avoiding loss or delay on the input and output data streams.* At 26.5 Mbytes/sec of input per BE node and a total of 62 data processing nodes (for an aggregate of 1.6 Gbytes/sec), this implies a minimum of 1.5 Gbytes of memory per BE processing node.
3.3.2.5	Storage – *The BE System shall have sufficient disk storage with sufficient access speed to meet short duration Correlator bursting demands plus a standby reserve to hold at least 12 hours of output data.* At 25 Mbytes/sec, this implies a minimum of 40 Gbytes of disk storage per BE processing node.

3.3.3 Software

Req. ID	Description
3.3.3.1	Applications – *all math/science application software shall take optimal advantage of all language, compiler, and system computational features and resources to reduce run times to the minimum practical level. They shall be coded in such a manner as to minimize the possibility of floating point exceptions during processing.*
3.3.3.2	Management – *all management software functions shall take optimal advantage of all language, compiler and system features and resources to reduce overheads to the minimum practical level.*
3.3.3.2	I/O – *all input and output, and storage and retrieval operations shall take optimal advantage of all system resources to reduce overhead and latency to the minimal practical level.*
3.3.3.4	Processing – *all data processing functions shall be chainable (outputs pipeable to inputs) and repeatable in the processing pipeline in cases where this makes computational sense.*
3.3.3.5	General - *Operating system, message passing and other middle-ware, and programming language(s) used shall follow industry standards and be commonly available and widely used. Availability of source code for the OS will be very important.*

3.4 Reliability/Availability

Req. ID	Description
3.4.1	Auto-correction – *the Backend System shall be self-monitoring.* It will be capable of detecting, reporting on and automatically taking action to remedy or lessen the impact of, at a minimum, the following types of abnormal conditions: processor hardware failure, operating system hangs or crashes, computational performance below minimum specifications, computational error rates above maximum specification, internal communications failures, and external (with the Correlator and e2e) communications disruptions.
3.4.2	Software – *the software part of the system shall be able to perform without*

	total system restart due to internal failure between system maintenance windows.
3.4.3	Hardware *– the hardware part of the system shall be able to perform indefinitely without complete loss of service, except in the event of total failure of primary and backup power.*
3.4.4	Network *– the internal BE network(s) and the networks connecting the BE to the Correlator, M&C, and e2e Systems shall be considered to be out of spec if packet loss exceeds 0.1%.*
3.4.4	Correlator mode changes *– the system shall be capable of responding in a loss-less manner to I/O and processing changes arising from Correlator mode changes.*
3.4.5	Loss of e2e *– the system shall continue to operate in a loss-less manner in the event of a temporary loss of availability of the e2e System.*
3.4.6	Loss of Correlator *– the system shall be able to complete processing of all onboard data, deliver the results to the e2e and maintain availability for immediate resumption of operations once Correlator access is restored.*
3.4.7	Loss of M&C *– the system shall continue to operate during the absence of the M&C System until the first encounter of unavailable critical auxiliary data*. The system will cache a predetermined amount of correlator data after the first encounter of unavailable critical data and complete all requested operations on cached data if the unavailable critical data is ultimately obtained.
3.4.8	Standby Mode *– the system shall be able to sit at idle and resume operations with minimal delay.*

3.5 Serviceability

Req. ID	Description
3.5.1	Hardware Accessibility *– all system processing and interconnect hardware shall be readily accessible for maintenance, repair, replacement and/or reconfiguration.*
3.5.2	Software Accessibility *– all systems and application source code shall be available to or on the systems that execute it.*
3.5.3	Debugging *– all software application modules shall be debuggable.*
3.5.4	Processes *– all software processes shall be killable, restartable, debuggable and testable without affecting normal operations.*

3.6 Maintainability

Req. ID	Description
3.6.1	Software tools *– software tools and pre-built applications that do not have source code available shall come with a complete diagnostic package and customer support.*
3.6.2	Operating Systems *– operating system software shall either have source code available or come with sufficient diagnostics and customer support.*

3.7 Scalability

Req. ID	Description
3.7.1	Hardware – *I/O, communications, and processing hardware shall be easily expandable, reconfigureable, augmentable and replaceable to meet increasing data transfer and processing demands imposed by EVLA science, Correlator changes, and availability of new technology.*
3.7.2	Transparency – *3.7.1, above, shall be accomplished in manner that is transparent to processing, communications and I/O software functions with the possible exception of recompilation of executables.*
3.7.3	Seamlessness – *3.7.1, above, shall be accomplished in a manner that is seamless, in that it does not affect hardware modules or software functionality that it meets at interfaces.*
3.7.4	Performance – *the Backend system shall be scaleable to an extent limited only by hardware technology and budget constraints. An ultimately upper bound is set by the Correlator upper limit of three Gbytes per second per Correlator output channel (baseline board) in real-time.*

3.8 Security

The Backend System needs a robust security mechanism in place so that unauthorized users are not allowed access. Authorized users are expected to be restricted to software and hardware development, testing, maintenance and operations personnel.

All users of the Backend System must be uniquely identified. This could be done via a username and associated password scheme that would authenticate and authorize the user access to the system and, if applicable, grant the user access to restricted or controlled parts of the system. If a user cannot be identified, they will not be given access. In order to monitor all past access to the system, all attempts to access the system should be logged.

Users' needs and expectations from the system will be different. Systems operations should be given unrestricted access to all aspects of the system and should have the authority to grant and revoke privileges on a per-user basis. Development, testing and maintenance personnel, on the other hand, require access to some parts of the system, but not all, indicating that an access level is needed that allows privileges to be granted on a per-user and what-do-you-need-to-do basis.

Req. ID	Description
3.8.1	*All users of the system shall login using some form of unique identification.* (e.g., username and password)
3.8.2	*All login attempts shall be done in a secure manner.* (e.g., encrypted passwords)
3.8.3	*A system administrator shall have unrestricted access to all aspects of the system.*
3.8.4	*Each user shall have a set of system access properties that defines the user's privileges within the system.* (e.g., the subsystems a user may control or system tools the user may access).
3.8.5	*The administrator shall have the ability to create and add a new user to*

	the system.
3.8.6	*The administrator shall have the ability to remove a user from the system.*
3.8.7	*The administrator shall have the ability to edit a user's system access properties.*
3.8.8	*The administrator shall have the ability to block all access to the system for all users or selectively by user.* (All blocked users with active sessions shall automatically be logged off.)

3.9 Installation and Upgrades

Req. ID	Description
3.9.1	Operations Activities – *the system shall continue operations, although not necessarily at full capacity, on all unaffected resources during partial shutdowns for maintenance, repair and/or upgrade.*
3.9.2	Test Mode – *the system shall be able to handle non-real-time operations in a transparent fashion (i.e., as if real-time).* Note: non-real-time refers to input data from a source other than the Correlator.
3.9.3	Replaceability –*modular design principles shall be employed to the maximum extent possible. Maximal practical use of available "hot-swappable" devices and components shall be made.*

3.10 Documentation

Req. ID	Description
3.10.1	Hardware – *complete and comprehensible hardware systems specifications and configuration information shall be readily available.*
3.10.2	Software Coding Practices– *software system and application code shall be well documented and written in a generally familiar language or languages (preferably not more than two). Software shall be written in a style that is easily readable and using practices that allow for minimal confusion.*

4 Functional Specifications

This section presents a plan for implementing the functionality, as defined by the requirements of the previous sections, of the Correlator Backend System. Data I/O and processing throughput requirements dictate the use of a fairly substantial computer system for the Backend connected by high speed and high capacity networks to the data sources and sinks. In addition, scalability, serviceability and maintainability requirements along with the overall structure of the main data elements (i.e., correlation lags organized by baselines that can be handled completely independently from one another in a 100% data parallel approach) have lead to the selection of a cluster of networked commodity PC's (Intel or Intel clones) as the primary platform for the Correlator Backend. Use of the distributed memory, cluster approach for system hardware has the additional benefit of being both low cost and familiar to the development staff. The additional use of Linux as the operating system compliments this choice of platform. Furthermore, the functional requirements of internal monitor and control, data input, data handling and organizing activities, numerical data processing and data output lead naturally to seven main software executables operating in a functionally parallel mode on the Backend Cluster. Figure 1 diagrams the mapping of these executables onto the cluster processing nodes. Each node is expected to be a multi-CPU Intel type PC with significant amounts of memory and local disk storage.

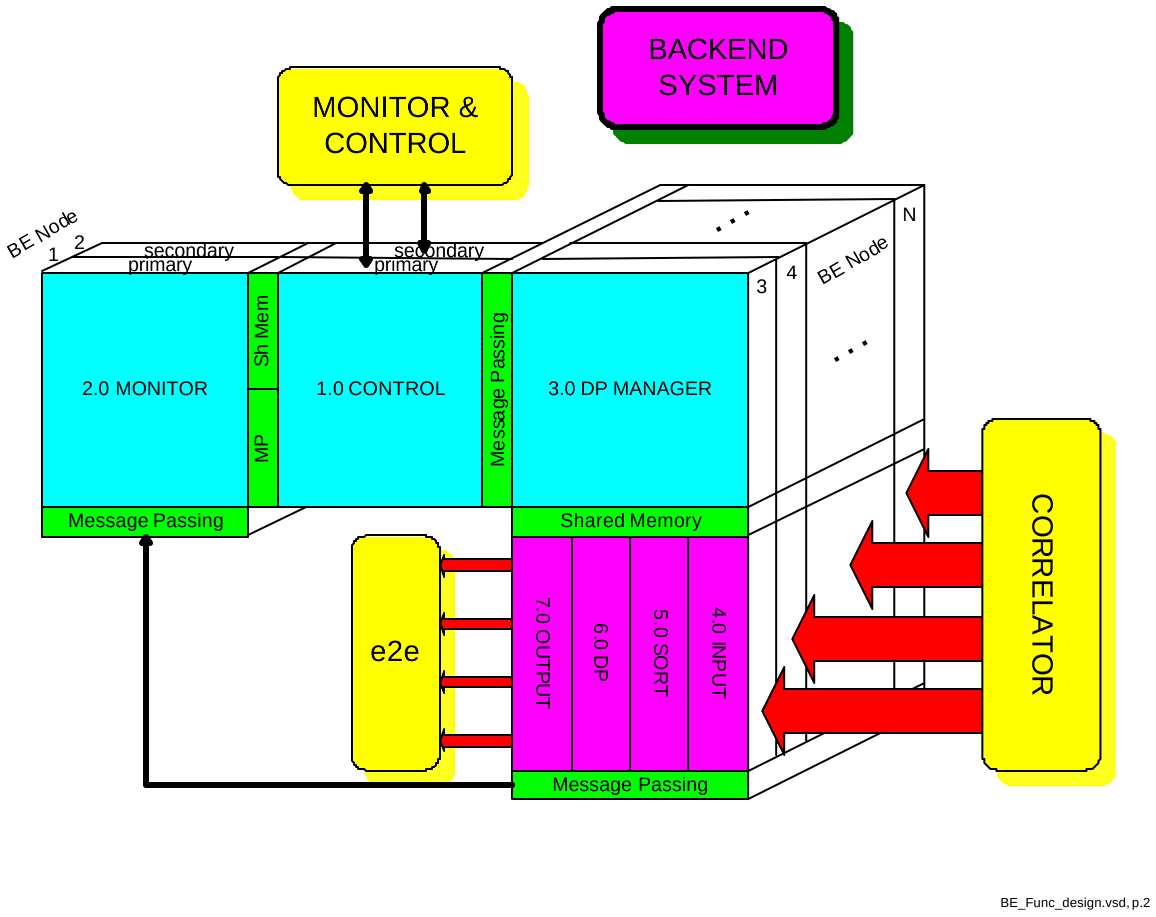


Figure 2. Correlator Backend System block diagram with software tasks mapped to hardware components. Internal communications layers (message passing and shared memory) are shown in green. External systems (Monitor & Control, Correlator and e2e) are marked in yellow. Width of lines to external systems indicates relative data volume flows across the networked interfaces.

Only one copy of the Monitor and Control executables is needed for the running system. They will run together on one node with a backup pair running on a second node. Nothing else will run on the Monitor and Control node and its backup/standby node. All other nodes will run one copy each of the data I/O, management and processing executables. Data parallelism was chosen as the method of dividing work among the nodes. Each node will do all processing for a unique subset of baselines. This approach minimizes the amount of inter-processor communication on the Backend Cluster since all data for a given baseline is sent directly by the WIDAR Correlator to the node that will process it. Once the baseline data is on the node, no further transfer is needed until the final output is consolidated for archival after processing is completed. On a node, functional parallelism was chosen to distribute the various processing tasks in time and among the multiple CPUs of the node. The names of the executables reflect the highest level of this distribution of effort. Functional parallelismat this level allows isolation of the Input and Output tasks that are expected to demand large amounts of communications services from the Data Sorting and Processing tasks that are expected to demand large amounts of CPU time. In this way the four major processing tasks compete among themselves for CPU time and overhead services. This places the burden of load balancing on the operating system rather than the application software. Communication among the various Backend executables is via message passing middle-ware or shared memory when copies of the executables reside on the same system node.

Correlator Backend execution is initiated by starting one copy of the Control process on one Backend Cluster node. All other processes are started internally in a cascaded fashion across the Backend PC Cluster. All processes including Control will self terminate when given the shutdown message, otherwise execution will continue indefinitely within the limits of hardware and operating system dependability.

