

AD-A142 777

2nd AFSC STANDARDIZATION CONFERENCE

COMBINED PARTICIPATION BY:
DOD-ARMY-NAVY-AIR FORCE-NATO



30 NOVEMBER - 2 DECEMBER 1982
TUTORIALS: 29 NOVEMBER 1982

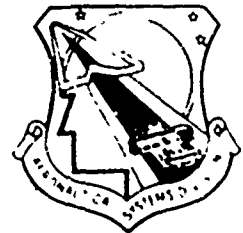
DAYTON CONVENTION CENTER
DAYTON, OHIO

DTIC FILE COPY

SPONSORED BY



HOSTED BY



PROCEEDINGS

NOTICE

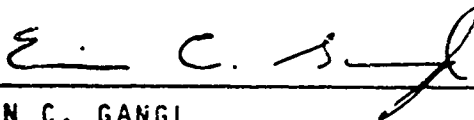
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



JEFFERY L. PESLER
Vice Chairman
2nd AFSC Standardization Conference



ERWIN C. GANGL
Chief, Avionics Systems Division
Directorate of Avionics Engineering

FOR THE COMMANDER



ROBERT P. LAVOIE, COL, USAF
Director of Avionics Engineering
Deputy for Engineering

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify ASD/ENAS, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ASD(ENA)-TR-82-5031, VOLUME II	2. GOVT ACCESSION NO. ADA 142777	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proceedings Papers of the Second AFSC Avionics Standardization Conference	5. TYPE OF REPORT & PERIOD COVERED Final Report 29 November - 2 December 1982	
	6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) Editor: Cynthia A. Porubcansky	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS HQ ASD/ENAS Wright-Patterson AFB OH 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
11. CONTROLLING OFFICE NAME AND ADDRESS HQ ASD/ENA Wright-Patterson AFB OH 45433	12. REPORT DATE November 1982	
	13. NUMBER OF PAGES	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same as Above	15. SECURITY CLASS. (of this report) Unclassified	
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) N/A		
18. SUPPLEMENTARY NOTES N/A		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer Instruction Set Architecture, Multiplexing, Compilers, Support Software, Data Bus, Rational Standardization, Digital Avionics, System Integration, Stores Interface, Standardization, MIL-STD-1553, MIL-STD-1589 (JOURNAL), MIL-STD-1750, MIL-STD-1760, MIL-STD-1815 (ADA), MIL-STD-1862 (NEBULA).		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This is a collection of UNCLASSIFIED papers to be distributed to the attendees of the Second AFSC Avionics Standardization Conference at the Convention Center, Dayton, Ohio. The scope of the Conference includes the complete range of DoD approved embedded computer hardware/software and related interface standards as well as standard subsystems used within the Tri-Service community and NATO. The theme of the conference is "Rational Standardization". Lessons learned as well as the pros and cons of standardization are highlighted.		

DD FORM 1473

1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This is Volume 2

Volume 1	Proceedings pp. 1-560
Volume 2	Proceedings pp. 561-1131
Volume 3	Governing Documents
Volume 4	MIL-STD-1553 Tutorial
Volume 5	MIL-STD-1589 Tutorial
Volume 6	MIL-STD-1679 Tutorial
Volume 7	MIL-STD-1750 Tutorial
Volume 8	MIL-STD-1815 Tutorial
Volume 9	Navy Case Study Tutorial

PROCEEDINGS OF THE

**2nd AFSC
STANDARDIZATION CONFERENCE**

30 NOVEMBER - 2 DECEMBER 1982

**DAYTON CONVENTION CENTER
DAYTON, OHIO**

Sponsored by:

Hosted by:

Air Force Systems Command

Aeronautical Systems Division

FOREWORD

THE UNITED STATES AIR FORCE HAS COMMITTED ITSELF TO "STANDARDIZATION." THE THEME OF THIS YEAR'S CONFERENCE IS "RATIONAL STANDARDIZATION," AND WE HAVE EXPANDED THE SCOPE TO INCLUDE US ARMY, US NAVY AND NATO PERSPECTIVES ON ONGOING DOD INITIATIVES IN THIS IMPORTANT AREA.

WHY DOES THE AIR FORCE SYSTEMS COMMAND SPONSOR THESE CONFERENCES? BECAUSE WE BELIEVE THAT THE COMMUNICATIONS GENERATED BY THESE GET-TOGETHERS IMPROVE THE ACCEPTANCE OF OUR NEW STANDARDS AND FOSTERS EARLIER, SUCCESSFUL IMPLEMENTATION IN NUMEROUS APPLICATIONS. WE WANT ALL PARTIES AFFECTED BY THESE STANDARDS TO KNOW JUST WHAT IS AVAILABLE TO SUPPORT THEM: THE HARDWARE; THE COMPLIANCE TESTING; THE TOOLS NECESSARY TO FACILITATE DESIGN, ETC. WE ALSO BELIEVE THAT FEEDBACK FROM PEOPLE WHO HAVE USED THEM IS ESSENTIAL TO OUR CONTINUED EFFORTS TO IMPROVE OUR STANDARDIZATION PROCESS. WE HOPE TO LEARN FROM OUR SUCCESSES AND OUR FAILURES; BUT FIRST, WE MUST KNOW WHAT THESE ARE AND WE COUNT ON YOU TO TELL US.

AS WE DID IN 1980, WE ARE FOCUSING OUR PRESENTATIONS ON GOVERNMENT AND INDUSTRY EXECUTIVES, MANAGERS, AND ENGINEERS AND OUR GOAL IS TO EDUCATE RATHER THAN PRESENT DETAILED TECHNICAL MATERIAL. WE ARE STRIVING TO PRESENT, IN A SINGLE FORUM, THE TOTAL AFSC STANDARDIZATION PICTURE FROM POLICY TO IMPLEMENTATION. WE HOPE THIS INSIGHT WILL ENABLE ALL OF YOU TO BETTER UNDERSTAND THE "WHY'S AND WHEREFORE'S" OF OUR CURRENT EMPHASIS ON THIS SUBJECT.

MANY THANKS TO A DEDICATED TEAM FROM THE DIRECTORATE OF AVIONICS ENGINEERING FOR ORGANIZING THIS CONFERENCE; FROM THE OUTSTANDING TECHNICAL PROGRAM TO THE UNGLAMOROUS DETAILS NEEDED TO MAKE YOUR VISIT TO DAYTON, OHIO A PLEASANT ONE. THANKS ALSO TO ALL THE MODERATORS, SPEAKERS AND EXHIBITORS WHO RESPONDED IN SUCH A TIMELY MANNER TO ALL OF OUR PLEAS FOR ASSISTANCE.


 ROBERT P. LAVOIE, COL, USAF
 DIRECTOR OF AVIONICS ENGINEERING
 DEPUTY FOR ENGINEERING

Accession For		
NTIS GRA&I	<input checked="checked" type="checkbox"/>	
DTIC TAB	<input type="checkbox"/>	
Unannounced	<input type="checkbox"/>	
Justification		
A-1		



jj h-a-b



DEPARTMENT OF THE AIR FORCE
HEADQUARTERS AIR FORCE SYSTEMS COMMAND
ANDREWS AIR FORCE BASE DC 20334

28 AUG 1982

REPLY TO
ATTN OF CV

SUBJECT Second AFSC Standardization Conference

TO ASD/OC

1. Since the highly successful standardization conference hosted by ASD in 1980, significant technological advancements have occurred. Integration of the standards into weapon systems has become a reality. As a result, we have many "lessons learned" and cost/benefit analyses that should be shared within the tri-service community. Also, this would be a good opportunity to update current and potential "users." Therefore, I endorse the organization of the Second AFSC Standardization Conference.

2. This conference should cover the current accepted standards, results of recent congressional actions, and standards planned for the future. We should provide the latest information on policy, system applications, and lessons learned. The agenda should accommodate both government and industry inputs that criticize as well as support our efforts. Experts from the tri-service arena should be invited to present papers on the various topics. Our AFSC project officer, Maj David Hammond, HQ AFSC/ALR, AUTOVON 858-5731, is prepared to assist.

ROBERT M. BOND, Lt Gen, USAF
Vice Commander

2nd AFSC STANDARDIZATION CONFERENCE

ORGANIZATION COMMITTEE

EXECUTIVE CHAIRMAN

Erwin C. Gangl

EXECUTIVE VICE CHAIRMAN

Jeffery L. Pesler

PROGRAM CHAIRMAN

Jerry L. Duchene

CO-CHAIRMAN

Harold J. Alber

CO-CHAIRMAN

Maj Lee Cheshire

CO-CHAIRMAN

David J. Krile

CO-CHAIRMAN

John Slivinski

EXHIBITS CHAIRMAN

Lt C.W. (Bud) Meynard

PUBLICATIONS CHAIRMAN

Cindy Porubcansky

SPECIAL ARRANGEMENTS CHAIRMAN

Lt Dennis A. Shoulders

PROTOCOL OFFICER

Capt Francis A. DeCurtis

ADMINISTRATIVE CHAIRMAN

Marie P. Jankovich

TREASURER

Richard H. McBride

CONFERENCE MANAGER

Systems Productivity & Management Corporation

Tuesday Luncheon

Keynote Speaker

Major General Marc C. Reynolds

Major General Marc C. Reynolds is Commander of the Air Force Acquisition Logistics Division, and Deputy Chief of Staff for Acquisition Logistics, Air Force Logistics Command, Wright-Patterson Air Force Base, Ohio.

General Reynolds was born in Chamberlain, S.D., on June 2, 1928, and graduated from Chamberlain High School in 1946. He subsequently attended Dakota Wesleyan University and the University of Denver until the outbreak of the Korean War. He holds a Bachelor's Degree in Political Science from the University of Rhode Island and is a graduate of the Air Command and Staff College and the Naval War College.

General Reynolds entered the Air Force as an aviation cadet in January 1951 at Perrin Air Force Base, Texas, and was commissioned upon graduation from pilot training at Vance Air Force Base, Okalahoma, in February 1952. He then attended jet interceptor training at Moody Air Force Base, Georgia, and Tyndall Air Force Base, Florida.

In July 1952, General Reynolds was assigned pilot duty with the 83rd Fighter-Interceptor Squadron at Hamilton Air Force Base, California, and in September he moved with the squadron to Paine Air Force Base, Washington. In March 1953, he was transferred to the 4th Fighter-Interceptor Squadron at Naha Air Base, Okinawa, where he continued to serve as a fighter-interceptor pilot, flying the F-94B.

His next assignment, in September 1954, was Otis Air Force Base, Mass., where he served with the 437th and 60th Fighter-Interceptor Squadrons as a tactical and training flight commander, flying the F-94C and F-101B, and with the 602d Consolidated Maintenance Squadron as a maintenance officer.

General Reynolds was transferred to Europe in November 1961, assigned to the 10th Tactical Reconnaissance Wing, with duty at RAF Station Bruntingthorpe, England, as a Flight Commander, and later at Toul-Rosieres Air Base, France, as Chief of the Wing Standardization Evaluation Branch.

After Command and Staff College at Maxwell Air Force Base, Alabama, General Reynolds was assigned to the 22d Tactical Reconnaissance Squadron, Mountain Home Air Force Base, Idaho. In November 1966, he moved to the 460th Tactical Reconnaissance Wing at Tan Son Nhut Air Base, Republic of Vietnam, and flew 230 combat missions over North and South Vietnam in RF-4C.

(over)

vt work vii

Following his Southeast Asia tour, he served in Japan as Deputy Chief of the Reconnaissance Division, Headquarters Fifth Air Force, Fuchu Air Station. In April 1970, he moved to Misawa Air Base as Commander of the 16th Tactical Reconnaissance Squadron.

General Reynolds returned to the United States in February 1971, assigned to Shaw Air Force Base, S.C., where he served as Assistant Deputy Commander for Operations in the 363d Tactical Reconnaissance Wing. He attended the Naval War College at Newport, R.I., in 1972-73 and was subsequently assigned to Ogden Air Logistics Center, Hill Air Force Base, Utah, initially as the Director of Distribution and later as Director of Maintenance. In July 1976, he was transferred to McClellan Air Force Base, California, as the Director of Materiel Management, Sacramento Air Logistics Center. In March 1978, he became the Center Vice Commander. He transferred to the Air Force Acquisition Logistics Division in May 1980, where he served as Vice Commander until October 1981, when he assumed his present duties.

General Reynolds is a command pilot with more than 5,200 hours flying time, including 475 combat hours. His military decorations and awards include the Distinguished Service Medal, Legion of Merit, Distinguished Flying Cross, Meritorious Service Medal with one oak leaf cluster, Air Medal with 15 oak leaf clusters, and Air Force Commendation Medal with two oak leaf clusters.

He was promoted to Major General Sept 8, 1980, with date of rank July 1, 1977.

General Reynolds was married to the former Judy Coppage of Falmouth, Mass., who died in February 1982. Their children are Barbara and Scott.

30

1

COMPONENT PART NOTICE

THIS PAPER IS A COMPONENT PART OF THE FOLLOWING COMPILATION REPORT:

(TITLE): Proceedings Papers of the AFSC (Air Force Systems Command) Avionics
Standardization Conference (2nd) Held at Dayton, Ohio on 30 November-
2 December 1982. Volume 2.

DTIC
S ELECTRIC D
JUL 13 1984

(SOURCE): Aeronautical Systems Div., Wright-Patterson AFB, OH.

To ORDER THE COMPLETE COMPILATION REPORT USE AD-A142 777.

A

THE COMPONENT PART IS PROVIDED HERE TO ALLOW USERS ACCESS TO INDIVIDUALLY AUTHORED SECTIONS OF PROCEEDINGS, ANNALS, SYMPOSIA, ETC. HOWEVER, THE COMPONENT SHOULD BE CONSIDERED WITHIN THE CONTEXT OF THE OVERALL COMPILATION REPORT AND NOT AS A STAND-ALONE TECHNICAL REPORT.

THE FOLLOWING COMPONENT PART NUMBERS COMPRISE THE COMPILATION REPORT:

AD#: .	TITLE:
AD-P003 561	Standards and Integrated Avionic Digital System Architecture.
AD-P003 562	Achieving the Benefits of Modular Avionics Design.
AD-P003 563	Standard ISAS (Instruction Set Architectures) and VLSI (Very Large Scale Integration): Two Interacting Trends.
AD-P003 564	Navy Packaging Standardization Thrusts.
AD-P003 565	An Introduction to the Avionics Integrity Program.
AD-P003 566	Elements for Successful Implementation of Computing Standards.
AD-P003 567	The Application of Standards to the TDY-750 (Tigershark) Mission Computer.
AD-P003 568	Pave Pillar; A Maturation Process for an advanced Avionics Architecture.
AD-P003 569	Advanced Cockpit-Systems Integration,
AD-P003 570	Defense Industry Attitudes About Air Force Interface Standards Report of an Electronics Industries Association Survey.
AD-P003 571	Digital Avionics Design for Validation.
AD-P003 572	Westinghouse Uses U.S. Air Force-Developed Standards.
AD-P003 573	A General Purpose Computer Architecture Investigation Facility.
AD-P003 574	Integrated Approach to a Successful Embedded Computer Resource Project.
AD-P003 575	Concepts for LEX Avionics.
AD-P003 576	MIL-Prime Program System,
AD-P003 577	Options and Opportunities for Standards: A NATO/AGARD Viewpoint.
AD-P003 578	Proposed MIL-STD for Avionics Installation Interfaces.
AD-P003 579	Fiber Optics for the Future - Wavelength Division Multiplexing.

COMPONENT PART NOTICE (CON'T)

AD#:	TITLE:
AD-PO03 580	Integrated CNI (Communication Navigation and Identification) Avionics and Future Standardization.
AD-PO03 581	Architecture, Hardware and Software Issues in Fielding the Next Generation DOD Processors.
AD-PO03 582	Standard Avionics Software: The Future Strategy for Cost-Effective Avionics.
AD-PO03 583	Lantern (Low Altitude Navigation and Targeting Infrared System for Night) - Tommorrow's Software Development Today.
AD-PO03 584	Quantum Leap in Avionics.
AD-PO03 585	Standardized Computing System SDS80.
AD-PO03 586	Tri-Service Combined Altitude Radar Altimeter (CARA): The Army Perspective.
AD-PO03 587	Mate Standardization.
AD-PO03 588	System Planning Tool to Measure Cost Avoidance Resulting from Indendent Assessment Techniques Applied to Test Program Software Development.
AD-PO03 589	A Corporate Approach to an Embedded Software Development and Support Standard.
AD-PO03 590	MIL-STD Defense System Software Development.
AD-PO03 591	Cost/Schedule Management for Software Development.
AD-PO03 592	Software Configuration Management in a Project Environment.
AD-PO03 593	APSE Database User Scenario.
AD-PO03 594	Architectural and Control Considerations for a High Speed Signal Processor Implemented with an Ada (Trademark) Executive.
AD-PO03 595	Planning of Operational Software Implementation Tool.

Approved for Distribution	<input checked="" type="checkbox"/>
Not Approved for Distribution	<input type="checkbox"/>
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
1A-1	

This document has been approved for public release and sale; its distribution is unlimited.

Wednesday Luncheon

Keynote Speaker

Dr. Alan M. Lovelace

Effective 1 Sep 82, Dr. Lovelace was named VP, Productivity and Quality Assurance.

Dr. Lovelace joined General Dynamics Corporation as Vice President, Science and Engineering in July 1981. He had served as Acting Administrator of the National Aeronautics and Space Administration since January of 1981.

Dr. Lovelace joined NASA in 1974 as Associate Administrator for the Office of Aeronautics and Space Technology. He was named Deputy Administrator in June 1976 by President Ford.

Since entering federal service with the U.S. Air Force in 1954, he has held many research management positions. He served at the Air Force Materials Laboratory, Wright-Patterson Air Force Base, Ohio, from 1954 through 1972, having been named Director in 1967.

From 1972 to 1973, he served as Director of Science and Technology with the Air Force Systems Command, Andrews AFB, Washington, D.C. From 1973 to 1974, Dr. Lovelace was Principal Deputy Assistant Secretary of the Air Force for Research and Development.

Dr. Lovelace retired as Deputy Administrator of NASA in December 1980, but stayed with the Administration through the first flight of the Space Shuttle Columbia and the appointment of a new Administrator.

Born in St. Petersburg, Florida, in 1929, Dr. Lovelace received Bachelor's, Master's and Doctoral Degrees in Chemistry from the University of Florida. Awards he has received include the Presidential Citizens Medal, the Department of Defense Exceptional Service Medal, the Air Force Decoration for Exceptional Service, the National Civil Service League Career Service Award, and the Office of Aerospace Research Award for Outstanding Contributions to Research.

He is a Fellow of the American Institute of Aeronautics and Astronautics and the American Astronautical Society, and is a member of the National Academy of Engineering, Air Force Association, Sigma XI and Phi Beta Kappa.

Thursday Luncheon

Keynote Speaker

Charles P. Lecht

Mr. Lecht is President of Lecht Sciences, Inc., a research and think-tank recently established in New York City.

Mr. Lecht is founder and former President/Chairman of the Board of Advanced Computer Techniques Corporation (ACT), a computer software consulting firm.

He holds a B.S. Degree in Mathematics from Seattle University and a M.S. Degree, also in Mathematics, from Purdue. His involvement in the computer field stretches back to 1951, making him an "old-timer" in a very young industry.

Among his earliest professional activities were programming for IBM's Service Bureau and for the MIT community's Lincoln Laboratory/MITRE organizations on a variety of scientific and military simulation projects.

From 1960 to 1962, Mr. Lecht served in the U.S. Army Ordnance Corps, first as Chief of its Programming Division and subsequently of its Mobilization Application Division; Ordnance Industrial Data Agency.

Mr. Lecht came to New York City in 1962, where he founded ACT. In the 17 intervening years, the Company has grown from a one-man show to an international complex employing over 450 persons and deriving more than 50% of its revenues from operations in Europe, Canada and the Middle East as well as the U.S.

In addition to building and presiding over ACT, Mr. Lecht has found time to hold a number of technical posts, author five books and innumerable articles and maintain a heavy schedule of speaking engagements in the U.S. and abroad. In addition to THE WAVES OF CHANGE, his books include three on computer languages and one on project management.

He is a member of the Young Presidents Organization, The Hudson Institute, the Data Processing Management Association, the Association for Computing Machinery and the New York Academy of Sciences.

In 1976, Mr. Lecht was designated by "The Gallagher Presidents' Report" as one of the "10 Best Businessmen in the USA" representing companies with income below \$1 billion. Profiles of Mr. Lecht have appeared in the New Yorker and Datamation, among other publications.

Table of Contents

Volume 1

MIL-STD-1589 Jovial (J-73) High Order Language

	Page
The Evolution of the Jovial/J73 Language from Definition to Use, James T. Pepe, SofTech, Inc.	3
Jovial Standardization, Austin J. Maher, The Singer Company - Kearfott Division	13
Management Overview of the Benefits of Efficient JOVIAL J73/1750A Software Tools, Joel Fleiss, Proprietary Software Systems, Inc.	15
Object Code Optimization in a Standard Compiler, Terence E. Devine, Software Engineering Associates	19
J73AVS: A JOVIAL J73 Automated Verification System, Carolyn Gannon, General Research Corporation	39
JOVIAL Language Control Procedures with a View Toward Ada, Patricia Knoop and Bobby R. Evans, ASD/ADOL, W-PAFB, OH	49
Feasibility Assessment of JOVIAL to Ada Translation, Daniel H. Ehrenfried, AFWAL/AAAF-2	65
The Multiple System OFP Support (MSOS) System, A Pre-PMRT Capability for Evaluating Tactical Software, Marjorie Kirchoff, Intermetrics, Inc. and Victor S. Vajo, Harold Lowery, Air Logistics Command, Warner Robbins AFB	85
Application of JOVIAL (J-73) to Digital Flight Controls, J. H. Robb and P. J. Boatman, General Dynamics, Data Systems Division, Central Center and P. H. Lang, General Dynamics, Fort Worth Division	93

Table of Contents

MIL-STD-1760 Standard Store Interface

	Page
The MIL-STD-1760 Development Program, Claude M. Connell and Bryce M. Sundstrom, Air Force Armament Laboratory, Eglin AFB	103
MIL-STD-1760 Implementation Strategy, Frank T. Woodall, Teledyne Brown Engineering	109
Aircraft-Store Electrical Interconnection System (AEIS) Functional Requirements, J. R. Perkins and D. E. Lautner, Vought Corporation	111
Signal Set Standardization for the Aircraft-Store Electrical Interconnection System, D. E. Lautner and J. R. Perkins, Vought Corporation	123
Consideration of MIL-STD-1760, Aircraft/Store Electrical Interface Standard on Stores Management System Architectures, John E. Sill, Fairchild Space and Electronics Company	139
Aircraft Multiplexing System Architecture and Stores Compatibility, A. DeRuggiero, Grumman Aerospace Corporation and B. Zempolich, NAVAIR	161

Table of Contents

MIL-STD-1862 Nebula 32 Bit Instruction Set Architecture

	Page
The NEBULA and MCF Standardization Programs, Edward Lieblein, US Army Communications - Electronics Command, Ft. Monmouth, NJ	177
The Nebula Standard Computer Architecture, William B. Dietz, Tartan Laboratories, Inc.	187
Transportability of Nebula Software, Guy L. Steele, Jr., Carnegie-Mellon University, Pittsburgh, Pennsylvania	207
The Impact of NEBULA, MCF and ADA on Real-Time Embedded Computer Systems, F. E. Wiebker, RCA Government Systems Division	219
PAS/NEB: A PASCAL Language Tool for the MCF Elizabeth Souren, Guido Lonardo and Dr. Robert Couranz, Raytheon Company	225

MIL-STD-1553 Multiplex Data Bus

A Programmable Bus Control Interface, Robert M. Salter, Sperry Univac	241
Single Chip MIL-STD-1553B Bus Interface Unit Keeps Pace With Chip Sets, Scott Schaire, Grumman Aerospace Corporation	255
MIL-STD-1553B Marconi LSI Chip Set in a Remote Terminal Application, Albert DiMarino	269

Table of Contents

	Page
MIL-STD-1553 Interface Application Notes, Steve Friedman, ILC Data Device Corporation	277
Application of 1553B to MRASM - A Systems Look, John E. Leib, General Dynamics Convair Division	295
MIL-STD-1553B in MRASM - The Designer's Challenge - Vicki L. Elmore, General Dynamics Convair Division	305
Third Generation MIL-STD-1553B LSI Chip Set, Robin D. Beasley, Marconi Avionics, Ltd, Rochester, Kent, UK	315
Data Word Standardization, Francis E. Peter, Naval Avionics Center	327
MIL-STD-1553B Validation Testing, Duane J. Thorpe and Kumar V. Vakkalanka, ASD/ENAS (SEAFAC), W-PAFB, OH	337
MIL-STD-1553: Testing and Test Equipment, Leroy Earhart, Test Systems, Inc.	349
A Common 1553B I/O Channel for the F-16, Stephen Alford, General Dynamics - Fort Worth Division	367

MIL-STD-1750 16 Bit Instruction Set Architecture

MIL-STD-1750A Users Group, C. Ray Turner, The Boeing Company and Marlin L. Wagner, Sperry Univac Defense Systems	385
---	-----

Table of Contents

	Page
MIL-STD-1750A Microprocessor Chip Set Development, Dr. Thomas A. Longo and Dan Wilnai, Fairchild Camera and Instrument Corporation	395
A High Performance MIL-STD-1750A Microprocessor, T. L. Rasset and J. H. Lane, McDonnell Douglas Aeronautics Company	405
A MIL-STD-1750A Computer Employing the MDAC CMOS-SOS Chipset, Charles Frank and Larry Speelman, ROLM Corporation	413
A New Silicon-on-Sapphire MIL-STD-1750A Microprocessor, Joseph R. Burns, Henry Silcock, Gregory Portanova and Steven Nicholas, Mikros Systems Corporation	429
Delco Electronics Standard Architecture Military Computers, Dr. Clive D. Leedham, Delco Electronics Division, General Motors Corporation	445
The Use of Computer ISA and Software Standards at Westinghouse Defense Electronics Center, Dr. Marvel A. Geyer, John G. Gregory and Harvey R. Moran, Jr., Westinghouse Defense Electronics Center	447
Transportable GPU Chip Set Technology for Standard Computer Architectures, Robert E. Fosdick and Harvey C. Denison, Tracor Aerospace	457
MIL-STD-1750A As A Spaceborne Instruction Set Architecture, Robert N. Constant, Richard C. Fleming, Edwin M. Garcia Marvin Lubofsky, and Donald R. O'Bell, The Aerospace Corporation	479
MIL-STD-1750A Verification Testing, Luis E. Velez, ASD/ENAS (SEAFAC), W-PAFB, OH	493

Table of Contents

MIL-STD-1815 Ada High Order Language

	Page
Update of the State of Ada Language Standardization and Other Ada Related Standards, Col Lawrence E. Druffel, HQ USAF, DARPA Program Office	503
Navy Transition to Ada - Potential for a Fresh Start, Owen L. McOmber, HQ Naval Material Command	505
Introducing Ada into the USAF, Maj David A. Hammond and Capt Michael C. Vinyard, HQ AFSC	507
Ada as a Program Design Language - A Rational Approach to Transitioning Industry to the World of Ada Through a Program Design Language Criteria, Robert M. Blasewitz, RCA Government Systems Division	509
The KAPSE Interface Team, Patricia A. Oberndorf, Naval Ocean Systems Center	519
A Standard Run-Time Executive for Compiled Ada, Ben Hyde, Intermetrics, Inc.	527
The Ada Run-Time Environment, Dr. Joseph K. Cross, Sperry Univac, Defense Systems Division	533
The Ada Work Center - Its Features, Capabilities and Developments, David Babcock, Rolm Corporation	539
A Code of Practice to Constrain ADA, Dr. Tim Swann, Marconi Avionics Limited	541
Use of Ada in System Design: A Case Study, Michael B. Patrick and Hal C. Ferguson, General Dynamics Data Systems Division Central Center	543
Ada Training Considerations, Christine L. Braun, SofTech, Inc.	545

Table of Contents

Volume 2

Standardization Issues - Near Term

	Page
Standards and Integrated Avionic Digital System Architecture, Edward L. Griffin, Martin Marietta Orlando Aerospace	563
Achieving the Benefits of Modular Avionics Design, Stephen W. Behnen, Fred M. Lightfoot and Peter R. Metz, Boeing Military Airplane Company	583
Standard ISA's and VLSI: Two Interacting Trends, Dr. Peter M. Kogge, IBM Federal Systems Division	597
Successful Development/Supply of Standardized Computers in a Period of Rapid Technological Change, Keith B. Dixon, Ferranti Computer Systems Ltd	611
AN/UYK-43 (V) and AN/UYK-44 (V) Program Overview, Capt James P. O'Donovan, Naval Sea Systems Command	613
AN/AYK-14 (V) Program Overview, Henry H. Mendhall, Naval Air Systems Command	615
Navy Packaging Standardization Thrusts, John R. Kidwell, Naval Avionics Center, Indianapolis, Indiana	617
An Introduction to the Avionics Integrity Program, James E. Verdier, ASD/ENASA, W-PAFB, OH	633

Table of Contents

Advanced Systems Architecture

	Page
Avionic Architecture - Past and Future, T.V. McTigue, McDonnell Aircraft Company	641
Elements for Successful Implementation of Computing Standards, Gordon R. Enoland, General Dynamics Corporation	643
B-1B Avionics Applications of Military Standards, L. M. Carrier and G. A. Kinstler, Rockwell International	655
Standards Application to B-1B Avionics Program, H. L. Ernst, Boeing Military Airplane Company	657
The Application of Standards to the TDY-750 (Tigershark) Mission Computer, David W. Geyer, Teledyne Systems Company	659
PAVE PILLAR: A Maturation Process for an Advanced Avionics Architecture, D. Reed Morgan and Lt Col R. Bellem, AFWAL/AAA, W-PAFB, OH	675
FACET - Integration and Standardization Thrusts in US Army AVRADA CNI Development Efforts, Arthur W. Lindberg, US Army Avionics R&D Activity, Fort Monmouth, New Jersey	691
Avionics Control Architecture of Army Helicopter Improvement Program (AHIP), Glenn P. Tomlin, Jr., US Army Avionics R&D Activity, St. Louis, MO	693
Advanced Cockpit - Systems Integration, Graham Roe, British Aerospace P.L.C., UK	695

Table of Contents

The Digital Interface Challenge

	Page
Defense Industry Attitudes About Air Force Interface Standards: Report of an Electronics Industries Association Survey, Peter N. Pocalyko, IBM Corporation and Chandler E. Swallow, Jr., Sperry Univac	721
Digital Avionics Design for Validation, Ellis F. Hitt, Battelle, Columbus Laboratories	729
HH-60D Advanced Avionics Architecture, Ira Glickstein, IBM Federal Systems Division	751
Westinghouse Uses US Air Force - Developed Standards, Carl S. Shyman, Westinghouse Defense Electronics Center	753
A General Purpose Computer Architecture Investigation Facility, Lt. Dean W. Gonzalez, RADC/COEA, Griffiss AFB, NY	767
An Integrated Approach to a Successful Embedded Computer Resource Project, Leo G. Egan, ITT/Federal Electric Corporation	777
Is A Federal Software Engineering Series Needed?, Gwendolyn Hunt, Data Processing Service Center/West	813
Concepts For LHX Avionics, LTC Russell H. Smith, US Army Aviation Center, Ft. Rucker, Alabama	815

Table of Contents

Standardization Issues of the Future

	Page
MIL-Prime Program System for Military Specifications and Standards, Frederick T. Rall, Jr., ASD/EN, W-PAFB, OH	823
Options and Opportunities for Standards - A NATO/AGARD Viewpoint, John T. Shepherd, Marconi Avionics Limited and Louis J. Urban, ASD/AX, W-PAFB, OH	843
Proposed MIL-STD for Avionics Installation Interfaces, Maj Gerald Schopf, ASD/XRX, W-PAFB, OH	861
Fiber Optics for the Future - Wavelength Division Multiplexing, J. Larry Spencer, NASA Langley Research Center	871
Integrated CNI Avionics and Future Standardization, Darlow Botha, AFWAL/AAAI, W-PAFB, OH	889
Architecture, Hardware and Software Issues in Fielding the Next Generation DOD Processors, Ole Golubjatnikov, General Electric Company	899
Standard Avionic Software - The Future Strategy for Cost-Effective Avionics, Edward C. Straub, ARINC Research Corporation	927
Application of Standard System Specification Techniques to the Design of Very Large Scale Integrated Circuits, Dave Jordan, Marconi Avionics Limited	947

Table of Contents

Advanced Standardized Systems/Subsystem

	Page
LANTIRN - Tomorrow's Software Development Today!, Kenneth B. Hawks, ASD/RWNM, W-PAFB, OH	951
AFTI/F-16 Digital Flight Control System Development, James K. Ramage, AFWAL, W-PAFB, OH	957
Quantum Leap in Avionics, W. E. Cantrell, General Dynamics Corporation	959
Integrated Digital Avionics System (IDAS), Peter Boxman, US Army Avionics R&D Activity, St. Louis, MO	975
Embedded Computer Standardization in the Submarine Advanced Combat System (SUBACS), Ronald L. Ticker, Naval Sea Systems Command	977
Standardized Computing System SDS 80, J. Olsson, Ericsson	979
Navy Real Time Signal Processor Development: Second Generation Planned Service Standard, Capt C. B. Robbins, Naval Sea Systems Command	991
Tri-Service Combined Attitude Radar Altimeter (CARA) The Army Perspective, William M. Gill, US Army Avionics R&D Activity, Fort Monmouth, NJ	993
MATE Standardization, Capt Richard E. Farmer, ASD/AEGB, W-PAFB, OH	1005
Digital Intercom, Capt Darian Ross, ASD/AEAC, W-PAFB, OH	1013

Table of Contents

Standardized Software Development

	Page
System Planning Tool to Measure Cost Avoidance Resulting from Independent Assessment Techniques Applied to Test Program Software Development, P. D. Kidd, Technology Development of California	1017
A Corporate Approach to an Embedded Software Development and Support Standard, D. D. Doe, D. E. Hilt and G. B. Wigle, Boeing Aerospace Company; J. P. Bateman, Boeing Commercial Airplane Company; L. L. Tripp, Boeing Computer Services Company and W. F. Jackson, Boeing Military Airplane Company	1029
MIL-STD Defense System Software Development, Deane F. Bergstrom, Rome Air Development Center, Griffiss AFB, NY	1043
Cost/Schedule Management for Software Development, Maj H. Wendt, DCAS PRO/Ford Aerospace Corporation and M. W. Evans, Ford Aerospace Corporation	1053
Software Configuration Management in a Project Environment, Maj H. Wendt, DCAS PRO/Ford Aerospace Corporation and M. W. Evans, Ford Aerospace Corporation	1071
Revising MIL-STD-1679, William J. Egan, Naval Material Command	1083
APSE Database User Scenario, Elizabeth S. Kean, Rome Air Development Center, Griffiss AFB, NY	1085
Architectural and Control Considerations for a High Speed Signal Processor Implemented with an Ada Executive, Steven E. Adams, Intermetrics, Inc. and Thomas R. Butler, Magnavox Company	1097
Avionic Systems Integration Facilities, Mark van den Broek and Paul M. Vicen, AFLC/LOE	1113
Planning of Operational Software Implementation Tool, Aaron Spinak, Proprietary Software Systems, Inc.	1115

EXHIBITORS

ARINC RESEARCH	RAYCHEM
BENDIX	RAYTHEON
BOMAR INSTRUMENTS	RCA
CIRCUIT TECHNOLOGY	ROLM
CONTROL DATA CORPORATION	SANDERS ASSOCIATES
DELCO ELECTRONICS	SCI SYSTEMS
FAIRCHILD CAMERA & INSTRUMENTS	SEAFAC
FAIRCHILD SPACE & ELECTRONICS	SINGER KEARFOTT
GARRETT	SMITH INDUSTRIES
GENERAL DYNAMICS, FT. WORTH	SOFTech
GENERAL ELECTRIC	SPECTRAL SYSTEMS
GRUMMAN AEROSPACE	SPERRY UNIVAC
HARRIS SEMICONDUCTOR	STC COMPONENTS
IEEE	SYSTEM PRODUCTIVITY & MANAGEMENT
ILC/DATA DEVICE CORPORATION	SYSTEMS RESEARCH LAB
INTELLIMAC	TELEDYNE SYSTEMS
INTERMETRICS	TELEFONAKTIEBOLAGET LM ERICSSON
KAISER	TEST SYSTEMS
LITTON	TROMPETER
LORAL	TRW
MCDONNELL DOUGLAS ASTRONAUTICS	UTC/MOSTEK
NORTHROP	WESTINGHOUSE DEFENSE

Authors Index

Adams, Steven E., 1097
Alford, Stephen, 367

Babcock, David, 539
Bateman, J. P., 1029
Beasley, Robin D., 315
Behnen, Steven W., 583
Bellem, R., 675
Bergstrom, Deane F., 1043
Blasewitz, Robert M., 509
Boatman, P. J., 93
Botha, Darlow, 889
Boxman, Peter, 975
Braun, Christine L., 545
Burns, Joseph R., 429
Butler, Thomas R., 1097

Cantrell, W. E., 959
Carrier, L. M., 655
Connell, Claude M., 103
Constant, Robert N., 479
Couranz, Robert, 225
Cross, Joseph K., 533

Denison, Harvey C., 457
DeRuggiero, A., 161
Devine, Terence E., 19
Dietz, William B., 187
DiMarino, Albert, 269
Dixon, Keith B., 611
Doe, Dennis D., 1029
Druffel, Lawrence E., 503

Earhart, Leroy, 349
Egan, Leo G., 777
Egan, William J., 1083
Ehrenfried, Daniel H., 65
Elmore, Vickie L., 305
England, Gordon R., 643
Ernst, H. L., 657
Evans, Bobby R., 49
Evans, M. W., 1053, 1071

Farmer, Richard E., 1005
Ferguson, Hal C., 543
Fleiss, Joel, 15
Fleming, Richard C., 479
Fosdick, Robert E., 457
Frank, Charles, 413
Friedman, Steven M., 277

Ed Clark

Gannon, Carolyn, 39
Garcia, Edwin M., 479
Geyer, David W., 659
Geyer, Manvel, 447
Gill, William M., 993
Glickstein, Ira, 751
Golubjatnikov, Ole, 899
Gonzalez, Dean W., 767
Gregory, John G., 447
Griffin, Edward L., 563

Hammond, David A., 507
Hawks, Kenneth B., 951
Hilt, David E., 1029
Hitt, Ellis F., 729
Hunt, Gwendolyn, 813
Hyde, Ben, 527

Jackson, W. F., 1029
Jordan, D., 947

Kean, Elizabeth S., 1085
Kidd, Paul D., 1017
Kidwell, John R., 617
Kinstler, G. A., 655
Kirchoff, Marjorie, 85
Knoop, Patricia A., 49
Kogge, Peter M., 597

Lane, J. H., 405
Lang, P. H., 93
Lautner, D. E., 111, 123
Leedham, Clive D., 445
Leib, John E., 295
Lieblein, Edward, 177
Lightfoot, Fred M., 583
Lindberg, Arthur W., 691
Lonardo, Guido, 225
Longo, Thomas A., 395
Lowery, Harold, 85
Lubofsy, Marvin, 479

Maher, Austin J., 13
McOmber, Owen L., 505
McTigue T. V., 641
Mendhall, Henry H., 615
Metz, Peter R., 583
Moran, Harvey R., 447
Morgan, D. Reed, 675

Nicholas, Steven, 429

O'Bell, Donald R., 479
Oberndorf, Patricia A., 519
O'Donovan, James P., 613
Olsson, J., 979

Patrick, Michael B., 543
Pepe, James T., 3
Perkins, J. R., 111, 123
Peter, Francis E., 327
Pocalyko, Peter N., 721
Portanova, Gregory, 429

Rall, Frederick T., 823
Ramage, James K., 957
Rasset, T. L., 405
Robb, J. H., 93
Robbins, C. B., 991
Roe, Graham, 695
Ross, Darian, 1013

Salter, Robert M., 241
Schaire, Scott, 255
Schopf, Gerald, 861
Shepherd, John T., 843
Shyman, Carl S., 753
Silcock, Henry, 429
Sill, John E., 139
Smith, Russell H., 815
Souren, Elizabeth, 225
Speelman, Larry, 415
Spencer, Larry, 871
Spinak, Aaron, 1115
Steele, Guy L., 207
Straub, Edward C., 927
Sundstrom, Bryce M., 103
Swallow, Chandler E., 721
Swann, Tim Jr., 541

Thorpe, Duane J., 337
Ticker, Ronald L., 977
Tomlin, Glen P., 693
Tripp, L. L., 1029
Turner, C. Ray, 385

Urban, Louis J., 843

Vajo, Victor S., 85
Vakkalanka, Kumar V., 337
VandenBroek, Mark, 1113
Velez, Luis E., 493
Verdier, James E., 633
Vicen, Paul, 1113
Vinyard, Michael C., 507

Wagner, Marlin L., 385
Wendt, H., 1053, 1071
Wigle, G. B., 1029
Wilnai, Dan, 395
Woodall, Frank T., 109
Wuebker, F. E., 219

Zempolich, B., 161

STANDARDIZATION ISSUES - NEAR TERM

SESSION CHAIRMAN: Robert Harris
AFWAL/AAAI

MODERATOR: Colonel Hugo Weichel
AFWAL/AAR

STANDARDS AND INTEGRATED AVIONIC DIGITAL SYSTEM ARCHITECTURE

Edward L. Griffin

Martin Marietta Orlando Aerospace
P. O. Box 5837
Orlando, Florida 32855
Telephone 305-671-2680

ABSTRACT

Integrated digital system design and development of the hardware, software, and interfaces that integrate the avionic flight control, fire control, and man-machine display and control must emphasize the man-rated weapon system's availability and survivability. The scope of tasks including detailed trade studies such as CMOS/SOS versus ECL semiconductor use, and parallel pipelining versus multi-microprocessor architecture usually requires an engineering team with backgrounds from requirements and integration, electronics hardware, packaging, and software. System attributes of fault tolerance, fail safe, and fail soft operation requires total team adherence to a set of design, documentation, implementation, and test standards of which few have complete familiarity. Since use of these standards has prevented costly errors and overruns in procurement, and decreased maintenance costs over the life cycle, this paper shows how to make each effective contributor on the team understand the standards controlling performance and product specifications, change and configuration control, test planning, and test procedure generation for the other areas of expertise.

INTRODUCTION

The air forces of today face a wide spectrum of offensive and defensive weapon systems that contain many types of sensory subsystems, complex flight and maneuver performance characteristics, and multi-mission adaptation capability. The Soviet hunter-killer antisatellite (ASAT) has the potential of depriving battlefield units of command, control, communications and intelligence information (C³I);¹ especially access to the extensive intelligence data bases available in the continental United States. The SHEFFIELD, equipped with missile, gun, and chaff dispenser weapon systems, SEA DART and ADAWS-4 fire control systems, standard naval communications plus Marconi SCOT satellite communication terminals, and the WLR-8 Abbey Hill early warning air surveillance and radar signal processor with a dictionary of 476 radar pulses, evidently did not interpret the frequency agile radar terminally guided Exocet missile as a threat.² From these facts, it is evident that our future attack or defensive weapon systems should be able to support a semiautonomous battlefield element that has only limited command and control interfaces and no intelligence data base updates for periods of 36-72 hours after initiation of, and during hostile actions. These systems

must be low cost to allow sufficient weapon inventory, flexible to respond to reprogramming of mission profiles and target types, responsive to simple controller commands, and reliable, with fault detection and fail soft operation capability. Digital systems engineering techniques can decrease cost, increase flexibility and responsiveness, and when considered in context with the total system, increase reliability.

Secretary of Defense Casper W. Weinberger, in an address to the National Security Seminar, Army War College, Carlisle Barracks, Pa, in June stated:³

"We must also be able to increase the sustainability of our forces --to have the ammunition, fuel, and stocks needed to balance the Soviets' ability to endure a prolonged conventional conflict. In assuming that a U.S.-Soviet conventional war would necessarily be of a shorter duration, previous administrations were forced to accept the probability of escalation to nuclear war if conventional forces were insufficient to end the conflict."

...

"But to rectify the past neglect of sustainability, we have much work to do. For example, the Soviets have enough ammunition to last twice as long as NATO. The discrepancy in stocks of some other items is even more serious. If we permit such a condition to continue, we encourage an adversary to think that he can win a war by outlasting us."

This statement reinforces the need for integrated digital system design and development to support low cost, flexible, responsive and reliable weapon systems. The necessary improvement in our present development techniques can be achieved by pursuit of several of the advanced software techniques, consideration of total digital systems, and use of innovative development management.

As the digital subsystems (digital hardware, computer programs, and serial bus interfaces) have expanded to become an increasing larger part of most modern systems; the usual approach of definition by a systems engineering staff, hardware implementation by an electronics group, and computer programming by software engineers is leading to cost inefficient functional duplication and increasingly difficult integration efforts. A team consisting of systems requirements and integration personnel, electronic engineers, and software engineers must perform and document the interdisciplinary trades that synthesize the system, the electronics, and the software performance requirements as well as the integration and test plans. The people on this team must subjugate pride of membership in their own technology specialty, and be knowledgeable of the practices and standards governing the other disciplines. Elements of this approach are evident in the F-15 Digital System Architecture, Figure 1. Further recognition of this approach's viability are apparent in Wright-Patterson's Pave Pillar program thrusts. Integrated avionic digital system development is the methodology to produce the cost effective, reliable, survivable systems needed today.

F-15 DIGITAL SYSTEM ARCHITECTURE

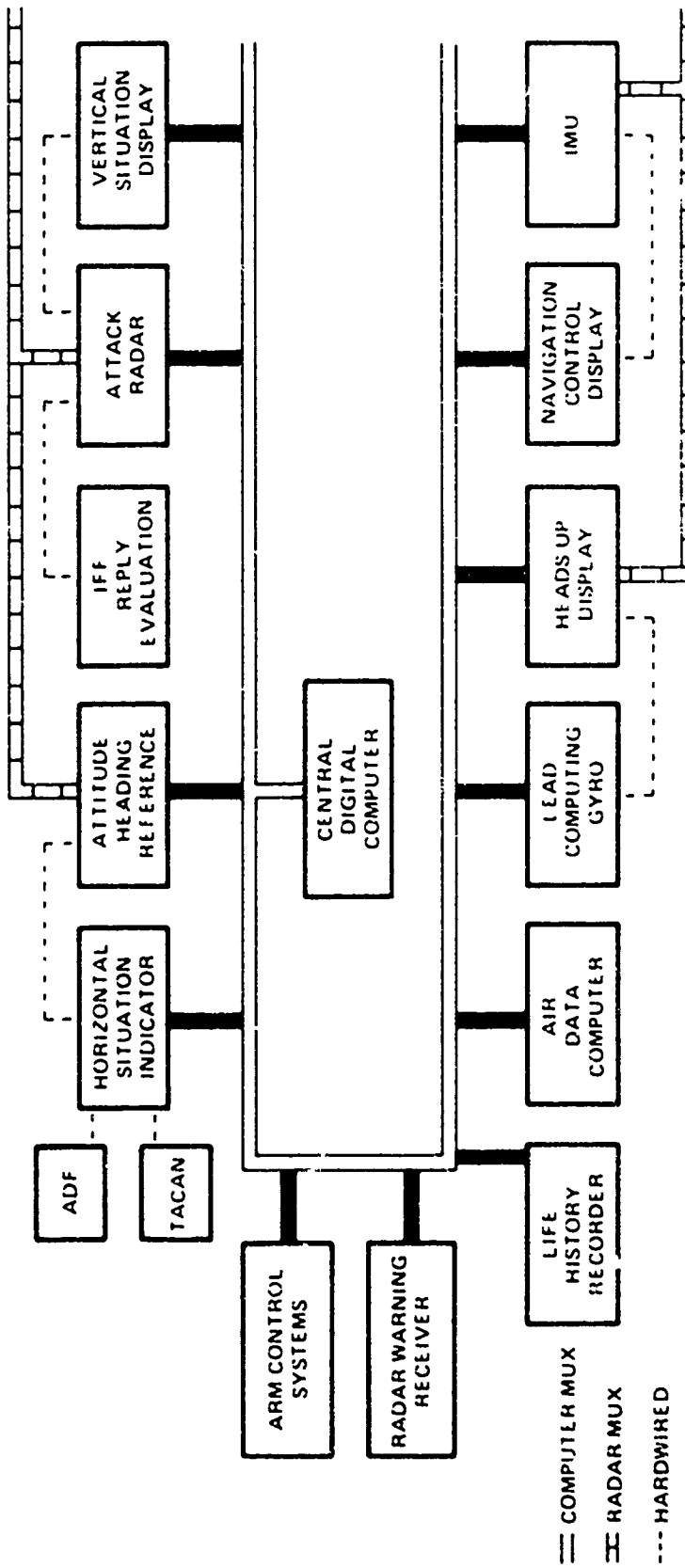


Figure 1

MARTIN MARIETTA

DIGITAL SYSTEM DEVELOPMENT⁴

The methodology for buildup of a digital system is shown in block diagram form in Figure 2. The upper portion of each block defines the main function performed, the center defines type of documentation generated, and the bottom the engineering skill mix required. The key to digital system design and development lies in the front end architectural definition during the conceptual and system performance requirements phase and is reinforced during design refinement, implementation, integration, and acceptance through system test.

Conceptual Design for the system must be attentive to the fact that most modern systems include one or more digital computers, and most modern systems are integrated, tested, and maintained in the digital domain. Additionally, if a system function can readily be implemented in either hardware or software, software will be much more cost effective and more flexible. Finally, system timing requirements are critical to computer selection, Bus-I/O architecture, and software structure. Therefore, consideration of the computer, Bus-I/O architecture, and software structure during the system conceptual design will ensure development of a cost effective digital system to support the total system. The tasks that must be accomplished during the system conceptual design phase to support digital system design are shown in Table 1.

System Trades - Digital interface definition for programmable sensors

- Timing analysis and range of value for responses

HW/SW Trades - Hardware vs. software implementation choices

- Central or multiple computer implementation

- Bus or hardwire interface architecture(s)

- Software vs. programmable hardware

Conceptual Design Tasks

Table 1

The primary thrust during this period of the digital system design is to accentuate an integration, test and maintenance philosophy, decrease hardware replication costs, maintain flexibility for change and growth, and look to simplicity for subsequent producibility.

System Performance Requirements are written to accommodate digital system implementations that preserve a balance of compliant system responses to accomplish the defined mission. The system specification (B-1) should define and allocate performance requirements in terms so that traceability can flow to applicable prime item hardware and software specifications. This definition of performance requirements must be stated so that response requirements and error budget margins are spread cost effectively across the sensor, digital system and interface, and electro-mechanical actuation systems. The tasks that must be accomplished during the system performance

REQUIRED DIGITAL SYSTEM BUILD METHODOLOGY

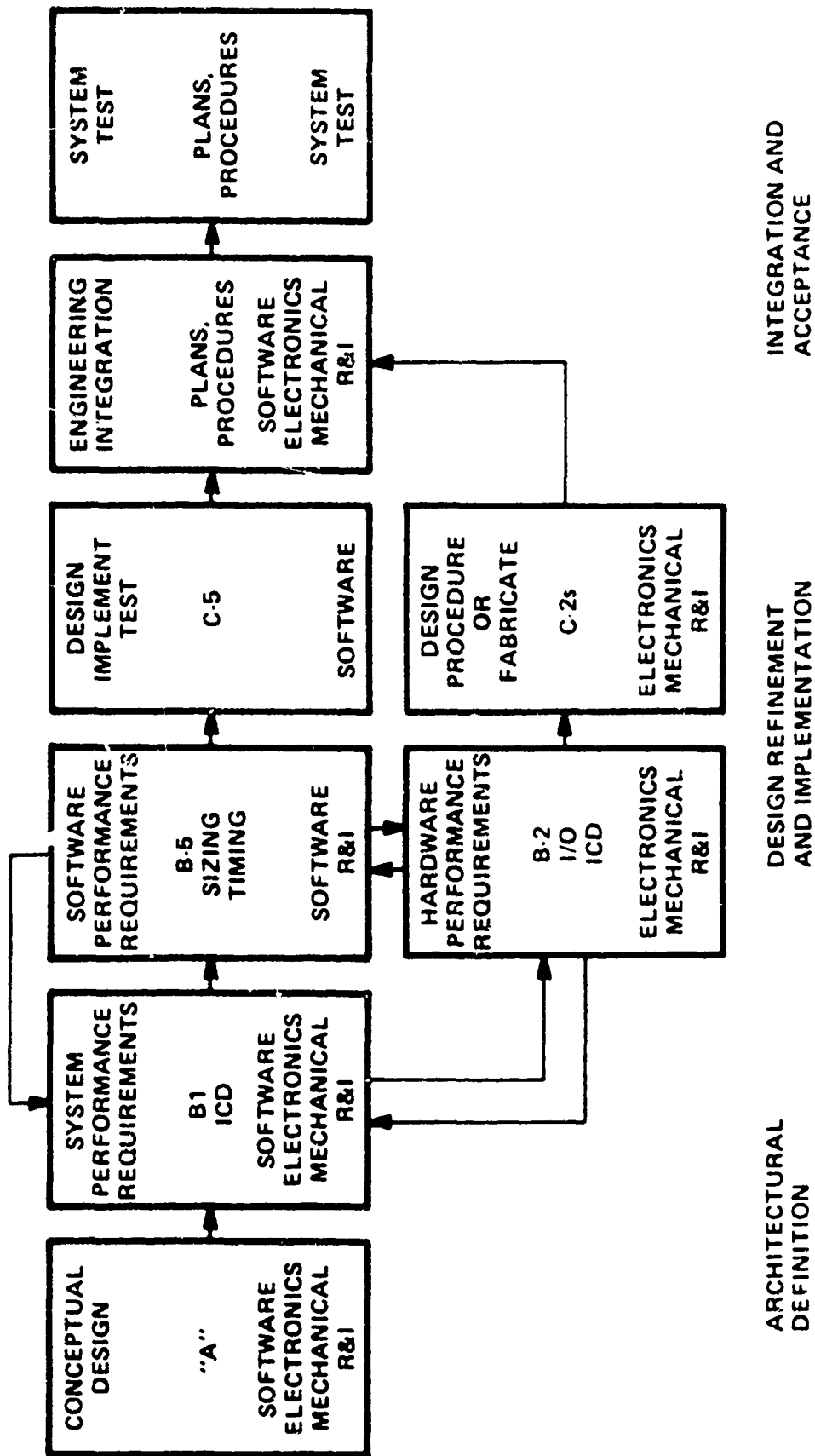


Figure 2

MARTIN MARIETTA

requirements phase to support the digital system design provide a second level of detail to the conceptual studies. These are outlined in Table 2.

- | | |
|----------------------|--|
| Functions/Operations | - Evaluate software flexibility in HW/SW equal choice trades |
| | - Evaluate digital flexibility in analog/digital equal choice trades |
| Interfaces | - Evaluate new design risk vs. standard I/O in sensor interfaces |
| | - Evaluate multiplex bus flexibility in interface design |
| Timing Analysis | - Evaluate special features to support data rate and critical response times |
| | - Evaluate need for high speed interrupt structure |
| | - Evaluate need for special signal processing |
| | - Evaluate need for dedicated I/O |
| | - Evaluate need for special software techniques. |

System Performance Requirements Definition Tasks
Table 2

The primary thrust of the functions/operations considerations is maximum flexibility for the cost expended. The interface definition should maximize ease of integration, test, checkout, and subsequent maintenance. The timing analysis are crucial in that they may dictate the need for special, complex hardware or expensive special features that an alternative approach could minimize. As system complexity is increased, the ease of producibility is decreased.

Hardware/Software Performance Requirements are the result of the detailed hardware/software trades that achieve final minimization of the costs. The hardware and software specifications (B-2's and B-5's) will reflect all of the applicable requirements allocated by the B-1 specification.

The allocation of these performance requirements to hardware and software will be defined to continue to balance cost, flexibility, and growth potential. Additional definition in the form of Interface Control Documents (ICD's) System Design Requirements, and I/O designs will complement the performance requirements where critical interfaces need further design requirement information. The tasks that must be accomplished during the critical item performance specification phase to support the final detailed digital design are shown in Table 3.

- | | |
|------------|---|
| Hardware | <ul style="list-style-type: none"> - Evaluate programmable logic flexibility in equal choice of hardware/programmable - Provide at least 100 percent computational memory and throughput reserve - Evaluate large processor throughput multiplication capability in a multiprogramming/multiprocessing equal choice. |
| Software | <ul style="list-style-type: none"> - Complete detailed sizing and timing analysis - Evaluate HOL economy in an assembly/HOL choice - Analyze HOL memory and throughput overhead. |
| Interfaces | <ul style="list-style-type: none"> - Evaluate standard I/O, buses, and protocol for ease of integration - Evaluate maximum digital domain integration in design guidelines. |

Hardware/Software Performance Requirement Definition Tasks
Table 3

The hardware tasks are oriented toward decrease of cost through increased flexibility, growth potential, and ease of integration. The software tasks are oriented toward definition of critical sizing and timing functions and simplification of software development. The interface tasks are oriented toward design simplicity, ease of integration and checkout, reliability, maintenance, and producibility. In this complex set of trades and definitions, it should be remembered that computational throughput or cost advantages of a scheme can be entirely negated by poor I/O design, inefficient programming, or inefficient software development facilities.

Integration and Acceptance Plans and Procedures are begun by the digital design team during performance requirements definition and completed while the hardware and software implementation is carried out by specialists in the respective disciplines. Working from the basis of hardware and software functions, comprehensive integration planning that must include testing for proper operation should develop procedures that are adaptable for reuse in software qualification testing, hardware acceptance testing, reliability growth analysis, factory testing, and post delivery maintenance. An additional responsibility of the digital design team is to provide direction for implementation of changes when hardware/software implementation at the detailed level discovers the inevitable interdisciplinary inconsistencies. Although the previous emphasis on growth and flexibility should have done much to minimize these impacts, occasionally a hardware change could be found to be more cost effective than a software change.

INTEGRATED AVIONIC DIGITAL SYSTEM ARCHITECTURE

The Air Force at Wright-Patterson has approached Integrated Avionic Digital System development from a total digital system design approach (Pave Pillar contracts). This initiative requires hardware/software trades to optimize a system design for testability, fault tolerant reliability, producibility, and growth. The cost effective trades between VLSI and VHSIC programmable logic arrays, central processing with multiprogramming or multi-processing, and sensor interface definition to adapt to hard-wire or bus interconnect architecture are the keys to this concept. The combined talents of system requirements and integration, software, and electronics engineers are required to accomplish this task. The battle damage survivability/reliability addition to the approach outlined in the digital system development methodology is the major difference between the two techniques.

An integrated avionic architecture using dual and triple redundancy in processors, busses and selected software modules, and combined with fail-soft attributes for selected functions, is a typical system that will meet manned flight control safety margins and provide required fire control accuracy with reduced system component count and cost for present and future systems.

Fly-by-wire flight control with its five computer-three agreement processing safety requirement can be combined with fire control's space throughput inherent in multi-sensor, multi-weapon control and the detect, recognize, track and fire timeline to give a more cost effective architecture that meets total system requirements. An illustration of this type of approach is shown in Figure 3. The major operations of man-machine interface, fire control, and flight control operate on local redundant busses with appropriate speeds, and are interconnected with synchronizers to allow reallocation of system functions to available resources under stress of component failure or battle damage. This approach increases total weapon system reliability and allows increased sortie count by always providing some level of available subsystems based on reconfiguration of available sensors, busses, and processors.

The primary aircraft bus, besides handling man-machine control inputs and displays, has communications (radio) processing control and interface as well as main system storage and processing control. Pilot control and display for the fight weapon delivery, and aircraft operation functions is well understood if not always optimally designed. The use of heads up display, CRT's, keyboards, and multi-function control sticks will be augmented in the future by voice input and output as well as other sensory communication. Much work is underway on use of common electronics for the radio communications and navigation devices. Main system storage and processing control is the function that implements reallocation of subsystem, bus, and processing functions as well as the in system fault detection, diagnostic, and test functions. This approach allows the pilot station to operate as a maintenance console with application of ground power and availability of a technician trained in its use.

The fire control bus has the potential of being multispeed. The highest speed functions, sensor signal processing, run on direct busses between the sensor and its signal processor. A medium speed bus is used for correlation

MULTIBUS, MULTIPROCESSOR, FAULT TOLERANT AVIONIC ARCHITECTURE

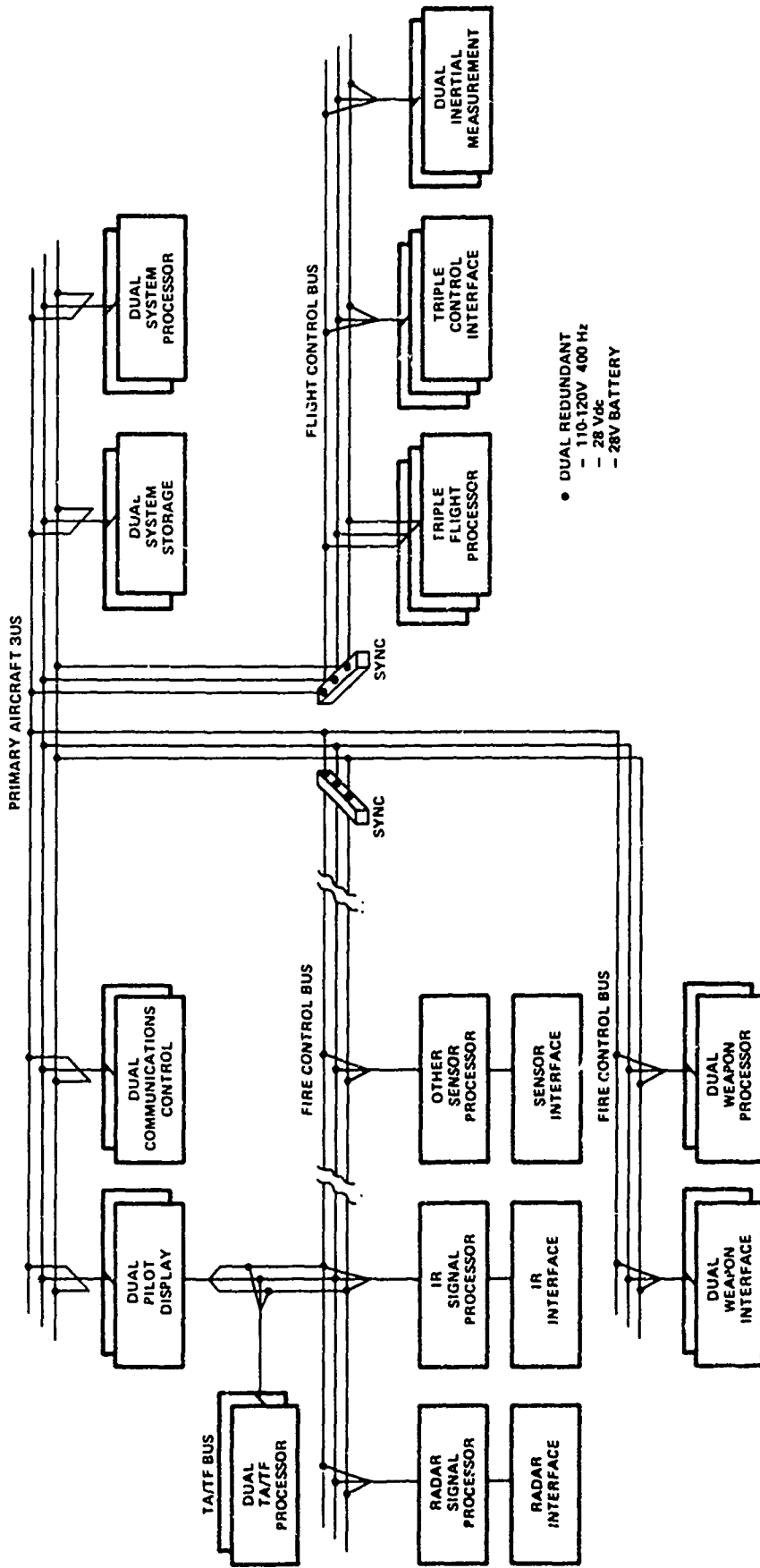


Figure 3

MARTIN MARIETTA

of sensor information and application to terrain avoidance/terrain following (TA/TF) and generation of weapon preset or control signals. A lower speed bus, perhaps equivalent to the 1553 architecture presently used, integrates the primary aircraft bus and the fire control bus.

The flight control bus also has multi-speed potential. The medium speed characteristic of the autopilot which stabilizes the aircraft in its environment interfaces with the lower speed primary bus to accommodate pilot interface command and display. These command signals serve as inputs to flight control guidance functions, and displays are generated from both autopilot attitude data and navigation position/velocity/heading data.

An architecture of the type discussed and illustrated here offers maximum potential for system fault tolerance and higher reliability without the complication of full federated architecture design. The environment of changing state-of-the-art sensors, weapons, and operational methods can be accommodated by the reconfiguration capability.

FAULT TOLERANCE ATTRIBUTES OF INTEGRATED AVIONICS

A digital system design that includes redundancy, flexibility, and growth capability can also be used to increase fault/battle damage tolerance and as a flight line diagnostic fault detection and test tool.

The sets of processing, bussing, and interface functions from a typical flight/fire control integrated avionic system is tabulated in Table 4 and categorized according to speed. The use of a simple, table driven executive with system status monitoring input and access to main system storage would allow system reconfiguration of processing, bus, and interface elements where faults or damage have impacted the baseline, and would also allow use of diagnostic routines resident in main storage. Alteration of tables would extend the system capability to acceptance of partial or complete new subsystems.

As an example of the concept, consider that the system has three flight control processors dedicated to that function, while five are required with three in agreement for man-rated fly-by-wire system operation. The redundant flight environment, radar signal, or IR signal processors can all fill the need for additional flight control processing, since only one of the three systems should be operating at the same time in normal flight. If a fault occurred, or battle damage impacted use of a baseline of three flight control computers, a flight environment computer, and an IR signal processor computer, the main system control could reload and assign any of the other two dual redundant computers to the task.

Given the requirement to monitor system status for dynamic reallocation of sensor, bus, and processing resources, plus the sophisticated controls and displays required for operation of modern aircraft, only a small extension of test stimuli-response philosophy will give self-diagnostic capability. A trained ground crew technician could operate from the pilot's position using ground power to call diagnostic routines from main system storage for test to a line replaceable unit. Where a fault unit was detected and a replacement

Lower Speed	Medium Speed	Higher Speed
Main System Storage	Flight Control	Flight Environment
Main System Control	Flight Control	Flight Environment
Man-Machine/Comm.	Flight Control	Radar Signal
Man-Machine/Comm.		I/R Signal
Weapon Control		Other Sensors
Weapon Control		

Processing Capability Inventory

Lower Speed	Medium Speed	Higher Speed
Primary Aircraft	TA/TF Process	Radar Signal
Primary Aircraft	TA/TF Process	I/R Signal
Primary Aircraft	TA/TF Process	Other Sensor
	Fire Control	
	Fire Control	
	Fire Control	
	Flight Control	
	Flight Control	
	Flight Control	

Bus Capability Inventory

Lower Speed	Medium Speed	Higher Speed
Pilot Control	Weapon Control	Pilot Display
Pilot Control	Weapon Control	Pilot Display
Flight Operation	Inertial Measurement	Communication (Radio)
Flight Operation	Inertial Measurement	Communication (Radio)
	Flight Control Surfaces	
	Flight Control Surfaces	

Interface Inventory

Fault Tolerant Adaptability of Integrated Avionics
Table 4

was unavailable, the system could be reconfigured (i.e., radar sensor and IR signal processor) to complete the majority of the assigned missions in a higher stress wartime environment.

Aside from the physical interface problems associated with integration of new sensor and weapon systems on an aircraft, the bussing and processing functions required would be available by use of the table driven, reconfiguration approach. The initial design for growth and flexibility insures that the integrated flight/fire control avionic architecture will meet the challenge.

STANDARDS APPLICABLE TO THE INTEGRATED AVIONIC DIGITAL SYSTEM

The applicable documents section of a recent Air Force RFP⁵ contained 129 references, of which 48 were standards. The digital design and development team were required to be familiar with 45 of the references, of which 24 were standards. A further analysis of familiarity by discipline gave the following areas of responsibilities for familiarity: (1) Electronics - 35, (2) Software - 30, (3) Requirements and Integration - 24, and Packaging - 9. This included five quadruplicates, 13 triplicates, and 12 duplicates of the same references. The learning curve for total team cognizance of required references for effective design is then: (1) Electronics - 10, (2) Software - 15, (3) Requirements and Integration - 21, and (4) Packaging - 36. The cost of the effort required for these personnel to become familiar with this spectrum of documentations clearly points out the need for a dedicated digital design team that is well trained in the realities of detailed practices within their discipline, but prepared to participate in cross discipline design decisions.

The requirements and standards that have been somewhat neglected previously, but need special consideration when sustainability and reliability are considered are listed in Table 5.

The integrated avionic digital system design is incomplete if the inherent system capability for reconfiguration and self test is not exploited to improve reliability and simplify maintainability. This consideration must encompass integration, factory, depot, field, squadron, and flight line operations. A philosophy that does not look at the total requirement for testability to meet the total spectrum of these requirements will again lead to system life cycle cost inefficiency. The approach must also consider the available equipment and personnel at each level that will contribute to the system reliability and survivability for increased sortie count.

PROCUREMENT, LIFE CYCLE COSTS AND STANDARDS

The use of standards during the procurement cycle has had a substantial impact on costs. The imposition of MIL STD's 483, 490, and 1521A alone has improved program management techniques and provided the customer with sufficient phased visibility into the procurement cycle to allow a meaningful input into contractor development procedures and techniques. The controls, documentation, and reviews support a customer/contractor team rather than adversary relationship.

MIL-S-8512D	Support Equipment, Aeronautical, Special, General Specification for the Design of, 14 Mar 80
MIL-T-28800B(1)	Test Equipment for Use with Electrical and Electronic Equipment, General Specifications, 20 Jul 77
MIL-H-46855B	Human Engineering Requirements for Military Systems, Equipment and Facilities, 31 Jan 79
MIL-STD-470	Maintainability Program Requirements (for systems and equipment), 21 Mar 66
MIL-STD-781C Notice 1 20 Mar 81	Reliability Tests Exponential Distribution, 21 Oct 77
MIL-STD-785B	Reliability Program for Systems and Equipment Development and Production, 15 Sep 80
MIL-STD-810C Notice 1 7 Apr 81	Environmental Test Methods, 10 Mar 75
MIL-STD-1472C	Human Engineering Design Criteria for Military System, Equipment and Facilities, 2 May 81
MIL-HDBK-217D	Reliability Predictions of Electronic Equipment, 15 Jan 82
MIL-HDBK-472	Maintainability Prediction, 24 May 66
AFR 66-1, Vol 3	Maintenance Management-Squadron Maintenance, 2 Jan 80
MATE	Modular Automatic Test Equipment Guides, 15 Jun 81

Applicable Documents for Special Consideration
Table 5

The literature resonates with the merits of higher order language and large scale integrated circuitry as the solution to decreased maintenance cycle costs. Imposition of MIL STD's 785B and 1589B has allowed much improvement of the previous procurement to maintenance cost ratios of 9/91 percent for manpower intensive, low electronic density missile systems and 30/70 percent for computer programs.

To obtain insight into the impact of the standards for a typical example, this excerpt on MTBF computation⁶ for a computer program will serve as a baseline. The 12000-line avionic program was assumed to be procured with either 0.5, 1.0, or 2.0 percent of the lines of code (instruction error confidence levels) in error. The procurement costs were defined as \$2,400,000 (25 lines of code per workmonth), \$1,200,000 (50 lines of code per workmonth), and \$800,000 (75 lines of code per workmonth) for the respective error confidence levels. The program scenario defined it as having 10 years of useful life and being installed in 180 processors distributed in the same number of aircraft among 10 tactical squadrons. In a squadron, 10 aircraft make two flights of two and one-half hours duration per day. The total operating time of the fielded system may then be computed by:

$$1.825 \times 10^6 \text{ hours} = 100 \text{ aircraft} \times 5 \text{ hours/aircraft-day} \times 3650 \text{ days.}$$

Using the definition for failure rate, and its inverse, mean time between failure (MTBF), the following equation was applicable:

$$\text{Errors/100,000 hours} = (\text{error confidence level}) \times (\text{program size}) / (\text{fielded system operating time})$$

The 0.5, 1.0, and 2.0 percent error confidence levels gave 3.29, 6.58, and 13.20 program errors per 100,000 hours, respectively. Using Monte Carlo techniques with a normal distribution for error difficulty (two, four and six workmonths to fix one, two, and three sigma errors), and an exponential distribution for time of error occurrence the following table was developed:

Typical Average Maintenance Workloading

Error Rate	MTBF	People	Cost
0.5	36395	1.5	\$ 900,000
1.0	15923	2.4	\$1,440,000
2.0	7575	4.8	\$2,880,000

The part of the life cycle spent on maintenance for the three error rates are seen to be 27, 55, and 78 percent, respectively. A plot of procurement and maintenance curves, as well as a plot of their composite, is included in Figure 4. This composite conforms to the classic life cycle curve with its saddle point of cost effectiveness. Even when the saddle point of the life cycle curve is used, three people are required for software maintenance.

The impact of increased attention to reliability and maintenance standards, as well as advanced software engineering techniques during development will extend the procurement curve at the higher end. Maintenance effects resulting from use of the standards has been magnified, by

TYPICAL EXAMPLE OF LIFE CYCLE CURVES

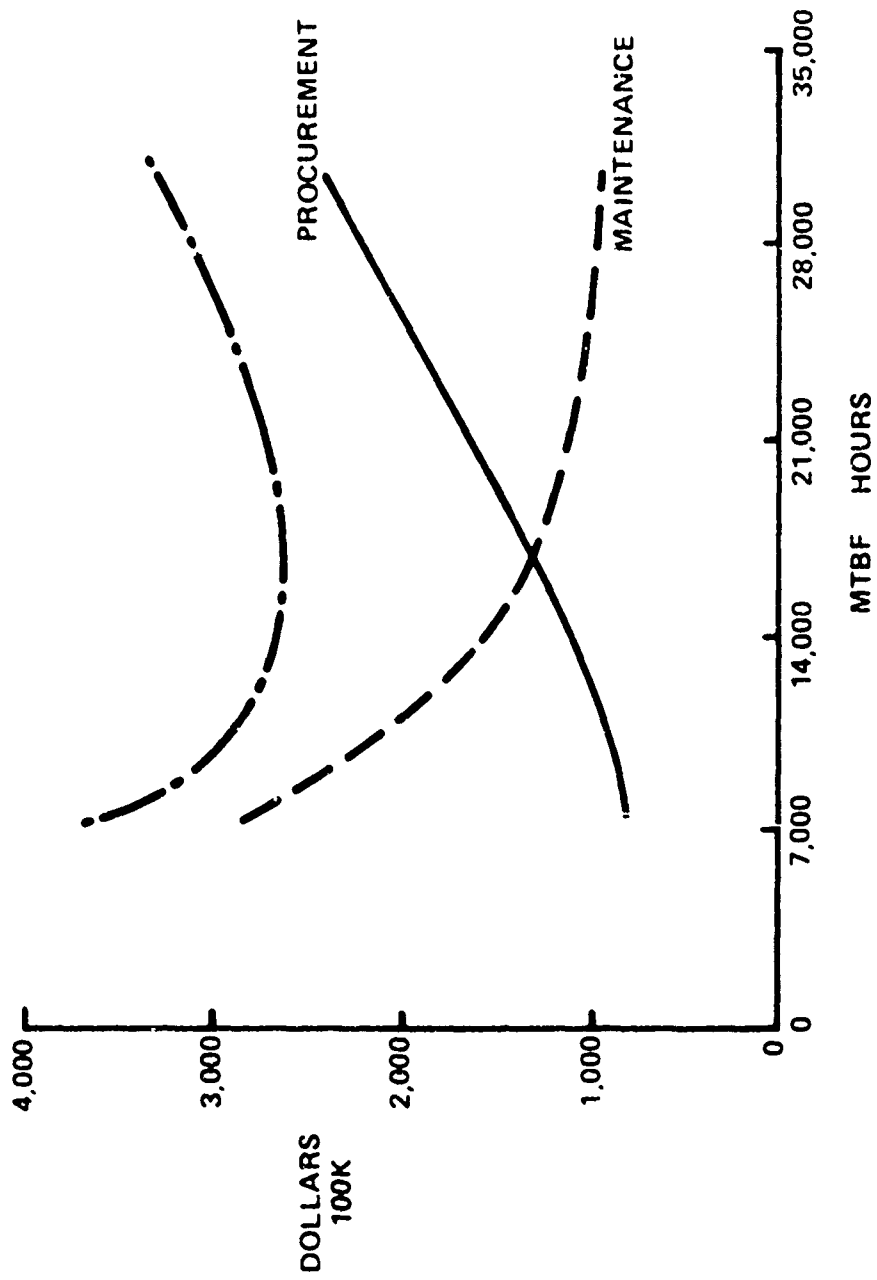


Figure 4

MARTIN MARIETTA

both further decrease of error rate and an additional decrease of error difficulty (or time to repair). The personnel required to staff the maintenance activity will decrease and the maintenance and composite cost curves will be lowered. Although the saddle point in the composite curve will shift to a lower MTBF, if higher MTBF's are required, they can be obtained at lower total cost.

EFFICIENT UTILIZATION OF STANDARDS FOR INTEGRATED AVIONICS

The digital system design and development group manager for an avionic program is faced with the familiar schedule and budget constraints, but also with an available staff unversed in the digital system design approach. If the staff are experienced in their respective electronics, software, requirements and packaging disciplines, the manager should be able to create a team through use of some innovative motivational and educational techniques. The motivational task should be simple, if the personnel have worked in the avionic area previously and if they are aware of the system challenges inherent in the level of automation required to decrease pilot workloads in today's high speed, low altitude, mission profile environment. The additional step that is necessary is stress of survivability, sustainability, reliability, and maintainability.

The educational process must usually be accomplished while conceptual design and system performance requirements work is in progress. Although most contractors cannot afford the training effort involved in giving their entire engineering staff detailed familiarization with the document set in Table 6, the personnel available to the integrated avionic group should be well schooled in their own department's applicable documents. The cross training process is not, as one reliability engineer stated, "a one to three year training course in reliability prediction techniques, but a one month part time familiarization exposure." The time is available for this effort because the personnel will use the tried techniques of strawman system generation and interdisciplinary negotiation.

The initial training will be through close exposure in the negotiation processes. The electronics, software, and requirements people will be forced to understand the documents listed under the Electronics and Software, Support, and Procedures columns in the table if the manager requires backup documentation for the trades. This process will also serve as the basis for post trade study documentation to be placed in the project data banks.

Briefings from the specialty skills of human factors, reliability, and maintainability will round out the educational process. The cost effective learning mechanism is to invite these specialists to the group's concept or design walk-throughs, and then request a critical briefing on the concept and its impact on their specialty. This series will usually require four hours of personnel contact time, but often results in eight to ten hours of individual effort per affected person to correct or refute design decisions in question.

These techniques have been used to mold a digital system architecture group in less than three months. The extension to the avionic digital system architecture group should add only one additional month. With the permeating need for this new approach to avionic digital architecture, management must be prepared to accept the challenge.

PRODUCT DEVELOPMENT

ELECTRONICS AND SOFTWARE

MIL E 5400T	Electronic Equipment
MIL E 6051D(1)	EM Compatibility
MIL I 23659C	Initiators
MIL P 27733A	Equipment Install.
MIL STD 499D	RF Spectrum
MIL STD 454G	Electronic Equipment
MIL STD 461A	EM Interference
MIL STD 462	EM Interference
MIL STD 810C	Environmental Test
MIL STD 1589B	JOVIAL Language
MIL STD 1629	Failure Mode Analysis
MTI STD 1670	Air Launch Environment
MIL STD 1750A	Computer Instruction Set
MIL STD 1763	AC/Store Certification
MIL STD 1815	Ada Language
AFSCP (AFLCP)	Design to Cost
ANSI x 3.9-1978	FORTRAN 77
NACSGM 5201	TEMPEST Guide
NACSIM 5100A	Laboratory Tests

SUPPORT

MIL S 8512D	Support Equipment
MIL T 28800B	Test Equipment
IEEE STD 716	ATLAS Language
IRIG 106-77	Instrumentation
MATE	ATE Guide

PROCEDURES

MIL Q 98S8A	Qual Program
MIL S 52779A	Software QA
MIL STD 480A	Config. Control
MIL STD 483	Config. Mgmt.
MIL STD 490	Spec. Practices
MIL STD 499A	Engr. Mgmt.
MIL STD 1519	Test Req. Doc.
MIL STD 1521A	Tech. Review
MIL STD 1535A	Supplier QA

PRODUCT MAINTENANCE

HUMAN FACTORS AND SAFETY

MIL-H-46855B	Human Engineering
MIL STD 882A	System Safety
MIL STD 1472	Human Engineering
AFSC DH 1-6	System Safety

RELIABILITY/MAINTAINABILITY

MIL STD 470	Maint. Program
MIL STD 471A	Maint. Demo.
MIL STD 781C	Rel. Test
MIL STD 785B	Rel. Program
MIL HDBK 217D	Rel. Prediction
MIL HDBK 472	Maint. Prediction
AFR 66-1, Vol. 1	Maint. Mgmt.
AFR 66-1, Vol. 3	Sqd. Maint. Mgmt.

Applicable Integrated Digital Avionic Documents
Table 6

SUMMARY

The need for flexible, responsive weapons to equip a sustainable force must be met in a cost effective manner to avoid stressing available national resources. The use of digital systems engineering techniques by a multi-disciplined team whose members are aware of the standards and design guidelines influencing the other team personnel can do much to keep system procurement costs lower. This can be accomplished by decreasing the redundancy in hardware, interfaces, and software development that often occurs if the separate disciplines work independently. In addition, the team approach will allow more cost effective decisions to be made in areas of post procurement maintainability and reliability.

The manager of this digital system design team can use the interdisciplinary trade and negotiation conceptual definition phase to have each expert cross-train others in requirements related to his discipline. Forcing referenced documentation to be generated for each trade and solution will force member training as a credential for team participation. Some more formal briefings may be necessary as maintainability and reliability specialists interact with the team. The result of the team approach with member cross-training in others' standards will be development of weapon systems that will meet the severe challenges imposed on them in today's environment.

BIBLIOGRAPHY

1. Russell, David M., "NORAD Adds Radar, Optics to Increase Space Defense", Defense Electronics, July 1982 Vol. 14 No. 7, (EW Communications, Inc., P.O. Box 50249, Palo Alto, CA 94303-9983) pp 82.
2. Russell, David M., "How EXOCRT Sank the HMS Sheffield," Defense Electronics, July 1982 Vol. 14 No. 7 (EW Communications, Inc., P.O. Box 50249, Palo Alto, CA 94303-9983) pp 43-47.
3. _____ "Our Protected Response Capability," Air Reservist, July/August 1982, Vol. XXXIV Number 6, (Superintendent of Documents, Government Printing Office, Washington, D.C. 20402) pp 3.
4. Griffin, Edward L. and Johnson, Floyd, "Missile Systems Division Software Compendium, Monthly Review of Significant Technology Topics, Volume 2, Number 1, May 1982", Martin Marietta Corporation, Missile Systems Division, Orlando Aerospace, P.O. Box 5837, Orlando, Florida 32855.
5. _____ "Conventional Standoff Weapon, Statement of Work, 10 March 1982", Guided Weapons System Program Office, Armament Division, Eglin Air Force Base, FL 32542, pp 185-192.
6. _____ Griffin, Edward L., LTC, "Considerations in Standardization of Computer Languages," Project 77-18, ASD/XOR, Wright-Patterson AFB, OH 45433.

BIOGRAPHY - EDWARD L. GRIFFIN

Mr. Griffin is the Missile Systems Division Software Manager with Martin Marietta Aerospace, Orlando, Fla. He has worked in and managed software development and test for the past eight years with previous experience in systems engineering and computer hardware applications. He serves as a Colonel in the Air Force Reserve, working computer hardware and software development with Wright-Patterson AFB.



ACHIEVING THE BENEFITS OF MODULAR AVIONICS DESIGN

Stephen W. Behnen
Fred M. Lightfoot
Peter R. Metz

Boeing Military Airplane Company
Advanced Avionics Systems
Seattle, Washington 98124
(206) 655-9722

Stephen W. Behnen

Since joining Boeing in 1978, Mr. Behnen has been active in the area of digital avionics system integration, leading to his present position as technical manager for the Integrated Systems activity. Prior to 1978, he worked on satellite systems and on combat systems for new Navy ships. He received an M.S. degree in Physics from the University of California in 1974.

Fred M. Lightfoot

Fred M. Lightfoot has been associated with The Boeing Company for the past 25 years. His present position is Manager of the Advanced Avionics Systems organization, Boeing Military Airplane Company. During his employment at Boeing, Mr. Lightfoot has worked in engineering and management positions in avionics integration and communications, command and control technology, as well as on major projects. He obtained an M.S. degree in Electrical Engineering at the University of Texas in 1958 and is a Professional Engineer, State of Washington.

Peter R. Metz

Peter R. Metz has been with the Boeing Company since 1976 in the field of advanced avionics development. Prior to that time, he taught and did research in communications and optical data processing at the University of Washington and also at two universities in Brazil. He obtained a Ph.D. in Electrical Engineering from the University of Washington in 1965.

ABSTRACT

New system development programs are adopting the principles of modular design to reduce the number of unique parts, increase capability, improve fault tolerance, lower costs, and encourage transition to new technologies. Programs such as Integrated Communication, Navigation, Identification Avionics (ICNIA) are proving the value of this approach. Even greater benefits will be obtained from modular design, however, when the use of common modules spreads across, and into, dissimilar subsystems. The creation and adoption of new military standards, which will complement existing standards, are needed to encourage widespread use of compatible modular avionics. Examples are given, as well as suggestions as to the most

efficient way of circumventing inherent industry reluctance to adopt national standards.

I. INTRODUCTION

A. FORCE MULTIPLICATION

An important problem that must be addressed by the aerospace industry in conjunction with the Air Force, is how to develop avionics systems that will increase the effectiveness of a given pool of aircraft so that it can perform a mission that currently requires larger numbers of aircraft. The result is commonly referred to as "force multiplication". There are at least four avionics system goals that will result in improved aircraft effectiveness. These goals are: (1) improved reliability; (2) improved maintainability; (3) increased flexibility to support different missions; and (4) increased aircraft capability under day/night all-weather conditions in advanced threat environments. We will address each of these in turn, and in so doing, indicate how each can be attained as a result of modular design (where modular design, or "modularization", is defined as the organization of unique hardware line-replaceable-units (LRUs) and system software structures, into families of common or canonical modules).

1. RELIABILITY

Reliability is an aspect of availability that is particularly important when aircrafts are deployed at forward bases with minimal support equipment. Study results show that avionics system reliability (mean-time-between-failures) is directly related to the number of LRUs in an avionics suite (Reference 1). Reducing the LRU count will reduce the failure rate. With proper modular design, LRU count can be reduced because of the capability of reallocating standard modules from a common pool as the need arises. In complex systems, a common pool of modular processors could be shared among different functions throughout a mission, or even time-shared among several functions simultaneously. Alternatively, new functions could be synthesized from a given modular system by software changes alone. For example, existing modules might be reconfigured to synthesize a SAR in a frequency band other than originally intended.

Taken together, the synthesis and sharing abilities should reduce the total LRU count in an avionics system with pooled modules by a factor as large as 3 to 10. Overall system reliability will then increase accordingly.

Another factor which would increase reliability is improved cooling design due to the standardization of module construction. Reduced thermal cycling is a well-known method for increasing MTBF.

2. MAINTAINABILITY

A second aspect of availability is maintainability. Improvements in maintainability can also be expected when the total and number of different types of LRUs are reduced in the manner already described. A reduction in LRUs permits a reduction in spares, reduced training time, and reductions in mean-time-to-repair (MTTR). MTTR improvement is possible through the

simplified fault detection and isolation achieved by having fewer unique elements, as well as improved familiarity with individual modules.

3. FLEXIBILITY

Another potential benefit expected from modularity is that of being able to reconfigure an avionics system in response to battlefield changes in threat description, real-time target list changes, and the ability to accommodate widely varying and changing weapons and expendables. Forward area reconfiguration permits the aircraft to achieve true multi-mission capability by adapting its capabilities to respond to the current battlefield environment, as that environment is changing. Modular design can permit mission-to-mission changes in the characteristics and capabilities of the avionics system. Before the next mission is flown, flight safety requirements will demand that the reconfigured system be tested to insure that flight safety is not compromised. This testing must be possible with little or no support equipment required at the forward area. Extensive use of modular design permits a distributed approach to architecture design that will allow rapid, on-board verification that the reconfigured system will not endanger safety of flight. The resulting flexibility will allow an aircraft to fly more than one kind of mission in a single day.

4. CAPABILITY

An approach to increase overall effectiveness by raising survivability levels, and improving and extending weapon delivery capability, is to employ sensor blending and multiple use of common sources of information. Sensor blending of information from all on-board and remote sources, however, may temporarily saturate available dedicated computational resources. If the means exist to reallocate standard processing modules from a common pool during peak demands, increased performance may be achieved over that obtained from dedicated processors. This is very similar to the system synthesis process discussed earlier. As an example, target classification processing may require billions of signal processing operations per second. The total time interval per mission, however, may be only minutes. A system design that provides the necessary processing power in dedicated target classification processors may be too expensive to implement since the processing is extensive and the need occurs during only a small part of the mission. Implementation is much more likely if the temporary need for additional processing power can be satisfied by a common pool of modular processors.

B. ENABLING TECHNOLOGIES

Significant progress has been achieved in the development of capabilities and technologies important to mission avionics which, in conjunction with efficient crew system design, should enable the force multiplication process even further. Some examples follow.

- a. Semi-automatic target acquisition. Programs include Advanced Target Acquisition System (ATAS), Multi-Functional Infrared Coherent Optical Sensor (MICOS), Covert Strike (an advanced radar program), and Forward Looking Active Classification Technology (FLACT).

- b. Survivable low-level penetration. Programs include Purple Haze (an advanced terrain masking display concept), Advanced Digital Avionics Map (ADAM), and Sensor Blending for Terrain and Obstacle Avoidance.
- c. Crew systems. Developing technologies include pictorial format displays, automated display evaluation, and integrated flight/propulsion controls.
- d. Very large scale and very high speed integrated processors (VLSI/VHSIC).

The latter development will provide the processing power to realize the potential represented by the other programs through data fusion at reasonable cost.

II. ADVANCED AVIONICS MODULAR ARCHITECTURE

A. FEATURES

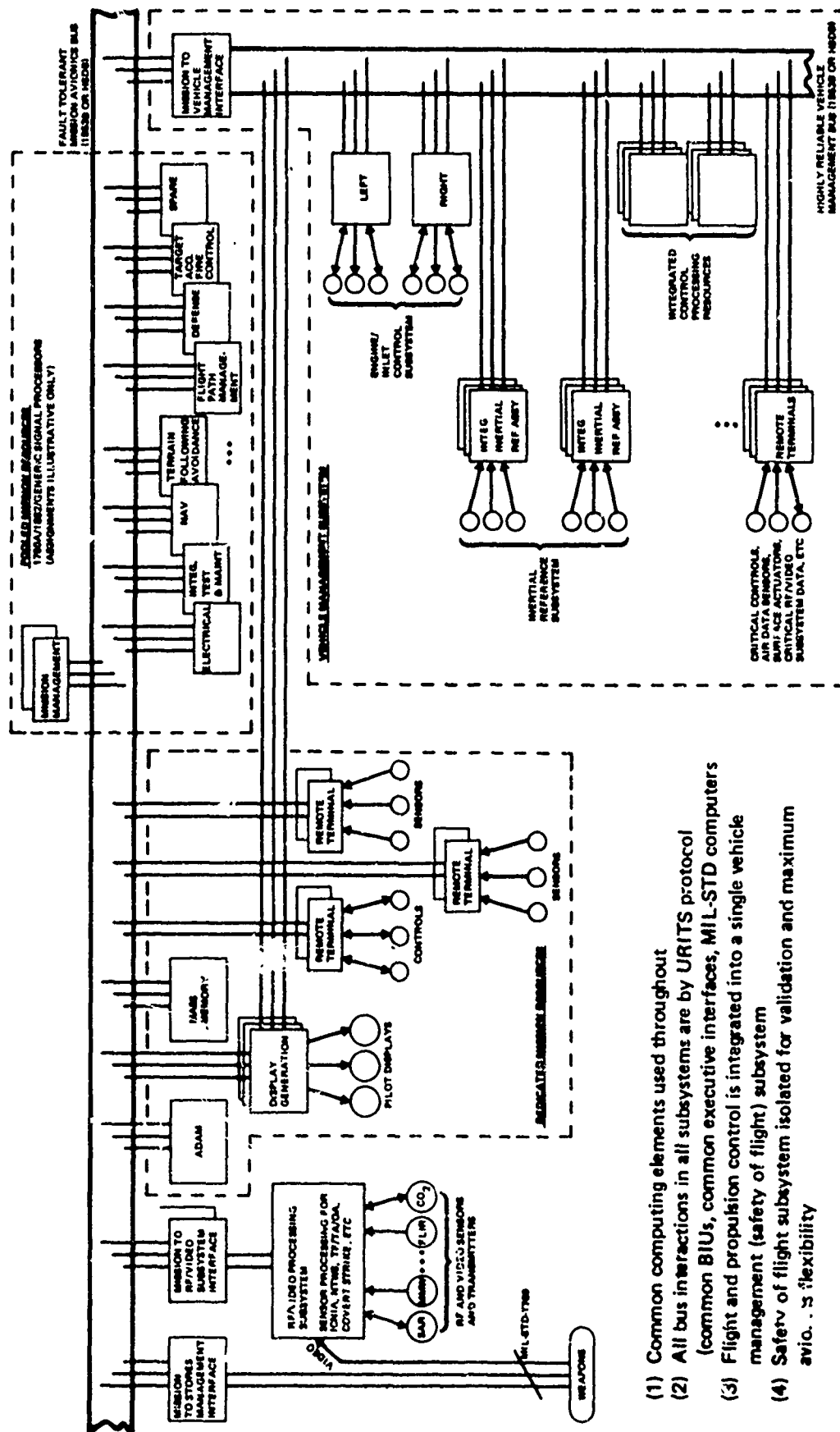
An architecture which embodies modular design principles leading to force multiplication is shown in Figure 1. The emphasis is on commonality of elements while providing for reliable integration of very large and complex subsystems using an ultra-reliable information transfer system (URITS). URITS is a fault-tolerant arrangement of MIL-STD buses, bus interface units, and executive software. Multiple modular units are a key part. Principle features of this architecture are:

- a. High system availability by employment of resource pooling, analytic redundancy; advanced fault detection/isolation through integrated test and maintenance design; automatic reconfiguration; flight critical/mission critical partitioning; simplified logistics by common module design.
- b. Flexibility through the use of generic signal, vector and data processors and high data rate buses; rapid system design update through functional partitioning; modular verification and validation through use of software development standards and a common executive family.
- c. Modular synergism which permits the flexibility to create new functional operations with little incremental investment - "The whole is greater than the sum of its parts".

B. RF/VIDEO SUBSYSTEM

The RF/video signal processing subsystem is a highly modular portion of the suggested architecture. The subsystem is a high data flow design that extends the AFWAL ICNIA approach by including additional functions. Redundant, time division multiplexed, VLSI general purpose filtering and processing modules provide a system with high levels of performance, availability, real-time reconfiguration, and sensor blending/fusion. Figure 2 shows this subsystem.

The RF/video subsystem functions considered include the communications, navigation, and identification functions of ICNIA, threat warning functions of NTWS, radar capabilities of COVERT STRIKE, laser radar/designator,



- (1) Common computing elements used throughout
- (2) All bus interactions in all subsystems are by URITS protocol (common BIUs, common executive interfaces, MIL-STD computers)
- (3) Flight and propulsion control is integrated into a single vehicle management (safety of flight) subsystem
- (4) Safety of flight subsystem isolated for validation and maximum avio. flexibility

Figure 1. Advanced Avionics System Architecture

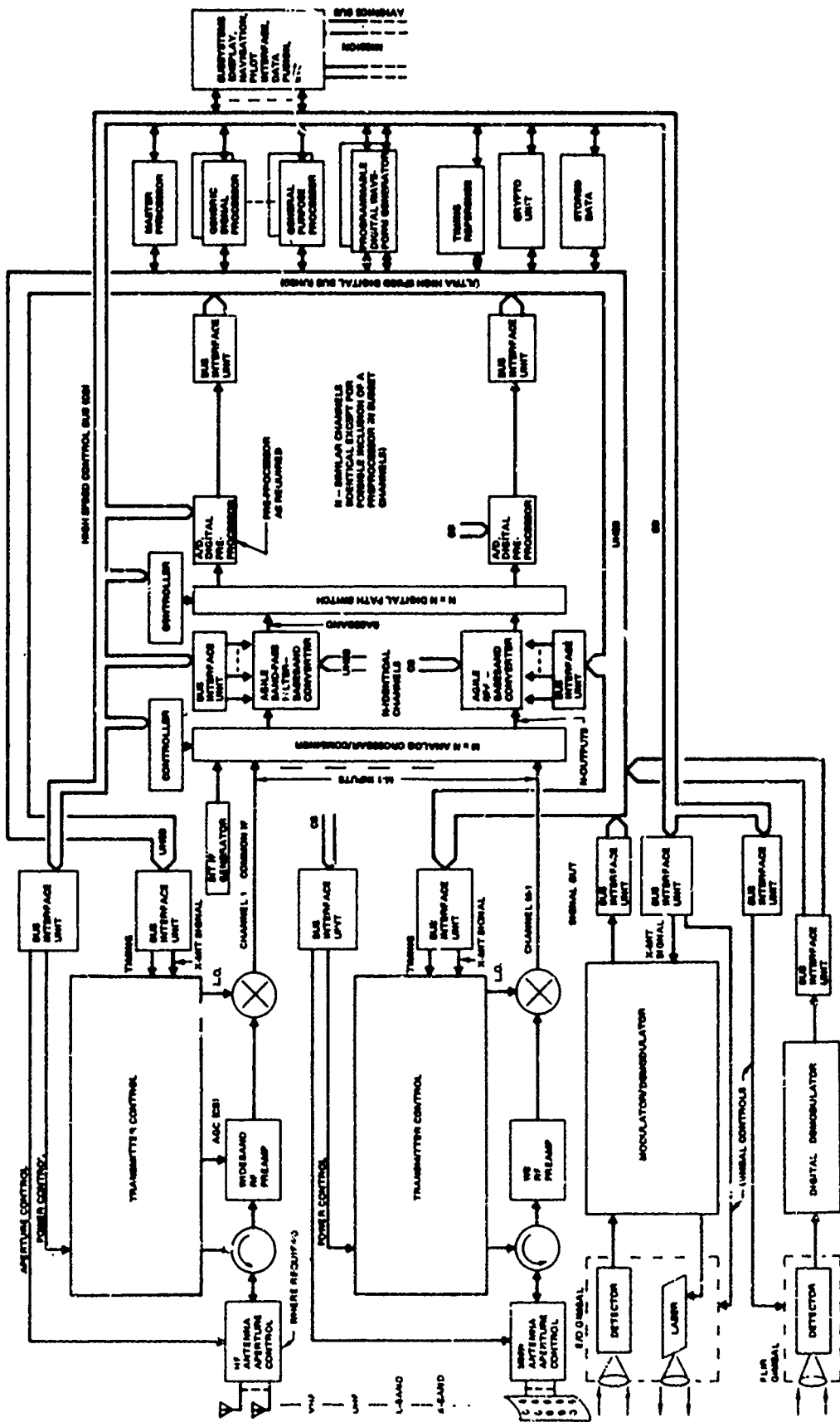


Figure 2. RF/Video Subsystem

non-optical radar, IR/EO warning and countermeasures, and RF jamming capabilities. The subsystem architecture maximizes the use of resource sharing and modularity. Included is an imbedded system of fault detection and isolation, as well as duplication of critical hardware, thereby permitting substitution of on-line modules for failed units wherever possible. Extrapolating from ICNIA program estimates, this approach can be expected to reduce weight, volume, and cost by at least 50%, 60%, and 30% respectively, compared to what would be required to achieve each of the baseline RF functions in a conventional implementation. Even greater reductions can be expected here since the higher degree of resource pooling, as compared to ICNIA, allows realization of new functions beyond those of the intended baseline.

The operation of the RF/video subsystem can be illustrated by considering the signal flow in the receiving mode. With the exception of IR/EO, signals from antenna elements/arrays (in some cases with controlled apertures) are amplified by wideband amplifiers and then subjected to bandpass filtering and baseband conversion. The latter operations are carried out by a bank of identical channels with specific channel selection via an analog crossbar/combiner under master processor control.

Several choices for realization of this portion of the circuit are (or soon will be) available for consideration: programmable CCD finite impulse response transversal filters; programmable SAW correlators; and integrated optical signal processors.

The output of each transversal filter channel consists of a baseband signal which has been digitized and is ready for further processing. To provide for time sharing of signal processors among the many RF channels, digital data from each individual channel is then placed on the ultra-high-speed data bus (UHSB).

The collective data rate on the UHSB will, of course, vary throughout any particular mission. The maximum rate is estimated to be of the order of several hundred Mbps. Clearly, such a rate calls for an extraordinarily high speed bus and special attention to various optical and RF coaxial techniques would be required for realization.

Baseband data on the UHSB is accessible to the processors, all of which are under the control of a master processor. Processed data, as well as additional information required for certain calculations, is then available via connections to the mission avionics bus.

Additional modules attached to the UHSB include a timing generator, and programmable waveform generators. System timing is broadcast over the UHSB. Waveforms originating with the programmable waveform generators are available for transmission by the various transmitters, as well as reference signals required by SAW correlators if they are used for bandpass filtering, and baseband conversion.

Fault-tolerant design features which are incorporated so as to fully exploit possible improvements in availability afforded by the extensive modularity include the following:

- a. All URITS aspects, including fault-tolerant standard buses, bus interface units (BIUs), and executive interfaces.
- b. General purpose and signal processing resources which employ a redundancy management technique consistent with the system and vehicle management subsystem designs.
- c. Crossbar switches and controllers redundant to levels required to meet aggregate reliability needs.
- d. An IF test generator which continually tests the agile bandpass filter and pre-processing subsections. The pulse nature of most signals allows these tests to be performed in totally transparent fashion.
- e. Physical dispersion so as to meet survivability needs.

