DOT/FAA/AR-07/39

Air Traffic Organization
Operations Planning and Development
Office of Aviation Research
Washington, DC 20591

Real-Time Operating Systems and Component Integration Considerations in Integrated Modular Avionics Systems Report

August 2007

Final Report

This document is available to the public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof. The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the objective of this report. This document does not constitute FAA certification policy. Consult your local FAA aircraft certification office as to its use.

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.act.faa.gov in Adobe Acrobat portable document format (PDF).

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-07/39		

4. Title and Subtitle	5. Report Date
REAL-TIME OPERATING SYSTEMS AND COMPONENT INTEGRATION CONSIDERATIONS IN INTEGRATED MODULAR AVIONICS SYSTEMS REPORT	August 2007
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Jim Krodel[1] and George Romanski[2]	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
[1]United Technologies–Pratt & Whitney Aircraft 400 Main Street East Hartford, CT 06108 [2]Verocel 234 Littleton Road, Suite 2A Westford, MA 01886	
	11. Contract or Grant No. DTFA03-03-P10486

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Federal Aviation Administration Air Traffic Organization Operations Planning Office of Aviation Research and Development Washington, DC 20591	Final Report
	14. Sponsoring Agency Code AIR-120

15. Supplementary Notes

The Federal Aviation Administration Airport and Aircraft Safety R&D Division COTR was Charles Kilgore.

16. Abstract

The combination of rigorous development assurance and verification assurance has led to safe and reliable operation within civil aviation systems and equipment. Historically, such systems were designed as federated architectures, yet significantly successful efforts in integrated modular avionics (IMA) system integration have occurred, such as the Boeing 777 aircraft.

This technical report can be used to formulate a basis for evaluating the integration of real-time operating systems (RTOS) and other associated modules that support partitioning in space, time, input/output, communications, and other shared resources on an IMA system.

Several role players (platform supplier, RTOS supplier, application supplier, and the IMA system integrator) in IMA system development are discussed, and their roles of integrating multiple functions at different integration stages are detailed.

17. Key Words	18. Distribution Statement
Integrated Modular Avionics, Software, DO-178, DO-297, RTOS, Partitioning	This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	51	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AC	Advisory Circular
ACR	Avionics Computer Resource
AIMS	Airplane Information Management System
APEX	Application/EXecutive
API	Application Programming Interface
ARINC	Aeronautical Radio, Inc.
ARP	Aerospace Recommended Practice
ATM	Air Traffic Management
BSP	Board support package
CNS	Communication, Navigation, Surveillance
COTS	Commercial off-the-shelf
CPU	Central processing unit
EUROCAE	European Organisation for Civil Aviation Electronics
FAA	Federal Aviation Administration
FHA	Functional hazard assessment
I/O	Input/output
IMA	Integrated modular avionics
LRU	Line replaceable unit
MMU	Memory management unit
MOS	Module Operating System
OS	Operating System
POS	Partition Operating System
RAM	Random access memory
ROM	Read only memory
RTOS	Real-time operating system
SAE	Society for Automotive Engineers
SC	Special Committee
SPDA	Secondary power distribution assembly
SSA	System safety assessment
TSO	Technical Standard Order
WCET	Worst-case execution time
XML	eXtensible Markup Language

EXECUTIVE SUMMARY

The combination of rigorous design assurance and verification assurance has led to safe and reliable operation within civil aviation systems and equipment. Historically, such systems were designed as federated architectures, and although some significantly successful efforts in integrated modular avionics (IMA) system integration have occurred, such as the Boeing 777 aircraft, documentation of practices and guidelines for integrating such complex systems is lacking.

This technical report presents the results of a research effort intended for use by both the certification authorities and industry to formulate a basis for evaluating the integration of real-time operating systems (RTOS) and other associated modules that support partitioning in space, time, input/output, communications, and other shared resources on an IMA system.

This report covers the subject of software module and component integration with an emphasis on RTOSs in airborne systems. These modules and components include RTOSs, schedulers, communications, board support packages, system health monitors, configuration databases, and software tools used specifically to assist in IMA system integration.

Integrating multiple applications on a common IMA system provides benefits and challenges. Sharing information is easier on a system specifically designed to exchange data in a controlled way. However, the challenge is raised in an integrated system where common mode failures may be introduced, and dependencies may exist that would not be present in a federated system.

This report presents approaches to apply system safety assessment methods to IMA systems. Several role players (platform supplier, RTOS supplier, application supplier, and IMA system integrator) in IMA system development are discussed, and their roles of integrating multiple functions at different integration stages are detailed. Each module or component built for integration into the system, software, or hardware has certain commitments to other modules or components as they are integrated at various integration stages. These commitments come in the form of limitations, assumptions, performance restrictions, behavioral restrictions, configuration features, and other shared resource considerations.

The IMA system and its associated RTOS(s) are featured in this report, along with the difficulties that occur for RTOSs when sharing such resources. Particular attention is given to IMA system schedulers and how application worst-case execution time predictions to accommodate determinism are very difficult to accomplish with sophisticated RTOSs in such systems. A companion handbook has been written on this topic that discusses in more detail the activities and responsibilities of the various role players when developing an IMA system.

1. INTRODUCTION.

1.1 PURPOSE.

The purpose of this report is to aid the aviation industry and the certification authorities in the integration of real-time operating systems (RTOS) and other associated modules and components that support partitioning in space, time, input/output (I/O), communications and other shared resources, and their integration to form operational systems.

1.2 SCOPE.

This report covers the subject of hardware and software module and component integration with a focus on software integration in airborne systems. The report addresses modules and components in context with integrated modular avionics (IMA) systems. These include RTOSs, schedulers, communications, board support packages (BSP), system health monitors, configuration databases, and software tools used specifically to assist in IMA system integration. A companion Handbook has been written on this topic that discusses in more detail the activities and responsibilities of the various role players when developing an IMA system [1].

1.3 BACKGROUND.

This work was based on previous studies conducted for the Federal Aviation Administration (FAA) that discussed issues regarding the use of software, electronic hardware, and commercial off-the-shelf (COTS) components in aviation systems. This work resulted in two reports [2] [3].

The COTS software report [2] provides a snapshot of portions of the industry domains related to safety. Avionics, nuclear, medical, space, and elevator domain information was surveyed. Key industry COTS components were identified, and potential alternate methods for verifying a COTS component's applicability to the avionics domain application were studied. RTOSs and communications software emerged as key eminent COTS technology offerings in the aviation community.

The COTS electronic hardware report [3] provided findings about the state of the industry relative to the design objectives identified in the RTCA guidance document DO-254, with a focus on the implications of the use of COTS electronic hardware components in safety-critical airborne systems. The use of complex electronic hardware components in airborne systems poses a challenge to meeting safety requirements. For complex components, complete verification is at best very difficult, and at worst not achievable.

A follow-on study provided a detailed look into the safety and compliance issues of using a COTS RTOS in aviation applications and further explored the COTS RTOS domain and its safety implications [4]. RTOS attributes were detailed and their safety-related properties were discussed, along with considerations to address when integrating a COTS RTOS with an application in an aviation system. A stress or robustness test was presented as an example for the basis of an RTOS vulnerability analysis.

Another study documented the safety implications of a partitioned COTS RTOS and its integrated architecture features [5]. Some features included the following:

- BSPs that isolate the RTOS from the target computer
- Central processing unit (CPU) chip cache jitter between partition switches
- Analysis of worst-case execution time (WCET) estimations on pipelined microprocessors
- Health monitoring and associated fault handling
- System integration

The integration of the RTOS and other software modules and components into the overall IMA system was identified as an area of further study, and is the subject of this report.

1.4 RELATED GUIDANCE, INDUSTRY ACTIVITIES, AND DOCUMENTS.

The following documents relate directly to the issues addressed herein. A brief description of their applicability to this study is provided.

a. DO-178B, "Software Considerations in Airborne Systems and Equipment Certifications" and amendment 1 (October 1999), contains guidance and standards concerning software development and certification issues for the operational safety of the aircraft. The FAA recognizes DO-178B as an acceptable means of compliance for airborne software in Advisory Circular (AC) 20-115B. ED-12B is the European Organisation for Civil Aviation Equipment (EUROCAE) equivalent of DO-178B [6].

b. DO-248B, "Final Report for Clarification of DO-178B 'Software Considerations in Airborne Systems and Equipment Certification'" (December 2001). The purpose of this document is to provide clarification of the guidance material in DO-178B. The material clarifies a specific section or topic of DO-178B. This document is also intended to resolve any inconsistencies between DO-178B and any other relevant civil aviation standards. ED-94B is the EUROCAE equivalent of DO-248B [7].

c. DO-254, "Design Assurance Guidance for Airborne Electronic Hardware" (April 19, 2000). This document provides guidance for design assurance of airborne electronic hardware from conception through initial certification and subsequent postcertification product improvements to ensure continued airworthiness. The FAA recognizes DO-254 as an acceptable means of compliance for airborne electronic hardware in AC 20-152. ED-80 is the EUROCAE equivalent of DO-254 [8].

d. DO-255, "Requirements Specification for Avionics Computer Resource (ACR)" (June 2000). This document contains the requirements for an Avionics Computer Resource (ACR), which provides the shared computation resources, core software, and signal conditioning necessary to interface with a variety of aircraft systems. The ACR provides a highly adaptable computer environment for a variety of software applications. This document is intended to assist manufacturers and software application suppliers and installers in achieving an ACR and installed software applications that will perform their intended functions satisfactorily under all conditions normally encountered in routine aeronautical operations. ED-96 is the EUROCAE equivalent of DO-255 [9].

e. DO-278, “Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance” (March 2002). This guidance applies to software contained in CNS/ATM systems used in ground- or space-based applications shown by a system safety assessment process to affect the safety of aircraft occupants or airframe in its operational environment. ED-109 is the EUROCAE equivalent of DO-278 [10].

f. DO-297, “Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations” (November 2005). This guidance applies to IMA system development for application developers, integrators, certification applicants, and those involved in the approval and continued airworthiness of IMA systems in civil certification projects [11]. EUROCAE does not currently have an equivalent for this document.

g. FAA Order 8110.49, “Software Approval Guidelines” (June 3, 2003), Chapters 4, 5, 7, 8, and 12. This order guides Aircraft Certification Service field offices and designated engineering representatives on how to apply DO-178B for approving software used in airborne computers [12].

h. Society of Automotive Engineers (SAE) Aerospace Recommended Practice 4754 (ARP 4754) “Certification Considerations for Highly Integrated or Complex Aircraft Systems” (1996). This document discusses the certification aspects of highly integrated or complex systems installed on aircraft, taking into account the overall aircraft operational environment and functions. It is intended primarily as guidelines for use with electronic systems, which, by their nature, may be complex and are readily adaptable to high levels of integration. The purpose of these guidelines are intended to provide designers, manufacturers, installers, and certification authorities a common international basis for demonstrating compliance with airworthiness requirements applicable to highly integrated or complex systems. ED-79 is the EUROCAE equivalent for this document [13].

i. SAE ARP 4761, “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment” (1996). This document presents guidelines for conducting an industry-accepted safety assessment consisting of a functional hazard assessment (FHA), preliminary system safety assessment, and system safety assessment (SSA). This document also presents information on the safety analysis methods needed to conduct the safety assessment. These methods include the fault tree analysis, dependence diagram, Markov analysis, failure modes and effect analysis, failure modes and effects summary, and common cause analysis. EUROCAE does not currently have an equivalent for this document [14].

j. Aeronautical Radio, Inc. (ARINC) 651_97-ARINC, “Design Guidance for Integrated Modular Avionics, ARINC Specification 651,” Airlines Electronic Engineering Committee (September 1991). This is a top-level design guide for the design and implementation of modular avionics systems. ARINC Specification 653 is a result of specific requirements identified within ARINC Report 651 [15].

k. ARINC 653, including Part 2, “Avionics Application Software Standard Interface, ARINC Specification 653” (January 2007). This document expands and clarifies the original content of ARINC 653 by focusing on avionics operating system partition management, intrapartition communication, and health monitoring. It is intended for use with federated avionics and IMA systems [16].

