WCAP-13822

EAGLE-21/AMSAC Diversity Evaluation

Authors

J. P. Doyle
L. E. Erin
H. J. Murphy
J. J. Patnesky
T. C. Tuite

August, 1993

Approved: [illegible] for W. L. Miller
W. L. Miller, Manager
Nuclear Applications
Process Control Division

Approved: R. J. Sterdis
R. J. Sterdis, Manager
Safety System Licensing
Nuclear & Advanced Technology
Division

Westinghouse Electric Corporation
Process Control Division
200 Beta Drive
Pittsburgh, Pennsylvania 15238-2987

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

1.1 Purpose

The purpose of this evaluation is to demonstrate the diversity between the Diablo Canyon Power Plant (DCPP) Eagle 21 Process Protection System upgrade instrumentation and the ATWS Mitigation System Actuation Circuitry (AMSAC), in the areas of functional design, system design, hardware design, and software design.

1.2 Background

To meet the requirements of 10 CFR 50.62, Pacific Gas & Electric (PG&E) installed Westinghouse supplied AMSAC equipment at DCPP Units 1 and 2 during refueling outages 1R3 and 2R2 (Fall 1989 and Fall 1988 respectively). At DCPP, AMSAC initiates a turbine trip, starts auxiliary feedwater and isolates the steam generator blowdown and sample lines, when the steam generator inventory is below the AMSAC low level setpoint. The AMSAC equipment is required by 10CFR 50.62 and the ATWS Final Rule to be diverse from the reactor trip system.

At the time of AMSAC installation at DCPP, the process protection portion of the reactor trip system was implemented with the Westinghouse 7100 Series Process Protection System (PPS). During refueling outages 1R6 and 2R6 (Spring 1994 and Fall 1994), PG&E plans to replace their aging, 7100 Series PPS equipment with the Westinghouse Eagle 21 PPS upgrade instrumentation. AMSAC is required to be diverse from Eagle 21 which is an integral part of the DCPP reactor trip system. In anticipation of some Nuclear Power Plants utilizing both an Eagle 21 PPS and a Westinghouse AMSAC, Westinghouse took deliberate measures during the AMSAC product definition and development phases to provide for diversity between AMSAC and Eagle 21 instrumentation, in the areas of system design, hardware design, and software design.

1.3 Regulatory Guidance

Regulatory guidance for AMSAC implementation is provided by 10 CFR 50.62 which is supplemented by the AMSAC Final Rule. The specific guidance is as follows:

- 10 CFR 50.62 — "Each pressurized water reactor must have equipment from sensor output to final actuation device, that is diverse from the reactor trip system, to automatically initiate the auxiliary (or emergency) feedwater system and initiate a turbine trip under conditions indicative of an ATWS. This equipment must be designed to perform its function in a reliable manner and be independent (from sensor output to interruption of power to the control rods).

- AMSAC Final Rule

  Mitigating Systems — Equipment diversity to the extent reasonable and practicable to minimize the potential for common cause failures is required from the sensors to, but not including, the final actuation device e.g., existing circuit breakers may be used for

auxiliary feedwater initiation. The sensors need not be of a diverse design or manufacturer. Existing protection system instrument sensing lines may be used. Sensors and instrument sensing lines should be selected such that adverse interactions with existing control systems are avoided.

In cases where existing protection system sensors are used to provide signals to the diverse equipment, particular emphasis should be placed on the design of the method used to isolate the signal from the existing protection system to minimize the potential for adverse electrical interactions.

2.0 ORGANIZATIONAL AND FUNCTIONAL DIVERSITY

2.1 AMSAC Design Evolution

In order to ensure that AMSAC instrumentation would be diverse to the extent reasonable and practicable from the Eagle 21 process protection system, Westinghouse organized AMSAC hardware and software design teams separate from the Eagle 21 hardware and software design teams. This deliberate separation allowed for the development of the two products to evolve from different design groups, through the efforts of independent designers, under the direction of different managers, during slightly different time periods.

The Westinghouse AMSAC design program was initiated in 1984 just prior to the issuance of the ATWS Rule. A design team, separate from any of the other Westinghouse digital system product design teams, was established. The team organization was as follows:

+a,c

The AMSAC System Design Specification and Software Design Requirements were prepared by individuals in a group separate from the group preparing these types of specification and requirements documents for the Eagle 21 system.

The hardware design, primarily mechanical components, power supplies and select boards, was based upon the Reactor Vessel Level Instrumentation System (RVLIS-86) equipment. Other modules (processor boards, relays, etc...) were selected with hardware diversity to a module level as a key consideration. Modules not used or planned for use in the Eagle 21 design were selected to perform the AMSAC mitigative functions.

The AMSAC system design was completed in 1986 with the first system delivered to Houston Lighting & Power for their South Texas Plants. The first system installed and operated was at Alabama Power Company's J. M. Farley Plant in 1987.

2.2 Eagle 21 Design Evolution

The Westinghouse Eagle 21 design program was initiated in 1985. A design team, separate from any of the other Westinghouse digital system product design teams, was established. The initial team organization was as follows:

[ ]+a,c

The hardware design for Eagle 21 was derived from the Integrated Protection System (IPS) which Westinghouse has developed for new plant applications. This hardware platform is based upon processor boards, I/O interface boards, and an architecture different from that used in AMSAC.

As the AMSAC design was being finalized, the Eagle 21 design program was in its early stages. Prototyping of the Eagle 21 conceptual design was completed in 1987 and equipment qualification tests were completed in 1988. The first delivery of Eagle 21 occurred in 1989 for implementing RTD Bypass Elimination at Tennessee Valley Authority's (TVA's) Watts Bar Unit 1.

By the time the Watts Bar Unit 1 Eagle 21 system was installed in 1989, a new design organization had evolved into place. This organization implemented design changes and improvements which were first introduced for the Eagle 21 Process Protection System upgrade at TVA's Sequoyah Unit 1 in 1990. This design team is depicted below:

+a,c

Since the Sequoyah Unit 1 Eagle 21 design was completed, design activities relating to Eagle 21 have been minimal and are now considered maintenance activities. These activities are being processed by the following organization:

+a,c

Three individuals were common to both the AMSAC and Eagle 21 programs. The first individual's involvement in Eagle 21 was limited to modifying diagnostic routines originally utilized in the Integrated Protection System (IPS) for use in Eagle 21 error handling and reporting routines. His work on AMSAC consisted of directing the development of the software design, coding significant portions of the software and participation in the hardware/software integration.

The second individual was a technician on the project and had no design responsibility.

The third individual's activities on Eagle 21 have consisted of being the project engineer for the Zion Unit 1 & 2 systems for a period of approximately two years, creating manufacturing configuration drawings (no detailed circuit design), and more recently as the responsible project manager. His management responsibility has been primarily focused on cost and schedule performance for ongoing projects. He has managed application of the completed design rather than development of the original design. His role in the AMSAC program was that of lead engineer for system development, hardware configuration design, and interfacing with customers to assist in plant implementation.

2.3 Functional Diversity

2.3.1 AMSAC Functional Design

The Diablo Canyon AMSAC system provides a diverse means to initiate a turbine trip, start auxiliary feedwater, and isolate the steam generator blowdown and sample lines through equipment which is independent from the Reactor Protection System (RPS). At DCPP, these functions are initiated when the water level in three out of four steam generators is below the AMSAC low level setpoint. The steam generator level and first stage turbine impulse pressure inputs to AMSAC are derived from the existing transmitters which will be powered by supplies located on the Eagle 21 analog input board as shown in Figure 1. These signals will be isolated in the Eagle 21 equipment prior to any signal processing, or analog-to-digital conversion, and will be connected as analog signals to AMSAC for processing of the mitigative functions. This configuration assures independence of the AMSAC signals, downstream of the field sensors and transmitter power supplies, from the RPS as required by the ATWS Rule. As a result, any failures within Eagle 21, such as processor lockups, will not affect AMSAC. This design is functionally the same as the existing licensed design using 7100 process electronics.

Figure 2 shows the existing and proposed designs. Both of these designs isolate the non-safety AMSAC system from the protection system and thus comply with the RPS diversity requirements set forth in WCAP-7306. This isolation prevents faults from the non-safety AMSAC system from affecting the RPS.

A simplified block diagram of the AMSAC architecture is shown in Figure 3. The architecture is designed to be both fault tolerant and highly reliable by using three separate sets of input boards and processors. Each processor performs identical 3 out of 4 coincidence logic calculations using the same inputs and sends the resulting component actuation demand signals to two majority voting modules. Each voting module performs a 2 out of 3 coincidence logic

vote on the processor demand signals and drives output relays for interfacing with the turbine trip and auxiliary feedwater start circuits.

2.3.2 Eagle 21 Functional Design

Eagle 21 will be installed to replace and perform all functions of the existing 7100 Series PPS equipment. This upgrade is referred to as a form, fit and functional replacement since it utilizes existing racks, minimizes impact on field wiring, and maintains all existing signal interfaces.

