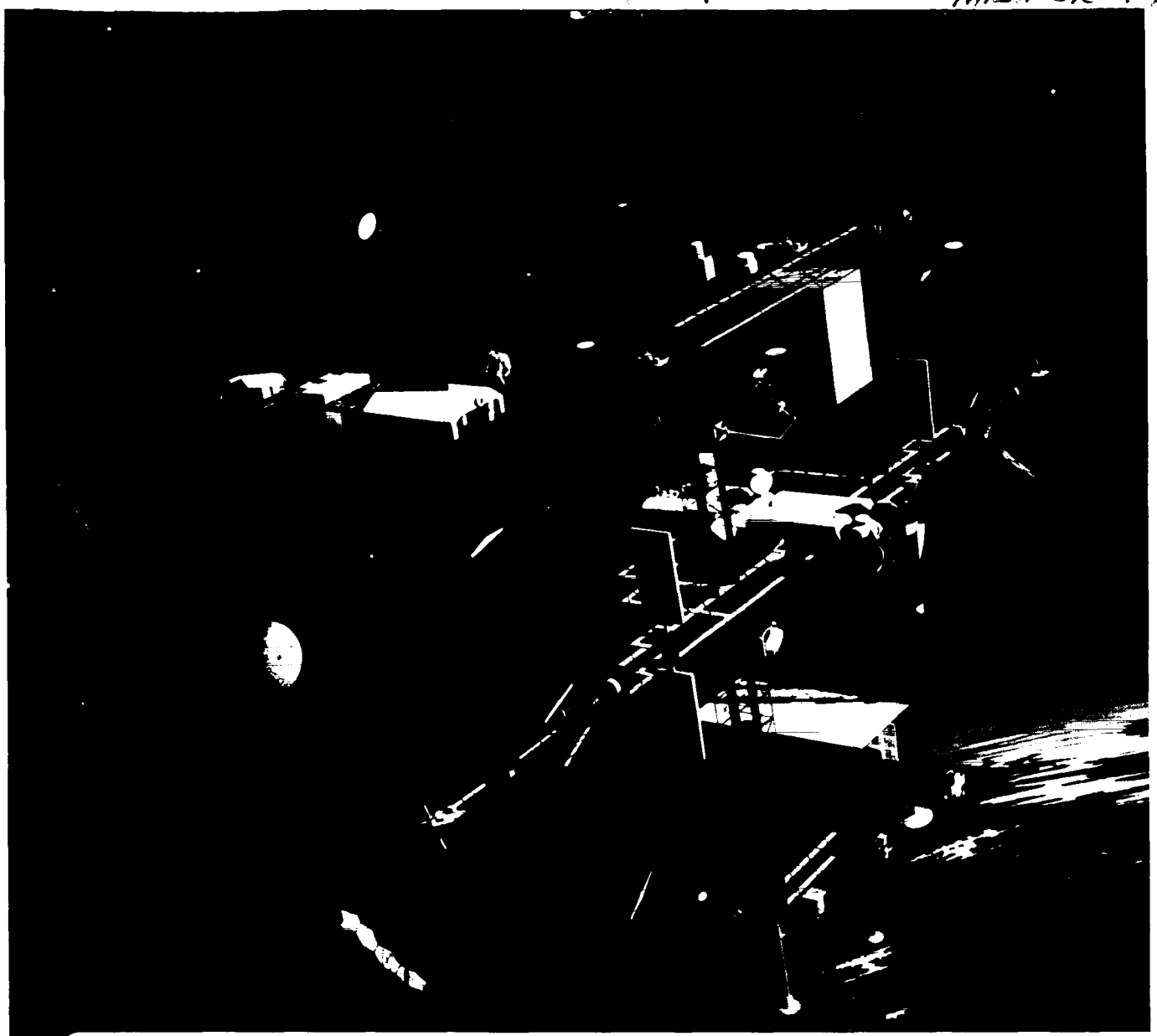
NASA CR-172003



(NASA-CR-172003) AUTOMATED SUBSYSTEM CONTROL FOR LIFE SUPPORT SYSTEM (ASCLSS) Final Report (Honeywell) 65 p Avail: NTIS EC A04/MF A01 CSCL 06K

N87-29117

Unclas G3/54 0098647

Automated Subsystem Control for Life Support System (ASCLSS)

FINAL REPORT

NASA CONTRACT NASA-9-16895

Honeywell

NAS-9-16895

12590
15 July 1987

AUTOMATED SUBSYSTEM CONTROL FOR LIFE SUPPORT SYSTEM (ASCLSS)

FINAL REPORT

Written By:

Roger F Block

Dr. Roger F. Block
Honeywell SSAvD Division

Honeywell

SPACE & STRATEGIC
AVIONICS DIVISION
CLEARWATER FLORIDA

ABSTRACT

This final technical report describes the objectives, plans and accomplishments of the NASA Johnson Space Center sponsored program entitled "Automated Subsystems Control for Life Support System (ASCLSS), NAS9-16895." The program objectives were to define a generic automation approach for Space Station subsystems and to demonstrate the selected automation technique by controlling and monitoring the Air Revitalization Group (ARG) of a regenerative Environmental Control and Life Support System (ECLSS). The ARG consists of three ECLSS processes: O_2 generation, CO_2 concentration, and CO_2 reduction.

The ASCLSS automation approach developed by Honeywell and Life Systems Inc. consisted of a hierarchy of distributed controllers implemented with 1750A microprocessors and a high speed busing network. System level, local process control, and real-time operating system software were developed and integrated with the controller hardware to demonstrate the automated control and monitoring of the three Space Station ECLSS processes. The ARG processes were simulated by three ARG simulators, implemented in individual personal computers. A crew man-machine interface (MMI) was included in the automation system to develop and demonstrate the control authority allocated between the crew, the upper level system controller, and the lower level ARG process controllers.

The completed ASCLSS system has successfully demonstrated many important features that are currently being considered for Space Station automation and control. These important features emphasized commonality of hardware and software, implementation of standards, incorporation of a high level of system autonomy, and the placement of the crew operator in a role of supervisory control.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ACRONYMS

A&C	Automation and Control
AI/ES	Artificial Intelligence/Expert Systems
ARG	Air Revitalization Group
ASCLSS	Automated Subsystem Control for Life Support System
CPU	Central Processing Unit
DMS	Data Management System
DOS	Distributed Operating System
ECLSS	Environmental Control and Life Support System
EDC	Electrochemical CArbon Dioxide Depolarizer
EDP	Embedded Data Processor
EPS	Electric Power System
E2PROM	Electrically Erasable Program Read Only Memory
GN&C	Guidance, Navigation and Control
HOL	High Order Language
ICD	Interface Control Document
I/O	Input/Output
IOC	Initial Operational Capability
ISA	Instruction Set Architecture
JSC	Johnson Space Center
Kips	One Thousand Instructions per Second
LSI	Life Systems, Inc.
MDM	Multiplexer-Demultiplexer
MMI	Man-Machine Interface
MPAC	Multipurpose Applications Console
NIU	Network Interface Unit
OMS	Operations Management System
PC	Personal Computer
RAM	Random Access Memory
RM	Redundancy Management
RTOS	Real-Time Operating System
S-CRS	Sabatier CO2 Reduction Subsystem
SDP	Standard Data Processor
SFE	Static Feed Electrolyzer
SLOC	Single Line of Code
TAVERNS	Test and Verification Environment for Remote Networked Systems
TMIS	Technical, Management and Information System
V&V	Verification and Validation

Section 1

INTRODUCTION

Early in 1983, NASA initiated a technology development program entitled, "Automated Subsystem Control for Life Support System" (ASCLSS, NAS 9-16895). The national commitment for a manned Space Station was evolving and NAS recognized that a permanent presence of man-in-space would require highly automated systems. The ASCLSS program was funded by the NASA Office of Aeronautics and Space Technology and the Crew and Thermal Systems Division of NASA Johnson Space Center.

The ASCLSS program primary objectives were to define and develop a generic approach to automation and control of Space Station systems and to demonstrate the selected automation approach by controlling and monitoring the Air Revitalization Group (ARG) of a representative Space Station regenerative Environmental Control and Life Support System (ECLSS). The autonomous operation of the ECLSS Air Revitalization Group is considered one of the most important requirements for an ultimately successful Space Station operation. An additional ASCLSS program objective was to demonstrate the importance of subsystem simulators for the development, demonstration and verification of the generic automation and controls approach.

The contract, which was initiated in May 1983, was managed by Honeywell Space and Strategic Avionics Division (SSAvD Clearwater, FL), with participation and important contributions by Life Systems, Inc. (Cleveland, OH), and Honeywell Systems and Research Center (Minneapolis, MN). The ASCLSS program team roles and technical responsibilities are summarized in Table 1-1. The program represented a blend of Honeywell's extensive automation and controls experience in commercial, industrial, and manned space applications with the environmental control and life support systems technology and experience of Life Systems, Inc.

The overall program task plan and schedule shown in Figure 1-1 followed a logical sequence of development steps starting with an extensive applications study during the first year of the program. The applications study evaluated and assessed the automation and control (A&C) requirements for the Space Station DMS, GN&C, ECLSS, and electric power (EPS) subsystems. Based on these requirements, a generic approach to Space Station A&C was defined. The approach featured a hierarchical and distributed controller architecture with a man-machine interface for crew supervisory control and individual simulators for each ARG process. Following design and fabrication of the automation system hardware and software, the major elements were individually tested prior to system integration. Final integration of the ASCLSS demonstration system took place at Honeywell SSAvD with delivery and system acceptance test at NASA JSC on 28 July 1986. Subsequently, a small add-on development program was conducted to develop enhanced demonstration features of the system. Final delivery of the ASCLSS system to NASA JSC occurred on 7 April 1987.

The remainder of this report summarizes the major objectives and significant results of the ASCLSS program tasks. Section 7 identifies the important "lessons learned" for the development program and Section 8 provides recommendations for additional areas of investigation and study. More detailed program information and technical data may be obtained from the documentation and references listed in Appendix A.

Table 1-1. NASA JSC ASCLSS Program Responsibilities

Honeywell SSAvD

- Program Lead and Management
- Automation Systems Engineering
- Automation and Control Hardware Development
- Generic, Distributed, Layered Operational Software Development
- System Integration, Test and Demonstration

Life Systems Inc.

- ECLSS System Engineering
- ARG System and Process Application Software Development
- ARG Simulators/Simulations Hardware and Software Development
- ARG Application/Simulation Software Test and Integration Support

Honeywell SRC

- Applications Study Lead
- Human Factors/Crew Operator Interface Development
- Automation System Man-Machine Interface (MMI) Hardware/Software Development
- MMI System Test and Integration Support

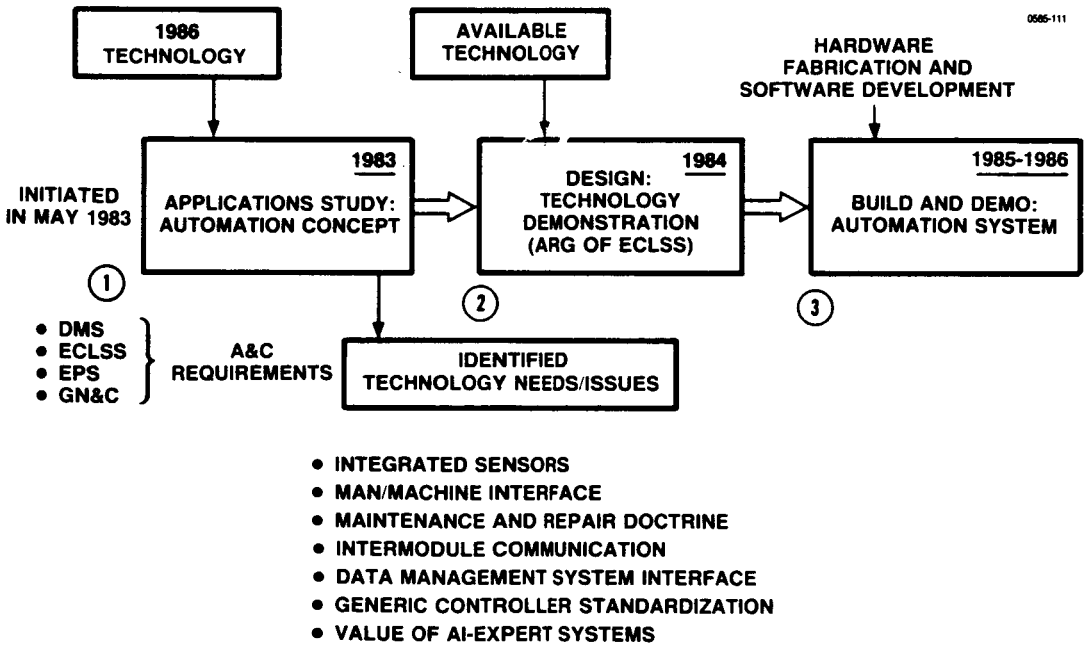


Figure 1-1. ASCLSS Task Plan and Schedule

Many organizations and individuals have contributed to the ASCLSS technology development and this acknowledgment identifies their significant commitments and achievements:

- NASA Johnson Space Center (Crew and Thermal Systems Division)

  Frank H. Samonski, Jr.
  Nick Lance, Jr.
  Dr. Hatice Cullingford

- Honeywell Space and Strategic Avionics Division (Clearwater, Florida)

  Dr. Roger F. Block
  Mel Lamb
  Hoyt Layson
  Mohammad Khavarian
  Russell Cooner
  Warren Jokinen
  Tom Berendzen
  Rozanna Miller
  Mark Baron
  Jack Churchward

- Life Systems, Inc. (Cleveland, Ohio)

  Dr. Richard A. Wynveen
  Dr. Dennis B. Heppner
  James D. Powell, III
  Edward S. Mallinak, Jr.
  Lawrence D. Noble, Jr.
  Steven Hendrix
  Jeffrey H. Birkel

- Honeywell Systems and Research Center (Minneapolis, Minnesota)

  Herb Lindquist
  Paul Rebelein
  Kenneth Hoyme
  Dr. Pamela Jamar
  Mike Schroeder
  Daniel Sudlin
  Amy McFarlin

Section 2

APPLICATIONS STUDY (Task 3.1.1)

2.1 TASK OBJECTIVES

The primary objective of the ASCLSS application study was to evaluate the feasibility and advantages/disadvantages of implementing a generic "automation technique" for several Space Station systems. The Electric Power System (EPS), Environment Control and Life Support System (ECLSS), and the Guidance, Navigation, and Control (GN&C) system were to be evaluated. Divisions of automation and control (A&C) responsibilities between the operator/crew, the system level control, and subsystem level of control were to be defined and partitioned. These A&C responsibilities were to consider automated command and control, subsystem and system performance monitoring, fault detection and isolation, fault tolerance, performance trend analysis, incipient failure detection, redundancy management, and onboard maintenance operations.

After defining the automation requirements including controller architecture, processing throughput, memory, power, and word sizes across the Space Station systems, a generic automation approach was to be defined to meet the system needs and achieve, as well, the significant benefits of commonality of hardware and software.

Also, a conceptual design for the Space Station ECLSS automation and control was to be defined based on the selected, generic automation technique. And, finally, a technology assessment was to be conducted to evaluate the availability and/or development required to implement the generic automation and control approach on the Space Station at initial operational capability (IOC).

2.2 TASK APPROACH

Space Station A&C requirements were synthesized from NASA sponsored Phase A Space Station studies and on-going system studies at Honeywell, Life Systems, Inc., and Rockwell (a subcontractor for the applications study). These requirements established an overall generic controls architecture and the functional partitioning for A&C responsibility for the ground support, the on-orbit crew, the Data Management System (DMS) man-machine interface, and the hierarchy of system automation and control elements.

The generic A&C approach was evaluated against three major Space Station systems (ECLSS, EPS, and GN&C) which were considered representative of the full spectrum of applications across the Station. This evaluation tested the viability of the generic A&C approach having broad applicability. And, finally, an assessment was conducted to evaluate the available technology required to meet the generic A&C requirements and to identify any required new developments.

2.3 APPLICATION STUDY RESULTS

2.3.1 Requirements

After an extensive review of NASA reports, Space Station Phase A studies, and on-going system studies, the following general requirements were selected to drive the development of the generic automation and control approach for Space Station:

- Modular Space Station architecture with accommodation for incremental buildup.
- Technology readiness for the automation approach based on 1986 design start for core modules and 1992 design start for added (evolutionary) modules.
- Maximum practical autonomy from the ground (reduce costs of ground facilities and personnel).
- Maximum automation of routine subsystem operations to minimize crew workload and free them up to conduct system user activities.
- Strong emphasis on commonality of hardware and software.
- Twenty years on-orbit operational life achieved through on-orbit maintenance.
- Flexible for growth in performance and technology insertion.

2.3.2 Automation and Control Architecture

These requirements led to the definition of a basic automation and control architecture for Space Station systems shown in Figure 2-1. The architecture features a module controller which interfaces to the Space Station Data Management System (DMS) high speed communications network via a Network Interface Unit (NIU). A module level man-machine interface (MMI) or work station also interfaces to the module controller. All critical Space Station system controllers (ECLSS, GN&C, EPS, etc.) that reside in that specific module interface to the module controller via an intra-module bus.

Any given system controller like the GN&C or ECLSS, may have several basic controllers at the lowest level of the control hierarchy. The basic or process level controllers perform the critical control functions for a given process (e.g., the CO_2 concentrator for the air revitalization group (ARG) of the ECLSS or the inertial measurement unit for the GN&C). This automation architecture accommodates a high level of autonomy and allows for a descending hierarchy of control authority functions which are partitioned as shown in Figure 2-1. This module-based, three-level hierarchy of distributed controllers provides for generic commonality, efficient communications, localized processing and control, and accommodation for fail-safe, redundant operation. Under this architecture, the Space Station DMS may be viewed as a system level *manager* of the complete Station-level data base and a global communications network between modules and the major Space Station systems and space operations segments.

2.3.3 Generic Controller Concept

Having defined a basic automation and control architecture, the following additional requirements were defined for the generic controllers making up the architecture:

- Emphasize commonality of hardware and software.
- Minimize cost of design, development, test, integration, maintenance, and ownership.
- Provide for growth and technology insertion.
- Emphasize implementation of standards.

