

✓ JOHNSON SPACE CENTER

IN-14-CR

239257
530.

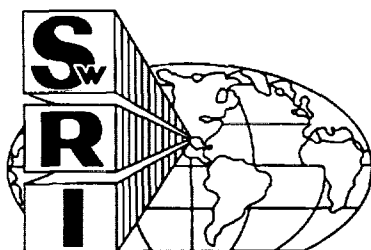
ADVANCED MANNED SPACE FLIGHT SIMULATION AND TRAINING

AN INVESTIGATION OF SIMULATION HOST COMPUTER SYSTEM CONCEPTS

Grant No. NAG9-394
SwRI Project 05-3050

Prepared for:
Simulator/Training Systems Development
National Aeronautics and Space Administration
Johnson Space Center
Houston, Texas

10 November 1989



SOUTHWEST RESEARCH INSTITUTE
SAN ANTONIO **HOUSTON**

(NASA-CR-185998) ADVANCED MANNED SPACE
FLIGHT SIMULATION AND TRAINING: AN
INVESTIGATION OF SIMULATION HOST COMPUTER
SYSTEM CONCEPTS Final Technical Report
(Southwest Research Inst.) 53 p

N90-10911

Unclas
0239257

CSCL 14B G3/14

**Southwest Research Institute
Post Office Drawer 28510, 6220 Culebra Road
San Antonio, Texas 78228-0510**

ADVANCED MANNED SPACE FLIGHT SIMULATION AND TRAINING

AN INVESTIGATION OF SIMULATION HOST COMPUTER SYSTEM CONCEPTS

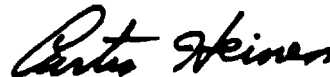
**Grant No. NAG9-394
SwRI Project 05-3050**

Prepared by:
**Bruce C. Montag
Alfred M. Bishop
Joe B. Redfield**

Prepared for:
**Simulator/Training Systems Development
National Aeronautics and Space Administration
Johnson Space Center
Houston, Texas**

10 November 1989

Approved:



**Curtis Heinen, Director
Training Systems and Simulators Department**

Executive Summary

Manned space flight crew training is one of the most challenging and computer intensive applications for real-time simulation technology. The computing power required to create a convincing artificial reality sufficient for mission training and flight software verification has traditionally pushed the state of the art in real-time computing, often exceeding the capabilities of commercially available computers. In some cases, custom one-of-a-kind simulation solutions have been utilized for space flight training to overcome the limitations of commercially available computing systems. The state-of-the-art has now advanced to the point that many powerful simulation technologies are available for solving the complex requirements of space flight simulation.

is presented. It is
This report presents the findings of a preliminary investigation by Southwest Research Institute (SwRI) in simulation host computer concepts and is designed to aid NASA in evaluating simulation technologies for use in spaceflight training. The focus of the investigation is on the next generation of space simulation systems that will be utilized in training personnel for Space Station Freedom operations.

The planned Space Station Training Facility (SSTF) promises to offer many significant, computer intensive simulation challenges that must be solved through the use of efficient, effective, and enduring simulation solutions. It is no longer necessary nor desirable to accept expensive, unique solutions to space flight simulation problems.

Space Station Training Facility (SSTF)
SwRI concludes that NASA should pursue a distributed simulation host computer system architecture for the SSTF rather than a centralized mainframe based arrangement. A distributed system offers many advantages and is seen by SwRI as the only architecture that will allow NASA to achieve established functional goals and operational objectives over the life of the Space Station Freedom program.

Several distributed, parallel computing systems are available today that offer real-time capabilities for time critical, man-in-the-loop simulation. These systems are flexible in terms of connectivity and configurability, and are easily scaled to meet increasing demands for more computing power. New technologies for tightly coupling parallel processors allow distributed architectures to provide main frame class computing power without any of the problems associated with developing, maintaining, and upgrading main frames for real-time applications.

New trends in the development of real-time Ada applications also suggest that industry standards and practices are emerging that promise to eliminate the need for custom, nonstandard Ada implementations in real-time environments. These trends should be researched in further detail, so that the benefits of new real-time Ada developments may be best exploited to achieve the established SSTF functional goals and objectives.

In addition, it is recommended that further research be performed in investigating SSTF simulation challenges and the best application of enabling simulation technologies. Identification and planning for these challenges will aid in the process of translating the SSTF concept into an enduring training system design. The following is a list of candidate areas that SwRI recommends for further research:

- Automation & Robotics Simulation
 - Investigation of Robotic Sensor Simulation Issues
 - Investigation of Embedded Training Issues
- Robotics/Visual System Requirements
 - Investigation of CAD/Visual Scene Database Correlation Issues
 - Investigation of Visual and Tactile Correlation Issues
- Embedded Avionics
 - Investigation of Avionics Simulation Requirements
 - Investigation of Simulate vs. Stimulate Issues
 - Investigation of Knowledge Based System Simulation Issues
- Crew Vehicle Interface
 - Front-End Analysis of Crew Training Needs
- SSTF Concept Development
 - Investigation of Real-Time Ada Directions
 - Identification of Real-Time CASE Requirements
 - Investigation of Real-Time Operating Systems
 - Investigation of Processor Communication Issues

Focused research addressing technical issues in each of these areas is needed to identify, plan for, and reduce risks associated with SSTF concept development. Early warning of space station simulation challenges and identification of candidate technical solutions will better enable future manned space flight simulation facilities to become economical, maintainable, supportable, and instructionally effective training systems.

Table of Contents

1.0	Introduction	1
1.1	Document Background	1
1.2	Document Purpose	1
1.3	Document Overview	1
2.0	SSTF Concept Definition	3
2.1	Integral DMS Role	3
2.2	S/W Standardization	3
2.3	Single Family of Computers	3
2.4	SSE Engineering Model Incorporation	4
2.5	Training Modes	4
2.6	Computer Resource Requirements	4
3.0	SSTF Concept Rationale	5
3.1	Decrease Reliance on Custom Simulation Solutions	5
3.2	Provide Realistic System Responses for Mission Rehearsal	5
3.3	Provide Environment for Training Freedom Station Ground/Space Team	6
4.0	SSTF Systems Simulation Challenges for Concept Development	7
4.1	Concurrent/Independent Crew Training	7
4.2	Crew Vehicle Interface Challenges	7
4.3	Embedded Space Systems Simulation Challenges	8
4.3.1	Embedded Avionics Systems Architecture	8
4.3.2	Embedded Knowledge Based Systems/Autonomous Systems	11
4.3.3	Robotics Technology/Telerobotics	11
4.3.3.1	Predictive Display & Control	13
4.3.3.2	CAD/CAE Correlated Sensory Perception	13
5.0	Enabling Simulation Technologies	15
5.1	Conceptual Technologies	15
5.1.1	Modular Simulation Concepts	15
5.1.2	Distributed Processing for Modular Simulation	16
5.1.3	Integrated Environmental Modeling	18
5.2	Architectural Technologies	18
5.2.1	Loosely Coupled Networking	22
5.2.1.1	Ethernet	23
5.2.1.2	Token Ring	24
5.2.1.3	VME Bus and Multibus II	24
5.2.1.4	Real-Time Networks	25
5.2.2	Tightly Coupled Memory Linkage	25
5.2.2.1	Broadcast Memory	26
5.2.2.2	Shared Memory	26
5.2.2.3	Crossbar Memory	26
5.2.3	Processor Selection	30
5.2.3.1	Processor Architectures	30
5.2.3.2	Data Flow Machines	30
5.2.3.3	Parallel Processing	31
5.2.4	Reliability and Maintainability	31
5.3	Software Technologies	32
5.3.1	Real-time Ada	32

5.3.2	Operating System Functions	33
5.3.3	Ada Programming Support Environments	33
5.3.3.1	Simulation Oriented CASE	33
5.3.3.2	Real-time Debugging Tools	34
5.4	Processing Technologies	34
5.4.1	Symbolic Computing	34
5.4.2	Vision Computing	35
5.5	Visual System Technologies	35
5.5.1	Digital Image Generation	35
5.5.2	Digital Image Processing	36
6.0	Concept Objectives and Technology Assessment	38
6.1	SSTF Computing Objectives	38
6.2	Architecture Assessment	39
Appendix A References		A1

List of Figures

Figure 4-1	Aerospace Avionics Architecture	9
Figure 4-2	GN&C/DMS Architecture	10
Figure 4-3	Robotics/DMS Integration	12
Figure 5-1	Space Station Modular Simulation Concept	17
Figure 5-2	Simulation Computer System	19
Figure 5-3	Integrated Environmental Modeling Concept	20
Figure 5-4	Operational Computer System Concept	21
Figure 5-5	Typical LAN Topologies	23
Figure 5-6	FDDI Token Ring	24
Figure 5-7	Typical Broadcast Memory Configuration	27
Figure 5-8	Typical Shared Memory Configuration	28
Figure 5-9	Typical Crossbar Memory Configuration	29
Figure 5-10	Robotics/Visual System Integration	37

List of Abbreviations

AJPO	Ada Joint Program Office	MIPS	Million Instructions Per Second
APSE	Ada Programming Support Environment	MPAC	Multipurpose Applications Console
ANSI	American National Standards Institute	NASA	National Aeronautics and Space Administration
ARTEWG	Ada Real-time Environment Working Group	NASREM	NASA Standard Reference Model for Telerobot Control System Architecture
C&T	Communication and Tracking	NST	Node Systems Trainer
CAD	Computer Aided Design	OADP	Operations Automatic Data Processing
CAE	Computer Aided Engineering	OCC	Operation Computer Complex
CASE	Computer Aided Software Engineering	ORU	Orbital Replacement Unit
COTS	Commercial Off-the-Shelf	OS	Operating System
CPU	Central Processing Unit	SDP	Standard Data Processor
CSMA/CD	Carrier Sense Multiple Access with Collision Avoidance Detection	SIB	Simulation Interface Buffer
CVI	Crew Vehicle Interface	SMTF	Shuttle Mission Training Facility
DIG	Digital Image Generator	SPOT	Space Proximity Operations Trainer
DMS	Data Management System	SSCC	Space Station Control Center
ECLSS	Environmental Control and Life Support System	SSE	Software Support Environment
EPS	Electrical Power System	SSP	Space Station Program
EVA	Extravehicular Activity	SSSC	Space Station Support Center
FDDI	Fiber Distributed Data Interface	SSTF	Space Station Training Facility
FMS	Fluid Management System	STD	Standard
FSW	Flight Software	STE	Student Training Environment
FTS	Flight Telerobotic Servicer	STS	Space Transportation System
GDB	Global Database	SwRI	Southwest Research Institute
GFE	Government Furnished Equipment	TCS	Thermal Control System
GN&C	Guidance Navigation & Control	TDRS	Tracking & Data Relay Satellite
IEEE	Institute of Electrical and Electronic Engineers	TERPROM	Terrain Profile Matching
I/O	Input/Output	TDM	Time Data Multiplexing
JPL	Jet Propulsion Laboratory	VME	VERSAbus-E (Motorola bus interface standard)
JSC	Johnson Space Center	VSDB	Visual Scene Data Base
KB	Knowledge-Base		
KBS	Knowledge-Based System		
LAN	Local Area Network		
Mb/s	Megabit per Second		
MB/s	Megabyte per Second		
MCC	Mission Control Center		
MDM	Multiplexer/Demultiplexer		
MIL	Military		

1.0 Introduction

Manned space flight simulation requirements place a heavy demand on real-time computer resources and have historically exceeded the capabilities provided by commercially available systems. Southwest Research Institute (SwRI), in investigating the technical issues associated with host simulation computer systems applicable for manned space flight simulation, has elected to focus this research on the needs of Space Station Freedom oriented simulation.

1.1 Document Background

Future astronauts, scientists, mission specialists, and ground support personnel assigned for Space Station Freedom operations will learn the hands-on specifics of their job tasks within the Space Station Training Facility (SSTF) at the Johnson Space Center (JSC) in Houston, Texas. The SSTF will provide a simulated near real-world environment sufficient for conducting mission rehearsal training for entire flight crews, verifying candidate flight software loads, and instructing personnel in the detailed operation of space systems.

The SSTF is planned to become one of the most sophisticated man-in-the-loop simulation facilities ever developed. Many advanced simulation challenges are to be expected in developing the SSTF concept due to the scope and breadth of the overall training problem and the high technology content of planned embedded space systems. Successful concept implementation will rest in large part on the ability of SSTF planners and decision makers to forecast these challenges, and accommodate them in the SSTF concept as it evolves.

1.2 Document Purpose

The purpose of this document is to address one of the most critical areas in the concept development of the SSTF environment. This area is the host simulation computer resources that will serve as the backbone of the SSTF student training environment. Conceptual planning for the host computer complex must give consideration to potential simulation challenges and technological issues associated with the future direction of real-time simulation computing. Information is provided in this document at a high level and covers a broad scope. This approach provides an overview of potential simulation problems and candidate solutions so that they may be identified and investigated in further detail.

1.3 Document Overview

This document describes a preliminary SwRI investigation of technical issues surrounding the host simulation computer resources and the potential simulation challenges that can be expected in developing the SSTF concept. During the period of investigation, over sixty technical reports and publications have been reviewed, and suppliers of state-of-the-art real-time simulation capable computers were interviewed in order to achieve a firm understanding of the technical issues associated with the definition of the SSTF host simulation computing needs.

The information presented in this document is divided into six sections and one appendix. Section 1 provides the background purpose and overview of the document. Section 2 provides a brief review of key development concepts that lay the foundation for the SSTF design. Section 3 identifies the rationale behind the development concept in terms of the functional goals and objectives that NASA has established for the SSTF. Section 4 discusses potential simulation challenges and how these challenges affect the host computer resources concept. Section 5 is review of state-of-the-art simulation technologies that are candidates for implementation in the SSTF host simulation computer system design. Finally, Section 6 assesses the technical issues and potential effectiveness of these

enabling simulation technologies to meet anticipated simulation challenges and attain the established SSTF goals and objectives. The appendix contains a listing of the reference technical reports and publications used as a basis for this study.

2.0 SSTF Concept Definition

The SSTF will be the primary training facility for instructing crew members, ground support personnel and space station customers in the operation of on-orbit systems. The Simulation and Training Environment (STE) will support real-time, man-in-the loop operation of embedded core space station systems, mission specific payloads, and free-flying space vehicles. Several functional and operational goals concerning life cycle cost, system reliability, configuration supportability, and training utility have been identified for development of the SSTF concept. To facilitate achievement of these goals, several SSTF development ground rules have been established. These ground rules involve the incorporation of Data Management System (DMS) flight software, standardization of simulation software, computer system compatibility (a single family of computers), reuse of available software (engineering models), and flexible training modes.

2.1 Integral DMS Role

The on-board space station operational management system and associated avionics, otherwise known as the DMS, will be stimulated within the SSTF to allow for realistic mission rehearsal and provide a near real-world environment for the verification of flight software. This capability will be provided through the integration of a government furnished equipment (GFE) "DMS kit" within the SSTF simulation environment. The DMS kit consists of a functional shipset of Standard Data Processors (SDPs) that execute space station flight software, and a Simulation Interface Buffer (SIB) that allows the SDPs to be functionally and physically stimulated by a host simulation computer. Flight equivalent rather than flight qualified SDPs will be utilized within the DMS kit. These SDPs will execute the same unmodified flight software that will be utilized in the space station.

2.2 S/W Standardization

The SSTF is one of the many customers of the Software Support Environment (SSE) contractor that provides a common Ada Programming Support Environment (APSE) for all space station related software development. The SSE establishes the tools and rules for software standardization across the Space Station Program (SSP). The SSE will also serve as a software repository for SSP simulation software developed by the C/D Workpackage prime contractors. The SSTF concept calls for heavy use of SSE software to minimize the need for SSTF unique software development. Commercial off-the-shelf (COTS) software will be utilized to the greatest extent possible (i.e. operating systems and support software) to enable the SSTF configuration to be as flexible and as supportable as possible, without reliance upon any single source for maintenance and reconfiguration.