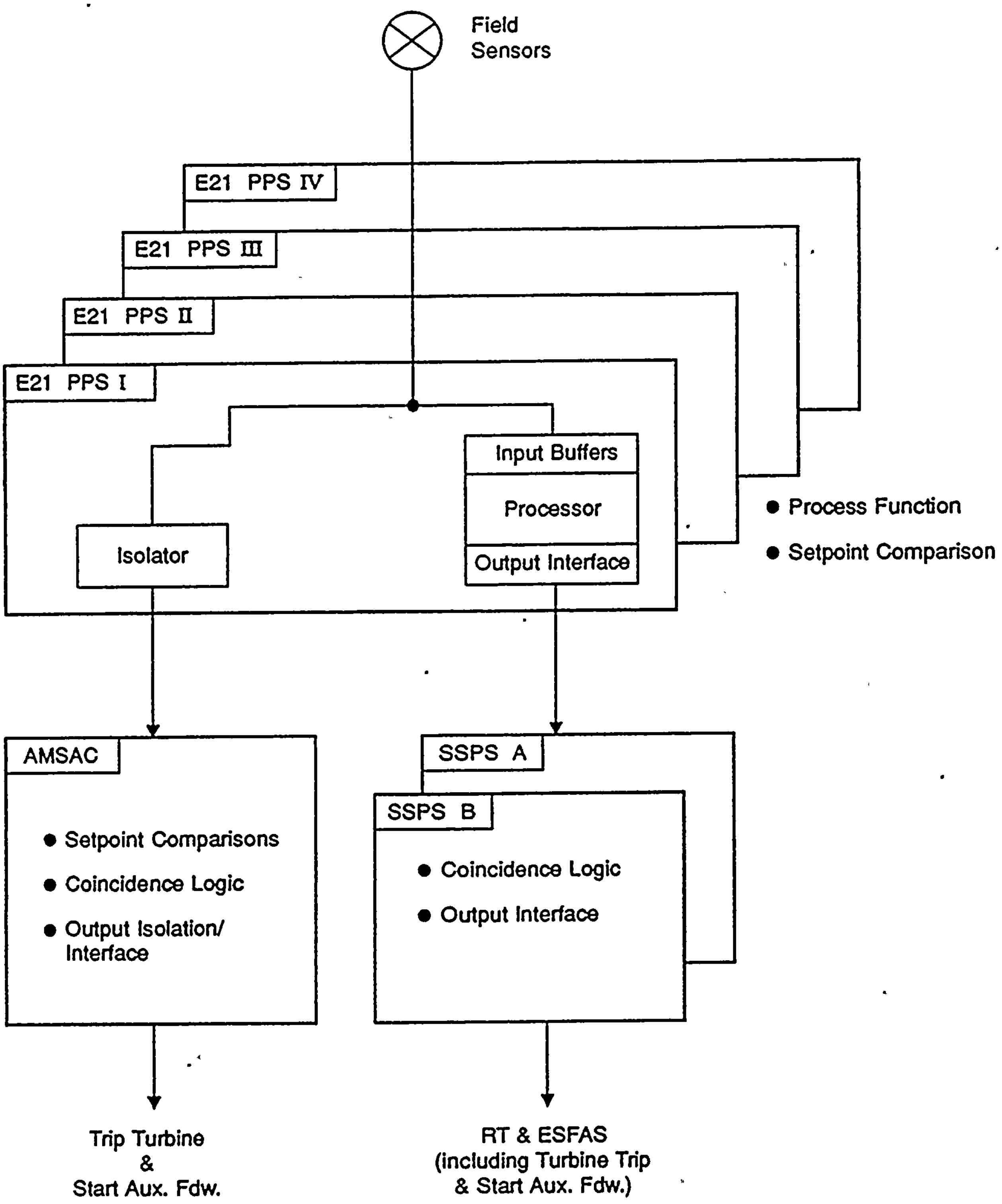


Figure 1. Proposed AMSAC/RPS Functions

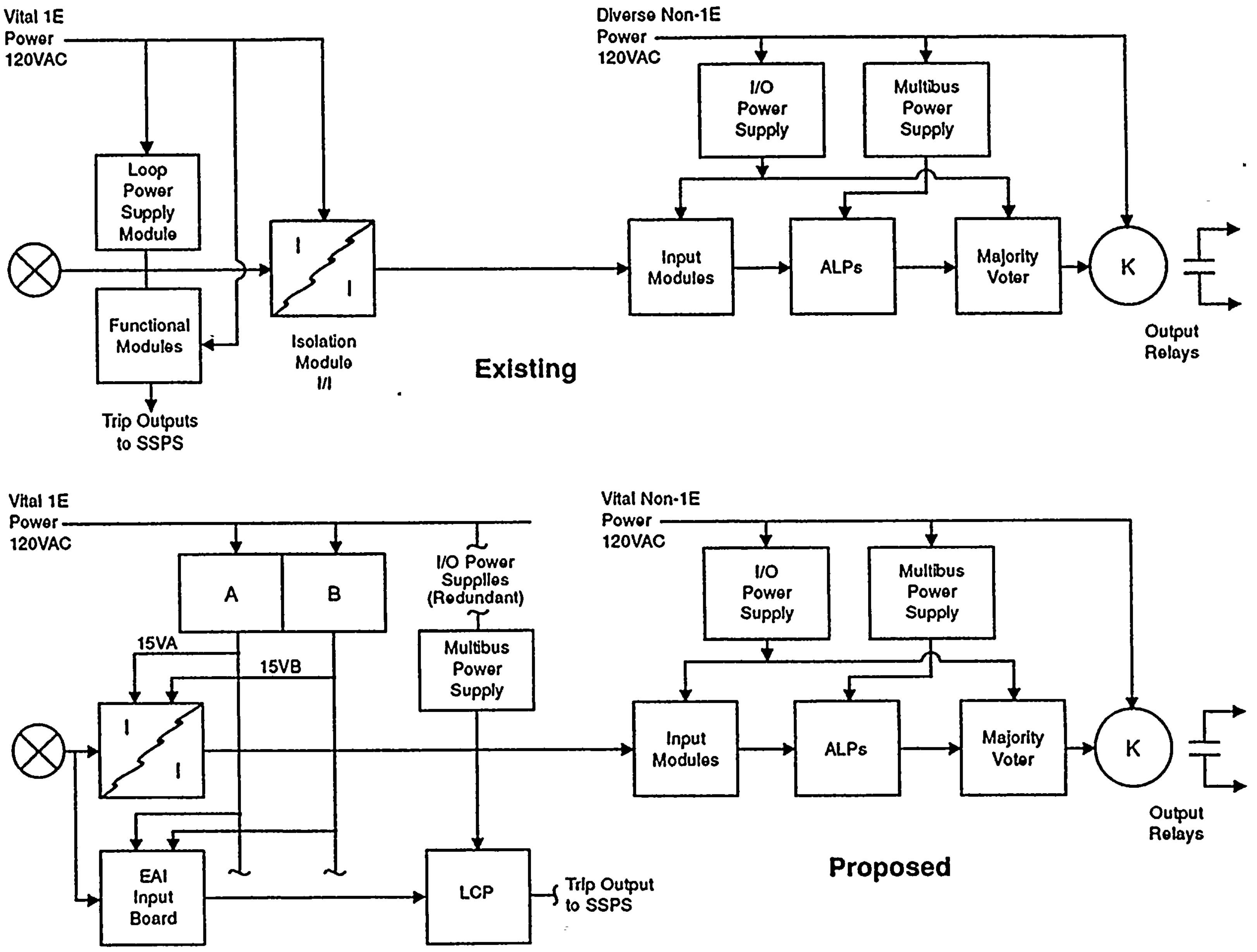


Figure 2. Power Supply Interfaces

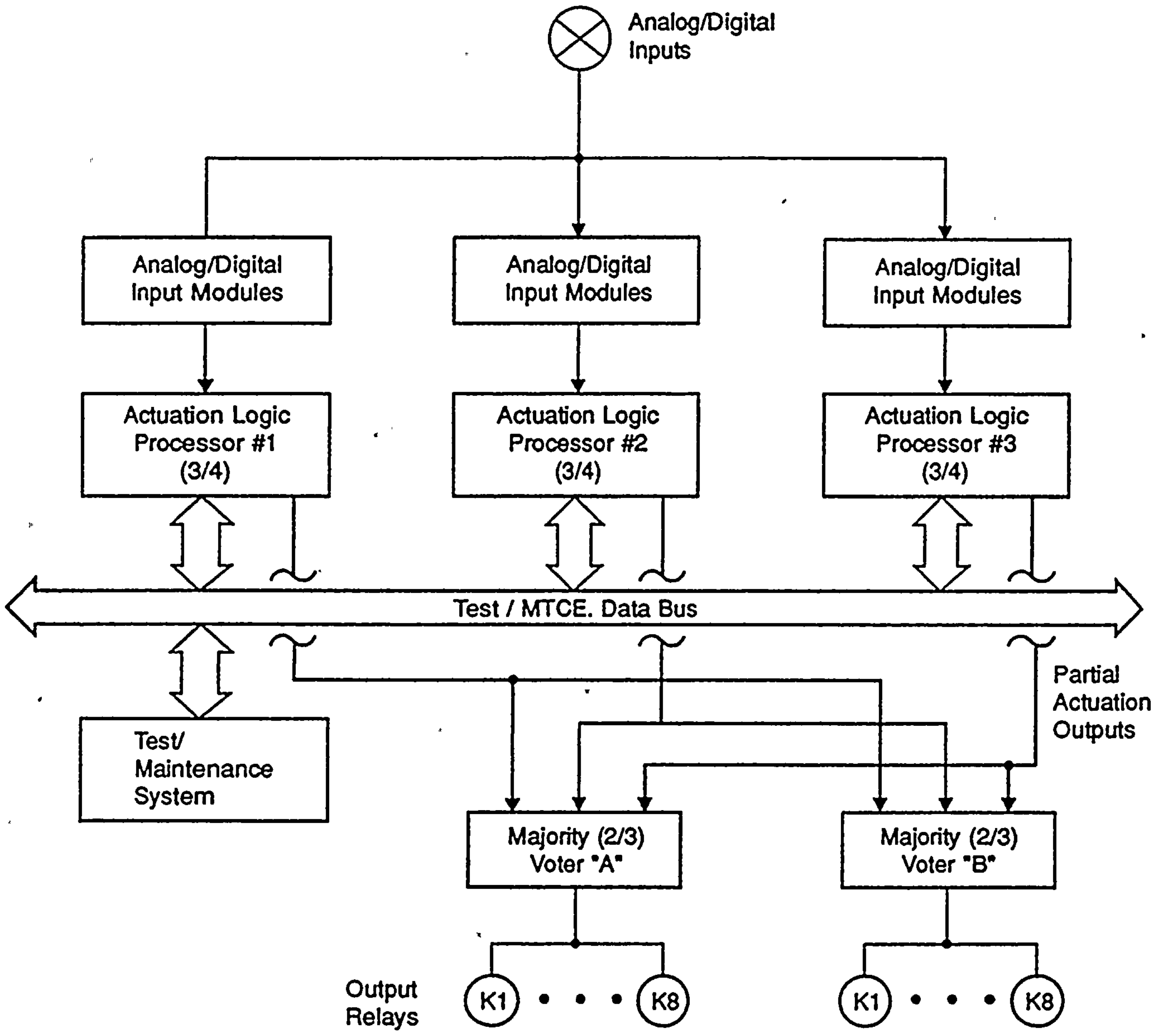

Analog/Digital Inputs
Analog/Digital Input Modules
Analog/Digital Input Modules
Analog/Digital Input Modules
Actuation Logic Processor #1 (3/4)
Actuation Logic Processor #2 (3/4)
Actuation Logic Processor #3 (3/4)
Test / MTCE. Data Bus
Test/ Maintenance System
Partial Actuation Outputs
Majority (2/3) Voter "A"
Majority (2/3) Voter "B"
Output Relays
K1
K8
K1
K8

Figure 3. Simplified AMSAC Block Diagram

Unlike AMSAC, Eagle 21 is not redundant within itself. Four protection sets of Eagle 21 cabinets are utilized along with two logic trains to assure safe operation of the plant. A simplified block diagram of the internal architecture of a single Eagle 21 cabinet is shown in Figure 4. Sensor inputs are processed by analog input modules and accessed by a Digital Filter Processor (DFP) which performs software filtering and A/D conversion. The Loop CalculationProcessor (LCP) processes the filtered values and provides reactor trip and engineered safeguards initiation signals. These signals are output to the Solid State Protection System (SSPS) via the Digital-to-Digital Converter board and the Eagle Partial Trip (EPT) output boards. The SSPS logic trains A and B perform coincidence voting logic using inputs from Eagle 21 (from the four protection sets) and sends reactor trip signals to the Reactor Trip Switchgear (RTS) and engineered safeguards actuation signals to appropriate plant components. It should be noted that the SSPS is independent and diverse from both the Eagle 21 and AMSAC systems.

2.3.3 Failure Modes and Summary

Each of these systems has been designed to react to failures in differing ways. The Eagle 21 system has been designed to fail in the safe direction. AMSAC has been designed to prevent inadvertent actuations since it is a backup to the protection system. The following table addresses the results of various failures in both of the systems:

Failure	Eagle 21		AMSAC	
	Action	**Indication**	**Action**	**Indication**
Loss of Input Channel	Inadvertent Trip of Affected Channel or Failure to Trip Affected Channel	MCR Alarm and Local Ind.	None, Mitigative Action Provided if Called for by Other Channels	MCR Alarm and Local Ind.
Processo r Halt	All Rack Trips Initiated	MCR Alarm and Local Ind.	None, Mitigative Action Provided if Called for by Other Processors	MCR Alarm and Local Ind.
Loss of Power	All Rack Trips Initiated	MCR Alarm and Local Ind.	None, Mitigative Action not Available	MCR Alarm and Local Ind.
Loss of Output Channel	Inadvertent Trip of Affected Channel or Failure to Trip Affected Channels	MCR Alarm and Local Ind.	None, Mitigative Action Still Possible	MCR Alarm and Local Ind.

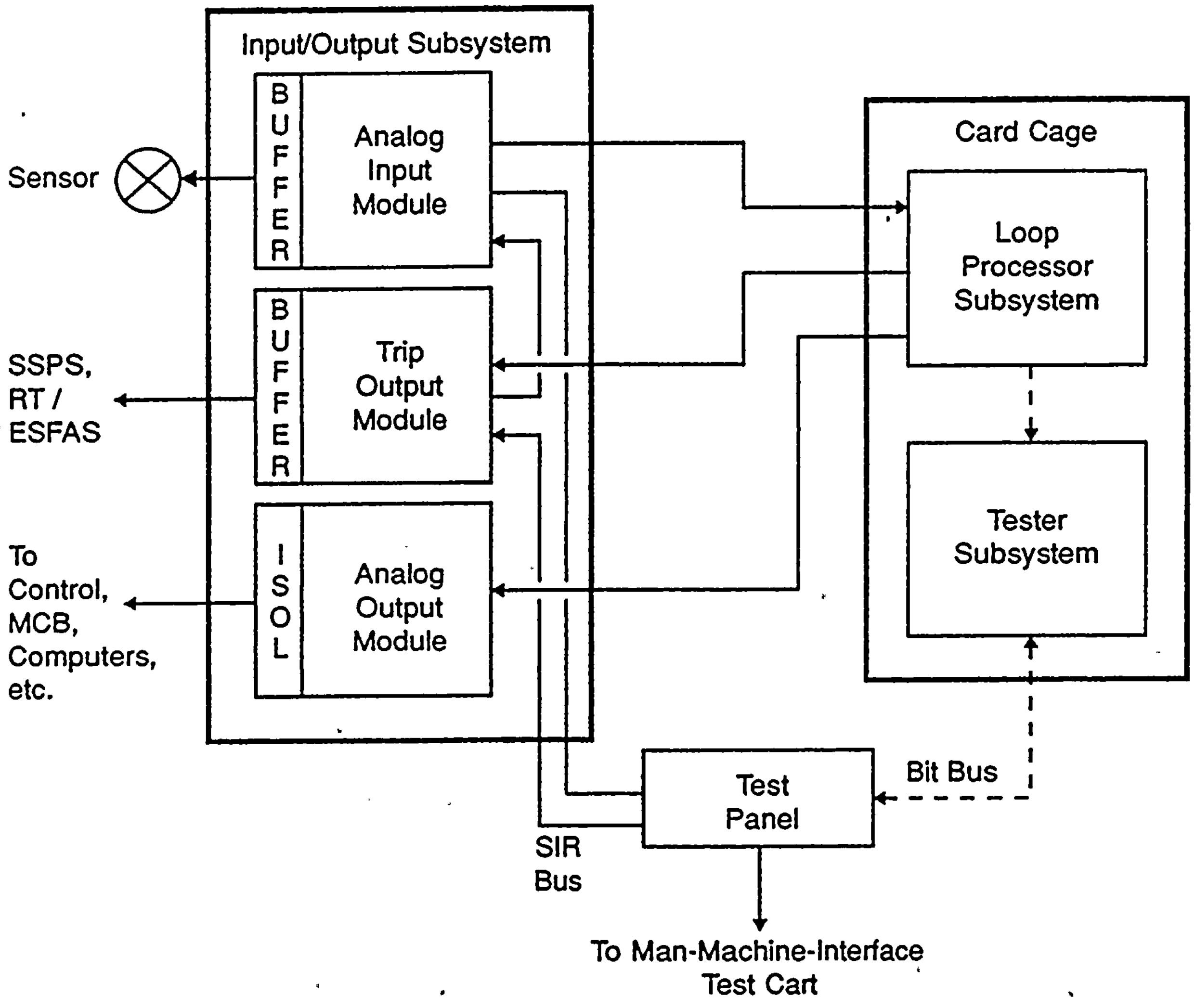


Figure 4. Eagle 21 Block Diagram

The power supply interfaces for the existing and proposed installations are shown in Figure 2. In the existing design the AMSAC signal is provided from a 7100 series isolation module. The sensor and loop power supply are shared with the process protection system as permitted by the AMSAC rule. This design is not immune to a single power supply failure. Failure of the vital 1E bus or internal failures of the DC power supply circuits in either the 7100 series loop power supply or isolator modules will disable the signal to AMSAC. In the proposed design, the AMSAC signal is provided by a Current Loop Input (CLI) isolation module. In the proposed design, the sensor and loop power supply (provided by the EAI board) are also shared. However, as with the existing design, failure of the vital 1E bus will disable the signal to AMSAC. The proposed design is more robust than the existing design in that a single failure of a DC power supply will not disable the signal to AMSAC since the CLI isolation module receives redundant DC power.

2.4 System Diversity

The AMSAC and Eagle 21 architectures, illustrated in Figures 2 and 3, are significantly different in structure and have been designed with different performance characteristics as a basis.

The AMSAC Actuation Logic Processors (ALP) read process inputs, perform the AMSAC algorithm and then provide either a "TRUE" or "FALSE" signal to the Majority Voters. Upon initiation of a "TRUE" signal, AMSAC initiates a turbine trip, starts auxiliary feedwater flow, and isolates steam generator blowdown.

The Eagle 21 Loop Calculation Processor (LCP) reads process inputs, performs the protection algorithm and outputs a "TRIPPED" or "NOT-TRIPPED" signal to the voting logic system (SSPS) via the Trip Output Module. The "NOT-TRIPPED" signal is a pulse and the "TRIPPED" signal is the absence of a pulse. The Trip Output Module is designed with a deadman timer such that an absence of a pulse will TRIP the output. This design is fail-safe in that a processor halt will trip the output and generate a partial trip or engineered safeguards request in the affected channel.

A failure that would prevent the Eagle 21 system from tripping and the AMSAC from actuating is extremely difficult to postulate due to this difference in output processing. Such a failure would have to cause the Eagle 21 LCP to pulse the Trip Output module with the same failure preventing the AMSAC from outputing a "TRUE" signal to the majority Voters.

This system level, or functional, diversity provides very high assurance that either Eagle 21 or AMSAC will provide their protective or mitigative functions. This principle is referenced in the following hardware and software discussions when addressing certain failures.

3.0 HARDWARE DIVERSITY

3.1 Introduction

AMSAC is a stand alone system, one rack per unit, which is designed to initiate auxiliary feedwater flow and turbine trip. The output of each majority voter is fed to a set of train related relays that provide the actuation signals to initiate these functions. The design of this system is in accordance with WCAP 10858P-A; AMSAC Generic Design Package, which was approved by the Nuclear Regulatory Commission (NRC).

Eagle 21 is being installed at Diablo Canyon as a digital replacement for the Westinghouse 7100 Series Process Protection System (PPS). The Eagle 21 was designed as a form, fit and functional replacement for the 7100 system. The system inputs and outputs are preserved and the replacement has no affect on the existing external interfaces. As a result, the Eagle 21 PPS has four redundant protection sets as did the 7100 system it replaced.

This section evaluates significant components common to the two systems. The evaluation supports the judgement that sufficient diversity exists to meet the requirements of 10CFR 50.62 and the ATWS final rule.

3.2 Hardware Differences between AMSAC & Eagle 21

During AMSAC system development, AMSAC hardware modules and components such as microprocessors were specifically chosen to be diverse from those planned for use in the Eagle 21 System.