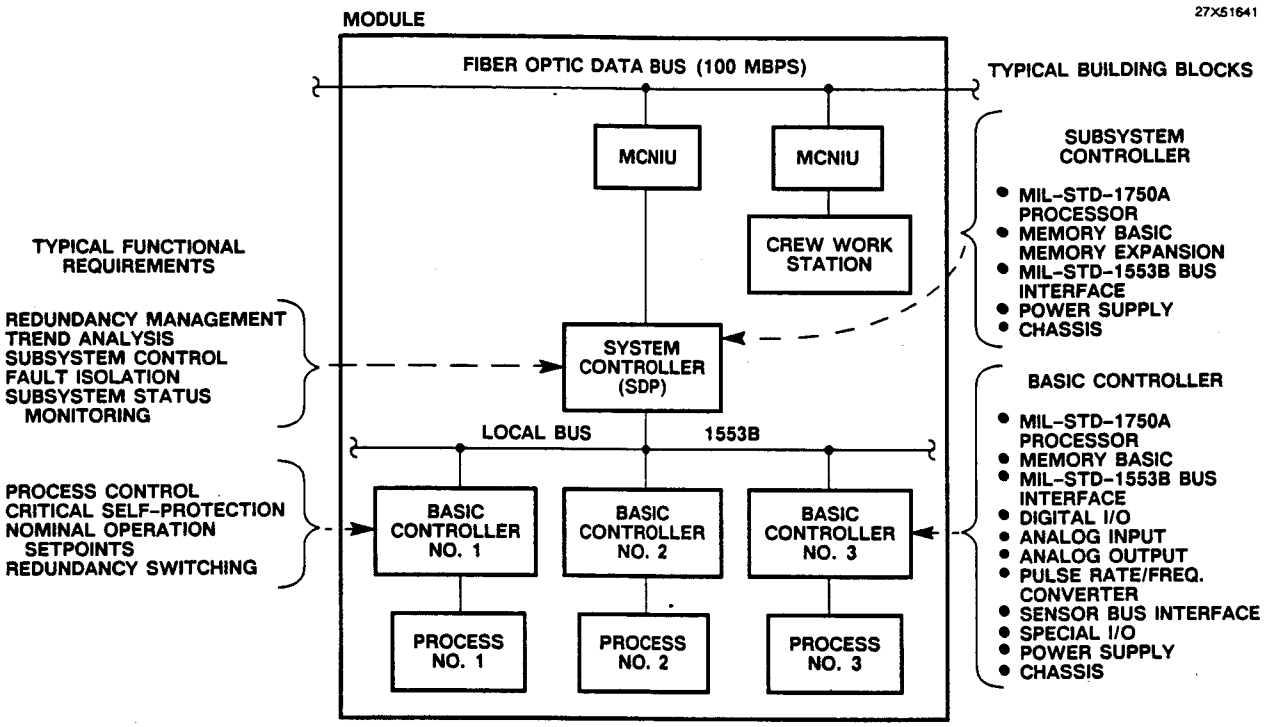


Figure 2-1. Requirements and Building Blocks for a Generic Controller Applied Anywhere in the A&C Architecture

A single controller could not fulfill all these requirements at every level of the architecture without penalties in size, weight, power, and cost. However, a family of controllers assembled from standard card level elements could come very close to meeting the requirements with minimum penalties.

The Honeywell generic controller concept is shown in Figure 2-2. The three basic controller functions consisting of a computer function (CPU/memory), a communications function (serial data/command buses), and input/output functions (analog/digital signals and commands) can all be implemented in card level ORUs and several standard chasses capable of meeting the controller requirements at each level of the automation hierarchy shown in Figure 2-1. This selected generic A&C approach is similar to the current Space Station concepts featuring common DMS elements (SDP, EDP, MDM, NIU) populating a hierarchal and distributed controls architecture in all systems.

The baseline A&C architecture and generic controller concept was then applied to the specific requirements of the ECLSS, GN&C, and EPS systems in order to verify the approach. Out of this detailed analysis and study emerged the family of controllers and the data bus recommendations shown in Tables 2-1 and 2-2, respectively. The card level building blocks for the controllers would have the performance characteristics listed in Table 2-3.

Two additional important results of the ASCLSS applications study were the Space Station automation system technology assessment summarized in Table 2-4 and a set of design doctrines for the Station Level work station or man-machine interface listed in Table 2-5. Many more details from the applications study may be reviewed in references 1, 2, and 3 in Appendix A.

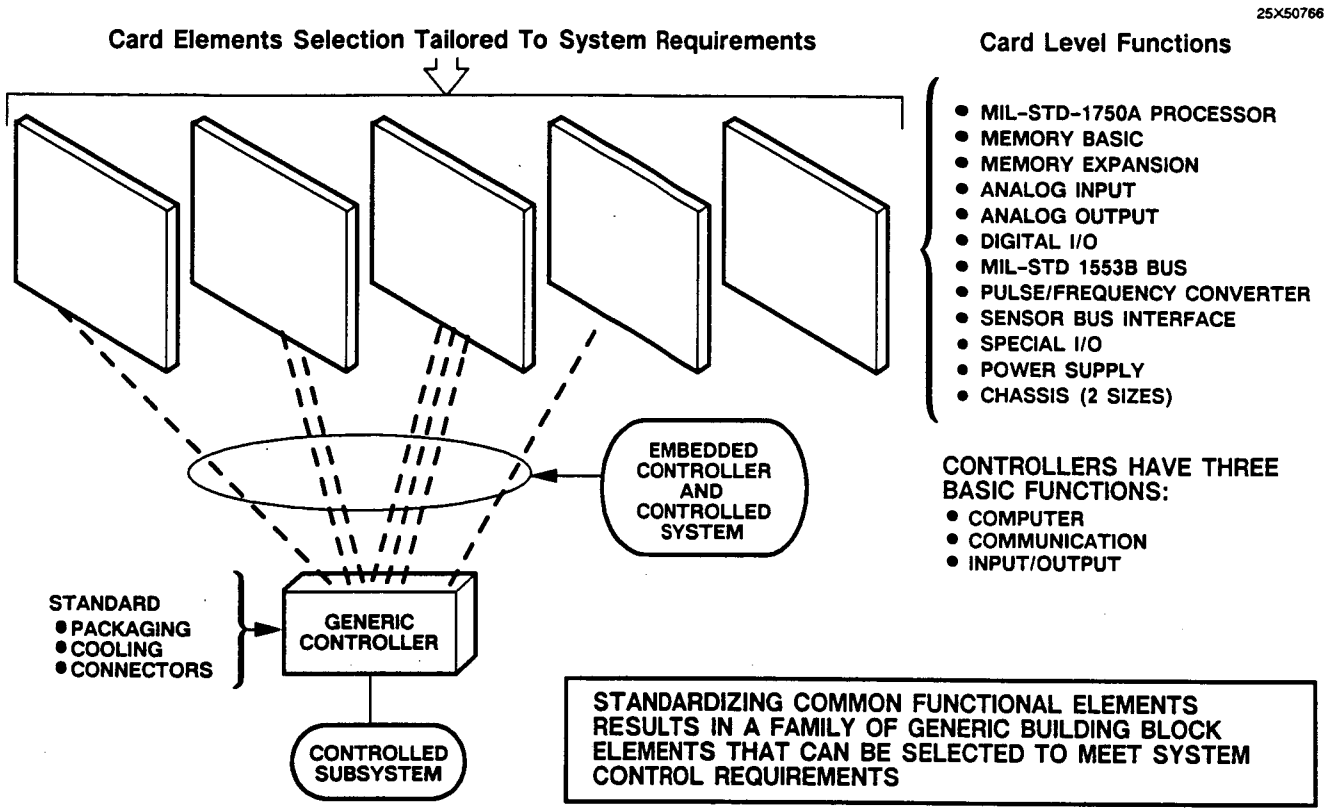


Figure 2-2. Honeywell Generic Controller Concept

Table 2-1. Worst-Case Requirements for Basic Controller and Interconnect

	ECLSS	EPS	G&NC
CPU	185 KIPS	50 KIPS	240 KIPS
ROM (16-Bit Words)	32K	40K	32K
RAM (16-Bit Words)	32K	16K	32K
Mass Storage(1) (16-Bit Words)	92K	0	258.2K
Bandwidth	64K bps	400K bps	50K bps

(1) Assumes one day of storage. No trend analysis in EPS basic controller, thus no storage.

Table 2-2. Bus Requirements and Recommendations

	Basic	System	Module
Media	Twisted Pair	Coax	Fiber Optics
Type	Serial	Serial	Serial
Data Rate	1 MBPS	10 MBPS	10-500 MBPS
Access Method	Single Bus Controller	Single Bus Controller	Distributed Bus Control
Delay*	58 Millisec Worst Case Transport Delay		

* Worst Case for Shuttle On-Orbit Operation

Table 2-3. Controller Family Performance and Capacity Recommendations

Family	Family Member	CPU (16 Bits) MIPS	RAM (X16) K	EPROM (X16) K(1)	Mass Storage (X16) (M^2)	Number of Dumb Sensors	Number of Dumb Actuators	Size(3) (In^3)	Weight(3) (lbs)	Power(3) (Watts)
Module Controller	A	2	80	336	400	0	0	TBD(4)	TBD	TBD
System Controller	A	2	80	336	10	0	0	TBD	TBD	TBD
	B	1	80	96	1	0	0	TBD	TBD	TBD
Basic Controller	A	1	80	96	0	72	10	300	11	44
	B	1	80	96	0	8	2	140	4	21
	C	1	16	32	0	72	10	250	10	40
	D	1	16	32	0	8	2	100	3	17

(1) Assumes most controller functions executed from EPROM

(2) Module controller data base includes maintenance procedures and logistics data
System controller data base includes data structures for all control points in the controller's domain

(3) Size and weight include card cage, enclosure and power supply but not cooling. All values are based on current technology

(4) Depends on type of mass memory selected.

Table 2-4. Controller Technology Assessment

	Technology Element	Recommendation	Rationale
A.	Instruction Set Architecture	MIL-STD-1750A (16 bit word)	Capitalize on availability of development tools, multiple hardware implementations, vast user base, facilitates portability of software
B.	CPU Throughput	1 mip CMOS, dais mix, single chip	Low power dissipation at sufficient speed
C.	Semiconductor logic circuitry for I/O and bus interfaces	Bipolar at 0.5 ns gate delay CMOS at 5 ns gate delay	Available
D.	Semiconductor Memory	CMOS static RAM (64K to 256K, 50 mw, 75 ns access) CMOS EPROM (64K to 256K, 50 mw, 150 ns access)	Low power with sufficient density and access time
E.	Bus Interfaces	MIL-STD-1553B (1 MBPS)	Well known standard used in many control applications
F.	A/D and D/A conversion	12 to 16 bit, ±10Vdc, 1 microsecond per bit	Available
G.	Integrated Sensors	On silicon sensors with associated microprocessor/memory/bus interface	• Not available until later 1980's • Offers control, size, weight, and cost advantages
H.	Chassis/Power Supplies	Standard size packages with mother/daughterboards and 80 percent efficient switching regulators	Modular packaging that accommodates on-orbit maintenance at the card level
I.	Software Higher Order Language	• ADA (MIL-STD-1815A) • Closest relative to ADA is Pascal	All future DoD and NASA flight software will require ADA. Space Station will select the ADA language

Table 2-5. Human/Machine Interface Doctrine

- Minimize crew workload for routine operations
  - Infrequent maintenance intervals (month or longer)
  - Maintenance tasks <1/2 hour
  - Subsystems automatic startup, shutdown, mode changes
  - Automatic fault detection, reconfiguration
  - Trend data automatically stored and interpreted
- Scheduled maintenance no sooner than 8 hours
- Display system designed to be user friendly
  - Menu driven data/displays
  - Built-in/integrated diagnostic capability
- Crew maintained in supervisory control
  - Caution, warning, alarms provided with explanatory procedures
  - Continuous system status provided at all times.

Section 3

DESIGN (TASK 3.1.2)

3.1 TASK OBJECTIVE

Based on the selected generic automation approach, a design for the automation and control of the air revitalization group (ARG) in the ECLSS system would be developed. The design would partition the functional requirements for performance monitoring and control, fault detection/isolation, trend analysis, and include various aspects of redundancy management and maintenance provisions.

As part of the preliminary design, differences in hardware and software implementation from the generic A&C technique defined for the Space Station in the application study (Task 3.1.1) and the proposed automation approach to be developed and demonstrated would be explained. The final design process was to include design requirements and specifications and software flow charts. Special emphasis was to be directed at defining and designing subsystem process simulators capable of checking out, testing, and demonstrating the generic automation technique to be developed on the ASCLSS program.

3.2 TASK APPROACH

Significant design tradeoffs were conducted to select the ASCLSS baseline automation design to be demonstrated. Concepts ranged from available commercial microprocessors, Life Systems, Inc. process controllers, to Mil-Standard control elements. Final design selection emphasized strong commonality of hardware and software, military standards, and real-time control features that are being considered in the Space Station DMS. The ASCLSS program goal was to implement the automation and control approach with high fidelity to Space Station so as to develop the most relevant lessons learned.

3.3 TASK RESULTS

3.3.1 Hardware System

The baseline ASCLSS demonstrator configuration is shown in Figures 3-1 and 3-2. The configuration consists of three basic (subsystem) controllers that provide the automation and control function for the three processes making up the air revitalization group (ARG) of the regenerative ECLSS subsystem. The ARG processes include:

- The O_2 generation process (static feed electrolyzer [SFE]).
- The CO_2 concentrator process (electrochemical depolarized concentrator [EDC]).
- The CO_2 reduction process (Sabatier CO_2 reduction subsystem [S-CRS]).

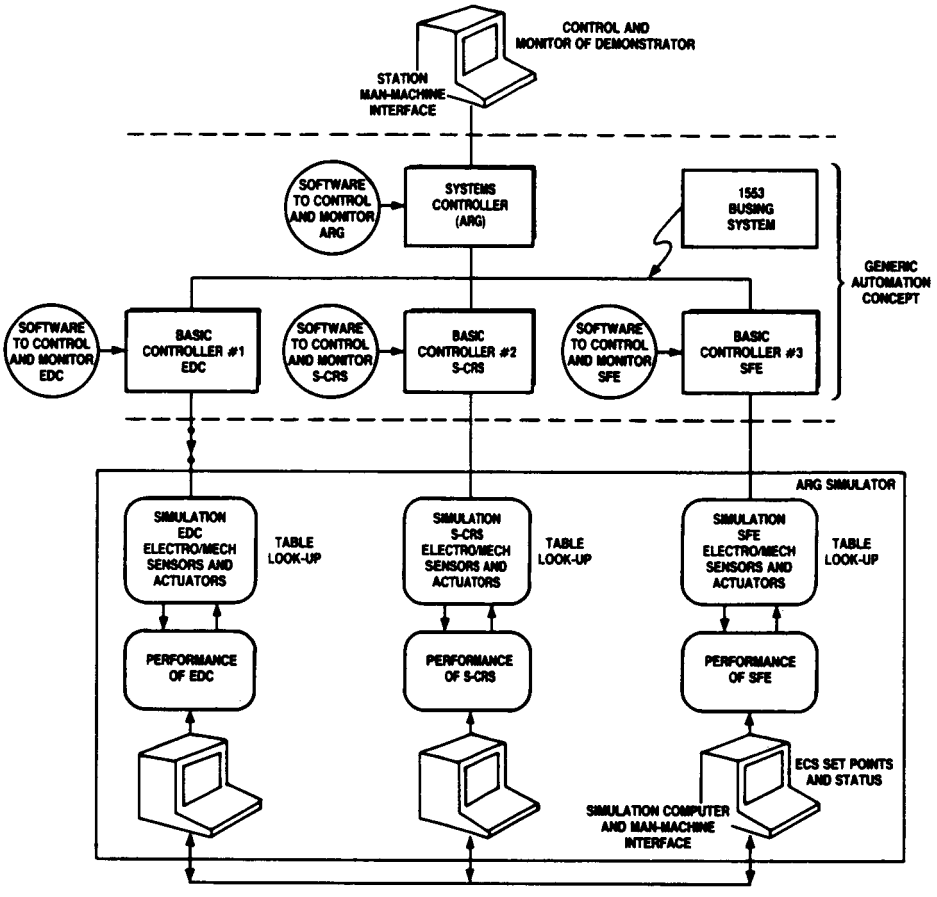


Figure 3-1. ASCLSS Demonstration System for ARG Automation and Control

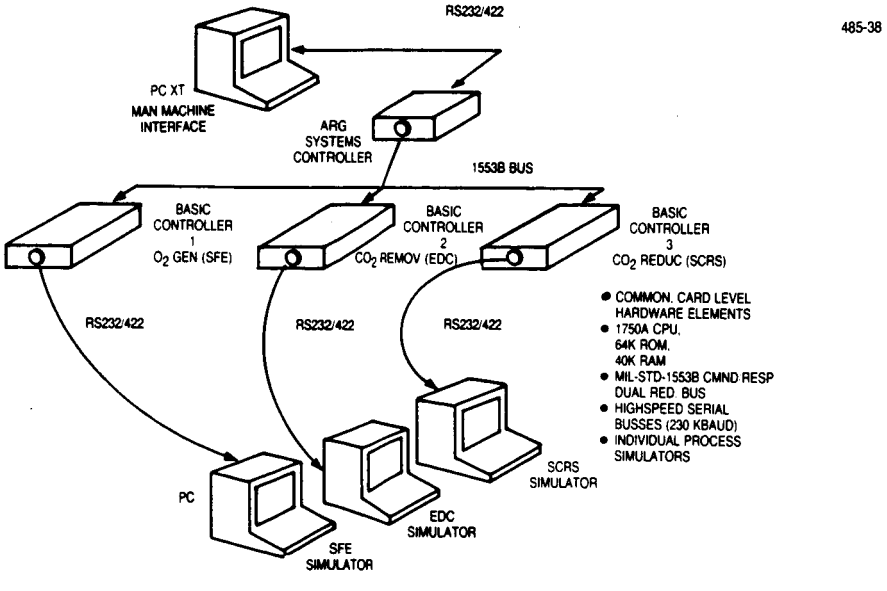


Figure 3-2. ASCLSS Automation and Control Demonstration System

The three basic controllers communicate to an ARG system controller via a dual redundant MIL-STD-1553B busing network. In the ASCLSS demonstration system, the ARG process hardware is represented by three real-time simulators implemented by personal computers. A man-machine interface (MMI) communicates with the system controller via an RS232/ RS422 serial bus. The MMI provides for ASCLSS automation system and ARG performance monitoring by the operator and provides for crew operational command authority.