1.5 DOCUMENT STRUCTURE.

This report is comprised of 12 sections. A brief summary of the contents is provided here for reference.

Section 1 provides introductory material, including the purpose, scope, related documents, background, document structure, and general use of the report.

Section 2 discusses systems safety considerations for IMA systems and introduces the reader to the various roles in the development of IMA systems.

Section 3 discusses the architectures of IMA systems with a focus on the APplication/Executive (APEX) and its inferred architecture from ARINC 653.

Section 4 discusses the staged integration concept that shows the various stages of integration when building IMA systems. This report limits the integration stages to only the first three stages: the integration of components/modules to form a platform, the integration of a single application with a platform, and the integration of multiple applications with a platform.

Section 5 discusses the elements of IMA systems that include the functional, behavioral, and performance characteristics of IMA systems.

Section 6 focuses on RTOS considerations in IMA systems. RTOSs governing partitioning and RTOSs within partitions are discussed. Shared resources governed by the RTOS, along with timing and scheduling considerations, are also discussed.

Sections 7 and 8 give attention to the issues involved with IMA system integration, installation, initialization, and overall system health monitoring and recovery.

Section 9 provides a summary of the research to date, an introduction to the Handbook, and an overview of recommended future research.

Section 10 lists the references used.

Section 11 lists related documentation.

Section 12 is a glossary that provides key definitions and acronyms used in this report.

1.6 CURRENT IMA-BASED RTOSs.

IMA-based systems were installed on aircraft with associated RTOSs. Interviews were conducted with a representative sample of some IMA systems developers, and when appropriate, nonproprietary information is shared in this report as part of the discussion of IMA system topics in an effort to better understand current IMA system integration-related issues. There are other IMA systems in production and under development. Tables 1 through 3 show a sample for this research effort.

Table 1. Airplane Information Management System IMA

IMA System Name	AIMS
Function	The AIMS provides pertinent information concerning the overall condition of the airplane, its maintenance requirements, and its key operating functions including flight, thrust, and communications management.
IMA Developer	Honeywell
Aircraft(s)	Boeing 777
RTOS	Internally developed – DEOS
RTOS Supplier	Honeywell
Brief RTOS Description	The Honeywell DEOS operating system was developed pre-ARINC 653, but retains many of the features of ARINC 653, as it was the precursor to the development of the A653 APEX interface definition.
Years System in service	Over 10
Notes	Information regarding some plans for 7E7 IMA systems was also provided.

AIMS = Airplane Information Management System

Table 2. Secondary Power Distribution Assembly IMA System

IMA System Name	SPDA
Function	Power distribution to regional or business jet aircraft.
IMA Developer	Hamilton Sundstrand / United Technologies
Aircraft(s)	Embraer ERJ-170
RTOS	Internally developed, but based on an ATI-Nucleus COTS product.
RTOS Supplier	Hamilton Sundstrand / United Technologies
Brief RTOS Description	The operating system for the SPDA differs from ARINC 653 in that certain features that might compromise safety were not incorporated. The RTOS used the memory management unit for protecting the data and code partitions and scheduled tasks based on a Deadline Monotonic Analysis method.
Years System in service	3
Notes	Information regarding some plans for a B-737 Wedgetail IMA system was also provided.

SPDA = Secondary Power Distribution Assembly

Table 3. Software Core Operating Environment IMA System

IMA System Name	Software Core Operating Environment
Function	Many
IMA Developer	Smiths Aerospace
Aircraft(s)	B-767/Tanker, C-130-AMP, B-787, others…
RTOS	PSC-ARINC-653
RTOS Supplier	Wind River, Inc.
Brief RTOS Description	The operating system provides an ARINC 653 API as well as a proprietary VxWorks API. APEX Parts 2 and 3 extensions have been included in the implementation. A comprehensive set of I/O capabilities is provided by the system, which are conformant with the requirements for robust partitioning.
Years System in service	At the time of writing this report, the B-767/Tanker has flown with the PSC-ARINC-653-based IMA system for approximately 1 year.

API = Application programming interface
PSC = Platform for safety critical

2. SYSTEM SAFETY CONSIDERATIONS FOR IMA SYSTEMS.

Integrating multiple applications on an IMA system provides both benefits and challenges. The benefits are that information is easier to share in a system specifically designed to exchange data in a controlled way. The challenge is that common mode failures may be introduced, and dependencies may exist in an integrated system that would not be present in a federated system.

For example, consider the communication capabilities using two systems that could substitute for each other in an emergency, such as Air Traffic Control and a Flight Management System. If one of these systems was to fail, the other could be used to compensate and ensure that the pilot maintains situational awareness. If both of these systems resided on a single IMA system that failed, then both systems could fail, causing a higher criticality situation than the criticality of each system in isolation.

2.1 SYSTEM SAFETY ASSESSMENT METHODS.

ARP 4761 focuses on system safety assessment methods for complex systems. The relationship for many of the techniques offered in ARP 4761 for RTOSs based in IMA systems can be challenging to apply. Careful consideration should be given to these methods, as the RTOS enables multiple complex applications to operate together and to share resources that have a large influence on the overall system safety.

The FHA identifies each system failure condition and provides a rationale for the condition's classification. "After aircraft functions have been allocated to systems by the design process, each system which integrates multiple aircraft functions should be re-examined using the FHA process. The FHA is updated to consider the failure of single or combinations of aircraft functions allocated to a system" [14].

The fact that an RTOS can govern the overall operation of an IMA system demonstrates the need for several special systems-related considerations. An integrated modular system should be carefully scrutinized with respect to common cause failures or the sensitivity of the system behavior when groups of applications contend for resources. In addition, just as the FHA has allocated safety functions to the IMA system, the IMA system architecture and the architecture of the IMA system's shared resources should now be re-examined for additional potential system failure conditions.

Once all the system failure conditions are accommodated in the system architecture and associated IMA system design consideration, the IMA system and its associated RTOS should be subject to a vulnerability analysis and robustness test [4]. In particular, since many IMA systems contain system health monitors and recovery functions, the IMA system should be exercised in such a way as to understand the system's behavior in multiple failure scenarios that satisfy the system safety assessment.

2.2 THE IMA SYSTEM DEVELOPMENT PROCESS.

ARP 4754 discusses the categorization of requirements, including safety, functional, customer, operational, performance, physical, installation, maintainability, interface, and derived. All of these requirements are considered during system development through various system safety assessment techniques. The requirements are allocated to certain portions of the system, including the allocation of some requirements to software. This activity results in architectural approaches that may meet overall requirements such as redundancy, monitoring, or partitioning. Partitioned systems, such as the IMA systems under study, should consider the approaches presented in ARP 4754.

ARP 4754 defines partitioning as a design technique for providing isolation to contain and/or isolate faults, and to potentially reduce the effort necessary for the system verification processes. IMA functions contribute to aircraft functions of different criticality; a partition can be used to protect the cross-functional effect of system design errors in separate parts of the IMA system. The partition should be developed at the assurance level corresponding to the highest applicable failure condition classification.

ARP 4754 covers integration of functional system components and partitioning. Conmy, et al. [17], offers an approach to IMA system considerations with respect to ARP 4754. Figure 1 shows the safety assessment process and the associated system development process noted in ARP 4754.

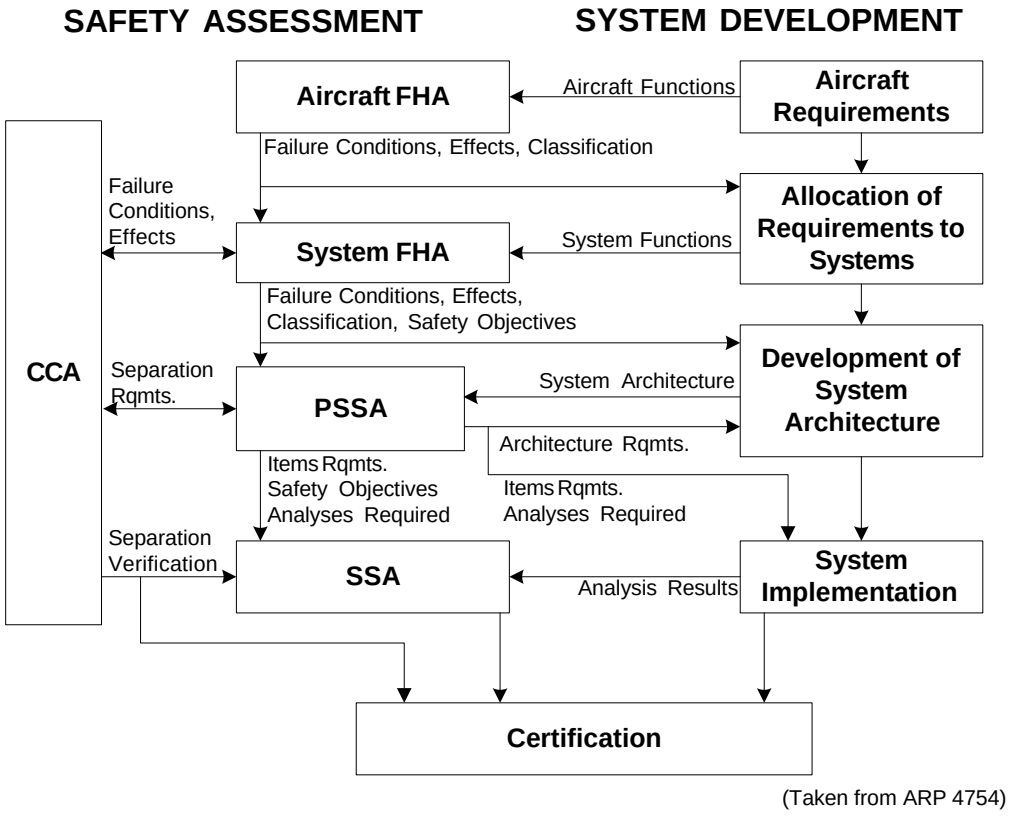


Figure 1. ARP 4754 Safety Assessment and System Development Processes

Conmy, et al. [17] applies the ARP 4754 paradigm to the system development process of an IMA system to demonstrate the IMA system analyses necessary to accommodate ARP 4754 (see figure 2).