The configuration and hardware differences between AMSAC and Eagle 21 are listed as follows:

	AMSAC	Eagle 21
Input Processor	Triple redundant Isolated input	Single channel input Xmitter power supply
Analog/Digital Conversion	Triple redundant [ ]+a,c Multimodules	Single [ ]+a,c Digital Filter Processor (DFP)
Microprocessor	Triple redundant [ ]+a,c Single board computers	Single [ ]+a,c Single board computer
Output Processing	Two train majority voter output	Single channel isolated output
Actuation Interface	Two trains (A&B) output relays	Solid State Protection System

3.3 Analysis of AMSAC and Eagle 21 Microprocessors

3.3.1 Introduction

As part of the Eagle 21/ AMSAC diversity justification, the [ ]$^{+a,c}$ processor used by the Eagle 21 system Loop Calculation Processor was evaluated versus the [ ]$^{+a,c}$ processor used by the AMSAC Actuation Loop Processor. The results of this evaluation are provided below and demonstrate that the [ ]$^{+a,c}$ processor is sufficiently diverse from the [ ]$^{+a,c}$ processor to meet the requirements of 10CFR 50.62 and the ATWS Final Rule.

The processors are compared on four levels:

- Performance Parameters
- Physical Packaging
- Internal Architecture
- External Interfaces

3.3.2 Performance Parameters

The [ ]$^{+a,c}$ processor was introduced by Intel in the late 1970's. The [ ]$^{+a,c}$ processor was introduced by Intel in the mid 1980's. The [ ]$^{+a,c}$ processor was developed as an upward compatible upgrade to the [ ]$^{+a,c}$ processor. Although some external interfaces, such as instruction sets, are the same to support upward compatibility, the [ ]$^{+a,c}$ design is completely different from that of the [ ]$^{+a,c}$. These differences are required to support the enhanced performance of the [ ]$^{+a,c}$.

The performance improvements of the [ ]$^{+a,c}$ are due to an improved internal architecture. The [ ]$^{+a,c}$ interfaces to external devices (memory, DMA controllers, external bus interface chips) via the local bus have been greatly enhanced versus the [ ]$^{+a,c}$. These improvements further enhance system performance, and will be discussed further as part of the Internal Architecture and External Interfaces discussions which follow:

- [ ]$^{+a,c}$ Performance Parameters
  Data Path = [ ]$^{+a,c}$ bits
  Clock Speed = [ ] $^{+a,c}$ MHz (as used in AMSAC ALP)
  Memory Address Capability = [ ]$^{+a,c}$ M
  Local Bus Interface = [ ] $^{+a,c}$ bus, time multiplexed bus

- [ ]$^{+a,c}$ Performance Parameters
  Data Path = [ ]$^{+a,c}$ bits
  Clock Speed = [ ]$^{+a,c}$ MHz (as used in Eagle 21 LCP)
  Memory Address Capability = [ ]$^{+a,c}$ M ([ ] $^{+a,c}$ M of this is used by Eagle 21 LCP)
  Local Bus Interface = [ ] $^{+a,c}$ bus, pipelined timing allows overlapped bus cycles

The increased performance requirements of the [ ]+a,c versus the [ ]+a,c required a different physical package, different internal architecture and different external interfaces. These differences are described below.

3.3.3 Physical Packaging

The [ ]+a,c processor is packaged in a 40 pin DIP package. The [ ]+a,c processor is packaged in a 68 pin leadless chip carrier (LCC) package. The [ ]+a,c is implemented using HMOS technology whereas the [ ]+a,c is implemented using more advanced HMOS III technology. Figure 5 shows the physical packaging of the [ ]+a,c processors.

The packaging of these two products is completely different. The performance requirements of the [ ]+a,c require it to have a much larger gate count than the [ ]+a,c. This larger gate count necessitates greater chip density which requires a different manufacturing process.

The [ ]+a,c also requires more pins (63 vs. 40, 5 pins on the 68 pin [ ]+a,c package are not used) than the [8086]+a,c. Separate address and data lines are provided for the [ ]+a,c local bus interface whereas multiplexed address/data lines are provided for the [ ]+a,c local bus interface. Separating the address and data lines allows the [ ]+a,c local bus to utilize pipelined addressing which improves processor access times to memory, system busses and other resources. This significantly improves processor performance.

The structural and functional differences require different manufacturing processes. The components thus are not subject to common defects introduced by manufacturing processes.

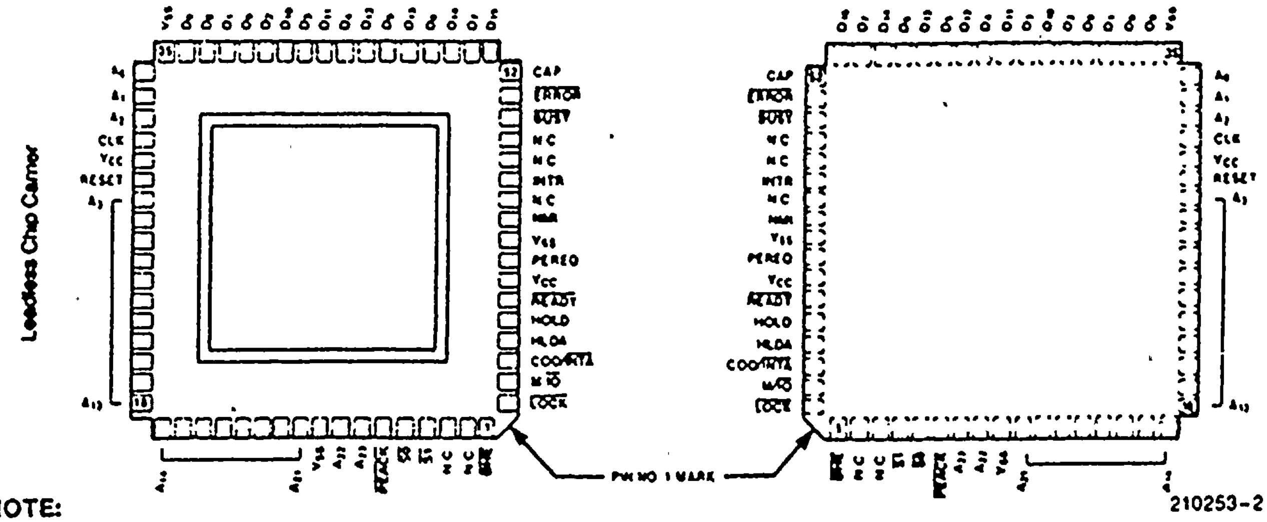
3.3.4 Internal Architecture

The [ ]+a,c internal architectures are shown in Figure 6. In general, the [ ]+a,c internal unit functions have been optimized and further segmented to improve processor performance. A discussion of each unit's function and the differences between the [ ]+a,c is presented below.

- Bus Unit [ ]+a,c — This unit is called the Bus Interface Unit on the [ ]+a,c. The Bus Unit performs all bus operations for the CPU, generating the address, data, and command signals required to access external memory and I/O. The Bus Unit also contains the interface to processor extensions. When not performing bus duties, the Bus Unit "looks ahead" and pre-fetches instructions from memory.

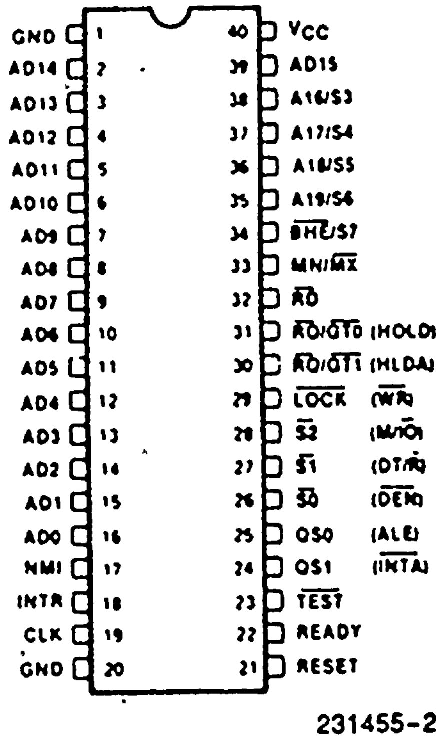
  For both the [ ]+a,c, the Bus Unit will pre-fetch up to six instructions and store these instructions in the pre-fetch queue. The Bus Unit will attempt to fill the pre-fetch queue when at least two bytes are empty.

  The significant difference between the Bus Unit for the [ ]+a,c and the [ ]+a,c is the local bus interface. As was described in the Performance Parameters discussion above the [ ]+a,c utilizes a 2-cycle pipelined timing local bus. The [ ]+a,c utilizes a 4-cycle time multiplexed bus. The [ ]+a,c physical design does not require multiplexing of the local bus address and data lines thus allowing pipelined timing which provides improved local bus performance.



NOTE:
N.C. signals must not be connected

[ ]+a,c



40 Lead

[ ]+a,c

Figure 5. [ ]+a,c

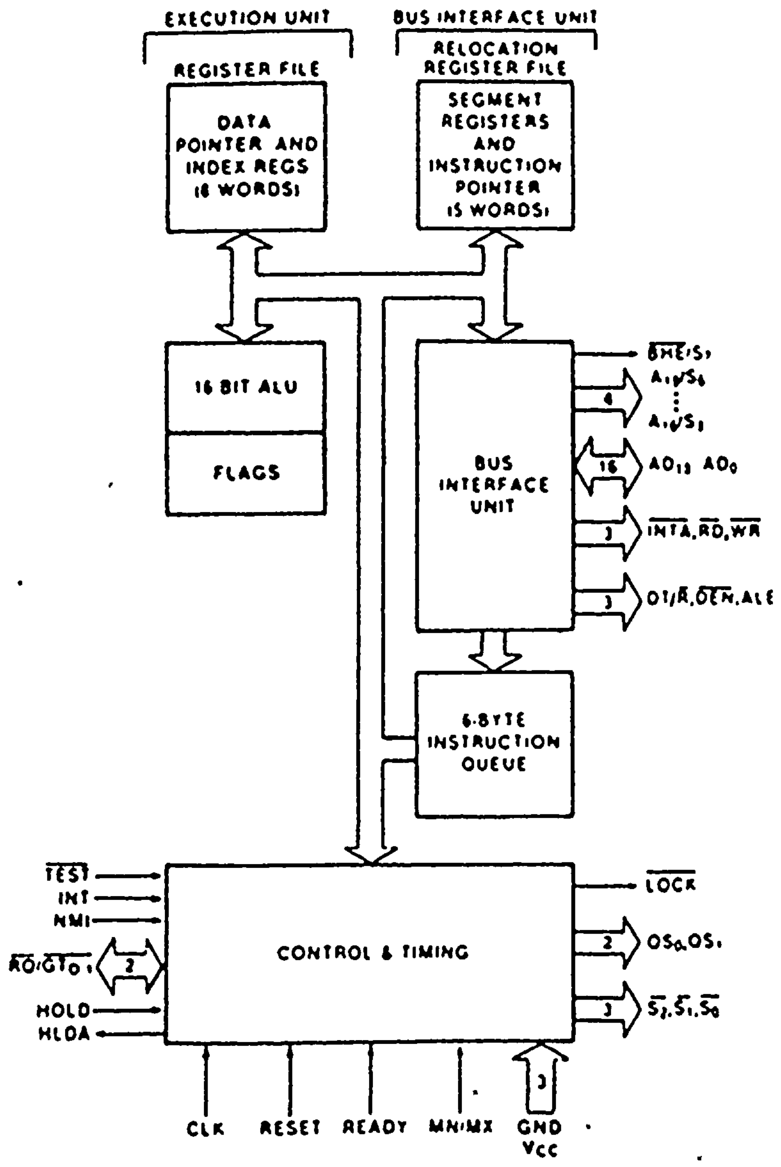
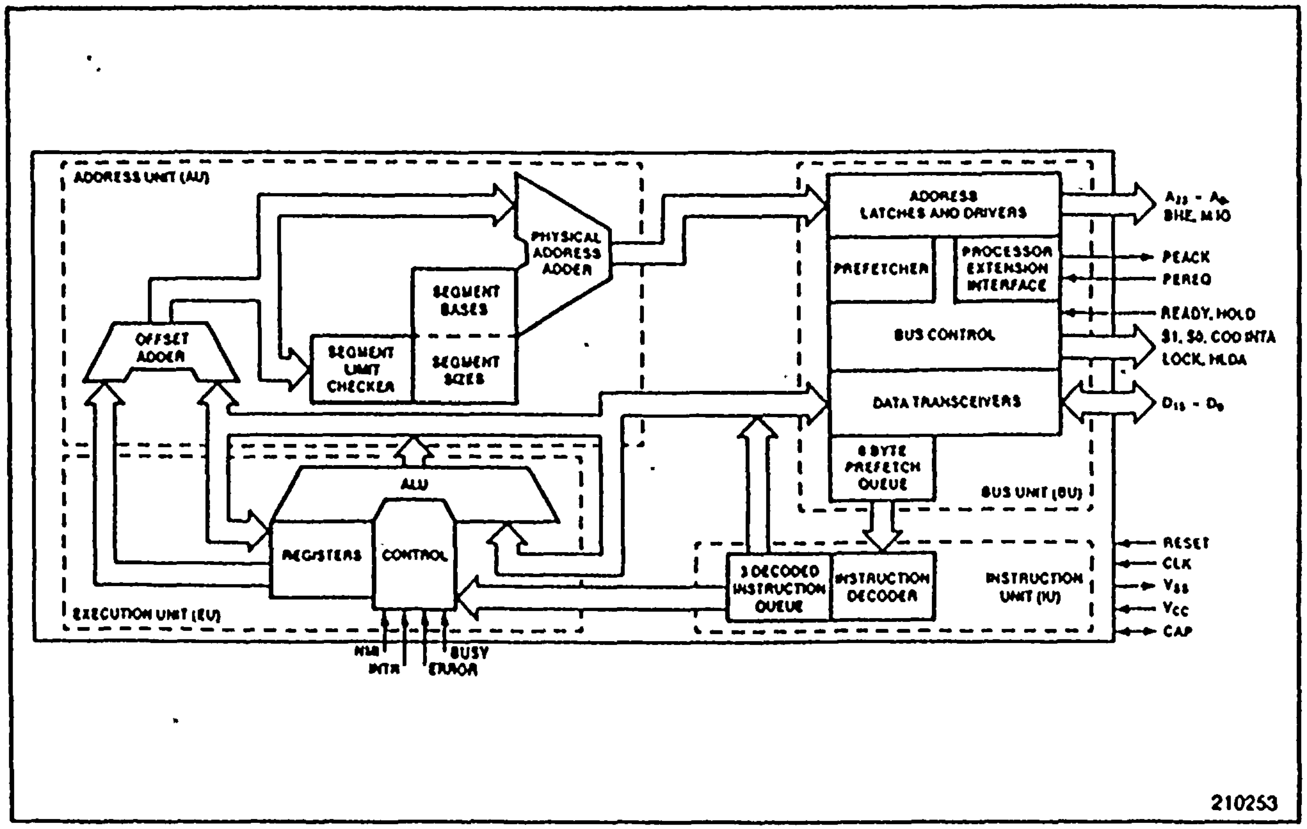


Figure 6. [ ]+a,c Internal Architecture



[ ]+a,c

Figure 6. [ ]+a,c Internal Architecture (Cont'd)

- Instruction Unit [ ]+a,c — The Instruction Unit on the [ ]+a,c receives instructions from the pre-fetch queue, decodes them, and places these fully decoded instructions into a 3 deep instruction queue for use by the execution unit. There is no comparable unit in the [ ]+a,c processor.