2.3 Single Family of Computers

The SSTF concept also calls for participation where applicable in the Operations Automatic Data Processing (OADP) plan that will establish a single family of computers for mission operations support. The intent of this plan is to establish computer system commonality among the various mission operations elements such as the Shuttle Mission Training Facility (SMTF), Space Station Support Center (SSSC), Mission Control Center (MCC) and the SSTF. The current OADP concept includes minimal considerations for the special real-time simulation needs of the SSTF. In general, the OADP packaged system requirements address real-time capabilities as an "upgrade" option, which does not currently reflect industry practice. As discussed in Section 5.2, real-time simulation computers must be designed from the ground up with much consideration given to bus design, processor integration, and operating system control to assure that real-time simulation performance is deterministic (i.e. predictable, repeatable, and adhering to strict timing requirements), fault tolerant, and easily scalable to accommodate expanding simulation requirements. Data processing systems not originally designed for real-time operation do not typically serve well in the real-time

role due to inherent architectural deficiencies and the need for custom solutions for real-time simulation applications.

2.4 SSE Engineering Model Incorporation

A key element in the SSTF concept is the incorporation of engineering software simulation models for each space station system. Defined systems are: Guidance Navigation and Control (GN&C) system, Electrical Power System (EPS), Thermal Control System (TCS), Environmental Control and Life Support System (ECLSS), Communication and Tracking (C&T) system, Fluid Management System (FMS), the propulsion system, robotic manipulators, payloads, international partner elements, free-flyer systems, and extra vehicular activity (EVA) system. These models, developed by the C/D workpackage prime contractors and maintained by the SSE, will be modified for real-time training simulation usage and will execute within the simulation host computer.

2.5 Training Modes

Several training modes of operation will be supported by the SSTF simulation environment. These modes include up to two simultaneous stand-alone part-task training sessions, combined systems training among two or more crew environments, integrated training that includes the Space Station Control Center (SSCC), and joint-integrated training sessions that include participation by other external-to-JSC simulations.

2.6 Computer Resource Requirements

The space station simulation environment is expected to become one of the most complex man-in-the-loop simulation systems ever developed, surpassing the Space Shuttle Mission Simulator in terms of processing throughput, training scenario complexity, and instructional system demands. The National Aeronautics and Space Administration (NASA) has estimated that the SSTF will require in the neighborhood of 3 million executable lines of code and over 150 million instructions per second (MIPS) of processing power to provide a convincing artificial reality for rehearsing on-orbit operations. These requirements are expected to expand as the station's core system capabilities and customer payloads become more sophisticated and incorporate an increasing level of automation and robotics technologies.

3.0 SSTF Concept Rationale

The SSTF concept rationale is intended as a foundation for the design of a training facility that provides for the achievement of certain functional goals and operational objectives dealing with system reliability, instructional effectiveness, and efficient utilization of facility resources. A three-part strategy is apparent in the establishment of the SSTF ground rules. The first part concerns the huge expense involved in maintaining custom developmental hardware and software. Development, purchase, and maintenance of computer software currently accounts for over 20% of the NASA budget each year. In addition, 10% of the total NASA budget is spent on writing new software. Due to the software intensive nature of the space station, these allocations are expected to skyrocket as development progresses. NASA wishes to control this trend during SSTF development by minimizing the need for custom simulation solutions that may prove expensive to support. The second part of the concept rationale is to ensure that a near real-world simulation environment exists in the SSTF. This will provide mission development and verification activities with a high degree of confidence in the expected on-orbit performance. The third part of the strategy deals with the need to train the entire Freedom Station Mission Support Team as a single unit for highly transferrable and effective training sessions. The following sections discuss each of these three strategies in further detail.

3.1 Decrease Reliance on Custom Simulation Solutions

Space flight simulation during the Apollo, Skylab, and Space Transportation System (STS) eras relied mostly on custom simulation solutions that were necessary to support the demanding training requirements associated with manned space flight operations. Space flight simulation is one of the most difficult types of simulation, requiring huge amounts of computer processing power to adequately support the safety critical aspects of manned space flight. Commercial support of real-time simulation resources did not exist during previous space flight simulation development efforts, and unique solutions were innovated so that the training problem could be effectively solved. Although technically effective, these custom solutions were difficult and expensive to support, maintain, reconfigure, and modify as new mission requirements were identified. In the past five years many changes have occurred in the real-time simulation industry. The size of the industry has nearly tripled and is currently estimated to be worth over \$2.5 billion. As a direct result, a variety of real-time simulation capable computers and associated software are becoming commercially available to meet the demand for simulation and training systems. Industry standards and practices are beginning to emerge that promote the use of open architectures for real-time applications. Custom, one-of-a-kind, proprietary simulation solutions are no longer necessary or desirable. NASA is therefore intending to develop the SSTF with as high a content as possible of commercially available real-time simulation resources. NASA also wishes to minimize the need for developmental software by capitalizing on other SSP software applications that can be effectively reused with minimum modifications in the SSTF. This is a basis for the incorporation of SSE maintained engineering software models of Freedom Station systems.

3.2 Provide Realistic System Responses for Mission Rehearsal

The need to verify flight software and effectively train personnel for mission operations is a basis for the ground rule to incorporate DMS flight equivalent hardware and unmodified flight software. It is expected that by incorporating the actual flight software, mission to mission reconfiguration can be accommodated quickly, and a high level of system fidelity can be assured.

3.3 Provide Environment for Training Freedom Station Ground/Space Team

The SSTF will also be the center of training for all SSP participants. To accommodate the wide variety of training needs required by astronauts, scientists, mission operations specialists, and ground support personnel, the SSTF concept provides for many different modes of training operations. The rationale for these modes is to ensure that the SSTF resources may be flexibly configured to adequately provide for the training needs of all SSP participants.

4.0 SSTF Systems Simulation Challenges for Concept Development

The high degree of technological sophistication envisioned for the Freedom Station promises to present many interesting and challenging systems simulation requirements. These simulation challenges must be identified and planned for during the system requirements phase to minimize the number of potential surprises that may arise during the SSTF development cycle. Three of these technically challenging advanced simulation needs are discussed in the following sections. These SSTF simulation challenges are: provisions for concurrent yet independent SSTF training sessions, contextually rich display and control systems for crew/vehicle interfacing, and the stimulation of multi-layered, embedded avionics systems for highly realistic, on-orbit operations training.

4.1 Concurrent/Independent Crew Training

The initial SSTF concept supported up to five simultaneous training sessions utilizing the Operational Computer Complex (OCC). As a result of the scrub effort, this requirement has been reduced to two simultaneous SSTF training sessions, with provisions for the future accommodation of additional sessions. A training session is considered to be the crew operation of a flight environment training device such as a Node Systems Trainer (NST) or the Station Proximity Operations Trainer (SPOT). During stand-alone operation, each trainer is individually configured to support part-task training objectives. These concurrent yet independent training sessions will allow several crew members to simultaneously interact with and focus on the operation of a single embedded system such as the GN&C, TCS, or EPS in separate mission/trainer specific situations. For example, NST #1 could be configured for instructing crew members in the operation of the GN&C system, while NST #4 could be configured for TCS related operation. The separate crews would then train in the NSTs simultaneously, without the other NST affecting the particular training session. The individual flight environment trainers may also be configured to support combined, integrated, or joint integrated training sessions to provide varying degrees of ground/space team coordinated mission training.

Configuring the OCC in a hardware efficient manner to provide for multiple, simultaneous yet separate training sessions is a significant simulation challenge. The operation of each trainer is required to be completely independent and free of interaction with other training sessions. If one trainer should experience a hardware or software failure, the other training session(s) should not be affected. These requirements lead to a multicomputer configuration, due to the need for physical partitioning in order to meet the operational trainer performance objectives. Specific technical issues associated with stand-alone vs. combined trainer operation include multiple SIB communications, functional allocation of simulation hardware and software, and run-time executive distribution. Several architectural technologies discussed in Section 5 hold promise for meeting the technical challenges posed by the multiple yet independent/concurrent training problem.

4.2 Crew Vehicle Interface Challenges

The sophisticated display and control features planned for the Multi-Purpose Applications Consoles (MPACs) present another simulation challenge for effective crew training. Several types of MPACs (fixed, portable, etc.) will be utilized on board Freedom Station. The display and control capabilities provided by the MPAC are contextually rich, offering provisions for multiple high resolution (High Definition Television type) visual displays featuring color 3-D graphics and pan/scroll/zoom, voice recognition capability, and force sensing/reflecting hand controllers. To aid in robotic operations, the MPAC may also feature aural/tactile cues for signaling critical situational information (i.e. when load limits are exceeded on manipulator joints or when end effector contact is achieved). The MPAC is the crew members' interface into the inner (and outer) workings of the Station. Through use of an MPAC, the crew member will have the power to: control the station's

orbit; solve embedded systems operating problems; monitor and control the flight of co-orbiting vehicles within the station's control zone; and install, orient, operate, and service payloads.

A key training objective is to adequately model the richness of the Freedom Station operating environment within the SSTF to promote the direct transfer of training from ground to orbit. Sophisticated simulation techniques are needed for synthetic image generation and environmental modeling to provide a convincing artificial reality that is effective for transferring lessons learned from the SSTF to on board operations. Successful training of time critical, man-in-the-loop operating scenarios depends upon the ability of the SSTF to accurately cue the crew member. Example scenarios include reboosting the station orbit, maneuvering the station for thermal control, proximity operations control of multiple vehicles, and robotic manipulator operation. Effective flight safety and emergency procedure training in the SSTF will rest upon the training system's ability to replicate the sensory environment and the physical factors associated with critical condition Freedom Station scenarios. The cognitive perception and response patterns developed by crew members in training should be directly applicable to space operations, without the crew having to "relearn" new patterns in space to overcome the deficiencies of their ground based training. The key Crew Vehicle Interface (CVI) simulation challenge therefore, is to adequately define and configure the SSTF simulation systems in terms of training capabilities so that all anticipated training needs may be easily accommodated as the development progresses.

4.3 Embedded Space Systems Simulation Challenges

The Congressionally mandated initiative to integrate automation and robotics technology into the Freedom Station design presents many significant challenges for the implementation of the SSTF concept. Embedded expert systems, autonomously operating systems, adaptive control structures, on line intelligent tutors, and teleoperated robotic systems serve as examples of the type of technologies that are planned for space station incorporation. The SSTF concept definition must plan to accommodate the space station hooks and scars that will allow these technologies to be integrated into on-orbit operation. The real-time computer complex configuration concept should be considered with these technologies in mind to avoid unwanted surprises during SSTF development.

4.3.1 Embedded Avionics Systems Architecture

The DMS architecture provides a multi-layered communication structure for interfacing embedded avionics systems with the core station information system. The DMS architecture is designed to be avionics application independent, so that a variety of systems, sensors, and control effectors may be accommodated as the station development progresses. This is a challenging structure for time critical, man-in-the-loop simulation due to the separation and allocation of functions between DMS layers. Current aerospace avionics architectures are typically single layer configurations, with multiple busses providing the system to system connectivity, as shown in Figure 4-1. Avionic systems are tightly coupled in that they utilize compatible clock cycle speeds (i.e. 50 Hz, 25 Hz, etc.) and use extensive time tagging of data items to ensure functional synchronization between the individual avionics tasks. In current simulator architectures, avionic system elements that are centralized and perform CVI critical functions are typically stimulated within the simulator training device. Other avionic system elements are functionally simulated at the level of resolution necessary to feed the stimulated elements with realistic flight data. Theoretically, this approach provides for realistic mission oriented training, and also reduces new software development requirements for providing mission specific trainer configurations.

Figure 4-2 shows an example of the multilayered DMS architecture applied to the GN&C system. The upper layer consists of the supervisory and management functions that are allocated to the SDPs. The bottom layer consists of the distributed subsystems and sensors that make up the GN&C avionics suite. The bottom layer also feeds sensed and conditioned data to the upper layer,