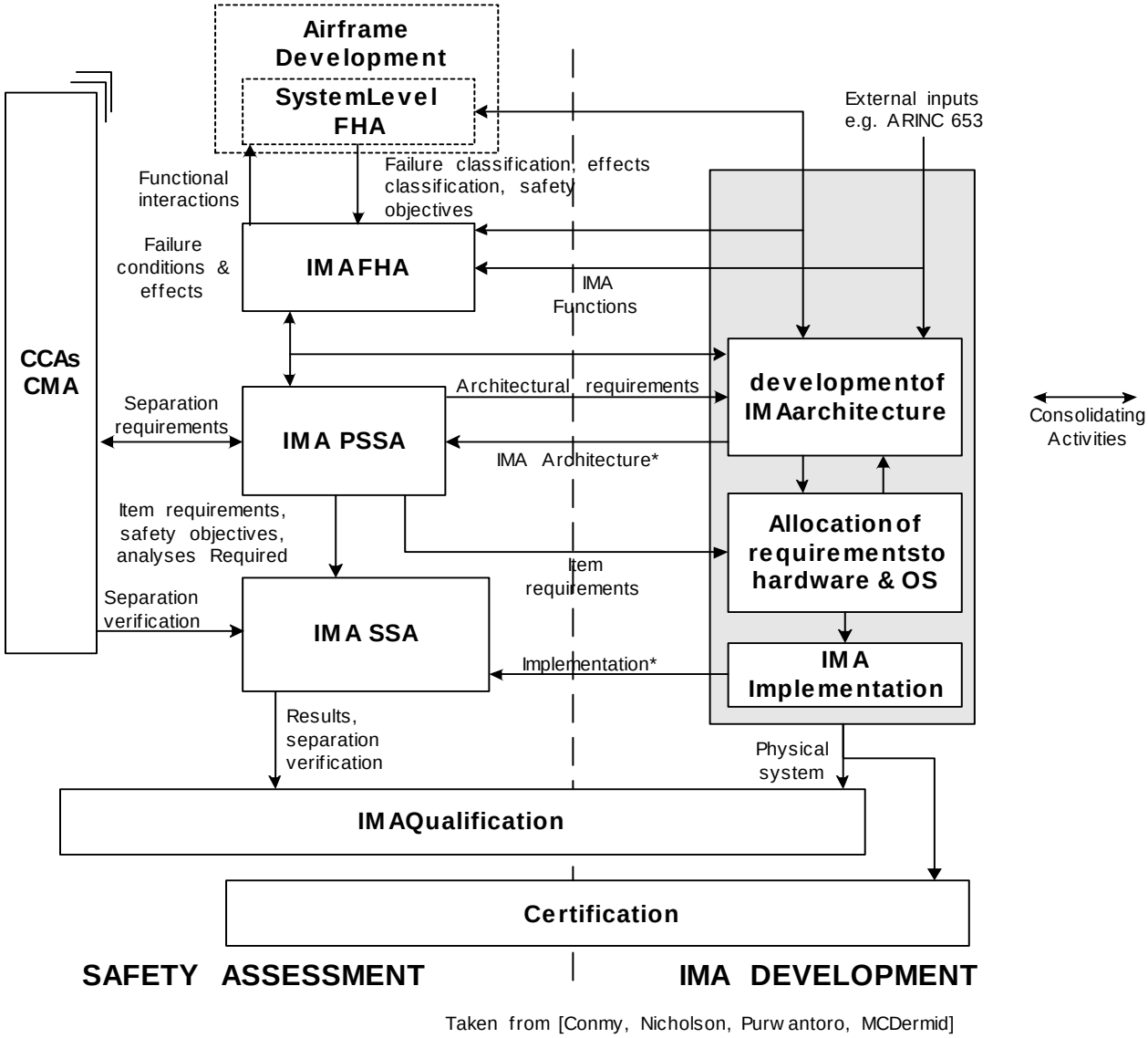


Figure 2. The IMA System Certification Considerations

The difficulty with this approach is that it segregates the IMA system FHA and SSA analysis from the rest of the aircraft system FHA and SSA, where clearly dependencies exist between the two. What is clear is that, as Conmy, et al. states, interaction activities exist between the IMA system integrator and the application supplier [17]. Indeed, these activities also exist between the other roles of the overall system: the platform supplier, RTOS supplier, application supplier, and the IMA system integrator. These activities form commitments between these various roles of the system development. A commitment is an assumption, limitation, performance restriction, behavioral restriction, configuration, or function provided by a module or component, which must be observed by the user of the module or component for proper operation of that module or component.

These interactions show information flowing from the IMA system developer to the application supplier in the form of constraints, and vice versa, where fulfillment of those constraints to the IMA system developer is depicted. The Conmy, et al. approach [17] is sound; however, more than constraints and fulfillments are needed, and more than just the IMA system developer and the application supplier should be involved.

2.3 THE IMA SYSTEM DEVELOPMENT ROLES.

The fundamental idea behind an IMA system is that it is comprised of a set of modules and components that are to be merged into a system that provides certain aviation functions. This provides a mechanism for developing parts of the system separately and then integrating them together into a functioning system. Frequently, modules and components are developed by different associates or organizations with defined roles. DO-297 has defined the following stakeholders.

- Platform and module suppliers
- Application supplier
- IMA system integrator
- Certification applicant
- Maintenance organization
- Certification authority

The roles defined in this report support the defined stakeholders above, with the role of RTOS supplier added.

2.3.1 Platform and Module Supplier.

The platform supplier provides the processing hardware resources with the core software, such as the operating system (OS) kernel and platform module service functions. The core software also includes interfacing software that links the hardware components with the core software. The platform supplier should publish the system development assurance level, electronic hardware design assurance levels, and software levels to which the IMA system was developed. It is the IMA system integrator's responsibility to ensure that all shared hardware and software elements and resources for the platform meet the system's highest levels of safety, integrity, and availability.

2.3.2 The RTOS Supplier.

For proper development and integration of the RTOS to the platform, the RTOS supplier should work closely with the platform supplier to specify the hardware feature commitments of the IMA system to the RTOS. The RTOS supplier provides services for temporal and physical partitioning, along with communications and segregating I/O and other shared resource management. The RTOS supplier is addressed separately from the platform supplier for this study, since several COTS RTOSs exist that may operate on a variety of hardware platforms.

2.3.3 Application Supplier.

The application supplier develops the hosted application (e.g., a particular aircraft function). Application development should be within the commitments conveyed by the other role players; yet application development is typically accomplished without consideration of other application functions, unless those functions are related, e.g., communication with other functions.

2.3.4 The IMA System Integrator.

The IMA system integrator performs the activities necessary to incorporate one or more avionics functions into a system. The IMA system comprises one or more platforms or modules (hardware and core software) and a specified set of hosted applications. The IMA system integrator has the obligation to convey system safety assessment information flow to the SSA process. The IMA system integrator should address all interfaces entering and exiting the IMA system configuration, including the mix of selected applications to be hosted, resource allocation, configuration tables, system integration, and overall performance of the system.

2.3.5 Additional Information.

A companion Handbook on IMA systems has been written in conjunction with this report [1]. This Handbook provides additional and more detailed information regarding the specific roles and responsibilities of each of these role players. The Handbook also describes activities and practices to be considered during the IMA system development.

3. ARCHITECTURES OF IMA SYSTEMS.

Various architectures can be used in the implementation of IMA systems. Most are proprietary, and their specifications are not published. This makes it difficult to discuss details of the architectures that were studied. ARINC 653 is a specification for an API, which supports an IMA system architecture. The ARINC document is published, and it describes many features that could be used in the implementation of an IMA system. In this report, ARINC 653 is described in outline form only and introduces IMA system concepts so that subsequent sections may discuss some of the issues with IMA system implementations. It should be noted that this report does not endorse the use of the ARINC 653 specification or any specific implementation of it.

3.1 ARINC 653 OVERVIEW.

ARINC 653 is a standard developed by aviation system developers that specifies a baseline-operating environment for application software used within the IMA system domain. ARINC 653 provides a general-purpose application executive (APEX) interface between the RTOS of an avionics computer resource and the application software. Included within this specification are the interface requirements between the application software and the RTOS and the list of services that allow the application software to request scheduling, communication, and status information of its internal processing elements. The majority of ARINC 653 describes the run-time environment for embedded avionics software and, as such, specifies methods or approaches

for the allocation and management of shared resources [16]. Since the publication of ARINC 653 in 1997, the ARINC 653 Committee has published Part 2 and Part 3 extensions.

Figure 3 provides a high-level overview of the ARINC 653 architecture.

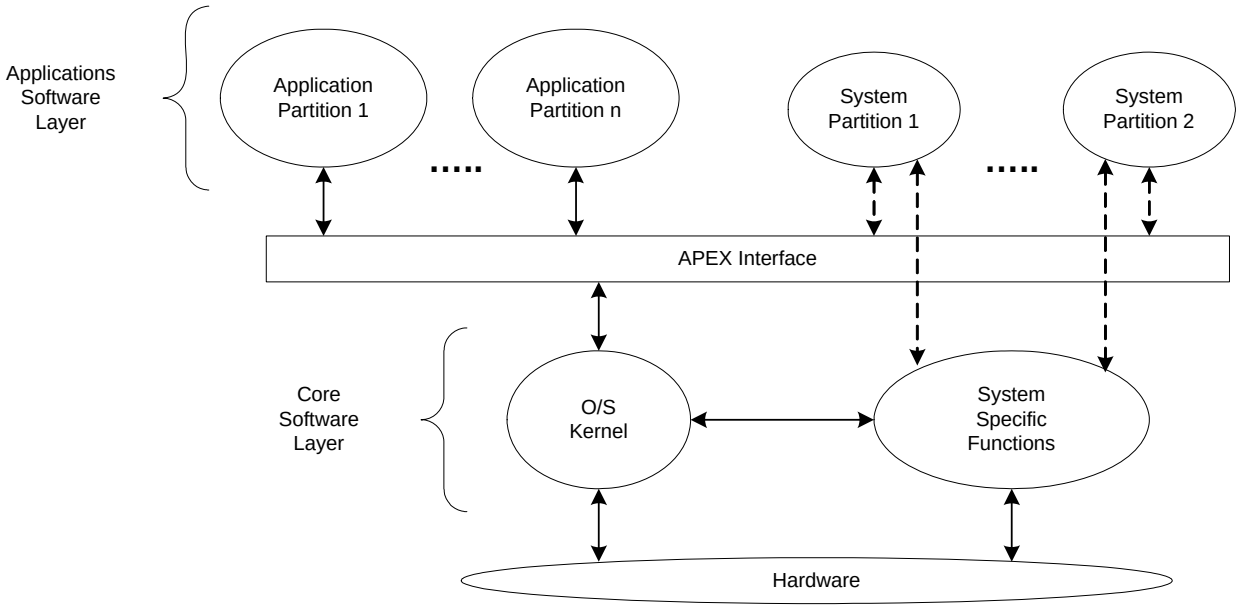


Figure 3. ARINC 653 Basic Architecture—Part 1

3.1.1 Hardware Components.

Any computing system has a set of hardware components on which processing can operate. There are several basic assumptions made about the hardware, as specified in ARINC 653. These assumptions are the commitments made by the platform that are provided to meet the requirements of the IMA system:

- The processor should provide sufficient processing throughput to meet each application's time requirements.
- The processor should have access to or provide required I/O device and memory resources.
- The processor should have access to time resources to enable sharing of CPU cycles based on time.
- The processor should provide a mechanism for the RTOS to take control if an application attempts to perform an operation that is not valid for the specific application.

Valid operations are internal to an application, or they may cross the boundary between an application and the modules or components with which they interact. If they cross the boundary, the interactions should be identified, agreed, and verified. They form a set of commitments that

the application supplier must convey to the IMA system integrator, platform supplier, and RTOS supplier.

3.1.2 Scheduling.

ARINC 653 allows applications to be mapped into one or more partitions. The specification describes how system functionality may be accomplished through partition management (attributes definition, partition control, and partition scheduling). Using configuration tables, the characteristics of partitions, their interactions, and their time and resource requirements may be specified. After the configuration description is loaded and processed, the RTOS provides the behavioral ARINC 653 support by allocating memory for the system partitions, setting up memory protection structures, and allocating other system resources.

The RTOS schedules the partitions by running them in a repetitive sequence as specified in a scheduling table supplied by the IMA system integrator. Supplement 2 of ARINC 653 permits several schedules to exist. Only one schedule is active at any time, and schedule switching may occur providing there is a mode-switch capability. Schedule switching in this sense is essentially transitioning to different modes of operation that have been predesigned and verified. Within a partition, a process (task) is managed with defined ARINC 653 process state transitions (dormant, ready, running, and waiting), and process scheduling is managed with priority pre-emptive type scheduling. The RTOS has total responsibility for process transitions and scheduling. The RTOS assists applications with time management by providing control if a scheduling operation does not respond in the specified time and by allowing periodic processes (tasks) to have assigned deadline commitments. The RTOS recognizes violations of deadline commitments within the partition. The applications may contain an error handler process to take control, or it may need to be handled by the RTOS. Appropriate responses may be programmed within the error handler to cope with the error or fault.

3.1.3 Partition Communication.

Communication between partitions (interpartition) is message-based to provide isolation from fixed partition dependencies. Each partition may create logical ports. The IMA system integrator, in the configuration record that connects the ports, specifies communications channels. Messages are written to and read from ports by processes (tasks) within partitions. Messages between partitions can be handled using various architectures. Special hardware, such as a communications data bus, may provide interpartition communication, or the core RTOS may provide protected message transfer mechanisms by copying data between memory locations. Such architectures can impose very different behavior on the system, and the RTOS must accommodate them by providing semantics that comply with the ARINC 653 specification. Messages can be of fixed or variable length and can be sent periodically or aperiodically. By permitting single or multiple connections on ports, messages may be unicast between two partitions or broadcast to several partitions as an extension of ARINC 653. The mechanism for data transfer is supported by two different modes: sampling—where the transmitted message data is overwritten and queued—where the transmitted message data is unique in that instance of time and must be queued so that information is not lost. The ports have attributes (partition identifier, port name, mode/direction of transfer, and message, storage, and mapping requirements) that should be established by the RTOS for proper communications.