With the extensive array of modules contained within the RF/Video subsystem, it is possible to realize system capabilities not included in the original baseline with minor additional investment (software). For example, with the existing SAR processing capability, a SAR system at other than the intended wavelength could be created. Alternatively, a transmitter intended for use in a radar mode only, could be utilized in a communication system. Several other applications are now under evaluation.

III. IMPLEMENTING STANDARDS FOR MODULAR AVIONICS DESIGN

The previous section of this paper provided examples that illustrate the benefits inherent in a modular design approach to avionics systems. These benefits are fully realizable only when a large library of compatible modules is available for general use by system designers. Two aspects of standardization must be addressed: the development of standard modules, and the development of standard interfaces and packaging to link existing and planned modules into new systems.

A. INTERSYSTEM AND INTRASYSTEM CONSIDERATIONS

Standard modules and interfaces can be treated at two levels - intersystem and intrasystem. The initial application of standard interfaces is generally most beneficial in the areas of intersystem communication and control. Intrasystem design is often too specialized to immediately benefit from standards enforcement, especially since intrasystem activities are often controlled by a single design group. Intersystem integration realizes more immediate benefits from standardization because the increasing complexity of modern avionics suites often leads to the participation of many different companies in the development of individual subsystems that will eventually be joined to form the complete avionics package. Standards provide a mechanism for each group to work independently and yet be confident in the ability to integrate the final product. Properly designed standards define interfaces that can link common processing modules through standardized control procedures. Once the necessary intersystem standards are adopted and are generally accepted for use in system design, the introduction of the standards into the intrasystem domain will tend to occur without additional effort. As an example, the emergence of VHSIC processing technology and high speed buses are allowing the common treatment of inter-

and intra-system modules and interfaces in systems that apply these technologies. VHSIC chips can be used to package powerful general purpose computers or even signal processors on one or two cards; the resulting processing elements can then be treated as individual subsystems or can be fully embedded as a part of a larger system. High speed buses allow massive transfers of data between processing elements, permitting physical separation of modules that otherwise would be linked through the backplane of a single subsystem.

While it may not be cost effective to develop standards solely and specifically for intrasystem use, it is most certainly cost effective to extend an existing standard into the intrasystem domain. The value of standardization can be enhanced, therefore, by developing standards according to a plan that calls for potential extension of its applicability and scope.

An advanced avionics system will be striving to fully incorporate emerging technologies such as VHSIC and high speed buses. An advanced system should benefit greatly from the development and adoption of standards to control the application of these technologies. Several of these benefits have already been demonstrated in the previous section of this paper. The issue that remains to be addressed is how to identify, develop, and enforce the use of standards appropriate to modular design.

B. STANDARDIZATION OPTIONS

Standards are needed for both hardware and software. Processor standardization can be considered at six different levels that are interrelated as shown in Figure 3. An instruction set standard should be considered the minimum essential step in processor standardization. A review of the six processor standardization options was performed as part of a Boeing contract to study microprocessor applications to airframe related avionics (References 2, 3). The most suitable option was found to be a standard device family. This approach provided the most flexibility to the system designer while achieving the major goals in the standardization process. With a device family, all subsystems can adhere to the standard, and yet use only the specific elements that are needed to satisfy the (perhaps) limited needs of that subsystem. At the same time, the standard device modules can be combined to create data processors, generic signal processors, or other processing elements that can then be conveniently coupled in a complex avionics system if communication standards and packaging standards are in place.

The existing communications standard for avionics systems, MIL-STD-1553B, will be joined by additional high speed bus standards. The SAE-A2F subcommittee on high speed buses is expected to introduce a standard for a 20 MHz fiber optic bus. In addition, a previous example in this paper points to the need for a standard fiber optic bus that exceeds 100 MHz.

The processing and communication standards should be complemented by a packaging standard that covers not only packaging for individual modules, but also packaging of larger elements that are built up from those individual modules. This standard must be broader in scope and more flexible than the draft MIL-STD-XXX installation standard currently in work.

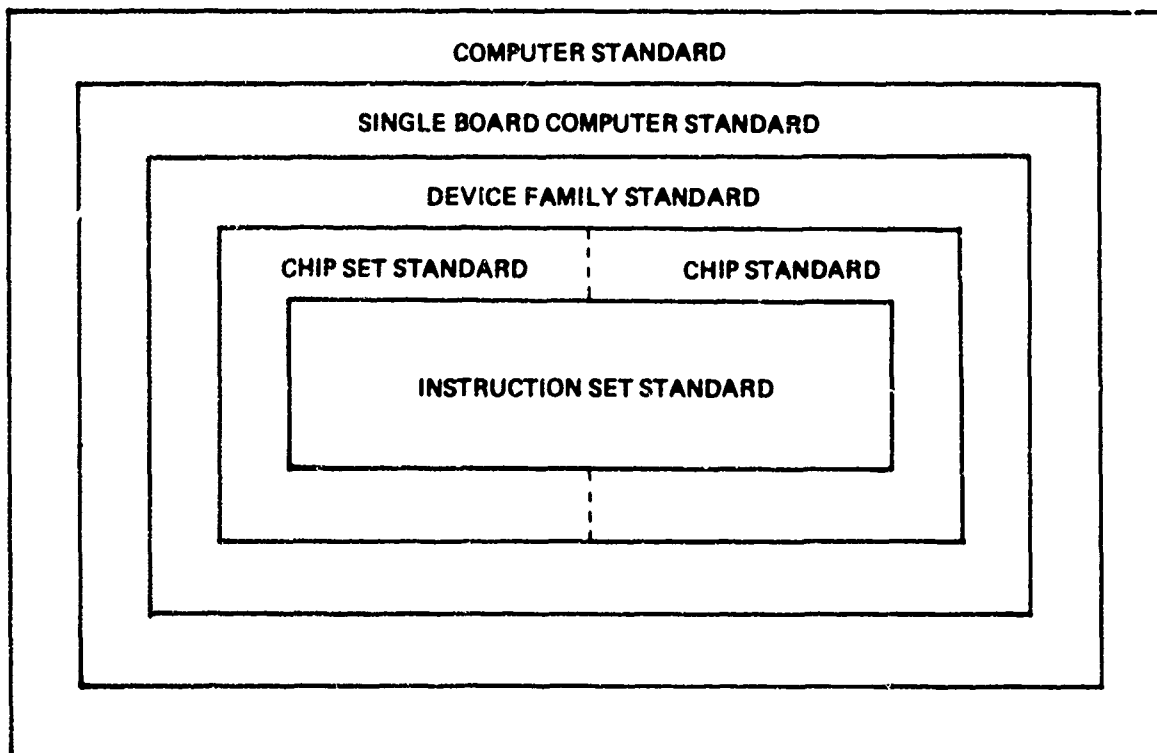


Figure 3. Hierarchy of Processor Standardization Approaches

Intersystem needs are generally served by developing modules to implement functions such as data processors, vector processors, and high speed bus interface units. Once these capabilities exist, the standardization effort can focus on modules useful in intrasystem design. These might include a programmable transversal filter, SAW correlator, programmable waveform generator, and a timing generator.

Once hardware modules and bus communication standards are in place, it is necessary to control the access to and use of these system elements. The system control function as a whole is provided by executive software that follows the protocol laid down by a set of system control procedures. A traditional executive provides the functions of task control, bus control, and fault tolerance and failure recovery in abnormal situations. The structure of a modular real-time avionics executive was developed and partly implemented during the Digital Avionics Information System (DAIS) program (Reference 4). The DAIS executive provides a useful starting point for the development and validation of a standard avionics executive.

One of the possible executive standardization approaches is to adopt a modular family structure analogous to the processing hardware family outlined above. This approach permits a variety of processing elements, either standalone or embedded in subsystems, to be controlled with modules selected from a family of closely related modules. The entire set of modules can then be maintained as a single configuration unit. This

procedure is entirely analogous to using a hardware device family to provide the various levels of processing demanded by a complex, integrated system. As an example, a hardware device family may provide two or more memory chips that allow a designer to select the chip that best meets the memory requirements for the subsystem or function at hand. An executive family may similarly provide modules that provide different levels of fault tolerance (i.e., flight critical, mission critical, or non-critical). The system designer can then select the level of fault tolerance that is appropriate to the function in question. The modular structure allows the designer to tailor his design within broad constraints, minimizing the need for overhead associated with providing high levels of capability for less complex jobs. The method of creating members of an executive family from a single configuration element is shown in Figure 4.

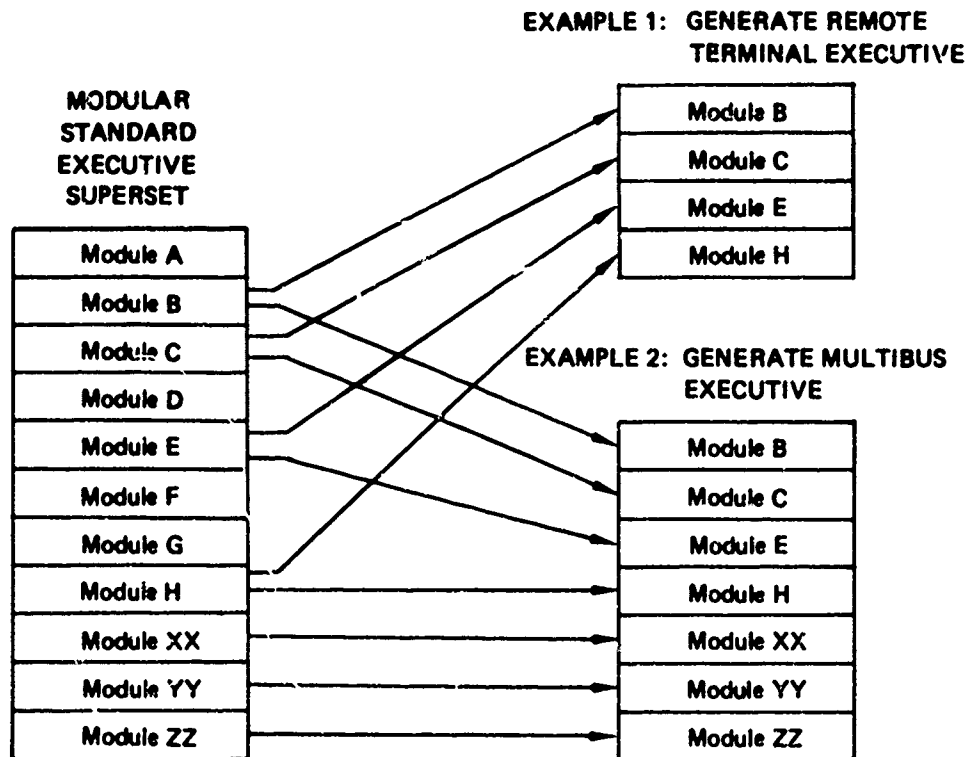


Figure 4. Generating Executive Family Members from a Single Module Superset

With standardized control software available, it will become easier to develop and apply standardized applications modules. The most promising areas for standardized applications modules include functions common to many aircraft types and missions (e.g., navigation and mathematical techniques that can be applied to a variety of problems (e.g., Fourier transforms, Kalman filters)).

C. ACCEPTANCE OF STANDARDIZATION

The success of any standard is determined by its acceptance in the community at large. It is not enough to simply introduce a standard, it must be applied. The degree of acceptance is often affected by the manner in which the standards are developed and introduced to system designers. To improve the speed and effectiveness of the standardization process, it is necessary to choose an appropriate administrative approach to standardization.

Four major administrative approaches have been used to introduce standards (Reference 2). These are:

- a. De facto industry standard - an unofficial standard is adopted by manufacturers to increase product compatibility.
- b. Technical society committee - the standardization process is officially sponsored and monitored by a recognized technical society, such as IEEE, SAE, or EIA.
- c. User Group - a committee of interested military and industry personnel meets regularly to develop or mature a standard. Examples include the JOVIAL User's Group and the 1750 User's Group.
- d. Unilateral government - an interested government organization develops a standard and requires its use on related programs.

Neither the de facto industry nor the unilateral government approach have high success rates since only one side of the product development partnership is involved. Both the technical society committee and user group approaches have worked very well. For systems with purely military applications, the user group approach is favored since the military can sponsor the group. The military can then determine the participants in the meeting, set the frequency of the meeting, and fix target dates for the availability of draft standards.

By itself, standard modular executive software provides only limited improvements in the system software design and integration effort. Much greater improvements can be achieved if the standard modules are combined with standard interfaces between the executive and applications tasks, and to the buses. A rigid executive-to-applications interface, such as the one developed for the DAIS program, permits the applications software design task to be undertaken without detailed knowledge of either the executive or the system control procedures. In addition, the applications software can be functionally partitioned, allowing independent design groups to define and develop portions of the system. As long as each software module adheres to the standard interface, and as long as this standard interface includes the bus control functions, the system integration process becomes a simple mechanical task.

V. CONCLUSION

Technology is becoming available to significantly increase the effectiveness of military aircraft operating at night, in weather and in a

severe threat environment. The potentiality of this technology can be realized through improved integration design based upon modular hardware and software concepts and the proper application of a program of military standards acceptable to industry.

When the modularity concept is fully exploited, resultant availability and performance levels will be equivalent to a larger operating fleet, thus providing force multiplication. Current Air Force avionic integration technology program should be supported to provide a forum and proving ground for these initiatives.

The technical approaches selected during these efforts need to be rapidly reflected in additional military standards that will encourage industry-wide acceptance of common modular design techniques.

REFERENCES

1. Dighton, R. G., "Designing the Hornet for Improved R and M", AAIA 19th Aerospace Science Meeting, January 12-15, 1981, St. Louis, Mo.
2. Microprocessor Applications to Airframe Related Avionics (ARA), Final Report, April 1981, Contract F33615-80-C-0120, Aeronautical Systems Division, WPAFB.
3. S. W. Behnen, M. B. McCall, "Embedded Microprocessors for Avionic Applications", NAECON '82, p. 154.
4. S. W. Behnen, "Implementing the DAIS Executive", NAECON '81, p. 1149.



STANDARD ISAs AND VLSI:
TWO INTERACTING TRENDS

Dr. Peter M. Kogge

IBM Federal Systems Division
Owego, NY 13827
607-687-2121

Biography

Peter M. Kogge is a senior engineer responsible for advanced computing system architectures at IBM Federal Systems Division facility in Owego, NY. Past duties have included system architect on the MCF project, the IBM 3838 Array Processor, and the Space Shuttle Input/Output processor. He has taught at the University of Massachusetts, and is a current adjunct professor at the State University of New York at Binghamton in Computer Science. He is the author of "The Architecture of Pipelined Computers," published by McGraw Hill in 1981.

His educational background includes a BS in EE from the University of Notre Dame, an MS in systems and information sciences from Syracuse University, and a Phd in EE from Stanford University.

Abstract

This paper provides some practical observations regarding the relationship between DoD standard Instruction Set Architectures (ISAs) and Very Large Scale Integration (VLSI). Relevant properties of current ISAs and available VLSI technology are noted. Projections are made on how VLSI may be used to implement these ISAs over the near term. Some problem areas that tend to limit the applicability of today's VLSI under certain circumstances are also addressed.

The final subject is the possibility of new ISA standards in the future: why this might become desirable and how such future ISAs may, or should, be influenced by a desire to take advantage of the full potential of VLSI.

Introduction

This paper attempts to relate some observations about DoD standardized Instruction Set Architectures (ISAs) with the use of Very Large Scale Integration (VLSI) technology in computers that implement them. For this paper, a "DoD standard ISA" means an ISA either appearing on the DoD 5000.5X list of standard ISAs for new programs, or ISAs currently used in computers already in the field or in programs completing development.

Within this paper, three separate topics are addressed:

- Observations about DoD standard ISAs and VLSI as they exist today,
- Comments about how VLSI in the near term may be used in such machines,
- Some initial thoughts as to how VLSI might influence future standard ISAs.

Throughout this paper it is important for the reader to distinguish between the ISA of a computer and its organization or "machine architecture." The first has to do with the image of a computer as seen by the programmer. This includes such things as instruction formats and opcodes, memory management mechanisms, exception and interrupt processing, and Input/Output (I/O) interfaces. It is largely independent of a particular computer implementation, since it indicates how the machine will work on a program, and not such matters as how fast it runs or what kind of memory hierarchy is actually implemented, nor even what technology is used or how the major hardware elements of the machine are interconnected. All these latter items more rightfully fall under the second term of computer "organization."

VLSI as a technology is one step beyond LSI (Large Scale Integration), and represents the ability to place on a single chip of silicon tens to hundreds of thousands of semiconductor devices, enough to implement complete digital subsystems such as a CPU (Central Processor Unit), I/O Processor, Memory Management Unit, etc. DoD's VHSIC program is an example of an attempt to develop VLSI technologies, and apply them in the development of relevant systems.

There are several different ways in which VLSI technologies can be categorized. One is the basic type of semiconductor device used, for example MOSFET, bipolar TTL, ECL, etc. Another is the physical dimensions used in the chip's fabrication, such as chip size, line width, number of levels of metal interconnection available, etc.

A third, and perhaps most relevant to this discussion, is the kind of logical structures available to the computer designer when he attempts to produce a new computer implementation out of VLSI. There are at least three general categories:

1. Gate Arrays - the chip is prefabricated to contain an array of identical elements, usually logic gates, with only the interconnection of these gates left to the designer as his options.

This is the cheapest of VLSI approaches, since the bulk of the chip processing can be done en masse for many different chips, with only the final steps tailored to each design.

2. "Cell Library," or "Master Image" - the designer has a menu of logic blocks such as gates, adders, multiplexers, registers, local stores, etc., from which he can configure a chip. The actual patterns for each of these macros have been worked out in advance, and are placed on the chip in accordance with the designer's wishes.

The greater development expense of this approach in comparison to gate arrays is offset by greater densities of devices on a chip. It also may provide building blocks, such as small memory arrays, that simply are not available on a pure gate array, and in many cases at higher performance levels as well.

3. Custom - the designer has complete freedom to specify down to the individual device level what he wants done.

This permits very specialized and optimized structures to be built, but is the toughest and most time-consuming of the three design approaches.

RELEVANT PROPERTIES OF CURRENT ISAs

Today's DoD-standard ISAs reflect (as do most commercial ISAs) the experiences of the last 20 years of computer design. However, because of their more specialized areas of application and the smaller production runs of computers implementing them, they tend as a group to have some very specific characteristics. Some of those most relevant to this paper include:

- "Single processor" image - i.e., not designed at the beginning with multiple computer configurations in mind.

Examples would include ISAs without synchronization or interprocessor communications instructions, or that have important internal CPU registers addressable as absolute memory locations (which CPU "owns" such locations and when?) or that have very minimal memory management systems not permitting separate or shared areas.

- "Reflections" in the ISA of the original technologies used to implement them.

This might include, for example, opcode assignments reflecting the control signals used to run a particular kind of Arithmetic Logic Unit (ALU) MSI device, or "undefined" opcodes whose functioning happens to depend on the exact logic used to decode the "defined" opcodes, and which programmers have discovered and used for some purposes.

- More complex decoding rules for instructions than found in, say, common microprocessors.

Such decoding rules or instruction formats were often employed to reduce the size of instructions and thus reduce the amount of memory needed to store programs.

- "Families" of computers with near but not total ISA compatibility.

The primary reason for this was often different I/O configurations, different interrupt structures, or often specialized instructions tailored to enhance the efficiency of that ISA for one particular program.

- Largely "microprogrammable" oriented, i.e., much of the ISA is implemented in the computer by using bits out of each instruction as it is executed to select a microprogram that directs proper execution.

This is largely an outgrowth on the previous two observations, plus typically short development cycles where a microcoded machine often permits concurrent hardware debug and microcode preparation.

- Tendency to regard memory as a scarce resource.

This is reflected in the complex decoding rules described above, plus often limited addressing capabilities, and reflects the historical premium placed on memory size due to physical volume and cost constraints.

- Large differences in ISAs in areas where common VLSI "coprocessors" might be applied.

Many different ISAs have, for example, radically different floating point data formats due to different accuracy requirements and/or what was cost-effective given a design with the constraints listed above. Further, even within a family of machines with the same ISA, differences in I/O structure (reflecting different target applications) are common.

- Holdovers from the "pure Von Neumann" view of computers that execute instructions strictly sequentially from the same memory that contains the data, with the main purpose of changing the values stored in memory locations.

A prime example of this is a view in many ISAs that permit and in some cases even requires modification of their own instruction stream as execution proceeds. In heavily overlapped machines, such ISAs often force large amounts of logic to detect when important cases of such sequentialism have occurred, and to stop the machine until the "hazard" has run its course.

SOME COMMENTS ON TODAY'S LSI/VLSI

Today, many people look upon VLSI as a "superman" technology, that is universally applicable and capable of making any computer smaller, cheaper, and faster. In fact, except for the low end of the computing spectrum, we actually have very little idea how to use VLSI, even when not under the constraints imposed on a military-environment computer. For example, 1980 saw the first shipment of many new computers, including microprocessors, mainframe computers, and supercomputers. It is both interesting and significant to note that the microprocessors used custom VLSI with tens of thousands of devices (not gates) per chip, while the higher performance mainframes used gate arrays of up to 2000 devices per chip, and the supercomputers were limited to a few hundred devices per chip. Interestingly, many consider the use of even 500 devices per chip in the supercomputer class as a real accomplishment.

In the arena of computers with DoD-standard ISAs there is a strong recognition that VLSI is coming. However, even more so than with commercial machines, we are still struggling with effective use of lower density LSI, let alone VLSI. The rest of this section tries to pinpoint where the difficulties come from in terms of three generic kinds of LSI/VLSI devices as they are applied today in machines with DoD-standard ISAs:

- Memory, and the "Memory is cheap" view
- Bit Slices
- Chip Sets

Memory

The "memory is cheap" view results from observations that the cost of memory chips is dropping dramatically with the introduction of 64K and 256K devices. Commercial computers may have very large memories, often measured in multiple megabytes, with a major portion of the machines' parts counts coming from memory chips. Consequently, the new VLSI memory technology can have a tremendous impact.

When addressing DoD computers for DoD applications, however, two constraints tend to limit the cost effectiveness or applicability of these larger devices. These are driving requirements for nonvolatility and radiation hardening. These often require either avoiding the VLSI memory chips entirely, using specialized but much lower density parts, or creative combinations of semiconductor and

classical core memories in ways that permit achievement of some, but not all, advantages of the VLSI memories.

The second constraint is that DoD embedded computers typically have not had multimegabyte memories - and may have trouble addressing them cleanly even if they were available. Consequently, such machines still have a relatively large and constant proportion of parts devoted to interfaces, memory management, refresh generation, etc. As mentioned above, very often the original ISAs were often designed to minimize memory usage, even at the cost of more complex instruction decoding.

In the author's view, what is actually needed for modern DoD ISA-based machines are advancements in memory technologies that tend to go in directions other than large and dense, including:

- Short but wide and fast ROMs and RAMs suitable for microstores,
- Multiported register files that permit consideration of higher performance implementations involving pipelining and multiple simultaneous register access,
- Fast FIFO queues for such things as prefetch buffers, storeback queues, I/O buffers, etc.

Bit Slices

"Bit slice" parts attempt to provide a "slice" of a computer's major data flow, for example an 8-bit wide piece of the main ALU, registers, and interfaces. The goal is a part set which permits different ISAs to be implemented by paralleling different numbers of the appropriate slices, with customization to the particular ISA done in microcode. While fine for controllers and relatively simple ISAs, they suffer from some critical drawbacks when applied to more complex ISAs or high speed implementations:

- They often do not have exactly the right paths to implement the different floating point/arithmetic formats. Including the necessary paths external to the bit slices often either results in an excessively slow computation rate, or external hardware rivaling in complexity the original data flow the bit slices were trying to replace.
- They often require external random logic for such things as picking out and aligning displacement fields, computing proper local store register file addresses, FSW and condition code formation, etc., all of which are ISA dependent.
- They still require specialized external interfaces to memory, I/O, AGE, etc.
- Finally, they normally reflect relatively conventional non-overlapped CPU organizations. Attempts to use them in high

performance heavily pipelined machines usually fail because of lack of interfaces that can be run simultaneously for connections to different stages.

Chip Sets

A "chip set" is a set of VLSI chips that together make up a complete computer. While this is the same goal as the bit slices addressed above, the partitioning of functions is different. Each chip does one whole subsystem of the computer, such as major data flow, microprogram decode and sequencing, memory management, floating point extensions, etc. Different performance level implementations of the same ISA can be implemented from such chip sets by adding or deleting various chips, and microcoding (at a lower performance level) those functions for which chips are not present.

Despite the obvious advantages brought about by chips sets, there are still reasons to consider them at best an interim solution to using VLSI:

- The basic chip set is still more or less good for only one ISA, for many of the same reasons addressed above for bit slices.
- The technology used for many chip sets often causes a significant performance penalty due to the need for frequent chip to chip data transmission.

This shows up most often in accessing of microinstructions from external microstores where several chip interface crossings are needed to get microaddresses off one chip, buffered through others to drive the microstore memory arrays, through the arrays themselves, and back into some other chips. In fact, this is often the limiting factor in minimizing cycle time, and thus maximizing performance.

- Even with the reduction in parts count, there are still a relatively large number of chips needed to configure a computer. Many of these come from the need for large offchip RAMs or ROMs for the microstore, and the buffer circuits needed to repower the chip set interfaces to drive them.

An obvious comment that one might have about chip sets is that commercial microprocessors seem to have "larger" chips in terms of functions. For example, microstores are almost always on the CPU chip. In response, one must remember that in most cases microprocessor ISAs and the available technologies were closely coupled (as were early DoD ISAs with their technologies), with the ISA tailored to what the technology could give easily. Further, few of the microprocessors really addressed problems where significant amounts of floating point or other unique instructions requiring complex hardware were desired. Finally, commercial microprocessor designs usually have large production runs of the same chip, permitting amortization of the more costly, but higher density, custom design approach. In contrast, as mentioned before, the smaller

production runs and variations among members of a common DoD-standard ISA family of machines make the cell library approach most attractive, even with its lower overall density than full custom designs.

NEAR TERM VLSI MACHINES WITH TODAY'S ISAs

There is no doubt that VLSI will be employed in implementations of current DoD-standard ISAs in the near future. Several vendors, for example, are working on chip sets for 1750A. However, as discussed above, the general density/performance characteristics of such technologies may be less than one might hope from observing equivalent events in the commercial arena. These reasons, in summary, include:

1. ISAs that practically demand microcode support - and microcode that often changes from program to program.
2. Performance requirements that include a healthy proportion of floating point instructions, as well as time-sensitive I/O response requirements.
3. Widely varying I/O device and interface characteristics.
4. Memory volatility and initial program load issues.
5. Low production runs that limit the investment that can be made in unique parts.

What then might one expect to see in the near future? First might be the extensive use of commercial microprocessors in the very low end of the performance spectrum (embedded controllers, etc...). Perhaps the major barrier here are requirements to use DoD standard High Order Languages (HOLs) such as Ada¹, CMS-2, JOVIAL, etc.

In the higher performance ranges where most DoD-standard ISAs are implemented, (and where the DoD standard HOLs have most applicability) two trends may develop, both of which are becoming visible in the parts being produced by the VHSIC program:

1. Extension of the "chip set" concept, with attempts to reduce parts counts and unique chip developments by developing "coprocessors" that execute only certain parts of the typical ISA, but which can be largely self-contained. Examples include floating point, I/O controllers, some memory management schemes, simple caches, etc. Onboard microstore that can be changed without totally redesigning the entire chip offers some hope of carrying a design over from one ISA to another, or at least among programs using the same generic ISA with minor changes. VLSI "cell library" design techniques are a natural for this since they allow minor changes to the chip to be done without an entire redesign.

¹Registered Trademark of the U.S. DoD.

Pipelining [1] within each chip will probably come into vogue in an attempt to buy back the performance lost by having to traverse multiple chips.

Also, physical partitioning of the ISAs implementation may have positive effects on other parts of the computer, such as in the main CPU, where there may still be ISA-unique designs, but where much of the overhead required to do these other functions can be removed, resulting in narrower microstores and thus fewer off chip microstore memory chips. An example of this approach as applied to the System/370 ISA can be found in [2].

The major problem with this approach is that most DoD-standard ISAs were not designed in ways that permit clean partitioning of functions. The result may be that unique interfaces may be needed to communicate ISA-specific information from one chip to another.

2. Development of less than VLSI density components - those that commercial vendors have tended to ignore, but which can significantly simplify the construction of high speed implementations. The small specialized memory-like devices mentioned earlier are prime candidates; others might include faster multipliers, barrel shifters, comparators permitting "duplicate and compare" or triple modular redundancy for fault tolerance, cache controllers, etc.

NEXT GENERATION DOD-STANDARD ISAs

This section addresses three major questions relating to the future of DoD-standard ISAs in an increasingly VLSI-oriented implementation environment:

1. Why might we consider changing ISAs at all?
2. What would be the characteristics of such ISAs from the vantage of VLSI?
3. What might be the time frame for such new introductions?

Why Change ISAs

Introducing a new ISA and computers implementing it cannot be done casually. There are substantial costs involved over and above the obvious ones of machine development. These costs come largely from providing the same level of software support available for existing ISAs, and must be carefully weighed against the advantages provided by the new ISA/implementation. Some valid arguments can be made that much of these costs will be avoided by the introduction of Ada and the various standard support systems to be built along with it. However, even with Ada it will still be necessary to retarget the Ada compiler to the new ISA, and modify the system code that will run on these computers to use the new features offered by the ISA.

With these comments in mind, then what might be the reasons for changing ISAs? First might be to fix limitations in current ISAs, such as:

1. Expansion of memory addressing limits to take advantage of potentially cheaper memories.
2. More accuracy in calculations such as 32 bit fixed point and more generalized and easier-analyzed floating point formats (e.g., IEEE standard floating point).
3. Better memory management and run time protection checks.
4. Elimination of architectural "features" that often have undesirable side-effects.

The second reason would be to provide features not available in the current suite of DoD-standard ISAs, but of either real value to or required by future embedded systems. These features might include:

1. Efficient, and verifiably correct, support for secure systems.
2. Better compiler target providing:
 - Simpler compiler with consequently more trust in the code that it produces.
 - More hooks to support debugging real-time code from the HOL source directly.
 - Easier and more verifiable implementations of advanced HOL features like exception handling and multitasking.
3. Fault tolerance support, such as checkpoint/restart instructions and ability to establish program state from outside a failed processor.
4. More performance.
5. Ability to be specialized to unique applications. For example, could we toss out some characteristics such as floating point, and still have a coherent enough ISA to use by the hundreds or even thousands in onboard satellite imaging applications.

Their Characteristics as Influenced by VLSI

In the opinion of many, the ultimate VLSI-based computers will be characterized by extensive parallel processing multiple computers with very significant fault tolerance, and where the ISA for each computer is relatively simple. The reasons tend to revolve around the major efforts needed to design more and more complex single computers. Truly high performance single CPU computers with today's ISAs either

seem to be unable to efficiently use VLSI at all (for reasons given above), or require inordinate numbers of unique chips, all out of proportion to the gain in performance. In contrast, "easy" VLSI implementations seem to be going in the direction of integrating whole, and eventually multiple, simple computers (CPU + memory) of moderate performance on a chip. This is exhibited by many commercial microprocessors that now offer significant on-chip memory, and by some very advanced research as in Texas Instruments' RIC chip (4 computers on a single chip - see [3]) and H. T. Kung's systolic arrays ([4], Chapter 8).

Further, significant recent research has investigated the properties of very "simple" Reduced Instruction Set ISAs consisting of a small number of primitive instructions from which more complex processing sequences can be built (cf [5] and [6]). The resulting machines are far simpler hardware-wise, and actually faster on both a cycle and a program basis than older, and more complex, ISA-based machines (cf [7] -the IBM 801). The reason is often that no microcode is needed (reducing both hardware and the cycle-limiting microstore access path), and what does have to be decoded is so regular and simple that very little logic and very short cycle times result. Further, these studies also indicate that the lack of "complex" instructions neither complicate compilers that target to such ISAs, nor do they greatly explode the amount of code required to represent programs processed by such compilers. In fact, many compilers for current DoD-standard ISAs are often not capable of using the more complex instructions - relegating them to either disuse or use in specially written assembly code subroutines.

This combination of good compiler target, simple hardware, and high performance seems to be an excellent match to VLSI capabilities.

The area of non-Von Neumann architectures is also one that has significant promise to achieve many of the above objectives when married to VLSI implementations. "Data Flow" architectures eliminate the idea of a highly sequential central processor doing one instruction at a time by distributing control among all instructions that are ready to execute (cf [8]) "Reduction architectures" take advantage of many of the ideas from applicative and functional programming to produce highly verifiable and highly concurrent systems (cf [9]).

Possible Timeframe

It is unrealistic to expect that such ISAs and the VLSI implementations that make them attractive will be available in the next 3 to 5 years. Too much is invested in the current generation, and too little known about exactly where current ISAs break down. Further, many of the desirable ISA features are still very much of a research area today. For example, today we do not know how to make fault tolerance in the hardware totally isolated from the software running on it except by massive applications of redundancy. Neither do we know how to take programs written in today's HOLs and automatically partition it across multiple computers. Both of the

above today require extensive programmer intervention, as can be seen by the kinds of new constructs being added to modern HOLs (e.g., the exception and multitasking facilities of Ada).

When it comes to tying together multiple computers, we are just beginning to understand appropriate interconnection patterns to use for certain well structured problems. The ISA support to handle the more general cases, particularly when fault tolerance is thrown in, is an open issue.

Finally, many computer scientists feel that the future wave in computers will definitely be non-Von Neumann, and literally dozens of groups are proposing new languages, ISAs, and machine organizations. However, until the really important features of this research can be distilled into sufficiently robust ISAs to cover all the problems addressed by today's ISAs, and palatable transition paths found from the huge investments made in today's support technology, such technologies will be limited - at least in DoD applications - to perhaps one-of-a-kind specialized applications.

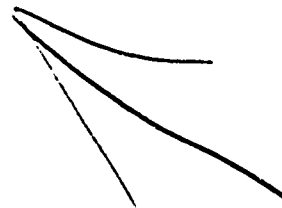
CONCLUSIONS

This paper has tried to analyze the relation between standard ISAs and VLSI. The general conclusions were two-fold. First, in the near term, VLSI may not have as much effect on the implementation of DoD-standard ISA based computers as might be surmised from general technology growth curves, at least at the higher performance end of the spectrum. This is due to some of the general characteristics of such ISAs (such as heavily microprogram oriented), and in the kinds applications (varying I/O, special functions, significant floating point, etc.). Perhaps the most probable trend is in the direction of tailorable VLSI chip sets targeted to specific ISAs.

The second major conclusion is that in the long run (perhaps a decade) it may be necessary to develop new ISAs to fully exploit the capabilities of VLSI so that new kinds of missions not possible today can be attempted. However, while we may have some inklings of what such ISAs might look like, it is too early to seriously consider details. Time is needed for the present generation of standards to mature and their effectiveness determined, particularly in terms of Ada and Ada program support, and in standards in other areas such as I/O.

BIBLIOGRAPHY

1. Kogge, P. M. The Architecture of Pipelined Computers, McGraw Hill, New York, NY, 1981.
2. Agnew, P. W. and A. S. Kellerman. "Microprocessor Implementation of Mainframe Processors by means of Architecture Partitioning," IBM Journal of Research and Development, Vol 26, No. 4, July 1982, pp 401-412.
3. Budzinski, R. et al. "A Restructurable Integrated Circuit for Implementing Programmable Digital Systems," COMPUTER, March 1982, pp 43-54.
4. Mead, C. and L. Conway. Introduction to VLSI Systems, Addison-Wesley, Reading Mass, 1980.
5. Patterson, D. and C. Sequin. "A VLSI RISC," COMPUTER, September, 1982, pp 8-22.
6. Fairclough, D. "A Unique Microprocessor Instruction Set," COMPUTER, May 1982, pp 8-17.
7. Radin, G. "The 801 Minicomputer," 1982 ACM Symposium on Architectural Support for Programming Languages and Operating Systems, March 1982, pp 39-47.
8. Treleaven, P. C. et al. "Data-Driven and Demand-Driven Computer Architecture," Computing Surveys, Vol 14, No. 1, March 1982, pp 9-143.
9. Backus, J. "Function-level computing." IEEE Spectrum, August 1982, pp 22-38. .eol.



- SUCCESSFUL DEVELOPMENT -
SUPPLY OF STANDARDIZED COMPUTER HARDWARE AND SOFTWARE
IN A PERIOD OF RAPID TECHNOLOGICAL CHANGE

Keith Dixon
Ferranti

BIOGRAPHY

Keith Dixon has been associated with Ferranti Computer Developments since 1965, initially for Naval Applications but since 1970 with emphasis on avionic and other hard environment applications areas. He assumed marketing responsibilities for Military Computer LSI development and investment in 1975, and is now responsible for marketing and sales of FCSL Standard Military Computer Products, and of Army, Avionic and ATC/Air Defense Systems produced by Ferranti, South Wales.

ABSTRACT

This paper overviews the experiences of Ferranti Computer Systems as major developers, suppliers and users of UK Mod Standard Computer and interface hardware and software items over fifteen years. Ferranti's leading role in this field currently centers on the development of the Military Argus Computer Range and Associated Coral/Mascot Compilers and Operating Systems with emphasis on the high performance bipolar VLSI radiation hard M700/40, due for release early 1983. These developments follow the earlier computer/module ranges of the FM1600 series, widely applied in Naval Command and weapons control and air traffic control/Air Defense, and the F-100-L Military Microprocessor, mainly for missile and space applications. For the future, Ferranti is heavily committed to the MIL-STD-1553 System Interface, and is engaged in developments of MIL-STD-1750 in preparation for VLSI Implementation, probably in the ADA/APSE era.

The paper seeks, in the light of experience, to identify the vital ingredients of success for standardization policies. Where market size can be limited, but performance and quality requirements are extremely high, during a period of rapidly developing technology.

AN/UYK-43(V) and AN/UYK-44(V) PROGRAM OVERVIEW

The AN/UYK-43 and AN/UYK-44 programs will provide replacements for the current Navy standard computers, the AN/UYK-7 and AN/UYK-20 respectively. The AN/UYK-43 and AN/UYK-44 are planned for use in all Navy shipboard tactical systems requiring digital computers. The AN/UYK-43 is a militarized general purpose large scale 32-bit computer. The AN/UYK-44 is a militarized general purpose 16-bit embeddable processor or a stand-alone fully packaged minicomputer.

Both the AN/UYK-43 and AN/UYK-44 are modular in construction to permit optimal configuration according to the unique requirements of each user system, enhance maintainability and logistics supportability, and permit orderly infusion of advanced technology into the computer hardware.

The AN/UYK-43 and AN/UYK-44 are both micro-programmed emulators of their respective antecedent computers. This emulation permits use of the same support software for systems application software development, and it allows capture of existing software that runs on the respective antecedent computers. Moreover, the Instruction Set Architecture (ISA) can be extended or enhanced to meet new requirements.

Biography

CAPT James P. O'Donovan
Project Manager (PMS-408),
Naval Shipboard Tactical Embedded Computer Resources Project
Naval Sea Systems Command
Washington, D. C. 20360

BSEE Manhattan College
MSEE Naval Postgraduate School
MBA George Washington University
Graduate, Industrial College of the Armed Forces

Experience:

Captain O'Donovan has served in a variety of R&D, Fleet Support and acquisition engineering billets during his career.

Tours at the Long Beach Naval Shipyard, Electronics Officer aboard the USS Hornet, and Fleet Maintenance Electronics Officer on COMSERVPAC and CINCPACFLEET staffs have been interleaved with Washington, D. C. assignments as NTDS Project Officer for the CGN-36 and CGI-38 ship class Combat Direction System designs, and as Director for Electronics Programs on staff of the Assistant Secretary of the Navy (Installations & Logistics).

Current Assignment: Captain O'Donovan comes to his present job as PMS-408 from a tour as Commanding Officer of the Naval Electronics Engineering Center, Charleston, South Carolina.

AN/AYK-14(V) PROGRAM OVERVIEW

The AN/AYK-14(V) Program represents a family of Navy designated standard logic modules. These modules can be configured in a variety of combinations to produce processing elements for incorporation into subsystems of a weapon system or produce complete self contained general purpose computer systems. The design of the AN/AYK-14(V) modules includes features that permit AN/AYK-14(V) users to select only those functions necessary to meet individual applications needs. The flexibility of this "building block" approach also allows future hardware reconfiguration and growth to accommodate changes in an applications needs with minimal impact to the weapon system. The functionally partitioned modular design also enables the AN/AYK-14(V) Program to capitalize on advances in technology in a planned orderly manner which is transparent to the users.

This brief presents program status and an overview of the AN/AYK-14(V) as a computer system and as a processor. Examples will be provided which demonstrate the inherent flexibility in the AN/AYK-14(V) design that enables it to meet a wide range of applications and take maximum advantage of technology infusion and pre-planned product improvement.

Biography

Henry H. Mendhall
Computer Resources and Avionics Systems Division
Naval Air Systems Command
Washington, D. C. 20360

BSEE Drexel University, Philadelphia 1968

Experience

14 years with the Naval Air Systems Command in Avionics computer and software systems.

Current Assignment:

Section Head for Advanced Computer Systems

Responsible for planning, engineering and acquisition of Navy Airborne standard and planned standard computer resources.

Navy Packaging Standardization ThrustsAbstract*Needed for**For a long time to*

Standardization is a concept that is basic to our world today. The idea of reducing costs through the economics of mass production is an easy one to grasp. Henry Ford started the process of large scale standardization in this country with the Detroit production lines for his automobiles. In the process additional benefits accrued, such as improved reliability through design maturity, off-the-shelf repair parts, faster repair time, and a resultant lower cost of ownership (lower life-cycle cost). The need to attain standardization benefits with military equipments exists now. Defense budgets, although recently increased, are not going to permit us to continue the tremendous investment required to maintain even the status quo and develop new hardware at the same time. ~~We need~~ more reliable, maintainable, testable hardware in the Fleet. ~~We must~~ recognize the obsolescence problems created by the use of high technology devices in our equipments, and find ways to combat these shortfalls.

The Navy has two packaging standardization programs that will be addressed in this paper; the Standard Electronic Modules (SEM) and the Modular Avionics Packaging (MAP) programs. Following a brief overview of the salient features of each program, the packaging technology aspects of the program will be addressed, and developmental areas currently being investigated will be identified.

Biographical Sketch

John R. Kidwell was born in Pleasureville, Kentucky on June 22, 1943. He began his affiliation with the Naval Avionics Center (NAC), Indianapolis, IN, in 1962 as a co-op student from the University of Evansville. In 1966 he received his BSEE and also became a full-time employee at NAC in the Engineering Department. Some of the programs in which he has been involved are Polaris - Poseidon, missile telemetry, electronic warfare, and advanced packaging techniques. He was a major contributor, as a technical editor and writer, to the Naval Air Systems Command (NAVAIR) Avionics Master Plan. This document provides "...long-range plans and alternatives for the appointment of a high level avionic equipment capability projected over the next decade." Mr. Kidwell is presently Head of the Advanced Avionics Packaging Branch (Code 965). NAC efforts for both the Standard Electronic Modules (SEM) and the Modular Avionics Packaging (MAP) programs are performed in this Branch.

Introduction

Standardization is a concept that is basic to our world today. The idea of reducing costs through the economies of mass production is an easy one to grasp. Henry Ford started the process of large scale standardization in this country with the Detroit production lines for his automobiles. Henry's primary concern was getting his costs down so he could undersell his competition. In the process, however, additional benefits accrued, such as improved reliability through design maturity, off-the-shelf repair parts (without the need for one-of-a-kind craftsmanship), faster repair time (due to interchangeable parts), and a resultant lower cost of ownership (lower life-cycle cost). Standards exist in every facet of life, from light bulbs, to shoe sizes, to socket wrenches. Use of standards within military equipments provides the means to increase the reliability, enhance the logistic support posture, simplify maintenance, and--as a result of the above--reduce the overall life-cycle cost of military hardware.

There are many areas where standards should be applied in military equipments. These areas can be broken down into the general categories of hardware, software, and systems architecture. The hardware area can be further subdivided into the areas of systems, subsystems, modules, and devices. The software area can be broken down into selection of processor architecture, higher order language, algorithms, and support considerations. System architecture considerations include bus structures/interfaces, fault tolerant techniques, electrical interfacing, built-in test techniques, and the general area of packaging and thermal management.

Each of the areas addressed above, and subdivisions of those areas, are actively being pursued by the services. This paper presents an overview of the Navy's module standardization approaches, provides the status of Navy efforts toward standardization of avionics system level packages providing improved thermal management, and assesses the impacts of advanced forms of integrated circuit technology such as very high speed integrated circuit (VHSIC) devices on these standardization efforts.

Why Standard Modules?

In numerous cases, military systems/subsystems have been developed to satisfy a specific requirement for a specific platform, resulting in numerous systems of similar characteristics, but each tailored to satisfy a unique platform requirement. In other words, equipment

developments have not been approached with wide-angle vision of all potential Navy and other military service platforms to establish an optimum level of hardware commonality within these weapons systems.

The high cost of maintaining current systems is, in part, due to this multiplicity of equipments performing similar functions. In addition, the escalating inflation factor has "shrunk" the buying power available for use in the development of new systems. In light of this dichotomy, with no reversal of the trends in the foreseeable future, steps must be taken to achieve a higher degree of platform commonality in the future.

If we examine the stimuli for non-commonality of electronic systems between platforms, we find that newly specified weapon systems "can't use" existing items because of either functional limitations, electrical interface incompatibility, installation incompatibility, inadequate or undefined reliability and maintainability provisions, or because the standard item "belongs" to another service branch. However, as formidable as achieving commonality might appear in light of these dilemmas, the application of a coordinated systems engineering approach, coupled with advanced technologies, afford the opportunity to achieve significantly increased platform commonality of electronic systems/subsystems.

Several ongoing Navy programs are designed to address these factors. These programs have found that a digital mechanization of subsystems lends itself to partitioning into groups of identifiable functional elements. The size of these functional elements may be large or small, depending on the amount of functional subdividing required by the system specifications. By comparing functional elements resulting from the partitioning of a variety of digitally-mechanized subsystems, a core set of functional circuits which have high commonality across these subsystems can be identified. Hence, functional standardization at a level below the subsystem is feasible and practical. The complementary effects of these programs allow the Navy to:

- * partition electronic functions in a manner that will achieve high hardware commonality within many equipment applications,
- * ease the logistics support burden on the congested supply system by extensive intersystem commonality of a limited number of module types,

- * reduce life-cycle costs,
- * minimize the technology dependence by documenting modules with functional specifications, allowing a measure of "technology transparency," and
- * achieve high reliability through stringent quality assurance and design requirements focused upon these fewer module types.

These programs are the Standard Electronic Modules (SEM) program and the Standard Avionics Module (SAM) sub-program. The SAM subprogram is a part of the Modular Avionics Packaging (MAP) program. The MAP program not only addresses standard modules, but also improved box-level standard packaging approaches which provide lightweight, thermally efficient system level packaging approaches.

The SEM Program

The Navy's Standard Electronic Modules (SEM) program was established in the mid-1960's as a module level standardization approach to the design, development, production, and logistic support of Navy electronic equipments and systems. The SEM program provides the system developer with a family of commonly used, low cost, reliable, functional building blocks for implementation in a variety of military hardware. The goals of the program are to favorably impact system life-cycle costs, availability (reliability and maintainability) and supportability, and to minimize the impact of technology obsolescence. By specifying the functional, mechanical, and thermal interfaces for modules, current technologies and manufacturing techniques may advance with the state-of-the-art and be applicable to all previous and new systems specifying the SEM functions.

The SEM program is documented in a hierarchal structure of tri-service coordinated military standards and specifications. Individual modules are documented by use of Military Specification MIL-M-28787 "slash sheets" (i.e., detail specifications) which set forth their form, fit, and function requirements. SEM functional specifications not only encourage vendor innovation to provide a truly competitive procurement environment, but also provide a means for system technology updates at the plug-in module level. The program does more than use a form, fit, and function approach. The mechanical hardware is defined, not only to the module size and shape, but to the environmental parameters the module must survive. The thermal capability

of the module, in both shipboard and avionics application, is defined. Not only are these conditions specified, but modules provided to the specification are rigidly tested to ensure that they meet the requirements.

The SEM program has progressed to the degree that in excess of 300 standard module types have been applied to more than 100 various Navy electronic equipments. Some five million standard modules have been committed to service use in these applications. While primary usage has been in shipboard equipment, there are also some applications in ground based, avionics, and missile equipments.

The progress of the SEM program can be measured in terms of program achievements. The success of any standardization program can be assessed by ascertaining the level of commonality achieved. Table 1 indicates the level of commonality achieved by SEM standards in several Navy systems, large and small. As can be seen in Table 1, commonality for these systems ranges from some 60 percent to over 90 percent. These results point to a reduction in logistic costs through a reduction of overall spares requirements. Fleet reliability data have been accumulated for several SEM-configured systems. Table 2 indicates these reliability data in terms of MIL-HDBK-217B predicted failure rates, accumulated field data, and the ratio of achieved field data to predicted data. As shown in Table 2, a significant reliability improvement has been achieved through the use of qualified SEM standards. This reliability improvement can be attributed directly to the rigid, independent quality assurance procedures applied to SEM. Success in driving acquisition costs down is illustrated in Figure 1. These data show module unit production costs for several categories of module standards for successive year buys plotted in "then year" dollars. These data indicate that a true competitive environment in which vendor innovation is permitted reduces module costs to a minimal value in spite of inflation.

This brief review of SEM commonality, reliability, and cost data indicates that the program is indeed moving toward achievement of its goals.

The MAP Program

Historically, avionics have been produced as discrete "black boxes," each dedicated to a specific avionics function, and with little control over details of internal construction. This approach to avionics procurement has resulted in a proliferation of unique system designs,

hardware implementations, and unique logistic support considerations. Standardization of, and commonality in, avionics equipment has typically been applied only at the subsystem level.

Although the unique "black box" approach may appear to provide the best cost, performance, reliability, and timely availability of new avionics due to the extremely competitive environment provided by multiple suppliers, each exploring different approaches, frequently the reverse is actually true. The life-cycle cost of equipment so acquired is higher due to unique design costs, additional spare parts cost, unique ground support equipment and maintenance manuals, training costs, etc. The performance has often been degraded due to environmentally induced failures caused by inadequate packaging. The reliability is often lower because insufficient emphasis was placed on thermal performance. The system may not be timely in its availability because designers "re-invented the wheel." To attack the problem areas, the Naval Air Systems Command initiated a program known as Modular Avionics Packaging (MAP). The program is structured to address the retrofit market, accomplished through Engineering Change Proposal (ECP) and Conversion In Lieu Of Procurement (CILOP) efforts, as well as new aircraft applications.

The objectives of the MAP program are to identify, develop, and implement packaging concepts that will satisfy the stringent high density, lightweight, high reliability and high maintainability requirements of Navy avionic equipment/systems. Primary packaging approaches under consideration are the use of SAM, standard enclosures (SE), and the integrated rack (IR). The SE provides a lightweight, thermally efficient enclosure for subsystems implemented in SAM. The IR combines groups of SAM into subsystems or multiple subsystems and provides all required interconnections, mechanical and environmental protection, and cooling. By standardizing the module interfaces, power supplies, and cooling techniques, the IR will provide a more reliable, maintainable lighter weight avionics package.

The Standard Avionic Module

The "lessons learned" across 15 years of Navy effort devoted to the SEM program are not being ignored. The concept of functionally specified, tightly controlled modules will be fostered. Figure 2 illustrates the module versions under development. These versions are the Format B, which is currently a standard module form factor of the SEM program, and a compatible new module form factor known as the

1/2 Austin Trumbell Radio (ATR), which is based on the 1/2 ATR enclosure size. Hardware elements for the Format B are currently being multiple sourced in industry. An intensive program to design, develop, test and multiple source the 1/2 ATR hardware is in process. Commonality of hardware elements between the Format B and the 1/2 ATR module (such as connectors, keying bushings, keying pins, etc.) will allow acceleration of the development process and result in further reduction of life-cycle costs.

The Standard Enclosure

The commercial airline industry long ago recognized the need to standardize on equipment types that could be utilized not only by different airlines, but also by different aircraft types on an airline. Standards are now in place for a variety of elements, one of which is a standard dealing with the sizes, thermal capability, and other mechanical characteristics of the enclosures used to house the electronics. The enclosures come in a variety of sizes, from a 1/4 ATR box in increments to a 1-1/2 ATR box. (New commercial specifications use the terminology Modular Concept Units, or MCU's, but this paper will retain the ATR terminology.) The boxes have a fixed height dimension of 7.62 inches, a variable depth (either 12.52 inches or 19 inches), and a variable width from 2.25 inches for the 1/4 ATR up to 15.29 inches for a 1-1/2 ATR. The "ATR short" box (the one with a depth of 12.52 inches) has become a de facto standard, and is the only standard depth included in the new MCU box specification. Most commercial avionics boxes are configured in these size enclosures.

The military has long recognized the need to standardize on box sizes and types, and has had "preferred" sizes of boxes as specified in MIL-STD-172 (in general the same sizes as 1/4, 1/2, 3/4, and full ATR boxes) for many years. These boxes have been used in a variety of military equipments, but never to the desired extent.

Both the Air Force and Navy are developing the necessary tools to provide SE for packaging avionics equipments. The Air Force is in the process of developing a form and fit specification for SE. The Navy SE development subprogram is a part of the MAP program. The SE being developed are in the 1/4, 1/2, 3/4, and full ATR short sizes. Use of the current SEM format B, span 2 module is possible in all four configurations of boxes and the 1/2 ATR module is to be used in the 1/2 and full ATR box sizes. The SE effort is currently under contract to Boeing Aerospace. The Air Force has provided funding to augment this effort to address

specific areas applicable to joint Air Force/Navy use of the SE. Figure 3 illustrates the Navy's SE concept. Preliminary hardware developments indicate that a lightweight (less than three pounds), thermally efficient (500 watts), MIL-E-5400 Class 2X enclosure in the 1/2 ATR size is attainable. Additional hardware developments will be performed in fiscal year (FY) 1982.

The Integrated Rack

The IR is planned primarily for application to new aircraft, although it could be retrofitted into certain existing aircraft. The IR provides the capability of housing more than one subsystem in the same enclosure, and providing the mechanical, environmental, and electrical interfaces for each subsystem. Studies show this concept can provide a weight and volume savings over separately packaged subsystems of up to 30 percent. The concept has been developed for the Navy by General Electric, Lockheed California, and most recently by Grumman Aerospace. The concept is far reaching in nature in that it will allow maintenance at the module level in flight (for larger, multi-personnel aircraft) or on the aircraft carrier deck (for smaller, high performance aircraft). Figure 4 illustrates one possible configuration of an IR. It should be noted that it is planned to make a single "tier," or row of modules in an IR, mechanically and electrically interchangeable between the IR and SE. A system developed in one enclosure could be inserted into the other enclosure configuration. An IR specification is currently on distribution to a large segment of Government and Industry. It is planned to perform fabrication and environmental test on one version of an IR in 1982.

The Packaging Impact of Very High Speed Integrated Circuits (VHSIC)

New device programs, such as VHSIC, can have a significant effect on packaging efforts. The VHSIC program is a major Department of Defense (DOD) thrust to advance the state-of-the-art of integrated circuit technology to support high speed, high throughput signal and data processing requirements of defense electronic equipments in the mid-1980's and beyond. This program is developing monolithic, very large scale, high speed, integrated circuit technologies which can produce complex devices to mechanize critical electronic subsystems required to meet future military needs. Completion of the VHSIC program will enable DOD to reduce life-cycle costs of military electronic systems, and insure future utilization of advanced integrated circuits in

defense equipments. A goal driving the entire program is to realize the capability for fielding advanced systems based on an availability of military-qualified integrated circuits with sub-micron feature size in the mid-to-late 1980's time frame.

The packaging characteristics required for VHSIC are not unique to a standard module program. Higher speed capabilities, increased number of required interconnections, and improved thermal dissipation capability, are applicable to any packaging technique selected. One advantage that a form, fit, and function module approach has is that new techniques or technologies may be implemented within the philosophy of functional standardization. The capability for replacing a family of technologically obsoleted devices is maintained without a drastic impact on the overall system.

There are, however, certain attributes of VHSIC and other very large scale integrated (VLSI) circuit devices which will impact module standardization. For a given module size, increased levels of circuit integration resulting from VLSI will allow greater functional density per module. This will likely increase thermal density, cause difficulties in functionally specifying the module, increase testability and fault isolation requirements, and increase individual module price. Increased levels of integration will also decrease the number of standard modules that can be utilized across system types. However, forms of standardized modules employing VLSI should be realizable in future Navy programs in various areas of digital and low level analog signal applications. It is speculated that these common modules will include the following general functions: Data Processing Family (memory, input/output, processor, arithmetic units), Signal Processing, Signal Distribution, Sensor Interfaces, Actuator Interfaces, Display Interfaces, Control Circuitry, Manual Entry Devices, Signal Conversion, Low Level Analog, and Power Supplies for the above. These "higher powered" functions will replace those currently used.

The potential system life-cycle cost benefits of increased levels of circuit integration have been well documented. Module standardization would appear to be supportive of, and complementary to, the life-cycle cost benefits inherent in VLSI circuitry. It is concluded that the use of advanced forms of circuit technology is compatible with a module standardization program, and together they have the potential for significant reductions in military electronic equipment life-cycle costs.

There are development efforts underway within the SEM and MAP programs to provide the capability to accommodate VHSIC and other LSI/VLSI devices. These efforts include the development of female contacts for use with multilayer boards, the investigation and evaluation of heat pipe and flow-through modules, use of chip carriers, investigations of alternative substrates, and the development of thermally efficient enclosures to house standard modules. Detailed discussions are being conducted with VHSIC contractors to establish their packaging needs. These developments will lead the Navy to the selection and development of VHSIC compatible technologies and module types.

Summary

The progress of module standardization efforts within the Navy demonstrates that standardization can result in highly reliable, low life-cycle cost electronic systems while still accommodating technology advances. The keys to success have been flexible module configurations that allow incremental growth; the use of functional, mechanical, electrical, and thermal specifications; and use of a strong design review and quality audit function. The continued success of these programs depends on their ability to evaluate advanced technology devices, functions, and system requirements, and to accommodate these elements within the programs. These programs provide a vehicle to address the problems of poor reliability, maintainability, and high life-cycle cost that the Navy faces. If future acquisitions of electronic systems are to gain the advantages offered by standardization, concentrated efforts must be sustained and supported. The message is clear: "don't reinvent the wheel."

COMMONALITY ACHIEVEMENTS

SYSTEM	STANDARDS				
	TOTAL MODULE TYPES	TOTAL MODULES/ SYSTEM	TYPES	SYSTEM TOTAL	PCT STDS
AN/BQQ-5 SONAR	138	16,000	21	12,000	75
TRIDENT FCS	156	15,069	88	13,110	87
TRIDENT SONAR (BQQ-6)	146	13,467	62	10,882	80
MK-116 MOD 1 FCS	30	737	18	517	70
PORTABLE TEST SET	24	140	18	86	61
FIN STABILIZER	30	75	27	72	96
FSK DEMODULATOR	8	23	5	20	87

TABLE 1

SEM RELIABILITY ACHIEVEMENTS

SYSTEM	PREDICTED FAILURE RATE F/10 ⁶ HOURS	OBSERVED FAILURE RATE F/10 ⁶ HOURS	PREDICTED <u>OBSERVED</u>	MODULE HOURS
MK-88 POSEIDON FCS	0.43	0.017	25.3	134 × 10 ⁶
ERSCAN	0.24	0.028	8.5	249 × 10 ⁶
AN/BQQ-5 SONAR	1.20	0.170	7.1	825 × 10 ⁶
AN/BQQ-6 SONAR	1.20	0.150	8.0	363 × 10 ⁶

TABLE 2

SEM MODULE PRICES

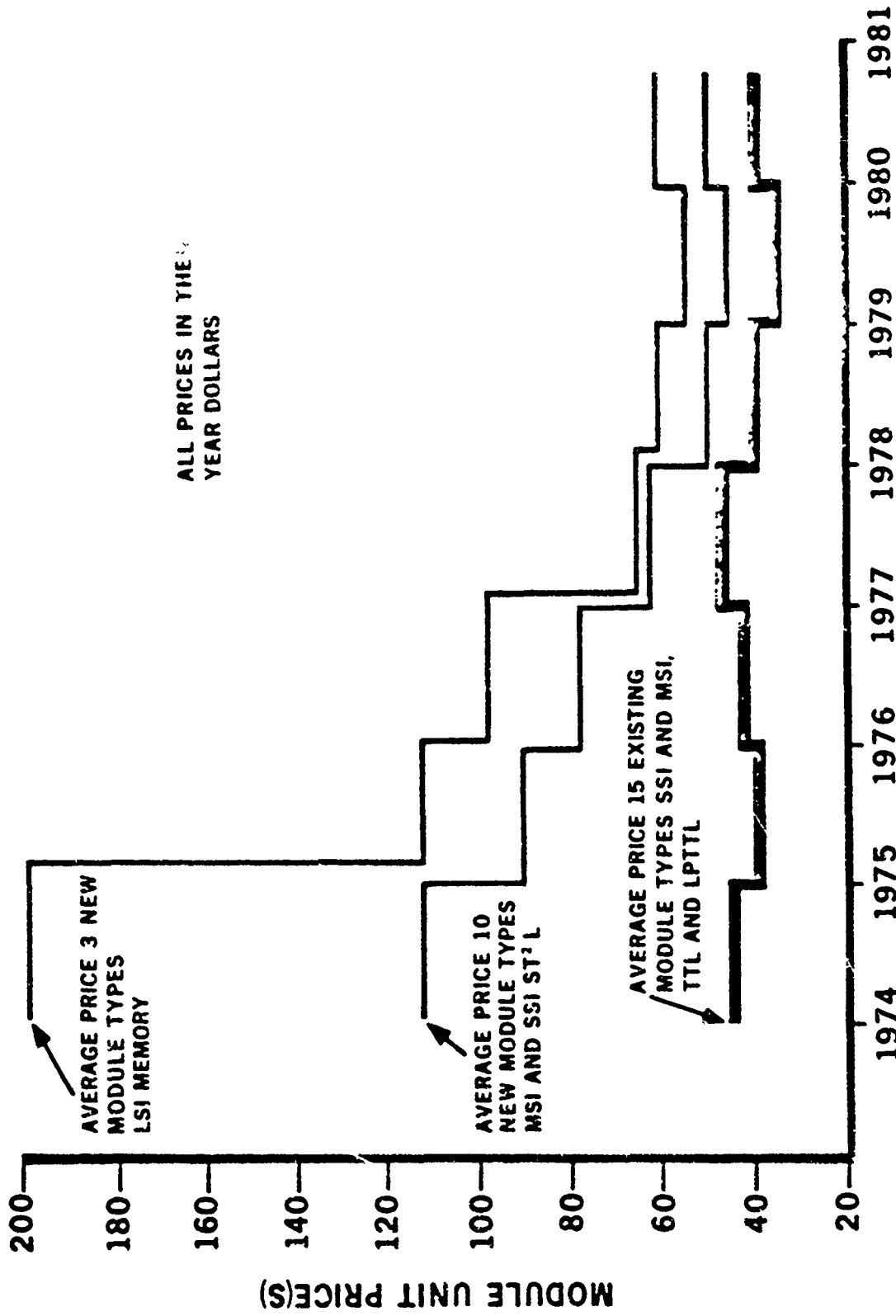


FIGURE 1

SEM FORMAT B

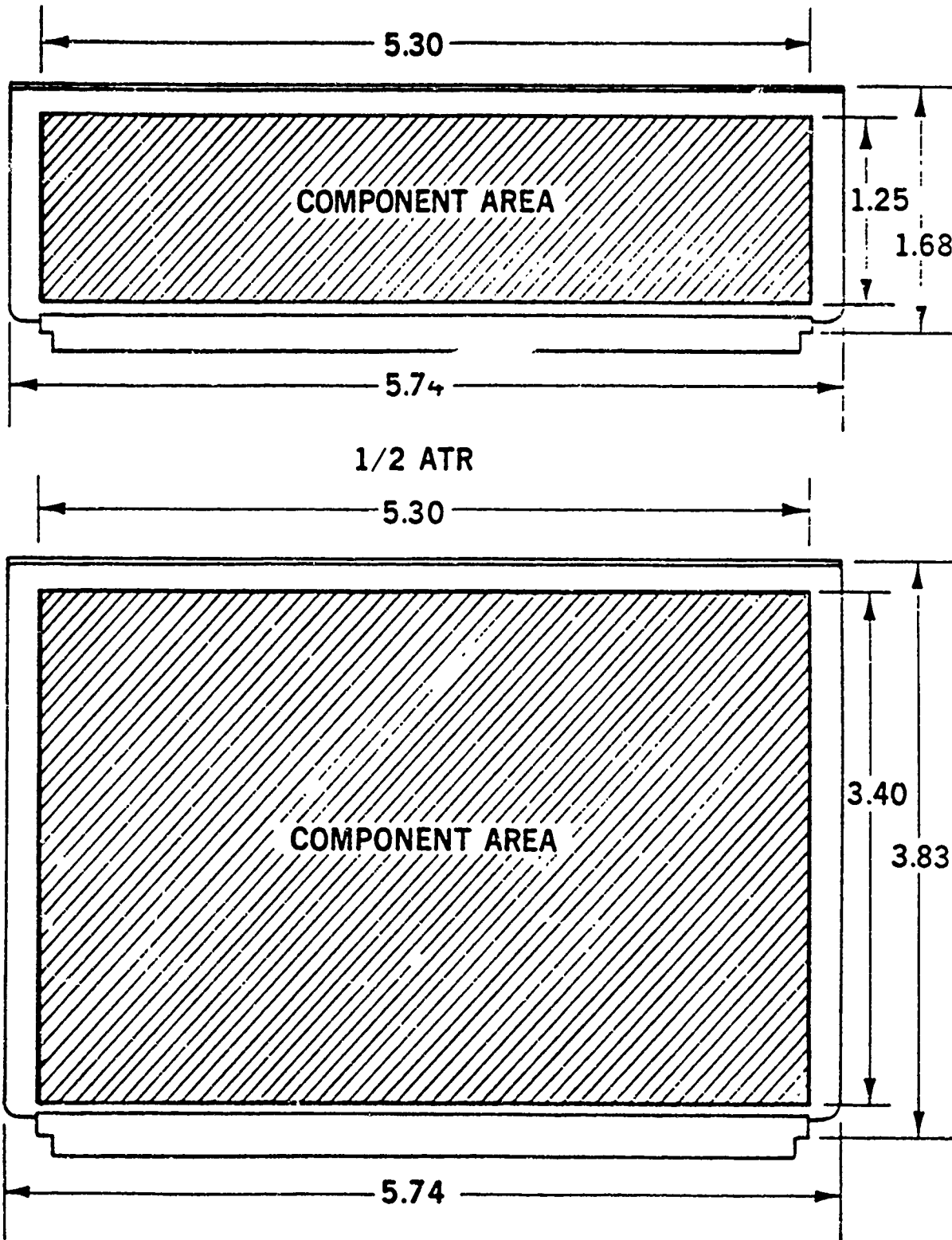


FIGURE 2

STANDARD ENCLOSURE CONCEPT

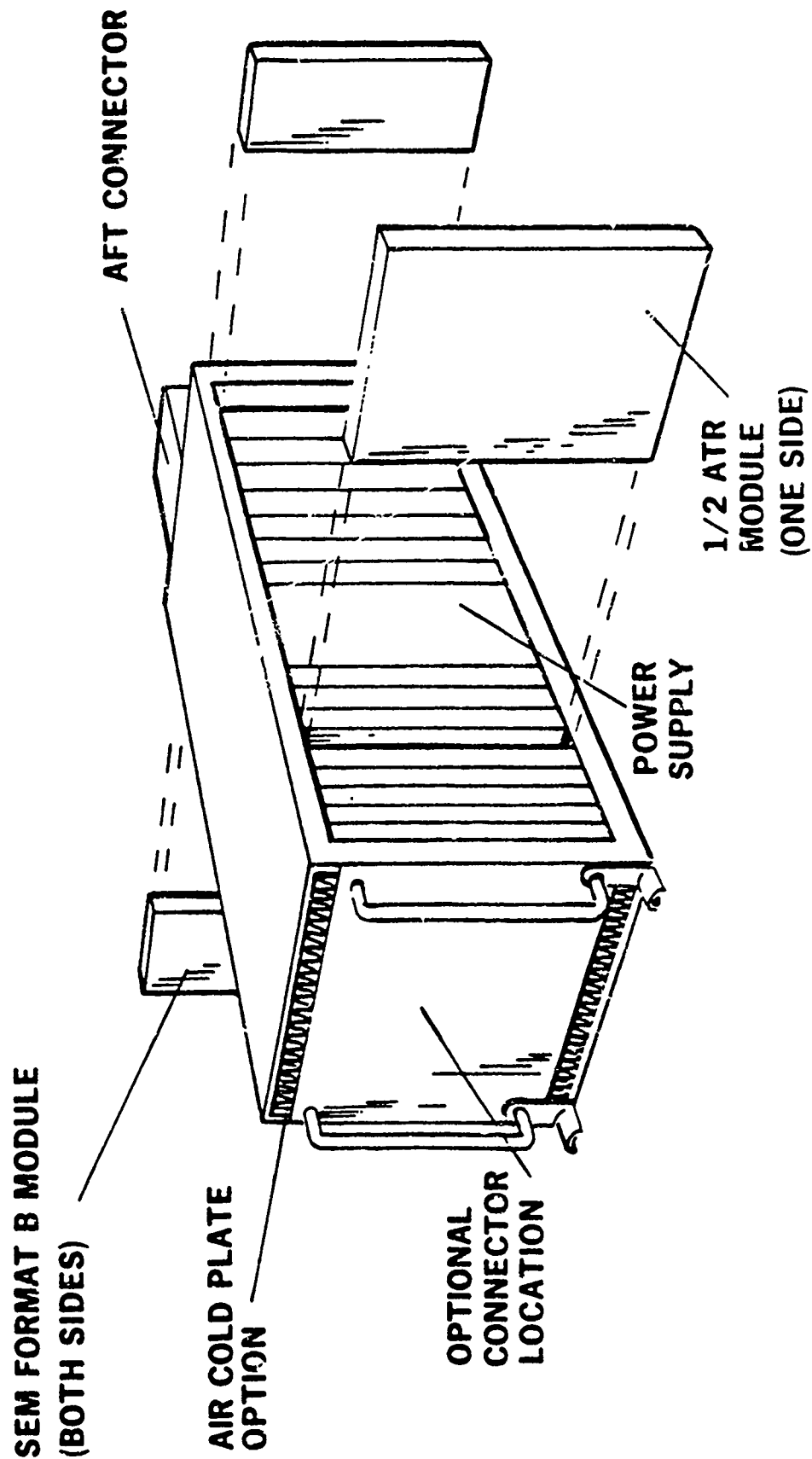
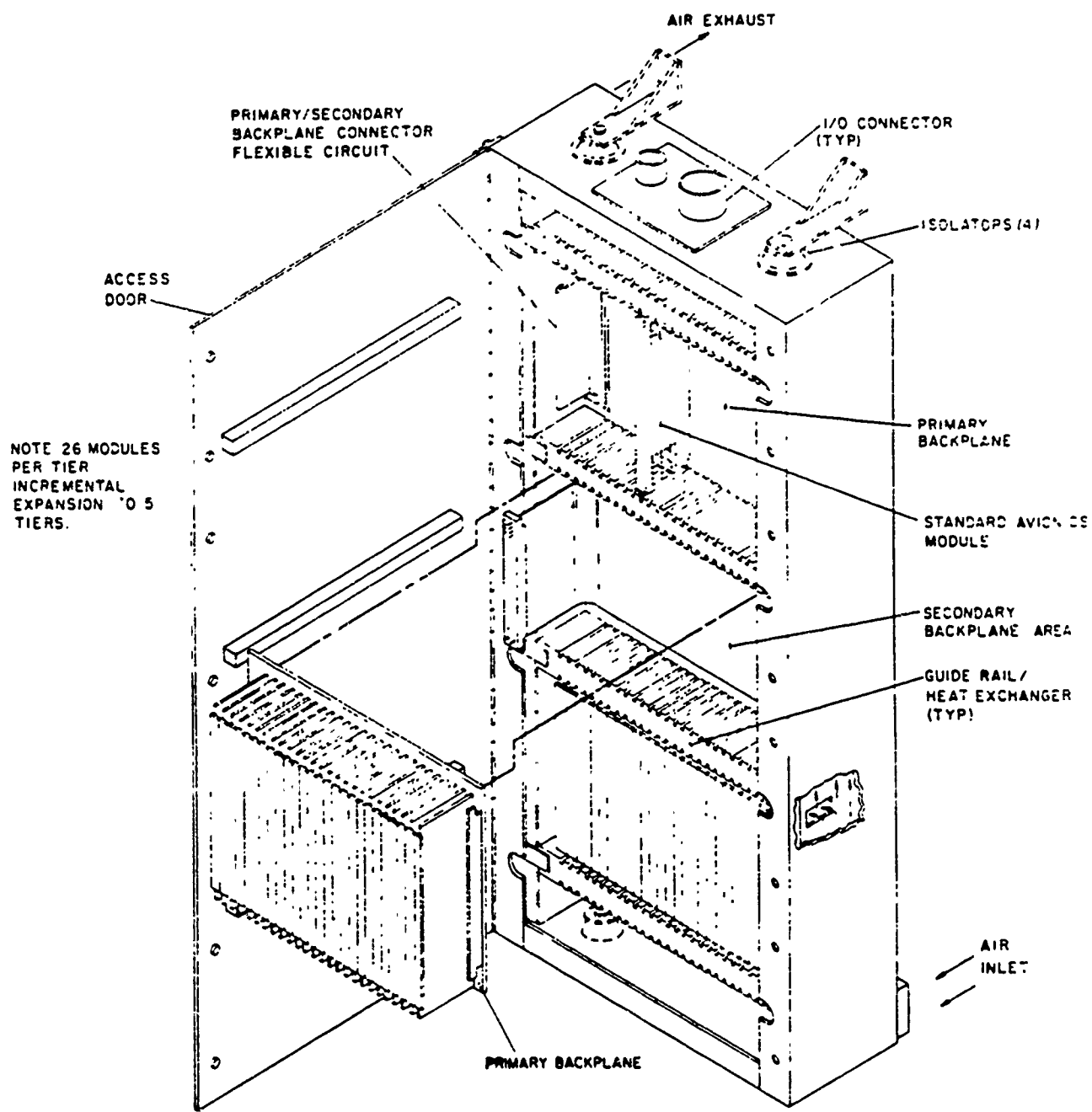


FIGURE 3



INTEGRATED RACK WITH AIR-COOLED GUIDE RAIL

FIGURE 4

AN INTRODUCTION TO THE AVIONICS INTEGRITY PROGRAM

James E. Verdier

Aeronautical Systems Division/ENASA
Wright-Patterson Air Force Base, Ohio (513) 255-6749

ABSTRACT

This paper describes the Avionics Integrity Program (AVIP) which will develop a MIL-STD and an implementation handbook for the development and operation of avionics systems. The objectives of the program are to reduce the cost of ownership and increase availability while meeting user requirements. The basic approach is presented and the outline of the MIL-STD and handbook are explained.

I. INTRODUCTION

Integrity, as defined within the context of this paper, means delivering what you said you would, or that what you deliver can be expected to perform as promised. There has long been concern over the integrity of Avionics systems and equipment. The primary reason for this concern is the high cost of ownership, the low availability of operational systems, and the fact that mission performance does not always meet expectations. Since individual avionics systems perform well in one or more of these areas of concern (i.e., life cycle cost, availability and performance), it is reasonable to assume that identifying why each system was successful in a particular area should, hopefully, enable us to identify optimum methods for developing avionics systems with integrity.

Strong evidence that integrity can be built into a system is demonstrated by the success achieved by the aircraft structure integrity program. This program was a deliberate attempt to identify the technology, organization and definition of tasks needed to develop and field a system with dependable structure. This program has evolved into a Military Standard (Mil-Std-1530) and is now a respected community reference. As a reference, it defines what must be done, and sometimes specifically how it is to be done, so that a structure with "integrity" evolves. The Avionics Integrity Program (AVIP) will be an attempt to evolve a similar reference for the avionics community.

II. APPROACH

After extensively reviewing the structural program, we decided to use it as a baseline for the AVIP. Avionics and structures are, of course, two different technical worlds, and an exact one-for-one transfer of structural approaches to avionics is not appropriate. However, we can identify and use the basic structure of the AVIP, tailoring it for the avionics community.

For example, the concept that certain activities must be accomplished in a time phased manner to insure integrity appears valid and can be applied.

Currently, we plan to develop AVIP for non-flight critical electronics first. A second effort will establish integrity programs for flight critical avionics.

There are two principle objectives for the AVIP. First, we hope to define the processes that will lead to higher levels of integrity and secondly, we will develop an implementation plan which will influence designers to make design choices that automatically insure systems with higher levels of integrity.

The plans are that the Air Force will develop draft documentation for the AVIP and then involve the avionics community, via forums, to critique and contribute to the finalization of these draft documents. The final output of these efforts will, hopefully, be a consensus by the avionics community of things that, if accomplished, will improve the integrity of avionics systems and equipments. Not all activities and events that go into programs are significant for inclusion in AVIP, so some philosophy as to what is important is in order.

First, AVIP cannot be just a technical document, telling what technical steps are best. It must deal with integrating technical steps and management concerns, recognizing the needs and limitations of schedules and resources.

Second, AVIP must aim at influencing the design choices made at all levels beginning in time with the early promotional conception, going through the life cycle.

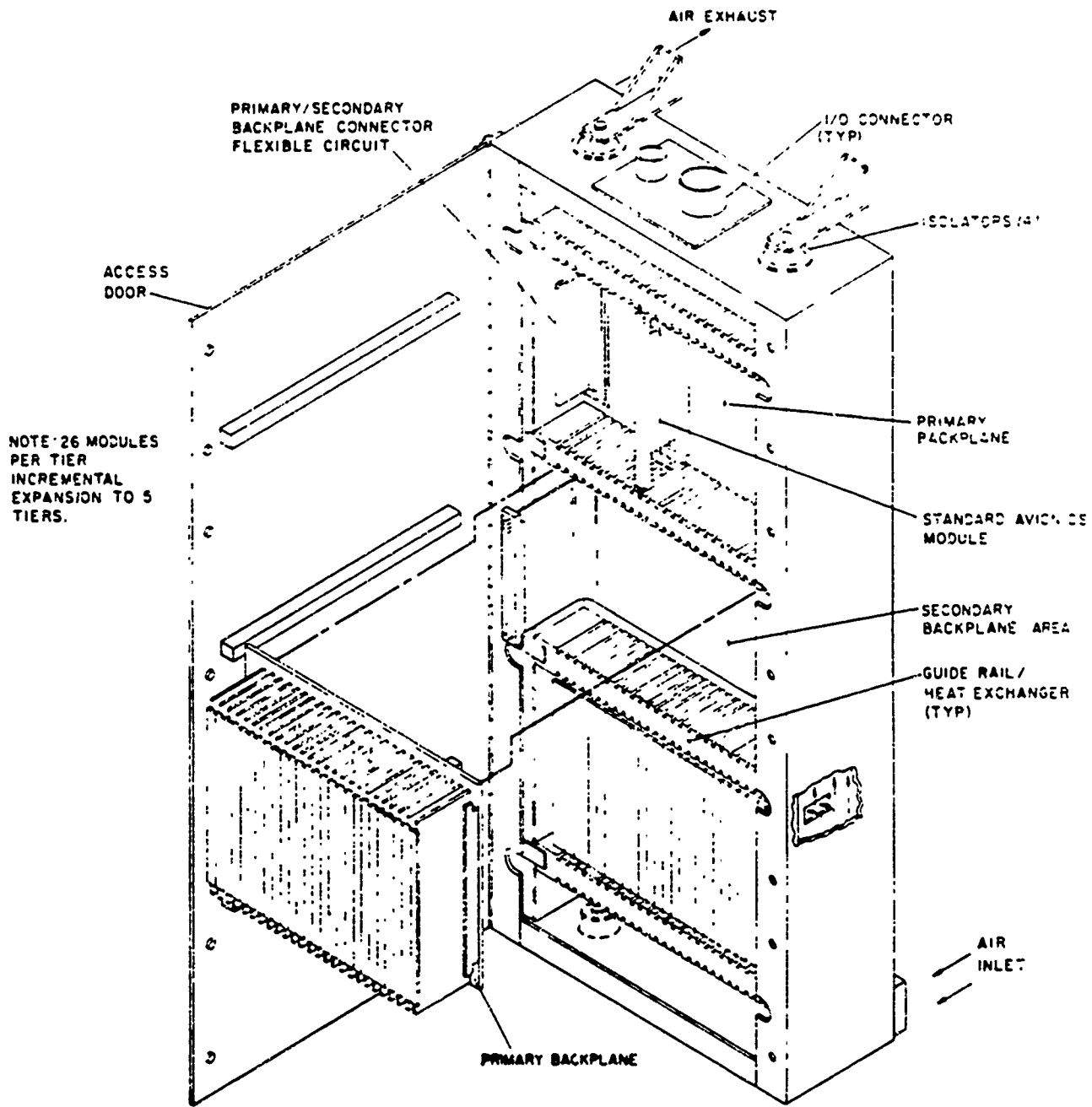
Third, the core document must communicate to a broad audience that includes people in sales promotion, management, budget planning, engineering, and logistics.

Fourth, AVIP must be an effective input to management's activity. If it cannot influence the planning of adequate resources to achieve verification or cannot assure proper timing of technical inputs, it cannot serve the accomplishment of integrity. Even where it is impossible to be part of the course of action it can cause an effective indicator of accomplishment, or the lack thereof. A primary service to management that AVIP provides is the forcing of the identification of risk and risk reduction action.

III. DOCUMENTATION

The planned documentation will consist of three volumes. Volume I will be the Executive Summary or top policy document. Volume II will be a draft Mil Standard. Volume III will be a draft handbook.

a. Volume I - Executive Summary. This document will consist of a few pages that outlines the overall AVIP requirements, establishes authority, identifies responsibilities and establishes program objectives.



INTEGRATED RACK WITH AIR-COOLED GUIDE RAIL

FIGURE 4

AN INTRODUCTION TO THE AVIONICS INTEGRITY PROGRAM

James E. Verdier

Aeronautical Systems Division/ENASA
Wright-Patterson Air Force Base, Ohio (513) 255-6749

ABSTRACT

This paper describes the Avionics Integrity Program (AVIP) which will develop a MIL-STD and an implementation handbook for the development and operation of avionics systems. The objectives of the program are to reduce the cost of ownership and increase availability while meeting user requirements. The basic approach is presented and the outline of the MIL-STD and handbook are explained.

I. INTRODUCTION

Integrity, as defined within the context of this paper, means delivering what you said you would, or that what you deliver can be expected to perform as promised. There has long been concern over the integrity of Avionics systems and equipment. The primary reason for this concern is the high cost of ownership, the low availability of operational systems, and the fact that mission performance does not always meet expectations. Since individual avionics systems perform well in one or more of these areas of concern (i.e., life cycle cost, availability and performance), it is reasonable to assume that identifying why each system was successful in a particular area should, hopefully, enable us to identify optimum methods for developing avionics systems with integrity.

Strong evidence that integrity can be built into a system is demonstrated by the success achieved by the aircraft structure integrity program. This program was a deliberate attempt to identify the technology, organization and definition of tasks needed to develop and field a system with dependable structure. This program has evolved into a Military Standard (Mil-Std-1530) and is now a respected community reference. As a reference, it defines what must be done, and sometimes specifically how it is to be done, so that a structure with "integrity" evolves. The Avionics Integrity Program (AVIP) will be an attempt to evolve a similar reference for the avionics community.

II. APPROACH

After extensively reviewing the structural program, we decided to use it as a baseline for the AVIP. Avionics and structures are, of course, two different technical worlds, and an exact one-for-one transfer of structural approaches to avionics is not appropriate. However, we can identify and use the basic structure of the AVIP, tailoring it for the avionics community.

For example, the concept that certain activities must be accomplished in a time phased manner to insure integrity appears valid and can be applied.

Currently, we plan to develop AVIP for non-flight critical electronics first. A second effort will establish integrity programs for flight critical avionics.

There are two principle objectives for the AVIP. First, we hope to define the processes that will lead to higher levels of integrity and secondly, we will develop an implementation plan which will influence designers to make design choices that automatically insure systems with higher levels of integrity.

The plans are that the Air Force will develop draft documentation for the AVIP and then involve the avionics community, via forums, to critique and contribute to the finalization of these draft documents. The final output of these efforts will, hopefully, be a consensus by the avionics community of things that, if accomplished, will improve the integrity of avionics systems and equipments. Not all activities and events that go into programs are significant for inclusion in AVIP, so some philosophy as to what is important is in order.

First, AVIP cannot be just a technical document, telling what technical steps are best. It must deal with integrating technical steps and management concerns, recognizing the needs and limitations of schedules and resources.

Second, AVIP must aim at influencing the design choices made at all levels beginning in time with the early promotional conception, going through the life cycle.

Third, the core document must communicate to a broad audience that includes people in sales promotion, management, budget planning, engineering, and logistics.

Fourth, AVIP must be an effective input to management's activity. If it cannot influence the planning of adequate resources to achieve verification or cannot assure proper timing of technical inputs, it cannot serve the accomplishment of integrity. Even where it is impossible to be part of the course of action it can cause an effective indicator of accomplishment, or the lack thereof. A primary service to management that AVIP provides is the forcing of the identification of risk and risk reduction action.

III. DOCUMENTATION

The planned documentation will consist of three volumes. Volume I will be the Executive Summary or top policy document. Volume II will be a draft Mil Standard. Volume III will be a draft handbook.

a. Volume I - Executive Summary. This document will consist of a few pages that outlines the overall AVIP requirements, establishes authority, identifies responsibilities and establishes program objectives.

b. Volume II - Military Standard - This document will identify the top level tasks and critical subtasks, and the attendant coordination and decision guidance. Current plans are that this document will parallel the structural integrity program's Mil Std. Like the structural program, the AVIP will be partitioned into five (5) major tasks. The five tasks are described below:

(1) Task I - Develop Overall System Requirements. This task will identify those analysis and trade studies that will be required to establish the system level requirements. The objective is to establish the architecture, balanced measurable performance parameters, and the operating environment. In Task I, the emphasis is on the "balanced and measurable." The basic idea is that attention must be given to the effect of increased performance requirements on cost. Balance of performance vs. support cost must be identified early, then achieved. Too often the verification of desired performance is ignored or left to later phases, only to find that there is no way to validate required performance. Other subtasks of Task I establish the basis of systems or equipment designs. One of these will be a definition of the architecture. Other subtasks that are accomplished by Task I are the identification of risk, risk management and the AVIP master plan.

(2) Task II - Develop Subsystem Performance and Configuration Criteria. Task I identifies requirements in the form of ideas on paper. Task II uses these as inputs to design, develop, and use the real world black boxes that make up systems or equipments. The general efforts under Task II includes trade studies, paper designs, subsystem partitioning, prototype testing, coordinating subsystem design integrations, design and construction of production tooling and processes, and some initial field installation and debugging. Out of the heart of this activity come the system performance that the customer is seeking. In Task II the process that is required to get to optimized systems through successive iterations will be accomplished.

(3) Task III - Validation. This task will include those subtasks required to verify, validate and demonstrate that the equipments and subsystems are meeting their functional and performance requirements. Subtasks will address resource provisions, define what and how verification, validation and tests are to be accomplished, the phasing of activities relative significant decisions, and the actions that are appropriate for the outcomes of validation.

(4) Task IV - Development of Operational Support Data. The title of this task is descriptive of what is involved. During operational use of the system, the performance will be monitored by an established system. The objective of such activities are to detect degradation and to guide the action when performance is below standard. Some appropriate data gathering that give useful performance measures is the significant foundation of such activity. This is an area that requires heavy involvement of Air Force logistics command and the developing agencies.

(5) Task V - The Management of Operational Life of the Equipment. The activities of Task IV designed the primary tool for this task. In addition to monitoring performance, this task will guide the technical tasks and processes that are needed to revise and retire the system or the equipment. In the revision of the systems, and in the application of planned

product improvement programs, the basic AVIP processes of Tasks I, II, III and IV will apply.

c. Volume III - Handbook - This document will provide detailed instructions and guidance on the tasks and subtasks identified in Volume II. It will be technically oriented and will contain lessons learned to support the reasoning, concept, rationale and requirements of the AVIP. The handbook will recommend detailed approaches, good design practices, analysis methods, test procedures and other details necessary to accomplish the various tasks and subtasks.

IV. BENEFITS

There are several benefits which the implementation of AVIP should achieve .

a. Establish a Large Experience Base. Each person has, from his experience, a concept of what-needs-to-be-done and how to go about it. Some have considerable experience and success while others have little. Since AVIP will be a community generated reference, it can take advantage of a larger experience base, and have some "lessons learned" to support the impacts of short cuts. An increased awareness of "what works" can evolve as a result of the AVIP coordination and improvement.

b. Keep Common Sense in Programs. If AVIP is applied, as a program requirement, it is likely to enforce common sense in the development process. This common sense tends to get lost in the complexity of many functions that are needed in program accomplishment. The pressures of optimism in program management tend to look for the answer that gets what is wanted, so that any one man's position tends to get lost, in favor of what is wanted. A larger community reference would be harder to ignore. The pressure of schedule and limited resources forces trades onto program management that result in compromise of original goals. It is not the function of AVIP to stop these compromises, but to identify that the compromise is happening and to identify impact, if possible.

c. Support of Planned Produce Improvement Philosophies. Current philosophies in procurement include the concept of planned product improvement. Most programs start with the promise and intent to get all that is wanted with the resources and time allowed. Very rarely does a program do that. Earlier money or time must be expanded, or some lesser performance is accepted. Although not the original concept, the planned produce improvement program (PPI) can be used to get back the lost performance, whether it be primary mission related or maintenance and reliability performance parameters. The role of AVIP, in this case, is to be the basic from which the need for the PPI would become evident.

d. Provide a Basis for Product Assurance Review. Recent changes in Air Force organizational structure have resulted in the establishment of a product assurance function whose function it is to alert the commander to situations where reliability maintainability and quality assurance

performance may be sacrificed. They are very interested in "what should be happening to get success." AVIP provides a reference, from which to investigate what a program "is" and "should be" doing.

e. Provide a "Common Language" for Both Industry and Government. Being jointly created it would have the foundation for a common way to think about what needs to be done, and then how to communicate about it. Here also management by identifying the exceptions is easier, and identifies where the issues are.

f. Be an Aid to Organizational Structuring. Management activities and organizational structuring are built around the tasks to be done. A commonality among programs would lead to similar structuring of organizations within all agencies.

g. Basis for Improving Cost Projections. Most cost project is critically founded on identifying the man hours that are likely to be needed to complete the tasks. Two parts of this projection are (1) that the tasks are properly and completely identified and (2) that good projections of man hours needed can be done. As tasks begin to become commonly thought of, and subdivided as in AVIP, there could become a larger comparable manhour experience base, which would make possible more accurate prediction and subsequent tracking of the application of manhours. The issues that are highlighted by AVIP will point to areas that may require risk reserve. Also risk identification and planning are direct output products of the AVIP activity.

V. SUMMARY

This paper has presented basic approach, philosophy and some proposed descriptions of the five tasks required to impliment AVIP. The challenge now is to define the subtasking, the coordination and decision guidance, as well as to develop detailed support data that AVIP should contain. Industry will be asked to participate in this challenge through an open forum to be held possibly as early as late 1983. Although the goals are high and somewhat general, hopefully, as we work together, a clear path to improved avionics integrity will evolve.

Biography

James E. Verdier

Mr. James E. Verdier is presently a Senior System Development Engineer in the Systems Branch of Avionics Engineering at Wright Patterson AFB. He is the team leader for the Avionics Integrity Program. Mr. Verdier received a degree in Industrial Engineering from Ohio State University. Mr. Verdier has 20 years of engineering experience at ASD, having worked on the Air Launched Cruise Missile, Ground Levelled Cruise Missile Maverick and avionics developments for the F-4, F-106 and F-102.

ADVANCED SYSTEMS ARCHITECTURE

SESSION CHAIRMAN: Major I. Caro
AFWAL/AAAF

MODERATOR: Colonel David J. Teal
F-16 Deputy System Program Director

AVIONIC ARCHITECTURE -- PAST & FUTURE

T. V. McTigue
McDonnell Aircraft Company
McDonnell Douglas Corporation
St. Louis, Missouri

In the space of two decades avionic systems at McDonnell Aircraft have progressed from an all analog, all vacuum tube, loosely integrated group of black'boxes as employed in early F-4 Phantoms to an almost all digital, no vacuum tube, mux interconnected, software controlled avionic weapon system such as exists in the F-18 Hornet. The future offers the challenge of complex strike fighter missions through high threat areas and poor weather with an increased demand on integration of hardware, software, sensors and the pilot. This paper reviews the architecture and achievements of the past, examines the alternatives of the future and provides insight and guidelines for the selection of next generation avionics architecture.

THOMAS V. MCTIGUE
BRANCH CHIEF
AVIONICS DIGITAL COMPUTERS AND SOFTWARE ENGINEERING
MCDONNELL AIRCRAFT COMPANY

Education:

B.S.E.E. (Magna Cum Laude)	St. Louis University	1959
M.S.E.E.	St. Louis University	1964

Experience:

Mr. McTigue has been engaged for the past 23 years in the system design, development, test, and integration of digital Avionics systems and software. This includes work with Avionics computers, mission software for operational flight and operational test programs, digital data link communication systems, and automatic test equipment. He was responsible for the management of the design and development of the airborne tactical operational flight programs for the U. S. Air Force F-15.

Presently, Mr. McTigue is responsible for the design, development, test, integration and documentation of the F/A-18 Mission Computer Operational Flight Program. He is also responsible for design and operation of the F/A-18 Software Development Facility and the real-time Software Test Facility.

Organizations:

Pi Mu Epsilon Mathematics Honor Society,
Eta Kappa Nu Electrical Engineering Honor Fraternity,
Sigma Xi Science Honor Society
American Institute of Aeronautics and Astronautics
AIAA Technical Committee on Computer Systems
Jane's Who's Who in Aviation and Aerospace

AD-P003 566

ELEMENTS FOR
SUCCESSFUL IMPLEMENTATION OF
COMPUTING STANDARDS

Gordon R. England
Director, Avionic Systems
GENERAL DYNAMICS CORPORATION
P. O. Box 748
Fort Worth, Texas 76101
(817) 734-4811, Ext. 2439

ABSTRACT

The F-16 avionics implements what is likely the broadest application of standards of any USAF weapon system. Standards available in 1976 were applied which consisted of the MIL-STD-1553B Multiplex Data Bus, JOVIAL J3B which was the de facto software HOL and precursor to JOVIAL J73 dialect and the MIL-STD-483/490 software documentation standard. These standards were instrumental in making the F-16 a very successful program. The F-16 avionic system is now being greatly expanded to accommodate advanced sensors and weapons currently in USAF funded development. Once again the F-16 is at the forefront in implementing the latest USAF standards. A key feature of the enhanced avionics is the application of JOVIAL J73 (MIL-STD-1589B) for all subsystems, the MIL-STD-1553B Multiplex Data Bus, the MIL-STD-1750A Computer Instruction Set Architecture and the MIL-STD-1760 Stores Interface. This paper describes the implementation of standards in both the current and the enhanced F-16 avionics.

TEXT

The F-16 is a fully operational, high performance aircraft that operates equally well in the air-superiority and ground-attack missions. This multirole capability is largely provided by the avionics system which is configured for all-weather air-to-surface and air-to-air weapon delivery and enroute operations with enhanced survivability. It consists of an integrated fire control system, an electronic warfare complement, and communication and identification equipment. The fire control system consists of a highly accurate inertial navigation set, a fire control computer, an integrated digital stores management set, a head-up display, a head-down display (for radar and electro-optical sensor or weapon video), and a fire control radar. The electronic warfare complement consists of a threat warning system, an active electronic countermeasures capability, and a chaff/flare dispenser set. The communication and identification equipment consist of UHF/VHF radios, an intercommunication set, air-to-surface IFF, TACAN, and ILS.

SYSTEM DESCRIPTION

System capability is realized largely through a federated network of real-time digital computers and small processors embedded in the avionic subsystems. Principal avionic elements which contain embedded computers are the Fire Control

"Copyright © 1982 by General Dynamics Corporation
All Rights Reserved"

Radar (FCR), the Inertial Navigation Set (INS), the Stores Management Set (SMS) and the Head-Up Display (HUD). The Radar/Electro-Optical (R/EO) Display and the Air Data Computer also contain limited processing capabilities. However, the Fire Control Computer (FCC) and its Operational Flight Program (OFP) serve as the focal point for integrating the various avionic elements into a total system. As such, the FCC OFP provides the key system management processing which causes the system to respond in a properly coordinated manner to pilot commands.

SYSTEM PARTITIONING

The F-16 avionic system features a partitioned MIL-STD-1553 DAIS type architecture as illustrated in Figure 1. The FCC functions as the bus controller. A backup bus controller is contained in the INS processor.

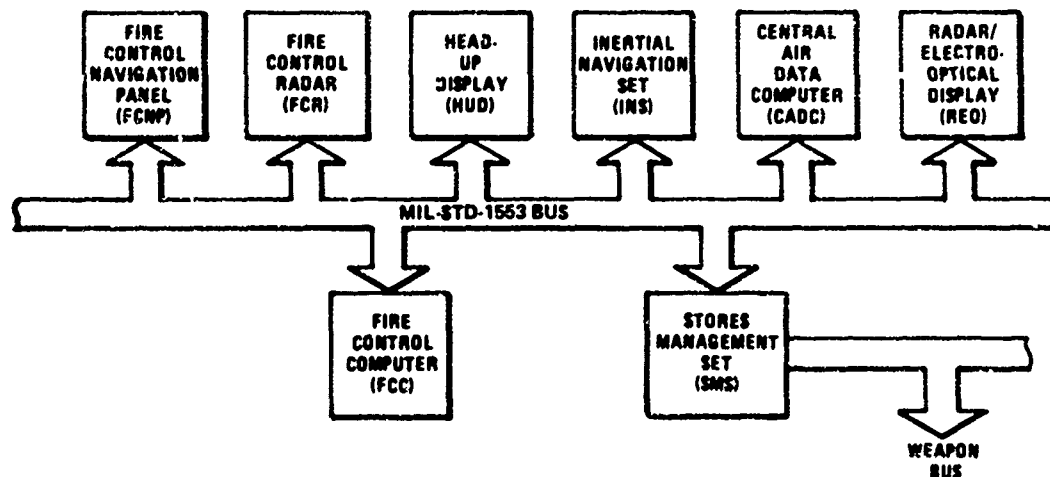


Figure 1. A MIL-STD-1553 DAIS-Type Architecture Implemented

The 1553 bus allowed the system to be partitioned into logical physical and functional units. Interfaces were established not only to minimize inter-sensor data flow but also to minimize control and timing interdependencies between avionic elements. This has resulted in a system with few critical intersensor dynamic loops and few unexpected sensor-to-sensor side effects. In other words, the system has simple interfaces, a simple structure, and a predictable operational behavior.

Since the sensors were partitioned so that system level integration is done within the FCC, there is considerable flexibility in mode redefinition, sensor replacement or addition and system enhancements. The software controlled MIL-STD-1553 multiplex data bus was an important factor in providing this capability.

Most significant are the program and contractual advantages provided by a smart implementation of 1553. With simple interfaces the subsystem procurement specifications were simple and well defined. Software elements were smaller, simpler and less interdependent. Thorough independent subsystem level check-outs were possible before full system integration. In addition, a large list of

development advantages were realized that were also instrumental in meeting schedule and reducing cost. Some of the more significant additional benefits realized from this bus architecture are as follows:

- o System integration largely reduced to software
- o Most changes possible without hardware impacts
- o Quick change turn-around
- o Direct simulation of subsystems on MUX
- o Simplified data recording, reduction and analysis
- o Ready integration of new subsystems

PROGRAMMING LANGUAGE

Approximately 90% of the FCC OFP is programmed in the JOVIAL J3B-2 Higher Order Language. Use of an HOL was based upon the expected benefits of:

1. Improved program clarity and readability
2. Improved programming reliability
3. Increased programmer productivity and
4. Easier use and enforcement of structured programming and other FCC OFP standard conventions.

The J3B-2 language was chosen specifically because it had all the features necessary for real-time avionic software. It also had maturity. Previous experience with the compiler indicated that it was capable of producing efficient code for avionic processing. In addition, military standard documentation existed for J3B-2. In 1976 the F-16 program was fortunate to benefit from the experience and application of J3B-2 by The Boeing Company on B-1.

HOL provided the expected benefits which are defined in Figure 2. These benefits were the basis of the later decision by General Dynamics to implement the J73 HOL (MIL-STD-1589B) for the enhanced avionics program. It was anticipated that as an offspring of JOVIAL J3B and J73/1, MIL-STD-1589B would have similar beneficial results.

DEVELOPMENT

- CLEAN ERROR-FREE CODING
- INCREASED PROGRAMMER CODING EFFICIENCY
- REDUCED PROGRAMMER TRAINING
- SELF-DOCUMENTING LISTINGS AND AUTOMATIC FLOW CHARTS

TRANSITION TO ALC

- HOL SOURCE EASY TO EXPLAIN/UNDERSTAND
- DOCUMENTATION CURRENT AND CORRECT

Figure 2. HOL Provided Many Direct Benefits

SOFTWARE DOCUMENTATION

Full MIL-STD-483/490 documentation was implemented for the FCC, RDR, and INS. This primarily consisted of a Development Specification (B5), a Product Specification (C5) and a comprehensive User's Manual. MIL-STD-490 like documentation was also utilized for support software and the SMS.