- Execution Unit [ ]+a,c — The Execution Unit on the [ ]+a,c decodes instructions from the pre-fetch queue and executes. The Execution Unit on the [ ]+a,c fetches decoded instructions from the Instruction Unit and executes. Both the [ ]+a,c Execution Units contain Arithmetic Logic Units (ALU) for manipulation of registers and instruction operands.

  The significant difference between the Execution Unit for the [ ]+a,c is that the [ ]+a,c Execution Unit does not need to decode instructions since this function is performed by the Instruction Unit. In addition, the Instruction Unit contains a 3 instruction code queue which allows the units to run asynchronously. This feature improves [ ]+a,c instruction processing speed versus that of the [ ]+a,c.

- Address Unit [ ]+a,c — The address unit on the [ ]+a,c provides the memory management and protection services for the CPU and translates the logical addresses into physical addresses for use by the Bus Unit. A register cache in the Address Unit contains the information used to perform the various memory translation and protection checks for each local bus cycle.

  There is no unit comparable to the [ ]+a,c Address Unit in the [ ]+a,c. Translation of logical addresses into physical addresses is performed by the Bus Interface Unit on the [ ]+a,c..

  The internal architecture of the [ ]+a,c is significantly different from that of the [ ]+a,c. The [ ]+a,c Bus Unit local bus interface utilizes pipelined timing which allows overlapped bus cycles versus the [ ]+a,c time multiplexed bus. The [ ]+a,c Instruction and Address Units offload the Bus and Execution Units which increases instruction processing speed.

3.3.5 External Interfaces

The [ ]+a,c processors interface to external devices via the local bus. As has been described above, the local bus for the [ ]+a,c is significantly different from that of the [ ]+a,c. See Figure 7 for a diagram of local bus implementations used for the [ ]+a,c processors which is representative of those used for the Eagle 21 LCP and the AMSAC ALP processor boards.

The local bus for both the [ ]+a,c processors connects memory and I/O resources to the processor using address, data, status and control signals. These address, data and control signals allow the processor to fetch and execute instructions, to manipulate information from both memory and I/O devices, and to respond to processor extension requests.

The [ ]+a,c local bus connects memory and I/O resources to the [ ]+a,c processor, using 24 separate address lines, 16 data lines and a number of status and control signals. The [ ]+a,c local bus uses pipelined address timing. To achieve high bus throughput, the pipelined address timing used by the [ ]+a,c allows overlapped bus cycles when accessing

memory and I/O. The resulting increase in bus throughput is achieved without requiring a proportional increase in memory speed.

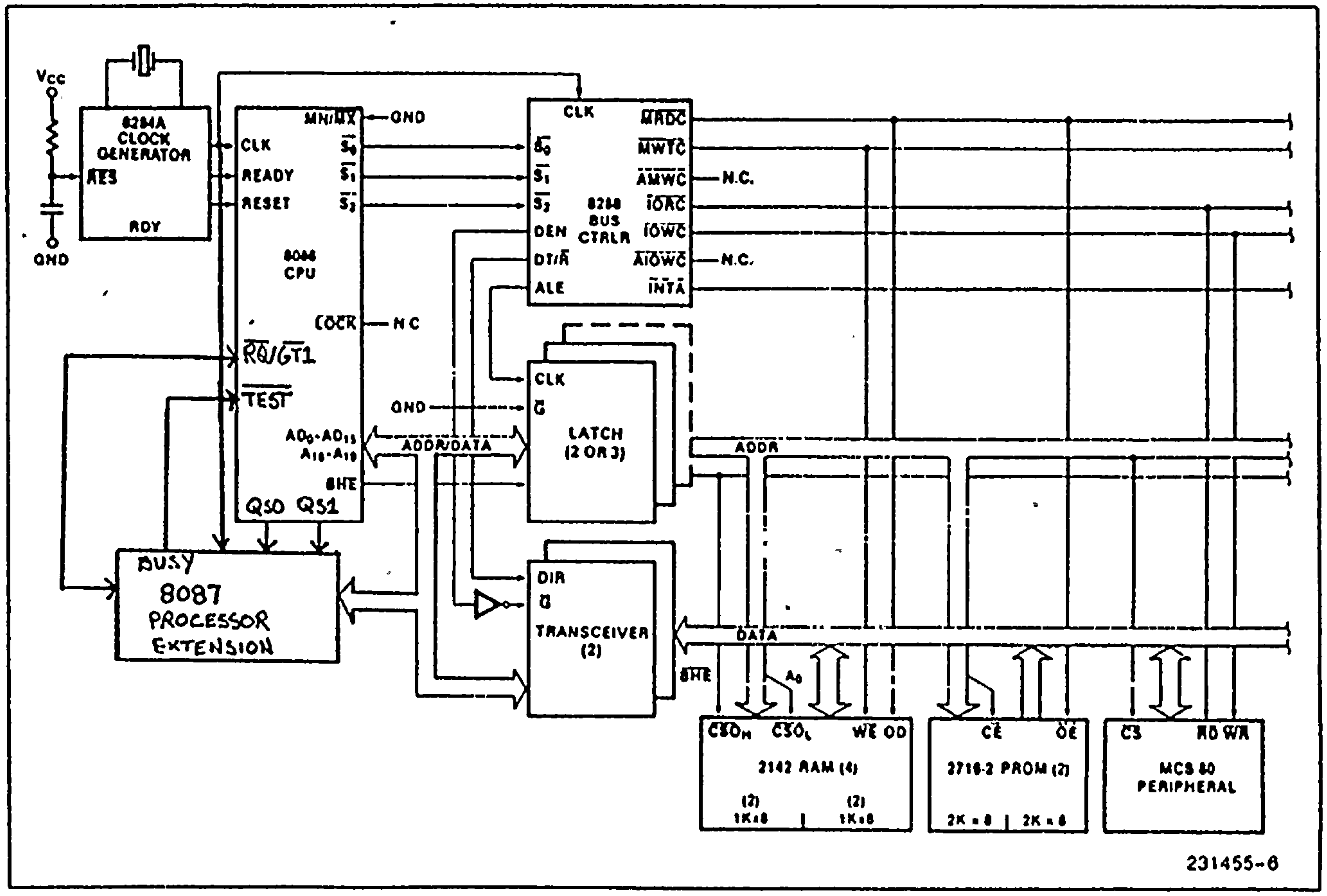
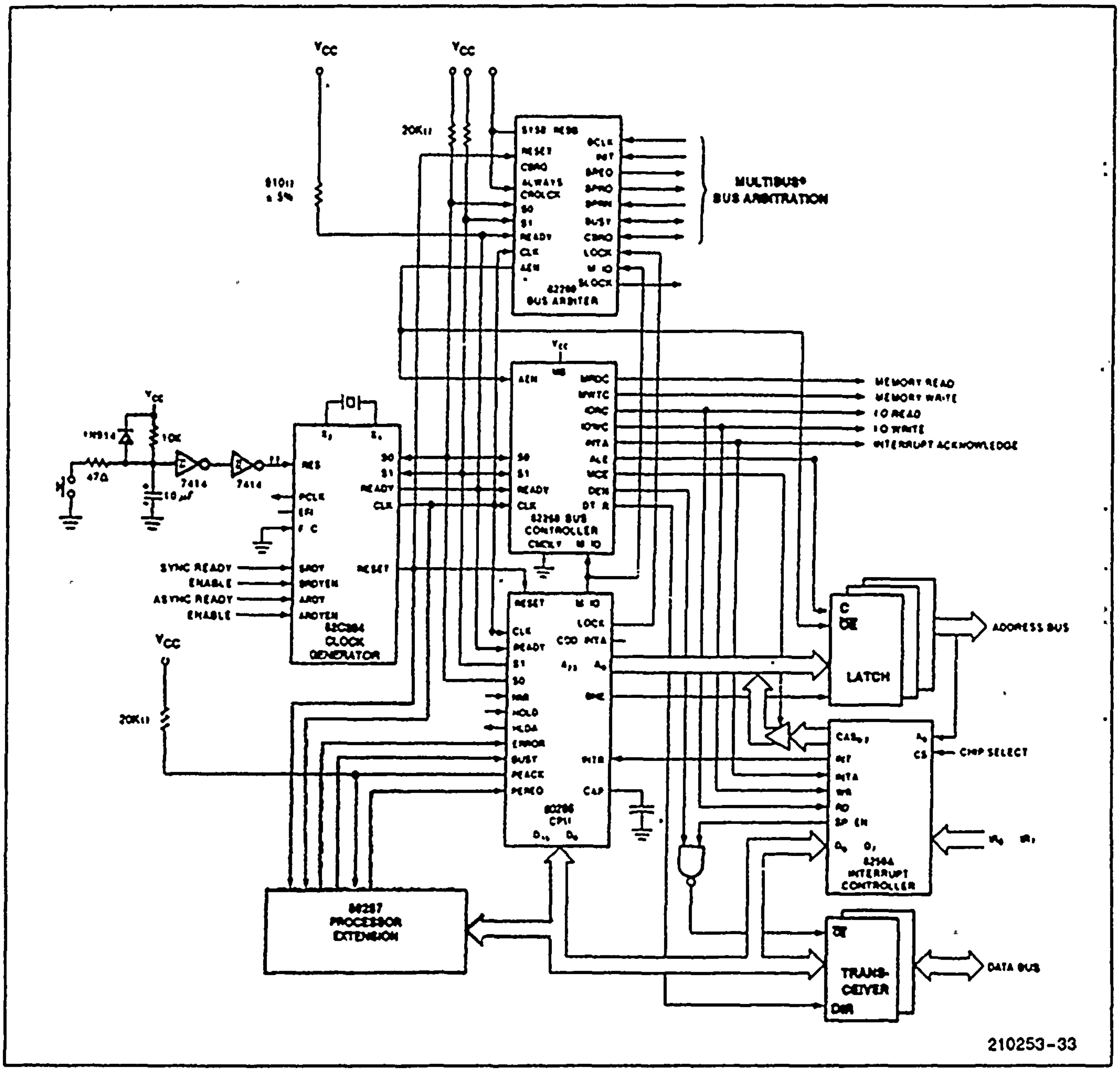


Figure 7. [ ]+a,c External Interfaces

[ ]+a,c



[ ]+a,c

Figure 7. [ ]+a,c External Interfaces (Cont'd)

The [ ]$^{+a,c}$ utilizes a two period clock bus cycle with the address for the next bus operation available during the second clock cycle (pipelining). Data is available during the second bus cycle. Using pipelined timing, the [ ]$^{+a,c}$ places the address for the next memory or I/O operation on the bus even before the previous bus operation has been completed. This overlapping (pipelining) of successive bus operations permits the maximum address setup time before data is required by the CPU or memory. The use of pipelined timing allows a [ ]$^{+a,c}$ bus cycle to complete in two clock periods and greatly improves bus throughput and processor performance.

The [ ]$^{+a,c}$ local bus connects memory and I/O to the [ ]$^{+a,c}$ processor using 16 multiplexed address/data lines and 4 multiplexed address/status lines and a number of additional status and control signals. The [ ]$^{+a,c}$ uses a time-multiplexed bus. Each bus cycle consists of a minimum of 4 clock cycles as follows (1) address, (2) buffer (3&4) data.

As can be seen from the above descriptions the [ ]$^{+a,c}$ local bus designs are significantly different.

3.3.6 Evaluation Conclusion

The [ ]$^{+a,c}$ processors have been evaluated at four key levels for diversity. Although the processors utilize the same instruction set all other facets of the design of these processors are different. Use of the same instruction set will be addressed as part of the software diversity evaluation. This evaluation illustrates the significant differences between the [ ]$^{+a,c}$ processors, and supports the judgement that the AMSAC and Eagle 21 microprocessors meet the diversity requirements of 10CFR 50.62 and the ATWS Final Rule even though they are supplied by the same manufacturer.

3.4 Evaluation of A/D Conversion in AMSAC and Eagle 21

3.4.1 Introduction

The following evaluation will compare the A/D conversion process implemented in both AMSAC and Eagle 21.

3.4.2 A/D Conversion Execution

A/D conversion as executed in AMSAC, is performed by an [ ]$^{+a,c}$ A/D Conversion Multimodule. A/D conversion as executed in Eagle 21, is performed by an [ ]$^{+a,c}$ slave processor board. The components executing the A/D conversion on each board are the same and are listed as follows:

+a,c

[HI-508A	Analog Multiplexer
LF398	Sample and Hold Register
DAC-80	Digital to Analog Converter
DM2504]	Successive Approximation Register

The hardware sequence for the A/D conversion is as follows:

1. Channel for conversion is selected by the Analog Multiplexer, [ ]$^{+a,c}$.
2. Analog voltage stored in the Sample and Hold Register, [ ]$^{+a,c}$.
3. Successive Approximation Register (SAR) sequences bit by bit from the most significant bit to the least significant bit to establish the 12 bit digital representation of the analog input. [ ]$^{+a,c}$.
4. As each bit is cycled by the SAR, the D/A converter, [ ]$^{+a,c}$, inputs an analog voltage into the Sample and Hold Register to compare against the analog voltage. The bit by bit voltage comparison generates the twelve bit word out of the SAR that represents the analog input voltage.

The ALP software executes A/D conversions in AMSAC by controlling the [ ]$^{+a,c}$ multimodule over the onboard [ ]$^{+a,c}$ bus. This function is controlled by the firmware which is resident on the host processor, the [ ]$^{+a,c}$ processor board (ALP).

The DPF software that controls the A/D conversion in Eagle 21 is resident on the [ ]$^{+a,c}$ slave processor board. The control of the conversion is executed via the on-board bus.

The software controls the multiplexer operation, while the remainder of the A/D conversion is executed by hardware under control of the on-board clock.

The common components listed above are extremely reliable and field proven and were first utilized by Intel in 1981. They continue to be manufactured by several component suppliers and are still being used for A/D conversion processes by a variety of original equipment manufacturers today. It should be noted that these components are not unique to Westinghouse or Intel products. They could be part of a common solution to A/D conversion found in many suppliers' digital equipment.

In the unlikely event a common component in the A/D conversion path should fail, there are diagnostics in both AMSAC and Eagle 21 that perform limit checks on the A/D converter outputs. The operator will be informed of such failures and will take corrective action.

3.4.3 Conclusion

There are safeguards in the form of diagnostics; that generate alarms in both AMSAC and Eagle 21 that would alert operations personnel that there is a problem with either of the systems. Immediate action could then be initiated to diagnose and correct the problems with the systems.