Messages can also be passed within partitions by intrapartition communications. In this case, buffer messaging that permits queuing of the information can be used. Blackboard messaging can be used, where any message written to a blackboard (e.g., scratchpad, common shared memory area) will remain there until the message is either cleared or overwritten by a new instance of the message. Processes (tasks) can also use semaphores or events to synchronize their messaging activities.

3.1.4 Health Monitoring.

Health monitoring is specified in ARINC 653. The health monitor is the RTOS function responsible for identifying, responding to, and reporting hardware, application, and RTOS faults and failures. Health monitoring helps to isolate faults and prevents failures from propagating. By classifying and categorizing faults and the health management responses, a range of possibilities exists that helps the application supplier and IMA system integrator select appropriate behavior. Configuration tables are used to describe intended recovery of identified faults, such as ignoring the fault, reinitializing a process, restarting (warm or cold) a partition, performing a total system reset, or calling a system-specified routine to take system-specified actions. Techniques for health monitoring and associated recovery actions are discussed in section 8.

3.1.5 Suppliers and IMA System Integrator Commitments.

When developing applications using a standard ARINC 653 IMA interface, certain commitments between the platform supplier, RTOS supplier, application supplier, and IMA system integrator must be established. These commitments come in the form of assumptions and constraints, many of which are safety commitments. The commitment method that ARINC 653 uses is the configuration table. Tables specified in ARINC 653 include configuration tables for system initialization, interpartition communication, health monitoring, and allocation of all shared system resources, including time, memory, and I/O resources.

3.2 ALTERNATE ARCHITECTURES.

Other architectural concepts exist that differ from the ARINC 653 paradigm. This section briefly discusses some of the attributes of these architectures. The difficulty in discussing IMA system architectures is the general nature of these systems. An IMA system may be implemented as a single line replaceable unit (LRU) or a complex configuration representing a network of systems. The scope of the architectures discussed here will be limited to the early stages of integration. This discussion will also be based on architectural concepts with respect to information flow. Information can flow in a variety of ways between partitions.

- Physical memory sharing between partitions
- Message communications between partitions
- Data bus communications between partitions

3.2.1 The RTOS-Controlled Memory Sharing.

In the first approach, the RTOS controls the physical memory sharing to assure proper data protection. Dynamically accessing data in this manner has certain risks, but it is still a viable architecture providing that robust memory partitioning is established. A simple form of memory sharing may be used to provide information flow in one direction only. In such a system, one partition may produce or update information that other partitions consume. Appropriate memory protection mechanisms would be set up to prevent any accesses that could compromise robust partitioning. Bidirectional information transfers could be arranged by changing the memory protection structures dynamically. Such a system would need to be carefully designed, implemented, and verified, because dynamic memory configuration would make verification of robust partitioning difficult. Any proposed dynamic memory configurations must be discussed with the certification authority early in the IMA system development.

3.2.2 Message Communications Control.

The second technique uses message communications control for data flow. The ARINC 653 specification describes this approach. Although the implementation is not described, the semantics of the message protocols are described. An implementation may use some operating system features or may offload the information transfer to some memory transfer device. As long as the behavior of the communication mechanism complies with the ARINC 653 specification, then the hosted application may use the API provided, irrespective of the actual implementation.

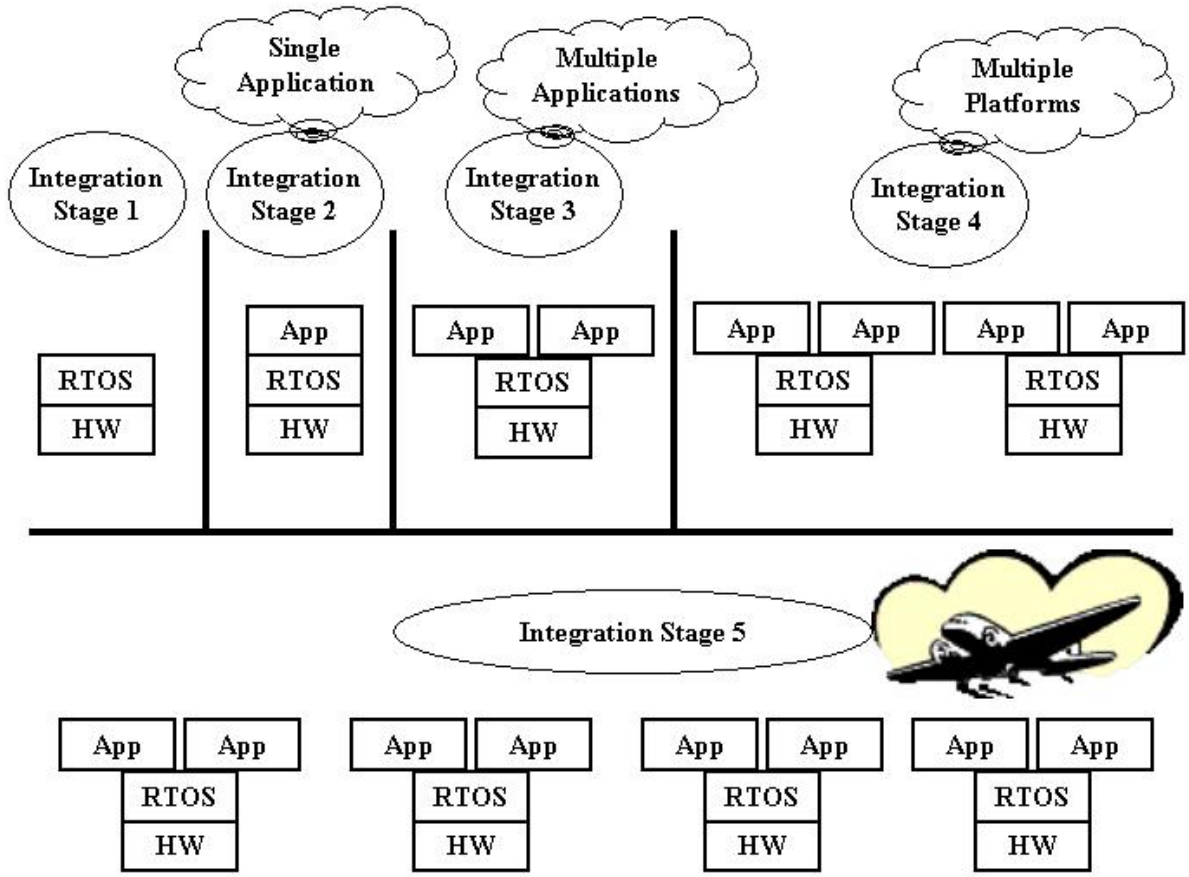
3.2.3 Data Bus Communications and Memory Mapping.

In the third technique, data regions can be mapped to memory locations, which are connected to a data bus directly. Memory protection relies, in part, upon the implementation of the data bus itself. This reduces the complexity of the software partitioning system design, but may limit the type of data that may be transferred. Information that is sampled, where the latest data is required, lends itself well to this type of data bus transfer. However, if messages must arrive in sequence, then additional communication protocols would be required. Time triggered data bus protocols may be used to control data access and transfer. Such protocols can be used to ensure information is shared through robustly partitioned communication channels. This approach can also provide effective time partitioning, whereas the other two architectures require the RTOS to provide robust time partitioning.

4. INTEGRATION STAGES.

The wide range of views as to what an IMA system is precisely presents difficulties in discussing IMA systems and the steps in the integration of the IMA system. An IMA system can range from a simple hardware platform and associated operating software to a complex system of systems whereby each system can contain large numbers of functions. To properly scope this wide range of IMA system views, integration stages of IMA system development that grow increasingly in functionality and complexity are presented. Figure 4 shows an overview of the IMA system development integration stages.

- Integration Stage 1—Integration of components/modules to form a platform
- Integration Stage 2—Integration of a single application with a platform
- Integration Stage 3—Integration of multiple applications with a platform
- Integration Stage 4—Integration of multiple platforms into an IMA system
- Integration Stage 5—Integration of IMA system(s) onto the aircraft (aircraft-level integration)



Acronym	Definition
App	Application
RTOS	Real Time Operating System
HW	Hardware

Figure 4. Staged Integration View of IMA System Development

At each stage and within stages, the commitments and compliance credit should be effectively conveyed between all interested parties as defined by DO-297: platform and module suppliers,

application supplier, IMA system integrator, certification applicant, maintenance organization, and certification authority. Because RTOSs can be purchased off-the-shelf and are central to proper IMA system operation, an RTOS supplier role has been added.

4.1 INTEGRATION STAGE 1—INTEGRATION OF COMPONENTS/MODULES TO FORM PLATFORM.

Integration stage 1 is the integration of an operating system to the hardware components and other modules to form an operational IMA system. This is the simplest level of integration and is necessary for proper operation of any IMA system.

The platform supplier makes the hardware available to support the applications that could possibly operate on the IMA system, usually in the form of a microprocessor and associated memory, timer resources, and I/O devices. This hardware can range from a customized board or set of boards in a system, to a set of composable modules that may already have some level of regulatory acceptance.

Layered between this hardware, the RTOS supplier typically provides a BSP of software that will permit the RTOS supplier's software to run on a variety of hardware configurations.

The RTOS supplier must adhere to hardware specifications and, in most cases, work in concert with the platform supplier. The platform supplier, possibly with the RTOS supplier, may also provide some additional higher-level functionality to support specific I/O devices. Device drivers often require special access to specific memory locations, or registers, or may need to request memory settings that prevent caching of memory locations dedicated to memory mapped I/O.

Device drivers may also be supplied in special system partitions, which may have additional privileges, but are scheduled in line with the application partitions. These system partitions become part of the platform and are treated as part of integration stage 1.

Verification of an integration stage 1 system is planned by an integration stage 1-specific integration test plan that will thoroughly exercise the IMA system, including interfaces, communications, and partitioning robustness testing. A previous study detailed methods for verifying the robustness of an RTOS in an IMA system [5]. If done properly, this activity can result in a certification authority accepted platform module that has documented capabilities and some compliance credits. Types of documentation needed at this step include a set of platform module commitments that denotes systems constraints with respect to safety, functions, architectures, behaviors, and performance. Documentation will also be needed for identifying full or partial compliance credit for certain objectives of the applicable version of DO-178, DO-254, DO-297, and other guidance material.

4.2 INTEGRATION STAGE 2—INTEGRATION OF A SINGLE APPLICATION WITH A PLATFORM.

Integration stage 2 is the integration of an application to the hardware platform. This step integrates a completed application to the platform to demonstrate how this single application will meet its functional requirements.

The integration step in integration stage 2 will focus on the links between the application and the APEX interface, assuming ARINC 653 architecture is used. A linker can accomplish this, although a special linking mechanism may be used to optimize access to shared memory with large address ranges.

It is the application supplier's responsibility to verify the application's accuracy and completeness. It is typically the application supplier who has the domain knowledge and is able to verify through test and other means that the application can satisfy its requirements using the resources and capabilities of the IMA system. In this case, the application supplier can act as the IMA system integrator. To ensure that the test is performed using a realistic setting, the application supplier should be able to specify the allocation and configuration of memory, timing, I/O, and other shared resources to represent the scenario expected by the application when the IMA system is fully populated with additional applications.

To ensure that the scenarios are comparable, the application supplier and the IMA system integrator should agree on the minimal resources to be provided to the application so that it can be verified with those settings. As an example, the timing specifications for the application should be used to verify its behavior under its temporal configuration settings. Other resources such as memory, inputs, outputs, and communications should be verified.

Verification of an integration stage 2 system is planned by an integration stage 2-specific integration test plan that will thoroughly exercise the application's use of the IMA platform, resources, and associated constraints.

If the platform provides a robust partitioning environment, then some verification compliance credit may be obtained in integration stage 2, such that repeating this verification may not be needed on a fully populated IMA system.

Integration stage 2 testing builds on the integration stage 1 testing already completed, and performs coverage analysis of the application. This is white-box testing for the application. Timing analysis may be performed on the application using the agreed upon time constraints with the IMA system integrator.