A significant aspect was the utilization of a development methodology in which the documentation was progressively constructed and served as an in-line design aid rather than an after-the-fact documentation activity. As indicated in Figure 3, this documentation was embedded into the design process. This approach provided the following development benefits:

- o Improved control of development process
- o Better Air Force visibility of product and status
- o Current, accurate knowledge of software configuration
- o Enhanced capability to identify and implement desired changes

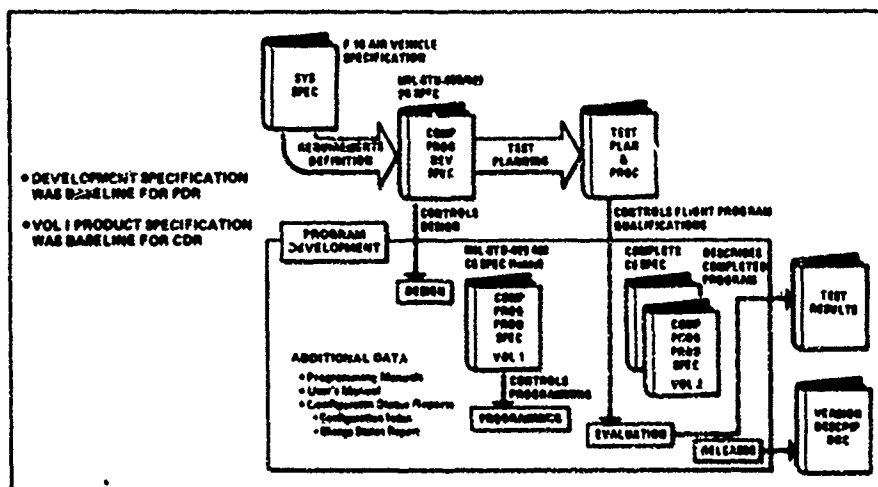


Figure 3. MIL-Standard Documentation Embedded into Design Process

ENHANCED F-16 AVIONICS (MSIP)

This F-16 avionic system capability is now being greatly expanded to accommodate advanced sensors and weapons currently in USAF funded development, as well as maintaining design awareness for even more advanced technologies now in view. This expanded capability will: (1) provide improved air superiority with the addition of Beyond Visual Range air-air missiles capability, (2) increase anti-armor effectiveness through the capability for low-level night penetration, and (3) accommodate extensive new sensors and weapons under development. This improvement program is known as the Multinational Staged Improvement Program (MSIP).

ENHANCED SYSTEM ARCHITECTURE

The advanced avionic system architecture (Figure 4) is an expansion of the existing F-16 avionic architecture. A new dual-redundant display multiplex bus is added to the existing dual-redundant avionics multiplex bus. This expanded architecture possesses the capacity and flexibility to accommodate the integration of the many new sensors and weapons while retaining the desired and proven qualities of the existing architecture, such as simple, non-time-critical interfaces and distributed processing.

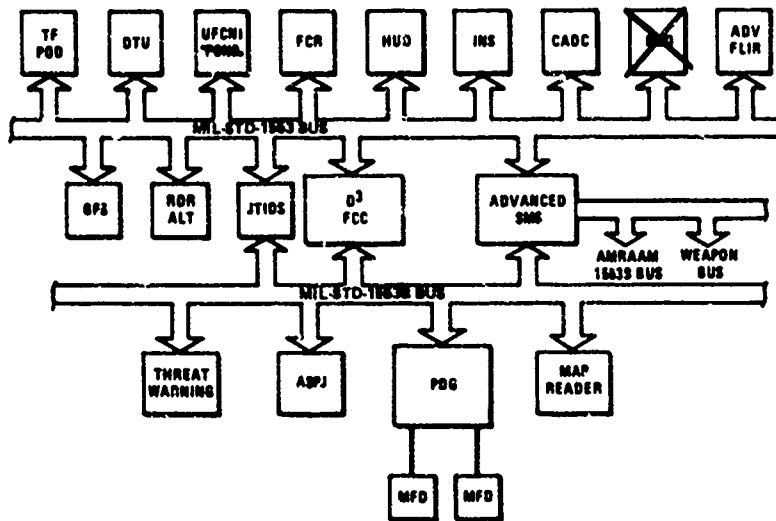


Figure 4. System Architecture

As in the existing F-16 architecture, the fire control computing element is the system's integrator; however, in the advanced systems architecture, the computing element is a double-speed, double-memory, double mux bus interface (D³) computer. This new D³ FCC interfaces with and is the primary bus controller for both avionic and display. A dual protocol feature is provided so that the D³ FCC can communicate with both 1553 and 1553B multiplex bus terminals on either of two multiplex buses. This is a very important feature in that it allows the co-existence of current F-16 avionic subsystems (1553 interfaces) and advanced sensors/weapons (1553B interfaces) on both buses.

The stores management set Advanced Central Interface Unit (ACIU) has three multiplex bus ports. Two ports interface with the two avionic multiplex buses

and the other interfaces with the special weapons multiplex bus. The ACIU, also dual protocol (1553, 1553B) capable, provides the backup bus control for the avionic buses. It is also three protocol capable (1553/1553B/Weapons MUX) and acts as the primary bus controller on the weapons bus. The weapons bus is available at all nine of the F-16 weapons store stations to provide both 1553 and 1553B interface capability for advanced weapons and sensors. Digital data can easily be passed between the avionic buses by either the D³ FCC or the ACIU. Also, the ACIU can easily pass data between these buses and the weapons bus. This results in a highly flexible digital communications network formed by the advanced avionics system architecture bus structure.

This F-16 advanced system architecture provides the foundation for a system with several major advantages. First, the F-16 architecture minimizes development costs, risks and maintenance costs. This is realized by careful partitioning which facilitates procurement of reliable, on-cost, on-schedule subsystems, thereby minimizing software difficulties during integration of the avionic elements into a reliable, operationally sound system.

Second, this architecture design minimizes avionic system vulnerability to faults and failures, and raises the level of fault tolerance. The level of tolerance required by the total system and its various avionic functions were used as a guide to define each subsystem's self-test structure.

Third, and finally, this architecture easily supports the capacity of the system to grow or be modified when required. This capability is of vital concern during the operational phase of a modern aircraft life-span, since recent history shows a user desire to modify and update digital avionic systems annually.

KEY USAF STANDARDS INCORPORATED

Rational standardization is an integral part of General Dynamics' design philosophy. Accordingly, the F-16 advanced avionic design features use of JOVIAL J73 High Order Language (MIL-STD-1589B), the MIL-STD-1553B Multiplex Data Bus, the MIL-STD-1750A Computer Instruction Set Architecture, and the MIL-STD 1760 Stores Interface.

Similarly, software and hardware commonality among avionic subsystems is a top priority goal of the MSIP Program, and use of the standards provides an effective means to that end. In fact, application of the standards is being extended in two key areas. First, a common interface between MIL-STD-1750A processors and MIL-STD-1553 has been defined and specified. This common interface definition will result in a uniform set of commands, communication protocol, and software modules required for multiplex bus communication. The application and development benefits of standardization are characterized in Figure 5.

Second, a complete, integrated J73 support software system is also being procured. This integrated system will be common for all MSIP software and is being offered for a wide range of USAF sensors and weapons. Design of this common support software system includes HGL debug features, MIL-STD documentation aids, and other features to aid in development and laboratory testing of flight programs. These design features, coupled with MIL-STD-1750A processors allow development of a common set of real-time software debug commands and procedures in the laboratory hot bench stations. The ultimate beneficiary of the common support package will be the receiving Air Logistics Command, where the benefits will accrue for decades.

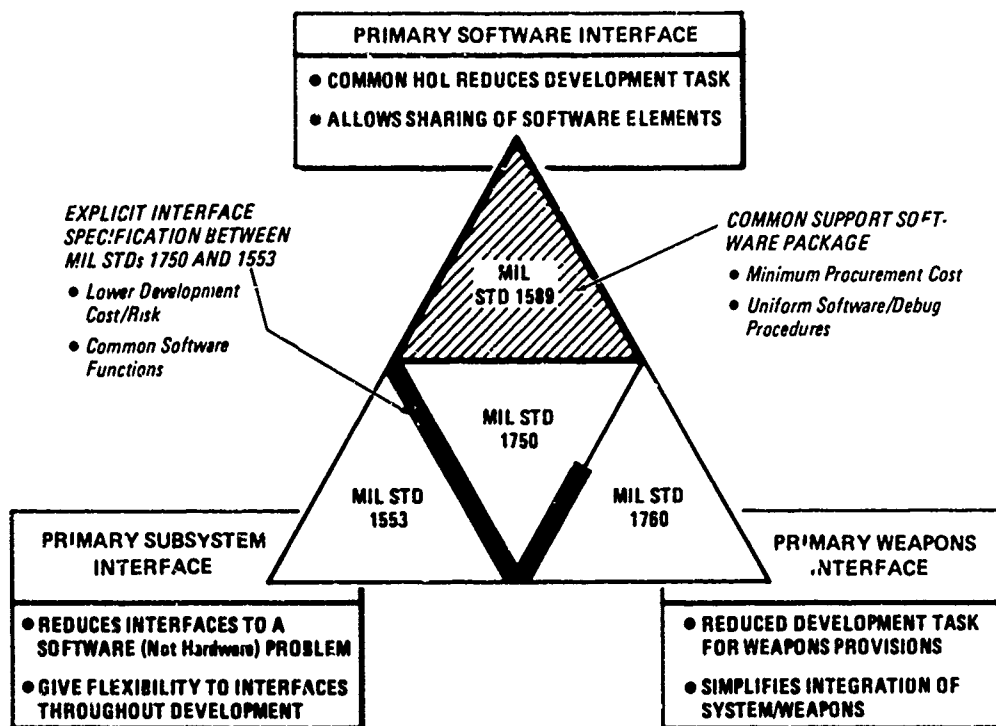


Figure 5. Application and Benefits of Standardization

Both avionic suites feature use of a MIL-STD-1553 bus network. Not only has the MIL-STD-1553 multiplex data bus proven to be an extremely reliable, testable interface, but it essentially reduces integration to a software-only issue. This means quick interface changes without hardware impacts; a significant plus in a development program.

Similarly, use of the MIL-STD-1750A Instruction Set Architecture provides three significant software benefits:

- (1) An identical compiler for all subsystems
- (2) Enables earlier software development
- (3) Provides a standard interface between processor and multiplex functional elements.

However, implementation of 1750A is a hardware issue and it is therefore being applied in a two-phase program. In Phase I of this program, a low power, single card, high speed processor is being developed under F-16 SPO sponsorship to meet near term F-16 schedules. The specification for this 1750A processor is tailored to allow the maximum applicability to the broad spectrum of emerging USAF sensors and weapons. In parallel with this Phase I development, the various MSIP II subsystems are being developed using the Z8002 processor. During Phase II, the 1750A processor will replace the Z8002 processor in production.

The application details for implementation of the various standards into the enhanced F-16 avionics are briefly discussed in the following sections.

APPLICATION OF MIL-STD-1760

MIL-STD-1760 addresses the electrical and data interconnection system between aircraft and stores. It has the goal of significantly reducing the problem of aircraft/store integration by requiring one standard electrical interconnection system for all aircraft and stores. Application of the standard is expected to reduce and stabilize the number and variety of signals required at the aircraft/store interface, minimize the impact of new stores on future stores management systems, and increase store interoperability among the services, within NATO, and with other allies.

The standard will ultimately include three interface areas: electrical, logical, and physical. The original release of the standard in July, 1981, defined the first of these, the electrical interface signal set between the stores and aircraft. The second interface area, logical (including communications architecture and data transfer protocol) is to be added in the near future. The physical aspect of the standard interface which includes connectors, umbilical cables, and common launchers is being addressed now and will be added to the standard in the near future.

The F-16 aircraft design anticipated release of the standard by incorporating wiring provisions in airplanes that are now coming off the production line. These early provisions will give the F-16 MSIP airplanes all of the key capabilities defined by the currently released standard. This consists of required provisions for AC and DC power, dual MIL-STD-1553 multiplex buses, release consent, video lines, radio frequency lines, audio, store/airplane interlocks and multiplex bus address lines. The F-16 will also provide auxiliary power as defined by MIL-STD-1760 at selected stations where high power usage is anticipated.

The first store incorporating the MIL-STD-1760 interface on the F-16 will be the AMRAAM missile which will be added to the existing AIM-9 stations.

APPLICATION OF MIL-STD-1589B

JOVIAL J73 as described by MIL-STD-1589B is the current Air Force standard higher-order language for embedded computer applications software. JOVIAL is a block-structured, strong type-checking, procedure-oriented language. This version combines the features of many earlier dialects of the language, e.g., J3, J3B, J4, and J73/I. General Dynamics is implementing all of the flight programs on the F-16 MSIP avionics in JOVIAL J73. These OFPs include the Fire Control Computer, the Data Transfer Unit, the Stores Management Set, the Multi-function Display Set and the Up-Front Control processor. An Integrated JOVIAL J73 Support Software System (ISSS) consisting of three separate computer programs (a compiler, assembler, and linker) operating in a common IBM 370 type host environment is being developed to support this use of JOVIAL J73.

The host environment forms the major interface between the programs and the user, and provides the means for running the programs and supplying inputs and outputs. Figure 6 illustrates the major interfaces within the system.

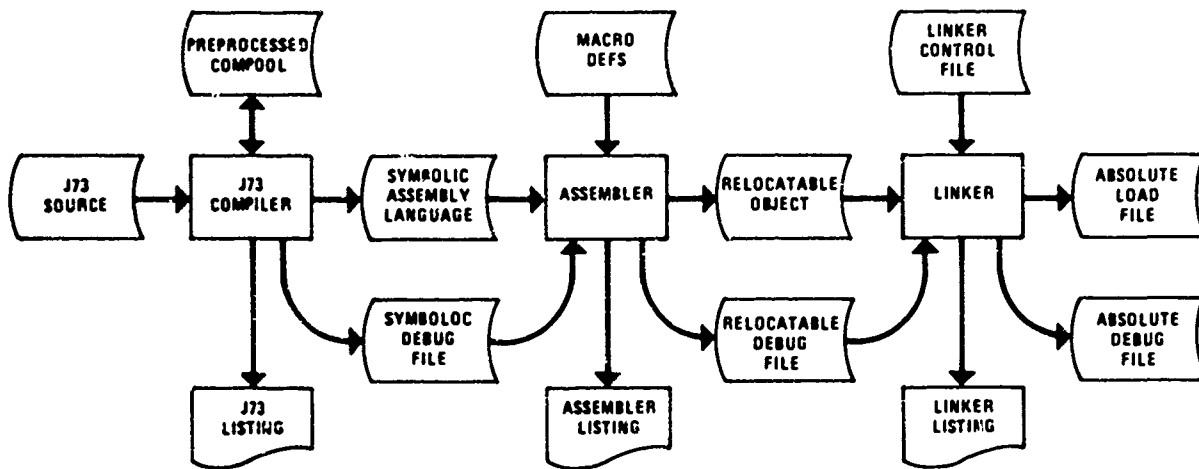


Figure 6. JOVIAL Integrated Support Software System Interface Diagram

General features of the JOVIAL ISSS are as follows:

- o Portability. Host dependent portions of the system are being minimized and isolated to allow the system to be rehosted with a minimum of effort.
- o Retargetability. Target dependent features of the system are parameterized and isolated to better facilitate changes in the target computer or to totally retarget the system.
- o Appropriateness. The ISSS is being specifically designed to support the performance requirements associated with real-time avionics software.
- o Maintainability. The ISSS will be maintainable in source form by organizations other than the developer.

General Dynamics is working with the USAF to extend this common support software package to encompass all F-16 avionics, including GFE; multiple users results in multiple benefits. Cooperative application will result in faster maturing of the support package and will provide a single, unified, support software package at the ALC.

APPLICATION OF MIL-STD-1553B

As previously discussed, MIL-STD-1553B allows a system architecture that provides technical, contractual and growth benefits (reference section on Enhanced System Architecture). However, to provide as much design commonality as possible between subsystems and to minimize the risks inherent in designing new hardware, General Dynamics has also defined a common architecture for the hardware/software interface of the communication channel. On the F-16 this means potential sharing of communications software among four different subsystems, as well as assuring interoperability of terminals and controllers.

A basic precept of the F-16 approach is that an I/O channel should not burden the host processor with the mundane tasks of transferring data to or from memory. The power of the host processor is always present to handle exceptional

situations, while the general nature of the interface assures flexibility for future growth. This architecture defines both controller and terminal operation within the context of MIL-STD-1553.

An I/O channel operating as a bus controller executes a channel program stored in memory as a set of linked tables. Once provided with the start address and enabled by the processor, the I/O controller executes its program independent of the processor. Each instruction in the channel program is four words long and is capable of specifying a complete MIL-STD-1553 data transaction or channel command. The channel always interrupts the processor for error handling and places the reason for the interrupt in a set of processor accessible registers.

The I/O channel operating in the terminal mode provides for the management of consistent data sets, time tagging of transmissions, selective interrupts to the processor, and for message queuing to insure that data sets do not get lost. The subaddress vector table located in the main computer memory supports the terminal mode of the I/O channel. Using the subaddress and transmit/receive bit from the received command word, the I/O channel references a four word block containing all the information necessary to locate the data buffers and complete the data transaction.

APPLICATION OF MIL-STD-1750A

The Air Force has adopted a standard instruction set for sixteen-bit computers used in airborne applications. When implemented using standard components, the central processing unit has proven too large and power inefficient for embedded computer applications. A new current state of the art technology design is therefore being developed for the F-16 MSIP avionics.

By establishing MIL-STD-1750A as the standard sixteen-bit computer instruction set architecture, the Air Force intends to reduce the life cycle costs of digital avionics. The expected benefits of this standard include common support tools such as compilers, simulators, and debug packages for multiple computers within a weapon system. In order to realize these benefits on a major program, the F-16 System Project Office contracted with General Dynamics to procure a small, low-power, cost effective implementation of MIL-STD-1750A for use on F-16 MSIP.

An instruction set architecture as described in MIL-STD-1750A includes not only the instruction set, but also the interrupts, fault handling provisions, extended memory addressing, and protection mechanisms as viewed by the machine language programmer. In this design, all features of the standard are partitioned into three sets of requirements: (1) the Central Processing Unit (CPU) incorporating all mandatory requirements for the F-16; (2) the Memory Management Unit (MMU) combining the optional features of extended memory addressing and operating system paging protection; and (3) the Block Protect Unit (BPU) holding the memory write-protection maps. Other optional features within the standard are left to the embedded computer system designer where they may be incorporated easily with standard digital components.

SUMMARY

General Dynamics has had considerable experience in implementing the key USAF Standards into a major weapon system. The conclusion from this experience is that a smart application of these Standards reduces development task, schedule

and cost. If reasonable Standards did not exist then contractors would select their own to achieve commonality of design in interface protocols, language and instruction sets. Experience indicates that while Standards of themselves do not make a successful program, they do make a successful program possible.

GORDON R. ENGLAND

BSEE - University of Maryland
MBA - Texas Christian University

Mr. England is Director of the Avionic Systems Department with management responsibility for systems design, development, integration and installation of avionics for all Fort Worth Division programs of General Dynamics including determination of requirements and architectures for advanced systems.

Prior to this assignment, Mr. England was the Avionics Project Manager for F-16 Avionics with direct technical responsibility for the development, flight test, and operation of delivered F-16 avionic systems. He has a broad background in the development of the F-111 weapon systems, including supervisory responsibility for both the software mechanization of the F-111F and FB-111A and for development of various hardware subsystems for the FB-111A and F-111A/E.

Mr. England also has an extensive background beyond fire-control related activities. He was Program Manager for the E2C Passive Detection System and was lead engineer in the development of the GEMINI Inertial Guidance System.

Mr. England is an active community and professional leader. He has been recognized by membership in engineering, business, and leadership honor societies and has presented and published numerous technical and managerial articles and has served on a wide range of panels, including National Research Council/Air Force Studies Board, Aeronautics and Astronautics, Industry Week, NAECON, Advanced Management Journal, DoD Standardization Seminar, AFSC Standardization Conference, AIAA Computers and Aerospace III, and many others.

ABSTRACT

B-1 AVIONICS APPLICATIONS OF MILITARY STANDARDS

By

L. M. Carrier, Director - Avionics Systems, Rockwell International
and G. A. Kinstler, Supervisor - Avionics Integration and System
Requirements, Rockwell International.

Military standards applied to the B-1B Avionics Program are discussed. Emphasis is placed on aircraft electronics systems design and interface with the aircraft. Subsystems discussed include the central integrated test system (CITS), electrical multiplex (EMUX), controls and displays, weapons interfaces, and communications and traffic control. Standards applied to offensive and defensive avionics are also summarized. Program constraints and rationale pertinent to the partial or deferred application of some standards are discussed. The extent currently applied and options planned for future incorporation in these areas (e.g., MIL-STD-1589B, -1750A, and -1760) are presented.

**Standards Application to B-1B Avionics Program - Boeing Military
Airplane Company (BMAC), H. L. ERNST**

This paper covers the B-1B Offensive Avionics Program as related to the recently developed USAF Avionics Standards. These USAF Avionics Standards include MIL-STD-1553B Data Bus System, MIL-STD-1589B High Order Language (HOL), and the MIL-STD-1750A Computer Instruction Set Architecture (ISA). The B-1B Avionics System is described covering the system architecture, major subsystem, and equipment. The recently conducted B-1B Standards (MIL-STD-1589B and MIL-STD-1750A) Program is treated. This treatment defines the tasks, summary of results, and conclusions.

The B-1B Program action taken subsequent to the Standards Program wrap-up is discussed. Finally, future B-1B Program Planning as to application of the Military Avionics Standards is discussed.

Bio-Sketch - H. L. Ernst, Boeing Military Airplane Company

Mr. Ernst was Program Manager of the BMAC B-1B Avionics Parallel Standards Program from January-June, 1982. He is currently Chief, B-1B Avionics Technical Staff. After receiving his BSEE degree from the University of Kentucky, and graduate work at Notre Dame, Mr. Ernst joined The Boeing Company in 1953. He joined the Boeing Military Airplane Company in 1972, after 19 years on transport programs (KC-135, 707, 727, SST, and NAP).

He was Systems Technology Chief on the AMST Prototype (YC-14) Program and Avionics Technology Chief on both the AMST (C-14) and C-X studies and proposals.

THE APPLICATION OF STANDARDS TO THE
TDY-750 (TIGERSHARK) MISSION COMPUTER

David W. Geyer

Teledyne Systems Company
19601 Nordhoff Street
Northridge, California 91324
213-886-2211

→ The design objective of the Teledyne TDY-750 computer series was to be completely compliant with the three Air Force Standards applying to aircraft mission computers: MIL-STD-1750A, MIL-STD-1553B, and MIL-STD-1589B (Jovial J73). A secondary design objective was to build a machine substantially lighter and faster than its competition while using standard discrete parts available from multiple sources. All objectives were met and, on 8/31/82, the first production prototype machine completed acceptance testing ahead of schedule. The 15 pound machine includes 64K memory, housing, power supply, and a microprogrammable 1553B Bus Controller/Remote Terminal. Formal SEAFAC testing to Change Notice 1 was successfully completed 8/25/82. A powerful software development system, including a real time Jovial Symbolic Debugger, complete the total hardware/software system. ←

INTRODUCTION

The Teledyne Systems Company has a long history of supplying digital processing systems for various militarized activities. Teledyne participation in the MIL-STD-1750 Users Group began during calendar years 1978 and 1979. During 1979, the decision was made by Teledyne to start an in-house development effort aimed at completing a functional breadboard by mid-1981. Groundrules for the development effort were as follows:

- Incorporate all MIL-STD-1750A options.
- Be fully compliant with MIL-STD-1750A Change Notice 1 including memory management unit.
- Optimize the CPU microcode for the MIL-STD-1750A instruction set.
- Design for the highest possible thruput to chip count ratio.
- The only parts that could be used in the breadboard would be those where there was a high probability that discrete parts, with MIL-STD-883B quality screens and/or MIL-STD-38510 specifications would be available in calendar year 1982.
- Recognizing that the VHSIC programs and/or chip sets would eventually become available, design the CPU from the beginning with provisions for multi-processing and redundancy.

- o Allow for a modular input/output system that could accept not only MIL-STD-1553B but analog and discrete signals as well.
- o Allow the use of modular memories of various technologies and timing requirements on a plug-in interchangeable basis.
- o Allow for provisions for cache "look ahead" should future speed enhancements be necessary.

Concept design studies were completed in the Fall of 1979, and in the beginning of 1980 a team was brought together to begin the detailed design and fabrication of the breadboard machine. This team not only reduced the machine to practice, but also tracked and incorporated the various interpretations of MIL-STD-1750 - drawing on the experience of the other developers reported during the MIL-STD-1750 Users Group meetings.

Power on was first applied to a complete breadboard in early 1981, and by the middle of 1981 the basic design of the CPU/memory/resource controller, and input/output was essentially complete.

In July 1981, we requested SEAFAC to take our breadboard for compliance testing. Permission was received, and in September 1981 certification testing began. It became immediately apparent that while our breadboard incorporated Change Notice 1, the certification software was still for the basic 1750 specification. Working across a weekend, we worked with SEAFAC to update their acceptance test software to Change Notice 1, and also to change our microcode in the two or three areas where we had interpreted the specification differently than the acceptance test software tested the requirement.

At the same time the breadboard was being completed, the Northrop Corporation began their competition for a flight computer for the Tigershark program. This lightweight high performance aircraft needed a MIL-STD-1750 computer that was small, low power, lightweight, and low cost - just the marketplace for which we had designed our breadboard machine.

We entered into a proposal phase and were subsequently announced the winners of the competition. By this time, we had also determined that we would have a complete series of MIL-STD-1750 airborne computers which we named the "TDY-750" series.

TDY-750 COMPUTER SERIES

Figure 1 shows a picture of the first production prototype of the TDY-750 computer. This computer successfully completed its acceptance test in late August 1982 and is now in operation at Northrop Aircraft. From go-ahead to completion of acceptance testing took approximately 11 months.

The machine was designed for an aircraft environment when no external cooling air was provided. The computer contains an exhaust blower and integral heat exchangers to provide for adequate thermal cooling (see Figure 2). Although the machine is very compact, we provided significant amounts of internal heat paths so that all junction temperatures of all

active integrated circuits/transistors are kept below 110°C at an airflow inlet temperature of 71°C. This is the absolute key to providing reliable operation. In a normal laboratory environment, the machine is only slightly warm to the touch. The air plenums are designed so that the air does not contact any active or passive electronic parts and/or electrical connections.

The machine was designed to provide for memory growth as can be seen from Figure 3. Figure 3 also shows how the higher power dissipating elements of the machine (the power supply and the CPU/resource controller) were mounted on the outboard sides of the computer so as to provide large flat contact with the cooling air streams. This figure also shows the battery that is utilized to transform our low power CMOS RAM memory into a non-volatile memory.

The machine is very easily disassembled into its major components (see Figure 4). Each individual circuit card is easily tested prior to inserting the assemblies together into the motherboard for final assembly and test. This Figure also shows that, while the packaging is reasonably dense, all size, weight, and power objectives were met using only discrete parts and standard packaging technologies.

Table 1 shows the overview of the major performance characteristics of the TDY-750 computer. Both the specification limits and the nominal values are shown so that an appropriate comparison can be made with other MIL-STD-1750A machines.

The predicted inflight reliability of our machine is quite high - both due to its low parts count, and our very conservative thermal design. We have warranted the machine to have an "in use" reliability of greater than 2000 hours - something we feel is unusual in the industry.

The major environmental specifications for the computer are shown in Table 2. These environmental specifications are stringent enough so that the basic machine can be used, with little (or no) requalification testing, on almost any aircraft.

One of the major design objectives of our TDY-750 computer series was to build a "modular" machine. Although the 1750 instruction set architecture completely defines the arithmetic logic of a standard computer, almost every application requires tailoring of memory and/or input/output. To minimize non-recurring costs, we wanted to create an architecture that will allow users to pick and choose from modules designed on other programs and to be able to restrict their design costs to only new modular units.

Figure 5 shows the basic architecture of the machine. Note that the machine is built around a very high speed internal parallel buss. We named this internal buss the "T-Buss". All modular elements run off of the T-Buss. Note the potential for additional 1750 CPUs to plug into the single T-Buss. Note also the provisions for a removable program monitor interface unit. The program monitor interface unit is key to be able to rapidly write and debug software for our 1750 computer series.

Figure 6 shows the basic concept of the T-Buss. Each of the modules on the buss are hard wired to the resource controller. The resource controller allocates buss resources on a first in/hard wired priority basis.

The present buss runs at a 25 MHz rate. Every 40 nsec, the resource controller polls the list of potential users to see who gains access to the buss in the next buss access cycle. This is a very simple concept and allows DMA channels to be interleaved with memory modules to be interleaved with multiple CPUs with minimum buss contention. In the present design, less than 5% buss contention time is achieved with a single CPU, high speed memory, and a 1553 data channel. Although we do not have the exact numerical results (and they are somewhat I/O/program dependent), buss contention (at the present clock rate) would not become to be significant until more than three processors existed on the buss simultaneously with high speed (greater than 500K words per second) I/O channels.

After designing the basic multi-ported T-Buss, the design flexibility inherent in such a concept makes it extremely easy to adapt from a single processor to a multi-processor configuration. Each individual module on the buss simply needs to be designed so that it can take a varying amount of time in between the module's buss request, and the buss allocation given. Any number of synchronous modules can thus be plugged into the buss - including central processing units.

Dual processing/multi-processing/fault tolerant processing becomes extremely easy with such an architecture. Note, were multiple CPUs to be running on the internal buss, that were one CPU to fail it would simply drop off the internal buss as it would no longer be outputting buss requests.

Since the total T-Buss contains less than 40 wires, there is a statistically insignificant probability of one of the modules on the buss physically shorting the buss to ground. The T-Buss itself, being a tri-state buss, is particularly amenable to a design where the failed module simply "does not exist" as far as buss operation is concerned. Redundancy management becomes very simple as every CPU can poll and access the status of every module on the buss except the other CPUs (CPU to CF communication is through memory.)

The basic arithmetic and logic unit of the TDY-750 machine is built from AMD 2901B components. We have provided a 32 bit wide ALU to give the speeds that we felt were necessary for most 1750 applications (See Figure 8). Note, however, that we were planning ahead from the point of view of technology insertion. We were well aware that numerous chip sets would be coming on line executing 1750 code in the years to come. We can incorporate any of the known chip sets into our machine's architecture simply by removing the present CPU card based on the 2901 components and replacing the card with a chip set based card. The overall computer would not change - there would simply be one circuit board that was cheaper, depopulated, and used lower power. As the total power dissipation of the entire machine would then reduce, the total machine's reliability would increase.

In fact, the entire machine was designed from the point of view of being able to rapidly incorporate technology upgrades - be it in the central processor, input/output, and/or memory simply by replacing a plug-in card with

no other retrofit of the machine.

Having designed a machine that was completely compliant, both in detail and architecture, with a MIL-STD-1750 instruction set, we then incorporated an input/output card compliant with MIL-STD-1553B. Figure 9 shows the basic concept of the 1553B interface card. Many early designs, by both our and other organizations, of a 1553B interface card required much of the buss housekeeping function to be performed by the central processor. Early analyses of the impact of 1553B on the central processor indicated there was a potential for an inordinate processor load to service the 1553 buss.

More modern designs (such as shown in Figure 9) utilize an on-card microprocessor to do the normal buss housekeeping functions. This also transforms the 1553B card into a simple direct memory access card internal to the machine. All of the 1553B buss mechanization peculiar elements (formats, fault tolerance strategy, retry strategy, etc.) are stored in a 8K x 16 on-card ROM and/or PROM. Using this mechanization, a reasonably loaded 1553B buss will take less than 30 KOPS of the central processor's time to bring the messages in and/or transmit the messages to the 1553B card. We have also designed the card so that by simply changing the on-card firmware, the card can function either as a remote terminal, a controller, or, under program control, it can be changed inflight from a remote to a controller or vice versa. The mechanization of this card is completely compliant with MIL-STD-1553B and has found wide application with all of our actual and potential customers.

SOFTWARE

No discussion of MIL-STD-1750 would be complete without a detailed discussion of the software support tools. MIL-STD-1750 goes hand-in-hand with the usage of the Jovial J73 language as defined in MIL-STD-1589B.

The interface between the Jovial J73 compiler (and the remainder of the Air Force tool set) is a small standalone mini-computer we have named the Program Control and Monitor System (PCMS). This is a simple Z/80 and/or 8086 based microcomputer based system that interfaces between the large scale machine hosting the Air Force tool set and the TDY-750 computer series. The interrelationships between these three components are shown on Figure 11. A simple RS232 link is all that is required between the external computer (be it a 3033 and/or VAX11/80 system) and our Program Control and Monitoring System. We supply a number of software tools over and above the Air Force tool set that are hosted inside our PCMS. The PCMS also contains the loader that loads the absolute object modules received from the Air Force tool set into the computer's main memory. This load can be done either via RS232 datalink, 1553B datalink, and/or the program monitor interface card.

Many early embedded computers had problems in gaining access to the internal workings of the computer and trying to find out exactly what was happening during high speed computer operation. The computer input/output is normally designed for inflight operation - not program debugging. To this end, we have designed a program monitor interface module that buffers in between the T-Buss and the external Program Control and Monitoring System (see Figure 12).

The PMI module not only allows a complete sequence of halt, stop, single step, etc. operations to be performed but it also contains an on-card 1024 word hindsight register. Under program control, the contents of this entire register can be dumped so that one can see how one arrived at a particular machine state. The PMI card in some applications is also "intelligent" in that only selected buss transactions are stored on the hindsight file.

A complete list of the functions provided by the PMI module is shown in Table 3. A unique feature of our design is that the PMI module is powered from the external test equipment. In some programs, the PMI module flies unpowered. The reliability detriment to the computer is therefore negligible inflight. In other programs, the PMI module is installed only in those computers assigned to the software development process. In yet other programs, the PMI module is externally inserted into the T-Buss through a cover plate and no provision is made for the module inside the airborne housing.

Figure 13 shows the display presented to the operator. Every software development engineer who has used the console of our Program Control and Monitoring System has been quite excited with the capabilities provided the individual to see what is happening inside the 1750 machine as they debug it a particularly balky piece of software.

No program would be complete without early versions of computers provided to the software engineers to allow the software process to begin in parallel with the hardware design process. We normally supply a complete TDY-750 lab computer that is identical to the packaged units except that it is packaged on large circuit cards and utilizes DIPs for lower cost. We normally provide one of these functionally equivalent units to all of our customers within 12 month of contract go-ahead.

Even with the 12 month span time to receive a functionally equivalent unit, there exists a need in most programs for earlier software support for our users. To this end, many of our users elect to purchase a simple 64K/1553B/TDY-750 CPU mechanization in the lab DIP configuration. We normally supply these units within 3 to 4 months of contract go-ahead to our users so that they can begin to become familiar with the Air Force tool set and actually watch code executing on a 1750 machine in their lab far in advance of the specific configuration they are ordering for their program. Significant cost savings have been predicted on every program that has planned to use this very early version of a software development system.

Table 5 shows a complete list of the software tools that can be provided by the Teledyne Systems Company in addition to the TDY-750 hardware. Note, in particular, the existence of the symbolic debugger. In most programs, the majority of the code will be written in a high order language. Although much of the debugging will be done on the Air Force tool set on the 3033/VAX11780, bugs will still remain that need to be resolved running on the embedded machine itself. There is nothing more frustrating than trying to find where a variable is in absolute storage after it has gone through the compiler/linker process. Our symbolic debugger allows halting at Jovial breakpoints, and reading/writing values of Jovial variables in english units. Our symbolic debugger handles all memory location and data type conversions.

BIOGRAPHY

Mr. Geyer is presently Vice President, Advanced Technology, for the Teledyne Systems Company in Northridge, California. He received his BSEE degree from the University of Colorado in 1957, and his MSEE from New York University in 1959 after starting his engineering career as a member of the Technical Staff of the Bell Telephone Laboratories. Prior to joining Teledyne Systems Company, he was with General Dynamics/Convair Division for over twenty years.

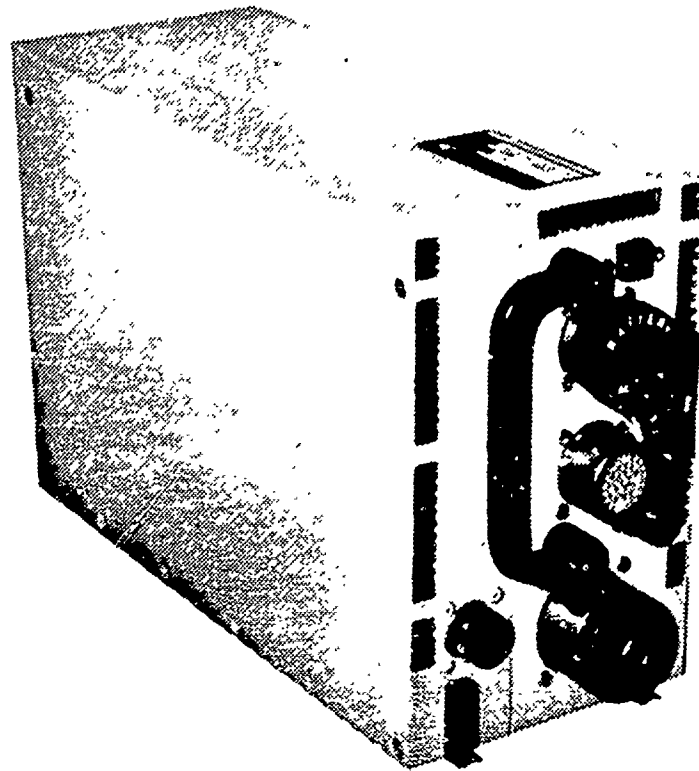


Figure 1 A Detailed Mockup Supported Our TDY-750/200
Mechanical and Thermal Designs

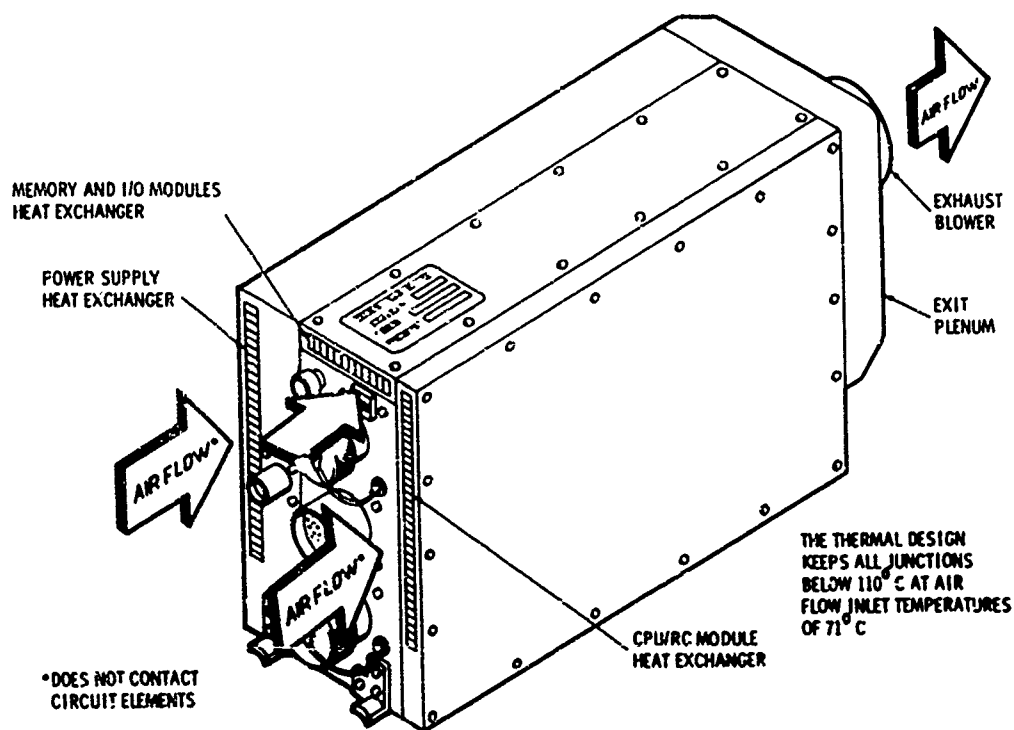


Figure 2 The Air Cooled Thermal Design Utilizes Three Separate Heat Exchangers for Short Thermal Paths

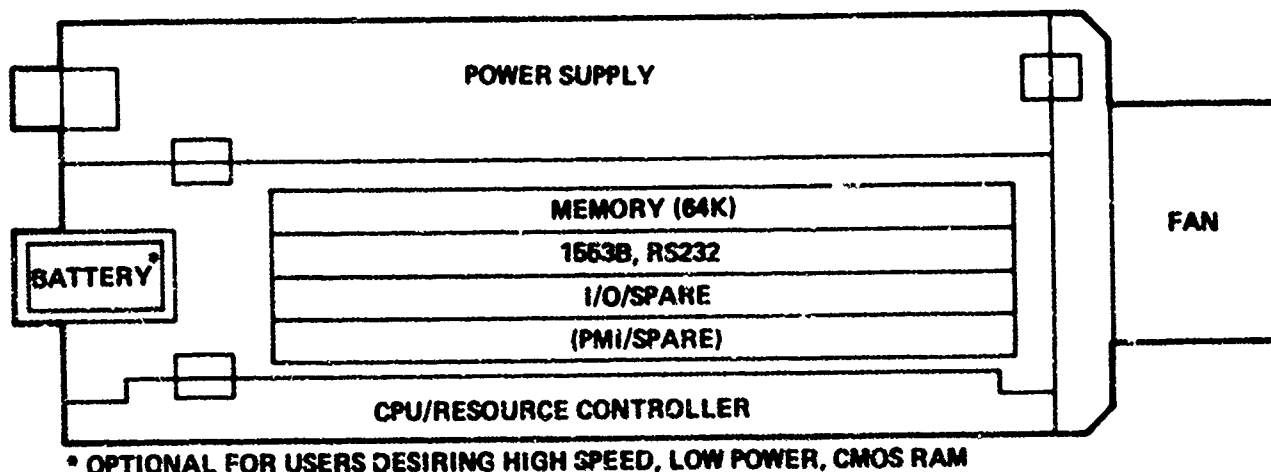


Figure 3 The TDY-750/200 Mission Computer Functional Arrangement Allows for Growth

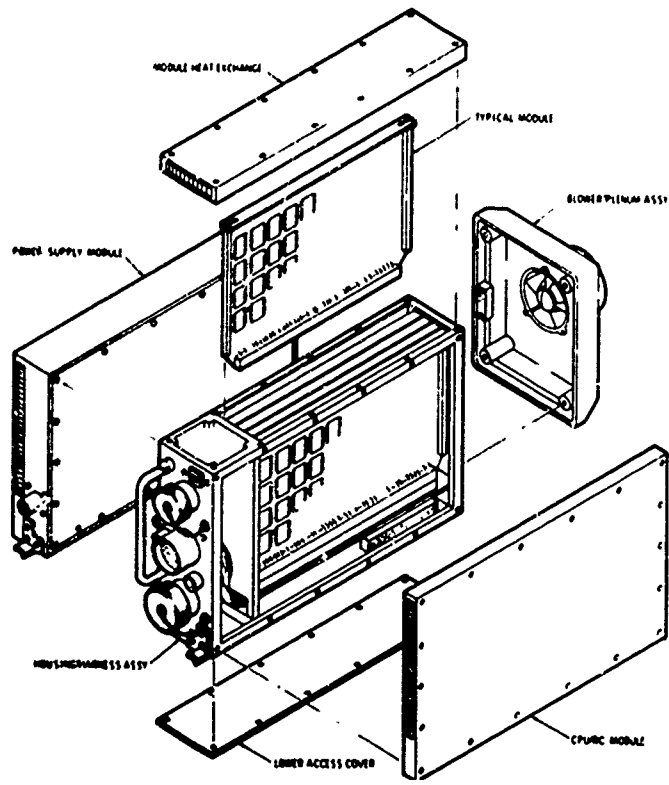


Figure 4

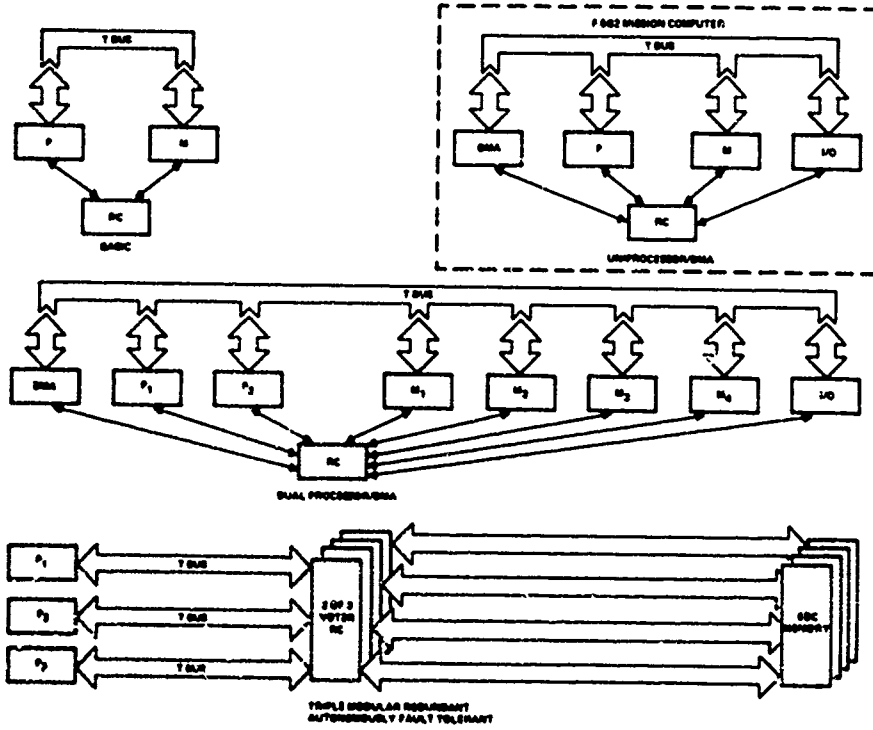


Figure 6

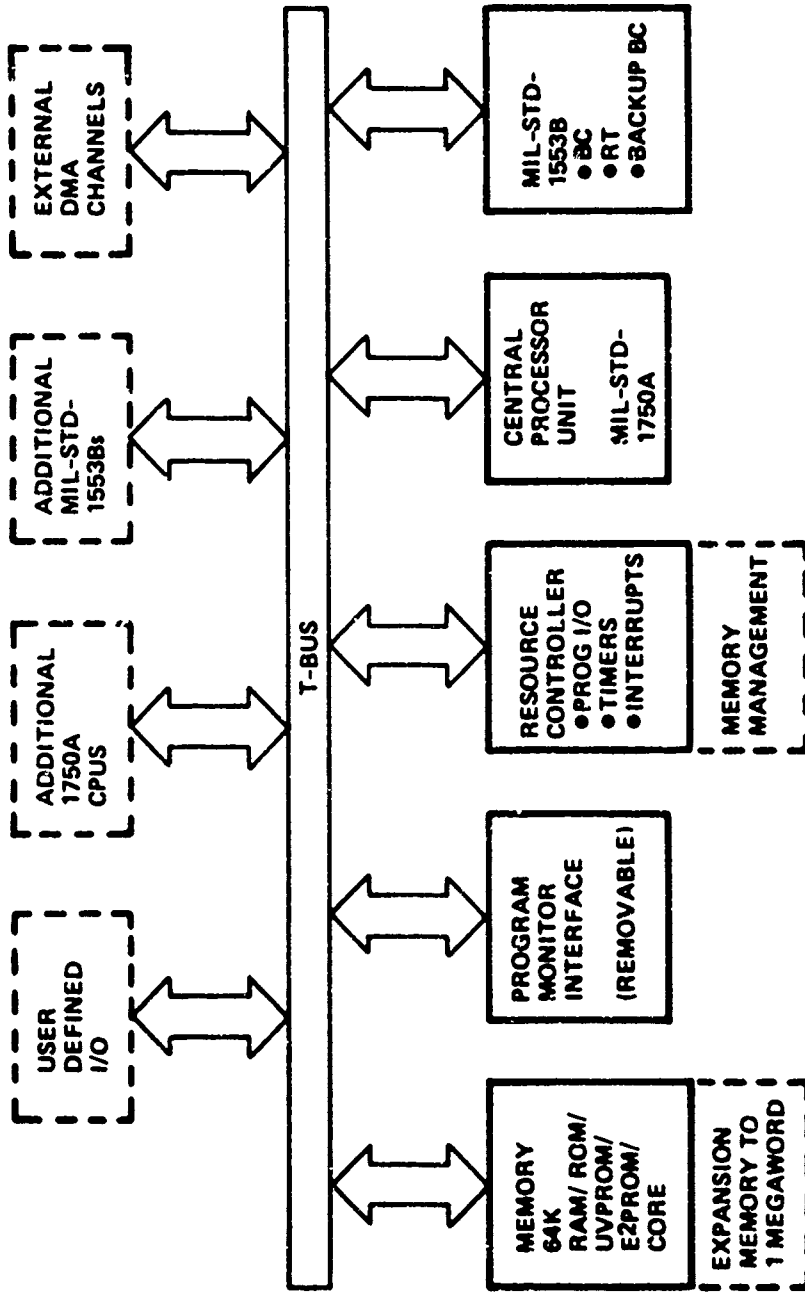
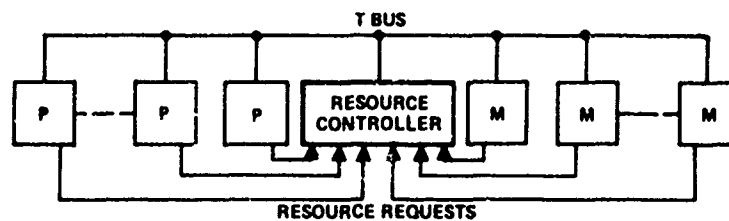


Figure 5 The TDY-750 Computer Utilizes the T-Bus to Achieve an Expandable System



DEFINITION

THE TBUS SHARES ADDRESS, MODE, AND DATA FOR GIVEN PROCESSOR/MEMORY CONNECTION. THE RESOURCE CONTROLLER ALLOCATES THE BUS RESOURCES.

TIME-MULTIPLEXING

THE TBUS IS DESIGNED TO TIME MULTIPLEX MULTIPLE USERS BASED ON SYSTEM NEEDS.

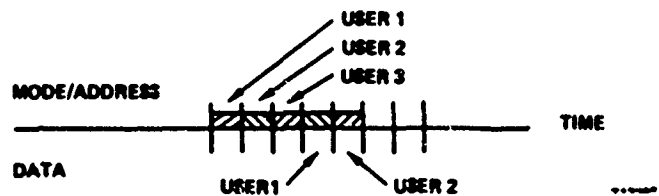


Figure 7

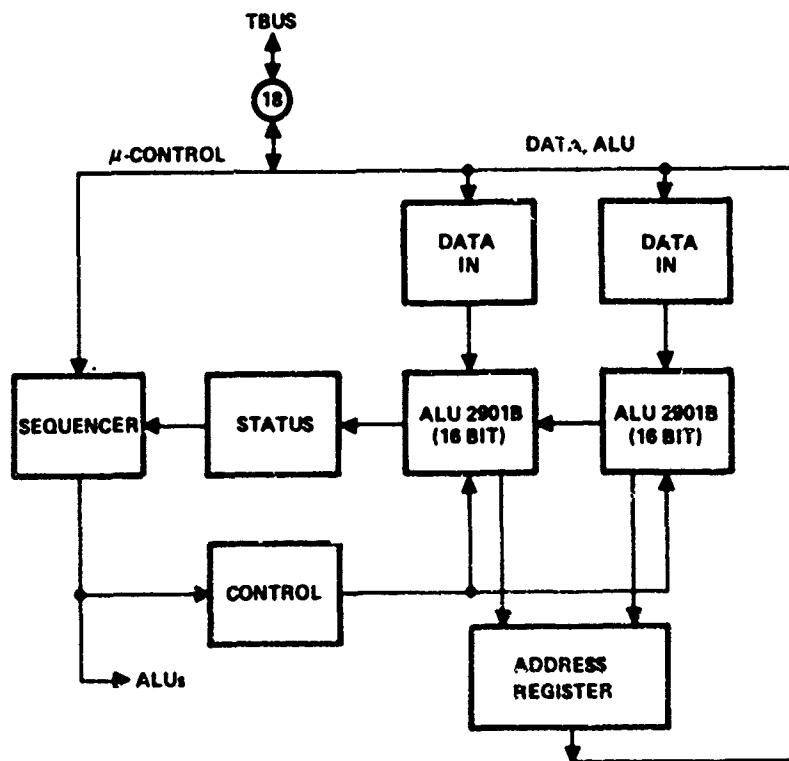


Figure 8

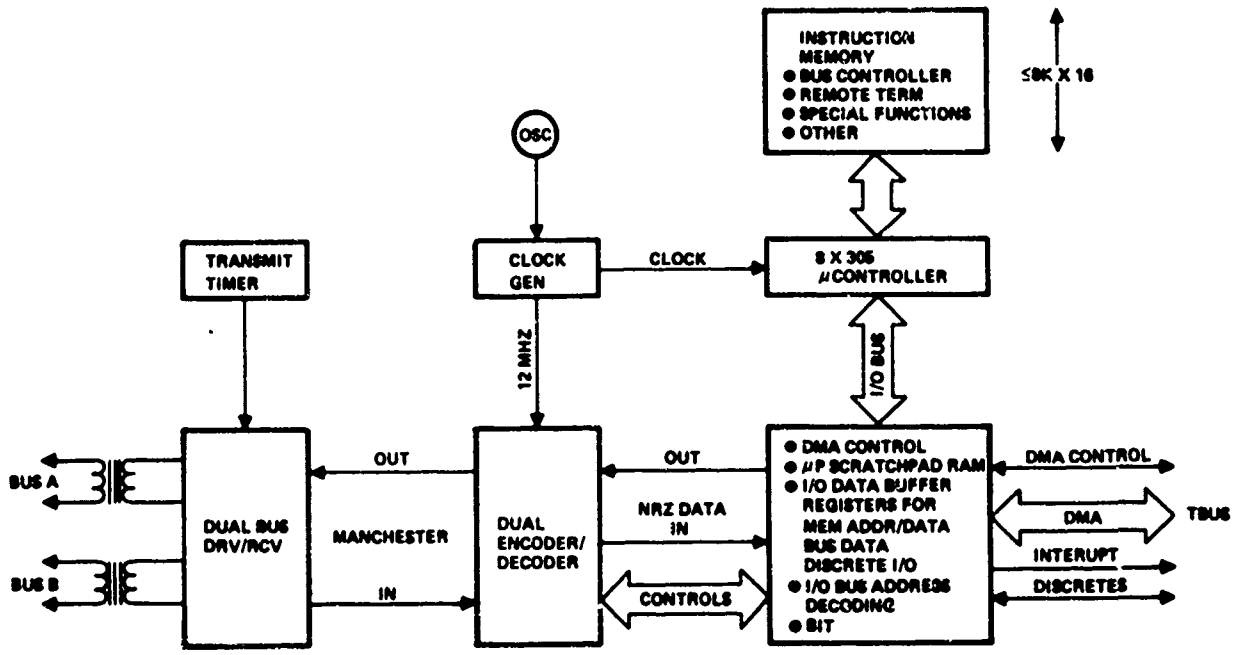


Figure 9

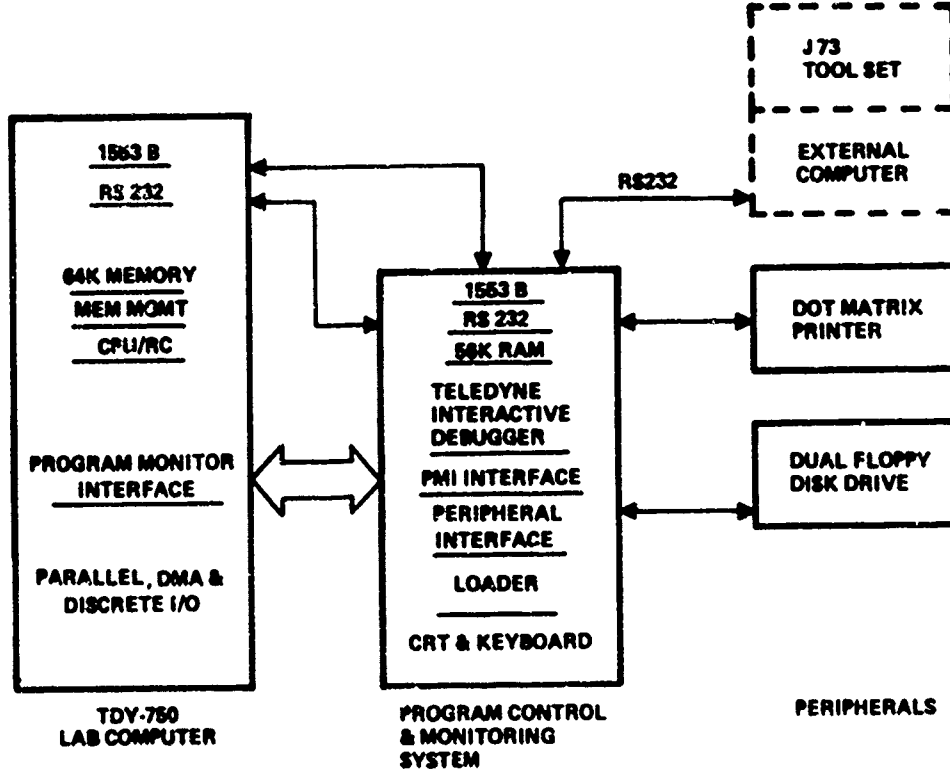


Figure 11 Teledyne's Software Development System

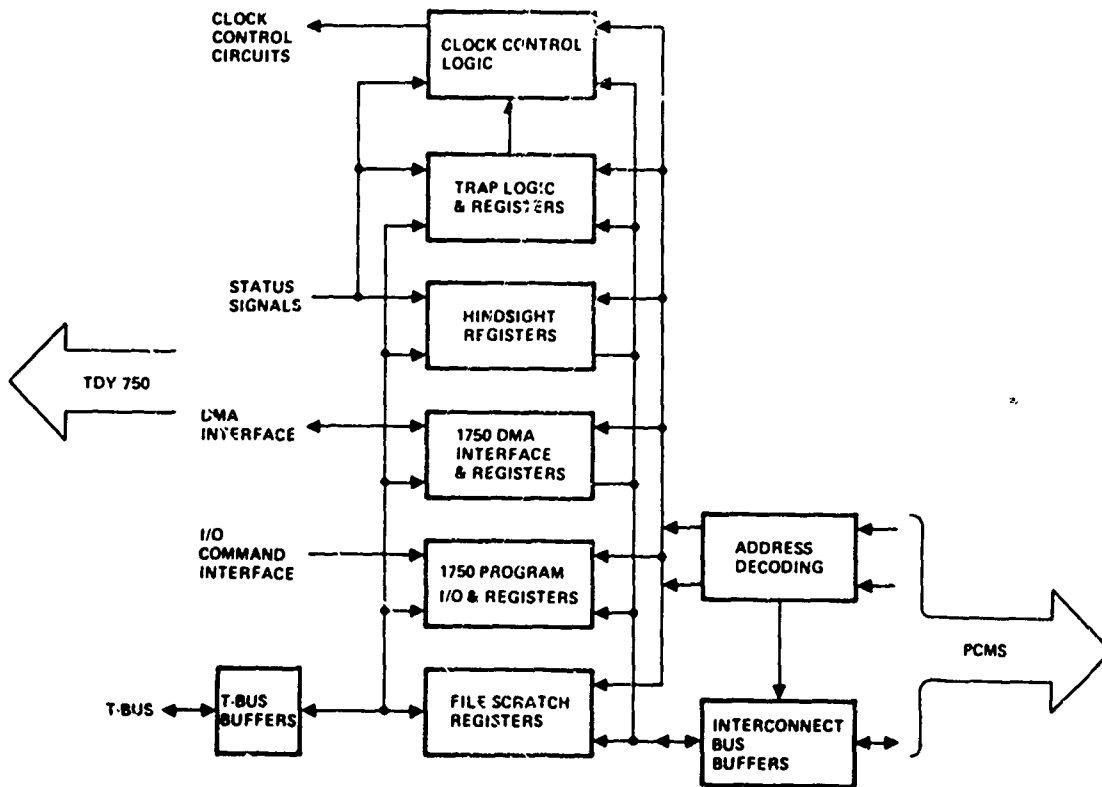


Figure 12 The PMI Module Interfaces the T-Bus Under PCMS Control

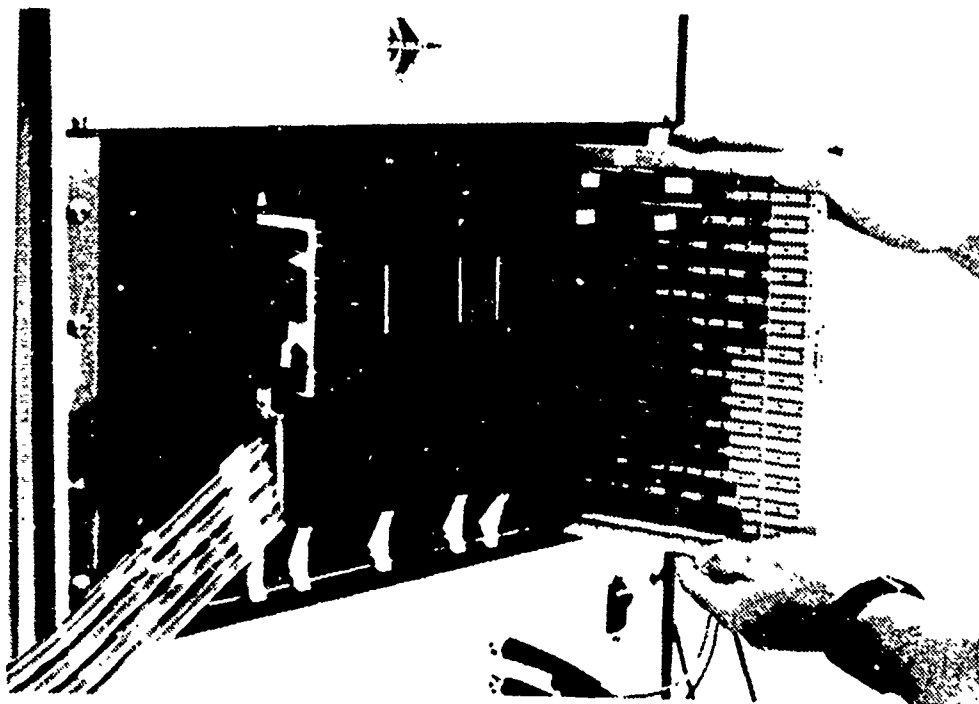


Figure 14 The TDY-750 Lab Computer Utilizes DIPs and Wire Wrap for Low Initial Cost and Easy Maintenance

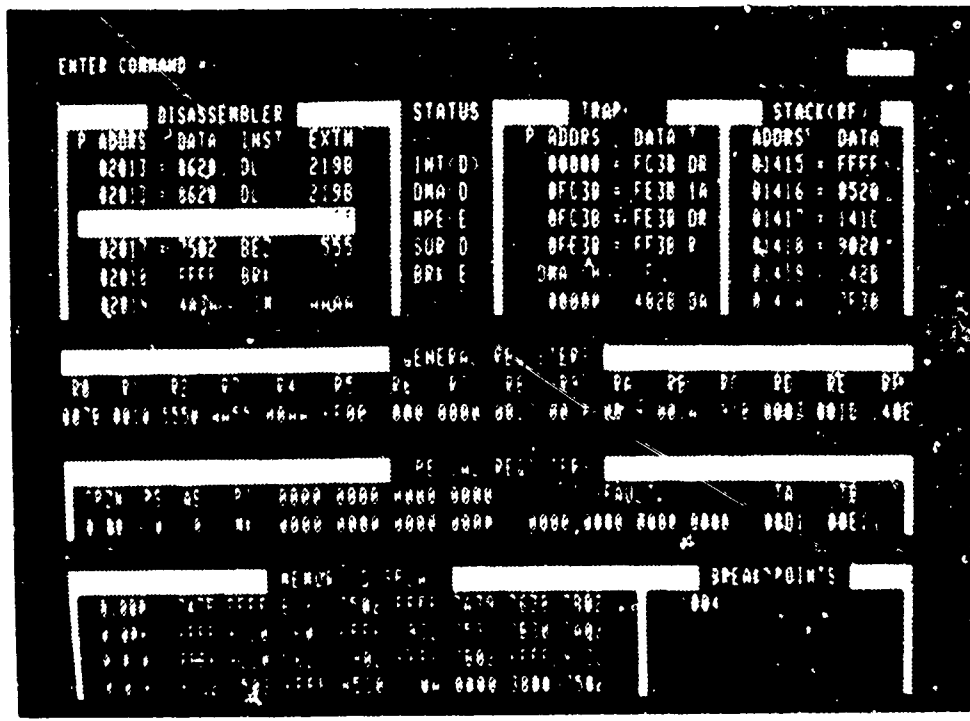


Figure 13 Teledyne's Display Provides Easy Identification of 1750 Parameters

FEATURES*	TDY - 750/200	
	SPEC LIMITS	NOMINAL
WEIGHT (POUNDS)	17	15
SIZE (CUBIC INCHES)	600	520
INPUT POWER (WATTS) INCLUDING FAN	150	135
THROUGHPUT (KOPS) (DAIS MIX)	635	650
MTBF (HOURS)	2100	5000

*WITH 64K SEMICONDUCTOR MEMORY, ONE DUAL CHANNEL 1553B I/O CARD, TWO SPARE SLOTS.

Table 1. TDY-750/200 Characteristics

- MIL-E-5400, CLASS I, OPERATING TEMP -55⁰C TO +55⁰C
NON-OPERATING TEMP -62⁰C TO +85⁰C
- MIL-STD-704A, CATEGORY B, WITH 290 TO 480 Hz EMERGENCY POWER
- MIL-STD-401A, NOTICE 3 EMI/EMC
- MIL-STD-781, TEST PLAN XIVC RELIABILITY
- MIL-STD-882A, SYSTEM SAFETY
- MIL-STD-1472B, HUMAN ENGINEERING
- MIL-STD-883, CLASS B MICROCIRCUITS
- MIL-STD-810, ENVIRONMENTAL TEST
- MIL-STD-471, METHOD 9 MAINTAINABILITY
- SUPPLEMENTAL 30MM GUNFIRE VIBRATION AND ACOUSTIC NOISE
 - + 85⁰ C FOR 10 MINUTES

Table II. The TDY-750/200 is Designed to Meet These Specifications

- PROGRAM LOAD AND VERIFICATION
- PROGRAM CONTROL
 - RUN, HALT, SINGLE STEP, FAST STEP, SAVE
 - BREAKPOINT HALT
 - TRAP HALT AT SPECIFIED ADDRESS
 - CHANGE PROGRAM COUNTER VALUE
 - PROCESSOR OR SYSTEM RESET
 - DIRECT I/O COMMANDS
- PROGRAM MONITORING
 - SELECTED MEMORY LOCATIONS (VIA INTERNAL DMA)
 - SELECTED I/O FUNCTIONS (VIA TBUS)
 - TRAP ON SPECIFIED ADDRESS
 - INTERRUPTS AND FAULTS
 - HINDSIGHT AND TRACE FILE
 - TRAP ON SPECIFIED FUNCTIONS

Table III. The FMI Module Provides Extensive Software Development and Debug Capabilities

AD-P003 568

PAVE PILLAR: A MATURATION PROCESS
FOR AN ADVANCED AVIONICS ARCHITECTURE

D. Reed Morgan
LT COL R. J. Bellem

AFWAL/AAA
WPAFB, OHIO
(513-255-6444)

ABSTRACT

Recent speed and density advancements in microelectronics will now permit the development of powerful and affordable avionic architectural elements - viz. processing, memories and wide band data buses. An advanced architecture making use of these elements and coupled with highly flexible software, will enhance the capability to fully exploit information integration and automation processes. An abundance of real-time data is available for integration from diverse subsystems aboard advanced military aircraft. Dramatic improvements in avionic system availability, crew workload reduction, weapon system survivability and supportability are possible using this approach.

The introduction of these system integration technologies into the force structure is needed at the earliest opportunity to meet expanding mission requirements. However, care must be exercised to ensure that concepts and standards have been matured through validation testing to avoid potentially costly mistakes.

The Air Force has established the PAVE PILLAR Program to provide the needed maturation of advanced avionic system architectural approaches, system elements and potential system standards during advanced development. This Program also initiates the concept of establishing system integration technology as a separate discipline within the Laboratory framework.

This paper describes the PAVE PILLAR Program being pursued within the Avionics Laboratory of the Wright Aeronautical Laboratories. Enhancements to operational force effectiveness resulting from system integration will be described, along with advanced system technology elements and potential standards which will be developed and demonstrated.

INTRODUCTION

This paper is written to apprise the advanced system architecture community of the rationale and planned activities under the PAVE PILLAR Program. This Program will be of fundamental interest in that: (a) the use of existing standards for projected architecture applications in the 1990's will be demonstrated; (b) the need for new potential standards will be assessed, with necessary developments accomplished and demonstrated in the framework of a backward compatible advanced architecture; (c)

validation of the advanced architecture and its associated hardware/software elements will be accomplished through the demonstration of several representative avionic system applications; and (d) through a sequence of both ground and flight tests, the resulting architecture will be matured, along with associated potential standards, in order to establish a framework for avionics into the next century.

The organization of this paper first describes the current "situation" in avionics - viz. what are the current problems which must be solved? Future system requirements which an advanced architecture must accommodate is presented, followed by a discussion of the strategy, or rationale for the PAVE PILLAR Program. Advanced architecture system characteristics are described, along with a discussion of the developmental and demonstration plan which will lead to flight validation.

BACKGROUND - A PERSPECTIVE ON AVIONICS

In formulating a strategy for advanced avionics architecture development and maturation, it is first necessary to establish a perspective as to how avionics are currently used, their current limitations and what can be improved.

Today's avionics are placed on aircraft as a means to aid the aircrew in mission accomplishment. With a few exceptions, these electronic devices are separately developed and functionally integrated autonomously. (NOTE: Current architectures have mostly been used to replace wires - this physical integration has not yet substantially affected a change in functional integration). Figure 1 captures the concept of current avionic subsystem autonomy. This Figure further conveys another feature of today's avionics - viz. the dedicated outputs of these separate subsystems are processed by the crew through the controls and displays subsystem. It is the crew's cognitive and psychomotor capabilities which are employed to perform information assimilation and to affect an action to a control element. In that our aircrews are already workload saturated, it is obvious that we cannot continue to merely add boxes or subsystems in a single thread manner. Later discussions will argue that an advanced architecture will be needed to permit functional automation for many missions in the future.

Secondly, today's avionics are difficult and expensive to maintain. Figure 2, typical of current fighters, shows the distribution of flight line maintenance actions for avionics in comparison with other subsystems. Recent data show that approximately 25% of the removed avionics line replaceable units are judged to be fault-free at the intermediate shop for typical fighters. One quarter of our avionic maintenance personnel's time and one quarter of our spares are not being effectively utilized. Reasons for this situation run the gamut from simple to complex. For example, cables and connectors account for a great deal of the intermittent and "cannot duplicate" problem. (Although data is not routinely collected to substantiate the degree of the problem, interviews with maintenance personnel and a limited survey of reports indicates from 20% to 50% of maintenance actions originate from faulty cables and connectors). Further, maintenance difficulties have been compounded by failure prone BITE and inadequate failure monitoring and recording. In short, improvements need to be made in our ability to

isolate avionic faults at the flight line. Again, inadequate diagnostic capability is in part due to autonomous subsystem development. Later discussion will argue that an advanced architecture will be an essential element in improving availability.

What then is the root cause of current avionics problems? Obviously, the physical way which avionics is integrated in piecemeal fashion explains why automation and availability is lacking - this current architecture is however only symptomatic of the problem.

One school of thought blames much of our avionics-derived problems on advanced technology - viz. we are using complex systems which fail often and cannot be properly maintained. The argument is that we would be better off by using simple, inexpensive avionics but build more aircraft. Not only do the authors feel that such a position does not satisfactorily respond to survivability needs downstream, such a view does not correspond to factual data. For example, R. Little et al provide an excellent comparison of F-4 and F-15 capabilities, availabilities and technologies (Ref 1). Their conclusion, supported by data, is that technology has been falsely accused in limiting availability and support, as well as contributing to complexity.

The authors believe that many shortfalls in current avionics are due to three major factors:

(1) Lack of Technology

(a) Operating System/Architecture

Exploitation of available information on the aircraft, including automated process control between classical subsystems is needed to not only reduce workload but to provide a means for integrated diagnostics. Such an approach requires a highly interactive operating system executive supported by wide band data distribution, high speed processing and extensive mass memory. Such technology was not available or adequately understood for inclusion in recent aircraft.

(b) Improved Subsystem Reliability

Increasing the inherent reliability of only a few high failure rate avionics will be extremely beneficial (see Figure 3). Reduction in the number of cables and connectors through extensive multiplexing, deletion of mechanical elements (e.g., radar antenna drive train, mechanical gyros), deletion of components requiring high voltages (e.g., traveling wave tubes and CRTs) are keys to basic reliability improvements. The Air Force currently has several programs underway which will improve the reliability of radar, CNI, navigation and EW subsystems.

(2) Cultural Limitations

As avionics capabilities have grown over the years, organizations (both within and outside Government) have evolved which specialize in the development of functionally-oriented subsystems. For example, flight control, engine control, navigation, communications, electronic warfare, radar, stores management (etc., etc.) are considered

"separate" entities and are for the most part developed and integrated separately (two notable exceptions are navigation/weapon delivery integration and terrain following radar/flight control integration). The authors believe that such "localized" thinking has, in the past, created a mind set which has slowed down possible progress in the automation arena. It is worthwhile noting however that substantial improvements in automation of coordinated subsystem functions is dependent on the architectural technology.

(3) Avionics Maturation

Another school of thought contends that avionics reliability and testability will be improved if a more lengthy and iterative "fly and fix" approach were followed before commitment to production. As applied to advanced architectures, such an argument appears to be extremely sensible because of the fundamental role played by the architecture in influencing the entire avionics system over the life of the aircraft. "Guessing wrong" or inadequate testing may result in an extensive and expensive integration phase, may lead to frequent and expensive retrofits, may inhibit the isolation of faults, further compounding sparing difficulties, etc., etc.

In summary, current day avionics problems are not fundamentally due to technology; rather, had the technology existed and matured, many current-day problems would not exist.

FUTURE TRENDS IN AVIONICS: THE NEED FOR AN ADVANCED ARCHITECTURE

Projected threat density increases and threat mobility will result in a high flux environment where decisions must be made quickly and accurately conveyed to affect the appropriate action. Access to information and its subsequent exploration will be a key element of many successful operations.

External to the aircraft, communications, radio navigation aids, IFF and JTIDS information will play an important role in providing this information. The opportunities offered by this new "radio" capability will not be fully realized until several fundamental issues are resolved:

- 1) Automated data handling/presentation of the information (particularly for threats)
- 2) Affordability (plus weight and volume constraints for tactical aircraft) of the plethora of radio functions available
- 3, Availability of the information in light of equipment failures and jamming environments.

The high flux environment expected will also require similar automated processes to be invoked on information internal to the aircraft. One recent study into future automation requirements concluded that trajectory and attitude control, engine control, weapon delivery and navigation were likely candidate functions (Ref 2). Both Air Force in-house and contractual studies show that information from across classical subsystem boundaries must be collected, blended or coordinated through automated

process control and then distributed to appropriate displays or effectors, again back across classical boundaries. For example, automated trajectory control will require integration and coordination of navigation parameters (where am I?), JTIDS, stored threat files and electronic warfare receivers (where are the threats relative to me - which ones are new - which ones can cause harm?), stored terrain data for both terrain following/terrain avoidance as well as threat masking, propulsion/flight control and targeting and fuel data (how far is the target - what time am I supposed to be there - how much fuel do I have?). The coordination of this data will be necessary to determine new ingress/egress paths brought on by new threats or target redirect commands.

Human control over these coordinated processes and assimilation or monitoring of the resulting actions will also require new automation concepts in crew station design as well as substantial refinement and intuitive presentation of information. Use of voice control to change display modes, extensive use of color graphics to display distilled, overlaid imagery/stored data are examples of approaches which must be seriously pursued.

Figure 4 captures the rationale behind the need for a new architecture in supporting weapon system/crew automation processes in the future. An advanced architectural approach will be needed to accomplish the integration, dissemination and presentation of information. Most obvious is the need for high speed data and video buses. For example, future aircraft are expected to employ large amounts of mass memory for terrain and cultural data as well as threat information. Correlation of this type of data, along with distributing large quantities of data for automated process control will lead to data bus requirements in excess of several MIL-STD-1553B buses (preliminary study indicates the need for a 20-50 Mbits per sec bus). Further, full compliance with MIL-STD-1760 in providing bi-directional video information between stores strongly suggests the need to explore video busing strategies to obviate the need for a large number of point-to-point cables.

Figure 5 further summarizes the need for an advanced architecture to provide availability improvements. For example, extensive use of wide band buses can reduce the number of cables/connectors by up to approximately 90% (thereby reducing a major reliability problem). Extensive use of VLSI/VHSIC circuitry and distributed computing will also inherently increase availability, again through cable/connector reduction. Further, an integrated diagnostics capability which would permit the in-flight monitoring of BITE, correlation of data from similar information sources and recording of environmental data would reduce cannot-duplicate (CND) and Re-Test OK (RTOK) problems. Such a capability will also be required to achieve fault tolerant operation for both physical and functional redundancy. Thus, achieving improvements in availability as well as automation will be dependent on high speed busing to affect the needed connectivity.

The required topology and system control of the advanced architecture will be derived from consideration of several key factors. These factors include growth capability (i.e., to support both pre-planned and unplanned product improvements), degree of fault tolerance and failure containment, processing/bus efficiency, etc. Consideration must also be given to prime

contractor/vendor responsibilities to ensure appropriate consideration of the functional partitioning and interfaces between advanced subsystems and the system. Further, continued use of MIL-STD-1553B buses for overall system control and to permit future use of compatible hardware must be included in the topology. An example architectural approach which satisfies these conditions is shown in Figure 6.

Note that this architecture supports the use of MIL-STD-1553B both at the global (system) level and/or at the subsystem level. Extrapolation of present trends indicates that many future subsystems will likely be configured as a bus oriented structure - hence, hierarchical busing interaction will be required. As with MIL-STD-1553B, the architecture should support high speed busing both at the global as well as at the subsystem level. Ultimately, high speed busing is expected to be used between standard modules within a subsystem to replace failure-prone connectors. Finally, a video bus structure is shown under the control of MIL-STD-1553B. The latter bus will be needed to accommodate the bi-directional video distribution between stores, per MIL-STD-1760. It is envisioned that a frequency allocated approach similar to cable television will be used to distribute the large amount of video information between sensors, displays and "smart weapons." Development of a standard high speed data bus and a video bus will be needed to support this architecture.

Although the above topology will support virtually any projected system application or downstream retrofit, the relative simplicity of the associated executive operating system will be the key to utilizing the topology. The advanced operating system will be required to dynamically interact and control system and system/subsystem processes in near real-time. The operating system must accommodate fault tolerant processes at the global network level (e.g., failed bus) as well as directly interact with application software executing automated fault tolerant/safety of flight critical processes between subsystems. The degree to which the operating system can be exhaustively tested before airborne system use will determine ultimate acceptance. Consideration must also be given to standardization of application to executive software interfaces as well as subsystem/system standard interfaces in order to mature the operating system.

PAVE PILLAR STRATEGY - ARCHITECTURE MATURATION THROUGH DEMONSTRATIONS

Two key issues must be settled before deployment of the advanced architecture: (a) which new standards require development, and (b) determination of the scope/complexity of the resulting software intense approach which accompanies the architecture. Simply stated, confidence needs to be established in the design before commitment. We collectively need to determine what we should do as well as what we should not do.

In recognition of this challenge, the PAVE PILLAR Program has been established. The strategy is one of maturing the system integration architecture through sequential validation demonstrations. The approach provides a low cost, rapid means of testing new integration concepts and high technology architectural elements and to develop design, performance and cost guidelines at the advanced development level. In so doing, the Program will greatly assist the Air Force in avoiding mistakes in

attempting to implement approaches found to be too complex or inadequate to support availability needs, as well as assist in the earlier introduction technology shown to be effective. A large, non-proprietary data base describing designs, algorithms and software will be made available to industry. In providing the data base, it has been concluded that two levels of testing are desirable. The first level would take the form of a laboratory-based "avionics wind tunnel" - a means to quickly configure, demonstrate and test a given system configuration or potential standard at low cost. After determining high payoff approaches in the laboratory, the second level of testing would occur through flight testing on a generic test bed aircraft to gain further confidence in the results. Maturation of system integration technology requires coordination and inputs from the community in order to improve technology transition. In order to affect this participation, the PAVE PILLAR Program is coordinating its activities with AFSC Laboratories, ASD, AFLC, and the Using Commands. A wide range of contractual activities, as explained in the next Section will involve a large spectrum of industry participants.

PROGRAM DESCRIPTION

The approach to be utilized in the core architecture development in the PAVE PILLAR Program can be viewed as four distinct, yet highly related efforts. As shown in Figure 7, these efforts are:

(a) System definition, design, development and integration which includes activities to validate the next generation architecture, establish a baseline for initial flight configuration, and the design, development and implementation of an advanced simulation facility for ground based testing.

(b) Efforts to perform availability and mission analyses and to bridge technology developments. Mission analysis will help merge technology with requirements, establish baseline performance, and based on continuing system performance analyses, document performance improvements and associated life cycle cost benefits.

(c) An advanced technology validation effort wherein promising laboratory exploratory and advanced development hardware and software will be integrated into the in-house Government simulation test bed and real-time man-in-the-loop evaluation of the PAVE PILLAR architecture to provide an early concept validation and perform availability demonstrations.

(d) Development of several key architectural building block elements which can be used for an early validation in the in-house facility. These building blocks will be transitioned to the system design, development and integration effort in (a) above.

A key element of the program is the early technology validation of the individual architectural elements in (c) above. Validation of these elements in the Avionics System Analysis and Integration Laboratory (AVSAIL) at Wright-Patterson will provide a lower risk implementation of these elements by the System Designer. The elements being validated at AVSAIL are:

(a) Video Bus - A wide band (20 MHz frequency allocation per modem), multi-drop, multi-user (up to 10 simultaneous connections) video distribution system which meets MIL-STD-1760 requirements.

(b) High Speed Bus - Specify and prototype several high speed data buses employing a range of competing characteristics and technologies (i.e., speed, media, signal format, interface).

(c) VHSIC 1750A Processor - Design, build, test and integrate VHSIC 1750A processors to support stand-alone and subsystem-embedded computer requirements. A MIL-STD-1553B and high speed bus interface will be developed. Preliminary studies indicate a speed regime of 3-4 MOPs (DAIS instruction mix) is possible.

(d) Common Signal Processor - Analyze existing signal processing functions and design, develop and test a signal processor having a standard architecture which will permit the reuse of software across several applications.

(e) Executive Software Coded in Ada - Apply Ada to real-time avionics software systems and assess the language features for future applications.

(f) Advanced Digital Avionics Map - Develop an electronic terrain map system that can be integrated into avionics systems to improve TF/TA, threat avoidance and navigation functions through advanced fusion algorithms.

(g) Fault Tolerant Architecture Concepts - Demonstrate fault tolerance concepts through reconfiguration and resource sharing to provide an early "Proof of Concept" of selected integration technologies. The System Design, Development and Integration effort in (a) above will implement many of the elements in (a) through (g) above. The total System will be validated after designing an advanced crew station and developing the system automation algorithms. The total system validation will make use of simulation technology in order to have sensor data inputs and interface characteristics. Flight test opportunities will be exploited following the advanced technology validation efforts.

The payoffs of the PAVE PILLAR Program will be evaluated in the following areas: sustainability; retrofitability; workload reduction; survivability; and standardization potential. Figure 8 summarizes the technology payoff areas for each of the validated technologies discussed in (a) through (g) above.

SUMMARY

Mission capabilities will be developed, in part, by the coupling of information and control processes both within selected subsystems and across virtually all subsystem capabilities projected for advanced aircraft. Coupling and sharing of these resources will require a breadth of knowledge and experience that spans virtually the entirety of air vehicle systems as well as innovation in architecture and integration technology. Figure 9 captures the gross structure of coordinated system operations in the future. This integrated approach deviates substantially

from the single thread subsystem approach used in today's aircraft; however, the approach is a culmination of integration trends which are occurring in piecemeal fashion. Exploitation of the benefits associated with information integration will require that both technical and cultural challenges be met.

In order to meet these challenges, the philosophy of the PAVE PILLAR Program will embody: (a) bringing to bear the breadth of knowledge and experience contained within the Air Force Laboratories, ASD, AFLC, Using Commands and the contractor community, (b) the use of baseline architecture parameters and standards which are currently being employed in Air Force programs; (c) the modification and extension of these standards and parameters as required, including the exploration of new standards (e.g., Ada, high speed buses); (d) innovation for coupling within and between classically defined mission critical and flight critical functions to preserve system safety while producing advanced capabilities; (e) the development and validation of advanced architectures, executive software, subsystem hardware and application software to implement the required fault-tolerant/automated coupling of information external and internal to future aircraft; and (f) development of advanced cockpit systems and concepts to better couple man and machine, while giving man the overall system management functions. With this approach, mature system interaction technologies/standards will be made available to establish the architectural framework for the 1990's.

REFERENCES

1. "High Technology Raises Fighter Force Readiness," R.C. Little, W.P. Murden, R.K. Schaefer, *Astronautics and Aeronautics*, June 1982, p. 38.
2. "Automation in Combat Aircraft," Air Force Studies Board, Assembly of Engineering, National Research Council, National Academy Press, 1982.

BIOGRAPHIES

D. REED MORGAN

Mr. Morgan is currently the Technical Director of the PAVE PILLAR Program within the Avionics Laboratory at the Air Force Wright Aeronautical Laboratories. Prior to this assignment in 1981, Mr. Morgan has been involved in application studies involving DAIS architecture for new systems. In 1972, he was involved in the early planning and execution of the DAIS PROGRAM.

RAYMOND D. BELLEM Lt Col, USAF

Lt Col Bellem is the Program Manager of the PAVE PILLAR Program within the Avionics Laboratory of the Air Force Wright Aeronautical Laboratories (AFWAL). Prior to this assignment, Lt Col Bellem was Chief, Support Systems Branch within the Avionics Laboratory, responsible for developing and operating the Avionics System Analysis and Integration Laboratory (AVSAIL) which is a major DOD avionics simulation and integration facility.

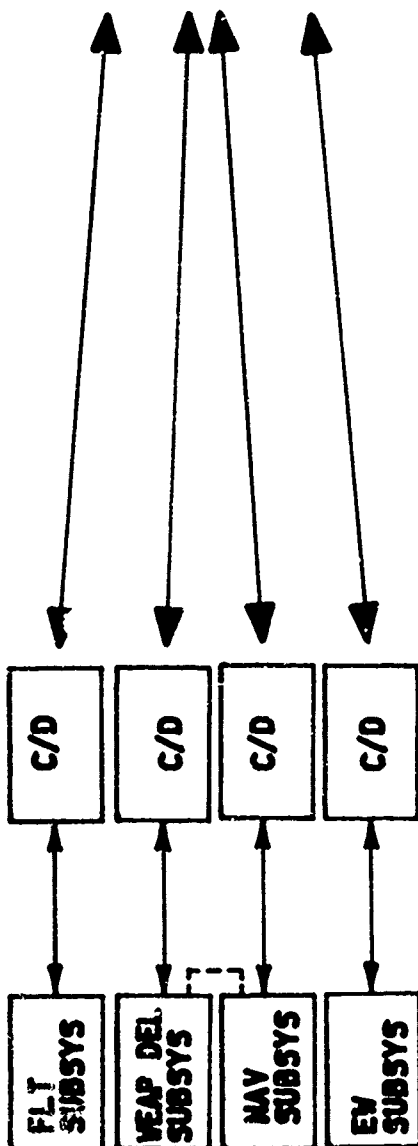


FIGURE 1. UNCOORDINATED AIRCRAFT SYSTEMS TODAY
(RELATIVE TO FUNCTIONAL INTEGRATION)

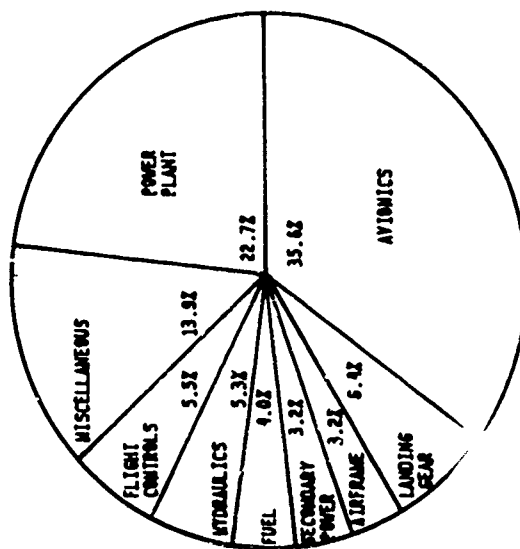
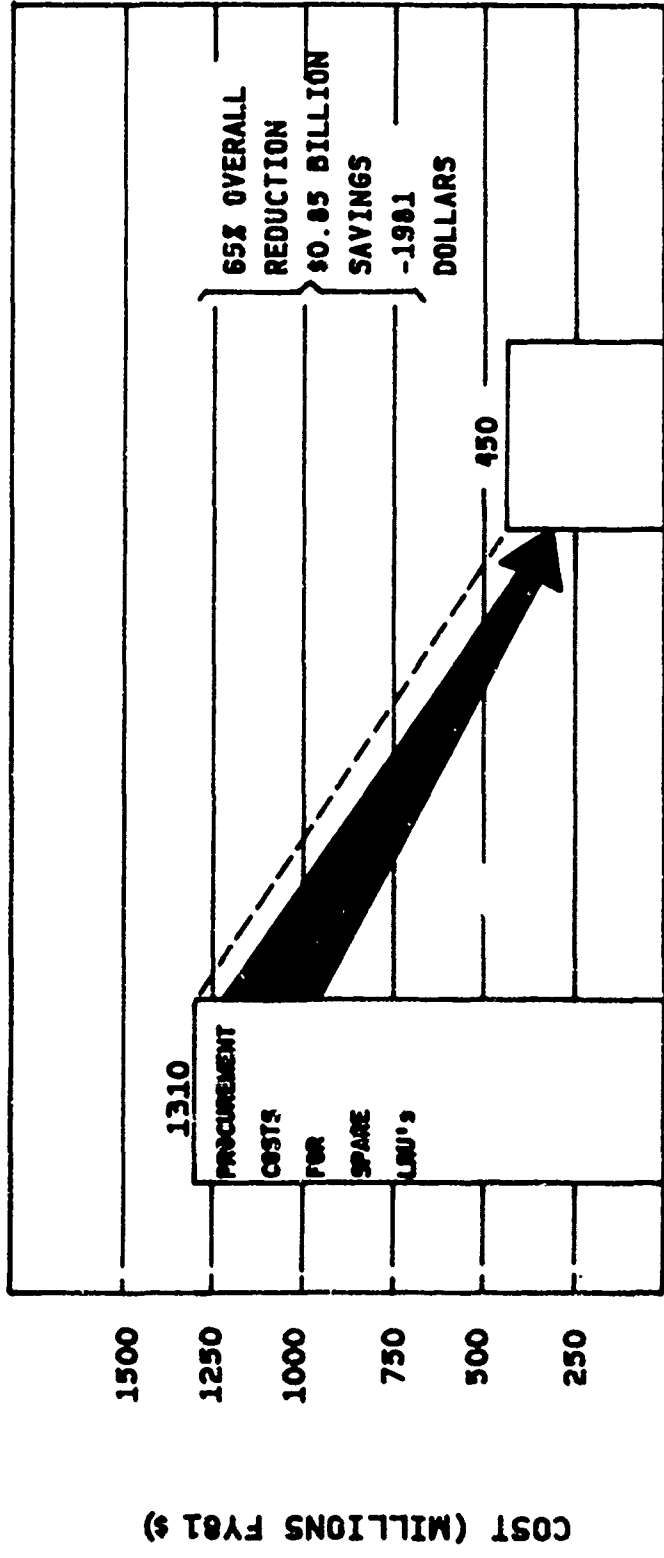


FIGURE 2. SOURCES OF AIRCRAFT MAINTENANCE ACTIONS FOR TYPICAL FIGHTER

- 11 LRU's (FROM 4 SUBSYSTEMS) 4 X MORE RELIABLE
- 18 COMBAT SQUADRONS
- 45 DAY DEPLOYMENT WITHOUT TEST EQUIPMENT (ELIMINATE AIS)

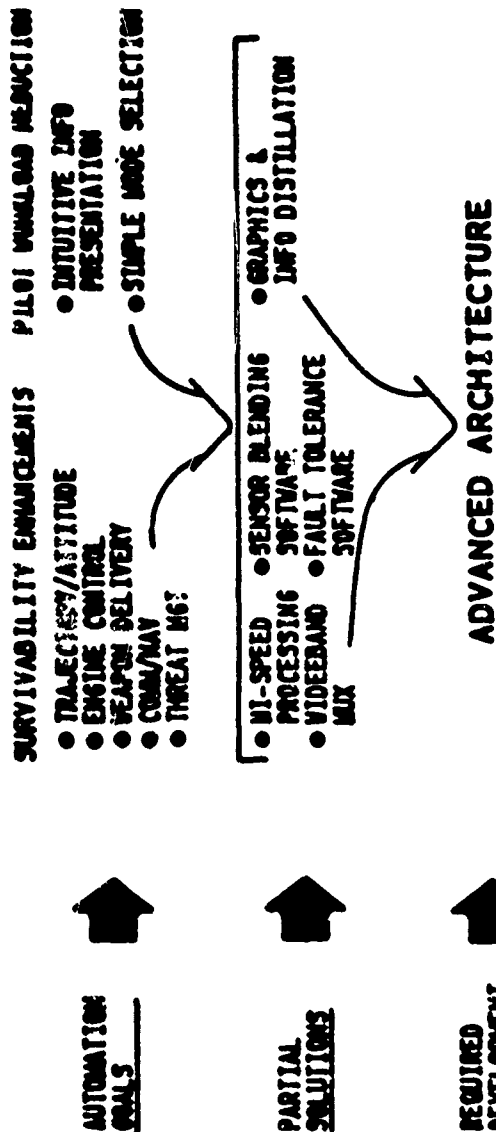


4 X CURRENT MTBF RATE FOR 11 LRU'S, (10% OF AVIONICS)

CURRENT F-15 MEAN TIME BETWEEN REMOVAL RATE

- RADAR (6)
- INS (1)
- HUD (1)
- WD (1)

FIGURE 3. RELIABILITY LEVERAGE ON LIFE CYCLE COSTS
 (EXTRACTED FROM "AN INTEGRATED VIEW ON IMPROVING COMBAT READINESS",
 M.D. RICH ET AL, RAND NOTE N-1797-AF, FEB 1982)



*AUTOMATION IN COMBAT AIRCRAFT, 1961 A.F. STUDIES BOARD-WOODS HALL

FIGURE 4. AUTOMATION IMPROVEMENTS THROUGH DIGITAL SYSTEM TECHNOLOGY

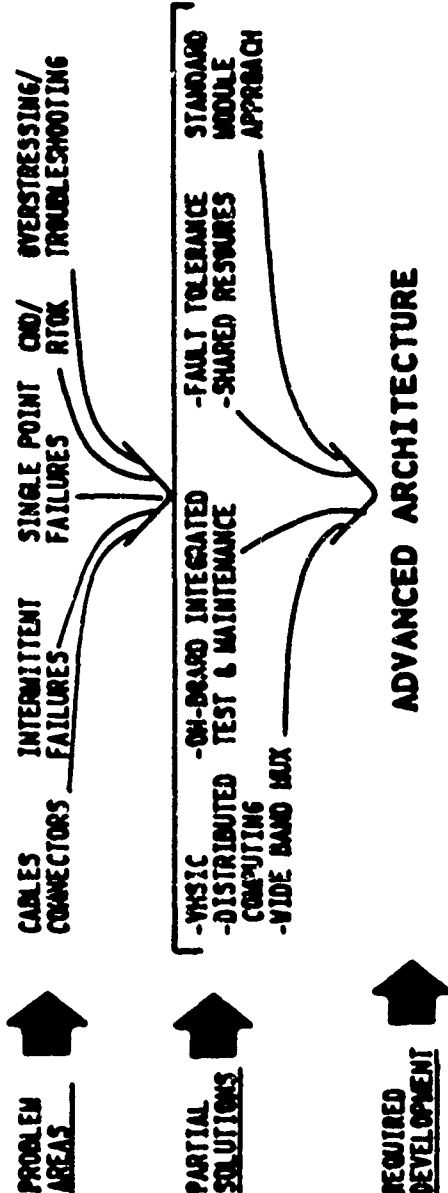
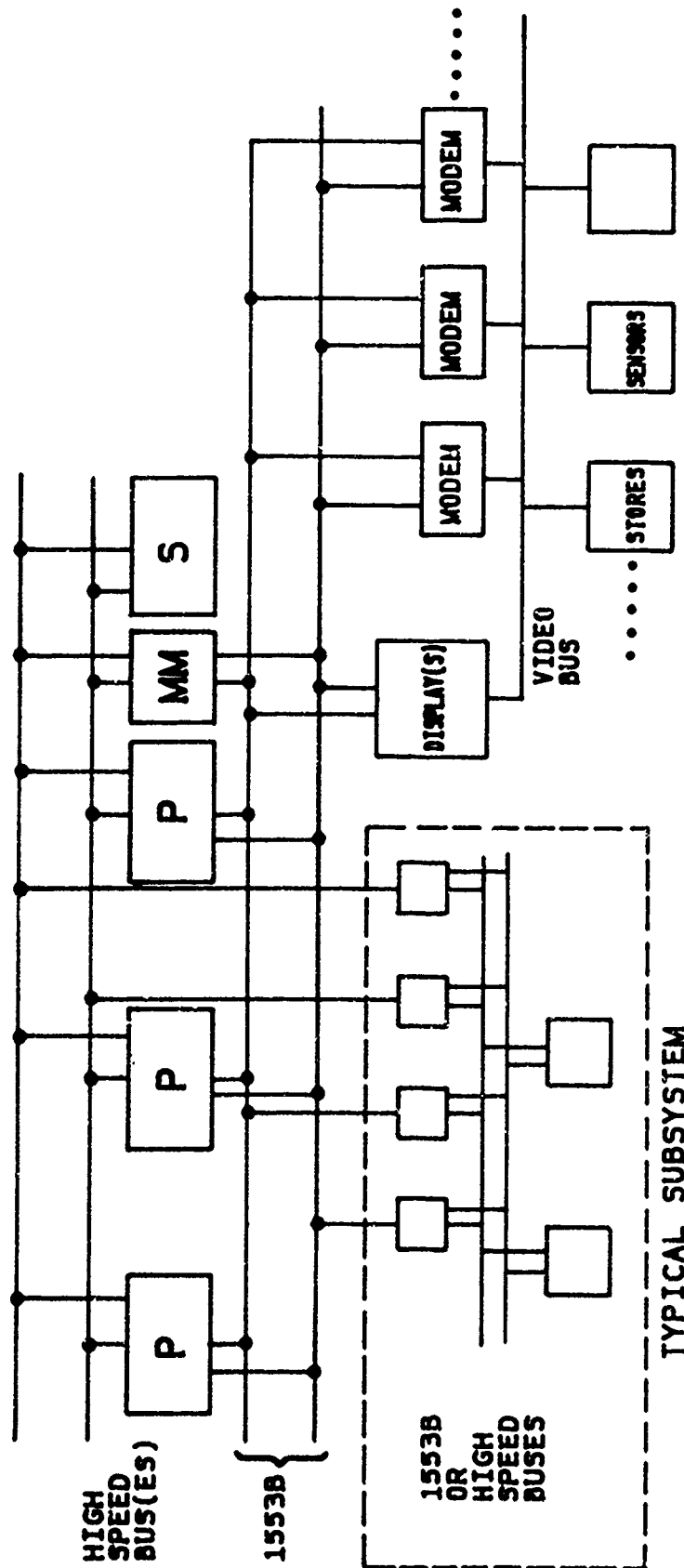


FIGURE 5. AVAILABILITY IMPROVEMENTS THROUGH DIGITAL SYSTEM TECHNOLOGY



P ~ PROCESSOR
 S ~ SUBSYSTEM
 MM ~ MASS MEMORY

FIGURE 6. ADVANCED ARC ARCHITECTURE TOPOLOGY

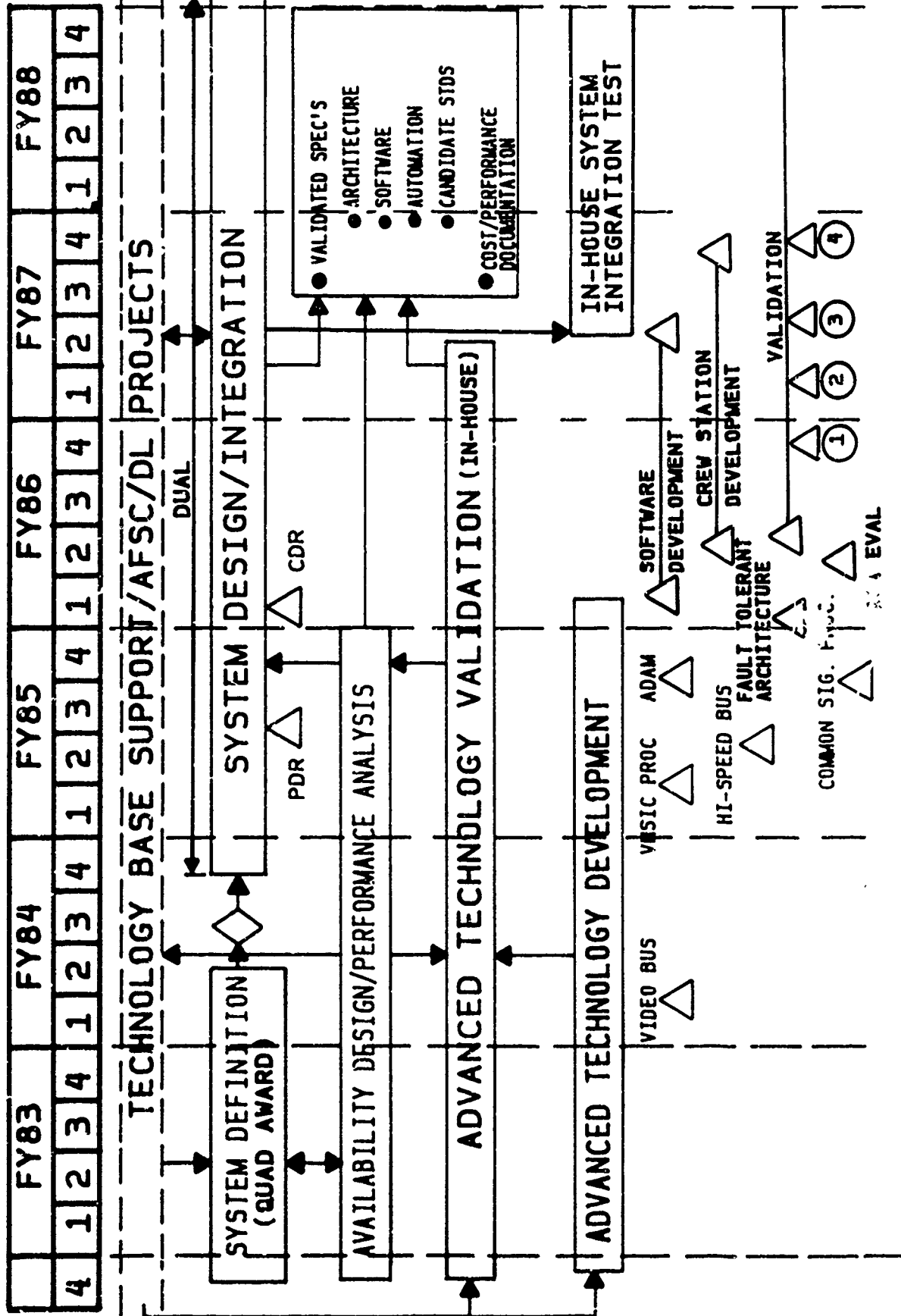


FIGURE 7. PAVE PILLAR PROGRAM FLOW

TECHNOLOGY	SUSTAINABILITY ★	RETROFITABILITY (P31)	WORKLOAD REDUCTION	SURVIVABILITY	STANDARDIZATION VALIDATION
HI-SPEED DATA & VIDEO BUSES	●	●	●	●	●
ADA EVALUATION					●
VHSIC DATA PROCESSORS	●	●			●
COMMON SIGNAL PROCESSORS	●	●			●
FAULT TOLERANT ARCHITECTURE	●		●	●	
ADVANCED CREW STATION	●	●	●	●	
SYSTEM AUTOMATION ALGORITHMS	●	●	●	●	

★ SUSTAINABILITY = f(SPARES, AVAILABILITY, FAULT TOLERANCE, TESTING) **FIGURE 8. DIGITAL SYSTEM TECHNOLOGY PAYOFFS**

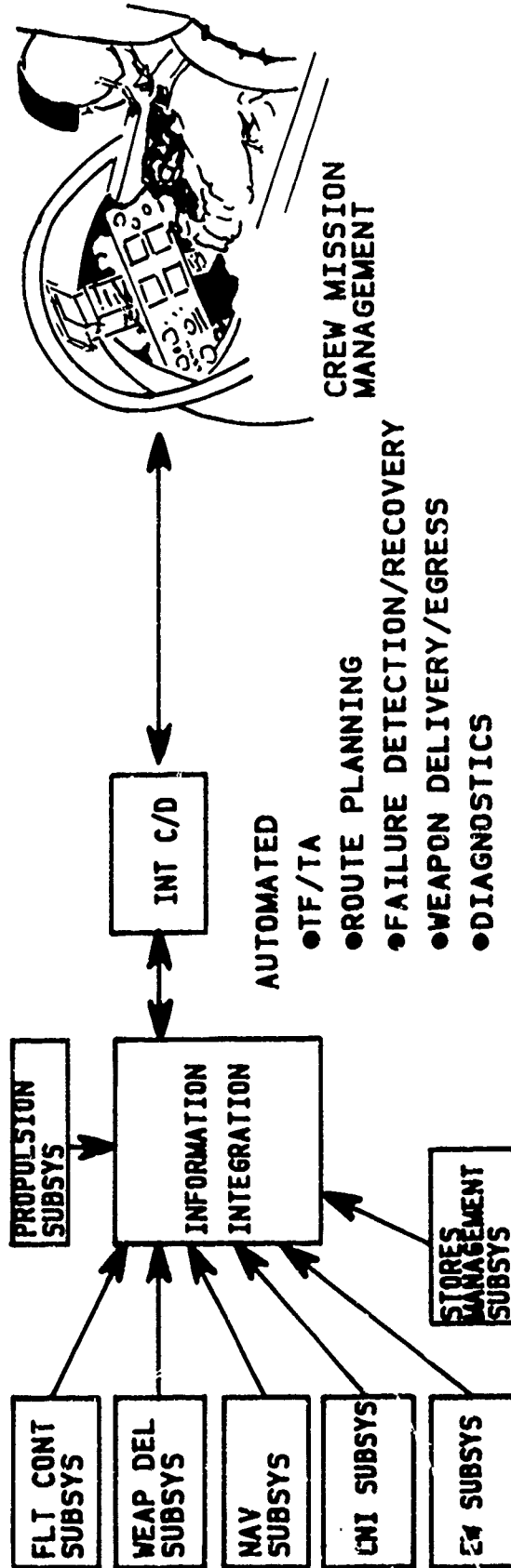


FIGURE 9. COORDINATED SYSTEM OPERATION OF THE FUTURE

FACET
INTEGRATION AND STANDARDIZATION THRUSTS
IN US ARMY AVRADA CNI DEVELOPMENT EFFORTS

Arthur W. Lindberg
US Army Avionics R&D Activity
ATTN: DAVAA-E
Fort Monmouth, New Jersey
07703

The current airborne Communications, Navigation and Identification (CNI) systems in Army, Air Force and Navy aircraft are comprised of a variety of discrete "blackboxes" with little or no standardization of hardware or systems integration among the combat services. The impacts of this lack of standardization are:

- a. continued proliferation of unique devices,
- b. duplicative development and production costs, and,
- c. increased life cycle costs.