Some of the more important features of this hierarchal and distributed automation system are shown in Figure 3-3 and listed below:

- All controllers at each level in the automation hierarchy are constructed from exactly the same building blocks or cards. All controllers feature the same, single board MIL-STD-1750A central processing unit (CPU) implemented with a Fairchild F9450 single chip microprocessor (750 KIPS). Selection of the MIL-STD-1750A microprocessor was based on its effective, functional architecture for high speed control processing and the availability and maturity of the software development tools, programming environments, and hardware. Figure 3-4 provides a photograph of the generic controller hardware and the associated 1750A CPU card.

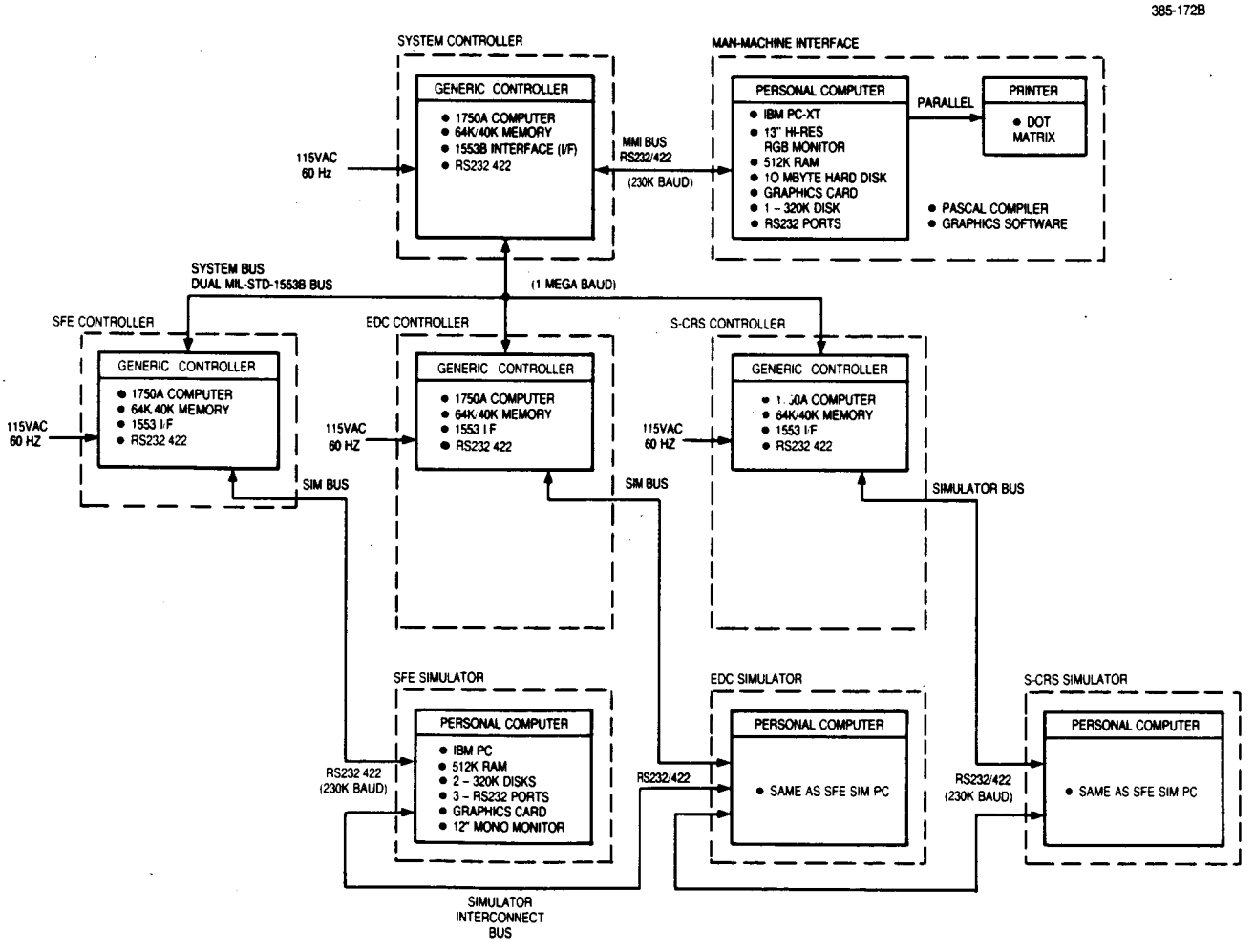


Figure 3-3. Approach for Automation and Control

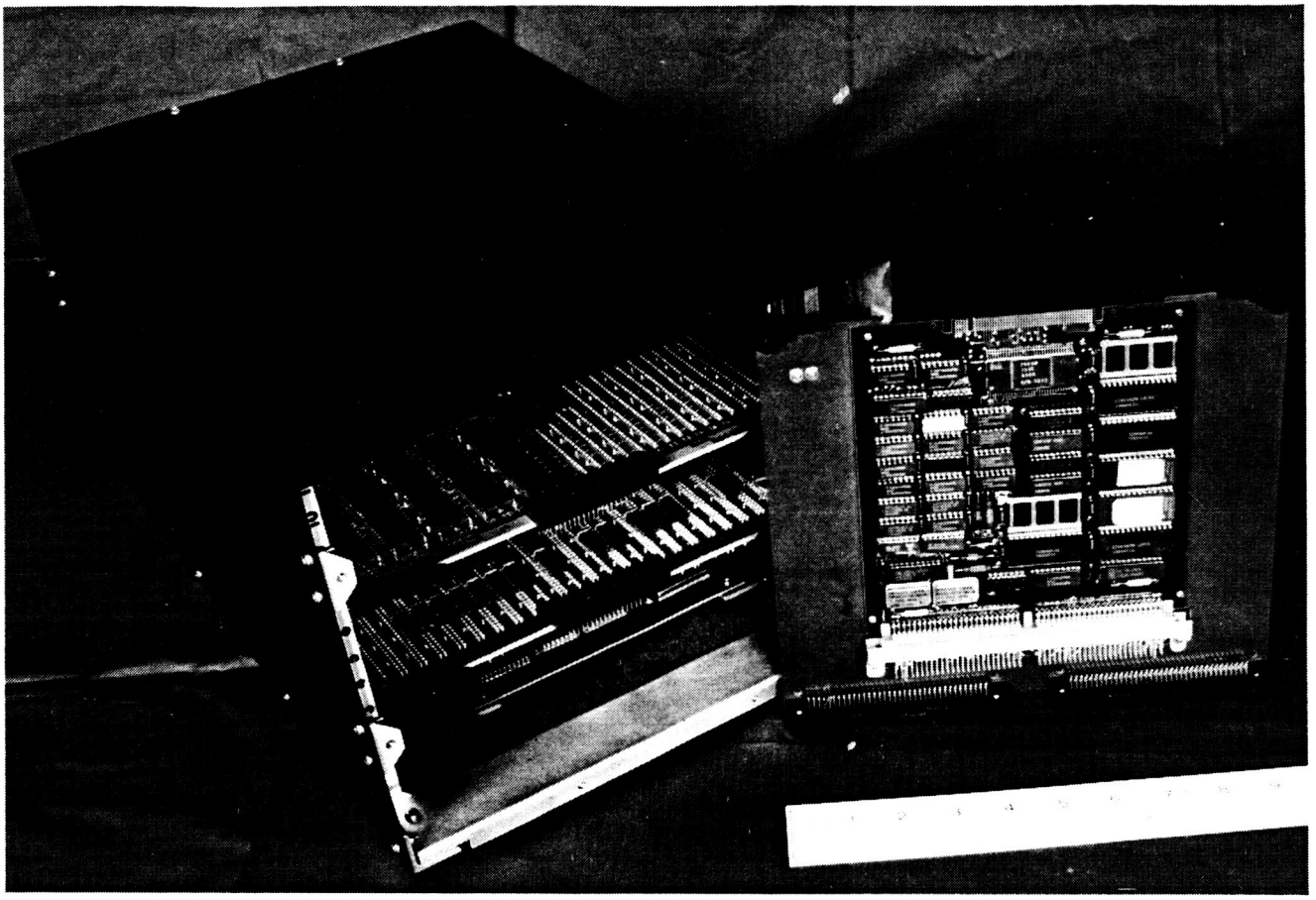


Figure 3-4. 1750A Microprocessor and CPU Card

- The memory cards in each computer/controller are the same (64K words E^2PROM, 40K words RAM). Where additional memory capacity is required, 8K words of memory can be easily added or additional memory cards of the same type can be added onto the CPU computer bus. The 1750A CPU can address up to 1 million words each of data and instruction memory.

- Bus interface cards are the same in each computer/controller where it interfaces to a specific communications bus (RS232/RS422 or 1553B). Bus protocol differences (i.e., "master" or "slave") are implemented in operating system software packages. The MIL-STD-1553B bus was selected because it is a high speed, standard bus that meets the demonstrator intercommunication requirements (1 MBPS).

- Although not implemented in ASCLSS, the input/output (I/O) or signal conditioning circuits required for a given process (i.e., EDC, S-CRS, or SFE) can be designed and/or implemented from standard/available electronic cards (e.g., signal amplification, Analog to Digital (A/D) and Digital to Analog (D/A) conversion cards).

- Note, in Figure 3-2 the ASCLSS demonstrator system also includes personal computers (PC) employed as ARG process simulators and as an MMI. The significance of simulators to the automation and control development cycle is discussed in Section 3.3.3 of this report. The high speed (230 KB) RS232/422 serial bus linking the PC simulators to the basic controllers was necessary to achieve real-time control performance from the simulations and controller interfaces.

The ASCLSS automation system was designed to demonstrate several important benefits that can be achieved through a common approach to automation across all Space Station systems:

- The generic design and implementation in common hardware and standard software allow automation techniques to be developed and validated to a high degree and then applied uniformly to specific applications across all of the Space Station systems.
  - Each subsystem or process vendor or component supplier does not have to conceptualize, implement, and validate his own automation approach.
  - All parties to the Space Station (NASA, subcontractors, crew) will develop and have a better understanding of all Space Station systems (important to system upgrades/evolution over the operational lifetime and to rotating crew and program personnel throughout the Space Station life cycle).
  - The many contractors and the large personnel base working with the generic automation concept and its implementation will provide a high degree of refinement, a broad integration and test base, and a greater assurance of hardware and software verification prior to Space Station initial operational capability (IOC).
- The generic automation approach allows a more universal approach to subsystem and system redundancy management (RM). Redundancy management requirements can be studied "across-the-board" and generic approaches and techniques devised to fit specific categories for system requirements (i.e., triple redundant, dual-dual, active standby, controller cross-strapping, etc.).
- The generic automation approach provides incentive for and accommodates the implementation of common and standard controls hardware and software (important for on-orbit maintenance requirements).
- The real-time automation and control hardware can be divided into three general categories:
  - Computer functions
  - Communications functions
  - Signal conditioning functions.

  Within each functional category, a library of standard electronic cards can be designed so each subsystem or process vendor selects from the standard cards to meet his particular controller requirements. Process or component controls can be designed and built from a standard set, or library, of generic controller electronic cards, chasses or packages, and power supplies. Each vendor selects from the generic equipment to build a controller for his specific process, component or subsystem.

Thus, commonality and implementation of standard or generic hardware and software demonstrated in ASCLSS appears to offer significant benefits in lower development and life cycle costs for Space Station automation and control. The ASCLSS approach also strongly supports the requirements for on-orbit maintenance, repair, and sparing (i.e., in this case the orbital replaceable unit [ORU] is at the electronic card level). Greatly simplified ground test checkout and hardware/software verification should also be realized eliminating the need for an extensive Station Avionics Integration Laboratory (SAIL) effort.

Many of these hardware features and their associated benefits are being considered in the current Space Station DMS and distributed control system designs throughout the Station.

3.3.2 Software System

The same benefits associated with commonality of hardware control elements are also achievable by providing a generic automation system with a significant content of common or generic software elements. This concept is currently being promoted in the present Space Station DMS architecture in the form of the distributed operating system (DOS). In the ASCLSS program, Honeywell developed a challenging layered software approach which facilitates independent verification and validation of application and operating system software.

Figure 3-5 illustrates the ASCLSS layered software approach employed in each of the four controllers. The approach has the following features:

- Each sublayer is transparent in functionality to the process control application software.
- Systems control and operating system software which are common in each controller require no repeated validation (developed and verified – once!).
- The controller I/O is performed by generic hardware and software modules.
- The native processor or target hardware (1750A) is not a development concern for the application supplier or process vendor (LSI in this case). The application software strictly deals with an I/O data base software architecture.

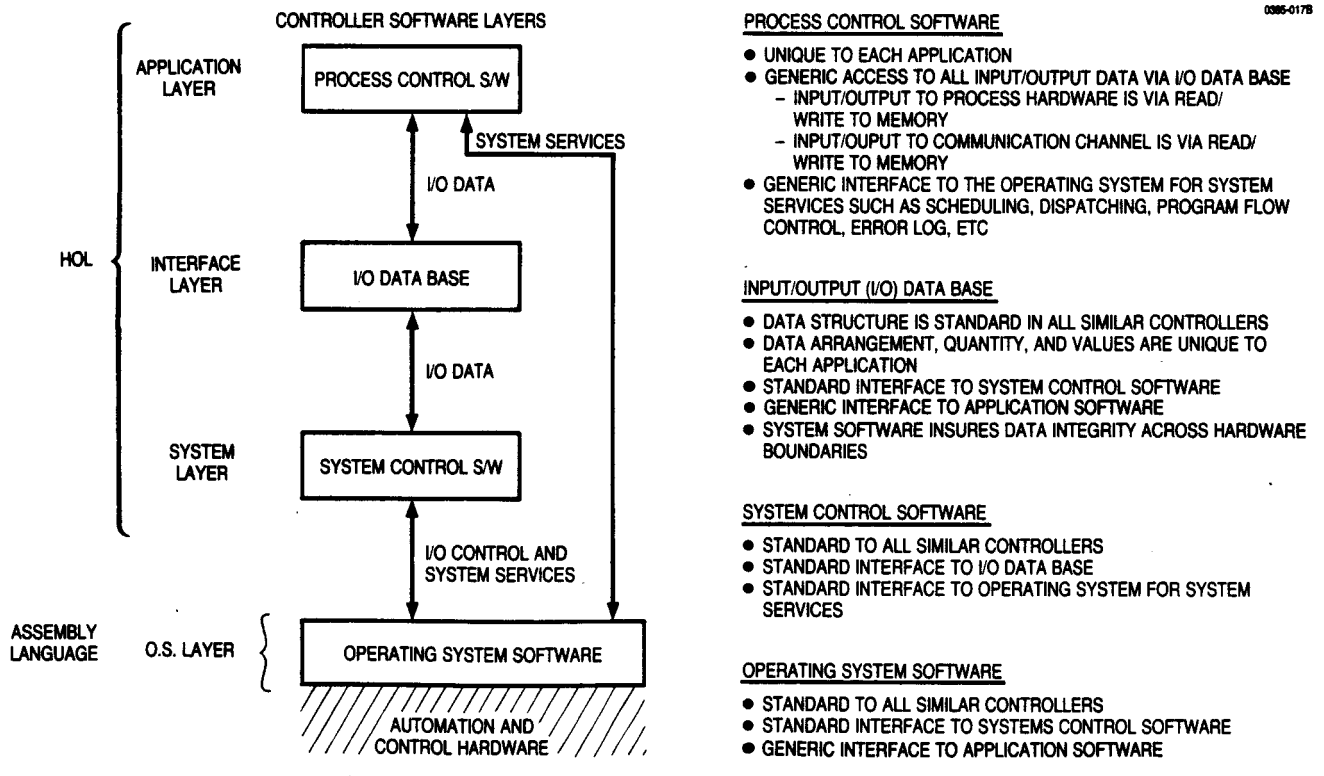


Figure 3-5., Generic Controller Layered Software Approach

- Process control application software development can take place with existing HOL tools at the process vendor's facility.

- Much of the automation/control operating system software is written directly in 1750A assembly language for reasons of computing speed and the fact that some lower level instructions cannot be implemented by an HOL statement.

- Only one software interface needs to be debugged at the integration phase (the I/O data base).

The ASCLSS program stressed software commonality and modularity. The ASCLSS layered software architecture shown in Figure 3-6 is divided into four layers:

- The operating system is common to all controllers. It is written in 1750A assembly language and performs all network and I/O data transfers as well as task scheduling.

- The system control software acts as the agent for the application software, with its I/O data base, and the operating system. Due to protocol features, there are some minor differences between the system control for the system controller and the subsystem basic controllers; the system control software in the three basic controllers are identical. The system control layer is written completely in Pascal.

- The I/O data base is unique to each controller. It is created through data declaration constructs and is used for transparently transferring data between controllers on the data bus and from/to sensors/effectors. The format/constructs for the I/O data base are the same in each controller, however the data is unique to the application.

- The process control (and process management) application software layer is the process application software which is unique to each application. This layer is written entirely in Pascal.

The ASCLSS application software and much of the system control software is written in Pascal. The ASCLSS software design and the selection of the Pascal language provide an easy path for future upgrading to ADA.

Honeywell addressed the ASCLSS baseline approach by defining a set of generic hardware and software elements that would be delivered to a subsystem and/or system developer that would address the problems of common hardware and software for a wide diversity of applications.

For a subsystem and/or system developer, the highest risk occurs at integration time. If system and subsystem requirements have been poorly defined or not implemented correctly, serious integration problems arise and often ripple into the design of the entire system. Most of these problems manifest themselves when the efforts shift from testing in a development environment to testing in the target environment. This is due mostly to operating a different hardware environment for the software development effort and then shifting from non-real-time to real-time events in the target hardware.