3.5 Analysis of Input Signal Conditioning in AMSAC and Eagle 21

The Input boards for both AMSAC and Eagle 21 convert the 4-20 ma input signal to a 0-5 VDC output signal that can be processed by the A/D converter. These boards have a common component on them, the [ ]$^{+a,c}$ isolation amplifier.

These isolation amplifiers are used throughout industry, are highly reliable and are also used in analog systems. Therefore, it is not considered to be practicable or reasonable to maintain diversity of components at this level.

In the unlikely event of a common mode failure of the [ ]$^{+a,c}$ isolation amplifiers, the Eagle 21 would still provide a reactor trip output. However, should the amplifier fail either high or low, the failure would be detected by the A/D converter limit checks and an Eagle 21 Trouble Alarm and an AMSAC General Warning would be generated (see A/D conversion discussion in Section 3.4). These scenarios are described below.

If all [ ]$^{+a,c}$ isolation amplifiers failed either high or low in every Eagle 21 rack, the result would be a reactor trip due to the following:

- Should all signals fail high, the pressurizer high pressure reactor trip and, indirectly, a P14 Turbine Trip and Feedwater isolation would cause a reactor trip since setpoints would be exceeded.
- Should all signals fail low, the reactor coolant low flow reactor trip and steam generator narrow range level reactor trip would be initiated since the setpoints would be exceeded.

Additional functions would also provide reactor trip and engineered safeguards actuations in the unlikely event of this common failure within Eagle 21.

An amplifier drift problem could also be postulated as a common mode failure. In such a case, one or more of the following three situations will occur in Eagle 21:

- Amplifiers drift above or below a setpoint will cause a partial trip or reactor trip.
- Amplifiers drift but do not exceed a setpoint will cause a input quality problem which will generate a Trouble Alarm
- Amplifiers drift but do not cause an input quality problem will be discovered during Plant Shift Check

There are three possible consequences in AMSAC:

- Amplifiers drift above or below a setpoint, cause a trip or partial trip of the AMSAC functions which would cause a General Warning
- Amplifiers drift but do not exceed a setpoint, could cause a hardware failure alarm which will generate a General Warning
- Amplifiers drift and there is no hardware failure, could go undetected until the next surveillance period

3.6 Impact of Common Components on AMSAC and Eagle 21 Microprocessors

There are two active components which are common to the [ ]$^{+a,c}$ in AMSAC and the [ ]$^{+a,c}$ in Eagle 21. They are as follows:

- [ ]$^{+a,c}$ Programmable Interval Timer (PIT) - provides loop cycle time

- [ ]+a,c Programmable Peripheral Interface (PPI) - provides digital outputs for trip and annunciation

Both of the above components, [ ]+a,c, have been widely used throughout industry. They are highly reliable and proven through field use in many systems.

Should the [ ]+a,c fail in Eagle 21, a timer diagnostic periodically compares the output of two timers driven by separate crystals to verify that the counts correlate. Failure of this diagnostic would cause the Eagle 21 processor to initiate error handling which causes a Channel Set Failure Alarm, a processor halt and places all trips in the preferred failure mode as a result of dead man timer actuation.

A failure of the [ ]$^{+a,c}$ in AMSAC is detectable through a timer diagnostic and will result in the processor halting and a General Warning alarm being initiated.

Since the deadman timer refresh pulse is output through the [ ]$^{+a,c}$ in Eagle 21, a common mode failure either high or low will prevent the refresh pulse from resetting the dead man timer in the EPT board. This would cause all trip outputs to be set to the preferred failure mode which would cause a reactor trip. Such a common mode failure would be detected by the Tester Subsystem (partial trip mismatch test) and a channel set failure alarm in all protection sets would be visible within the control room.

The [ ]$^{+a,c}$ provides the output interface for outputs going to the train related majority voter modules within AMSAC. Depending on the failure mode of the chip, there could be affects from no action to an AMSAC actuation being sent to the voter modules.

Since the Eagle 21 utilizes a pulsed output and deadman timer arrangement on the trip output, a common mode failure of the [ ]$^{+a,c}$ will not result in a failure to initiate trip outputs.

3.7 Impact of Power Supply Failures

The power supplies used within both systems are [ ]$^{+a,c}$ series power supplies. These supplies are utilized to power the input/output signal conditioning boards and the Multibus boards in both systems. The Eagle 21 and AMSAC system are designed to operate differently under conditions where a loss of power is encountered. Eagle 21 is designed as a fail-safe system and as such will initiate all reactor trip outputs and provide an alarm output (Channel Set Failure) to the main control room in the event of a common mode loss of either type of power supply.

Unlike Eagle 21, AMSAC requires power to perform its mitigative functions. A similar common mode loss of either power supply would result in an alarm (General Warning) in the control room and no other action.

Power supplies convert AC line voltage to DC for signal conditioning and microprocessor operation and are used in analog as well as digital equipment. Design solutions such as switching power supplies are common throughout the industry. Thus even power supplies made by different manufacturers will have many common components.

In the unlikely event that these power supplies would either fail high or drift low in a common mode, the following actions would result. A power supply failing high will trigger overvoltage protection circuits which will turn off the supply. A power supply failure low will halt the microprocessor due to sensitivity to low voltage. In both cases the Eagle 21 system will fail-safe and initiate all reactor trip outputs and provide an alarm output to the main control room. The AMSAC system will initiate an alarm in the main control room.

3.8 Industry Standard Components

Tables 1, 2 and 3 list all active components found on the microprocessor boards and input/output boards utilized within Eagle 21 and AMSAC. An X indicates where a particular device is used in the systems.

A summary of the active components on the multibus boards (Table 1) which are in the protection/mitigation paths within Eagle 21 and AMSAC is provided below:

Board Type	System	Number of Active Components	Number of Common Active Components*
[ ]$^{+a,c}$	AMSAC	102	
	AMSAC	14	33
	Eagle 21	88	
	Eagle 21	158	

*Number of components common to both systems.

Many of the devices listed in Table 1 are industry standard parts and are produced by several manufacturers. These devices can be broken down into three major categories:

- Microprocessors and Their Peripherals
- A/D Conversion Components
- Common Logic/RAM Chips

In order to illustrate this fact, three parts (one from each category) were chosen from Table 1. The manufacturers for each are listed below:

- [ ]$^{+a,c}$ was originally manufactured by Intel. The present listed manufacturers are AMD, GEC Plessey, Intel, IMI, Krueger, NEC, OKI and Toshiba.
- [ ]$^{+a,c}$ Successive Approximation Register was originally manufactured by Advanced Microdevices. The present listed manufacturers are AdvLinear, National and Rochester.
- [ ]$^{+a,c}$ Hex Buffer/Driver was originally manufactured by Texas Instruments. The present listed manufacturers are Lansdale, National, Rochester, Signetics and TI.

There are certain active devices that are common to each board but are not included in the comparison since they are not common to the protection path in Eagle 21 and the mitigation path in AMSAC. Examples of these are as follows:

- AMSAC does not use the Multibus for microprocessor signal outputs to the field, therefore, no multibus operating devices are listed.
- Interrupts are prohibited on the microprocessors in both AMSAC and Eagle 21. Therefore, the [ ]$^{+a,c}$ Programmable Interrupt Controller (PIC) is disabled and is not listed.

A summary of the active components on the input/output boards (Tables 2 and 3) which are in the protection/mitigation paths within Eagle 21 and AMSAC is provided below:

Board Type	System	Number of Active Components	Number of Common Active Components*
Input Board	AMSAC	28	1
EAI Input Board	Eagle 21	50	
Majority Voter	AMSAC	14	0
EPT Output Board	Eagle 21	39	

*Number common to the AMSAC & Eagle input/output boards.

The only active component on the input/output boards that is common to the protection/mitigation paths in Eagle 21 and AMSAC is the [ ]+a,c Isolation Amplifier. This device is manufactured by only one supplier (Analog Devices) but the common failure modes of this device have been analyzed and found to cause safety actuation or mitigation in Section 3.5

As in the case for the multibus boards, there is one device, the AD2710LN 10V regulator, which is common to the Eagle 21 and AMSAC input boards but is not included in the comparison since it is not part of the AMSAC mitigation path. The AD2710LN is used in AMSAC for off-line testing and thus is not part of the signal path when AMSAC is on-line.

3.9 Conclusions

As can be seen from the material presented in Sections 3.1 to 3.8, the AMSAC and Eagle 21 hardware is very diverse. Their architectures are different in that the AMSAC is a standalone system designed with redundancy in the input processing, microprocessor and output processing. The Eagle 21 is not redundant within each rack; however, there are four protection sets which provide system redundancy.

In the evaluation, performed in Section 3.3, of the microprocessors that are the heart of each system; it was shown that there is a significant difference in architecture between the AMSAC microprocessor [ ]+a,c, and the Eagle 21 microprocessor [ ]+a,c. Although the microcode resident on each chip was not available due to proprietary issues, it is judged that it is different since the architecture is so different.

In the sections where common components or assemblies were identified, 3.4, 3.5, 3.6 and 3.7, failure modes and effects were defined and in every case there were no common failures that would prevent the safe shutdown of the plant.

In the review performed in Section 3.8, there was only one common component, the [ ]+a,c, Isolation Amplifier. Common mode failures of this device were shown in Section 3.5 to have no effect on the safety of the plant. In addition, it was noted that this an industry component with a broad operational base, both analog and digital.

TABLE 1. MICROPROCESSOR ACTIVE COMPONENTS LIST

AMSAC	Eagle

+a,c

TABLE 1. MICROPROCESSOR ACTIVE COMPONENTS LIST (Cont'd)

AMSAC	Eagle

+a,c

TABLE 1. MICROPROCESSOR ACTIVE COMPONENTS LIST (Cont'd)

AMSAC	Eagle

+a,c

TABLE 2. INPUT BOARD ACTIVE COMPONENTS LIST

+a,c

Device	Type	AMSAC Input Board	Eagle 21 Input Board
	5V Regulator	X	
	Dual Peripheral or Driver	X	
	Opto Coupler	X	
		X	
	Operational Amplifier	X	
	Isolation Amplifier	X	X
	DC Converter		X
	Quad Isolated DC/DC Converter		X
	DC/DC Converter		X
	DC/DC Converter		X
	RS 422 Line Driver		X
	2 Input and Gate		X
	Retriggerable Monostable		X
	Hex Inverter		X
	Octal Tri State Buffer		X
	Dual 4:1 Analog MUX/DEMUX		X
	Voltage Regulator		X
	Voltage Regulator		X
	Microcontroller		X
	Op Amp		X

TABLE 3. OUTPUT BOARD ACTIVE COMPONENTS

+a,c

Device	Type	AMSAC Majority Voter	Eagle 21 Trip Output Board
	5V Regulator	X	
		X	
	Optocoupler	X	
	Optocoupler		X
	Optocoupler		X
	Optocoupler		X
	RS-422 Line Driver		X
	2 Input and Gate		X
	Hex Inverter		X
	Monostable		X
	Retriggerable Monostable		X
	Octal Tri State Buffer		X
	Voltage Regulator		X
	Dual Comparator		X
	Microcontroller		X
	Op Amp		X
	Solid State Relay		X

4.0 SOFTWARE DIVERSITY

4.1 Introduction

In evaluating the level of diversity between the software in the Eagle 21 and AMSAC systems, it is most important to note that there are NO software modules shared by the two systems. This includes application, utility and diagnostic software. When developing the AMSAC software, a separate software design team was established to assure that AMSAC would not use software designed for existing products or products under development (e.g. Eagle 21). The AMSAC software was designed and coded from the ground up to address this concern. Separate libraries and controls are utilized by AMSAC and Eagle 21 to assure continued "separation" of the systems' software.

Verification and Validation

Eagle 21 Verification and Validation was performed by an independent V&V team in accordance with Regulatory Guide 1.152 and ANSI/IEEE Std. 7-4.3.2-1982. The Eagle 21 Verification and Validation group was independent of both the Eagle 21 and the AMSAC design groups.

Regulatory guidance did not require formal Verification and Validation to be performed on the AMSAC system. However, the Westinghouse V&V organization did perform independent source code reviews and validation testing on the AMSAC system. In, addition, AMSAC validation was performed by individuals who were not involved with the Eagle 21 validation testing and were from a different organization.

4.2 Source Code Thread Path for Steam Generator Narrow Range Level

4.2.1 Purpose

To show that the Eagle-21 Process Protection System and the AMSAC ATWS Mitigation System Actuation Circuitry software implementations are diverse.

4.2.2 Introduction

The Eagle-21 and AMSAC software implementations will be analyzed to determine the diversity aspects of the systems. The following types of reviews will be accomplished to demonstrate their diversity:

- Thread path of the common input signal low low steam generator level for both the Eagle 21 and AMSAC systems.
- Calling Hierarchies for both systems will be completed; however, these calling hierarchies will only include procedures that are identified in the thread paths and are not comprehensive of both systems.

- Side-by-side comparisons of the procedural calls and functional description and common source code excerpts will be identified for common functions between Eagle 21 and AMSAC systems. These comparisons will be represented in tabular form and will give a clear understanding of the software implementation for both the Eagle 21 and AMSAC systems.

Once the analysis is complete and all data is compiled a final summary identifying areas of commonality will be identified to show common function and the diversity aspects of both systems.

EAGLE-21 Thread Path for Low-Low Steam Generator Level Signals

The following figures, descriptions and source code excerpts will show a thread path for the Low-Low Steam Generator Level Signal as it is processed by the Eagle 21 Process Protection system software. Figure 8 shows an overview of the Low-Low Steam Generator Level Signal as it is processed by the Eagle 21 process protection system.

The signal originates at a differential pressure transmitter and enters Eagle 21 as an analog signal through the signal conditioning board "EAI". The signal is converted to a digital signal by the A/D board 88/40A. This signal is an input to the Trip Time Delay function algorithm which is part of the Loop Calculate Processor (LCP). The signal is then output from the Eagle 21 Process Protection System through the I/O board [ ]$^{+a,c}$ and the Partial Trip Board EPT to the Solid State Protection System (SSPS) for Reactor Trip and ESF actuation.