AVIONIC ARCHITECTURE

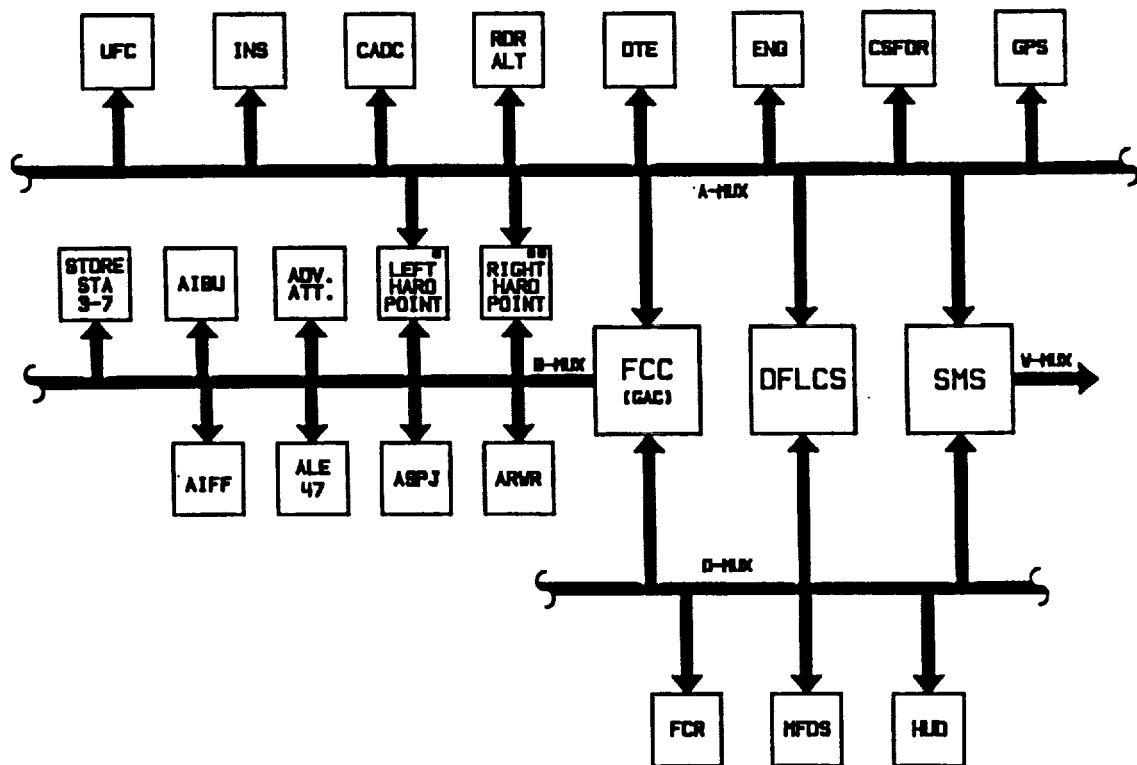


Figure 4-1
Aerospace Avionics Architecture

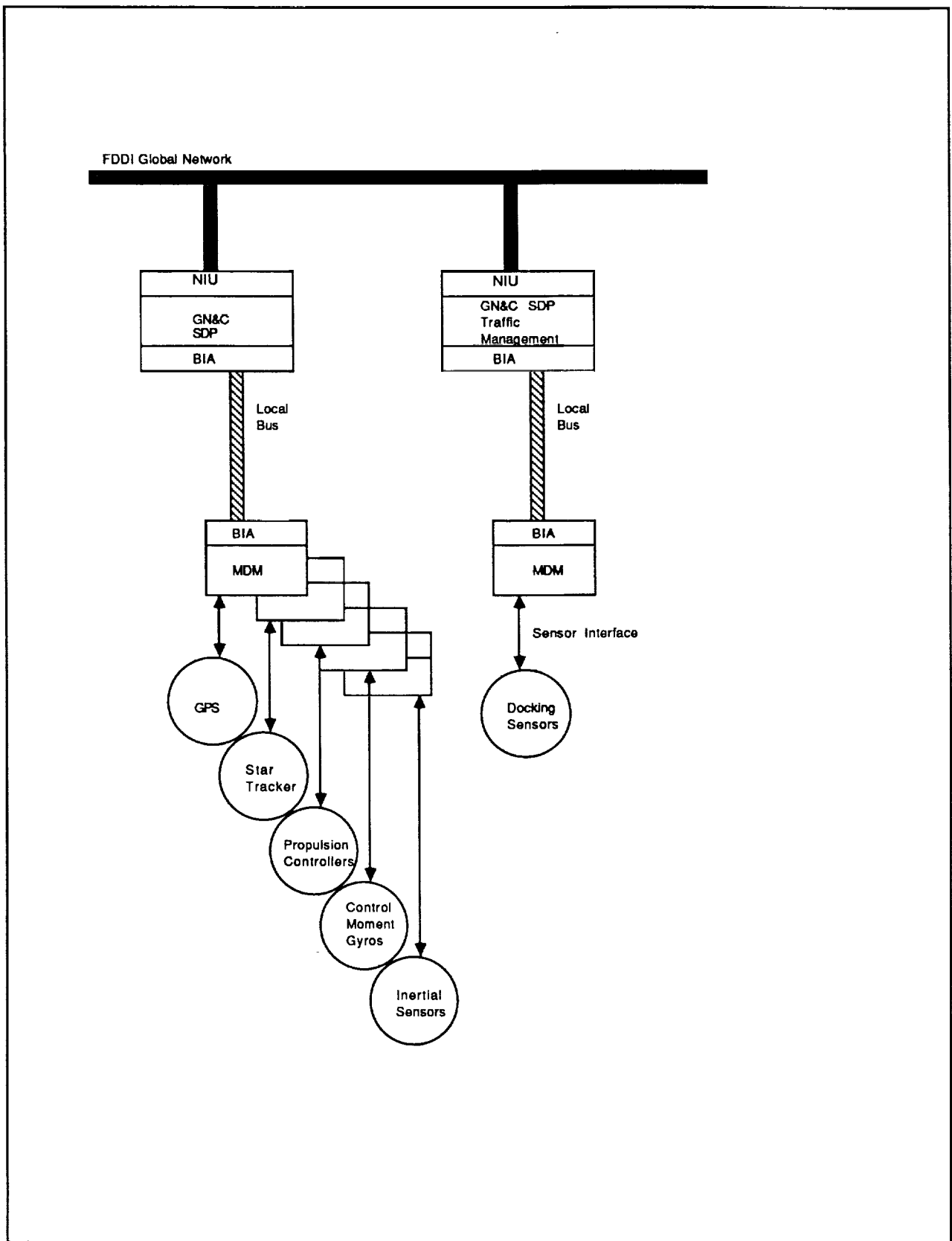


Figure 4-2
GN&C/DMS Architecture

where it is transformed into information for communication to the crew or to other space station systems. This division of functions between layers provides for loose coupling between the supervisory layer and the real-time control layer. Upper layer elements are interconnected via a global network consisting of a 100 megabits/second (Mb/s) token ring bus conforming to the Fiber Distributed Data Interface (FDDI) standard, while lower layer elements communicate with the upper layer via a 1 Mb/s local mux bus. The loose coupling approach offers the potential for application dependent clock cycle speeds within the lower layer, which may present synchronization problems. For example, an embedded imaging sensor/processor may run at video speeds, while an embedded rotary joint processor may be required to run at a different speed that is derived from a dynamic stability analysis. Automation and adaptive control technologies applied to the upper layer do not present as much of a simulation problem due to the incorporation of flight equivalent SDPs running FSW within the SSTF.

Advanced automation technologies, such as imaging machine vision type sensors and knowledge based controllers will call for trade-offs to be made as to the method for providing the functionality of the embedded system in the SSTF. Alternatives are to either **Functionally Simulate** the embedded system within the host computer, **Stimulate** flight equivalent hardware executing the embedded system software, or **Emulate** the embedded system on special purpose COTS hardware.

4.3.2 Embedded Knowledge Based Systems/Autonomous Systems

Autonomous or knowledge based systems (KBS) embedded within the lower layer of the DMS architecture may present special simulation problems. Implementation of these systems currently requires special purpose processors that perform symbolic computing operations. Functional simulation of expert systems is a relatively uncharted territory due to the limited scope and small number of KBS systems that have been incorporated into production aerospace vehicles. SSTF accommodation of these technologies may require that flight equivalent symbolic processors be stimulated to ensure representative embedded system performance. Stimulation of these processors would in turn put special requirements on the host based environment simulation model. The environment model would be required to provide for all of the scenarios programmed into the KBS world database for contingency training in mission operations. Heuristic behavior of the KBS would also need to be accounted for by the environmental model.

4.3.3 Robotics Technology/Telerobotics

Simulation of advanced telerobotics operation may be the most challenging SSTF simulation problem. The goal of the automation and robotics initiative is to help relieve the Freedom Station crew from many of the demanding EVA and time intensive operations associated with station assembly and maintenance. To facilitate meeting this objective, hooks and scars are being incorporated in the station design to accommodate automation capabilities as they become available. Anticipated Freedom Station automation and robotics capabilities will utilize machine intelligence, sensory aided perception, and advanced man-machine interfaces. The robotics technologies planned for incorporation in Freedom Station pose several potentially formidable simulation challenges, particularly in the visual system area.

A flexible robot control structure has been identified for utilization in the Flight Telerobotic Servicer (FTS) and other Freedom Station robotic systems. This scheme, known as the NASA Standard Reference Model for Telerobot Control System Architecture (NASREM) provides for varying levels of system autonomy and operator control regarding manipulator operations. Figure 4-3 indicates how the NASREM hierarchy would map into the DMS architecture in terms of system interfaces and levels of operator/manipulator interaction. The NASREM architecture provides for operator control over robotic operations via job oriented task commands ("remove & replace Orbital Replacement Unit (ORU) #2 in forward service bay"), object oriented E-move commands ("move

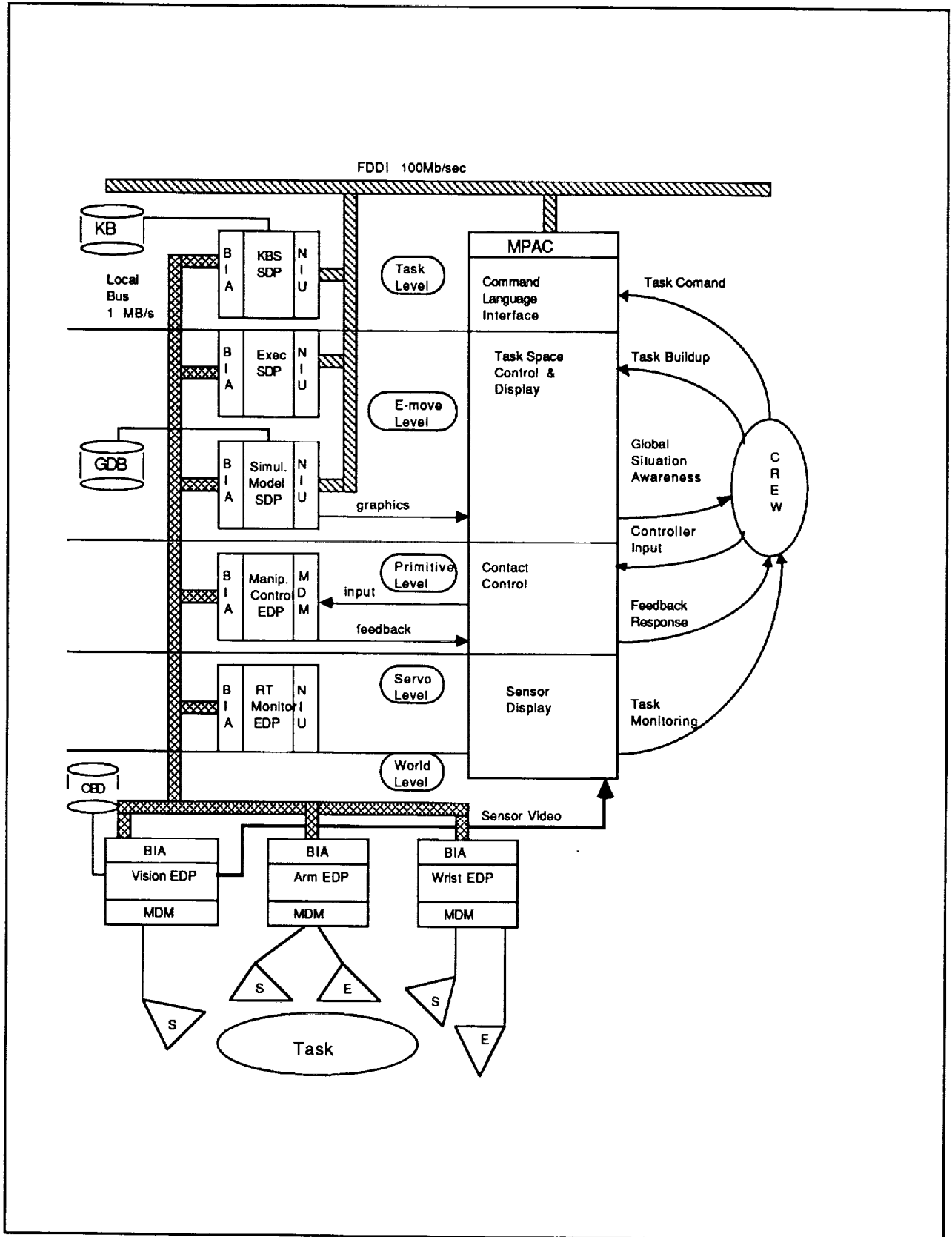


Figure 4-3
Robotics/DMS Integration

<object> from <source> to <destination>"), or by manual control of arm and end effector movements. To accomplish these operations safely and effectively, the operator will need extensive real-time queuing of the task situation. The operator must be able to accurately perceive what is happening, make cognitive decisions concerning the task process, and effect control over the task situation. Advanced display and control technologies are planned for aiding the operator in utilizing the NASREM configured robotic system, from within the station and from ground based controller workstations. The following sections discuss several technologies associated with robotics technologies that are challenging for SSTF simulation. These include predictive display and control, and Computer Aided Design (CAD) / Computer Aided Engineering (CAE) correlated sensory perception.

4.3.3.1 Predictive Display & Control

Remote operation of space based robotic manipulators from a ground based workstation is a difficult technical challenge due to the inherent time delay in the round trip transmission of data from earth to space. Delays of as much as 2 seconds are anticipated for ground based manipulator control loops transmitted to the station via the Tracking and Data Relay Satellite System (TDRS) geosynchronous relays. Research at the Jet Propulsion Lab (JPL) has indicated that control loop delays above 500 ms cause manipulator operators to fall into a move and wait strategy for accomplishing task objectives, resulting in very poor task performance. To overcome the time delay problem, research in predictive display and control technologies is being performed to increase the effectiveness of teleoperated control schemes. By executing a macro simulation of the robotic system within the ground based workstation in response to input commands, a real-time display can be generated that predicts the behavior of the space based manipulator. Researchers at JPL believe that by coupling this predictive simulation with CAD/CAE data regarding the object being remotely manipulated, a total manipulator/object signature can be computed in real-time that is sufficient to drive interactive 3-D graphics displays with correlated force feedback signals for the hand controllers. The CAD data would provide the spatial geometry and physical characteristics data needed to determine when end effector contact is achieved. The CAE data will allow the computation of the inertial response of the object to a given input force. Real-time sensor data from the manipulator in space would then be overlaid with the virtual graphics display to provide operator overall awareness of the task situation. This predictive display and control technology, in addition to providing a solution to the ground control time delay problem, could also be utilized on board Freedom Station to provide an embedded training capability that would permit crew members to remain proficient at robotic operations, without having to exercise the actual robotic equipment.

4.3.3.2 CAD/CAE Correlated Sensory Perception

Autonomous robotic operation at the NASREM task or E-move level requires that the manipulator be capable of "seeing" and "feeling" its way throughout the performance of the task. Research is currently being performed in the integration of visual imaging sensors and laser range finders with CAD/CAE databases to increase the robots' ability to maintain situational awareness of the task space. By correlating real-time task imagery and ranging data with the CAD/CAE database, the manipulator will be able to navigate through the task space with a higher degree of confidence than if driven by sensor data alone. This concept is analogous to the Terrain Profile Matching (TERPROM) approach for passive terrain avoidance utilized in some tactical fighter aircraft. By correlating radar altimeter readings and periodic position fixes with an on board digital map of the route being flown, the TERPROM equipped aircraft is able to anticipate and react to terrain contours and hazardous obstacles along the aircraft's flight path. The correlation of on board data with sensor data will allow the robotic manipulator to anticipate collisions or unsafe operating conditions in a manner similar to that of the TERPROM navigation system. This technology presents a challenging simulation task for the SSTF. If the CAD/CAE correlation is mechanized within the upper layer of the DMS environment (i.e. within the SDPs), then the Visual Scene

Database (VSDB) will need to be directly correlated with the Freedom Station CAD database to ensure realistic training scenarios. If the mechanization occurs in the bottom DMS layer (within an embedded processor), then other alternatives may be available, dependent upon whether the embedded processor is stimulated or functionally simulated. Correlation of the visual scene with the robot flight software is a challenging simulation task, because simulator VSDB's descriptors for man-made objects (i.e. aircraft, vehicles, etc..) are typically entered into the database through manual modeling techniques. The resolution of these models is also typically very loose. VSDB/CAD correlation will require precise matching of the visual database models with the CAD object descriptions. This correlation will be particularly critical if pattern recognition flight software is executed within the SSTF. Automatic generation of visual scene database models based on CAD data would be an ideal solution to this problem.

5.0 Enabling Simulation Technologies

Development of the SSTF concept will call for many decisions to be made regarding concept implementation. Many challenging simulation problems are anticipated in implementing the SSTF concept that must be solved by effective, efficient, and enduring technical approaches. It is in NASA's best interest to seek an SSTF architecture that offers maximum flexibility, scalability, and adaptability to change. Pursuit of these implementation objectives will better position the training facility for accommodating new missions, new on-orbit technologies, and new user requirements over the anticipated life of the Space Station Freedom program. The information in the following sections is offered to SSTF planners and decision makers as a broad look at the most critical technical issues surrounding SSTF concept implementation.

Several enabling simulation technologies hold promise for answering the challenges posed by the variety of simulation problems anticipated for the SSTF. These include conceptual technologies that offer life cycle advantages through modular systems definition techniques, architectural technologies that provide flexible and adaptable system configurations, software technologies that support the architectural technologies, processing technologies that allow simulation of advanced automated systems, and visual system technologies that, in conjunction with the architectural and software technologies, provide an overall integrated training environment for highly transferrable crew training. The pursuit and application of these technologies may better enable the SSTF to weather the evolutionary changes which challenge the initial system design. But these technologies must not be taken at face value. Many technical issues and considerations must be accounted for when implementing a given technology. The following discussions are directed at uncovering some of these issues, so that they may be more closely examined in further detail.

5.1 Conceptual Technologies

Several new trends in simulation system definition offer distinct advantages over traditional methods in the design of complex, multi-crew training systems like the SSTF. These concepts help assure that training simulation devices are designed as modular, expandable, and flexible systems. Application of these conceptual technologies in the SSTF design process will allow the SSTF configuration to accommodate hooks and scars necessary to provide training support for evolutionary automation and robotics technologies, minimize the potential problems involved in integrating with the SMTF, accommodate a variety of diverse payload technologies, and ease the tasks associated with mission-to-mission reconfiguration. Technologies that play a key role in this front-end system engineering process include modular simulation concepts, distributed processing concepts that support modular simulation, and the concept of integrated environmental modeling. These goals can be achieved through the application of front-end systems engineering analysis which categorize and quantify the simulation system's logical, physical, and functional processes.

