457149

NASA

National Aeronautics and
Space Administration

NASA CR-188253

Lyndon B. Johnson Space Center
Houston, Texas 77058

AN ANALYSIS OF THE MODES AND STATES FOR GENERIC AVIONICS

(NASA-CR-188253) AN ANALYSIS OF THE MODES AND STATES FOR GENERIC AVIONICS (Lockheed Engineering and Sciences Co.) 57 p
N94-10664
Unclas
G3/06 0182636

July 1993

Richard B. Wray

Prepared by:

Lockheed Engineering & Sciences Company
Houston, Texas

Job Order 60-911
Contract NAS 9-17900

for

FLIGHT DATA SYSTEMS DIVISION
JOHNSON SPACE CENTER

LESC-30749

AN ANALYSIS OF THE MODES AND STATES FOR GENERIC AVIONICS

July 1993

Richard B. Wray, Advanced Systems Engineering Specialist

APPROVED BY:

G.L. Clouette, Project Integration Specialist
Flight Data Systems Department

D.M. Pruett, Manager, Advanced Programs
Flight Data Systems Division

Prepared by:

Lockheed Engineering & Sciences Company
Houston, Texas

Job Order 60-911
Contract NAS 9-17900

for

FLIGHT DATA SYSTEMS DIVISION
JOHNSON SPACE CENTER

LESC-30749

DOCUMENT CHANGE RECORD

The following table summarizes the change activity associated with this document.

ISSUE AND DATE	CHANGE SUMMARY	SECTION

PAGE ii INTENTIONALLY BLANK

PREFACE

This document has been produced by Mr. Richard B. Wray of Lockheed Engineering and Sciences Company (LESC), developer of the avionics architecture model represented in this document. Special acknowledgment to Mr. John R. Stovall of LESC, who codeveloped the avionics architectural model and also provided special understandings of the Space Station Freedom modes and states used as the basis for the representations in this document. Special acknowledgment is also given to Mr. Dave Pruett, Johnson Space Center, for supporting the Advanced Architecture Analysis and assisting in the development of the modes and states, and for constructive criticisms.

CONTENTS

FIGURES

ACRONYMS

AV_CTRL	Avionics Controller
C&T	Communications and Tracking
CRTL	Controller
D&C	Display and Control
DFI	Digital Flight Instrumentation
DMS	Data Management System
ELS	Environment and Life Support
EPS	Electric Power Subsystem
EVA	Extra Vehicular Activity
FOC	Full Operational Capability
GN&C	Guidance Navigation and Control
IBM	International Business Machines
IOC	Initial Operating Capability
ISE	Integrated Station Executive
JSC	Johnson Space Center
LESC	Lockheed Engineering & Sciences Company
LS	Launch Support
MCC	Mission Control Center
MSN-CTRL	Mission-Controller
NASA	National Aeronautics and Space Administration
PC	Personal Computer
POST	Power-on Self Test
PSO	Payload and Science Operations Subsystem
RODB	Remote Object Data Base
SDSS	Space Data System Services
SGOAA	Space Generic Open Avionics Architecture
SOCS	Space Operations Control Subsystem
SSF	Space Station Freedom
TDRSS	Tracking and Data Relay Satellite System

1. INTRODUCTION

When new space vehicles are developed, typical technical approaches concentrate on defining the functionality needed in those vehicles and the capabilities which must implement the functions. Specifications often describe the avionics processes, inputs and outputs. However, it is often late in the program development before the behavior of the functions and their system/subsystems is addressed; e.g., the definition of which outputs respond to which inputs, when and under what conditions. Yet the behavior of avionics systems is key to the successful operation of space vehicles, and more thorough understanding of space avionics behavior is needed.

A view of the shuttle and station avionics, as suggested in Figure 1-1, typically shows the different hardware black boxes and their interfaces, but the behavior of those boxes across their interfaces (which is represented by the gray arrow arcs) is often not described well -- it is difficult to show behavior in a diagram. Behavior (in this context) is described by the mission, vehicle and system/subsystem configurations and potential interactions; by scenarios which map out how interactions should proceed; and by the rules set up by developers on what inputs may be given to and what responses are desired in return of the avionics systems. This behavior is captured in part by descriptions of the modes and states of the system.

An analysis of the possible modes and states in space generic avionics has been performed by Lockheed Engineering & Sciences Company (LESC) for the National Aeronautics and Space Administration's (NASA) Johnson Space Center (JSC). One purpose of this analysis was to develop a more precise definition of the modes and states to which generic avionics can be subjected. Another purpose was to provide inputs to a simulation of the behavior of the Space Generic Open Avionics Architecture (SGOAA) model for the entities in flight vehicle core avionics being developed. (The SGOAA reference model is described in [WRA93]). The modes and states analysis was based on the modes analysis being performed for the Space Station Freedom (SSF) program (see [SUL92]), in particular the integrated avionics modes being defined. Examples referencing the SSF are based on SSF design prior to the 1993 restructuring.

1.1 OBJECTIVE

The objective of this study was to develop a topology for describing the behavior of mission, vehicle and system/subsystem entities in new flight vehicle designs based on the use of

open standards. It also had to define and describe the modes and states which may be used in generic avionics behavioral descriptions, describe their interrelationships, and establish a method for applying generic avionics to actual flight vehicle designs.

1.2 ORGANIZATION

This analysis report consists of the introduction in Section 1, including the definitions in section 1.3. Section 2 presents a description of the mission modes, vehicle modes, avionics subsystem states, and Space Data System Services (SDSS) subsystem states. Section 3 presents some conclusions and recommendations about using modes and states in developing space systems.

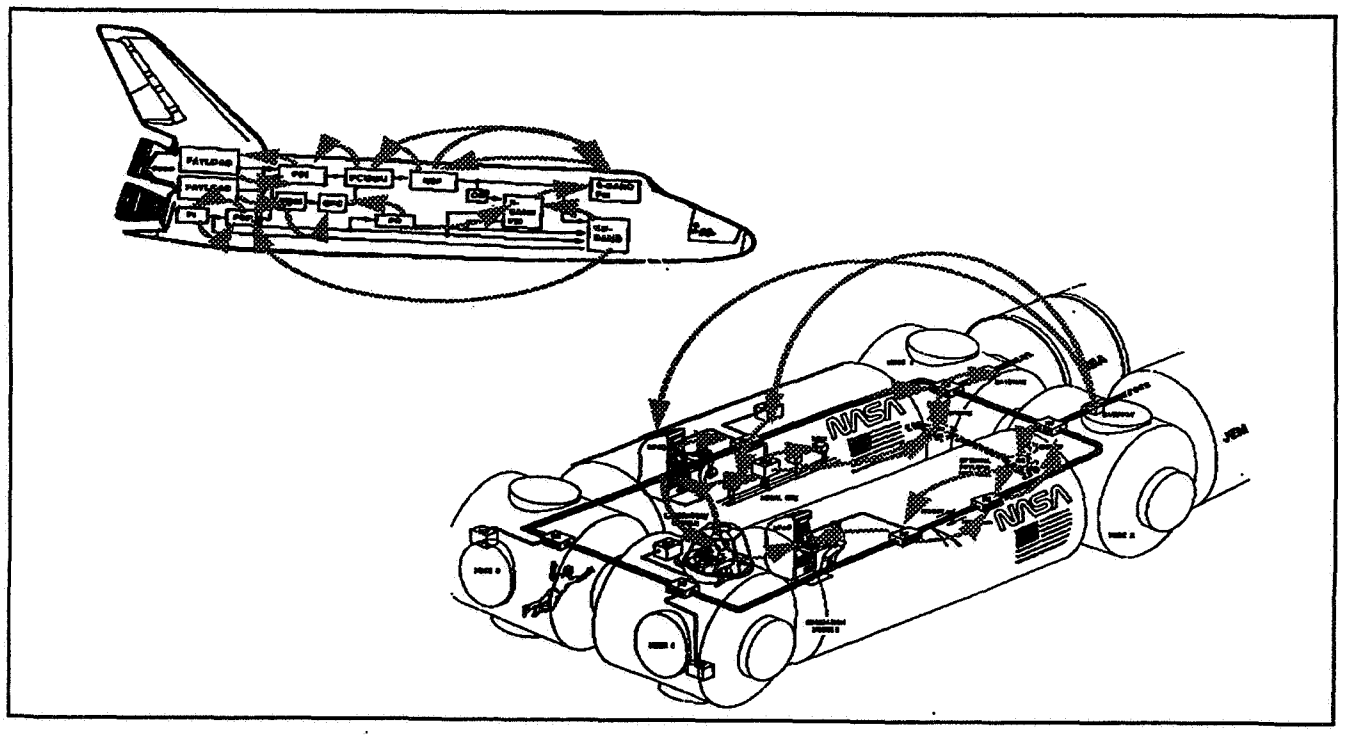

Figure 1-1. Apriori Knowledge of Functional Behavior is Important!

1.3 DEFINITIONS

An Avionics System is defined for the purpose of the SGOAA model as the set of all electronic and processing based subsystems on a space vehicle, including all hardware, software and other electronics needed to control and operate the space vehicle. It is the collection of capabilities that provides the coordinated functionality for end-to-end

processing in handling the information needed to know the platform's elements, to control its interaction with its environment, and to respond to human commands. Avionics provide for information acquisition, transmission, and storage of analog or digital signals and include the sensors, intra-platform communications, processing hardware, software and subsystems, data storage, human-machine interface subsystems, and response actuator controls used in the platform. (Adapted from JSC 31000, Vol 3, Rev E, Para 3.1.24.1.1)

An entity is an abstract element that represents a real world unit, its data attributes and essential services with their respective performance and quality characteristics. It is used similarly to the term object from object oriented analysis, without intending to convey the implicit assumptions associated with object by practitioners of object oriented analysis and design. It is the classification of items (i.e., things) related to application portability.

A function is an action/task that the system must perform to satisfy customer and developer needs.

A logical interface is defined as the characteristic requirements associated with an interaction between a source of data and the end user of the data.

A mode is a predefined set of hardware and software configurations, and associated procedures used to organize and manage the conditions of operation for an avionics system's behavior, as planned (in advance) or directed by a human. Modes may have several levels of expanding detail.

A direct interface is defined as the routing requirements associated with passing data from the source of the data to the end user of the data. Data is used by an entity in a physical manner if it passes the data on without changing the data; thus for example, network operating systems are direct interfaces to applications because they package or unpack data and send it to another network node. Direct interface requirements are normally a design issue unless the physical implementation has implications for the logical use or need for data, only then should the physical implementation be specified as a requirement. For instance, a service such as a Reports Generator getting data from a Data Base Manager might not need to know the inter-network addressing of the Data Base Manager, but the Network Manager providing the data would need to know the routing requirements of the hardware services.

The Space Data System Services is a generic functional architecture designed to provide a comprehensive set of data processing services to all space vehicles and subsystems.

The Space Generic Open Avionics Architecture is defined as the target open architecture umbrella standard being developed to establish the preliminary standard for functional, hardware, software and interface models. It is a generic architecture, meaning that the elements of the architecture do not depend on any one mission or program for their definition. The elements of the architecture can be tailored to apply to many different space missions and programs. Tailoring may result in subsets of requirements applicable to a mission or program, but will retain architectural interface compatibility. The initial focus of the SGOAA architecture is for Space Vehicles; Other-Planet Bases are part of the SGOAA architecture, but are not being addressed for now until the architecture has been refined. The SGOAA model includes a functional architecture to define the generic functions and structure needed for specific avionics capabilities.

A state is a set of hardware and software configurations which identify the intervening or instantaneous organization and constraints on the conditions of operation for an avionics subsystem's behavior, resulting from mode commands, as established by the system. Several states may result from one mode command. The existence of unexpected configurations in states indicates potential operating problems.

A system is the composite of equipment, material, computer software, personnel, facilities and information/procedural data that satisfies a user need. (Electronic Industries Association Bulletin SYSB-1)

2. MODES AND STATES

Modes and states were developed by considering the interfaces for a space vehicle. The skin of the spacecraft was defined to be the boundary for partitioning between external and internal interfaces. The boundary between modes and states was based on partitioning between activities planned and directly controlled by humans (i.e., a mode) and those activities directed, in response, by the machine (i.e., a state). Some activities were recognized as transcending all boundaries, such as operational procedures. However, even such transcending activities can be implemented by partitioning them along an established boundary; when one does so, then the procedures can be more closely linked with the activities and subsystems defined by that boundary. Such linkages can enable transcending activities to be more clearly implemented.

2.1 GENERAL BEHAVIOR

Modes are those activities planned in advanced and accordingly implemented along a timeline established by humans, or are activities directly under human control. A mode represents a unique set of behaviors, with mutually exclusive functions. A mode establishes the set of commands and resulting functions and controls which are available to personnel at a point in the mission of a vehicle; that is, modes establish control sets for operator activities and procedures. They insure all activities commanded by humans are consistent across the alternative operating configurations, functions and system behaviors. Modes do not establish controls over systems per se, although modes would be implemented in part by systems. A mode model can be developed which is the set of modes governing the behavior of a specific mission or vehicle. A mode model identifies the set of allowable commands and responses (e.g., functions, signals or activities) for each command. The internal/external vehicle interface boundary is used in distinguishing between commands affecting the externally-observed behaviors which could affect other vehicles or facilities. Modes affecting other external entities are defined as mission modes. Modes affecting other internal entities are defined as vehicle modes.

Mode transitions may (and probably will) have conditions and checks required to be performed before a transition may take place to insure it is safe to perform the transition. Safing is a condition of reduced functionality since its purpose is to facilitate transitions. Since mode transitions are caused by human command, the combinations and timings of all possible commands may not have been fully explored and tested. Prior to a routine transition, a safety check should be performed to insure that all desired activities in the mode being exited have

been completed, no residual conditions have been created which would create a risk in the mode being entered, and that conditions are ready for entry into the commanded mode. However, capabilities for special mode transitions (or exits, such as on crew command) would be needed. It would be desirable for special mode transitions if specific safety procedures were in place to be used to perform special or emergency terminations of on-going activities. Thus the specific activities of a "safing" check are context dependent and will be different for each mode transition. Safing is particularly needed after a failure when the system is more likely to be perturbed; however, it is also relevant in normal conditions for complex missions where command confusion may be possible.

States are the activities that a system or subsystem adopts in response to a mode input. A system/subsystem may step through several states in transitioning between one mode and another. Systems respond to mode commands by automatically entering a sequence of states, controlled by closed loop control algorithms which are specific to the selected mode. A state represents a unique set of stable system or subsystem behaviors in a closed loop control algorithm. Different states have mutually exclusive functions (e.g., the state "operate" is mutually exclusive to the state "standby".) States must allow override and manual controls by crew. Manual control over low level entities must not be precluded by logic designed into any system, although inhibition logic in the form of downloadable (and thus changeable) tables of rules may be used to establish a-priori crew guidelines on activities and capabilities which can be disabled through software.

State transitions are typically caused by mode changes. The transition between modes causes a change in the states of subsystems, with subsystems moving through a sequence of states until stable operation in the new (directed) mode has been achieved. State transitions can be used to define and describe special subsystem behaviors such as needed for autonomous operations, closed loop controls or as a response to failures. Safing is also needed in state transitions, but should be less complicated than mode safing since all states should have been pre-defined. Selected combinations of states will have been tested; however, in complex systems not all combinations can be tested due to cost considerations. Capabilities for special state transitions (e.g., exits, such as in response to crew commands) may be needed if specific safety procedures were to be used to perform special or emergency terminations of on-going activities. Thus, a safety check should be performed to insure that all activities in the state being exited have been completed, no residual conditions have been created which would create a risk in the state being entered, and that conditions are ready for entry into the directed state.

The key difference between a mode and a state is that a mode is planned in advance or directed by a human, while a state is the response configuration adopted by a processing controlled system/subsystem. A state may be a complex configuration with many subsets, or it may be a simple configuration at the lowest level consisting of a switch setting such as on-off.

