Paul Schneck
Guest Editor

Architecture of the Space Shuttle Primary Avionics Software System

PASS, perhaps the most complex flight computer program ever developed, epitomizes the benefits to be gained by establishing a well-structured system architecture at the front end of the development process

GENE D. CARLOW

Many lessons have been learned over the last 20 years about the development of large, complex systems. Structured programming techniques used for implementation have led to improved software quality and maintainability. Compilers and other tools have been developed to improve productivity and reliability. Management procedures have evolved that improved the control and stability of the development process. However, as effective as they all are, these tools and techniques alone cannot prevent or overcome the problems that will exist if a well-structured system architecture has not been established on the front end of the development process.

Such an architecture, as described in this article, has been developed by the IBM Federal Systems Division for NASA's Space Shuttle Orbiter Primary Avionics Software System (PASS). PASS performs a critical role in virtually all operational aspects of the Space Shuttle Orbiter (Figure 1). It is by far the most complex flight computer program ever developed. It employs state-of-the-art multiple computer redundancy management concepts and incorporates a multitude of functions required to support Shuttle operations on the ground and in flight.

Influenced by software engineering concepts evolved in recent years, the software organization responsible for development of this system has implemented a design architecture and executed plans and procedures that resulted in the top-down integration and delivery of incremental software releases. User schedule and capability requirements have been satisfied and numerous requirements modifications have been accommodated in a very controlled and timely manner.

The present tense of this article, published in 1980–1981, has been retained in republication.

PASS ARCHITECTURAL DRIVERS

The PASS is central to all of the Space Shuttle Orbiter Avionics System functions, including vehicle flight control (Figure 2). Many factors, in addition to the critical timing and redundancy management requirements of the avionic system design, have influenced the architecture of the PASS. These include the Data Processing System (DPS)/IBM AP-101 General Purpose Computer (GPC) design and memory/CPU constraints; the multicomputer redundancy management and synchronization; the operational sequencing/mode control and man/machine interface requirements; the applications functional and performance requirements; and the requirement for design modularity and modification flexibility necessitated by the R&D nature of the Shuttle Program. Added to this are the consideration and effect of literally hundreds of people participating in the requirements definition, development, verification, certification, utilization, and support of the PASS.

The successful accommodation of these many diverse and sometimes conflicting requirements and considerations is dependent on the existence of a sound software system architecture. The PASS architecture has met that challenge and survived the test of time for the past several years.

OPERATIONAL STRUCTURE

The PASS architectural structure reflects a blend of operational, reliability, and functional requirements plus several physical constraints. The main memory capacity of the AP-101 (106K 32-bit words) is not sufficient to contain all of the software (500K) required to satisfy all of the PASS requirements. The mass memory is a serial rather than direct-access storage device. Reliability considerations dictate that the software required

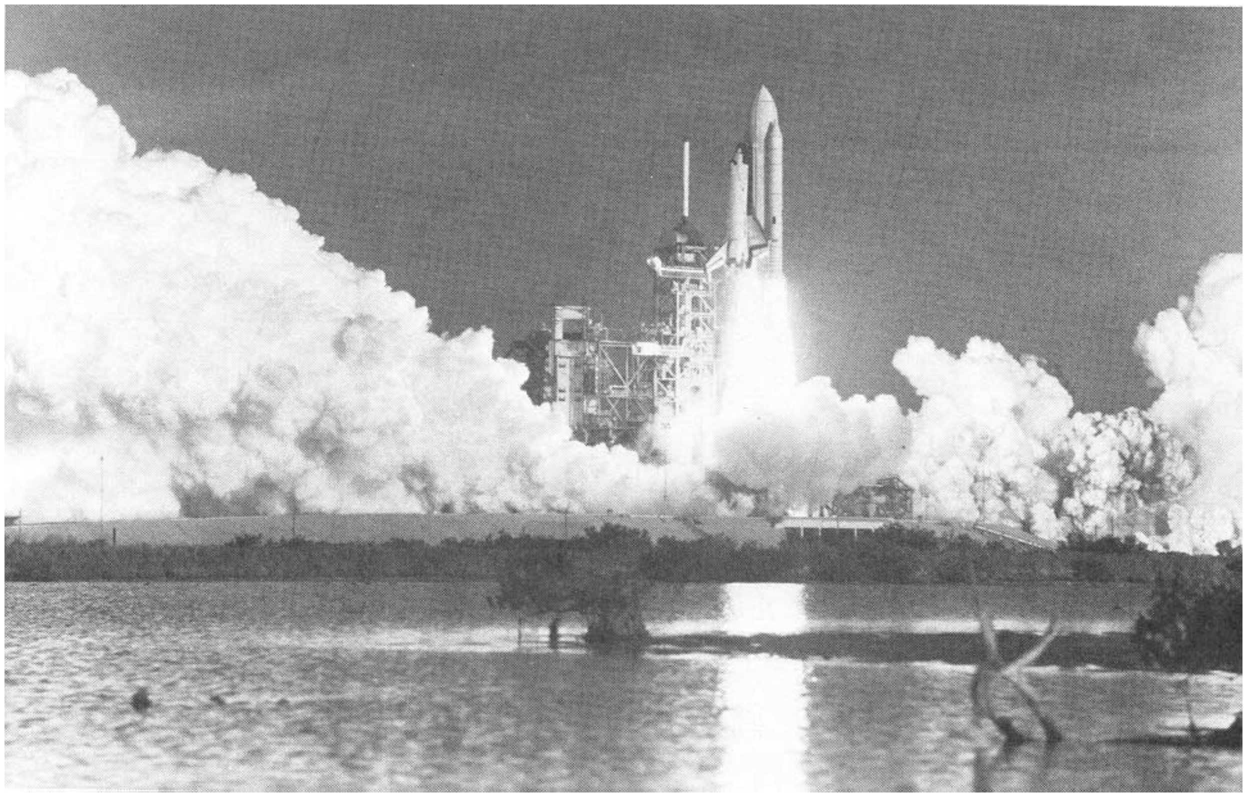


FIGURE 1. The Launch of Columbia Provided One of the First Critical Tests of PASS

to support a single critical mission phase such as ascent must be totally resident in each redundant GPC main memory throughout that phase. These constraints have been addressed with an operational structure that is based on the combination of mission phases and major application functional requirements. The result has been the segmentation of the PASS into eight different phase/function combinations (Figure 3), each of which is identified with a unique operational sequence (OPS) designation. The software required for each OPS is loaded into the GPC main memory from the mass memory at the initiation of the OPS. Except for a few specific nonmission critical situations, such as the loading of a display format during on-orbit coast periods, transfers between mass memory and a GPC main memory do not occur during the execution of an OPS.

The PASS is stored on the mass memory in a manner that minimizes the amount of code and data that must be transferred to main memory when a new OPS is initiated. The main memory load for each OPS has three parts (Figure 4): the resident or systems software, which contains code and data common to all OPS loads; the major function base, which contains code and data common to a major applications function used in more than one OPS load; and the OPS overlay, which contains the applications code and data unique to an OPS load. Main memory loading occurs during the transition from one OPS to another in response to a major application function switch selection and keyboard entry from the crew. The contents of the OPS in progress determine which of the three parts must be loaded for support of the new OPS. At best, just the OPS overlay will have to be loaded.

MAN/MACHINE INTERFACE

The operational environment within which a software system will be utilized in terms of man/machine interface must be considered when the architecture is established. If this is not done, a "user beware" situation will evolve and result in redesign and modification after the user is exposed to the product. The PASS man/machine interface is structured to accommodate a knowledgeable user, with minimum time and effort required to communicate with the computers to monitor and control a very complex Orbiter avionic system. This structure is implemented as a substructure of the OPS and consists of major modes, specialist functions, and display functions (Figure 5).