5.1.1 Modular Simulation Concepts

The concept of modular simulation focuses on reducing the complexity of simulator systems by logically and physically compartmentalizing the design into independent, well defined subsystems. It is a building block approach that describes the overall simulation problem in terms of separable tasks. These tasks are then allocated to functional simulation entities with well-defined interfaces and internal processes. The functional process is then allocated to physical hardware and software components. The modular design approach is based on fundamental systems engineering principles that are applied specifically to configure real-time simulation systems. Adherence to these principles provides clear development paths and provides for graceful growth capabilities as the simulation requirements are expanded.

The front-end systems engineering approach is intended to prevent the integration, maintenance, and upgrade problems experienced with simulation and training systems currently in the field. Industry standard practice has been to settle on the physical design of the simulation system prior to the identification of the overall tasks and functional processes that the system needs to perform to support achievement of training system objectives. It is very important to consider the evolutionary requirements of the simulation system early in the system conceptualization process. History has shown that centering a training system design around specific computer equipment or proprietary software components can be very problematic in terms of life cycle cost, system expandability, and flexibility in meeting new requirements. By dividing the logical simulation processes based on interface considerations and separating them from the physical environment, a system configuration can be established that is easy to understand, implement, and modify. Figure 5-1 shows an example of how space station system simulation requirements can be logically allocated to simulation processes. Each process within the simulation environment performs a function required to meet training requirements. Once each process is defined in terms of functions and logical interfaces, the process may be easily allocated to hardware and software.

Real-time system oriented Computer Aided Software Engineering (CASE) concepts are emerging that support the principles of modularity in the systems definition process. These tools will allow the system designer to identify processes which are inherently parallel, process timing synchronization requirements, and data flow considerations early in the design phase. This capability allows system designs to be configured that specify and optimize the best physical computer resources needed to perform the simulation task. CASE concepts for real-time simulation are discussed further in Section 5.3. The modular simulation approach offers an alternative to the myriad of problems caused by the premature selection of specific computer systems prior to complete system requirements identification.

5.1.2 Distributed Processing for Modular Simulation

Distributed processing concepts involve allocating simulation functions to individual processor environments. A processor environment can be thought of as a collection of individual computing units that are grouped to allow collective processor-to-processor communications. Each computing unit contains a CPU, local memory, and I/O capability. Computing units are interconnected via an inter-node link to form an integrated computing node that is managed by a node controller. Computing nodes are connected by an intra-node link to form a distributed computing system. The distributed processing approach divides the overall computation load into independent processes that may be performed in parallel. Individual node controllers allow the computer resources to easily be configured to support specific applications. A distributed architecture can be dynamically configured to support independent applications (part task training) or a single massive application (combined training). This approach has numerous advantages over the single virtual machine approach commonly used for real-time simulation.

The single computer approach is typically represented by a master central processing unit (CPU) and several slave CPUs that communicate with each other through shared memory. This configuration performs as a single virtual machine since only one CPU may access shared memory at a time, resulting in a serial execution environment. This approach is also not very scalable in terms of processing power expandability, since only a fixed number of CPU's may share the computer's main bus, and only the master CPU is capable of handling I/O. To overcome these limitations, the simulation industry is developing distributed processing concepts that support the principles of modular simulation and provide flexible, scalable computer resources. The distributed processors may be interconnected through either loosely coupled networks or tightly coupled memory linkages. Issues associated with each of these interconnection methods are discussed in Section 5.2.

Flight Hardware/Software

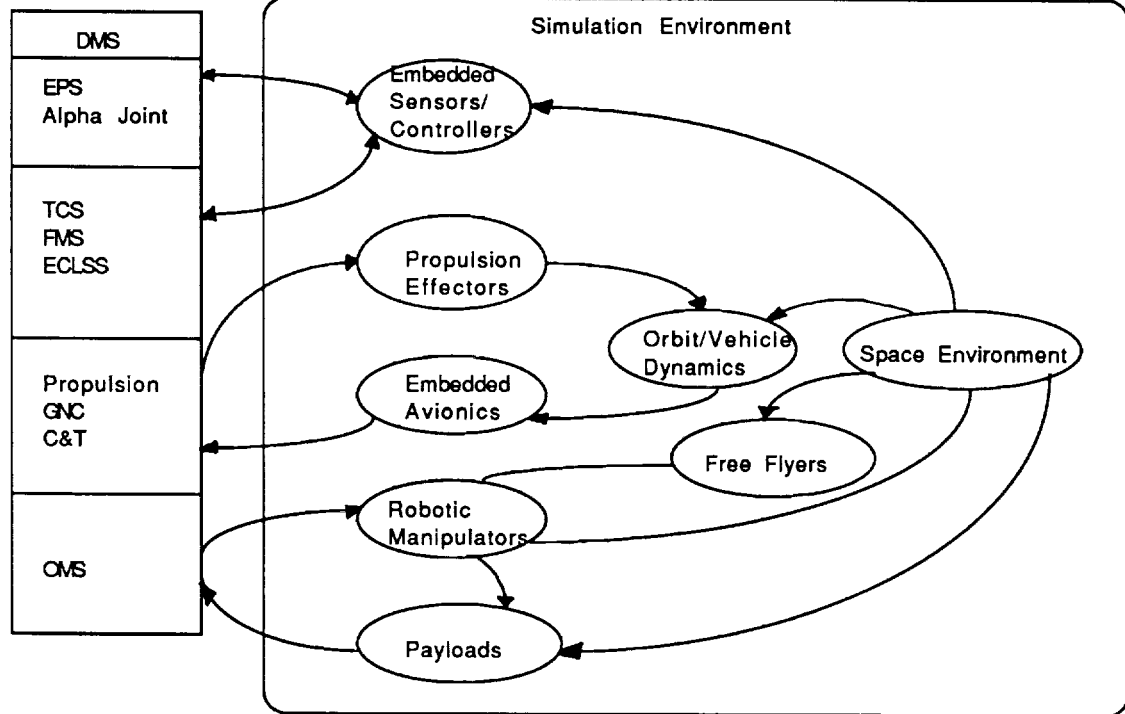


Figure 5-1
Space Station Modular Simulation Concept

Figure 5-2 describes the differences between the typical simulator computer system and the distributed simulation system. The monolithic approach typically centers around a proprietary internal bus and is designed to support centralized, local applications that are easily handled by a single computer. The distributed approach relies upon grouping computing unit to form computing nodes. The key to the distributed architecture is the processor-to-processor linkage (intra-node bus and inter-node bus). Several technologies discussed in section 5.2 allow independent processors to communicate reliably and predictably at real-time speeds. The computing units may be easily interconnected and sized to form computing nodes that support dynamic, large scale applications that often cannot be handled by any one computer.

The strength of the distributed processing approach is that each individual computing node may be uniquely sized to match the simulation process that has been allocated to it. As the simulation requirements expand and mature, additional nodes can be added or easily re-sized to accommodate the requirements. In addition, the simulation system configuration is better able to support advanced simulation requirements by matching process to processor. The distributed approach also provides a growth path for computer technology migration. As more capable real-time processors emerge, they may be swapped for existing processors without seriously impacting the simulation functional allocation.

5.1.3 Integrated Environmental Modeling

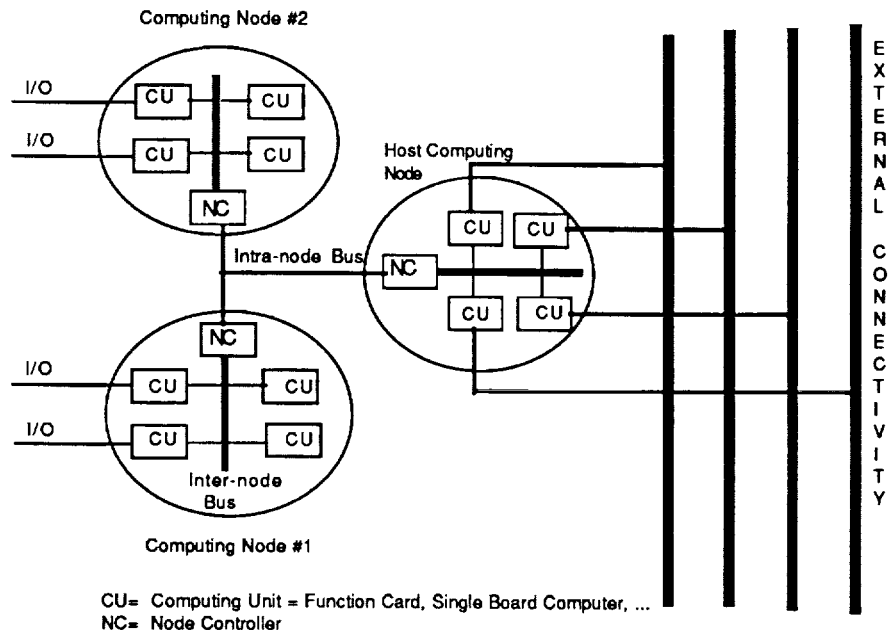
Integrated environmental modeling refers to a centralized approach to simulating external processes. An example of an external process is the characteristic behavior of the earth as a planet. A corresponding internal process would be the sensing of an earth attribute by an embedded space station system. The concept of integrated environmental modeling focuses on the definition of a unified environment that may be sensed in a consistent manner, independent of the sensor type. The concept is illustrated in Figure 5-3. In the non-integrated approach, each sensor models its own world, totally independent of other sensors and other environmental effects. Non-integrated environmental models are difficult to understand and maintain, and may result in inconsistent cues to the crew. Integrating the environmental model as a separate simulation function ensures that all cues conveyed to the crew will be correlated and provides for the modular incorporation of new sensors as they become available.

5.2 Architectural Technologies

Advanced computer architectures offer flexible alternatives for meeting the demanding and evolutionary simulation requirements posed by the SSTF. The concept objective is to configure a real-time simulation environment that can evolve and grow gracefully as the space station training and operating requirements expand to include new missions, payloads, and embedded system capabilities. The core computing resources utilized within the SSTF must be based on architectural technologies that have a bright and predictable future, offer a clearly defined growth path to accommodate increasing performance requirements, and promise to be fully supported now and in the future by the commercial computer industry. Figure 5-4 identifies the current operational computer system concept in terms of functional interfaces and SSTF connectivity required to support multiple, concurrent and independent training sessions. Many technical issues must be considered in determining the configuration for implementing the host simulation computer system concept.

The information presented in this section is designed to provide decision support for the assessment of candidate architectures that are capable of fulfilling the real-time computer resource requirements of the SSTF. The architectural technologies considered for analysis here are important to the current and future trends in real-time simulation oriented computing systems. These technologies have been grouped according to the method utilized to implement the specific capability