4.3 INTEGRATION STAGE 3—INTEGRATION OF MULTIPLE APPLICATIONS WITH A PLATFORM.

The integration stage 3 activity will normally commence after integration stage 1 and 2 activities for each application that was completed, and when the configuration development for a set of applications is complete. The IMA system integrator performs this activity. The objective of this integration activity is to verify that multiple applications on an IMA platform operate as

intended and specified, without any unintended effects. This integration activity is an increment to integration stage 1 and 2 activities, and assumes that each separate application has been through integration stage 2 verification on the same IMA platform. For the purposes of this activity, a platform is considered a single software execution environment.

At this stage, the lower-level testing of the application had been completed and does not need to be repeated. If a robust partitioning environment is provided and the same configuration settings are used as was verified with the individual applications, the behavior of the applications should be unaffected by the execution of other applications. However, in actuality, some lower-level testing of the application(s) and the RTOS may need to be conducted to confirm intended function in a fully populated IMA system.

Verification of an integration stage 3 system is planned by an integration stage 3-specific integration test plan that will thoroughly exercise all of the applications use of the IMA platform, resources, and associated constraints. Additional verification activities will include verifying that interactions between applications are appropriate and effectively controlled. The applications should have been verified under the same timing parameters through the integration stage 2 verification process, and the accuracy of the I/O messages should have been verified. The integration stage 3 verification should verify that the temporal specifications of message traffic are met for all applications.

4.4 INTEGRATION STAGES 4 AND 5.

The scope of this report is within the activities listed for integration stages 1, 2, and 3. Therefore, integration stage 4, the integration of multiple platforms into an IMA system, and integration stage 5, the integration of IMA system(s) onto the aircraft, will not be elaborated here.

4.5 SAFETY AND COMPLIANCE CREDIT COMMITMENTS.

The goal of the IMA system certification process is to obtain incremental compliance credit for IMA system components by incremental acceptance of modules and applications. Modules are software, hardware, or a combination of hardware and software used to form an IMA system. To obtain incremental compliance credit, an IMA system should have a design concept (e.g., time, space, and I/O partitioning) and a defined development process that allows independence of concurrent system development, verification, and compliance demonstration.

Each incremental acceptance requires a set of commitments be established between all system modules and components. These commitments must support the overall system safety and effectively communicate information such that compliance can be demonstrated.

Safety and other commitments are met by the roles of the interested parties (aircraft integrator, IMA system integrator, platform supplier, RTOS supplier, and application supplier). Many times these accepted modules have assumed commitments that should be conveyed with the module configuration index (e.g., version description document). Many safety (and other) commitments are applied during the integration process. Incremental acceptance of modules at different integration stage levels permits a vehicle for acceptance of the IMA system.

5. ELEMENTS OF IMA SYSTEMS.

In the study of the IMA system topic, it became evident that there are four key elements of IMA systems:

- Architectural
- Functional
- Behavioral
- Performance

The architectural element was discussed in section 3, so it will not be further discussed in this section. The other three elements (functional, behavioral, and performance) are described below.

5.1 FUNCTIONAL.

The functions incorporated into the IMA system must use the guidance of ARP 4754 and 4761, where the functional hazard analysis, preliminary system safety assessment, and the allocation of these functions to hardware and software are assigned. Once these allocations are in place, the IMA system, including the RTOS, should carefully consider all modes of operation, such as start, push back, taxi, takeoff, climb, cruise, descend, land, crew alerts, maintenance, backup functions, and other operational functions in context of their safety assessments. In addition to the flight or operational modes, the IMA system developer needs to consider the various states of operation within these modes, such as IMA system initialization, normal operation, degraded operation, and shutdown.

When considering these modes or states of operation, certain commitments may have to be declared between the IMA system modules or components that would require proper handling from the various roles developing the IMA system. For example, the degraded operation of the IMA system may require specific RTOS services to attain that operational state, for example, fail safe operation.

5.2 BEHAVIORAL.

The overall behavior of the system under certain scenarios should be designed into the system. For example, an RTOS response to an attempted memory violation should be specified and verified, and have an understood behavior so that all the modules and components of the IMA system will know if the RTOS will shutdown the faulty application, call for an additional backup system to be employed, etc. There may be alternate tasks or communications paths employed, or there may be properties imposed on data by applications. The sequencing of events and synchronization commitments must be documented, and system behavior in various modes and states must be designed and verified.

5.3 PERFORMANCE.

The platform supplier defines many of the performance parameters of the IMA system. Features such as the CPU and its associated clock speed, floating point processing capability, memory management units (MMU), timers, delays, memory types, and their associated access times are

all part of the IMA system's performance definition. The platform supplier must provide these commitments to the other IMA system development role players. Correspondingly, for example, if an application is critical to the system, it may require priority of resources or dedicated resources that in turn may affect the performance of other applications and the RTOS itself. The application supplier must provide these commitments to the other IMA system development role players. The performance commitments are critical to the overall successful operation of the IMA system.

As mentioned in section 6.6.2, careful study should be made of the IMA system's WCET since, as discussed, absolute determinism of this time in a complex RTOS is extremely challenging to achieve.

6. THE RTOS CONSIDERATIONS IN IMA SYSTEMS.

RTOSs are a focal point for any IMA system. They govern the sharing of resources and data. Several vendors offer off-the-shelf RTOSs with some DO-178B pedigree. For example, some RTOSs were reverse engineered to provide the data and verification for (at least partial) compliance with the guidance of DO-178B, and some were approved in previously certified airborne systems. Thus, a new specific IMA system integration role, strongly coupled to the platform supplier, has emerged. This section provides information on the RTOS in the context of IMA systems.

6.1 RUN-TIME KERNELS.

The job of a run-time kernel is to apportion available memory, time, and I/O resources between the applications.

These resources are typically identified and shared through configuration records, tables, or files. Configuration records are used to document the resources available on the platform and the resources allocated to the OS kernel and the hosted applications.

The RTOS design will show, and RTOS verification will verify, that the information presented in the configuration records is used to set up structures in the RTOS and in the associated computer. These structures may control access to memory using hardware (page/segment tables) and timetables, which cause the scheduler to switch and select partitions.

At this level, it is not always possible to allocate safety concerns to the RTOS because of the generic nature of RTOSs. It is the applications that may compromise safety, and it is primarily the applications that use the RTOS. The approach used to help classify these safety concerns is to identify RTOS vulnerabilities [4]. These vulnerabilities will address the safety concerns caused by interference between applications, which may not be robustly partitioned.

Identification of RTOS vulnerabilities should be systematic. A catalog of design features used to provide robust partitioning might be incomplete. The completion criteria would be difficult to verify if the list is informal. Vulnerabilities should be treated in the same way safety concerns are mapped onto fault trees, that is, decomposing levels of detail about possible fault origins. Vulnerability trees should identify the possible faults, for example, robust partitioning being

violated in a systematic, hierarchical way. Each branch of the vulnerability tree will identify how robust partitioning can be compromised. Lower branches will be more detailed, until there is sufficient detail to identify one or more RTOS requirements that ensure the vulnerability is mitigated. Tracing these vulnerabilities to RTOS requirements, and then verifying the RTOS requirements, can complete the vulnerability assessment.

The assessment of the kernel should be performed with the assumption that a partition may contain code that could deliberately attempt to compromise robust partitioning, health monitoring, fault management, or other limitations or constraints of the system. It should not be possible to develop an application that could use more resources than those allocated to it in the configuration records. It also should not be possible for an application to modify any configuration record or any part of the system that implements robust partitioning, or alter any system provided health monitoring, fault management, or other system limitations or constraints.

6.2 NONPARTITIONING RTOSs.

While some of the information in this document applies to nonpartitioned RTOSs, the notion presented by integration stages 2 and 3 assumes robust partitioning. Robust partitioning is a means for ensuring the intended isolation and independence in all circumstances of aircraft functions and hosted applications using shared resources [11].

When application suppliers verify their software during integration stage 3, then any compliance credit obtained (partial or full) may be carried forward. However, the IMA system integrator may need to repeat some verification of the RTOS and applications for the fully populated IMA system. This verification may not need to be repeated by the IMA system integrator; indeed, the integrator may be working with several applications supplied by different application suppliers, each of which has the domain knowledge and experience to verify their own applications.

In a nonpartitioned environment, such assumptions cannot be made. The domain level testing will need to be performed with all applications linked together. As applications become more powerful, their interconnections will carry more data at faster rates. Nonpartitioned systems complicate the integration and verification activity since the interconnections are physical and point-to-point. In partitioned IMA systems, because the interconnections are virtual, it may be easier to intervene with mechanisms to probe or record message traffic.

6.3 THE RTOS WITHIN A PARTITION OF AN IMA SYSTEM.

The IMA system includes a defined set of process management services that provides the means of handling processes within partitions. One configuration of an IMA system is shown in figure 5. In this configuration, a Module Operating System (MOS) runs at the system level and schedules several partitions, which run at user level and where each partition contains an application and a Partition Operating System (POS).

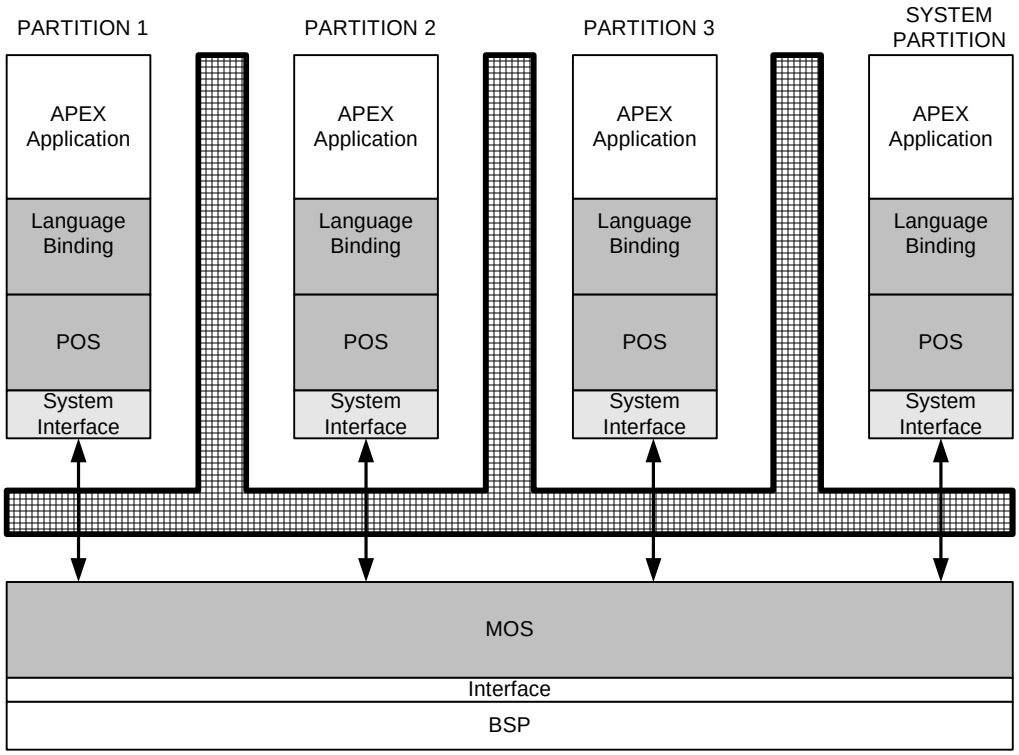


Figure 5. Configuration of an IMA System

As described in the COTS RTOS study:

> "From the user perspective, the programming model may be the APEX specification (i.e., the API may comply with ARINC 653). The implementation may alternately be offered as a layer, which is based on the proprietary implementation offered by a specific vendor. One vendor's accommodation of ARINC 653 may be implemented with only one scheduler in the kernel providing scheduling operations for both the processes within applications, and scheduling operations of the applications. Another vendor may use a two-level scheduler: one that manages applications and the tasks that manage their resources and a process scheduler that manages user mode processes within an application. Other approaches are also possible." [5]

Using the paradigm of an application being equivalent to an LRU, the selection of, and mapping to, a POS may need to be flexible. If the POS is provided in the user space, then the application must be linked with the POS. A simple linking will mean a copy of the POS in each application's memory space. A more optimal mapping may require just one copy of the POS in physical memory, and links to this from each virtual memory space of each partition. With extremely large memory address ranges, calls to the POS may need to be made indirectly to circumvent various hardware address limitations. This is accomplished using indirection tables that provide the link between a call and the desired address of the function being called. Calling tables are often used to achieve a level of abstraction that permits two or more versions of the POS to be used/loaded at the same time. By changing the indirection tables, a version of the

POS may be reinstalled in the field for one application without affecting other applications that do not use it.