The major mode provides the capability to segment the primary processing within an OPS into major steps or sequences. Each OPS has one or more modes. Major modes are further substructured into blocks that are linked to CRT display pages so as to establish an orderly sequence for the crew to communicate and maintain control of the software. Sequencing from one mode/block to another and the processing performed

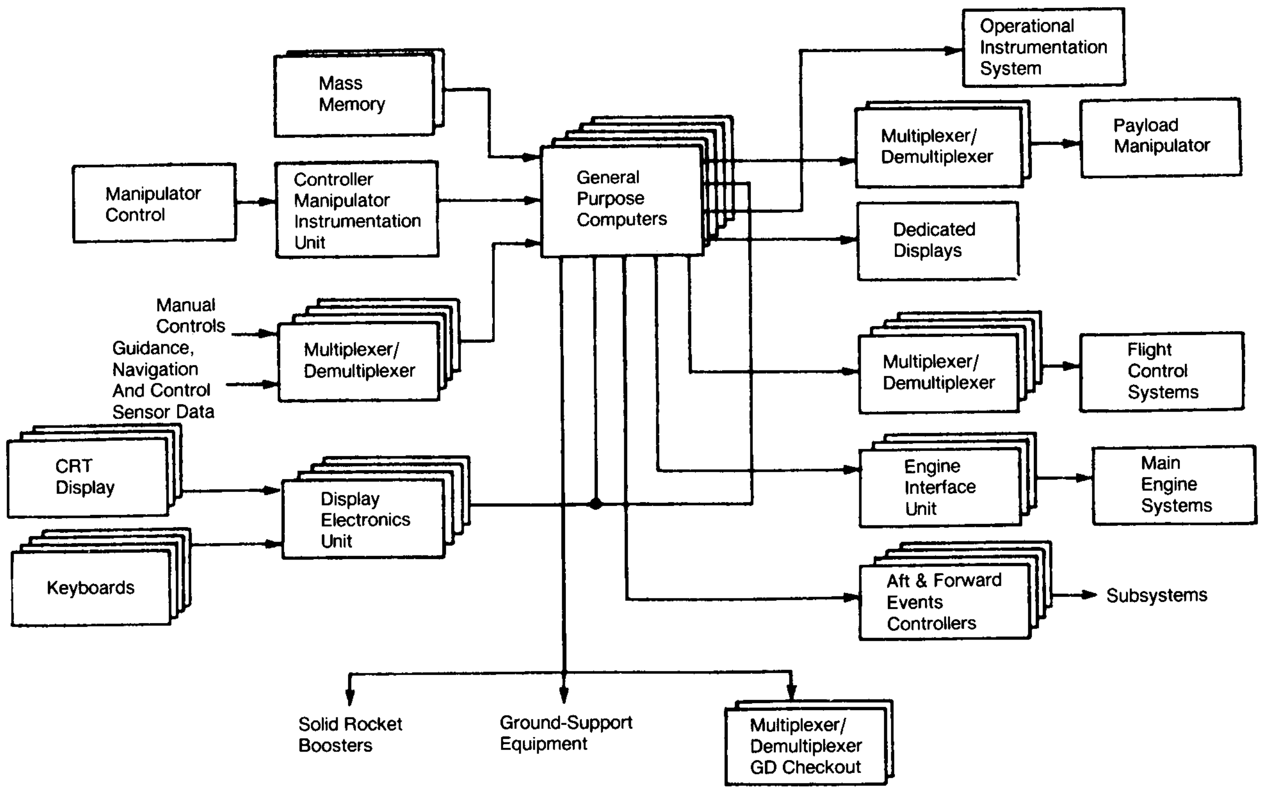

Mass Memory
Manipulator Control
Controller Manipulator Instrumentation Unit
General Purpose Computers
Operational Instrumentation System
Multiplexer/ Demultiplexer
Payload Manipulator
Dedicated Displays
Manual Controls
Guidance, Navigation And Control Sensor Data
Multiplexer/ Demultiplexer
Flight Control Systems
CRT Display
Display Electronics Unit
Engine Interface Unit
Main Engine Systems
Keyboards
Aft & Forward Events Controllers
Subsystems
Solid Rocket Boosters
Ground-Support Equipment
Multiplexer/ Demultiplexer GD Checkout

FIGURE 2. Shuttle Orbiter Avionic System (The GPCs are the central point of control within the Orbiter avionics system.)

within are initiated either by keyboard entry from the crew or, in some cases, automatically as the result of a specific event or condition detected by the software. At least one major mode will be entered any time an OPS is initiated.

The second element of the OPS substructure is the specialist function (SPEC). Unlike the major mode, the SPEC is initiated within an OPS only upon a keyboard entry from the crew. Once initiated, the SPEC executes independent of and concurrent with other processing within the OPS. The SPEC also has a substructure of blocks, which are linked to CRT displays and establish the valid keyboard entry options that are available to the crew for controlling the operation and/or monitoring the processing results. Whereas major modes are used to accomplish the primary functions within an OPS, the SPEC is used for secondary or background functions.

The third element of the OPS substructure is the display function (DISP). The major difference between the DISP and major mode or SPEC is that no processing function is initiated by the DISP. It is used only to monitor the processing results of a SPEC or major mode function.

A key feature of the software architecture that facilitates the implementation of the operational structure is the control segment. Control segments consist of a series of standardized logic blocks that establish the structure of an OPS, major mode, SPEC, or DISP. The sequencing of all processing required within an OPS, major mode or SPEC, and the associated CRT displays and keyboard entry options is defined in the design of a control segment. A library of macros called the control segment grammar has been developed to standardize and modularize the design and implementation of control segments.

SYSTEMS SOFTWARE

Flight software systems have been characterized by their small size (15K to 30K) and the limited number of functions they perform. Their software architectures reflect a synchronous design approach within which the dispatching of each application process is timed to always occur at a specific point relative to the start of an overall system cycle or loop. A relatively simple streamlined operating system or executive is used, but a lot of care must be taken in the implementation to synchronize the start and completion of the different application processes, and their associated I/O, to prevent overruns at both the process and system levels. A major benefit of this approach is repeatability; however, there is only limited flexibility to accommodate change.

Given the increased number and variety of different applications functions that the Shuttle computers are required to perform, and the anticipation that requirements would change as the Shuttle Program evolved, a nonsynchronous approach was adapted for the PASS architecture. Other basic architectural decisions included the isolation of applications processes from the external I/O and computer redundancy management.