DISTRIBUTED SIMULATION COMPUTER ARCHITECTURE



TYPICAL MONOLITHIC SIMULATION COMPUTER ARCHITECTURE

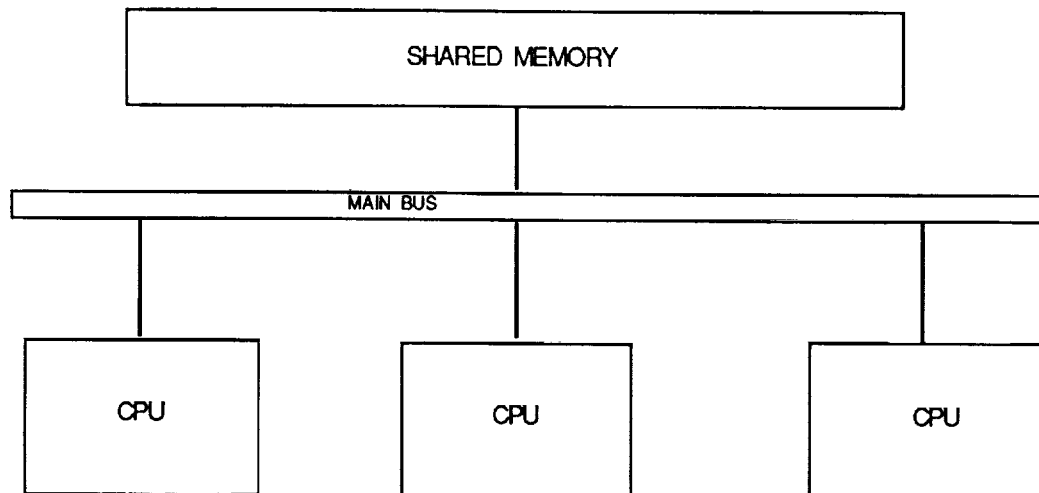


Figure 5-2
Simulation Computer System

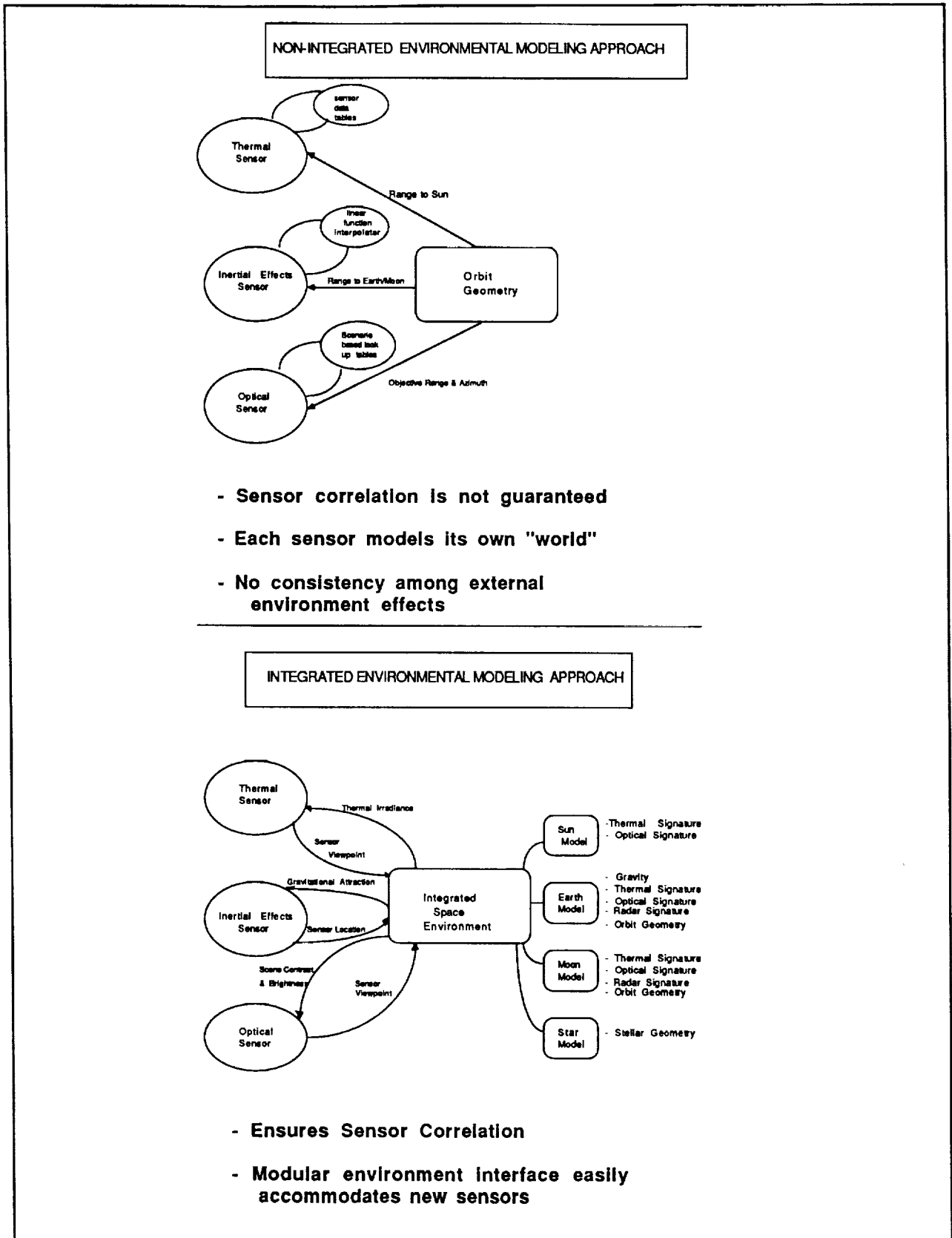


Figure 5-3
Integrated Environmental Modeling Concept

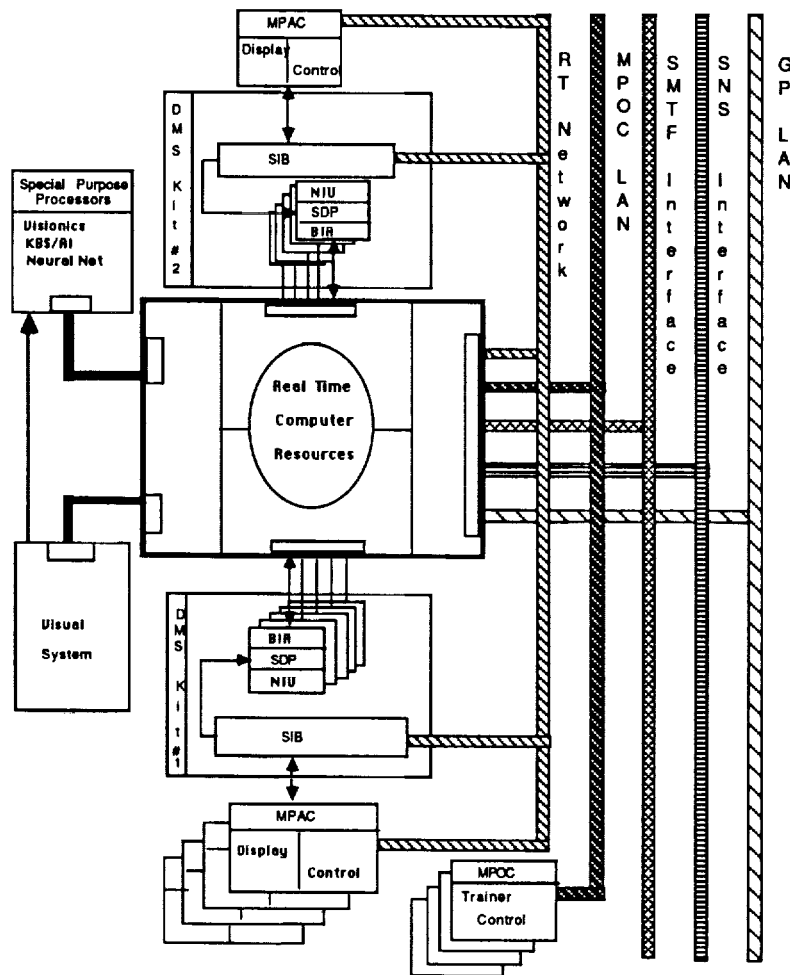


Figure 5-4
Operational Computer System Concept

that the technology offers. These groups are: loosely coupled networking methods, tightly coupled memory linkage methods, and processor selection. A number of issues are presented for evaluating each of these technology areas.

The SSTF is characterized by a diverse set of computing needs representing a wide range of real-time and batch requirements. In addition, the integrated SSTF must provide connectivity to several other training devices to fully implement training goals (i.e. the Shuttle Mission Training Facility-SMTF, Neutral Buoyancy Lab, Mission Control Center, etc.). The need for a modern Local Area Network (LAN) is clear but there are a number of issues which determine their effectiveness in a given situation. In a complex computer system containing several local computing nodes with requirements for the various nodes to communicate with each other, the system should be analyzed on at least two levels: the tightly-coupled, local level and the loosely-coupled, global level.

There are several important issues associated with selection of a communication method when dealing with real-time applications: scalability, error detection and recovery, overhead, and performance. Scalability deals with issues of flexibility and the ease of expansion to accommodate changing communications requirements. This is one of the most critical features needed by a complex system with unknown growth needs such as the SSTF. Performance issues are significant because communication speed will affect frame processing time when the processes are distributed over a network. Determinism (of message propagation time) is also a critical performance issue.

To date, a front-end training analysis has not been performed to determine precise training objectives and curriculum for the SSTF. In order to size the communication requirements of a complex system, one must be able to estimate the peak data communications rates for each communications path using average message size, message density, system overhead and bandwidth. These estimates will not be possible for SSTF until design options and the training requirements are stabilized. This makes the selection of scalable and open standard architectures a most important factor.

5.2.1 Loosely Coupled Networking

Loosely coupled networking schemes allow dissimilar processors to communicate with one another according to a standard interface protocol. This protocol can be separated into physical and logical layers of communication such as defined by the International Standards Organization (ISO) Open Systems Interconnect (OSI) standard for use in LANs, or by integrated protocols such as VME bus and MultiBus that standardize the signal lines and interface modules required for processor-to-processor communication. VME bus interfaces also offer a means of tightly coupling processors and devices on a one-to-one basis.

LANs are used to connect a number of computer systems (global) for the purpose of sharing or exchanging data. Data is organized into messages or message packets for transmission, with the transmission speed being dependent on the type of physical transmission line used. For example, a twisted-pair wire can transmit up to 1 Mb/s, while thin or thick coaxial wire can attain rates above 10 Mb/s, and fiber optics cables can extend into the gigabit per second range.

There are several popular LANs which have received wide industry support and are formally documented by various international standards organizations. Loose coupling through LANs is characterized by the transfer of message packets, flexible configuration options, and the ease with which nodes with different vendors' products can be connected. Processors of varying architectures and capabilities that accept the LAN protocol can be connected over fairly large distances (one to two Km) and repeaters are available which increase these limits. They tend to be very scalable but can have dramatic performance degradation when overloaded.

Communications for complex distributed real-time systems must form the backbone upon which predictable, stable, scalable system solutions are built. To be successful, the real-time communications must be able to predictably satisfy individual message level timing requirements. In a nonreal-time setting, it is sufficient to verify the logical correctness of the communication; however, in a real-time setting it is also necessary to verify timing correctness. Timing correctness refers to ensuring the ability to schedule synchronous and sporadic messages as well as ensuring that the response time requirements of asynchronous messages are met. Ethernet, for example, has very low predictability of message transmission and arrival times because of the unpredictable increase of message collisions and retransmission as the network loading increases.

Each computing node is connected to the LAN by interface modules which implement a protocol: the rules which determine the format, speed, and addressing for data traversing the LAN. Computing nodes can be connected to LANs in a variety of ways: rings, stars, trees, busses, and other topologies as shown in Figure 5-5.

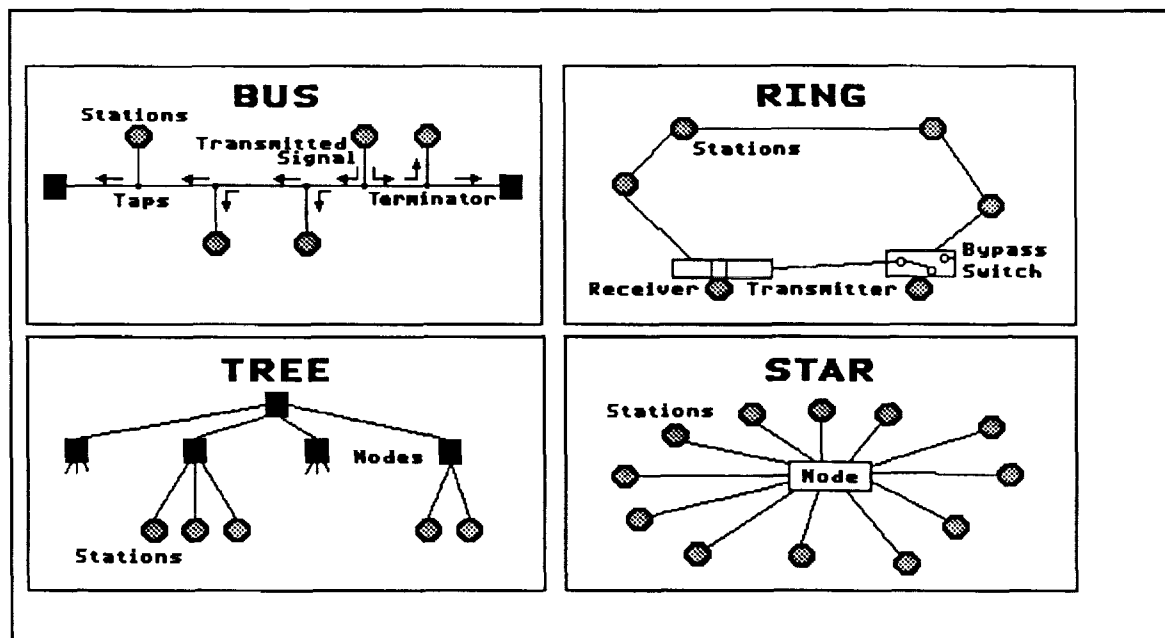


Figure 5-5
Typical LAN Topologies

5.2.1.1 Ethernet

Ethernet (IEEE 802.3 standard), is a 10 Mb/s coaxial cable bus using baseband modulation and implementing a "carrier sense multiple access with collision avoidance detection" (CSMA/CD) protocol in hardware. Ethernet has very low predictability of message transmission and arrival times because of the unpredictable increase of message collisions and retransmission as the network loading increases. These collisions have been shown to be exponentially increasing with respect to bus loading and can begin to have an effect at relatively low loading levels (about 20%). It is a very scalable network, but its use is questionable where high speed, deterministic, hard deadline communications are required.

5.2.1.2 Token Ring

The Fiber Distributed Data Interface (FDDI) fiber optic cable being considered for the Freedom Station global network provides relatively inexpensive connectivity with transmission rates of 100 Mb/s and sustained transfer rates of 80 Mb/s. FDDI (ANSI ASC X3T9.5 standard) is a counter-rotating double ring token passing arrangement, as shown in Figure 5-6.

Token bus and token ring networks have a somewhat better deterministic performance than other LAN structures and are very scalable. Priority schemes, built into the token's data structure, allow message priority to be easily implemented. Heavy loading and turn-taking (resulting from token passing) are effectively handled by this priority system. The current network trend is toward increased use of Time Domain Multiplexing (TDM) implemented by a cyclical executive such as the token-passing scheme within FDDI. Research has shown that as the number of independent communicating tasks increases, the uncertainty of message timing also increases. Also, reliability and availability issues are driving systems to become increasingly reconfigurable, which exacerbates the problem of determinism. The problem of non-deterministic networks is currently the focus of scheduling theory research activities and it seems that currently available products are not particularly well suited (theoretically) to the task.

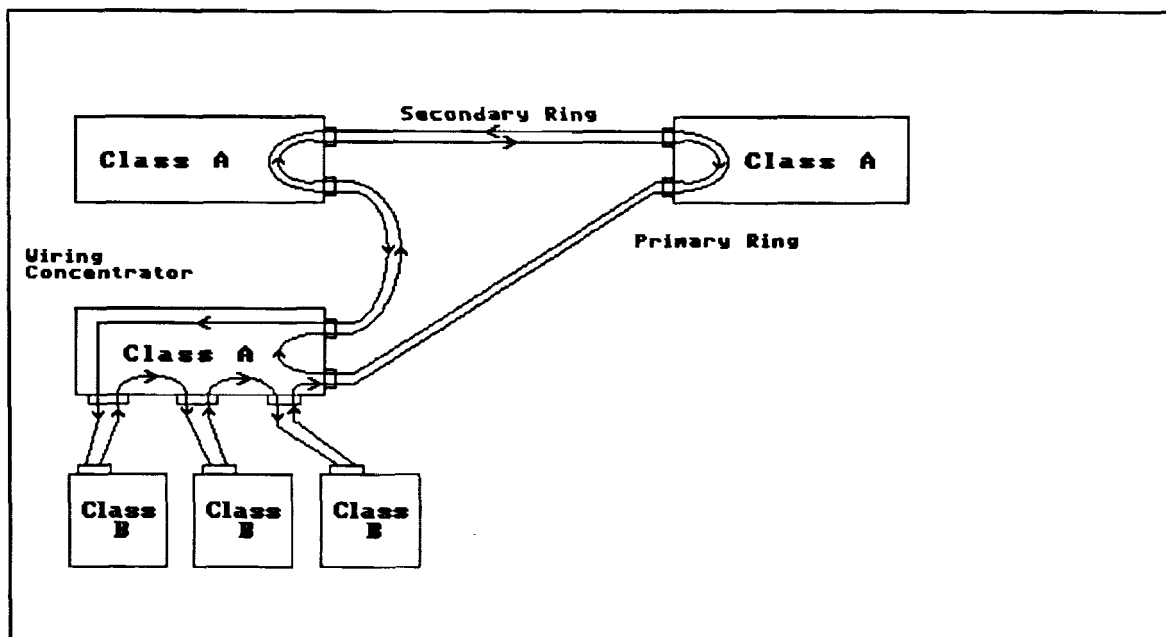


Figure 5-6
FDDI Token Ring

5.2.1.3 VME Bus and Multibus II

VMEbus and Multibus II are standard parallel busses which have popular support as local network interfaces. Each has a maximum bandwidth of about 40 MB/s (megabytes per sec) but have very different operational philosophies. Both are capable of processor independence, allowing computers from different manufacturers running at different clock speeds to operate on the same bus. Multibus has gained considerable favor in industrial automation applications while VMEbus has a larger installed base and is more popular in scientific real-time applications. Either bus would be a satisfactory interface for localized communications within the SSTF.

Multibus II uses a message-passing protocol. Message packets contain source and destination addresses as well as several bytes which can be used to describe up to 255 virtual interrupt sources or destinations. The message passing is usually handled by a special coprocessor rather than the board's main computer and eliminates the need for dual ported memory. Multibus II is synchronous and multiplexed which (according to some authorities) provides higher reliability. Multibus cards must synchronize to the bus clock rate of 10 Mhz for transmission, and if a clock edge is missed, a wait state of 100 nsec is needed to catch the next clock pulse, potentially degrading performance. Bus arbitration is distributed among several "bus masters" who arbitrate for bus services based on a prioritized ID assigned by a central services module.

VMEbus is asynchronous and non-multiplexed, using seven prioritized hardware interrupt lines to apportion tasks among multiprocessors. The asynchronous bus allows each processor in a multiprocessor system to transfer data at its fastest possible rate. VMEbus' arbitration method is "centralized" in that one global arbiter board, slot one of the card cage, handles bus-access requests over dedicated, daisy-chained bus-request and bus-grant lines to the other boards' requester circuitry. VMEbus allows four priority levels and three modes: prioritized, round robin, and single level. Prioritized arbitration assigns the bus to the board driving the highest priority bus-request line. In round robin, the arbiter assigns access on a rotation priority basis, from highest to lowest. Single level arbitration serves only one priority, relying on the daisy-chained bus-grant lines to determine the order in which boards get access to the bus.