Development tools often become more primitive in the target environment and multiple events need to be examined at the same moment in time. An added problem is that application software tends to be very non-deterministic in execution, especially software which monitors and controls a subsystem. When a generic set of hardware and software are to be developed for application developers, all of these issues must be addressed.

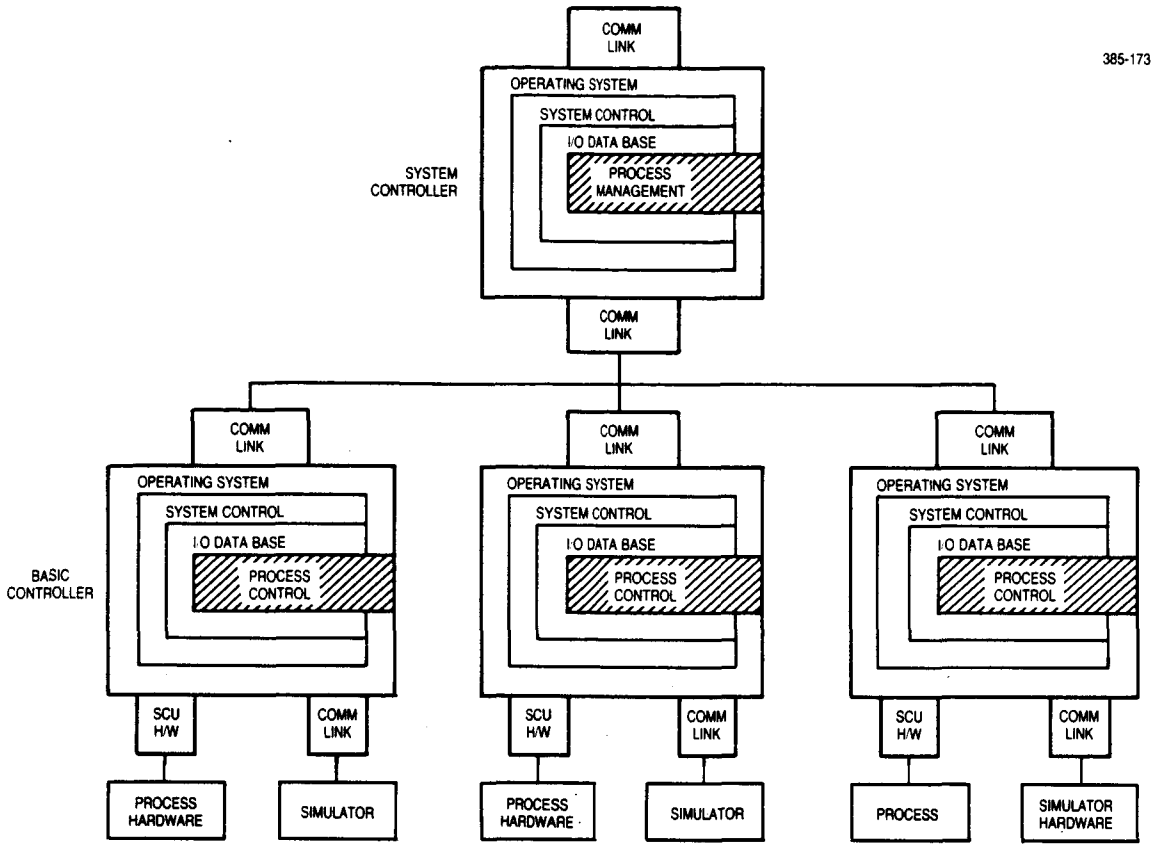


Figure 3-6. Layered Software Architecture Established Software Commonality

The automation and control of the ARG is typical of most systems which are comprised of distributed control elements. Generally these subsystem elements must be autonomous but they each affect each other through their roles in the ARG as a system. Therefore an overall supervisory monitor and control hierarchy for monitor and control is required. Each subsystem control element provides for autonomous control of the subsystem. All subsystem controllers communicate with a system level controller but not with each other, therefore a strict hierarchy is established. For ASCLSS, the MMI performs as the higher level control authority for the system controller, but in larger systems this control could be a module level controller or the operations management system (OMS) on the Space Station. The system controller can perform its functions autonomously without the MMI or the higher level control authority.

If the layered software architecture were not implemented, each Space Station application software developer would have to address the following major design issues:

1. Operating System Interface
2. I/O to sensors and actuators
3. Communications interfaces and protocols
4. Timing and Synchronization
5. Tasking Knowledge

6. Location of data in a multiple controller set

7. Delivering commands and data in a distributed controller set.

Honeywell addressed these design issues as a generic problem in order to implement its baseline common software approach. In ASCLSS, the application software developer (Life Systems, Inc) developed application and test software on an IBM PC which was targeted for a controller with 1750A instruction set, different system architecture, different operating system, different communications interfaces, and different execution speed.

The Honeywell layered software approach allowed the application software developer to develop software as a collection of finite state machines independent of the design issues described above. The finite state machine concept is an approach to decomposing a large, complex software task (i.e., the application control software for the ARG system) into easy-to-code modules and state-tables. The resulting software avoids convoluted code dependent on many different input conditions and facilitates test and verification of each clearly defined software module. Thus, the application software in each ASCLSS controller consists of a multitude of modules each responsible for a task or operation which it performs based on the state it's in and the current values of specific input parameters.

A table driven software layer defined as "System Control" software was developed which presented a total data interface to the application software in each controller. System Control software provides for synchronous and asynchronous communications between the system controller and subsystem controllers via symbolic data structures whose content is defined by the application software. The I/O to sensors and actuators is performed in the same manner. By using data structures as the only interface to the application software, the application software can be developed and tested on *any* development system since memory is handled the same on any computer from a High Order Language (HOL). These data structures collectively were defined as the I/O Data Base which comprise a distributed data base among all controllers. System Control software updates the required data structures each system minor cycle (100ms for ASCLSS). These data structures are the tables which drive the System Control software in each controller.

The innovative layered software architecture allows the subsystem or process developer to retain the controls responsibility and accountability, and in addition, achieves a consistent, common approach to operating system and communication system software.

The software design and development responsibilities for the ASCLSS program are shown in Figure 3-7 and listed below:

- Honeywell Design and Test Responsibility

  a. Automation and control software services programmed in PASCAL and in 1750A assembly language.

     - Operating systems services or scheduling
     - Communications services
     - Hardware I/O
     - Data base management
     - Controller redundancy management.

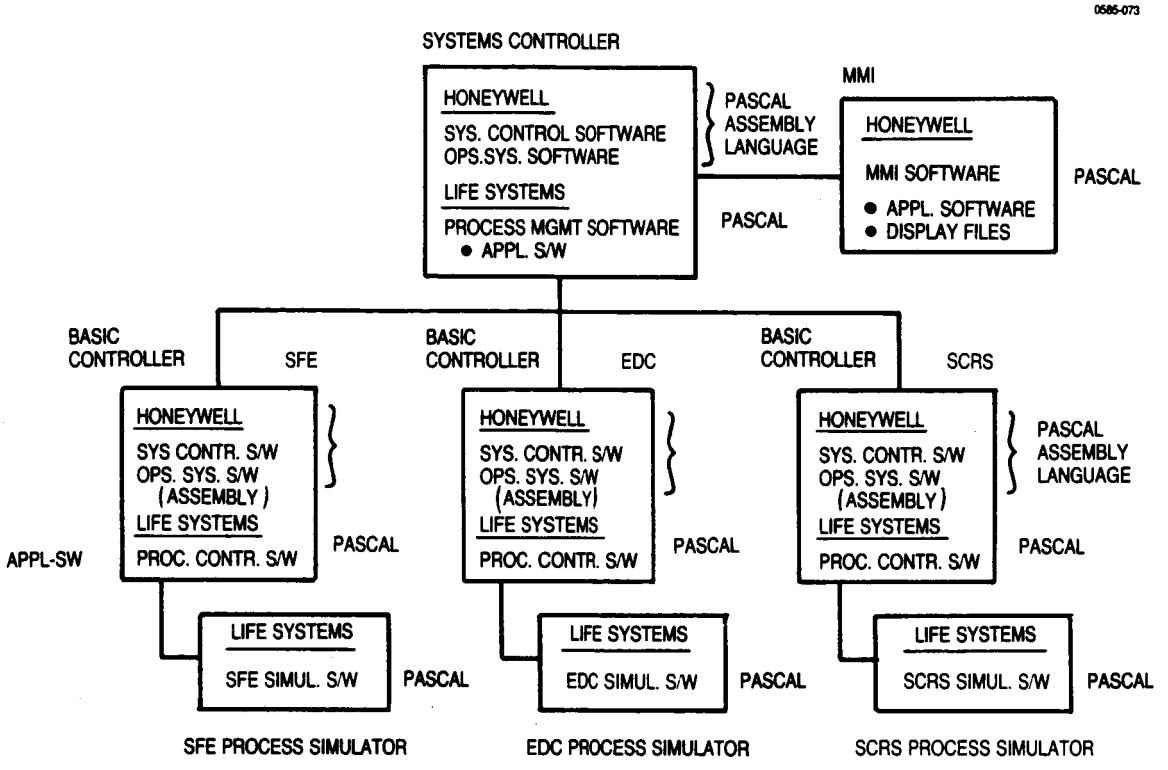


Figure 3-7. ASCLSS Software Partitioning

b. Man-machine interface displays and control operations programmed in PASCAL.

- Keyboard commands from operator
- Display formats/sequences
- ARG status, warnings, alarms, event logs, sensor/actuator set points.

- Life Systems, Inc. Design and Test Responsibility

a. ARG system control level and ARG subsystem process control or application software programmed in PASCAL.

- Control algorithms
- Startup/shutdown procedures
- Mode transition sequences
- ARG system control policies/procedures
- System level fault detection
- Monitor functions.

b. ARG process simulations programmed in PASCAL.

3.3.3 Application of ARG Process Simulators

The ASCLSS demonstrator architecture shown in Figure 3-1 illustrates the use of simulators as a replacement for the subsystem process hardware during development. The three ARG simulators (hardware and software) were developed by Life Systems, Inc. The simulators (implemented with commercially available personal computers) can be used to develop and checkout the process controller software before the flight hardware is available or they can be used to demonstrate contingencies or failures with greater flexibility than real hardware. In the latter example, the simulators can play an important role in testing or verifying the automation and control hardware and software including redundancy management aspects and artificial intelligence or expert systems (AI/ES) software without requiring the process hardware to be present.

Each of the three ARG process simulators respond to actuator output commands from their respective basic controllers and duplicate the response of the actual ARG process hardware to the same commands. This is accomplished by a dynamic computer model residing in each simulator. The ARG subsystems are the CO_2 Electrochemical Depolarized Concentrator (EDC), the Sabatier CO_2 Reduction (S-CRS) and the Static Feed Electrolyzer (SFE) for O_2 generation. Each simulator replaces its corresponding mechanical subsystem hardware package and provides a useful tool in the demonstration of the basic controller concept. Each simulator provides a friendly human interface that enables the user to monitor the simulation during system operation and to choose from a variety of operational scenarios.

The ARG simulations were developed and targeted on IBM PCs using PASCAL and RTOS (Real-Time Operating System). Each simulator was connected to a controller over an RS232/422 serial bus and to the other two simulators over an RS232 LAN. The RS232 serial bus carried the messages associated with sensor and actuator data and the RS232 LAN carried environmental and subsystem fluid flow interface data. Subsystem interface data is data relating to fluids, gasses, and temperatures shared where the subsystems physically interface to each other.

The ARG simulators also simulated an on-orbit crew loading in the module and the module environment in addition to the subsystem simulation. This was required since the ARG also interfaces and its performance is affected by the local atmospheric environment and the environment is driven by crew loading.

The following list describes the general features of each of the ASCLSS ARG simulators:

1. Retrieves commands (actuator signals) from the basic controller and calculates via software routines a new state of the subsystem. The simulator will subsequently send back to the basic controller information (sensor values) describing the new state of the process. It also provides process data to basic controller on command.

2. Displays a subsystem schematic continuously updated with new sensor, actuator, flow, valve setting, and fluid level data (figure 3-8).

3. Displays sensor/actuator status listings in tabular form in display windows. Prints out display content on operator request.

4. Displays the ARG subsystem's effect on the environment and the interfaces with other subsystems.

5. Provides an option of running the simulation in scaled time (real-time, faster that real-time or slower than real-time).

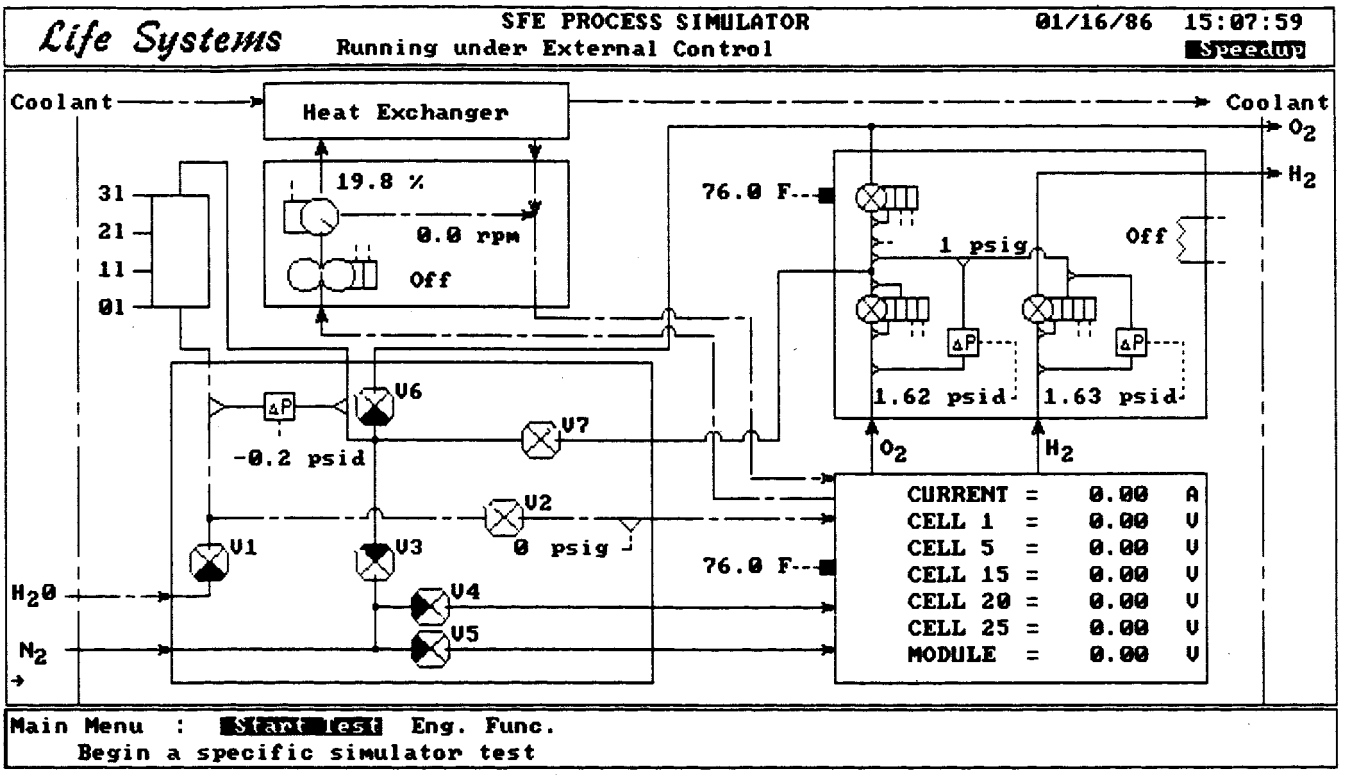


Figure 3-8. Simulator Schematic Screen

6. Provides the capability of conducting the simulation as a steady-state process with operator keyboard input or a dynamic process with communication (i.e., actuator signals) with the basic controller.

7. Allows the user to pause the simulation, store the data base on a floppy diskette, load an existing data base or restart the paused simulation.

8. Provides an option of operating the simulator in a stand-alone subsystem simulation mode or connected to other subsystem simulators to form an ARG network.

9. Provides for each subsystem up to ten preprogrammed, common failure events that the user can select and test the automation system response.

The primary benefit of utilizing the ARG process simulators is to checkout controllers before actual process mechanical hardware is available. Another benefit of the simulator is the ability to introduce hardware failures to demonstrate the controller's capability to handle a range of failures. Mechanical hardware would not aid such a demonstration without risking damage to the process equipment by operating at out-of-range conditions. The simulator can also be instrumental in the development of improvements to existing designs (rapid prototyping). By conducting feasibility studies, one can determine whether a new control or change to an existing control would be useful. Simulators can also provide a means to conduct parametric studies to determine interrelationships between system process variables. All of these benefits reduce subsystem development costs. More detailed information on the ARG simulators and the ARG application software development are contained in the Life Systems, Inc. Final Report, TR-569-30, May, 1986.

3.3.4 Generic Man-Machine Interface (MMI)

The success of the permanently manned Space Station will depend heavily on man's demonstrated productivity in space. Through the crew's ability to be productive and their flexibility to adjust to mission contingencies, man will establish his value in future Space Station operations.

Currently, manned space operations with the STS (Shuttle) depend heavily on the crew to control the complex subsystems and on-orbit activities. Though most Shuttle orbiter subsystems are to a large extent automated, the basic design philosophy demands a flight or ground crew control input/monitor function. The future Space Station design will exhibit high levels of automation including automated redundancy management. To meet these new challenges, the on-orbit crew role will transition from a "controller" to a "manager" of vast quantities of information.

Space Station systems studies indicate that the design of a system's crew interface should be integral to the design of the automation controller of that system. Experience has shown time and again that the MMI must be designed to reflect the levels of automation in the control system. At one time, for example, control systems were largely manual and the MMI reflected this automation approach by the provision of numerous dedicated controls for process mode selection and process control. Similarly, for virtually every control (or set of functionally related controls) there was a dedicated display for monitoring the process(es) under control.

The ASCLSS program emphasizes an automation technique featuring a high level of subsystem autonomy or system "self-control" (e.g., mode selection, system initialization and startup, and mode transitioning) and "self-monitoring". However, the MMI must be designed so that the crew member never is in a position of not being able to override the automatic control features of any Space Station subsystem.