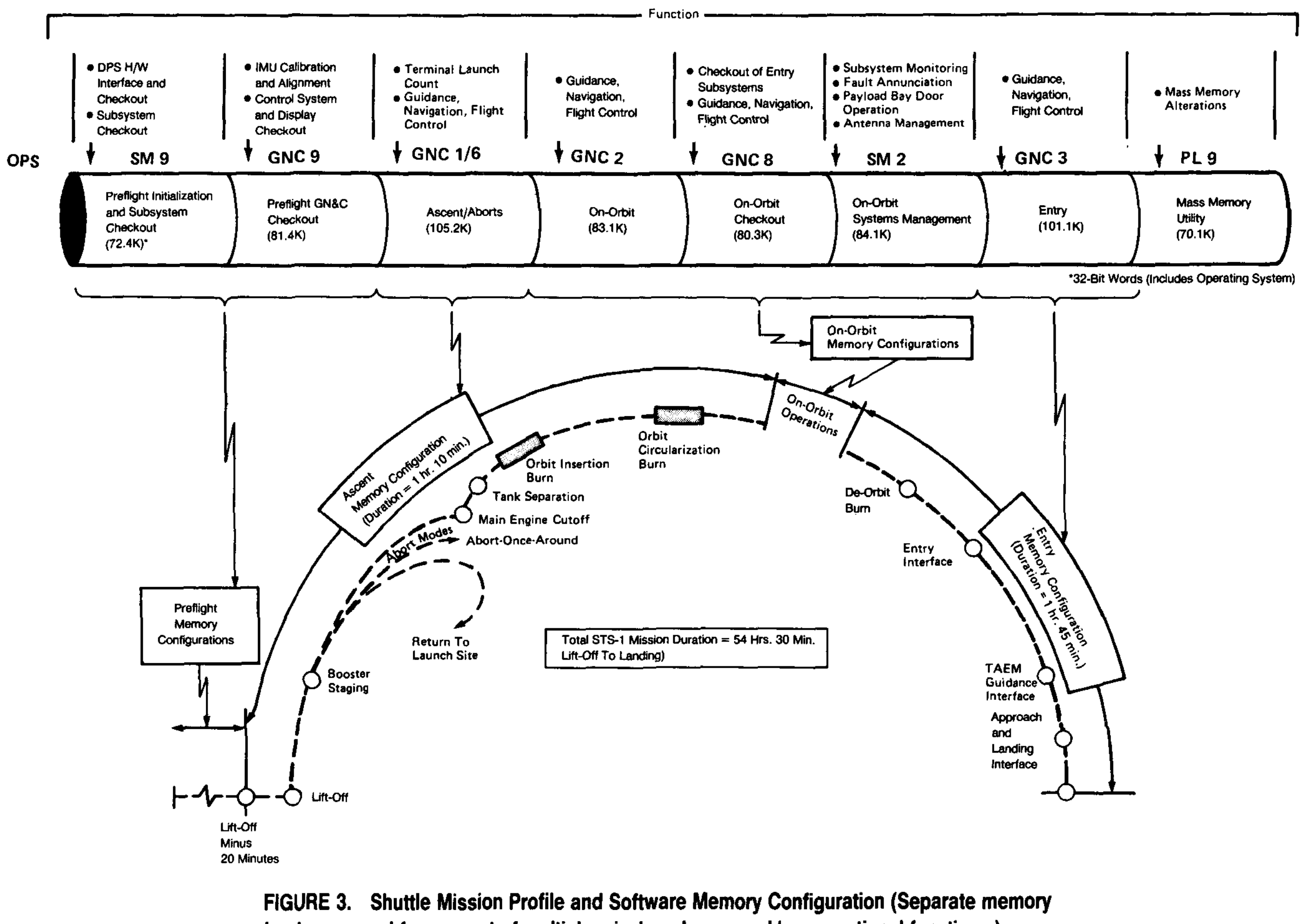
These overall system level architectural decisions are reflected in the design and implementation of the system software (Figure 6). Sequencing and control of the applications are accomplished through interfaces with the major elements of the systems software. Management and control of the GPC internal resources and external interfaces are accomplished by the Flight Computer Operating System (FCOS). Loading and initialization are performed by the System Control. Communications between the user and systems or applications software are managed by the user interface. Intercomputer communication and redundant computer synchronization are also handled by the systems software.

FLIGHT COMPUTER OPERATING SYSTEM

The large variety of jobs performed by the FCOS (Figure 7) can be grouped into three major functions. First is process management, which controls the allocation of all internal computer resources. Using a multitasking priority queue structure, it schedules and allocates CPU resources in response to requests from other systems and/or applications processes. These resource requests are made through a standard set of service interfaces (SVCs) with reference to specifications that define the time and/or event, frequency, and priorities for servicing the request. Process phasing and overall system load balancing are maintained through the management of the specifications. The processing management function ensures that the highest priority system or application process, ready with work to perform, is given the CPU resource required to accomplish it.

Similar to the process management control of the CPU resource, the input/output management function controls the allocation of the input/output processor (IOP) resources. Each GPC commands data buses assigned to it for I/O control. The IOP of each GPC contains a master sequence controller (MSC) and 24 bus control elements (BCEs). Each GPC component (CPU, MSC, and 24 BCEs) is actually an asynchronous processor communicating with the next higher and/or lower processor during I/O operations. The IOP software is considered part of the input/output management function.

The redundancy designed into the orbiter avionics



Function
• DPS H/W Interface and Checkout
• Subsystem Checkout
• IMU Calibration and Alignment
• Control System and Display Checkout
• Terminal Launch Count
• Guidance, Navigation, Flight Control
• Guidance, Navigation, Flight Control
• Checkout of Entry Subsystems
• Guidance, Navigation, Flight Control
• Subsystem Monitoring
• Fault Annunciation
• Payload Bay Door Operation
• Antenna Management
• Guidance, Navigation, Flight Control
• Mass Memory Alterations
OPS
SM 9
GNC 9
GNC 1/6
GNC 2
GNC 8
SM 2
GNC 3
PL 9
Preflight Initialization and Subsystem Checkout (72.4K)*
Preflight GN&C Checkout (81.4K)
Ascent/Aborts (105.2K)
On-Orbit (83.1K)
On-Orbit Checkout (80.3K)
On-Orbit Systems Management (84.1K)
Entry (101.1K)
Mass Memory Utility (70.1K)
*32-Bit Words (Includes Operating System)
On-Orbit Memory Configurations
Ascent Memory Configuration (Duration = 1 hr. 10 min.)
On-Orbit Operations
Orbit Insertion Burn
Orbit Circularization Burn
Tank Separation
Main Engine Cutoff
Abort Modes
Abort-Once-Around
Return To Launch Site
De-Orbit Burn
Entry Interface
Entry Memory Configuration (Duration = 1 hr. 45 min.)
Total STS-1 Mission Duration = 54 Hrs. 30 Min. Lift-Off To Landing)
Preflight Memory Configurations
Booster Staging
TAEM Guidance Interface
Approach and Landing Interface
Lift-Off
Lift-Off Minus 20 Minutes

FIGURE 3. Shuttle Mission Profile and Software Memory Configuration (Separate memory loads are used for support of multiple mission phases and/or operational functions.)

and data processing systems has resulted in an extremely complex data network (Figure 8). All communications on the data network are managed by the input/output management function, including the requirements to ensure that redundant computers receive identical data from the avionic hardware and to perform I/O fault isolation and correction. Any system or application requirement for I/O is requested, scheduled, and performed through the input/output management function based on availability of I/O channels and the priority of the request. I/O can be initiated through a standard service request (SVC) or tied to a timer interrupt to start a process execution cycle. An interface is established with the process management function to support any coordination required between I/O completion and suspended process resumption or event notification.

Loading of the GPC memories and sequencing and control of the GPC and IOP operating states, including hardware initialization and status checks, are accomplished by the DPS configuration management function. All transfers of code/data between the mass memory and GPC main memories, including program overlay and inflight mass memory modification, are performed by this function. Additionally, the configuration management function establishes the redundant computer set for support of critical mission phases.

Synchronization of multiple computers in a redundant configuration is accomplished approximately 350 times per second by a special process in conjunction with the execution of most other FCOS processes. A sync discrete pattern is transmitted to and monitored for receipt from each GPC in the redundant set. Nonreceipt of the correct pattern from another GPC within four milliseconds from the time of transmission constitutes "failure to sync" by that GPC and results in its being "voted" out of the redundant set. Management of redundant set membership is a manual action performed by the crew.

SYSTEM CONTROL

Initialization and configuration control of the DPS and associated avionic data network are performed by the system control. Several systems-level SPECS give the crew control over these functions. After loading a GPC memory with the resident systems software and synchronizing internal timers with the avionic system master timing unit (MTU) by FCOS, system control is evoked to establish an interface and primary/secondary relationship with other GPCs. Communication paths are established with the keyboard/display units and those avionic subsystems that are supported during the execution of all OPSs. A systems-level cyclic process is initiated to continuously service these interfaces and establish the 40-millisecond timing cycle that is inherent in the avionic system design.