Where a system is complex, and consists of several subsystems, each of which also consists of lower level subsystems, states may be very complex. If (as is normal in many programs) the subsystems have been parceled out to different subcontractors for development, then each subsystem so developed may have its own set of states which may be independently developed from any other subsystem states. Then the total aggregation of states for several subsystems may not be well understood, and may not have been rigorously tested under all possible input sets. While states have to be predefined and known for each low level subsystem when they are developed, as these are aggregated into larger subsystems, the combinations of predefined state sets may become very large, with unknown or unexpected sets of combinations becoming possible. Three possibilities can exist: (1) unknown non-operational states which are errors, (2) operational states which are unexpected, and (3) operational states which are expected. The last is obviously the preferred condition. However, planning and design must also consider handling of the first two cases. The modes and states approach in this document can facilitate such planning and design.

When hazardous activities are being commanded in a mode, or being implemented in a state, the operation should take place completely within the one mode, and be divisible into a sequence of states each of which contains any hazardous activity completely within the one state. The objective is to not have any hazardous activities underway when a mode or state transition occurs since transitions are, by definition, inherently unstable. Transitions with hazardous activities underway potentially increase any risk from the hazard.

Four types of mode and state models, as illustrated in Figure 2-1, have been defined:

- Mission mode model governing mission element behavior
- Vehicle mode model governing internal vehicle behavior
- Subsystem state model identifying potential subsystem element behavior, with the Avionics Subsystem used as an example
- Function state model identifying potential functional element behavior, with Space Data System Services used as an example.

On the left in Figure 2-1 is the hierarchy of elements required in a mission, from the mission entities, to the vehicle internal entities, to the subsystem entities, to the functional entities. Within each level, there are multiple entities controlled by one entity called a controller (CTRL). The behavior of the CTRL entity can be described by a model of either modes or states, as represented by the rounded rectangles in the center of the figure. Mode and state behavior at each level of the hierarchy is linked by the commands issued, controls generated, events occurring, and resulting actions which can all be detected throughout the hierarchy by various checks and tests. Such detections can in turn cause additional actions to take place.

Mission modes are behaviors across those external interfaces to a flight vehicle which enable the flight vehicle to perform a mission in concert with other supporting vehicles or facilities, such as the tracking and data relay satellite system (TDRSS) or the mission control center (MCC).

Vehicle modes are behaviors across those internal interfaces to a flight vehicle which enable humans and/or the flight vehicle software to organize the vehicle's capabilities to perform directed missions.

Subsystem states, such as in the avionics subsystem, are behaviors internal to a flight vehicle (or internal to a supporting facility) which identify the potential response configurations a subsystem, such as avionics, may adopt or make in changing between modes.

Function states are behaviors internal to a flight vehicle which identify the potential response configurations a major function (or lower level subsystem such as the data services function or subsystem, guidance navigation and control function or subsystem or other applications function or subsystem) may adopt in or make in changing between modes or subsystem states. Figure 2-1 depicts the relationships of modes and states in a flight vehicle.

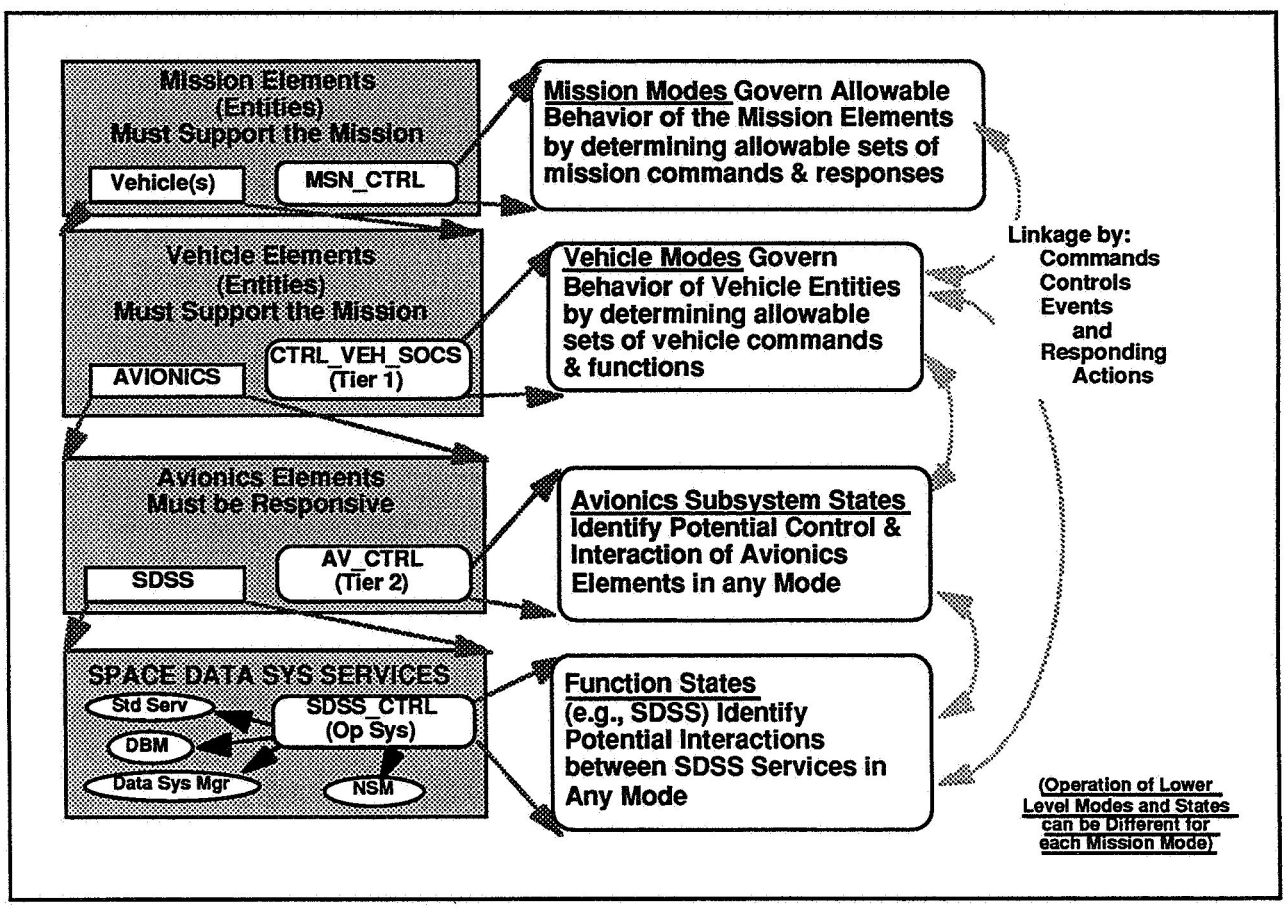

Figure 2-1. Levels of Modes and States in a Flight vehicle

Each mission mode may cause multiple vehicle modes to occur or be blocked. Each vehicle mode may in turn cause multiple subsystem states to occur or be blocked. Similarly, each subsystem state may cause multiple function states to occur or be blocked. Activities at each level are linked to activities at other levels through the creation of commands, controls and events, which are detected by other entities at the same or other levels. These detections act as inputs to the detecting entity, causing it to act if necessary according to the rules governing that entity's behavior. The resulting sets of lower level configurations may be different for each higher level mode or state; they may use similar names for similar behaviors (e.g., scheduled vs. interruptive testing), or they may use similar names for totally different behaviors (e.g., standby mode vs. storage state).

The specific modes identified below in the mission and vehicle mode sets are not intended to be the only possible modes. Additional modes may be needed in specific

2.2 MISSION MODES

Mission modes describe the control of the interactions of entities at the highest level of a mission, such as the space vehicle with other facilities or platforms as shown in Figure 2-2. The mission entities (shown in the upper left of the figure) are the major standalone elements or systems involved in a mission, and are under the operational control of a mission control entity. The behavior of the mission controller (MSN_CTRL) entity is shown in the lower right of the figure in the mission mode model.

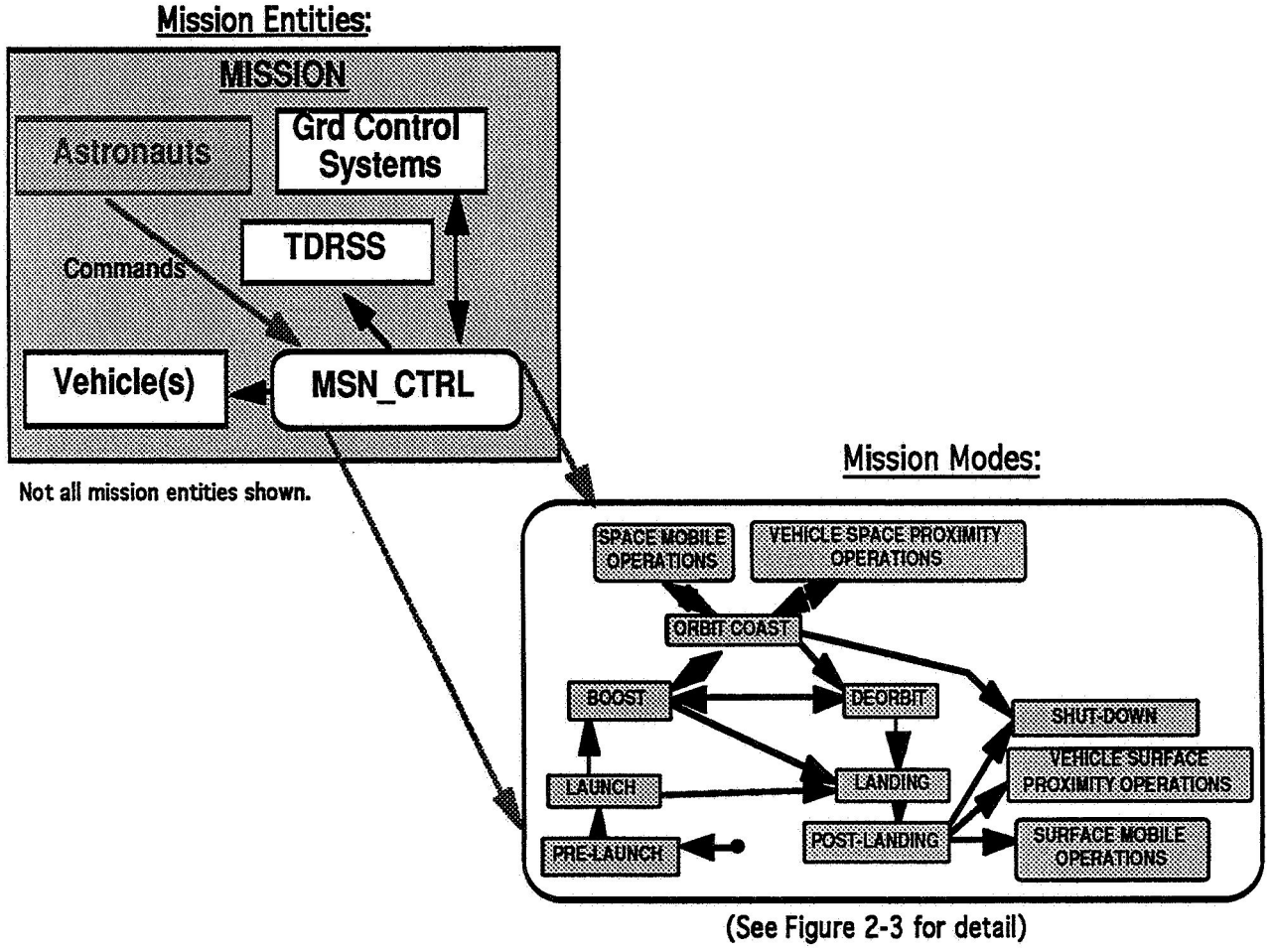

Figure 2-2. Mission Mode Transitions Implement the Mission Controls

Although not all mission entities are shown in the figure; the vehicles are the entities carrying crew, payloads and/or supplies for the mission; and the ground control systems are the support systems on the ground, the launch systems, the test and checkout systems, etc. needed to get the mission off the ground and into orbit. The TDRSS is usually thought of as

a "pipe", but is in fact an end system in its own right, since it may have to be configured specially to support a mission. For instance, on the Common Lunar Lander Artemis program (at one point in its development), constraints on TDRSS coverage were limiting Artemis' ability to send telemetry back to the ground control systems, and would have required special TDRSS configuration control changes to insure sufficient telemetry coverage (until the requirements were changed). The mission control entity is the overall control entity for the mission; although this is usually associated with the MCC, in this instance, the control is the generic function which may or may not necessarily be centered in the MCC.

The key is that the MSN_CTRL is a processing entity which governs the interactions of its "sibling" mission level entities. This controller determines what sets of mission commands and responses are allowed under what conditions, and at what times or in what sequences. This behavioral control is based on rules and procedures planned and established by humans, and only human-directed behaviors are allowed under the conditions set up by the human plans. Such directed behaviors may be immediate, or may have been preplanned and implemented by a timelined sequence. As a result of these modes; command, controls, and events may be established which in turn constrain or initiate activities and procedures in lower level entities.

Thus, the mission modes define the sets of allowable mission element behavior which the MSN_CTRL can effect. Each mode encompasses the set of commands, controls, events, conditions and procedures needed to safely operate the mission in a stable manner. The transition checks from one mission mode to another establish how the MSN_CTRL entity can establish transitions from one stable set of operations and ground rules to another set of operations with different ground rules. For instance, changing from ORBIT COAST to REBOOST involves a number of specialized functions which insure that reboosting does not occur in an unsafe manner due to unsafe combinations of subsystem configurations and crew activities. For the generic case of any type of space vehicle, twelve modes are identified. These modes and their allowable transition paths are shown in more detail in Figure 2-3. Note the identification of examples of specific functions to be provided in selected modes.

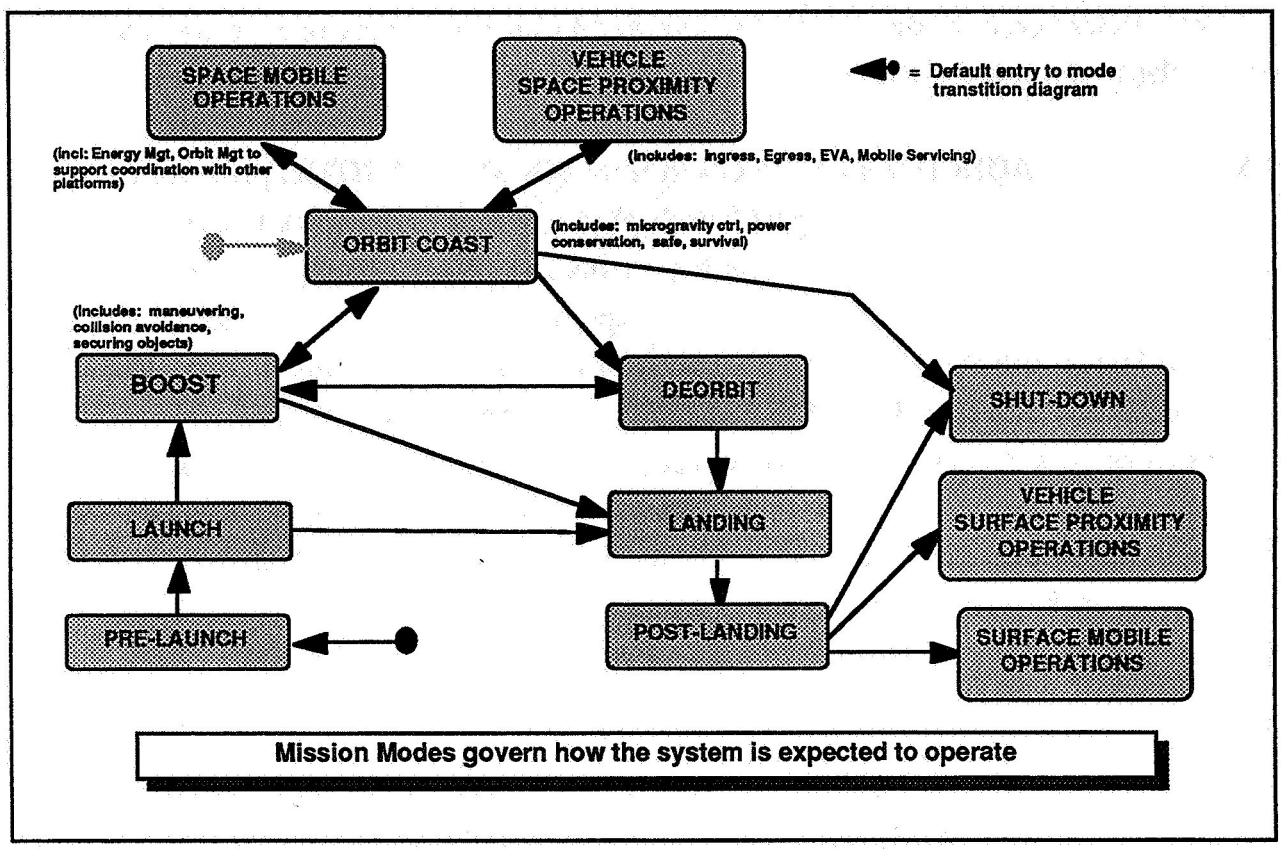

Figure 2-3. Generic Mission Mode Transitions are Constrained

The default entry for normal missions is shown by the black ball and arrow in the lower left. The first set of modes involves getting the mission launched. PRE-LAUNCH activities prepare the vehicle using the appropriate ground launch systems, checkout consoles, test equipment, pre-launch procedures, etc. When the space vehicle is approved for launch, the MSN_CTRL entity enables the set of activities allowed under LAUNCH, while disabling the set of activities under PRE-LAUNCH. Only then can special events such as fuel and oxidizer loading take place with the appropriate safety controls and systems in place. From LAUNCH, two paths are allowed: one to BOOST to orbit, and a second for a direct return to landing.