The Correlator Backend has its own internal Monitor and Control functions. These will reside on a single node of the PC Cluster and handle all communication with the external environment, the internal data handling processes, plus a number of system wide internal functions. Communications with external systems will be via the main EVLA M&C System network. One copy each of the Monitor and Control processes is needed. Since the Monitor and Control processes represent single points of failure, a second copy of each on another node will shadow the primary ones and be available to takeover in case of failure.

All remaining nodes on the Correlator Backend Cluster will be available for processing of lag data from the WIDAR Correlator. Each node will have the same set of five executables operating in concert. The Manager process is responsible for receiving parameters from the Control function and setting up appropriate tables and data structures needed by the other data handling processes. These additional processes form a data processing pipeline and include Input, Sorting, Data Processing and Output. Lag data from the WIDAR Correlator is handled in sequence by these four processes resulting in final spectral channel output. Communication among the five data handling processes running on a given cluster node is done via shared memory to achieve maximal data access and communication rates with minimal latency. Communication with the Monitor and Control processes is via message passing.

Data will be received from the WIDAR Correlator via a dedicated high speed, switched network connecting each data processing node to one or more Correlator Baseline Boards. Data will be sent to the Archive and e2e System components via a second dedicated high speed, switched network. Each Correlator Backend Data Processing node will have its own output path into this network.

4.1 Software Components

The Backend System software components will be described using a series of flow charts. The following key should be used while examining the charts.

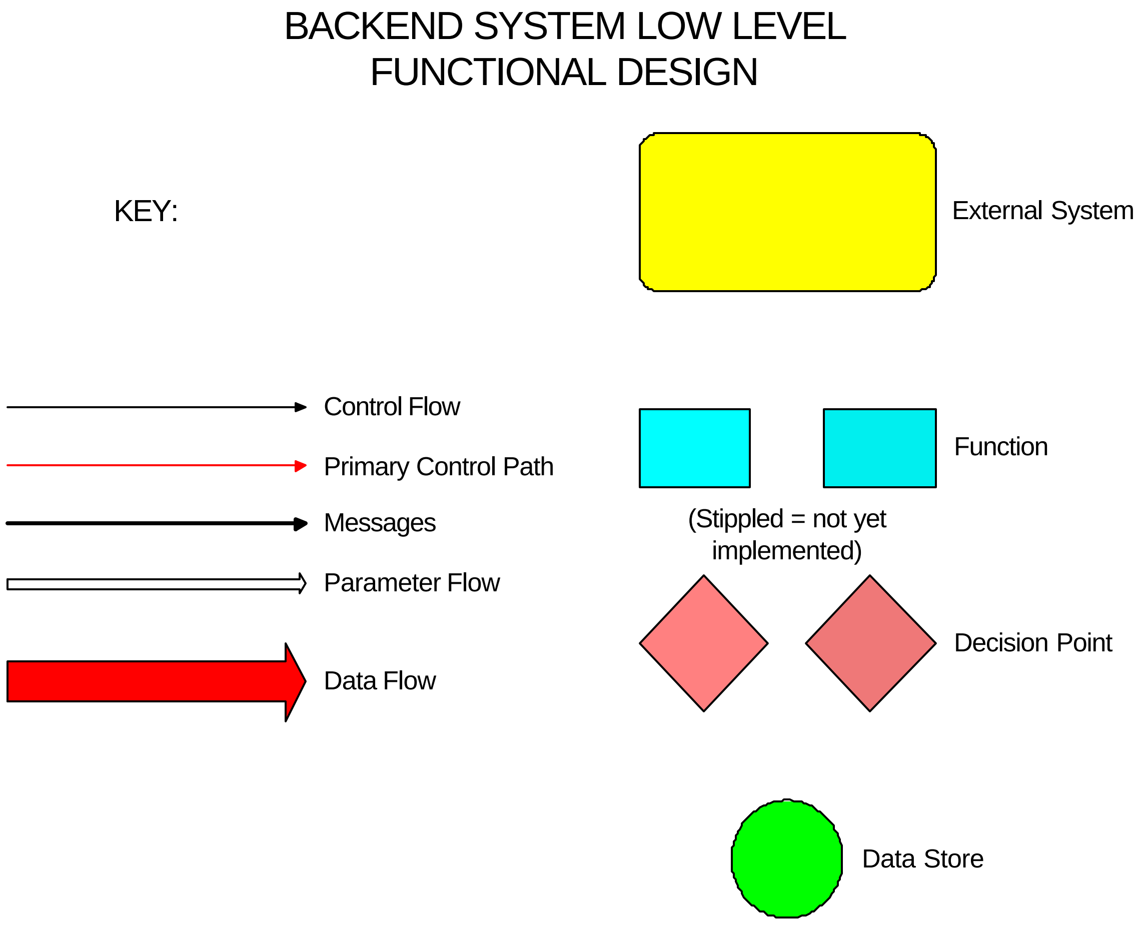


Figure 3. Software flow chart key.

4.1.1 Control Process

The Control process is the first and only Backend software executable that is started by the external environment. It starts the execution of the Monitor process, and then starts the execution of the Manager processes on all of the Backend cluster nodes that will handle Correlator lag frames.

The Control process acts as an intermediary between the Correlator Backend and the external environment. It receives all incoming messages and data other than WIDAR Correlator data frames and forwards them to the appropriate destination within the Backend. It also collects all internally generated messages and sends them out across the external interface. The Control task remains active until the Monitor task has shut itself down.

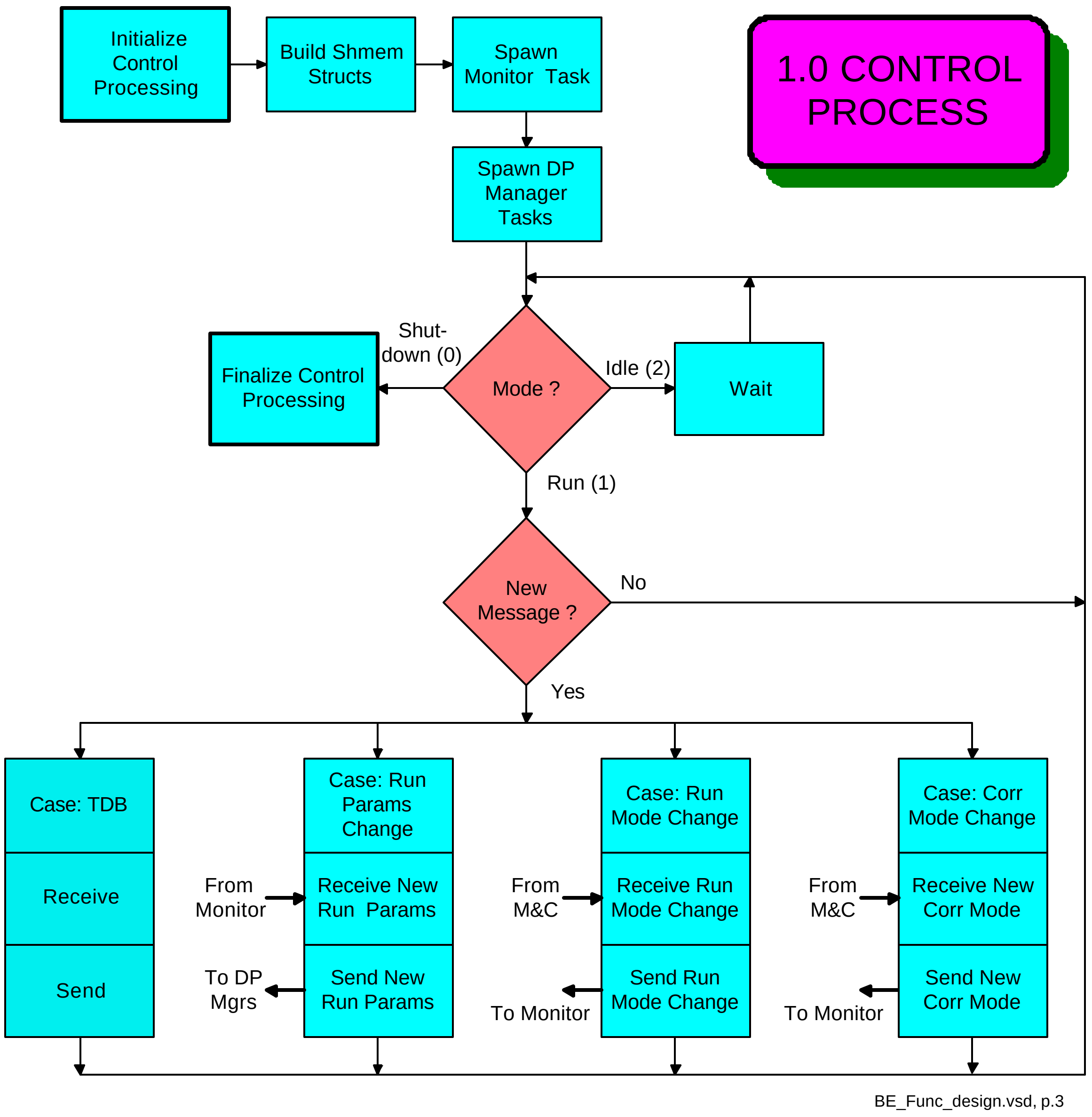


Figure 4. Control process flow chart.

4.1.2 Monitor Process

The Monitor process is started by the Contorl process. It maintains a catalogue of all data handling processes versus the nodes they are stared on as well as the apportionment of baselines among the nodes. The main job of the Monitor process is to keep track of the Backend hardware and software environment. It does periodic checks on the operational status of the cluster network connections and processor states. It also monitors the state of all software processes and receives periodic reports from the lag data handling processes containing performance and status information. It will have the ability to act upon certain failure modes and report a variety of error and warning codes to the external environment. The Monitor process will remain active until all the DP Manager processes have shut themselves down.

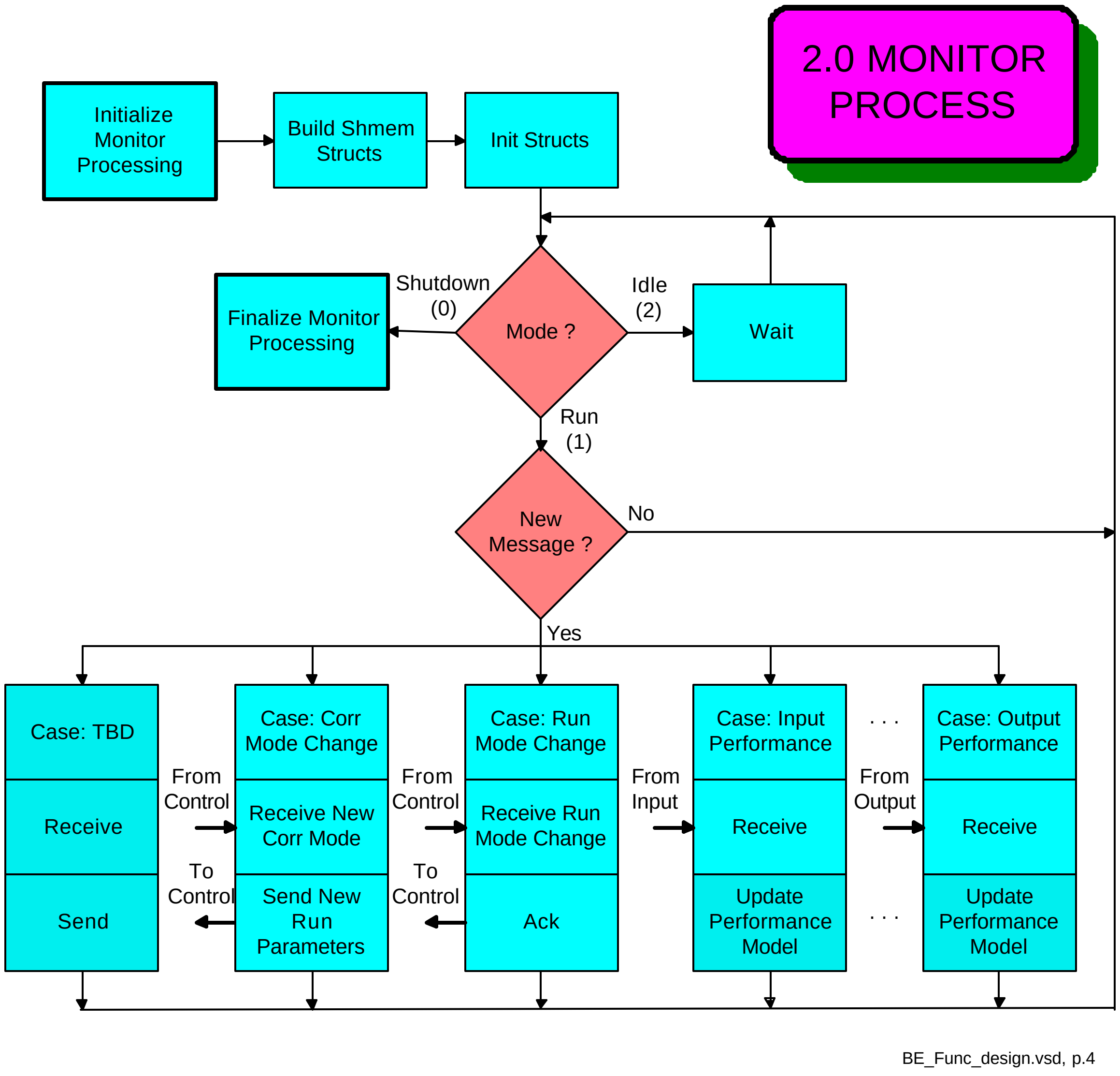


Figure 5. Monitor process flow chart.