The last step in the GPC start-up and initialization process is to establish the basis for further system reconfiguration. This is implemented in the form of a control segment. The GPC is placed in the idle OPS (OPS-0) ready to modify the configuration of the DPS and/or data network assignments, to initiate one or more systems-level SPECs, or change into another OPS in response to crew commands. Functions performed by the systems-level SPECS include resetting the MTU, initiating a memory dump, examining or changing a specific core location in memory, or changing the con-

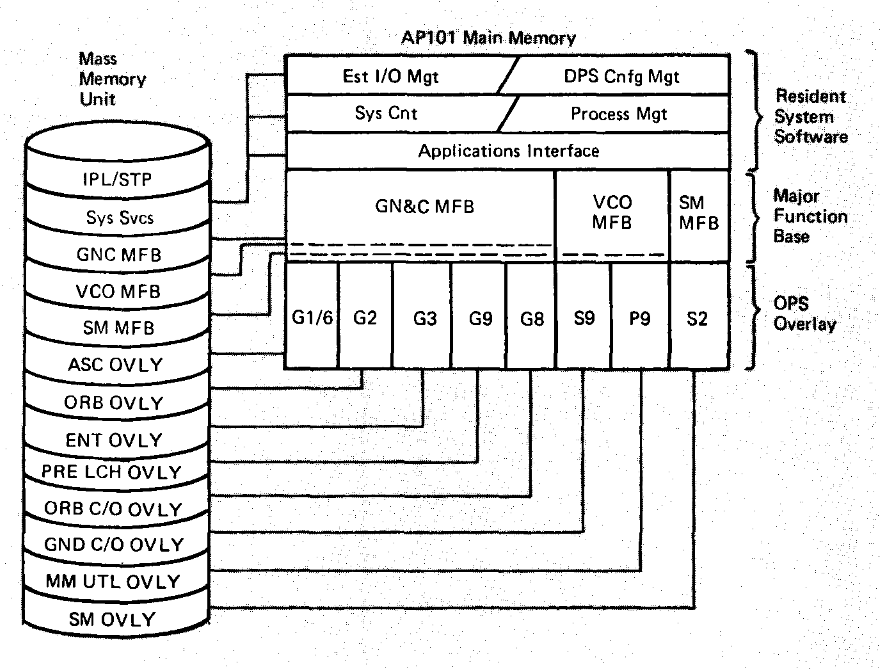


FIGURE 4. Memory Configuration Structure (Each memory load requires all three components to perform operational functions.)

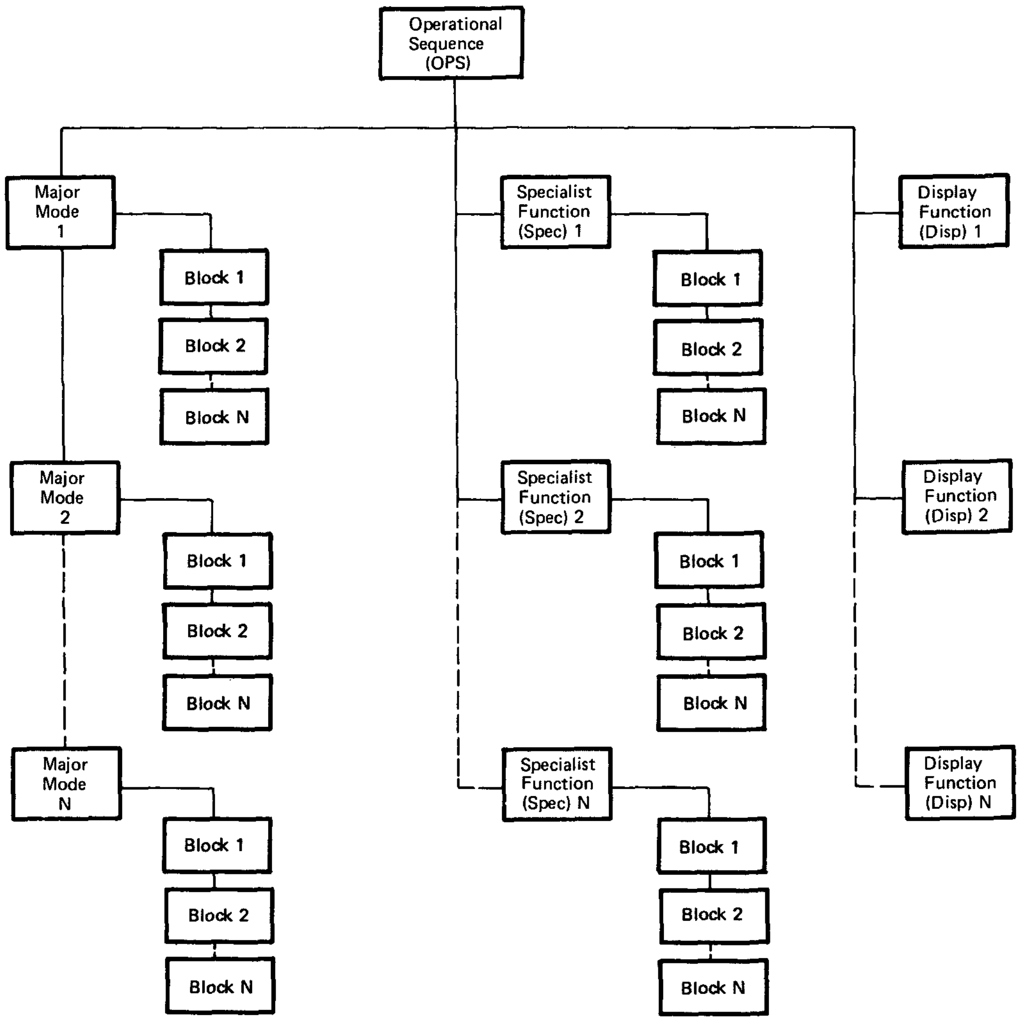

Operational Sequence (OPS)
Major Mode 1
Block 1
Block 2
Block N
Major Mode 2
Block 1
Block 2
Block N
Major Mode N
Block 1
Block 2
Block N
Specialist Function (Spec) 1
Block 1
Block 2
Block N
Specialist Function (Spec) 2
Block 1
Block 2
Block N
Specialist Function (Spec) N
Block 1
Block 2
Block N
Display Function (Disp) 1
Display Function (Disp) 2
Display Function (Disp) N

FIGURE 5. PASS Operational Sequence Structure (Using this simple three-element structure and associated function keyboard, minimum crew action is required to communicate with the computers.)

figuration of the DPS. Systems SPECs can be initiated in other OPSs as well as OPS-0.

USER INTERFACE

Support of the crew interface, keyboard entry, and display generation is accomplished with both on-board software and off-line processors. Off-line processors translate data entry and display format requirements into parametric data tables for use by the on-board software. These data tables are used to generate CRT displays and define/process valid data entries associated with a particular display.

The user interface facilitates external control of systems or applications processing as defined in a control segment. It consists of three major functions: command input processing, operations control, and output message processing. Two user interfaces in addition to the keyboard and CRTs also are supported. These are the launch data bus (LDB), used to communicate with the Launch Processing System at the Kennedy Space Center prior to vehicle lift-off, and the network signal processing (NSP) interface, used to process data and/or commands received from the Mission Control Center at the Johnson Space Center during in-flight operations. A fourth interface, the intercomputer channel (ICC), is used to maintain communications and synchronization between the GPCs.