BOOST performs the needed ascent activity control, such as atmospheric maneuvering, booster/staging separation, etc.; places the space vehicle in orbit; separates the launch vehicle from the mission spacecraft; performs the needed maneuvering to achieve interim and final orbits; recommends and/or performs collision avoidance maneuvering; insures securing object messages are transmitted to on-board crew, etc. BOOST is constrained by conditions enabled by the crew or ground systems to insure mission, vehicle and safety

critical activities occur as desired by the crew, and includes appropriate overrides and interlocks, as discussed later.

The paths from LAUNCH to LANDING and from BOOST to LANDING provides a control option for abort from launch and post-launch phases of the mission back to earth. Depending on the specific capabilities of the mission, this abort prior to achieving orbit could be through use of an emergency escape capsule, by ejection of the manned spacecraft from the launch vehicle and return to landing, or by other mechanisms to be developed. Note also that a path from BOOST to DEORBIT is provided to allow the control mechanisms for a change from one set of activities while boosting in orbit to a return to earth, if a return (for instance due to significant problems) were necessary. Safe condition checks in transitioning from BOOST to LANDING might include verification that solid rocket boosters have been jettisoned, and are not a hazard to the command vehicle.

Finally, ORBIT COAST is achieved as the MSN_CTRL completes the last boost operation and carries out the controls necessary to change from the BOOST mode to the ORBIT COAST mode, which is assumed to be a primary mode and reason for many of the space missions. During ORBIT COAST, the appropriate payload activities can be enabled, experiments can take place such as in the microgravity controlled environment, power conservation activities may be carried out, safe and survival activities are enabled as needed. Controls over all activities internal to the flight vehicle are included in the ORBIT COAST mode. Controls over any activities external to the mission spacecraft, such as extravehicular activity (EVA), are treated as different mission modes, requiring special sets of rules to be carried out and subsystems to be used. Due to the potential for problems or interference between external and internal flight vehicle activities, and the potential risk to crew members while outside the flight vehicle, external activities have been included under their own modes, rather than included in ORBIT COAST, as described below. Communications required with external entities is another important reason.

From ORBIT COAST, other modes may be entered, such as SPACE MOBILE OPERATIONS, VEHICLE PROXIMITY OPERATIONS, SHUT-DOWN, BOOST, and DEORBIT. Each transition would have a set of rules to be carried out before the old mode could be disabled and the new mode could be enabled. Some transitions would be caused or inhibited by predefined command, controls or events detected by condition sensing processes. SPACE MOBILE OPERATIONS is provided to establish the control structures needed in a flight vehicle which must constrain its orbit (or other) operations due to needs associated with

other flight vehicle. For instance, if an orbiting laboratory must constrain its ability to move to a new orbit because an untethered vehicle or satellite will be returning later that day, then SPACE MOBILE OPERATIONS would provide the mixture of joint controls, such as energy or orbit management modes, that affect both the laboratory and the satellite's need for effective operations to enable them to later perform a rendezvous. VEHICLE SPACE PROXIMITY OPERATIONS is provided to support local external activities around the flight vehicle, such as egress, EVA, mobile servicing, and ingress. For instance, an astronaut on EVA in VEHICLE SPACE PROXIMITY OPERATIONS mode, would constrain or interlock the flight vehicle from going into BOOST or DEORBIT mode unless overridden. System safety checks would verify such interlocks had been released before systems activation was allowed.

DEORBIT is the beginning of those activities leading the flight vehicle back to the surface. The controls to do this are partitioned into several sets: DEORBIT, LANDING, and POST-LANDING. DEORBIT provides those controls needed to insure a safe departure from orbit into the reentry window. LANDING provides those controls needed to insure a safe setdown on the surface. Allocations of specific controls such as for guidance to the landing site could be to either mode; a reasonable rule for such partitioning is that all control activities associated with leaving orbit and entering atmosphere are part of DEORBIT, while all control activities associated with maneuvering in atmosphere to the touchdown site, touching down, and operations to immediately shutdown or secure appropriate subsystems on the flight vehicle (e.g., engine shutdown and oxidizer and fuel safing) would be part of LANDING. POST-LANDING is envisioned to encompass those control activities which provide for spacecraft security (e.g., airlock-engine interlock release) after touch down, the activities inside the flight vehicle after touch down (for rest and experimentation) and those activities that are needed after touch down to prepare the vehicle for its next set of mission activities. From the flight vehicle operations viewpoint, since POST-LANDING is concerned with securing the vehicle, this is interpreted to include those vehicle security aspects related to protecting and safely operating the vehicle while payload experimentation is conducted. After landing, vehicle activities will probably take place that interact with the environment external to the flight vehicle, including SHUTDOWN, VEHICLE SURFACE PROXIMITY OPERATIONS, or SURFACE MOBILE OPERATIONS.

The set of control behaviors permissible after the POST-LANDING checks have been completed depend on the purpose of the landing. If the vehicle has returned to earth, then the probable control structure is SHUT-DOWN. However, if the landing is on another

planetary body such as the moon or mars, then the probable control structures are continued POST-LANDING experiments, VEHICLE SURFACE PROXIMITY OPERATIONS or SURFACE MOBILE OPERATIONS. Since on-surface experiments conducted inside the flight vehicle are usually treated as payloads, they are relatively independent of the flight vehicle operations per se, and would have their own control structures. Support (if any) for them from the flight vehicle subsystems would be provided for under the POST-LANDING mode. Similarly, experiments conducted outside the flight vehicle (on another planetary body) would have their own control structures, and would be supported by the surface operations modes described next.

VEHICLE SURFACE PROXIMITY OPERATIONS is provided to support local external activities around the flight vehicle, such as egress, surface EVA, surface vehicle servicing, and ingress. For instance, an astronaut on going to EVA in VEHICLE SURFACE PROXIMITY OPERATIONS mode, would be constrained from opening both airlock doors unless overridden and engines could not be fired if an astronaut were on EVA. SURFACE MOBILE OPERATIONS is provided to establish the control structures needed in a flight vehicle which must constrain its local operations or experiments due to needs associated with other vehicles. The most likely vehicle requiring flight vehicle constraints would be rovers being prepared for operations, performing tasks or being recovered. For instance, if a surface lander vehicle must constrain its ability to refuel from an unmanned supply vehicle because a manned rover is operating locally, then SURFACE MOBILE OPERATIONS would provide the mixture of joint vehicle operations controls, such as collision avoidance, fluids management and joint navigation modes, that affect both the flight vehicle, the supply vehicle and the rover's need for effective operations to enable safe operations.

2.3 VEHICLE MODES

Vehicle modes describe the control of the major vehicle entities in support of the current mission mode, as shown in Figure 2-4. The vehicle entities are the major subsystems, such as the power, propulsion, fuels, oxidizer, and avionics subsystems, which are needed to properly configure a spacecraft. Each of these major subsystems must be under the operational control of some control entity, which is defined in the generic architecture to be part of the Space Operations Control Subsystem (SOCS). In the Space Station Freedom program, CTRL_VEH_SOCS is performed by the Integrated Station Executive (ISE), with perhaps the support of the Timeliner.

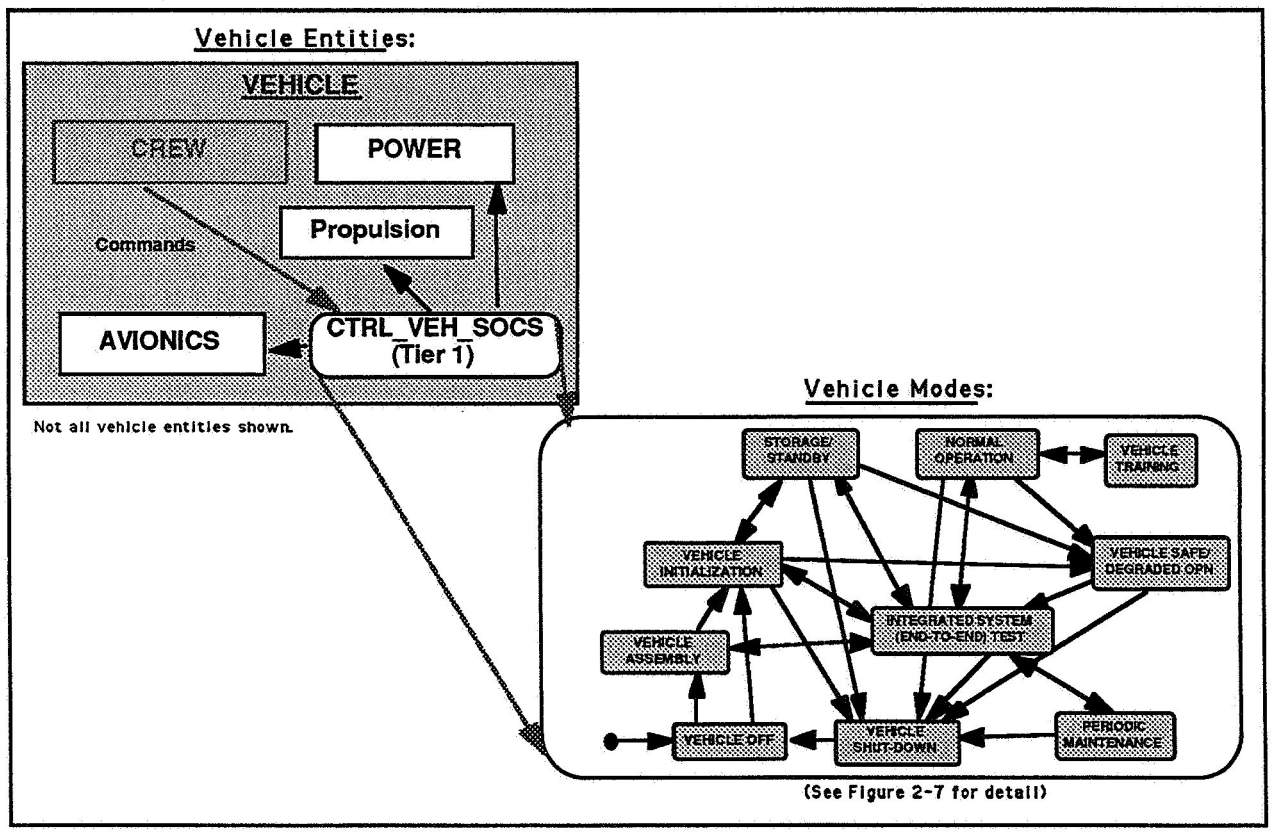

Figure 2-4. Vehicle Mode Transitions Implement the Vehicle Controls

The vehicle entities that are under CTRL_VEH_SOCS are shown in Figure 2-5 for the generic architecture. Each of these entities contains all the hardware, software, procedures and rules needed to operate as a subsystem in the flight vehicle. While the figure appears to be SOCS or SDSS centric, this is only because the figure is showing the control by the SOCS carried out through the SDSS. Note that all subsystems must respond to the crew, through

the Crew Display and Control (D&C) subsystem, which is the crew's entry point into to the automated and processing subsystem of the flight vehicle.

The Payload and Science Operations Subsystem (PSO) is under the flight vehicle SOCS control because it must be subservient to the needs of the flight vehicle and safe vehicle operation. However, it is assumed that it operates relatively independently, with its own controllers, and control modes which are not covered in this document. Similarly, the Launch Support (LS) Interface Subsystem provides the interface controls from the flight vehicle to the launch center subsystems, especially prior to vehicle liftoff. The Digital Flight Instrumentation (DFI) Subsystem includes the instruments needed in experimental flight vehicle flights.

While environmental control will be needed in essentially all vehicles, the life support in the Environment and Life Support (ELS) Subsystem would obviously only be needed in spacecraft operated by humans. The environmental controls in the ELS may be active (such as an electronics heat dissipation capability) or passive (such as insulating blankets). ELS is provided for generality since the SGOAA can be applied to both crewed and uncrewed vehicles. The Electric Power Subsystem (EPS) provides the power either from batteries, solar arrays, nuclear power generators, a combination of these or other means. The Communications and Tracking (C&T) subsystem provides the multiple link communications, the interfaces to external systems such as TDRSS and tracking radars and processing as needed. The Guidance Navigation and Control (GN&C) subsystem provides these capabilities, while Propulsion is under GN&C subsystem control due to its sensitivity and time latency requirements.

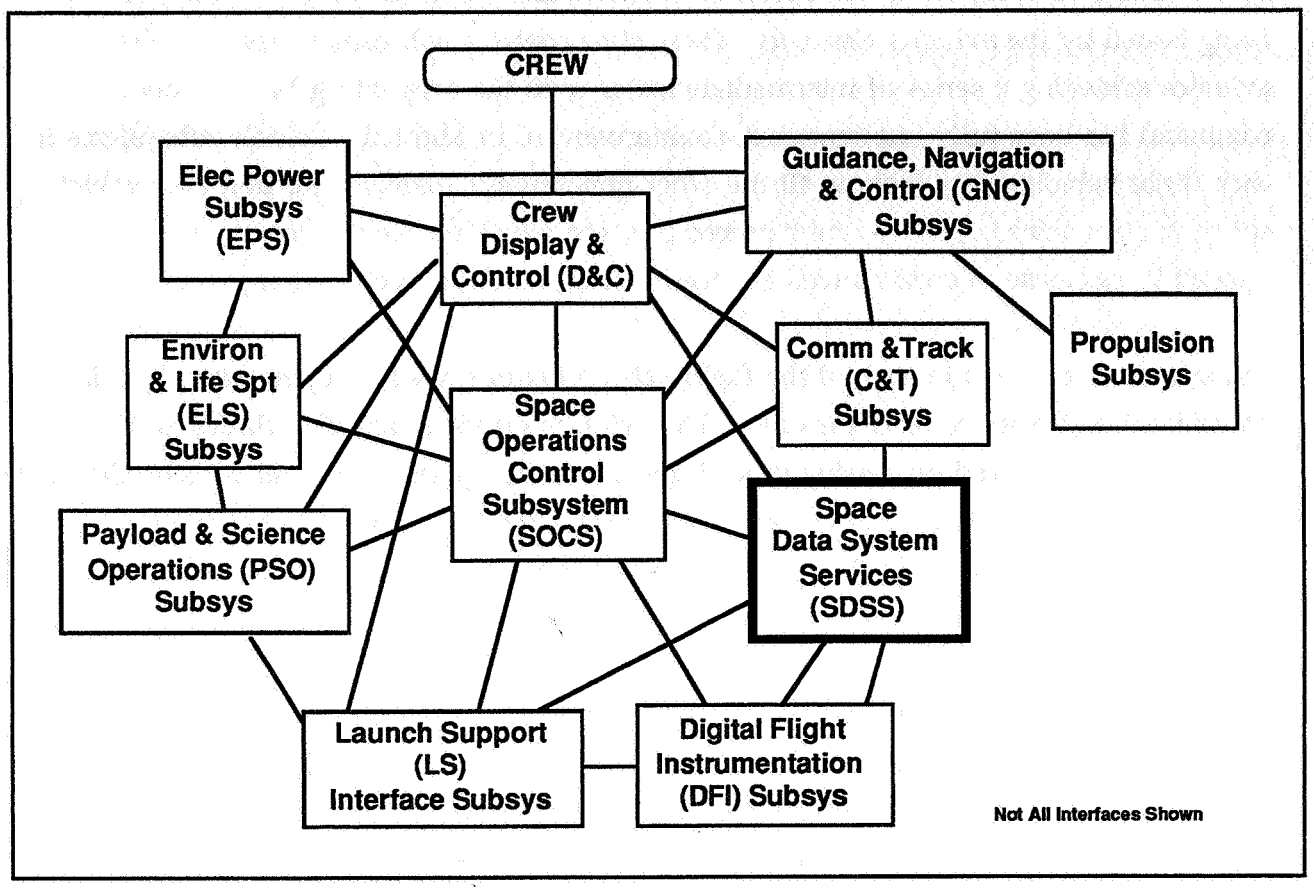

Figure 2-5. Space Vehicle Entities Must be Controlled in All Modes

Figure 2-6 represents a view of the major vehicle subsystems in a spacecraft (represented in white), and the associated avionics elements (represented in grey), partitioned by a plane drawn horizontally through their centers representing that each vehicle subsystem has embedded avionics. The vehicle subsystems and their associated commands and controls can be partitioned into modes and states by a plane through their centers corresponding to the partitioning between subsystems and avionics elements. Thus, for each white box, there are associated mode commands applicable to that box. The mode commands result in the vehicle subsystems achieving the mode associated with the command. For each grey box, there are associated state controls applicable to that box. The state controls result in the avionics elements achieving the state associated with the control signal.