5.2.1.4 Real-Time Networks

Network communication schemes are beginning to emerge that provide deterministic message passing communications between dissimilar processors. One real-time network supplier (SYSTRAN) offers an innovative approach known as the Shared Common Random Access Memory Network (SCRAMNet) that combines the benefits of tightly coupled shared memory linkages with the open system flexibility provided through loosely coupled communications methods. The SCRAMNet system consists of a set of host interface cards connected by fiber optic cable. The host interface cards plug in to standard card slots provided for computer interfaces such as 9U sized VME, MultiBus II, and Q-Bus. Memory located on the interface card is mapped into the host computer's memory, and data changes to this memory area are detected and transmitted to the other network nodes. This approach tends to eliminate unnecessary bus traffic. By not implementing complex message passing protocols, and through use of fiber optic cable, SCRAMNet is able to achieve message transmission times of less than 7 microseconds over distances exceeding 700 meters. The major limitation of the SCRAMNet scheme is that currently, only 512 K bytes of shared memory can be supported between network nodes. Current SCRAMNet applications utilize between 2 and 10 network nodes.

5.2.2 Tightly Coupled Memory Linkage

Processor-to-processor communications utilizing tightly coupled memory linkage methods allow compatible processors to almost instantaneously transmit and receive data. The tightly coupled memory linkage method avoids the need for data format and protocol compliance by promoting direct memory-to-memory read and write capability. There are many very fast vendor-proprietary bus structures available that perform very well on that vendor's equipment. They are often limited to a short total length (up to a few hundred feet) with speeds of 20-50 MB/s.

Use of proprietary interconnect busses is fine for use within a localized (functional) computing node for distributed processing (intra-node) as long as it can grow to meet the localized functional need. However, all localized computing nodes which require widespread communications within the SSTF (internode) should also support interfaces which have an open system architecture. These open architectures are those which are backed by industry standards organizations (e.g. ANSI,

IEEE) and have received wide support from the computer manufacturing industry and third party suppliers.

5.2.2.1 Broadcast Memory

The broadcast memory method for tightly coupled processor communications involves the broadcasting of shared memory areas between all processors accessing a common bus. Whenever a processor writes to a location in its copy of shared memory, the bus then "reflects" this memory area to all other processors accessing this same shared memory region. This approach eliminates the bottleneck caused by multiple processors attempting to access a central shared memory area. Multiple broadcast memory buses can be utilized to form a matrix of computing nodes capable of tightly coupled communication.

Encore/Gould offers an innovative Reflective Memory design in their Concept 32/2000 product line. In addition to a high speed local memory bus and an external Sel BUS, the reflective memory bus allows write-only memory updating to all connected processors over an electrically passive bus. The currently available reflective memory bus has a 26.6 MB/s bandwidth, but the bandwidth will be doubled by product enhancements planned for 1990. Electrically, the reflective memory bus connects to local memory through the CPU card but does not impact CPU performance. Up to eight processors can be connected by the reflective memory bus, and in turn, each of the eight processors can, by a multi-drop connection, reflect memory to another string of eight processors, forming an 8x8 matrix of processors connected by reflective memory as illustrated in Figure 5-7. Reflective memory is very deterministic with a worst-case propagation delay of about one-half microsecond. Additional hardware lines offer cross-coupled interrupts among processors for timing synchronization.

5.2.2.2 Shared Memory

Shared memory is the traditional simulator architecture that uses a memory bank to which several processors can read or write, as illustrated in Figure 5-8. Key variables, which are of interest to more than one processor, are assigned to this common memory. This is a common technique found in many of today's simulators. Information transfers are fast because no packing, formatting or message software is involved, although some wait states are likely when memory contentions occur. Scalability of shared memory systems is limited and they are usually proprietary to a particular vendor. As the number of attached processors increases, the longer each must wait to be granted a memory access. Typically, all processors must be in close proximity to the shared memory resource, and complicated software semaphores must be used to ensure coherent data access. Shared memory can also be a single point of failure within the system.

5.2.2.3 Crossbar Memory

Crossbar memory is a linkage technique that utilizes a separate bus for each interconnected processor, so that each processor has a direct communication path to every other processor, as illustrated in Figure 5-9. The crossbar approach provides for a very scalable architecture, with constant memory access times independent of the number of interconnected processors. Both I/O and processing can be performed in parallel.

Bolt Beranek and Newman, Inc. offers an innovative crossbar memory system known as the TC200 System which uses a "Butterfly switch" to provide efficient and transparent access by each processor to all locations in memory, whether local to a processor, remote on another processor, or external on a disk or tape. Every processor is connected to the switch via an 80 MB/s, 32 bit bus. The switch paths are eight bits wide and provide 38 MB/s bandwidth and allow multiple remote memory accesses to occur in parallel. The switch is also modular, providing an additional 38 MB/s

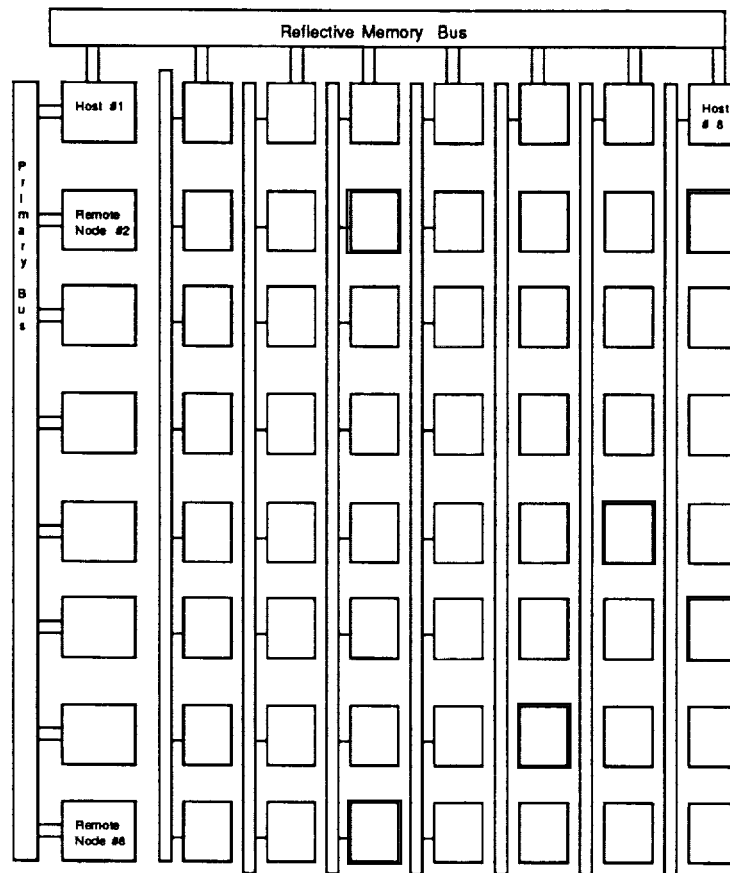


Figure 5-7
Typical Broadcast Memory Configuration

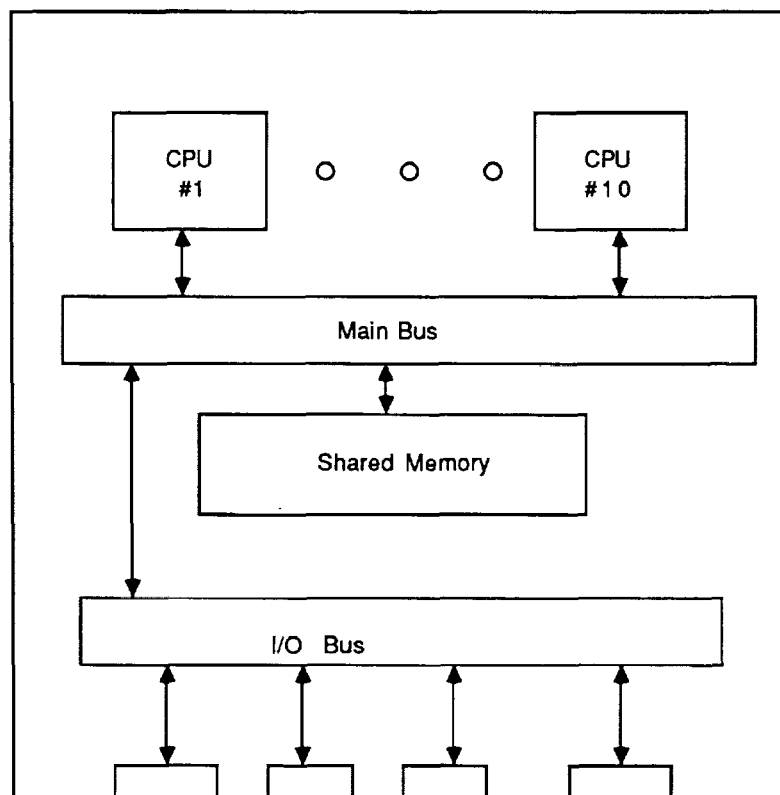


Figure 5-8
Typical Shared Memory Configuration

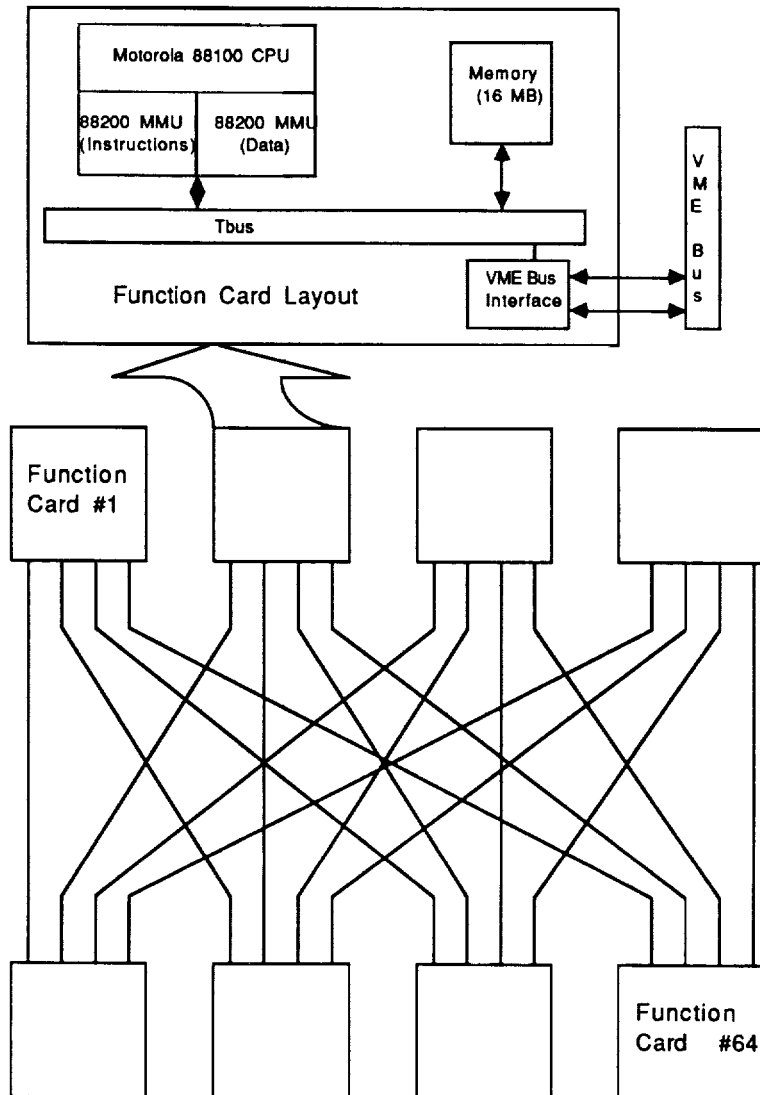


Figure 5-9
Typical Crossbar Memory Configuration

in communications bandwidth with the addition of each cluster of eight processors. The architecture is very scalable and supports up to 64 parallel processors, with growth capability to accommodate up to 504 processors.

5.2.3 Processor Selection

Today, computer system designers have a wide variety of processors (CPUs or board-level products) from which to choose. Some of the CPU designs are commercial products with open (i.e. non-proprietary) and well-documented features, while others are considered proprietary by the computer manufacturer. In today's market, where the total life span of a CPU is less than 8 years, chip designers and system integrators are very careful at the onset of a new product design to build in an orderly growth path for the product. Proprietary CPUs are designed with a particular market or application niche in mind and frequently offer a speed advantage in that area when compared to chips designed for wide distribution. The availability of software is generally better for systems based on standard designs although proprietary designs are well supported in the target application area by the manufacturer through internal development and arrangements with third party OEM software houses.

5.2.3.1 Processor Architectures

The majority of today's computer architectures are based on the classical von Neumann scheme. This paradigm includes a global addressable memory which holds both program and data objects, and a program counter which holds the address of the next instruction to be executed. The program counter is implicitly updated by machine instructions to provide the machine with a sequence of instructions to execute, implying a single locus of control, a fundamental bottleneck of the von Neumann model for parallel processing.

The most serious problem with any form of memory sharing in a multiprocessor machine is the underlying execution model, which implies a global, updatable object-oriented memory. It is common for processors to compete for the right to update a memory location. The program fragments, which execute in parallel and share data, must be synchronized by operations such as test-and-set, semaphores, or message-based primitives which require considerable overhead and reduce performance. There are other fundamentally different schemes which are beginning to show success in certain application areas, some of which are necessary to fulfill the diverse computing requirements of SSTF. These are discussed in following sections.

5.2.3.2 Data Flow Machines

Data flow machines deal only with values and not addresses (of values). The basic operators produce a value which is used by other operators. The model has no instruction counter: an instruction is enabled in and only if all the required input values have been computed. Enabled instructions consume input values, execute, and produce sets of output values which are sent to other instructions that need these values. An instruction in data flow has no other side effects, and a language based on data flow concepts does not introduce sequencing other than the ones imposed by data dependencies in the algorithm. In principle, it is possible to expose all of the parallelism in a data flow program.

Using this concept, it is possible to develop very powerful and deterministic real-time processes known as event-driven execution. Software modules are only scheduled for execution when a module input changes, thus signalling the algorithm to compute a new output value(s). This removes the system from the high overhead burdens of continuous polling schemes to detect when a module requires execution.

5.2.3.3 Parallel Processing

Multiprocessing is a platform architecture strategy in which two or more processors perform as one integrated system. True parallel processing is achievable on such an architecture and is characterized by:

- A high speed memory bus to a shared memory area. Since all processors are typically in the same equipment rack, the bus is physically short and can provide high bandwidth.
- A shared I/O bus allowing all processors access to a common set of peripheral equipment.
- An operating system which dispatches tasks from a central dispatch queue to processors in an available processor pool without bottlenecks associated with master/slave relationships. All processors are available for all jobs. The classical task of load balancing real-time tasks is eliminated.
- A scalable configuration, capable of easily accommodating additional processors.

Language compilers must be capable of identifying and exploiting the parallelism inherent in applications, splitting the identified code segments into "parallelized" packets. There are some cases in which execution order of subroutines or packets is extremely important. For example, in aerodynamic models it is critical that velocity and acceleration components be calculated from data collected within the current data frame. The system must have some means of assuring that tasks will execute within a required timing window and that tasks dependent on results of other tasks' calculations observe the proper dependency relationships. At the present time, the compilers and toolkits required for production quality parallel processing environments are still evolving and it may be some time before they are found in wide use.

5.2.4 Reliability and Maintainability

The architecture of the computer system plays an important role in determining the availability (fault tolerance) of the computer resources for operations and the speed with which problems may be isolated so that the system may be repaired and returned to operational status. Reliability and maintainability issues are key items for consideration in the SSTF host computer complex concept because of the high level of system availability required to support crew training.

Computer system designs that rely upon a centralized mainframe type architecture are susceptible to loss of operational capability when a key component (such as the shared memory) fails. Although the probability of a critical failure occurring is small due to the minimum complexity of the mainframe architecture, when a failure does occur it usually brings down the system due to the high degree of component integration.