The US Army AVRADA has a development program, called Future Airborne Communications Equipment Technology (FACET), containing three major sub-thrust areas:

- a. Integrated Communication, Navigation, Identification Avionics (ICNIA) - a joint AF/Army CNI system utilizing an integrated bus architecture of real-time programmable functional modules to replace the current families of discrete receivers and radio sets.
- b. Digital Multiple Channel Audio System/Digital Audio Distribution System (DMAS/DADS) - a tri-service universal replacement airborne audio system based on the latest voice coding techniques, digitized audio and a distributed network architecture.
- c. Voice Interactive Systems Technology Avionics (VISTA) - an audio enhancement development effort, with high multi-service participation potential, to provide voice I/O capabilities in aircraft for reduced aviator workload, hands-off subsystems control and increased mission effectiveness.

The goals of the FACET program are:

- a. universal application,
- b. shared development costs (multi-service),
- c. simplified system improvement and expansion (modular, software - intensive systems),
- d. weight, size and power reductions,
- e. improved maintainability (low-cost modules, BIT), and
- f. lower life cycle costs (DTUFC, multi-service systems support).

BIOGRAPHICAL DATA

ARTHUR W. LINDBERG

Project Leader for Audio Processing Programs, including DMAS, DADS, and overall responsibility for the FACET Program; Communications/Sensors/Instrumentation Division, Avionics Research and Development Activity, Fort Monmouth, N.J. Lead Engineer on various communications (R/F, Audio) communications systems for new generation Army aircraft. Employed at Fort Monmouth, N.J. from 1968 to present.

Bachelor of Science (Electrical Engineering), Drexel University, Philadelphia, PA, 1968; Master of Science (Electronic and Computer Engineering), University of Michigan, Ann Harbor, Michigan, 1973.

Participating Author:

Military Standard, MIL-STD-1294, dated 9 Mar 81 "Acoustical Noise Limits in Helicopters."

RTCA Document No. RTCA/DO-170, Jan 80, "Audio Systems Characteristics and Minimum Performance Standards - Aircraft Microphones (except carbon), Aircraft Headsets and Speakers, Aircraft Audio Selector Panels and Amplifiers."

AVIONICS CONTROL ARCHITECTURE OF
ARMY HELICOPTER IMPROVEMENT PROGRAM (AHIP)

Glenn P. Tomlin, Jr.

US Army Avionics R&D Activity
4300 Goodfellow Blvd.
St. Louis, MO 63120
314-263-1634

ABSTRACT

The Army Helicopter Improvement Program (AHIP) provides the Army's OH-58 Scout aircraft with a day/night Mast-Mounted Sight, Laser Designation capability, and a fully-integrated digital avionics/sensor control display package. Major functions are controlled and displayed through use of a central Control Display Subsystem (CDS), which consists of two identical Master Control Processor Units (MCPUs), one keyboard control, two multifunction displays.

Avionics systems interface either directly with the MIL-STD-1553B multiplex data bus or through Input/Output circuitry in the MCU. Displays are driven by digital symbol generation within the MCU.

The navigation system for AHIP is a self-contained doppler/inertial system controlled from the MIL-STD-1553B data bus.

The UHF/VHF - AM/FM -- communications radios are standard Army radios controlled through Input/Output in the MCU. An Airborne Target Handoff System (ATHS) (digital data burst), additional FM power amplifier and HF radios are controlled directly from the data bus.

Problems encountered with incorporation of MIL-STD-1553B relate to lack of timely definitions in MIL-STD-1553 of standard multiplex bus data word/message formats and actual hardware implementation (bus couplers, black box connectors).

BIOGRAPHICAL SKETCH

AUTHOR: GLEN P. TOMLIN, JR.

TITLE: AVIONICS PROJECT LEADER, ARMY HELICOPTER IMPROVEMENT PROGRAM (AHIP)
US ARMY AVIONICS RESEARCH AND DEVELOPMENT ACTIVITY
HQ US ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND
ST. LOUIS, MO

EDUCATION: BSEE, 1969, University of Missouri at Rolla

PROFESSIONAL SOCIETIES: INSTITUTE OF NAVIGATION
ARMY AVIATION ASSOCIATION

ADVANCED COCKPIT-SYSTEMSINTEGRATION

G. ROE
Project Leader Advanced Cockpit Studies
British Aerospace P.L.C.
Brough
North Humberside
U.K.

ABSTRACT

The present paper describes two major complementary activities funded by the United Kingdom Ministry of Defence which are being undertaken at the Brough site of British Aerospace. These studies are addressing the problem of pilots task optimisation and the overall system architecture needed to meet the operational requirements of the next tactical combat aircraft. These activities are the Advanced Cockpit Design Studies and the Tactical Combat Aircraft Avionic Demonstrator Rig.

The Advanced Cockpit Studies have been underway for some 6 years. The scope of these studies has been extensive, covering escape system design, alleviation techniques, advance pilot and equipment cooling techniques, information and control task rationalization and the development of workload prediction and measurement techniques. The studies have after a number of iterations culminated in the development of a dynamic cockpit mockup. The studies specifically related to the information and control task rationalisation will be discussed in this paper in some detail.

The Tactical Combat Aircraft Avionic Demonstrator Rig (TCAADR) is presently at the mid point of a 3-4 year evolutionary design programme investigating such topics as, total system integration, standardisation of interfaces, effective sub-system inter communication, graceful degradation of the system and improved maintenance procedures. The architecture being developed has a multi bus hierarchy and implements the 'data' transmission standard 1553B for sub system to sub system and bus to bus communications. A review of this phased programme will be presented with particular emphasis being given to the impact of the pilots needs on the system design and implementation.

1.0 Introduction

The pilot of any aircraft is provided basically to undertake one task "Handle the Unexpected". Yet in the past and many current aircraft this relatively simple task is made extremely difficult by the pilots need to integrate and interpret a proliferation of discrete display devices to enable him to firstly control the aircraft and secondly to be aware of the unexpected occurrence. Additionally, the combat aircraft is expected to fly and survive in an extremely hostile battlefield environment while the pilot performs precision aiming and weapon release tasks. This increase in what is called pilot taskload has occurred because every new combat aircraft which enters

operational service is expected to out fly its predecessors, this is usually defined as an increase in what has been termed its "Mission Effectiveness". The traditional response to this has been to install more avionic systems each of which requires its own display and control facility, while assuming that the pilot is adaptive enough to cope as the system integrator. This trend has produced a near exponential growth rate in displays and controls, Figure 1 illustrates that the design syndrome is not unique to aircraft in the UK.

In addition to creating increases in pilot task loading the trend of increased avionic facilities has created a situation in which one contemporary aircraft utilises a central computer to control about 1000 data transfers. This computer required 300 man years of effort to develop to 50.000 word control program. Whenever a system is modified within an architecture such as this, the control program needs to be modified and revalidated for flight, which is a costly and complex process. It has been recognised that with even more complex systems required to meet the operational effectiveness targets of the next aircraft and that these aircraft will be in service well into the first quarter of the 21st Century a new system design concept was required. This should be capable of simple, inexpensive equipment update programmes to meet new as yet unknown threats and changes in operation role.

The present paper will describe two major complementary activities funded by the UK Ministry of Defence being undertaken by British Aerospace Brough aimed specifically at addressing the pilot task optimisation and overall system architecture problems for the next tactical combat aircraft. These are the Advanced cockpit and Tactical Combat Aircraft Avionics Demonstrator Rig (TCAADR).

2.0 The Advanced Cockpit

The trend towards an ever increasing number of facilities in modern aircraft and an associated increase in the number of controls, switches and display surfaces has led to a self defeating situation in which the pilot has become severely inhibited in his ability to exploit the facilities provided. Further problems have been caused by the need to operate under controls high 'g' loads and over extensive periods of high speed low altitude flight.

2.1 Initial Studies

The overall objective of the present studies has been to produce an easily workable, highly flexible display and control system in which the pilot becomes more of a system manager instead of system integrator. An essential first step in this was the need to develop an understanding of what the pilots tasks are expected to be. This required the generation of formalised "missions" and segments of these missions, purely for the purposes of identifying essential facilities and the pilots actions. It was assumed for the purposes of this activity that the cockpit would utilise addressable display devices, the aircraft systems would be integrated together by means of a digital data transmission system and that where necessary sub systems would contain sufficient intelligence management capabilities to autonomously monitor and control their internal functional capabilities. The initial stage was to develop functional flow diagrams from the missions, which transform the requirements into functional items, this technique is illustrated in Figure 2.

It was then necessary to define what tasks the man and machine are best suited to undertake, the definitions used are presented in Table 1.

MAN GOOD AT	MACHINE GOOD AT
<ul style="list-style-type: none"> ● DETECTING A WIDE VARIETY OF STIMULI ● PATTERN RECOGNITION ● EXERCISE JUDGEMENT ● REACT TO THE UNEXPECTED ● ORIGINALITY IN PROBLEM SOLVING ● APPLY EXPERIENCE TO PROBLEM SOLVING ● FINE SHORT TERM CONTROL ● NON LINEAR CONTROL ● GRACEFUL DEGRADATION UNDER OVERLOAD ● INTUITIVE REASONING 	<ul style="list-style-type: none"> ● LONG TERM MONITORING. ● CONTROL OF REPETITIVE LONG TERM TASKS ● FAST RESPONSE ● DOES NOT TIRE ● RAPID COMPLEX COMPUTATION ● PARALLEL MULTI TASK OPERATION ● FAST RECALL AND STORAGE OF LARGE AMOUNTS OF DATA. ● PRECISE, SMOOTH EXERTION OF GREAT FORCE

Table 1 Distribution of Abilities.

This information in addition to an initial definition of acceptable levels of system automation, allowed a detailed information and task analysis to be performed. In this the functional blocks Figure 3 were assessed to provide a definition of the information and task requirements to adequately fulfil the mission goals of that stage, the task requirements then being allocated to the portion of the system which is best suited to handling them based upon the Human Factors and system criterion previously defined. The major output of this study in the context of the current paper was a detailed description of the pilots information display and control requirements during the various phases of the mission.

2.2 Display Concept

The cockpit layout developed as a result of this and other studies Lyons J.W et al 1980, Roe G. 1981, Roe G. 1981 is illustrated in Figure 4. The display system comprises a HUD presenting primary flight and weapon aiming data. The units currently under investigation have improved fields of view to enable the presentation of external low light T.V. or Forward Looking Infrared images for night time low level high speed flight and in addition these are required to occupy little if any space below the coaming. This facilitates the installation of a head-level display (HLD) in our case a Ferranti COMED (Combined Optical Map and Electronic Display) which provides a detailed full colour topological moving map onto which CRT data such as aircraft track way points etc may be superimposed. In addition when the map is not required, the unit may present sensor or aircraft system data. The two multipurpose displays (MPD's) are positioned either side of the HUD/HLD under the coaming. These displays present all the appropriate systems information during flight. They may also be used in connection with the system control panels on the left console to select system options and detail presentations.

The flexibility offered and the task rationalisation achieved by the display system design may be illustrated with simple examples. Firstly, during normal operation the pilot may select by means of the appropriate "mission phase" key on the Mission Systems Keyboard (MSK) the data relevant to his current flight phase. Figure 5 shows the displays configured for the cruise flight mode. This gives a simple flight data presentation on the HUD and short term navigation data overlaid on the moving map presented on the HLD. The two MPD's 1 and 2 provide long-term navigation and systems monitoring status data, respectively. On selection of the air combat key, the data shown in Figure 6 would be presented. Now available are weapon aiming and manoeuvre management data on the HUD, with attack sensor data presented on the HLD. MPD 1 now provides weapon availability and release data and MPD2 still presents systems status data. During normal operating conditions the fact that many of the sub systems contain resident intelligence and management capabilities allows the pilot to be relieved of mundane monitoring and control tasks presently undertaken. For example it is assumed an advanced fuel management system is utilised, this provides under normal operating conditions the ability to present to the pilot only gross parameters such as total contents. In the present cockpit this would represent one line on the MPD screen instead of possibly six instruments in a conventional cockpit Figure 7.

In the situation where the pilot requires a more detailed appraisal of the system, he may select a full systems diagram, by means of the Systems keys in the 'MSK'. This will show the pilot Figure 7. Much more data in a more coherent fashion than previously possible. This philosophy when applied to systems such as engine control hydraulics and primary electrical power significantly declutters the cockpit. This approach also provides the opportunity to allow the pilot to select corrective procedures and detail data presentation relating to the malfunctioning system and reconfigure the sub systems appropriately. This is achieved by depressing the warning system keys located around the coaming edge when these illuminate.

2.3 Control Concept

The control facilities have experienced a similar rationalisation for a detailed discussion; consult Roe G. 1982. The flight controller is located on the starboard console in an attempt to improve dynamic tracking performance under high acceleration manoeuvre. The control handle contains all those facilities required by the pilot to perform both Air to Air and Air to Ground attacks. To minimise excursions off the flight controller all once-a-flight selectors are located outboard of them, these being arranged in order of frequency of use around the arm rest. These selectors are principally those for the basic aircraft systems which are now self monitoring and regulating. The major avionic system selectors are located outboard of the reduced displacement throttle and may be used in connection with the MPDs and MSK to select and modify system configuration. In an attempt to reduce pilot cross monitoring during selection procedures, console control processors are proposed which perform error and sequence checking ensuring that if the pilot misses visual verification of an invalid change this does not enter the system and an error is signified on the screen when he ultimately selects the accept key.

There is little doubt of the benefits the present cockpit design has to offer in achieving the future mission effectiveness targets. However, although the cockpit and its associated systems have been developed and tested the supporting aircraft system architecture required to exploit these in a cost effective manner was until recently an area of great controversy and receiving little if any practical study. It was in recognition of this and the potential benefits which could be derived from the integration of the system via a data bus network for example weight saving, interface standardisation, improved maintenance, easy system modifications and graceful degradation, that the TCAADR Programme was instigated.

3. Tactical Combat Aircraft Avionics Demonstrator Rig

The intention of the current programme is to develop as a ground rig a totally integrated system to meet the mission requirements of the next tactical combat aircraft with the major objectives being:

1. To reduce the risks associated with the application of new technology.
2. To understand the strengths and limitations of a system based on data buses and particularly those using the transmission standard MIL-STD-1553B.
3. To address the task of constructing a totally integrated system involving in addition to the traditional offensive systems, the control of general services systems such as fuel hydraulics etc.
4. To provide a facility to allow future system and sub system development and study.
5. To demonstrate system acceptability to the pilot.

The development of the Rig is based on an evolutionary approach and progresses via a series of identifiable intermediate stages.

3.1 Stage 1

The initial activity was to derive an intimate knowledge of MIL-STD-1553B and provide a simple but flexible and effective design tool which could be built upon during future stages. The configuration which was designated Stage 1 is shown in schematic form in Figure 8. The system at this level satisfied several purposes. It provided a mini system, connected via a dual redundant data bus to the pilots displays and controls. The sub systems, navigation - fuel, allowed a simple navigation mission to be flown and provides a mix of high and low repetition rate data to be communicated across the bus.

A major achievement of this has been the development of an embryo-executive function which is capable of being developed for later stages of the rig. This function is defined as that in which is vested the control of the total systems to achieve the required overall system state. This function is principally responsible for the pilot becoming a system manager rather than integrator and as such will be discussed in some detail.

The executive controller uses pilot selections and sub system status data feedback in the form of status words and state response words to generate the control commands to the sub systems. The executive function must therefore be capable of coping with any operating condition that may occur and must include sufficient redundancy to meet the system integrity and availability requirements. The executive function interacts very closely with the bus controller in the case of Stage 1 of the programme it is resident within the same unit. The bus controller is responsible for initiating the various information transfers as required by the data MIL-STD 1553B and the timing of message transfers. The information transfers can be commands, status or data words. As stated the commands arise as a consequence of executive control initiative. Much of the rest of the bus communication consists of data transfers between the sub system as a consequence of the normal system operation. The bus controller directs the bus traffic by causing the system to cycle through a set of data sequence tables which are stored within it. In the present implementation fixed sequence labels are used to achieve a reduction in bus traffic and these are organised into tables related to the phases of flight. The appropriate table is passed upon command from the executive which responded as described previously. For example when the pilot selects a new flight phase a flight phase sequence table is mapped from the executive into the operating table Figure 9. In the event of the executive having knowledge of a sub system malfunction it would attempt to reconfigure the present working table by mapping across commands generated within the Translation Address Generator, additionally these commands would be passed when possible to the other mission phase tables to maintain system coherence Figure 10.

It will be seen that each sequence table is a complex combination of fixed and dynamic table generation and in addition the data must be organised into packages of different iteration rates between 64 Hz and 1 Hz . Hence the operational table is arranged on a cyclic basis with the major cycle iterating of the slowest data rate. Within this high and intermediate rate data is packed. Sufficient time is allowed during each major cycle to allow message re-tries and acyclic transactions to take place.

Returning to the present programme, the Stage 1 configuration allows the demonstration of cyclic and acyclic data transfers at various data rates by means of inputs via the cockpit controls and sub system reconfiguration in response to sub system failure. The general purpose computer (ref. Figure 8) provides the "outside world" stimulation to the system in the form of aerodynamic and engine models, outside world display data and the rig command and monitor function.

3.2 Stage 2

The Stage 2 rig is shown in Figure 11. A second data bus has now been introduced to provide the means of controlling the general aircraft services function. The general services has its own bus controller and executive but looks like any other remote terminal on the avionic bus.

Another important feature of this stage is the clear distinction between the aircraft systems and the outside world stimulation. The outside world bus is provided to pass stimulus data from emulation of various sensor packages, to interfaces at the appropriate system where it appears in a realistic format.

Redundancy of the executive and bus control on the avionics bus and of the waveform generation for the display system have been added. In addition extensive monitoring recording and off-line analysis facilities have been developed.

The issues which are to be investigated and resolved at Stage 2 include:-

- a) Data transfer between asynchronous data buses.
- b) Devolution in the event of failure of executive control to the general services executive.
- c) The design and operation of a bus controller which is interfaced to a second bus network.
- d) The design and operation of dual bus controller, in particular the hand - over of control due to malfunction.
- e) The ability of the display to service failure.
- f) Monitoring and analysis of bus traffic.

3.3 Stage 3

The Stages 1 and 2 of the programme are evolving while recognising the final system architecture of the Stage 3 rig. Functionally the system has been split into four groups (Figure 12): Mission Group, Aircraft Group Nav aids Group and Pilot Group with the requirements for a ground rig adding an Outside World Group to provide the necessary stimulations Figure 13 shows a much simplified view of the present architecture.

3.2.1 Mission Group

The mission group of systems involves those facilities which allow the aircraft to perform its mission role and typically include basic sensors for target detection, recognition and tracking. These sensors interact with the weapon aiming function while they may under certain conditions automatically signal safe and effective weapon release and orchestrate the essential electronic defensive aids deployment. To provide realistic signal traffic around the bus network a number of weapon simulations will be developed each capable of "release" and the instigation of typical failure cases. The weapon release system will provide safety critical outputs to the aircraft/weapon station interfaces. It will utilise prebriefed tactical data and pilot inflight selection to prepare attack packages. After receiving committal demand signals from the pilot, the release outputs will be generated.

The defensive aids system will sense and assess the priority of threats within the operational environment. It will additionally select the appropriate counter measures for deployment manually or automatically.

3.2.2 Aircraft Group

This functional group consists of all the basic aircraft systems which are essential to keep the aircraft in flight. This group therefore investigates the problems of integration of principally safety critical systems.

The flight control facility will comprise full authority ACT flight and engine control system. The inner loop of this function will be provided with all appropriate sensor data via the Outside World bus. The flight critical nature of this system means consideration is being given to the optical isolation of this facility from all other bus systems.

The general services sub systems will be arranged so that the management and control functions are distributed and associated with at least two processing units. The processors would be distributed geographically through the aircraft to allow data from the sensors to be collected and co-ordinated. The processors would be configured to carry out one of the main management functions, secondary data control and local data collection while in the event of bus failure it would revert to a primitive reduced mode of operation to ensure system functioning.

The maintenance system will passively monitor various system performance parameters and will store these for subsequent retrieval and analysis.

3.2.3 Nav aids Group

The Nav aids Group provides those functions required to navigate the aircraft and communicate with external sources. The facilities provided consist of inertial sensor processing, radio nav aids communication transceivers and briefing aids.

The inertial sensor function will provide aircraft attitude, body rates and flight vector data, from stimulus data provided over the outside world bus network. The navigation function will provide aircraft heading, velocity and position in an earth reference frame, in addition to required heading, track, ground speed and time to achieve a desired destination or route. The system will be capable of planned or unplanned 'on-top' and 'off track' fixing. The management function will also carry out fault detection and system reconfiguration based on simulated fault demands received via the outside world bus.

The communication system is included principally to load the bus network with relevant pilot demands and will as such emulate the communication control sequences with forward air control, air traffic and ground control.

The briefing aids allow the insertion of pre-flight briefing information into the system and during flight the facility is provided to gather and store intelligence data.

3.2.4 Pilot Group

The Pilot Group for this Stage of the programme will utilise the Advanced Cockpit design concepts discussed earlier in this paper. The area of detailed study will be the display system architecture as this is very dependent upon the design assumptions incorporated. The trade offs between redundancy, cost and failure absorption will be studied. The current system philosophy allows the first failure to be absorbed with no apparent effect to the pilot, and the second failure allows all the information to be accessed but not necessarily on the chosen display surface.

4.0 Future Rig Activities

At the completion of the current TCAADR programme, in about 18 months time, it is believed the UK will have a unique facility capable of addressing many fundamental areas of contention regarding the implementation of MIL-STD-1553B as a system integration mechanism.

As technology in the avionics area moves so rapidly it is felt important that such a tool as that being developed is available to rapidly evaluate new concepts and weigh the cost/benefits of these. Currently there are a number of questions which are being considered for future study.

- . What degree of standardisation is desirable?
- . What are the benefits to be derived from MIL-STD-1750A?
- . What is the requirement for high speed data buses and how should they be implemented?
- . What are the implications of VHSIC technology on system partitioning?
- . What degree of automation should be incorporated and what will be the impact on the pilots operation capabilities?

5.0 Concluding Remarks

The present paper has attempted to review and draw together the cockpit integration aspects of two major research and development programmes currently underway at British Aerospace Brough. These being the Advanced Cockpit Studies and the Tactical Combat Aircraft Avionics Demonstrator Rig.

One of many lessons learned thus far from both programmes is that as 'one integrates' systems more a very detailed understanding of the pilots needs and abilities at various stages of a mission is required. This is seen to be important if the apparently limitless potential offered by integrated systems is to be realised and exploited in an effective manner.

The TCAADR programme is providing a detailed understanding of how to exploit the potential offered by MIL-STD-1553B as a system integration tool. In addition, it is providing a unique system design and assessment facility for Future Studies. While the Advanced Cockpit Studies are intended to develop an easily workable affordable interface between the pilot and his aircraft inboard systems. A major criterion for measuring the success or otherwise of the concepts developed in the TCAADR programme will be pilot acceptability. This will be gauged by how well he performs the tasks expected of him and just as important by whether he feels 'at one' with his systems. The fact that the two activities are closely integrated at British Aerospace, Brough is seen to be extremely important.

Although it may not have been conceived as such, our work is beginning to show that the MIL-STD-1553B not only allows total cost effective system integration and weight savings but it is also providing an interesting catalyst. This is seen as allowing the development of a truly Symbiotic relationship between the pilot and his aircraft, dramatically improving mission effectiveness.

6.0 References

1. Lyons, J.W and Roe G., 1980 "The Influence of Visual Requirements on the Design of Military Cockpits" AGARD-AG 255 page 12-1 to 12-25.
2. ROE G., April 1981 "Displays for Avionics", Display Application and Technology, Volume 2, Number 2 IPC Press Pages 159-165.
3. Roe G., May 1981 " Design of the Future Military Cockpit" AGARD CP 312 (Supp) page 2-1 to 2-14.
4. Roe G., April 1982 "Head Up Hands Back Control Concept" AGARD CP 329 page 21-1 to 21-12.

7.0 Acknowledgements

The opinion expressed in this paper are the authors and do not necessarily represent those of British Aerospace. Permission to publish the paper is by courtesy of British Aerospace.

The author would like to acknowledge the help of Mr J.D Bannister, Project Leader TCA Rig in producing this paper.

AUTOBIOGRAPHICAL NOTES

Mr. G. Roe T.Eng(RET). A.M.R.Ae.S.,

Project Leader - Advanced Cockpit Studies

Future Projects Department,
British Aerospace P.L.C.,
Aircraft Group,
Kingston-Brough Division,
BROUGH,
North Humberside.
HU15 1EQ
U.K.

The author has been with British Aerospace for 16 years during which time he served an engineers apprenticeship and had experience of various technical departments. He became a member of the Future Projects Department 6 years ago where he was given the responsibility to design and develop Advanced Cockpit Concepts for a range of project aircraft many of which will not enter R.A.F. Service until the first quarter of the 21st Century. The author is a registered Technical Engineer and an Associated Member of the Royal Aeronautical Society. He has presented a number of papers to various AGARD panels, to other societies and associations, while he has had technical papers covering human factors to display techniques published in national and international journals. 704

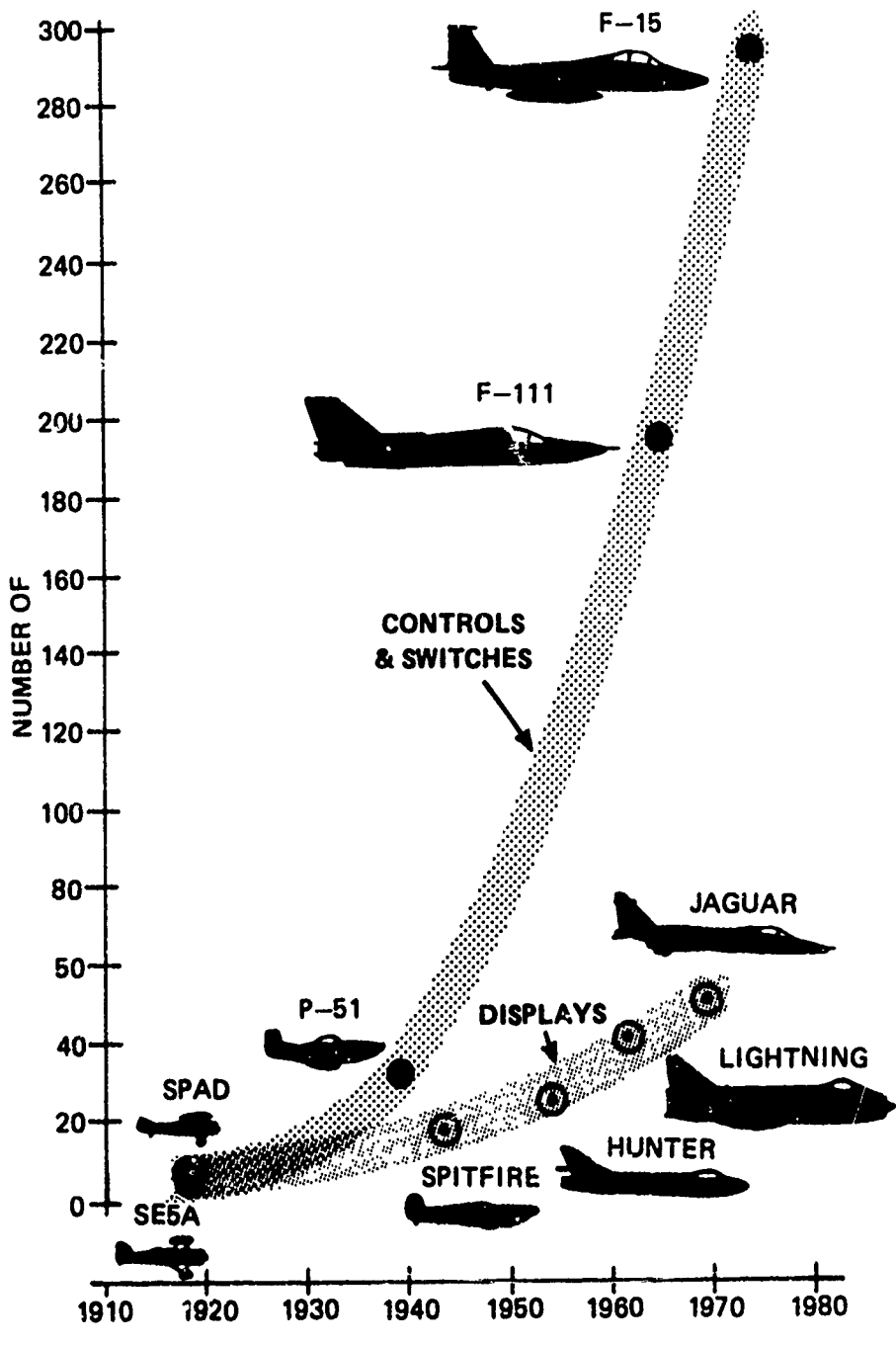


FIG. 1 Cockpit Facilities Growth.

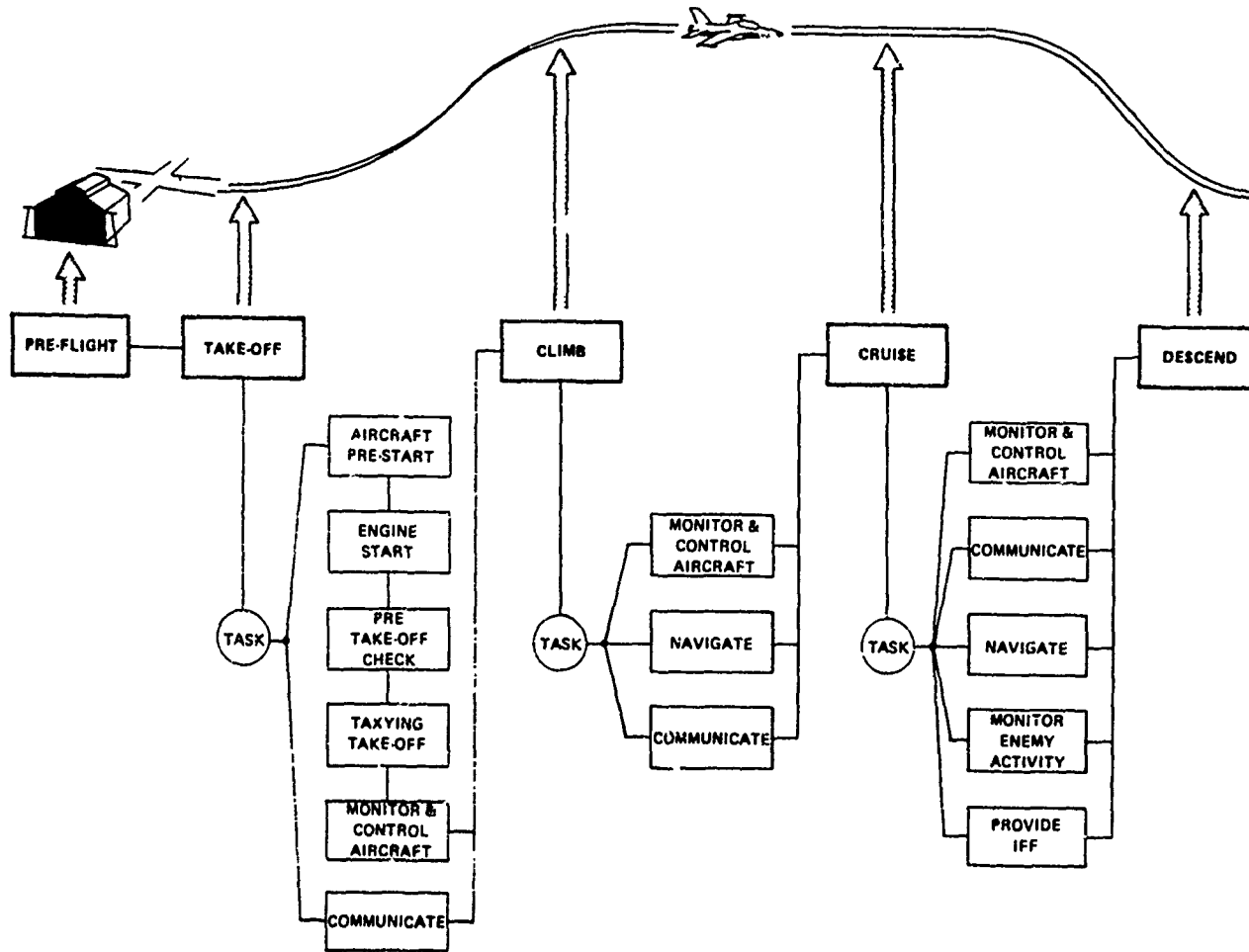
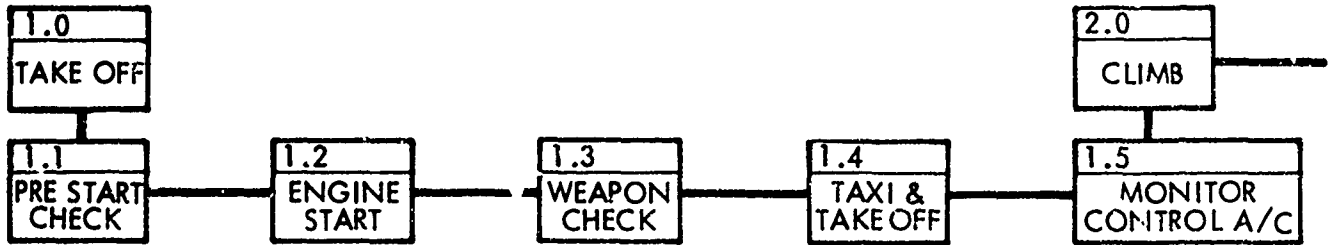


FIG.2 Functional Task Analysis.



Function Number	Function Name	System Information Requirements	System Action Requirements	Machine Tasks	Man Tasks
1.1	Prestart check	Cockpit check: 1. Integrity of system 2. Switches and controls 'safe' before power on 3. Position seat and rudder 4. Umbilicals 5. Straps 6. Brakes	1. Place all switches and controls to safe 2. Adjust seat and rudder 3. Connect umbilicals 4. Connect straps 5. Ensure brake 'on' Measure, monitor, inform, select, test and warn	Measure, monitor inform, select, test and warn.	1. Inspect 2. Set switches to safe 3. Adjust seat to attain eye level 4. Connect umbilicals 5. Fasten straps 6. Apply brakes 'on'
1.2	Power 'on'				

FIG.3 Information / Task Definition.

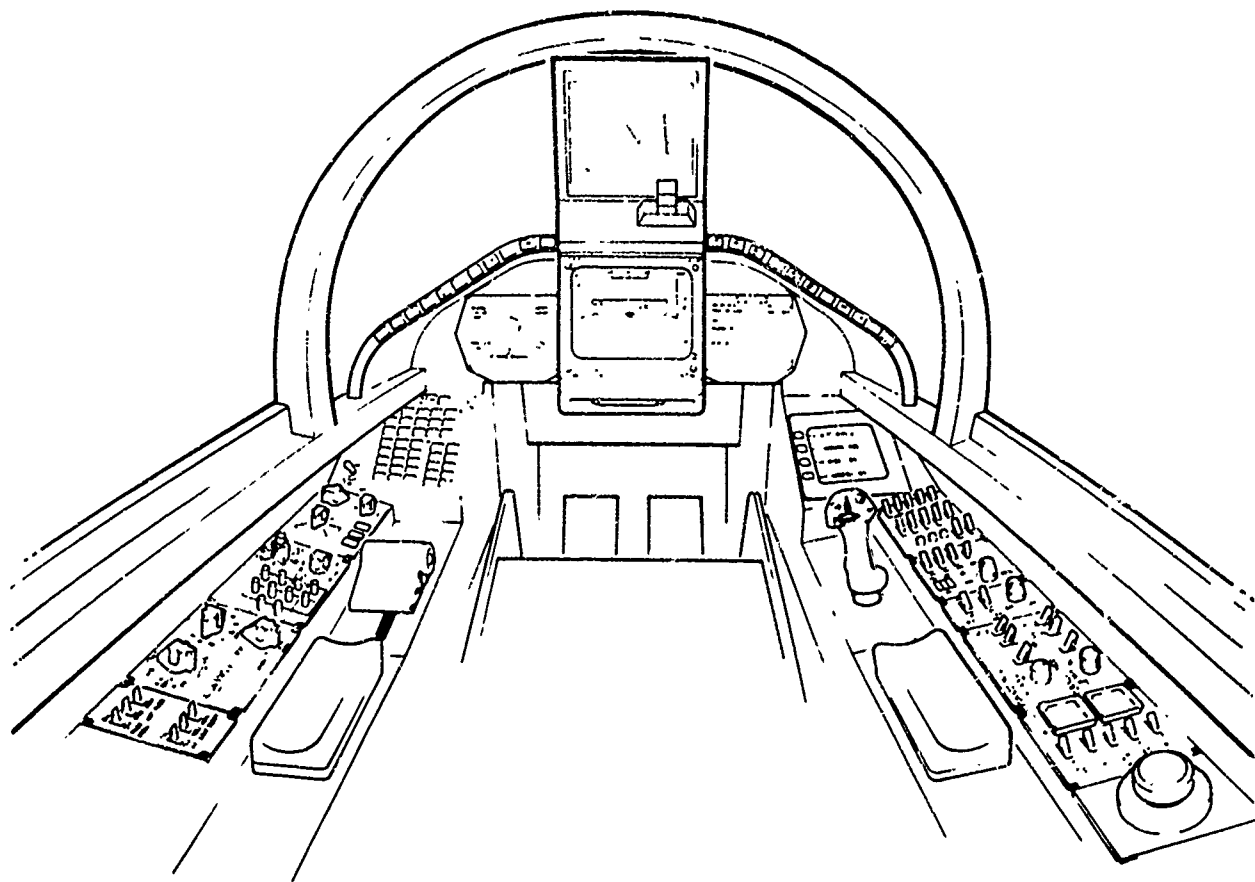


FIG.4 Advanced Cockpit.


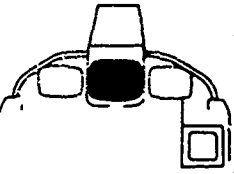
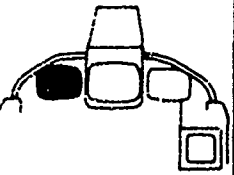
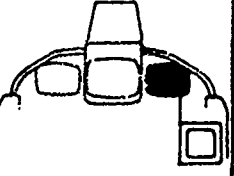
	FLIGHT MODE	CRUISE
DISPLAY	Head Up Display 	A/C Symbol Pull Away Heading Pitch Precision Course Steering Vectors Radar On And Range To Target
	Head Level Display 	MAP SHORT TERM NAV. Map With Waypoints Known Threat Data Targets Present Position Distance To Go Time To Go Heading Speed (Knts) Height
	Multi Purpose Display 1 	LONG TERM NAV. Pre Planned Flight Plan (Grid North Up) Waypoint Update Points Targets Base Present Position Required Heading Actual Heading Nav. System Mode A/C Speed Height Time & Time Next Feature
	Multi Purpose Display 2 	A/C SYSTEMS A/C Speed Height Heading Present Position Fuel Remaining Comms Status Thrust Level Time GMT and E.T. Advisory Warnings

FIG.5 Displays Cruise Flight.


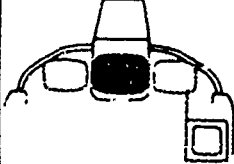
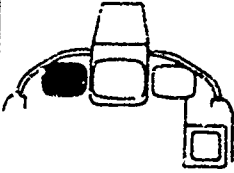
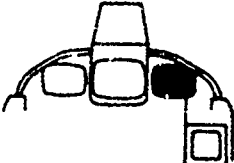
	FLIGHT MODE	AIR COMBAT
DISPLAY	Head Up Display 	A/C Symbol Pull Away Heading Air Manoeuvre Management Data Weapon Release Data
	Head Level Display 	<u>SENSOR DATA</u> Radar Modes Targets Position Target Ranges A/C Symbol Pitch & Roll Heading Speed Height
	Multi Purpose Display 1 	<u>WEAPON STATUS</u> Weapon Location Weapon Type Weapon Availability Weapon Launch Status External Threat Data
	Multi Purpose Display 2 	<u>A/C SYSTEMS</u> A/C Speed Height Heading Present Position Fuel Remaining Comms Status Thrust Level Time GMT and ET Advisory Warnings

FIG.6 Displays Air-A: Combat.

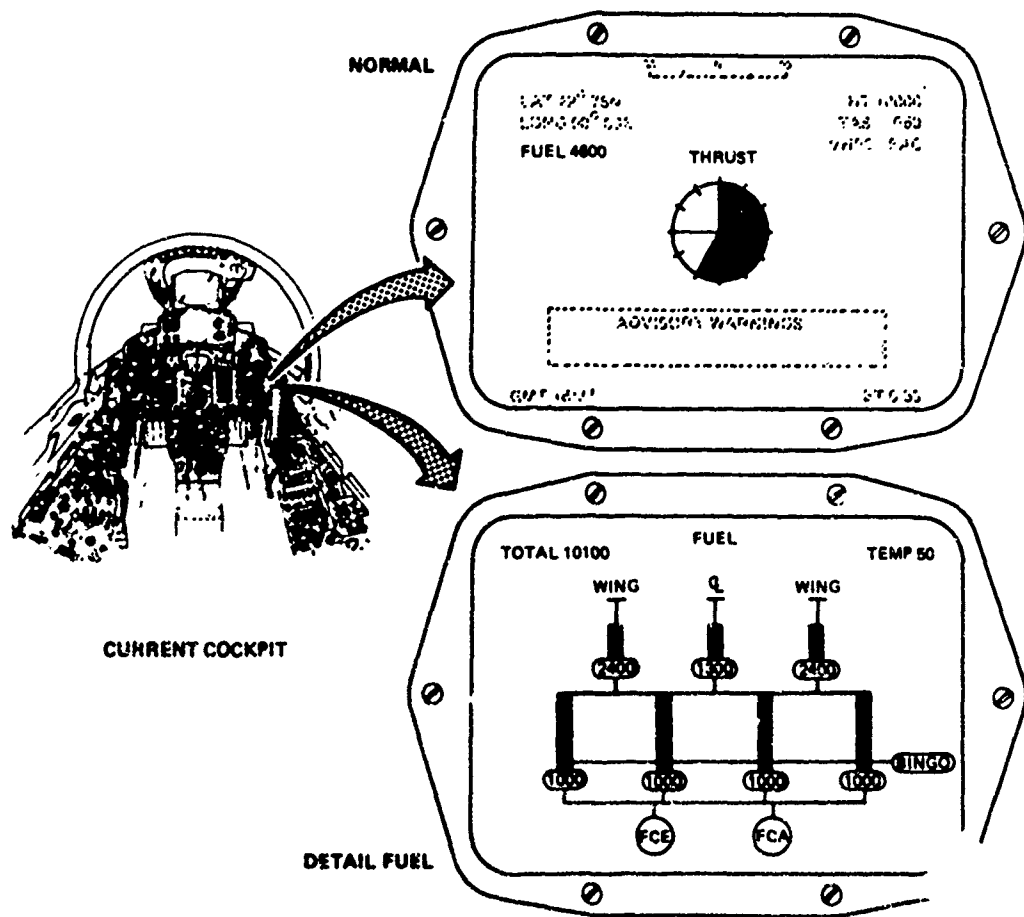


FIG.7 Fuel System Data Rationalisation.

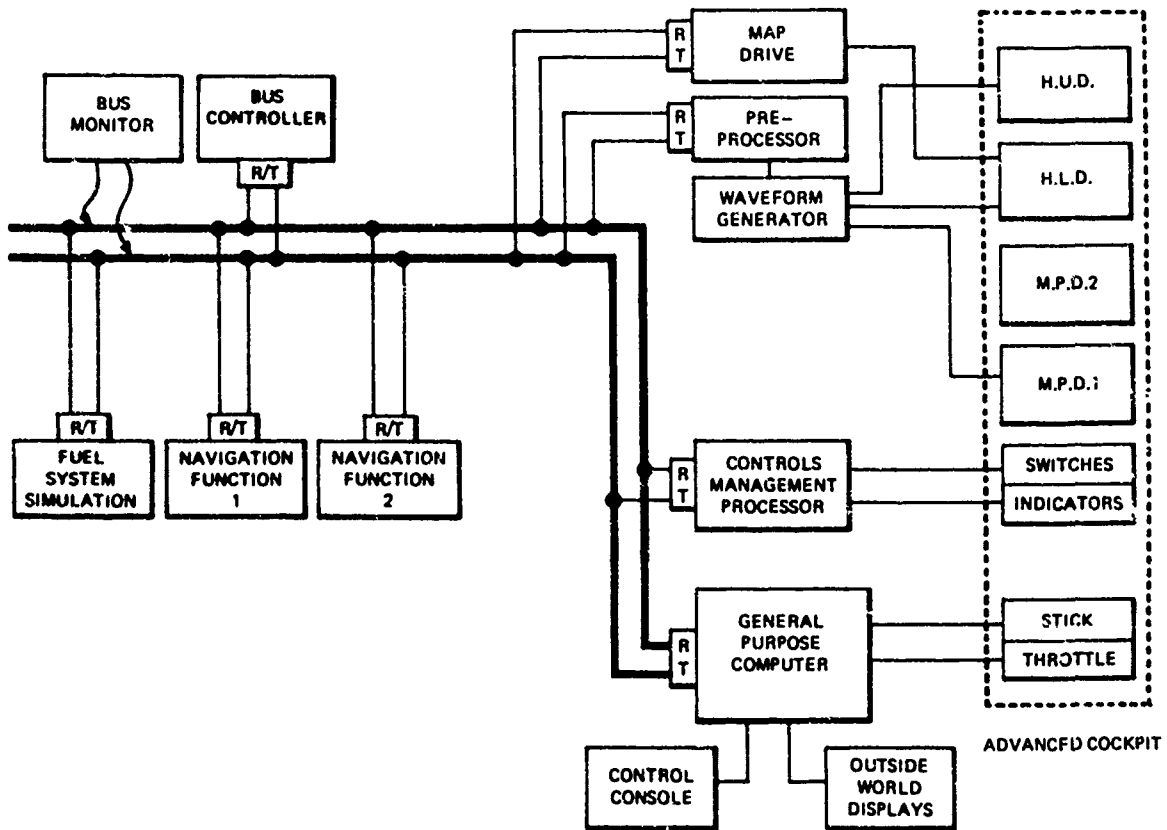


FIG.8 TCAADR. Stage 1 Architecture.

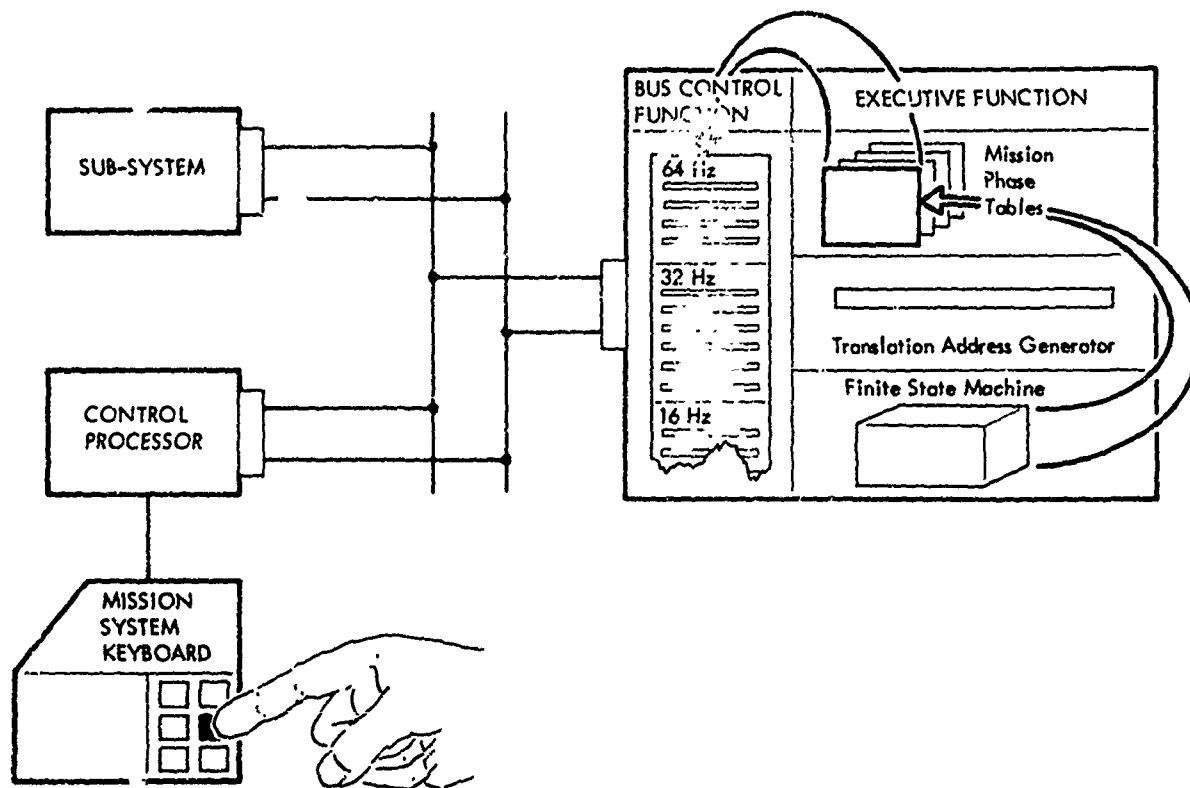


FIG.9 Mission Phase Selection Sequence.

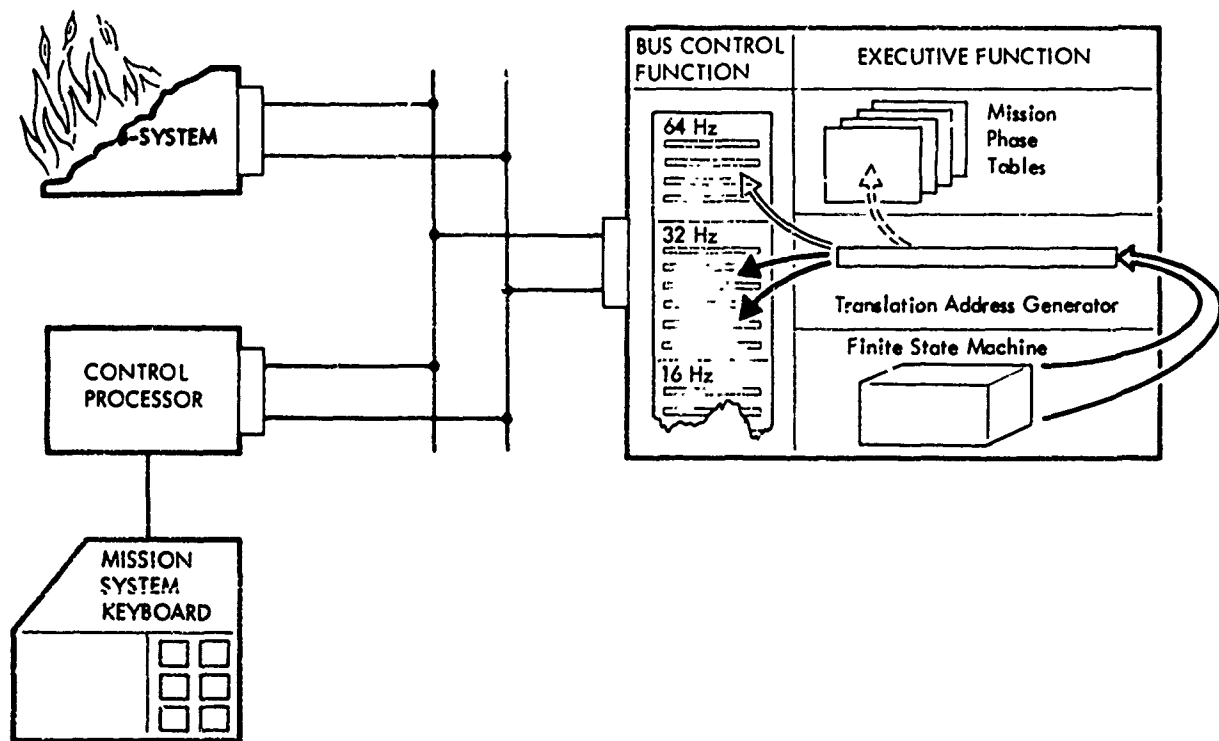


FIG. 10 Sub-System Failure Sequence.

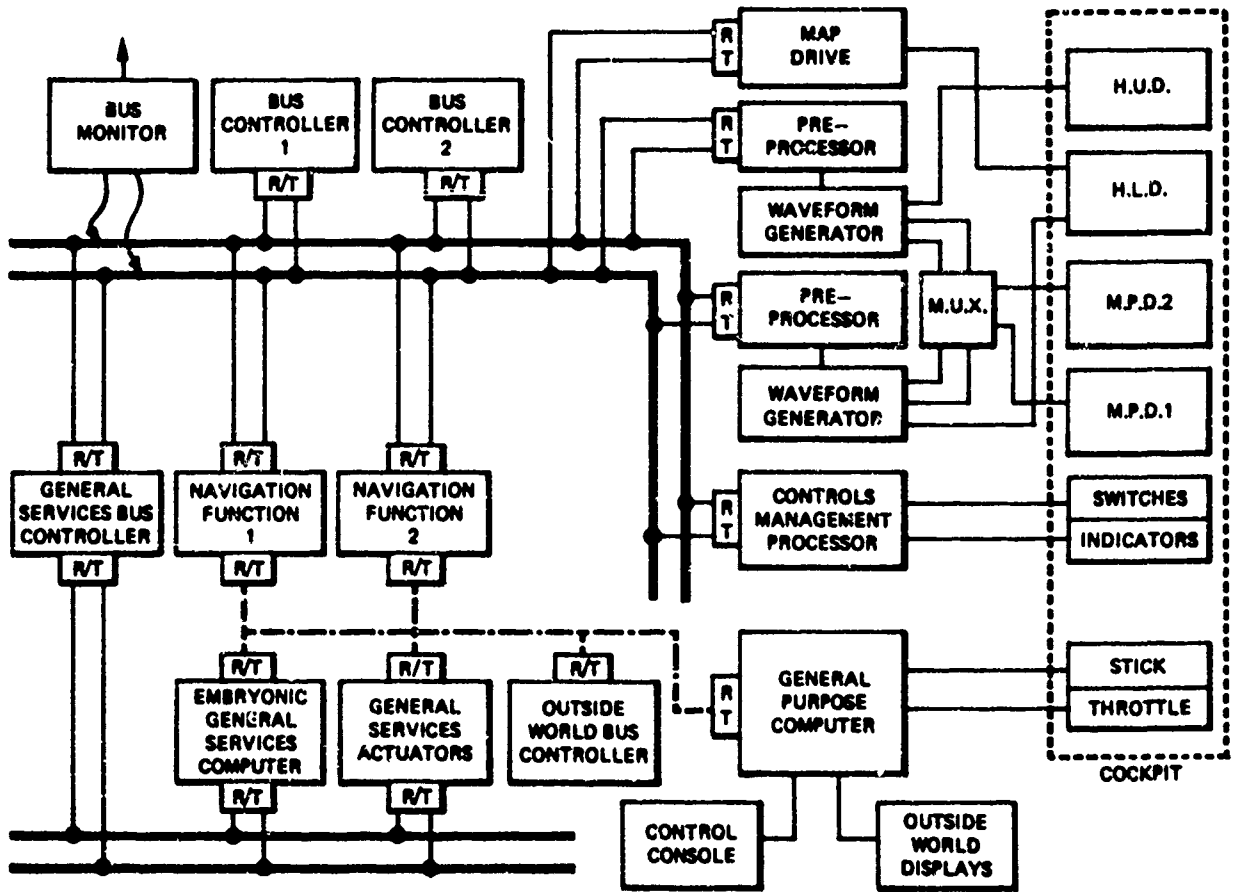


FIG. 11 TCAADR Stage 2 Architecture.

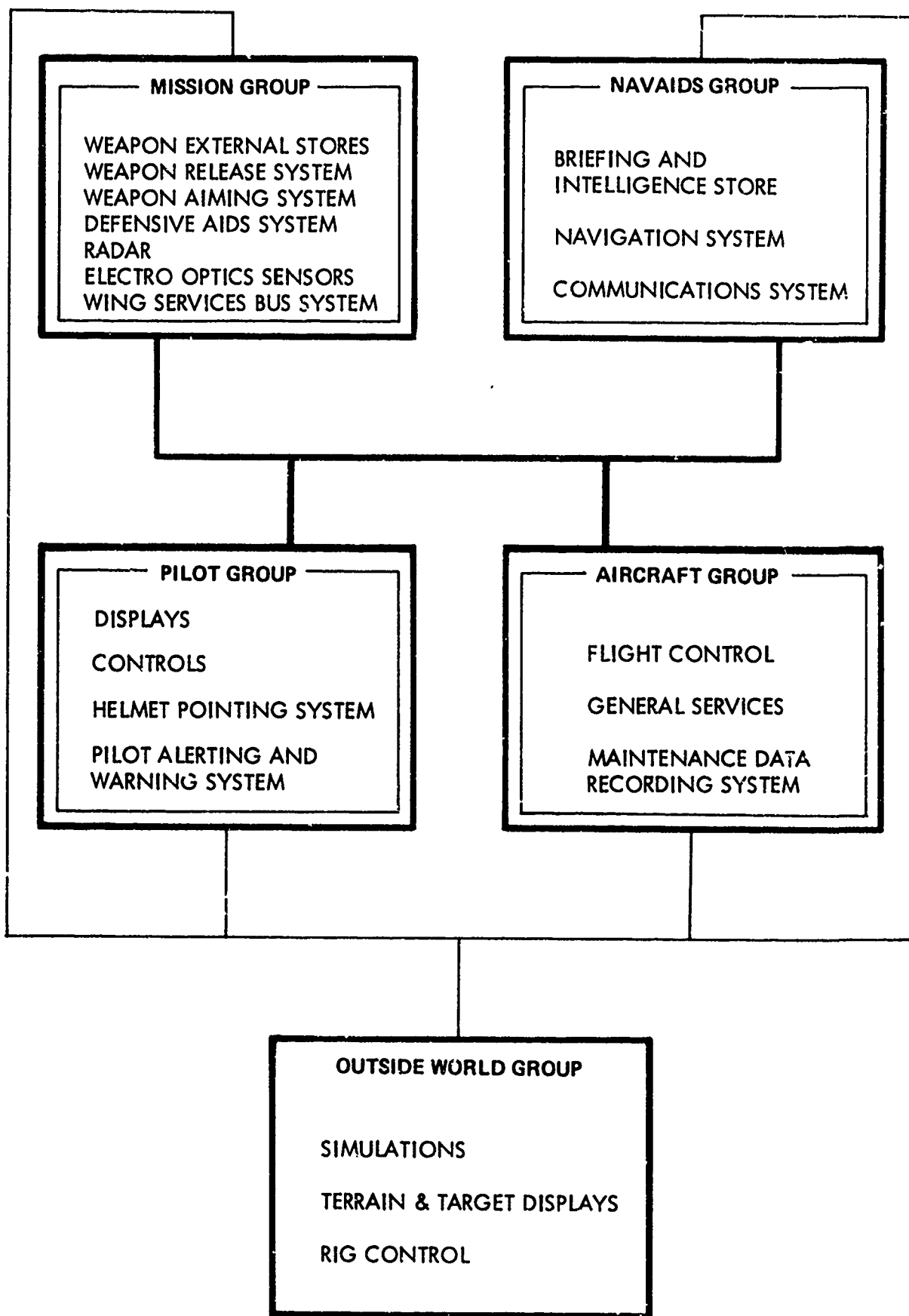


FIG. 12 Functional System Architecture.

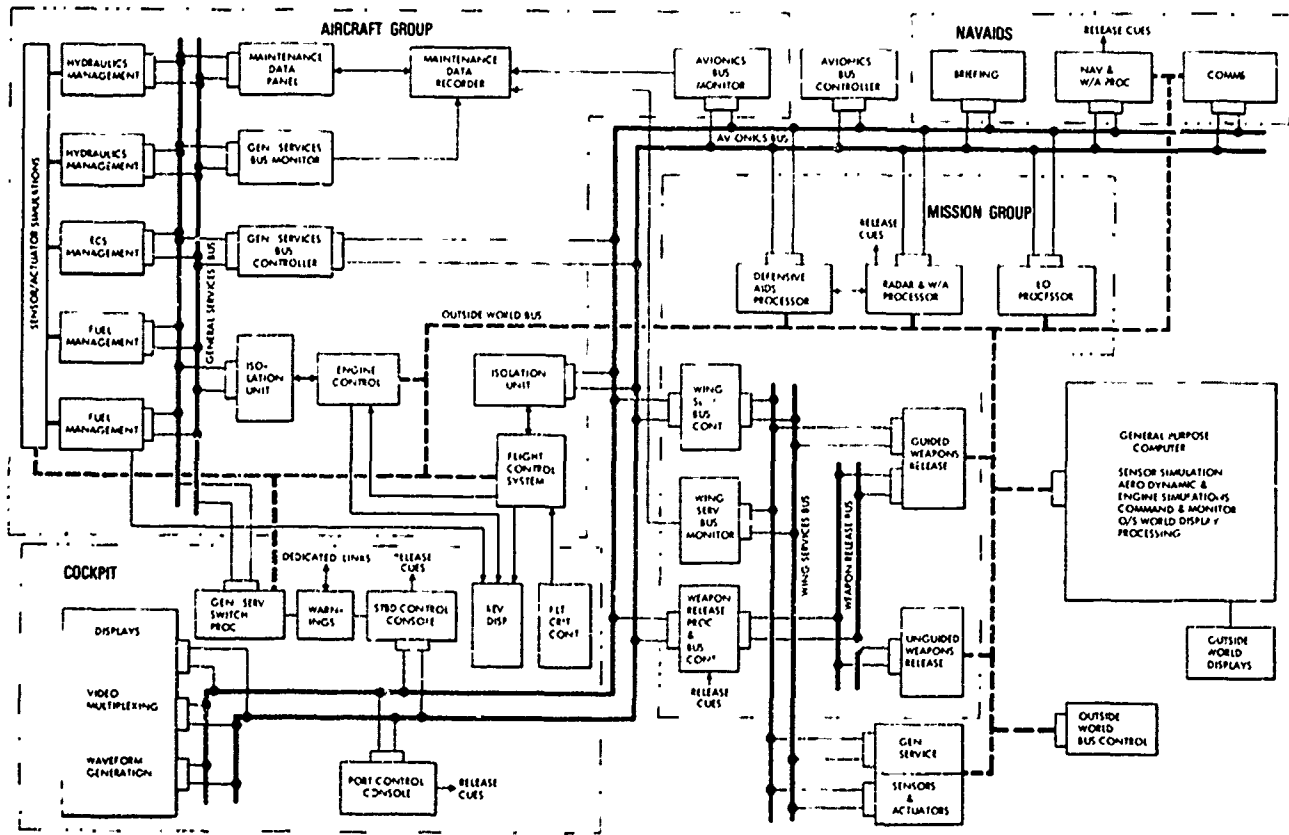
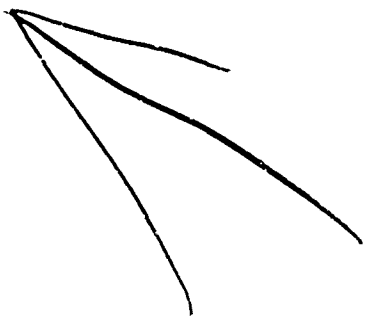


FIG. 13 TCAADR Stage 3 Architecture.



THE DIGITAL INTERFACE CHALLENGE

**SESSION CHAIRMAN: Major Lee Cheshire
ASD/ENAS**

**MODERATOR: Colonel J. Clifford
ASD/AF**

AD-P003 570

DEFENSE INDUSTRY ATTITUDES

ABOUT AIR FORCE INTERFACE STANDARDS

REPORT OF AN ELECTRONICS INDUSTRIES ASSOCIATION SURVEY

Peter N. Pocalyko

IBM Corporation
Federal Systems Division
6600 Rockledge Drive
Bethesda, MD 20817
(301) 493-1460

Chandler E. Swallow, Jr.

Sperry Univac
Defense Systems Division
Eastern Engineering Center
1555 Wilson Blvd.
Arlington, VA 22209
(703) 558-7251

P. N. Pocalyko - Technology Directorate, Federal Systems Division, IBM; over 20 years in research and development of data processing hardware and software for Department of Defense and NASA. Involved in industry interactions with government military standards since 1970.

C. E. Swallow, Jr. - Systems Engineering, Defense Systems Division, Sperry Univac; 20 years experience in the military in tactical data systems research and development and 12 years with industry engineering and program management of computer military systems development. Involved in NATO tactical systems interface standardization since 1974.

ABSTRACT

Major General Welch USAF, Asst. Deputy Chief of Staff for Research, Development and Acquisition, has asked the Electronic Industries Association for policy level participation in the Air Force's avionics standards program. This paper reports on the initial step of the response of industry. It analyzes a survey made under the sponsorship of the EIA. Defense industry managers and senior engineers experienced in the development and production of mission-critical avionics and software were questioned about their experiences and opinions concerning the Air Force standards for J-73 (JOVIAL), ADA, 1553 Data Bus, and 1750 Instruction Set Architecture. The responses are cross-correlated with experience levels and nature of the respondent's field of expertise.

Results are presented as a summary of current attitudes which can serve as data base for focusing issues for further discussion with industry.

INTRODUCTION

The genesis of this paper was a briefing at the Technical Council of the Electronic Industries Association by Major General J.A. Welch, USAF, Assistant Deputy Chief of Staff, Research Development and Acquisition. He made the point that the Air Force must promulgate policies that can apply to the different industry groups involved in avionics standardization; that is, prime contractors, avionics systems contractors and subsystem/component manufacturers. Therefore, he indicated, the Air Force needs policy level industry participation to address the interests of each of these groups. The Air Force is concerned with such issues such as resource allocation to foster and promote standards, transitioning from current to future standards, and incorporating improvements in reliability and maintainability of avionics.

To achieve the needed industry interaction with the Air Force, General Welch requested EIA establish a panel/committee to provide industry feedback on avionics standards. He also requested that EIA initiate a symposium where industry could exchange information and experience with standards implementation. He concluded his remarks with a request for direct EIA feedback to him and other individuals in the Air Force concerned with avionics standardization. The technical Council offered the General full support and formed a small task group to work out the best means for responding to the General.

About the same time during an unassociated address to the National Research Council's Board on Computer Science and Technology, Mr. Peter McCloskey; President of the Electronic's Industry's Association was voicing the view that there is a wide diversity of opinion as to how to provide computer systems to the government among its member companies which represent some of the government's largest contractors and some of their smallest.

He noted "On one side, some companies agree that a firm policy of direction, requirements and standardization are in their best interest. There is also a middle group that believes moderate regulation and standardization is beneficial, and gives them the necessary latitude to be competitive in this very volatile market." He went on to remark that "A third side is composed of those companies favoring an open market concept in which the company with the product considered the cheapest and the best wins the bid."

With these perceptions in mind and following more detailed discussion with MGEN Welch, in which he focused his concerns on interface

standards, it was decided that an initial step to further the discussion with the Air Force should be an industry survey. After some deliberation, it was decided to direct this survey to persons in industry who are knowledgeable of those Air Force interface standards aimed at mission-critical avionics and software. MGEN Welch pinpointed the following standards as those he would like feedback about; namely the J-73 (JOVIA) and ADA programming languages, MIL-STD 1553 Data Bus and MIL STD 1450 Instruction Set Architecture. The emphasis of the questionnaire which was sent out, reflected these concerns and followed the line of questioning he suggested.

Companies were asked to solicit responses to the questionnaire from experienced managers or senior engineers involved in Air Force programs. In order to obtain reasonably straight-forward answers and encourage uninhibited replies, respondents were asked to submit their answers anonymously. The report, of course, will be on the usual EIA not-for-attribution basis. However, an attempt will be made to correlate replies of big contractors and small contractors to see if there are significant differences.

SURVEY QUESTIONNAIRE

The complete questionnaire is included here in the proceedings. The first part covers some questions about the respondent, mostly to classify him by experience and job description. The next part asks about his perceptions of the overall intent of the Air Force's Interface Standards Program. This was included because early discussions of this matter amongst the EIA Technical Council members and some of their staffs indicated this might be an area of disparate views.

Following these questions are a number which call upon the technical judgement of the respondent as to what interfaces and elements of the computer systems ought to be standardized. Finally some opportunities are given for suggestions (and comments) about the Air Force's standards.

The instructions to the respondents and the questions submitted to EIA companies are as follows:

***REQUEST-** Respondent's familiar with one or more of the following interface standards are asked to fill in answers to the following questions. Since it is expected that viewpoints may vary widely, responses from different vantage points e.g. marketing, systems engineering, contracts hardware design, software production, etc. are welcomed.

*RESPONDENT'S PROFILE

1. Are you familiar with any of the following Air Force interface standards? (Please circle appropriate answer)
- A. General Specifications for a Aircraft Multiplexed Data Bus (MIL-STD 1553)
- a. No b. A little bit c. Some what
d. Have worked on products conforming to standards

- B. Instruction Set Architecture (MIL-STD 1750)
 - a. No
 - b. A little bit
 - c. Some what
 - d. Have worked on products conforming to standards
- C. JOVIAL Programming Language (J-73)
 - a. No
 - b. A little bit
 - c. Some what
 - d. Have worked on products conforming to standards
- D. ADA Programming Language (MIL-STD - 1815)
 - a. No
 - b. A little bit
 - c. Some what
 - d. Have worked on products conforming to standards.

2. Years in data processing business? _____ , years working on military systems? _____

3. Area of current job? _____

COMMENTS ON STANDARDS (assumed to apply to the standard with which the respondent has greatest experience unless otherwise noted.)

- 4. What do you believe was the reason why the Air Force established the particular interface standard?
- 5. Do you believe it is an enabling or constraining standard? In what way?
- 6. How do you feel about modifications and changes to standards?
- 7. Suggested areas of change (Please elaborate)
- 8. What circumstances warrant replacement of a standard?
- 9. Do you feel the Air Force standards encourage or discourage competition?

What about innovation?
Insertion of new technology?

10. From a system viewpoint where are good places to position an interface? e.g.

- a. between the pin connectors and cards?
- b. between functional modules and cabinet back panel wiring?
- c. between cabinets in a sub system?
- d. between sub systems in a major system?
- e. at connection points to a data bus?
- f. other?

11. What elements of a system ought to be standardized?

- a. chips?
- b. cards?
- c. functional processing units?

- d. data bus?
 - e. other?
12. As a interface between processing equipment and people, i.e. programmers, which approach do you favor?
 - a. standard instruction set architecture?
 - b. standard compilers?
 - c. standard higher order languages?
 - d. other?
 13. As a manufacturer would you prefer to see a "build to print", "form, fit and function" or "accreditation to a standard instruction set architecture" approach to data system element standardization.
 14. Will the availability of competent VHSIC circuits change your attitudes to your answers to the questions on this page?

In what way?
 15. What other standards in this area do you think the Air Force should endorse?

When might they be needed?
 16. Do you consider these Air Force standards as guidelines, procurement specifications, or across the board standards?

Firm or subject to change?
 Permanent or evolving?
 Mandatory or suggested?
 17. Any other comments you may wish to make about these standards?
 18. Do you have any suggestions as to how industry might help the Air Force interface standards program?
 19. Any other comments, issues that should be addressed with the Air Force Avionics Standardization?"

CONCLUSIONS AND EXPECTATIONS

This project is expected to provide a good summary of current attitudes which can serve a data base for highlighting issues for further Industry/Air Force exchange of ideas on this very important and technically changing area of Department of Defense tactical system acquisition.

At the time of preparation of this report for the proceedings of this conference, a large majority of EIA companies had not responded to the initial questionnaire mailing. In fairness to late respondent's, detailed results are not published here. Rather, they will be

collated after a broad data base has been thoroughly analysed. Graphics summarizing the results will be presented .

Some of the concepts emerging from the early returns are as follows:

- a. The perception of Air Force motivations in establishing the interface standards emphasize "easing integration" for the data bus standard and "software portability" for the other standards. Lessor reasons given are to "reduce life cycle costs" and "reduce proliferation".
- b. There is a strong feeling that the standards are enabling. The reasons vary but are expected to focus on the fact they are providing a "bench mark or common base". A very positive indication that the standards encourage competition is shown by the returns but at least more than 50% are concerned that they may constrain innovation. Others see innovation allowed within the confines of these standards as a plus. More analysis is needed to extract the reasons for a strong response that the standards do not inhibit the insertion of new technology.
- c. The majority somewhat grudgingly accept the need for change, i.e. "necessary evil", "inevitable", but plead for control. Almost all imply that an old standard is not replaced until a new technology offers a better substitute or can supplement an earlier function.
- d. An overwhelming majority say interface standards should address connections between subsystems and to a data bus. Similar support indicates some specific aspects of functional processing units and most characteristics of data buses should be standardized.
- e. Respondents support very strongly standardization of Higher Order Languages and to, a lesser degree, Instruction Set Architectures. Accreditation to a standard ISA seems to be a preferred way (2 to 1) to define acceptability of manufactured data system equipments. VSEIC is not expected to change these attitudes by a majority of the answerers. However, the reasons behind this answer will need more analysis of the data.
- f. Mixed in the data are indications that there is some industry uncertainty about the permanence of the Air Force standards, the degree and extent that they will be applied and the policy on waivers and exceptions. The final report will try to delineate these concerns.
- g. Many suggestions, mostly in very brief form, but wide ranging were included in the responses. Areas of high interest for further standards are a higher performance data bus and software environment items. Increased industry participation in both the formulation of new standards and the updating of old was a noticeably frequent suggestion.

The final analysis of the questionnaire responses will discuss these areas and possibly others in more detail. A final report will form the basis for the EIA Technical Council response to Major General Welch.



DIGITAL AVIONICS DESIGN FOR VALIDATION

Ellis F. Hitt
Battelle, Columbus Laboratories
505 King Avenue
Columbus, Ohio 43201
(614) 424-6595

B.S., Electrical Engineering, University of Kansas (1960)
M.S.E., Air Force Institute of Technology (1962)

Mr. Hitt is a Projects Manager with extensive experience in conceptual, preliminary, and final design of avionics including navigation, guidance, control, communications, controls and displays, and electrical power subsystems; integration, testing, and analysis of avionics; development of mathematical models and computer programs for performing error analysis, systems simulation and evaluation, life-cycle costs analyses; mission software design, development, verification and validation.

Mr. Hitt's recent work at Battelle has included: development of a "Handbook of Validation Processes for Advanced Digital Integrated Flight Control and Avionics Systems--Volume I"; Digital Avionics Executive Software and Programmable Graphics Generator Software; Digital Avionics Control System; VOR/DME Dynamic Navigation Signal Model; Cockpit Display of Traffic Information; Integrated Control Core Software Concept Development; Comparative Analysis of Techniques for Evaluating the Effectiveness of Aircraft Computing Systems; Simulation Methods for the Validation and Failure Effects Analysis of Advanced Digital Flight Control and Avionics Systems; Formulation of a Methodology for Evaluation of the Mission Effectiveness of Fault Tolerant Integrated Control Systems.

Mr. Hitt has authored numerous papers. He is Battelle's delegate to the Radio Technical Commission for Aeronautics.

ABSTRACT

The designer/developer of fault tolerant avionics must consider the requirements to validate these digital systems. These requirements should be primary factors influencing the design and hence the sustainability of these systems. This paper presents a synopsis of a methodology of design and validation of digital avionics and flight control systems based upon early consideration of validation requirements. Avionics developed using this methodology will provide real time fault detection and isolation and hence reduce aircraft down time due to avionics failures. Changes in mission requirements will be reflected in the need to modify or add software modules throughout the system's life cycle. This

necessitates design and control of hardware and software interfaces in order to keep the time required to validate the change to a minimum and speed retrofit of the modification in the operational units.

INTRODUCTION

The requirements for digital avionics have been based on functional mission performance, reliability, and safety factors. In addition, operational factors including natural and combat environments impact the ability of a system configuration to meet the reliability and safety requirements. Advanced digital avionics must be developed to satisfy not only the foregoing requirements, but also be sustainable in order to maximize aircraft availability. Sustainable avionics can be, and must be, developed using fault tolerant design techniques for both hardware and software. Modern software engineering practices should be used throughout the life cycle.

These advanced digital avionics must be validated throughout the life cycle to assure the system requirements are satisfied. Validation must begin in the development phase. It is impossible to adequately validate the system if the start of validation is delayed until the development is completed. Cost effective validation must be conducted in parallel with the system design and development. The products (tools and documentation) required for validation can strongly influence the design and development process. These same tools, and validation methodology, are used throughout the system's operational period to validate updates to the digital system hardware and software.

These validation needs strongly interact with the system design and hence directly influence the avionics sustainability and life cycle costs.

BACKGROUND

Digital avionics validation has been the subject of research sponsored by the FAA, NASA, and USAF. Battelle's Columbus Laboratories have developed the "Handbook of Validation Processes for Digital Integrated Flight Control and Avionics" (Ref. 1) for the FAA Technical Center. The purpose of this handbook is to identify techniques, methodologies, tools, and procedures in a systems context that may be applicable to aspects of the validation and certification of digital systems at specific times in the development and certification portion of the system life cycle. The application of these techniques in the development of discrete units and/or systems will result in a completion of a product or system which is verifiable and can be validated in the context of the existing regulations/orders for the government

regulatory agencies. The handbook uses a systems engineering approach to the integration and testing of software and hardware during the design, development, and implementation phases. The handbook also recognizes and provides for the evaluation of the pilot's work load and utilization of the new control/display technologies, especially when crew recognition and intervention may be necessary to cope with/recover from the effects of faults or failures in the digital systems, or from crew-introduced errors in periods of high work load due to some inadvertent procedure or entry of incorrect or erroneous data.

The recommended validation methodology contained in the handbook is equally applicable to military avionics as well as civil avionics. The civil safety requirements for flight critical functions are more stringent than those in MIL-F-9490D for the same functions.

These methodologies require standardization of hardware and software interfaces and strict configuration control in order to assure the integrity of the validation process.

VALIDATION METHODOLOGY

The methodology for validation and maintenance of advanced digital integrated flight controls and avionics has been extracted from Ref. 1 and is synopsized in the following paragraphs. Greater detail is contained in the handbook and it is recommended that those interested in the detail request a copy of the handbook from the FAA Technical Center.

The roles and responsibilities of the various agents (customers, developers/manufacturers, and independent test organization (ITO)) are described in Ref. 1. Specific analysis and design aids (models and tools) are discussed in detail in Ref. 1 and are synopsized below.

Figure 1 depicts many of the major activities that must take place during the system definition, design, full scale development, operational test and evaluation, air worthiness/certification, production/deployment, and operation/maintenance phases of the system life cycle. The figure depicts the major activities and their time relationship and provide supplemental information for each activity. In addition, major decision points are indicated by the diamond shaped logical decision box. The following discussion synopsizes for each of the major phases of the system life cycle those activities depicted in Figure 1.

CONCEPT FORMULATION PHASE

Two primary activities occur during this phase. The general plan of approach will be developed and a system analysis performed to develop system requirements fully. The report presents

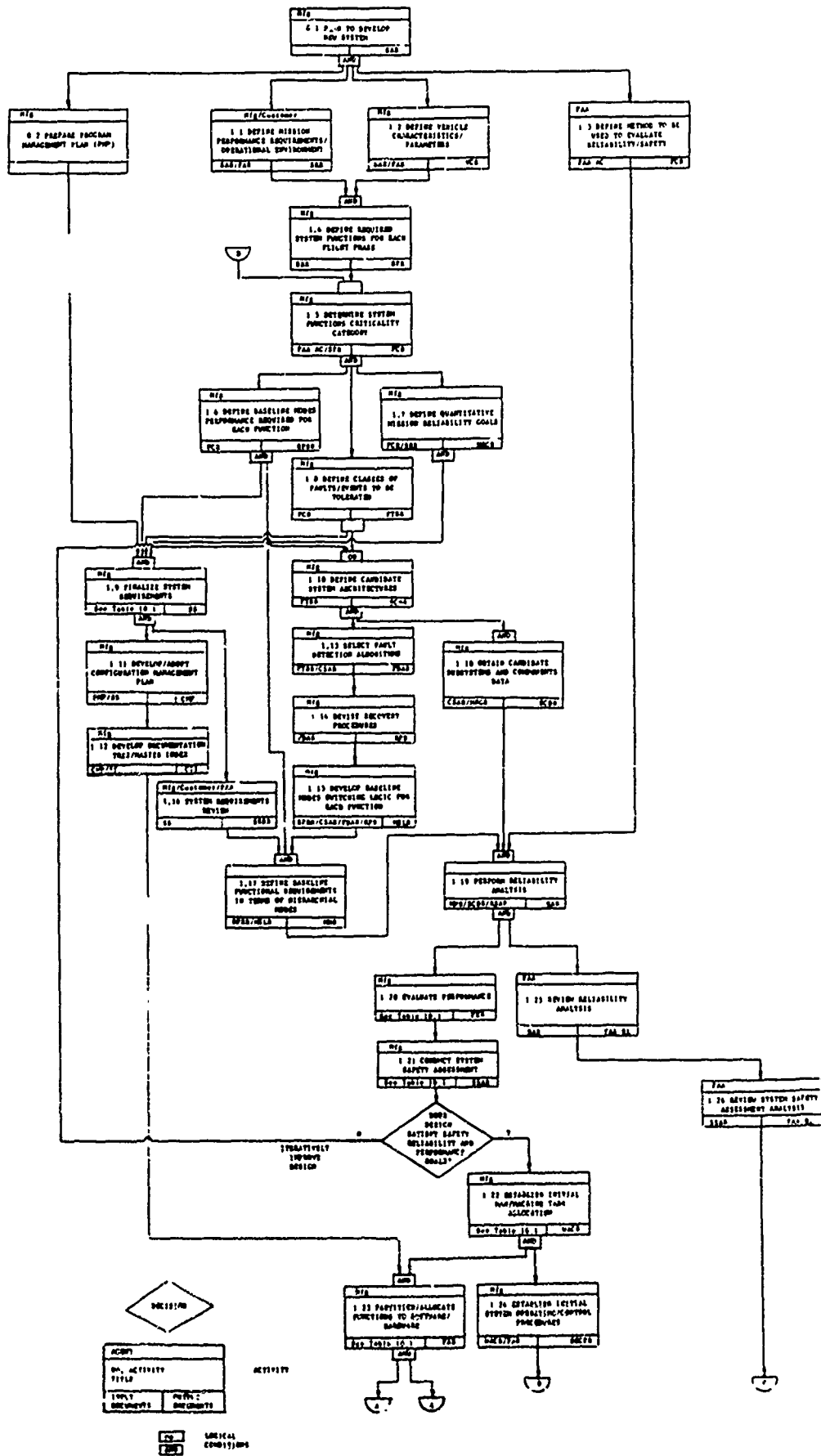


FIGURE 1. DIGITAL SYSTEMS VALIDATION ACTIVITIES SEQUENCE

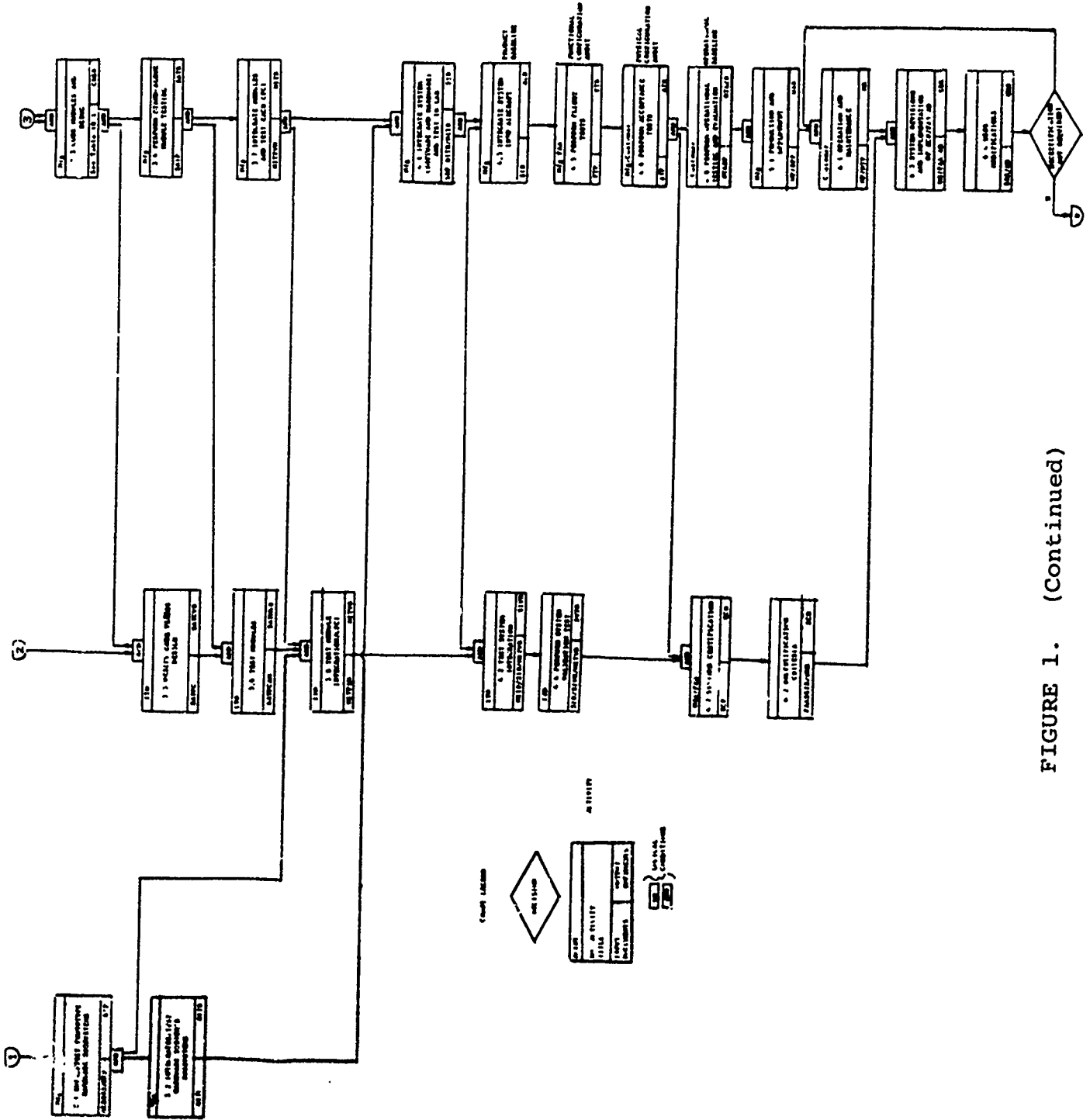


FIGURE 1. (Continued)

the results of a comprehensive functional analysis of the system's elements (personnel, equipment, facilities, and support).

A program management plan will be prepared to describe the procedures which will be used to control the work during the subsequent phases of the system life cycle. It includes definition of both technical and financial management tools and describes the reporting procedures in detail. The program-management methods such as project-control tools, required back-up staff, and line-management structure are defined.

SYSTEM DEFINITION PHASE

1.1 Define Mission Performance Requirements/Operational Environment. The work that takes place in the problem definition is critical since the performance requirements defined form the basis for the subsequent design and evaluation of candidate systems against these requirements. The care taken in this step will impact all subsequent activities.

Basic mission related factors and operational environment considerations must be included. The use of formal systems requirements engineering tools such as the Design Analysis System (DAS) (Ref. 2), Systematic Activity Modeling Method (SAMM) (Ref. 3), and others described in Ref. 4 may be of advantage in establishing the requirements of a large system comprising many subsystems. The output of this task is a compilation of the mission performance requirements and operational environment description.

1.2 Define Vehicle Characteristics/Parameters. Vehicle characteristics including mass properties as a function of time, aerodynamic data, propulsion system data, and data on all systems the avionics and flight control interface with including aircraft electrical system, flight control surface actuators, and air data sensors require careful definition as we proceed from generic avionics to fault tolerant avionics. In order to design sustainable avionics, the sensors must be located to provide a high probability of survivability in a combat environment as well as provide the information needed to make use of software observers, e.g., Luenberger. Software observers are key to the fault detection and isolation in fault tolerant systems.

1.3 Define Method to be Used to Evaluate Reliability/Safety. This activity is normally the responsibility of the procuring or regulatory agency. Recent evaluations of existing computer implemented reliability analysis models (Ref. 5) concluded that many of these models had significant limitations when applied to fault tolerant systems. The FAA recently directed the use of fault trees (Ref. 6). Whatever method is used, it should have been validated prior to application to the systems under consideration.

1.4 Define Required System Functions for Each Flight Phase.

This activity requires the application of the methodology of functional decomposition of the overall mission requirements into specific system functions required for each flight phase. If a formal methodology is used, it should provide the capability to "support a flexible approach and functional decomposition which may be successfully embellished to accommodate the functional, performance, and operational requirements. In other words, this specification and requirements may best be envisaged as the accumulation of a data base which will be progressively updated with the information regarding functional, performance, and operational requirements" (Ref. 7).

1.5 Determine System Functions Criticality Category.

An analysis must be performed of each system function to determine its criticality. All the assumptions, sources of reliability data, failure rates, system function type category, etc. should be precisely documented in a report providing the criticality category of all system functions.

1.6 Define Baseline Modes Performance Required for Each Function.

In some cases, the performance requirements are specified by the procuring or regulatory agencies. In other cases, the developer of the system will have to establish the accuracy, data channel capacity, computer throughput, etc. for the baseline mode for each function for each flight phase. Since the baseline mode for all functions should provide the highest overall performance of the system, it is important that all functions which simultaneously require the use of certain resources be included in the analysis. At this stage in the system definition, it is probable that subsequent design changes may necessitate changing some parameters. Therefore, provision must be made for growth in terms of the demand on resources and data throughput.

1.7 Define Quantitative Mission Reliability Goals.

MIL-F-9490D specifies mission accomplishment reliability and quantitative flight safety goals. The system developer must translate these requirements into quantitative values which can then be allocated to the components of the system. This reliability apportionment is often done using the "similar familiar system's reliability apportionment approach" or the "factors of influence method" (Ref. 8). As stated in this reference, "both the familiar system factors and the influence methods have their weaknesses when they are used individually. However, combining the two methods produces better results because data are used from similar subsystems as well as when new subsystems are designed under different factors of influence."

1.8 Define Classes of Faults/Events to be Tolerated.

A fault tolerant system designed to tolerate all known fault types, however small the probability that a fault of a specific type might occur, could be prohibitively expensive. The developer must, based upon the system function criticality category, define the classes of faults and events which the system is to tolerate.

This must be defined in sufficient detail to permit selection of fault detection algorithms and development of recovery strategies in subsequent activities.

1.9 Finalize System Requirements. The system specification documents all system level requirements for each function, its criticality, performance, and the apportioned reliability goals.

1.10 Define Candidate System Architectures. There are many candidate digital data bus architectures ranging from the single level, to many different hierarchical networks integrating local buses with dedicated functions (Ref. 9) to complex mesh networks (Refs. 10-11). Unless the procuring agency specifies a specific architecture, the developer must find one or more candidate system architectures which satisfy the system requirements, and specifically the reliability and safety requirements.

1.11 Develop/Adopt Configuration Management Plan. The manufacturer must either develop or adopt a configuration management plan which will be the basis for the management of the hardware and software configuration throughout the life cycle of the system. This activity is critical in maintaining the integrity of the system. Deviations from standard interfaces for hardware and software have serious impacts on system sustainability.

1.12 Develop Documentation Tree/Configuration Item Index. The documentation tree should be developed and a method adopted for identifying all documents and the current version of the document.

1.13 Select Fault Detection Algorithms. The fault detection algorithm selection should result in the selection of algorithms that will be implemented to assure correct operation of each processing unit, valid transmission of data between digital systems, data validity prior to use in subsequent computation, and system response to commanded outputs. Specific algorithms selected should be documented in a report for subsequent use in a hardware and software specifications.

1.14 Revise Recovery Procedures. Basic recovery procedures possible after isolating the fault are dependent upon the type of fault, the function and fault's impacts. If the fault involves incorrect operation of the processing unit's memory, self-checking circuits and error-correcting codes (dependent upon the codes selected) have the ability to detect and correct single bit and detect double bit errors. These error-correcting codes in effect assure that the computer produces the correct outputs, in spite of the fault.

In the case of data transmission errors, recovery techniques may involve a retransmission of the message, with switch to a redundant bus if communication between the digital subsystems is not established within the design number of retries or retransmissions. At a system level, recovery techniques may involve

reloading and restarting of programs as well as complete reconfiguration of the system. The reconfiguration may include the assignment of tasks to processors with the specific task assigned dependent upon the mission phases remaining. Note, that this approach may conflict with the absolute code address for modules discussed in Ref. 12.

1.15 Develop Baseline Mode Switching Logic for Each Function. The baseline mode switching logic implements the switching between modes as a function of performance measures not meeting those required for the function, or failures of sensors, computers, or data transmission paths.

1.16 System Requirement Review. This is the first formal review at which the agent responsible for validation normally participates. Often this does not occur and the system requirements review is limited to the developer and his customer. This review is generally conducted as a "walk through" in which a careful review of the system's specification is made against the system requirements with the objective of identifying any deficiencies. These deficiencies must be corrected prior to proceeding with further definition and design of the system. At the completion of this task, the functional baseline of the system has been established.

1.17 Define Baseline Functional Requirements in Terms of Hierarchical Modes. The previously defined baseline modes and switching logic for each function must be structured in a hierarchy of modes. Normally the priority of modes is ordered on the basis that the mode with the best performance is preferred. Depending upon the design approach taken, the system may or may not operate in a mode with the best performance. Generally, the switching between modes for an automatic subfunction is performed by the determination of the availability of the subsystems required for each mode. If all subsystems or all modes are available, the first mode is used. When a failure is detected, the system functions will be searched to determine the effect of the failure mode. The system will then revert to the backup mode with the next best performance. The hierarchy of modes must permit crew selection of any mode with a reversion to the automatic mode selection when the crew so elects.

1.18 Obtain Candidate Subsystems and Components Data. The data must be collected for use in subsequent reliability performance analysis as well as design of the system. These data should be implemented in a data base which is controlled.

1.19 Perform Reliability Analysis. An analysis of the theoretical reliability of the candidate system architectures for the various combination of candidate subsystems and components should be performed using the previously agreed upon methodology. The results of the analysis should be compared with the previously documented mission reliability goals.

1.20 Evaluate Performance. Performance evaluation makes use of simulation and modeling. For the digital system, a model is built using a simulation language that describes the basic configuration of the system. Performance characteristics can then be determined from the characteristics of the allocated components. The model used in this simulation should reflect not only the system and its components but also the environment the system will operate in.

The simulation model may be driven by either generating input data (probabilistic or Monte' Carlo simulation) or by feeding some representative input data (deterministic or trace-driven simulation) and then simulating the actual behavior of the system.

Performance evaluation seeks to determine performance characteristics due to algorithmic design, system allocation and configuration, and structure and interfaces. No matter whether the simulation is a self-driven simulation using a statistical description of inputs or a trace-driven deterministic simulation, the results must be properly analyzed to evaluate the performance for use in subsequent design refinements.

In addition to the performance evaluation of the digital system, performance evaluation using either analytical models or simulation should be done considering the complete closed-loop response and accuracy of the system including the sensor errors, data transmission delay times, computation delay times, and actuator characteristics to arrive at performance estimates of the closed-loop system. If a hierarchical structure is used in the development of the evaluation, it is possible that combinations of analytical and simulation models may be used.

1.21 Conduct System Safety Assessment. A system safety assessment should be conducted using the previously agreed upon methodology for the candidates systems.

1.22 Establish Initial Man/Machine Task Allocation. This activity assumes that the basic concept meets the safety, reliability, and performance goals. The man/machine task allocation is based upon workload and other factors discussed in Ref. 1. Prior to the application of some of the more sophisticated task allocation tools, functional sequence diagrams, operational sequence diagrams, and action sequence diagrams (Ref. 13) should have been developed. Reference 13 states "the allocation of functions is a trial and error type of process that proceeds according to the scale of the analyst". The goal of any system, in its ultimate use by and for humans, is to present the information required for human judgement in a manner that best aids decisions. Therefore, the goal is to allocate to the data processing system those functions which a human can not (because of time constraint) or will not (because of tedium) do for himself, and allocate to the human those functions that involve value judgements".

1.23 Partition/Allocation Functions of Software/Hardware.

"The concept of 'software first' is reaching acceptance. Thus, the software is designed through at least an intermediate design stage prior to determining hardware requirements, and software requirements are driving hardware selection rather than vice versa. Therefore, in the allocation to the system, the cost of developing software makes it attractive to allocate certain functions to hardware, particularly those functions that are simple, independent of other software functions, and can be expected to remain the same for a long span of the system life cycle. The functions are allocated to software where changes are anticipated over the life of the system. Also, and more important we allocate the software to those functions that integrate all the elements of the system into a smooth operation." (Ref. 13)

1.24 Establish Initial System Operating/Control Procedures.

System operating and control procedures involve not only those procedures associated with the operation and use of the digital avionics/flight control systems themselves, but also the procedures associated with the operation of the vehicle and its subsystems. This involves definition of the minimum equipment list required for dispatch, maintenance procedures in case of a detected fault, and the more complicated operating procedures necessary when operating in constricted airspace and attempting to utilize time-of-arrival algorithms. Various trade-offs between system operating procedures are possible just as there are trade-offs in the design.

1.25 Review Reliability Analysis/1.26 Review System Safety Assessment Analysis. These activities are conducted by the customer and consist of a thorough review of the analyses provided by the developer. The results of these reviews are provided to the developer so that he may correct any discrepancies or provide any additional information required.