The Space Station workstation or MMI will offer an operator a single console or workstation to monitor any of the many complex systems. The MMI can be extremely effective when the concepts of supervisory control, standardization, responsiveness and a forgiving interface are well thought out early in the system design phase. These issues are discussed below.

Supervisory Control. Provide the human operator with an integrated system representation at all times. This is especially important in the event of a critical system failure, when under high stress and mental workload, human operators may tend to narrow their attention to one or two displays and controls.

Standardization. Standardize display screen formats, display information coding techniques, control input formats and interaction procedures.

Responsiveness. Provide the human operator with the ability to move rapidly and easily between system, subsystem and component details. Minimize system response time and provide effective feedback for all user inputs. Never leave the user with a blank screen.

Forgiving Interface. Use software safeguards to prevent the user from making critical errors. Provides advise if improper command or unsafe request is made by the operator.

3.3.5 ASCLSS MMI Operational Requirements

The objective of the ARG MMI is to promote crew confidence in both the system performance and in the controls approach. The human operator is able to monitor system status and change system modes, sensor and actuator settings and responds to alarms or warnings.

Status Monitoring. The ARG operator can monitor the status and mode of all subsystems. Subsystem status will be OK, WARNING, or ALARM. Subsystem modes will be DEGRADED, SHUTDOWN, STANDBY, NORMAL, PURGE, or UNKNOWN (communication failure). If a subsystem process is in transition, text describing the transition is shown in the mode field (e.g., Normal –> Standby). The alphanumeric characters used are of normal polarity (white on black), without serifs and not slanted or rotated. Mixed upper and lower case is used to improve readability and uppercase alone is used in titles to focus attention. Figures 3-9 and 3-10 provide examples of the MMI displays for the ARG system and the CO_2 reduction process schematic.

Mode Transition Enabling. The ARG operator can enable system mode transitions. Required display information includes current mode and allowable mode transitions. Software safeguards preclude illegal transitions (and an error message will be displayed to the operator) or require the operator to provide additional inputs to confirm the transition.

Sensor Value Checking and Setpoint Changes. The ARG operator can access current subsystem sensor readings and setpoint values and modify the high and low alarm setpoints, the high and low warning setpoints and the high and low setpoint enables. Sensors that control an actuator also have high and low control setpoints that the ARG operator can modify. Setpoints may have any real value within a given range or the value NONE and setpoint enables will be ON or OFF. Software safeguards verify a high alarm value is greater than a high warning value which in turn is greater than a low warning value, etc. An error message notifies the operator of an illegal entry. A two-step control input process requires the operator to verify setting changes before implementation.

Actuator Setting Changes. The ARG operator has the ability to override and change subsystem actuator settings. Required display information includes current actuator setting and setting options. Depending on the actuator, setting options will be OPEN, CLOSED, ON, OFF, AUTO (no override), or a have a value. Again, a two-step control input process requires the operator to verify setting changes before implementation.

Software safeguards will either preclude illegal (e.g. unsafe) configurations and an error message will be displayed to the operator, or require the operator to provide additional control inputs to confirm such a setting. Subsystem status summary displays note overridden actuators.

Responding to Alarms. The ARG operator receives visual and auditory notification of subsystem alarm status, and visual notification of subsystem warning status. The ARG operator is required to acknowledge each alarm but not each warning. Alarm and warning status displays do not interfere with operator actions. An event log of subsystem events, alarms and warnings is available to the operator for display and printout.

In summary, the ASCLSS MMI was designed to move the on-orbit crew from an "operator" to a "supervisor" of multiple automated subsystems. The ASCLSS system automated the ARG and thus freed up the crew to perform user operations. However, the MMI allowed the crew to respond to a warning or alarm, exercise override command authority, and investigate system malfunctions, if required.

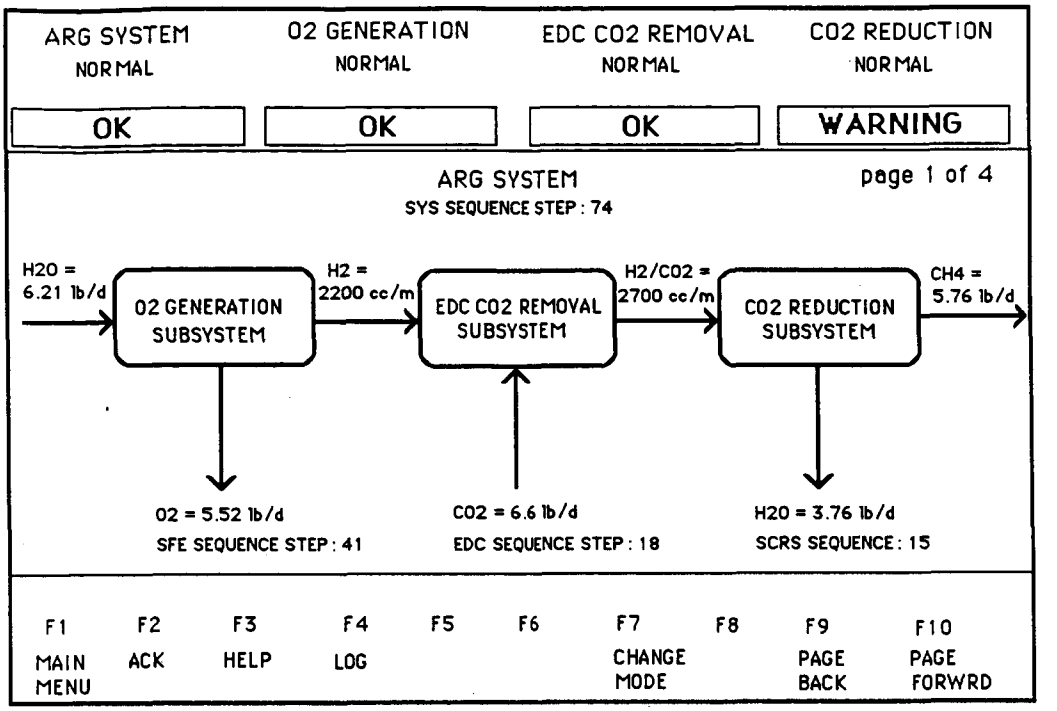


Figure 3-9. ARG MMI – Schematic Call-Up Display (ARG System Schematic is Shown)

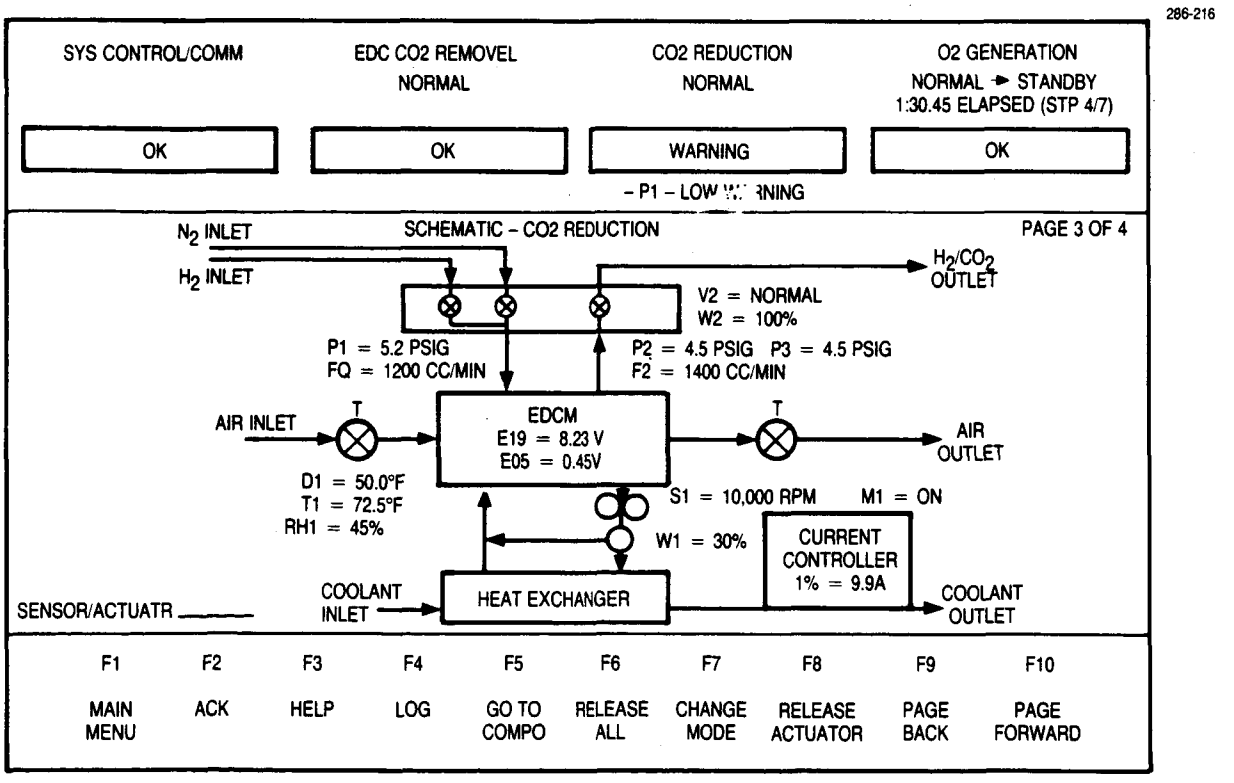


Figure 3-10. Representative MMI Schematics Required for Space Station Automated Subsystem Control

Section 4

FABRICATION (TASK 3.1.3)

4.1 TASK OBJECTIVES

This task included the purchase, fabrication, assembly, and unit testing of the generic automation and control hardware elements, with special emphasis on the subsystem simulators. The automation system software including application software, operating system software, and simulator software was to be written, debugged, and unit tested under this task.

4.2 TASK APPROACH AND RESULTS

The ASCLSS real-time controllers were fabricated with Mil-Spec electronics wherever possible. For example, the MIL-STD-1750A CPU cards and MIL-STD-1553B communication cards were provided from a Honeywell military helicopter program. All controller avionics boxes were exactly the same, allowing efficient build and spares strategies. All controller cards were individually tested in the controllers using test software and the target 1750A CPU.

In order to successfully develop the multiple ASCLSS software elements in three different company environments and effectively control and integrate them in the ASCLSS real-time controls system, a structured software development approach had to be employed. The following software specification and control documents were created and closely managed (Figure 4-1):

- *ASCLSS Software Specification (Reference 8).* The ASCLSS Software Specification defined the functional requirements and top level design approach for the Application software in the system and subsystem controllers, the system control software, the operating system software, and the ARG simulator software.

- *Pascal Software Development Guidelines (Reference 9).* The Pascal Software Development Guidelines defined the PASCAL language statement extensions (for the IBM PC) which could not be implemented in the controller and defined the Pascal system shell for the controller. This prevented use of statements not handled by the 1750A compiler.

- *I/O Data Base Control Document (Reference 11).* The I/O Data Base Control Document provided Life Systems with the definition of all of the data structures in the controllers. This was a living control document containing data sheets which Life Systems completed as they defined the content of the data structures.

- *Software Interface Control Document (Reference 12).* This document defined all the messages and formats that were to be communicated between the MMI, system controller, basic controllers, and the simulators.

- *Software Development and Integration Plan (Reference 10).* This document described the software environments at each facility and the test and verification requirements and the software development and integration plan for the entire ASCLSS program.

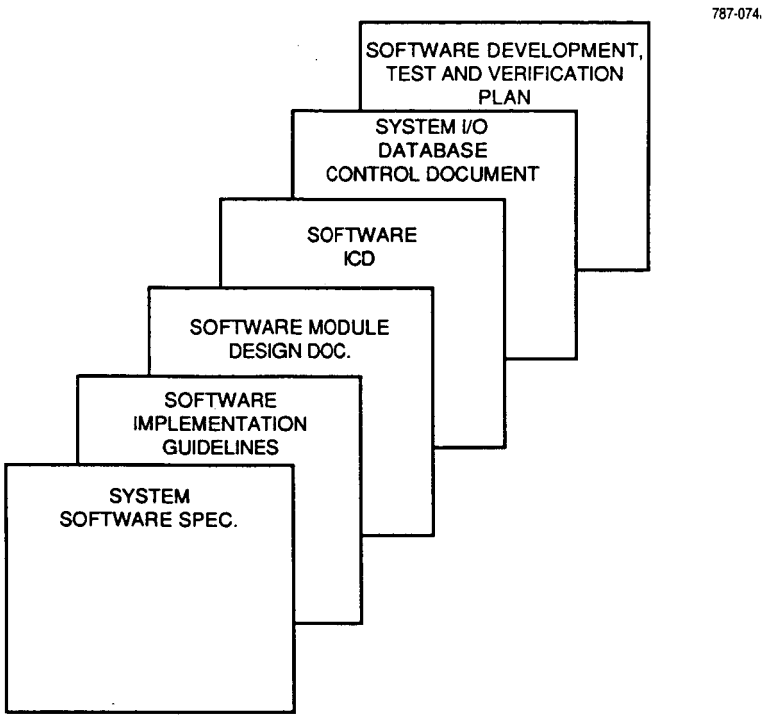


Figure 4-1. Structured Software Development Program

The generic layered software approach developed by Honeywell allowed for parallel software development of all software at the three development facilities. This was accomplished by the development of system control software in the generic controllers which provided virtual communications and I/O to the application software and a symbolic distributed data base for the MMI.

Since the application software developed at Life Systems, Inc. had no knowledge of the Instruction Set Architecture (ISA), the controller hierarchy, communications protocols, I/O interfaces, or the target operating system, the application software could be developed and tested on any software development system which supported a Pascal compiler. By avoiding extensions to the Pascal language standard and following the documents developed by Honeywell SSAvD, the application software could be statically and dynamically tested in the LSI software development system.

Since the system control software was isolated from the application software by a data interface and was table-driven (i.e., independent of any application), it was developed in isolation from the application. This provides confidence that the generic software developed on the ASCLSS program will work for future applications in different subsystems.

The MMI software issued commands to and requested data from the system controller on a symbolic basis. The MMI was able to request data from any controller and issue commands to any controller with no knowledge of the controller hierarchy or the specific data location. By following the Interface Control Document and the I/O Data Base Control Document, the development and testing of the MMI could and did proceed in parallel with all the software developed on the project.

The living software control documents, application source code, annotated software listings, program action registers, and monthly status reports were exchanged between the three geographically remote companies. This was accomplished by establishing a telecommunications network between Honeywell SSAvD, Honeywell SRC, Life Systems, and NASA JSC (Figure 4-2). By using modems, telephone lines, personal computers, and minicomputers, all documents and *software* were exchanged electronically providing instant feedback on changes, problems, and solutions. The effective application of telecommunications to exchange information and software was an essential component for the timely success of the ASCLSS program.

Communications between the ASCLSS participants was maintained through an early, TMIS-like network. LSI (the process experts) were able to develop both the control process software and the simulator software in their own facility on their development system (IBM PC), resulting in cost and time savings by using a development environment that LSI was accustom to. Updated versions of the ASCLSS application software were available to both Honeywell and NASA in a timely fashion through the use of bulletin boards and telecommunications. It is believed that this application of modern telecommunications to support software development and program technical management saved the ASCLSS program significant cost and greatly reduced program risk. Equally important, the ASCLSS layered software architecture and the tight control of the software interfaces also contributed to the parallel and independent development of the major software efforts on the program.

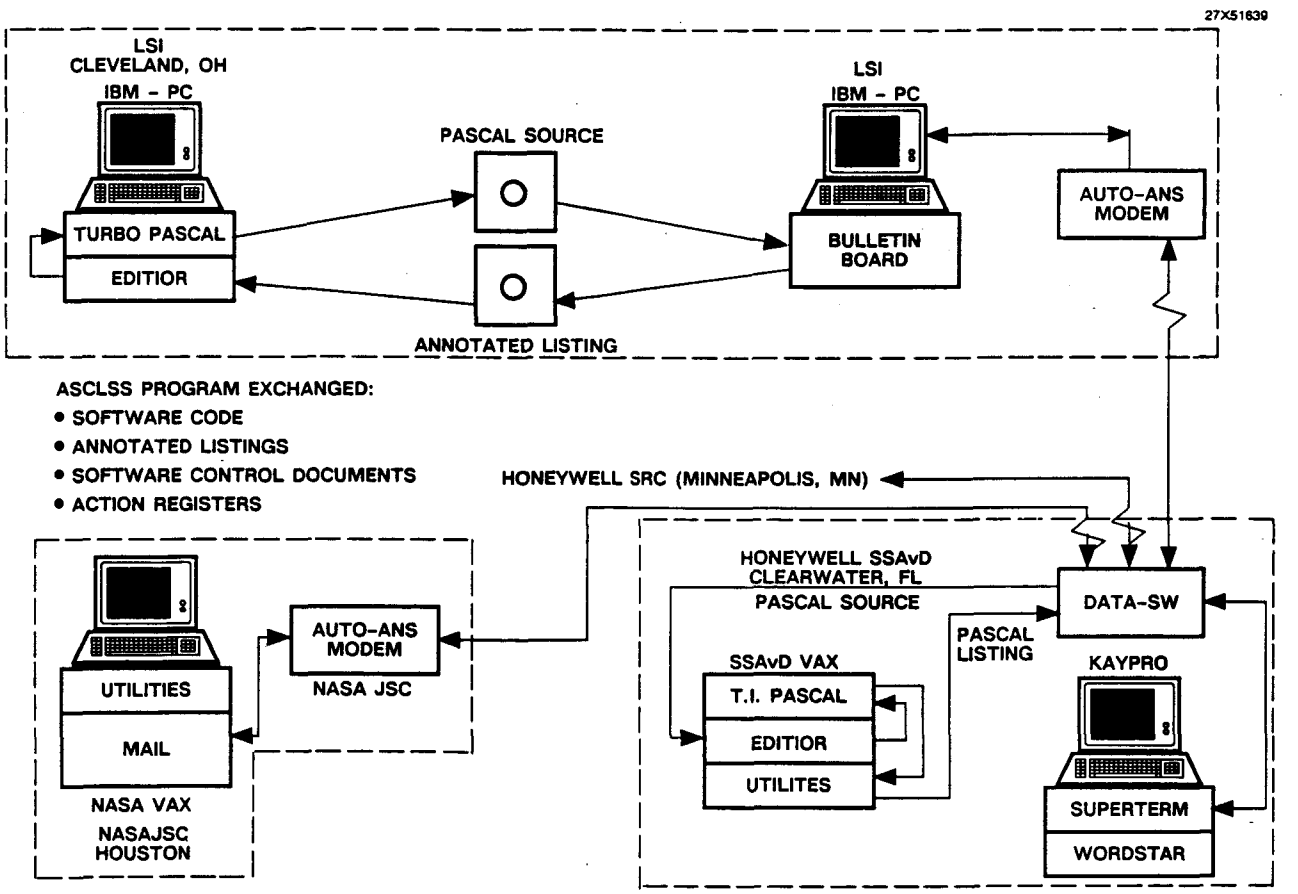


Figure 4-2. Telecommunications Were Key to ASCLSS Development and Integration Success

Figure 4-3 represents conceptually the ASCLSS application task or program and the data and command structures and information flow between the MMI, the system controller, the basic controllers, and the real-time process simulators. Note that commands and data are exchanged through table driven data structures established by the I/O database.