Through the interpretation of user input commands received from the crew keyboard and process control commands included in the control segments, user interface performs the initiation, sequencing, and termination of other systems or applications processes. User input data are routed to the appropriate systems or applications process as specified in the control segment. Requests for memory overlays to support the transition from one OPS to another are passed to system control. Requests to change the DPS configuration and/or data network bus assignments are also passed to system control. Output messages and displays are generated as specified in the control segments and then are routed to the user through the appropriate external interface.

APPLICATIONS SOFTWARE

On-board software has been developed for three major applications to support ground and in-flight Shuttle operations. These three applications are guidance, navigation, and control (GN&C); vehicle systems management (SM); and vehicle checkout (VCO). Vehicle checkout provides software support for the testing, integration, and/or certification of the Orbiter avionics subsystems during vehicle preparation on the ground and in-flight orbit coast period prior to entry.

GN&C software determines vehicle position, velocity, and attitude; performs sensor redundancy management; provides the crew with displays and data entry capabilities to monitor and control the avionics subsystems; and issues the engine and/or affector commands for a mission from lift-off through touchdown and rollout.

SM monitors the performance and configuration of Orbiter and payload subsystems to detect abnormal conditions and alert the flight and ground crews via CRT displays and/or lights and alarms so they can take corrective action.

Unique functional and performance requirements for each of these applications have necessitated different design structures for each. Overall sequencing and control for each application occur via the control segments. Off-line processors are used in some instances to generate parametric data tables, which on-board software uses to produce the required results. A facility is available to change, without going through a recompile/assembly, the initial state/value of certain parameters that are tied directly to the avionics hardware and/or mission profile characteristics.

GUIDANCE, NAVIGATION, AND CONTROL

The GN&C is a cyclic closed loop application, which performs a wide variety of functions with extremely tight timing and phasing relationships. GN&C functions, of which there are approximately 200 called principal functions, are included in six different OPSs, three of which must execute in a redundant computer configuration. There are from one to ten major modes included in the six OPSs, and ten SPECs, each of which is available in one or more of the OPSs. A control segment defines the initiation, sequencing, and termination for each OPS, major mode, and SPEC. The execution rates for the principal functions within an OPS vary from 25

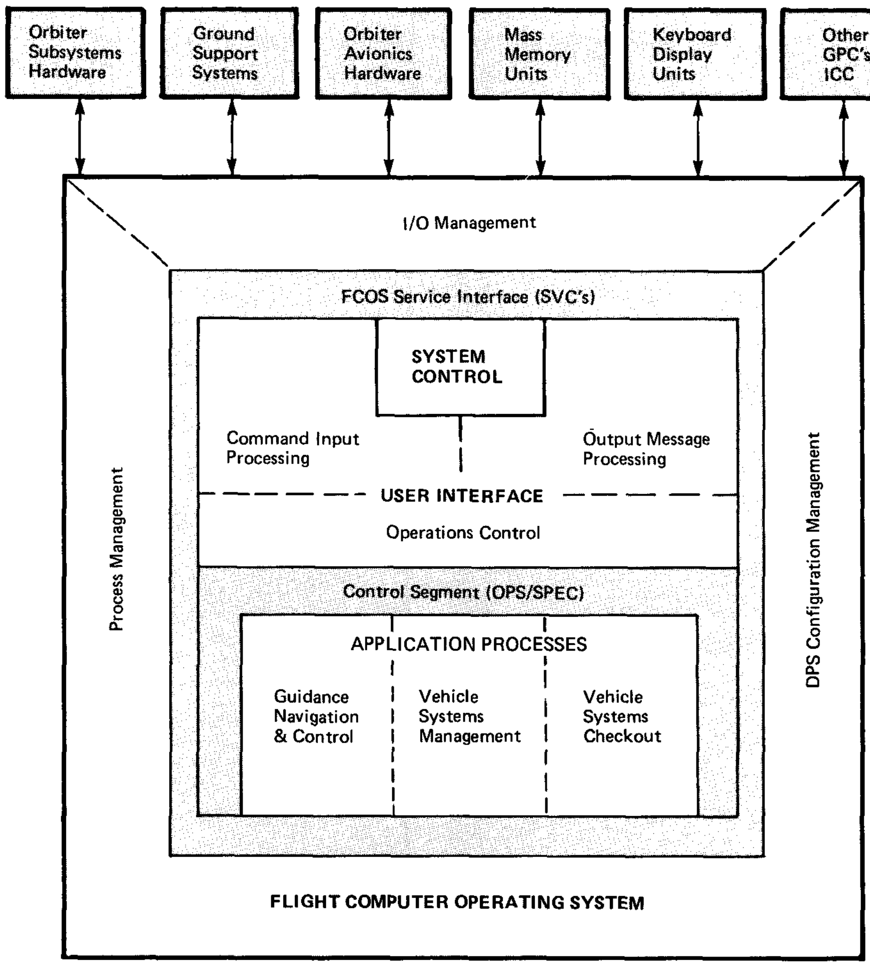

Orbiter Subsystems Hardware
Ground Support Systems
Orbiter Avionics Hardware
Mass Memory Units
Keyboard Display Units
Other GPC's ICC
I/O Management
FCOS Service Interface (SVC's)
SYSTEM CONTROL
Command Input Processing
Output Message Processing
USER INTERFACE
Operations Control
Control Segment (OPS/SPEC)
APPLICATION PROCESSES
Guidance Navigation & Control
Vehicle Systems Management
Vehicle Systems Checkout
Process Management
OPS Configuration Management
FLIGHT COMPUTER OPERATING SYSTEM

FIGURE 6. Software Architecture (Isolation of the applications software from the external environment is accomplished by the PASS architecture.)