1.27 Prepare System Hardware Designs/Interface Specifications. This activity involves documenting the system hardware design specification and the interface specification describing all hardware interfaces between subsystems.

1.28 Complete System Software Requirements Definition. The inputs to this task are the preceding reports which define all the software functions, and the fault tolerant algorithms, recovery procedures, switching logic functions allocated to software, and initial system operating/control procedures. Software architecture will be defined and the software functions to be performed by each processor defined in terms of their control structure, data structure, data flow control, and application structures. The system software development specification will describe the overall system software requirements and be the primary reference document for all system software. Software located in the individual processors must be traceable back to the systems software development specification.

1.29 Validate System Design Against Requirements. The system design review evaluates all work leading up to this point in the system development. This review normally occurs at the end of the validation phase or early in the full-scale development phase of the system life cycle. At the completion of this review, the allocated configuration baseline is complete.

SYSTEM DESIGN PHASE

2.1 Select/Design Avionics/Flight Control Hardware. The system design may make use of existing components as well as require the complete design and development of new components and subsystems. Analysis and experimentation may be required to determine which existing component may be candidates and to select from these components. In addition, the design for new components must be completed. Mathematical models and simulation are often used in evaluating the performance and reliability characteristics of existing equipment as well as in the design in new equipment. The primary difference is the depth and detail required since the design of a new item entails working with piece part component and individual integrated circuit characteristics as well as the design of components such as large scale integrated circuits or hybrid circuits that incorporate the functions of many individual piece part components.

In the design of new subsystems, it is also necessary to perform the electronic packaging design include mechanical, thermal, and other environment modifying design techniques such as sealing and pressurizing the line replaceable unit. To complete the hardware design process, specifications and drawings must be updated to reflect the specific characteristics of each component of the selected system configuration.

2.2 Design System Integration/Support Facility. The system integration/support facility is a tool to be used for development and integration of the avionics. This activity involves establishing the requirements and characteristics of the facility, and development of a program plan for the development of this facility. The facility processors shall host the support software required for development, test, and integration of the object code for each processor used in the avionics. Support facility host processors will host the compilers, assemblers, linkers, editors, and loaders for the flight processors. Many support facilities include one or more processors capable of emulating the micro-code of the actual flight processors.

In addition to the processors, the support facility shall include the network interconnecting the support facility processors, the data collection instrumentation, and other associated electronic test instrumentation. The support facility generally contains the test control center for controlling the use of the simulators in the support facility.

2.3 Prepare Software Development Plan. The software development tasks and schedule, methodology, configuration item and deliverables, configuration management plan, and documentation standards. The plan should make allowances for correction of errors found through testing and retesting to verify that the errors have been corrected and no new errors introduced.

2.4 Define Computer Program Configuration Items Requirements. A detailed set of computer program configuration items (CPCI) specifications, which are a statement of the development requirements for each CPCI, whether they are routines, programs, groups of programs, or the entire software subsystem (if it is small), shall be developed. Individual CPCI specifications shall be traceable to the software development plan, configuration item index, and the system software development specification.

2.5 Develop System Software Interface Specification. The system software interface specification describes in detail the requirements for all data transmitted between digital subsystems. The format of each word and, in multiple word messages, the format of each message is totally specified. If the data transmissions are currently on a synchronous basis, the transmission rate is specified. If a command response protocol is used in which addresses and subaddresses are used for communication rather than a broadcast protocol, the addresses and subaddresses in each message or word is given. This specification serves as a basic software interface control document and should be under configuration control. Any data transmission between subsystems other than those prescribed in the software interface specifications should be invalid.

2.6 Develop/Modify System Integration/Support Facility. If no system integration/support facility exists, the manufacturer must develop a facility which meets the requirements previously formulated in the design of the system integration/support facility. Should the manufacturer presently have a system integration/support facility, the activity may merely involve making minor hardware modifications or modifying software data acquisition programs to require that test data to be obtained during the system test and integration. At the conclusion of this activity, the system integration support facility should be complete including all hardware and software.

2.7 Define Test Requirements. The system should have been designed from the beginning to be testable. The test requirements document describes the test approach and addresses (1) the testing philosophy followed, (2) responsibility for the various levels of testing, (3) test performance measures and standards, (4) method to be followed in handling software change proposals emanating from each test activity, and (5) test report requirements. This document is used for the detailed test planning and development of test procedures for each test plan.

2.8 Commence Software Real-Time Operating System, Programs, and Modules Design. Using a structured design

procedure (Refs. 14 and 15), each module is designed using the allowed basic constructs and the algorithms defined in the CPCI development specifications. Standard coordinate systems, definitions, symbols, and mnemonics must be used in the development of the software. Software interface standards between sensors/equipment modules and core software modules which may be used across more than a single aircraft type must be established. Each input/output variables mnemonic, units, range, and resolution must be established. This permits the designer the flexibility of combining standard data words into the MIL-STD-1553B bus messages required to transfer data between software modules in different processors.

The system hardware architecture selected by a designer determines primarily the address/subaddress of the data to be transmitted/received between software modules, whether core (standard) or sensor/equipment modules. Repartitioning the software in a multiple processor system can be done with relative ease if the standard data words to be transmitted between core modules are adhered to and the core element integrity is preserved. Attempts to further partition core elements (a single processing function involving solution of an equation set yielding a single output or vector components) into different processors will reduce the processing efficiency and efficacy of the core software modules (Ref. 16).

2.9 Hardware Preliminary Design Review. Individual hardware design description documents and development specifications are reviewed to evaluate hardware trade-offs, functional interfaces, errors due to lack of understanding of the critical design areas, and the interfaces of the system integration support facility with each of the hardware items.

2.10 Verify Software Requirements Against System Requirements. The software preliminary design review is held after authentication of the CPCI development specification and the accomplishment of preliminary design efforts but prior to the state of the detailed design. Designers normally "walk" the reviewers through the design in a step-by-step fashion that simulates the function under investigation. The material is reviewed in enough detail so that the concerns expressed at the beginning are either explained away or identified as action items.

Activities 2.11-2.14. These activities consist of specification reviews, updates to specifications, breadboard and evaluation of circuits, and the hardware critical design review. Further details are contained in Ref. 1.

2.15 Develop System Validation Test Plan and Procedures. The validation test plan encompasses verification and should describe the techniques or methods to be used in the validation of the system. Reference 1 describes various methods including those associated with design validation, hardware testing, software testing, and system level tests. The validation test plan

should specifically identify each of the selected test concepts which will be used for the foregoing. It contains test objectives, test description, description of the test environment, including required hardware and software, delineation of the requirements being validated, and an evaluation plan. The evaluation plan consists of the acceptance criteria and a description of the techniques to be used in analyzing the test data in order to determine compliance with the acceptance criteria. Observations of the test itself and the evaluation of the test output data constitute the basis on which it is determined whether the test objectives have been met, pertinent requirements validated, and the acceptance criteria satisfied.

2.16 Develop Test Plans. This activity is developing the test plans for each of the test levels including stand alone hardware testing, stand alone testing of software modules, software integration, system integration, and flight tests. Each test plan traces a testing sequence from unit level test to final acceptance tests and identifies each individual test. Test procedures, keyed to the test plan, provide step-by-step instructions for the execution of the test and specify precisely what outputs are to be expected. Test support software for the hardware test bed to be used should be identified as well as all testing inputs. Test procedures shall be sufficiently detailed so they can be used in the complete integration, replication, and validation of the system software. The test procedures must also provide all information required for the integration of the system and the flight test of the system. These test plans and procedures should be furnished to the procuring agency for review prior to the critical design review.

2.17 Perform Detailed Software Design. The final software design is often done using a formal design methodology such as structured design or other methods. During the final design effort, a design walk-through should be used by the developers to verify the flow and logical structure of this system while design inspection should be performed by the test team.

2.18 Software Critical Design Review. "The critical design review (CDR) is a formal technical review of the CPCI detailed design conducted prior to the start of coding. The CDR is intended to ensure that the detailed design solutions, as reflected in the draft of the CPCI product specifications, satisfy performance requirements established by the CPCI Development Specification. CDR is also accomplished for the purpose of establishing integrity of the computer program design at the level of flow charts or computer program logical design prior to coding and testing. The principal items reviewed are the complete draft of the CPCI product specification and drafts of test plans/procedures. All changes to the CPCI Development Specification and available test documentation are examined to determine compatibility with the test requirements of a development specification" (Ref. 17).

SYSTEM FULL SCALE DEVELOPMENT

3.1 Build/Test Prototype Hardware Subsystems. Component screening, acceptance testing, and the environmental qualification testing are conducted in the hardware subsystems. In addition, failure modes and effects tests should be conducted at the individual subsystem level to verify the system redundancy design for the classes of faults the system is to tolerate. Any discrepancies identified in the tests should be analyzed and modifications required to make the system operate properly identified and submitted to the Change Control Board.

3.2 Integrate/Test Hardware System's Subsystems. A sequence of integration tests should be performed to integrate each of the hardware subsystem's. A simulator may be used in this testing to provide the test driver signals for items not yet integrated.

3.3 Code Software Modules and Debug/3.4 Perform Stand-Alone Module Testing. These activities involve coding and debugging software followed by stand-alone testing. The stand-alone test may use the techniques of static analysis, dynamic testing with or without instrumentation probes, symbolic execution, or proofs of correction. Code execution testing may be done on a host computer which simulates or emulates the target computer or the actual execution may be done on the target machine. Whichever module testing approach is taken, one basic criteria for the set of test cases is to ensure that they cause every instruction in the module to be executed at least once. All logical paths should also be traversed. The testing should be done in the sequence specified by the test plan and procedure.

3.5 Verify Code Versus Design/3.6 Test Modules. These activities are conducted by the independent test organization. A walk-through or inspection may be used. In addition, a static analyzer may be used. An independent test organization is likely to use a dynamic analyzer and execute the code for each of the modules. The data collected by the instrumentation probes in the dynamic test mode will be analyzed and a test report prepared noting any anomalies.

3.7 Integrate Modules and Test Each CPCI. Integration testing is primarily functional with the main emphasis on the interaction between the software components and the interface. Testing also takes place in a laboratory containing the target computers and enough equipment to simulate the application with considerable fidelity. As each test is conducted by the developer, a test report will be generated. After all testing is completed, the final report is generally prepared which includes all errors detected and status of their correction.

3.8 Test Module Integration/CPCI. The independent test organization conducts this test for the purpose of verifying interfaces, computational accuracies, timing, and sizing. While some of the tests may be run using an emulation of the target processor and the instrumentation probe, final module integration testing for each CPCI should be tested in the actual computer and hardware environment. These tests will be run under "live" conditions using test drivers in the avionics integration support facility.

SYSTEM INTEGRATION, TEST ACTIVITY

These activities consist of integrating the system (software and hardware) in the laboratory and making use of simulation facilities. The independent test organization will normally conduct failure modes and effects tests. After completing the laboratory system integration testing, the system will be integrated into the aircraft which may or may not necessarily be the only aircraft the system will ultimately be used in. In this case, the aircraft interface must be specifically noted; special instrumentation may be required if it is expected that the interface in another aircraft could be greatly different. At the completion of this test, a product baseline will have been fully defined which then becomes the baseline used by the configuration management organization.

The system validation tests conducted by the independent test organization are designed to demonstrate that the system will correctly operate in the environment it's designed to operate in and tolerate system transients and other faults the system was designed to tolerate. These independent validation tests may occur in the same time frame as the flight test performed by the manufacturer.

At the completion of the flight test, a functional configuration audit may be performed on the software. This functional configuration audit verifies that the CPCI's actual performance complies with the requirements of the development specification. Requirements of the development specification not validated by the CPCI test are identified and the solution for subsequent validations is proffered.

The acceptance tests are either conducted by the developing organization while being witnessed by the customer or performed by the program's customer or end user. At the completion of the acceptance testing, the Physical Configuration Audit is conducted.

Prior to or during the flight test, actual data collected shall be compared with that used in early analyses such as the reliability and safety assessment analyses to determine if there is a great discrepancy in the data or a need to redo the analyses. Results of simulation tests are being used as a substitute for the many costly hours of flight tests where simulation can be

shown to yield valid results. In many cases, simulators are being used for conducting hazardous or high risk tests instead of actual flight tests (Ref. 18).

Operational tests and evaluation tests are conducted to determine the operational effectiveness and suitability of the system. The operational effect portions of the test are concerned with the capability of the system to perform its intended function in the operational environment while the operational suitability is concerned with the degree the system supports a mission and is maintainable (Ref. 19). These tests are normally conducted by the end user.

PRODUCTION AND DEPLOYMENT

These activities consist of production of the quantities of the system required by the user, the acceptance testing of each system by the user, and the introduction and operation of each of the new systems as they are delivered from the manufacturer.

OPERATION AND MAINTENANCE

The user of the system (e.g., TAC and AFLC) must continue the configuration management activities. "Changes to system functional capability required by the user or discovery of design errors during service will necessitate post-certification in software changes. Such changes can lead to "secondary errors" in the software, i.e., errors that were not present or whose affects were not detected, when the system was first accepted. Thus careful consideration must be given to verification/validation of the changes." (Ref. 20) The users should establish a formal data collection data base system for the digital avionics system. This information will be of great use in the maintenance of the digital system's hardware and software.

CONCLUSIONS

Fault tolerant digital avionics must be designed to permit validation of the system. The methodology synopsized in this paper and presented in detail in Ref. 1 is an approach to the development of digital avionics based upon the reality of updates and modifications to these systems necessitated by mission requirements changing throughout the life cycle. The tools and documentation required for validation have direct application to the design and development cycle. The observance of a structured design process based upon the recognition of the need for validation will result in sustainable avionics.

REFERENCES

1. Hitt, Ellis F., Webb, Jeff J., Lucius, Charles E., Bridgman, Michael S., Eldredge, Donald, and Sulzer, Richard, "Validation Processes for Digital Integrated Flight Control and Avionics Systems", Vol. I, FAA-CT-82-115, Battelle Columbus Laboratories, October 1982.
2. Willis, R. R., "DAS An Automated System to Support Design Analysis", pp. 109-115, Proceedings of 3rd International Conference on Software Engineering, May 10-12, 1978, IEEE Catalog No. 78CH1317-7C.
3. Stephens, Sharon A. and Tripp, Leonard L., "Requirements Expression and Verification Aid", Ibid, pp. 101-104.
4. Schindler, Max, "Today's Software Tools Point to Tomorrow's Tool Systems", Electronic Design, Vol. 29, No. 15, July 23, 1981, pp. 73-110.
5. Ness, W. G., McCrary, W. C., Bridgman, M. S., Hitt, E. F., and Kenney, S. M., "Automated Reliability and Failure Effects Methods for Digital Flight Control and Avionic Systems; Volume I: Evaluation, Volume II; Methods Summary", NASA CR-166148, Lockheed-Georgia Company and Battelle Columbus Laboratories, March 1981.
6. "Airplane System Design Analysis", Advisory Circular (AC) 25.1309-XX, Department of Transportation, Federal Aviation Administration, Federal Register, Vol. 46, Issue 214, November 5, 1981, p. 54958.
7. Smoliar, Stephen W., "Operational Requirements Accommodation in Distributed System Design", IEEE Transactions on Software Engineering, Vol. SE-7, No. 6, November 1981, pp. 531-557.
8. Dhillon, B. S. and Singh, Chanan, Engineering Reliability, New Techniques and Applications, Wiley-Interscience Publication, John Wiley & Sons, New York, 1981, pp. 43-45.
9. "MIL-STD-1553 Multiplex Applications Handbook", Air Force Systems Command, Aeronautical Systems Division, ENASD; Boeing; SCI Systems, Inc., Revised February 1, 1982.
10. McCuen, James W., "Higher Order Information Transfer. Systems are Coming", AIAA 81-2317, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
11. Brock, Larry D., Hopkins, Albert L., Jr., and Spencer, J. Larry, "Onboard Communications for Active-Control Transport Aircraft", AIAA 81-2321, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.

12. "Software Considerations in Airborne Systems and Equipment Certification", Document No. RTCA/DO-178, Radio Technical Commission for Aeronautics, November, 1981.
13. Jensen, Randall W. and Tonies, Charles C., Software Engineering, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979, pp. 125-132.
14. Stevens, W. P., Myers, G. J., and Constantine, L. L., Structured Design, pp 114-122.
15. Dolbey, S. C. and Cary D. R., "Management of Software Design--A Structured Approach", Proceedings of the IEEE 1981 National Aerospace and Electronics Conference, pp. 172-178.
16. Hitt, Ellis F. and Broderson Robert L., "Integrated Control Core Software Concept Study", Vol. 1-5, AFWAL-TR-81-3141, Battelle Columbus Laboratories, December 1981.
17. Jensen, Randall W. and Tonies, Charles C., Software Engineering, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979, p. 399.
18. Archibald, Don M., "The Role of Simulation Methods in the Aircraft Certification Process", Report No. FAA-RD-77-17, Lockheed-California Company, March 1977.
19. Murch, Walter G., "Operational Test and Evaluation of Software", Proceedings of the IEEE 1981 National Aerospace and Electronics Conference, pp. 1390-1398.
20. "Software Considerations in Airborne Systems and Equipment Certification", Document No. RTCA/DO-178, Radio Technical Commission for Aeronautics, November, 1981, pp. 34.



HH-60D ADVANCED AVIONICS ARCHITECTURE

Ira Glickstein
IBM
Owego, NY

BIOGRAPHY

Ira S Glickstein is a Senior Engineer at IBM's Federal Systems Division, Owego, New York. He served as lead systems engineer on the HH-60D proposal, with responsibilities in the system architecture and controls/displays areas. He is currently in the HH-60D systems engineering department.

Ira has been involved in new business and initial development engineering for several projects at IBM, including: Coherent Emitter Location Testbed (CELT), Army Command and Control Master Plan, Joint Tactical Information Distribution System (JTIDS), Adaptable Surface Interface Terminal (ABIT), A-4M Angle Rate Bombing System, Light Airborne Multipurpose System (LAMPS), AC-130E Gunship, and A-7D/E Navigation and Weapon Delivery System. He received outstanding contribution awards for his work on a computer memory correction method and for systems engineering efforts.

Prior to coming to IBM, he was employed by Lockheed Electronics and Norden. He holds an Electrical Engineering Degree from City College of New York and a Professional Engineering License (New York).

ABSTRACT

The HH-60D avionics system design makes extensive use of USAF/DoD Interface and Processing Standards:

Standard Interfaces.

- All data interfaces use a dual-redundant MIL-STD-1553B data bus, where cost and safety considerations permit. This provides operational reliability and growth flexibility.

- Signal conversion for equipments that are not data bus compatible is performed by four separately located RTUs that serve the cockpit/nose and transition areas. This permits use of unmodified inventory units, and reduces risk, schedule, and life-cycle costs.
- The control and display subsystem is compatible with either 525 or 875 line TV (EIA RS-343A/(RS-170) and can handle either 1:1 or 4:3 aspect ratios. This allows use of current FLIR, MMR, and map technology, and infusion of technology improvements.

Standard Processing Hardware and Software.

- All mission processing is performed in dual-redundant MIL-STD-1750 Mission Computers, using the standard USAF high-order language, JOVIAL J73, and structured software development disciplines. This controls the software development process and provides inherent growth flexibility through technology infusion and transferable software.
- Existing distributed processors, with proven software and firmware, are used for peripheral tasks, where cost effective. The embedded processors and software in the INS, MMR, and DEUs, are examples of cost-reduction by use of software and designs already paid for by other military programs.
- The proposed HH-60D architecture is fully compatible with distributed processor avionics now in development for use on multiple military air vehicles.

WESTINGHOUSE USES U.S. AIR FORCE-DEVELOPED STANDARDS

Carl S. Shyman, General Manager

Westinghouse Defense Electronics Center
Avionics Division
Baltimore, Maryland 21203

(301) 765-4766

Abstract

Westinghouse has applied digital standards advantageously for the U.S. Air Force on its latest weapon systems. At present Westinghouse is applying Mil-Std-1750A (ISA), Mil-Std-1589B (Jovial 73 HOL), and Mil-Std-1553B (multiplex busing) to three major programs: B-1B Offensive Radar System, Improved AN/APG-66 Radar for the F-16, and AFTI F-16 Electro-Optical Sensor/Tracker.

Westinghouse has gone one step further than the digital standards. With U.S. Air Force encouragement Westinghouse has a program for maximum radar commonality among the B-1B ORS, F-16C, and the U.S. Army Sgt. York DIVAD Gun System. This paper will cover Westinghouse's approach toward managing the application of the military standards across multiple programs with different prime contractors and services. Additionally, the method by which configuration control of standard module hardware (i.e., rational standardization) maintained at Westinghouse will be discussed.

Paper

Westinghouse developed a modular radar series a decade ago that was called the WX series of radars. The basic architecture of the radars is shown in figure 1. The main subunits were the antenna drive, transmitter, stalo, receiver, programmable signal processor, antenna pointing sensors, backup displays and controllers, computer complex, and radar interface. All of these subunits were interconnected into a radar system by a high speed multiplex bus called the digibus which was modulo 8-bit parallel protocol at 1 MHz speed. Most of the radar family used 16-MHz bit rates on the digibus.

The radar configurations from low performance to high performance were achieved by varying the types of subunits and changing the software and memory capacity. All of the radars used common support software and support hardware. The interface of these subunits all contained the same terminal type connecting to the digibus. The interface to the onboard

Modular Radar Architecture

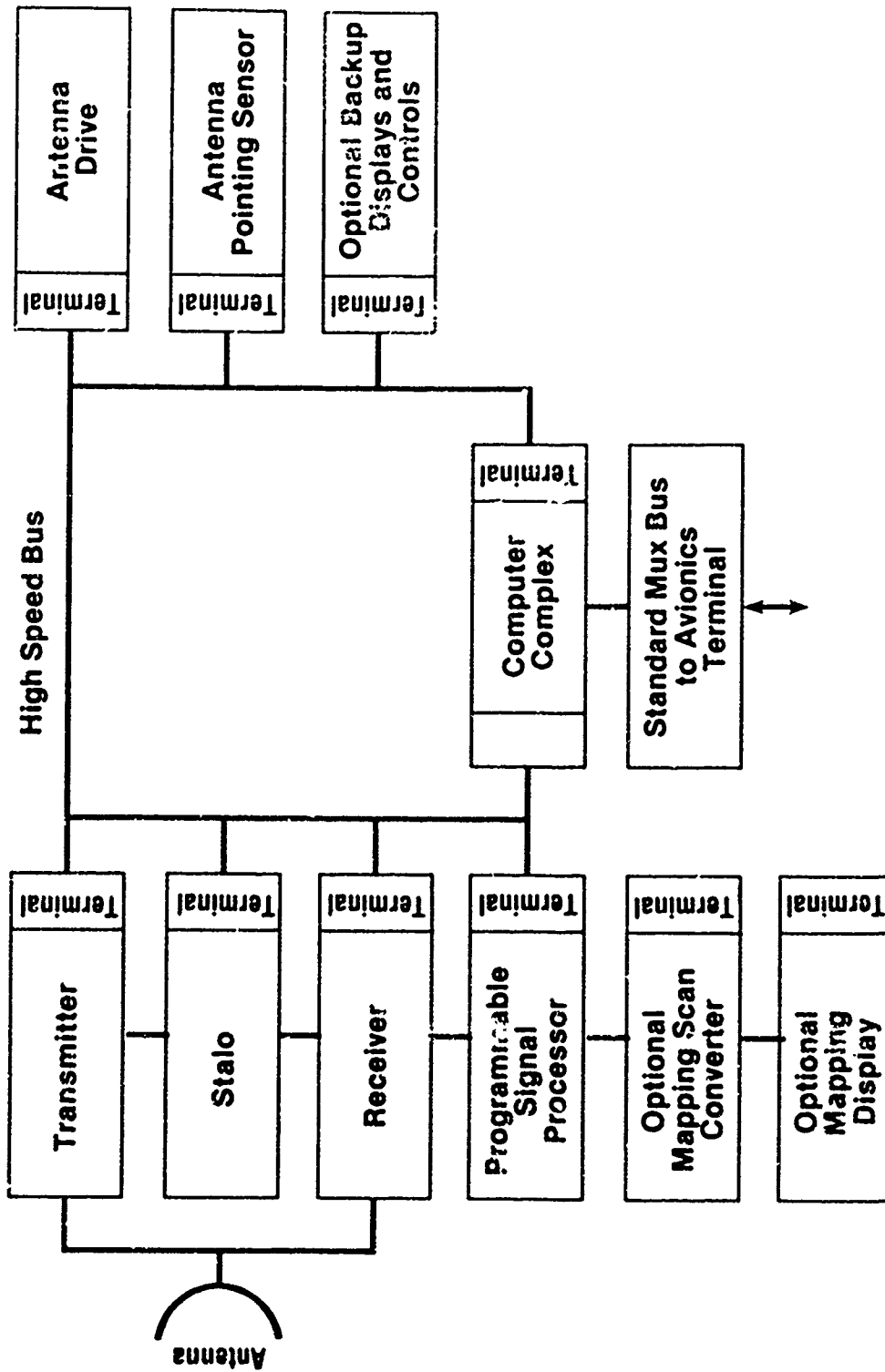


Figure 1

avionics was by a multiplex terminal. All information and control was computer driven, including all built-in-tests, system status checks, and instrumentation data transfer.

The WX Program led to a successful Westinghouse competition for the F-16 Radar. During the development of this radar (later the AN/APG-66), optimization of the modularity concept continued. The result was the block diagram shown in figure 2. As you can see the antenna functions were combined in one module; the receiver and stalo into one module called the low power RF; the computer complex. The mux terminals and the control of the digibus were combined into a computer module and the processor and scan converter into a digital signal processor module. Each module had its own self-contained power supply and the necessary hardware to perform self-test and fault isolation.

The AN/APG-66 then formed the basis for several other modular designs. The U.S. Custom Service Integrated Sensor System involves an APG-66 radar integrated with a forward looking infrared (FLIR) set. Both are installed in a Cessna Citation II business jet and utilized for the intercept and apprehension of border intruders and smugglers. Radar integration into the Citation was facilitated by the addition of a second radar computer, reprogrammed to interface the APG-66 with the rest of the avionics system.

This same radar architecture was the basis for the fire control radar of the Sgt. York (DIVAD) Air Defense Gun for the U.S. Army. The Sgt. York system is depicted here in block diagram form (figure 3).

As F-16 mission requirements evolved, it became necessary to expand the APG-66 mode complement adjustable to new tactics, scenarios and weapons. Initially, many radar changes were adopted via OFP software change (radar improvements are being implemented), but eventually there will be sufficient hardware changeover required to warrant the incorporation of a programmable signal processor (PSP), replacing the existing radar computer and digital signal processor (see figures 4 and 5). The PSP extended the software flexibility inherent in the radar computer into the high-speed processing portion of the radar.

At the same time, the decision was made to employ a higher order language (HOL) in software and adopt a more advanced computer instruction set architecture based on the Westinghouse Milli-EP design. A short time later, these were modified to be fully compliant with the latest U.S. Air Force standards, namely J73 HOL (Mil-Std-1589B) and Mil-Std-1750A. The Improved APG-66 system, incorporating two other new LRUs beside the PSP, is being developed for F-16C production effectivity.

To address the radar mission requirements of the B-1B Bomber, Westinghouse was able to use a great deal of the Improved APG-66 designs and fully employed the same digital standards. The Operational Flight Program (OFP) for the B-1B is very different from F-16 in terms of modes. It is a true measure of radar technology maturity and modular design that these requirements are met with a high degree of hardware commonality and

APG-66 Simplified Block Diagram

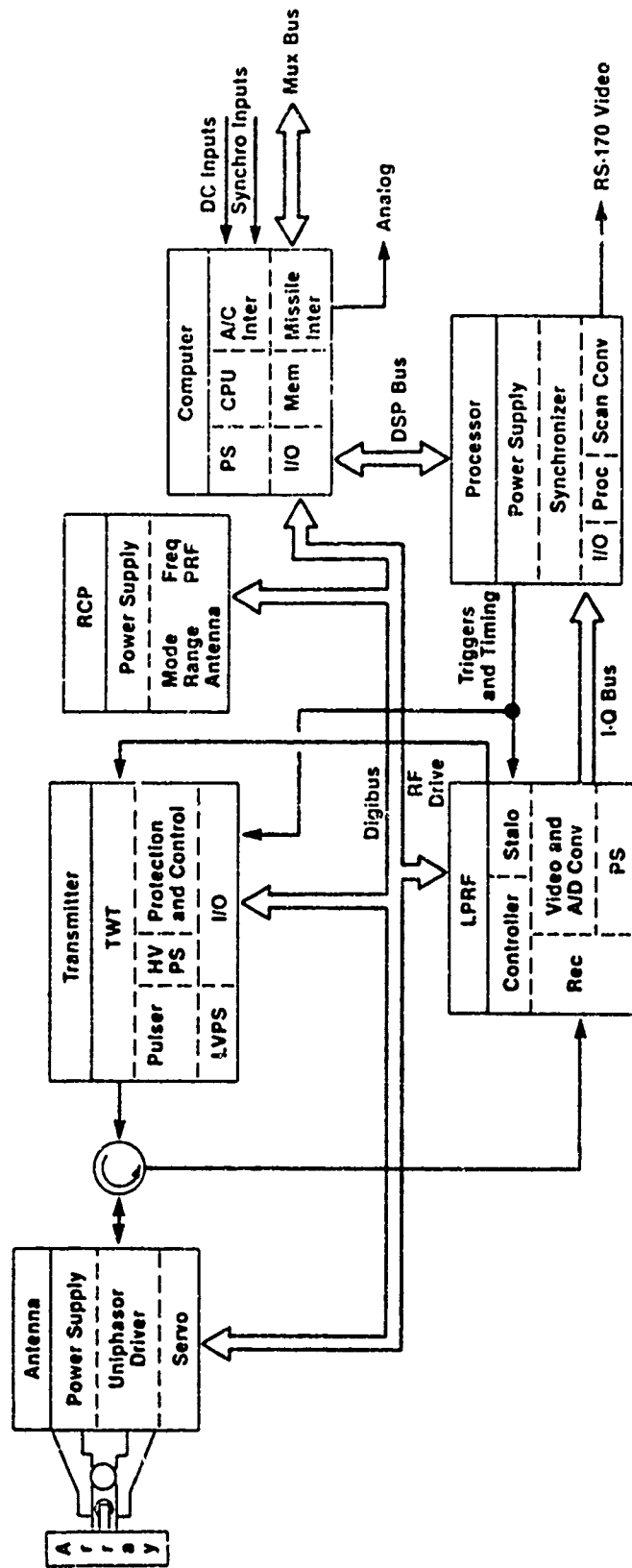
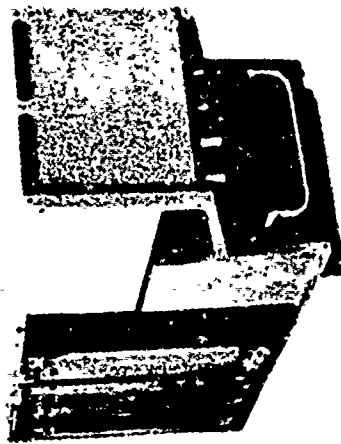
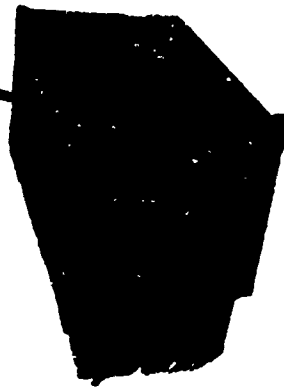
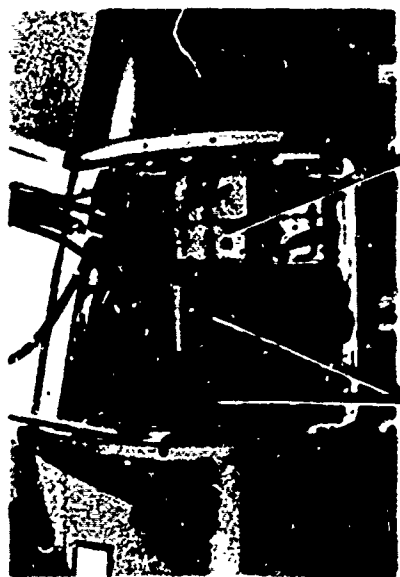


Figure 2

Improved APG-66



New Programmable
Signal Processor (PSP)

New Modular
LPRF

New Dual Mode
Transmitter (DMT)

DTIC 64112

Figure 4

Improved APG-66 Block Diagram

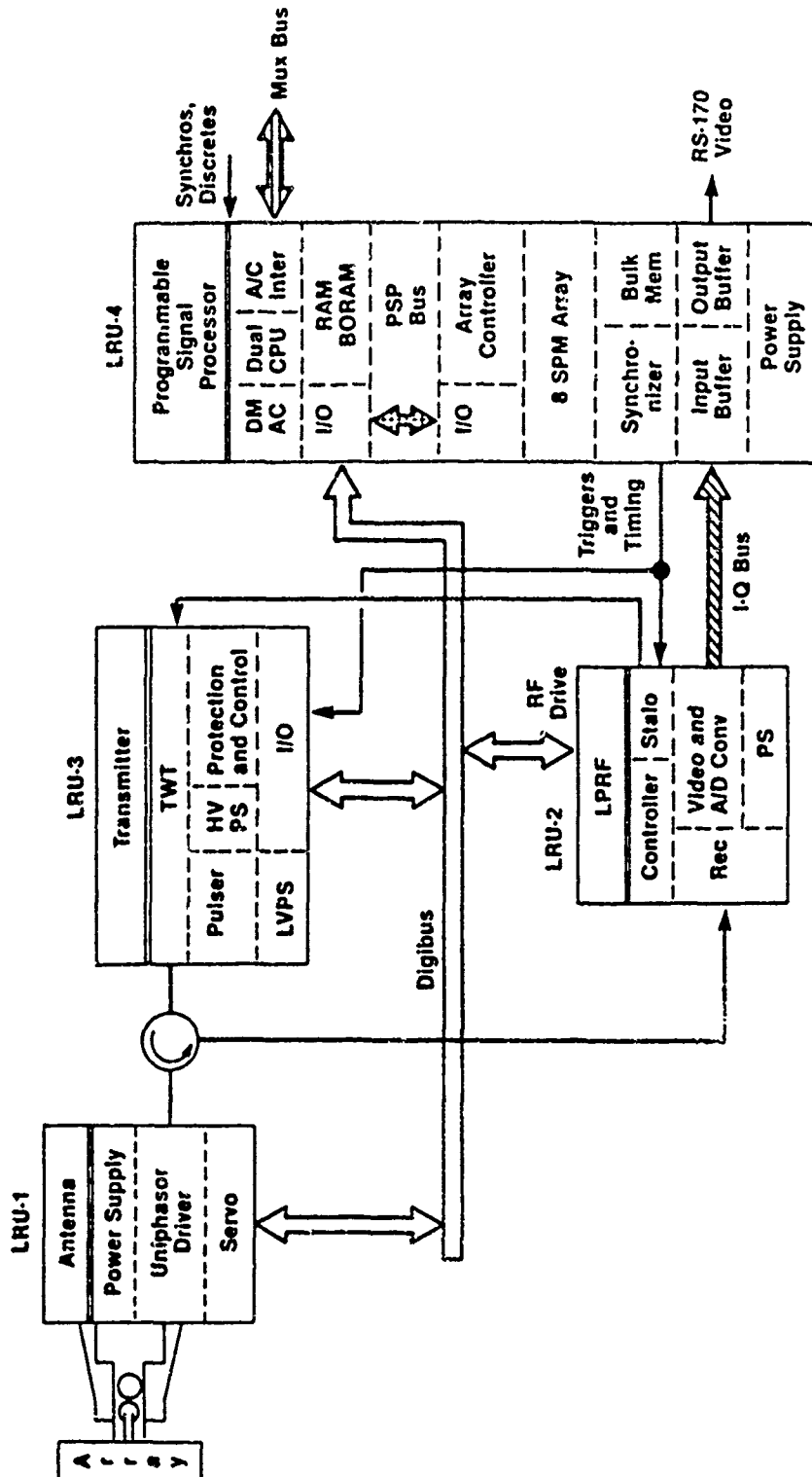


Figure 5

identity. The B-1B Offensive Radar System, shown here (figure 6), consists of essentially two Improved APG-66 systems which utilize a common electronically scanned phased array antennas. The redundant hardware is employed solely for increased reliability and is not related to individual mode performance. Only one set of LRU's is required or used to perform all radar modes. Sophisticated LRU/SRU interconnections assure continued safe radar operation in the event of a failure in one part of the ORS system.

When the Air Force standards came about, Westinghouse changed the interface terminal to Mil-Std-1553 series. The computer went from a Westinghouse standard millicomputer to the Air Force Mil-Std-1750 series instruction set architecture. The operational software went from assembler language programming to Jovial language initially, J73I, and presently J73 using the Air Force-developed compiler and support package with some additional Westinghouse-generated packages.

The basic architecture and Air Force standards are now applied to Westinghouse's three major radar programs - The B-1B Offensive Radar System, Improved AN/APG-66 Radar for the F-16 and the Army's Sgt. York DIVAD Gun System. Through use of the Air Force standards and the fundamental radar architecture Westinghouse has thus developed the first multimode radar family that uses commonality in major Department of Defense programs. One of the major challenges was to design the modules for vastly different environments such as the latest bomber, fighter, and air defense tank.

In order to ensure maximum commonality of the major production programs, Westinghouse structured its management of the programs into an organization as shown in figure 7. I will now discuss the program managing aspect ; which to my knowledge is a first in the radar field.

The Executive Committee was formed to manage commonality, disseminate information, and represent Westinghouse in a multi-customer situation. Composed of the program managers of all involved program activities, the committee is chaired by a manager at the division level. A full time Configuration Coordinator serves as the arms and legs of the Committee in dealing with the diverse functional groups making up each program organization. The Executive Committee and its procedures are in addition to normal Configuration Management procedures. The cornerstone of these procedures is the functioning Program Configuration Control Board (CCB). The CCB serves as the forum for analyzing proposed changes to the functional or physical parameters of the hardware. The Program Manager is the Chairman of the CCB and is the final arbiter in change control decisions. The members of the CCB who assist him in making the decisions represent all functional organization such as Engineering, Manufacturing, Quality and Reliability Assurance, Software, Integrated Logistic Support and Configuration Management. The Executive Committee is also the final authority when individual configuration control boards do not agree on changes that will reduce existing identicalities or commonalities among systems, unless incorporated by all involved programs. The Committee serves as a forum for introducing problems uncovered by individual

B-1 B Offensive Radar System

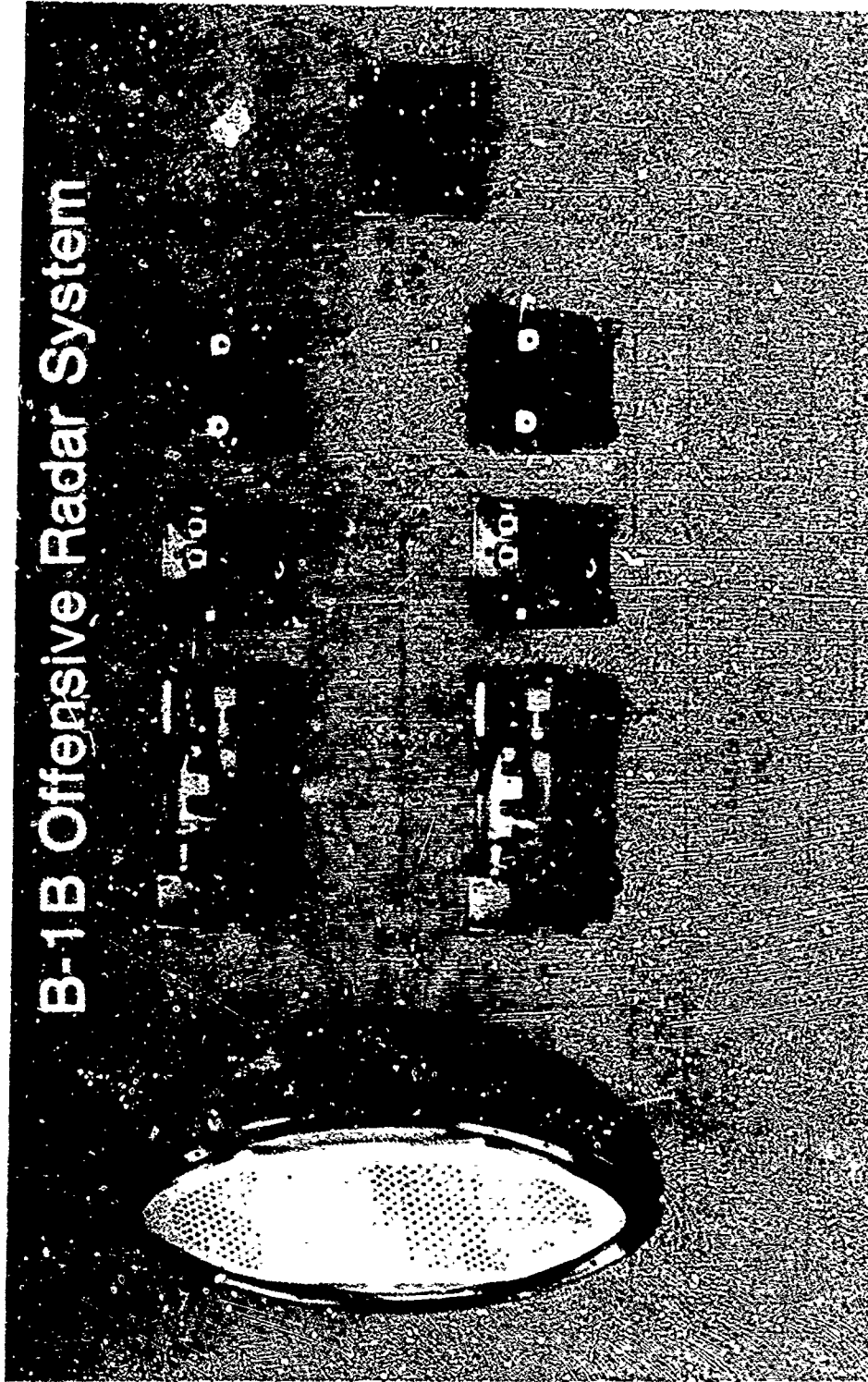


Figure 6

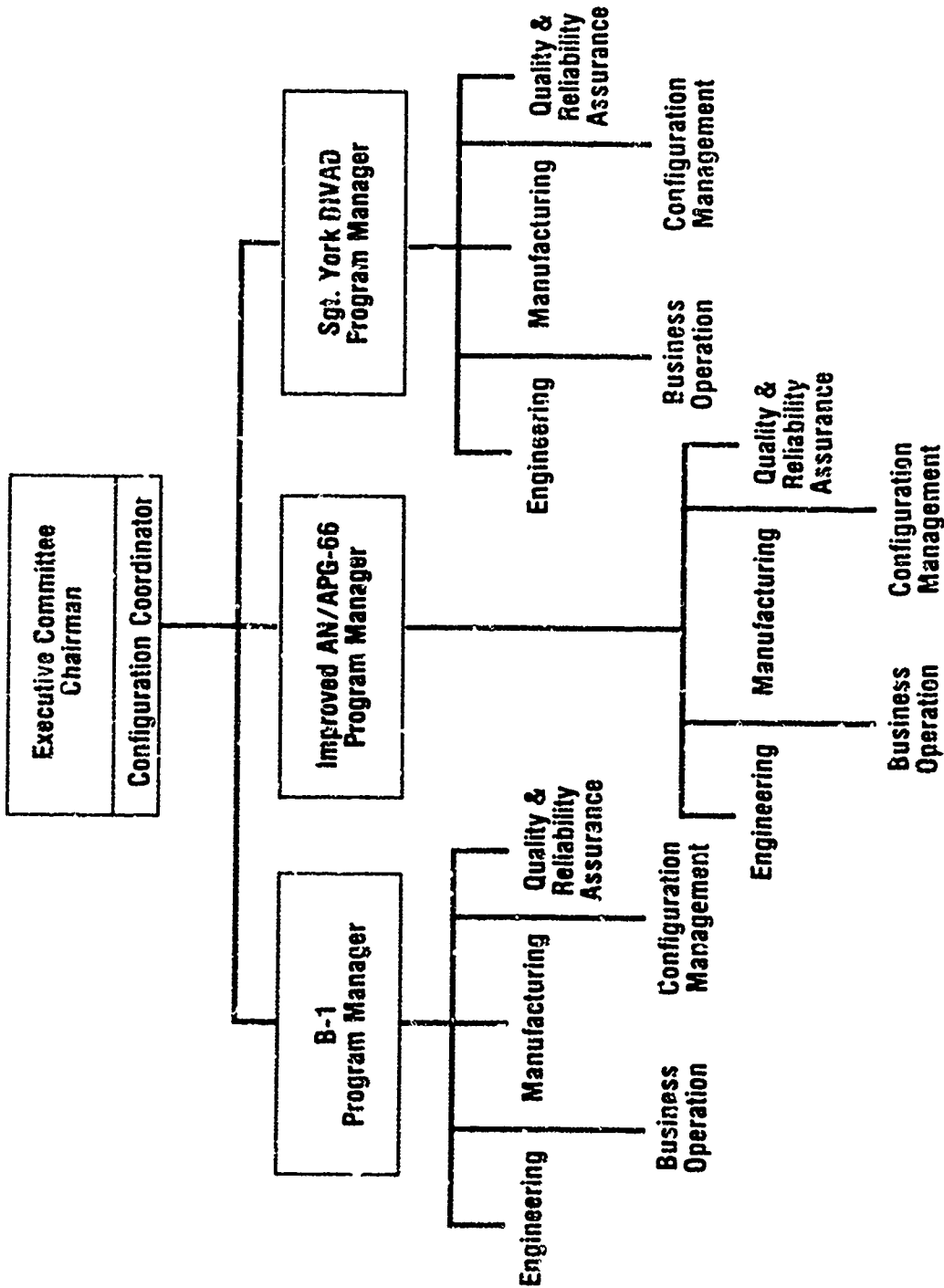


Figure 7

programs which impact other programs. It also has the authority for assigning activity responsibility to individual programs for the solution of problems that impact more than one program. Another one of the major functions of the Committee is to coordinate the logistic support and planning to capitalize for the customers the advantages of identity and commonality.

In the documentation area the Committee objective is to control multiple program use of identical documentation. Due to the shared identical hardware and its supporting documentation, new procedures describing the handling and routing of Change Requests (CRs) and Revision Notices (RNs) had to be developed. The procedures are to deal with the use of drawings and hardware across program lines. It is intended that this be accomplished with a minimum of extra paperwork processing while maintaining the necessary interprogram control and traceability.

A Multiprogram Commonality Report is the means by which identical drawings used on one or more programs are identified. It is updated and distributed weekly.

The procedure for processing multiple program CRs is as follows. When each CR is received by the originating program's Configuration Management Office (CMO), in addition to whatever processing is done for that program, the drawing number on the CR is checked against the Multiprogram Commonality Report. When the document number on the CR matches one on the Commonality Report, the CMO attaches a Multiple Usage Change Approval Sheet to the CR and notes on that sheet the affected program(s). While the originating program's Configuration Control Board (CCB) deliberates on that change, these copies are routed to the other user program's CMO for CCB consideration. If any or all other user programs do not agree on any particular change, that CR shall be brought before the Executive Committee for review. However, approval of a CR by any program is sufficient authority for that program to proceed with work on that change. Once action has been taken by all user programs, the approved/disapproved copy of the CR shall be routed back to the originating program's CMO.

For Revision Notices handling and routing is as follows. When an RN is received by the originating program's CMOs, the document number is checked against the Commonality Report. If a match is found, the Multiple Usage Change Approval Sheet is filled out and attached. The RN Continuation Sheet for multiple program sign-off will also be attached. The change effectivity assigned by the originating program on the RN shall be the only authorized effectivity. After the originating program's CCB has acted on the change, it shall be handcarried to the next user's CMO for processing and CCB consideration. Each CCB shall act on the RN and pass it to the next CCB in turn. When all CCBs have reviewed the RN, the change is authorized. If any program does not approve the RN it shall be diverted and brought to the attention of the Executive Committee for a decision regarding continuation of identity. The responsibility for creating new unique documentation shall belong to the program(s) not approving the change.

In the operational software area a common kernal approach is used for the executive and LRU controls. The executive software permits differences for the various individual requirements. Even the basic general-purpose Mil-Std-1750A computers and common programmable signal processors have a common kernal control. All three programs use a common testing approach. They all use a common configuration control approach. Software testing breaks down into two areas; software benches and system benches. Both areas are common to all three programs.

The use of identical hardware for different applications carries both advantages and disadvantages. The major problem is one of coordination. Different contractual and environmental requirements, different schedules, configuration management milestones, and multiple customer relationships all contribute to a tendency for each program element to satisfy their own unique requirements with minimum regard for the overall picture. The management techniques discussed earlier strongly minimize the impact of this parochial attitude. On the positive side there are several significant advantages to the usage of identical hardware. First there is a great relief to the cost and schedule risk of developing hardware, software, and tooling more than once. Concomitant with this risk relief is the ability to procure material in larger and therefore less costly lot buys. Also there is a savings in both time and money in the manufacturing cycle due to quicker progress along the learning curve. Lastly, a great potential for cost savings exists in the logistical support arena (common spares, test equipment, repair facilities, and handbooks are examples).

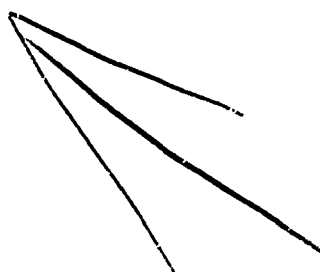
The pursuit of standardization of radar hardware for multiple applications has produced great advantages. It has also taught a valuable lesson: start early. Any attempt to achieve modular commonality or other standardization efforts are handicapped if they are not begun in the definition phase of a program. Once the contracts are written and accepted, commonality is doomed without subsequent customer concessions. The closer to the front end that this effort is initiated the greater the rewards.

Conclusions

Westinghouse has made good progress by utilizing Air Force standards and intends to continue their use in future programs. Westinghouse has structured its management of programs to handle standardization matters in an organized and highly efficient manner, providing great experience in organizing and running such programs where standardization and commonality are important.

Biographical Sketch

Carl S. Shyman. Mr. Shyman received a BSEE from the University of Washington in 1948 and a MSEE from Stanford University in 1949. From 1949 to 1955 he worked for North American Aviation (later Autonetics) where he developed an automatic landing system for the Navaho Missile and participated in the development of the NASAAR Radar for the F-104. From 1955 to 1964 he worked for the Boeing Company where he was responsible for managing the Bomarc Target Seeking Radar programs and later the Bomarc and Dynasoar Airborne Guidance systems. He moved to Westinghouse in 1964 and since that time has had a number of radar and guidance systems assignments, including Department Manager for the F-16 radar system. He is currently General Manager of the Avionics Division at the Westinghouse Defense Electronics Center.



A GENERAL PURPOSE COMPUTER ARCHITECTURE INVESTIGATION FACILITY

Dean W. Gonzalez, 1Lt., USAF

RADC/COEA
Griffiss AFB NY 13441
315-330-2558

Lt. Gonzalez is a 1980 graduate of the Air Force Academy. After receiving his Bachelor's Degree in Computer Science, he was assigned to the Computer Architecture Section of the Information Sciences Division at the Rome Air Development Center. His current work includes emulator development and performance monitoring, and tool development for the architecture investigation facility.

ABSTRACT

This paper discusses a set of hardware and software tools that have been integrated into a general purpose emulation facility. This facility is available across the United States via the ARPAnet. The tools allow users to produce an emulator using a high order programming language, execute the emulator via a remote interface to a Nanodata QMI and execute assembly language software on the emulator. Experiences with the development and use of the facility are discussed. Views on the extension of the facility's capabilities are also discussed.

INTRODUCTION

Prospective users of any of the multitude of new computer architectures benefit greatly if they are able to evaluate an architecture before committing themselves to it. During this evaluation they examine the performance of the architecture and the ease or usefulness of programming the particular machine. To perform this evaluation an implementation of the computer, a method of interfacing to it and software development and support tools are necessary. Unfortunately, each computer development system uses a different set of tools and, of course, different hardware. The cost of this package for the investigation of an architecture can be prohibitive.

Computer simulations can reduce the cost and trouble of architecture investigation and software development. This technique is typified by large software packages running on multiple user mainframe processors. A serious drawback to using these packages is the speed at which the simulator implements execution of the target machine.

Emulation ("hardware mimicking") of the target machine is a second alternative. This term has been used to mean several different approaches to hardware

mimicking but is used here to mean the microprogrammed implementation of an architecture. Microprogramming is a technique of computer control where user defined sequences of binary digits are used to control the opening and closing of specific hardware logic gates in a computer and is a standard approach to implementing new computer architectures. This method of control gives the user more power over how the machine functions and causes the machine to run faster than one under software control. By allowing any user to remicroprogram a host machine, many target machines become available.

User microprogrammable machines are not generally available to a casual user due to the physical unavailability of such machines and the difficulty in programming them. It is possible, though, to make a dedicated emulation engine available to a broad user community by connecting it to a computer network. A Nanodata QM-1 user microprogrammable computer¹ at Rome Air Development Center (RADC) has been connected to the Department of Defense (DoD) sponsored ARPA network and is accessible to users across the United States, Hawaii and England.

Two computers on the ARPAnet host the tools necessary for use of the investigation facility that is the subject of this paper. A DECSYSTEM-20 computer (RADC-TOPS20) serves as the front end for the QM-1. The QM-1 is physically connected to the DEC-20 and is used as an emulation engine peripheral by the QPRIM (QM-1 Programmer's Research Instrument) software running on the DEC-20 (Figure 1). Thus, the first boundary to widespread use is eliminated; anyone who can access the ARPAnet can also access the QM-1. A Honeywell 6180 (RADC-MULTICS) hosts other tools necessary for the facility; the Smite compiler and the Meta Assembler. A compiler for a High Order Language (HOL) called Smite removes the last boundary to the widespread use of emulation engines. This compiler and a general purpose table-driven cross-assembler, the Meta-Assembler used for developing software to run on an emulator, are the two major tools on the 6180 that contribute to this facility. The combination of these software tools and the QM-1 hardware provides a reconfigurable computer architecture investigation and software development facility.

SMITE

Smite is a Hardware Description Language (HDL) that allows users to create functional descriptions of computer architectures². These descriptions are functionally equivalent to the machine they describe but may not and need not be in one to one correspondence with the target architecture. The Smite description need only mimic the actions of the target architecture at the level of target machine register transfers. That is, only the interaction of accumulators, general purpose registers, status registers and the like must be described.

Smite is a procedural HDL that has high order language control constructs. There are three basic repetitive constructs:

```
DO WHILE ...  
DO UNTIL ...  
DO FOR ...
```

and two basic conditional constructs:

```
IF ... THEN ... ELSE ...      CASE ,exp. ...
```

Smite also has a complete set of arithmetic and logic operators (+, -, =, >=, <=, /=, OR, AND, XOR, NOT), allows block structuring of programs (BEGIN ... END) and extensive use of subroutines.

All Smite data items must be built from individual bits. The data items may include flags, registers, memories and several types of hardware data items (switch, light, port, clock). Arrays of all data items may also be constructed and data items may overlap one another. Since all data items are declared by the user as groups of bits, type checking is nonexistent and all data items are type compatible.

Take the MIL-STD-1750A add register (AR) instruction as an example³. The instruction adds two general purpose registers, RA and RB, and the result is stored in RA.

Assembly version Machine version

AR RA,RB



{A1 is the eight bit operation code and 6 and 8 (four bits each) are two of sixteen general purpose registers (0 - 15)} Assuming the basic instruction cycle of the machine has been implemented, using the following declarations,

```

DECLARE RA <0:3> REGISTER, "pointer to reg. A"
        RB <0:3> REGISTER, "pointer to reg. B"
        IR <0:15> REGISTER, "instruction register"
        GP-REGS [0:15] <0:15> REGISTER; "16 register file"

```

execution of the following Smite statements implement the instruction.

```

RA <- IR <8:11>; "get pointer to A"
RB <- IR <12:15>; "get pointer to B"
GP-REGS[RA] <- GP-REGS[RA] + GP-REGS[RB];

```

(The addition operator implicitly performs the two's complement addition operation.)

Notice that this description does not deal with any of the register, bus and the arithmetic logic unit implementation details. There also is not a one to one mapping to the registers of the 1750A. For example, registers RA and RB are temporary data items that have no counterpart in the 1750A. This allows for a straight forward description of the functional characteristics of the hardware that is not overwhelmed with low level details.

Smite is used to write emulators of architectures. These are cross-compiled from Smite to vertical microcode on a Honeywell 6180 (Multics) at Rome Air Development Center (RADC) (Figure 2). The microcode is targetted only to the Nanodata QM-1 and is an extension of a "standard" vertical microcode language used on all QM-1s⁴. When running, the emulator microcode resides in user writeable control store and is executed by horizontal microcode running in a lower level microcode memory, nanostore.

Before the QPRIM system existed the user of an emulation had to be physically present at the QM-1. There, after downloading a microcoded emulator from Multics to a QM-1 disk file through a hardware link, one could test, debug and run the emulator using the Smite Application Support Software (SASS) ². This package is hosted on the QM-1 but is a single user system and requires that the user be physically located at the QM-1 in the RADC computer facility; which are serious drawbacks to the use of Smite and the QM-1.

QPRIM

QPRIM now connects the QM-1 to the ARPAnet via a complete software system replacing SASS ². The QM-1 is physically connected to a DEC SYSTEM-20 and shares the System-20's main memory. Control of the QM-1 is implemented via a specific software system that uses the QM-1 as a TOPS-20 peripheral. Thus, QPRIM is a multi-user system, replaces SASS and eliminates the requirement of being physically located at the QM-1.

To use QPRIM, one merely logs in to RADC-TOPS20 from anywhere on the ARPAnet and runs the QPRIM program. The system is completely interactive with extensive prompting of user commands (Figure 3).

One must initialize QPRIM with two files before QPRIM does anything significant. First, an emulator that is written in Smite, compiled on RADC-MULTICS and transferred to TOPS-20 must be loaded using the LOAD command. Second, a previously assembled file that describes the emulation of the target machine must be loaded using the TABLES command.

The descriptor table is created by assembling a descriptor table source file (Figure 4). This file contains a list of calls to standard (QPRIM standard) macros written in the TOPS-20 MACRO language. The descriptor table is a database that defines the target machine architecture as implemented in the emulator. It also supplies conventions for numbers, character sets and target machine instructions which QPRIM uses to parse interactive user input. This allows for a great range of user friendliness since the user of QPRIM tells QPRIM how friendly to be!

QPRIM can be used in either of two modes. In the first mode, an emulator writer creates, tests and debugs an emulator and its descriptor table. Once he is satisfied with the package, he may "install" the emulator as a tool under QPRIM. Now the second mode of QPRIM can be entered. A user who has no expertise with Smite, emulator testing and debugging or descriptor tables uses the tool as if he was using the target machine instead of QPRIM. Now, QPRIM is used merely as an interface to the tool. QPRIM has, in effect, become a development system for the target machine.

QPRIM has a large set of commands that implement the full functional spectrum of a typical development system. The user of QPRIM may:

1. use all types of I/O devices,
2. examine and change emulator microcode interactively,
3. examine and change the contents of QM-1 general purpose registers and memory interactively,
4. examine and change the contents of target machine registers and memory interactively,

5. assemble and insert, disassemble and display target software interactively.

META-ASSEMBLER

Given QPRIM and Smite, one can insert machine code into target computer memory and execute it. However, for the generation of large amounts of code this is too laborious and use of a symbolic assembler is necessary. The Meta-Assembler is a general purpose cross assembler that is target machine independent⁶. It can be used to assemble target machine language for virtually any machine.

The Meta-Assembler is hosted on RADC-MULTICS. To use it, one must supply a data set of target machine characteristics (e.g. memory size, program counter width, instruction width and the character set) and description of target machine assembly language syntax and semantics. This definition file is the database used by the Meta-Assembler. It contains all the target machine specific information the Meta Assembler requires. This information is the set of rules for translating the source code to machine code (Figure 5).

The syntax of most assembly languages can be represented by a set of forms of syntax. Each form must be listed in the definition file. The forms that can be used by a particular instruction are then listed side by side with the instruction mnemonic and opcode value. The semantics of instructions is defined by this list. The final necessary ingredient of the definition file is a brief target machine description. This description states the registers and memory size that the Meta-Assembler may use.

The Meta-Assembler is reconfigurable and can be tailored to the syntax conventions of familiar target machine assemblers. Assembler language directives are defined to operate just as in other assemblers. This means that one inputs assembly language programs to the Meta-Assembler as if one was using an assembler hosted on the target machine!

The Meta-Assembler provides many of the built-in services of high quality assemblers. Macros can be defined and used in the assembly language program. Also, one can define structured assembly language constructs in the definition file and then use these in the assembly language. The Meta-Assembler also produces relocatable object code.

A linker must be used to link the Meta-Assembler output before the object code can be used. The linker allows one to piece together object modules that were assembled separately. Object code modules may also be relocated at link time. The linker is typically used with a loader that prepares object code for loading into QPRIM. Both tools are also hosted on Multics.

The use of these three tools (assembler, linker, loader) results in an object code file that may be directly loaded into QPRIM and used by an emulator. The code must, of course, be transferred over the ARPAnet to RADC-TOPS20. The QPRIM RESTORE command is used to load object programs prepared with this method into emulator main memory. This is the path a tool user follows to create and run programs on a target machine.

EVALUATION OF THE FACILITY

At RADC, several emulators have been created and used within the framework of this architecture investigation facility. Three completed emulators, ready for tool users, are the Intel 8080, Motorola M6800 and the AN/UYK-20. An emulator for the MIL-STD-1750A instruction set architecture is nearing completion and an emulation of MIL-STD-1862A is being created. Development of these emulators has stressed the facility enough to highlight good qualities, uncover problem areas and show where enhancements are required.

The Intel 8080 emulator served as the initial test case for QPRIM. It was used to prove that the system worked. The emulator is a description of the 8080 CPU and no I/O devices are supported. This emulator is based on an 8080 emulator written in Smite and running on the QM-1 in stand-alone mode. Therefore, its usefulness in an evaluation of the facility is minimal.

The Motorola M6800 emulator is the first to support any I/O devices. This emulator was used to test the coherency and efficiency of all elements of the facility. Extensive amounts of target software were worked through the path from the Meta-Assembler to running on the 6800. Some of the software used was obtained from a commercially available microcomputer system. The software ran exactly as it does on off-the-shelf computer systems.

The next step in test evolution was the AN/UYK-20 emulator. This emulator includes a full complement of I/O devices running under one input/output Controller. These devices include a console, a paper tape punch/reader, magnetic tape unit and a disk drive unit. This emulator is written in MULTI and has proven QPRIM's ability to effectively simulate several types of I/O devices.

The MIL-STD-1750A emulator is the largest (the 1750 is a 16 bit machine) that has been used successfully in the facility. It is written in Smite and 1750 assembly language is being run successfully on it. The Meta-Assembler is used to assemble the software which is then automatically prepared for loading and running under QPRIM.

Two other emulators, MIL-STD-1862A and Motorola MC68000 have uncovered problems with parts of the facility. Both of these architectures are 32 bit machines. The Smite language allows for the description of these architectures as easily as it does for the 8 bit machines. However, the Smite compiler has problems mapping the many complex 32 bit operations onto the 18 bit QM-1 registers. Mapping of 8 and 16 bit operations onto the QM-1 registers is not nearly as difficult and the process works well. Unfortunately, though, the compiler has many bugs in the logic which maps operations of words longer than 18 bits. The emulation of the 1862 is serving as a verification tool for the compiler and, using it, bugs are being found and corrected.

The final issues in the evaluation of this facility is the efficiency of the tools and the efficiency of emulators produced by the Smite compiler. Unfortunately, the Smite compiler itself is very inefficient. Compile times for the larger emulators (e.g. MIL-STD-1750A, M68000 (about 2000 lines)) are near one hour of CPU time. The Meta-Assembler is also an inefficient tool that requires about 20 minutes of CPU time per 1000 lines of target software. The emulators produced by the Smite compiler, though, have a slowdown factor of less than 100 when compared to actual hardware and the capability exists to speed them up even more.

A new, more efficient version of the Meta-Assembler is installed on the DEC-20. Preliminary results show that run time is decreased by more than one order of magnitude. Once a linker and loader are available for this version of the assembler, it will replace the Multics hosted Meta-Assembler.

TOOL IMPROVEMENTS

Based on experience with the tools in this facility, two approaches can be taken to the future of the facility: 1) improve the tools already available and, 2) develop more powerful tools.

A needed improvement is in input/output hardware description with Smite. Historically, I/O processing has been the sore point of both hardware and software systems. Smite was designed to allow efficient description of computer components but I/O descriptions were given no real attention. A very primitive facility for sending information to the outside world is available, so I/O is possible, but description of complex I/O processes, channel controllers and I/O processors is cumbersome. Evaluation of the requirements for efficient I/O description could produce extensions to the syntax and semantics of Smite to provide a complete computer description capability. This evaluation would either accept the QPRIM I/O support package as standard and interface with it, or develop a new support package.

Steps have already been taken to integrate a more efficient Meta-Assembler into this facility. A more useful tool than this Meta-Assembler is one that allows the independent definition of opcodes and operand elements. For example, the MIL-STD-1862A instruction set allows one to use up to three operands for some instructions. Each of these operands can be any of 180 types ! The Meta-Assembler requires the user to explicitly list the more than five million combinations of these operands! A more reasonable approach is a meta assembler that requires one to enumerate only the twelve generic 1862 operand formats. Then the user provides a list of mnemonics, opcodes and a set of operands allowed in each respective position. This allows one to write very compact descriptions of assembly languages. The existence of such a tool is not known.

CONCLUSION

Smite, QPRIM and the Meta-Assembler provide a support facility for both architecture and software investigation. One creates an emulation of any architecture, using the Smite HDL, which runs on the Nanodata QM-1 computer. This emulation, supported by the QPRIM user interface, is then used to execute machine code assembled by the Meta-Assembler. Since these tools are accessible through the ARPAnet, many users have the chance to evaluate, with relative ease, the performance and usefulness of an architecture before committing themselves to it.

REFERENCES

1. Nanodata Corporation, QM-1 Hardware Level User's Manual, Buffalo New York, December 1981.
2. TRW/Defense and Space Systems Group, Advanced Smite Reference Manual, RADC-TR-80-66, Rome Air Development Center, New York, February 1980.
3. "Sixteen-Bit Computer Instruction Set Architecture," MIL-STD-1750A (USAF), 2 July 1980.
4. Burkhard, W. A., R. D. Tuck and R. L. Hartung, QM-1 Multi Microprogrammer Guide, NSWC/DL TR-3834, Naval Surface Weapons Center, Dahlgren, VA., September 1978.
5. Goldberg, Joel and Louis Gallenson, QPRIM System: Tool Builder's Manual, ISI/WP-20, USC/Information Sciences Institute, Marina del Rey, CA., October 1981.
6. TRW/Defense and Space Systems Group, MDAC Meta-Assembler and Generalized Linkage Editor User's Manual, RAFB-6404-1-78-126, Warner-Robins Air Logistic Center, GA., 1 Jan 1981.
7. "Nebula Instruction Set Architecture," MIL-STD-1862A, 1 July 1981, pages 7-16.

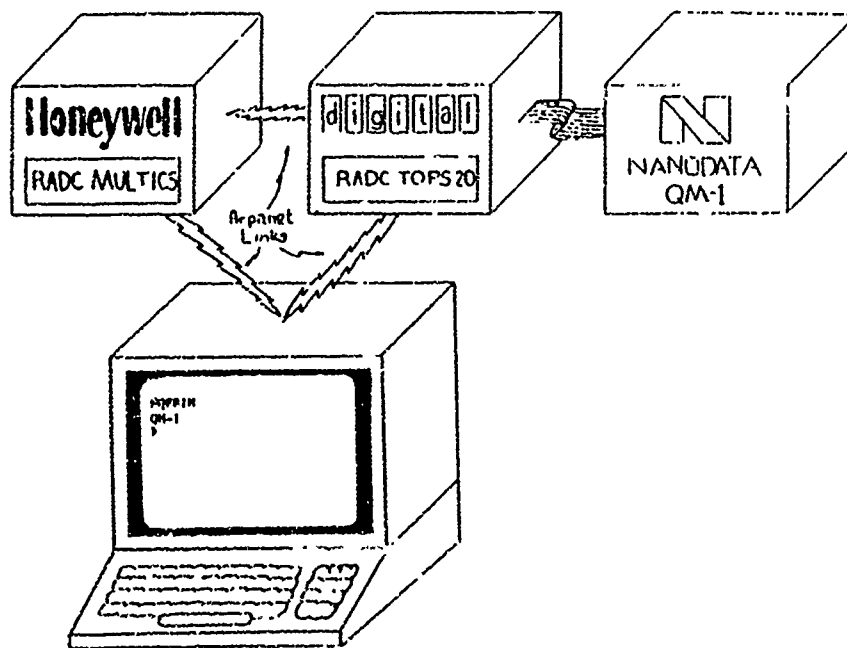


Fig. 1 Facility Hardware

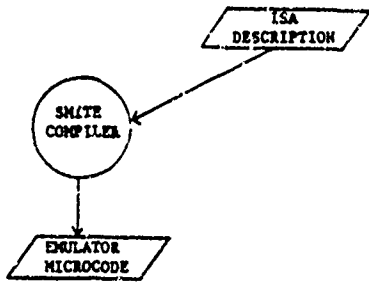


Fig. 2 Creating the emulator.

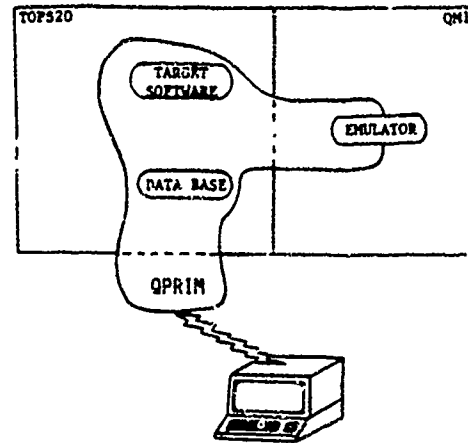


Fig. 3 Control of resources.

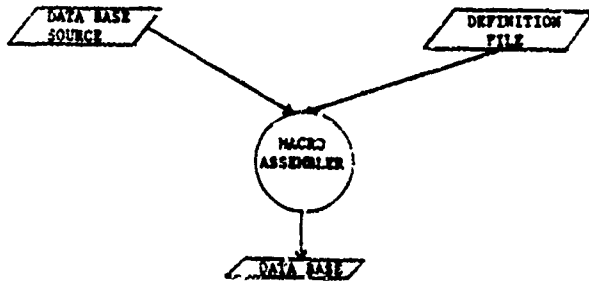


Fig. 4 Creating the database.

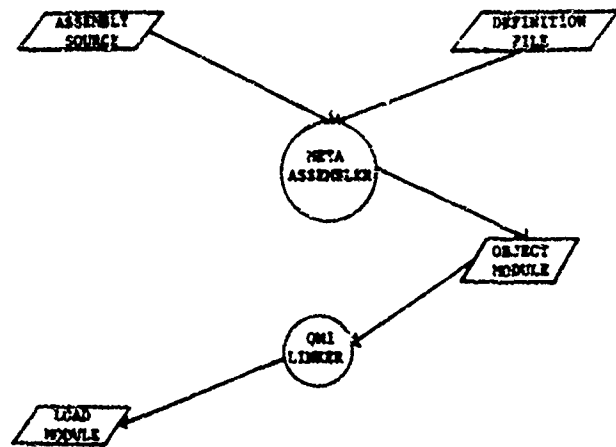


Fig. 5 Creating target software.



AN INTEGRATED APPROACH TO A SUCCESSFUL
EMBEDDED COMPUTER RESOURCE PROJECT

L. G. Egan, Jr.
ITT/Federal Electric Corp.
P. O. Box 1886
VAFB, CA. 93437
805-866-9059

Abstract

This paper describes a number of key milestones and techniques that should be accomplished in order to produce cost effective, embedded computing systems. It stresses the necessity of improved project management to effect an integrated hardware/software system. Improved project management requires a thorough understanding and implementation of the DoD/MIL standards and specifications. If Program Project Managers from both the Government and contractor are knowledgeable in, and motivated to follow, the approved standards/specifications, embedded computer resources, i.e., equipment, computer programs, personnel, facilities, and logistics, will be provisioned at lower costs and per the requirements allocated to each resource. The paper will emphasize the need for systems engineering, work breakdown structures, the systems development process, documentation milestones, DT&E for hardware/software, and integration leading to system certification. This can only be effective via standards compliance.

Introduction

This paper will be an oral presentation using viewgraphs throughout. No additional text material is provided.

BIOGRAPHICAL SKETCH

Leo G. Egan, Jr. has 32 years experience in electronics, missile, and aerospace programs, the last 17 of which have been directly involved in command/control/communications hardware/software and embedded computer resources related to data processing, computing equipment, firmware and software. At present he is Management System Specialist for the Federal Electric Corporation. He has designed, developed and implemented a number of management systems in the areas of project management, configuration management, data management, and software quality assurance, particularly in the management of computing hardware/firmware/software. He lead the design and implementation of the first software configuration management system structured to meet DOD MIL-STD5/SPECS.

Mr. Egan is a recognized expert in Systems Management and System Engineering and has lectured at seven universities and colleges throughout the U.S., and in Europe. He prepared and conducted a 4 day course on the Management of Embedded Computer Resources in Systems. He holds a BS and MS in Electrical Engineering and is presently preparing his dissertation as a Ph.D. designate, having completed all source work. He is listed in "Who's Who in Technology, 1981".

AN INTEGRATED APPROACH
TO A
SUCCESSFUL
EMBEDDED COMPUTER RESOURCE PROJECT

L.G. EGAN
ITT/FEDERAL ELECTRIC CORPORATION

..... THE SOFTWARE PRACTICES MOST APPLICABLE TO
EMBEDDED COMPUTERS SHOULD CLOSELY MATCH THOSE
MANAGEMENT PRACTICES WHICH HAVE BEEN DEVELOPED
FOR HARDWARE ACQUISITION.

BARRY C. DE ROSE AND
THOMAS H. NYMAN
'THE SOFTWARE LIFE CYCLE'
SIGNAL, NOV./DEC. 1977

COMPUTER RESOURCES

- THE TOTALITY OF COMPUTER EQUIPMENT, COMPUTER PROGRAMS, ASSOCIATED DOCUMENTATION, CONTRACTUAL SERVICES, PERSONNEL AND SUPPLIES.
- INCLUDES MICROPROCESSORS, MICROCOMPUTERS AND FIRMWARE

SYSTEMS MANAGEMENT IMPACT RELATIONSHIPS

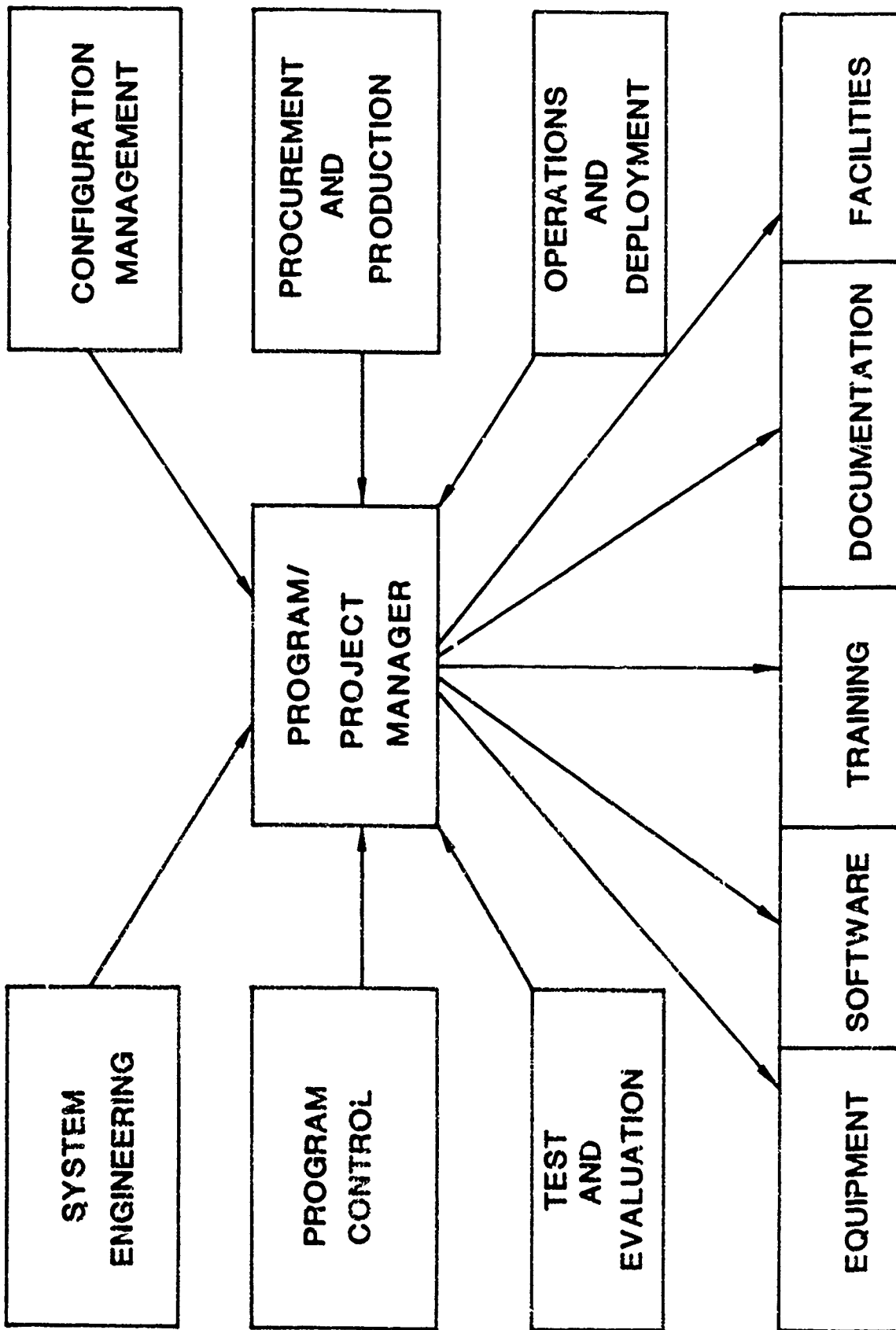


Figure 1.

MATRIX ORGANIZATION

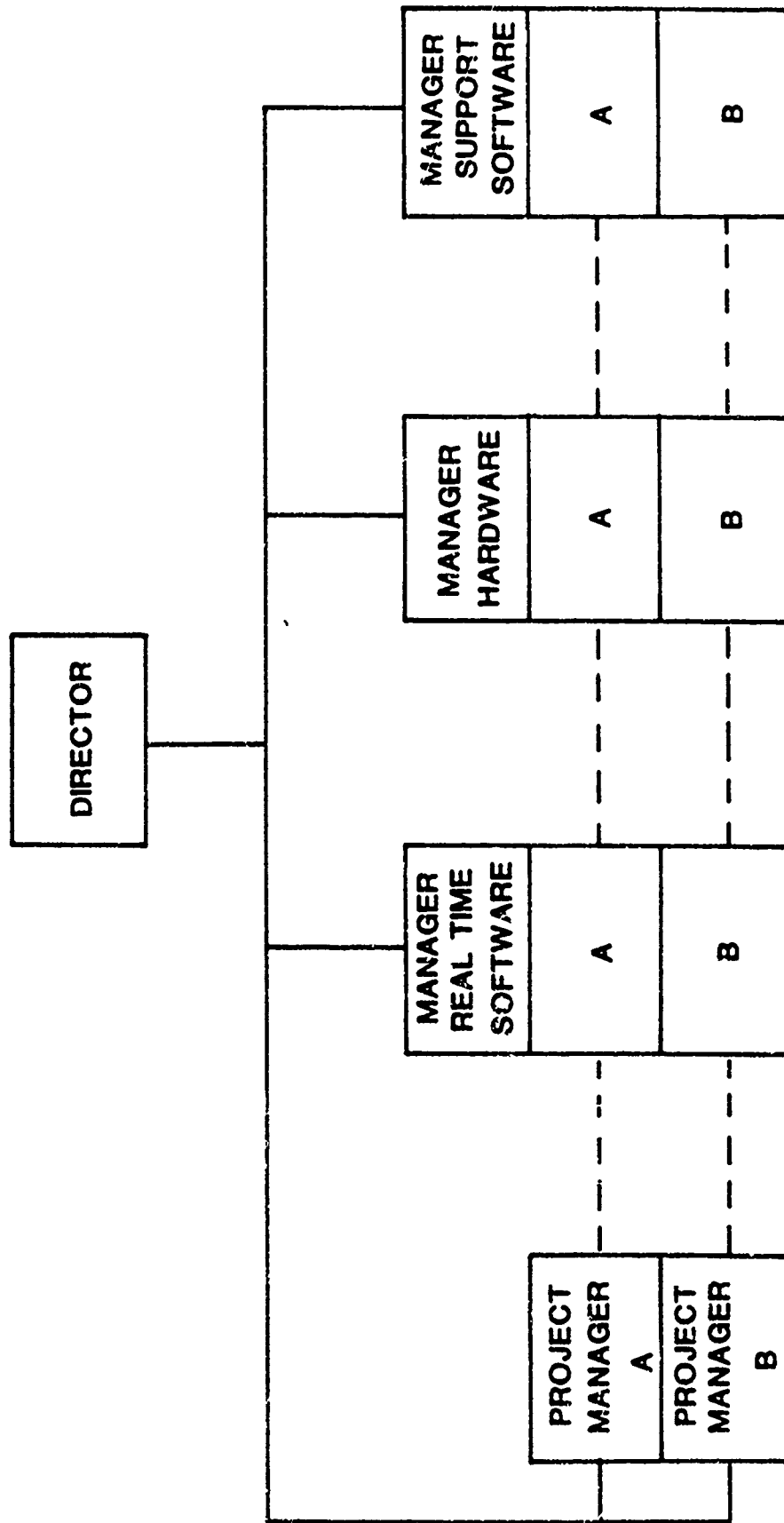


Figure 2.

SYSTEM PHASES

<u>CONCEPT FORMULATION</u>		<u>DEFINITION</u>	<u>ACQUISITION</u>	<u>OPERATIONAL</u>
CONCEPTUAL PHASE PHASE 0 (CUSTOMER)	PHASE I (CUSTOMER OR CONTRACTOR)	VALIDATION PHASE PHASE II (CUSTOMER AND/OR CONTRACTOR)	PHASE III (CONTRACTOR)	PHASE IV (CUSTOMER OR CONTRACTOR)
	USER REQUIREMENTS CONCEPTUAL DESIGN RESULTS IN FEASIBILITY SCHEDULE FUNDING	SYSTEM TRADE STUDIES SYSTEM SYNTHESSES SYS. ANALYSES REQUIREMENTS ALLOCATION RESULTS IN SYSTEM SPEC. SYS. SEG. SPEC. SCHEDULE FUNDING	EFFECTIVENESS STUDIES AND ANALYSES SYSTEM SEGMENT DESIGN REQUIREMENTS DEFINITION AND ALLOCATION DETAIL PLANNING RESULTS IN C/CPCI SPECS. PROGRAM PLANS INTERFACE SPECS. PROPOSAL (TECH, COST, SCHED)	DESIGN STUDIES AND ANALYSES REQUIREMENTS ALLOCATION TO CPC's DESIGN AND DEVELOPMENT PRODUCTION TEST AND LAUNCH RESULTS IN CPC SPECS. ENGINEERING DATA MANUALS HARDWARE SOFTWARE

Table 1.

THE SYSTEMS ENGINEERING PROCESS

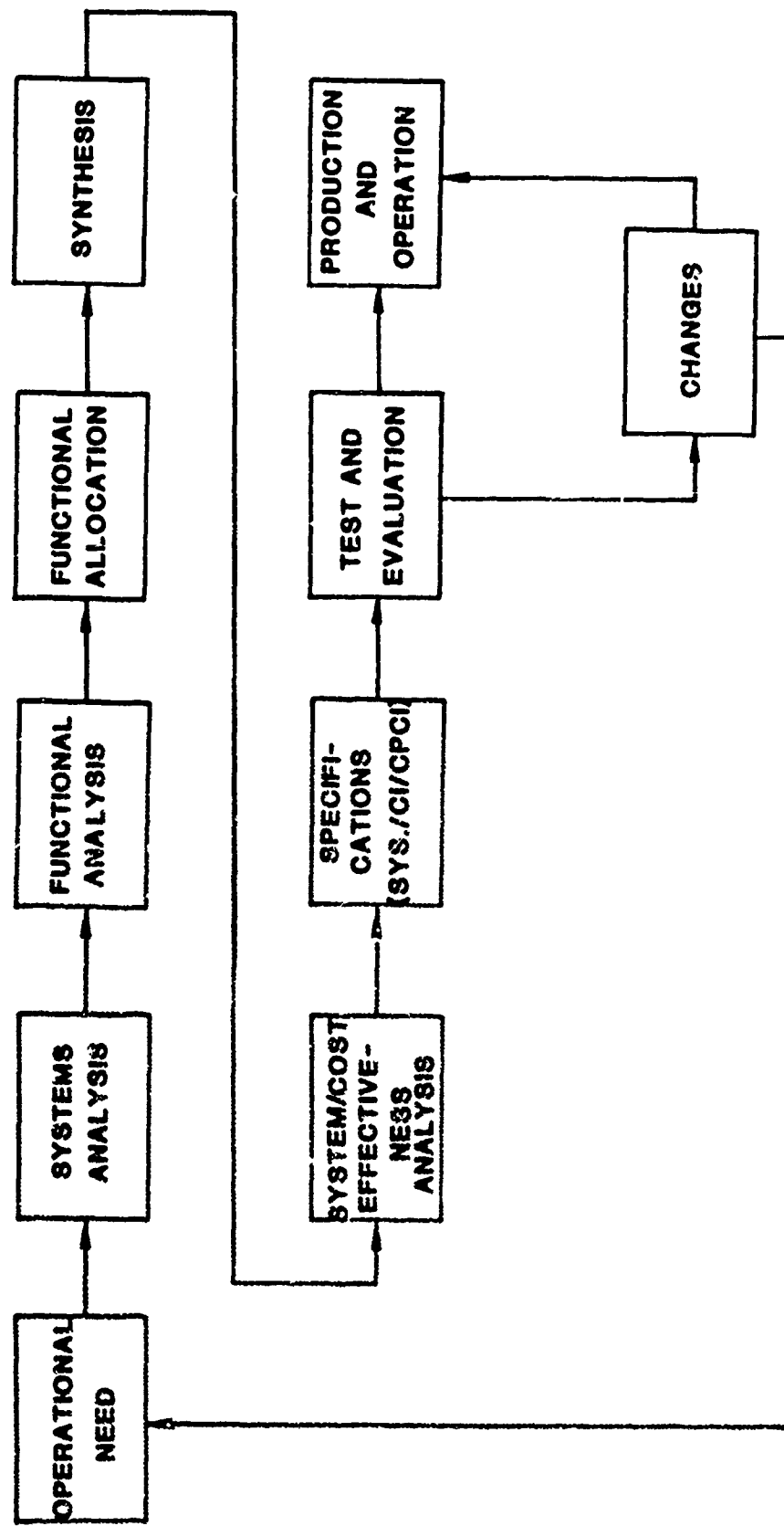
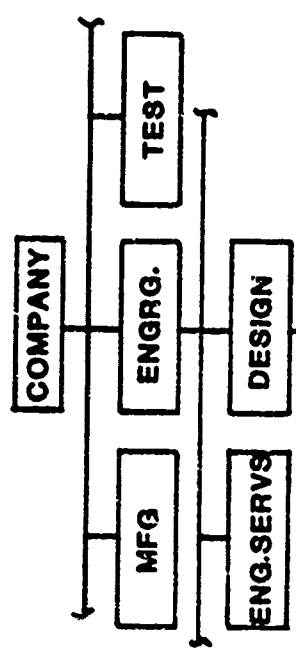


Figure 3.

TYPICAL INTEGRATION OF CWBS AND COMPANY ORGANIZATION (HARDWARE)

FUNCTIONAL ORGANIZATION



CONTRACT WORK BREAKDOWN STRUCTURE

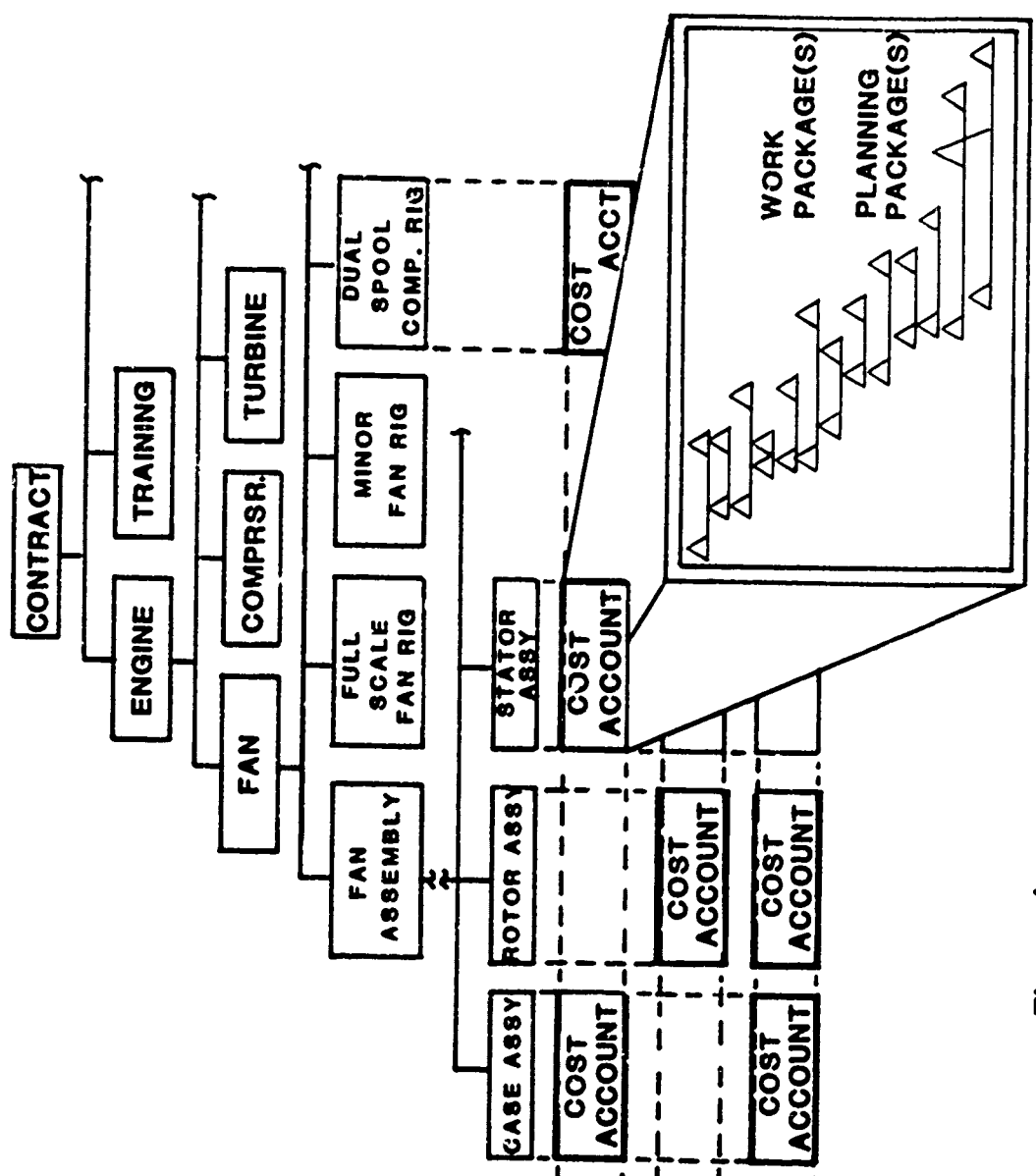


Figure 4.

CI/CPCI GENERAL CONSIDERATIONS

- FUNCTION OF ANTICIPATED DESIGN.
- SATISFIES AN END USE FUNCTION.
- ELEMENTS OF A SYSTEM SEPARATED FOR THE PURPOSE OF MANAGING THEIR DEVELOPMENT.
- DELIVERABLE ENTITY TO WHICH CERTAIN SYSTEM FUNCTIONS HAVE BEEN ALLOCATED.
- DETERMINED BY GOVERNMENT'S NEED TO CONTROL INHERENT CHARACTERISTICS OR INTERFACES.
- SUBSYSTEMS OR SE THAT WILL BE COMMON TO MORE THAN ONE SYSTEM.
- LIMITED TO MAJOR SUBSYSTEM LEVELS OF THE WBS.
- CRITICAL ITEMS OF A LOWER WBS LEVEL.
- SAFETY.
- HIGH DEGREE OF INTERFACE WITH GFE.
- HIGH DEGREE OF FUTURE CHANGE ACTIVITY AFTER BEING PLACED IN OPERATION.
- POTENTIAL USE IN MULTIPLE SYSTEMS
- SCHEDULED FOR DEVELOPMENT, TESTING AND DELIVERY AT DIFFERENT TIMES.

CONFIGURATION MANAGEMENT BY BASELINES

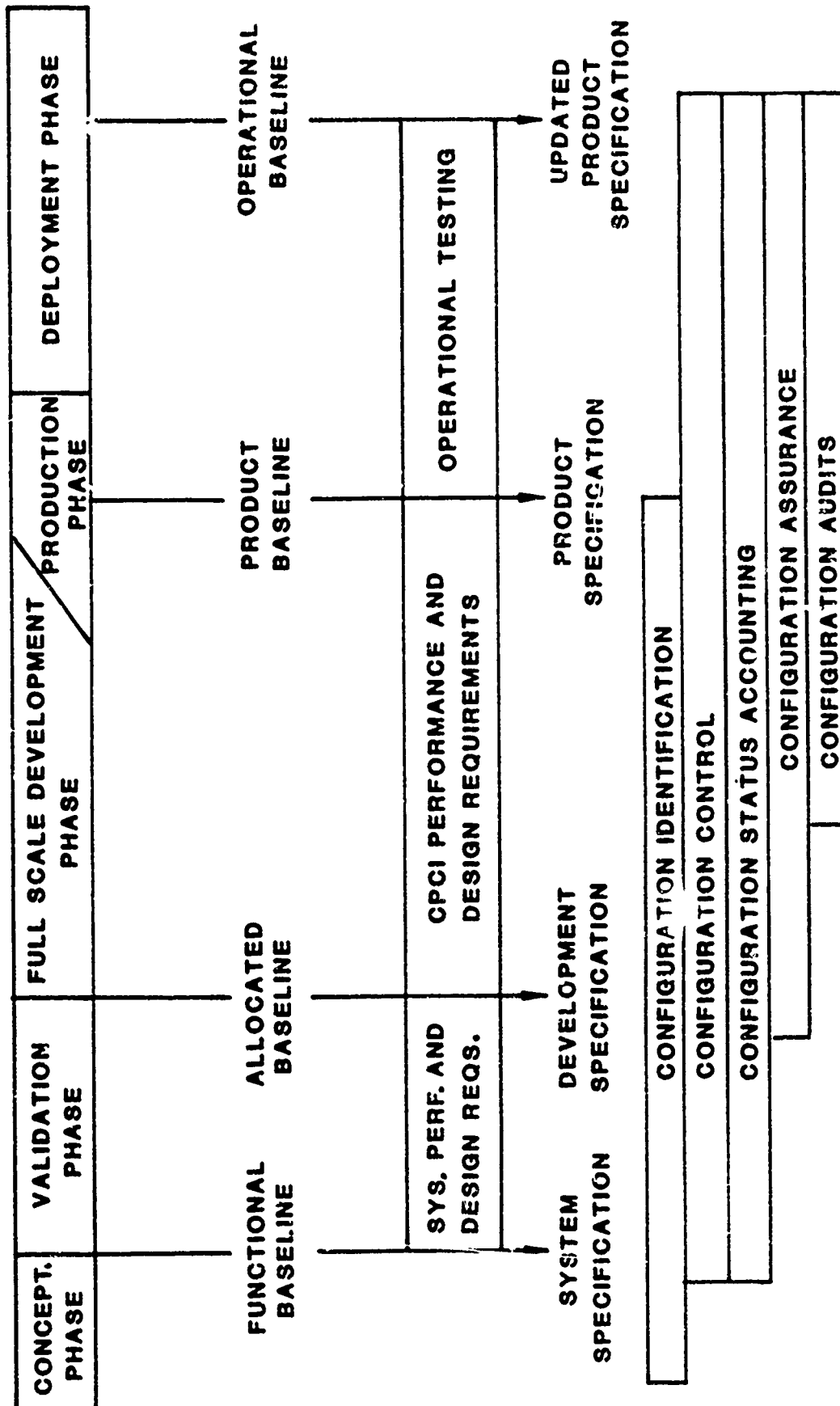


Figure 5.

BASELINE SUPPORT

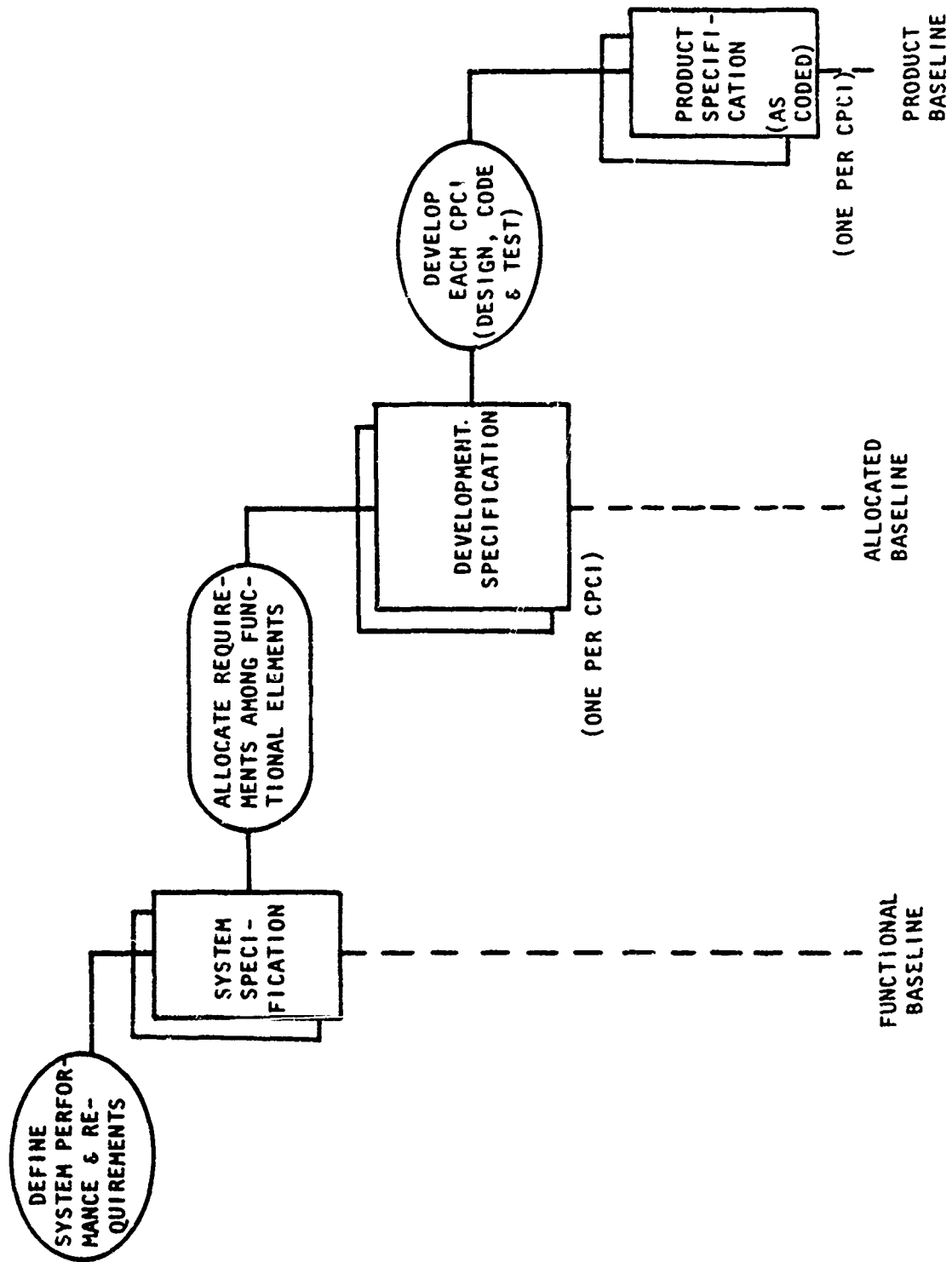
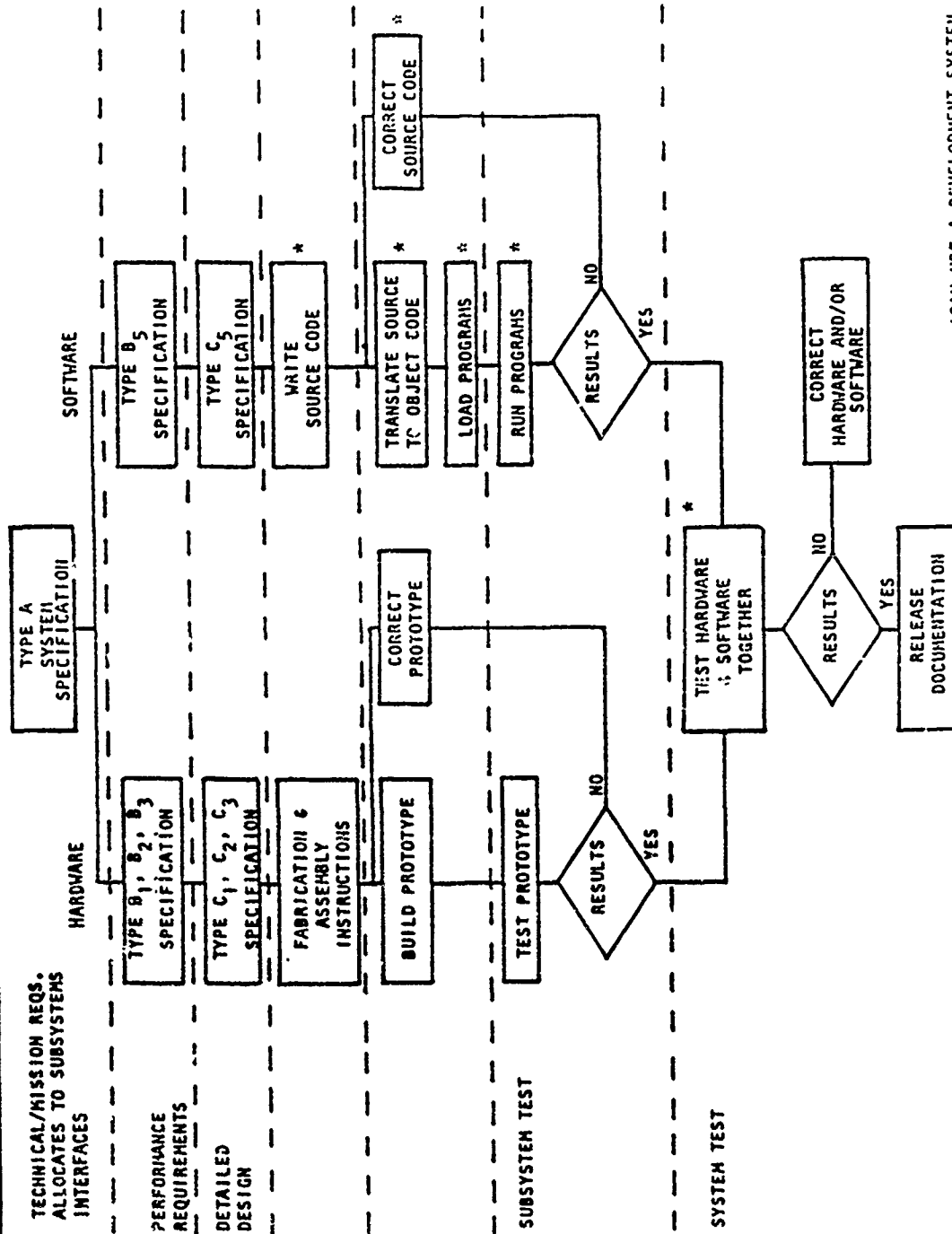


Figure 6.