6.4 SHARED RESOURCES AND RESOURCE MANAGEMENT.

The RTOS is typically responsible for management of shared resources, including (1) management of memory and its various types and modes; (2) I/O management; (3) management of CPU, including process timing; and (4) communications. I/O system management, along with file system allocations and sharing, are also governed by the RTOS. It should be noted that it may sometimes be easier to dedicate a resource to a partition and have the partition share access to that particular resource. For example, one design choice would make the filing system a shared resource and partitions would contend for use of this resource. Another design option would dedicate the filing system to one partition and have the applications send messages for storage to this one partition that implements the filing system.

The RTOS of any IMA platform has the task to manage all shared resources among the various applications using those resources. There are four major areas of shared resources.

- Memory
- I/O
- CPU and other processing units (e.g., floating point)
- Communications between partitions

6.4.1 Memory.

The RTOS controls a variety of memory types with the help of the CPU control registers and the MMU. Memory types include cache (instruction cache and data cache), program random access memory (RAM), data RAM, program read only memory (ROM), data ROM, and other nonvolatile memory types. Each type of memory may impose certain constraints due to its electrical characteristics, which the RTOS must take into account. Furthermore, the RTOS must consider the differing characteristic features of these memory types. For instance, cache can be set up in various modes of operation, such as copy-back or write-through [5]. The cited reference also discusses the mechanism for using cache in a more deterministic manner, such as switch cache off, specify a huge jitter margin, selective flushing, and selective invalidate. Reading and writing to nonvolatile memory is typically slower, and certain types of flash memories have special requirements on the protocol used for writing information to them. More recently, special attention to single-event upsets that perturb memory cell states due to radiation affects at high altitudes may also require special memory maintenance updates such as memory refresh, use of error correcting memory, or other design mitigations. Regardless of the mechanism used, each has it own set of assumptions and constraints, or commitments, that the RTOS supplier and platform provider must consider in its design and in how the RTOS and each application will use the shared memory with other applications. Further, DO-255 states that for higher-criticality-based IMA systems, the core software shall prevent any adverse behavior of software contained in a partition from affecting the operation of any other software outside of the partition.

6.4.2 Input/Output.

The RTOS in an IMA system must effectively control access to shared I/O ports, devices, and channels. As with other shared resources, I/O operations should be partitioned or protected in some fashion. Different approaches are possible, such as kernel-centralized I/O control or partition I/O control. Many implementations incorporate some type of RTOS-governed partition or task permission table that permits only certain partitions or tasks to access specific I/O. A health monitor is then employed to identify any undesignated accesses by other tasks or partitions. I/O considerations differ for different RTOSs. Memory and timing can be effectively controlled at the integration stage 1 level by the RTOS and used by the application; however, I/O transcends integration stage 1 and can easily affect integration stage 2, 3, 4, or even 5 levels of integration. Because of this, constraints and assumptions made by the RTOS on behalf of the I/O must be effectively conveyed to the IMA system integrator, and perhaps require not only special integration considerations, but also special verification activities.

6.4.2.1 Input/Output—BSP Management.

Connecting I/O to the RTOS is typically done through an API. Information is passed between the application and the I/O device using this API. On an IMA system, the I/O is more controlled and the applications may have dedicated devices that are not shared, or the I/O messages are communicated through virtual ports, which the IMA system integrator connects to communication channels. In this way, the application is unaffected by other applications performing similar I/O operations.

The platform supplier integrates device drivers through API calls to the BSP; this permits the generic API interface to access the specific hardware being used. Certain constraints and assumptions may also exist for the BSP, and need to be communicated between the platform supplier, RTOS supplier, and application supplier. Either these constraints will be specified in interface specification documents, or if they are changeable, they will be communicated in configuration records created specifically for such information.

6.4.2.2 Input/Output—Interrupt Management.

Recognition of interrupts and the distribution of related events are usually accomplished by the RTOS. The interrupt mechanism must be conveyed from the platform supplier to the RTOS supplier and the application supplier. As interrupts are asynchronous, they may affect an application that does not use them.

Certain interrupts are unavoidable. If an external clock is used, then interrupts may be a way of keeping track of real time in the system. Two options are possible. If a ticking clock is used, then the time overhead is periodic and can be accounted for, since it is deterministic. If an alarm clock is used, then the timer interrupt will occur when the programmed time expires. Again, this is deterministic since it depends on the programmed value.

Interrupts due to faults or exceptions will be treated specially. A power loss could result in an interrupt, which starts a special timer. If the loss persists for longer than a specified duration,

then some system prescribed action may be programmed. Other interrupts may not be permitted at all, or they may be permitted but they will require special handling.

Commitment information, such as minimum and maximum allowed interrupt rates, interrupt types (e.g., edge-or level-triggered), interrupt latency, exceptions and exception handling, masking, and relative priorities, must be documented and verified.

6.4.3 Central Processing and Other Processing Units.

The IMA system RTOS controls and schedules the common processing allocations between partitions on a processor. The control of this processing element and other processing elements on the device being used is generally exclusively controlled by the RTOS. As such, the hardware- and platform-specific constraints are well known by the RTOS supplier, and must also be conveyed to the application supplier and IMA system integrator.

DO-255 also specifies that the allocation of processing time must lead to predictable real-time behavior with demonstrable properties. It also specifies that the ACR manufacturer must document the method of time allocation to include the maximum number of partitions supported by the allocation scheme, the partition scheduling (e.g., fixed at manufacture, set at integration, set at load, set at power-up), and the scheme for external I/O (e.g., continuous, at the beginning or end of period, between context switch). There may be specific features of the hardware where the time when a processor run does not equate to the number of instructions executed in all cases, as discussed in section 6.6.2 on WCET.

6.4.4 Communications.

The RTOS provides communication services to support information flow between partitions hosted on a processor and within a partition. This can come from a data bus or RTOS-supported message interface. The communication services support streams of information and event information that are typically platform-independent, using a variety of communication mechanisms. For example, ARINC 653 discusses the approach where the RTOS sends messages directly to a communications bus, or alternately uses memory locations within the core OS process shared memory. Messages can be fixed or variable and can be sent either periodically or aperiodically. Broadcast messages may also be permitted. Some implementations require a port interface that requires attributes such as a partition identifier, port name, mode and direction of transfer, and message, storage, and mapping requirements. Again, these communication services require commitments from the platform supplier, who has designed potential data bus communications for the RTOS supplier and application supplier.

6.5 THE RTOS EXCEPTION HANDLING.

An exception may be raised while an application is executing. This may be a system-defined exception, e.g., a system level exception handler for corrupted memory (fault management). Depending on the programming language being used, it may be a user-raised exception. A user-raised exception will be identified by a programming language-specific, run-time support system that will search for the appropriate exception handler and give that handler control to affect user-specified recovery, or it may need to be handled at a higher level, e.g., executive level, rather

than application level. For example, a data consistency check may raise the exception Data_Inconsistent, which is propagated to an exception handler that repairs the data structure with a default value.

A system-defined exception may still be the result of an application error, but it may be trapped by the hardware. For example, a divide-by-zero operation may be the result of an expression in the partition or application, but the error is raised by the hardware and the exception handler is located in the core RTOS. The hardware exception handler must identify which partition was running and propagate this hardware exception to the run-time support system to take language specific actions.

Architectures that comply with ARINC 653 require that a system health monitor be in place. If a system-defined exception cannot be handled by the application, either because the application programming language and its associated run-time support system does not support exception handling, or because a handler routine for the raised exception does not exist, then the exception is propagated to the health monitor, which must determine the appropriate behavior responses and determine how best to recover from the event.

Even though exception raising and propagation may use system resources (i.e., the core RTOS), it is important that the memory and time used must be taken from the allocation to the partition that was running when the exception was raised. The core RTOS and the partition RTOS must be carefully designed so that if a system-defined exception is raised just as a partition switch is about to occur, then the processing of the exception is deferred until the exception-owning partition is resumed. If this cannot be done, then the timing margins on partition switching must take into account the jitter that could be introduced by exceptions occurring just prior to a partition switch.

6.6 THE IMA SYSTEM SCHEDULERS.

Robust partitioning is necessary for scheduling in IMA systems. ARINC 653 details two levels of schedulers: (1) a scheduler for the partitions, which by ARINC 653 definition are cyclic, and (2) a scheduler for task scheduling within the partition. Processor CPU time, memory, I/O, and bus capacity must be properly distributed to all partitions so that timing and other constraints are guaranteed when tasks are sharing resources.

A robustly partitioned RTOS provides temporal (time) and spatial (memory) partitioning for applications or partitions hosted on a processor. There are a variety of scheduling approaches possible at the partition level. The Distance Constraint Guaranteed Dynamic Cyclic [18] scheduler applies three basic operations: left sliding, right putting, and compacting, to dynamically schedule aperiodic tasks and, in the meantime, will guarantee the distance constraint characteristics of a cyclic partition schedule.

Because ARINC 653 specification provides cyclic partition scheduling and a number of periodic process scheduling operations, the combination lends itself well to rate monotonic analysis. The ARINC 653 specification also permits process priorities to be changed. This means that an earliest-deadline-first scheduling paradigm may be used within a partition.

The architecture drawn in figure 3 shows a logical relationship between applications and the core OS. As described in reference 5, the separation may be made in different locations depending on the implementation. There are two schedulers specified in ARINC 653. In the MOS, the scheduler arranges partitions to run cyclically, and in the POS, the processes run using a priority pre-emptive scheduler. The core OS will run in a privileged mode on the IMA system hardware. Calls to the core OS from the application level must be made using a special system call mechanism. The RTOS implementer may choose to have both POS and MOS in the core OS, simplifying the implementation. All calls to the APEX layer would then be implemented by a set of system calls. The interface at the application layer is very thin, with the APEX interface being little more than a calling mechanism.

The other implementation approach is to supply the POS and MOS as separate implementation layers and perform process scheduling at the application level and partition scheduling at the core OS level. In this implementation, more code is executed at the application level, and calls to the POS are simple calls not requiring the special system call mechanism. The resultant context switch time is reduced, making process scheduling more efficient; however, there is more code at the POS layer, and special linking arrangements may need to be made. This interface may be called a thick binding to differentiate it from the one described above.

In both cases, the application interface needs to be linked with the application using the applications address space. Both the thin and the thick bindings require RTOS code to be linked with the application. The integration stage 1 results in a core OS layer that interfaces to the hardware. It has access to all of the system resources and the POS layer, which may be just a simple system calling interface or a complete process scheduler in addition to a system calling interface.

The RTOS supplier provides a system with the integration between the POS and MOS completed. The application will link with the APEX interface, be it a thin or a thick binding, to form a monolithic program running in one address space.

The obligation of the platform supplier is to ensure that the appropriate life cycle data exists for all of the code up to the APEX layer.

Two aspects of IMA system schedulers warrant further discussion: initialization and WCET analysis.

6.6.1 Initialization.

Task spawning takes system-provided resources and sets up complex data structures. One approach to task spawning is to preallocate all of the resources and set up static memory structures. Another approach is to allocate and initialize these structures dynamically, but only performing these initializations during start-up time to avoid indeterminist behavior. Whichever technique is used, the start-up time is difficult to predict, but will be consistent if the parameters of the application do not change. If the start-up time does not change, then it may be measured to see if the start-up time satisfies requirements.

If tasks are created while the application is running, then the time taken to spawn the task compromises the application timing considerations. Task creations are typically forbidden for highly critical applications once initialization is complete. ARINC 653 provides a special state that permits scheduling only after processes are created, and process creation in the running state results in an error.