4.1.3 DP Manager Process

The DP Manager processes are started by the Control Process on each of the Backend Cluster nodes that will handle Correlator lag data frames. They in turn each start Input, Sorting, Data Processing, and Output tasks on their own nodes. All communication between the DP Manager and these four lag data handling processes is via shared memory. The DP Manager process initializes the share memory areas and communicates address information to the four data handling processes. A given cluster node and its DP Manager, Input, Sorting, DP, and Output processes will handle all data for a subset of baselines. That is, all data for a given baseline will be sent from the WIDAR Correlator to only one Backend data processing node. A given node will handle several baselines at once. The DP Manager task will remain active until it's four processing tasks have shut themselves down.

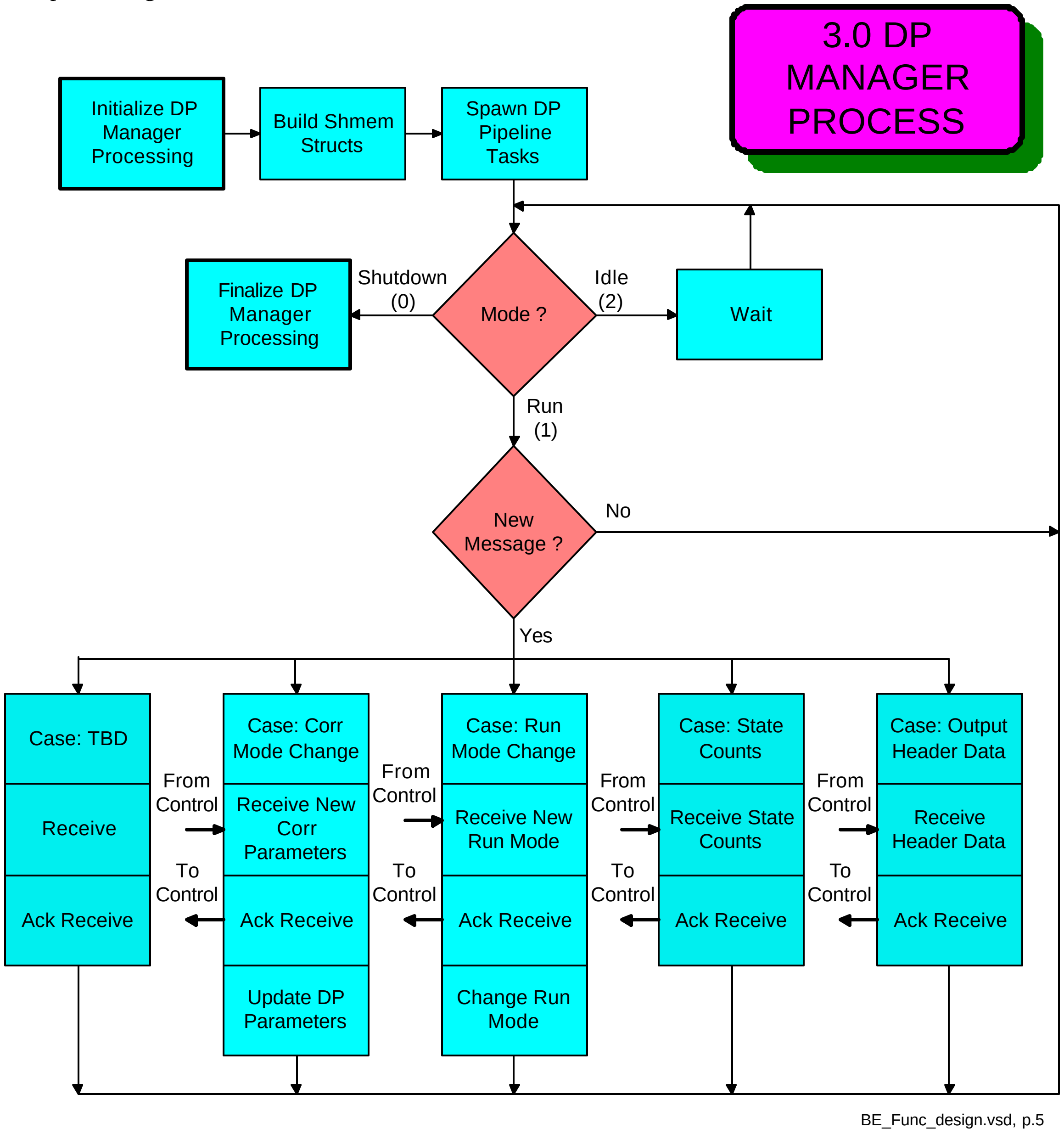


Figure 6. Data Processing Manager flow chart.

4.1.4 Input Process

The Input process receives WIDAR Correlator lag data frames via UDP packets over a dedicated, high-speed switched network. It deposits them in the order received into a set of data buffers called the Lag Frame Store. After a buffer is filled, UDP receives are briefly halted to check for messages and send a performance report, UDP receives are then resumed. The overall size of the storage space, as well as the size of a buffer can be set by control parameters. A list is maintained of the status of each of the buffers.

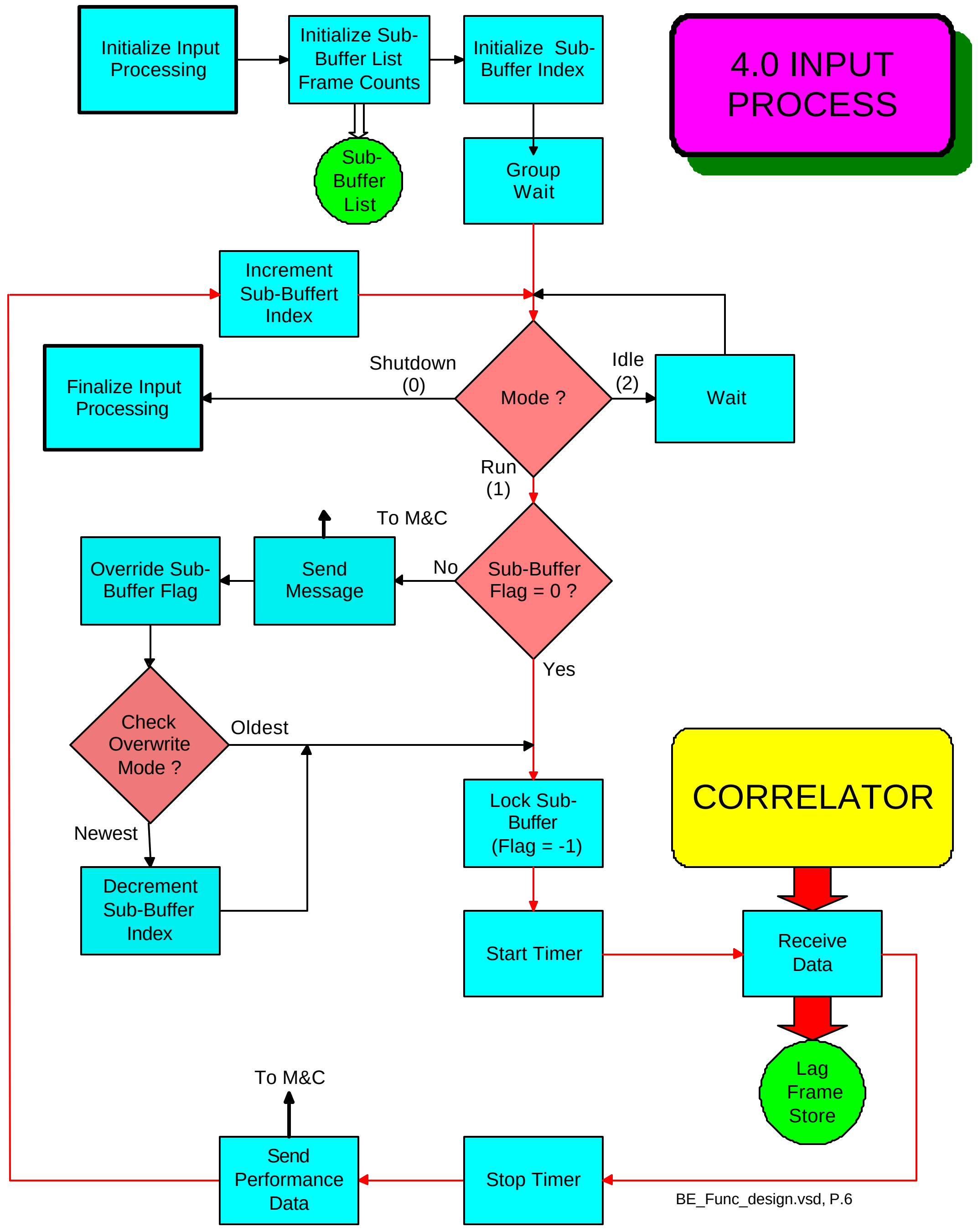


Figure 7. Input process flow chart.

4.1.5 Sorting Process

The Sorting process sequentially reads lag fra mes deposited in the Lag Frame Store by the Input process. It examines parity and header data and performs a number of validity checks. Frames that are determined to be invalid are purged from the Lag Frame Store. For valid frames header parameters are used to calculate an address in the Lag Set Store data structure. The address of the storage location of the frame data in the Lag Frame Store is placed in the appropriate address slot for the lag set of the frame (see Lag Set Table entry data structure in Section 4.2.3.). Each lag set in the Lag Set Store has several fields of header data that are set as the frame addresses are accumulated. The Lag Set Store is organized in correct time order. Once all frames for a lag set have been accumulated, the lag set is marked as ready for processing by setting a value in the header. Each time a new Backend integration interval starting point is encountered, all incomplete lag sets up to the most recent Correlator dump are marked completed and ready for processing. It is assumed that a missing frame is lost for good if more than one Correlator dump has occurred since the dump to which it belonged.

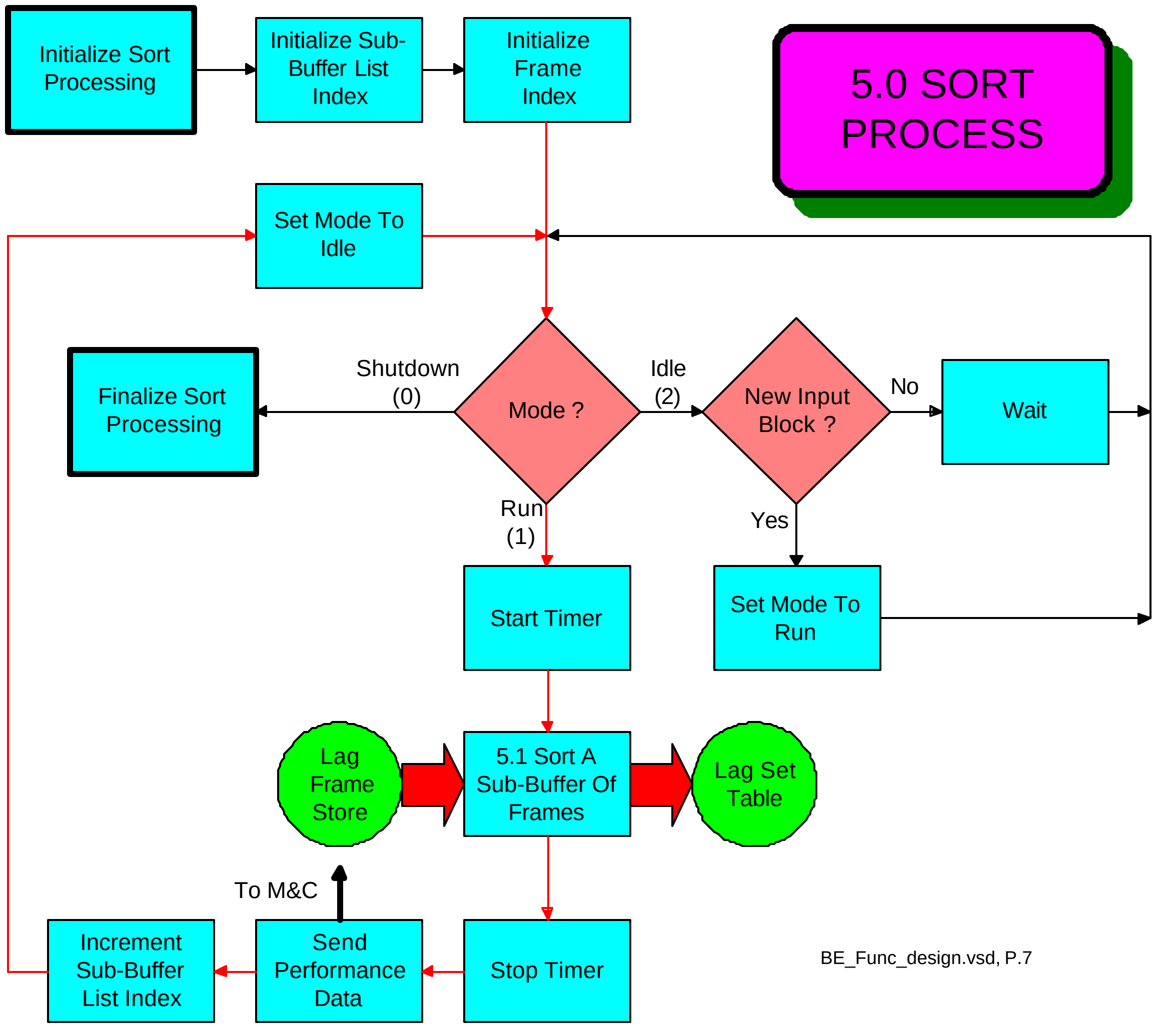


Figure 8. Sort process flow chart.