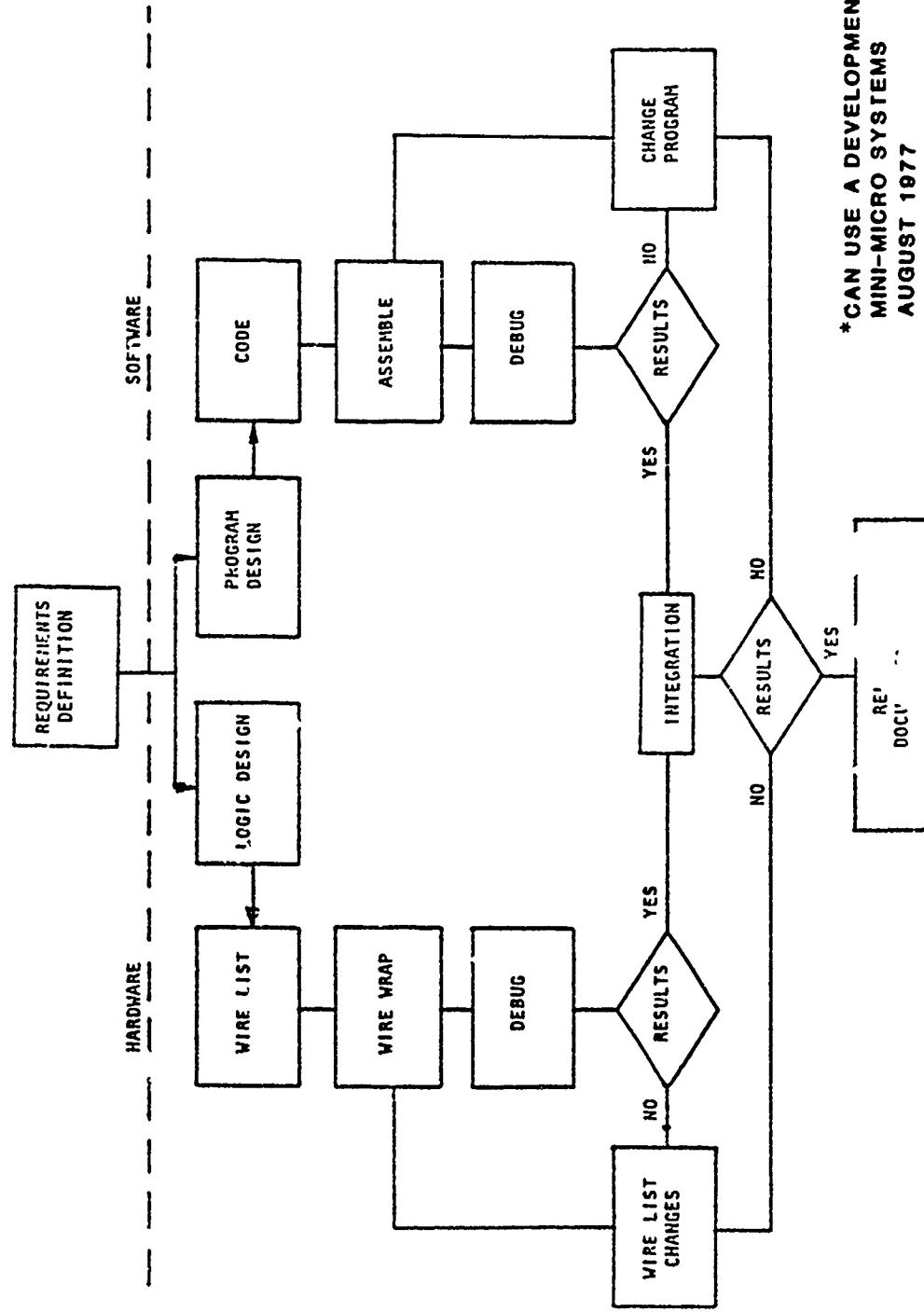
SYSTEM DESIGN PROCESS (MICROCOMPUTER)



*CAN USE A DEVELOPMENT SYSTEM!
 MINI-MICRO SYSTEMS
 AUGUST 1977

Figure 7.

SYSTEM DESIGN PROCESS (MICROPROCESSOR-BASED SYSTEM)



*CAN USE A DEVELOPMENT SYSTEM
MINI-MICRO SYSTEMS
AUGUST 1977

Fig. 1

CPC/CPCI DEVELOPMENT

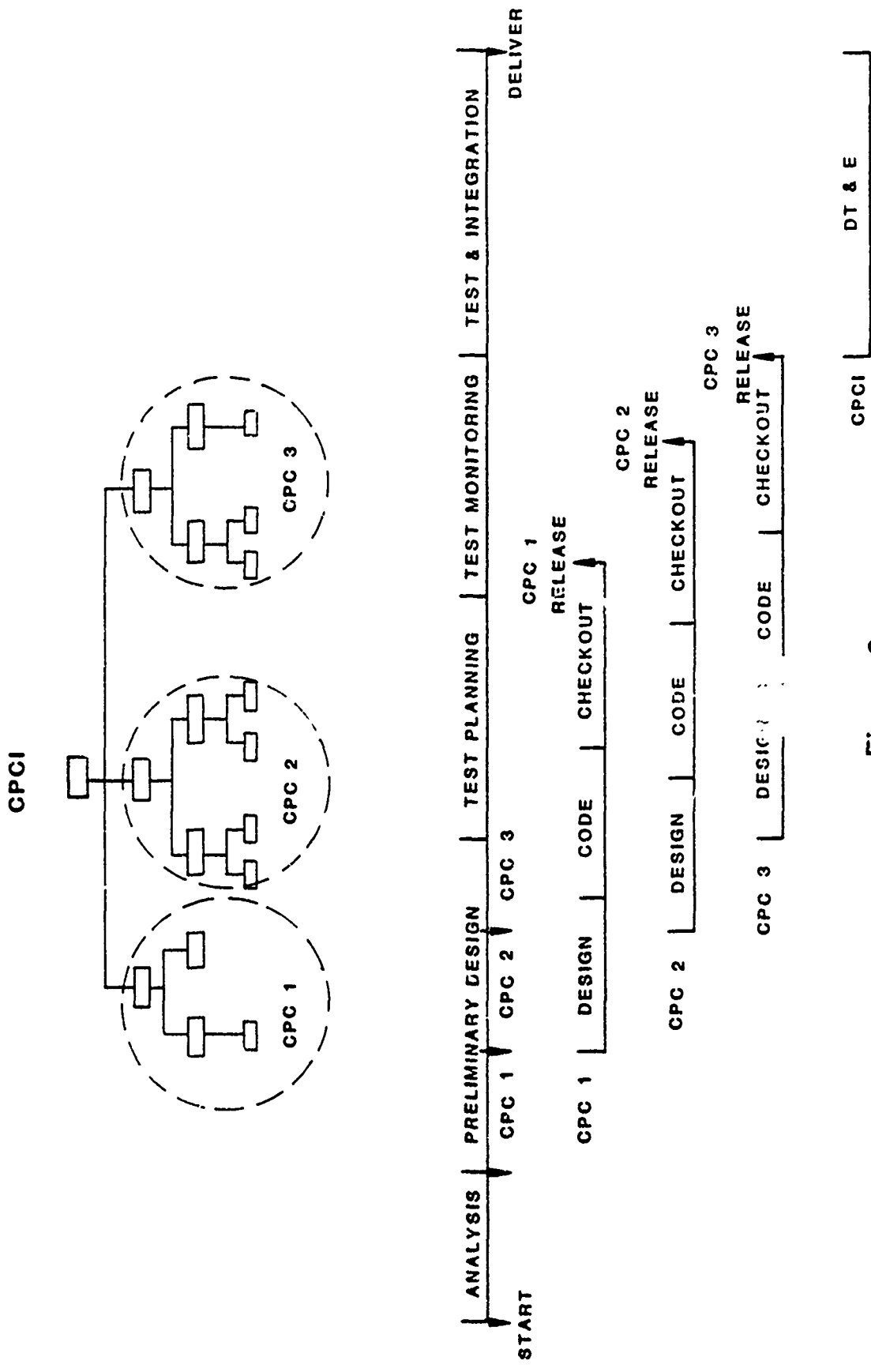


Figure 9.

CPCI's DEVELOPED AS PARTS OF A SYSTEM

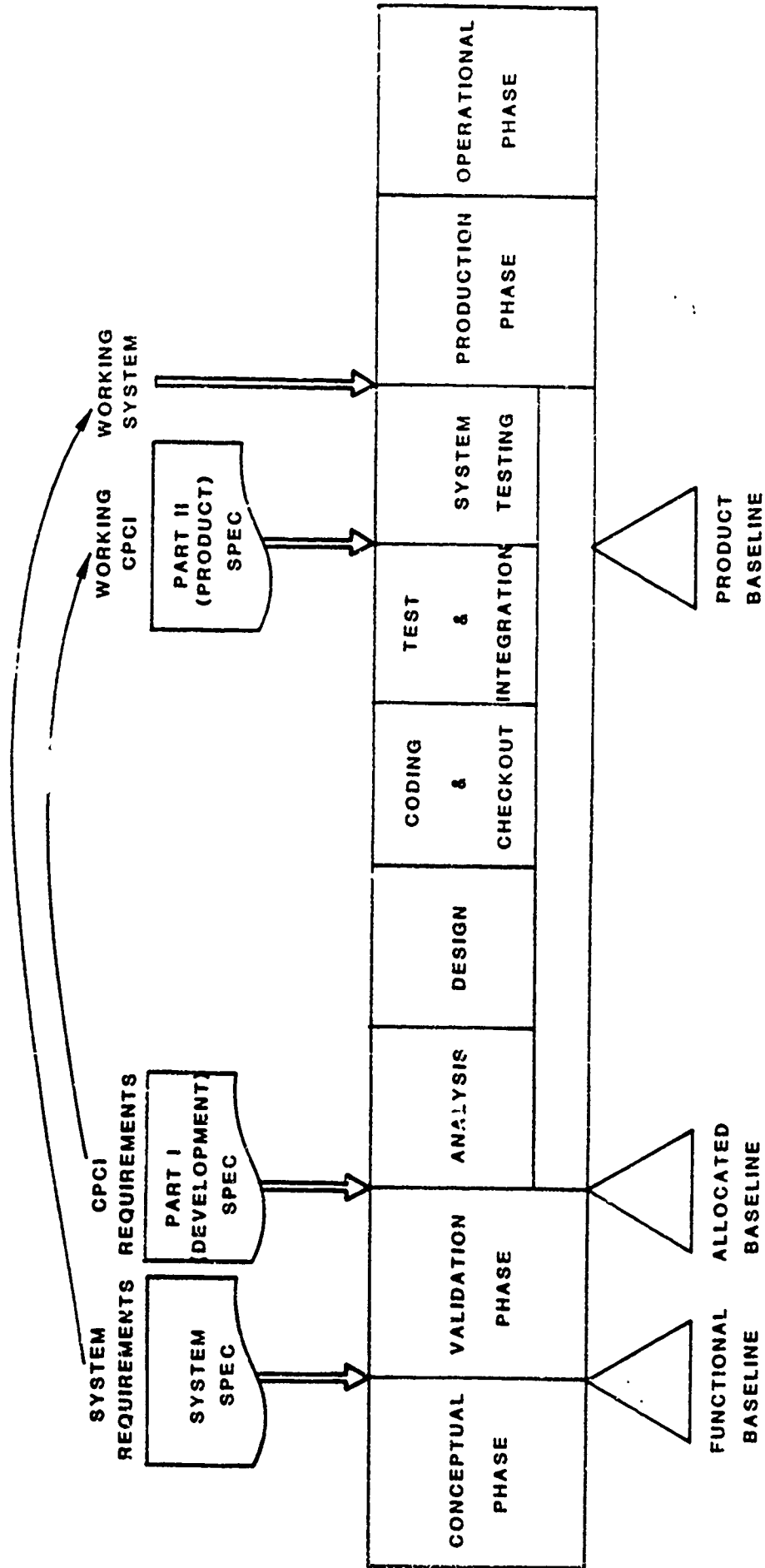


Figure 10.

CPCI'S DEVELOPED INDEPENDENTLY OF A SYSTEM

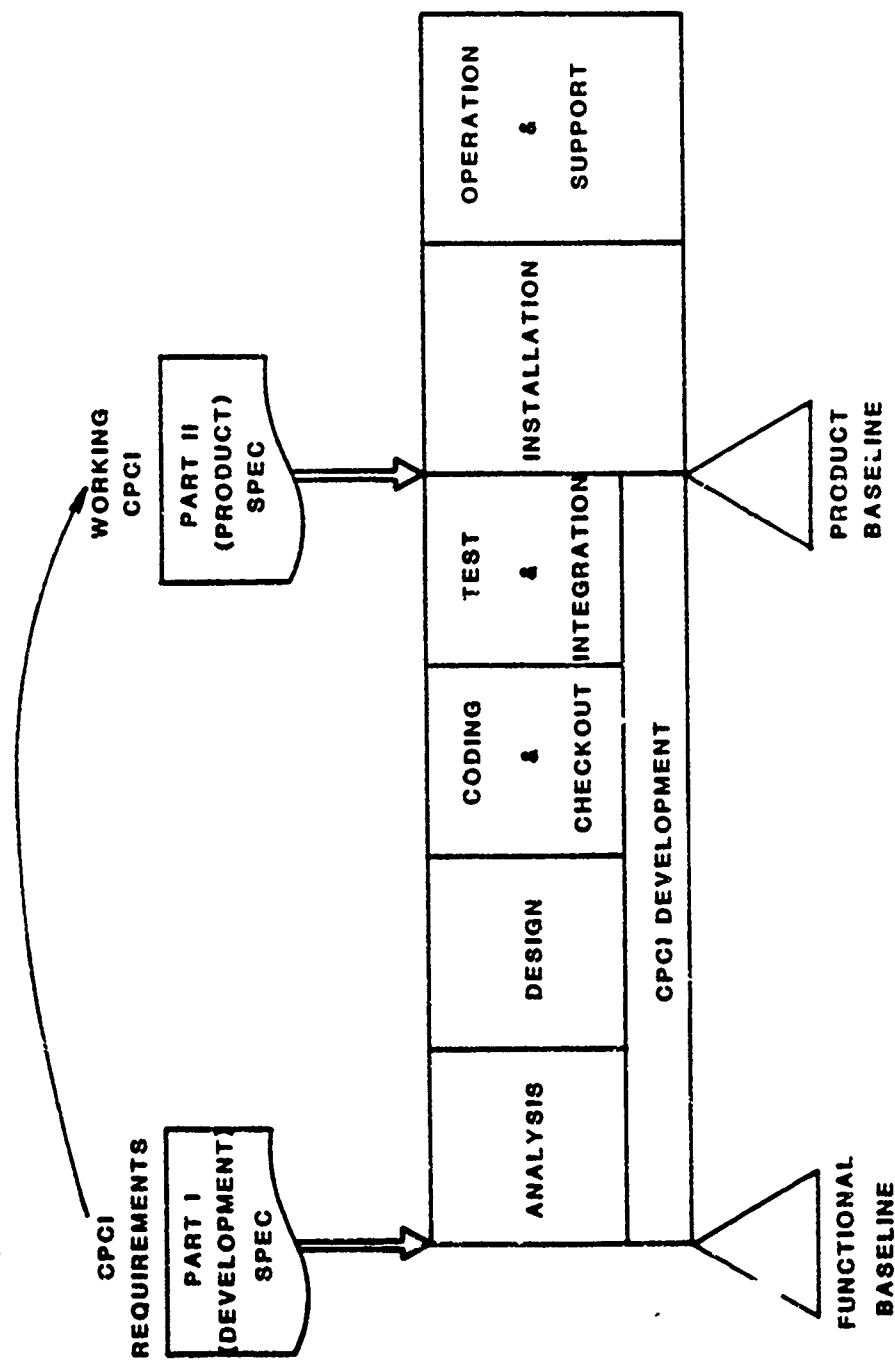


Figure 11.

SPECIFICATIONS

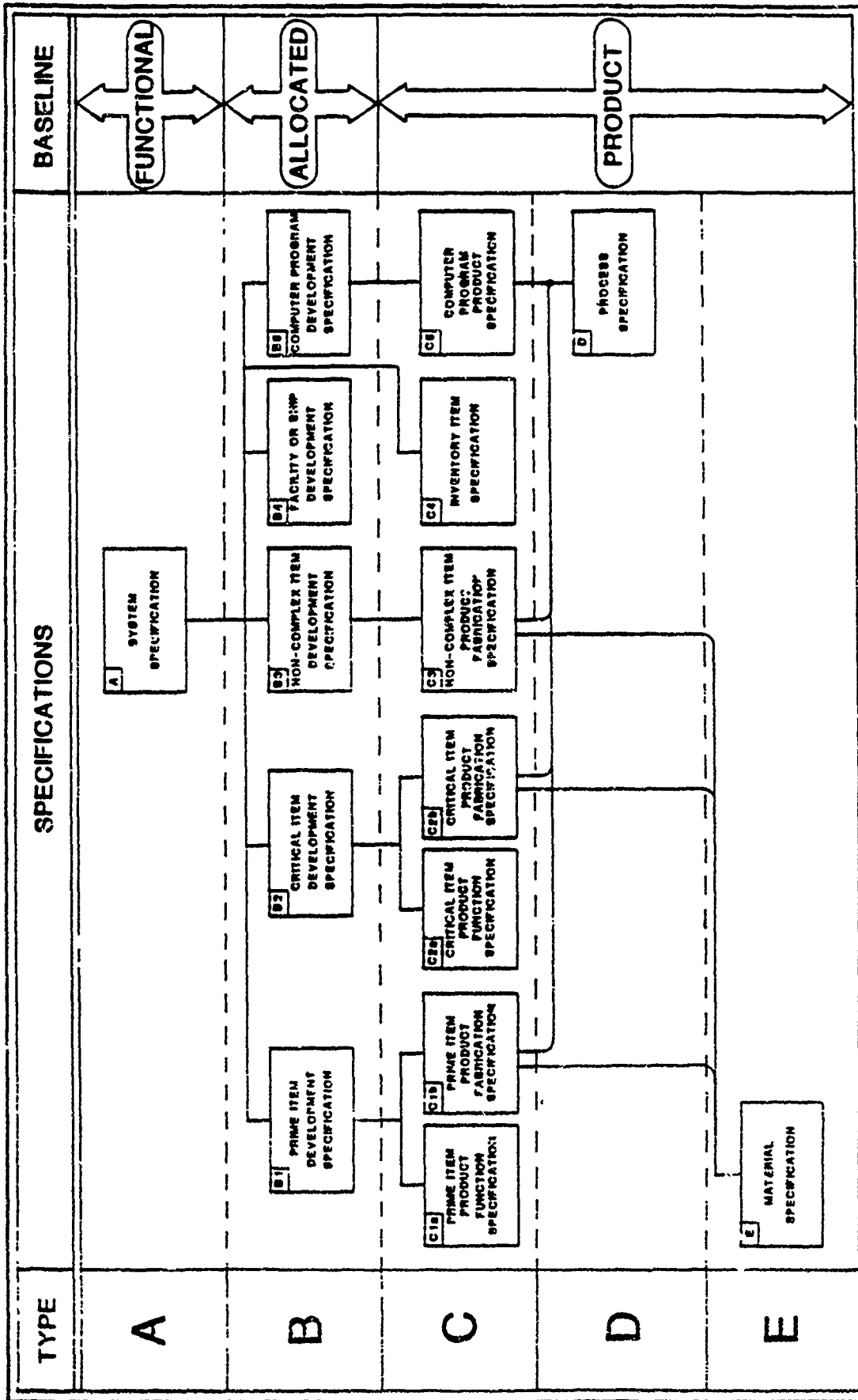


Figure 12.

TYPES

TYPE A -- SYSTEM SPECIFICATION
TYPE B -- DEVELOPMENT SPECIFICATIONS
 TYPE B1 -- PRIME ITEM
 TYPE B2 -- CRITICAL ITEM
 TYPE B3 -- NON COMPLEX ITEM
 TYPE B4 -- FACILITY OR SHIP
 TYPE B5 -- COMPUTER PROGRAM
TYPE C -- PRODUCT SPECIFICATIONS
 TYPE C1a -- PRIME ITEM FUNCTION
 TYPE C1b -- PRIME ITEM FABRICATION
 TYPE C2a -- CRITICAL ITEM FUNCTION
 TYPE C2b -- CRITICAL ITEM FABRICATION
 TYPE C3 -- NON-COMPLEX ITEM FABRICATION
 TYPE C4 -- INVENTORY ITEM
 TYPE C5 -- COMPUTER PROGRAM
TYPE D -- PROCESS SPECIFICATION
TYPE E -- MATERIAL SPECIFICATION

STANDARDIZED FORMAT FOR SPECIFICATIONS

SECTION 1	SCOPE
SECTION 2	APPLICABLE DOCUMENTS
SECTION 3	REQUIREMENTS
SECTION 4	QUALITY ASSURANCE PROVISIONS
SECTION 5	PREPARATION FOR DELIVERY
SECTION 6	NOTES (ADMINISTRATIVE INFORMATION)
SECTION 10	APPENDIX (NARRATIVE AND LISTING SEPARATED FROM MAIN BODY OF SPECIFICATION FOR CON- VENIENCE)

SPECIFICATION RELATIONS

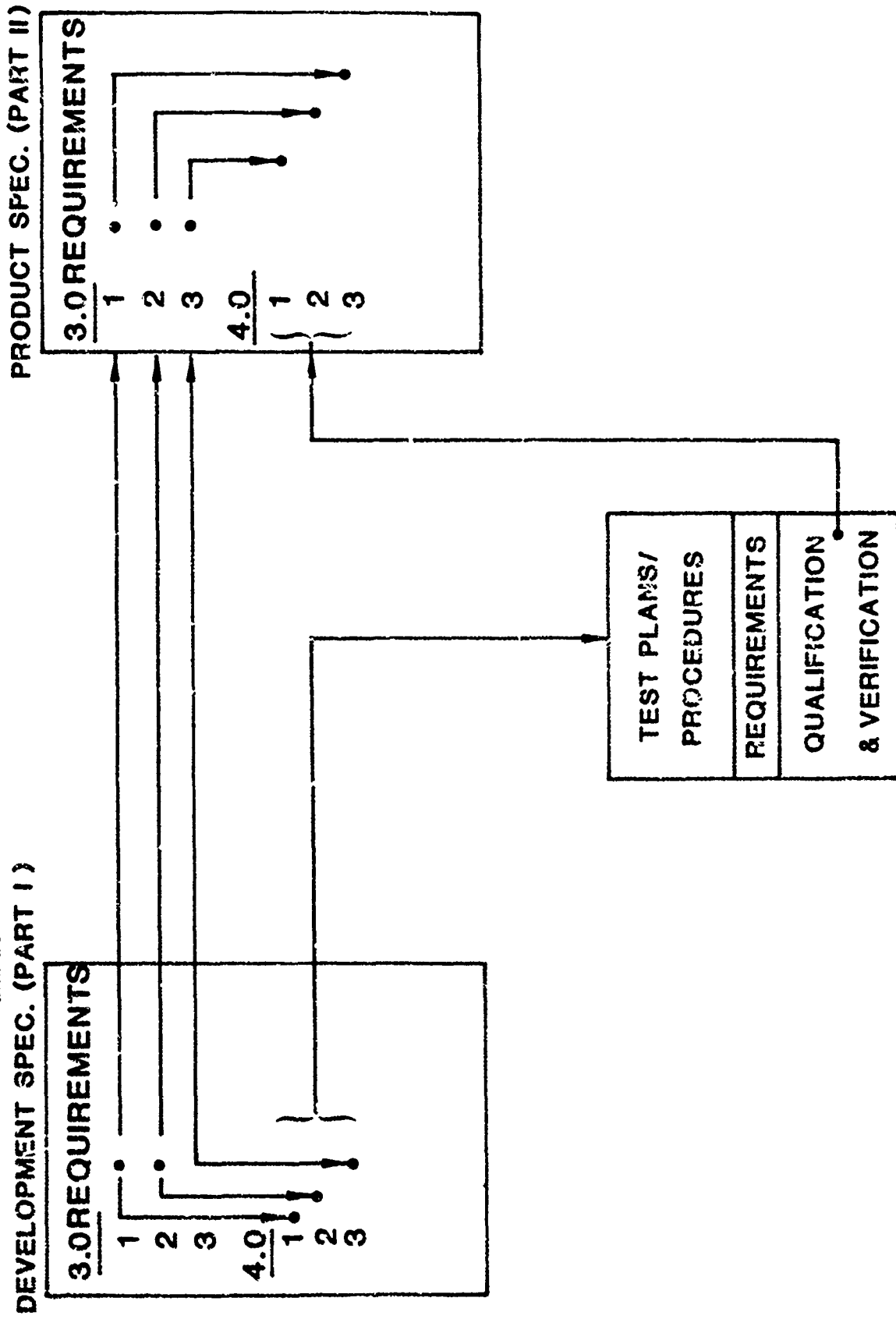


Figure 13

MINIMUM AF DIDS

SYSTEM SPECIFICATIONS	DI-E-3101
SYSTEM SEGMENT SPECIFICATION	DI-E-3117
COMPUTER PROGRAM DEVELOPMENT SPECIFICATION	DI-E-3119
COMPUTER PROGRAM PRODUCT SPECIFICATION	DI-E-3120
ADDENDUM SPECIFICATION	DI-E-3104
SPECIFICATION CHANGE NOTICE	DI-E-3134
TEST PLAN/PROCEDURE	DI-E-3706
USER'S MANUAL	DI-M-3410
TEST REPORT	DI-T-3718
VERSION DESCRIPTION DOCUMENT	DI-E-3121

Table 2

MIL-STD-1679 (NAVY) DOCUMENTATION

A. INTERFACE DESIGN SPECIFICATION (IDS)	DI-E-2135
B. PROGRAM PERFORMANCE SPECIFICATION (PPS)	DI-E-2136
C. PROGRAM DESIGN SPECIFICATION (PDS)	DI-E-2138
D. PROGRAM DESCRIPTION DOCUMENTATION (PDD)	DI-S-2139
E. DATA BASE DESIGN DOCUMENT (DBD)	DI-S-2140
F. PROGRAM PACKAGE DOCUMENT	DI-S-2141
G. COMPUTER PROGRAM TEST PLAN	DI-T-2142
H. COMPUTER PROGRAM TEST SPECIFICATION	DI-E-2143
I. COMPUTER PROGRAM TEST PROCEDURES	DI-T-2144
J. COMPUTER PROGRAM TEST REPORT	DI-T-2156
K. OPERATOR'S MANUAL (OM)	DI-M-2145
L. SYSTEM OPERATORS' s MANUAL (SOM)	DI-M-2148
M. SOFTWARE QUALITY ASSURANCE PLAN	DI-R-2174
N. SOFTWARE CONFIGURATION MANAGEMENT PLAN (SCMP)	DI-E-2175
O. SOFTWARE DEVELOPMENT PLAN	DI-A-2176
P. S / W CHANGE PROP. (SCP) / SOFTWARE ENHANCEMENT PROP. (SEP)	DI-E-2177
Q. COMPUTER SOFTWARE TROUBLE REPORT (STR)	DI-E-2178

Table 3.

TYPES OF SOFTWARE LIBRARIES

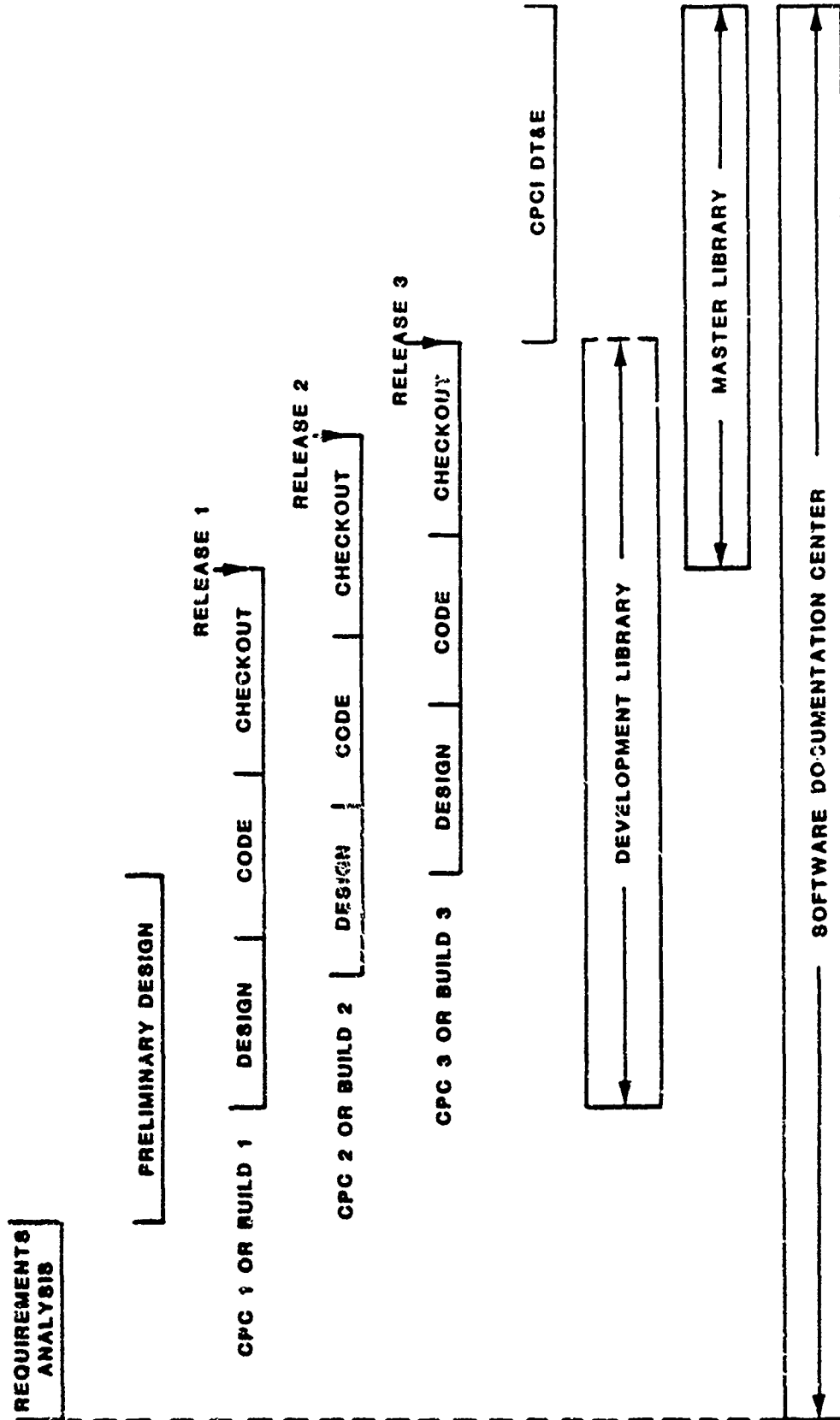


Figure 14.

FOUR BASIC TYPES OF REVIEWS & AUDITS

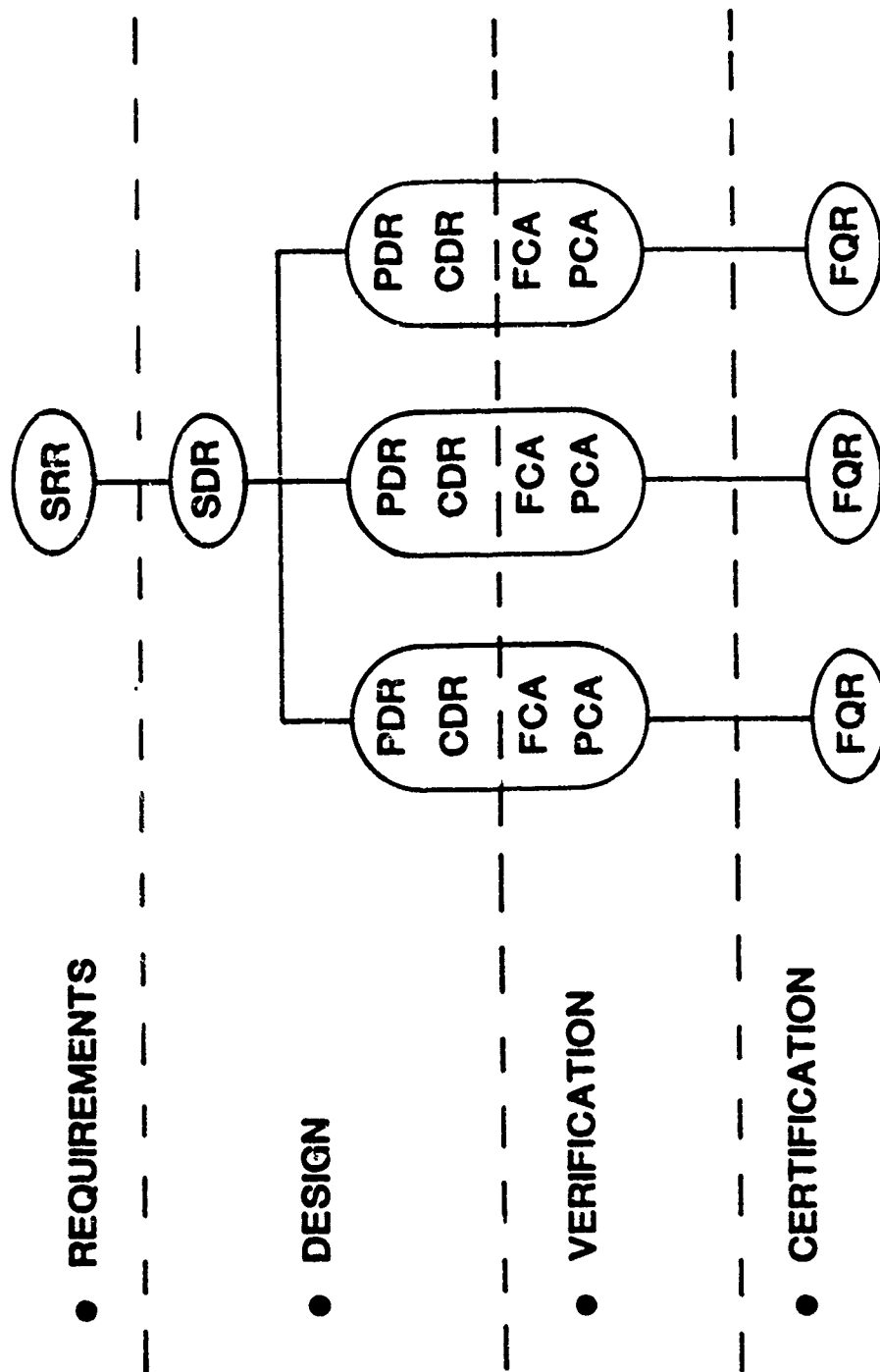


Figure 15.

QUALITY SOFTWARE

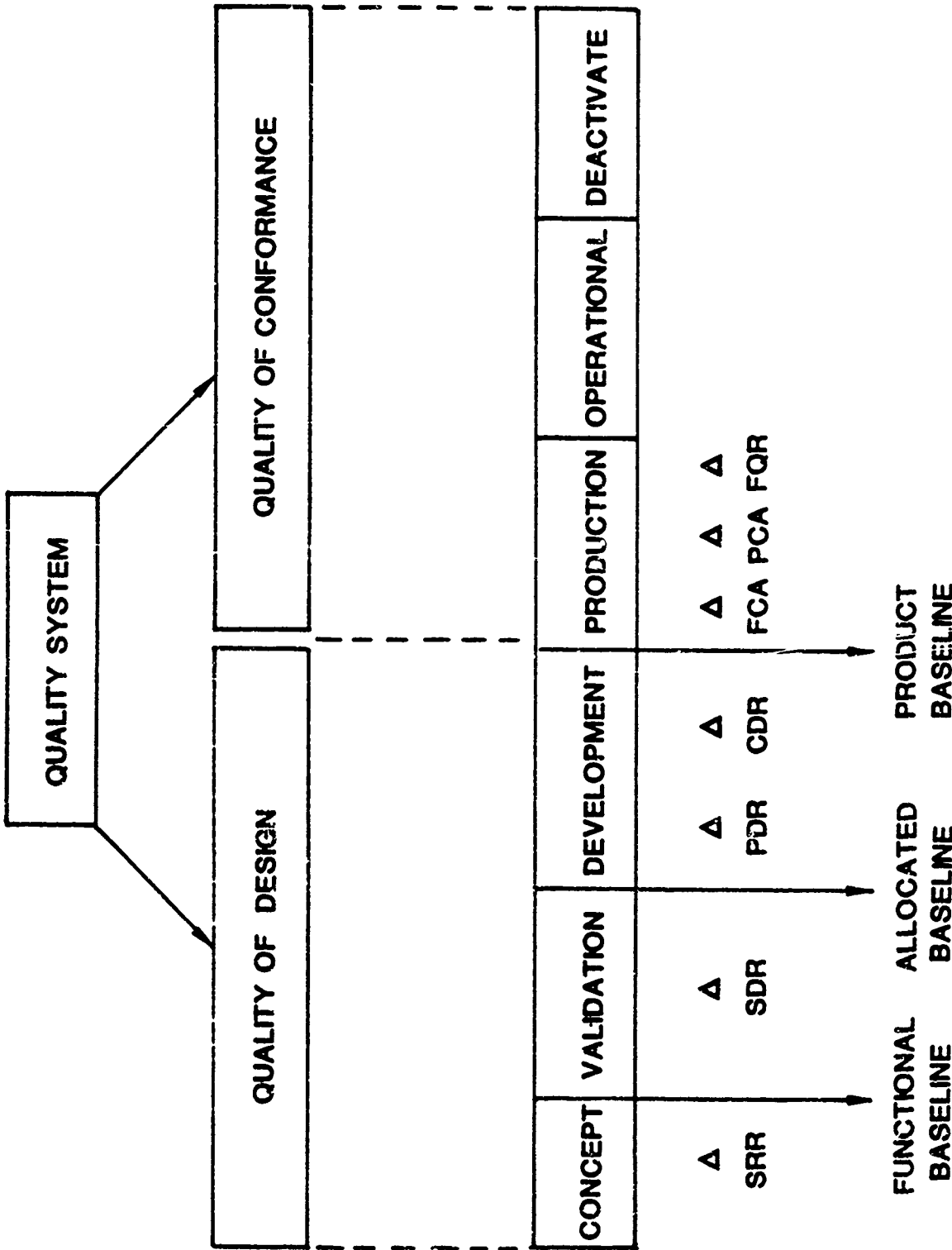


Figure 16.

AN INTEGRATED QUALITY ASSURANCE PROGRAM

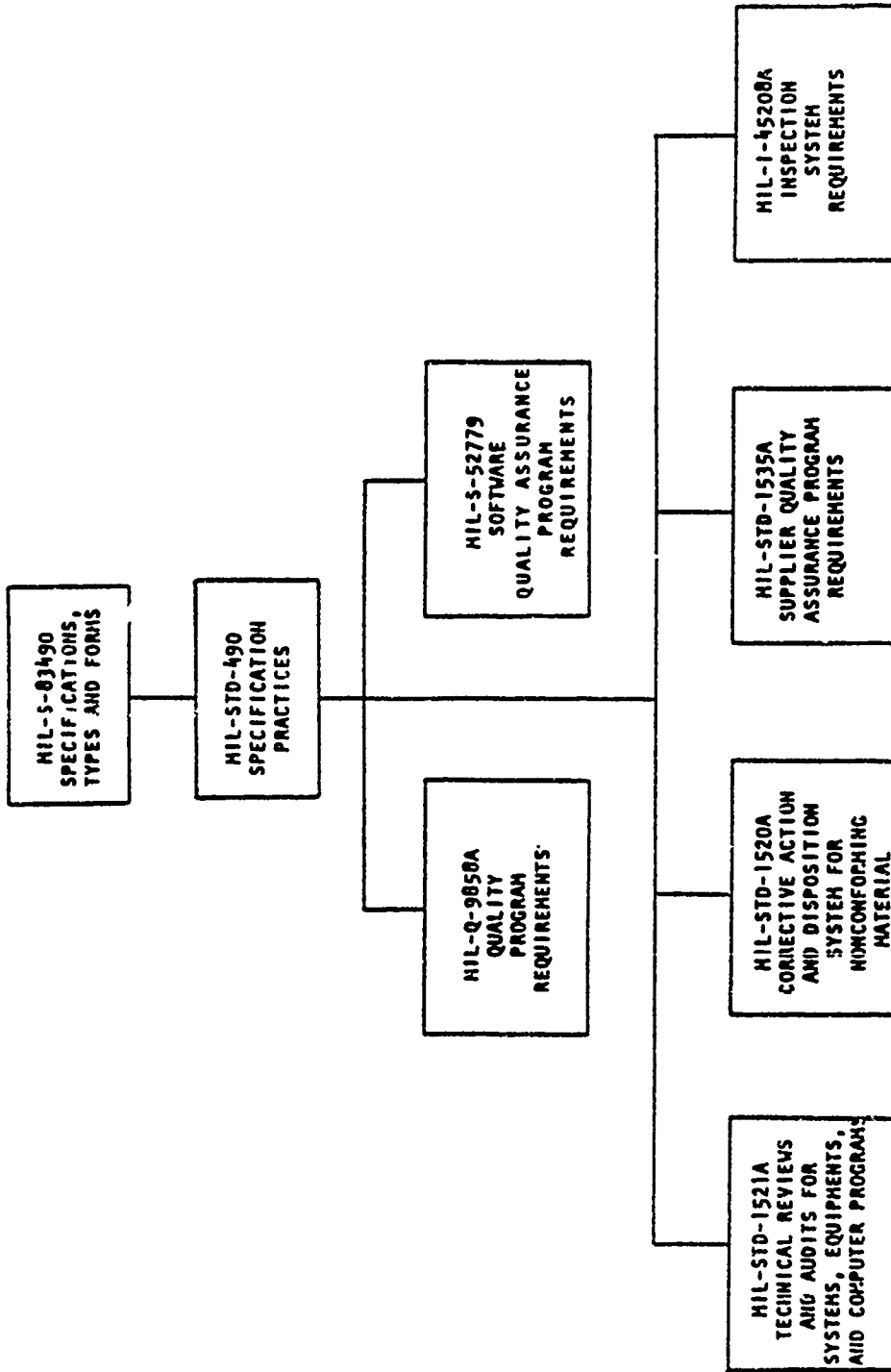


Figure 17.

TESTING STRUCTURE

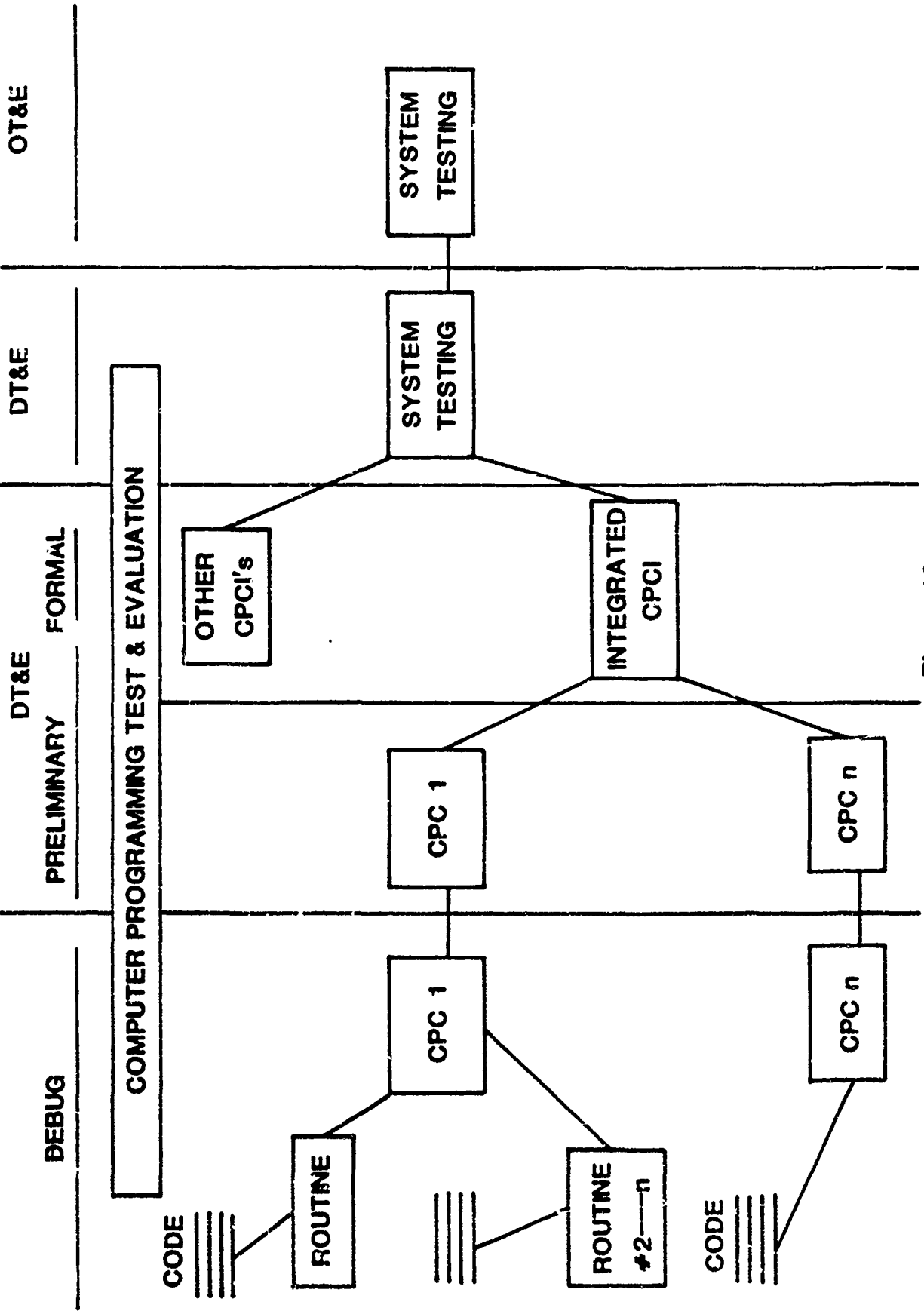


Figure 18

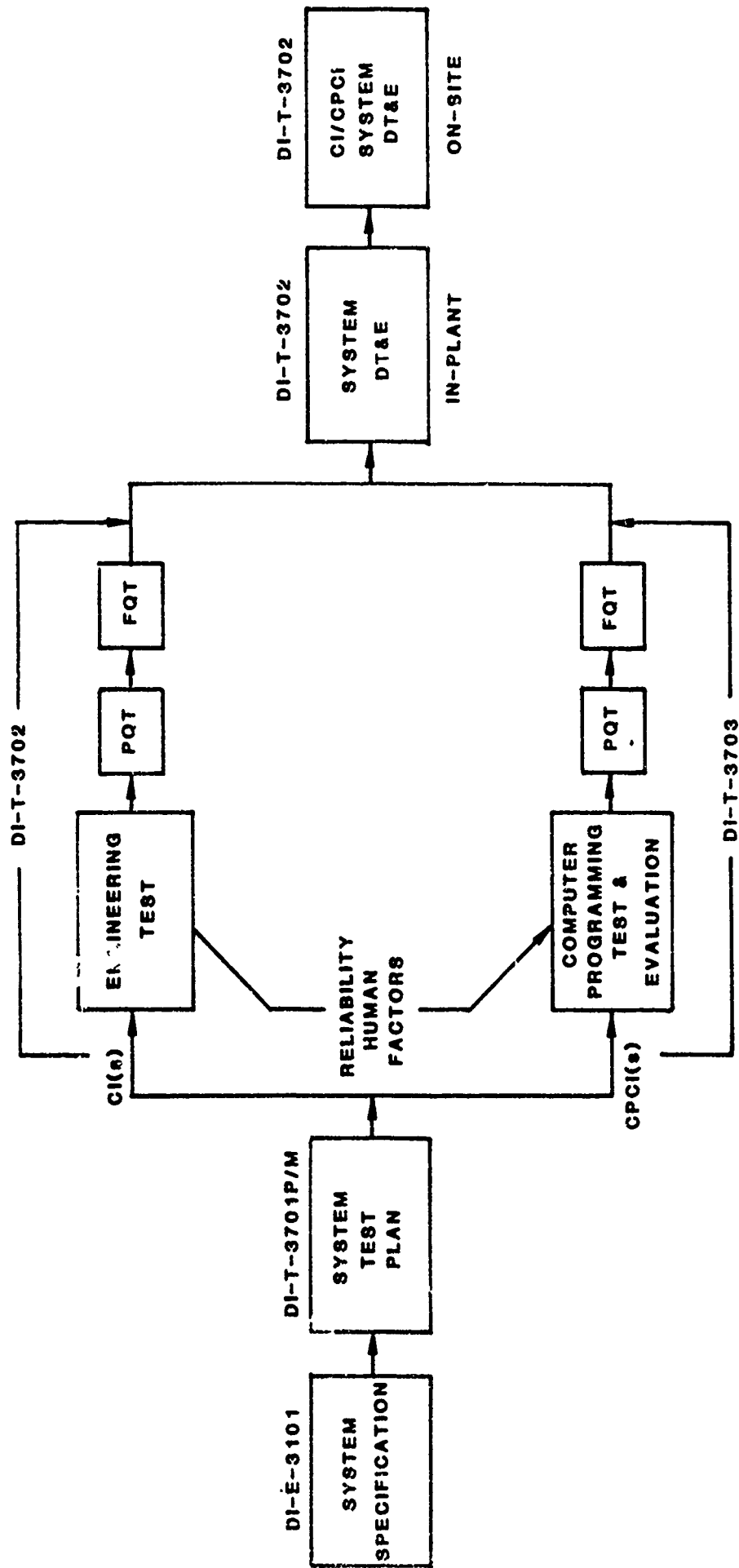


Figure 19

LOGISTICS SUPPORT OF COMPUTER RESOURCES

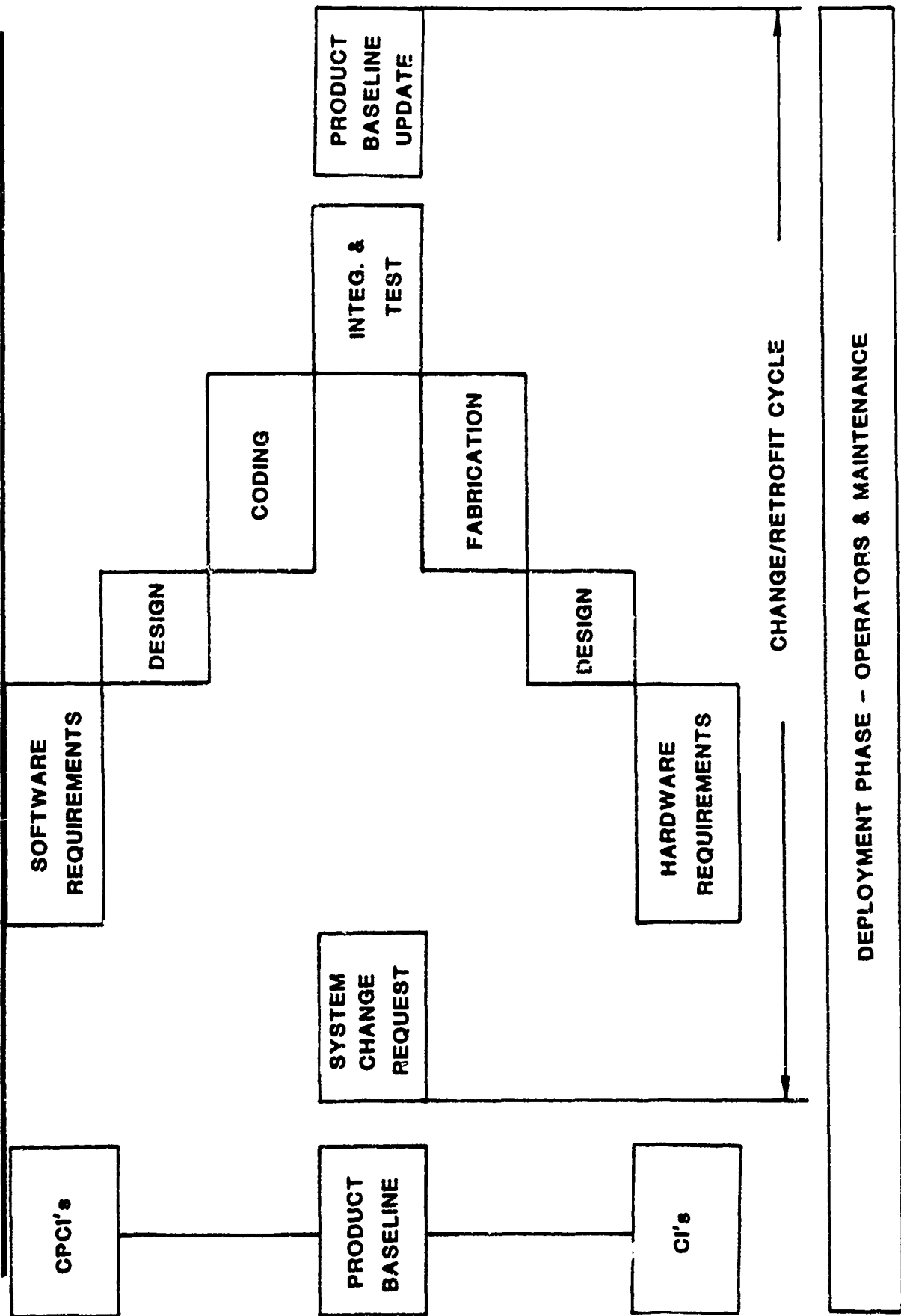
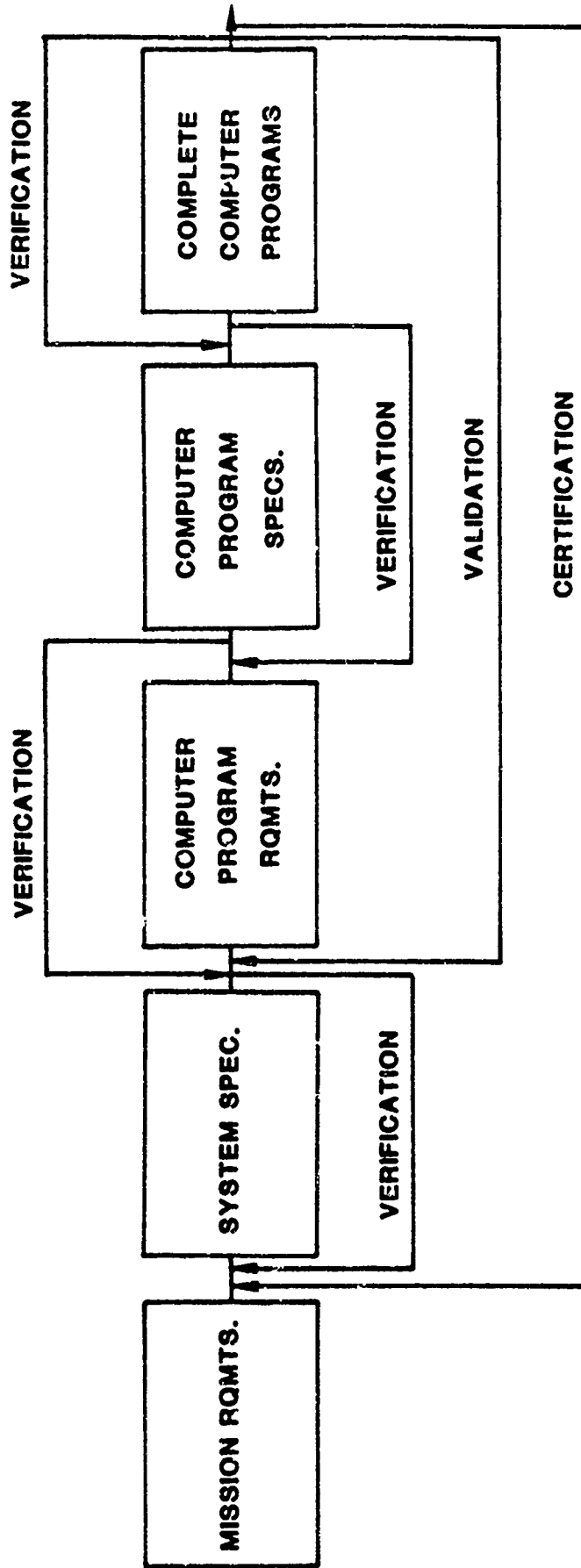


Figure 20

ASSURANCE MODEL, ECS SOFTWARE



A.E. PATTERSON
DEF. SYS. MGMT. COLL.
MAY 1977

Figure 21.

MANAGEMENT TECHNIQUES

MANAGEMENT BY PROCEDURES (MBP) MANAGEMENT BY OBJECTIVES (MBO) MANAGEMENT BY EXCEPTION (MBE)

DETAILED STEP-BY-STEP

INNOVATION TAILORING

PROBLEM SOLVING -

ONE WAY ONLY

WHAT NEEDS TO BE DONE,
WHAT DO WE DO TODAY

PEOPLE

MANAGER

PEOPLE & MANAGER

(STEPS NOT FOLLOWED
WILL DOOM THE PROGRAM)

(CREATIVE, PROVEN, INITIATIVE,
LEADERSHIP & DEDICATION)

(COORDINATED TEAM EFFORT)

BLEND

Figure 22.

REMEMBER THAT THE IMPETUS AND PRIORITIES OF YOUR
PROJECT ARE REALLY REGULATED BY THOSE THINGS THAT PLEASE,
PLACATE, INTRIGUE, GRATIFY, STIMULATE, AND FLATTER THE PEOPLE
WHO INFLUENCE ITS SUCCESS.

POWER ALONE WILL GET YOU NOWHERE.

THERE IS NOTHING YOU CAN'T ACCOMPLISH AS LONG AS YOU
DON'T CARE WHO GETS THE CREDIT.

DON'T BE CASUAL ABOUT YOUR PROJECT. IF YOU FEEL IT IS
IMPORTANT, OTHER PEOPLE WILL SENSE THAT FEELING.

REGULATIONS/STANDARDS/SPECIFICATIONS

MIL-S-83490	SPECIFICATION, TYPES & FORMS
MIL-STD-490	SPECIFICATION PRACTICES
MIL-STD-483, Notice 2	CONFIGURATION MANAGEMENT PRACTICES FOR SYSTEMS, EQUIPMENT, MUNITIONS, & COMPUTER PROGRAMS
MIL-S-62779A	SOFTWARE QUALITY ASSURANCE PROGRAM REQUIREMENTS
MIL-Q-9858A	QUALITY PROGRAM REQUIREMENTS
MIL-STD-499A	SYSTEMS ENGINEERING MANAGEMENT
MIL-STD-881	WORK BREAKDOWN STRUCTURES
DOD-D-1000	DRAWINGS, ENGINEERING, & ASSOCIATED LISTS
DOD-STD-100	ENGINEERING DRAWING PRACTICES
DOD-STD-480A	CONFIGURATION CONTROL - ENGINEERING CHANGES, DEVIATIONS, & WAIVERS
MIL-STD-1521A	TECHNICAL REVIEWS & AUDITS FOR SYSTEMS, EQUIPMENT & COMPUTER PROGRAMS
DCDA-5000.19-L	ACQUISITION MANAGEMENT SYSTEMS & DATA REQUIREMENTS CONTROL TEST
AFR 65-3	CONFIGURATION MANAGEMENT
AFR 800-14 Vol. I	MANAGEMENT OF COMPUTER RESOURCES IN SYSTEMS
AFR 800-14 Vol. II	ACQUISITION & SUPPORT PROCEDURES FOR COMPUTER RESOURCES IN SYSTEMS

Table 4.

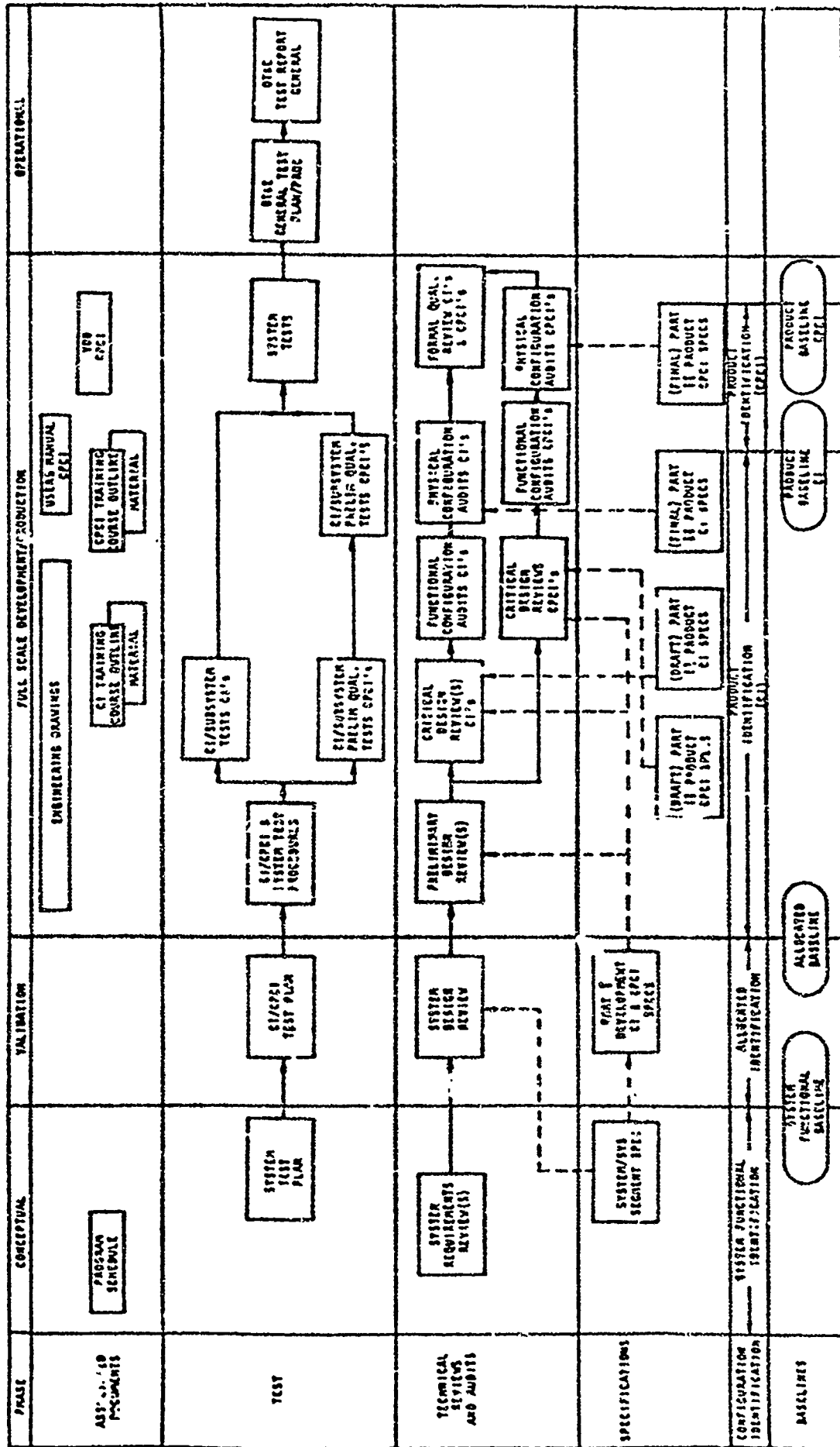


Figure 23.

THIS IS ALL DONE BY

INTEGRATED MANAGEMENT SYSTEMS



Abstract

IS A FEDERAL SOFTWARE ENGINEERING SERIES NEEDED?

There continues to be a growing shortage of software engineering and management personnel within the Federal government, especially within the Naval Material Command (NAVMAT). During the last few years, many NAVMAT in-house study groups have been commissioned to examine the software engineering problem. There are individuals classified as electronics engineers, physicists, computer scientists, computer specialists, general engineers, and mathematicians that are doing "software engineering." Many view the software engineering problem in totally different ways, some see it as one of compensation, some as classification, and others center on the issue being that of qualification standards. The Office of Personnel Management is prepared to conduct an occupational standards study to develop a new classification series for software engineering, with the Navy designated as lead agency. This talk will address the progress by the NAVMAT working group assigned to formulate a Navy position on software engineering, identify problem areas, and develop a plan of action for solving the software engineering issue.

Gwendolyn Hunt
Director, Data Processing Service Center, West
Pacific Missile Test Center
Point Mugu, California 93042

BA	Tennessee A&I	1956
MS	University of Southern California	1976
	Graduate, Industrial College of the Armed Forces	1979

Experience:

26 years of software engineering involvement including:
Pacific Missile Range - Range Software Development
Pacific Missile Test Center - Head, Tactical Software Design
Pacific Missile Test Center - Assoc Director, Systems Technology Dept.

Current Assignment:

Director, Data Processing Service Center West

Responsible for all Automatic Data Processing support for NAVMAT Activities located within the proximity of the Pacific Missile Test Center.

CONCEPTS FOR LHX AVIONICS

LTC Russell H. Smith
US Army Aviation Center
Directorate of Combat Developments
Fort Rucker, Alabama 36362
(205) 255-5902

LHX is the acronym for a family of light, highly capable aircraft intended for operational use in the airland battle well beyond the year 2000. They will be capable of operation in a wide variety of adverse environments on a very hostile battlefield (lasers and other directed energy weapons will be commonplace). Accordingly, the conceptual designs being considered are very different from today's helicopters (fig. 1). One major thrust is toward automation of crew duties, with a goal of achieving single pilot operation.

The cockpit concept for LHX is to mechanize the design so that intelligent systems either operate automatically or in response to simple voice commands or minimal manual inputs to controls grouped within the envelope of the pilot's seat. The key feature in the design will be a large field of view panoramic display of the outside scene (fig. 2) (no, the pilot does not look outside) which will be automatically annotated with aircraft system and battlefield information. The pilot will be able to overlay magnified views of selected terrain and digital map/navigation information or aircraft systems monitoring information. This display will contain all the information necessary to fly the helicopter, acquire and identify targets, and monitor onboard systems. With sensor fusion (coupling of infrared, TV, and millimeter wave radar into one composite scene) and image generation, the display could remain undistorted for operation in any atmospheric environment within the capabilities of the aircraft to fly. To further reduce pilot workload, a new design for flight controls is being developed.

The approach being taken for LHX flight controls is to utilize a four-axis side arm controller combined with a very sophisticated automatic pilot. Positioned on the arm rest, the flight control will have less than $\frac{1}{2}$ inch of full travel in any direction. Pilot input for forward, rear, and turning flight will be as with today's cyclic control. What is today's collective input will be a vertical input to the side arm controller. Hovering pedal turns will be generated by a twisting motion of the controller. The automatic pilot will fly a predetermined flight path at any given altitude above the terrain, allowing the pilot to devote his attention to fire control, battlefield management, or other tasks.

To explain the conceptual designs, let's look at a typical mission using the proposed systems.

At the start, the pilot will receive a small cassette which has been programed with operational information including enemy and friendly situation, map of the operational area with navigation waypoints,



FIG 1

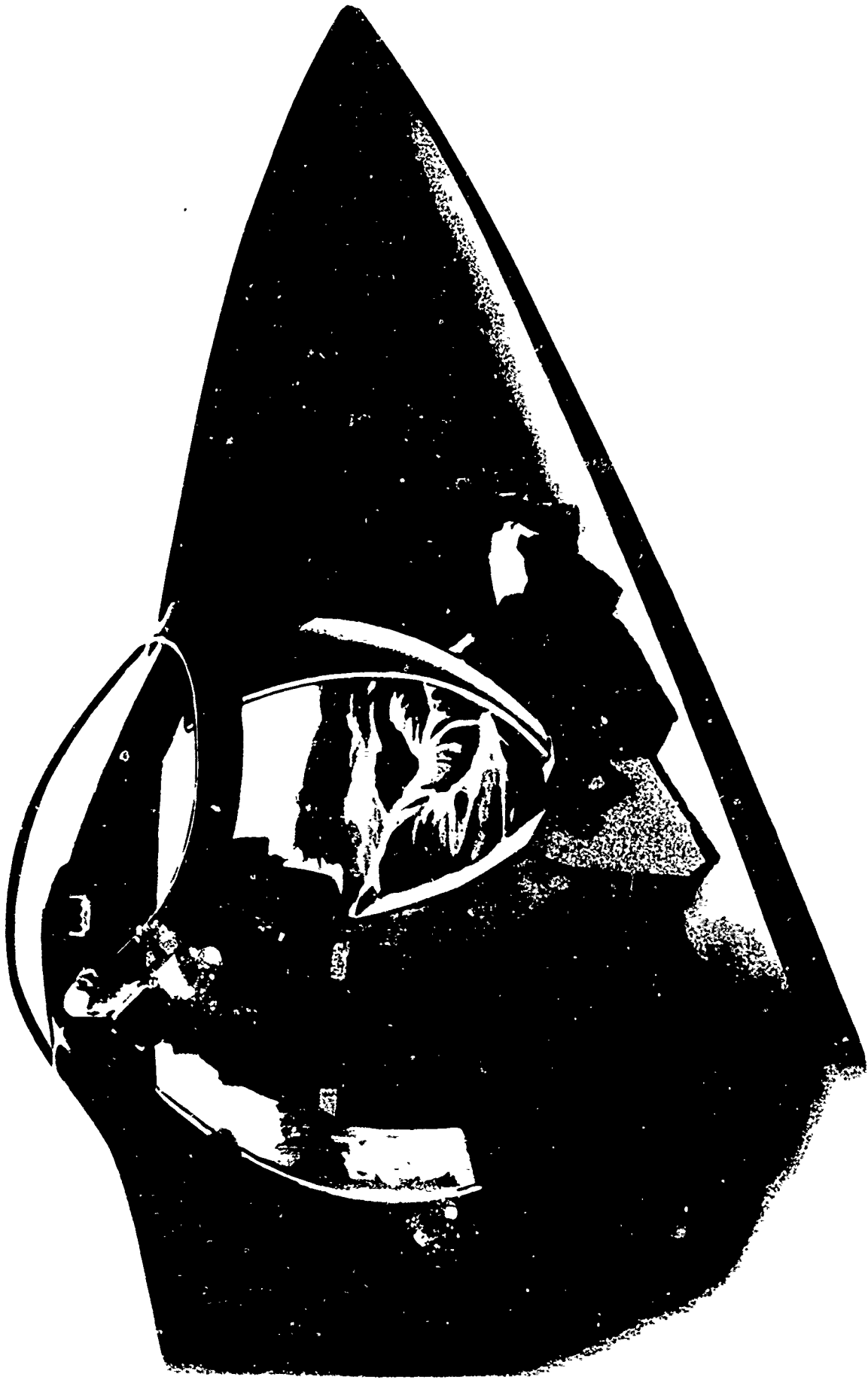


FIG 2

communications frequencies and codes, and the pilot's voice template. The pilot will insert the cassette in the cockpit console which will activate the voice interactive and keyboard systems. The pilot can then command all systems to automatically energize and check for malfunctions. The aircraft engine will be started through voice or keyboard command and operational checklists will be called up for display if necessary. The maintenance management system will record system operational parameters for later recall by maintenance personnel. Pilot cautions and warnings will be automatically displayed, but normal operational information will be displayed only on command. Control of operating modes for fire control, communications, and automatic flight controls would be effected by using either the voice or keyboard systems.

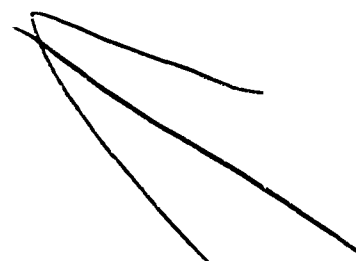
The navigation system coupled to the autopilot will automatically fly the aircraft over the predetermined flight path. Once in the battle area, the LHX could be commanded to automatically hover while contact was made with supported units and intelligence sources. The pilot could fly to a target engagement area and command the aircraft to automatically unmask, acquire targets and remask. The onboard computer will automatically select and prioritize the threat weapons for engagement and display the information for the pilot. Depending on the mission, the pilot could either engage the targets or transmit the data to other members of the combined arms team. By virtue of its low visibility design and terrain masking, the LHX could operate nearly covertly while penetrating well into enemy-held terrain. One other factor still remains to be considered, the maintenance of the sophisticated systems.

The LHX concept is being formulated with maximum pilot and maintenance technician workload reduction as key elements in the system's architecture. In order to achieve the high dependability necessary for sustained tactical operations, a robust avionics architecture is being developed. Highly reliable, very high speed, integrated processors (10,000 hours mean time between failure), employing self-healing electronic rerouting of data through multiple processors, will be used throughout the LHX. Functional system backup will be provided for mission-critical systems through the use of electronically reconfigurable system arrangements without the high cost and weight of traditional duplicative redundancy. The avionics system may make 100-hour and longer continuous developments with no electrical component maintenance confidently achievable. Simple removal and plug-in replacement of components is envisioned with nonoperable components being discarded instead of repaired. These designs will help meet the needs for reduced manpower to fight and win on the battlefields of the future.

Some of you are thinking, "This guy watched 'Star Wars' too many times." Let me assure you that all of the capabilities are possible by the late 1980's. As part of the LHX concept design, the US Army Aviation Research and Development Command (AVRADCOM) has contracted with Boeing-Vertol to investigate the one-man cockpit concept. As part of the program, Boeing has built a simulator which has the side arm controller and panoramic view display using televisions. US Army pilots from field units have flown the simulator through typical nap-of-the-earth (NOE) flight profiles and praised the handling qualities of the side arm controller and automatic flight control features. They claim the panoramic view is far easier to fly than using night vision goggles. The study is to be published in the first quarter of FY83.

Of course, a great deal of development is still required, but we are planning on engineering models of the LHX to be flying before 1990.

We are developing the tactical concepts for the future in the form of Airland Battle 2000. In order to employ the tactics, we must begin now to develop the technical concepts that will insure our aircraft are not only lethal but able to survive in future battles. Old aircraft designs have served us well, but they will not meet the needs of the airland battle of tomorrow.



STANDARDIZATION ISSUES OF THE FUTURE

SESSION CHAIRMAN: Darlow G. Botha
AFWAL/AAAI

MODERATOR: Frank A. Scarpino
AFWAL/AAA

MIL-PRIME PROGRAM SYSTEM

Frederick T. Rall, Jr.

ASD/EN

Wright-Patterson AFB, Ohio

BIOGRAPHY

Wright-Patterson Air Force Base, Ohio -- Mr. Frederick T. Rall, Jr., is the Technical Director, Deputy for Engineering, Aeronautical Systems Division (ASD). He was formerly Chief Engineer on the F-15 Program.

A Forest Park, Illinois native, Mr. Rall graduated from the Proviso Township High School, Maywood, Illinois, in 1946. He earned a BS Degree in Aeronautical Engineering from the Massachusetts Institute of Technology in 1950 and an MS Degree in Aeronautical Engineering from the California Institute of Technology in 1951. An MS Degree in Industrial Engineering from the Massachusetts Institute of Technology was earned in 1964.

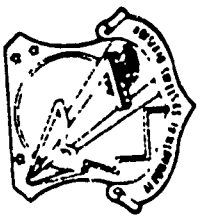
Mr. Rall's engineering assignments have included Project Engineer for the Cooperative Wind Tunnel, Pasadena, California, and an Aerodynamicist with the Douglas Aircraft Company, El Segundo, California. His USAF assignments have included duty as a first lieutenant, Aerodynamics Branch, Aircraft Laboratory, ASD; Chief, Internal Aerodynamics Unit, Aircraft Laboratory, ASD; Chief, Aerodynamics Branch, B-70 Engineering Office, ASD; Chief, Aerodynamics Division, Deputy for F-111, ASD; Chief, Airframe Division, Deputy for F-15/JEPO, ASD; and System Engineering Director, Deputy for F-15/JEPO, ASD.

Among Mr. Rall's awards are the Exceptional Civilian Service Award, the DOD Distinguished Civilian Service Award, the first American Institute of Aeronautics and Astronautics Air Breathing Propulsion Award, the Air Force Association Citation of Honor, and the Presidential Rank Awards of Meritorious and Distinguished Executive. He is a member of the Air Force Association, the American Defense Preparedness Association, and the Association of Old Crows, and is an Associate Fellow in the American Institute of Aeronautics and Astronautics.

Mr. Rall was born 16 September 1928 at Oak Park, Illinois. He is married to the former Peggy McDonald of Springfield, Ohio. They have six children and reside at 7960 South Oak Court, Centerville, Ohio. Mr. Rall is the son of Mr. Frederick T. Rall, Sr., 6451 Far Hills Ave, Dayton, Ohio.

ABSTRACT

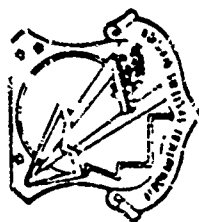
The MIL-PRIME Program was developed to provide the Aeronautical Systems Division with a specifications and standards program for the development of new weapon systems. This program was started in 1976 and is still in the process of being implemented. This presentation provides the background and insight into the MIL PRIME System, the use of these new documents in the acquisition process, and the current status of the program.



MIL-PRIME PROGRAM SYSTEM

MILITARY SPECIFICATIONS AND STANDARDS

AERONAUTICAL SYSTEMS DIVISION



MILITARY SPECIFICATIONS/STANDARDS

- 14,000 ACTIVE DODISS DOCUMENTS
- 40,000 REPROCUREMENT
- 4,000 DEVELOPMENT OF NEW EQUIPMENT DOCUMENTS
- ✓ • 500 KEY AERONAUTICAL DEVELOPMENT DOCUMENTS



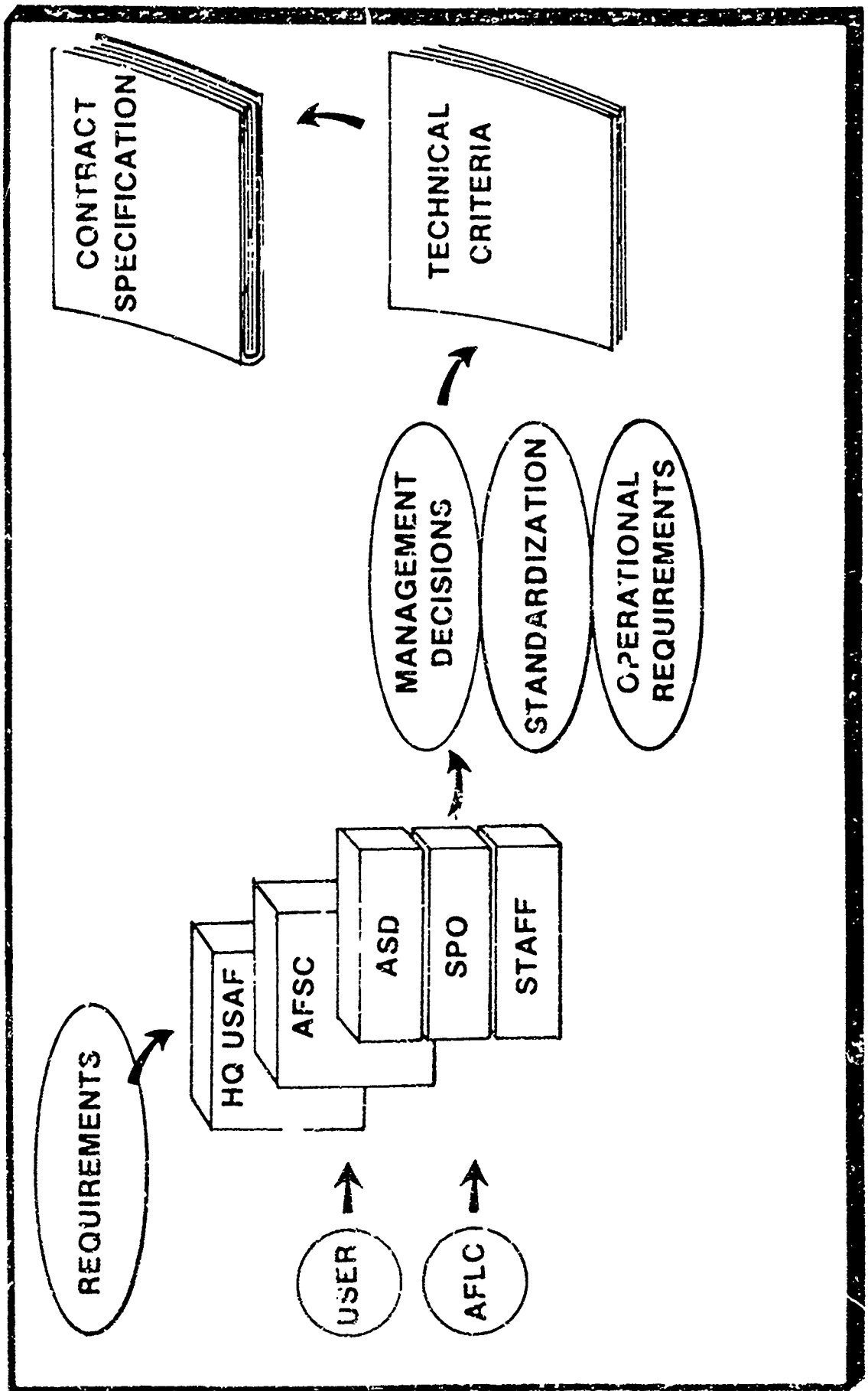
DEVELOPMENT MILITARY SPECIFICATIONS/STANDARDS/HANDBOOKS

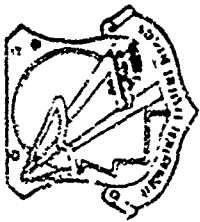
OBJECTIVES

- TECHNICAL DIRECTION
- BASELINES
- LESSONS LEARNED
- DEFINE
 - NEEDS
 - TESTS
 - CRITERIA
- CONSISTENCY & EXPERTISE
BETWEEN PROGRAMS



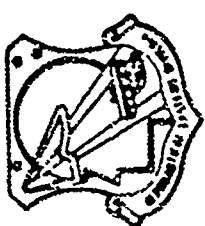
REQUIREMENTS/DIRECTION FLOW



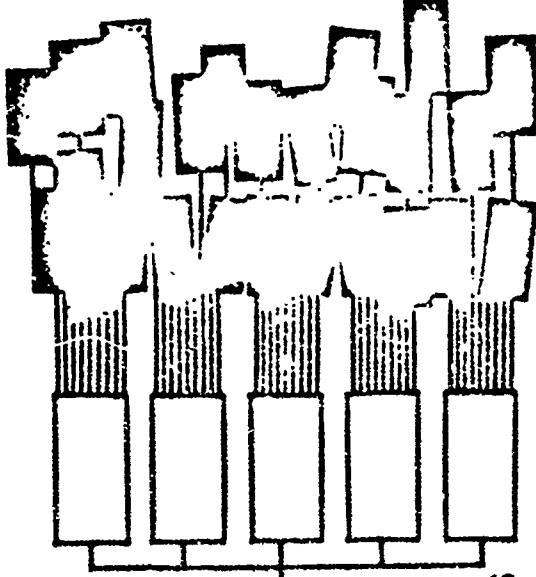


DEVELOPMENT MIL-SPEC DEFICIENCIES

- EXCESSIVE TIERING
- DIFFICULT TO TAILOR
- INHIBITS INNOVATION
- ALL REQUIREMENTS NOT VERIFIED
- GOLD PLATING
- NOT UNDERSTOOD BY USER
- REQUIREMENT RATIONALE LACKING



EXAMPLES



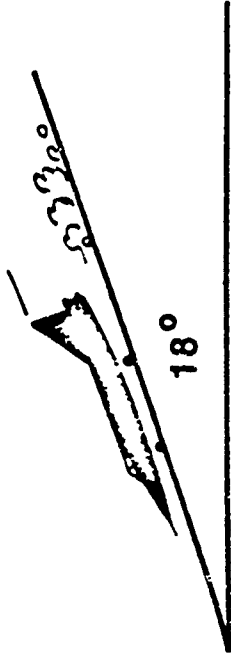
MIL-S-8512

2ND TIER

3,111 DOCUMENTS

MIL-I-3584

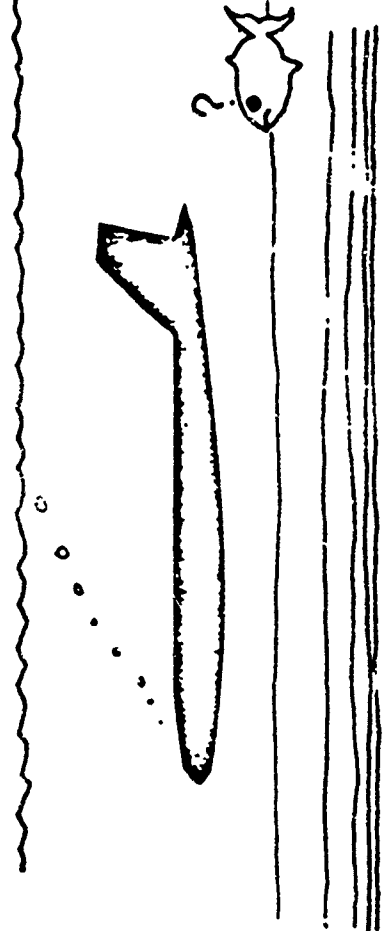
PARKING BRAKE



NO VERIFICATION

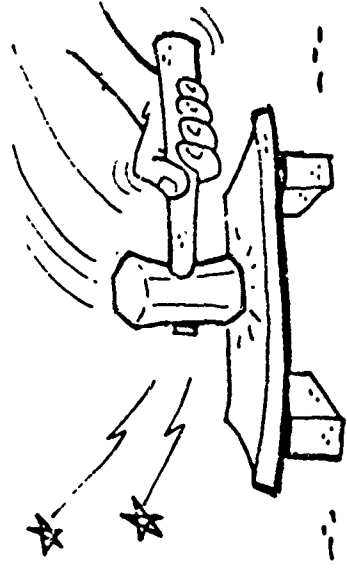
MIL-A-8065

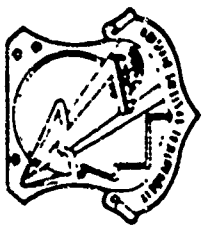
DITCHING REQUIREMENTS IN PSI ON FUSELAGE BOTTOM



MIL-L-38207

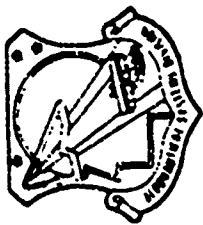
3.4.6.1 THE MATERIAL SHALL BE AN UNBREAKABLE ALLOY.





MIL-PRIME-PROGRAM DEVELOPMENT SPECIFICATIONS/STANDARDS

- **TO DEVELOP SPECIFICATIONS AND STANDARDS THAT HAVE**
 - **REQUIREMENTS STATED IN TERMS OF OPERATIONAL NEEDS**
 - **INTERFACE REQUIREMENTS**
 - **BUILT IN TAILORING**
 - **GENERAL CRITERIA VALUES PROVIDED**
 - **REQUIREMENTS VERIFIED BY TEST, ANALYSIS/ INSPECTION**



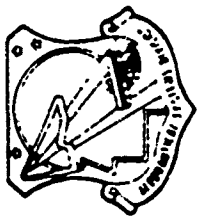
DEVELOPMENT DOCUMENTS

MIL-S-X-64

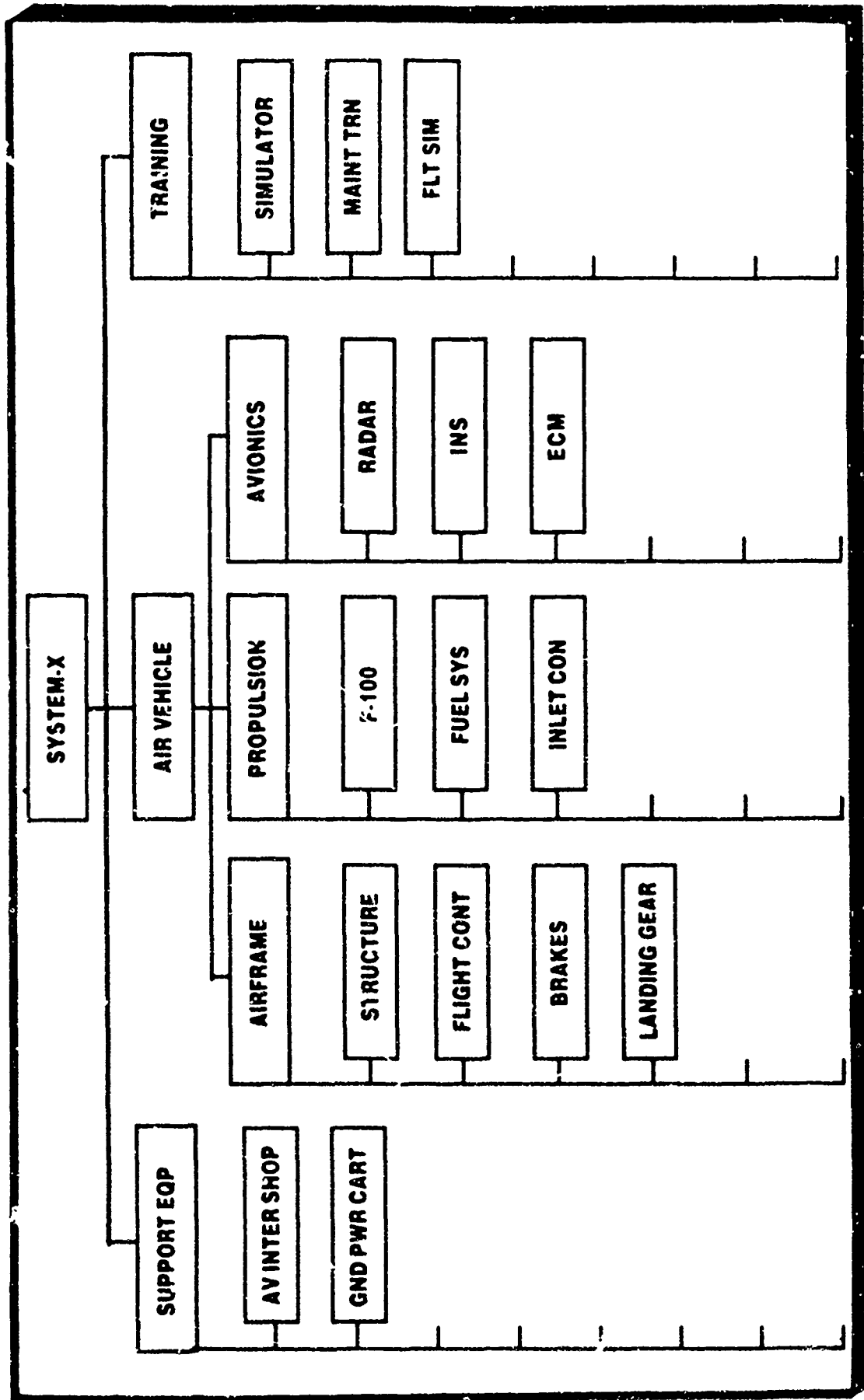
SPECIFICATIONS
OPERATIONAL NEEDS,
OPERATIONAL PARAMETERS
GENERAL INTERFACE FOR A
AND INTERFACES FOR A
REQUIREMENTS PRODUCT
PHYSICAL PRODUCT
FAMILY

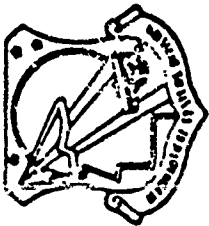
MIL-STD

STANDARD
CRITERIA AND
QUALITIES TO A
APPLICABLE TO A
PHYSICAL PRODUCT



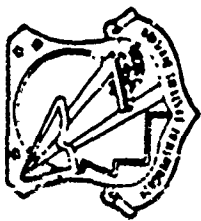
SPECIFICATION TREE





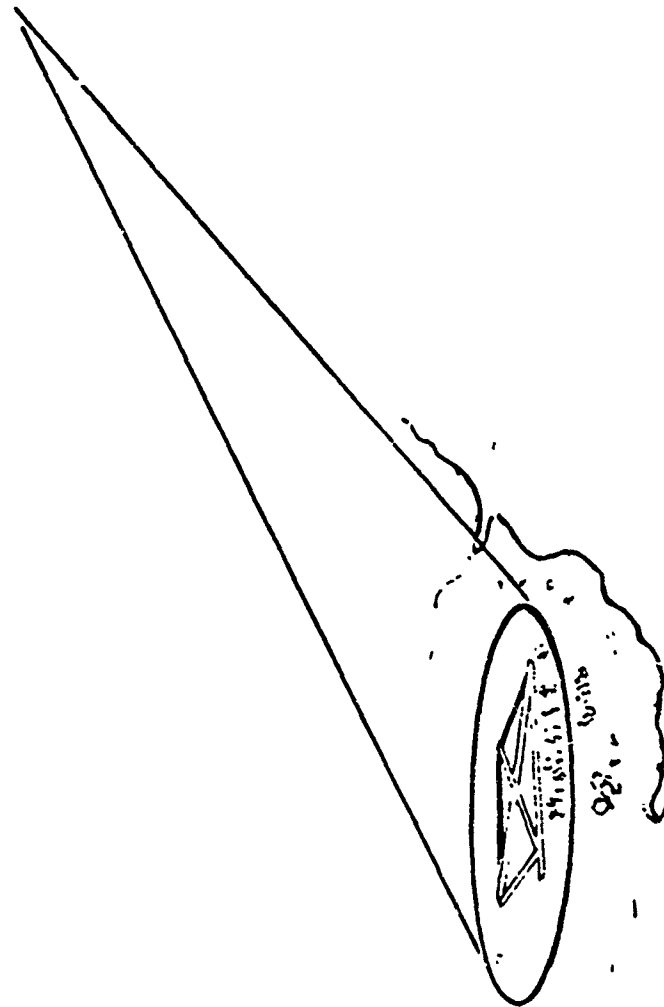
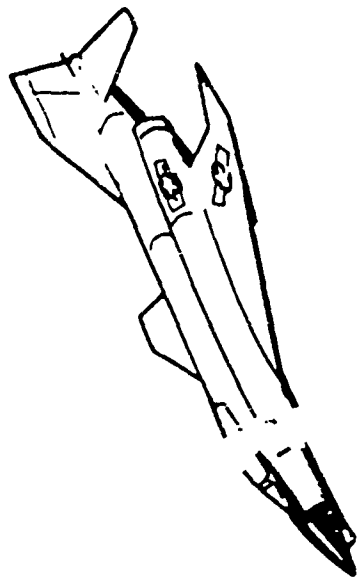
NEW SPECIFICATION EXAMPLES

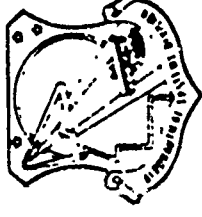
<p>LANDING GEAR SPECIFICATION</p> <table border="0"> <thead> <tr> <th></th> <th>OLD</th> <th>NEW</th> </tr> </thead> <tbody> <tr> <td>SPEC'S</td> <td>13</td> <td>1</td> </tr> <tr> <td>1ST TIER</td> <td>250</td> <td>2</td> </tr> <tr> <td>DUPLICATIONS</td> <td>60</td> <td>0</td> </tr> </tbody> </table>		OLD	NEW	SPEC'S	13	1	1ST TIER	250	2	DUPLICATIONS	60	0	<p>LANDING GEAR</p> <p>3.2.3.2.D A PARKING BRAKE SHALL _____ TO HOLD THE AIR VEHICLE</p> <p>STATIC UNDER THE FOLLOWING CONDITIONS</p> <p>_____</p> <p>_____</p> <p>_____</p>
	OLD	NEW											
SPEC'S	13	1											
1ST TIER	250	2											
DUPLICATIONS	60	0											
<p>STRUCTURES</p> <p>3. DITCHING IS _____</p> <p>UNDER THE FOLLOWING CONDITIONS:</p> <p>SEA STATE _____</p> <p>NUMBER OF _____</p> <p>PERSONNEL FOR EGRESS _____</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>	<p>AERONAUTICAL INSTRUMENTS</p> <p>3. A CLOCK SHALL BE _____</p> <p>TIME DESIGNATION SHALL BE IN _____</p> <p>(12) (24) _____ HOURS</p> <p>SIZE _____</p> <p>ACCURACY _____</p>												



DEVELOPMENT SPECIFICATIONS/STANDARDS

- RANGE _____, ALTITUDE _____
- INTERFACES...MS5495, POWER, BAY SIZE,
- FREQUENCY _____
- TEST PROGRAM _____

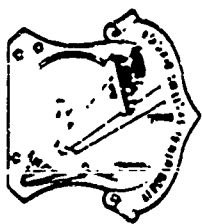




SPECIFICATION/STANDARD APPENDIX HANDBOOK

MIL-SPEC-APPENDIX
HANDBOOK

- RATIONALE FOR REQUIREMENTS
- STANDARD CRITERIA
- LESSONS LEARNED
- GUIDANCE FOR USE OF SPECIFICATION AND STANDARD



LANDING GEAR

C-5 **C-17** **NGT**



722 PAGES

REFERENCE PAGES

8 REF

SOW PAGES

35 PAGES

9 PAGES

3 REF

20 PAGES

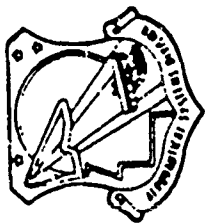
15 PAGES

15

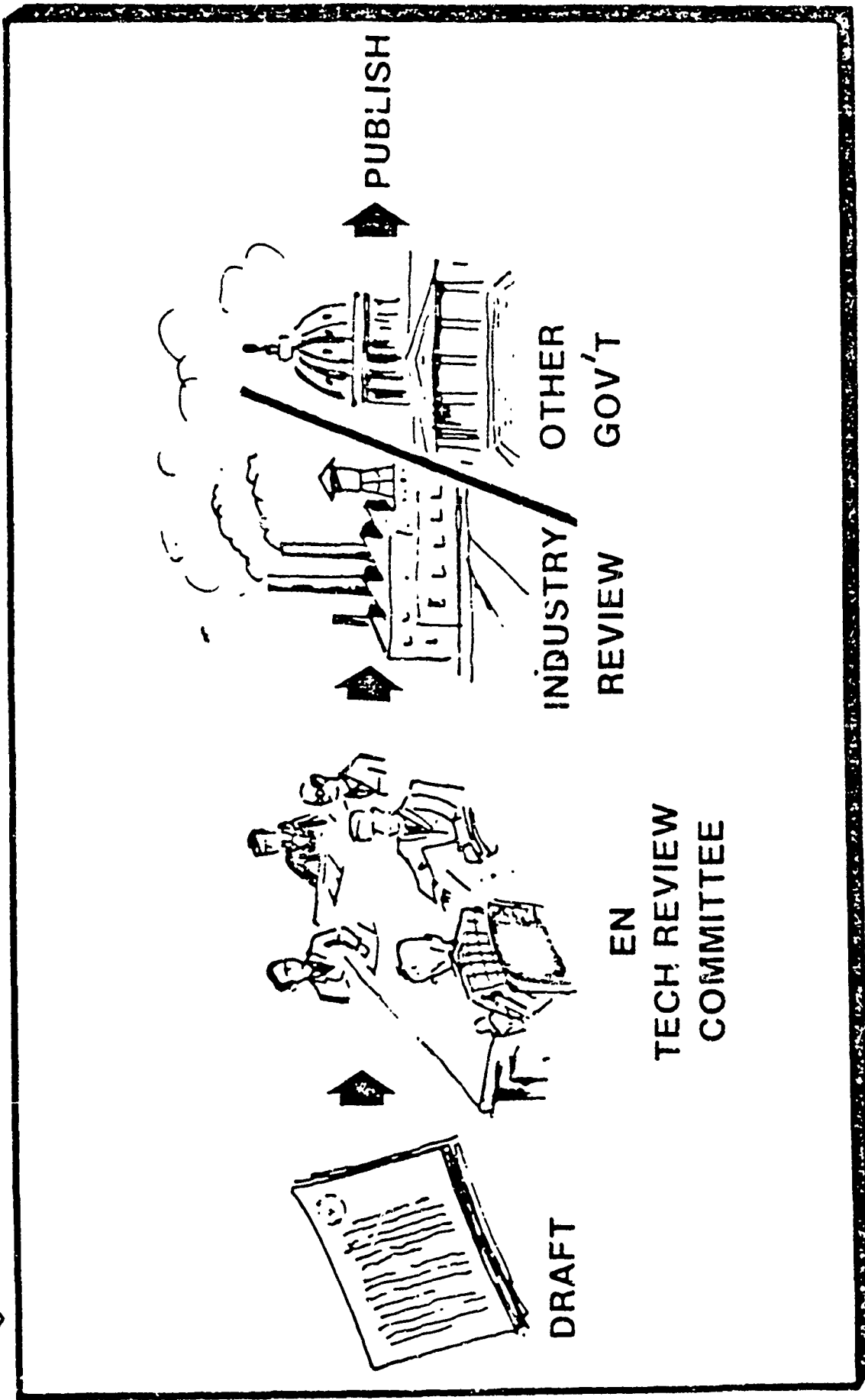
29

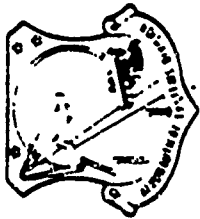
757

TOTAL PAGES



TECHNICAL REVIEWS/COORDINATION

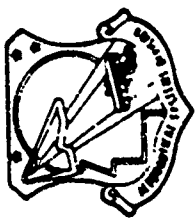




MIL-PRIME

ASD PROGRAM

- IMPLEMENT NEW PHILOSOPHY
- REWRITE DEVELOPMENT SPECIFICATION
- DEVELOP LESSONS LEARNED DEPOSITORY



STATUS

COMPLETED

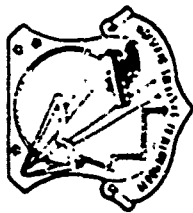
PARACHUTE SYSTEMS (DOD)
RELIABILITY PROGRAMS (DOD)
CREW STATIONS
AIR VEHICLE FLIGHT PERFORMANCE
AIRCRAFT/AVIONICS ENVIRONMENTAL
INTEGRATION
BEARINGS PULLEYS AND CABLES
ENVIRONMENTAL CONTROL AIRBORNE
FIRE AND EXPLOSION HAZARD
PROTECTION
FUEL SYSTEMS
LANDING GEAR
AERIAL REFUELING RECEIVER
SYSTEM

FINAL REVIEW

ENSP
FLIGHT AUXILIARY POWER
AIRCRAFT DOOR/CANOPY
PNEUMATIC SYSTEMS
AIRCRAFT STRUCTURES IN
MISSION COMPLETION IN
COMBAT
AIR DATA
AVIONICS MULTIPLEXING
ELECTRIC POWER AIRBORNE
INSTRUMENTS
RECONNAISSANCE
AVIONICS
MAINTENANCE
TRAINING SIMULATOR
COMPUTATIONAL SYSTEM
FOR REAL TIME TRAINING
BONDING LIGHTNING PROTECTION
DEFENSIVE AVIONICS
NAVIGATION, AIRBORNE

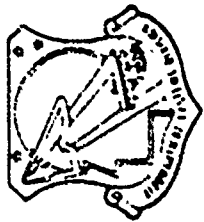
IN WORK

ENGINES RAMJET
ENGINE INSTALLATION
ENGINES TURBOJET TURBOFAN
FASTENERS
HYDRAULIC SUBSYSTEMS
COMPUTATIONAL SYSTEMS
DISPLAYS AIRBORNE
OFFENSIVE AVIONICS
RADIO AIRBORNE
AIR TRANSPORTABILITY
AIRCRAFT OXYGEN SYSTEM
PERSONAL PROTECTIVE EQUIP
SURVIVAL/FLOTATION EQUIP
PROPELLANT COMPONENTS
AIRCRAFT ESCAPE
EMERGENCY ESCAPE
DISPLAY SYMBOLOLOGY
HUMAN FACTORS
LIGHTING EQUIPMENT
MOBILITY GROUND EQUIP
SUPPORT EQUIPMENT
FLIGHT SIMULATOR



MIL PRIME PROGRAM SCHEDULE

	FY76				FY77				FY78				FY79				FY80	FY81	FY82	FY83	FY84
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4					
STUDY EFFORT																					
REVIEW MILITARY SPECIFICATIONS/ STANDARDS																					
DEVELOP CONCEPT																					
DEVELOP MODELS PROCEDURES																					
INFORMATIONAL BRIEFINGS																					
PREPARE/REVIEW/ COORDINATE DOCUMENTS																					
AIR VEHICLE, SUPPORT EQP, TRAINING SPEC'S																					



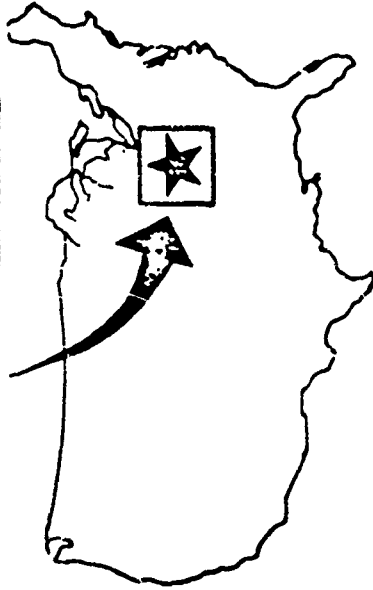
SUMMARY

CONCEPT VIABLE

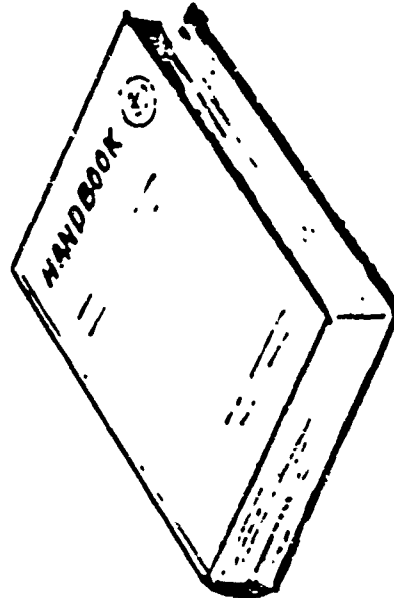
DEVELOPMENT INNOVATION
MEETS USER NEEDS



MIL PRIME COUNTRY



LESSONS LEARNED



OBJECTIVES MET



STANDARDIZATION FEEDBACK

DEVELOPMENT SPECIFICATION



AD-P003 577

OPTIONS AND OPPORTUNITIES FOR STANDARDS
A NATO/AGARD VIEWPOINT

by

John T SHEPHERD
Technical Director
Marconi Avionics Limited
Airport Works
Rochester, Kent, England.
Telephone no. MEDWAY 44400

and

Louis J URBAN
Deputy for Avionics Control
Technical Director
ASD/AX WPAFB, Dayton, Ohio, U.S.A.
Telephone No. 513 255 4768

4.1 GENERAL

This paper presents a summary of the findings of AGARD Working Group O6. This working group was established to consider 'Distributed Micro Processor Application to Guidance & Control Systems'. The results of this Study are presented in AGARD AR-178. One of the areas considered by the working group was option and opportunities for standards and it is this area that is being considered in this paper.