The crew is both a "subsystem" and the ultimate command and control element in the spacecraft (as represented by the white oval). The crew is responsible for the activity of the vehicle subsystems (white) through mode commands. Each mode command is interpreted

by the subsystems, especially the avionics elements, and results in a series of state controls being issued by the avionics elements. These state controls will usually result in the avionics achieving a series of intermediate states until the originating human mode command has been fully implemented, countermanded or aborted. Vehicle subsystems in each flight vehicle entity include all the other non-avionics elements needed by a vehicle entity, such as vehicle physics, crew procedures, etc. which enable the vehicle to act and interact in response to crew as well as other external environment entities.

Avionics state controls in each of the flight vehicle entities result in operation which is transparent and responsive to the crew. The avionics for state control includes all the hardware, software and processing procedures needed to operate the flight vehicle. Selected interfaces between avionics entities (grey) are represented by the lines in Figure 2-6. Although interface activity is directed by humans between entities, control over use of such linkages is a state control issue.

The key to vehicle control is the SOCS, which provides the overall vehicle control function. It governs the interactions of its "sibling" vehicle entities as shown in Figures 2-4 and 2-6. It determines the allocation of resources to other vehicle entities, controls the timing or sequencing of their activities, and generally determines what sets of vehicle commands and responses are allowed under what conditions. As in the mission modes, this behavioral control is based on rules and procedures established by humans, and only human-directed behaviors are allowed under the conditions set up by the human plans. As a result of these modes, command, controls, and events may be established which in turn constrain or initiate activities and procedures in lower level entities.

The vehicle modes define the sets of allowable functional behavior which the CTRL_VEH_SOCS can effect. Each mode encompasses the set of commands, controls, events, conditions and procedures needed to safely operate the vehicle in a stable manner. The transitions from one vehicle mode to another establish how the CTRL_VEH_SOCS entity transitions from one stable set of operations and ground rules to another set of operations with different ground rules. For instance, changing from INTEGRATED SYSTEM (END-TO-END) TEST to NORMAL OPERATION involves a number of specialized functions not normally needed, and specialized data which must be properly controlled and purged from the system so test data is not erroneously interpreted as operational data.

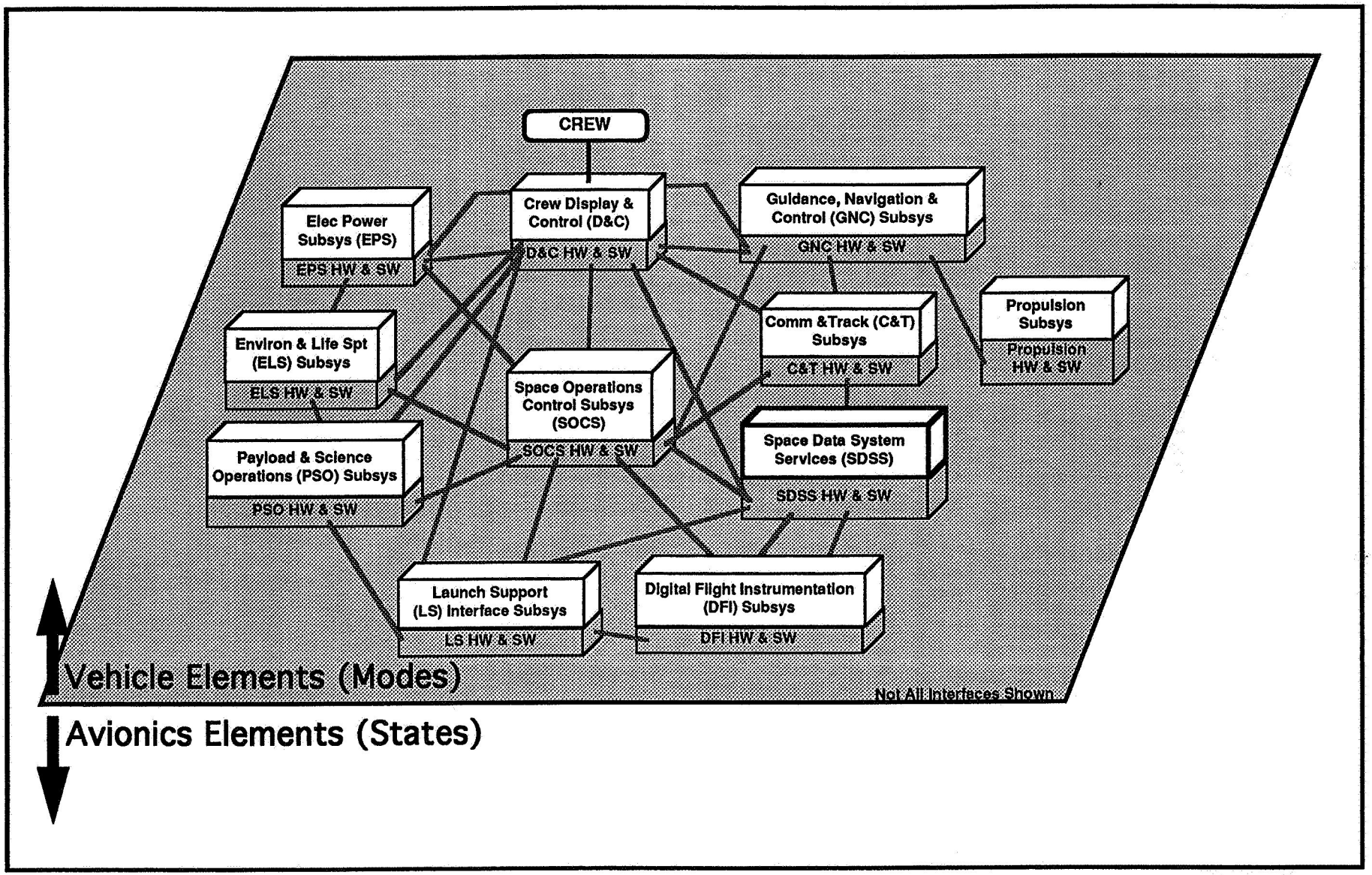

Figure 2-6. Space Vehicle Entities Contain Hardware, Software and Procedures

For the generic space vehicle, ten modes are identified, directly based on the SSF modes analysis (as reported in [SUL92]. These modes and their allowable transition paths are shown in Figure 2-7. Any mode may be further decomposed into lower level modes showing the details of lower level control structures. Note the nominal path to achieve normal operations, if everything is working toward a standard mission with no anomalies or special activities planned. All the other paths become executable under varying sets of special circumstances, such as VEHICLE ASSEMBLY for new vehicle construction or VEHICLE TRAINING for directed training missions.

The default entry for normal vehicle operations is into a vehicle with no power applied, which must be powered-up (and booted). For instance, many spacecraft are powered up on the pad prior to launch, even though many hours or even days may pass before the spacecraft needs to operate independently from its launch vehicle, because issues such as the timing may need special synchronization, the position fix may need special determinations, or special test and checkout on activation may be needed by the mission flight vehicle that is not met by the launch vehicle. When VEHICLE OFF mode is entered, power must be applied to enable the flight vehicle to become active, to be assembled, or initialized. Some external input is usually needed to enable the VEHICLE OFF mode to be exited.

From VEHICLE OFF, only two paths are allowed: one to VEHICLE ASSEMBLY and another to VEHICLE INITIALIZATION. VEHICLE ASSEMBLY enables pieces of the flight vehicle to be put together. These pieces may be hardware requiring assembly before the VEHICLE INITIALIZATION (or startup) may take place. Alternatively, the flight vehicle may have subsystems or subassemblies which need to be separately initialized and checked out prior to VEHICLE INITIALIZATION, or it may have subsystems or subassemblies already operating in a safe condition which need to be put together and checked out for some partial capability before VEHICLE INITIALIZATION can take place. From VEHICLE ASSEMBLY, two modes may be entered: VEHICLE INITIALIZATION, or INTEGRATED SYSTEM (END-TO-END) TEST.

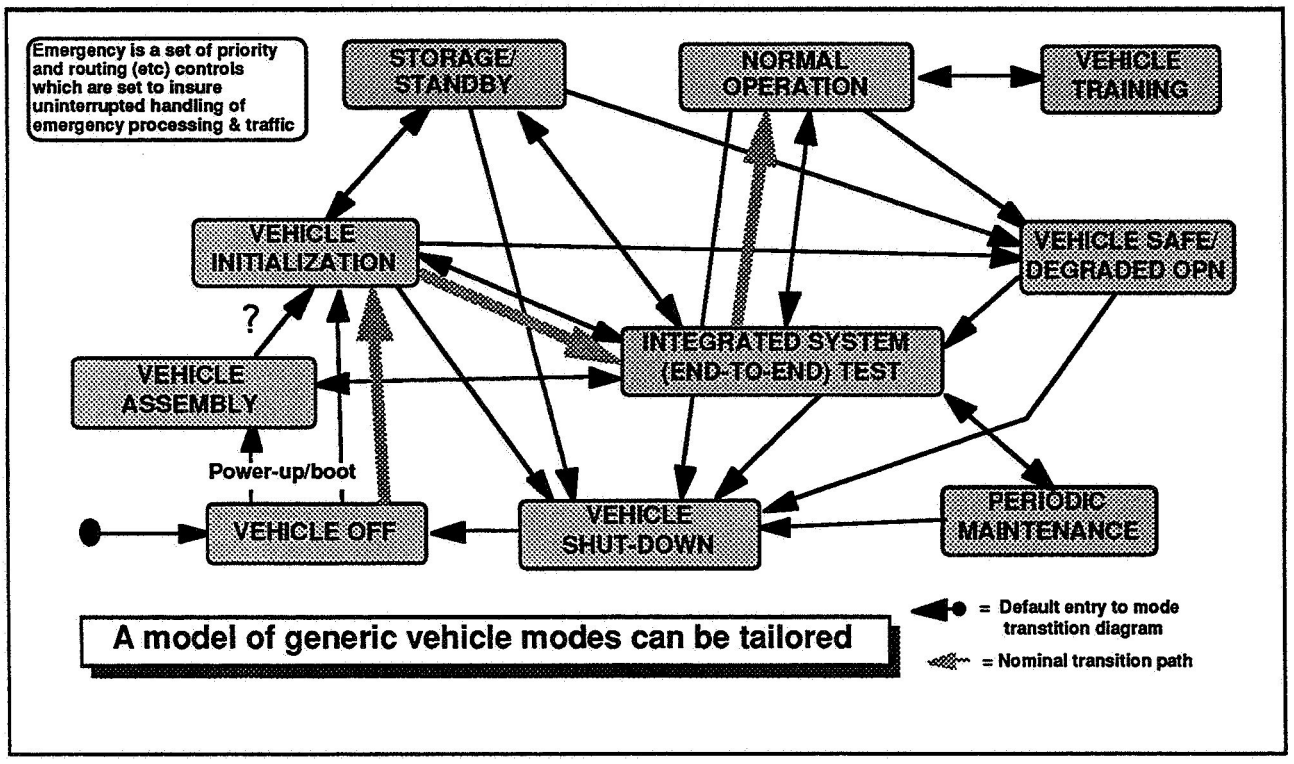

Figure 2-7. Vehicle Mode Transitions Must be Constrained

VEHICLE INITIALIZATION is the startup effort to activate operational capability in the flight vehicle, which may be full operational capability (FOC) or may be some lesser level of initial operating capability (IOC). This mode ensures that the flight vehicle subsystems are started up in the proper sequence, with appropriate safety controls, so they can integrate their operating activities correctly. Part of VEHICLE INITIALIZATION is the giving the command for a power-on self test (POST), needed to insure the vehicle is operating properly before exiting from this mode. Commanding of the POST is not the same thing as entry into the INTEGRATED SYSTEM (END-TO-END) TEST; POST is a restricted startup activity to test specific components, while INTEGRATED SYSTEM (END-TO-END) TEST is a full scale system test sequence. If the POST is failed, then entry into VEHICLE SAFE/DEGRADED OPN, INTEGRATED SYSTEM (END-TO-END) TEST, or VEHICLE SHUT-DOWN may be mandatory.

From VEHICLE INITIALIZATION, several modes may be entered: STORAGE/STANDBY, VEHICLE SAFE/DEGRADED OPN, INTEGRATED SYSTEM (END-TO-END) TEST, or VEHICLE SHUT-DOWN. The normal transition out of initialization is into INTEGRATED

SYSTEM (END-TO-END) TEST to insure the system is fully functional before normal operations are allowed to start. In some missions, the VEHICLE ASSEMBLY and VEHICLE INITIALIZATION modes might be the same thing, as denoted by the question mark in the Figure 2-7.

INTEGRATED SYSTEM (END-TO-END) TEST provides the full test control structures and test data needed to checkout a system from end-to-end after it has been initialized, or at any time when problems must be identified. This is a vehicle mode because it is used when the whole vehicle is being used in a dedicated manner for just this end-to-end testing; special test data will probably be introduced into injection points (such as at sensors) to determine if expected results occur. Interruptive test programs will probably be running to checkout all paths and operating conditions of all subsystems. This is more general than POST because INTEGRATED SYSTEM (END-TO-END) TEST interferes with normal vehicle operation, while POST is performed before a vehicle has started operating and will normally not address end-to-end system test issues. This is also different from the testing that would be conducted in NORMAL OPERATION, which would not require the attention of the entire vehicle and crew.

STORAGE/STANDBY is a mode of operation used when an initialized and tested flight vehicle is to be placed into an almost non-operating condition for some time period, from which it will later be recovered and re-used. STORAGE is used when there is no crew in place and the system must operate in a minimally powered configuration, probably just performing periodic built-in-test and awaiting commands to re-activate systems. STANDBY is used when there is crew in control; the presence of humans creates a different type of operation with a higher level of environmental and systems support. An example of vehicle STANDBY was demonstrated by the Lunar Excursion Modules while stowed with the command vehicle in the Apollo missions, prior to arrival at lunar orbit. This mode can only be entered from VEHICLE INITIALIZATION or INTEGRATED SYSTEM (END-TO-END) TEST. If the flight vehicle is being initialized but not intended for current and immediate operation, then after VEHICLE INITIALIZATION, it could (but is not required to) go into INTEGRATED SYSTEM (END-TO-END) TEST before storage in the STORAGE/STANDBY mode. Any other entry from another mode into STORAGE/STANDBY would have to be through the INTEGRATED SYSTEM (END-TO-END) TEST mode to insure the system had been tested and was functioning properly before it was left to operate in the minimal capability STORAGE/STANDBY mode.