The ACLSS automation and control system memory budget or memory map is shown in Figure 4-4. The real-time operating system is the same in all controllers and occupies 3600, 16-bit instruction words in the controller's E^2PROM. Note that the ASCLSS program software runs out of E^2PROM (nonvolatile memory) while the operational data structures run in RAM. The expansion factor experienced in converting Pascal lines of code to words in the 1750A averaged approximately five. The memory budget in Figure 4-4 indicates that the ASCLSS software program in each controller occupies only about 25-30% of the 64K words capacity of the E^2PROM and the data structures consume only 12-20% of the available RAM.

The 1750A microprocessor in each controller also has unused capacity for additional tasks. Diagnostic measurements made for the controllers indicate in Figure 4-5 that only about 60% of the 100 millisecond minor cycle is used by the ASCLSS controllers to complete the normal automation tasks during each cycle.

Thus, significant memory resources and CPU capacity exists to perform additional tasks like redundancy management, process and controller health monitoring, and embedded artificial intelligence/expert systems (AI/ES) in real-time control.

And finally, the ASCLSS program represents a mini-model of how the Space Station automation and control development will be accomplished through coordinated efforts between NASA, the DMS subcontractor, and multiple process and hardware subsystem developers. Many of the important ASCLSS lessons learned, which are summarized in Section 7 and detailed in Appendix B, will have direct applicability to the future Space Station automation and controls development.

Because of its relevancy to the future Space Station, and at NASA's request, an evaluation was conducted to assess the size and cost of the significant software development completed on the ASCLSS program. Table 4-1 summarizes the results of this software analysis. In this table, the term "SLOC" stands for Single Line Of Code written in high order language. The following points can be made:

- Honeywell software efforts included: (1) systems engineering, (2) development of the system control software, operating system, and test software, and (3) overall system software integration.

- The cost of systems and integration responsibility adds significantly to the software program software development cost (35%).

- Application software provided by LSI was ported from existing ARG process control applications to Pascal and, thus, LSI's rate of development in SLOCS/day are higher and Cost/SLOC are somewhat lower than Honeywells.

- These figures of merit for software development are significantly lower than currently being used for planning purposes on Space Station. However, the ASCLSS software is development software, not flight software. Also, redundancy management functions are not implemented and the software was written in Pascal instead of ADA. These additional features are expected to add significantly to the Space Station software development costs.

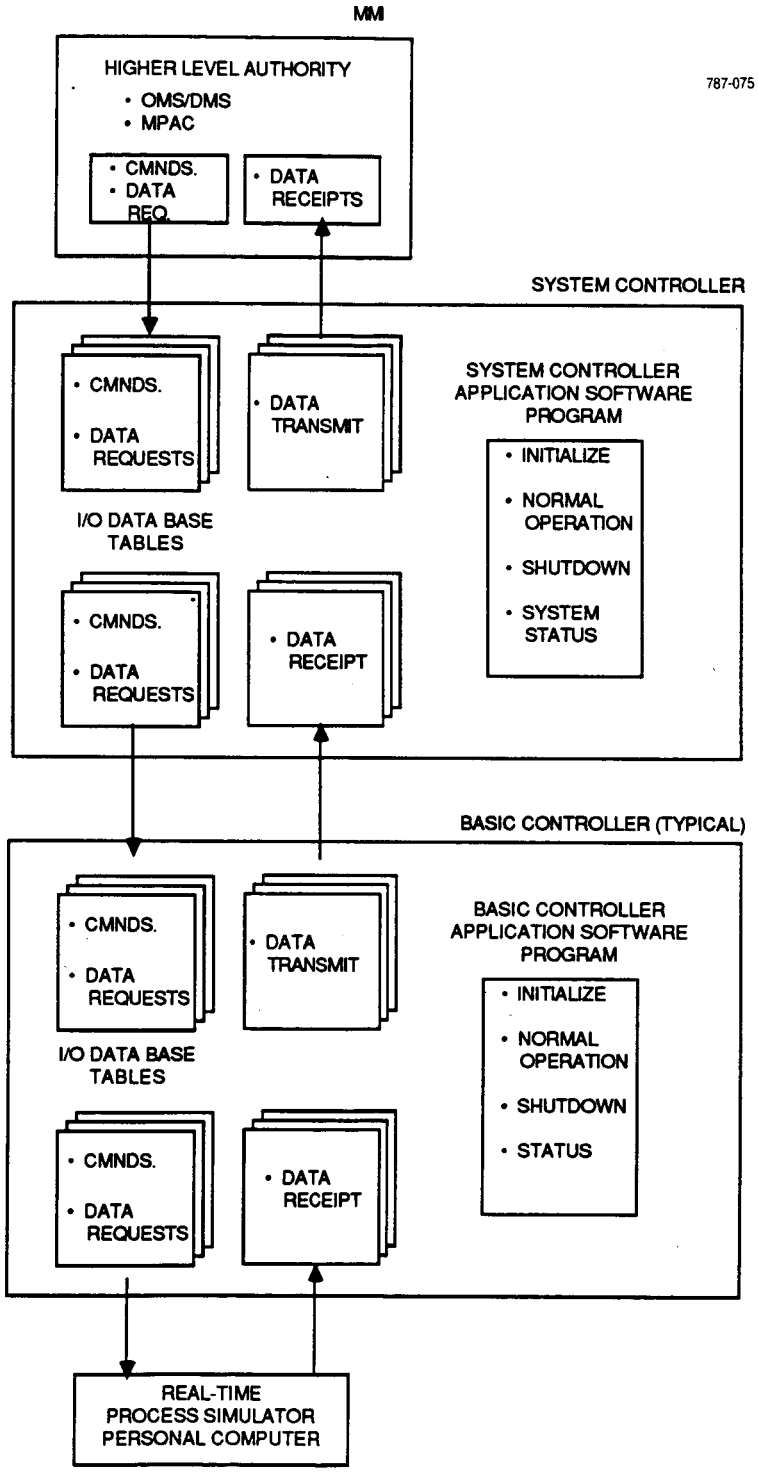


Figure 4-3. ASCLSS Top Level Software Architecture

ARG System Controller

727-076

Software Function	Pascal SLOC	Instruction Words	Data Words
Oper. Sys.	--------	3,600	9,133
Sys. Control	2,000	10,020	180
Appl. SW	1,040	5,290	465
TOTAL	3,040	18,910	9,778
Available Memory	--------	64,000	40,000
Margin for Growth	--------	45,090	30,222

SFE Basic Controller

Software Function	Pascal SLOC	Instruction Words	Data Words
Oper. Sys.	--------	3,600	4,783
Sys. Control	1,000	5,000	120
Appl. SW	2,474	12,370	675
TOTAL	3,474	20,970	5,578
Available Memory	--------	64,000	40,000
Margin for Growth	--------	43,030	34,422

EDC Basic Controller

Software Function	Pascal SLOC	Instruction Words	Data Words
Oper. Sys.	--------	3,600	4,723
Sys. Control	1,000	5,000	120
Appl. SW	1,691	8,455	630
TOTAL	2,691	17,055	5,473
Available Memory	--------	64,000	40,000
Margin for Growth	--------	46,945	34,527

S-CRS Basic Controller

Software Function	Pascal SLOC	Instruction Words	Data Words
Oper. Sys.	--------	3,600	4,273
Sys. Control	1,000	5,000	120
Appl. SW	1,556	7,780	510
TOTAL	2,556	16,380	4,903
Available Memory	--------	64,000	40,000
Margin for Growth	--------	47,620	35,097

Figure 4-4. ASCLSS System Memory Budget

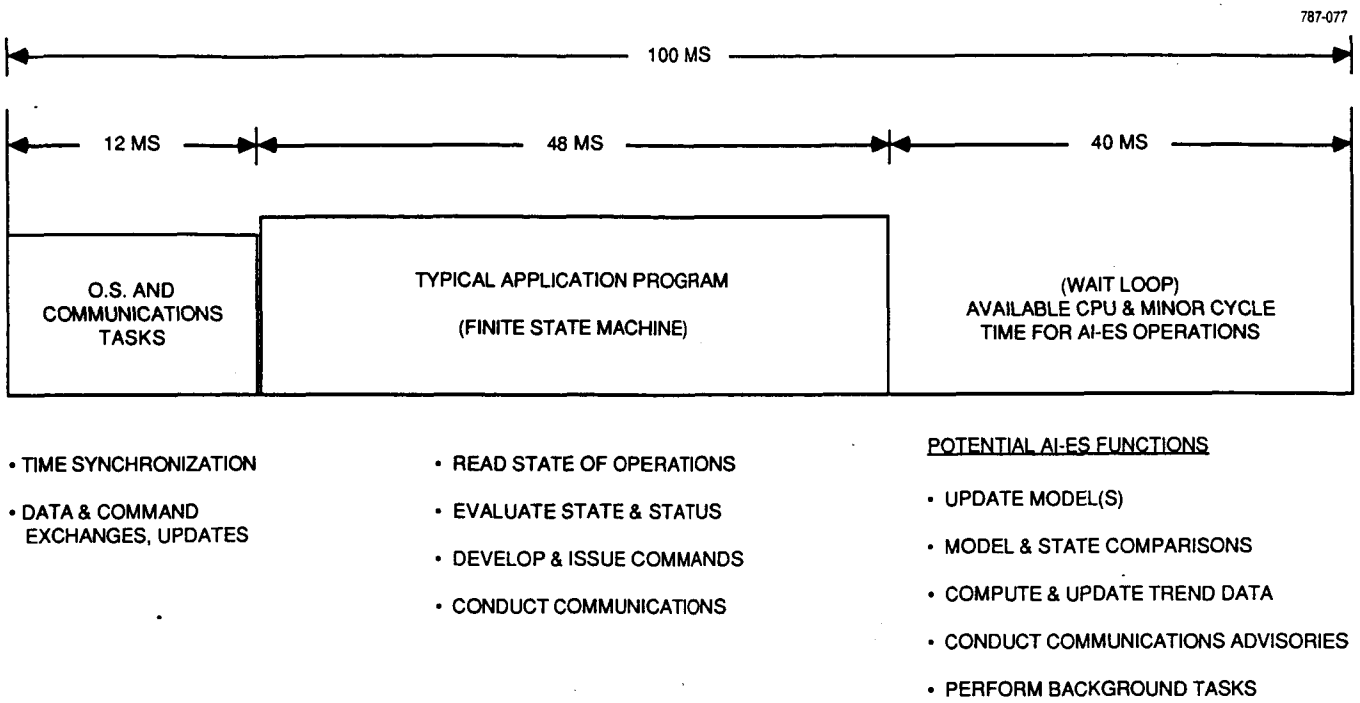


Figure 4-5. Controller Minor Cycle Headroom

Table 4-1. ASCLSS Software Development

Honeywell		**Life Systems, Inc.**	
Operating System (Assy)/Test SW	9,600 SLOCS (equiv)	Application Software	10,700 SLOCS
System Control SW (Pascal)	3,300 SLOCS	Simulator Software	10,900 SLOCS
Total	12,900 SLOCS	Total	21,600 SLOCS
Labor		Labor	
Systems	200 Man-Days	Application Development	300 Man-Days
Development	900 Man-Days	Simulator Development	350 Man-Days
Integration	260 Man-Days		
Total	1,360 Man-Days	Total (Design, Devel, Test)	650 Man-Days

Software Development	**Systems & Sw Devel**	**Systems & SW Dev & Integ**	**Application Development**	**Simulation Development**
$36/SLOC	$46/SLOC	$56/SLOC	$18/SLOC	$12/SLOC
14 SLOCS/Day	12 SLOCS/Day	9.5 SLOCS/Day	18 SLOCS/Day	31 SLOCS/Day

Comments:

- Not flight software
- Redundancy management not includled
- Software is not ADA
- LSI application software translated to Pascal from existing ARG software

Key:

- Single line of Code (SLOC) in high order language

4.3 APPROACH TO QUALITY ASSURANCE, RELIABILITY AND SAFETY

Though ASCLSS represents a technology development program, the implementation of the demonstration system design, development, and test was performed at a Honeywell division whose major business is man-rated, space qualified avionics hardware.

Within constraints of cost and the ASCLSS program development schedule, many features were incorporated and fabrication policies and procedures were followed which relate to quality assurance, reliability, and safety in the ASCLSS demonstration system.

A summary of the Q/A, reliability, and safety efforts performed on the ASCLSS program are listed below:

- Quality Assurance:

    1. The ASCLSS common controller hardware included 1750A CPU and 1553B cards from a military flight program (Mil-Spec parts and assembly).
    2. The basic controller chassis was redesigned to a small, compact configuration (nearer to flight size).
    3. The cooling fans in the controllers were replaced with high speed fans exhibiting double the flow initially designed in.
    4. An E^2PROM protection circuit and switch are provided to prevent inadvertent program erasures.
    5. All hardware assembly was performed by highly skilled aerospace engineers and technicians using standard practices.
    6. The controller cards were "burned-in" tested at least 300 hours prior to delivery.
    7. High quality electronic parts were used wherever possible.
    8. Higher flow rate fans were also installed on the IBM PCs to increase thermal margins.
    9. A step-by-step sequence of tests from card, box, and system was employed for both hardware and software.

- Reliability

    1. Added thermal margin and protection were provided by larger fans in the ASCLSS controllers and PCs which will improve electronic parts reliability.
    2. Differential driver cards were added to each IBM PC to increase communications reliability and performance.
    3. E^2PROM memory protection circuit was provided to maintain controller software integrity.
    4. The controller CPU and 1553B cards are military flight quality boards.
    5. The controller chassis, backplane, connectors, RS232 card, and memory card are fabricated from high quality commercial parts.
    6. The ASCLSS demonstration system does not preclude the efficient application of a more detailed reliability program in the future.

- Safety

  1. Detailed system assembly instructions were written to provide safe and effective assembly of the ASCLSS hardware.
  2. Current/circuit protection was provided by fusing at the controller rear panel and in the power supply.
  3. The controller chassis was designed to protect the user from electric shock with a box cover and fan guards were included for all controller and PC fans.
  4. Antistatic mats for floor and bench were employed for all assembly and repair operations. Cards were stored and shipped in antistatic envelopes.

Section 5

TEST OF THE AUTOMATION APPROACH (TASK 3.1.4)

5.1 TASK OBJECTIVE

Under this task, the hardware and software elements of the generic automation approach and subsystem simulators were to be thoroughly integrated and tested. The testing program objectives and plan were to test all elements for proper operation, and demonstrate the of generic aspects of the automation and control technique to the air revitalization group using the subsystem simulator(s). The completion of the task would be an acceptance test program designed to demonstrate these dual objectives.

5.2 TASK APPROACH

As discussed previously, the layered software architecture and division of development responsibilities between the ASCLSS contractors allowed a parallel and independent development and test plan to take place. The definitive software control documents and the use of telecommunications to coordinate changes and status of all activities greatly facilitated the parallel development approach.

The sequence of test and integration steps for the ASCLSS automation and control system are shown in Figure 5-1. At Honeywell, the operating system and I/O data base and system control software were tested in the system and basic controllers using test software that simulated ARG application software. Once a system controller and basic controller were tested successfully, the MMI and ARG simulator were added and tested using test software. Following a successful ASCLSS "single string" test, the remaining two strings or channels of the system were added. A full-up system test at Honeywell using strictly test software was conducted which included a systems communications "stress test". During a two hour test of the entire system, 1.36×10^6 messages, or 5.4 trillion bits of data, were exchanged successfully between the controllers on the 1553B busing network without a single error!

In parallel with the Honeywell SSAvD developments and tests, LSI developed their ARG application software (ARG system management and SFE/EDC/S-CRS process control) and SFE/EDC/S-CRS simulator software. These software packages were developed and tested both statically and dynamically at LSI on IBM PCs. At the same time, the MMI software was also being developed at Honeywell SRC (Minneapolis) using test software to simulate the ASCLSS system interface and data flow.

Thus, as Figure 5-1 indicates, all software was tested at the three separate development sites prior to final integration at Honeywell SSAvD in Clearwater. This resulted in a uniquely successful integration of all ASCLSS software in less than a week. This process essentially took place by removing the test software and linking in the ARG application software as illustrated in Figure 5-2. Following integration, software problems that arose were efficiently and effectively resolved by making software changes and using the telecommunications system discussed previously.