6.6.2 Worst-Case Execution Time.

WCET is notoriously difficult to determine in complex systems. Measuring and estimating how long an application takes to run is only academically interesting, as most real-time systems are repetitive and the application stops when the system is shutdown. The repetition rate may vary with the design, and there could be several repetitive loops in a specific system. The only visible effect of timing behavior is the application's response to external events, and the application's ability to produce some external events. An event could be an interrupt, a discrete voltage level on a pin, or simply the availability of some data value in memory.

DO-178B suggests WCET is to be estimated so that timing margins may be determined. The implication is that the application runs as a single thread to obtain data, process this data, and produce a result in a specific time window. This type of programming is popular and may not require the use of an RTOS.

Using an operating system, the execution paths in the IMA system computer may be more interleaved, such that several execution threads may cooperate to improve the throughput of the data. Within a specific time window, there may be several events that require appropriate responses, and timing of these responses may be critical to the overall timing assessment. For some cyclic repetitive functions, the important measures are the timing margins for the execution time within given time frames. For software that is scheduled by time frame, the deadlines correspond to specific points on the time frame. The assumptions are that I/O operations are synchronized with the end of the repetitive time frames. For software that is event driven, the important measures are the timing margins on the external events. For software that is scheduled on events, the deadlines correspond to the time of external events.

Application timing becomes important. The modules and components that support the application will have an impact because they share some system resources. The modules and components include the RTOS, BSP software, software support libraries (e.g., mathematical libraries or string-handling libraries), as well as infrastructure software, which provides I/O support, filing systems, etc.

To help the application supplier determine if they can meet their timing margins, the RTOS supplier may need to provide data on the latencies that may be induced in the application code by RTOS functions and services. For example, a change of a task's priority may require a search in a scheduling queue. This search may be linear and have a bound dependent on the number of tasks in the system and their relative priorities, or the search may be optimized and the time may be constant.

The safety concerns for WCET are the same in IMA system applications as they are in a federated system, with the following subtle differences as described in reference 5. In a

partitioned system, CPU time used by an application does not necessarily equate to processing throughput during the schedule window of the partition. As cache memories are shared resources, they carry an internal state between partitions. This does not affect data integrity in a truly partitioned system, but it may affect timing behavior on memory access. Special care must be exercised to take this into account, for example, flush or invalidate cache on a context switch, or set up caching policies, write through, and copy back.

Due to the difficulty in properly assessing WCET in IMA systems as noted above, an analysis of integration stage 1 and 2 integrated modules or components must be conducted with verification of the assumptions during testing.

7. INTEGRATION AND INSTALLATION.

As discussed in section 4, integration is a multistep process that can be viewed in steps or integration stages in this document. It should be noted that TSO-C153 authorization does not provide functional approval or installation approval [19]. However, AC 20-145, "Guidance for Integrated Modular Avionics (IMA) That Implement TSO-C153 Authorized Hardware Elements," provides guidance to build and certify a compliant IMA system from the hardware element level up to the highest level of integration at integration stage 5. Associated with these steps are the verification activities necessary to show the integration process has met its requirements.

The integration aspect of these systems can be likened to the development of the software code itself, where IMA system configuration databases, commitment information, compliance credits, component control, etc., can have a powerful influence on the resultant performance and success of the IMA system. This opens up a new level of integration activities where, at each integration stage, the integration activity has requirements, a design approach, a configuration basis, a robust verification plan, and quality assurance oversight.

The installation of modules and components on an IMA system must also consider reinstallation. To obtain the benefits of an IMA system, it should be possible to modify a single-application module or component, while maintaining compliance credit for the others that were unaffected by the modification. A data loader that works in conjunction with a manifest system must track all modules and components that have been successfully loaded. The configuration must be compared against an approved configuration to ensure that proper loading occurred and that the configuration is what was approved. This includes all code and data for applications, and also includes the configuration data used to initialize and manage the IMA system.

Attention has been given to increased demands on web-enabled embedded computer systems such that they need to perform at all times, including during installation. Architectures are in place to permit such activity. Some approaches include loading the upgraded code and the code to permit a smooth transition to the upgraded software. This transitional code will include system state behaviors during the transition to track the success of the transition [19].

7.1 SOFTWARE/SYSTEM INTEGRATION PROCESSES.

7.1.1 DO-178B Issues.

Currently, the FAA relies on the guidance of DO-178B to assist in the acceptance of the software in airborne systems and equipment. DO-178B recommends product and process guidance considerations. The study performed within this report finds DO-178B to be well suited to systems developed in a federated manner, that is, a single LRU with a single primary function. Many of the recommendations in DO-178B can also be applied to a system that is based on an IMA system. Not surprisingly, this study has identified some shortcomings in applying DO-178B to an IMA system.

DO-178B has identified specific fundamental processes (planning, development) that are operated on by specific integral processes (verification, configuration management, quality assurance, certification liaison). Guidance that is specific to integration development activity is located within the development process section 5.4.1, with only one objective:

> "The Executable Object Code is loaded into the target hardware for hardware/software integration."

Guidance that is specific to the integration verification activity is addressed in DO-178B sections 6.3.5 and 6.4.3: the reviews and analyses of the outputs of the integration process and the requirements-based software integration testing and hardware/software integration testing.

The high degree of communication and integration required from the various role players for a typical IMA system has demonstrated that DO-178B does not provide enough detailed guidance for such systems. The objective-based approach of DO-297 will require additional activities beyond the current DO-178B guidance. The Handbook provides some insight into some of these activities [1].

7.1.2 A Software/System Integration Process.

DO-297 is a guidance document for IMA system developers that provide specific guidance for the stakeholders: the platform and module suppliers, application suppliers, IMA system integrator, certification applicant, maintenance organization, and certification authority.

DO-297 guidance includes functional performance, aircraft safety and security, design process, design tools, partitioning, resource management, fault management, and health monitoring. Verification guidance discusses partitioning analysis, robust partitioning verification, fault isolation and reporting, and specific guidance for modular qualification of IMA system components, or subsystems at a variety of integration effort levels or integration stages.

7.2 SOFTWARE/SYSTEM INSTALLATION AND CONFIGURATION PROCESSES.

Installation and configuration management activities occur at different parts of the development, verification, and deployment life cycles of IMA systems.

7.2.1 Installation.

7.2.1.1 The IMA System Development and Verification Tool Installation.

The complexity of integrating IMA systems will require the use of new tools to develop the system and associated commitments, integrate the system, and verify that system commitments are sustained.

7.2.1.2 The IMA System Integration Assistance Tools Installation.

The integration of modules and components into a functional IMA system requires a concerted coordination effort with the system development roles and, at times, various configuration management systems.

Some IMA systems have designed integration tools for the RTOS that assist in the development and integration of the IMA system. Some of the system resource configuration items under the direction of these tools are flash size, RAM size, tick rate, page size, hard start times, cold start times, CPU utilization available, process quotas, thread quotas, event quotas, mutex quotas, mailbox quotas, semaphore quotas, etc.

These tools may also generate data bus registry configuration tables for operating systems. They will also determine if IMA system resources will meet the guaranteed requirements, and they will take information from each process, calculate, and report the ability to allocate the specified configuration.

7.2.1.3 Installation on an IMA System.

Installing applications and IMA system support software requires the use of a data loader. The software to be loaded must first be transferred from the transport medium into nonvolatile memory and the transfer confirmed. When the system is started, the software and data are transferred from nonvolatile memory to the IMA system's virtual memory space. This requires copying the data to RAM because this memory type provides better performance. The transfer of code and data must be correct and confirmed. This requires that the correct versions and configurations of the software are transferred and the correct version of the list that describes these configurations is used.

The software to be loaded and their versions will be held in a manifest that is given to the loader. The description of where the software is to be placed and where they are subsequently copied must be maintained in configuration tables. The loader must not compromise the manifest, configuration tables, software to be loaded, or the processes used. A common approach is to use cyclic redundancy checks and automated configuration management systems to verify the correct loading of the configuration items onto the IMA system. This includes part number checks, intermix checks, etc.

7.2.2 Configuration.

The configuration of an IMA system and the associated set of applications may require a specification that is several thousand lines long. The configuration data will be provided in some readable form so that it can be developed and reviewed. This readable form may be inappropriate for processing by the IMA operational system, and some translation of this data may be used instead. Tools may be used to make the specification, verification, and translation process more manageable.

Configuration record tools are a set of tools that manage the integration specification of an IMA system. This is distinct from a configuration management system, which is used to control the versions of modules or components in a specific baseline.

Configuration record tools allow application developers and IMA system integrators to manipulate information that is used to control the configuration data in an operational or test system. The tools must enable application developers and IMA system integrators to establish configuration record data and to review this data for compliance credit.

The configuration specifications may be in many forms. They may be in the form of some programming language data declarations, in eXtensible Markup Language (XML), or in some text-based proprietary format. The ARINC 653 Part 2 specification provides an XML schema format for describing the configuration information. The specification suggests that tools be developed to allow the configuration information to be transformed into the binary versions that should be used on the IMA system. The XML representation will be translated to a proprietary representation that may be input to an IMA system, where it is used directly by the core operating system.

If the configuration specification is produced in some user-readable form, such as XML, or a programming language and then translated to some internal representation, then the verification of the internal data structures must be considered. The internal data structures may be in a compact text form, where they will need to be translated into the information required by the RTOS, or the data structures may be in binary and be translated prior to be being loaded.

If the configuration specification is translated, then the integrity of the translation must be verified. One approach is to use a qualified translator. The input data would be reviewed by the IMA system integrator and checked for consistency by some qualified verification tools. If the translator was qualified as a development tool, then the data integrity check would be complete. If a qualified translator was not available, i.e., a programming language was used as a configuration specification language and a qualified compiler was not developed, then the configuration data would need to be verified either by test or by translation into a form where it could be verified.

The ARINC 653 Part 3 specification has developed a conformity suite to be used to check operating systems that provide the APEX interface. The intent of the conformity suite is to ensure that the operating system manages application code in conformance with the APEX interface. The conformance suite will check compliance with the specification at the language level, the behavior of the code using the API, and the robustness checks required.

The conformity suite is to provide an agreed set of requirements and test specifications that could be used to develop a validation suite. The development of a specific validation suite is outside the scope of the ARINC 653 committee; however, the specification for the suite content is being developed as part of Part 3.

8. THE IMA SYSTEM INITIALIZATION, HEALTH MONITORING, AND RECOVERY.

8.1 THE IMA SYSTEM INITIALIZATION.

Startup of a highly integrated IMA system can be difficult. Task priorities, resource priorities, data stream initializations, platform hardware chip initializations, etc., put additional commitments on the various modules and components of the IMA system and the IMA system integrator. Parameter defaults must be established and state machine assumptions must also be considered. System initialization may require a different balance of resources compared to steady-state execution. Also, synchronization between partitions exchanging information may affect their behavior. This may require different execution schedules during system initialization than other execution modes.

8.2 THE IMA SYSTEM HEALTH MONITORING.

Techniques for system health monitoring can vary. DO-255 states that the IMA system shall include internal hardware and software mechanisms necessary to ensure that software contained in a partition is prevented from violating the time and space requirements of other partitions. Any such attempt at partition violation shall be detected, blocked, and reported so that the IMA system is provided with a means of uniquely identifying and handling the error. The handling of all errors reported to the health monitor by one partition shall not affect software contained in other partitions running on the IMA system. Fault identification can occur in the partition, in the core RTOS (executive), at the IMA system level, or the aircraft level.

8.3 SYSTEM RECOVERY.

The IMA system must provide a health monitoring system, but because errors can occur at various levels, the provision of configuration data to control error detection and recovery actions belongs to each of the roles discussed in section 2.3. Each application should describe the recovery actions to be performed if an application is to encounter an error. An application error could be logged and then ignored, it could result in a cold or warm restart operation, or the application could be made dormant. The RTOS or IMA system will handle errors that cannot be handled by the applications. It is the IMA system integrator's responsibility to specify the health monitor responses based on the error classes and system states. The platform supplier may specify a set of health monitor responses for a class of errors and system states that can be attributed to the platform.