A distributed multi-computer architecture exhibits greater complexity than the centralized architecture due to the greater number of components that make up the distributed computer system. Even though expected reliability is reduced due to the greater number of system components, system availability of the distributed architecture can be greater due to the inherent advantages of the functional process distribution. The distributed approach allows for the allocation of spare computing units within each computing system node. Since each computing unit is under software control of the node controller, a high degree of configuration flexibility is provided. If a failure occurs that results in the loss of a computing unit, the node controller can easily reassign a spare computing unit

to perform the functions of the failed unit. Similarly, a spare node controller can be assigned the functions of a failed node controller. A failed computing unit or node controller can be electrically switched out and quickly repaired or replaced without affecting the overall operation of the computer system. This capability is made possible through the use of passive inter-node and intra-node communication links. Passive transmission methods (i.e. fiber optic cable) allow the failed computing unit to be bypassed on the inter-node bus without affecting the remaining computing units within the node. The capability to semi-automatically recover from system faults without impacting the operation of the total system allows the distributed multi-computer approach to exhibit high levels of availability, even though the architecture is complex.

5.3 Software Technologies

Emerging software technologies are becoming commercially available that support the architectural technologies discussed in Section 5.2. These new technologies, centered around industry standards for programming languages, operating systems and the interaction between the two, offer the promise of solving many of the costly operations and maintenance problems related to the use of complex, customized, and often proprietary simulation solutions that are in current field use. To bring these promises to practice, though, many technical issues associated with these software technologies must be identified and addressed. Successful SSTF computer complex concept implementation will require careful planning with regard to the impact of these issues on the chosen computer system hardware/software configuration. The following discussion is intended to provide a high level overview of important issues associated with Ada language simulation environments, Ada compatible real-time operating systems, and real-time simulation development tools.

5.3.1 Real-time Ada

Several deficiencies have been identified by industry regarding the ability of ANSI/MIL-STD-1815A compliant Ada to perform in hard real-time application environments. These deficiencies include the inability of Ada to support explicit task scheduling, dynamic task prioritization, task blocking, or strict time slice management. Other known problems are caused by nondeterministic real-time interrupt handling via task rendezvous entries, and the possibility of task priority inversion due to the way in which Ada handles task synchronization. These problems have resulted in several different approaches undertaken by industry to allow the Ada language to be utilized effectively in hard real-time applications.

One approach taken by Ada vendors (DDC-I) is to provide source code changes that support hard scheduling along with their validated Ada compiler. This approach is characteristic of the "make Ada work now" method that suppliers have pursued to meet the demand for real-time Ada. This approach is not acceptable in many circumstances because it violates the Ada standard and requires that a special waiver be granted by the government sponsor. This method results in Ada software that is not portable and does not conform to the established standard.

Another path to a real-time Ada environment offered by industry is that of a real-time Operating System (OS) kernel that is custom tailored to support ANSI/MIL-STD-1815A without modifications. The OS overcomes the known Ada deficiencies by implementing real-time executive functions through packaged interfaces to the Ada language. The Ada tasking model is not utilized. This approach allows run-time simulation functions such as time slice management, inter-process communication, and dynamic process prioritization. This approach may be unacceptable due to the custom nature of and lack of standardization among operating systems that offer these capabilities. Another deficiency is that a standard does not currently exist for the OS to Ada language interface.

In response to these issues and the difficulty experienced within industry in developing real-time Ada systems, the Ada Joint Program Office (AJPO) is pursuing a revision to ANSI/MIL-STD-1815A to include incorporation of real-time environment features. This initiative, known as the Ada 9X program, seeks to resolve known problems by offering run-time extensions to the language in a way that will minimize the impact to existing compilers and installed software systems. This program is very controversial and is expected to drag on past the planned September 1990 release date. If implemented, the Ada 9X initiative offers the possibility that real-time Ada may be run "bare machine," without the need for an OS kernel to implement executive functions. This capability offers significant advantages since reliance upon custom software for a real-time environment would be virtually eliminated. An alternative approach, promoted by the Software Engineering Institute (SEI), is to define a standard for a run-time kernel that supports the current ANSI/MIL-STD-1815A. The kernel would provide all of the real time features, and would also eliminate the need for a custom, proprietary OS kernel to support real-time Ada usage.

5.3.2 Operating System Functions

Operating system functions are critical in determining and controlling real-time simulation performance. Typically, real-time operating systems support dynamic process manipulation, inter-process communication, task scheduling, and time slice management. Dynamic process manipulation allows the OS to control task prioritization, blocking/removal of tasks from execution, and task creation. Inter-process communication controls the way in which tasks access memory. Some OSs implement resource locking to allow only one task at a time to access given memory locations. Real-time task scheduling is implemented on a cyclical, frequency synchronized basis. This approach ensures rigid task queuing for deterministic system behavior. To enforce simulation determinism, the OS may also manage the time slice events allocated for each process. The OS effectively manages the run-time environment by monitoring the time slice performance and manipulating tasks to ensure fault tolerance and real-time performance.

Efforts are currently underway to define and standardize the interfaces between an operating system and the Ada language. AJPO is working on defining the run-time interface to the OS through the Ada Real-time Environment Working Group (ARTWEG). The Army is currently working on an IEEE standard (P-100 3.5) that will define the interfaces between POSIX and Ada. Portable Operating System for Unix (POSIX) hopes to standardize the interface between an application and an operating system. POSIX currently applies to the UNIX, VMS, and OS/2 operating systems.

5.3.3 Ada Programming Support Environments

Ada programming support environments provide the tools and rules that programmers will use during software development and test. The SSE contractor is providing the APSE that will be utilized by the SSTF. The SSTF is just one of the many target users of the SSE tools and rules, and as such, the special APSE needs for simulation application development may be overlooked. The following discussion is intended to identify some of the issues surrounding APSE usage for real-time simulation software development.

5.3.3.1 Simulation Oriented CASE

Many Ada programming support environments exist, each offering different types of CASE tools. Industry standard CASE interfaces, methodologies, and information formats do not yet exist. The National Institute of Standards & Technology is currently working to define a government standard for an Integrated Software Engineering Environment. This standard is intended to promote commonality among the various CASE tools oriented toward embedded system software development. Many of the special needs of real-time simulation are not addressed by current CASE tools. CASE tools are needed for simulation that allow the software engineer to model and validate

designs based on functional and physical real-time requirements. To provide this capability, CASE tools need to incorporate real-time oriented modeling notations and timing flow considerations that address distributed, parallel system operations. Without these important real-time features, CASE usage will not offer the expected productivity benefits, and may actually impede the real-time software development process.

5.3.3.2 Real-time Debugging Tools

The APSE environment must provide for the capability to quickly debug and correct real-time simulation errors and run-time problems. The real-time debugger should not be intrusive, in that some APSE debuggers can potentially alter the behavior of real-time simulation programs. In addition to monitoring program performance symbolically (i.e. by symbol names rather than machine addresses), the debugger should be capable of providing graphic information to the user for memory analysis, symbol data logging, and processor load balancing. These features are especially critical for multiprocessor environments that are tightly coupled through interconnected memory.

Ada exception handling constructs are very limited due to the small number of predefined exceptions identified and the wide range of conditions that cause these exceptions to be invoked. These limitations reinforce the need for powerful run-time debugging tools.

5.4 Processing Technologies

In addition to performing core simulation computing tasks, the SSTF computer complex will need the flexibility of accommodating many different types of processing technologies. Training system effectiveness will stem from the SSTF's ability to create an artificial reality that is convincing and effective for transferring ground based lessons and experiences to on-orbit operations. To accomplish this goal, complex space station embedded systems incorporating advanced automation oriented technologies will need to be provided for in the SSTF design. As discussed previously in Section 4.0, embedded systems featuring artificial intelligence, adaptive control, or machine vision may need to be accommodated through either stimulation of flight equivalent hardware executing the embedded system software, emulation of the embedded system with commercially available special purpose processors, or functional simulation of the embedded system within the host simulation computers. The following sections discuss at a high level the technical issues associated with these special purpose processing technologies and the impact of their incorporation in the SSTF computer complex concept.

5.4.1 Symbolic Computing

Symbolic computing, used heavily within the Artificial Intelligence (AI) community, is based on data-driven programming which embraces data flow machine type architectures. Symbolic programs tend to be heterogeneous and diverse, involving a variety of mechanisms and conceptual tasks within a single program. For example, the manager of an autonomous space vehicle will perform a variety of tasks such as hierarchical classification, signal interpretation, hypothesis formation, matching, and logical inference in addition to conventional numerical tasks.

The large and complex problems typically addressed by symbolic computing utilize these unique techniques to achieve the needed levels of structure and abstraction to make the problems manageable. The knowledge representations they use require large and uniform address space within the computer. The computing style also tends to create large numbers of temporary data structures. This means that the system must support garbage collection, the process of reclaiming unused storage at a rate rapid enough to always provide for free, symmetrical storage. While these applications can be developed for traditional machine architectures, significant performance improvements can be achieved by using unique machine architectures. Symbolic computers view each computer word

as containing a data field and one or more tag fields. The tag fields are used to identify object type, to delineate their extent in memory, and to reclaim unused storage efficiently. Symbolic computers are usually microcoded to accommodate the tag processing task, but otherwise resemble conventional stack-oriented computers.

5.4.2 Vision Computing

Visual image processing avionics, otherwise known as "visionics" systems, provide an on board means of sensing, processing, and analyzing real-time imagery that is generated by a vision system. Visionics systems allow man and machines to perceive objects and spatial relationships that exist in the vision system's field of regard. Vision computing systems are typically high speed numeric processors that operate on vectorized data sets describing the intensity/color of individual picture elements (pixels) that make up the sensed image. Real-time image processing calls for the visionics system to accommodate video rate communication speeds. For TV displays, this rate is almost 63 Mb/sec ($512h \times 512v \times 8 \text{ bits} \times 30 \text{ Hz}$). Current visionics systems typically sense and process around 2 million pixels per second ($256h \times 256v \times 30\text{Hz}$). The sensor image is then reformatted to match the operator (TV) display characteristics. Processing all pixels at real-time speeds is a tremendous computing task. Visionics systems minimize the need for computing power by performing operations only on a designated area of interest within the field of view. For example, an algorithm requiring 30 instructions per pixel can be performed at video speeds on a 50×50 pixel area of interest with a 2.5 Mips class machine ($50h \times 50v \times 30\text{Hz} \times 30 \text{ instr.} = 2.25 \text{ Mips}$). The SSTF host simulation computer complex must be capable of providing the functionality that on board visionics systems provide for space station operations. Typically, this functionality can be provided for simulation through the use of dedicated image processing equipment capable of capturing and analyzing visual scene data from a digital image generator (DIG) in real-time. Many of these systems are available that communicate with host computing systems via standard bus interfaces such as VME.

5.5 Visual System Technologies

The SSTF computer complex is also required to support synthetic image generation equipment that will provide dynamic out the window visual imagery and correlated sensor video for a variety of simulated space systems that require operator interaction. The interfaces between these visual system devices and the host simulation computer complex should be very flexible and adaptable to ensure that the SSTF computer resources are utilized in the most efficient manner to support stand-alone, combined, integrated, and joint integrated training sessions. The following discussion focuses on two aspects of visual system integration within the SSTF, digital image generation and digital image processing. Specific technical issues that must be considered in planning for the integration of these visual system technologies into the SSTF concept are presented at a high level.

5.5.1 Digital Image Generation

DIG systems are now available that support multiple concurrent eyepoint access to a common database. This is a necessary capability to ensure cue correlation, especially during multi-crew training scenarios. Multiple eyepoint access to a common database is required to support training activities such as handing objects back and forth between independently controlled, cooperating manipulators; as in passing a simulated object between the SMTF and the SSTF. The key DIG feature for consideration is database compatibility. Multiple copies of scene databases to support varying fidelity viewing requirements is not desirable due to the additional operations support required and the potential for trainer-to-trainer correlation problems. Ideally, the image generation equipment utilized within the SSTF should be capable of sharing a common database format by either preprocessing the common database during mission generation or by constructing the necessary scene content at run-time based on embedded parametric data.

A major DIG consideration is the ability of the DIG to correlate with CAD/CAE based models as discussed in Section 4.0. Research needs to be pursued in the compatibility issues associated with commercial graphics descriptors and visual scene database descriptors commonly used within the simulation industry. Figure 5-10 illustrates the critical role that the visual system plays in providing an integrated simulation environment for robotics training. Other considerations deal with the ability of the visual system to feed back ray tracing data for robotic end effector range and contact sensors. When given a ray location and orientation within the database, the visual system traces the ray to identify the point at which the ray intersects a database object. Current flight simulator visual systems provide this range feedback capability in a limited form (around 30 range requests per second) for radar altimeter readings and weapon impact positioning. The demand for these services within the SSTF will be much greater due to the need to simulate several sensors on multiple manipulator arms.

The host simulation computer configuration should also be flexible in its ability to drive the visual system based on selected training mode (i.e. stand-alone, combined, etc.). For reliability and growth aspects, it would be desirable for the visual eyepoint interface to be provided by any of the host simulation computing nodes, rather than relying upon a dedicated node for visual system interfacing. This multi-channel approach would allow the computing resources to be flexibly allocated as a function of training mode requirements, as opposed to being allocated according to fixed hardware dependencies.

5.5.2 Digital Image Processing

Digital image processing technology provides for field- and frame-based analysis of video format images. Digital image processing is a central component of robotic machine perception systems that analyze and act on the robot task space. The machine perception system searches for recognizable objects within the sensor field of view, tracks operator designated objects, and provides image based navigation signals for manipulator path dynamics. Integration of digital image processing technologies into the SSTF simulation environment should be anticipated to support the training requirements for robotic manipulator operation. The SSTF host simulation computer concept should be capable of accommodating this technology to provide for adequate robotic systems training.

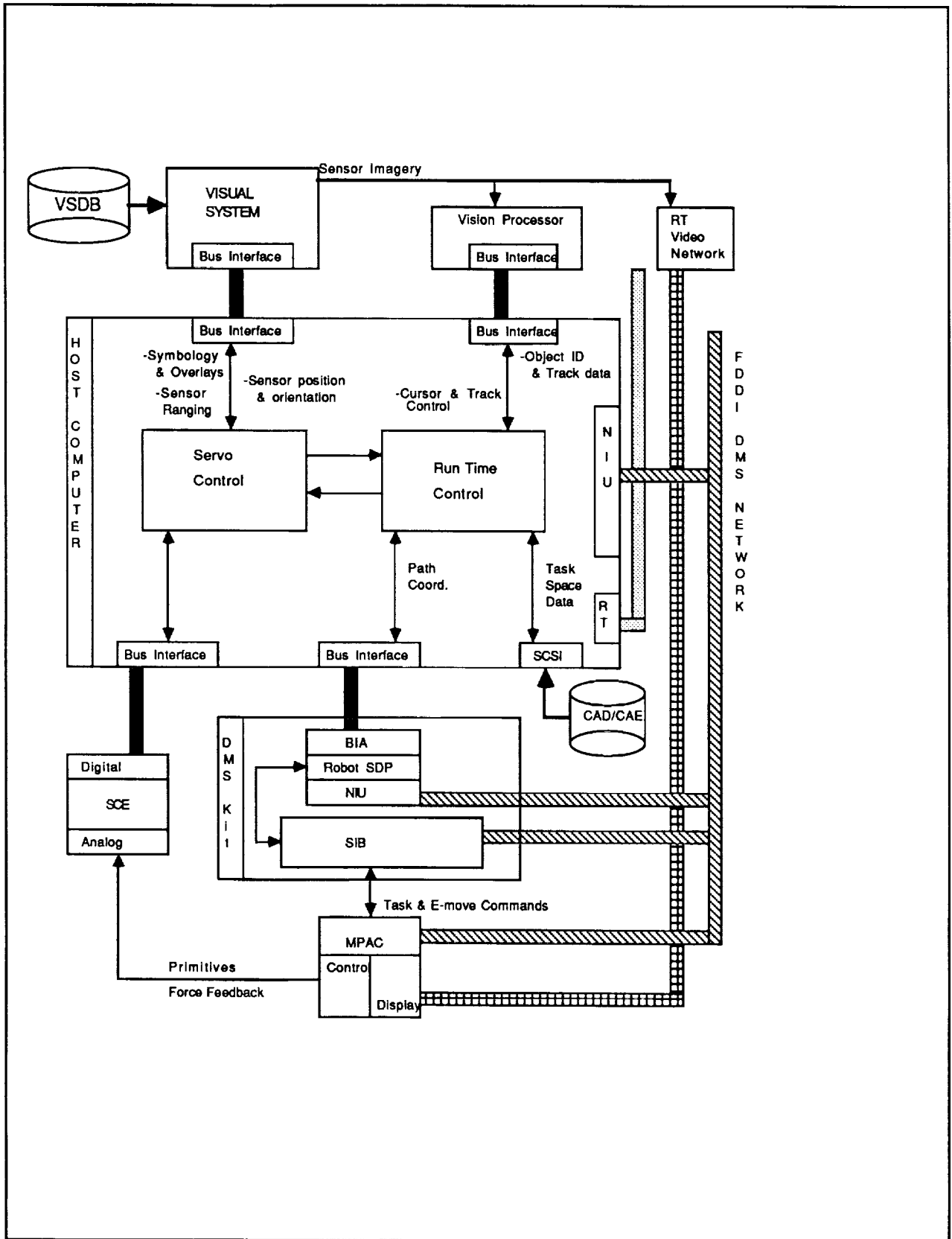


Figure 5-10
Robotics/Visual System Integration

6.0 Concept Objectives and Technology Assessment

Many of the state-of-the-art simulation technologies previously discussed hold promise for solving many of the simulation challenges discussed in Section 4.0. Application of these technologies to manned space flight simulation and training will better enable NASA to achieve established goals and objectives concerning the operation of SSTF resources. In assessing the feasibility for incorporating these technologies in the SSTF concept, many issues must be addressed concerning embedded system simulation and alternative host simulation computer configurations. These issues are presented below.