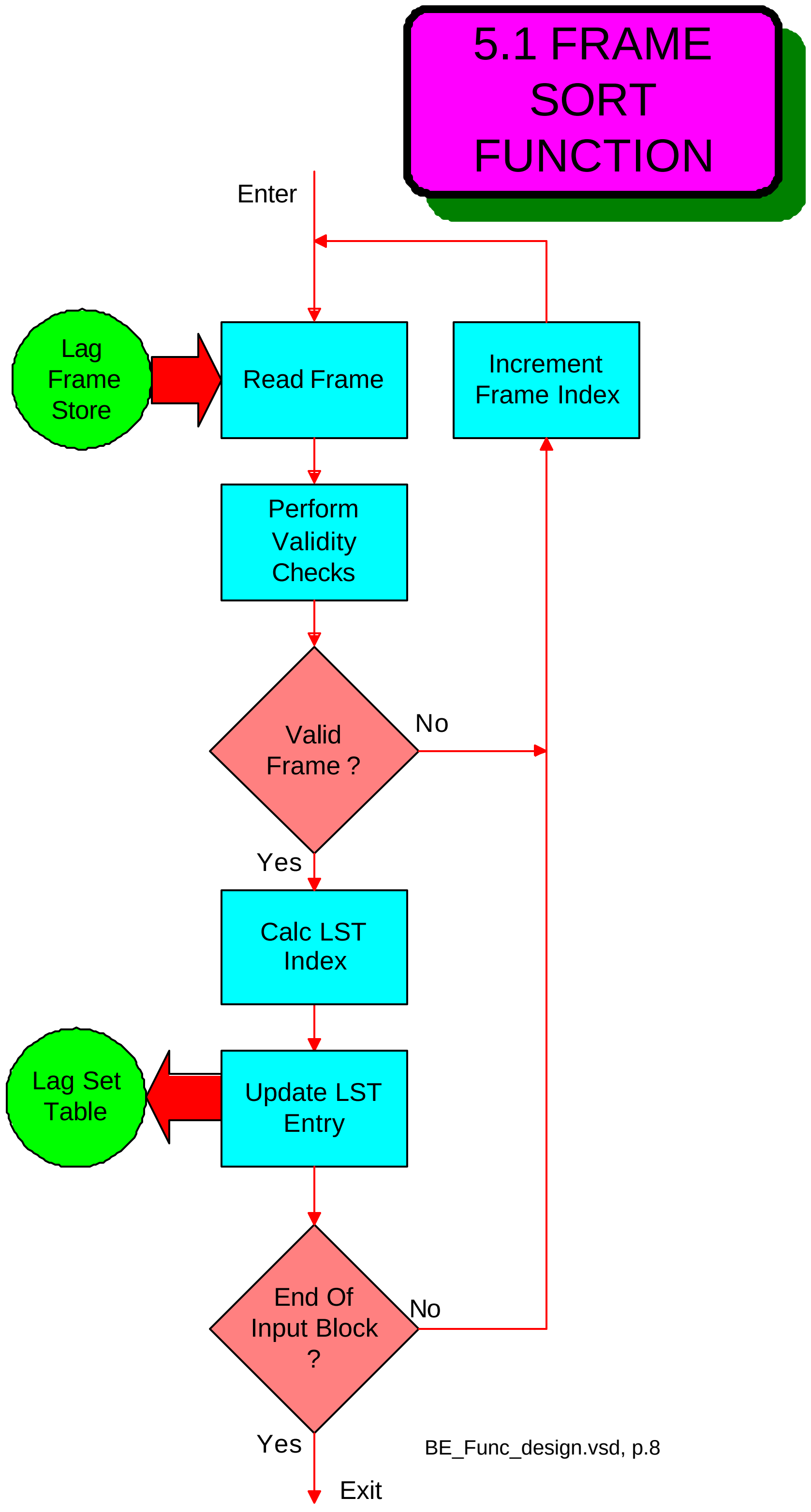


Figure 9. Flow chart of the Frame Sort function within the Sort process.

4.1.6 Data Processing Process

The DP or Data Processing process sequentially accesses the lag sets in the Lag Set Store. It must wait for the Sorting task if an incomplete entry is encountered. Each Lag Set Store entry is read and the lag frame addresses are used to pull the frame data from the Lag Frame Store and assemble the lags in proper order. The lags are Fourier Transformed and summed into the appropriate entry in the Integrated Spectra Store. A number of optional signal processing functions will be available for application either before or after Fourier Transform or after integration. These are detailed in the requirements section above. Completed integrations are marked as ready for output by setting the appropriate field in the companion Integrated Spectra Header buffer.

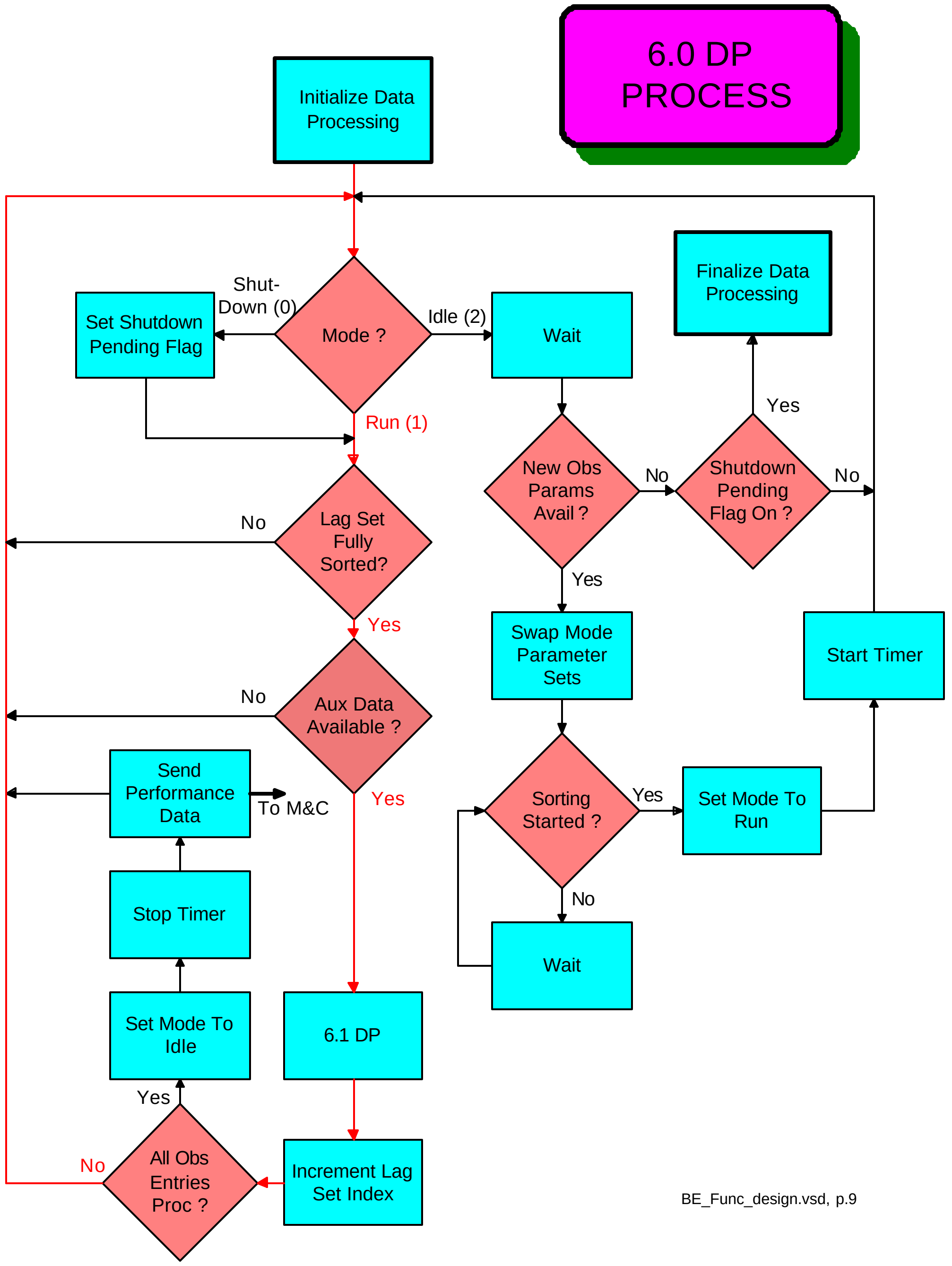


Figure 10. DP process flow chart.

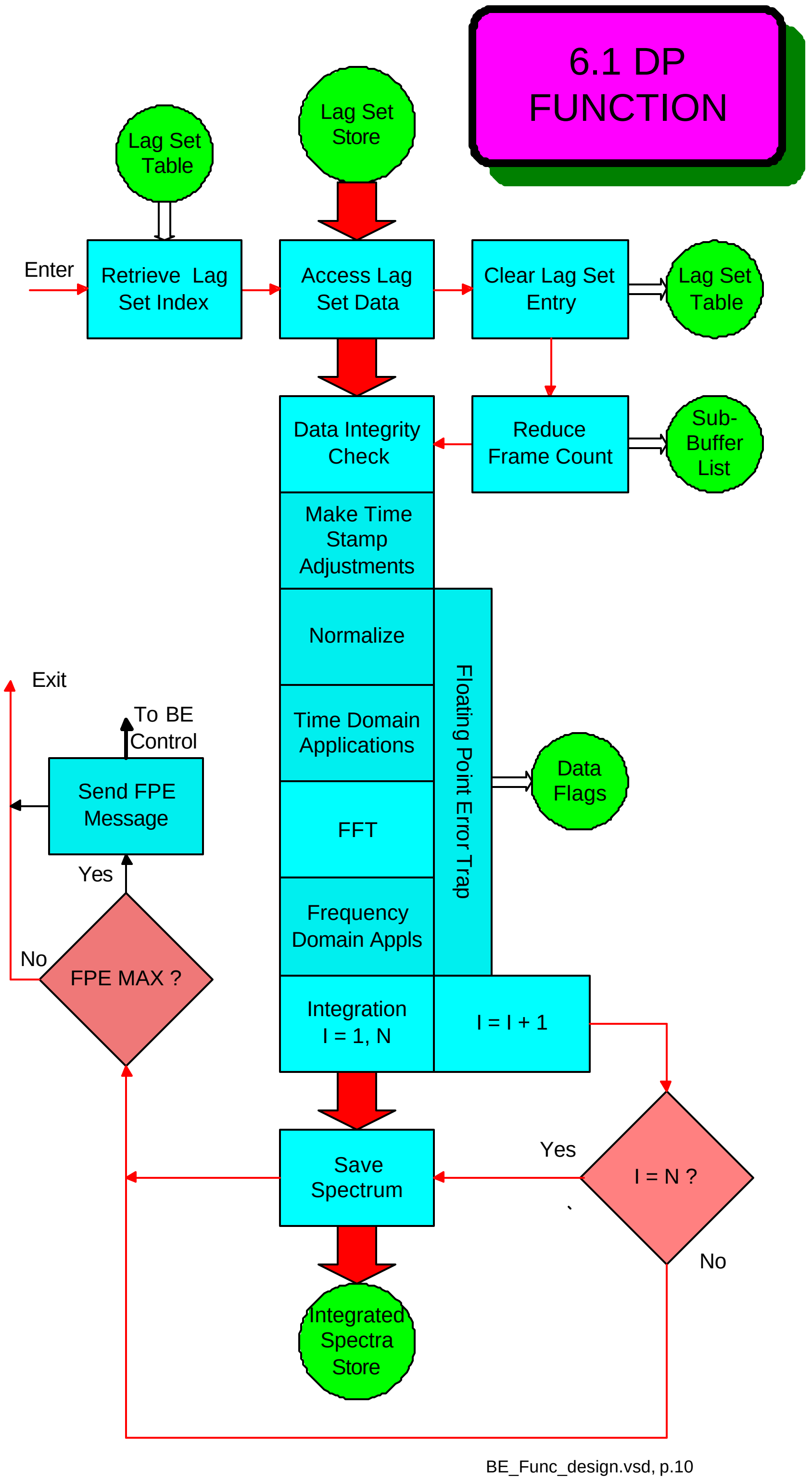


Figure 11. Flow chart of the DP function within the DP process.

4.1.7 Output Process

The Output process sequentially accesses the Integrated Spectra Store entries. It must wait for the Data Processing process if an incomplete entry is encountered. Competed entries are sent to the appropriate external destination via the high-speed, switched output network.