9. FINDINGS AND RECOMMENDATIONS.

Integrating multiple applications on an IMA system provides both benefits and challenges. The benefits are that information is easier to share in a system specifically designed to exchange data in a controlled way. Some challenges are that common mode failures may be introduced and

that dependencies may exist in an integrated system that would not be present in a federated system.

This report considers integration activities for modules and components in IMA systems with a focus on the RTOS. Existing guidance with respect to the industry was reviewed, and associated safety considerations were discussed. This report focuses on the lower integration stages of integration at the platform, RTOS, and application integration levels, and recommends a series of safety and compliance credit commitments be identified by the various role players in the development and integration process. Several role players in IMA system development are discussed such as the platform supplier, RTOS supplier, application suppliers, and the IMA system integrator, and their roles of integrating multiple functions at different integration stages.

Specific RTOS considerations are discussed. In particular, the difficulty of achieving determinism in an RTOS-based IMA system is elaborated. Discussion on system installation and initialization along with system health monitoring considerations are provided in this report.

It is recommended that a planned approach be taken to IMA system development, verification, and deployment. The Handbook developed concurrently with this report can be used to assist in the development and verification of IMA systems [1]. The Handbook is intended to provide a view of activities that the various role players should assume in order to effectively develop and integrate IMA systems and their associated modules and components.

In conducting this research, certain technology areas were found lacking. Research in data and control coupling in IMA systems is currently deficient, and an effort to formalize an approach for analyzing, documenting, and verifying data and control coupling is needed.

Adequate and definitive resolution of the WCET problem for IMA systems needs further research as well.

A further recommendation is to develop and provide a job aid specific to the approver of IMA systems, considering some of the activities noted in this report and companion handbook.

10. REFERENCES.

1. Krodel, J. and Romanski, G., “Handbook for Real-Time Operating Systems Integration Considerations in Integrated Modular Avionics Systems,” FAA report DOT/FAA/AR-07/48, 2007.

2. Krodel, J., “Commercial Off-The-Shelf (COTS) Avionics Software Study,” FAA report DOT/FAA/AR-01/26, May 2001.

3. Thornton, R., “Review of Pending Guidance and Industry Findings on Commercial Off-The-Shelf (COTS) Electronics in Airborne Systems,” FAA report DOT/FAA/AR-01/41, August 2001.

4. Halwan, V., and Krodel, J., “Study of Commercial Off-The-Shelf (COTS) Real-Time Operating Systems (RTOS) in Aviation Applications,” FAA report DOT/FAA/AR-02/118, December 2002.

5. Krodel, J., “Commercial Off-The-Shelf Real-Time Operating System and Architectural Considerations,” FAA report DOT/FAA/AR-03/77, February 2004.

6. RTCA, DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” December 1992.

7. RTCA, DO-248B, Final Report for Clarification of DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” December 2001.

8. RTCA, DO-254, “Design Assurance Guidance for Airborne Electronic Hardware,” April 2000.

9. RTCA, DO-255, “Requirements Specification for Avionics Computer Resource (ACR),” June 2000.

10. RTCA, DO-278, “Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance,” March 2002.

11. RTCA, DO-297, “Guidance and Certification Considerations for Integrated Modular Avionics (IMA),” November 2005.

12. FAA Order 8110.49, “Software Approval Guidelines,” June 2003.

13. SAE, ARP 4754, “Aerospace Recommended Practice 4754 Certification Considerations for Highly Integrated or Complex Aircraft Systems,” 1996.

14. SAE, ARP 4761, “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,” December 1996.

15. AEEC, ARINC 651_97—ARINC, Design Guidance for Integrated Modular Avionics, ARINC Specification 651, September 1991.

16. AEEC, ARINC 653_97-ARINC, Aviation Application Software Standard Interface, January 1997.

17. Conmy, Nicholson, Purwantoro, and McDermid, “Safety Analysis and Certification of Open Distributed Systems,” Department of Computer Science, University of York, York, YO10 5DD UK.

18. Lee, Yann-Hang, and Kim, Daeyoung, “Periodic and Aperiodic Task Scheduling in Strongly Partitioned Integrated Real-Time Systems,” 2001.

19. Hicks & Nettles, “Dynamic Software Updating,” ACM Transactions on Programming Languages and Systems, 2005.

11. RELATED DOCUMENTATION.

1. Ainsworth, Eastaughffe and Simpson, “Safety Cases for Software-Intensive Systems,” Praxis Critical Systems Limited.

2. Airlines Electronic Engineering Committee (AEEC), ARINC 653—Part 2: Avionics Application Software Standard Interface, January 22, 2007.

3. Audsley, N., Burns A., and Wellings, A., “Implementing a High-Integrity Executive using Ravenscar,” *Ada Letters*, Vol. XXI, No. 1, pp. 40-45, 2001.

4. Bate-Bate, Conmy, and McDermid, “Generating Evidence for Certification of Modern Processors for Use in Safety-Critical Systems,” Department of Computer Science, University of York, York, UK, 2000.

5. DiVito, “A Formal Model of Partitioning for Integrated Modular Avionics,” NASA/CR-1998-208703, August 1998.

6. FAA, TSO-C153: Integrated Modular Avionics Hardware Elements, May 6, 2002.

7. Ferrell and Ferrell, “Software Service History Handbook,” FAA report DOT/FAA/AR-01/116, January 2002.

8. Fox, Lantner, and Marcom, “A Software Development Process for COTS-based Information System Infrastructure,” *Fifth International Symposium on Assessment of Software Tools and Technologies*, June 2-5, 1997.

9. FAA, AC 20-145: Guidance for Integrated Modular Avionics (IMA) that Implement TSO-C153 Authorized Hardware Elements, February 25, 2003.

10. Alves-Foss, Rinker, and Taylor, “Towards Common Criteria Certification for DO-178B Compliant Airborne Software.”
Available at: http://www.cs.uidaho.edu/~jimaf/docs/compare02b.htm

11. Lewis and Rierson, “Certification Concerns with Integrated Modular Avionics (IMA) Projects,” Washington, D.C., October 2003.

12. Rierson, Rybecky, SC-200, WP#224, “Design Assurance,” August 6, 2003.

13. Rushby, John, “Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance,” SRI, NASA/CR-1999-209347, June 1999.

14. RTCA, DO-160D, “Environmental Conditions and Test Procedures for Airborne Equipment,” July 1997.

15. AEEC, ARINC 652, “Guidelines for Avionics Software Management,” January 1993.

16. ANSI-IEEE, Standard 729-1983, “Glossary of Software Engineering Terminology,” 1983.

17. FAA, AC 20-148, “Reusable Software Components,” December 7, 2004.

12. GLOSSARY.

This section provides key definitions and acronyms used throughout the report. If no source is identified, then the source of the definition is from the authors for the purposes of this report.

Aperiodic—Occurs predictably but not at regular time intervals.

Application—Software and/or application-specific hardware with a defined set of interfaces that when integrated with a platform(s) performs a function. The source of this definition is DO-297.

Application Supplier—The developer that supplies software and/or application-specific hardware with a defined set of interfaces that when integrated with a platform(s) performs a function.

Baseline—The approved, recorded configuration of one or more configuration items that thereafter serves as the basis for further development, and that is changed only through change control procedures. The source of this definition is DO-297.

Board Support Package—A software layer between the hardware and the RTOS that permits the RTOS supplier’s software to run on a variety of hardware configurations.

Channel—A path for interpartition communications, consisting of a set of logically connected ports.

Commitment—An assumption, limitation, performance restriction, behavioral restriction, configuration, or function provided by a module or component, which must be observed by the user of the module or component for proper operation of that module or component.

Component—A self-contained hardware or software part, database, or combination thereof that may be configuration controlled. The source of this definition is DO-297.

Core Software—The operating system and support software that manage platform resources to provide an environment in which an application can execute. The source of this definition is DO-297.

Cyclic—Actions that occur in a fixed repeated order, but not necessarily at fixed time intervals.

Database—A set of data, part or the whole of another set of data, consisting of at least one file that is sufficient for a given purpose or for a given data processing system. The source of this definition is DO-297.

Deadline—A time by which a process must have completed a certain activity.
Determinism/Deterministic—The ability to produce a predictable outcome generally based on preceding operations. The outcome occurs in a specified period of time with some degree of repeatability. The source of this definition is DO-297.

Dormant—A process that is available to execute but is not currently executing or waiting to execute. The source of this definition is ARINC Specification 653.

Failure—The inability of a system or system component to perform a required function within specified limits. A failure may be produced when a fault is encountered. The sources of this definition are IEEE Standard 729-1983 and DO-178B.

Failure, Common Mode—Coincident failures or faults resulting from an error present in several identical redundant hardware or software components; typically generic in nature, designed-in fault. The source of this definition is ARINC Report 652.

Fault—A manifestation of an error in software. A fault, if encountered, may cause a failure. The source of this definition is DO-178B.

Fault Detection—The ability to positively identify that a failure has occurred (i.e., a fault was triggered). The source of this definition is ARINC Report 652.

Fault Isolation—The ability of a system to identify the location of a fault once a failure has occurred. The source of this definition is ARINC Report 652.

Hardware/Software Integration—The process of combining software into the target computer.

IMA System Integrator—The developer who performs the activities necessary to integrate the platform(s), modules, and components with the hosted applications to produce the IMA system.

Independence—
1. Separation of responsibilities that assures the accomplishment of objective evaluation.
   - For software verification process activities, independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified, and a tool(s) may be used to achieve an equivalence to the human verification activity.
   - For the software quality assurance process, independence also includes the authority to ensure corrective action.
2. A design concept that ensures that the failure of one item does not cause a failure of another item. The source of this definition is DO-297.

Initialization—A sequence of actions that brings the system or component thereof to a state of operational readiness. The source of this definition is DO-297.

Interpartition—Refers to any communication conducted between partitions.

Intrapartition—Refers to any communication conducted within a partition.

Integrated Modular Avionics (IMA)—A shared set of flexible, reusable, and interoperable hardware and software resources that, when integrated, form a platform that provides services designed and verified to a defined set of safety and performance requirements to host applications that perform aircraft functions. The source of this definition is DO-297.

Linker—A program that assembles individual modules of object code, which reference each other into a single module of executable object code.

Loader—A routine that transfers object programs and data from some external medium into nonvolatile memory of the target system.

Message—A continuous block of data with a defined length, which is transported by the system (either by the communication network or within a module). The source of this definition is DO-297.

Partition—An allocation of resources whose properties are guaranteed and protected by the platform from adverse interaction or influences from outside the partition. The source of this definition is DO-297.

Partitioned Real-Time Operation System (RTOS)—An operating system that satisfies the requirements for temporal and spatial partitioning.

Periodic—Occurs cyclically with a fixed time period.

Platform and Module Suppliers—The developer that supplies a module or group of modules, including core software that manages resources in a manner sufficient to support at least one application. A component or collection of components can comprise a module.

Port—A partition-defined resource for sending or receiving messages over a specific channel. The attributes of a port define the message requirements and characteristics needed to control transmissions.

Pre-emptive—Undertaken or initiated to deter or prevent an anticipated process. The executing process may be suspended to allow a higher priority process to execute.

Process—A programming unit contained within a partition, which executes concurrently with other processes of the same partition. A process is the same as a task. The source of this definition is ARINC Report 651.

Robust Partitioning—A mechanism for assuring the intended isolation of independent aircraft operational functions residing in shared computing resources in all circumstances, including hardware and programming errors. The objective of Robust Partitioning is to provide the same level of functional isolation as a federated implementation.

RTOS Supplier—The RTOS supplier, as a member of the platform and module supplier role, has critical responsibilities of protection with regards to space, time, I/O, and other shared resources on the IMA system.

Suspended—Process in waiting state. Execution has been temporarily halted awaiting completion of another activity or occurrence of an event.

System Partition—A partition that requires interfaces outside the ARINC 653 defined services, but is still constrained by robust spatial and temporal partitioning. A system partition may perform functions such as managing communication from hardware devices or fault management schemes. System partitions are optional, and are specific to the core module implementation.

Task—A programming unit contained within a partition, which executes concurrently with other processes of the same partition. A task is the same as a process. The source of this definition is ARINC Report 651.