The usual entry into NORMAL OPERATION is from INTEGRATED SYSTEM (END-TO-END) TEST. A new vehicle, one which has been powered down, or one which has been operating in a degraded mode due to failures or has been undergoing maintenance cannot be placed into normal use without testing the entire vehicle to be sure it can safely perform its mission. After the vehicle has been performing training activities, a re-entry back into NORMAL OPERATION can also be done. NORMAL OPERATION is the ordinary method of vehicle operation with all subsystems capable of activity in accordance with their normal control rules.

VEHICLE TRAINING is a special mode in which the vehicle is assigned to a training mission, such as required to prepare for more complex later missions, to verify systems and crew capabilities, or to train new crew members. VEHICLE TRAINING requires the entire vehicle to be performing training activities, probably using special training data in the systems; such data would need to be purged before exit from the VEHICLE TRAINING mode to avoid confusing systems between operational and simulated data. While in this training mode, much activity would probably be simulated in the systems and subsystems, depending on the type of training and its purpose.

VEHICLE SAFE/DEGRADED OPERATION is a mode of control used to indicate that significant failures have occurred, during which the mission operation must continue to proceed, and a lesser capability than normal can be tolerated. As a vehicle level mode, this represents a serious degradation to the vehicle's overall capability, and is not isolated in affect to just a small area. The entire vehicle is being re-oriented for survival of the crew (first), with the mission (maybe second) depending on specific conditions of the emergency. Examples of this were plans during the Apollo missions to recover the astronauts by slingshoting around the moon if landing or lunar orbit had to be bypassed due to serious problems; or the mission change in the Apollo 13 mission after the command module explosion disaster. This mode can be entered from NORMAL OPERATION, STORAGE/STANDBY and VEHICLE INITIALIZATION. Failures detected during any of these other modes might require special safing routines to be followed to achieve safe vehicle operation, especially if crew is aboard.

PERIODIC MAINTENANCE is entered from INTEGRATED SYSTEM (END-TO-END) TEST during which it is determined that either failures of sufficient magnitude exist that the entire flight vehicle must enter the "shop" for repairs, or a sufficiently long time has passed since the last periodic checkout and test of the vehicle that maintenance personnel should

check it out as a routine precaution. This mode is used in lieu of VEHICLE SAFE/DEGRADED OPERATION because the vehicle is assumed to be docked with a repair depot-like facility where the crew can safely stay while maintenance personnel checkout, test and repair any failed capabilities. Before turning the flight vehicle back to the crew, a complete INTEGRATED SYSTEM (END-TO-END) TEST would again be run to verify it was safe for the crew to resume control. Otherwise, the vehicle would be shut down and then powered off and further crew operation could not occur. In some missions or vehicles, the INTEGRATED SYSTEM (END-TO-END) TEST and PERIODIC MAINTENANCE modes might be the same thing. Note that performance of checkout activities periodically by the crew would involve use of the INTEGRATED SYSTEM (END-TO-END) TEST mode, not the PERIODIC MAINTENANCE mode. The latter is intended for specialized use with specialized software and hardware intruding into the system when the vehicle in not involved in a mission, and hence there is no crew operating the vehicle. For example, when an aircraft has returned to the hanger or a spacecraft to the vehicle assembly building, then maintenance personnel could load special maintenance versions of the operational software and use the PERIODIC MAINTENANCE mode to debug problems reported by the crew during the operational mission. This mode would not normally be available to the operational crew because the corresponding system software would be using special maintenance structures with hooks for maintenance debugging which could interfere with performance of an operational mission. However, while not recommended, any software (such as special maintenance software) can be loaded and executed by the crew if they so desire.

VEHICLE SHUTDOWN may be entered from many of the other modes as shown when the vehicle mission is completed or when it is determined that the appropriate use of the vehicle can no longer be supported and operation must be terminated, temporarily or permanently. During this mode, each of the vehicle entities is prepared for safe shut down; with backing up of any needed data bases, completion of diagnostic storage, notifications to associated vehicles in the mission or home port facilities, and implementation of proper shut down sequences for subsystems. This mode may not be entered from VEHICLE ASSEMBLY since the vehicle must be started before it is shutdown; assembly is not a start up activity, but is only a route to the startup. Finally, the vehicle must be turned to VEHICLE OFF, with no power applied to any subsystem or subassemblies.

The nominal routing for a normal vehicle start up is shown by the grey arrows in Figure 2-7. From a power-down condition, the vehicle must initialize all its subsystems in the

proper sequence. Then after initialization power-on self tests and initialization is completed (successfully), an integrated system test from end-to-end should be run to verify correct operation. Only, then can the normal vehicle operations begin, in a normal vehicle start up (with no problems encountered).

2.4 SUBSYSTEM STATES

Subsystem states describe the control of the major entities within a vehicle entity such as avionics. In this document, we are only addressing avionics entities, i.e., the hardware, software and processing procedures previously shown in Figure 2-6, and the avionics subsystem states. The analysis could be easily extended to major vehicle entities other than the avionics. The avionics subsystem states describe the control of the major avionics entities in support of the current mission and vehicle modes, as shown in Figure 2-8. The avionics subsystem entities are the major avionics subsystems, such as GN&C, C&T, SOCS and SDSS, which are needed to properly operate the flight vehicle in accordance with human command and control in the mission and vehicle modes. An actual implementation might be accomplished by designing one centralized avionics_controller (AV_CTRL) module for all avionics, or by distributing individual _CTRL modules to each subsystem. In the SSF program, AV_CTRL is done partially by ISE, MCC and the crew manually.

Each of these major avionics subsystems must be under the control of some control entity which is designated as GNC_CTRL, C&T_CTRL, SOCS_CTRL, SDSS_CTRL, etc. Since this analysis is intended to be implementation independent, an AV_CTRL entity has been defined as the sum of each of these separate _CTRL entities. Another way of looking at the AV_CTRL entity is to view all the avionics from on high, with the _CTRL functions overlapped; that overlapped set of distinct subfunctions comprises AV_CTRL, and AV_CTRL can then be tailored to each avionics subsystem's control needs.

The key is that the AV_CTRL is a processing entity which governs the interactions of its "sibling" avionics level entities. This controller determines what sets of mission, vehicle and avionics commands and other inputs meet the predefined criteria for acceptance by the avionics, under what predefined conditions, what allowable (predefined) responses can be chosen, and at what times or in what sequences. This behavioral control is based on rules and procedures established by humans, and implemented by the avionics subsystems. These are states rather than modes because they are intervening steps the avionics establish in carrying out human commands. Only states which start from and end in human-directed behaviors are allowed under the conditions set up by the human plans. As a result of these states; command, controls, and events may be established which in turn constrain or initiate activities and procedures in lower level entities.

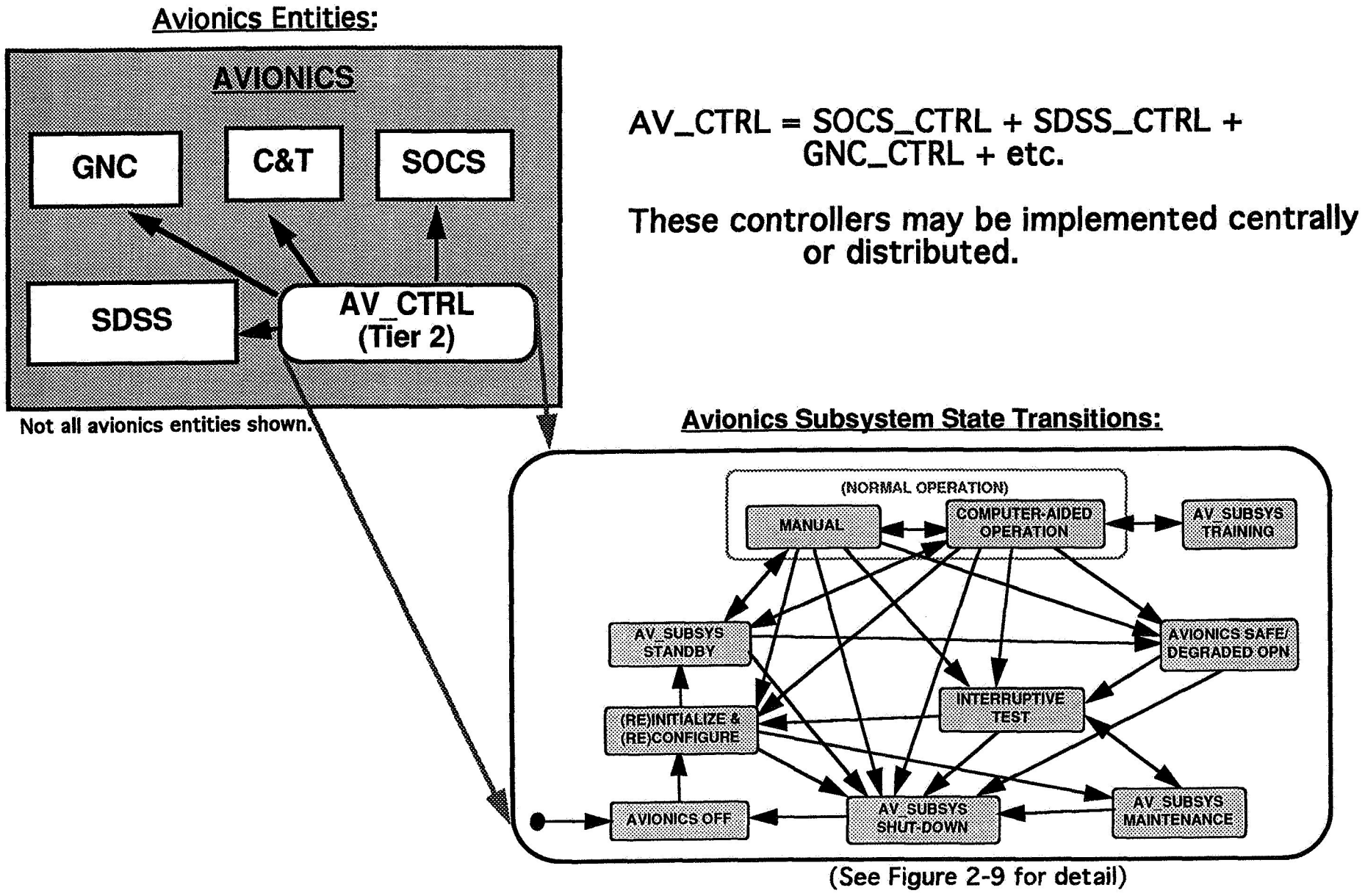

Figure 2-8. Avionics State Transitions Implement the Avionics Controls

For each mode change, there may be several intervening states until the final mode is achieved. The state and state transitions define the sets of allowable functional behavior which the AV_CTRL can effect. Each state encompasses the set of commands, controls, events, conditions and procedures needed to safely operate the avionics subsystem in a stable manner. The transitions from one state to another establish how the AV_CTRL entity transitions from one stable set of processes and condition look-up tables to another. For instance, entering MANUAL state is necessary to insure the appropriate automated controls are set up to enable a human to take manual control, otherwise, the possibility exists that as humans try to set switches for one operation, an automated routine elsewhere might be trying to counteract their efforts in accordance with its own (presumably obsolete) rules and procedures.

In a complex system with many independently designed subsystems, modules or components; each such independently designed unit will probably include its own states and state transitions, as its design team interprets the specifications. Since each avionics subsystem state consists of the sum of these states of individual subsystems and modules, the precise set of intervening states may not result as expected. Only exhaustive (and expensive) testing can identify all possible states and their interacting software paths; usually such testing can only be afforded for high priority paths.

Although each state is mutually exclusive, one avionics subsystem could be in one state while another is in a different state. So, GN&C might be in a STANDBY state while SOCS is in a TRAINING state -- this insures a training SOCS command to fire thrusters causes a simulated firing rather than a real firing. For the generic architecture, ten states are identified. These modes and their allowable transition paths are shown in more detail in Figure 2-9.

The default entry for normal vehicle operation of the avionics is for all power to be off (upon vehicle startup); power must then be applied to the normal avionics modules from some external source in order to boot the system, as discussed in the next paragraph. The application of power in VEHICLE OFF mode enables the vehicle to reach either the VEHICLE ASSEMBLY or VEHICLE INITIALIZATION mode (as shown in Figure 2-4). Avionics can thus be powered up as subassemblies during assembly or as complete subsystems during initialization. As can be seen from this point, each avionics subsystem state can support different mission and vehicle modes. However, the vehicle or mission mode change would carry specific sets of commands, controls, or events which would

indicate to the AV_CTRL which avionics subsystems were to be affected. So the impact on avionics of mode changes is due to the command, controls and events which would be detected, not directly by the human mode command per se.

AVIONICS OFF is a "holding" state during which no activity can occur until power gets applied; it exists because when systems are in orbit without any power, special planning is needed to enable power to be applied, especially under conditions of operation when a "cold boot" may be needed. From AVIONICS OFF, the application of external power or use of a self-contained battery operated boot module (perhaps in the EPS) can cause entry of the (RE)INITIALIZE & (RE)CONFIGURE state. This means that any time a module is started (by the application of power), it must first be initialized and configured before it can be used. Such initialization and configuration can be as simple as loading its startup data; or as complex as selecting an initial program load, determining the applicable load conditions, moving the software code into the appropriate processor memory, booting the code, linking the code with the appropriate data memory locations, and handing over control of activity to that software.

As indicated in the note in Figure 2-9, an emergency does not enable a different set of states, nor does it establish new functions (in general, although special emergency functions may be enabled). An emergency primarily enables a different set of priorities, routings and procedural rules which allow the system to function much faster to insure the uninterrupted handling of the data and/or traffic dealing with the emergency. It is not desirable to have a separate emergency state (or mode) because that would imply an emergency has a completely different set of functions and procedures not normally used which are only used for the emergency. Use of such emergency-only-functions would create an undue risk, since these functions might not be as safe or well designed and tested as the normal functions. Such emergency-only functions would also increase the size of the system without providing any significant increase in emergency condition safety. Since the functionality needed in an emergency is the same as (in general) or a subset of the functionality needed normally, what needs to change in an emergency is the speed of operation and the rules to be followed, not the functions. For instance, in an emergency, Remote Object Data Base (RODB) Writes and GN&C Navigation Vector Updates still are needed, but much faster (depending on the emergency).