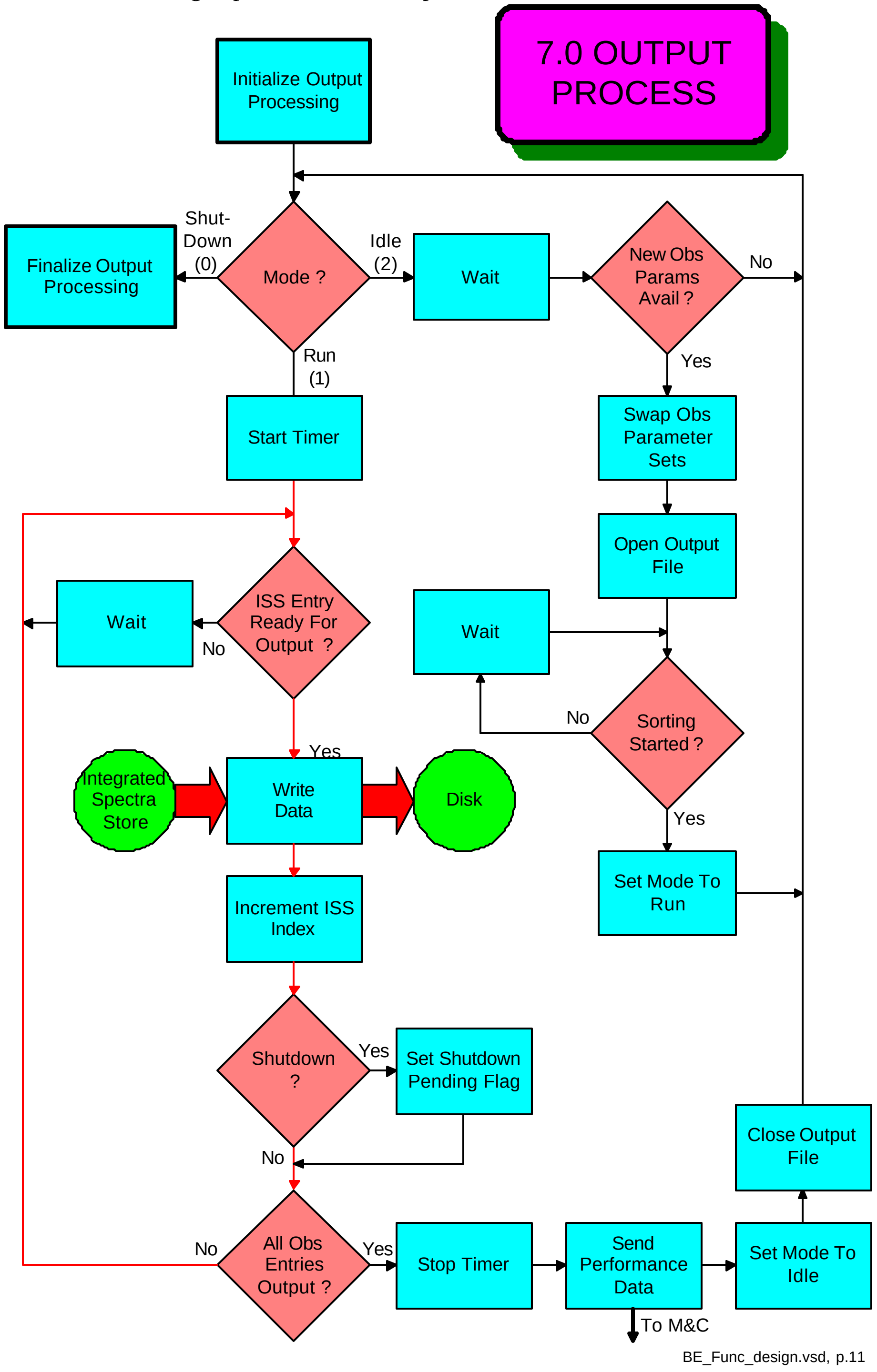


Figure 12. Output process flow chart.

4.2 Data Structures

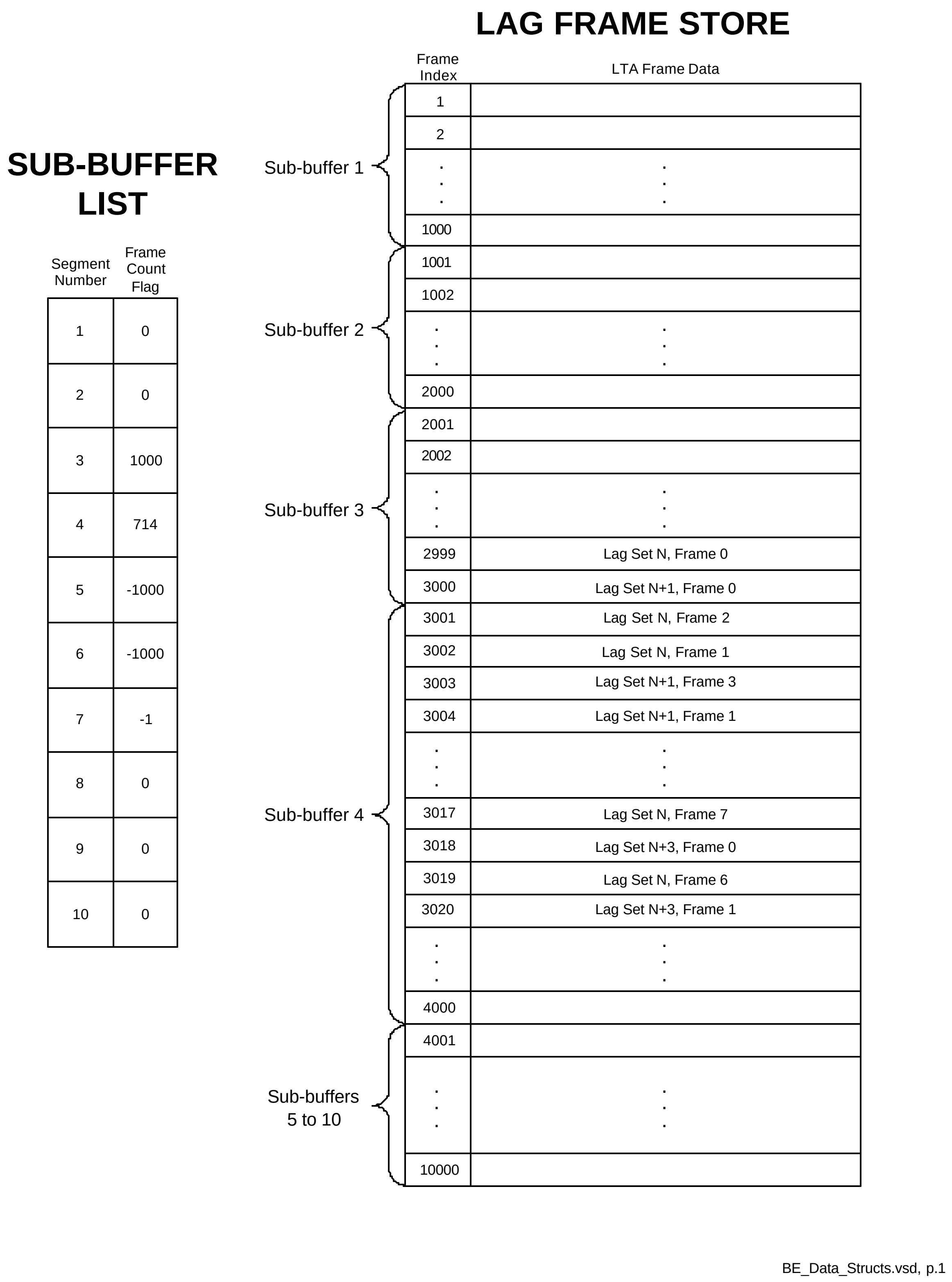
The major Correlator Backend data structures are the Lag Frame Store, the Lag Set Table and the Integrated Spectra Store. They are defined in a manner that minimizes the amount of memory-to-memory lag data movement in the processing node shared memory. The input lag frames from the WIDAR Correlator are written once into the Lag Frame Store by the Input task. The header fields are read and used by the Sorting process. The lag data fields are touched only during the initial checksum calculation. The Sorting task writes Lag Frame Store location indices into the much more compact Lag Set Table. The Lag Set Table is accessed for frame location indices by the Data Processing task that uses the indices to read appropriate lag frames from the Lag Frame Store. The Data Processing task accumulates spectra into the Integrated Spectra Store, and the Output task reads spectra and sends them to external destinations. In this way, the lags values are written and read only once, and the spectra are written and read only once.

There are several smaller data structures that will also be detailed below.

4.2.1 Lag Frame Store

The Lag Frame Store is a sequential circular buffer. If it becomes filled, either incoming data must be discarded or the oldest resident data must be over-written. Incoming WIDAR Correlator lag frames are all the same fixed size, so the size of an entry in the data structure is fixed and constant. This data structure can be initialize once at start-up and never needs to be changed. The data store or buffer is divided into a selectable number of logical sub-buffers. One sub-buffer is filled for every block of UDP frames received from the Correlator. A block is made-up of a pre-defined (but alterable) number of Correlator frames. The frames are stored in the buffers in the order received from the network, so they could be out of order with respect to the lag set processing order needed.

Data is removed from the Lag Frame Store by the Data Processing task. As each lag frame is read its space is freed for reuse. Once all frames in a sub-buffer have been freed, the sub-buffer becomes available for another block of input frames. A record of the status of each sub-buffer is kept in the Sub-buffer List data structure



Segment Number	Frame Count Flag
1	0
2	0
3	1000
4	714
5	-1000
6	-1000
7	-1
8	0
9	0
10	0

Figure 13. Internal structure of the Lag Frame Store and sub-buffer list array. Sub-buffers are filled one at a time by blocks of lag frames obtained from a series of UDP packet receives. The Sub-buffer List maintains a continuous status of all sub-buffers.

4.2.2 Sub-buffer List

This data structure is a simple array of length equal to the number of sub-buffers in the Lag Frame Store. Each element of the array records the status of its corresponding sub-buffer.

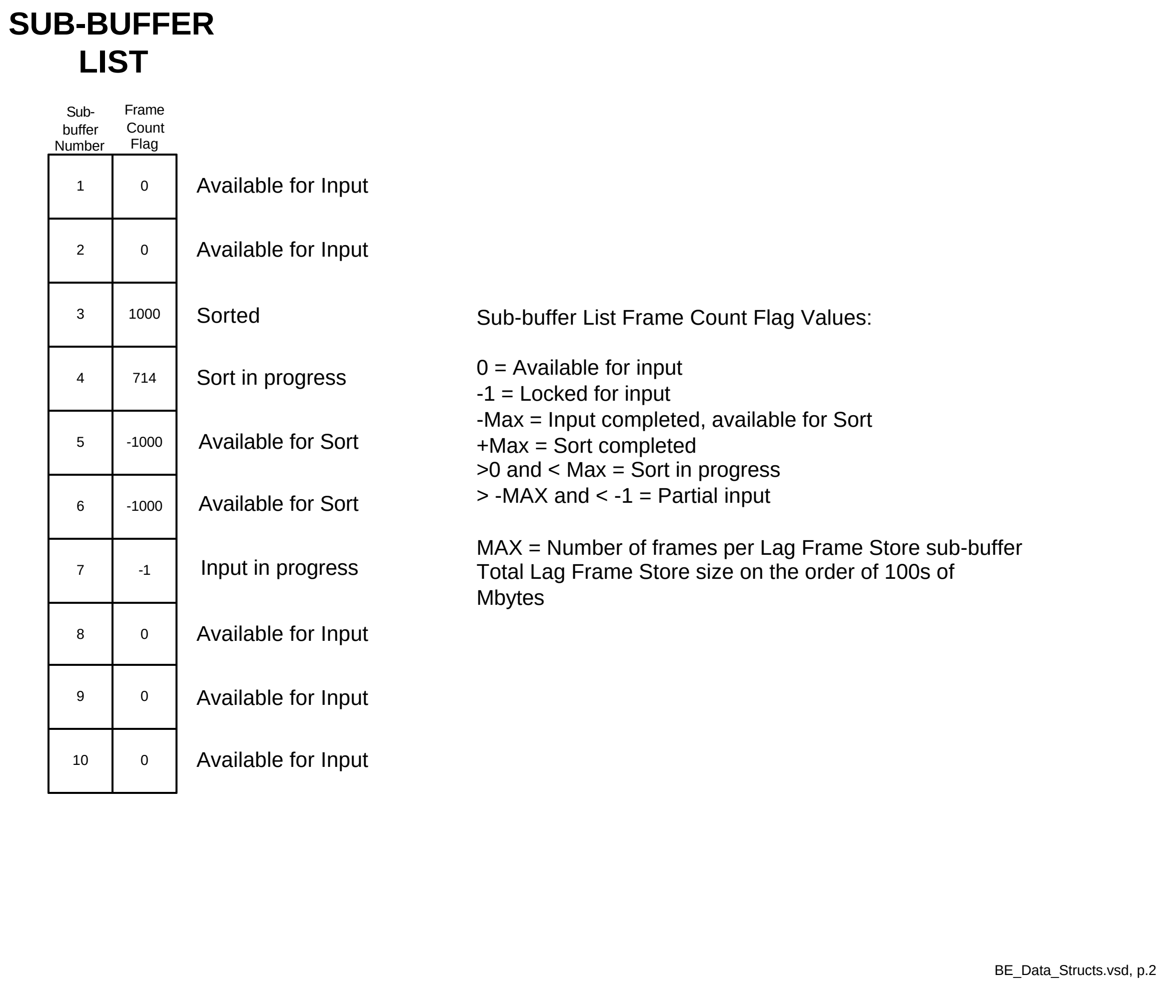


Figure 14. Lag Frame Store sub-buffer list array internal structure with example entries and an explanation of relevant entry values. A block size of 1000 frames has been assumed.

4.2.3 Lag Set Table

The Lag Set Table is a directly addressable data structure. Each lag set in an Observation will have a unique record in the Lag Set Table. Record size is dependent on the number of lags in a lag set and will consequently change from sub-band to sub-band within an Observation. The table dimensions may also change from Observation to Observation, so its parameters must be recalculated before each new Observation. The storage space is circular to allow for indefinite length (in time) Observations. If it is filled, the Sorting task will wait until space is freed by the Data Processing task. If the wait is too long it may impact the Input task by causing the Lag Frame Store to fill. Such an occurrence will generally be due to a software or hardware failure further along in the data processing pipeline and should be detectable by the Monitor task.