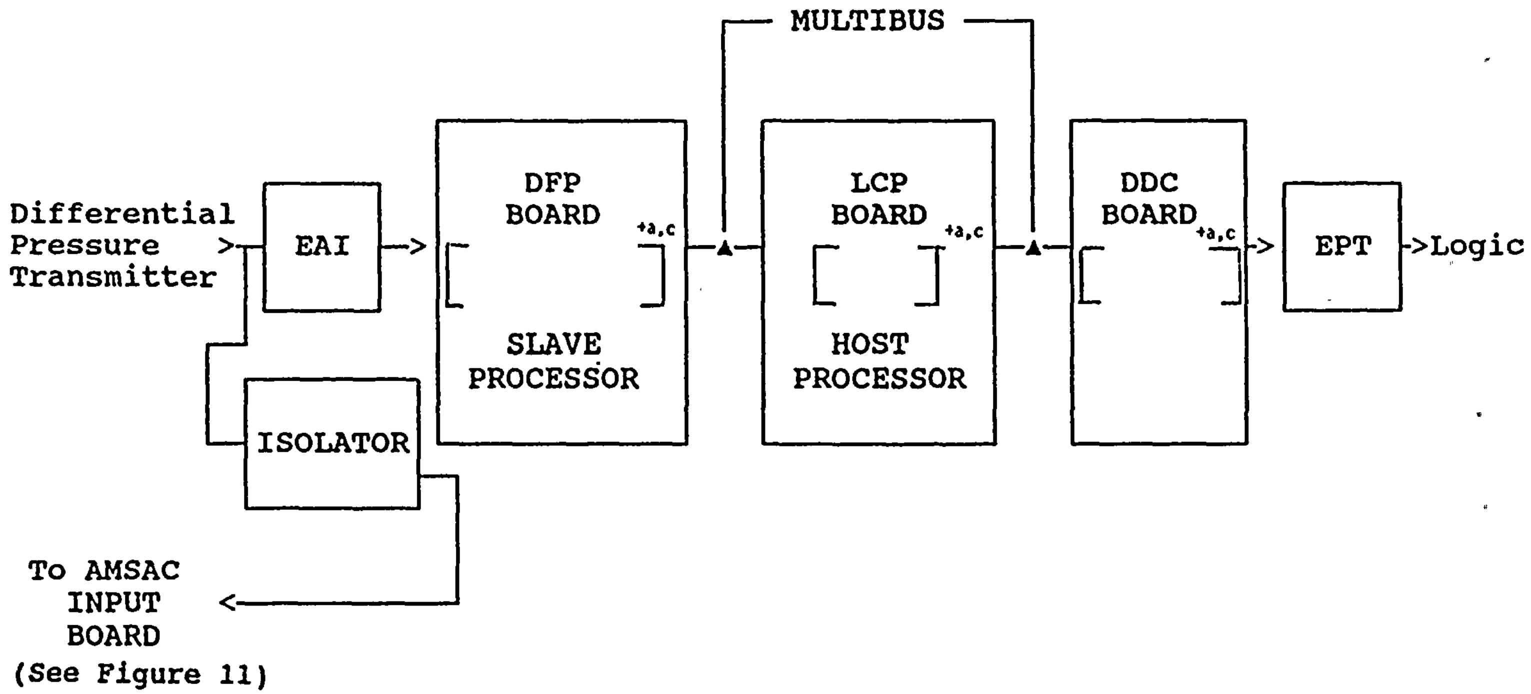


Figure 8. Eagle-21 Input/Output Overview Diagram

The Digital Filter Processors (DFP) in Eagle 21 perform the following procedures to read inputs, filter inputs and store the raw and filtered values in shared memory. These values are then used by the Loop Calculation Processor (LCP) to generate trip requests, plant computer outputs and outputs to the main control board.

1a MAIN_DFP_PROGRAM — This is the Main Program for the Digital Filter Processor board, its main function is to perform the A/D Conversions.

2a IADSM_ACQUIRE_RX_BUFFER — This procedure acquires the communications shared memory buffer between the slave DFP processor and the Host LCP processor.

3a PF_GET_ANALOG_READINGS — This procedure will read and process all the 16 A/D channels. All signals will be processed using three cascaded low pass filters.

4a IADSM_RELEASE_RX_BUFFER — This procedure will release the data buffer to the host LCP processor once the slave processor has placed the data into the shared buffer.

5a,6a IADSM_SEMAPHORE_ACQUIRE & IADSM_SEMAPHORE_RELEASE — These procedures are called from IADSM_ACQUIRE_RX_BUFFER and IADSM_RELEASE_RX_BUFFER and are used to acquire and release various semaphores which are used to test the LCP and DFP hardware locks and determine if they are functional.

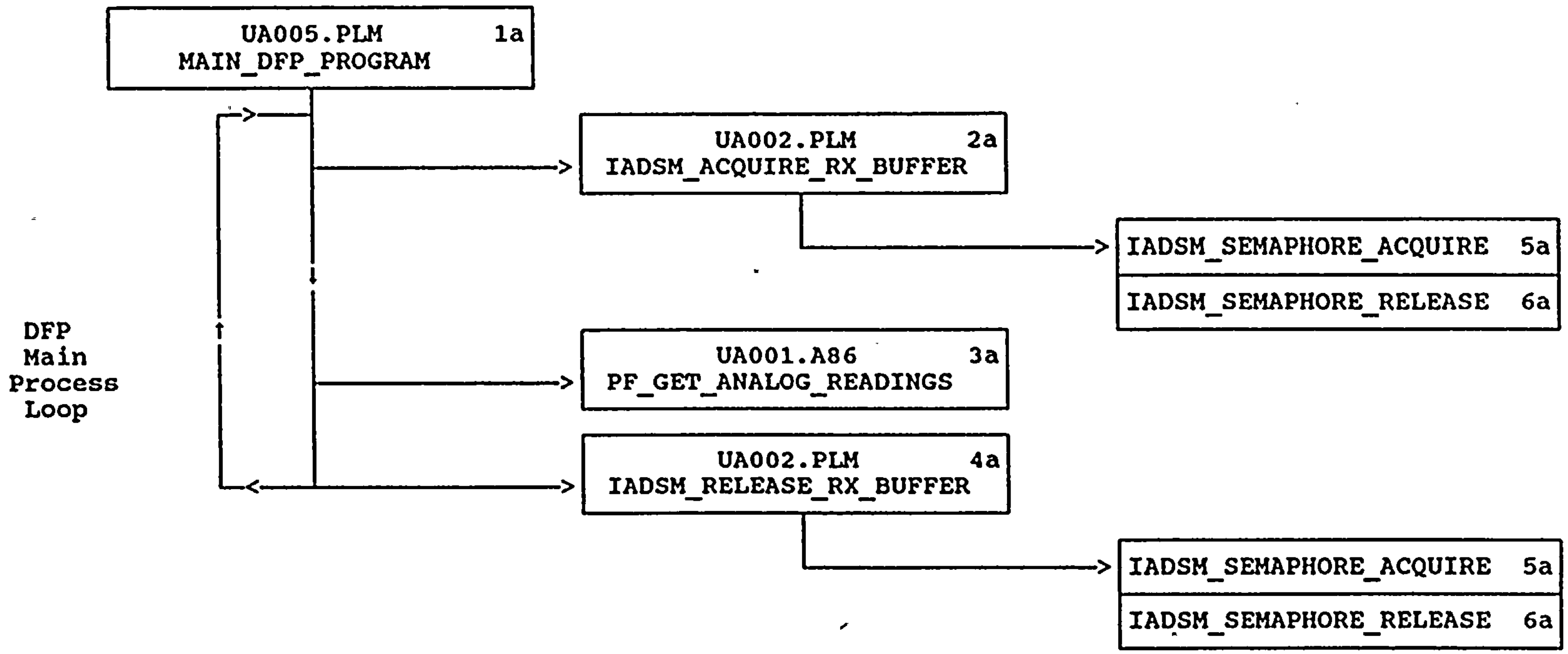


Figure 9. DFP Process Loop

The Loop Calculation Processor (LCP) reads raw and filtered values from the DFP shared memory and then calibrates each analog input, converts inputs to electrical and engineering units, executes protection algorithms, and sends trip outputs via the partial trip output board to the SSPS. An outline of these procedures is provided below and is illustrated in Figure 10.

1b LCP_MAIN — Top level procedure for the Loop Calculation Processor, its function is to perform all the calls to the top level procedures and the main loop cycle of LCP Software.

2b READ_ANALOG_INPUTS — This procedure reads all the analog input values on a per channel basis for an individual protection rack and sets the values into a common data structure.

3b GET_DFP_DATA — This procedure moves the analog input data from the shared memory on the DFP boards into the LCP RAM memory. It utilizes procedural calls HSM_RELEASE_RX_BUFFER and HSM_ACQUIRE_RX_BUFFER to acquire the latest DFP analog input data buffer.

4b HSM_ACQUIRE_RX_BUFFER — This procedure acquires the newest buffer from the slave DFP processor.

5b HSM_RELEASE_RX_BUFFER — This procedure will release the data buffer back to the slave processor once the host LCP processor has used the data buffer.

6b,7b HSM_SEMAPHORE_ACQUIRE & HSM_SEMAPHORE_RELEASE — These procedures are called from HSM_ACQUIRE_RX_BUFFER and HSM_RELEASE_RX_BUFFER and are used to acquire and release various semaphores which are used to test the LCP and DFP hardware locks and determine if they are functional.

8b CALCULATE_GAIN_AND_OFFSET — This procedure calculates the gain and offset for the valid analog input values.

9b READ_AI_SMALL — This procedure sets the value of each of the analog inputs by calling procedure READ_ONE_AI. The small data structure is used by this protection channel.

10b READ_ONE_AI — This procedure reads a single analog input and converts the raw count value to an engineering unit value. Electrical units of volts are then calculated for a sensor type of current transmitter.

11b PROCESS_REAL_VALUE — This procedure accepts a value read from the A/D converter and compensates for any analog input channel's gain and offset errors.

12b ELEC_UNIT_CONV — This procedure converts the calibrated count value to electrical values of volts using a linear gain and offset method.

13b UPDATE_ENGR_UNITS — This procedure converts the electrical value which are in volts to engineering units.

14b EXECUTE_LOOP — This procedure determines the protection channel algorithm to be executed for the protection rack.

15b EXEC_TTD_ALGORITHM — This procedure performs the Trip Time Delay (TTD) protection channel algorithm and stores the results in the common large data structure for use in other LCP procedures.

16b LOW_COMPARATOR — This procedure performs the low comparator trip function for tripping or resetting the bistable.

17b LEADLAG — This procedure performs the lead/lag function using a trapezoidal approximation.

18b RSA_2 — This procedure performs the 2 sensor redundant algorithm. It uses (2) two sensors and performs averaging and consistency checks to produce a single value which most accurately represents the parameter being measured.

19b RSA_3 — This procedure performs the 3 sensor redundant algorithm. It uses (3) three sensors and performs averaging and consistency checks to produce a single value which most accurately represents the parameter being measured.

20b CALCULATE_THIRD_ORDER_POLY — This procedure performs the third polynomial equation.

21b WRITE_DI — This procedure is called from the LCP main loop and writes all the digital outputs on a per channel basis from a protection rack to its associated hardware.

22b WRITE_DO_SMALL — This procedure writes all the digital outputs for a protection channel which use the small data structure. For the partial trip outputs, procedure PT_OUT is called to determine the value output to the 519 board.

23b PT_OUT — This procedure is used for partial trips only and outputs the appropriate pulse state or value.

24b SET_519_DIGITAL_OUTPUTS — This procedure outputs a digital value to a port on the SBC-519 board. This signal is then sent to the signal conditioning EPT board.

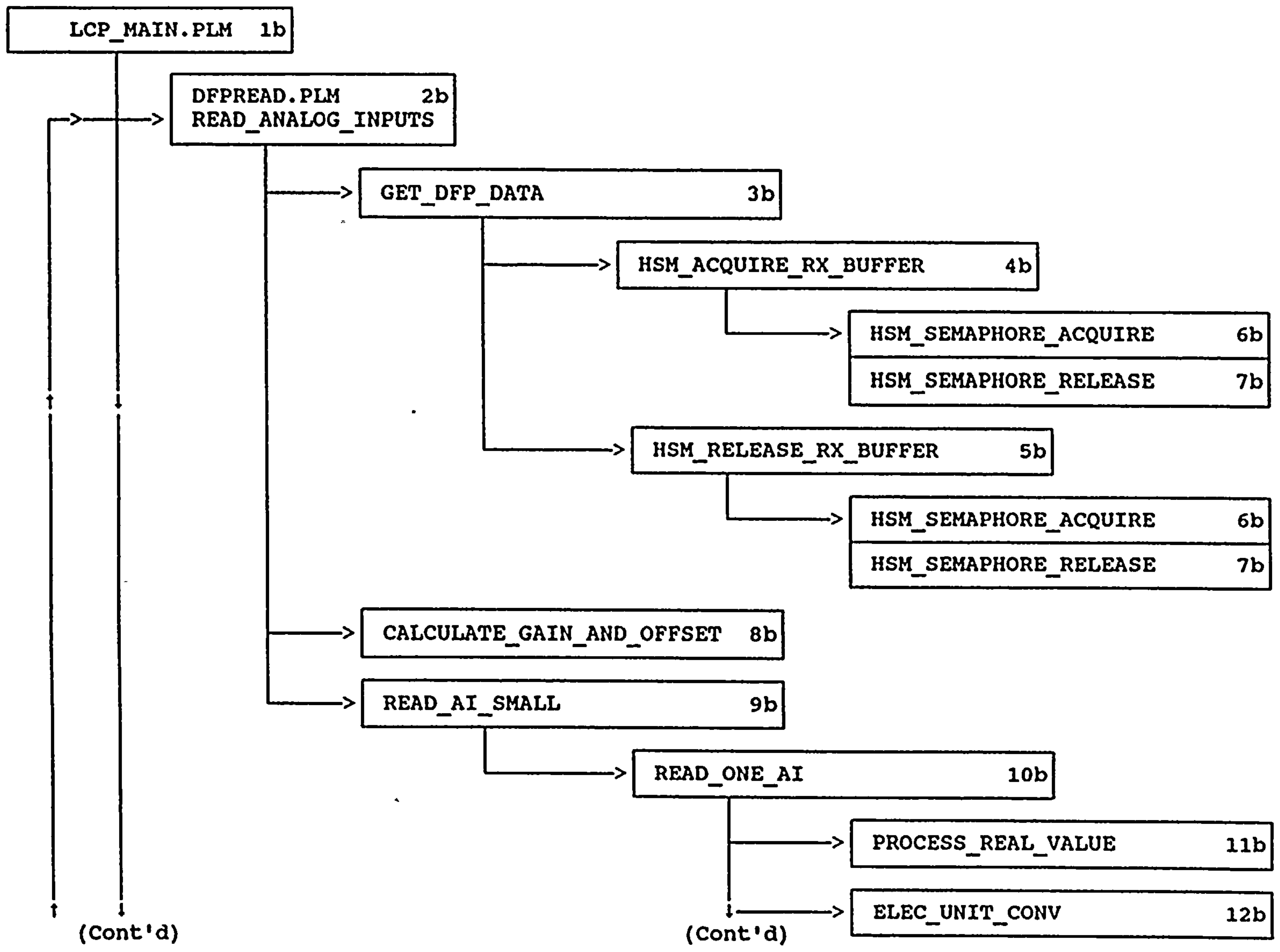


Figure 10. Eagle 21 Loop Calculation Prcessor (LCP)