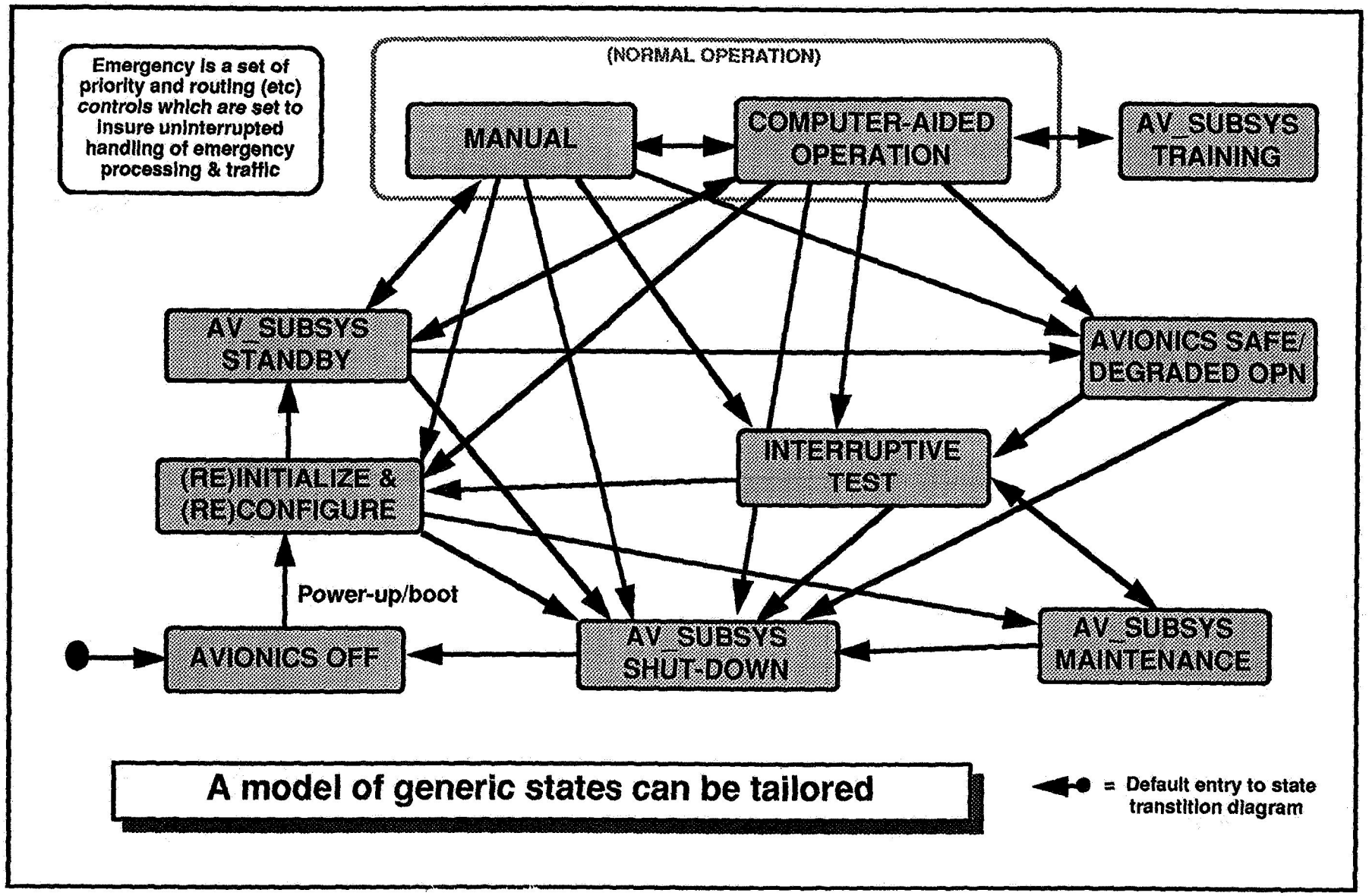

Figure 2-9. Generic Avionics Subsystem State Transitions are Constrained

The (RE)INITIALIZE & (RE)CONFIGURE state is thus the first state in which activity can take place. Its purpose is to enable the system to change the startup data sets or process configuration of end-mission applications supporting the spacecraft's mission. Re-entry to this state constitutes a "warm restart or reboot". It can be entered from most active states, except AV_SUBSYS STANDBY, AV_SUBSYS TRAINING, AV_SUBSYS SAFE/DEGRADED OPERATION and AV_SUBSYS MAINTENANCE. Entry from AV_SUBSYS STANDBY and AV_SUBSYS TRAINING is inhibited because only states in which end-mission applications processing is underway should allow reinitialization or reconfiguration; if no end-mission applications processing is underway, then there is no possibility that the crew or ground control may want to change the end-mission applications processing conditions. Thus, for example, reconfiguration training could be allowed whereby navigation applications are reconfigured to replace guidance control applications loads during an ORBIT_COAST mission mode when no guidance activity was needed, although as a training exercise, no actual reconfiguration would take place due to the lack of state connectivity. Entry from AV_SUBSYS SAFE/DEGRADED OPERATION and AV_SUBSYS MAINTENANCE is inhibited because if there is such a significant problem with the processing environment that either of these states were entered, then the possibility of distorted operating state conditions and data would necessitate either a cold restart or significant retesting and then a warm restart.

In the (RE)INITIALIZE sub-state, all subsystems in the affected modules are reset to their start up or last acceptable checkpoint condition. Power is not interrupted. Data in the affected modules is dumped (or stored for later analysis) and all stacks and other operating system references to the affected modules are reset. In the (RE)CONFIGURE sub-state, some or all resources in selected subsystems or modules are being re-directed to support other or additional tasks, or priority assignments are being changed. All subsystem resources in the affected modules are reloaded with new or alternative applications, data for the applications, or pointers and priority re-assignments, depending on the extent of the reconfiguration. Connectivity may be changed to support the re-directed assignments. From the (RE)INITIALIZE & (RE)CONFIGURE state, exit to the AV_SUBSYS STANDBY state is one possible direction while the changed system awaits a command to execute. This insures the system which has been initialized, re-initialized, configured or re-configured can stabilize and be ready to support crew modes and changes with no unexpected, transient activity; it also allows time for the crew to review and act on the state command which caused the re-configuration. Another possible exit from (RE)INITIALIZE & (RE)CONFIGURE state is to the AV_SUBSYS MAINTENANCE state, if a maintenance set

of applications and data need to and have been loaded for maintenance activity support. The other possible exit from (RE)INITIALIZE & (RE)CONFIGURE state is to shut down the system if necessary.

AV_SUBSYS STANDBY is a state in which the avionics subsystem has been initialized, processing has been activated and is awaiting commands, sensors and effectors are not operating, and in general, no end-mission applications activity is underway. It is a relatively temporary condition in which the avionics subsystem has been loaded and prepared to carry out its function, but has no direction yet to do anything while waiting for mission and vehicle mode sequences to reach a point at which the avionics subsystem must provide support. Direction to execute can come from crew commands provided in real-time or from pre-assembled timeline sequences. (In the latter case, AV_SUBSYSTEM STANDBY might be a very short, transient state.) From AV_SUBSYS STANDBY, exits to the NORMAL OPERATION (MANUAL or COMPUTER-AIDED OPERATION), AVIONICS SAFE-DEGRADED OPERATION, INTERRUPTIVE TEST or AV_SUBSYS SHUT-DOWN is possible.

The NORMAL OPERATION state for generic avionics provides for two substates in space vehicles. The first, MANUAL state, is essentially the current state of the art with manual setting of switches in a time ordered sequence, either performed by physically flipping myriad physical switches as is done on the shuttle for direct control, or by some combination of physical and software system-level switches as is done by some of the shuttle switches and will be done more extensively on station. . NORMAL OPERATIONS activities would include execution of built-in test capabilities that do not interfere with normal processing and operation of end-mission applications. For example, performance of continuous built-in-test functions would be expected in NORMAL OPERATION. NORMAL OPERATIONS would not include operation of maintenance functions by maintenance personnel; such activity would be part of the AV_SUBSYS MAINTENANCE state as described below. However, some execution of selected maintenance functions could be allowed in NORMAL OPERATIONS, depending on crew and ground control rules.

To cover contingencies requiring a greater degree of autonomous operation, a COMPUTER-AIDED OPERATION state has been provided to encompass the commands, controls, events and rules for automated (and possibly intelligent) processes which can act without apparent immediate direction. For instance, humans could set up a complex expert system to have the GN&C subsystem take star readings and the C&T select release of a satellite for

communications relay without immediate human intervention -- such activities would be based on a set of input conditions being detected (such as imminent communications loss), rules being in place to determine what alternatives might satisfy this condition, resources being available (such as communications link satellites), non-interfering mission activities underway (such as the crew is sleeping), etc. COMPUTER-AIDED OPERATION also addresses operation of vehicles such as Voyager or Pioneer which operate for extended periods of time without human intervention.

While in NORMAL OPERATION, there may be periods or circumstances when crew members want to train with specific avionics subsystems, without putting the entire vehicle into the VEHICLE TRAINING mode. AV_SUBSYS TRAINING state is provided to address those special controls needed to enable crew training on specified subsystems, with interface activity carefully controlled to insure training data is isolated from operational subsystems. Much of the data used would be simulation data, either running canned to drive the training subsystems, or being generated in real time from operating simulations in the flight vehicle or on the ground. Such simulation data would be uniquely tagged to clearly define it as simulation or training data, not to be confused with real operational data. Controls would be established to insure any operational priority issues automatically override training activities, with clearly identified "real data" markers for crew and systems to preclude any possibility of confusion when training activities are pre-empted by real mission activities. Since this state is based on use of simulations or simulated data, there is no entry from MANUAL state, because the training computer aids must first be executed (from COMPUTER-AIDED OPERATION) before TRAINING state can be executed.

An AVIONICS SAFE/DEGRADED OPERATION state is provided for operations under several circumstances. First, the VEHICLE SAFE/DEGRADED OPERATION mode may have been commanded from NORMAL OPERATION, STORAGE/STANDBY or VEHICLE INITIALIZATION mode (as shown in Figure 2-4), in which case the affected subsystems are most likely in AVIONICS SAFE/DEGRADED OPERATION (which may be temporary or permanent depending on backup and spares capability). The recognition of the VEHICLE SAFE/DEGRADED OPERATION mode is made in the avionics by detection in the AV_CTRL of the commands, controls and events generated by vehicle mode controls. A second possibility is that some subsystems may have suffered significant failures, during which mission operations must continue, but the VEHICLE SAFE/DEGRADED OPERATION mode has not been commanded.

As an avionics level state, AVIONICS SAFE/DEGRADED OPERATION represents a serious degradation to a specific avionics subsystem but not to the overall vehicle. The failure effect is isolated to a relatively contained area, subsystem or function. Failure response alternatives will be tried out, and may remedy the problem enabling a return (after INTERRUPTIVE TESTING to verify the re-established capability) to NORMAL OPERATION. While failure alternatives are being determined and have not yet been implemented, the AVIONICS SAFE/DEGRADED OPERATION state keeps other subsystems aware that there is a problem in progress. An example of this might be if a spacecraft command vehicle suffered a disastrous failure of several of its processors, with most other subsystems retaining capability and the mission could proceed, then while NORMAL OPERATION of the vehicle might be possible, AVIONICS SAFE/DEGRADED OPERATIONS might have been necessary for some subsystems until resources could be reassigned or other recovery alternatives implemented.

The INTERRUPTIVE TEST state is available to support more extensive testing than can be allowed in NORMAL OPERATIONS. The nominal procedure in built-in testing activity is to allow only that continuous built-in testing to proceed during NORMAL OPERATIONS which does not interfere with any end-mission or applications processing activities. Failure detected through these efforts call for normal or minimal impact recovery procedures as part of NORMAL OPERATIONS. If the failures are too extensive to be handled normally, then the AVIONICS SAFE/DEGRADED OPERATION state might be required to recover. Also, if the vehicle is down for maintenance, then the maintenance activity might call for more extensive testing in the INTERRUPTIVE TEST state also. Of course, crew members can command interruptive tests from MANUAL state at any time. When INTERRUPTIVE TEST is executing, those specific tests which have been apriori identified as interrupting the normal, acceptable flow of processing are selected, scheduled and execution is commanded. Unless specifically authorized as part of this state, no interruptive test could be scheduled. Thus, this state would have to include functions to verify that it was safe to select interruptive tests to perform based on the current mission profile.

If the avionics are being initialized, and the power-on self-test in the (RE)INITIALIZE & (RE)CONFIGURE state detects failures, then two possibilities are allowed. First, if the failure is sufficiently mild that NORMAL OPERATION can be entered, then INTERRUPTIVE TEST can in turn be entered to perform more extensive checks if the crew so desires or testing can be delayed until a later time (as long as the crew recognizes there has been a POST failure. The second possibility is that if the failure is too severe to enable

the system to achieve NORMAL OPERATION, then it may be necessary to load and initialize the AV_SUBSYS MAINTENANCE functions, as shown by the arrow from (RE)INITIALIZE & (RE)CONFIGURE to support more extensive repair. Entry of INTERRUPTIVE TEST from (RE)INITIALIZE & (RE)CONFIGURE is not provided because it is necessary to set up operating safeguards before INTERRUPTIVE TEST can be entered; (RE)INITIALIZE & (RE)CONFIGURE does not offer sufficient human control to do that since that is one purpose of NORMAL OPERATION and AV_SUBSYS MAINTENANCE.

The detection of NORMAL OPERATION or INITIALIZATION failures that can not be isolated, may require the more extensive testing that only INTERRUPTIVE TEST can accomplish. This state enables more extensive testing by taking an avionics subsystem off-line while more rigorous tests, such as injecting test signal data at the sensors, loading test case data for processors, running extensive dedicated tests, etc. are used to identify and isolate the fault condition or determine that there is actually no failure. Removal of the subsystem or module from the network is needed to insure that no test data or results contaminate the network. Exit from this state may take place to the (RE)INITIALIZE & (RE)CONFIGURE or the AV_SUBSYS SHUT-DOWN states. If INTERRUPTIVE TEST was entered from AV_SUBSYS MAINTENANCE, then it can return to AV_SUBSYS MAINTENANCE.

The AV_SUBSYS MAINTENANCE state is provided for the use of specialized maintenance analysis programs, operation of a maintenance version of an avionics subsystem program with special breakpoints and hooks for pulling off maintenance data, use of specialized maintenance data sets, etc. It can be only be entered if the (RE)INITIALIZE & (RE)CONFIGURE state has loaded this special set of applications first. After it has started, it may call on INTERRUPTIVE TEST to check various issues, and then continue maintenance checking and debugging. Exit from this state is by way of an avionics shut down of affected subsystems to insure any loaded code, data or operating conditions which are not ordinarily in use can be thoroughly purged from the system.

The transition to AV_SUBSYS SHUT-DOWN occurs when a VEHICLE SHUT-DOWN mode or the AV_SUBSYS SHUT-DOWN state is commanded. Such a command can be issued by SOCS, or by other authorized elements (such as special maintenance applications). AV_SUBSYS SHUT-DOWN may also be entered when an avionics subsystem fails to successfully (re)initialize or (re)configure upon command, is commanded off while in the AV_SUBSYS STANDBY state, is manually switched off, is directed by an automated process

to cycle power, has power cycled from a test routine in INTERRUPTIVE TEST or AV_SUBSYS MAINTENANCE, or is shutdown due to failures from the AVIONICS SAFE/DEGRADED OPERATION state. AV_SUBSYS SHUT-DOWN involves either cycling the power to force a cold boot restart, or to simply turn-off a specific avionics subsystem. Especially under failure conditions, there may be a need to physically shut down failed systems (or systems with failed components) to force them to stop erroneous processing and false data generation. (After they have been shut down, then additional maintenance or reconfiguration activities can take place to try and restore proper functional operation.)

2.5 FUNCTION STATES

Function states describe the control of the major functional entities within an avionics subsystem. In this document, we are only addressing the SDSS functional entities shown in Figure 2-10. The services such as standard data services, data base management, network services, and data system services are controlled by the operating system, shown in light grey shading. The operating system is also controlled as noted above because it is one component of the AV_CTRL entity. The operating system is defined to be the SDSS_CTRL entity or function. It controls its sibling SDSS functions. The analysis could be similarly performed for the functions in other avionics subsystems such as GN&C or C&T.