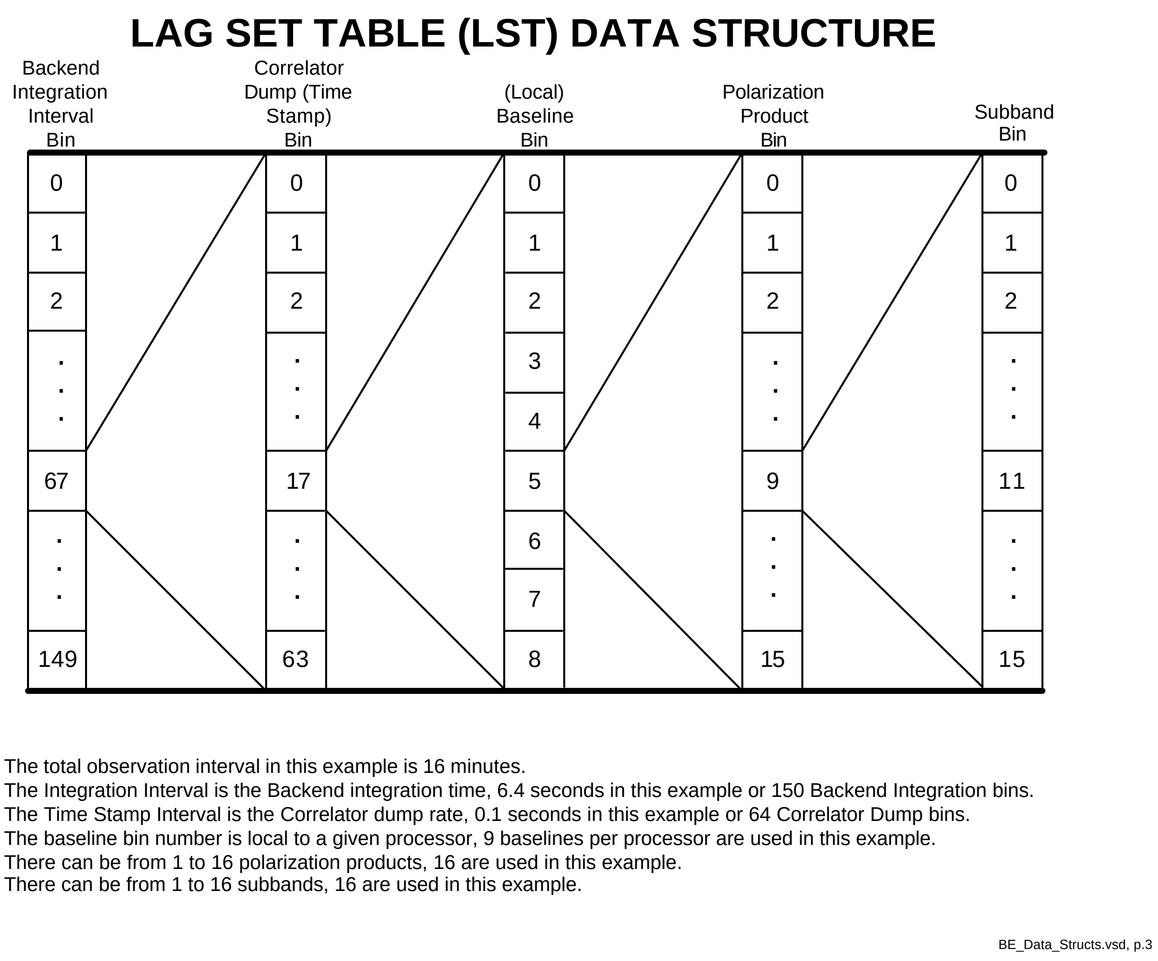
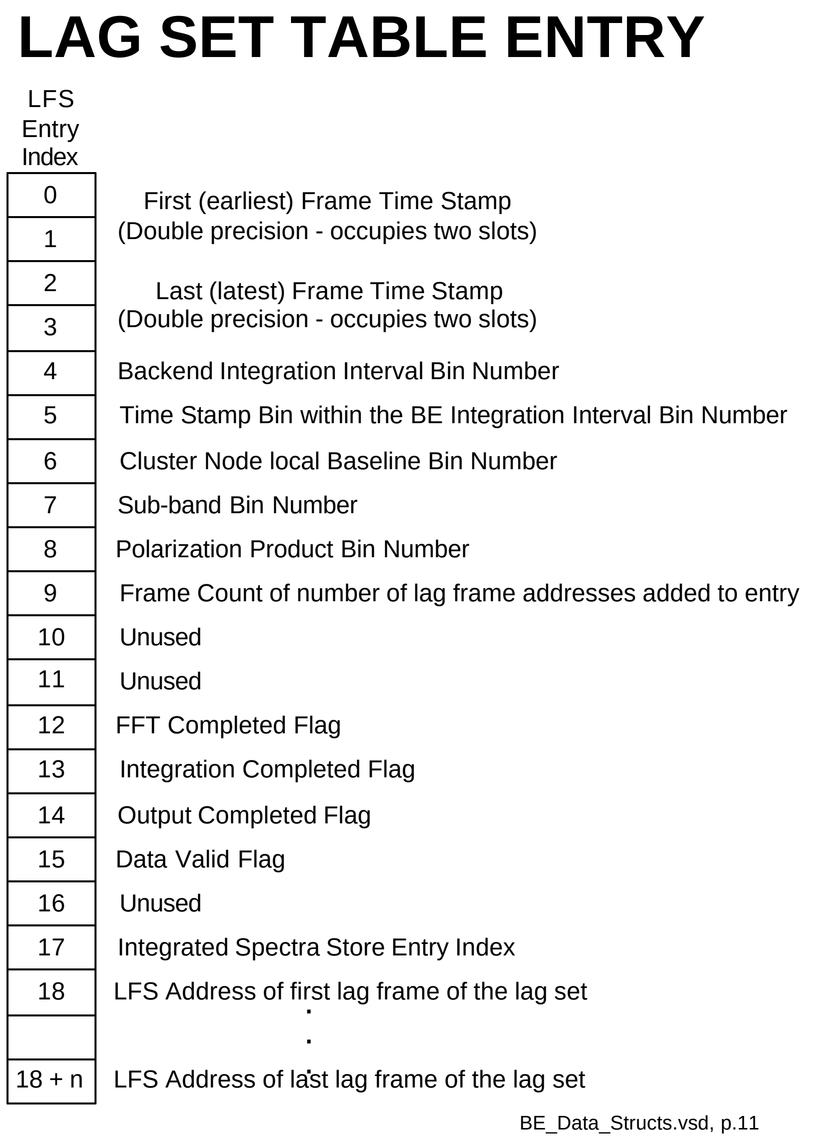

LAG SET TABLE (LST) DATA STRUCTURE
Backend Integration Interval Bin
Correlator Dump (Time Stamp) Bin
(Local) Baseline Bin
Polarization Product Bin
Subband Bin
0
1
2
67
149
0
1
2
17
63
0
1
2
3
4
5
6
7
8
0
1
2
9
15
0
1
2
11
15
The total observation interval in this example is 16 minutes.
The Integration Interval is the Backend integration time, 6.4 seconds in this example or 150 Backend Integration bins.
The Time Stamp Interval is the Correlator dump rate, 0.1 seconds in this example or 64 Correlator Dump bins.
The baseline bin number is local to a given processor, 9 baselines per processor are used in this example.
There can be from 1 to 16 polarization products, 16 are used in this example.
There can be from 1 to 16 subbands, 16 are used in this example.
BE_Data_Structs.vsd, p.3

Figure 15. Lag Set Table internal organization. The table is essentially a five dimensional array. Moving from left to right across the diagram a single element of a data axis (column in the diagram) is exploded in the column immediately to the right. The index of a table entry at the lowest level on the extreme right is obtained from the positions of the entry in preceding columns (axis) and the known length of the column (axis).



LAG SET TABLE ENTRY

LFS Entry Index	
0	First (earliest) Frame Time Stamp (Double precision - occupies two slots)
1	
2	Last (latest) Frame Time Stamp (Double precision - occupies two slots)
3	
4	Backend Integration Interval Bin Number
5	Time Stamp Bin within the BE Integration Interval Bin Number
6	Cluster Node local Baseline Bin Number
7	Sub-band Bin Number
8	Polarization Product Bin Number
9	Frame Count of number of lag frame addresses added to entry
10	Unused
11	Unused
12	FFT Completed Flag
13	Integration Completed Flag
14	Output Completed Flag
15	Data Valid Flag
16	Unused
17	Integrated Spectra Store Entry Index
18	LFS Address of first lag frame of the lag set
	. . .
18 + n	LFS Address of last lag frame of the lag set

BE_Data_Structs.vsd, p.11

Figure 16. Lag Set Table entry details.

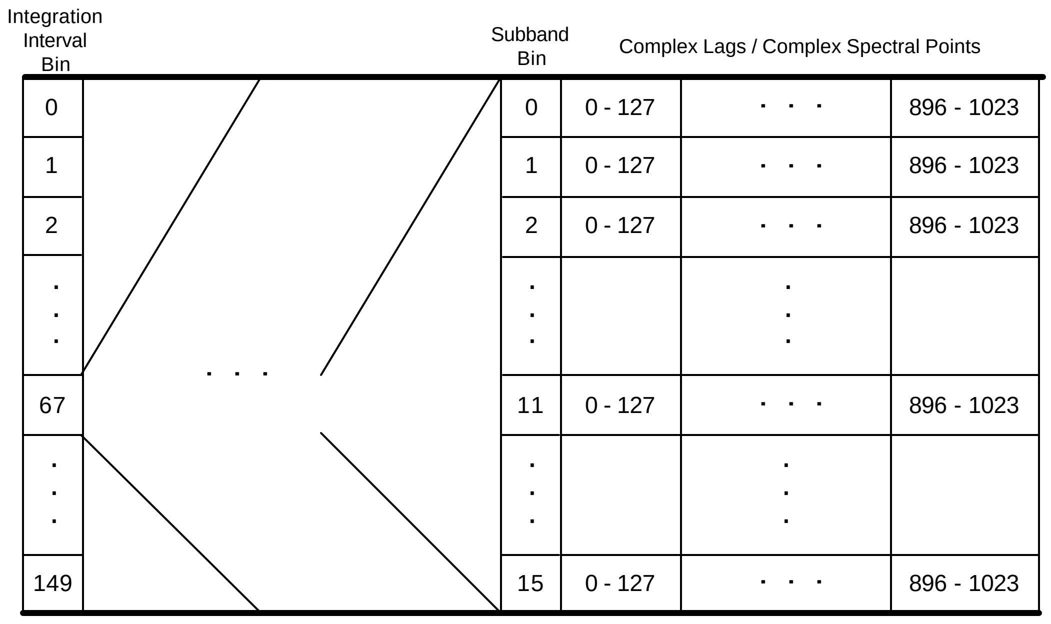
Each record in the Lag Set Table contains the Lag Frame Store indices of the lag frames that make up a single lag set. Header fields are used to record information unique to that lag set. The lag set records are in time sequential order by Backend Integration and Correlator dump numbers. The records for a given baseline are grouped together with the sub-bands for each polarization product sub-grouped. In this way the Lag Set Table record number for a given lag frame in the Lag Frame Store can be calculated directly by the Sorting process from the lag frame header information and the Observation parameters. Lag frame indices in the record are in lag sequential order. The position of a given lag frame in a sequence is also determinable from its header data. The ability to directly calculate the record number and sequential frame number in the lag set allows the Sorting process to handle input frames in random order thus simplifying handling of out-of-order arrival of frames over the Correlator-Backend network.

The Lag Set Table entries are accessed in order by the Data Processing task. As each entry is finished, it is freed for reuse.

4.2.4 Integrated Spectra Store

The Integrated Spectra Store is organized in a fashion analogous to the Lag Set Table. In this case, however, each record corresponds to an integrated set of Fourier Transformed Lag Sets and the data (rather than an index) is contained in the record. Header fields identifying the unique content of each record are also present. The records are time sequentially ordered by Backend integration number. The Correlator dump time sequential number has been eliminated by the integration, but the baseline, sub-band and polarization product groupings remain the same as in the Lag Set Table.

**INTEGRATED SPECTRA STORE (ISS)
and HEADER (ISH) DATA STRUCTURES**



The total observation interval in this example is 16 minutes.
The Integration Interval is the Backend integration time, 6.4 seconds in this example or 150 Backend Integration bins.
The Time Stamp Interval is the Correlator dump rate, 0.1 seconds in this example or 64 Correlator Dump bins.
The baseline bin number is local to a given processor, 9 baselines per processor are used in this example.
There can be from 1 to 16 polarization products, 16 are used in this example.
There can be from 1 to 16 subbands, 16 are used in this example.

BE_Data_Structs.vsd, p8

Figure 17. The Integrated Spectra Store is a four dimensional array similar to the Lag Set Table. The Correlator dump column has been eliminated by the integration. Index calculation proceeds in an analogous fashion.

4.2.5 Baseline Lookup Table

Each Backend Cluster node will handle lag frames from a limited number of baselines. In order to be able to place entries properly in the Lag Set Table, the So rting task must reduce the station ids provided in the frame header first to an absolute baseline number and then to a relative baseline number for a particular node. This is done via the Baseline Lookup Table and an absolute to relative baseline number conversion list for each Backend Cluster node. The conversion list is a simple array referencing a node's local baseline numbers to the equivalent global baseline numbers. The global baseline numbers are stored in a two-dimensional array with the X and Y axes being the X and Y lag frame header Station id numbers. Row and column are specified by the Station ids and an absolute baseline number is extracted from the table. The table is constructed once at Backend startup and only needs to be redone if more antennas are added to the array.