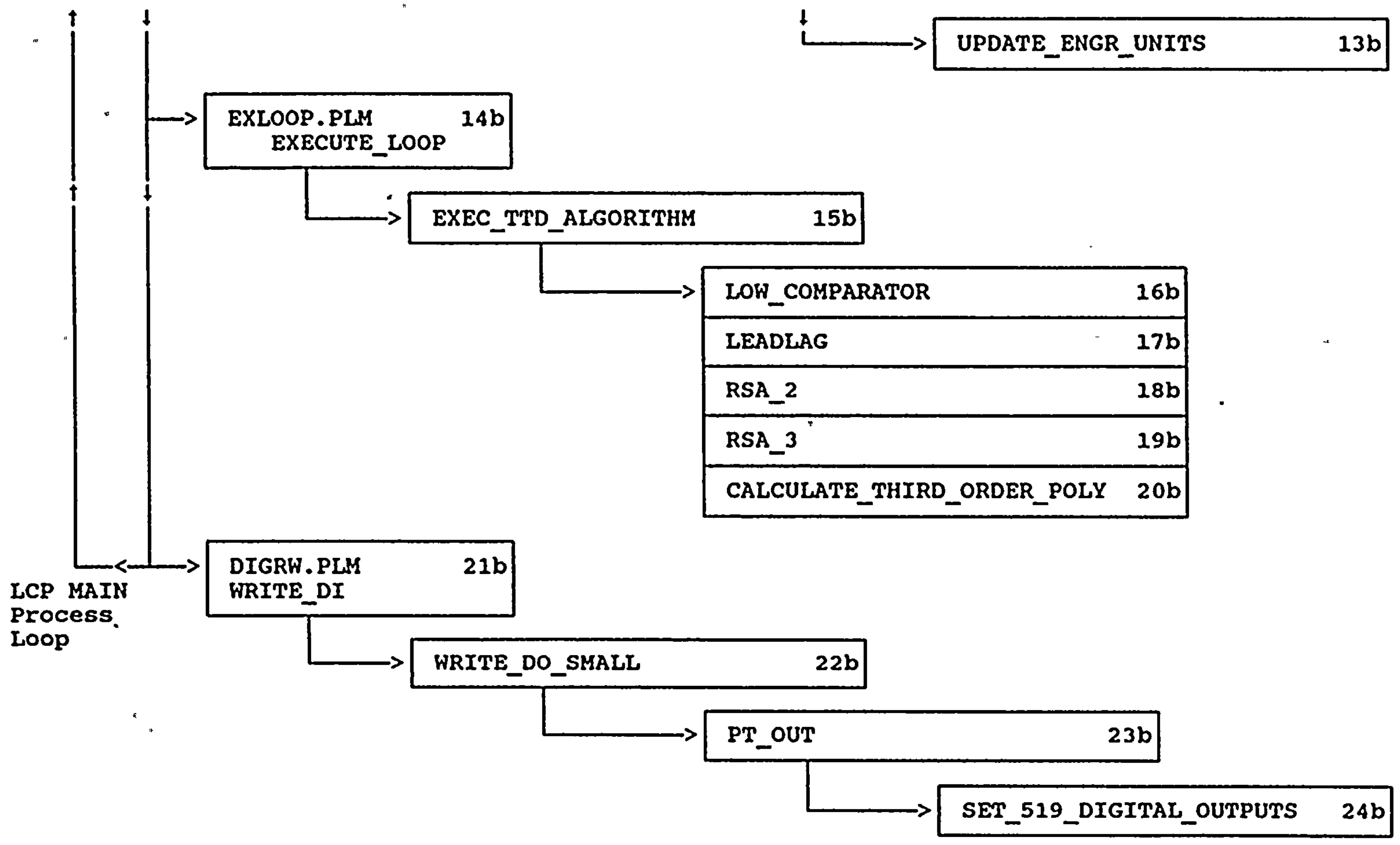


Figure 10. Eagle 21 Loop Calculation Processor (LCP) (Cont'd)

AMSAC Thread Path for Low-Low Steam Generator Level Signals

The following figures, descriptions, and source code will show a thread path for the Low-Low Steam Generator Level signal as it is processed by the AMSAC system software. Figure 11 shows an overview of the input signals as they are processed by the system.

The signal originates at a differential pressure transmitter, that is isolated by the Eagle-21 Process Protection System and enters AMSAC as an analog signal. The signal is conditioned by the Analog Input Board and is converted to a digital signal by [ ]+a,c card located on the Actuation Logic Processor (ALP) board. The ALP performs the necessary setpoint comparisons and logic and then initiates a turbine trip and auxiliary feedwater flow when required through the majority voter and interfacing output relays.

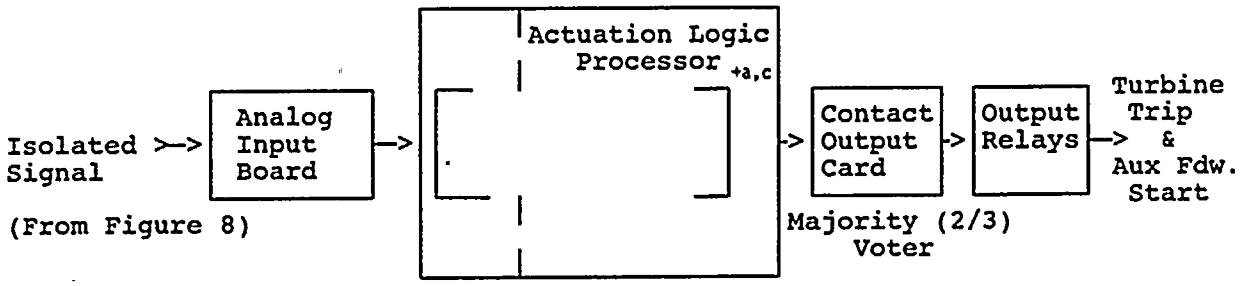


Figure 11. AMSAC Input/Output Overview

The AMSAC Actuation Logic Processors (ALPs) reads raw values and then calibrates each analog input, linearizes the calibrated input values, filters the inputs, calculates engineering values, perform setpoint comparisons and coincidence logic, and sends actuation outputs via the majority voters and output relays. These procedures are outlined below and are illustrated in Figure 12.

1c ALP_MAIN — This procedure implements the main loop of the ALP subsystem, calling appropriate procedures to perform the functions of the AMSAC software system.

2c SMIM_GET_VALUES_FROM_SM — This procedure transfers data from the shared memory area for the TMP and ALP processors and stores the data in a local area.

3c SMIM_XFR_DATA_TO_LOC_AREAS — This procedure transfers data from the TMP processor to the ALP processor via the shared memory area. This includes channel specific and plant specific calibration and configuration data.

4c AIDDM_CLBR_GAIN_OFFSET — This procedure accepts the Gain and Offset values which where part of the data values transferred from the TMP_to_ALP subsystems.

5c CITESM_SET_AI_CHNL_PRMTRS — This procedure checks the validity of all channel specific parameters and stores the data in the appropriate locations.

6c CITESM_CALC_AI_HYSTERESIS — This procedure calculates the hysteresis value for a specific channel. The hysteresis will be used as part of the low comparator function for the low low steam generator inputs.

7c CITESM_CALC_ES_FOR_ALL_PTS — This procedure checks the number of channels in the Analog Input Scan List. It checks each channel number for validity and calls lower level procedures to perform the engineering unit conversions. The engineering effective states are calculated based on whether the point is tripped, hardware status is bad or how the hysteresis and setpoint detention is set.

8c AIDIM_PRCSS_SNGL_CHNL — This procedure controls the processing for A/D and engineering unit conversions.

9c AIDDM_ATOD_CNVRT — This procedure reads field input data from the A to D Converter and computes the digital value. It determines if a hardware failure or power supply problem occurred, and scales and normalizes the digital value based on current gain and offset.

10c AIDIM_CNVRT_TO_ENGG_UNTS — This procedure linearizes the calibrated input value based on the linearization function for the input channel. It then computes the filtered value for the input channel from its linearized value, filter time constant, and prior filtered value, and converts the filtered value to engineering value.

11c CITESM_CALC_ES_FOR_AI_PTS — This procedure determines engineering effective states based on whether the point is tripped, hardware status is bad or how the hysteresis and setpoint detention is set.

12c EVALUATE_ACTIVE_LOW — This procedure sets the point status based on the prior effective state and the engineering units value as compared to the setpoint and hysteresis values.

13c PLANT_EXECUTE_ALGORITHM — This procedure assembles the input data and calls the appropriate plant algorithm.

14c CITESM_GET_ES_FOR_AI_PTS — This procedure returns the effective state for the appropriate channel.

15c ALG_EXECUTE — This procedure controls the software implementation of the ALP protection algorithm for the initiation of turbine trip on low-low steam generator water level (level setpoint both dependent and independent of turbine load) when the turbine load is above a pre-specified percent of nominal load.

16c ALG_LOW_LOW_4 — This procedure controls the initiation of turbine trip on low-low steam generator water level when the turbine load is above a pre-specified percent of nominal load.

17c VOTER_THREE_OF_FOUR — This procedure generates a 3/4 voter algorithm, returning the status.

18c ALG_TIMER_WAIT — This procedure checks and sets timer status, where a 3/4 vote for the low low steam generator is detected.

19c TIMER_STOP_SOFT_TIMER — This procedure disables counting in the timer; sets timer status to stopped.

20c TIMER_READ_SOFT_STATUS — This procedure returns the status of the specified timer; stopped, counting or timed out (halted).

21c TIMER_START_SOFT_TIMER — This procedure resets the specified timer and set its status to counting.

22c ALG_TIMER_HOLD 34 — This procedure checks and sets timer status, when the turbine load is above the specified percentage of nominal load.

23c PLANT_DISTRIBUTE_OUTPUTS — This procedure calls the appropriate output procedure to route the calculated output values to the proper output channel, based on hardware wiring information.

24c DO_UPDATE_OUTPUTS — This procedure calls DO_PORT_UPDATE to output the updated values.

25c DO_PORT_UPDATE — This procedure assembles the requested states of the digital output channels into bytes corresponding to the output ports, applies the data active and data polarity bit masks and writes the values to the output ports.

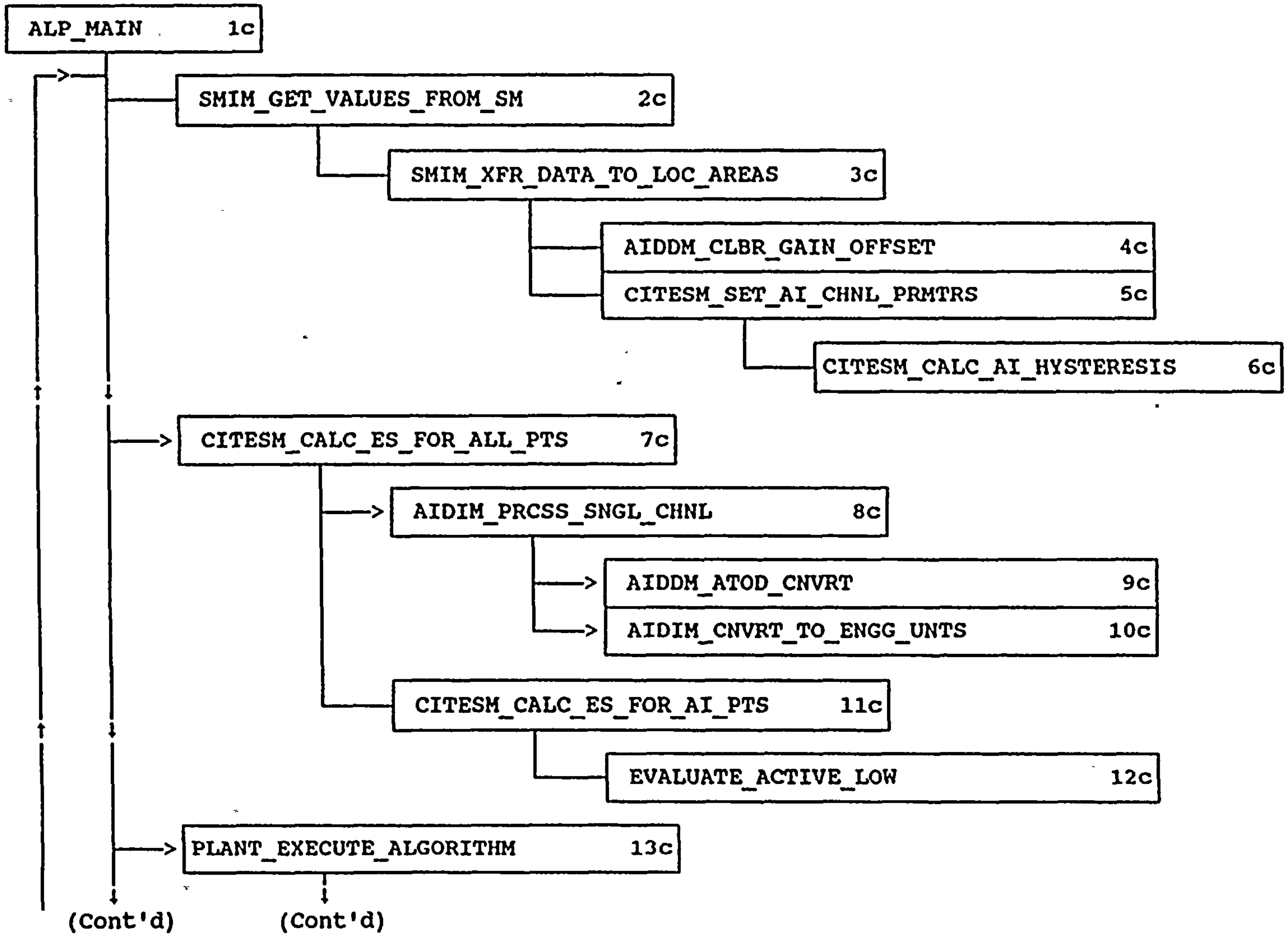

ALP_MAIN 1c
SMIM_GET_VALUES_FROM_SM 2c
SMIM_XFR_DATA_TO_LOC_AREAS 3c
AIDDM_CLBR_GAIN_OFFSET 4c
CITESM_SET_AI_CHNL_PRMTRS 5c
CITESM_CALC_AI_HYSTERESIS 6c
CITESM_CALC_ES_FOR_ALL_PTS 7c
AIDIM_PRCSS_SNGL_CHNL 8c
AIDDM_ATOD_CNVRT 9c
AIDIM_CNVRT_TO_ENGG_UNTS 10c
CITESM_CALC_ES_FOR_AI_PTS 11c
EVALUATE_ACTIVE_LOW 12c
PLANT_EXECUTE_ALGORITHM 13c
(Cont'd)
(Cont'd)