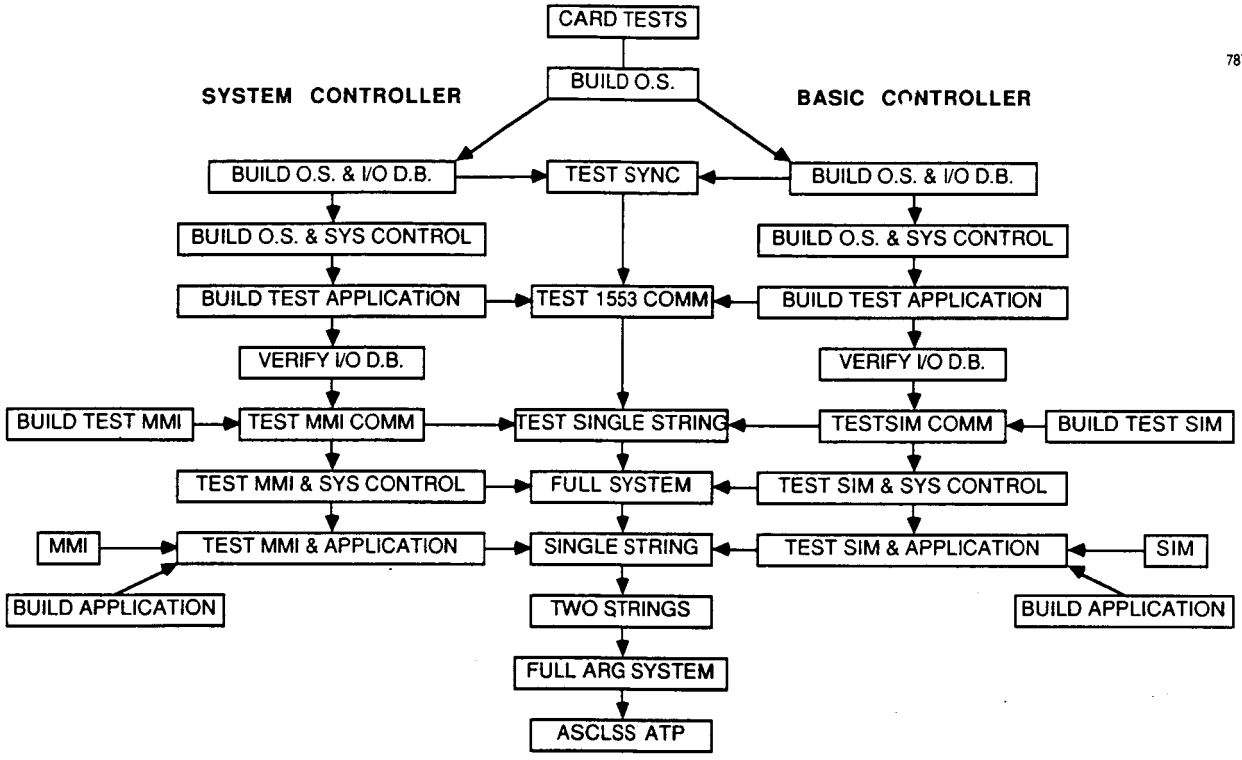


Figure 5-1. ASCLSS Integration Plan

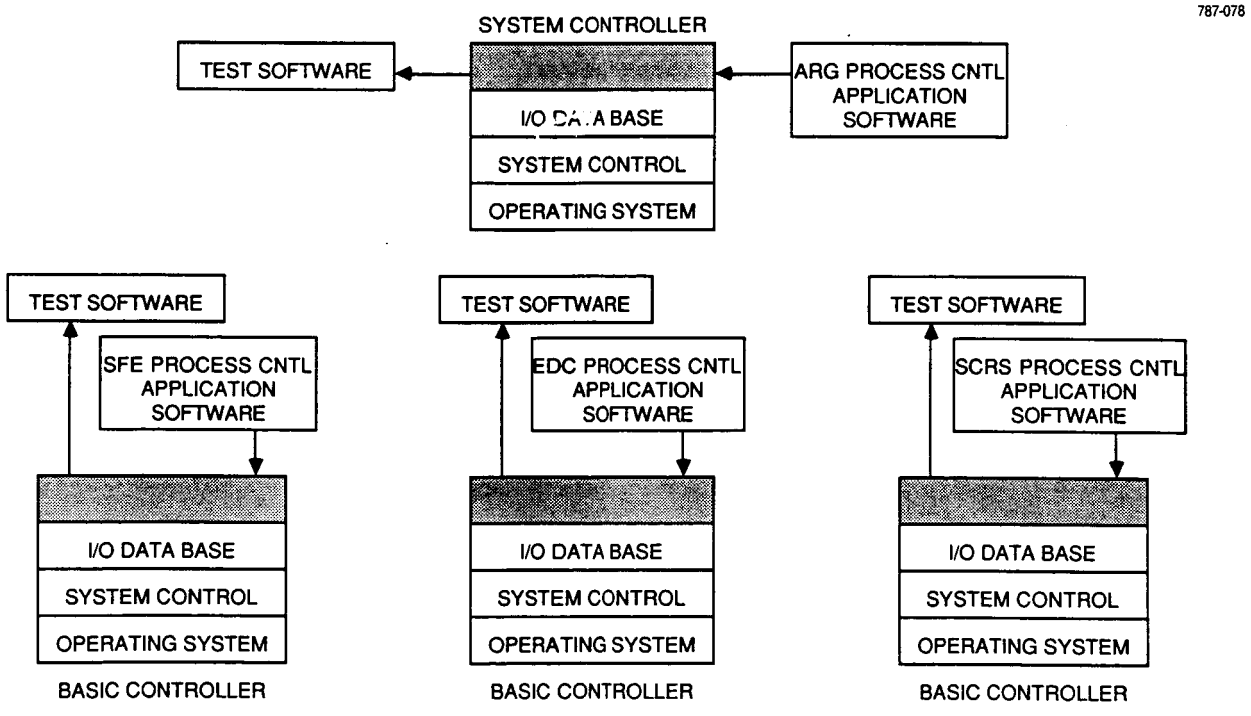


Figure 5-2. Application Software Integration Strategy

A detailed acceptance test plan and procedure (reference 17) was developed which included a logical sequence of tests from card testing to a complete ASCLSS system test that demonstrated the critical features of the automation and control approach. Table 5-1 summarizes the important tests included in the ASCLSS acceptance test program.

Table 5-1. ASCLSS Acceptance Test Sequence

1.0 Automation Hardware Tests (in each controller)

- 1.1 Memory Test
- 1.2 CPU Self-Test/Timer Tests
- 1.3 MIL-STD-1553B Test
- 1.4 Operating System/RS232 Communications Test

2.0 Automation System Tests (Test Software Driven)

- 2.1 I/O Data Base Test
- 2.2 Primary System Bus (MS 1553B) Failure
- 2.3 Secondary System Bus (MS 1553B) Failure
- 2.4 System Bus Channel Failure Test
- 2.5 Simulator Bus (RS232) Failure
- 2.6 Controller Hardware Failure Test

3.0 ARG System Tests

- 3.1 Power Up, Initialization Sequence Test
- 3.2 Mode Transition Tests
- 3.3 Controller Tests
- 3.4 MMI Monitoring and Control Tests
  - Sensor Failure Tests
  - Actuator Failure Tests
  - Process Autonomy Tests
- 3.5 Controller Failure Tests
  - System Bus (MS 1553B) Failure Test
  - System Controller RS232 Failure Test

4.0 Two-Hour System Test in Normal Mode

Section 6

DELIVERY TO JOHNSON SPACE CENTER (JSC) (TASK 3.1.5)

6.1 TASK OBJECTIVE

At the conclusion of the integration and test program, the automation equipment was shipped to NASA JSC, setup and demonstrated. Program documentation was delivered within one month of system hardware delivery.

6.2 TASK APPROACH

The ASCLSS system was successfully acceptance tested at NASA JSC in July of 1986. Following a brief period of system demonstration at NASA the system was returned to Honeywell to incorporate several enhancements that were to improve the demonstration features of the ASCLSS automation and control system:

- Incorporated system and individual ARG process sequence steps to be displayed on the MMI during system mode transitions
- Provided capability for hardcopy printout of MMI event log
- Incorporated a speed up mode in the ARG simulators that allowed the ASCLSS system to move through its transitions in significantly shorter times (roughly a factor of 10 faster than real-time representation).

These enhancements were completed in March of 1987. Following a demonstration of the ASCLSS system at NASA Headquarters on April 4, the system was again delivered and acceptance tested at NASA JSC. Figure 6-1 presents a photo of the delivered ASCLSS demonstration system.

Figure 6-1. ASCLSS Demonstration System

Section 7

SIGNIFICANT PROGRAM LESSONS LEARNED

The true legacy of the ASCLSS development program are the many and significant lessons learned. A list of 50 of these lessons learned are included as Appendix B. Several of the more important ones are listed below:

- The ASCLSS program demonstrates a control hierarchy which places maximum control authority at the lowest controller level. This controls hierarchy optimizes process and system autonomy and reduces subsystem interactions during integration. Maximum authority at the lowest controller level also provides an excellent vehicle for demonstrating distributed artificial intelligence/expert systems (AI/ES) which is discussed in Section 8.
- In the ASCLSS approach, the automation and control authority, philosophy, and implementation remains the responsibility of the subsystem developer. "The application expert is accountable."
- The program demonstrated the effectiveness in using modern telecommunications in software development and transmission, data exchange, documentation control, and software configuration management. This is an early "proof of concept" of the Space Station Technical, Management, and Information System (TMIS).
- The emphasis on standards such as: MIL-STD-1750A, MIL-STD-1553B, RS232, Pascal, and standard development systems such as IBM PCs and VAX mainframes, fostered portability, and use of readily available development tools for both hardware and software.
- The benefits of a common family of controller cards was demonstrated during system development where cards, and even whole controllers, were swapped to isolate hardware problems and to maintain development schedules. Also, one design fix applies to all common elements. An added benefit of a common card family is that one set of cards currently maintained at Honeywell serves as spares for nine ASCLSS controller units in the field.
- The user-friendly MMI effectively supports an unknowledgeable user or operator. The air revitalization group system and individual ARG processes are easily monitored for operational status or warnings/alarms. The operation of the ASCLSS system is automatic and places the user in full supervisory control.
- The subsystem simulators meet the testing requirements for real-time control applications. The simulators can be programmed for failures, alarms, and rare system configurations which will exercise automation and control system logic with no risk to the process hardware or when early in the program development cycle when process hardware may not be available. The simulators will also be effective in test and validation of future development of embedded AI/ES functions.

- The software architecture features a distributed data base which provides for growth in system capability and accommodates future implementation of embedded AI/ES functions.
- The ASCLSS layered software approach allows operational software to be application independent and the application software to be controller and microprocessor independent (generic applicability and high portability). The layered approach facilitated the parallel development and independent software testing and resulted in straightforward and efficient software and hardware integration.

Many of the important automation features pioneered and demonstrated in the ASCLSS program are providing direct payoff and benefit to NASA. Table 7-1 lists several ASCLSS features that are being planned or carried into the Space Station program.

Table 7-1. ASCLSS Program Features

ASCLSS Automation Features	Comparable Space Station Features
• Strong hardware commonality	• DMS common elements implemented across subsystems (NIU, SDP, EDP, MDM, etc.)
• Distributed controls hierarchy	• Similar three tiered control levels, established by OMS, SDP, and MDM
• Layered software with modularity and significant commonality	• Common distributed operating system (DOS) provided by DMS. Seven layered, OSI/ISO software architecture planned in NIU with common Network Operating System (NOS)
• Distributed control with control authority pushed to lowest level	• Not currently planned. Presently all application software resides in SDPs
• Effective use of process or subsystem simulators	• Planned TAVERNS concept will provide this test capability
• Use of telecommunications in program management and software development	• Planned TMIS and SSE will provide these capabilities
• Crew placed in supervisory control of automated systems	• Space Station OMS and MPAC will establish supervisory control role for on-orbit crew

ASCLSS HAS PIONEERED AND DEMONSTRATED MANY SPACE STATION CONCEPTS

Section 8

RECOMMENDATIONS FOR ADDITIONAL AREAS OF INVESTIGATION

The ASCLSS program represents a significant first step in defining, developing, and demonstrating an automation and controls approach for the Space Station. However, there are many additional studies, development tasks, and technology demonstrations that can be performed that will expand the ASCLSS lessons learned and exploit the existence of the ASCLSS system as an early automation and control test bed. Some of these areas recommended for additional investigation are listed below:

1. Develop an input/output I/O interface for one or more of the ASCLSS basic controllers, integrate the ARG process hardware to the ASCLSS system, and control the process with the ASCLSS system. This will complete the proof of concept to use real-time simulators in place of process hardware to develop and test automation and control systems.
2. Evaluate and test what level of fidelity is required for a real-time simulator to effectively represent process hardware.
3. Develop and demonstrate ARG system fault detection, isolation and recovery and system redundancy management.
4. Enhance the MMI by incorporating multiple display screens, adding windowing techniques, and incorporating animated graphics and touch screens.
5. Port the ASCLSS software system from Pascal to the Space Station HOL, ADA.
6. Develop and demonstrate a policy and approach to on-orbit maintenance repair and replacement of system hardware and software. Determine the viability of repair or replacement at the card level for on-orbit operations. Also, how do you change or repair software on-orbit?
7. Host different Space Station applications (GN&C, thermal, power, etc.) in the ASCLSS system. Consider hosting multiple and different applications in the ASCLSS system so as to emulate a shared SDP.
8. And finally, develop and demonstrate the inter-relationship and an approach to incorporating artificial intelligence and expert (AI/ES) systems embedded in a real-time controller.

These investigations are relatively straightforward by using the ASCLSS system as an existing automation and control test bed.

Section 9

CONCLUSIONS

The Automated Subsystem Control for Life Support System (ASCLSS) program has successfully developed and demonstrated a generic approach to the automation and control of Space Station subsystems. The automation system features a hierarchical and distributed real-time control architecture which places maximum controls authority at the lowest or process control level which enhances system autonomy. The ASCLSS demonstration system pioneered many automation and control concepts currently being considered in the Space Station data management system (DMS):

- Heavy emphasis is placed on controls hardware and software commonality implemented in accepted standards.
- An innovative layered software architecture was developed which accommodated independent and parallel software development and relatively straightforward hardware and software integration.
- The automation system architecture retains the controls responsibility and accountability with the subsystem or process developer.
- The approach demonstrates successfully the application of real-time process or subsystem simulators to support development and validation of automation and control hardware and software. This is an early "proof of concept" of the NASA TAVERNS (Test and Verification Environment of Networked Systems) V&V approach.
- The ASCLSS system completely automates a Space Station subsystem (air revitalization group of the ECLSS) which moves the crew/operator into a role of supervisory control authority.

The ASCLSS program developed over 50 lessons learned which will aid future Space Station developers in the area of automation and controls. And finally, the ASCLSS demonstration system represents an early test bed which will support additional technology studies and development activities including evaluation of controls requirements, application of ADA software to real-time control, and development and demonstration of embedded artificial intelligence and expert (AI/ES) systems in a distributed automation and controls system.

Appendix A

ASCLSS PROGRAM LIST OF REFERENCES

1. "Automated Subsystems Control for Life Support Systems (ASCLSS), Applications Study Report, Volume I"; Honeywell Systems and Research Center, Minneapolis, MN; January, 1984.

2. "Automated Subsystems Control for Life Support Systems (ASCLSS), Applications Phase Final Report, Volume II"; (Honeywell Proprietary), Honeywell Systems and Research Center, Minneapolis, MN; January, 1984.

3. "Automated Subsystems Control for Life Support Systems (ASCLSS), Applications Phase Final Report, Volume III, Electrical Power System"; (Honeywell Proprietary), Honeywell Systems and Research Center, Minneapolis MN; January, 1984.

4. "ASCLSS Primary Deign Review Report", (Vugraph Handouts), Honeywell Space and Strategic Avionics Division, Clearwater, FL; December 11-12, 1984.

5. "ASCLSS Final Design Review Report"; (Vugraph Handouts), Honeywell Space and Strategic Avionics Division, Clearwater, FL; April 3-4, 1985.

6. "Functional Requirements Document ACSLSS Demonstration Program"; Honeywell Space and Strategic Avionics Division, Clearwater, FL; October 1, 1984.

7. "ASCLSS System Design Specification", DSYG9152A1; Honeywell Space and Strategic Avionics Division, Clearwater, FL; April 30, 1985.
8. "ASCLSS Software Specification", CPYG9152A1; Honeywell Space and Strategic Avionics Division, Clearwater, FL.

9. "ASCLSS Pascal Software Development Guidelines", DS34062728; Honeywell Space and Strategic Avionics Division, Clearwater, FL; May 2, 1986.

10. "ASCLSS Software Development Plan", DS34062727; Honeywell Space and Strategic Avionics Division, Clearwater, FL; May 29, 1985.

11. "ASCLSS I/O Data Base Control Document", DS34062726; Honeywell Space and Strategic Avionics Division, Clearwater, FL; April 23, 1986.

12. "Interface Control Document for ASCLSS", ICD34062725; Honeywell Space and Strategic Avionics Division, Clearwater, FL; August 1, 1986.
13. "ASCLSS Operating System Design Specification"; Honeywell Space and Strategic Avionics Division, Clearwater, FL; May 22, 1987.

14. "ASCLSS Man-Machine Interface Functional Design Document"; Honeywell Systems and Research Center, Minneapolis, MN; April 1, 1986.

15. "ASCLSS Man-Machine Interface Software Description Document"; Honeywell Systems and Research Center, Minneapolis, MN; April 7, 1986.

16. "ASCLSS System Users Manual", DS34062774; Honeywell Space and Strategic Avionics Division, Clearwater, FL; January 19, 1987.

17. "ASCLSS Systems Acceptance Test Procedure"; Honeywell Space and Strategic Avionics Division, Clearwater, FL; July 10, 1986.

18. "Development and Demonstration of All Automated Subsystems Control for Life Support Systems, Final Report"; Life Systems, Inc., Cleveland, OH; May, 1986.

19. "Design Handbook for Static Feed Electrolyzer", TR-569-47A; Life Systems, Inc., Cleveland, OH; March, 1984.

20. "Design Handbook for Sabatier CO2 Reduction Subsystem", TR-569-46; Life Systems, Inc., Cleveland, OH; March, 1984.

21. "Design Handbook for Electrochemical CO2 Concentrator", TR-569-48; Life Systems, Inc., Cleveland, OH; March, 1984.

22. "Basic Controller Application Software Data Dictionary", TR-569-59A; Life Systems, Inc., Cleveland, OH; January, 1986.

23. "ASCLSS Basic Controller Application Software System Description", TR-569-66; Life Systems, Inc., Cleveland, OH; January, 1986.

24. "ASCLSS System Controller Application Software Data Dictionary", TR-569-61A; Life Systems, Inc., Cleveland, OH; January, 1986.