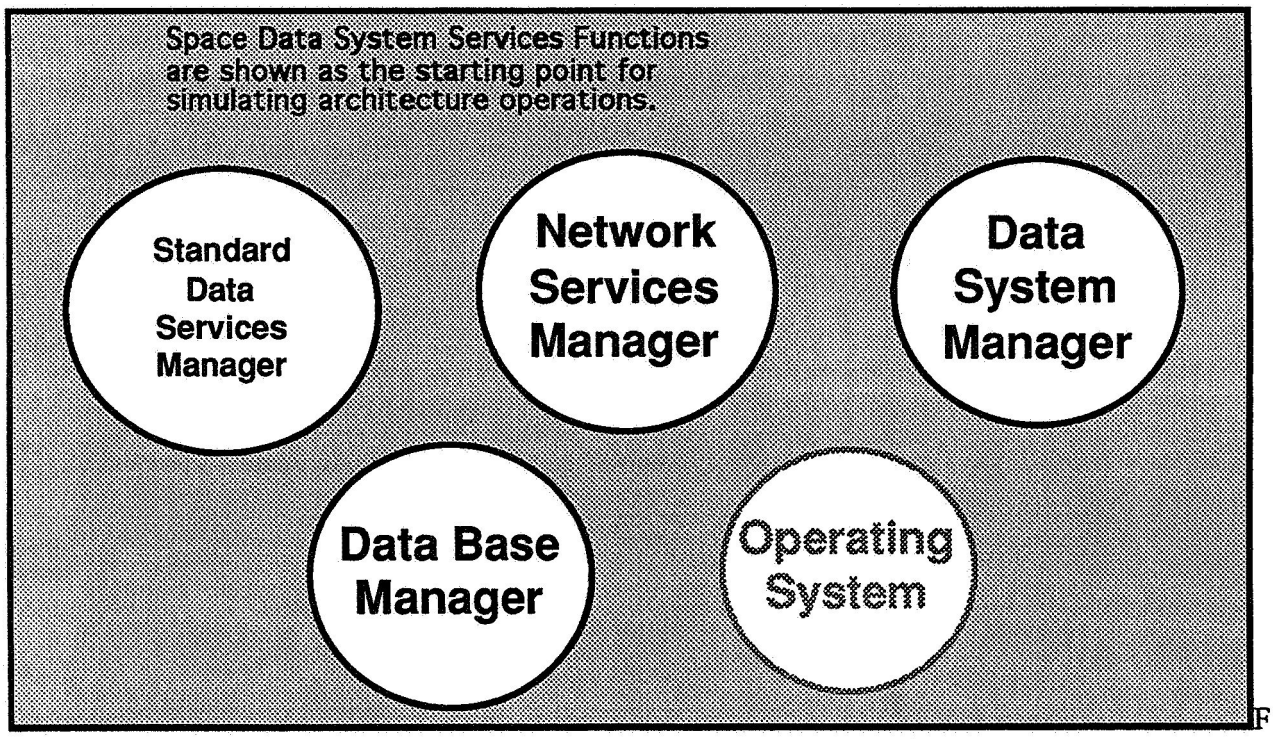

Figure 2-10. SDSS Functional Activities That Must Be Controlled

The function states describe the control of the functional entities in the SDSS. Regardless of the mission mode, vehicle mode, or avionics state, any interaction within the system in other than OFF modes or states will require some support from SDSS service functions. Figure 2-11 describes the relationships of the SDSS functional entities and the SDSS function state transitions. In the Space Station Freedom program, SDSS is performed by the Data Management System (DMS) and the SDSS_CTRL is performed by the Lynx-OS.

For each avionics state change, there may be several intervening operating system and data system service calls until the final avionics state change is achieved. The SDSS function states and state transitions define the allowable functional behavior which the operating system can effect. Each state encompasses the set of commands, controls, events, conditions and procedures needed to safely operate the data system services in a stable manner. The transitions from one state to another establish how the SDSS functional entities transition from one stable set of processes and condition look-up tables to another. For instance, SERVICE OPERATION establishes the allowable conditions for normal calls to services such as RODB Write, while SCHEDULED TESTING calls up the controls needed for specific built-in tests such as memory self-test to be run.

Since the operating system controls all interaction within data system services, it is the entity which determines what commands, controls, and events the data system will recognize and respond to, under what conditions, at what times, and in what sequences. This behavior control is based on the basic rules and procedures encoded in the operating system design. These are states because they are selected by the operating system controller based on the encoded responses allowed it for each input command, control or event and the conditions it satisfies. As experience with International Business Machines (IBM) compatible Personal Computers (PCs) and MacIntosh computers has shown, the command given by the human at the keyboard or mouse does not always translate into the system control activity anticipated by the human.

Although each state is mutually exclusive, whether or not this could be accomplished on a multi-tasking or thread basis would have to be determined case by case for each implementation. For example, using the last set of examples, one thread could be performing a RODB Write while another is performing a memory self-test. For the generic data system architecture, Figure 2-12 shows the SDSS states and allowable state transition paths.

The default entry for normal data system operation is into the SDSS_OFF state, since an external (to the SDSS) power source is needed to start or boot the system. The focus here is on taking a single processor from a "cold" condition with no power to a "warm" condition with power available. In a multiprocessor based system, this state would still be needed for each processor. As previously noted, power would be applied in the VEHICLE OFF mode

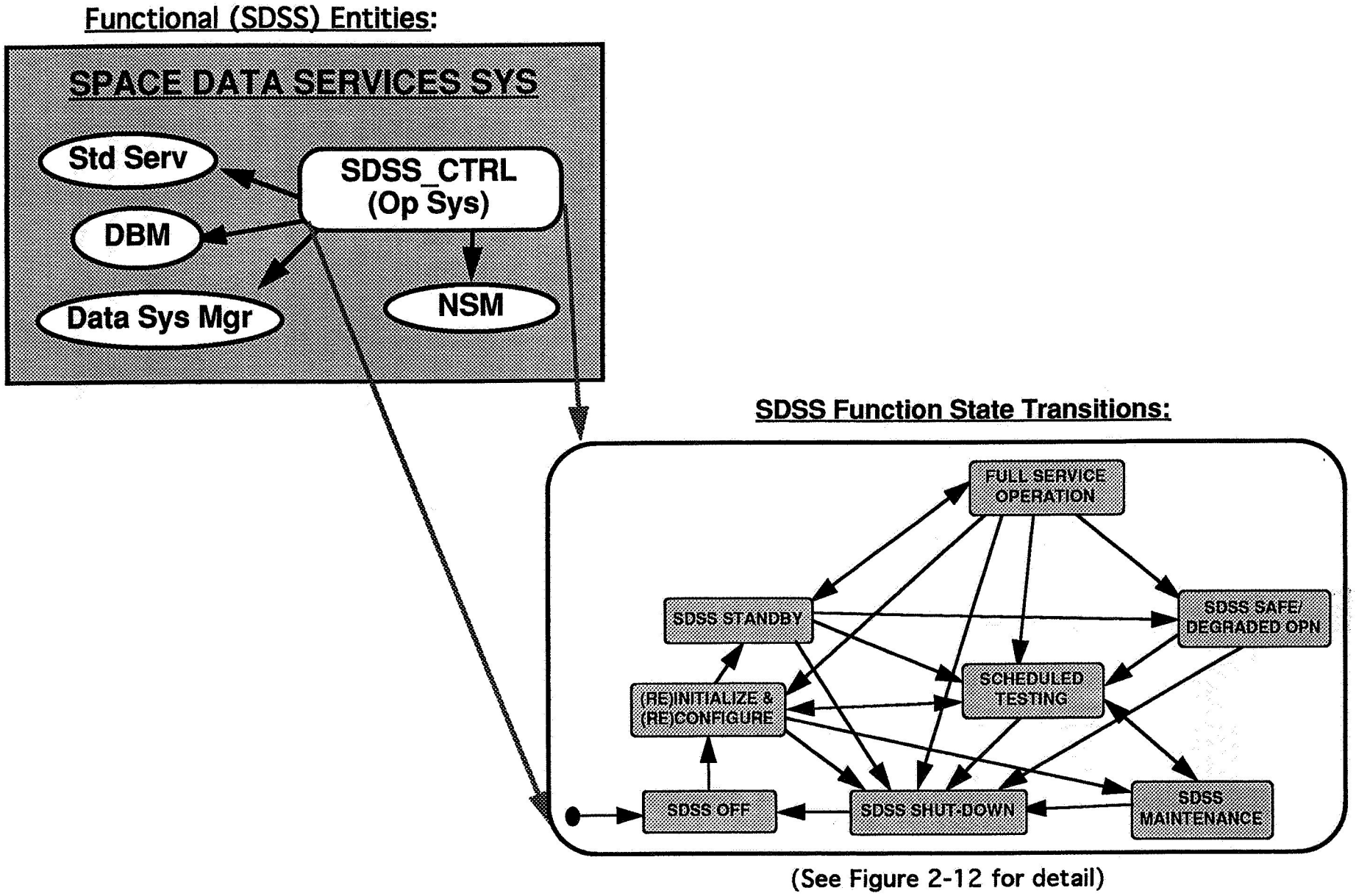

Figure 2-11. SDSS Function State Transitions Implement the System Controls

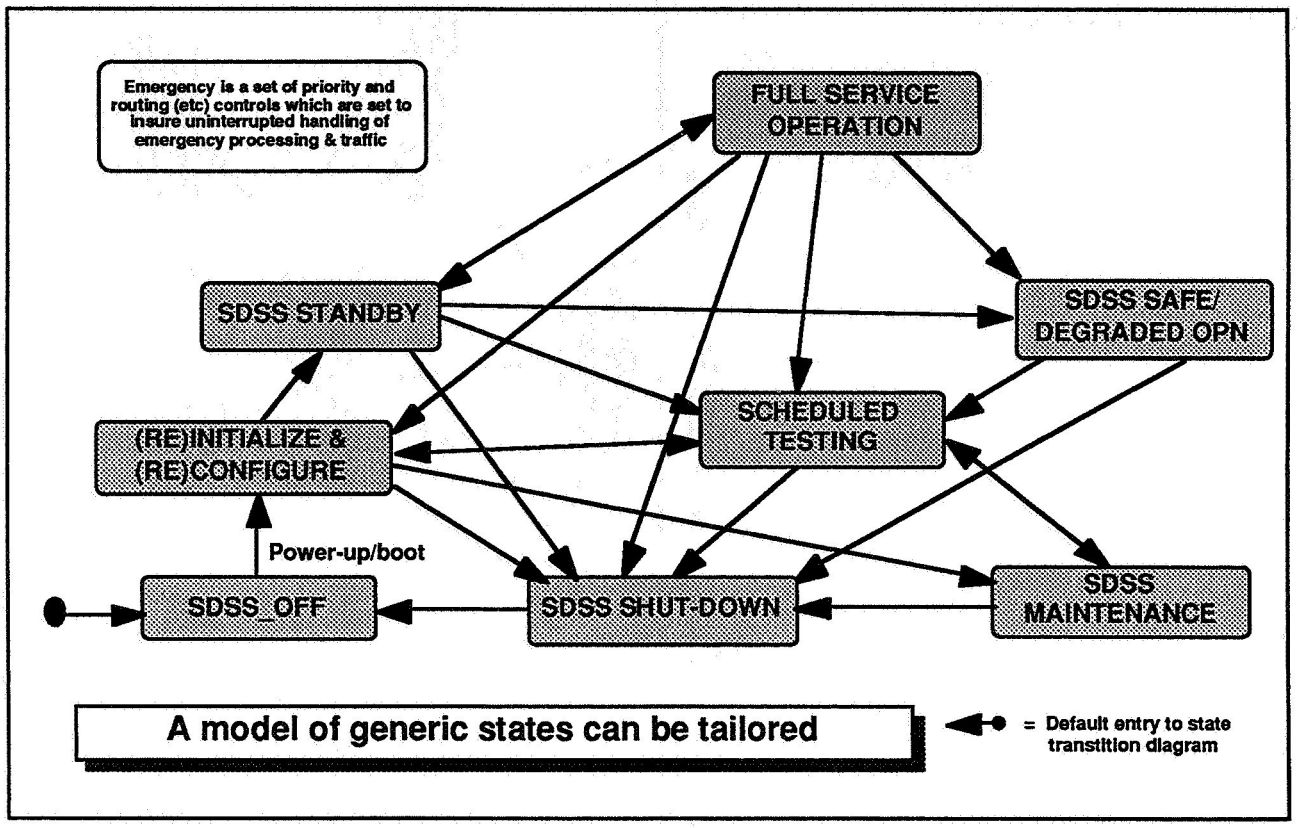

Figure 2-12. Function State Transitions Must be Constrained

and the SDSS Avionics Subsystem AVIONICS OFF state. As can be seen by this figure, the entire data system must be powered up as one entity. Once power is applied, the data system must initialize and configure all its elements.

Initialization and configuration of the data system is accomplished by entering the (RE)INITIALIZE & (RE)CONFIGURE state. When the data system is in the SDSS_OFF state, the entire avionics can not operate, and is essentially just "cold iron". Only the power subsystem, or any battery-backed specialized subsystems can operate, and they can only operate at a minimal level of built-in capability, such as a special battery timer with a boot-up read only memory device to kick start the application of power to the SDSS initialization routines.

When power is applied to the data system for the first time, it immediately enters the (RE)INITIALIZE & (RE)CONFIGURE state by starting its boot up and self initialization routines. This includes power-on self test, resource availability detection, detection and

initiation of designated applications, allocation of resources to applications, etc. Re-entry into this state later causes either a warm or cold restart, or a reconfiguration, depending on the condition in effect at that time. The normal exit from this state is to enter standby. An alternative exit is in support of maintenance activities. Otherwise, it can only transition to the SDSS SHUT-DOWN state if it is unable to successfully startup and configure.

After the data system has stabilized in an initialized, configured condition, it enters the standby state where it awaits an external command to start servicing activities, i.e., it is "warm", checked out, and ready to perform its function.. Upon direction, the data system enters the FULL SERVICE OPERATION state. Such direction may be explicit by crew, by timelines, or by special startup configuration controls. In the FULL SERVICE OPERATION state, the data system responds to service requests to support external applications (such as GN&C requesting a RODB Write). A service request to perform a data system test service (such as a SOCS call for or timer initiated performance of built-in testing) causes a transition into the SCHEDULED TESTING state. The only other possible transitions from SDSS STANDBY are to SDSS SAFE/DEGRADED OPERATION or to SDSS SHUT-DOWN. A transition from SDSS STANDBY into SAFE/DEGRADED OPERATION would occur when the system has successfully initialized with configurations such as use of partially-failed components, insufficient resources, or missing applications or modules. A transition to SDSS SHUT-DOWN could be commanded by the crew or SOCS while SDSS is in standby.

The FULL SERVICE OPERATION is the normal state of service for external applications, and integrates the behavior of the SDSS functions for standard services, data services, network services and data system services. It establishes the calls to the operating system to initiate these services, how they are handled, under what conditions they are accepted or rejected, what happens to call errors, etc. Direction can also be handled to return to the STANDBY state. This state also includes the support SDSS might have to provide to enable end-mission applications software to load and initialize. If commanded, this state can transition back to (RE)INITIALIZE & (RE)CONFIGURE for configuration changes or reboots, can change to the SCHEDULED TESTING state to run specific built-in test diagnostics, or can transition to the SDSS SAFE/DEGRADED OPERATION state if failures preclude normal operation.

The SCHEDULED TESTING state supports any directed test, whether it is off-line, on-line but not interfering, or disruptive of end-mission applications or other activities. This is a relatively "dumb" state which simply knows what <u>all</u> possible tests are, how to add more

tests if directed and how to insure the tests are safely executed. It does not know how to determine if a test is appropriate to the current operational modes or crew activities; such a determination would be made by as part of a higher level mode (such as NORMAL OPERATIONS) or state (INTERRUPTIVE TEST) functional check. This state includes special functional control checks required to insure this state is properly called and authorized; that is why a separate state is used. Another reason for providing this state is to facilitate manual control over testing and manual override over system initiated tests. Such manual or override control could more easily bypass any built-in restrictions and conditions tests included as part of higher level mode or state condition checking.

The SCHEDULED TESTING state may be entered from any power-up state in the SDSS Function State diagram. Entry from (RE)INITIALIZE & (RE)CONFIGURE is provided to support initialization power-on self testing. Entry from SDSS STANDBY is provided to enable system checks to continue while in a standby condition to insure the system will continue to operate when called upon. Entry from SDSS SAFE/DEGRADED OPERATION is provided to enable testing and re-testing of failed or partially failed elements in SDSS. Normal exit from SCHEDULED TESTING is either to power down and re-boot, or to reinitialize to insure no test data and results contaminate normal operational service. While maintenance activities are in process, exit to the SDSS MAINTENANCE state is acceptable also.

In the event failures force a transition into the SDSS SAFE/DEGRADED OPERATION state, special routines and priorities appropriate to the conditions of the failure, and authorized by the crew, would be instituted to enable crew survival, mission survival, and/or safety of systems. In the event an emergency is declared by the crew, then SAFE substate conditions would apply selected elements of the special emergency priorities and routings that are allowed only in emergencies.