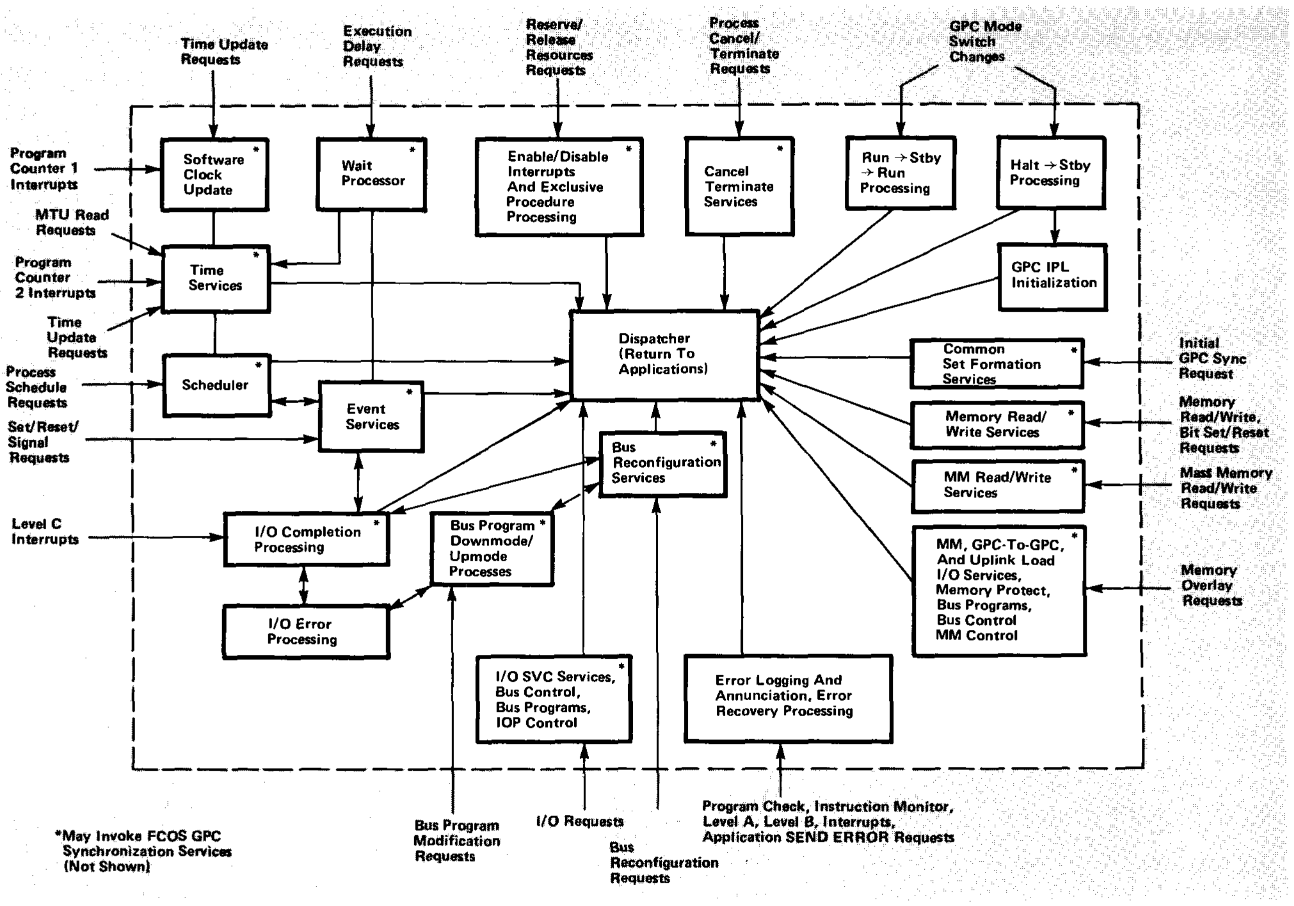

Time Update Requests
Execution Delay Requests
Reserve/ Release Resources Requests
Process Cancel/ Terminate Requests
GPC Mode Switch Changes
Program Counter 1 Interrupts
MTU Read Requests
Program Counter 2 Interrupts
Time Update Requests
Process Schedule Requests
Set/Reset/ Signal Requests
Level C Interrupts
Software Clock Update *
Wait Processor *
Enable/Disable Interrupts And Exclusive Procedure Processing *
Cancel Terminate Services *
Run → Stby → Run Processing
Halt → Stby Processing
GPC IPL Initialization
Time Services *
Scheduler *
Event Services *
Dispatcher (Return To Applications)
Common Set Formation Services *
Memory Read/ Write Services *
MM Read/Write Services *
MM, GPC-To-GPC, And Uplink Load I/O Services, Memory Protect, Bus Programs, Bus Control MM Control *
Bus Reconfiguration Services *
I/O Completion Processing *
Bus Program Downmode/ Upmode Processes *
I/O Error Processing
I/O SVC Services, Bus Control, Bus Programs, IOP Control *
Error Logging And Annunciation, Error Recovery Processing
Initial GPC Sync Request
Memory Read/Write, Bit Set/Reset Requests
Mass Memory Read/Write Requests
Memory Overlay Requests
*May Invoke FCOS GPC Synchronization Services (Not Shown)
Bus Program Modification Requests
I/O Requests
Bus Reconfiguration Requests
Program Check, Instruction Monitor, Level A, Level B, Interrupts, Application SEND ERROR Requests

FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

Hz for support of the basic vehicle flight control, down to 0.25 Hz for display update. Phasing of the function executions and associated I/O, coupled together with CPU loading constraints, have been a major influence on the GN&C design structure.

Central to the GN&C design structure (Figure 9) is a cyclic process called the executive, which controls the initiation and phasing of the principal functions and associated I/O. Executive cyclic processing is initiated by the OPS control segment through the use of an FCOS process scheduling SVC. The executive uses a table-driven dispatching design technique to initiate and control the processing of the principal functions. This technique enables phasing or sequencing of the principal function executions to be altered as required between major modes and OPS through a dispatcher table update (DTU) module.

To ensure that all critical flight control processing is completed within a 40-millisecond minor cycle, three executive structures are included in the GN&C design. The high-frequency executive is scheduled at a relatively high priority to cycle at a 25 Hz rate and initiate all principal function processes directly related to vehicle flight control. Mid-frequency and low-frequency executives are scheduled at lower priorities. They initiate principal function processes, which operate at rates of 6.25 Hz down to 0.25 Hz.

SYSTEMS MANAGEMENT

The systems management (SM) application provides the crew with status monitoring and controls for orbiter subsystems not directly involved with vehicle flight control. This support is also extended to payload subsystems. SM functions are included in a single OPS initiated during the orbit coast phase of a Shuttle mission. An interface is maintained with the GN&C computer(s); however, the SM application itself executes in a simplex configuration. Two crew selectable major modes are available for antenna and payload bay door control.

A relatively simple design structure has been implemented for the SM application (Figure 10). Central to this design are several sets of parametric data tables, which specify the parameters to be acquired and the processing to be performed on each. These tables are built through the use of off-line processors with input data that represent the orbiter/payload subsystems configuration and operating characteristics. This table-driven design technique was adopted to accommodate the reconfiguration required from one Shuttle mission