It should be emphasised that this document is not intended to suggest definitive standards or even to state categorically that any given standard should be developed. Rather its intention is to focus attention upon the need for standards and to point out areas where opportunities exist for standardization.

As will be seen from the previous sections in this report there is a vast proliferation in hardware and software. When systems are developed they often produce unique hardware and software, such as operating systems, executives, high level languages, etc. Since the life cycle of aircraft systems is at least twenty years from conception, it could be as much as thirty years after the initial design before the systems are finally phased out. This makes it almost impossible to maintain avionic systems in the later parts of their life cycle.

It is often argued that the application of standards can impede development and force the use of obsolete equipment. While these dangers exist, it is possible to produce a standard which is technology independent and which, therefore, does not prevent the introduction of new technology during the life of that aircraft.

To quote some examples of a technology transparent standard, the VOR navigation system initially conceived in the late 30's, was adopted in 1946 as a standard navigational aid by the U.S., becoming an international standard in 1949. This standard remains the same forty years after the original concept, although a VOR receiver today is rather different from one in the late 1930's or even the late 1950's. Equally, the Decca navigation system was introduced in 1944, has chains operating on that standard today although the technology used to implement that system has changed dramatically.

From these examples it will be seen that standards can be produced which are technology independent, and that hardware to meet these standards can be produced by a number of companies. It is precisely this approach that is advocated for standards in microprocessor based systems.

4.2 THE NEED FOR STANDARDS

In considering the need for standards in avionic systems it is worth bearing in mind that every avionic system yet produced has employed some form of standard. At any stage where two units need to interface it is necessary to establish a standard of one form or another so that the designers of the individual units know what that interface needs to be.

With a complex avionic system it is therefore general practice to establish a series of general standards for that avionic system and, using these standards as a framework, produce the detailed form, fit and function specifications for the various elements in the system. In many cases the standards established are unique to that particular system and as such may be produced hastily or without due consideration for the total implications of such a standard.

Such ad hoc standards are typically established to ensure that the various elements in the system are compatible one with another. It follows that if standards are to be established anyway, it is preferable that these standards are properly considered and evaluated by a wider body of people. Once properly established, they can be applied across a number of projects.

In order to achieve this it is essential that such standards are technology independent so that their application does not inhibit the use of advanced technology components where they would benefit the project.

The primary reason for establishing standards is one of cost effectiveness. In the past they have been principally applied to hardware. In general, no matter how large a project is, the economies of scale of procuring large numbers of standard components are such that hardware standardization on a project by project basis is undoubtedly beneficial. This is effectively standardizing on a particular technology base for a project whilst utilising more general technology independent standards to provide the overall requirement. Such an approach applied to both hardware and software undoubtedly increases the cost effectiveness throughout the total life cycle of a project. Today in particular the cost of developing support software is extremely high and, therefore, standards for support software are essential.

Considering each phase of a project in turn the cost benefits accruing from standards are as follows:-

1. DESIGN PHASE

(a) Since standards are already available, effort does not need to be expended in producing the project standards. Thus system specifications may be produced more quickly and with a high degree of confidence.

(b) The existence of standards implies that the designer may already have some familiarity with those standards and can incorporate them in his system design.

(c) The existence of standards mean that the designer can communicate more easily and again reduce the design effort required.

(d) The existence of standards permits the designer to concentrate on solving the real design problems, rather than being forced to waste time dealing with more routine requirements.

2. DEVELOPMENT PHASE

(a) The existence of standards means that suitable hardware components should already be available. These components may not be of the latest technology standard but will enable early prototyping to take place.

(b) The existence of standards means that standard test equipment for hardware and support software tools may be available and, therefore, less development effort will be required to build up the support environment.

(c) The existence of standards means that a high degree of competence already exists in those parts of the design based upon standards, thus less testing will be required and only the new elements in the system need to be proved.

(d) The existence of standards means that the development teams are familiar with the standard products and thus can concentrate upon the new areas of design.

3. PRODUCTION PHASE

(a) The existence of standards means that the economies of scale will apply and that component costs will be lower.

(b) The existence of standards means that a variety of sources should be available to supply standard equipment.

4. MAINTENANCE PHASE

(a) The existence of standards means that the maintenance personnel will require less training since they are already familiar with the standards. There are also substantial savings in technical data and technical maintenance documentation.

(b) The existence of standards reduces the number of test equipments, support software and host computers at a maintenance depot.

Since maintenance costs constitute 70% of the total life cycle of electronic equipment, the aforementioned logistic reasons are more than enough to justify technology independent standards. In practice, the cost benefits also extend to the acquisition phase.

The previous argument also implies that it is worthwhile to establish standards on a national basis and to apply them to a variety of projects. However, if each NATO country establishes its own standards in isolation, they will probably be incompatible with standards established by other NATO members. This means that the NATO community not only pays more than necessary in establishing standards but that interoperability between NATO forces is also potentially reduced.

It follows that what is needed are international NATO standards, rather than purely national standards. In order to

achieve this it is necessary to start considering standards at an early stage. It is of no use waiting until each country has developed its own standards and then attempt to reconcile all of the differing features to produce one international standard. It is far better for NATO countries to collaborate early in producing standards satisfactory to all.

4.3 AREAS SUITABLE FOR STANDARDS

4.3.1 General

In any consideration of standards for microprocessors applied to guidance and control systems, it is necessary to consider both hardware and software. Hardware systems suitable for standards are:-

- (a) Interfaces
- (b) Processors
- (c) Memory systems
- (d) Power supplies

while in software it is necessary to consider:-

- (a) High level languages
- (b) Executives
- (c) Operating systems
- (d) System design languages
- (e) System description languages.

The following subsections examine each of these areas in detail.

4.3.2 Elements of hardware suitable for standardization

4.3.2.1 Interfaces

The following interfaces are suitable candidates for standardization:-

1. Serial asynchronous interface
2. Serial synchronous interface
3. Parallel asynchronous interface
4. Parallel synchronous interface
5. Variable data interface
6. Discrete interface
7. Video data interface
8. Fibre optic bus interface.

1. Serial asynchronous interface

A serial asynchronous interface which operates upon a data bus is already widely used in avionic systems. It provides the capability of loosely coupling a federated system of processors

and sensors and, because of the loosely coupled nature of such an interface, is ideally suitable for the mission system elements in a total avionic system. Such a system enables the various elements to be redesigned almost independently of the total system concept since the loosely coupled nature of the system means that extremely detailed specifications to cope with synchronisation, etc., are not necessary and the system can be treated in more or less the same way as the earlier generation analog systems as far as conceptual design is concerned.

This technique is already widely used and standards have been produced to cope with it. In particular MIL STD 1553B exists and has gained wide acceptance on both sides of the Atlantic. The UK has recently adopted this standard as DEF Stan OO/18 (Pt 2) and NATO is about to adopt it as STANAG 3838.

2. Serial synchronous interfaces

While mission systems are suitable for implementation using a 1553 data bus structure, flight critical systems employing multiplexed redundancy techniques to achieve high integrity require data to be synchronised. Such systems are of necessity more tightly coupled than the mission systems since the system has to be considered as a total entity and designed to ensure that the integrity requirements are achieved. In particular it is essential that operations between the various elements are synchronised, and the data inputs are synchronised between elements and the various sensors providing data. MIL STD 1553B has synchronisation features that could be used for these systems.

3. Parallel asynchronous interfaces

Parallel data transfer is normally used when a high bandwidth data transfer rate is required. Its main application to date is in the internal transfers around a computer system. If tightly coupled multi-processing systems are ever employed in aircraft, then it may be necessary to utilise a parallel transfer mechanism to produce the necessary data transfer rate. Such a system, being tightly coupled and probably incorporating shared memory, would require a high degree of synchronisation. It is, therefore, probably not necessary to consider the development of an asynchronous parallel system.

4. Parallel synchronous interfaces

The synchronous version of the system described in 3 above, its main application would be high speed data transfer between elements in a tightly coupled multi-processing system employing shared memory. Such systems are becoming increasingly common in surveillance application. It is probably advantageous to consider the generation of a standard for such a system before

the pressure of project time scales forces various NATO countries to develop their own standards. Back plane or board level interface standards can be derived by taking advantage of de facto standards which exist, such as the Multi Bus (IEEE 796) S-100 Bus (IEEE 696), IEEE 896 Bus, or the IEEE 488 Bus.

5. Variable data interfaces

Another area where interface standardization can be achieved is standard data word formats for subsystems integrated on a multiplex 1553B bus. An example would be navigation position data from an inertial navigation subsystems transmitted on a multiplex data bus. The data word formats would be fixed as to bit values and definition. An example is the draft standard produced under a USAF contract (F33615-80-0124).

6. Discrete interfaces

In addition to variable data, a large number of discrete signals need to be transferred around the aircraft. It is therefore necessary to consider an international standard for discrete signals. In the UK this has been considered and Def Stan OO/18 (Pt 4) defines such a standard. This standard has already been presented to NATO as Study 3909 AVS. The UK has been invited to produce a draft STANAG.

7. Video data

As more and more electronic display systems and electro optic sensors are employed, it is necessary to consider publishing a standard for the transmission of video data. The advent of colour displays will complicate the issue since within countries comprising NATO there exists three incompatible commercial colour standards. It is therefore essential that a unified NATO video standard is established. This is currently being considered in NATO as Study 3936 AVS.

8. Fibre optics interfaces

The advent of fibre optic communication systems permit much higher band-widths to be achieved and also greatly improved EMI resistance. Therefore fibre optic data transmission will find increasing use in future systems. It is important to consider NATO fibre optic standards to ensure that compatibility will be achieved. A 1 MHz fibre optic standard is being considered in NATO as Study 3910 AVS, which has produced a draft STANAG. This is the first of a family of future fibre optic standards. Effort should next be directed to defining a wide bandwidth bus standard.

4.3.2.2 Processor systems

4.3.2.2 Processor systems

The development of processors in recent years has been largely devoted to the realization of a variety of architectures in single LSI chips. The most popular architectures are:-

- (a) The classical Von Neuman accumulator based architecture
- (b) The general purpose register architecture
- (c) The stack oriented architecture
- (d) The memory-memory organised architecture.

Architecture types (b), (c) and (d) were developed to make the operation of machines more efficient in comparison to the original Von Neuman approach by reducing the number of memory accesses required and by reducing the number of instructions, by reducing the number of get and put instructions. Since both memory and CPU costs have been reduced dramatically, many of the advantages of these architectures have disappeared. Also, the growing use of high level language and higher processing speeds from VLSI have made the fine details of these architectures of little importance to the vast majority of users.

Developments in software engineering are now focussing attention upon the need for a complete development environment. For example, the Ada Programming Support Environment (APSE) system is being developed as part of the ADA project. The high costs of such software support environments allied to the need to provide software maintenance facilities over the life of a system means that the vast number of available processors can no longer be supported and that a standard processor is needed.

In the US this has led to the adoption of MIL STD 1750A as the standard USAF processor. This specification defines an instruction set architecture (ISA) which can be provided by a variety of manufacturers in a variety of differing technologies. In addition this ISA should consider a standard interface for non multiplex data bus input/outputs, such as a 16 bit parallel I/O interface. This ISA is now being studied by the UK government as a member of the set of recommended architectures specified by MOD computer policy.

In addition to a general purpose (GP) architecture machine, there are needs for special purpose devices such as:

- (a) fast arithmetic processors, possibly working in conjunction with the GP machine. These processors would enable much faster operations to take place;
- (b) high speed parallel or array processors to handle such areas as signal processing.

NATO Study 3913 AVS is considering the subject of avionic computer standardization.

4.3.2.3 Memory systems

Memory systems still represent a considerable part of the total cost of a computing system. It is suggested that there is some need for standardization in this area. It is considered unwise to specify a specific memory technology, but it would be feasible to establish a memory interface standard which would specify at least:

- (i) Memory access techniques
- (ii) Addressing techniques
- (iii) Bus structure and communication protocols.

Item (iii) should be related to item 4 of 4.3.2.1 above.

4.3.2.4 Electrical supply interfaces

Power conditioning systems have advanced dramatically in recent years. However, it is still common practice to provide each LRU with its own power conditioning unit. It is suggested that advantages could accrue, if preconditioning was provided in the aircraft. These units would provide a partially stabilised supply. The individual LRUs would then use this supply. This would result in the power supplies of the LRUs being smaller, lighter, cheaper and more reliable. It is suggested that both power supply and power preconditioning standards be established.

4.3.3 Software options and opportunities

4.3.3.1 General

A consideration of software options is complex. However, software costs are becoming more dominant during both development and maintenance. Therefore it is necessary to consider standards in such areas as design techniques, language support systems, operating systems, and the software environment, in addition to standard languages.

In order to examine these other options, it is necessary to consider the requirements for a standard high level language. In general it should be remembered that any computer language, be it assembler or a high level language, is essentially a tool available to the designer to implement his design. The success of a language is dependent on the features it provides the designer to aid in the implementation process.

The software under consideration can be broken down into four main classes.

These are:

- (a) Support systems, in particular avionics test equipment
- (b) Crew training simulators
- (c) Mission critical systems, for example, fire control
- (d) Safety critical systems, for example, flight control.

Of these systems, test equipment uses ATLAS as a standard. Unfortunately, there are many dialects of ATLAS, which limits the usefulness of ATLAS. In the US the IEEE has produced an ATLAS standard which is specified as a standard for USAF programs.

Mission critical systems are those whose failure will affect the success of the mission. Safety critical systems are those whose failure may cause loss of the air vehicle. It should be noted that the safety critical systems form in fact a subset of the mission critical systems on an aircraft. In the past the number of safety critical systems on an aircraft have been comparatively few. In particular, flight control, engine control and stores management have been considered as safety critical items.

The advent of integrated systems and electronic displays have created more sub-system elements using digital software, which can affect safety. As a result the number of safety critical sub-systems is growing. For example, an all-electronic cockpit will force close examination of display sub-systems software to ensure that safety critical items, such as loss of display functions, which will place the aircraft in serious jeopardy, are considered. Furthermore, the advent of integrated fire/flight controls will cause critical evaluation of elements in the first control sub-systems to ensure they will not become safety critical, to the extent that they influence an unsafe flight path of the aircraft.

The careful distinction that currently exists between mission critical and safety critical items must be strictly maintained in future digital systems, particularly the software of these systems. The reason is that there will be significant differences in verification and validation and certification requirements of the software involved.

An examination of the system requirements will indicate that the following items are required from any support software system and, in particular, from a high level language:

1. Safety
2. Reliability
3. Ability to support structured or top down design

4. Ability to provide concurrency and synchronisation features.
5. Economy
6. Ease of use
7. Legibility
8. Good support environment
9. Compatibility
10. Good library procedures.

Considering each of these in turn.

1. Safety

The language design structure and implementation should be such that all conditions are reliably predictable and that no hazardous situations can occur in flight critical systems.

2. Reliability

Software reliability, which can affect safety, has become a major issue in systems and software design, primarily because software has been delegated increasing responsibility in integrating mission and flight critical sub-systems. It is essential therefore that the HLL support software, including the compiler and support tools, produce operational code with a high degree of reliability. Without this high degree of reliability, all design efforts to provide system safety and reliability would well be in vain.

3. Ability to support structured or top down design

The value of top down design has been proven many times over in software developments. It follows that any HLL considered for standardization must support top down design methodology.

4. Provision of concurrency and synchronisation functions

Because of the time critical nature of most guidance and control systems, concurrency and synchronisation features are essential in high level languages to support systems implementation.

5. Economy

Any language used should produce compact and economic code.

6. Ease of use

Since any software team is made up of personnel with various skill levels, it is essential that any high level language is easy to understand and use. The language should be easily learned, easily understood, and the effects of all of the

constructs available should be clear.

7. Legibility

Software quality requires wide use to be made of structured walk-throughs. It is essential therefore that code produced in a high level language should be extremely legible so that it is easily understood by personnel other than the original writer.

8. Provision of a good support environment

A high level language is only one tool used by the designer. It is essential, therefore, that a good support environment is provided for the language.

9. Compatibility with design procedures

The implementation phase of software design should follow naturally from the design established. As such it is desirable that visibility is maintained and it should be possible to move from the initial design through to the final implementation without losing visibility.

10. Good library procedures

One of the few strengths of FORTRAN is the vast range of mature library routines and defined I/O available within the language. Since real time programming should benefit from such reusable modules, it is essential that these features can be implemented in the language.

No existing language would appear to meet all of these requirements. The earlier languages such as FORTRAN obviously do not support modern structured design techniques and are doubtful in terms of economy. Since ADA is in its infancy, characteristics in terms of reliability, safety, ease of use and good library procedures must be developed and proven.

In practice NATO languages which are closest to providing all of these features are JOVIAL, CORAL and PEARL, largely because they have been in use for sufficient time for their reliability to be established.

A high level language specification is only one of the features needed to provide good software, and only one of the options which exist for standardization. In order to establish the other options, it is necessary to consider a total software design, maintenance requirements and the tools needed in each of these areas. These are addressed in the following sections.

4.3.3.2 Options for software standardization

The vast majority of effort in software standardization has been devoted to the standardization of higher order languages. Undoubtedly a higher order language is one of the software elements which needs to be standardized. Given the vast proliferation of languages, it is obviously important to standardize on a language. To reverse this proliferation, ADA has been established as a common DOD language.

A higher order language specification is only one of the elements needed in software development. Software tools are also needed throughout the total life cycle of software. The software life cycle is broken down into a number of areas. These are:

- (a) Software requirement and specification
- (b) Software system design
- (c) Software coding and implementation
- (d) System testing and integration
- (e) Maintenance and support.

Options for standardization exist in most of these areas and, unlike higher order languages, the vast majority are still in the formative stage. It is apparent therefore that options and opportunities exist for a concerted NATO effort to introduce standards across all elements of software development and maintenance. Examining each of these areas in detail:

(a) System description techniques

One of the major sources of error in software is the initial requirement statement for the software. Analysis of such systems as Safeguard in the US has indicated that the vast majority of faults discovered during system integration was due to problems in the initial requirements specification. Part of this problem is due to the fact that such specifications are written in a natural language and, as such, are prone to misinterpretation due to the ambiguity of such languages. English, for example, is prone to ambiguity and therefore specifications written in English are subject to misinterpretation throughout the design phase. To overcome this a number of artificial languages, known as problem statement languages, have been developed. Such languages enable the system's designer to specify his requirements in a more disciplined and unambiguous manner. One problem statement language, PSL/PSA, was developed initially by the University of Michigan. This problem statement language has been further developed in the UK by the British Aerospace plc under the SAFRA programme and in the US as CADSAT. There is no doubt that such languages can aid in the original problem statement phase of software development. Since it is obviously important for NATO members to exchange ideas and to conduct joint projects, it is obviously desirable to standardize on a common statement language.

(b) System design techniques

During the system design phase the software designer is concerned with the structure and architecture of his program. In particular he is concerned with the modules of the program and the inter-relationship between the modules. This design process is greatly aided by having a system design methodology readily available. The functions with which he is concerned are:

- (1) The activities of processors within his system;
- (2) Communications between these processors;
- (3) The degree of parallelism or synchronisation required between the processors or between co-operating processes in any one processor.

Most guidance and control systems consist of parallel processors and of parallel processes inside the processors. A number of system design methodologies exist, for example, SADT, structured hierarchy charts and, in the UK, MASCOT. Such system design tools should form a part of the total support environment needed to develop software. It is suggested that once again an opportunity exists for standardizing upon system description and design languages within NATO.

(c) Coding and implementation

It is during the coding, implementation and maintenance phases that higher order languages are necessary. As was mentioned above, there are a number of requirements which should be met by any higher order language, if it is to be used in guidance and control systems. As also was mentioned above, there would appear to be no existing language which satisfies all of those requirements. ADA can possibly meet most of the requirements, but, because of the early nature of its development, many of the most important features have not been proven. ADA, therefore, would have to be monitored carefully before it is firmly established as a standard guidance and control language - particularly for those areas of a guidance and control system which are safety critical.

A brief examination of ADA indicates that the language is far more extensive than required to code the vast majority of guidance and control subsystems. One possible approach to consider is to establish a proven subset of ADA for those constructs required for safe effective guidance and control. This subset would not be implemented implicitly as a separate ADA language specification compiler. Rather it would use the full compilers already available. The compiler would be given a discrete to identify a high integrity, flight critical software requirement. The compiler in turn would only execute those portions of the language which have been proven safe for guidance and control purposes. This approach would not infringe the concept that ADA should not be subsetted, which presumably

prevents the proliferation of a wide variety of ADA subsets and hence loses commonality. This implicit guidance and control usage of ADA would produce code compiled from a fully standard ADA compiler, only some features such as multi tasking, which are flight critical or unsafe, would not be used.

(d) System testing techniques

The system testing phase is largely concerned with detailed testing and integration. The tools used are largely those supplied as part of the system environment. As was mentioned above, one of the requirements of a higher order language is that a good support environment is provided. In this context the support structure proposed for ADA looks extremely powerful and should provide considerable aid in the validation and verification phase of software development. The implicit subset approach suggested above to cope with ADA as a high order language obviously does not invalidate the use of a support environment such as that provided as part of the total ADA package. Certainly the standardization of such a support package would be extremely valuable and once again represents a major opportunity for NATO to develop guidance and control standards.

(e) Maintenance

The technique suggested above would have direct read across into the area of maintenance and support of software during the remainder of its life cycle.

(f) Standards and procedures to provide design disciplines

In addition to the above tools, it is desirable to establish a series of codes of practice and standards to cover such areas as coding, documentation, etc.

4.4 Recommendations

The NATO Standardization Committees (MAS-AVSWP) should consider development of the following standards:

- (a) NATO STANAG 3838 should be expanded to include a Data Word Formatting, as recommended in para 4.3.2.5.
- (b) NATO STANAG 3838 should be expanded to include discrete signals, such as those included in UK Def Stan 0018 (Part IV).
- (c) High Speed MUX bus standards should also be developed to accommodate the requirements outlined in para. 4.3.2.6 (video) and 4.3.2.7 (fibre optics).

- (d) The input/output definitions in MIL STD 1750A should be expanded to include a 16 bit parallel I/O interface as per para 4.3.2.2.
- (e) Back plane/board interface standards described in para. 4.3.2.1.4 should be developed, with due caution, to ensure these hardware standards will be technology transparent and technology independent.
- (f) Power supply and power preconditioning standards should be considered, as recommended in para 4.3.2.4.
- (g) A NATO STANAG covering the ATLAS test language should be developed to provide a standard NATO equipment language
- (h) The Terminology and Nomenclature in Chapter III should be developed into a NATO STANAG.

4.5 In addition to the recommendation in para 4.4, an AGARD working group should be established to examine system description techniques, system design techniques, high level language requirements and support tools for avionic systems. This work should in particular address the development of the new ADA language.

5. ACKNOWLEDGMENTS OF DISCLAIMER

The authors wish to acknowledge the invaluable assistance of the other members of the working group:-

Dr Richard Smyth
 Dr Thomas Cunningham
 Mr Richard Bousley
 Dr David Geyer
 Mr Robert Hawkins
 Mr Richard Mejzak

All of the U.S.

Dr Antony Callaway of the U.K.

Mr Richard Bogenberger
 Mr Hartmut Thomas

of Germany

M: Gerard Boyer
 M; Jean M Valembois

of France.

The views put forward in this document represent a consensus of the personal views of the working group representatives and do not necessarily represent the views of any of the organisations or countries from which the working group members were drawn.

Equally, they do not represent the views of any panel of AGARD or of AGARD itself.



PROPOSED MIL-STD FOR AVIONICS INSTALLATION INTERFACES

Major Gerald Schopf
ASD/XRX, Wright-Patterson Air Force Base
Dayton, Ohio
(513) 255-3555

ABSTRACT

This paper describes the Military Standard (MIL-STD) now in development for avionics installation interface standardization. Originally based upon the interface standard used by the commercial airlines, this new MIL-STD, now extensively revised, is scheduled for coordination at the end of 1982.

The background which led to the development of the standard includes an analysis of the benefits expected to result from its application, the relationship between this standard and other military standards, and the similarities between this standard and the commercial (ARINC 600) standard. The "open forum" approach, using maximum industry participation, was used extensively over a two-year period to produce the document.

The technical highlights of the standard, including weight and power dissipation limits, environmental requirements, and LRU form factors are presented. A new electrical connector, which also serves as a hold-down device, is a key element in the design approach.

Air Force plans for implementation of the standard are aimed primarily at new airframes and major avionics updates of existing airframes. Also, those avionics subsystems being developed for multiple airframe application are prime candidates.

INTRODUCTION

The U.S. Air Force is proposing a new MIL-STD for the physical interfaces between an aircraft and its installed avionics. This standard is the culmination of nearly three years of investigations into the desirable attributes for the standard. The standard has been developed using an "open forum" approach with extensive participation by avionics suppliers, airframe manufacturers, and representatives from the three military services. The standard builds upon concepts developed by the commercial air transport industry; however, substantial modifications have been made to accommodate military requirements. When approved, the new MIL-STD will profoundly influence the design of both avionics and aircraft equipment areas. It should also substantially improve the reliability, maintainability, and ease of retrofit for military avionics. This paper presents the progress to date, and planned future activities for the standard.

BACKGROUND

The design of electronics for aviation use has historically required a compromise between those who build aircraft, those who operate the aircraft, and those who supply the equipment for installation. Avionics suppliers have long felt that the airframe manufacturers should provide a more benign environment for their equipment; airframe manufacturers would prefer to minimize the structural preparation necessary to provide adequately-cooled, vibration- and shock-isolated conditions. Operators would prefer ease of access to the avionics, with a minimum of requirements to test, disconnect, and reinstall failed units.

This seemingly irreconcilable situation has been reasonably approached on the commercial air transport world for many years. All avionics and aircraft intended for use in the commercial community follow an industry standard referred to as ARINC Specification 600 "Air Transport Avionics Equipment Interfaces."* This specification defines avionics case sizes, connector locations, maximum heat dissipation allowances, and other key parameters for installation. Figure 1 depicts the general configuration.

The Air Force formally recognized the need for a military installation standard at the 1979 Avionics Planning Conference. Formal analyses were initiated the following year and produced the following findings:

- The commercial (ARINC 600) specification appeared to be suitable, with minor changes, for military transport aircraft. These changes could be accommodated by a MIL-Addendum to ARINC 600.
- High performance aircraft would require substantially more stringent parameters in terms of environmental conditions.
- Exclusions to a general standard would be required to accommodate the need for placement of avionics in remote areas, or to accommodate higher heat-dissipation equipment such as radar transmitters.

These findings substantiated that a new standard would be required. This proposed standard has been referred to as MIL-STD-XXX. Figure 2 demonstrates the current concept for applying the current and proposed standard in a mixed aircraft type population.

The analysis efforts also identified other qualitative and quantitative benefits to the wider use of an installation standard:

Design and Development

The avionics are often not considered in detail until well after the airframe and propulsion system have been designed. The existence of an installation standard with well-defined dimensional and heat dissipation characteristics permits the airframe designer to consider the airframe-avionics interfaces long before the functional avionics suite has been

*Aeronautical Radio, Inc. (ARINC) is an airline-owned entity which maintains the avionics specifications as one of its services to the ownership.

Cooling Outlet
on Top

ARINC 600 Boxes
(12.6" Long x 7.6" High)
Fixed Height and Length.

Specified Low Insertion
Force Connector at
Specified Position on
Back

Cooling Air Inlet of
Specified Size, Shape,
Location at Bottom

Front Hold-Downs

Modular Cases in 1/8 ATR Increments, 1/8 to 1-1/2 ATR Sizes, as Required, to
Conform to a Maximum Heat Dissipation Limit Per Unit Size.

FIGURE 1 - KEY FEATURES OF COMMERCIAL INSTALLATION STANDARDS

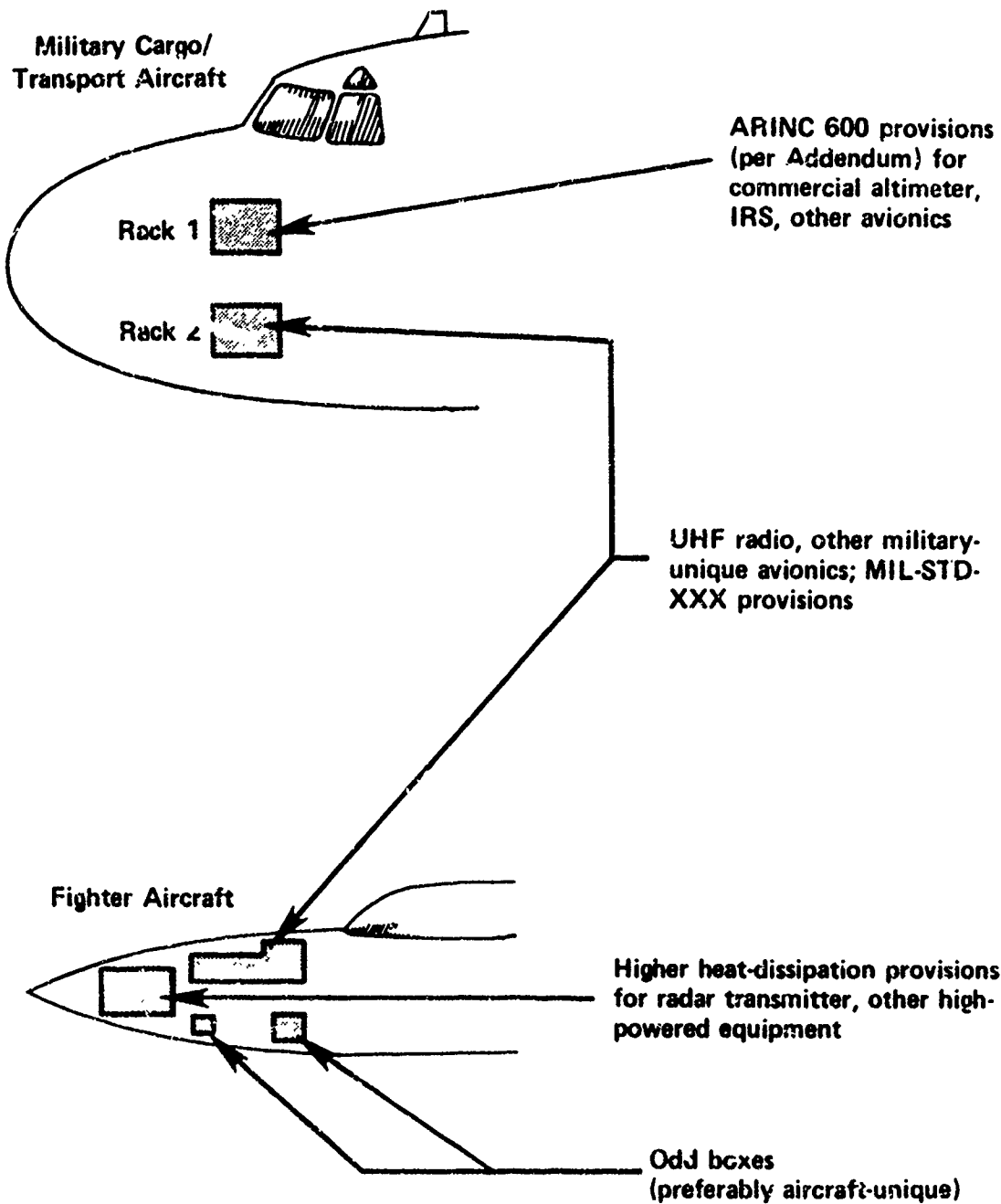


FIGURE 2 - EMPLOYMENT OF PROPOSED STANDARDS IN A MIXED AIRCRAFT TYPE ENVIRONMENT

finalized. This reduces time and cost, as well as providing for a more reliable and maintainable installation.

Operation and Support

A conservative estimate of the R&M savings achieved by improved installation interface practices is 20% over "business as usual." Further, the concepts proposed in MIL-STD-XXX would reduce the potential for battle damage, connector and cable damage, and CBW contamination by locating connections at the rear of the box.

Retrofit

Group "A" retrofit expenses are minimized if a comprehensive PME standard has been implemented. Adequate cooling is available and racks, hold-downs, and other mechanical interfaces can be adjusted to the required dimensional multiples. The wiring necessary to implement a digital bus should be in place. In the past, most of the Group "A" retrofits would have to be repeated for each new installation. Characteristically, aircraft integration costs exceeded the cost of the Group "B" equipment. Physical integration costs can be reduced by as much as one-half following implementation of the PME standard.

These potential benefits led the Air Force to proceed with more detailed development of a comprehensive packaging, mounting, and environmental (PME) concept.

DEVELOPMENT OF MIL-STD-XXX

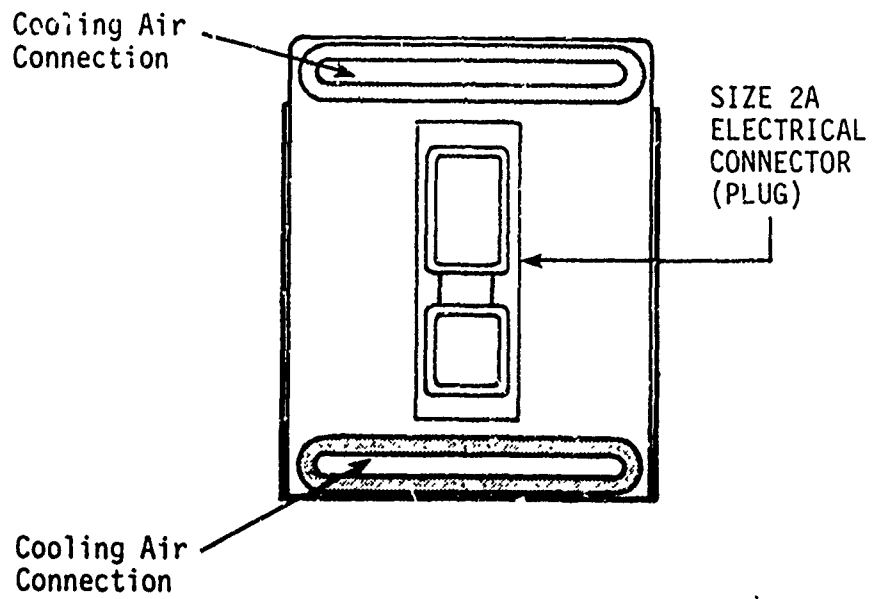
A series of open forums and smaller working group meetings were held during 1981 and 1982. Consensus seemed to change repeatedly during this period, as many new players joined the activity, and as analysis and testing of critical aspects of the standard continued. Some of the basic characteristics which have emerged at this writing are:

Form Factor

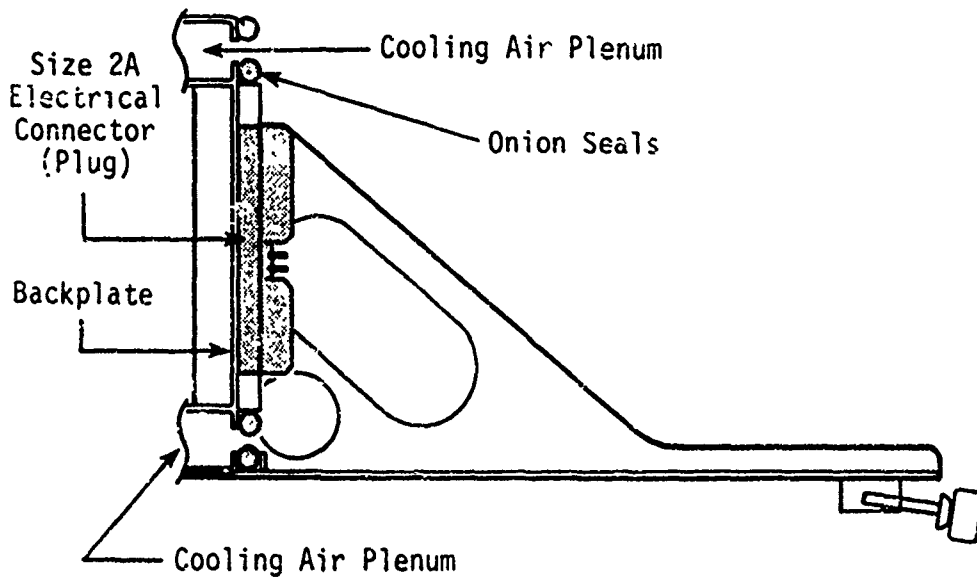
The basic ARINC 600 form factor has been retained. To permit installation in locations where the 7.6" height cannot be accommodated, the standard permits installation of the 2, 3, and 4 MCU sizes on their sides.

Connector

A rear-mounted, low-insertion-force connector, similar to the ARINC 600 connector but reduced in height, is now proposed. The connector forms the the only load-bearing mechanism at the rear of the box. A signal circuit capability up to 300 pins is available. Provisions for substituting radio frequency coaxial cable, wave guide, or fibre optic connector inserts have been made. Features of the connector attachment are shown in Figure 3.



Rear View LRU



Side View Rack Assembly

FIGURE 3 - LOCATION OF CONNECTOR ON BOX AND RACK ASSEMBLY

Cooling

Forced air cooling was selected over more exotic forms. The rationale for this choice was based primarily on battle damage considerations; however, it was also concluded that the advanced technology of thermal transfer techniques would support higher packaging densities. The air inlet locations are in the rear and exit at the front of the box. The size, weight, and thermal dissipation limits are as follows:

<u>Size</u>	<u>Width*</u> <u>(Inches)</u>	<u>Equivalent ATR</u> <u>Size**</u>	<u>Maximum Weight</u> <u>(Pounds)</u>	<u>Heat</u> <u>Dissipation</u>
2	2.25	1/4	14.3	250 w
3	3.56	3/8	22.0	375 w
4	4.88	1/2	29.7	500 w
5	6.19	--	37.4	625 w
6	7.50	3/4	45.1	750 w
7	8.79	--	52.8	875 w
8	10.09	1	60.5	1000 w
9	11.39	--	60.5	1125 w
10	12.69	--	60.5	1250 w
11	13.99	--	60.5	1375 w
12	15.29	1-1/2	60.5	1500 w

The concept does not require closed loop operation, but is adaptable to this practice.

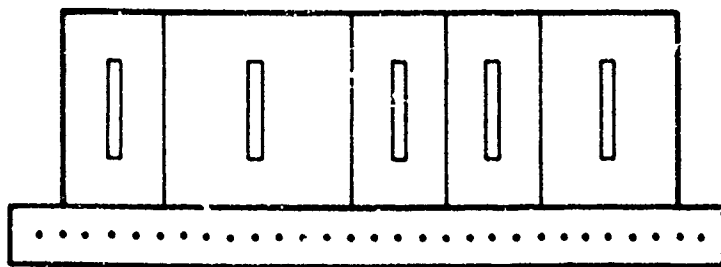
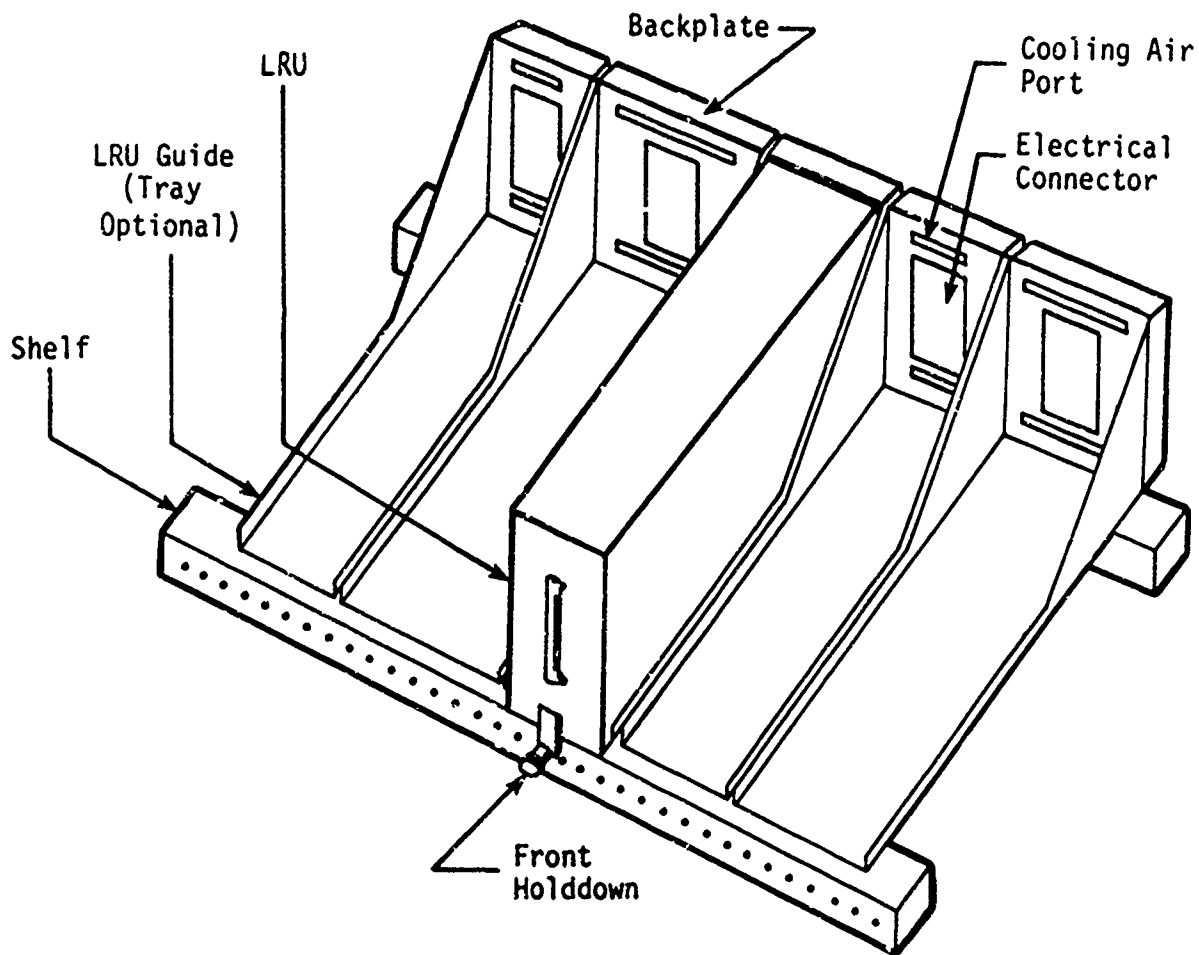
Other Features

A typical rack assembly for aircraft installation is depicted in Figure 4. The use of trays is optional; however, some means of guiding the LRU into the mating connector must be provided. The general design environment is for MIL-E-5400/MIL-STD-810 conditions. Comprehensive mechanical testing is stipulated in the standard, comprising primarily wideband, random frequency vibration. Also, a cooling evaluation test is stipulated to determine:

- (a) The total wattage input and actual heat dissipation for all modes of electrical operation
- (b) The temperature of equipment sidewalls at the thermal design condition
- (c) Pressure drop through the equipment versus coolant airflow rate
- (d) Internal temperature characteristics at the thermal design condition and other anticipated transient and abnormal environmental operating conditions

*Length is 12.76, height is 7.64 in all instances.

**Equivalent to older ARINC 404 "ATR Short" sizes.



Front View

FIGURE 4 - TYPICAL RACK ASSEMBLY

All in all, the standard is the most comprehensive consideration of avionics installation requirements that has ever been undertaken in the military. The standard has incorporated the best of many ideas that have been put forth by industry and the government. Mechanical testing and thermal analyses of prototype hardware built to the values now contained within the standard have verified that the concept is viable.

APPLICATION

Naturally, the question that arises at this time is where and when the standard will be applied. There are new aircraft programs such as JV-X, C-X, ATF, ATARS, which would present good candidates for application of the new standard. When these programs are firmer, decisions can be made as to the appropriateness of employing the new standard.

There are major block improvement programs and fighter-derivative programs for existing airframes, such as the F-15 and F-16, which present much earlier opportunities. As decisions are made to produce air-to-ground or reconnaissance versions of these aircraft, the potential for at least partial application of the new standard should be examined.

Finally, there are sizeable avionics swap-outs planned for the late 1980's and early 1990's for the major portion of the fleet. This includes the Global Positioning System (GPS), the Joint Tactical Information Distribution System (JTIDS), the Microwave Landing System (MLS), the MARK XV IFF system, the New Threat Warning System (NTWS), and several planned integrated Communications Navigation and Identification Systems (CNIs). All of these must be packaged in one or more LRUs for installation in production aircraft and for retrofit.

While the initial versions of some of these equipments are too far along to be resized to MIL-STD-XXX, there are longer-term opportunities to respecify the units as the equipment matures.

The other services are similarly planning block improvement programs and individual avionics swap-out programs. These activities provide a significant opportunity for the application of the standard if the programs can be planned for installation in the aircraft as an integral group. In that way, the necessary racks, cooling capacity, and other requirements can be installed economically with provisions for additional avionics, as appropriate.

The mechanism for applying the standard will be the same as for other interface standards in aircraft, such as MIL-STD-1553 or MIL-STD-1760. Program Management Directives will instruct program offices to utilize the new MIL-STD, or to provide trade-off analyses to demonstrate where it is not cost effective.

SUMMARY

The PME standard has progressed to a stage where only a few parameters need to be finalized. It will be ready for formal coordination at the end

of this year. If there are no serious problems during the coordination process, it is expected that it can be applied to procurements which will provide hardware for installation in the 1985 time frame.

When applied in conjunction with other interface standardization efforts, such as MIL-STD-1553 and MIL-STD-1760, this standard offers the opportunity to significantly reduce the enormous retrofit cost "bow wave" which is projected in the post-1985 time frame at the completion of the development cycles for the new avionics programs. Further, the proposed standard is very much in line with current thinking for the greater employment of two-level maintenance concepts, and quick remove and replace procedures needed to meet the threat of that period. Implementation will require changes to many practices now used in procuring military avionics; however, the grass roots reactions of the many industry participants has been that it is time for a change.

REFERENCE AND BIBLIOGRAPHY

1. *Standard Avionics Packaging, Mounting, and Cooling Baseline Study*, ARINC Research Publication 1753-01-1-2124, January 1980.
2. *Development of Avionics Installation Interface Standards*, ARINC Research Publication, 2259-03-2-2477, August 1981.
3. *Development of Avionics Installation Military Standard*, ARINC Research Publication 2256-41-TR-2811, October 1982.

ABOUT THE AUTHOR

Major Gerald Schopf is Program Manager for the U.S. Air Force PME Standardization project. He received a B.A. from Colorado State University and an M.S.B.A. from the University of Wyoming. He is currently attached to The Deputy for Development Planning, Aeronautical Systems Division, AFSC (Code: ASD/XR).

FIBER OPTICS FOR THE FUTURE - WAVELENGTH
DIVISION MULTIPLEXING

by J. Larry Spencer
NASA Langley Research Center
Hampton, Virginia 23665

Optical wavelength division multiplexing (WDM) systems, with signals transmitted on different wavelengths through a single fiber, can have increased information capacity and fault isolation properties over single wavelength optical systems. This paper describes a typical WDM system. The applicability of future standards to such a system are discussed. Also, a state-of-the-art survey of optical multimode components which could be used to implement the system are made. The components to be surveyed are sources, multiplexers, and detectors. Emphasis is given to the demultiplexer techniques which are the major developmental components in the WDM system.

INTRODUCTION

Optical wavelength division multiplexing (WDM) involves the simultaneous transmittal of information via different wavelengths of light. The various wavelengths after generation by separate optical sources are mixed by a multiplexer and transmitted over an optical communication link. At the receiving end of the link, the distinct wavelengths of light are separated by a demultiplexer and converted to electrical signals by a photo detector. By using WDM a single optical fiber will provide multiple transmission paths. WDM increases the information capacity of a single optical fiber and also provides a means for two-way simultaneous transmission (full duplex). For a given data transmission requirement a WDM system would require fewer optical fibers, repeaters, splices and/or connectors than a single wavelength system. WDM systems also have the standard advantages of single wavelength optical systems such as reduced weight and cost and immunity to lightning-induced transients.

An additional attribute of WDM is that a faulty or failed transmitter will be confined to a single communication path and will not disturb the information transmitted on the same fiber at different wavelengths. This fault containment attribute makes WDM a prime candidate for application in NASA-Langley Research Center's research program in flight crucial fault tolerant systems for advanced aerospace vehicles.

Aero-Space Technologist, Fault Tolerant Systems Branch, Flight Control
Systems Division, (804)827-3681

GENERAL SYSTEM DESCRIPTIONS

As discussed in the introduction, there are two general system design advantages using WDM techniques. The one for increasing the capacity of the transmission system and the second pertains to fault containment in the system. A four-channel WDM system block diagram for increased channel capacity is shown in figure 1. Each input channel has an optical source transmitting light at a given wavelength. The output of these sources are combined onto a single transmission fiber using a passive multiplexer. The multiplexed optical signals travel through the transmission fiber to a passive demultiplexer that separates the multiplexed signals into their optical wavelength components. A nonwavelength selective detector is used to convert each optical signal out of the demultiplexer into an electrical signal. For a WDM system, various signals such as analog and digital data, video signals, and audio signals can be transmitted simultaneously on the single transmission fiber. Systems of this type have been demonstrated for up to 12 channels (ref. 1).

A block diagram of a WDM data bus with four subsystems connected is shown in figure 2. Each optical transmitter at the subsystem will emit light at the given wavelengths ($\lambda_1, \lambda_2, \lambda_3, \lambda_4$). These transmitted signals are mixed in the multiplexer and produce four identical outputs. These outputs are distributed to the four subsystems via the fiber optic links. At each subsystem the output of the multiplexer is separated into its four optical components ($\lambda_1, \lambda_2, \lambda_3, \lambda_4$) by a demultiplexer. Each of the four optical outputs of the demultiplexer is converted from an optical signal to an electrical signal by the detector.

In a WDM data bus, all subsystems can transmit data to all other subsystems at the same time. Therefore, a system protocol is not required. An additional advantage of a WDM data bus pertains to the fault containment.

If one of the transmitters fails in a mode of transmitting erroneous data on the bus or transmitting continuous noise, it will not effect the other data transmissions because of the separation of optical wavelengths. Such a fault will be confined to its own transmission path. For a flight crucial application, the system shown in figure 2 would be one channel of a redundant data distribution system. The redundancy level would be determined by the criticality of the application. NASA-LaRC with a contract to Hughes Research Labs is developing a four wavelength system as shown in figure 2 (ref. 2). The remainder of this paper will describe the optical sources, multiplexers, demultiplexers, and detectors which are components that make up a WDM system.

OPTICAL SOURCES

Optical communication sources for aerospace applications can be divided into two classes: sources that transmit light in the wavelength region of 780 nm through 860 nm, which are usually Al Ga As devices, and sources that transmit light in the wavelength regions of 1100 nm through

1700 nm, which are usually In Ga As P devices. Currently for short distancedata transmission systems Al Ga As sources are the solution but for long-distance optical communications In Ga As P devices are extremely attractive. Both Light Emitting Diodes (LED) and Laser Diodes (LD) can be made of Al Ga As or In Ga As P for the optical sources in a communication system. The LED has a much poorer coupling efficiency to the fiber than the LD (at least 10 db less than laser-to-fiber coupling efficiency) (ref. 3). For a WDM system, the LD is more attractive than the LED. The emission spectra of a typical laser diode is less than 1.0 nm wide at the half amplitude point. This measurement is called the full width half max (FWHM) in the literature. For an LED source, the FWHM is typically 30.0 nm for the Al Ga As device and 100.0 nm for the In Ga As P devices. Optical filters can be used to increase the number of wavelengths in a WDM system using LEDs but each filter will introduce an insertion loss in the optical link. If optical filters are not used on each source, the WDM system using LEDs is limited to two wavelengths, one source constructed of Al Ga As and the second source constructed of In Ga As P. The laser diode also has the advantage of a larger modulation bandwidth. Two disadvantages of laser diodes are cost and more temperature effects on the emitted optical output. The schematic representation and operating characteristics of a typical Al Ga As laser diode is shown in figure 3 (ref. 4). The photograph in figure 4 shows a typical laser diode transmitter. The laser diode is located in the center of the printed circuit (PC) board and is mounted on a thermo-electric (TE) cooler. All the electronics on the PC board except one dual-in-line package are controlling the temperature of the laser via the TE cooler. The other IC is controlling the input current to the laser.

MULTIPLEXERS

The function of the optical multiplexer is to combine the outputs of several optical sources to form a composite signal. The multiplexer required for the system shown in figure 1 combines the outputs of the optical sources onto a single fiber whereas the multiplexer used in the system shown in figure 2 combines the outputs of the optical sources onto a multiple number of fibers.

For the multiplexer used in the system in figure 1, a wavelength sensitive component will be the most efficient. The system multiplexer will probably be the same component as the demultiplexer but it will be used in a reciprocal manner. For these wavelength sensitive devices, the attenuation of each optical signal through the device is not related to the number of channels being multiplexed. This means that channel expansion of the system will not be limited by the additional insertion losses caused by increasing the number of channels of the multiplexer and demultiplexer. A more detailed description of the wavelength sensitive multiplexing components will be given in the next section of this paper.

An interesting concept concerning the integration of optical sources and multiplexer onto a single chip has been described in the literature

(ref. 5). This single chip could be used for the sources and multiplexer in the system shown in figure 1. In figure 5a, an array of six distributed-feedback (DFB) lasers are shown coupled to a single-channel waveguide. A cross section view of the DFB laser construction is shown in figure 5b. The lasers have gratings of different spacings (.9 nm apart) and as a result, emit different wavelengths (2.0 nm apart) of light. The light from each laser is combined in a passive channel waveguide. The output of the chip waveguide is butt-coupled to a single transmission fiber.

The multiplexer used in the system in figure 2 will have an output for each subsystem in the data distribution system. A nonwavelength sensitive star coupler is used for the system multiplexer. The two basic types of star couplers are transmissive and reflective as shown in figure 6. Reflective star couplers are typically designed using a mixing rod with a mirror end plate that diffuses the light and reflects it back to all the fibers. The most popular type of transmission star is the biconical tapered coupler. The component is constructed by cutting the desired number of fibers, stripping the cladding from an area in the middle of the cut length of fiber and twisting together this uncladded portion of the fibers into a bundle or other close-packed formation. By putting the fibers under tension and subjecting them to a closely controlled heat, the fibers will soften and fuse together. By placing tension on the fibers during this fusing process, a biconical tapered region will form (figure 7). As light from the optical sources travels through the input fibers to this fused region, the taper will keep the light trapped in this fused portion of the fiber. If the device is constructed properly, the input light will be equally distributed to the output fibers. The throughput attenuation of an optical star coupler is equal to the reciprocal of the number of outputs of the star plus the excess loss in the coupler. This excess loss is typically 2 db. There are many other ways of mechanizing star couplers such as the use of lenses and optical planar devices but an indepth discussion of this subject is beyond the scope of this paper. Most of the devices require more elaborate fabrication techniques with little performance improvement.

Demultiplexer

The function of the demultiplexer is to separate the multiplexed optical signal into the individual wavelengths. The demultiplexer is the major development component in a WDM system.

The success of a WDM system is directly proportional to the crosstalk level and insertion loss associated with the demultiplexer. The demultiplexer must separate each wavelength despite the modal, linear, and angular dispersion of the composite signal caused by the transmission fiber waveguide. For the transmission system to operate with a 10^{-9} bit error rate, the crosstalk levels between the outputs should be less than 30 db. The spectral width of each output should be narrow to minimize inter-channel crosstalk. The insertion loss requirements of the demultiplexer are dependent on the overall system design of the WDM but with sources and detectors available today, allowable losses between transmitters and receivers of 50 db or more are possible (ref. 6).

The primary components used for performing the demultiplexing are (ref. 7) prisms, filters, and gratings. In this section of the paper, a general description of the principle of operation for each demultiplexing component will be given along with general performance comments.

Prisms

A functional schematic of a typical prism demultiplexer is shown in figure 8. A prism demultiplexer is classified as an angular dispersing device (ref. 8). The radiation from the input fiber is collimated by a lens and passed through the prism which dispenses the radiation according to wavelength. The separated wavelengths are focused on the output fibers by a second lens. Prism demultiplexers are expensive to produce because the design requires two lenses and an expensive, high-quality prism. A prism demultiplexer is also bulky. Few prism demultiplexers have been built because of high cost and difficulty of miniaturization and therefore, performance characteristics are sparse. In addition, precise mounting of the input and output fibers, both lenses, and the prism element is required.

Filters

In many WDM systems, filters are used to help improve the signal to noise ratio of the detected signals, but few are used as the wavelength discriminating device. In general, there are two classes of optical filters, the bandpass filter or interference filter and the high or low pass filter or the dichroic filter. For the interference filter there are essentially two types: (1) absorption filters, and (2) dielectric (non-absorbing) filters. The absorption filters are generally made of gelatin containing various organic and inorganic dyes. The filters are sensitive to temperature, humidity, and prolonged exposure to light. The filters are quite broadband (typical 100.0 nm) with a transmittance of approximately 50 percent. Because absorption filters are broadband and environmentally sensitive, they are not viable for use in WDM systems (ref. 9). The dielectric filters are usually made of multilayers of different refractive index dielectric materials. These filters are sealed against humidity effects and are quite flexible. The filters have bandwidths between .5 nm to 50.0 nm and a transmittance of 50 percent or greater. The dielectric filters are very sensitive to the angle and collimation of the entering light.

A dichroic filter is an optical component that discriminates wavelengths by transmitting certain wavelengths while reflecting energy of higher or lower wavelengths. These filters are usually used at an angle of incidence of 45° with the light beam. The longer wavelengths of light are passed through the filter while the shorter wavelengths of light are reflected at 90° to the passed beam. Temperature changes have about the same reaction, approximately 0.8 percent to 2.2 percent per 100°C, on dichroic filters as on dielectric filters.

Since a properly designed filter will only select a single wavelength out of a composite signal, an N channel WDM system would require at least N-1 filters and N+1 separate collimators. The insertion loss of a filter

type demultiplexer using filters will be proportional to the number of channels in the WDM system. The more channels the more a given wavelength may be attenuated as it passes through the demultiplexer. Also, the size of the demultiplexer is proportional to the number of wavelengths being separated. The attractive features of interference filters are ease of production, small size capability, and low cost.

Shown in figure 9 is a simplified schematic of a demultiplexer using interference filters. A four wavelength system (ref. 10) with 20 to 30 db isolation between channels and 8 to 11 db of insertion loss has been built and tested. A simplified schematic of a demultiplexer using dichroic filters is shown in figure 10. A full duplex bidirectional link has been built and tested using dichroic filters (ref. 11). The two wavelength system had approximate 12 db attenuation with -30 db to -40 db crosstalk. There are other systems for which each channel has equal path lengths (ref. 9) and for an eight wavelength system each channel passes through only three filters. Other demultiplexers have been tried using dichroic filters. One type uses fibers cut and polished at 45 degrees to their longitudinal axis. The filters are sandwiched between the fibers. By sending two wavelength signals down the fiber, one will pass through the filter while the other will be reflected 90° into a second fiber (ref. 11). Very little experimental data have been published on demultiplexers using dichroic filters.

Other demultiplexer approaches have been described in the literature (ref. 8) using multiple-grating filters. This approach has the advantage of eliminating all except one collimating lens and a single output focusing lens, thereby simplifying the packaging of the demultiplexer. Because of the difficulty of producing a photosensitive material suitable for making these devices, they are not realizable at this time but such a material might be developed in the future.

Diffraction Gratings

A diffraction grating operates in a similar manner as a prism in that it spreads out a light beam into its component wavelengths. This process, which is called dispersion, is performed in parallel with prisms and diffraction grating demultiplexers, and in series with filter demultiplexers. Any device which is equivalent in its action to a number of parallel equidistance slits of the same width is called a diffraction grating. There are two types of diffraction gratings; the transmission grating and reflection grating. All of the demultiplexers described in this section of the paper use the reflection type of grating. A reflection grating can be made by ruling parallel lines on a polished plate. The surface between the ruled lines is capable of reflecting light while the ruled lines will not reflect light. The surface reflecting the light in effect form reflecting slits. The efficiency of a diffraction grating can be improved by blazing the grooves. To blaze the grooves, each reflecting surface is constructed at an angle proportional to the wavelength at which the diffraction grating is designed to disperse. Blazed reflective gratings can be 75 percent or more efficient. Blazed gratings are constructed by ruling with a diamond point, chemical etching, photo etching, or ion milling.

To use a diffraction grating as the wavelength separating element in a demultiplexer, the light must be collimated before diffraction and must be focused on each output fiber after diffraction. The methods of performing this collimation and focusing form the major differences in the demultiplexer designs. Demultiplexers which use lenses, Graded-Refractive-Index (GRIN) rods, and planar waveguides will be described.

The optical demultiplexer described in reference 12 uses two conventional lenses, one for collimation and the other for focusing, and a blazed reflective diffraction grating as the wavelength sensitive element. The demultiplexer is mounted in a Littrow configuration as shown in figure 11. Radiation of wavelengths $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n$ enters the device through the input fiber and is collimated by the lenses. The collimated light strikes the reflection diffraction grating and is dispersed. The lens focuses the diffraction reflected beam of each wavelength on the receiving output fibers. A system of five channels with a wavelength spacing of 20.0 nm in the 800 nm region has been constructed and tested (ref. 12). The insertion loss was about 1.7 db in each channel, and crosstalk was less than -30 db.

The optical demultiplexers described in references 1 and 13 use a GRIN rod lens (also known as selfoc lens) for the collimating component. A GRIN rod lens consists of a cylinder of dielectric material with a refractive-index distribution that has a maximum at the rod axis and decreases approximately as the order of the square of the radial distance from the rod axis. In a GRIN rod lens, light rays follow approximately sinusoidal paths with nearly constant periods (L). In the demultiplexer, the GRIN rod lens is cut to length $L/4$. The lenses are used to convert the small-diameter, large-numerical-aperture beams from the input fibers of the demultiplexer into a large-diameter collimated output beam. An excellent description of GRIN rod lenses is given in reference 13. A schematic of a demultiplexer which uses a GRIN rod lens is shown in figure 12. As the demultiplexer receives the combined light radiation ($\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n$) from the input fiber the GRIN rod lens collimates the beam. The collimated beam strikes the reflection diffraction grating and is separated into its wavelength components. The reflected beams then reenter the GRIN rod lens which focuses each wavelength to a distinct position on the face of the rod. The output fibers of the demultiplexer are butted to the rod face where the output rays of the demultiplexer are focused. The space between the lens and the grating can be filled with a wedge-shaped dielectric spacer so that the entire device can be cemented together into a stable solid assembly. Experimental models of demultiplexers using the GRIN rod lens have been built to be about the size of a paper clip. A twelve channel demultiplexer has also been reported (ref. 1) which has an average channel spacing of 17.0 nm. The insertion loss averaged approximately 3 db and the average adjacent channel crosstalk was -32 db.

A planar Rowland spectrometer for an optical wavelength demultiplexer was described and reported in the literature (ref. 14). A schematic of the planar Rowland demultiplexer is shown in figure 13. If, in a Rowland spectrometer, a concave grating of radius of curvature R is placed tangentially to a circle of diameter R such that the grating center lies upon the circumference, then the spectrum of an illuminated point lying upon the circle will be focused upon this circle (known as a Rowland

circle). The Rowland geometry is unique in that, in the absence of aberration, the optics of the device produce a one-to-one image of the input spot at the output plane. If a planar waveguide is incorporated in the structure, no external collimating or focusing devices between the input and output fibers are required.

The demultiplexer is constructed by epoxying a thin optical waveguide (approximately .075 nm) between two support microscope slides. To the end face of the device, which has a radius of curvature R , a reflection diffraction grating is attached. At the front face of the structure, which is cut to radius $R/2$, the input and output fibers are butt coupled to the planar waveguide. The input beam to the demultiplexer, consisting of the wavelength components ($\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n$), enters the planar waveguide from the input fiber. The light travels through the planar waveguide where it strikes the blazed reflective diffraction grating at the end face of the structure. The beam is separated into its wavelength components where each component is reflected towards the front of the structure. Because of the Rowland geometry, each wavelength is focused to a different spot along the front face. For each wavelength position, there will be a fiber to collect the output signal. A four channel device has been constructed and tested with 10.0 nm separation of wavelengths. The cross talk measured was -18 db and typical insertion loss from input fiber to output fiber was approximately -9 db. More than half of the insertion loss was due to the poor diffraction grating efficiency (5 db). A demultiplexer using the Rowland geometry similar to the GRIN rod lens device can be cemented together into a stable solid assembly.

Optical Detectors

None of the WDM systems described in this paper require wavelength sensitive optical detectors. Depending on the particular WDM system design, a decision will have to be made on whether to use a PIN photodiode or an Avalanche photodiode. The Avalanche photodiode is a more sensitive detector but it requires a 100 to 400 volt bias. A detailed discussion of optical detection is beyond the scope of this paper.

In many WDM system implementations, the overall optical losses could be improved by mounting the photodetectors directly on the demultiplexing component. This technique would eliminate the losses associated with coupling the demultiplexer to the receiving fiber and the losses associated with coupling the receiving fiber to the photodetector. A drawing of such an arrangement using the planar Rowland demultiplexer is shown in figure 14. In this representation, many of the optical receiver components are mounted on top of the structure using a ceramic hybrid circuit wafer.

Standards Issues

Many of the same issues for standards apply to WDM systems as apply to a single wavelength system. The issues include terminology and symbols, interconnecting devices, sources and detectors, test methods and instrumentation, cables, and wiring installations. One issue which needs to be standardized in WDM systems is the optical wavelength and spectral bandwidth of the sources. In many designs of wavelength sensitive

multiplexers and demultiplexers, the receiving fibers must be standardized. If an optical source at a given wavelength fails, it could be replaced without major testing and repositioning of the receiving fibers if there are standard wavelengths of operation.

Because of WDM data bus is functionally equivalent to fully connected point-to-point links, a system protocol standard is not required as in MIL-STD-1553B and MIL-STD-1773. This is one area where the use of WDM in a data bus will negate the need for a standard.

Concluding Remarks

WDM is a viable technology for optical fiber transmission systems for both increased channel capacity and fault containment. Both the wavelength multiplexing and demultiplexing can be performed in optically passive components with the demultiplexer being the critical element in the system. The use of prisms as the wavelength sensitive element has the advantage over filters for separating the wavelengths in parallel but has the disadvantages of requiring collimating and focusing of all input and output fiber signals respectively. A prism has additional disadvantages at being large and expensive to construct with currently available materials. The use of optical filters has the advantage of being low in cost but a disadvantage of requiring collimating and focusing on all input and output signals respectively. Another disadvantage is that the wavelengths are separated in a serial manner causing the insertion losses to be a function of the number of channels demultiplexed. Finally, the physical size of the demultiplexer increases with increasing number of channels to be demultiplexed. At the present time, diffraction gratings are the best wavelength sensitive elements to be used in optical demultiplexers. Practical devices with currently available materials have been constructed and tested. The differences between the designs using diffraction gratings are that different elements are used to collimate and focus the input and output optical signals. Conventional lenses, GRIN rod lenses and Rowland configured planar waveguides are all viable candidates. Additional work is needed in evaluating the environmental effects on the components, reducing the component sizes and cost, and gathering field experience with the WDM system.

REFERENCES

1. Metcalf, B. D.; and Providakes, J. F.: High-Capacity Wavelength Demultiplexer with a Large-Diameter GRIN-Rod Lens. *Applied Optics*, vol. 21, no. 5, Mar. 1, 1982, pp. 794-796.
2. Spencer, J. Larry; and Himka, Roger L.: Description and Planned Use of a Data Distribution Evaluation System for Fiber Optic Data Buses. Paper presented at the National Aerospace and Electronics Conference, Dayton, Ohio, May 18-20, 1982.
3. Bolz, Don; and Herskowitz, Gerald J.: Components for Optical Communications Systems: A Review. *Proceedings of the IEEE*, vol. 68, no. 6, June 1980, pp. 689-731.
4. Ladany, I; and Hammer, J. M.: Single-Mode Laser Studies: (I) Design and Performance of a Fixed-Wavelength Laser Source and (II) Coupling of Lasers to Thin-Film Optical Waveguides. NASA CR-3341, 1980.
5. Aiki, K.; Nakamura; and Umeda, J.: Frequency Multiplexing Light Sources with Monolithically Integrated Distributed Feedback Diode Lasers. *Applied Physics Letters*, vol. 29, no. 8, Oct. 1976, pp. 506-508.
6. Barnoski, Michael K., ed.: *Fundamentals of Optical Fiber Communications*. Chapter 6. Academic Press, Inc., 1976, pp. 183-201.
7. Rickard, Eric: Wavelength Division Multiplexing: Overview of the State of the Art. RADC-TR-81-47, Mar. 1981.
8. Tomlinson, W. J.: Wavelength Multiplexing in Multimode Optical Fibers. *Applied Optics*, vol. 16, no. 8, Aug. 1977, pp. 2180-2194.
9. Stigliani, Daniel J., Jr.; Hanna, D. W.; and Lynch, R. J.: Wavelength Division Multiplexing in Light Interface Technology. IBM Report, NR 71-531-001, Mar. 18, 1971.
10. Hendricks, Herbert D.; Spencer, J. Larry; and Magee, Carl J.: Fiber Optics Wavelength Division Multiplexing for Data Systems. *Future Connections-NASA*, vol. 1, no. 2, Nov. 1981, pp. 9-12.
11. Nelson, A. R.; Gasparian, G. A.; and Beckel, G. W.: Bidirectional Coupler for Full Duplex Transmission on a Single Fiber. CODACOM-77-1798-F, ITT Electro-Optical Products Div., Aug. 1979.
12. Aoyama, Koh-ichi; and Minowa, Jun-ichiro: Low-Loss Optical Demultiplexer for WDM Systems in the 0.8- μ m Wavelength Region. *Applied Optics*, vol. 18, no. 16, Aug. 15, 1979, pp. 2834-2836.
13. Tomlinson, W. J.: Application of GRIN-Rod Lenses in Optical Fiber Communication Systems. *Applied Optics*, vol. 19, no. 7, Apr. 1, 1980, pp. 1127-1138.
14. Yen, H. W.; Friedrich, H. R.; and Morrison, R. J.: Planar Rowland Spectrometer for Fiber-Optic Wavelength Demultiplexing. *Optics Letters*, vol. 6, no. 12, Dec. 1981, pp. 639-641.

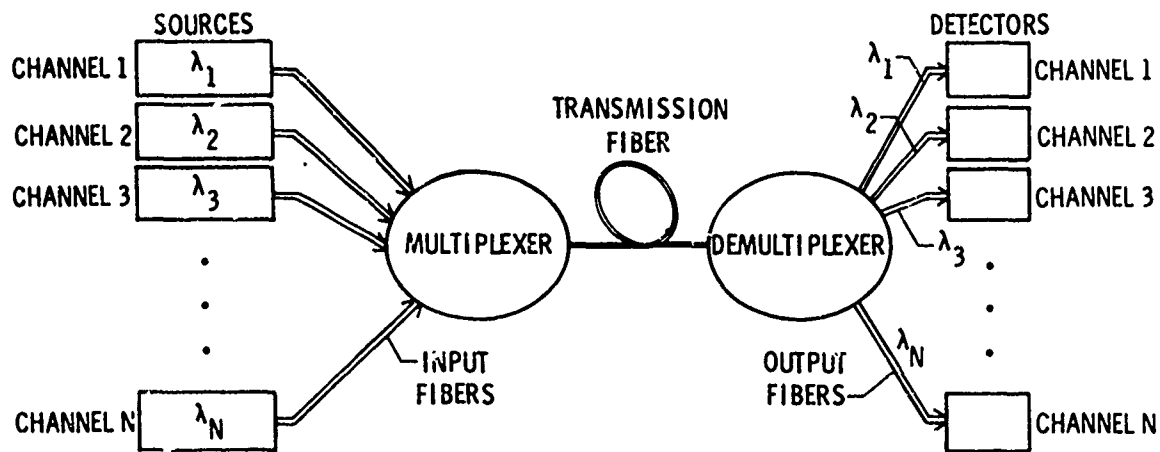


Figure 1. - WDM system for increased channel capacity.

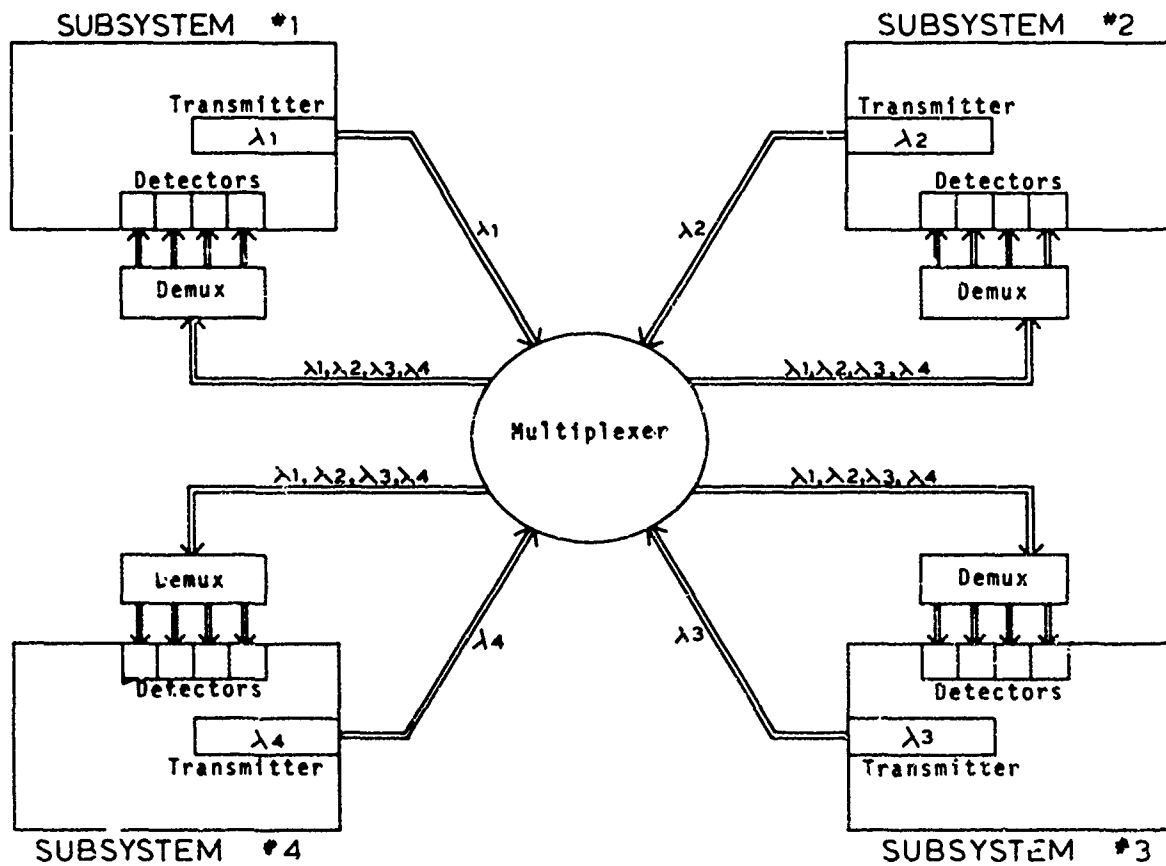


Figure 2. - WDM system for fault isolation.

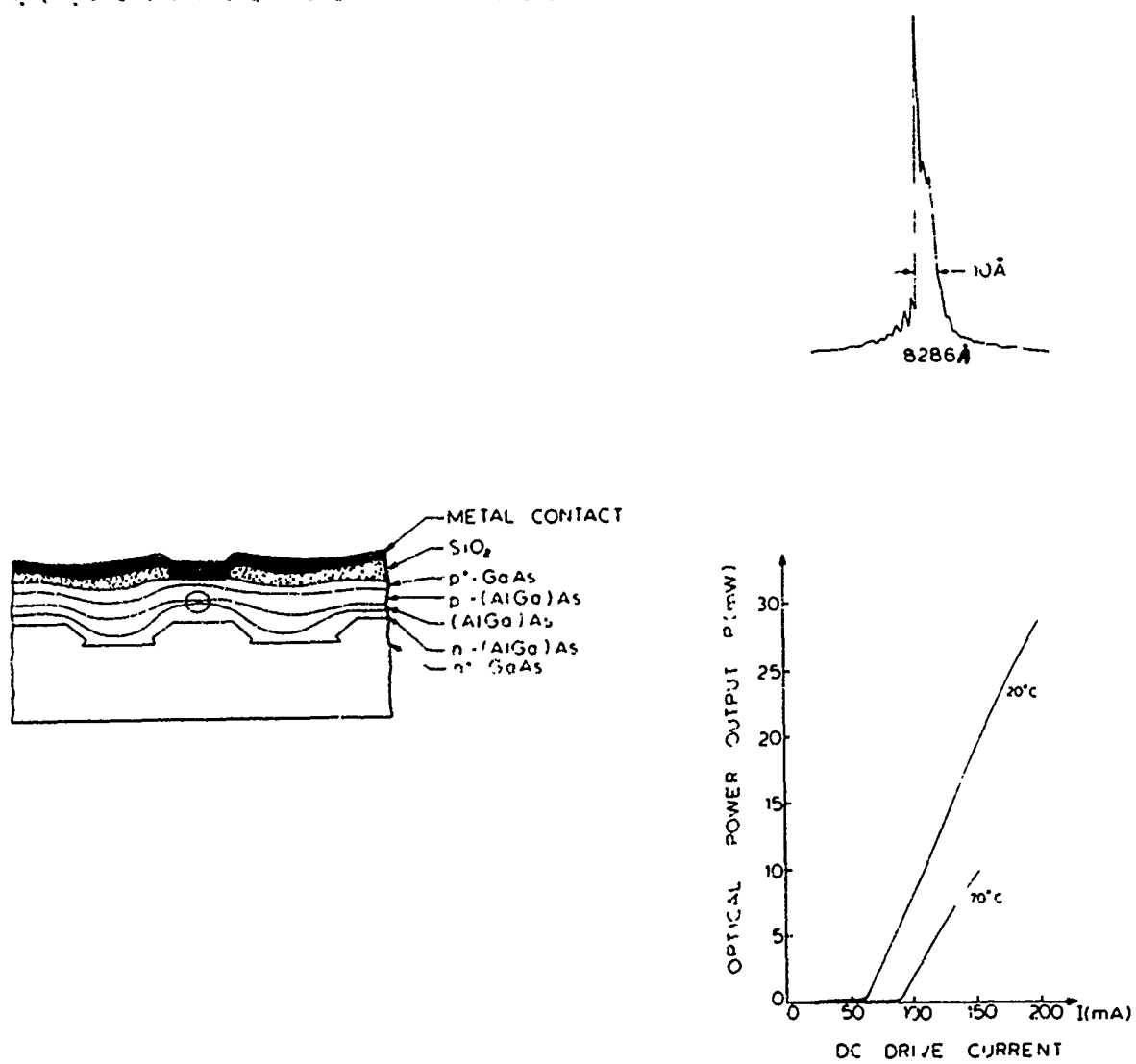


Figure 3.- Laser diode.

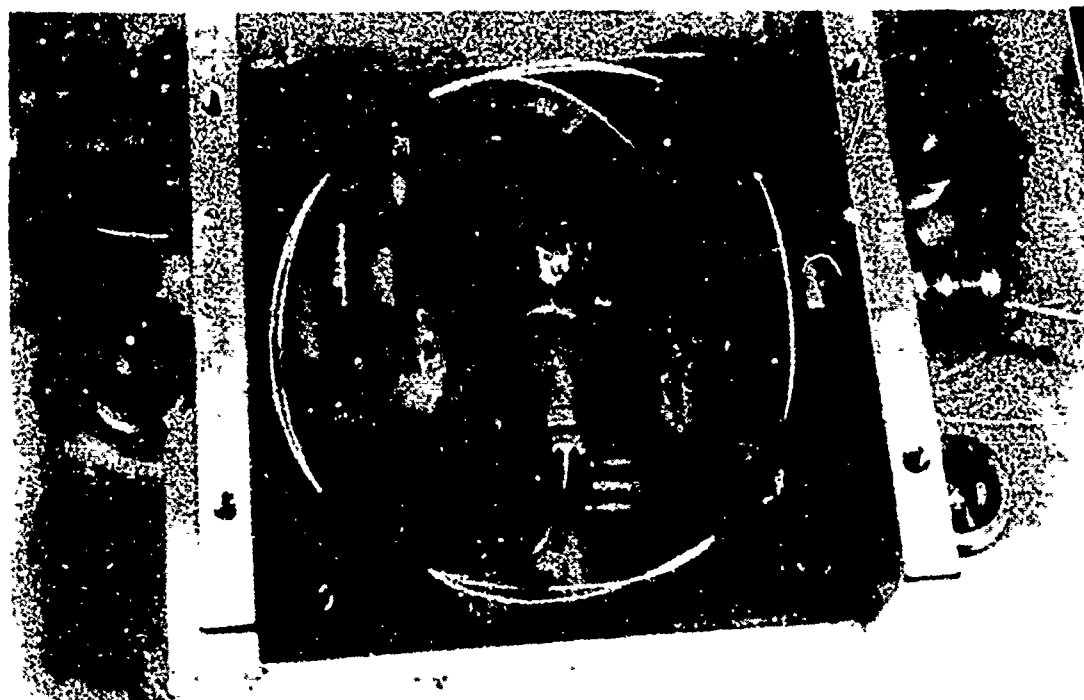
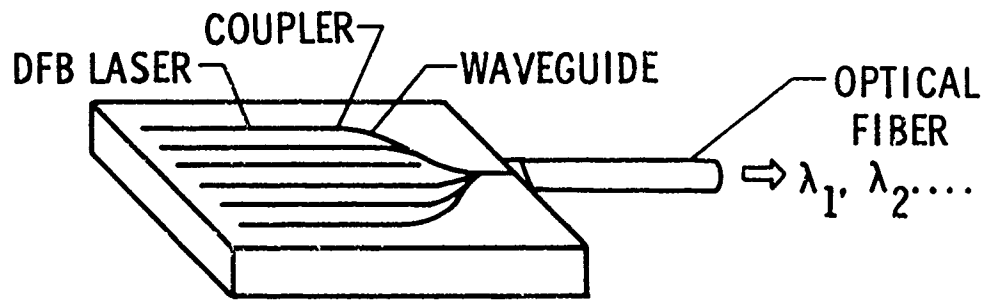
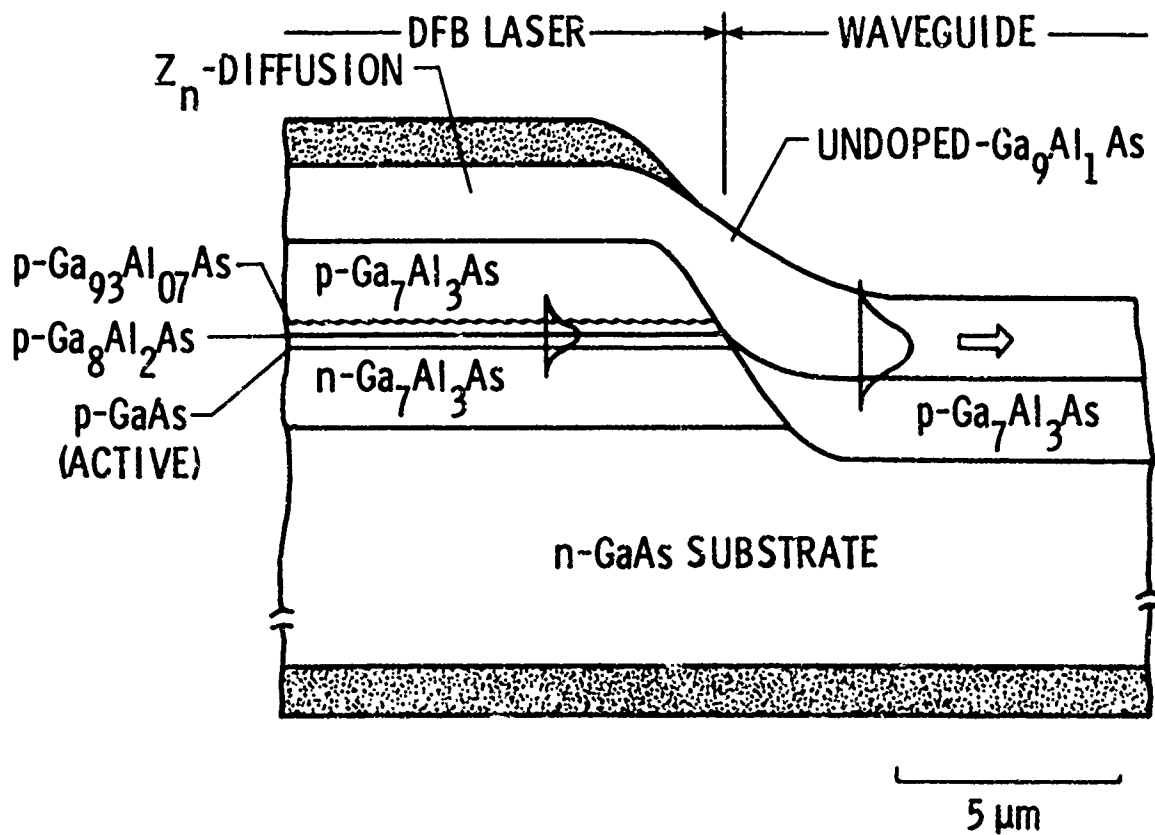


Figure 4.- Laser transmitter.

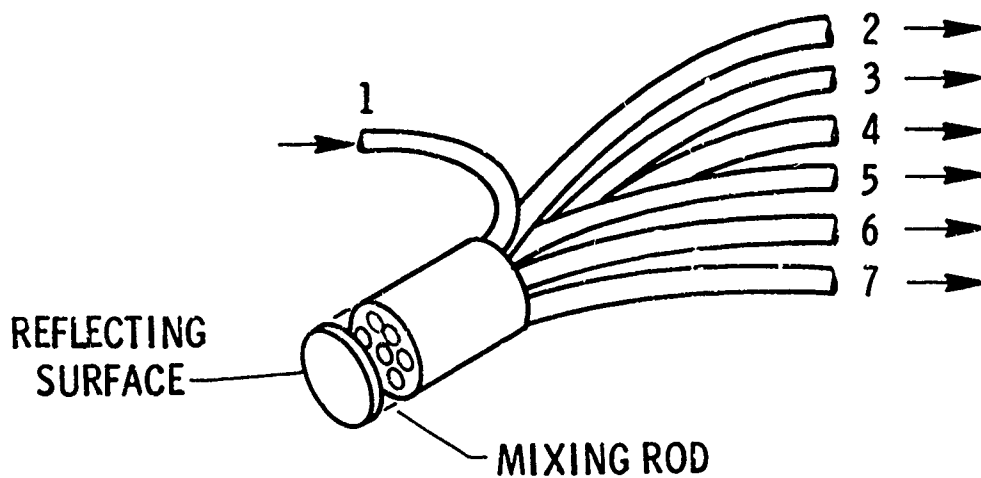


(a) Schematic

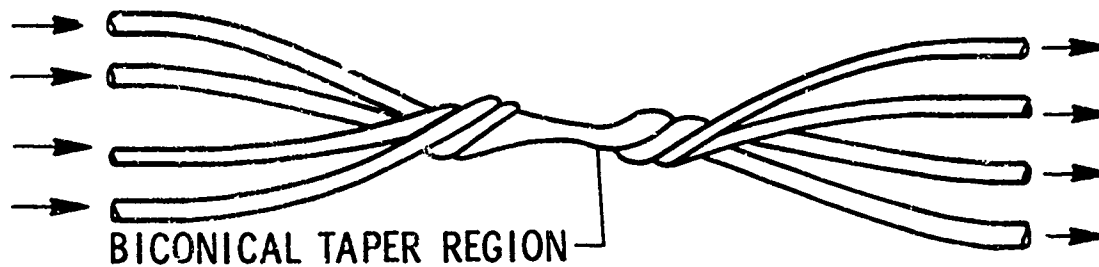


(b) Cross-sectional view

Figure 5. - Array of 6 DFB lasers.



● REFLECTIVE STAR



● TRANSMISSIVE STAR

Figure 6.- Optical Couplers

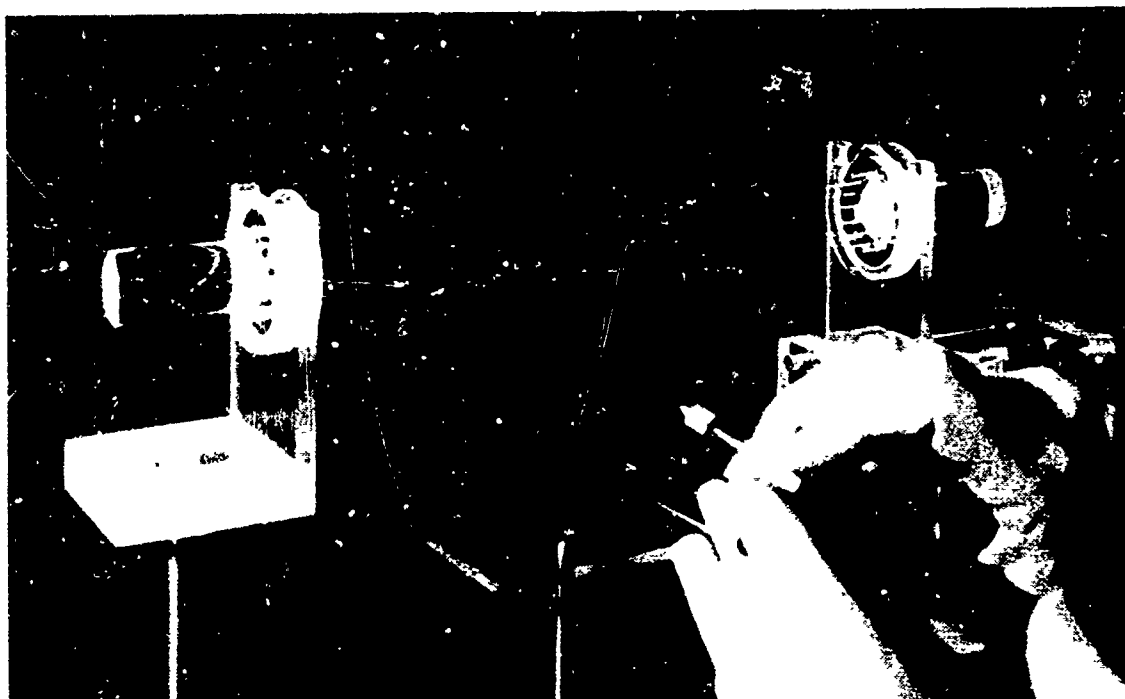


Figure 7.- Making of a fused biconical taper coupler.

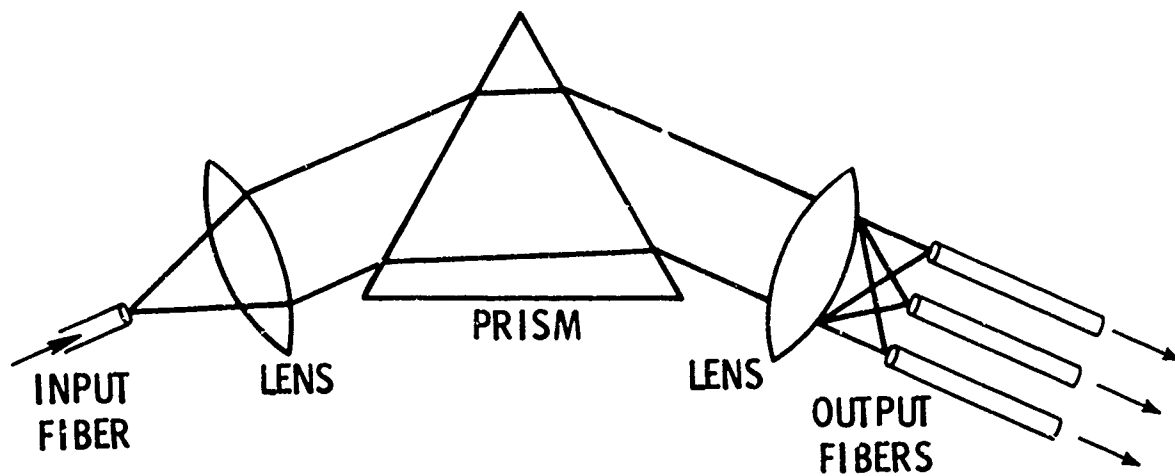


Figure 8. - Prism demultiplexer

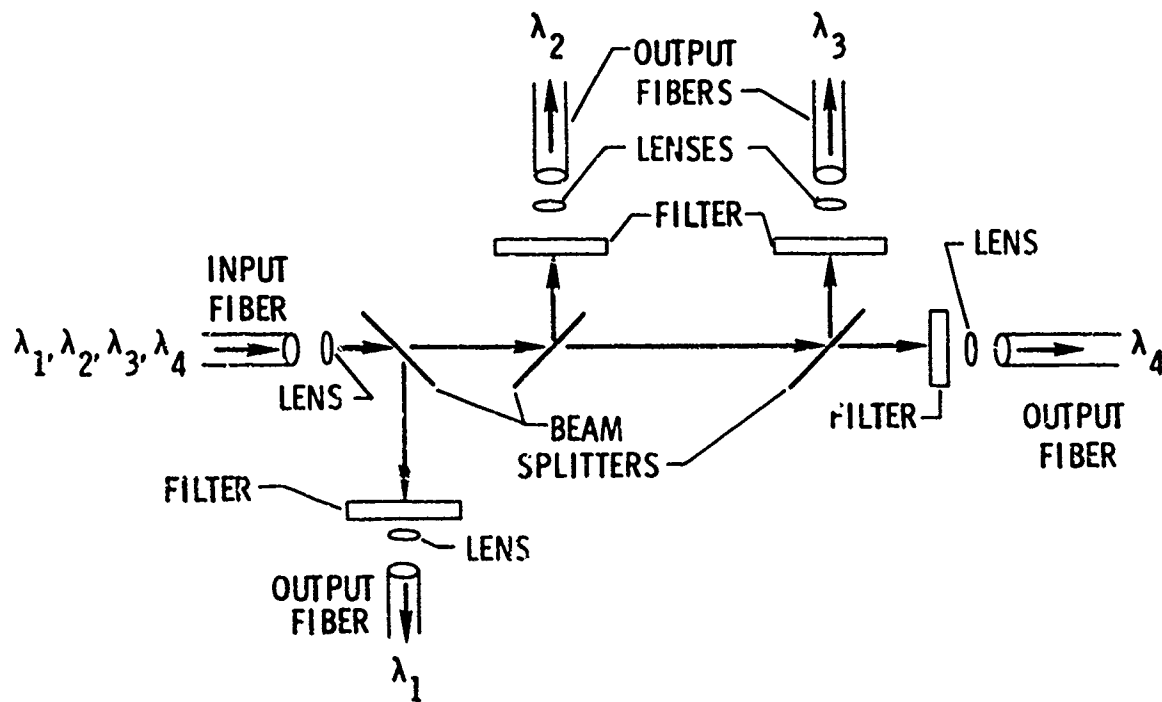


Figure 9. - Demultiplexer using interference filters.

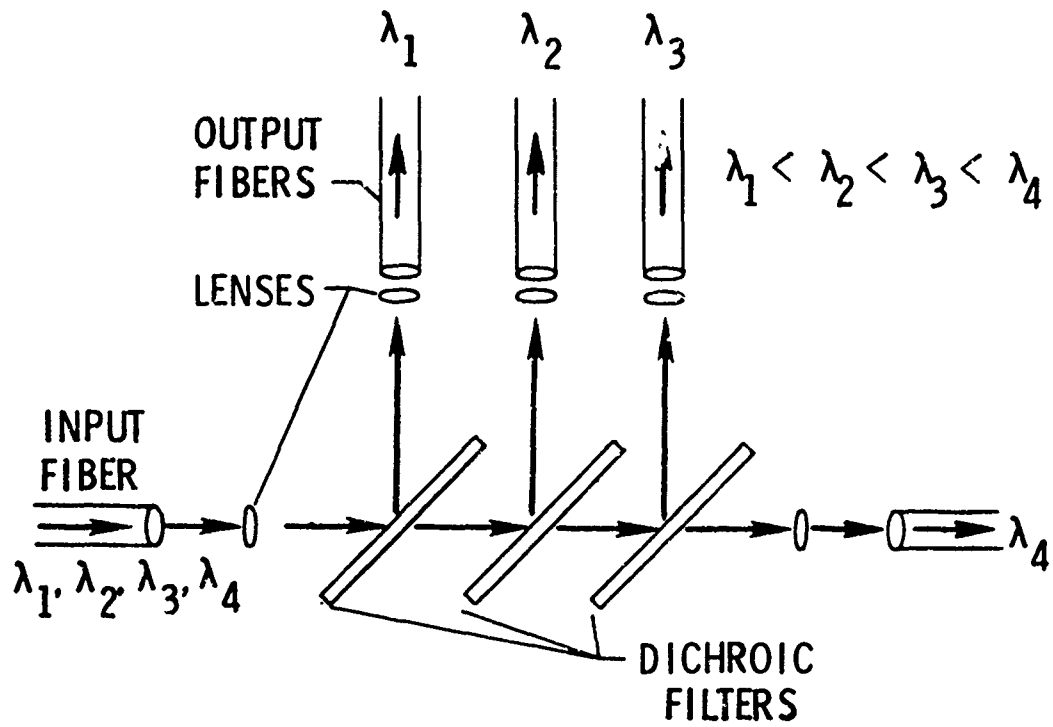


Figure 10.- Demultiplexer using dichroic filters.