25. "ASCLSS System Controller Application Software System Description", TR-569-65; Life Systems, Inc., Cleveland, OH; January, 1986.

26. "System Control Application Software Transition Requirements", TR-569-58B; Life Systems, Inc., Cleveland, OH; August, 1985.

27. "ARG Process Simulator Data Dictionary", TR-569-60; Life Systems, Inc., Cleveland, OH; January, 1986.

28. "ASCLSS Simulator Software System Descrition", TR-569-67; Life Systems, Inc., Cleveland, OH; January, 1986.

29. "ARG Process Simulator Functional Description", TR-569-57A; Life Systems, Inc., Cleveland, OH; January, 1986.

30. "Master Simulator Test Plan", TR-569-13A; Life Systems, Inc., Cleveland, OH; January, 1986.

31. "User's Guide and Test Procedure", TR-569-68; Life Systems, Inc., Cleveland, OH; March, 1986.

Appendix B

LESSONS LEARNED/DECISIONS MADE

ASCLSS PROGRAM ROADMAP
LESSONS LEARNED/DECISIONS MADE

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
1. Automation system micro-processor selection.	o Baselined 1750A processor - F9450 CPU chip - ADOP bit slice multi-chip as backup.	o SOW requests S.D.A. micro-processor (1750A is forward technology). o MIL-STD 1750A leads into Space Station. o Application study recommended the 1750A.	o Decision added cost and risk to program - but they were manageable. o Many tools and large user base available since 1750A is a DoD standard.
2. Automation system HOL selection for application software.	o Baselined Pascal; essentially a close relative to ADA.	o Appl. study recommended ADA - ADA tools and support were unavailable and immature. o Pascal SW tools & language well known, available, and offer easy migration to ADA.	o Pascal compilers and tools to 1750A exist on HI VAX. o ADA has been baselined for Space Station.
3. Individual simulators for each ARB process or a single simulator for 3 ARB processes.	o Baselined individual simulator and process simulation for <u>each</u> ARB process.	o Easier, less complex simulation development process. o Clearly defined, easy to demonstrate simulation/simulators.	o Enhanced demonstration potential. o Allows for growth and development.
4. Technical performance of MMI display monitor.	o Selected high quality, with medium resolution RGB color monitor for MMI display graphics (IBM XT).	o Represents the major interface for the crew control authority and operations interface. o Required for more impressive demonstration.	o Did not select ultra-high resolution monitor due to high cost of monitor and supporting computer.
5. Automation system communication bus.	o Selected a dual redundant MIL-STD-1553B bus approach, with command/response protocol.	o Similar to high-speed command/response bus planned for use on Space Station. Establishes desired controller hierarchy. o Military standard bus, capable of high speed (1 MBPS).	o Bus structure and controller hierarchy similar to anticipated Space Station bus and control hierarchy. o Communications on Bus is mainly data and status flow, not commands. Ethernet bus may be more appropriate.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
6. Generic signal conditioner I/O cards (not implemented but designed for).	o The ability to implement automation with generic cards allows a flexible, building block approach to controller design. Interfaces and hooks provided for future I/O implementation and spare card slots are provided for.	o Generic, common hardware will be essential to meet Space Station design-to-cost and maintenance, repair, and sparing requirements.	o Generic I/O will be a Space Station requirement. Not address in baseline ASCLSS program.
7. Memory architecture and capacity for the basic control computer.	o Defined memory 40K words RAM (data memory) & 64K words E2PROM (instruct. memory). Addition of 8K words can be accommodated.	o 104K words memory should be more than adequate for a non-redundant controller.	o Additional memory is just an implementation issue. o Using 1750A DI line can add 24K words of RAM easily.
8. Bus interfaces between controllers and the MMI and ARG simulators.	o Selected high-speed (230KB) RS232-RS422 digital serial bus. Differential drivers at bus terminations also provided.	o Need a high-speed bus that transfers simulator data within time frame of signal conditioning hardware and controller minor cycle (230KB is acceptable).	o Differential drivers allow separation of hardware by large distances. o Achieves a real-time simulation/simulator.
9. Automation Hierarchy of module, system, and subsystem controllers.	o Selected a hierarchal architecture with maximum control authority at the lowest level.	o This type of architecture is anticipated for the Space Station.	o Basic controllers are highly autonomous not needing comm. with systems controller to continue operations.
10. Automation system protocols.	MMI - bus controller for Syst. contr. - RTU Syst. Contr. - Bus Controller for the Subsyst. Contrs. - RTU Subsyst. Contr. - Bus Controller for the Simul. PC-RTU	o Again to reflect a hierarchal architecture.	o Space Station control protocol may be different, but this is a straightforward, conservative command-response structure.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
11. ASCLSS operating systems.	MMI - PC DOS (IBM XT) BCC - ASOS (Honeywell) Simulators - RTOS (IBM PC)	o A simple OS is adequate for MMI because it is not cyclic in real-time. o ASOS developed for this application is adequate, and exists. o A real-time OS is desirable for the simulation because it is cyclic real-time.	o Space Station will require full multi-tasking, distributed control and priority tasking/ rescheduling.
12. Automation system minor cycle time interval.	o Computer system hardware basic cyclic interrupt is 50 Msec. o The application minor cycle is 100 Msec.	o Want to experience running application at slower rate than OS minor cycle.	o Some Space Station subsystems may require shorter minor cycles (~25 msec). o Hardware minor cycle is selectable in increments of 25 Msec.
13. System bus data flow architecture.	o Periodic window - for comm. every cycle. o Aperiodic window - for most communications.	o Easy to analyze. o Solves conflict of cyclic time critical data and non-time critical bulk data.	o Periodic data - no retries. o Aperiodic data transfer-three retries.
14. Generic controller implemented in every control location.	o All system level and subsystem controllers have exactly the same controller HW.	o Demonstrate commonality of controller hardware, an important Space Station issue.	o Commonality of HW/SW is a high priority Space Station requirement.
15. Selection of PC for MMI and AR6 simulator.	o Selected IBM XT, for MMI. o Selected IBM PC, for simulator.	o Many home computers would have been adequate. o IBM PC has largest selection on off-the-shelf HW/SW. o LSI has IBM PC experience and software development tools.	o MMI and workstations on Space Station will be more advanced technology.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
16. Automation system software architecture and development approach.	o Layered software approach. o Separates HOL/application SW from real-time automation system software	o Demonstrate generic use of automation system software. o Allows use of different SW development environments in 3 remote locations.	o Application SW is decoupled from automation system software resulting in the system software being generic across multiple Space Station subsystems.
17. Local Area Network (LAN) ties 3 ARS simulators/simulations together.	o Maintains a consistent and identical set of environment conditions for all 3 simulators.	o External environment data calculated and maintained by one simulator and data provided to other simulators on LAN.	
18. Development of MMI SW, ARS simulator SW, ARS application SW, and automation SW in 3 remote development environments.	o An I/O data base controls SW interfaces and data exchanges. All major packages communicate from I/O data base structures.	o Allows separated SW developments on 3 different companies own software development systems.	o I/O data base successfully demonstrated that approach is possible and achievable.
19. Establish telecommunication links between SSAvD, LSI, SRC, and NASA JSC. This greatly aids the development process taking place in 3 different remote locations (SSAvD, SRC, LSI).	o Established PC-to-PC links between SSAvD/LSI/SRC. Also tied SSAvD PC to NASA VAX.	o These links greatly increase the efficiency of program communications, actions, and technical data evaluations.	o Results have been very successful. o Weekly status reports are exchanged. o Action registers are maintained. o Software files are transferred, compiled, tested, annotated and exchanged. This arrangement greatly helped program during HW/SW integration phase.
20. The initial packaging design for the ASCLSS BCC (17" x 14" x 11") was large to ease build and test C/O but too large to reflect Space Station controller features.	o Redesigned chassis around same cards and card cage. Ordered smaller power supplies. Resulting package is 40% smaller in size and more representative of Space Station controller configurations.	o Small controller will enhance applicability of controller to Space Station test bed efforts.	o The reduced size ASCLSS controller (16.6" x 11.2" x 8.6") is more representative of Space Station controllers.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
21. The system controller maintains time of day for all controllers (and MMI and simulators) and maintains all basic controllers synchronized.	o Real-time control requires a master clock and all controllers synchronized and data time stamped with time of day.	o System controller performs master clocking and broadcasts time of day and achieves synchronization of all controllers.	o DMS will provide station clock function on Space Station.
22. ASCLSS controller throughput driven by 15MHz clock. This should represent a 700K ops throughput for standard mix.	o Developed HW/SW using 15MHz clock. If problems occur at this speed, insert new clock to run system at 12MHz.	o No problems occurred at 15MHz, however, provisions are in place for system speed reduction (lowers power).	o Some subsystems and applications will require greater throughput.
23. ASCLSS demonstrator system initialization process allows all elements to initialize and assume a safe startup in any order.	o Desirable to allow all units (MMI, controllers, simulators) to startup and initialize in independent fashion.	o Initialization/startup gets entire system in safe state. o Proceed to operation requires controller communications.	
24. ARG simulators designed to operate stand alone as well as integrated to ASCLSS controller system.	o Needed to allow simulator/ simulation development and checkout at LSI and before ASCLSS HW/SW integration.	o Needed for successful ARG simulator development.	o Effective for training.
25. Define program functional responsibilities based on end product and clear interfaces.	o Honeywell SSAvD - controls HW and generic SW. o LSI - application SW and subsys. simulators. o Honeywell SRC - MMI HW/SW.	o Responsibility assigned based on company business objectives and technical strengths. Clear definable interfaces established by specifications and control documents.	o Established a clear working arrangement allowing independent development but dependent successful integration (determined "who").
26. Define tasks to be done in each functional/responsible area. Also, group tasks in definable, understandable groupings (i.e. process management operating systems, etc.).	o Tasks were defined to fit program cost/schedule parameters. Tasks were prioritized and some lower priority efforts were dropped.	o Highest prioritized tasks were always maintained in the development program.	o Some tasks such as hardware integration and MMI trend reporting had to be eliminated or downscoped. (Determined "what".)

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
27. Define functions and tasks and share with team to achieve integrated approach.	o Team agreed on functions and tasks in all areas.	o Established full team understanding.	o (Determined "where").
28. Define common understandable nomenclature and definitions to improve team understanding.	o Developed dictionary and definitions early and stayed consistent.	o Aids understanding by all parties and for outsiders.	
29. NASA must participate and be decisive in direction, wants, and guidance.	o NASA project monitor included in daily/weekly telecons, communications, and decisions as a full "team member."	o By getting NASA into a team relationship they became more decisive and understanding.	o Vital in any development program.
30. Address controller signal conditioning early.	o The approach to signal conditioning is vital to any controller approach.	o Key tradeoffs exist in the signal conditioning area for any controller application.	o This activity is currently planned as part of follow-on program.
31. Subsystem developer/application expert designs the application software.	o ECLSS expert (LSI) developed application software.	o Most qualified to do the application controls "accountability".	o Achieved through the unique features of layered software.
32. Use of simulators to test/verify automation and controls philosophy and design.	o Individual process simulators were effective in test of A&C.	o Simulators are baselined now for Space Station (TAVERNS) as I, T, V&V critical elements.	
33. Importance of MMI from users/operators perception.	o MMI must be well designed and effective.	o The "system" becomes what's on the MMI to the operator.	o Critical to user understanding and confidence in system operational performance.
34. Commonality of hardware.	o Full commonality of controller hardware.	o One set of cards functions as spares for nine units in the field.	

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
35. Restrictions of the selected Pascal compiler used in the Basic Controller and System Controller.	o The restrictions have caused a rethink and rewrite of areas of both the BC and SC applications software. Some of the routines, initialization in particular, have been coded in assembly language to circumvent the language restrictions. o In other areas, the compiler has been found to generate inefficient code again resulting in adjustments in coding techniques to avoid these inefficiencies.	o The selected Pascal compiler was the only one available for the 1750A processor.	o Some of these adjustments may result in awkward constructions and increasingly difficult maintenance problems.
36. Restrictions of the communications system used between the Basic Controller and System Controller.	o Data organizations have been manipulated to conform to the requirements of the 1553B communications system.	o The selected communications bus (1553) has a limit on the size of messages that can be sent on the bus.	o The communications system should have been designed to packetize the desired data to fit the requirements of the selected hardware applications without requiring the software to conform to communications restrictions. The packets should have been formed and unpacked in a manner that was completely transparent to the application software. The current system implies that if different hardware communications system is implemented, then the software might again have to reorganize its data structures to accommodate it. This also causes excessive software development and integration for the application software developer.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
38. Restrictions imposed by the compilers being used to develop the simulators.	o While the majority of the simulation ware is being developed in Pascal, both PLM and assembly language routines are also being written when necessary to provide either desired functions or desired speed.	o Some specialized functions must be provided for dealing with the specific hardware of the application and some routines must execute within a minimum of time. In these cases a more flexible or more efficient language is chosen. Keeping these exceptions to a minimum maintains the maximum portability and error checking capacity of the Pascal language.	o Pascal was chosen primarily because it has the best compiler error checking and correctness verification of the available compilers. The type checking inherent in the language can be expected to dramatically reduce the debugging time and many potential runtime errors. Pascal is also very close in style and constructs to Ada and the simulators could be expected to be ported to ADA at a later time with less effort. It should also be noted that the primary simulation equations are actually being developed using a different Pascal compiler than the one they will eventually be compiled against. The portability of the language is a great advantage in this regard.
39. Requirements of simulator cycle time for real-time simulation.	o Execution time for the simulation equations are continually monitored during development to ensure that they will preform in the time frame (100ms minor cycle) necessary for real-time simulation.	o It will be necessary to optimize the simulation equations to maintain the real-time operation. By monitoring this as development proceeds it was easier to pinpoint the areas of optimization in the simulation that produced the most gain for the least effort.	o The equations continue to be reduced in complexity and, as a result shorter, execution time, without seriously impacting the fidelity of the simulation.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
40. Use of telecommunications link for information interchange.	o A number of questions and documentation issues were resolved using telecommunications links between Life Systems and SSAvD.	o The electronic transfer was felt to be quicker and easier than surface mail and cheaper than Express Mail.	o Use of the electronic transfer exclusively at times caused added delays due to the fact that the exchange took place at fixed intervals. There was not enough equipment available to support the telecommunications continuously and software development equipment had to be diverted to this use periodically. Also, some questions would have been better handled in a personal conversation where additional questions could have been asked and an issue settled without additional electronic exchanges. Electronic transfer should be reserved for special instances unless the facilities to support the telecommunications system can be dedicated.
41. Elimination of data-dependency from operating system in Basic Controller and System Controller.	o The operating system in the BC and SC was designed to be independent of the data being exchanged.	o Data independence would eliminate any need to regenerate a new version of the operating system software when a new application is developed.	o The operating system software has not been entirely isolated from the data in the applications environment. The sizes and numbers of data structures in the BC and SC have to be made known to the operating system when the controller is generated. This information should be contained in a single fixed table that the applica-

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
			tion software generates to more completely separate the operating system routines. This one table could then identify the extent and the amount of data to be transferred.
42. Selection of immature software packages and lack of complete control over them in development of simulators.	o Abnormal operation of RTOS in certain areas of the operating system and its associated compilers are being circumvented.	o The selection of a real-time operating system (RTOS) and development environment was extremely limited.	o The operating system still contains some "rough edges" and the compilers required some additional care and impose restrictions not originally foreseen. All of these are manageable, although annoying. Not having access to the source for the operating system in order to make corrections, means that greater care must be taken in selecting a system. Both the company providing it and the operating system itself should be evaluated.
43. Selection of immature hardware packages in development of simulators.	o Questionable operation in certan areas of the simulator hardware are being avoided.	o The hardware was selected for its availability and advertised fitness for purpose.	o Some of the hardware is being used in ways not yet thoroughly tested by the manufacturer. The lack of available information from the manufacturer results in longer development times when interfaces are being developed. Any problems encountered must be resolved without factory assistance.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
44. Use of a real-time operating system for simulation.	o The real-time operating system "RTOS" was selected for use in the ASCLSS ARG simulators.	o The simulation is intended to be a real-time simulation.	o The only requirement for a real-time operating system comes from the Basic Controller communications system which insists on a response within a limited time frame. Otherwise, the simulation does not strictly require a real-time operating system to provide real-time simulation. A standard operating system might have been adequate. However, it should be noted that the selection of "RTOS" as the operating system also required the use of the associated Intel compilers which have proven to be significantly faster and more efficient than the best compilers for the standard IBM-PC system.
45. Presentation of simulator data and operator command input.	o The operator interface to the simulator was reworked to provide a cleaner and more efficient system.	o The most effective presentation of the simulation is that of a graphic display depicting the process.	o The operator interface was re-worked to provide the process schematic on the screen as much as possible. Instead of menus taking over the entire screen, "windowing" was employed without disturbing the process display. As much as possible, the process schematic was the primary display.

PROBLEM/ISSUE	ACTION/DECISION	RATIONALE	RESULTS/COMMENTS
46. Standard Pascal compilers consume excessive time consumption during compiling, linking and debug during software development.	o Selected high-speed Pascal compiler that sufficiently reduces the line of unpiling, linking, and debug.	o Must be configured for IBM PC and support standard Pascal.	o Selected TURBO Pascal.
47. Software development strategy at Life Systems.	o Implementation of Data Flow Diagrams, Data Dictionary and structure charts.	o Reduced overall cost of engineering and coding software.	o In addition to reducing developmental cost, this method provided good means of communicating software concepts.
48. Simulation process equation development.	o To define a standard methodology to determine interrelationship between process variable for any given simulation.	o Must define all sensor parameters in terms of actuator values and time.	o Developd sensor/actuator matrix to determine relationships (Direct, Indirect, No Relationship). Created a structure chart for each sensor showing all system parameters that will affect its value.
49. Different data structure names used by application and system control software.	o Require use of data structure generic names in application code.	o Code easier to understand, debug, and more efficient.	o More efficient, effective software code.
50. At times, telecommuniatons were prohibited between facilities which delayed schedule.	o Provide dedicated telecommunications (PC's) resources at each facility.	o Accommodate continuous availability of telecommunications for software, data and technical exchanges.	o Program schedule not impacted.