6.1 SSTF Computing Objectives

The SSTF host simulation computer complex must be capable of accommodating all of the computing needs required to provide accurate, near real-world training scenarios. The skills and lessons learned in the SSTF should be directly transferable to space flight operations, with minimal re-learning necessary to overcome ground based training deficiencies. Achievement of this goal requires that the host computer complex be capable of interfacing with a variety of special purpose processors that are dedicated to performing a specific function such as a knowledge based system or visionics system. In many cases, the appropriate level of system functionality may only be achieved by stimulating or emulating flight equivalent processors or special purpose processors executing flight software based algorithms. Special purpose processor interfacing will most likely be through a MIL-STD-1553 type mux bus or VME type interface. The host computer complex should offer many possible means of accommodating these interfaces, rather than relying upon a dedicated I/O chassis for all external communications. Distributed and open I/O interfacing will allow the computer resources to be utilized in the most effective manner for accomplishing simulation objectives, without having to resort to fixed, dedicated resources for specific training situations.

The host simulation computer resources must also be capable of quick reconfiguration in terms of run-time software loads and mission unique payload accommodation that may call for special purpose processing. The processing environment must be easily configurable between concurrent stand-alone and combined/integrated training activities.

Similarly, the host simulation computer resources should be flexibly configured to support the necessary training requirements associated with each training mode (i.e. stand-alone, combined, integrated, and joint integrated). These training requirements may involve multiple DIG visual channels to support various training scenarios involving simultaneous, independent systems operation. To support this mix and match capability among trainer stations for part-task training, it is recommended that the host computer resources be flexible in the ability to drive visual interface channels from any computing node.

The SSTF concept calls for variable embedded system fidelity based on the selected training mode. Part-task, stand-alone training sessions call for medium fidelity, single system oriented response, while combined full-task training sessions call for high fidelity, full systems interaction across all space station systems. To provide for these capabilities, the multi-computer host simulation resources should be easily reconfigurable to support multiple concurrent part-task sessions or combined full-tasks sessions. Conceptual simulation technologies may prove useful in determining system designs that support multiple types of training sessions. One approach is to utilize common interfaces (Ada packages) between simulation processes (i.e. GN&C avionics, TCS sensors, etc.) but utilize different programs (Ada bodies) depending upon the specific training scenario that is selected. A high fidelity program would offer full systems signature performance while a low fidelity program could support procedures training for part-task sessions. When a particular training mode and trainer configuration is selected, the appropriate program (Ada body) would be linked with the standard interface (Ada package) and the simulation process could then be installed on the

appropriate computing node within the host simulation computer complex. This approach would allow the computing resources to be dynamically configured and efficiently utilized as a function of the training requirements. For example, in stand-alone training mode, each node of the multi-computer host could be automatically configured in terms of connectivity and software load to support the selected training scenario. Selection of combined training mode could then be accomplished by automatically reconfiguring the nodes of the multi-computer host into a single node system, with the appropriate connectivity switching and software load to support full systems interaction.

6.2 Architecture Assessment

Issues associated with the architectural technologies discussed in Section 5.0 must be addressed when evaluating possible host simulation computer configurations. The real-time computer resources must be capable of not only providing an efficient Ada run-time environment, but must also be capable of accommodating special purpose processors, visual system interfaces, inter-facility real-time data communications, and be configurable into at least two independent processing nodes to support concurrent part-task training sessions. In addition, the computer resources must offer efficient economies of scale in terms of processing power and communications bandwidth expandability. Provisions should be available for doubling these capabilities without serious impact to the initial hardware/software configuration.

These various needs may best be fulfilled by a distributed processing simulation architecture as opposed to a monolithic, traditional mainframe type architecture. A distributed, multi-node architecture offers significant advantages in terms of reconfigurability, scalability, redundancy, and fault tolerance. Distributed architectures for real-time simulation applications are now being supported by the computer industry that offer unprecedented computing power and communications flexibility, and they provide an attractive solution for the host simulation computing needs of the SSTF.

Although a distributed, multi-processor architecture is clearly desirable, there are still many choices that must be made in configuring a distributed system to meet SSTF functional goals and operational objectives. Processor integration and operating system software are two crucial areas that will, in large part, determine the ability of the SSTF student training environment to meet design goals for life cycle cost minimization, evolution and growth, reconfiguration, and upward technology compatibility.

The method by which processors are integrated to form computing nodes is critical in the ability of the computer to provide scalable real-time simulation services. Loosely coupled networking methods allow communications over long distances between dissimilar processors, but do not as yet adequately support time critical, man-in-the-loop applications with dynamic system configurations and unknown message processing demands. The SCRAMNet method provides the best of both the loosely coupled and tightly coupled approaches, but currently is limited in the shared memory area size that can be supported among several network nodes. Other tightly coupled memory linkage methods are designed to support the needs of real-time simulation, but limit system configuration flexibility due to their proprietary designs and lack of standardization. But the advantages of a tightly coupled distributed computing system far outweigh the disadvantages. Distributed computing systems offering mainframe class performance are available today that are based on commercially supported CPUs with a clear migration path, high speed direct memory communications, and real-time operating system features. It is highly recommended that these systems be investigated further for application in advanced manned space flight simulation systems such as the SSTF.

The operating system software that manages the real-time computational performance is another critical area in determining the computer system's suitability for scalable real-time simulation applications. Many proprietary real-time operating systems are commercially available that have successfully proven that Ada can be used in real-time applications. Anchoring on one of these proprietary systems that have large amounts of machine-dependent and non-portable code limits flexibility and does not answer the call to reduce the use of custom simulation solutions. The industry efforts under way to standardize the way Ada is used in real-time applications should be closely followed and evaluated in conjunction with the evaluation of architectures.

Appendix A

References

REFERENCES

- 1) Space Station Training Facility, MSD Project Review, May 4, 1989
- 2) SSTF Contrast, 1986 Cost Commit vs. Current Concept, May 11, 1989
- 3) SSTF Conceptual Design Document, First Draft, August 1988
- 4) JSC 32060, SSTF Level A Simulation Requirements, June 30, 1988
- 5) SSTF Development Plan, Delivery Summary, July 27, 1989
- 6A) SSTF Development Plan, Appendix A, SSTF Model Requirements
- 6B) SSTF Development Plan, Appendix B, SSTF SLOC Requirements
- 7) Memo from Doug Morris, April 24, 1989; Simulation Model Development Plans
- 8) Memo from Doug Morris, Attachment A; Special Requirements for Simulation Models Used in Training
- 9) OADP Specification and Statement of Work, February 22, 1989
- 10) SSTF DMS Kit Outfitting, Herbert Long, June 1989
- 11) Data Interfaces to the Space Station Information System, Richard Carper, Fritz Schultz, IEEE 1988
- 12) The Evolution of the Mission Control Center, Michael Kearney, Proceedings of the IEEE Vol. 75, No. 3., March 1987
- 13) Space Station Data Management System Architecture, William Mallary, Virginia Whitelaw, Proceedings of the IEEE Vol. 75, No. 3, March 1987
- 14) An Overview of Space Station Operations, James Walker, Robert Anderson, SP-687, Society of Automotive Engineers, October 16, 1986
- 15) An Environment for the Integration and Test of the Space Station Distributed Avionics Systems, Thomas Barry, Terrance Scheffer, IEEE AES Magazine, November 1988
- 16) Space Station Data Management System: A Common GSE Test Interface for Systems Testing and Verification, Pedro Martinez and Kevin Dunn, Proceedings of the IEEE, Vol. 75, No. 3, March 1987
- 17) NASA Space Station Effort Drives Expert Systems Research at Johnson, AW&ST, April 22, 1985
- 18) Testing and Validation in Artificial Intelligence Programming, R.B. Purves et al, SPIE Vol. 851 Space Station Automation III, 1987

- 19) System Autonomy Hooks and Scars for Space Station, S.A. Starks and D.W. Elizandro, SPIE Vol. 851 Space Station Automation III, 1987
- 20) Knowledge-Based Simulation for Aerospace Systems, Ralph W. Will et al, Machine Intelligence and Autonomy for Aerospace Systems, 1988
- 21) Machine Intelligence and Crew-Vehicle Interfaces, Robert G. Eggleston, Machine Intelligence and Autonomy for Aerospace Systems, 1988
- 22) Vision Sensing Techniques in Aeronautics and Astronautics, E.L. Hall, Machine Intelligence and Autonomy for Aerospace Systems, 1988
- 23) Supervisory Control of Telerobots in Space, Thomas Sheridan, Machine Intelligence and Autonomy for Aerospace Systems, 1988
- 24) Manned Spacecraft Automation and Robotics, Jon D. Erickson, Proceedings of the IEEE, Vol. 75, No. 3, March 1987
- 25) Telerobotics for Space Applications, W.S. Otaguro et al, IEEE AES Magazine, November 1988
- 26) NASA Autonomy and Robotics Technology Program, Lee B. Holcomb, Melvin D. Montemerlo, IEEE AES Magazine, April 1987
- 27) Space Robotics in the '90s, Carl F. Ruoff, Aerospace America, August 1989.
- 28) West Germany's First Space Robot, Gerd Hirzinger, Aerospace America, August 1989.
- 29) Ground Operations of Space-Based Telerobots Will Enhance Productivity, Wayne R. Schober, IEEE NAECON 1988
- 30) The Flight Telerobotic Servicer Project and Systems Overview, Harry G. McCain and James F. Andary, IEEE NAECON, 1988
- 31) Servo Level Algorithms for the NASREM Telerobot Control System Architecture, SPIE Vol. 851 Space Station Automation III (1987)
- 32) Orbital Navigation, Docking, and Obstacle Avoidance, SPIE Vol. 851 Space Station Automation III (1987)
- 33) Automation and Robotics and Related Technology Issues for Space Station Servicing, Helmut P. Cline, SPIE Vol. 851 Space Station Automation III 1987
- 34) Telerobot Experiment Concepts in Space, Lyle M. Jenkins, SPIE Vol. 851 Space Station Automation III (1987)
- 35) Architecture For Dynamic Task Allocation in a Man-Robot Symbiotic System, Lynne E. Parker and Francois G. Pin, SPIE Vol. 851 Space Station Automation III (1987)
- 36) Telerobot Operator Control Station Requirements, Edwin Kan, SOAR 1988

- 37) Time Delayed Operation of a Telerobot via Geosynchronous Relay, Brian Wilcox, SOAR 1988
- 38) CAD Model Based Vision for Space Applications, Linda Shapiro, SOAR 1988
- 39) Spaceborne VHSIC Multiprocessor System for AI Applications, Henry Lum, Howard Shrobe, John Aspinall, SOAR 1988
- 40) Automation of the Space Station Core Module Power Management and Distribution System
- 41) Real-time Simulation for Space Station, Robert St. John et al, Proceedings of the IEEE, March 1987
- 42) Shuttle Mission Training Facility Upgrade, Kurt Frevert, Riley McCafferty, ITEC, November 30, 1987
- 43) The Shuttle Mission Simulator--From Design Concepts to an Operational Training Device, Pete Sivillo, Riley McCafferty, ITEC, November 19, 1985
- 44) A Computer Systems Upgrade for the Shuttle Mission Training Facility, Ankur Hajare, Patrick Brown, AIAA 1988
- 45) Reducing the Risks of Using Ada Onboard the Space Station, Terry Humphrey, IEEE AES Magazine, November 1988
- 46) Ada Adoption Handbook, John Foreman, John Goodenough, CMU/SEI-87-TR-9, May 1987
- 47) Experience in Implementing an Ada Real-Time Program for Flight Simulation Operation, Grecco Myren, ITEC, 1987
- 48) Lessons Learned from the Ada Simulator Evaluation Program, Jerry Hendrix, ITEC, 1987
- 49) Hybrid Ada/Fortran Systems for Flight Simulation, Vincent Rich, ITEC 1987
- 50) What Every Good Manager Should Know about Ada, Judy Bamberger, NAECON 1987
- 51) An Ada Language Software Architecture for Distributed Simulation Systems, Andrew Fuller, IEEE NAECON 1988
- 52) Strategic Computing: A Status Report, IEEE Spectrum, April 1987
- 53) A Unified Approach to Real-Time Systems Integration, R & D, March 21, 1989
- 54) Computer System Hardware, Steve Seidensticker, Flight Simulation Update, 1988
- 55) Parallel Compilers Are Coming Just in Time, Electronics, November 1988
- 56) Focusing Real-Time Systems Analysis on User Operations, Michael Deutsch, IEEE Software, 1988
- 57) A Comparison of 12 Parallel Fortran Dialects, Alan Karp, Robert Babb, IEEE Software, September 1988

- 58) Programming a Hypercube Multicomputer, Sanjay Ranka et al, IEEE Software, September 1988
- 59) Modular, Functionally Distributed Microprocessor-Based Simulation, Michael Ash, ITEC 1985
- 60) Distributed Processing for Complex Simulators, David Parkinson, ITEC 1985
- 61) A Parallel Processor Alternative to the Modular Simulation Architecture, Edward Kulakowski, David Kramer, ITEC 1986
- 62) Multiple Microcomputer Architectures, Charles Pope et al, ITEC 1986
- 63) Real-Time Simulators: Dealing with their Growing Complexity, Steve Seidensticker, ITEC 1985
- 64) Modular Simulators: Is a Local Area Network Sufficient?, Donald L. Johnston, NAECON 1985
- 65) Modular Simulators: A Comprehensive Unified Architecture, Steve Seidensticker, NAECON 1985
- 66) Space Station Training Facility Concept of Operations, JSC 32034, August 4, 1988
- 67) Distributed Ada Real-Time Kernel, Judy Bamberger and Roger Van Scoy, IEEE 1988.
- 68) International Workshop on Real-Time Ada Issues, Volume VII, Number 6, Special Interest Group on Ada (SIGAda), June 1988