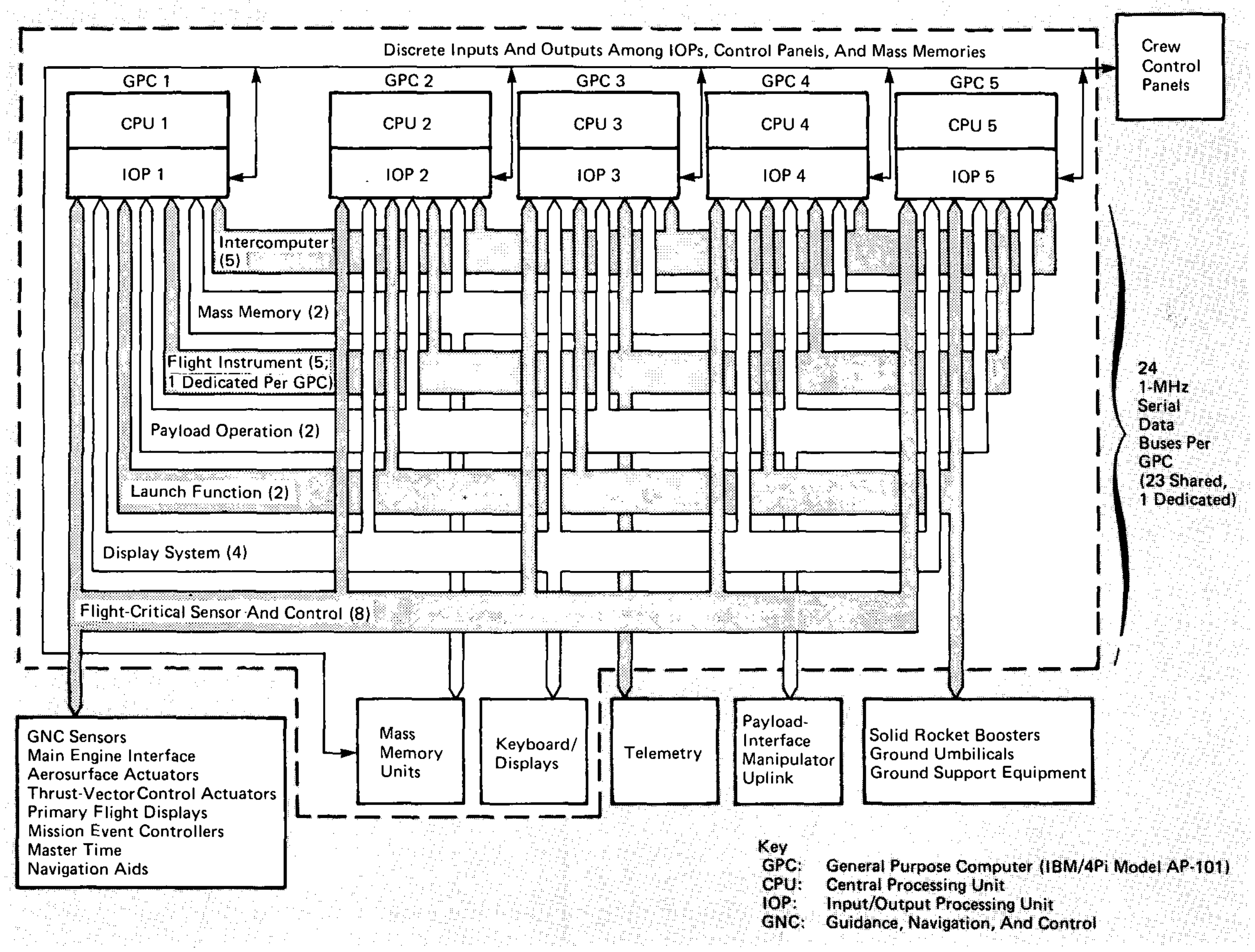

Discrete Inputs And Outputs Among IOPs, Control Panels, And Mass Memories
Crew Control Panels
GPC 1
CPU 1
IOP 1
GPC 2
CPU 2
IOP 2
GPC 3
CPU 3
IOP 3
GPC 4
CPU 4
IOP 4
GPC 5
CPU 5
IOP 5
Intercomputer (5)
Mass Memory (2)
Flight Instrument (5; 1 Dedicated Per GPC)
Payload Operation (2)
Launch Function (2)
Display System (4)
Flight-Critical Sensor And Control (8)
24 1-MHz Serial Data Buses Per GPC (23 Shared, 1 Dedicated)
GNC Sensors
Main Engine Interface
Aerosurface Actuators
Thrust-VectorControl Actuators
Primary Flight Displays
Mission Event Controllers
Master Time
Navigation Aids
Mass Memory Units
Keyboard/ Displays
Telemetry
Payload-Interface Manipulator Uplink
Solid Rocket Boosters
Ground Umbilicals
Ground Support Equipment
Key
GPC: General Purpose Computer (IBM/4Pi Model AP-101)
CPU: Central Processing Unit
IOP: Input/Output Processing Unit
GNC: Guidance, Navigation, And Control

FIGURE 8. Space Shuttle Data Processing System (Coordination of I/O operations and intercommunications between the GPCs is accomplished by asynchronous processors within the IOP.)

to the next.

The data tables produced by the off-line processors coupled with the data acquisition and performance monitor processing functions are the primary elements of the SM application. The data acquisition executes at a 5 Hz rate as established by the SM OPS control segment. Parameter preconditioning, and fault detection and annunciation (FDA) processes execute at a 1 Hz rate under control of the performance monitor.

In addition to the primary SM processes, several special processes perform more complex computations on the input data from, or issue command sequences to, specific Orbiter subsystems. This special processing is controlled by an executive dispatcher that operates at a 5 Hz rate. Included among the SM SPECs are the capabilities to make limited modifications to the SM tables, reconfigure the telemetry downlink formats, control payload operations, and operate the remote manipulator system for payload deployment or retrieval.

VEHICLE CHECKOUT

The vehicle checkout (VCO) is designed to support avionics system initialization and checkout under control of ground and/or flight crews. Its primary functions and associated I/O interfaces are configured into three ground checkout OPSs and one in-flight checkout OPS. The in-flight OPS and one of the ground checkout OPSs execute in a redundant computer configuration. A limited VCO function is included in all other OPSs to provide a ground systems interface to control the execution of a flight OPS while the vehicle is on the ground in a test configuration. The wide variety of available VCO functions falls into one of three categories. The subsystem initialization performs a preparation and configuration function of the avionics and DPS systems for subsequent VCO tests. The specialized subsystem test functions provide a series of self-contained subsystem test processes that execute to completion upon user initiation. The generalized test functions provide a catalog of subsystem testing processes, which are sequenced and controlled through ground or flight crew command inputs.

A unique feature of the VCO design is the test control supervisor (TCS). Initiated through a SPEC, it provides to the user a variety of command/response processes, which make it possible to develop test sequences external to the software for execution in a vehicle checkout OPS. The TCS will accept these commands one at a time or in the form of a series of commands (procedure) for which the required processes are automatically sequenced to completion. A maximum of three TCS command sequences/procedures can be processed simultaneously. The previously mentioned limited VCO function available in all flight OPS is a modified test control capability limited to one-for-one command processing.

IMPLEMENTATION

A comprehensive software production facility has been implemented to provide the tools, utilities, and simulation facilities required to build, integrate, test, and manage the PASS development. Included are the preprocessors used in the reconfiguration of those elements of the PASS that are sensitive to changes in Orbiter hardware configurations, mission profile characteristics, payloads, and crew displays.

Standardization of formats for external data input to these preprocessors has reduced the possibility of error through syntax and reasonableness checking. With a very few exceptions, the HAL/S compiler, developed for the NASA Shuttle applications, is used for all PASS software outside of the FCOS. Interface with the FCOS facilities is accomplished through a standard set of service macros (SVCs) supported in the HAL/S language.

The structural aspects of the software architecture and HAL/S language together with the standardization of interfaces through the FCOS SVCs and control segment grammar have been major factors in the successful implementation of PASS software. They have made it possible to pursue the implementation of software requirements in a top-down sequence and follow a release plan that used a building-block approach. System (Level A) and functional (Level B) requirements were defined relatively early and supported the devel-