GLOBAL BASELINE NUMBER LOOKUP TABLE

	0	1	2	3	SID_X	24	25	26
0	351	0	1	2	· · ·	23	24	25
1	0	352	26	27	· · ·	48	49	50
2	1	26	353	51	· · ·	72	73	74
SID_Y	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
24	23	48	72	95	· · ·	375	348	349
25	24	49	73	96	· · ·	348	376	350
26	25	59	74	97	· · ·	349	350	377

BE_Data_Structs.vsd, p.4

Figure 18. Station ids in X and Y form the two axes of the table. Absolute baseline numbers for cross-correlations are number from 0 to 350 for a 27-antenna array. Autocorrelation baseline numbers run from 351 to 377 along the diagonal. Note that the upper right and lower left triangles of the array are symmetric.

4.2.6 Polarization Product Lookup Table

The eight available Basebands can be grouped into up to four baseband pairs, used as eight separate Basebands or some combination of these. Baseband pairs can produce one, two or four polarization products. As a result there can be up to sixteen total unique polarization products across the Basebands and Baseband pairs. A two dimensional array of valid polarization products is generated prior to the initiation of each Observation. The X and Y axes are the lag frame Baseband id numbers and the entries are relative polarization product number needed to properly place an entry in the Lag Set Table.

POLARIZATION PRODUCT NUMBER LOOKUP TABLE

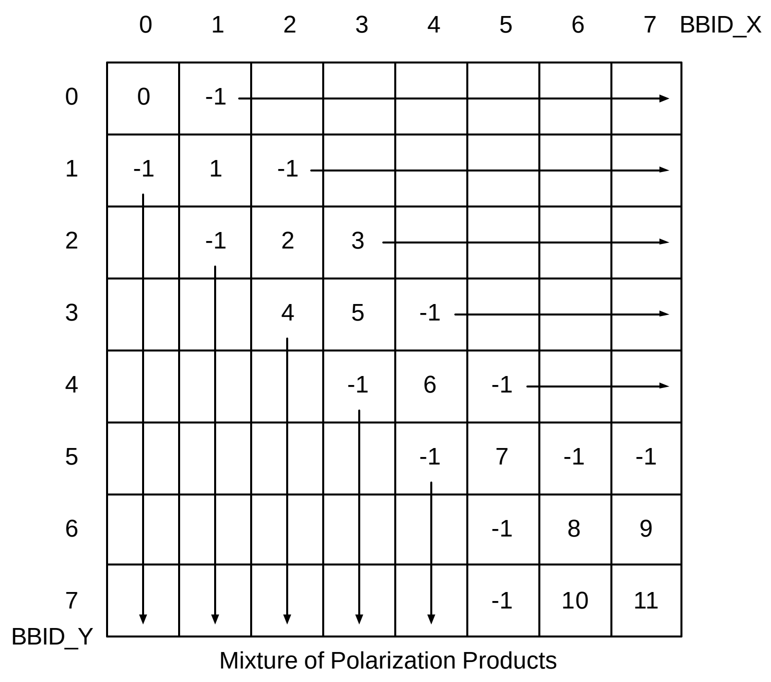
BBID_Y \ BBID_X	0	1	2	3	4	5	6	7
0	0	1	-1 →	→	→	→	→	→
1	2	3	-1 →	→	→	→	→	→
2	-1 ↓	-1 ↓	4	5	-1 →	→	→	→
3	↓	↓	6	7	-1 →	→	→	→
4	↓	↓	-1 ↓	-1 ↓	8	9	-1	-1
5	↓	↓	↓	↓	10	11	-1	-1
6	↓	↓	↓	↓	-1	-1	12	13
7	↓	↓	↓	↓	-1	-1	14	15

All 16 possible products are active

BE_Data_Structs.vsd, p.5

Figure 19. Up to sixteen polarization products can be generated when four baseband pairs and all polarization products for each is requested. This situation is shown above. Unused or invalid combinations are marked with a –1. For lesser numbers of basebands, baseband pairs and number of polarization products some additional entries will be marked as invalid.

POLARIZATION PRODUCT NUMBER LOOKUP TABLE


BBID_X
BBID_Y
Mixture of Polarization Products

	0	1	2	3	4	5	6	7
0	0	-1	→	→	→	→	→	→
1	-1	1	-1	→	→	→	→	→
2	↓	-1	2	3	→	→	→	→
3	↓	↓	4	5	-1	→	→	→
4	↓	↓	↓	-1	6	-1	→	→
5	↓	↓	↓	↓	-1	7	-1	-1
6	↓	↓	↓	↓	↓	-1	8	9
7	↓	↓	↓	↓	↓	-1	10	11

BE_Data_Structs.vsd, p.10

Figure 20. Only a total of twelve polarization products are requested. The first two basebands (0 and 1) form a pair for which all four possible polarization products is produced. The same is true for the last two basebands (6 and 7) that are also paired. Input observation parameters will indicate if any of the middle four basebands are paired. If for instance 2 and 3 are paired then the output polarization products will be RR and LL. If a baseband is not paired, its single polarization product will be determined by the polarization of the input signal.

4.3 Interfaces

The Correlator Backend has three interfaces to external systems, one each with M&C, the WIDAR Correlator and e2e. Physically, each of these interfaces will be network connections. Figure 21 is a diagram of an initial configuration that provides bandwidths necessary to meet minimum data throughput requirements. It can be realized using currently available switching, cabling and interface hardware similar to that being installed in the NRAO AOC building in Socorro.

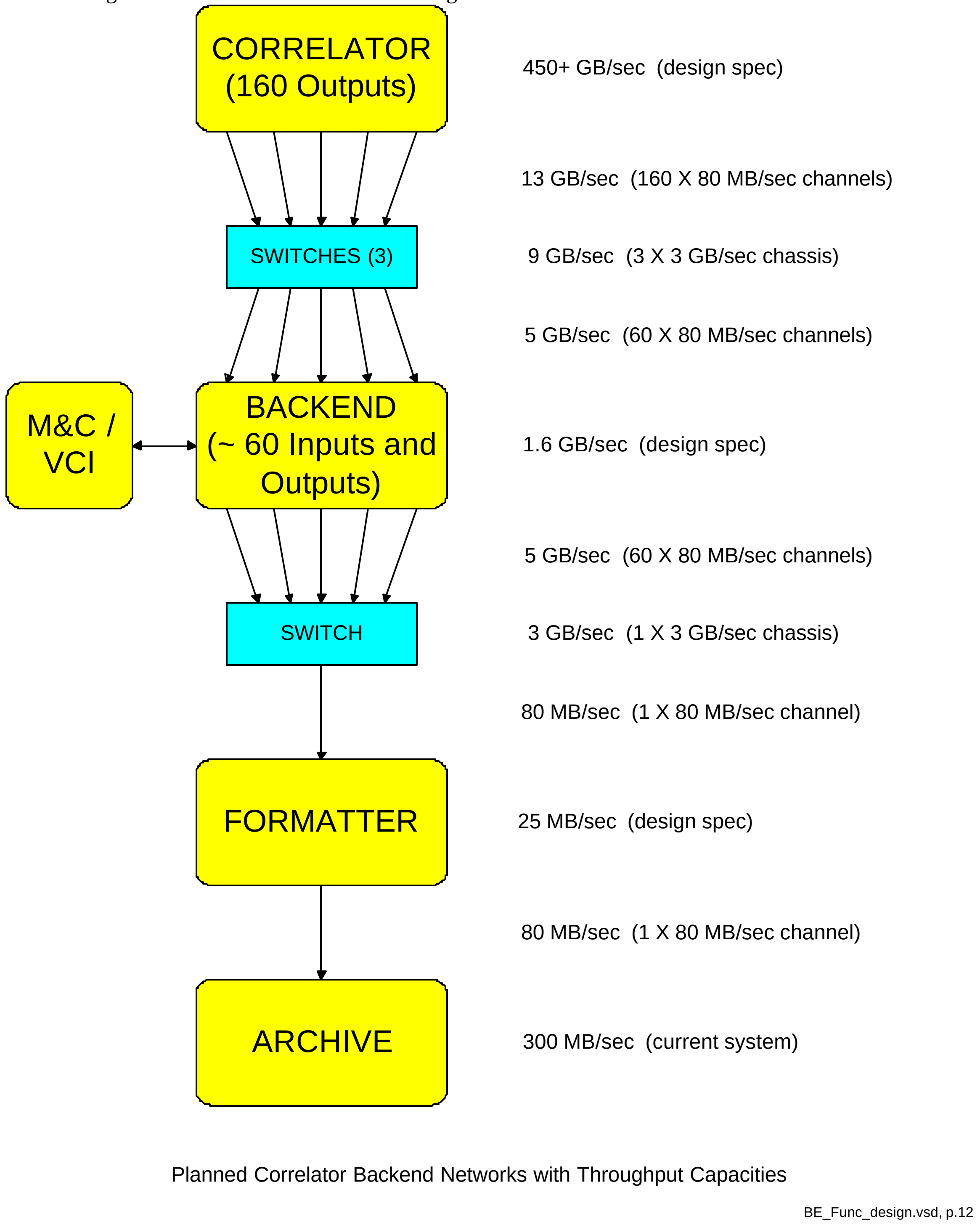

CORRELATOR (160 Outputs)
450+ GB/sec (design spec)
13 GB/sec (160 X 80 MB/sec channels)
SWITCHES (3)
9 GB/sec (3 X 3 GB/sec chassis)
5 GB/sec (60 X 80 MB/sec channels)
M&C / VCI
BACKEND (~ 60 Inputs and Outputs)
1.6 GB/sec (design spec)
5 GB/sec (60 X 80 MB/sec channels)
SWITCH
3 GB/sec (1 X 3 GB/sec chassis)
80 MB/sec (1 X 80 MB/sec channel)
FORMATTER
25 MB/sec (design spec)
80 MB/sec (1 X 80 MB/sec channel)
ARCHIVE
300 MB/sec (current system)
Planned Correlator Backend Networks with Throughput Capacities
BE_Func_design.vsd, p.12

Figure 21. Correlator Backend Networks.

4.3.1 WIDAR Correlator Interface

The WIDAR Correlator interface is completely specified by the Lag Frame definition provided in NRC-EVLA Memo # 014. The Backend Sorting process follows the format of the frame definition to extract header and lag data from incoming UDP data packets.

4.3.2 M&C Interface

Communications across the interface with the M&C System will be via message passing middleware. The specific M&C middleware product has yet to be determined, but the current Backend software employs PVM and will need to maintain compatibility with this product. A number of different kinds of information will flow in both directions across the interface. All will be sent and received as messages using sockets based TCP/IP packets.

The following information flows into the Backend:

1) Global parameters that govern the overall operation of the Backend and apply to all sub-arrays and Observations.
2) Observation parameters including ranges and valid values of all lag frame header fields needed during processing, Backend integration time and optional processes to be applied.
3) Auxiliary data such as State counts and/or requantizer data if VanVleck type corrections are to be applied.
4) Requests for system status.
5) System state instructions such as shutdowns.

The following information will flow from the Backend:

1) Autocorrelations.
2) System Status reports.
3) Error and warning messages.
4) Performance data.

4.3.2.1 Global Parameters

The following structure is maintained in the global parameters area of the backend:

Struct Array_Params Array;

It references the following data structures and parameters that must be supplied to the Backend. Any time there is a change in the configuration of the array, these parameters must be refreshed. This must be done before any Observations that depend upon the new parameters are communicated to the Backend.

```
/*------------------------------------------------------------------------------------------*/
/* Structure for the configuration parameters of a sub-array                                 */
/*------------------------------------------------------------------------------------------*/
struct Array_Params
{
        unsigned Num_SubArrays;         /* Number of sub-arrays currently defined            */
        struct SubArray_Config_Params SubArray_Config[MAX_SUBARRAYS];
                                        /* Array of structs for the currently active sub-arrays */
```

```
            struct Baseline_Config_Params Baseline_Config[MAX_BASELINES];
                                               /* Array of structs for all baselines (including          */
                                               /* autocorrelations)                                      */
};

/*---------------------------------------------------------------------------------------------------------------*/
/* Structure for the configuration parameters of a sub-array                                                   */
/*---------------------------------------------------------------------------------------------------------------*/
struct SubArray_Config_params
{
            unsigned Num_Anten;       /* Number of antennas in the sub-array (allowed values are 1    */
                                      /* to 32). Note: The initial Phase I WIDAR Correlator will be   */
                                      /* able to handle up to 32 antennas even though there will only */
                                      /* be 27 in the array. The total of all antennas for all sub-arrays*/
                                      /* may exceed the actual number of available antennas.           */
                                      /* A physical antenna can be defined to be in more than one      */
                                      /* sub-array at a time, but can only be active in one sub-array  */
                                      /* at a time.                                                    */
            unsigned Num_Baselines; /* Number of baselines for the antennas specified including     */
                                      /* autocorrelations (allowed values are 1 to Num_Anten +         */
                                      /* (Num_Anten * (Num_Anten – 1)) / 2)                            */
};

/*---------------------------------------------------------------------------------------------------------------*/
/* Structure for the configuration a baseline                                                                  */
/*---------------------------------------------------------------------------------------------------------------*/
struct Baseline_Config_Params
{
            unsigned Global_bl_num;   /* Global baseline number                                        */
            unsigned Antenna_1;       /* Antenna number of one of the two contributing to the         */
                                      /* baseline (analogous to the "X" station id)                    */
            unsigned Antenna_2;       /* Antenna number of the other antenna contributing to the      */
                                      /* baseline (analogous to the "Y" station id)                    */
            unsigned SubArray_num;    /* Number of the sub-array to which the baseline belongs.       */
                                      /* Unlike antennas, a baseline can only be assigned to one      */
                                      /* sub-array at a time.                                          */
};
```