If maintenance personnel are loading special versions of the operating system with maintenance diagnostics added, with extended built-in testing, or with debugging hooks enabled, then a direct transition from (RE)INITIALIZE & (RE)CONFIGURE to the SDSS MAINTENANCE state may be performed. From the SDSS MAINTENANCE state, more direct maintenance control over the system is possible, using specialized maintenance versions of the operating system and data tables as needed. Special calls to the SCHEDULED TESTING state can be effected, or a shutdown and restart can be done.

3. CONCLUSIONS AND RECOMMENDATIONS

3.1 CONCLUSIONS

There are four levels of modes and states possible in a system as summarized by Figure 3-1.

- The Mission Mode level governs the interactions of mission entities (e.g., vehicles, ground control, TDRSS, and astronauts) in the external environment with a flight vehicle that have a direct affect on the mission success.
- The Vehicle Mode level governs the interactions of subsystem entities (e.g., power, propulsion, avionics, and crew) within the vehicle which are directly responsible for accomplishing the mission of the vehicle. There is a separate Vehicle Mode diagram for each mission entity.
- The Subsystem State level identifies all the interactions of functions or functional entities within a vehicle subsystem which must interoperate to support the mode decisions of the crew and ground control. There is a separate Subsystem State diagram for each vehicle subsystem entity. In this document, the avionics subsystem functions (e.g., GN&C, C&T, SOCS and SDSS) are used to show the Avionics Subsystem State interactions.
- The Function State level identifies all the interactions of the processes within an vehicle subsystem's function which must interoperate to perform the function's defined purpose or to service requests for support from any other function. There is a separate Functions State diagram for each subsystem's functional entity. In this document, the Space Data System Services function state interactions for its processes; e.g., standard services, data base management, data system management, and network services; are shown and used as an example.

Each of the modes and states in such a hierarchy (as shown in Figure 3-1) behaves differently in supporting an actual mission scenario with its attendant mode and state changes. As shown in the figure, a mission mode change from COAST to BOOST, at the Mission Mode level causes operational activities to take place for all mission elements to insure they are functioning in a cooperative manner, and that multi-vehicle/center mission activities are properly sequenced.

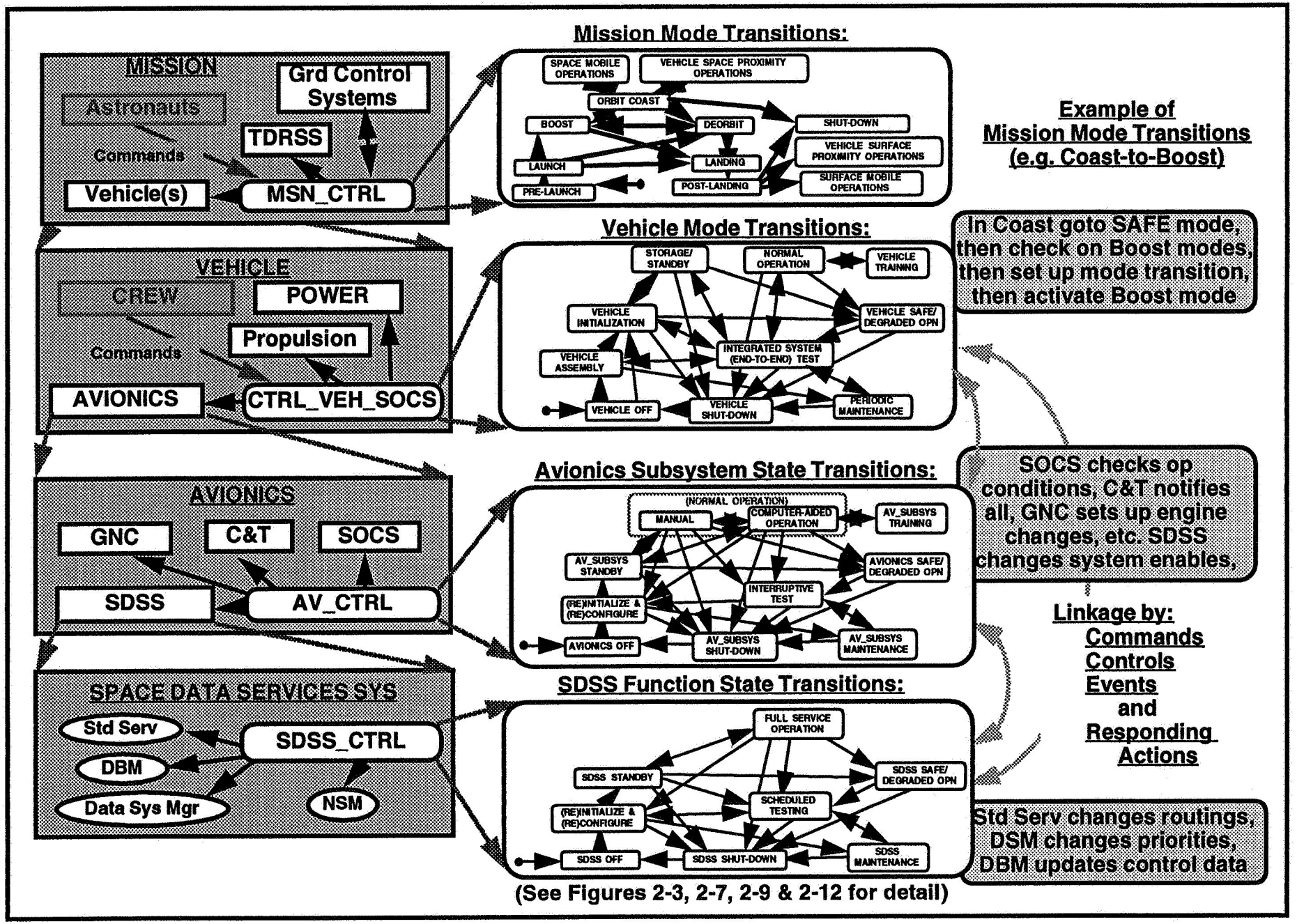

Figure 3-1. Behavior of Modes, Functions and States must be Integrated

Within a vehicle, the Vehicle Mode level then insures proper sequencing of safe vehicle activities, such as verifying and authenticating the mode change command, determining the BOOST mode requirements for the flight vehicle, setting up the transition, issuing the vehicle commands and controls which cause the Vehicle Mode level subsystems to behave properly, then activating the BOOST mode subsystem activities.

There is a Subsystem State transition diagram for each of the Vehicle Mode subsystems. For this example, only the Avionics Subsystem State transitions are addressed. In this Avionics Subsystem State level, SOCS is the overall operational controller working through the AV_CTRL controller, with responsibility for checking all operating conditions, vehicle subsystem status, and directing SDSS to distribute commands and controls to the appropriate avionics subsystems. SOCS directs C&T to distribute telemetry and ground control command acknowledgments back to the ground, directs GNC to follow the provided sequence of thruster firings, etc.

For each Avionics Subsystem State, there is a Function State transition diagram. For this example, only the Space Data System Services Function State transitions are addressed. At the SDSS Function State level, the operating system (SDSS_CTRL) is the overall controller, with responsibility for controlling the interaction of all SDSS services.

Some of the ramifications of this modes and states structure are described below.

The Mission and Vehicle Modes chosen by crew or ground control, and the Avionics Subsystem States subsequently selected, make no apparent difference on the definition of process states and their transition structure, as shown at the Function State transition level. Thus, whether the system is in the MANUAL Avionics Subsystem State or in the TRAINING Vehicle Mode has no apparent affect on the structure of the avionics functional processes -- i.e., the same guidance or operating system functions are needed in both cases. What is affected is the capability of these functional processes to operate, which depends on what conditions and controls are true/false (not their transition structure). The differences between the MANUAL state and TRAINING mode are not visible below the avionics subsystem level -- again, low level processes are required in any case. Mode changes can cause a switch from one state model to another, or from one set of allowable behaviors in a model to another set of allowable behaviors. Since it is the avionics functional processes and their state transitions which are actually implemented in hardware and software, this means their design is relatively insensitive to changes in desired use of modes and avionics

subsystem states, which can thus be definitized later without drastic cost impact (as long as their scope is consistent).

The mode commands cause different sets of lower level commands, controls, events and conditions to govern when activities take place, but this is a use of the state transition diagrams, not impacting the structure of the diagrams. Thus, for example, the same services offered by SDSS are available for any subsystem or function to call upon regardless of the modes of the system. However, some modes will not permit some operations to take place. The implementation of activities at any level may create events or cause conditions to become true/false; which in turn can be detected at other levels to cause further activities to take place at other levels.

At higher levels of states and modes, the effects of crew or ground control desires become more pronounced. Specific crew or ground control desired actions can be broken into their component parts using this model of system behavior in order to determine if the system will respond the way the developers intended, and whether that response is desired by the human users.

This analysis identified the boundary conditions applicable to a simulation of the SDSS function state. Development of such a simulation is underway to define the behavioral responses of the SDSS to possible activities generated in the other modes and states of an advanced generic avionics system.

3.2 RECOMMENDATIONS

The key question which needs to be understood in developing an advanced avionics system is:

How will the system behave?

While it is relatively easy is to identify the functions, inputs and outputs of the system; it is much harder to define how they must work or interact since that is intimately involved in the scenarios the crew might choose to follow, and in the responses they might choose to make. However, without knowledge of this behavior it is impossible to insure that the system will function the way the designers intended. Thus, this analysis defines a paradigm for organizing the possible behaviors of concern in developing an advanced avionics system.

A key follow-on that needs to be accomplished is development of simulations for this behavioral model. A mission mode simulation could enable the crew to investigate mission behaviors in a documented manner that could be captured by developers. A vehicle mode simulation could enable the potential interactions of the crew and subsystems to be investigated in a rigorous manner. An avionics subsystem state simulation could investigate the behavior of the system in different situations. A function state simulation for each of the avionics functions could establish the boundaries of acceptable behavior, and identify unacceptable system/subsystem responses. Definition of such behavioral patterns prior to system design or construction would reduce the costs of re-designing or re-building the systems or subsystems to meet unexpected and unacceptable behaviors discovered in development or flight testing.

APPENDIX A
LIST OF REFERENCES

APPENDIX A
LIST OF REFERENCES

[ISO7498] "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", International Standards Organization, 1984,.

[POSIX91] "Draft Guide to the POSIX Open Systems Environment", P1003.0/D14, IEEE Computr Society, November 1991.

[SUL92] Sullivan, B. (Team Chairman), "Integrated Avionics Software Description - PSAR Review Copy, "IBM, 6 November 1992.

[WRA91] Wray, R. B., "Requirements Analysis Notebook for the Flight Data Systems Definition in the the Real-Time Systems Engineering Laboratory (RSEL)," Job Order 60-430 for the JSC, LESC-29702, December 1991.

[WRA93] Wray, R. B. and Stovall, J. R., "Space Generic Open Avionics Architecture (SGOAA) Reference Model Technical Guide," Job Order 60-911 for the JSC, LESC-30374-A, NASA CR-188246, December 1992.

DISTRIBUTION LIST FOR LESC-30749
AN ANALYSIS OF THE MODES AND STATES FOR GENERIC AVIONICS

NASA

EK111/D. M. PRUETT (5)
PT4/E. M. FRIDGE (5)
AMES/E. S. CHEVERS (10)
JM-2/S. McDONALD (15)
EG1/D. P. BROWN
EG111/K. J. COX
JPL/MS 301-235/A. HOOKE
IA13/D. STONE

LESC

C18/J. R. THRASHER
C18/E. A. STREET
C18/R. E. SCHINDELER
C18/G. Y. ROSET
C18/J. STOVALL (10)
C106/P. G. O'NEIL
C07/J. E. MOORE

C87/M. W. BRADWAY
(FOR SATWG)

C18/JEAN FOWLER
(MASTER + 2 COPIES)

C18/G. L. CLOUETTE
C18/R. W. WRAY (10)
C18/M. W. WALRATH
C18/B. L. DOECKEL
B11/G. J. MOORMAN
C83/S. J.THOMAS
C22/D. CRAVEY
C29/P. HOPKINS

B15/LESC LIBRARY (2)

MITRE, 1120 NASA ROAD 1, HOUSTON, TX 77058
D. ERB

UHCL, UNIVERSITY OF HOUSTON - CLEAR LAKE, 2700 AY AREA BLVE. -
BOX 444, HOUSTON, TEXAS 77058
CHARLES HARDWICH

NIST/CSL, FRITZ SCHULTZ, BLDG 225, ROOM B266, GATHISBURG, MD. 20899

ROME LABS/OCTS, GRIFFIS AFB, NY 13441-5700
RICHARD WOOD

LASC, 86 S. COBB ST.,MARRIETTA, GA. 30063
RICK HARWELL, D/73-D2, ZONE 0685
JOHN WEAVER, D/73-D1, ZONE 0685
COX, JIM, D/73-MA ZONE 0081
REED, MIKE, D/73-MA, ZONE 0081
HUDSON, ROCKY, D/73-D2, ZONE 0685

DISTRIBUTION LIST FOR LESC-30749
AN ANALYSIS OF THE MODES AND STATE FOR GENERIC AVIONICS - CONTINUED

LMSC, 1111 LOCKHEED WAY, SUNNYVALE, CA. 94088-3504
CHARLES TADJERAN, ORG. 62-31, BLDG 150
ROY PETIS, ORG. 78-10, BLDG 264
JOHN McMORRIS, ORG. 81-90, BLDG 157
DUWAYNE DICKSON, ORG. 80-06, BLDG 154
F. L. (FRED) LORY, ORG. 68-15, BLDG 104
MERLIN DORFMAN, ORG. 62-80, BLDG 563

LMSC (RD&D), 3251 HANOVER STREET, PALO ALTO, CA 94303-1191
BILL GUYTON, ORG. 92-20, BLDG 254E
RAY MUZZY, ORG. 90-21, BLDG. 254E
STEVE SHERMAN, ORG. 96-10, BLDG 254E

LOCKHEED-SANDERS, 95 CANAL ST. NASHUA, NH 03061
RAY GARBOS (NAM5D-5002)
JOHN MILLER (NCA 09-1106)
JEFF E. SMITH (PTP2-B002)
DUNCAN MOORE (MER 24)

LADC, P. O. BOX 250, SUNLAND, CA. 91041
ALEX LOEWENTHAL, DEPT. 25-14, BLDG 311

LAD, P. O. BOX 17100, AUSTIN, TX. 78744-1016
CURTIS WELLS, ORG. T2-10, BLDG 30F

LOCKHEED CORP, 4500 PARK GRANADA BLVD, CALABASIS, CA 91399-0310
MICHAEL CARROLL
BART KRAWETZ

LAS Ontario, P. O. BOX 33, ONTARIO, CA. 91761-0033
C. R. (BOB) FENTON

LFWC, P. O. BOX 748, FORT WORTH, TX 76101
PAUL DANIEL, MAIL ZONE 2640

LSOC, 1100 LOCKHEED WAY, TITUSVILLE, FL 32780
L. J. (LEWIS) BOYD, ORG. 32-40, (Z/LSO-183)
ARTHUR EDWARDS, ORG. 11-42, BLDG. B/DX-D, Z/LSO-284)

DISTRIBUTION LIST FOR LESC-30749
AN ANALYSIS OF THE MODES AND STATES FOR GENERIC AVIONICS - CONCLUDED

MR. MARTIN FREED, (ASC/ENASC), 5565 BARBANNA LANE, DAYTON, OH 45415

ASC/YFMXT, WRIGHT-PATTERSON AFB, OH 45433
MR. BYRON STEPHENS

NAVAL AIR WARFARE CENTER, AIRCRAFT DIVISION, WARMINSTER, PA 18974-0591
RICHARD J. PARISEAU/CODE 102A
RICHARD S. MEJZAK/CODE 2021

EER SYSTEMS INC., 3027 MARINA BAY DR., SUITE 105,
LEAGUE CITY, TX 77573
RAY HARTENSTEIN

RESEARCH ANALYSIS AND MAINTENANCE INC., 512 AUDUBON ST.,
LEAGUE CITY, TX 77573
ROGER EVANS