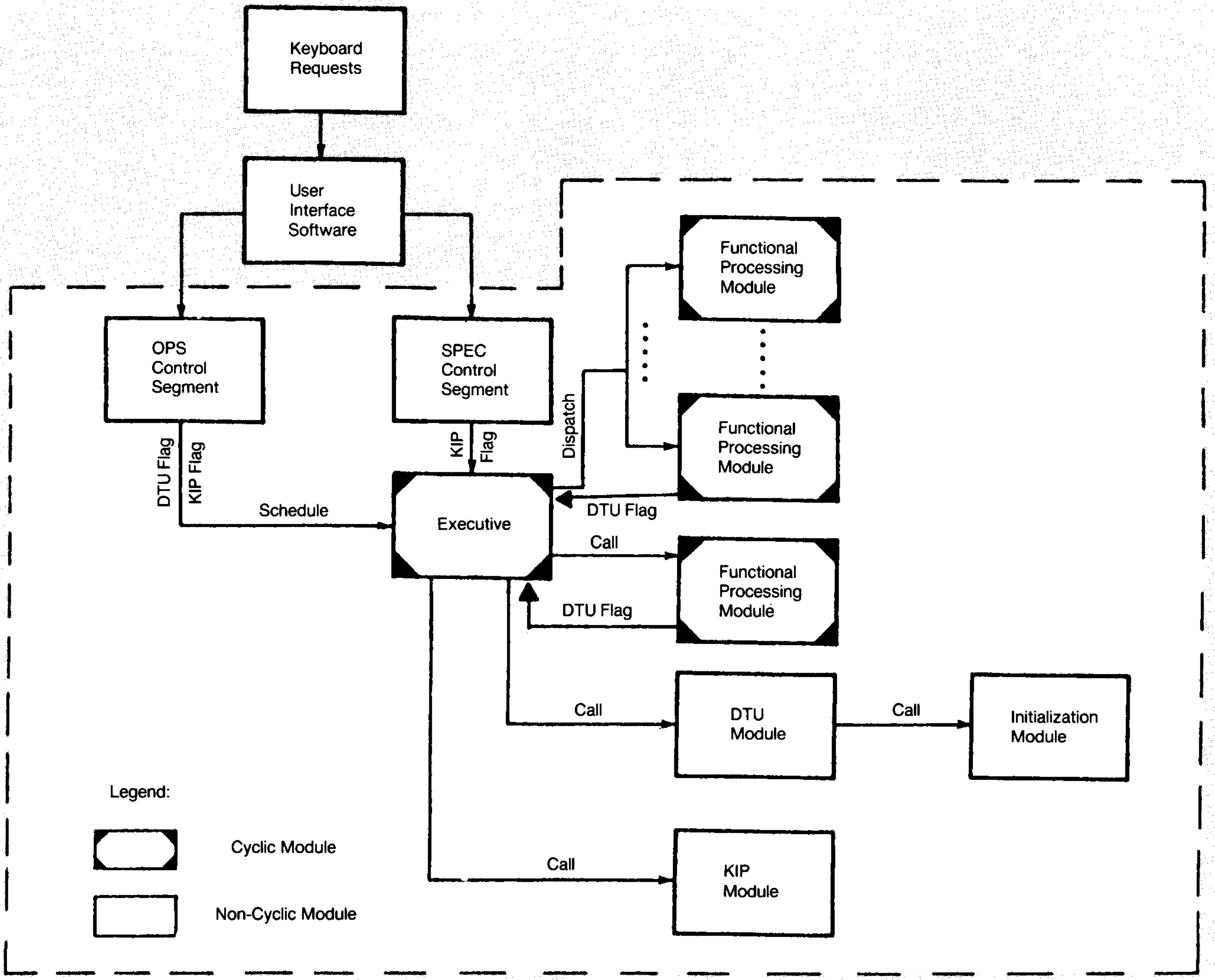


FIGURE 9. GN&C Design Structure (Synchronization of Orbiter avionics system operations is tied to the GN&C Executive.)

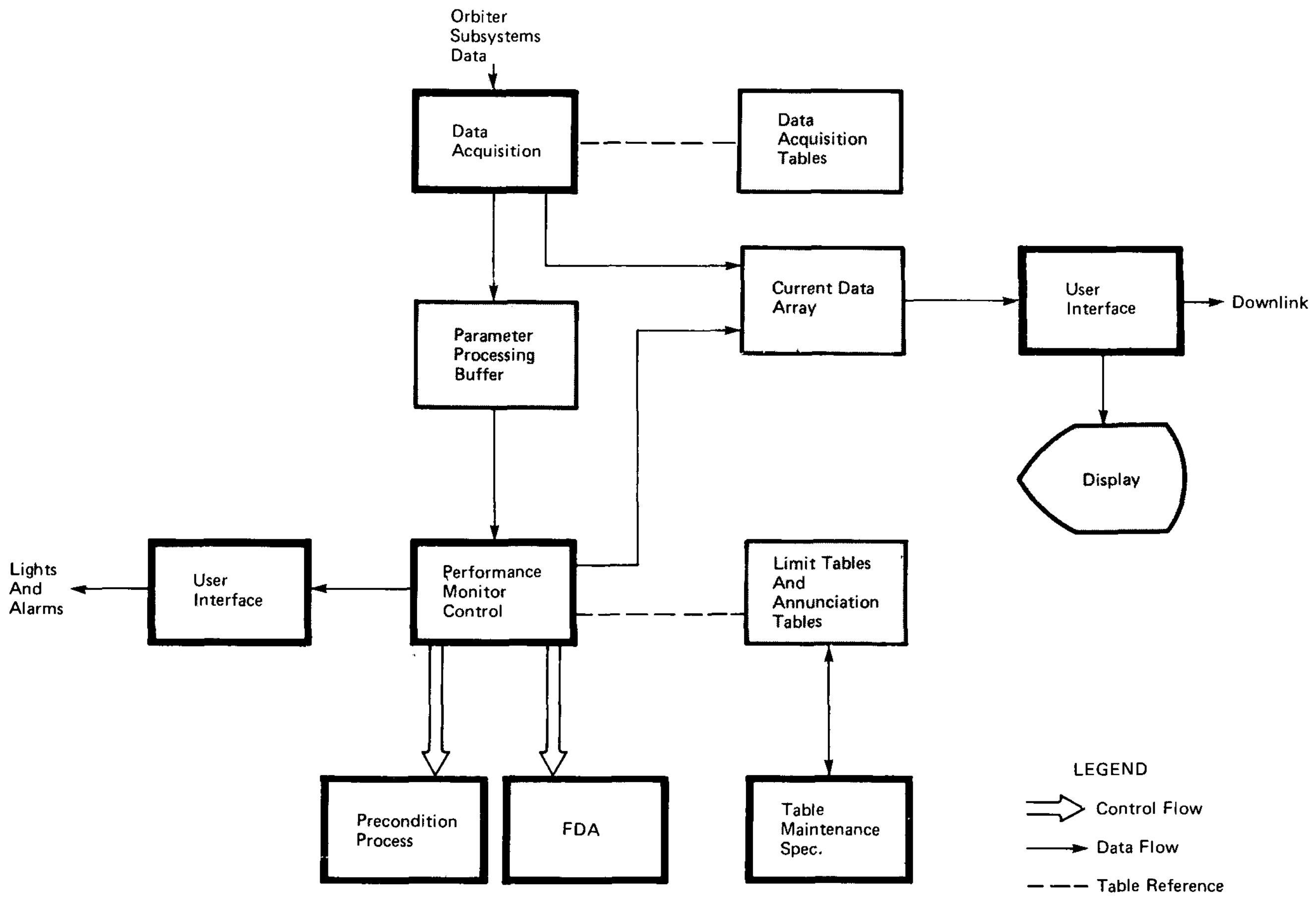


FIGURE 10. Basic SM Process Overview (Isolation of change impact from one mission to the next is accomplished with this table-driven design.)

opment of application control segment structures that could be tested using the initial versions of the systems software. This established a foundation for the subsequent modification and enhancement of the systems software and the design/implementation of detailed (Level C) application requirements as they became defined. A significant number of subsequent requirements changes was accommodated in a very controlled manner during the development cycle.

Having an established software architecture early in the development cycle also helped provide an insight into potential GPC capacity requirements. It was determined that the memory and CPU needed to implement evolving requirements were greater than the capacity available in the AP-101 computer. A "scrub" was initiated to identify areas of the software design that could be optimized and requirements that could be reduced in terms of function and rates. This analysis was accomplished early enough to implement the results without excessive impact on software schedules and costs.

SUMMARY

A characteristic of many IBM Federal Systems Division software projects is the evolution that occurs during development. Overall goals and objectives usually remain intact, but specific requirements tend to change significantly. Use of comprehensive architectural design strategies and effective configuration controls makes it possible to more readily absorb and manage the instability of a changing environment. The PASS architecture provides the foundation to accommodate a very dynamic environment while supporting the program schedules associated with the Orbiter avionic systems development, integration, and certification.

CR Categories and Subject Descriptors: C.4 [**Performance of Systems**]—*reliability, availability, and serviceability*; D.2.2 [**Software Engineering**]: Tools and Techniques; D.2.5 [**Software Engineering**]: Testing and Debugging; D.2.9 [**Software Engineering**]: Management—*software quality assurance*; D.4.5 [**Operating Systems**]: Reliability; J.2 [**Physical Science and Engineering**]—*aerospace*; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*software development, software maintenance*

General Terms: Design, Management, Reliability, Verification

Additional Key Words and Phrases: avionics system, PASS, space shuttle

Author's Present Address: Gene D. Carlow, IBM, Federal Systems Division, 1322 Space Park Drive, Houston, TX 77058.