Figure 12. AMSAC Process Loop

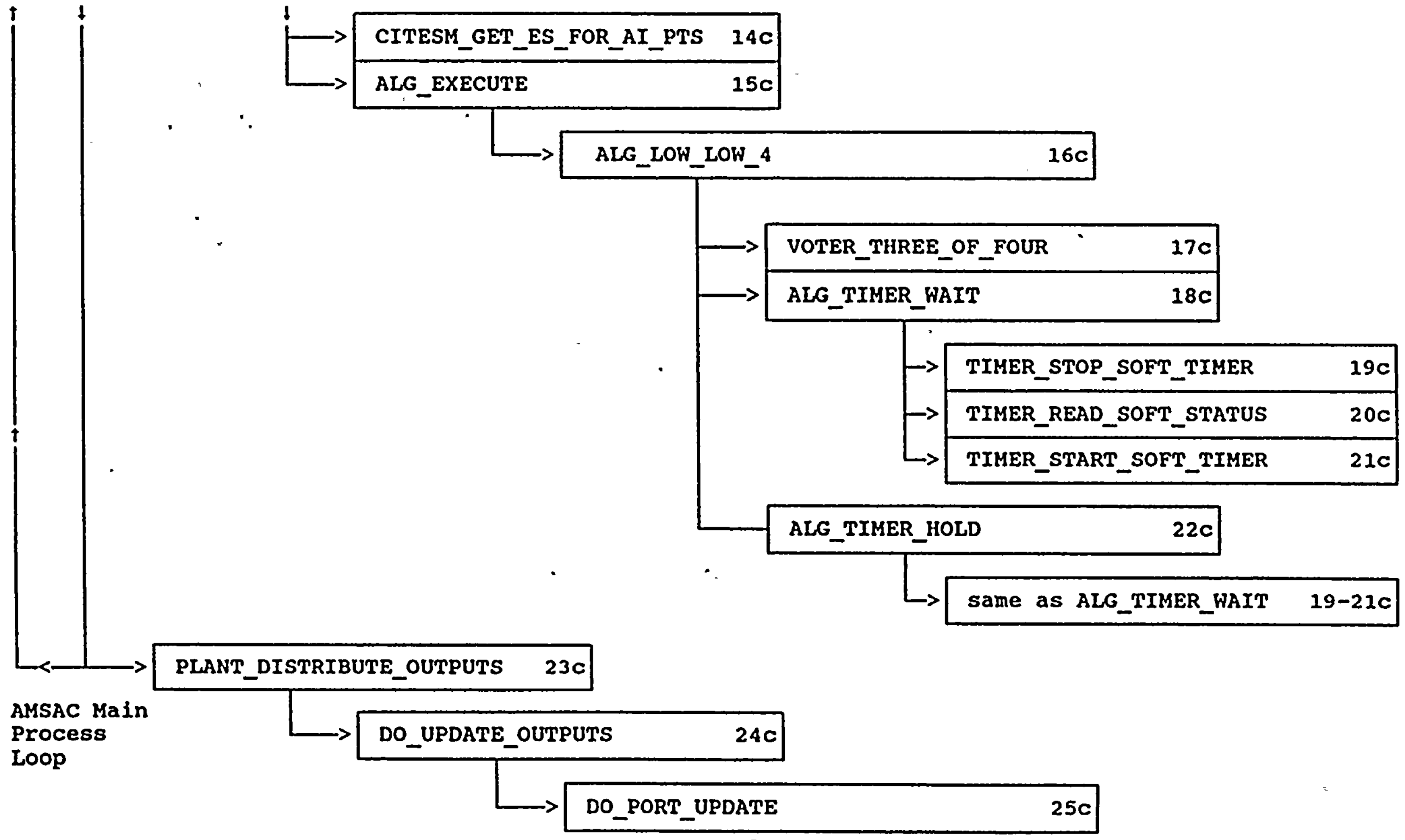

CITESM_GET_ES_FOR_AI_PTS 14c
ALG_EXECUTE 15c
ALG_LOW_LOW_4 16c
VOTER_THREE_OF_FOUR 17c
ALG_TIMER_WAIT 18c
TIMER_STOP_SOFT_TIMER 19c
TIMER_READ_SOFT_STATUS 20c
TIMER_START_SOFT_TIMER 21c
ALG_TIMER_HOLD 22c
same as ALG_TIMER_WAIT 19-21c
PLANT_DISTRIBUTE_OUTPUTS 23c
AMSAC Main Process Loop
DO_UPDATE_OUTPUTS 24c
DO_PORT_UPDATE 25c

Figure 12. AMSAC Process Loop (Cont'd)

To aid in reviewing the procedures and software of AMSAC and Eagle 21, the procedures and functions are presented side-by-side in Tables 4 and 5. From this side-by-side comparison, it can clearly be seen that the software design and implementation is completely different.

Software Comparison EAGLE-21 Versus AMSAC

+a,c

Table 4. Eagle-21/AMSAC Procedural Comparison

+a,c

Table 5. Eagle-21/AMSAC Functional Comparison

The source code for each of the major functions is presented side-by-side to illustrate the extreme diversity which exists between Eagle 21 and AMSAC in this area.

Read Analog Inputs "Source Code" Comparison Eagle-21/AMSAC

+a,c

Analog Input to Engineering Unit Conversion
Comparison of the Functional Algorithms for Eagle-21/AMSAC

+a,c

Analog Input to Engineering Unit Conversion
"Source Code" Comparison for Eagle-21/AMSAC

+a,c

Analog Input to Engineering Unit Conversion
"Source Code" Comparison for Eagle-21/AMSAC (Cont'd)

+a,c

Analog Input to Engineering Unit Conversion
"Source Code" Comparison for Eagle-21/AMSAC (Cont'd)

+a,c

Functional Algorithm Comparison for Eagle-21/AMSAC

+a,c

Functional Algorithm "Source Code"

Comparison of like functions for Eagle-21/AMSAC

+a,c

Functional Algorithm "Source Code"
Comparison of like functions for Eagle-21/AMSAC (Cont'd)

+a,c

Summary of the software analysis between Eagle-21/AMSAC

+a,c

4.2.3 Conclusion

Westinghouse has performed a side-by-side analysis of the software implementation for the software required to process the steam generator water level input for both the Eagle-21 and AMSAC systems. The following conclusions can be made:

- There is no common software source code between the Eagle-21 and AMSAC systems.
- Like functions have been implemented independently of each other. This has clearly resulted in significant diversity between the systems (i.e., code structure, digital filtering, and order of functions) such that the requirements of 10CFR 50.62 and the ATWS Final Rule are met.
- The implementation of both functional algorithms, Trip Time Delay (TTD) for Eagle 21 and Low Low Steam Generator Level for AMSAC are also diverse. AMSAC performs a 3/4 vote on the low low steam generator level to cause an actuation, while EAGLE 21 causes a partial trip on a low low steam generator level and the SSPS performs the 2/4 voting.

4.3 Evaluation of Use of Same Programming Language, Compiler, Linker and Locator in Eagle 21 and AMSAC

4.3.1 Introduction

The Eagle 21 and AMSAC microprocessors each execute program instructions from compiled code stored in EPROM. The compile, link and locate functions necessary to perform these program functions are all performed offline. A brief discussion of each function follows.

The compiler translates source code into object code. Both the Eagle 21 and AMSAC systems are primarily coded in [ ]+a,c with a small number of assembly language routines coded in [ ]+a,c.

The link function combines a set of object code modules into a single object code module and attempts to match all external symbol declarations with their public symbol definitions in library modules. The output of the link function is a relocatable object code module. The link function can link any set of object code modules which may have originated from various languages (eg. Fortran, PLM, Assembly) into a single object code module.

The locate function converts relocatable object modules to absolute object modules which are executable. Absolute object modules contain references that allow object module segments to be placed at particular locations in processor memory.

4.3.2 Evaluation

Both the Eagle 21 and AMSAC systems are programmed in [ ]+a,c is a language with a syntax similar to [ ]+a,c which has been optimized for use with the [ ]+a,c family of microprocessors.

The goal of the programming language is to provide the programmer with a human type language to program machine instructions. High level programming languages such as [ ]$^{+a,c}$ allow equations to be expressed in mathematical form and conditionals to be formatted in "IF THEN ELSE" or "CASE" type statements which are logical, easy to understand, and easily lend themselves to verification type activities such as mathematical proof and code inspection. These high level languages relieve the programmer from the constraints of programming in assembly language which is extremely cryptic, harder to understand and does not lend itself as easily to verification type activities.

Westinghouse has had extensive experience using the [ ]$^{+a,c}$ Compiler and the [ ]$^{+a,c}$ Link and Locate tools. [ ]$^{+a,c}$ and these tools have been used in the RVLIS/ICCM, PSMS/QDPS products as well as the Eagle 21 and AMSAC systems. This experience spans more than 10 years of operation in diverse functions. There has never been a language defect, a compiler error or a link/locate error which has manifested itself in the field. Westinghouse maintains version/revision control of [ ]$^{+a,c}$ and the compile, link and locate tools used in the Eagle 21 and AMSAC systems. New versions will not be used until changes have been evaluated and tested.

Because of this experience Westinghouse viewed use of this well-proven language and compiler as a reliability enhancement. Introducing a new language/compiler has the potential of introducing unknown errors which may have a tendency to degrade overall reliability. Since [ ]$^{+a,c}$ compiler have reliably translated the programmers instructions into machine code for over 10 years with no failures it was deemed desirable for use in both the AMSAC and Eagle 21 systems.

Although high level programming languages ease the programmers job, the language itself does no more than provide a way of expressing equations, algorithms and procedures as defined by the programmer. All decisions such as comparing process variables to setpoints and trip/no trip decisions are programmed in by the designer. The processor itself does not execute any particular language. As was stated above, high level language code is converted to executable object code by the compile, link and locate functions. The processor executes this object code. Thus two systems which are programmed in the same high level language, but are programmed to perform different functions, will execute different object code and therefore be diverse. A detailed software thread path of the Steam Generator Level function is provided as part of this analysis. This thread path compares Eagle 21 source code for Steam Generator Level processing to that of AMSAC. The thread path analysis proves that the source code for the two systems are diverse. Diverse source code will produce diverse object code. Simply programming the same functions in a different language using the exact same programming sequences and methods may produce object code which is not very diverse.

The compiler, linker and locator's jobs are to accurately translate source code to executable object code on the target hardware. The compiler is the most complex of the three functions since it translates the high level coding language into object code. Even though compilers can be relatively complex their job is to create object code from source code. Like the

programming language, the compiler, linker and locator do not have system level intelligence. Diverse source code run through the same compile, link and locate tools will produce diverse object code.

In addition, verification unit testing is performed on compiled and linked code. Verification unit testing executes all lines of code in the module under test. Validation testing is performed with compiled, linked and located code on the target hardware. Validation testing is designed to execute as many lines of code as feasible. Validation testing executes all critical process code such as algorithm calculations and process variable comparisons to setpoints. The AMSAC and Eagle 21 loop processor software is designed to the following specifications:

- ■Use of Interrupts is Prohibited
- ■Use of Re-entrant Code is Prohibited
- ■Processors execute in continuous loop
- ■Boundary Checks of Parameters
- ■All machine states are defined
- ■Coding standards prohibit non-deterministic coding practices

Software which is not designed in this deterministic manner (use of interrupts, re-entrant code) may be impossible to bound since the execution combination possibilities for non-deterministic code (one that relies on interrupts) is infinite.

To have safety significance, a language/compiler error would somehow have to cause the processor to skip or corrupt process variable comparisons or manipulations . Because of the deterministic software design practices used for the Eagle 21 and AMSAC code language/ compiler, these errors would be detected in either verification or validation testing.

As was discussed earlier AMSAC and Eagle 21 are diverse at the system level. Thus, even if a common mode language, compiler, link/locate error did exist the error would have to cause the Eagle 21 to continue to pulse the Trip Output Module to maintain a "NOT-TRIPPED" output to the voting logic system while the same error prevents the AMSAC from outputing a "TRUE" to the Voter to compromise diversity. Languages merely implement the programmers instructions while compilers translate source code to object code. The postulated error would have to be equivalent to a source code statement of "IF PROCESS EXCEEDS SETPOINT THEN DON'T TRIP". In Eagle this error would have to prompt the processor to execute two I/O commands (to a specific address) via the multibus to pulse the Trip Output Module. In AMSAC this same error would have to prevent the processor from executing an I/O command via the ALP local bus to output a "TRUE" to the voter.

As was discussed in the paragraph above, this scenario could not be caused by a common mode failure of the language, compiler or link/locate tools on deterministic systems such as the Eagle 21 and AMSAC that have gone through the extensive verification and validation testing. In the unlikely event that these subtle, low level errors truly exist, at most they may cause inadvertent

actuation or failure to actuate in a simple True/False system such as AMSAC but these errors would not be sophisticated enough to at the same time execute two active commands to prevent the Eagle 21 from tripping.

4.3.3 Conclusion

Source code of the same language [ ]+a,c but programmed in a diverse manner will produce diverse object code. The thread path analysis of Steam Generator Level functions in Eagle 21 and AMSAC proves that the source code for the two systems is diverse and will produce diverse object code. Since the AMSAC and Eagle 21 processors run on object code, not source code, use of the same programming language and the same compile, link and locate tools will still produce diverse object code and thus, not compromise diversity.

Extensive use and reliability of the [ ]+a,c language, compiler and link/locate tools have proven that subtle errors in these items that could cause inadvertent actuation or failure to actuate system level responses do not exist.

Because of the deterministic design of the Eagle 21 and AMSAC software, language/compiler errors which would somehow prevent the Eagle 21 from tripping and prevent the AMSAC from actuating would have been caught during the verification and validation testing of either system.

In the unlikely event of a common mode language, compiler type error, the system level diversity of the AMSAC/Eagle 21 systems will prevent the same error from causing a failure of Eagle 21 to trip the reactor and initiate safeguards and a failure of AMSAC to trip the turbine and initiate auxiliary feedwater.

5.0 CONCLUSIONS

The AMSAC and Eagle 21 systems have been reviewed at all key levels for diversity to meet the requirements of 10CFR 50.62 and the ATWS Final Rule.

Organizational diversity has been shown by demonstrating that completely separate design teams were deliberately used to develop the AMSAC and Eagle 21 systems.

Functional diversity has been shown by demonstrating that the AMSAC algorithm to initiate Auxiliary Feedwater and trip the turbine is diverse from that employed by Eagle 21 to initiate reactor trip and engineered safeguards actuations.

System diversity has been shown by demonstrating that the AMSAC and Eagle 21 system architectures and failure modes are diverse. This prevents the small number of industry standard common components from preventing Eagle 21 to trip with the same failure preventing AMSAC from actuating.

Hardware diversity has been demonstrated between AMSAC and Eagle 21 since there are very few components common to both systems. Microprocessor diversity [ ]$^{+a,c}$has also been demonstrated. Those components that are common are highly reliable industry standard components produced by multiple manufacturers. Hardware diversity down to low level industry standard component level (op-amps, logic gates etc.) is not reasonable and practicable. There is a high probability that these industry standard components would be used in any I&C process equipment, including other vendor's designs. It has also been demonstrated that failures of common components is either detectable in both systems via system diagnostics and alarms, operator shift checks, or will result in a fail-safe (tripped) output from the Eagle 21 system.

Software diversity has been demonstrated between AMSAC and Eagle 21 since the source code for a common function (Steam Generator Water Level) is diverse via thread path analysis. Use of a common programming language [ ]$^{+a,c}$ and compiler for the two systems has been evaluated by proving that diverse source code will produce diverse machine code even if the source code is of the same high level language. In addition, Westinghouse has over 10 years of proven experience with [ ]$^{+a,c}$ and its compiler, in many diverse applications, both safety and non-safety systems.

These reviews conclusively prove that the Diablo Canyon AMSAC and Eagle 21 systems are diverse to the extent reasonable and practicable, and satisfy the requirements for diversity as stated in 10CFR 50.62 and the ATWS Final Rule.