4.3.2.2 Observation Parameters

The following structure is maintained in the local Observation parameters area on each Backend processing node:

Struct SubArray_Obs_Params SubArray;

It references the following data structures and parameters that must be supplied to the Backend. Any time there is a change of Observation parameters for the sub-array, these parameters must be refreshed. This must be done before the Observation is scheduled to begin and they cannot be changed during the time the Observation is in effect.

```
/*-----------------------------------------------------------------------------------------------------------------*/
/* Structure for one Observation for one sub-array                                                                 */
/*-----------------------------------------------------------------------------------------------------------------*/
struct SubArray_Obs_Params
{
            double Obs_beg_time;      /* Observation start time                                              */
            double Obs_end_time;      /* Observation end time. A negative value indicates an                 */
                                      /* indefinite end time. A valid end time must eventually be            */
                                      /* be provide in order to end the Observation                          */
            unsigned BE_dumps_per_obs;         /* Number of Backend integration dumps (outputs)    */
                                      /* for the observation. If the end time is indefinite this value       */
                                      /* must be negative. A valid value must eventually be supplied */
                                      /* in order to end the Observation.                                    */
            struct Baseline_Params Baseline[MAX_BASELINES];           /* Array of baseline          */
                                      /* structs with baseline parameters specific to the Observation */
            struct Baseband_Params Baseband[MAX_BASEBANDS];       /* Array of baseband          */
                                      /* structs with baseband parameters specific to the Observation*/
            struct PolProd_Params PolProd[MAX_POL_PRODS];            /* Array of polarization    */
                                      /* product structs with polarization product parameters        */
                                      /* specific to the Observation                                         */
            unsigned total_pp;        /* Total number of polarization products across all requested   */
                                      /* basebands                                                           */

/* The folowing parameters define the sizes and addresses of the Lag Set Table and Integrated        */
/* Spectra Storage for various levels of bins in the storage hierarchies. These parameters are not     */
/* supplied from the outside. Instead, they are calculated from other Observation parameters.          */
/* The first three permit variable numbers of lags/spectral channels at the sub-band level.             */

unsigned LST_sb_entry_sizes[MAX_POL_PRODS][MAX_SUBBANDS];
                                      /* Array of lowest level Lag Set Table entry sizes. One for       */
                                      /* each sub-band of each baseband in the Observation.             */
unsigned ISS_sb_entry_sizes[MAX_POL_PRODS][MAX_SUBBANDS];
                                      /* Array of lowest level Integrated Spectra Store entry sizes.    */
                                      /* One for each sub-band of each baseband in the Observation.*/
unsigned LST_sb_entry_indices[MAX_POL_PRODS][MAX_SUBBANDS];
                                      /* Array of lowest level Lag Set Table entry indices. One for    */
                                      /* each sub-band of each baseband in the Observation.             */
unsigned LST_bl_bin_size;             /* Number of 4-byte values in one baseline block in the Lag     */
                                      /* Set Table.                                                          */
unsigned LST_cd_bin_size;             /* Number of 4-byte values in one Correlator dump block in      */
                                      /* the Lag Set Table.                                                  */
unsigned LST_in_bin_size;             /* Number of 4-byte values in one Backend Integration block   */
                                      /* in the Lag Set Table.                                               */
unsigned ISS_bl_bin_size;             /* Number of 4-byte values in one baseline block in the         */
                                      /* Integrated Spectra Store.                                           */
unsigned ISS_in_bin_size;             /* Number of 4-byte values in one Backend Integration block   */
                                      /* in the Integrated Spectra Store.                                    */
};
```

```
/*--------------------------------------------------------------------------------------------------------------*/
/* Structure for baseline parameters that are Observation dependent                                      */
/*--------------------------------------------------------------------------------------------------------------*/
struct Baseline_Params
{
        unsigned Global_bl_num;   /* Global baseline number                                   */
        unsigned Local_bl_num;    /* Local baseline number on the processing node             */
        unsigned BE_node;         /* Backend Cluster destination node for all WIDAR           */
                                  /* Correlator output frames for the baseline                */
        unsigned BE_IP_addr;      /* Backend Cluster destination node IP address for all WIDAR*/
                                  /* Correlator output frames for the baseline. (Note: this   */
                                  /* address can change during an observation if there has been */
                                  /* a failure on the node for the baseline. The new address will */
                                  /* be that of the standby node to which the baseline has been */
                                  /* transferred.)                                            */
        unsigned WC_IP_addr;      /* WIDAR Correlator source IP address of all Correlator     */
                                  /* frames for the baseline                                  */
        unsigned SID_X;           /* Station ID for the "X" antenna                           */
        unsigned SID_Y;           /* Station ID for the "Y" antenna                           */
        unsigned Antenna_1;       /* Antenna number of one of the two contributing to the     */
                                  /* baseline (analogous to the "X" station id)               */
        unsigned Antenna_2;       /* Antenna number of the other antenna contributing to the  */
                                  /* baseline (analogous to the "Y" station id)               */
        unsigned SubArray_num;    /* Number of the sub-array to which the baseline belongs. This*/
                                  /* parameter must be the negative of the actual sub-array   */
                                  /* number if one or both of the antennas is currently inactive */
                                  /* (That is, not being used by the sub-array during the current */
                                  /* Observation.)                                            */
        unsigned LTA_frames_per_cd;       /* Expected number of WIDAR Correlator LTA          */
                                  /* frames accumulated per Correlator dump                   */
        unsigned Corr_dumps_per_BE_dump;          /* Expected number of WIDAR Correlator      */
                                  /* dumps per Backend integration dump                       */
};

/*--------------------------------------------------------------------------------------------------------------*/
/* Structure for baseband parameters that are Observation dependent                                      */
/*--------------------------------------------------------------------------------------------------------------*/
struct Baseband_Params
{
        unsigned bb_active[MAX_BASEBANDS];   /* 1 = active, 0 = inactive baseband             */
        unsigned bb_mate[MAX_BASEBANDS];     /* baseband number of pairing mate               */
                                  /* (allowed values are 0 to 8, where 0 means the baseband is */
                                  /* unpaired)                                                */
        unsigned bb_num_pp[MAX_BASEBANDS];/* number of polarization products: 0 if            */
                                  /* inactive, 1 if unpaired, 2 or 4 if paired                */
        double bb_bw;             /* Baseband bandwidth in HZ                                 */
        double bb_lf;             /* Baseband lowest frequency in HZ                          */
        unsigned bb_num_sb;       /* Number of sub-bands in the baseband  (allowed values are */
                                  /* 0 to 16 where 0 is equivalent to the baseband being inactive)*/
        struct SubBand_Params SubBand[MAX_SUBBANDS];
                                  /* Array of structs containing sub-band specific parameters */
};
```

```
/*----------------------------------------------------------------------------------------------------------------*/
/* Structure for parameters that change with sub-band                                                            */
/*----------------------------------------------------------------------------------------------------------------*/
struct SubBand_Params
{
        unsigned SBID_X;        /* Correlator Sub-band ID (always = SBID_Y)                 */
        double sb_bw;           /* Sub-band bandwidth in HZ                                 */
        double sb_lf;           /* Sub-band lowest frequency in HZ                          */
        unsigned Num_bits       /* Number of data bits (either 4 or 7)                      */
        unsigned Num_Lags;      /* Number of complex lags for the sub-band (must be a power */
                                /* two from 128 to 262,144)                                 */
        unsigned Num_Frames;    /* Number of WIDAR Correlator data frames needed to         */
                                /* contain the requested number of lags (Num_Frames is      */
                                /* defined as Num_Lags / 128, acceptable values are 1, 2, 4, 8,*/
                                /* 16, 32, 64, 128, 256, 512, 1024, 2048)                   */
};

/*----------------------------------------------------------------------------------------------------------------*/
/* Structure for parameters that change with polarization product                                                */
/*----------------------------------------------------------------------------------------------------------------*/
struct PolProd_Params
{
        unsigned BBID_X;        /* "X" station baseband ID for the polarization product     */
        unsigned BBID_Y;        /* "Y" station baseband ID for the polarization product     */
};
```

4.3.2.3 Auxiliary Data

The types and definitions of auxiliary input data have not yet been specified.

4.3.2.4 Autocorrelations

Autocorrelations will be produced by the baseline boards of the WIDAR Correlator and will flow to the Backend Cluster in data frames at the same time as the cross-correlations. Some amount of these autocorrelations will be needed by other parts of the M&C system. The specific amounts, delivery locations, frequency of delivery and formats of this data have not yet been determined.

4.3.2.5 System Status Requests

The types and definitions of system status requests have not yet been specified.

4.3.2.6 System State Instructions

System state instructions are currently defined as three possible states conveyed by a single unsigned integer. The value zero (0) instructs the Backend System to shutdown, the value one (1) instructs the

Backend System to run, and the value two (2) instructs the Backend System to idle. The Backend will always be in one of these three states. During a state change different components (executables) may be in different states for some period of time as dictated by the internal condition of the component at the time of the state change request. For instance, an idle or shutdown from the run state will not be completed until all data in the Lag Frame Store, and Integrated Spectra Store for the current Observation have been completely processed and disposed of to the Archive. The Input process will not shutdown until all input for the Observation has been received, hence a shutdown during an active Observation will not take affect until the Observation ends. The Sort, DP and Output routines shutdowns will follow only as they complete their work for the data remaining in their buffers. A second idle or shutdown during an Observation while an idle or shutdown is already pending will result in an immediate cessation of processing by all Backend Components. The submission of an idle or run state request while a shutdown is pending will cancel the shutdown.

The Backend System is initialized in the idle state and returns to it every time an Observation is completed. A new run state request must be issued with each new Observation sent to the Backend. A shutdown causes all Backend executables to terminate and the entire Backend System must be restarted before operations can continue. It is recommended that an idle request always be submitted prior to a shutdown, and that the shutdown not be issued until the idle state has been fully reached and confirmed. A direct shutdown should only be done in extreme cases where some or all of the system components are hung and self-recovery has already failed. If a shutdown does not succeed at this point, the next recourse is to cancel the START_CBE command at the command line. This will kill all tasks, cleanup shared memory segments and close all files. There will generally be no need to halt the PVM console.

4.3.2.7 System Status Reports

The types and form of system status reports have not yet been defined.

4.3.2.8 Error and Warming Messages

The types and form of error and warning messages have not yet been defined.

4.3.2.9 Performance Data Reports

The types and form of performance data reports have not yet been defined.

4.3.3 e2e Interface

The Backend System will send spectra to the e2e System. Currently, records containing a single spectrum with its header data are written. All the output records generated for all baselines from a given Backend node for a given Observation are written to a separate local file on that node. The names of all the local files for an Observation obey a consistent naming convention. These local files form a single logical parallel file distributed across the Backend nodes. The data and auxiliary information in the components of this logical parallel distribute file will ultimately be incorporated into FITS tables or some other appropriate published format.

The current output format for spectral data is giving by the following structures. The output is written to a binary file.

```
/*-----------------------------------------------------------------------------------------------------------------*/
/* Structure defining the output spectra file header. One header record is written at the beginning    */
/* of the file                                                                                         */
/*-----------------------------------------------------------------------------------------------------------------*/
 struct fheader_format
{
        unsigned hlen;          /* Number of 4-byte integers in the file header                    */
        unsigned dlen[MAX_POL_PRODS][MAX_SUBBANDS];
                                /* Array of spectra data record data lengths in number of          */
                                /* complex pairs consisting of two single precision floating       */
                                /* point values. An array is needed because each sub-band can */
                                /* have a different number of spectral components so the          */
                                /* record length will vary                                         */
};

/*-----------------------------------------------------------------------------------------------------------------*/
/* Structure defining the output spectra data record header. One header record is written prior to    */
/* each spectral data record.                                                                          */
/*-----------------------------------------------------------------------------------------------------------------*/
struct header_format
{
        double TS_first;        /* Earliest time stamp of any spectral data in the data record     */
        double TS_last;         /* Latest time stamp of any spectral data in the data record       */
        unsigned in_num;        /* The Backend Integration number for the Observation              */
        unsigned num_cd;        /* The actual number of correlator dumps for the BE                */
                                /* integration                                                     */
        unsigned bl_num;        /* The baseline number of the spectral data                        */
        unsigned pp_num;        /* The polarization product number of the spectral data            */
        unsigned sb_num;        /* The sub-band number of the spectral data                        */
        unsigned num_ch;        /* The number of spectral channels in the spectral data            */
        unsigned FFT_flag;      /* Fourier Transform Flag. 1 = FFT has been done, the data         */
                                /* are spectra, 0 = the data are lag values                        */
        unsigned Int_flag;      /* Backend Integration Flag. 1 = The data have been                */
                                /* integrated, 0 = The data have NOT been integrated (i.e.,        */
                                /* there was more than one correlator dump per Backend             */
                                /* integration interval, but the integration was not performed)   */
        unsigned Vld_flag;      /* Data Valid Flag. 1 = All data in the spectral data record are   */
                                /* valid, 0 = Some or all of the data in the spectral data record  */
                                /* are invalid, the individual data element data valid flags       */
                                /* should be consulted. (Note: the form of these individual data*/
                                /* element data valid flags has not yet been specified)            */
};
```