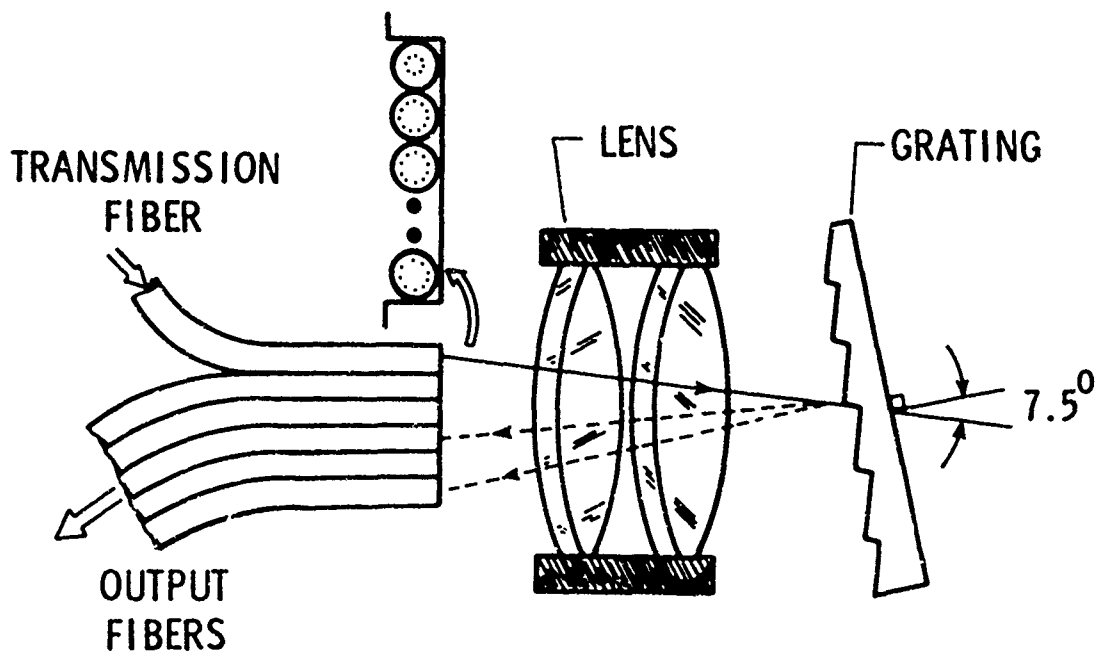


Figure 11.- Demultiplexer using conventional lenses in a Littrow mount.

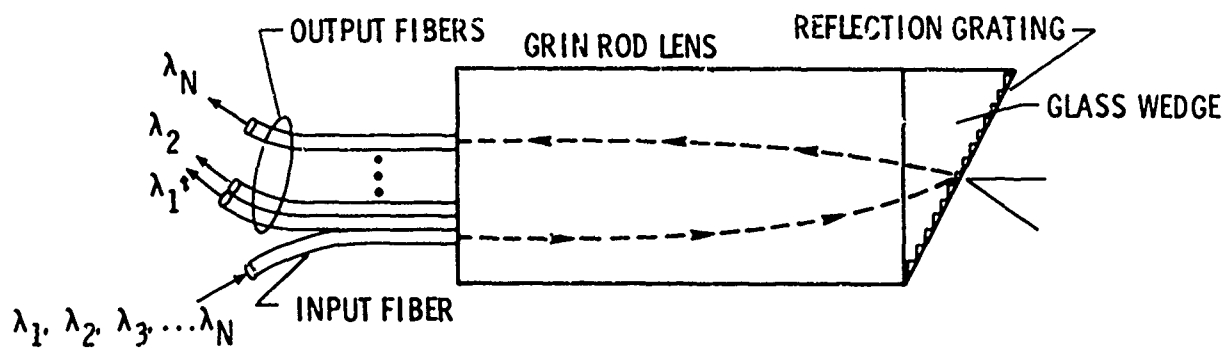


Figure 12. - Demultiplexer using a GRIN rod lens.

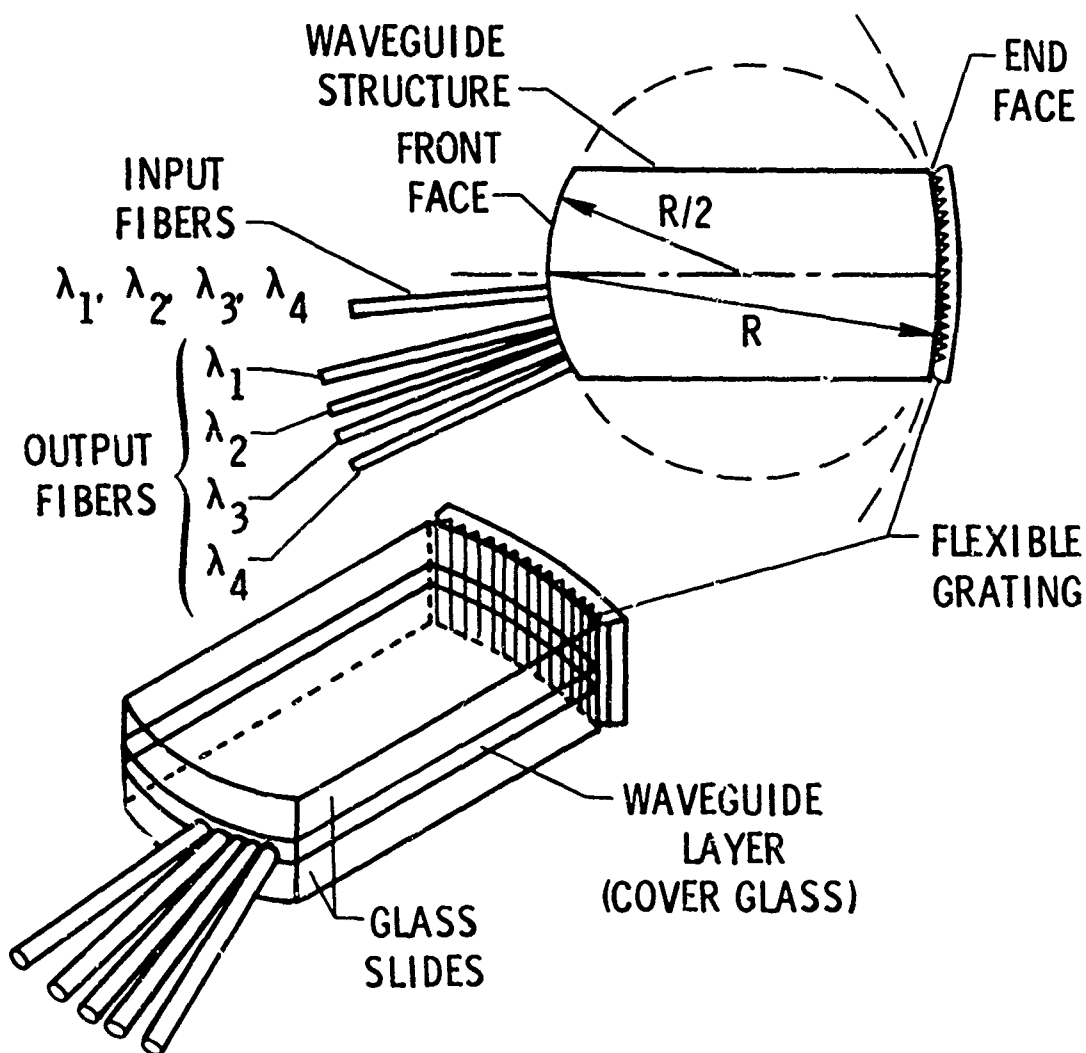


Figure 13. - Demultiplexer using a planar Rowland structure.

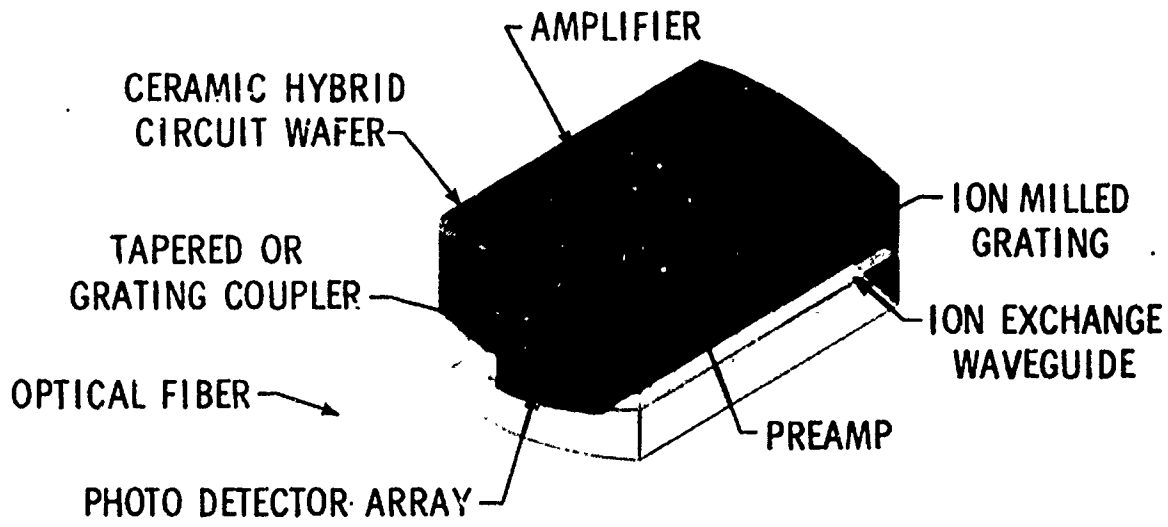


Figure 14. - Demultiplexer with adjacent photo detectors.

INTEGRATED CNI AVIONICS AND FUTURE STANDARDIZATION

Darlow Botha
Avionics Laboratory, AFWAL/AAAI
Wright-Patterson AFB, Ohio
Tel: (513)255-2766/4666

BIOGRAPHY: Darlow Botha is the initiator and technical director of the ICNIA project. His experience is in communication systems and in the application of advanced technology to integrated system concepts. Before joining the Laboratory in 1972, he worked for Texas Instruments in CNI systems, and at MITRE in advanced communication concepts for command and control systems. He is chairman of the Integration and Standardization subcommittee of the AIAA Digital Avionics Technical Committee and a member of the Standardization Panel of the Armament and Avionics Planning Conference.

ABSTRACT: The Integrated Communication Navigation and Identification Avionics (ICNIA) project is a system design and validation effort within service laboratories; the system includes the full suite of CNI functions, and is an alternative to a collection of independent C, N, and I system black boxes. The concept is that of a family of modules; front end, preprocessor, signal processor, data processor, clock, COMSEC/TRANSEC, etc. which can be tailored to an individual aircraft type and mission, implying that there could be unique LRUs for each aircraft type. This presents a logistically intolerable situation unless the LRUs, fleetwide, consist almost entirely of standard SRUs of relatively few types across the fleet; standardization at the module, sub-assembly, or SRU level is required. This paper discusses the system approach to CNI implementation in 1990s procurement, current standards already adopted in the program, and the future standards which appear to be necessary for fullest benefit from the ICNIA concept.

I. ICNIA: A TECHNOLOGY ALTERNATIVE

1. GOALS.

The goal of the ICNIA program is Affordable-Operational-Mission Effective CNI. This really means anti-jam CNI, since the current voice radio, TACAN, and IFF hardware are affordable, operational and effective -- for peacetime missions. However, the developing systems intended to provide the CNI functions in the face of electronic warfare measures have, in some cases, degenerated from the "universal" system for wide application in a tactical battlefield to terminals that can only be afforded on the highest value platforms, and an alternative is needed.

2. APPROACH.

AFWAL technical goals are to provide "technology alternatives... demonstrated options for...expanded operational capabilities...a foundation for development of innovative concepts...and options for current and future Air

Force needs." The current CNI approach is separate development of AJ radio avionics in several organizations within the Air Force. The ICNIA approach is to look at the avionics end of independent AJ voice, data, IFF and navigation systems, and treat the total radio suite as a single system for design purposes. Initiated in exploratory development, emphasis was on integration, commonality and time sharing as techniques to reduce electronic hardware costs and space occupied by the electronic boxes in the aircraft. There is growing evidence that the independent developments in many cases have reached a point at which the cost of integration into the aircraft exceeds the cost of the electronics. The ICNIA approach as a "technology alternative" has therefore evolved to include transition to operational application, and how technology concepts can contribute to transition, as a project requirement equal in importance to the more conventional development goals.

3. CONCEPT: A "TAILORABLE FAMILY OF MODULES"

Early concepts of integration as a design methodology to achieve goals of cost and space reduction evolved to include "integrability" as a key attribute of the demonstration hardware, that is, a technical approach which can impact the cost of installation and support as well as electronics cost. The concept is that of a "family of modules", "tailorable to aircraft and mission". The ICNIA project is thus the demonstration of a system design of building blocks readily integrated by the system integrator in any application, from a brand new airframe with minimum constraints, to retrofitting existing aircraft in which the size, shape and quantity of available holes in the avionics bay is highly constraining and in which one or more of the anti-jam CNI functions may already be implemented. This approach implies an interrelationship between the ICNIA system design process and the process of conceiving, generating and establishing standards.

FIGURE 1. A FAMILY OF MODULAR BUILDING BLOCKS

(follows this page)

THE ICKIA PROFILE A FAMILY OF MODULAR BUILDING BLOCKS

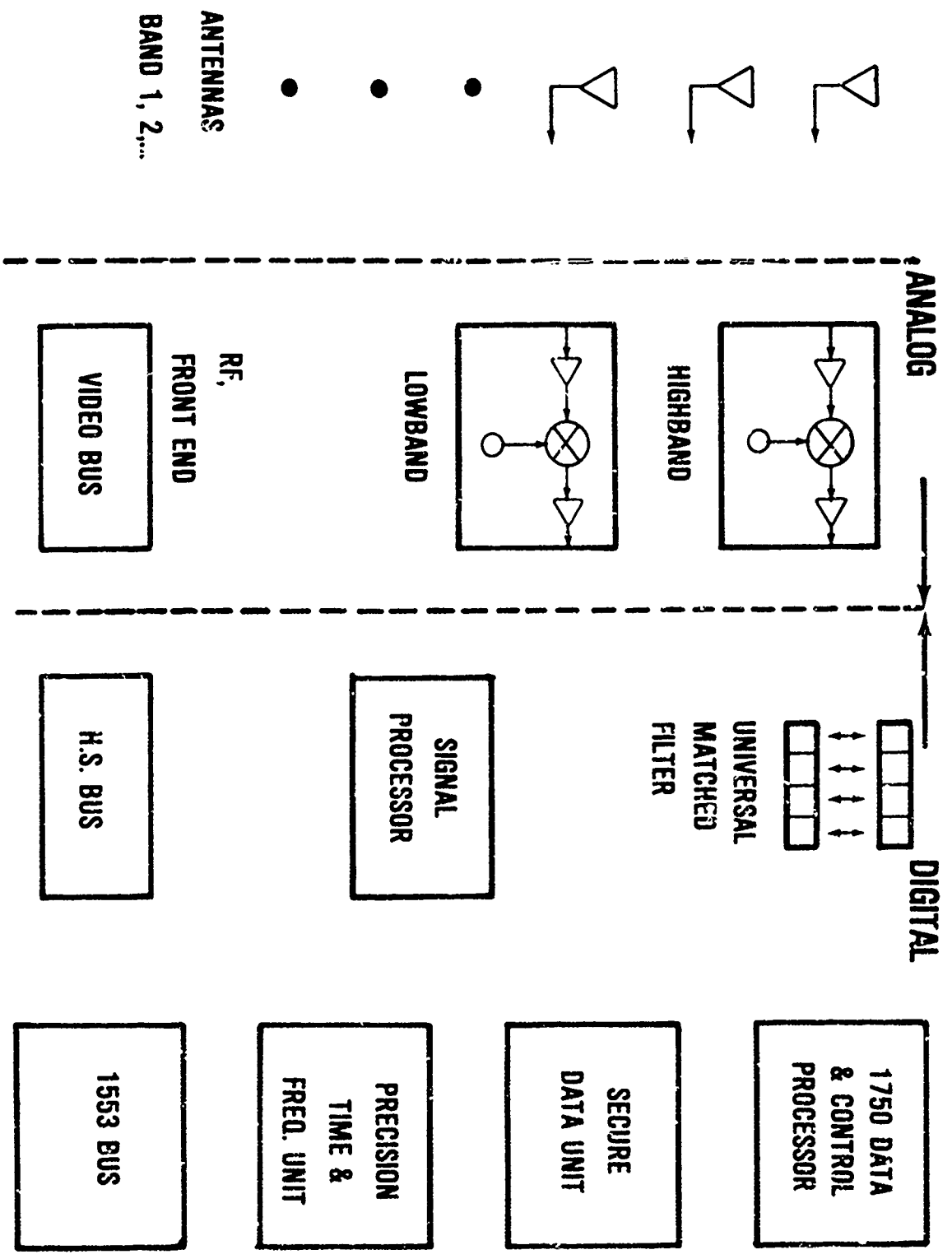


FIGURE 1. A FAMILY OF MODULAR BUILDING BLOCKS

G

II. THE ICNIA PROJECT

1. PROGRAM DETAILS

a. Management. ICNIA is a joint effort of the AFWAL Avionics Laboratory and the U.S. Army Avionics Research and Development Agency (AVRADA), with project office management at Wright Patterson. The establishment of a Navy role in the program is under discussion (Oct 82). ICNIA is now an advanced development program (6.3) to demonstrate the practicality of a modular integrated system design, and to provide a demonstration of both performance and benefits achievable sufficient to warrant continuation through full-scale development into procurement of operational hardware.

b. Approach. Two contracts will be awarded, to design and build demonstration terminals. A decision point at the critical design review provides against the contingency of reduced funding requiring the elimination of one contract, and encourages quality performance by the contractor.

c. Schedule. A draft of the Statement of Work has already been circulated through Government agencies. A revised SOW in a draft RFP has been submitted to industry (Oct 82); a final RFP will go out in January and a contract is anticipated in June or July of 1983. The first demonstration terminal will be delivered three years later followed by approximately two years of laboratory and flight test and evaluation.

2. PRIOR WORK

The first conceptual studies (started in 1978) showed that feasible digital-analog boundaries lay between the signal processor and the data processor, and that analog/RF hardware represented 50 to 80 percent of hardware costs, even when projected 1985 technology was considered. Design approaches therefore emphasized techniques to reduce the cost of the hardware. Two distinct approaches came out of exploratory development: one architecturally the same as that of current independent black box terminals but emphasizing commonality of hardware and exploitation of miniaturization technology such as RFLSI; the other, exploiting a programmable transversal filter concept distinctly different from the super-heterodyne architecture but relying on highly advanced technology. Exploratory technology validation work showed that miniaturization in RFLSI was feasible, in fact inevitable, but that the advanced CCD techniques for the "blue sky" approach are ten years or more in the future. Further iterations on the system design in system definition studies, coupled with arrival of a major DoD thrust in very large scale integration, show that the analog-digital boundary can be moved up to the IF, replacing analog implementation with digital. These design iterations resulted in both approaches converging on a single generic design, which is also compatible with the Tactical Information Exchange System (TIES) developed in parallel at the Naval Air Development Center. This is a result which both encourages and simplifies a tri-service approach.

3. ARCHITECTURE

With the exception of the Microwave Landing System (MLS), all of the CNI functions, which can be found in one or more tactical aircraft and must be implemented in ICNIA hardware, occupy spectrum between 2 MHz and 2 GHz. Clearly

no hardware is likely to cover three decades of frequency, so those building blocks with parameters related to location in the spectrum must consist of more than one type. The number of building blocks in a particular type (for example, antenna) will be determined by practical state of the art rather than the allocation of frequencies to specific functions. Currently, it appears that all signals can be handled in one type of digital preprocessor, provided they are contained in an instantaneous bandwidth less than 20 MHz. For the foreseeable future, all CNI signals will meet this criterion. The architecture is then the interconnection interface, replication of building blocks and control, shown in Figure 2. Each block of the architecture is discussed below, with reference to the family of modules of Figure 1.

GENERIC ICNIA TERMINAL

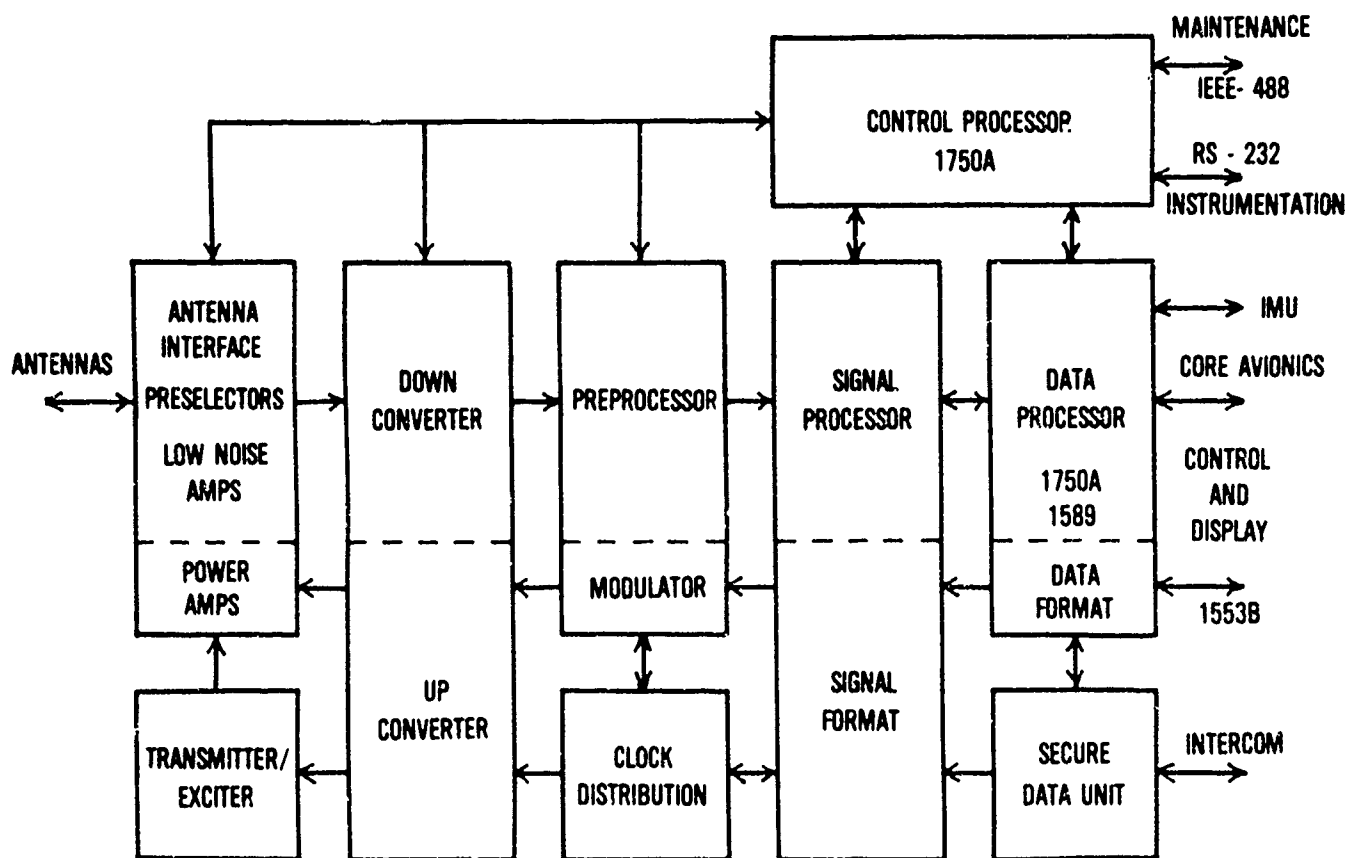


FIGURE 2. ICNIA GENERIC ARCHITECTURE

a. The Antenna Subsystem consists of as many antennas as required: (1) to cover the total spectrum of the specific set of CNI functions for a given platform and its mission (2) provide spatial coverage as determined by platform mission scenario and (3) provide isolation between transmit and receive functions. Currently tactical aircraft carry as many as fifteen CNI

antennas; although some are multi-frequency, most are dedicated to transmission and reception of a specific function. One criterion may be the dedication of antennas to the transmit function with physical separation to isolate transmit and receive functions to provide relief from on-board self interference.

b. RF Front End. The number of unique module types will also be determined by practical frequency coverage achievable, but is not required to be the same as the number of unique antenna types. Within the band covered by each unique module type, each channel in that band will be provided by frequency programming of identical modules. Their number is determined by the requirement for simultaneous (within a microsecond) frequency channels as dictated by aircraft mission requirements within that band. The output of each module is a signal of less than 20 MHz bandwidth, at a standard intermediate frequency and level, with sufficient rejection of signals outside of the desired signal bandwidth.

c. RF Interconnect. This subsystem interconnects separate channels derived from the front end modules to as many subsequent processing stages as are required. It can be implemented either as a matrix switch or as an IF multiplex system if it is found desirable to locate RF resources at the antennas rather than in the avionics bay (as in the signal distribution subsystem developed in the TIES program).

d. Preprocessor. Only one module type is required, a digital preprocessor to handle signals of instantaneous bandwidth up to 20 MHz. One such module must be dedicated to each signal for functions such as correlation of spread spectrum signals of 10 to 20 MHz bandwidth, but may be commuted between two or more signals occupying less bandwidth. The output of each preprocessor is fully digital with rates of 200K samples per second or less.

e. High Speed Digital Interconnect: This will be a high speed digital bus or an interconnect matrix switching network to connect limited bandwidth preprocessor outputs to a number of signal processors, not necessarily the same in number as the preprocessors.

f. Signal Processor: Capabilities projected for development of solid state "devices" using 1.25 micron technology will allow the design of signal processors with initial clock rates of 40 MHz, reaching as much as 100 or 200 MHz by the time ICNIA proceeds to full-scale development. The total processing throughput for the ICNIA currently estimated could be handled by one signal processor with a throughput of 100 MHz. However, fault tolerance and physical survivability require that the processing load be split between two or more modules. This module is fully digital, programmable in time sharing to perform functions of carrier and code tracking, data demodulation and decoding, and additional filtering.

g. Data Processing, Control Processing will be implemented by standard 1750A modules now being developed for embedded micro-processor applications.

h. Precision Time, Frequency Generation and Distribution: With suitable redundancy provided, this becomes the "common clock" subsystem which is a necessity for and can be updated by any of the several anti-jam functions performed within the terminal.

i. Secure Data Unit: This subsystem contains the traditional COMSEC/TRANSEC functions. Negotiations are underway with NSA to have them support the ICNIA program with a determination of the practicality of either an integrated subsystem, or a replication of identical programmable units to perform the crypto functions. The design will be driven by requirements of crypto security.

j. Other Avionics Interface: Design tradeoffs of the ICNIA program stop at the 1553 avionics system bus. Control and display in the form of an interactive graphics terminal will be provided in the demonstration terminal hardware and software for the purposes of test evaluation quick look and rudimentary demonstration of pilot interface. The operational pilot interface design must be driven by pilot, mission and mission sensor, and aircraft requirements. These will come out of the advanced system integration demonstration under PAVE PILLAR.

III. ICNIA AND TECHNOLOGY

1. ANTICIPATE AND EXPLOIT

A firm ground rule of the exploratory development started in 1978 was to project the technology which might be available in 1985 and exploit it in generating innovative concepts. This included technology in all senses: material, device, circuit, module, methodology and "science" of software, and design procedures both theoretical and practical. Device technologies initially anticipated have been validated in one case, invalidated in another. During this process, the DoD thrust in solid state technology was established, allowing the technical horizon of the project to be raised. "Looking ahead" is a continuing process; design effort is based on technology which can be expected to be mature at the time of transition to full scale development in 1988. The decision is then determining what technology can be applied to demonstrate that an integrated system is the best approach. Architectural tradeoffs in developing the interface structure will be made to enhance the attributes of technology transparency, ease of technology insertion or pre-planned product improvement. One of the tools of design for ICNIA is the standardization process.

2. STANDARDIZATION TECHNOLOGY

Under the broad definition of technology in the preceding paragraph, standardization is an evolving technology. Earlier approaches were frequently after the fact, defacto standards, based on requirements that the technology be mature, that the application be widespread and that the market justify standardization actions. In the ICNIA program we see standardization as being a process which must be a relatively quick reaction to galloping technology, to enable many designers to exploit new technology rapidly in system design, not a process which takes a long time to mature and is then cast in concrete to endure forever. At the current pace of technology we will be fortunate indeed if a particular standard approach survives more than two generations of technology. Standardization must be anticipated, developed and exploited, with the thought that standards have a dynamic characteristic and must evolve, rather than emerge full blown from a long drawn-out effort to create rules which will endure forever. In this light, ICNIA, with many other programs, anticipates the need for certain standards which are discussed below.

IV. ICNIA AND STANDARDIZATION

1. ICNIA APPROACH TO STANDARDIZATION

There is little in technology that is unique to ICNIA; even the RF technology shares theory and practice with applications in radar and electronic warfare. The ICNIA signal processor breadboard technology validation is in fact based on chips developed for an EW breadboard, with only one new chip (a universal digital matched filter) specifically for ICNIA. Standards applicable to ICNIA thus have potential for other avionics. This is a desirable condition, since standards have their greatest payoff in a large military "market", but one which makes its own problems in creating and coordinating sensible standards. ICNIA thus does not intend to promulgate standards, but the project principle of anticipating and exploiting technology wherever or by whomever generated extends readily to working with organizations with mutual interest in a common standard. ICNIA expects to make modest allocations of project resources to leverage other efforts in technology for the ultimate benefit of the project. A wide variety of standardization efforts can be crowded under this umbrella, limited only by the necessity of not dissipating effort over more areas than can be managed.

a. Integrability is the Payoff from Standardization. This presentation itself is advertising the willingness of the project to cooperate, in selected areas, to define widely applicable standards, in data and signal processing, and bussing of all kinds. The use of standards internal to the ICNIA system as defined enhances the ability to tailor ICNIA development to application: in some aircraft, the entire architecture may be applicable; in others, data processing may be incorporated in existing core avionics; conceivably, there is an application in which even the signal processing occurs in a core system. The 1553 bus, high speed data bus and analog (video or RF) standards built into ICNIA will provide this desired freedom of technology integration. These and other areas of future standardization with payoff potential for ICNIA are discussed in the following paragraphs.

2. STANDARDS: TODAY

When the ICNIA concept development was started, the DAIS program constituted a state of the art "core" of an integrated avionics architecture. DAIS provided the basis for the 1553 Bus, 1750 Instruction Set Architecture, and 1589 Higher Order Language standards. These, and the proposed new Packaging, Mounting and Environmental (PME) standard (LRU) constitute currently required standards within the ICNIA program. Since ICNIA started as an Avionics Laboratory program, it was natural to accept our own standards because of the familiarity of division personnel with the standards and because of the existence of the AVSAIL facility based on JOVIL and 1750 architecture. While the programming of the ADM will probably be executed in J73 (MIL Standard 1589), consideration is being given to adoption of Ada, certainly in full scale development if not in the ADM. The 1553 bus standard is adequate for the high level interface at which a full ICNIA subsystem will be integrated with other avionics. However, other partitionings of the ICNIA technology require new standards, and some are already expected.

3. STANDARDS: EXPECTED WITH CONFIDENCE

In the area of information transfer across functional interfaces the need has already been recognized for such standards as a high speed digital bus and a video or RF bus compatible with ICNIA. These are topics addressed in the charter of the newly formed SAE, AE-9 Committee on Aerospace Equipment and Integration. Standards are required in organization, protocol and control and for the implementing technology. While these busses are essential in ultimate application and desirable in demonstration, they are not mandatory now. When the time comes for design decisions involving the interface definitions in ICNIA, design contractors in conjunction with Government engineers will make a best guess about how these standards will finally be emerge and proceed with the building and programming of the system.

4. STANDARDS: EXPECTED, SOMETIME

The ICNIA concept of a family of modules from which system integrators, airframe builders and System Program Offices can select to tailor a CNI suite to a specific set of air frame and mission requirements, from scratch or as a retrofit, implies that each such program may generate a unique configuration of black boxes to implement ICNIA. Logistically, this is intolerable, unless the uniqueness is limited mostly to particular combinations of part numbers contained within standardized PME boxes. For most benefit in application of the concept, a "standard module" approach is required; the parts, cards or other subassemblies used must conform to form, fit and function whether the application is in aircraft or helicopter, in Army, Navy or Air Force platforms, possibly even in ground installations. In a previous session in this conference, the Navy Modular Avionics Packaging program has been discussed. This offers not only a physical form and fit, thermal interface standard proposal, but an alternative approach to even the traditional black box in the "Integrated Rack" approach. For the current state of complexity of the components in such modules, undesirable compromises are still necessary between the number of connector pins required for interconnect (large) and the thermal interface between card and heat transfer mechanism within the aircraft. However, new concepts of architecture are being offered which suggest that the ultimate form of information transfer, even between cards within a module, will occur completely through bus interconnects. These are the "maybes" of future standardization, and will be anticipated with interest, but limited expectation at this stage of the ICNIA project.

V. IN SUMMARY

ICNIA is an ADP with its prime thrust the demonstration of an alternative architecture to implement anti-jam CNI functions in manned aircraft. Studies point to an integrated, modular design, a "family of building blocks" which will require extensive application of standardization, in major assemblies and in components, in interfaces and in protocol. The need for such standardization is not unique to CNI functions, but extends across all avionics of the future; the ICNIA project will join with other organizations having mutual interests in the definition and development of new standards for the future.

ARCHITECTURE, HARDWARE AND SOFTWARE ISSUES
IN FIELDING THE NEXT GENERATION DOD PROCESSORS

Ole Golubjatnikov

General Electric Company
Undersea Electronics Programs Department
Syracuse, New York 13221
(315) 456-4744

Ole Golubjatnikov has been associated with the computer field since 1950. As an undergraduate student at the University of Illinois, where he received his BSEE and MSEE degrees, he worked in ORDVAC and ILLIAC computer developments.

During his professional career, he has been involved in the development of 12 military and commercial computers and signal processors, and scores of military and commercial embedded computer systems applications. The computer developments include the General Electric M236, M605, GE600, FFP, the Honeywell H6000 and the Army Military Computer Family (MCF). Typical computer applications include Atlas Guidance System, Project MAC, MULTICS, WWMCCS, CEDEC, Apollo Central Instrumentation Facility, American Stock Exchange, GE Timeshare System and Genigraphics. He has participated in the architectural design of the GE solid-state radar family, including the AN/TPS-59 radar and the AN/SQR-19 and AN/SQS-53C sonars.

From 1970 to 1977, he was president of COMPTA and an independent consultant specializing in computer architecture and embedded computer system applications for commercial and military applications. He is currently responsible for computational architecture for shipboard ASW systems. He is active in IEEE, Computer Society, NSIA and EIA and is currently coordinating the EIA review of the Joint Logistics Commanders (JLC) MIL-STD-SDS software development specification set.

ABSTRACT

Programmable hardware will be pervasive in DOD mission-critical embedded computer systems exploiting technology advances to meet unsatisfied needs. The embedded processors are absolutely critical in maintaining the superiority of our mission-critical systems and weapons. They provide a major advantage for mission-critical system performance and operational readiness. There are, however, major obstacles to achieving these advantages.

Commercial computer architectures and implementations are constrained by market considerations while military computers and custom-designed processors generally lack comprehensive software environments and are deficient in field upgradeability and support and have high life-cycle costs. The proliferation of computer types has aggravated the explosive DOD software cost growth. DOD is providing major processing architecture, hardware and software initiatives for fielding the next generation processors in support of their mission-critical

systems to overcome these problems, to contain the explosive software cost growth and to more fully exploit the computer advantage.

The paper examines the major architectural, software and hardware issues associated with the development of the next generation DOD processors. The impact of DOD mission-critical computer (MCC) architecture, hardware, software and policy initiatives on the Army MCF, the Navy TECR and EMSP, and the Air Force MIL-STD-1750 and MIL-STD-1862 processors is explored in a comparative manner.

The paper concludes that the next generation DOD MCCs are driving the military and commercial state-of-the-art in signal processing and the military state-of-the-art in scalar processing. In the case of one MCC, it may actually match or exceed the commercial processing state-of-the-art. The second major conclusion is that MCC policy initiatives should recognize the best balance between commercial and military computer technologies within the constraints of wartime survivability.

I. INTRODUCTION

The DOD mission-critical embedded computer costs are projected by the Electronic Industries Association (EIA) (reference 1) to explode during this decade from \$4.1 billion in 1980 to \$38 billion in 1990. Eighty-five percent of this cost explosion is projected to be MCS software. These costs will be driven by DOD automation needs and will be restrained by advances in computer technology and DOD mission-critical computer technical and management initiatives triggered by the Johns Hopkins and Mitre studies (reference 2) done in the spring of 1975.

The DOD automatic data processing (ADP) budget is projected by EIA (reference 1) to grow from \$2.6 billion in 1980 to \$7.8 billion in 1990. The DOD ADP cost growth will be eased by reduced hardware acquisition and support costs, achieved through increased competition and exploitation of VLSI technology, as well as reduced software development costs, achieved through increased software productivity. These gains will not offset the accelerated growth in software costs associated with the expanding applications which can be cost-effectively supported in programmable hardware and rising labor costs.

While the commercial computer technology is generally adequate to support the DOD ADP requirements in the 1980s, the DOD mission-critical computers present a number of DOD and specific-service-unique requirements. These unique requirements have their roots in the military operational needs, the limited size of the MCC market, the rapid obsolescence of computer hardware, the mission-critical system development and field support process, and the procurement and standardization practices of the specific services in the past.

The commercial computers are constrained by the need to provide upward compatibility for their installed base (and the associated long-term obsolescence of their single-source ISA), the market potential for the new product in a competitive market (volume considerations) and the proprietary considerations, and cannot respond to all modern software and major DOD and service-unique requirements. The traditional military computers and custom-designed processors offer performance advantages in the specific application for which they were developed. They represent significant product maturity, cost and schedule risks on programs for which they were developed. Once developed, they are generally produced in small quantities, lack comprehensive software environments, are

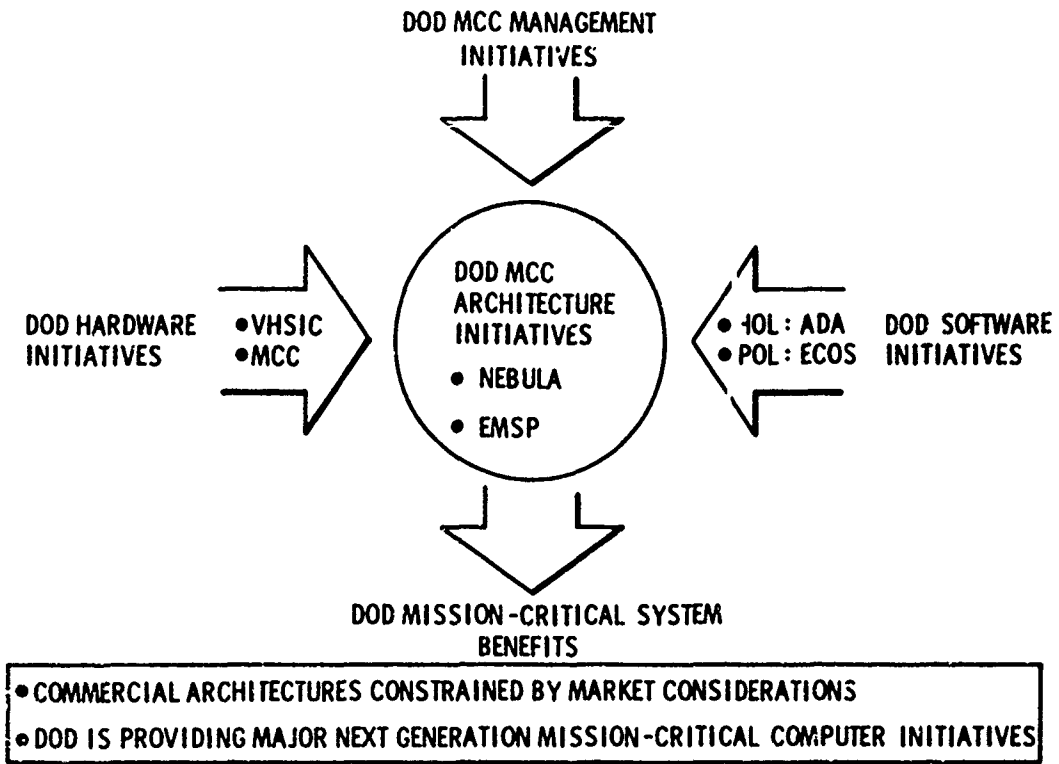


Figure 1. DOD Processing Initiatives Produce MCC Benefits

single source and deficient in evolutionary upgrades, are expensive to support in the field and contribute to the overall MCS survivability problem (reference 3).

To restrain the exploding software costs, to overcome the problems described above and to advance the state-of-the-art in military computer technology, the DOD has initiated a number of MCC policy, architecture, hardware and software initiatives. The architecture initiatives are centered on ISAs and a number of bus, peripheral and interface standards, many of which are described in the tutorials and papers of this conference. The hardware initiatives are focused on the VHSIC program and the application of commercial VLSI/VHSIC technology to military computer implementations. A full recognition of the acquisition policy significance of the separation of ISA standardization from hardware implementation standardization is evolving. The DOD software initiatives have as their objective the increased productivity of personnel resources through increased use of improved tools (reference 4). The DOD software initiatives are focused on the Ada* program and include Ada programming support environment (APSE). Other software initiatives include, among others, the development of MIL-STD-SDS and the establishment of a Software Engineering Institute. The MCC policy initiatives include DODI 5000.29, DODI 5000.31, DODI 5000.5X, the draft JLC Joint Policy on the Management of Computer Resources in Defense Systems and the specific service MCC regulation and instructions. The paper examines the MCC architecture, hardware and software issues and the impact of DOD MCC initiatives on the next generation DOD mission-critical computers. The salient standards and acquisition strategy of the next generation Army, Air Force and Navy MCCs are summarized in Figure 2.

The Army Military Computer Family is based on the NEBULA (MIL-STD-1862) ISA and Ada (MIL-STD-1815) HOL (higher order language) standards. The three fully compatible members of the MCF hardware implementation are the embeddable single board computer, the AN/UYK-49 microcomputer and the AN/UYK-41 superminicomputer. The Air Force has standardized on MIL-STD-1750 ISA and JOVIAL (MIL-STD-1589) for its 16-bit mini- and microcomputer needs and NEBULA (MIL-STD-1862) and Ada (MIL-STD-1815) for its 32-bit supermini/mainframe needs. A large number of suppliers are developing or producing the MIL-STD-1750 computer. The 32-bit supermini/mainframe is in the planning phase. The Navy has an extensive inventory and software investment in the previous generation AN/UYK-7 and AN/UYK-20 computers and CMS-2 HOL. The AN/UYK-44 is a next generation hardware implementation of the popular AN/UYK-20 ISA. The AN/UYK-43 is a next generation implementation of the AN/UYK-7 mainframe. The airborne implementation of the AN/UYK-20 ISA is the AN/AYK-14. The Enhanced Modular Signal Processor (AN/UYK-2) is the Navy's next generation signal processor incorporating the AN/UYK-44 as its control processor. The AN/UYK-2 is utilizing SPL-1 as its HOL and ECOS as its signal processing POL (problem-oriented language).

The Air Force MIL-STD-1750 is procured on a program-by-program basis with a number of potential competitors competing for each procurement. The Army and Navy next generation computers are competitively developed and will be produced on a commodity basis by the winning production contractor. All of these computers, except for the AN/AYK-14, are still in development with a number of manufacturers with different hardware implementation approaches still competing.

*Ada is a trademark of the Department of Defense.

SERVICE	PROCESSOR	ISA STD		IMPLEMENTATION STD	HOL/POL STD	ACQUISITION STRATEGY	COMPETING CONTRACTORS
		16-BIT	32-BIT				
ARMY	SINGLE BOARD	—	MIL-STD-1862	EMBEDDED	MIL-STD-1815	14-4-2-1	{ RAYTHEON, RCA, GE }
	MICRO	—	MIL-STD-1862	AN/UYSK-49	MIL-STD-1815	14-4-2-1	
	SUPERMINI	—	MIL-STD-1862	AN/UYSK-41	MIL-STD-1815	14-4-2-1	
AIR FORCE	MICRO/MINI	MIL-STD-1750	—	MANY	MIL-STD-1589	ISA-OPEN	MANY
	SUPERMINI	—	MIL-STD-1862	TBD	MIL-STD-1815	TBD	
NAVY	MICRO/MINI	AN/UYSK-20	—	AN/UYSK-44	CMS-2 (ADA)	2-0-2-1	IBM, UNIVAC
	MINI	AN/UYSK-20	—	AN/UYSK-14	CMS-2 (ADA)	1	
	MAIN FRAME	—	AN/UYSK-7	AN/UYSK-43	CMS-2 (ADA)	2-0-2-1	IBM, UNIVAC
	SIGNAL PROCESSOR	EMSP	(EMSP)	AN/UYS-2	SPL-1/ECOS	7-5-1-1	WESTERN ELECTRIC

NOTE: ACQUISITION STRATEGY KEY (W-X-Y-Z): NUMBER OF CONTRACTORS (TEAMS) DURING (PROPOSAL) - (ADM) - (FDM) - (PRODUCTION)

Figure 2. Mission-Critical Computers Defined

Therefore the comparative MCC analysis in this report is conducted utilizing specified design goals, rather than the actual production contractor product characteristics. The MCC design goal parameters are assessed as to their responsiveness to DOD hardware and software initiatives and their competitiveness against the projected 1985 commercial computer state-of-the-art.

II. MISSION-CRITICAL SYSTEMS DEFINED

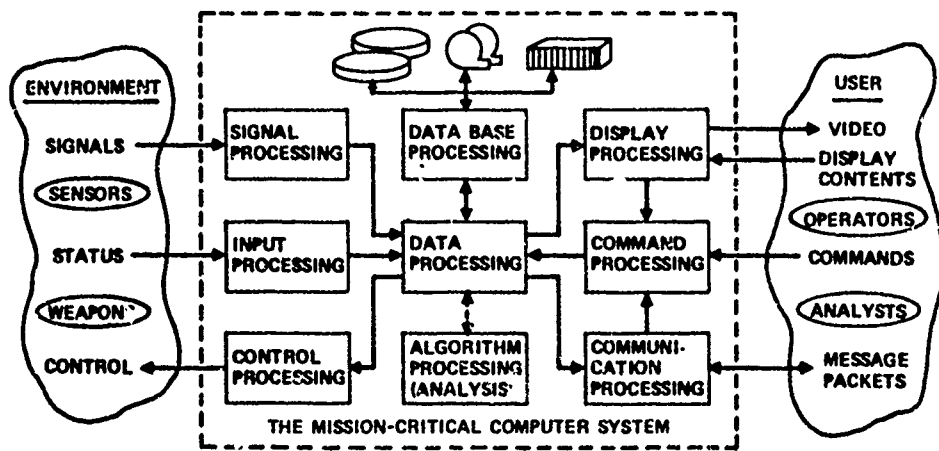
Virtually every sophisticated system in the current and planned military inventory makes extensive use of computer technology. Computers are integral to our strategic and tactical capabilities: they are at the heart of carrier battle group defense, they control the targeting and flight of missiles, they control and coordinate the systems within a high performance aircraft, and they integrate the complex activities of battlefield command. All of these MCS can be functionally characterized by the generic functional structure of Figure 3.

Of all the computer resources within a given defense system, some will be mission critical in the sense that successful performance of the intended mission depends on them daily. All mission-critical resources are embedded in the sense that they are considered within the "boundary" of the mission-critical system and provide functions which are integral to its proper operational functioning. Of these, some will be directly involved in system mission-operation but others will be mission-support in terms of providing specialized forward or base-level maintenance and logistics. There are other non-embedded computers which are also mission-support in the sense that they are applied primarily to a given mission for supply or rear-area maintenance. Outside every system will be a variety of general support systems providing standard logistics functions, financial management, etc. The computers for general support are likely to be commercial-soft machines in the sense that they have been designed for installation and use in a fixed, well-controlled physical environment and are intended for general commercial marketing by their vendors. These machines are specifically excluded from the considerations of this paper.

The mission-support computers may be commercial-soft or military computers, depending on the system physical environment and the overall system architectural and life-cycle support requirements. Finally, mission-operation computers are generally military computers or they might be mission-custom-designed processors for a very particular purpose. These and other mission-critical computers operate in demanding MIL-Spec environments; they are part of mission-critical systems, platforms or battlefield networks where survivability and logistics support requirements are most demanding. These mission-critical computers are the primary focus of this paper.

The generic functional structure of a wide variety of mission critical systems is depicted in Figure 3 so that their operational environment and processing functions can be characterized. The embedded computer system environment consists of a wide variety of sensors and actuators (weapons) requiring signal, control and status processing. The inputs are frequently uncontrolled and the control response is time-critical. The user environment is often complex, consisting of operators and analysts digesting data base and real-time information from the environment in a decision translation function.

Programmable hardware will be pervasive in these mission-critical systems. The processing functions can cover the complete spectrum from signal, input, control, status, algorithmic analysis, data base, display, command and



THE ENVIRONMENT

- INPUT UNCONTROLLED
- CONTROL RESPONSE CRITICAL
- DIVERSE SENSORS AND WEAPONS

THE PROCESSING

- FUNCTIONAL PARTITIONING
- EXPANDING USE OF GENERAL PURPOSE PROGRAMMABLE LOGIC
- UNIQUE DEMANDING REQUIREMENTS

THE USER

- INFORMATION CONTENT
- DECISION TRANSLATION

PROGRAMMABLE HARDWARE WILL BE PERVASIVE IN DOD MCS DUE BOTH TO NEEDS AND TO TECHNOLOGY

Figure 3. Mission-Critical System Functional Definition

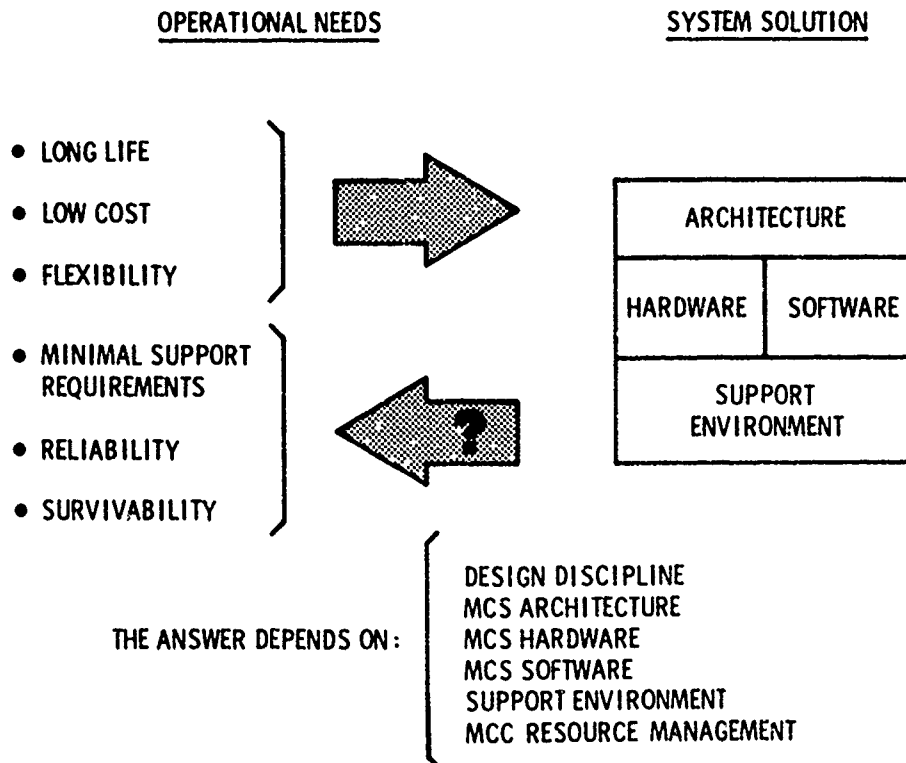


Figure 4. Can Mission-Critical Computers Meet Operational Needs?

communications processing. These functions will require expanding use of general purpose programmable logic. The programmable logic may be implemented in: 1) mask programmable, 2) microprogrammable, 3) assembly language programmable, 4) higher order language programmable or 5) problem-oriented language programmable implementations.

Programmability and software are the essential elements that control, even define, the system. Software is the embodiment of the system "intelligence." It provides the flexibility to respond to changing threats, missions, needs and requirements. The problem is that programmability and software also pose a host of difficulties that hinder realization of the full benefits possible from the advances in computer hardware technology. Development and support of software for major military systems are among the most complex human endeavors. Such development requires the resolution of complex system issues through system engineering and software development activities that are poorly defined and not well understood. The state of practice in DOD and industry ranges from effective, disciplined approaches in some systems to near chaos in others. The difficulty of software development is aggravated in situations where software support environments do not exist and concurrent processing equipment development is required.

The DOD mission-critical computer problem requires system level DOD initiatives and solutions as depicted in Figure 4, and cannot be met by processor "black box" level standardization solutions used during the 1970s. The basic question is: can DOD programmable systems exploit the VLSI-driven computer technology and meet operational needs of 1) long life, 2) low cost, 3) flexibility, 4) minimal support requirements, 5) reliability and 6) survivability? The answer depends on: 1) system design discipline, 2) MCC architecture, 3) MCC hardware implementation, 4) MCC software design discipline, 5) MCC support environment and 6) MCC resource management.

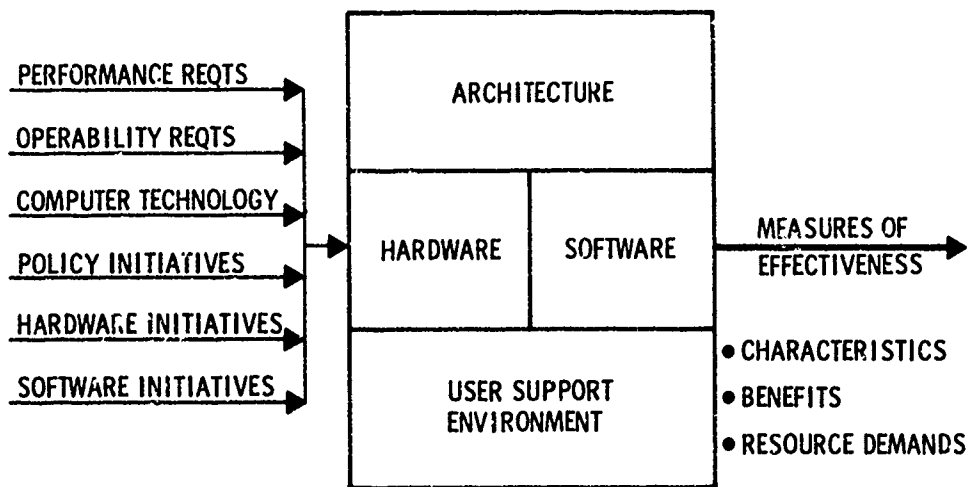
III. A STRUCTURE MODEL FOR MCC ISSUES

The primary objective of this section is to identify the major issues related to mission-critical embedded computers and to determine their interrelationships and provide a structured basis of discussion of these issues in the subsequent sections as to their impact on the development of the next generation DOD mission-critical embedded computers.

The issues related to mission-critical embedded computers are complex, span a wide range of technical and management disciplines and are highly interrelated (reference 5). A simplified MCC processing issue model is proposed in Figure 5. It identifies the major structural elements of MCCs and the major model input drivers and the output measures of effectiveness.

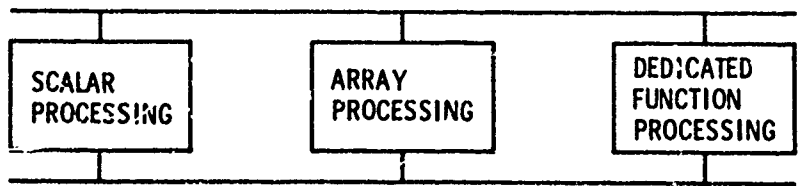
The major MCC structural elements are architecture (ISA), hardware implementation, software implementation and user support environment. The MCC issue model input drivers include a wide range of performance and operability requirements, the commercial computer technology and the DOD initiatives in architecture, software, hardware and policy.

The model output measures of effectiveness (MOE) are the MCC characteristics, user benefits and resource demands. A simplified set of output measures of effectiveness is later described for the Army MCF AN/UYK-41 and the



PROCESSING ISSUE MODEL PROVIDES A STRUCTURE FOR MCC COST BENEFIT AND TRADEOFF ANALYSIS

Figure 5. MCC Issue Model



- PERFORMANCE REQUIREMENTS
 - WIDE SPECTRUM OF COST-EFFECTIVE PERFORMANCE
 - PROCESSING PARALLELISM
 - EXPANDING SPECTRUM OF PROGRAMMABILITY
 - NEW HOL AND COMPILER REQUIREMENTS
- OPERABILITY REQUIREMENTS
 - COMBAT SYSTEM ARCHITECTURE REQUIREMENTS
 - FAULT TOLERANCE AND OPERABILITY REQUIREMENTS
 - MILITARY ENVIRONMENTAL REQUIREMENTS
 - EXPANDING SUPPORT ENVIRONMENT REQUIREMENTS

MCS PROCESSORS PROVIDE A MAJOR LEVERAGE FOR MCS MISSION PERFORMANCE AND OPERATIONAL READINESS

Figure 6. Spectrum of MCS Processing Requirements

Navy EMSP (enhanced modular signal processor) AN/UYS-2 processors, reflecting the projected mid-1980s state-of-the-art in military computers. The proposed 12 MOEs include:

- 1) Memory capacity
- 2) Performance (processing speed)
- 3) HOL compatibility
- 4) ISA compatibility
- 5) Support environment
- 6) Combat system architecture (CSA) compatibility, including I/O and peripheral equipments
- 7) Life cycle cost
- 8) Environmental ruggedness
- 9) Availability/reliability
- 10) Maintainability
- 11) Weight/volume
- 12) Power consumption.

A number of the above MOEs are similar to the commercial computer design objectives. Many others such as military support environment, combat system architecture, environmental ruggedness, life-cycle cost, power consumption and weight/volume are directly related to unique military mission-critical system requirements and are not the primary design tradeoff parameters in the development of commercial computer products.

IV. MCC DESIGN IS DRIVEN BY A WIDE RANGE OF PROCESSING REQUIREMENTS

Mission-critical systems impose a wide spectrum of processing and operational requirements for the DOD mission-critical computers, as summarized in Figure 6. The processing requirements can be categorized as a function of algorithms and data structures, as scalar processing, as array processing or as dedicated function processing. The processing rates for scalar processing can range from low KIPS (thousands of instructions per second) to tens of MIPS, while the array and the dedicated function processing rates can range into BIPS (billions of instructions per second). The storage requirements for simple applications may be satisfied by a few hundred locations, while large data base and simulation systems may require tens of millions of words of on-line storage. Similarly, I/O requirements may range from a few channels to hundreds of I/O channels.

The scalar processing requirements are generally satisfied by the general purpose computers loosely categorized into subfamilies based on cost performance considerations: microprocessors, microcomputers, minicomputers, superminicomputers, mainframe computers and supercomputers. The array processing functions are frequently performed in supercomputers, array processors and custom-designed hardware. The dedicated function processing has been traditionally performed in MCS-unique custom-designed hardware, although increased performance and device integration has permitted the use of ROM-driven special purpose processors in many functions.

VLSI, microprocessors and the associated trends towards increased parallelism and programmability are changing the architectural approach to array and dedicated function processing. These specialized processors are beginning to approach the degree of sophistication and user support assumed commonplace in

scalar processing, as is vividly demonstrated by the Navy EMSP procurement. Therefore, the next generation DOD processors must provide new HOL language, compiler and support tools capable of supporting architectures containing a high degree of distributed, parallel and pipelined processing. The new DOD language Ada with extensions appears to be particularly well suited to support the VLSI-based signal and dedicated function processors.

The wide range of DOD MCC performance requirements, and the opportunities presented by the VLSI technology for flexible and cost-effective solutions to these requirements, are major DOD and industry challenges.

V. MCCs MUST MEET DEMANDING OPERATIONAL REQUIREMENTS

Mission-critical computers are an element of the mission-critical system; they must be supportable, upgradeable, survivable and affordable. It is in the area of these operational requirements where significant and fundamental differences between the commercial and military computers exist. Typical priority ranking for a military computer's (e.g., MCF) design and tradeoffs is:

- 1) Reliability and maintainability
- 2) Life cycle cost and power
- 3) Size and weight
- 4) Speed and memory capacity.

An almost reverse order of priorities drives the design of commercial computers. The commercial OEM computer design is dominated by the in-house ISA compatibility (in-house standard) to support upgrade of existing customers and the competitive cost/performance considerations.

Similar to the commercial computer vendor or to the large corporate user, the issue of ISA standardization is absolutely crucial to the military as the benefits of supportability, upgradeability, survivability and affordability directly flow from the standardization strategy. In the battlefield and remote ship platform environments, a further standardization at the hardware implementation level is indicated to further benefit from the supportability, upgradeability, survivability and affordability savings and considerations.

The emphasis on reliability and maintainability for military computer requirements come, of course, from the fact that they must work not only in peace time, but also during the battle. This overall requirement precipitates numerous subrequirements, as described below.

Very high levels of ruggedness, reliability and maintainability must be inherent in the computers by design. The basic standards specifying the computer ruggedness requirements are MIL-STD-4158, MIL-STD-5400 and MIL-STD-16400. Typical environmental requirements include operating temperature (-54°C to 71°C), nonoperating temperature (-62°C to $+85^{\circ}\text{C}$), altitude, temperature shock, humidity, rain, salt fog, sand and dust, solar radiation, fungus, explosive atmosphere, shock (e.g., 40Gs for 11ms), vibration, acceleration, EMI, TEMPEST, and radiation hardening.

Mean time between failures (MTBF) for military computers typically ranges from 5,000 hours to 100,000 hours. MCCs (mission-critical systems) may specify system availability requirements beyond the single unit reliability numbers, requiring fault-tolerant multiprocessor configurations and special fault

detection and automatic reconfiguration capabilities.

Graceful degradation and interchangeable hardware in order to restore system operation without replacement or other maintenance actions are fundamental requirements in mission-critical systems consisting of 10 to 30 mission-critical computers. The issue of commonality and programming support are critical to reduce field support costs. These issues are being addressed by the Army MCF and the Navy EMSF programs.

MCC maintainability requirements are driven by the service maintenance environment and the difficulty of training personnel for servicing complex electronic systems. Therefore, it is generally required that MCCs contain a high degree of built-in-test (BIT) circuits, so that typically 98% of all faults are detected automatically, with single removable unit isolation requirement of around 95% of the time.

In addition to unit level maintenance, mission-critical system level maintenance is required for complex multiprocessor systems, remoted systems and distributed processing networks in the 1980s. The maintenance of distributed MCCs is facilitated through I/O interfaces, special maintenance and remote operation instructions, and the ability to remote fault detection and isolation operations.

A high degree of hardware implementation level standardization by Army and Navy MCC acquisition strategies provides for availability of spare units and spare parts in the combat environment. If spare units and parts are not available, faulty units can be replaced by operable units from lower priority systems or functions.

Finally, in the area of reliability and maintainability, MCS life-cycle longevity is extended in unsatisfied applications with growing or changing processing requirements through technology insertion strategies and the ISA level standardization. Because of resulting implementation-dependent differences between new and prior generation units, system-level revalidation testing will be required during system upgrades with next generation equipments.

Life-cycle cost is the second major MCC consideration. In a typical MCS, the life-cycle support costs frequently exceed the development and acquisition costs. Cost benefits of ISA and hardware level standardization accrue mostly in the following cost elements: production, software development, hardware development, spares, training, support and test equipment. The ability to produce a larger volume of standard hardware accounts for a significant portion of these potential savings because of the learning curve effect and the amortization of non-recurring hardware and software costs. Additional potential savings are realizable through logistics commonality of field-replaceable items, when multiple applications are deployed on a platform or in a system. The extent to which these potential savings are eroded in practice depends on a number of important factors that are not typically considered in life-cycle cost analysis, but which are. These include the use of products with obsolete technology, force fitting applications to existing products, etc.

With respect to power consumption, size and weight, these are not major considerations in commercial computer designs. In contrast, in military MCC designs these are frequently the most difficult characteristics to meet and

require tradeoffs with performance, cost and reliability.

Processing speed and memory capacity are device-technology driven. While fundamental design parameters in military computer design, they are frequently traded off to achieve the specified operability requirements.

VI. TECHNOLOGY DRIVES TO INCREASED MCC PROGRAMMABILITY AND PARALLELISM

To maintain its supremacy in defense systems, the DOD must extract the best processing technology available from the U.S. commercial/industrial base. This processing technology is complemented in certain areas not readily available from the commercial marketplace by the DOD hardware and software initiatives. The DOD VHSIC program is the primary DOD hardware initiative.

Digital device technology speed and density have been doubling approximately every three years and this rate is expected to continue. The advances in device technology result in directly related decreases in computer power, weight and cost performance and increases in reliability. While the device technology speed and density are expected to increase, the computer backpanel speeds are approaching the limits of physical constraints, indicating the need for new architectural approaches based on increased parallelism and programmability.

The mix of IC products used in computers is tending strongly to favor VLSI-LSI components. The custom, customizable and commodity VLSI-LSI components all are impacted by the increasing speed and density of the VLSI technology. In 1975, typically 25 percent of semiconductor products in computers were LSI-based. By 1980, the VLSI-LSI categories accounted for 75 percent and by 1985, approximately 90 percent. Not only is the VLSI-LSI content in computers increasing, but so also is the density of devices, providing for additional functionality and test features. The DOD VHSIC program is primarily targeted to signal processing and is projected to significantly impact the next generation DOD signal (array), dedicated function and scalar processors.

The increase in programmability and parallelism is the direct result of VLSI economics. It occurs at a number of different levels, as summarized in Figure 7.

The VLSI economics dictate that digital systems be programmable so that a minimum set of circuits can be used in the largest possible set of applications to offset the high non-recurring costs associated with VLSI device design. This result of VLSI economics is readily evident in the wide use of the programmable microprocessors in commercial and military applications.

Programmability provides flexibility for a wide variety of applications, as well as the ability to respond to changes in threat and mission in MCSs. Programmability is also a major problem in MCSs, as it tends to increase the need for highly skilled configuration management people and results in high software development and support costs. These negative factors are further aggravated by the high turnover rate of military personnel.

Programmability in MCCs can be used at several different levels. Mask programmability is used in custom-programmable circuits. Microprogrammability is used for hardware level control of instruction execution where a number of microinstructions are used for the execution of a single machine or assembly language level instruction. In a machine using parallel or pipelined

- | | |
|--|--|
| <ul style="list-style-type: none"> •MICROPARALLELISM •FUNCTIONAL PARALLELISM •PROCESSOR PARALLELISM | <ul style="list-style-type: none"> •MASK PROGRAMMABILITY •MICROPROGRAMMABILITY •VERTICAL, OR ASSEMBLY LANGUAGE, PROGRAMMABILITY •HOL PROGRAMMABILITY •POL PROGRAMMABILITY |
|--|--|

VLSI WILL INCREASE HARDWARE PARALLELISM AND PROGRAMMABILITY AS THE ONLY PRACTICAL MEANS FOR INCREASED PERFORMANCE AT REDUCED COST

Figure 7. VLSI Impact on Processing Architecture

- | | |
|---|---|
| <ul style="list-style-type: none"> •EXPLOIT VLSI POTENTIAL <ul style="list-style-type: none"> - PARALLELISM - PROGRAMMABILITY - TECHNOLOGY INSERTION •REDUCE LIFE CYCLE COSTS <ul style="list-style-type: none"> - APPLICATION MATCH - DEVELOPMENT - MAINTENANCE - RESPONSIVENESS TO APPLICATION CHANGES | <ul style="list-style-type: none"> •PORTABILITY <ul style="list-style-type: none"> - PROGRAMS - DATA - PROGRAMMERS •REDUCE CUSTOM HARDWARE •PROGRAMMER PRODUCTIVITY •INTEROPERABILITY •REUSABLE SOFTWARE |
|---|---|

HOL STANDARDIZATION PROVIDES MAJOR DOD MCS BENEFITS

Figure 8. Why a Modern Higher Order Language?

architecture, a number of microinstructions may be in concurrent or phased-concurrent execution, grossly complicating the programming problem. The chance of error in programming at this level is quite high and support tools are essential. Assembly language programming assigns mnemonics to encoded bit patterns in a computer instruction word and provides an assembler, significantly reducing the difficulty in machine programming.

The above programming levels are tedious at best and present a challenge to the programmer. The use of HOL and POL speeds up the programming process and improves code reliability. The impact of a modern HOL on MCC design is discussed in the next section.

The economics of VLSI also dictate increased levels of parallelism in MCC design. The parallelism occurs at all three levels -- micro, functional and processor. The use of parallelism in MCC architectures reduces non-recurring costs and results in significant production economies.

VHSIC devices are expected to differ from those that will be available in the commercial market in two important aspects. The main emphasis of the program is on the development of units for signal processing. MCSs (e.g., radar, sonar, electronic warfare and imaging) require fast real-time processing of signals. Limited numbers of less demanding comparable applications now exist in the commercial market. These include robotics and computerized tomography. Military MCCs are also built in rather small quantities. This means that approaches must be developed for the economical production of short runs of devices or that flexible designs must be chosen which can be readily configured and programmed for different applications.

The devices to be delivered under VHSIC contracts will also differ in a second important respect from commercial practice -- provisions for radiation tolerance will require significant differences in device design and the selection of process techniques.

The VHSIC program provides both incentives and opportunity for revolutionary signal processing architectures. The Navy, in its specifications for EMSP, is insisting that the architecture not be algorithm-dependent. However, it is inevitable that a complex parallel processor will perform some tasks more efficiently than others. The cost effectiveness of custom-designed dedicated function processors, in the past, was fully justified by their architectural efficiency in the processing of specific dedicated functions and data structures. The use of VLSI and programmability may reduce this practice and the associated non-recurring and field support costs.

VII. MODERN HOL IS KEY TO REDUCED LIFE-CYCLE COST

MCSs are characterized by complex, difficult-to-code software. The software is frequently asynchronous, external-event-driven, data-dependent and interactive. It is demanding of machine resources and subject to mission-, threat- and need-based changes. Typical programs are written mostly in assembly language, with increasing use of CMS-2, JOVIAL and FORTRAN. System integration and test and system reliability are frequently a major problem. PASCAL is gaining popularity in commercial applications, but has found only limited application in military MCSs. PASCAL is a highly structured, simple-to-use language and speeds up program development and test and improves system

reliability. Navy SPI-1 is a descendant of ALGOL and related to PASCAL. It provides a number of features that facilitate programming in signal processing applications and MCSs.

The MCC software problem is to design languages, compilers, support tools, environments and architectures that provide maximum hardware efficiency with minimum programmer effort. A modern HOL is key to VHSIC/VLSI technology exploitation; portability of programs, data and personnel; increased programmer productivity; reusable software; and reduced MCS life-cycle cost, as summarized in Figure 8.

The primary characteristics of such a modern HOL are synopsized in Figure 9. The DOD Ada meets all the key features listed in Figure 9 and is an excellent solution to DOD MCSs. It is a descendant of PASCAL. It facilitates top-down ECS design, applicative programming, concurrent processing and distributed development. It is a strongly typed language with separation of data structures from program structures. The definition of new data types and operations (e.g., hardware operators) is permitted. Ada task synchronization and concurrent processing features are sophisticated and support table-driven real-time multitasking systems. Ada is an excellent language not only for scalar processing, but also for signal and dedicated function processing. Ada has the fundamental structure to realize the full potential of VLSI/VHSIC hardware. The application of Ada to signal processing applications requires certain language and compiler extensions.

VIII. SUPPORT ENVIRONMENT MUST SUPPORT THE ENTIRE LIFE CYCLE

The DOD MCC support environments are based on commercial host/tactical target computer concepts and draw heavily from the commercial computer base. Commercial off-the-shelf computers, commercial peripherals and commercial software, including large time-shared operating systems, interactive text editors, file management systems, document generators and libraries, are used for the host facility. Ada, JOVIAL or CMS-2 tools are interfaced to the commercial host computer environment. The tactical MCCs are interfaced to the host facility. The MCC target machines contain the tactical executive, application programs, debuggers and performance measurement tools. Such an approach creates a cost-effective support environment for the entire life cycle, the objectives of which are summarized in Figure 10.

MIL-STD-SDS is a major Joint Logistics Commanders initiative to upgrade the MCS software development methodology. The initiative consists of a new MIL-STD-SDS, a set of 25 consolidated DIDS and compatibility and upgrade changes to MIL-STD-483, MIL-STD-490 and MIL-STD-1521. The total draft MIL-STD-SDS specification package consists of 1000-plus pages and is currently out for industry and service review. The EIA has received over 1500 industry comments. The overall industry response is highly supportive of the JLC effort, with a number of major issues identified as summarized in Figure 11. The resolution of the identified issues should result in a highly flexible standard encouraging software technical excellence and productivity.

IX. TECHNOLOGY INSERTION AND LIFE-CYCLE COST DICTATE ACQUISITION POLICY

Acquisition of mission-critical computers and computer-based systems entails

- ENCOURAGE TOP-DOWN ECS DESIGN
- PERMIT APPLICATIVE PROGRAMMING
- PERMIT DISTRIBUTED DEVELOPMENT (SEPARATELY COMPILED PROGRAM UNITS)
- SEPARATION OF DATA STRUCTURE FROM PROGRAM STRUCTURE
- PERMIT DEFINITION OF NEW DATA TYPES AND NEW OPERATIONS/OPERATORS
- PROVIDE FOR SEPARATION BETWEEN PROGRAM SPECIFICATION AND PROGRAM BODY
- PROVIDE FOR CONCURRENT TASKING
- PROVIDE FOR TASK SYNCHRONIZATION AND DATA TRANSFER
- PROVIDE FOR SPECIAL INSTRUCTIONS TO COMPILERS OF VARIOUS SOPHISTICATIONS

Figure 9. *A HOL for MCS Processing Should Have Several Key Features*

- SYSTEM DEVELOPMENT
- SYSTEM DEPLOYMENT
- SYSTEM MAINTENANCE
- SYSTEM UPDATE

FEATURES MUST ACCOMMODATE:

- DISTRIBUTED DEVELOPMENT
- HARDWARE CHANGE
- QUICK, RELIABLE REPROGRAMMING
- CONFIGURATION MANAGEMENT
- PERSONNEL TURNOVER

Figure 10. *Support Software Must Support the Entire Life Cycle*

- JOINT LOGISTICS COMMANDERS
 - DARCOM
 - NAVMAT
 - AFSC
 - AFLC
- DRAFT MIL-STD-SDS
 - SDS (NEW)
 - DIDS (25)
 - MIL-STD-483, -490, -1521
 - OTHER MIL-STDs
 - 1000+ PAGE PACKAGE
- 1500+ INDUSTRY COMMENTS
- DRAFT FAILS MANY AIA/EIA CONCERNS
 - SOUND DISCIPLINE WITHOUT CONSTRAINING DESIGN (2 ISSUES)
 - FLEXIBILITY FOR SCOPE AND APPLICATION (3 ISSUES)
 - METHODOLOGY RESPONSIVE TO TECHNOLOGY EVOLUTION (3 ISSUES)
 - EFFECTIVE STRUCTURING AND INTEGRATION OF POLICY, STDS, SPECS AND DIDS (4 ISSUES)
 - POST-DELIVERY SUPPORT NOT ADDRESSED (1 ISSUE)

MIL-STD-SDS IS A MAJOR TRI-SERVICE INITIATIVE AND WILL HAVE MAJOR IMPACT ON SOFTWARE PRODUCTIVITY

Figure 11. MIL-STD-SDS Provides MCS Design Methodology Initiative

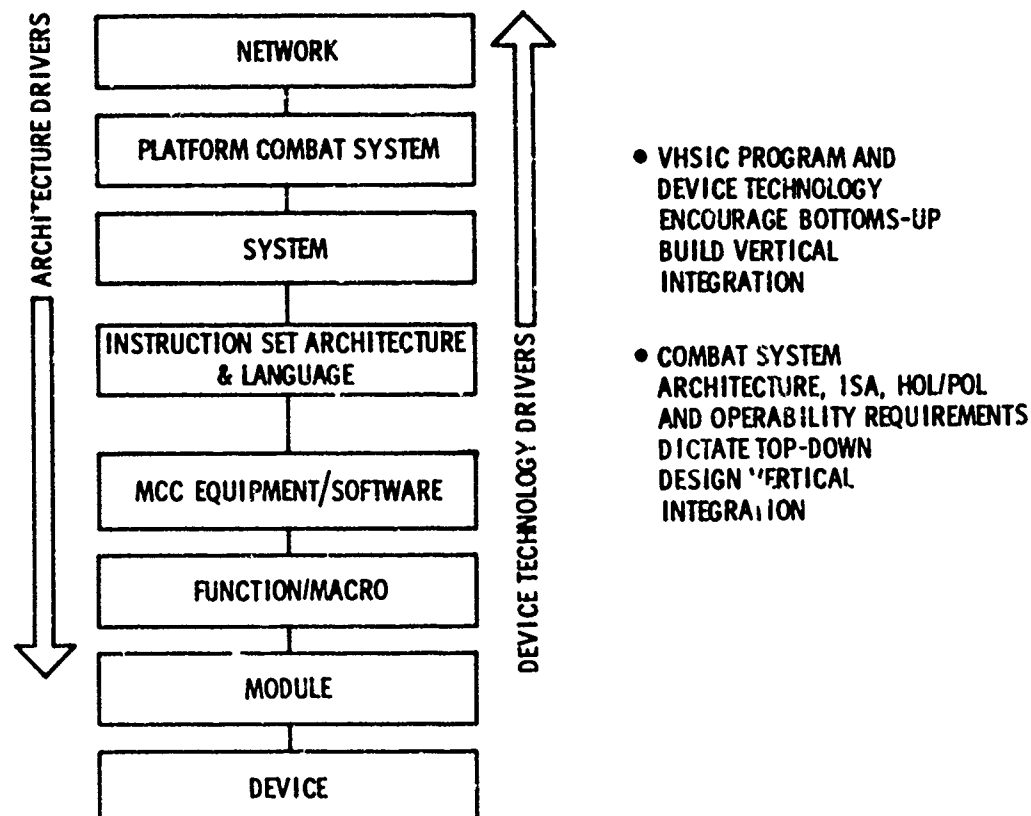


Figure 12. MCC Design Is Impacted by Vertical Integration Drivers

a significant investment over a period of years for development and production. Once acquired, the DOD incurs continuing costs for deployment, operation, retrofit and maintenance for the duration of operational use until the end of the life cycle. These costs frequently can exceed the initial development and production costs of the mission-critical system.

Typical DOD platform/facility operational life is estimated at 15 to 40 years; a typical embedded computer system operational life is estimated at 10 to 20 years, while the cost-effective embedded computer life in a commercial environment is 4 to 7 years. This disparity of operational life cycles between platforms/facilities, mission-critical computer systems and mission-critical computers dictates the need for technology transparent combat system architectures and ISA approaches and acquisition policies that encourage timely technology insertion in unsatisfied and evolving applications.

The Air Force acquisition policy is based on platform-based program procurements and an ISA level standardization (i.e., MIL-STD-1750 and MIL-STD-1862).

The Navy systems on a specific platform may evolve over the platform life cycle, resulting in a mixture of MCCs of different time frames. The platforms are engaged in extensive duration sea duty requiring on-board maintenance and support. This operational environment puts considerable premium on logistics support and training and has led to ISA as well as hardware implementation level standardization. The Army is faced with a similar logistics and survivability problem on the battlefield and has standardized at both ISA and hardware implementation levels (reference 3).

X. MCCs ARE INCORPORATED IN VERTICALLY INTEGRATED MISSION-CRITICAL SYSTEMS AND NETWORKS

Commercial computers are generally optimized for free-standing, general purpose operations. In contrast, the DOD mission-critical computers are frequently dedicated functional elements in mission-critical systems which in turn can be nodes in defense networks. Therefore, significant system and network level benefits result from architectural compatibility, interoperability and commonality of logistics and support in mission-critical computers.

While these combat system/network architectural and ISA drivers impact the upper levels of the MCC vertically integrated hierarchy, the device technology is driving the lower levels of MCC hierarchy for bottom-up vertical integration, as depicted in the eight-level hierarchy of Figure 12. VLSI/VHSIC device availability, module interconnection, bussing, packaging (e.g., SEMIP) and function/macro-level standardization are the major contributors to the bottom-up hardware implementation integration as the level of device technology densities continues to increase. Device and function level programmability (mask, micro, assembly and HOL) will provide application flexibility to a reduced number of VLSI devices so that they can address a wide range of applications.

XI. PROCESSING COST EQUATION PROVIDES GUIDANCE

The next generation DOD MCCs must be cost effective. The cost effectiveness must not only include the classical hardware and software factors of the life-

cycle cost, but also the fundamental effectiveness of the MCC product. Modern, highly expressive instruction sets that are efficient for military real-time applications are indicated. Extensive scalar, array and dedicated function algorithm and application analysis is required for the definition and validation of these ISAs. The ISAs must be higher order language/problem-oriented language oriented and optimized for DOD HOL standard language Ada. The scalar processors must further be suitable for multi-level military security. To assure technology insertion and a high degree of competition, the ISAs must be Government-owned or available to the Government and its defense contractors with no licensing restrictions.

The test of goodness for the MCC ISAs is the processing cost equation while executing a wide range of MCS applications. The elements of the processing cost equation are summarized in Figure 13. The non-recurring costs of the processing equation include the design of the compiler and support tools, the design of the ISA, the design of the hardware implementation, and the design of the application software. The recurring costs of the processing cost equation include the manufacture of the hardware, the execution of the compiler and the execution of the compiled program. In real-time systems, efficiency of execution of the compiled program in the target machine is the critical cost element, while the efficiency of the compiler execution in the host machine is of less importance in the next generation DOD (indirect execution) machines.

The higher level language (HLL) (direct execution) machines are still a subject of advanced research, and because of their limited performance in near-term VLSI/VHSIC device technology and the difficulty in execution validation, they are of limited interest in demanding real-time applications. The HLL machine performance implies more hardware operating at higher execution speeds, although it is not clear that these hardware issues will continue to inhibit the introduction of HLL machines in the longer term VHSIC era for lower performance MCS applications. The benefits of increased competition and reduced software costs with HLL machine acquisition strategy can potentially overcome the hardware-based reduced performance handicaps.

All processing cost equation elements are impacted by the commercial computer technology and the DOD architecture, hardware and software initiatives. The primary impact of the commercial VLSI and the DOD VHSIC device technology is to reduce the processing recurring costs, making the non-recurring costs more and more dominant. The VHSIC and commercial design automation initiatives are attacking the hardware implementation design costs which are currently dominating the limited-product-on-run VLSI device costs and schedules.

The DOD architecture initiatives are attacking all recurring and non-recurring elements of the processing equation. In demanding military real-time applications, the recurring manufacturing, compiled program execution and field support costs dominate the processing equation.

The DOD HOL initiative is focused on Ada and reduces both the non-recurring and recurring costs of the processing equation. The DOD standardization on a limited number of HOLs reduces application, compiler and architecture design non-recurring costs. The effectiveness of the HOL reduces the recurring costs of compiler execution and compiled program execution.

As can be seen from the discussion of the processing equation, significant

NON-RECURRING

- DESIGN COMPILER
- DESIGN ARCHITECTURE
- DESIGN HARDWARE IMPLEMENTATION
- DESIGN APPLICATION

RECURRING

- MANUFACTURE HARDWARE
- EXECUTE THE COMPILER
- EXECUTE COMPILED PROGRAM

	HOL	ISA	DEVICE TECHY
• DESIGN COMPILER	✓	✓	—
• DESIGN ARCHITECTURE	✓	✓	—
• DESIGN HARDWARE IMPLEMENTATION	—	✓	✓
• DESIGN APPLICATION	✓	✓	✓
• MANUFACTURE HARDWARE	—	✓	✓
• EXECUTE THE COMPILER	✓	✓	✓
• EXECUTE COMPILED PROGRAM	✓	✓	✓

ISA AND HOL STANDARDIZATION AND DEVICE TECHNOLOGY HAVE A MAJOR IMPACT ON DOD MCS PROCESSING COSTS

Figure 13. Processing Cost Equation Provides Guidance for MCC Optimization

changes and continued evolution of the next generation DOD MCC are anticipated as competition, device technology and DOD architecture, software and hardware initiatives continue to drive the design optimization of these machines.

XII. DOD MCC INITIATIVES ARE TAILORED BY SPECIFIC SERVICE-UNIQUE REQUIREMENTS

The Army, Air Force and Navy responses to the DOD policy, hardware and software initiatives are tailored by the unique requirements and acquisition practices of the specific service. These differences are real and deep-rooted in the evolution from previous generation standardization efforts, program procurement practices and differences in the combat environments. The specific service responses to the DOD MCC Directives 5000.29, 5000.31 and 5000.5X can be found in the various service regulations, instructions and joint policy statements. The Army AR 1000-1, Air Force AFR 800-14 and Navy TADSTANDS provide a top level view of service policy responses to DOD policy initiatives.

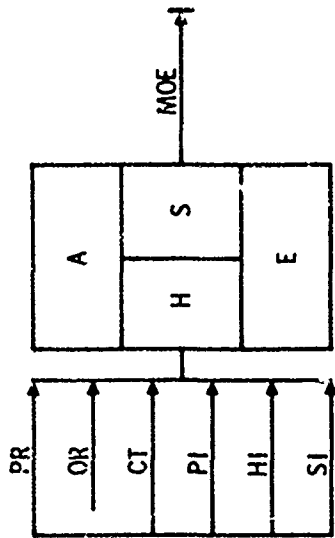
The analysis of the different acquisition strategies is beyond the scope of this paper. The primary purpose of this section is a comparative analysis of the next generation Army, Air Force and Navy MCC, and their projected effectiveness from a technological viewpoint based on DOD hardware and Ada-based software initiatives. It is assumed that these DOD processing initiatives will set the direction for a joint tri-service exploitation of the processing technology in the mid- to late 1980s.

The comparative analysis is based on projected mid-1980s commercial and military processing technology. All of the Army, Air Force and Navy MCCs identified in Figure 2 are still in competitive procurement. Therefore, the analysis cannot be based on actual product parameters. Instead, it is based on the specified goal requirements for these machines. Due to the rapidly evolving technology and the continued impact of the DOD processing initiatives and competition, the actual production machines in 1985 may significantly differ from the assumptions made in this goal-requirements-based comparative analysis. The analysis is further based primarily on personal experience with the above procurements and is obviously incomplete and based on limited and somewhat biased data. Apologies are due to all individuals and organizations where full credit has not been done to the characterization of their MCC.

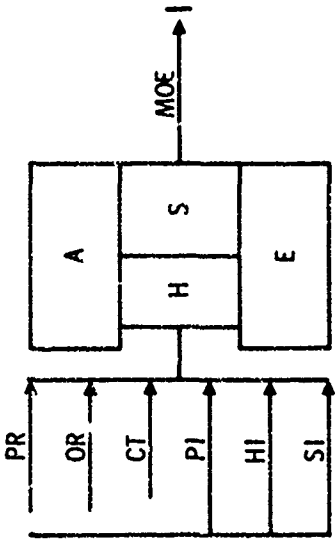
The comparative assessment of the Army, Air Force and Navy MCCs is summarized in Figure 14. The relative size of the product architecture, hardware implementation, software implementation and support boxes is indicative, on a relative comparison basis, of the effectiveness of the MCC in terms of computer technology and its compatibility with the DOD processing initiatives. The lengths of the input arrows are indicative of the performance and operability challenges assumed by the MCC development program, the compatibility with DOD hardware and software initiatives, and the industry response to the acquisition strategy.

An overview of the assessment summary indicates that the Navy EMSP is the most challenging MCC development and is a major contribution to the state-of-the-art in military and commercial signal processing. It potentially provides innovative solutions to ISA, combat system architecture, signal processing hardware and software, support environment, and system engineering methodology. It responds to a wide range of array and dedicated function processing requirements. It addresses a wide spectrum of demanding operational requirements

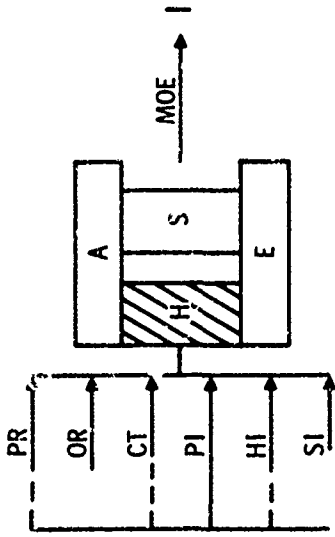
AN/JYK-41



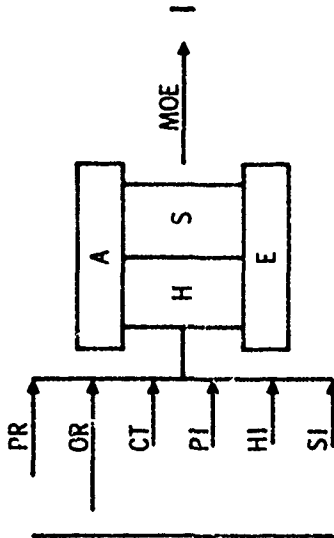
AN/JYK-49



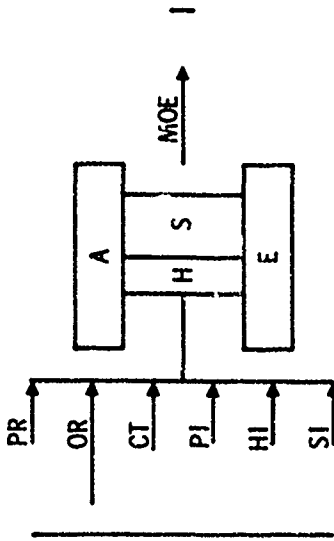
MIL-STD-1750



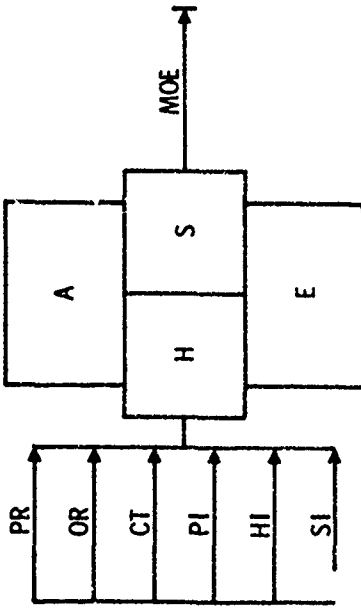
AN/JYK-43



AN/JYK-44



AN/JYK-2



KEY:

- A - ARCHITECTURE (ISA)
- H - HARDWARE PRODUCT
- S - SOFTWARE PRODUCT
- E - SUPPORT ENVIRONMENT
- PR - PERFORMANCE REQTS
- OR - OPERABILITY REQTS
- CT - COMPUTER TECHNOLOGY
- PI - POLICY INITIATIVES

- HI - HARDWARE INITIATIVES
- SI - SOFTWARE INITIATIVES
- MOE - MEASURE OF EFFECTIVENESS

Figure 14. Summary Assessment of DOD MCC Initiatives

and makes extensive use of future VHSIC and commercial VLSI technology. The EMSP model output measures of effectiveness (MOE) are projected in Figure 15. The parameter EMSP MOE footprint provides an assessment of the EMSP against current state-of-the-art versus the program goal. The actual EMSP footprint is subject to the winning contractor's design tradeoffs and technical approach and may differ considerably from the footprint described in Figure 15.

The Army MCF program AN/UYK-41 represents a significant design challenge against the Army stated goals and the commercial and military processing technologies. The AN/UYK-41 is the industry's first Ada design implementation at the superminicomputer level of performance. The stated AN/UYK-41 goals are: speed 3 MIPS, memory capacity 2M bytes, cost \$75K, reliability 10,000 hours MTBF, volume 0.52 cubic feet, power 100 watts and weight 40 pounds; the CSA requirement includes 62-user MIL-STD-1553 bus, 4 point-to-point parallel I/O and 2 point-to-point serial I/O.

The Air Force acquisition policy differs from the Army and Navy policies as it standardizes only at the ISA level and allows a large number of different hardware implementations. As a result, the MIL-STD-1750 computer implementations cover a range of different machine performances.

XIII. DOD ARCHITECTURE, HARDWARE, SOFTWARE AND MANAGEMENT INITIATIVES DRIVE THE MILITARY PROCESSING STATE-OF-THE-ART

Programmable hardware will be pervasive in DOD mission-critical systems driven by technology and unsatisfied service needs. The mission-critical computers provide a major leverage for MCS performance and operational readiness.

Hardware costs are dramatically reduced by VLSI technology, which drives to increased architectural parallelism and programmability. The benefits of VLSI and MCC are eroded by the skyrocketing non-recurring costs, risks and schedule delays of logic design in the form of software and irregular VLSI structures.

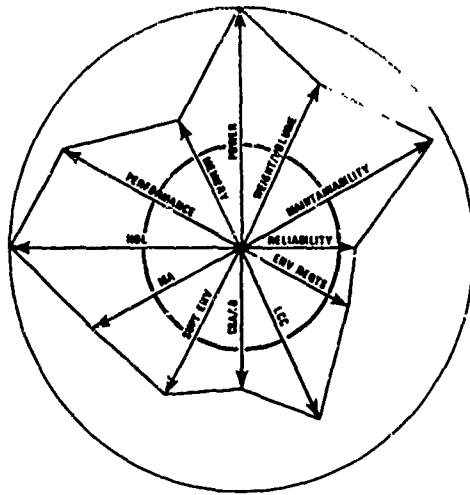
The DOD strategy of using the best of commercial computer technology in general-support and mission-support functions and providing DOD initiatives in mission-critical mission-operation functions is characterized. The fundamental question: "Can DOD next generation mission-critical systems and mission-critical computers meet their performance and operational requirements?" is raised and the need for a vertically integrated system solution, rather than a traditional "standard black box" solution used in the 1970s, is suggested.

The DOD processing initiatives complement the commercial computer technology in areas where the commercial technology is deficient for military needs. The DOD software initiatives are focused on Ada. The DOD hardware initiatives are focused on VHSIC and service-sponsored MCC developments. The DOD owned ISAs result in increased production volumes and competition with the associated standardization benefits. The DOD policy and technical initiatives assure our military supremacy through computer technology and constrain the projected explosive cost growth of mission-critical embedded computer systems and their associated software.

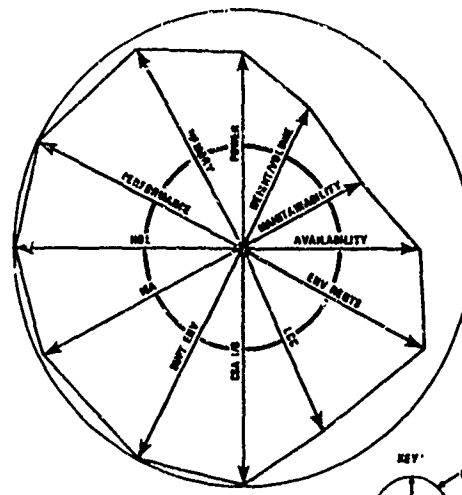
An 11-element processing issues model with 12 measures of effectiveness is suggested for the assessment of the effectiveness of the service MCC initiatives. A number of major issues identified by the MCC processing issues model are

NEBULA/MII-STD-1862

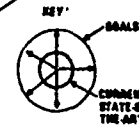
EMSP



AN/UYS-41 (MCF)



AN/UYS-2



DOD ARCHITECTURAL INITIATIVES ARE DRIVING THE PROCESSING STATE-OF-THE-ART

Figure 15. Assessment of DOD MCC Effectiveness

- PROGRAMMABLE HARDWARE PERSIVASIVE IN DOD MISSION-CRITICAL SYSTEMS
 - HARDWARE COSTS DRAMATICALLY REDUCED BY VLSI
 - COST OF DESIGNING IRREGULAR STRUCTURES VERY EXPENSIVE
 - VLSI DICTATES PARALLELISM AND PROGRAMMABILITY
 - SOFTWARE COSTS ARE SKY-ROCKETING
- MCC MODEL IDENTIFIES DOD UNIQUE REQUIREMENTS
 - VERTICAL INTEGRATION OF COMBAT SYSTEM ARCHITECTURE
 - OPERABILITY REQUIREMENTS
 - CHANGING THREAT/MISSION AND EVOLVING MCS REQUIREMENTS
 - COMPUTER PROLIFERATION PROBLEM
 - MISMATCH OF PLATFORM AND MCC LIFE TIME LONGEVITY
 - SURVIVABILITY
 - LIFE CYCLE COST
- DOD PROCESSING INITIATIVES COMPLEMENT COMPUTER TECHNOLOGY
 - ADA FOCUSES DOD SOFTWARE INITIATIVES
 - VHSIC AND MCC PROCUREMENTS FOCUS DOD HARDWARE INITIATIVES
 - DOD OWNED ISA'S RESULT IN INCREASED PRODUCTION VOLUMES AND COMPETITION WITH STANDARDIZATION BENEFITS
- MCC MODEL USED TO ASSESS EFFECTIVENESS OF DOD INITIATIVES
 - DOD INITIATIVES DRIVE THE SIGNAL PROCESSING STATE OF THE ART
 - DOD INITIATIVES DRIVE THE MILITARY COMPUTER STATE OF THE ART
 - DOD INITIATIVES INCREASE SOFTWARE PRODUCTIVITY AND CONTAIN THE PROJECTED EXPLOSION OF MCS SOFTWARE COSTS

DOD PROCESSING INITIATIVES PROVIDE THE BEST OF COMMERCIAL AND MILITARY COMPUTER TECHNOLOGY IN DOD MCS

Figure 16. Cost Benefits of DOD MCC Initiatives

discussed in the paper and summarized in Figure 16.

Finally, the Army AN/UYK-49, the Air Force MIL-STD-1750 and the Navy AN/UYK-43, AN/UYK-44 and the AN/UYK-2 processor requirements are assessed as to their responsiveness to the DOD processing initiatives and effectiveness as to their processing state-of-the-art in the mid 1980s. It is concluded that the service MCC initiatives drive the signal and dedicated function processing state-of-the-art in both commercial and military applications. The service scalar processing MCC initiatives drive the military computer state-of-the-art and, in the case of the AN/UYK-43, may actually match or exceed the commercial processing state-of-the-art in a military operational environment.

The second major conclusion is based on the observation that in certain MCC host functions and MCS general support functions, the commercial computer technology is cost effective and might be adequate for military operability requirements. At the same time, considerable care must be exercised, as any veteran of the European campaign in Germany can testify, because in the event of an all-out war there is no survivable soft facility, unless a high degree of redundancy is available.

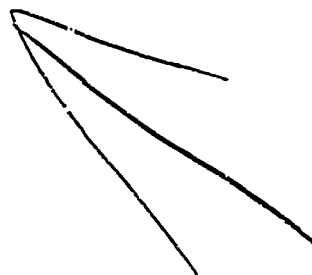
XIV REFERENCES

1. "DOD Digital Data Processing Study - A Ten Year Forecast," EIA, October 7-9, 1980.
2. Barry C. DeRoze, "An Introspective Analysis of DOD Weapon System Software Management," Defense Management Journal, Vol 11, No. 4, October, 1975.
3. Edward Lieblein, "The Army's Solution to Military Computer Standardization," USACEC, Fort Monmouth, NJ, 1982.
4. Draft "Strategy for a DOD Software Initiative," DUSD(R&AT) 27 August 1982.
5. O. Golubjatnikov, et al, "Final Report, Computer Accreditation Study," ONR Contract N00014-79-C-0987, General Electric Company, Syracuse NY, May 1980.

XV. DEFINITIONS

ADP	Automatic Data Processing
APSE	Ada Programming Support Environment
ASW	Anti-Submarine Warfare
BIPS	Billions of Instructions Per Second
BIT	Built-In Test
CSA	Combat System Architecture
EIA	Electronic Industries Association
EMSP	Enhanced Modular Signal Processor
HLL	Higher Level Language
HOL	Higher Order Language

IC	Integrated Circuit
I/O	Input/Output
ISA	Instruction Set Architecture
JLC	Joint Logistics Commanders
KIPS	Thousands of Instructions Per Second
LSI	Large-Scale Integration
MCC	Mission-Critical Computer
MCF	Military Computer Family
MCS	Mission-Critical System
MIPS	Millions of Instructions Per Second
MOE	Measures of Effectiveness
MTBF	Mean Time Between Failures
NSIA	National Security Industry Association
POL	Problem-Oriented Language
TADSTANDS	Navy Tactical Standards
TECR	Navy Tactical Embedded Computer Resource
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large-Scale Integration



STANDARD AVIONICS SOFTWARE
THE FUTURE STRATEGY FOR COST-EFFECTIVE AVIONICS

Edward C. Straub

ARINC Research Corporation
2551 Riva Road
Annapolis, Maryland 21401
Phone (301) 266-4854

Mr. Straub is a Principal Engineer and Senior Project leader in the Advanced Systems Program at ARINC Research Corporation. He has participated in development of the Air Force Avionics Master Plan, served as project leader for development of hardware and software acquisition strategies for the Common Multi-Mode Radar (CMMR) Program, was the principal engineer responsible for failure and cost-benefit analyses of the Air Force's Standard Avionics Integrated Control System (SAICS) Program, and the project leader for development of the USAF Wasp A/G missile-to-aircraft Integration Management Plan. Mr. Straub was also the project leader assigned to investigate Mark XII Identification Friend or Foe (IFF) operations in CONUS and Europe, and is currently performing cost analyses for the F-16 Advanced Radar.

In previous assignments with the U.S. Air Force, Mr. Straub was principal engineer for the advanced-technology electrostatic gyro (ESG) program in the Air Force's Avionics Laboratory. As Director of Electronics, AFSC Headquarters, Mr. Straub was responsible for more than 80 diverse programs ranging from small avionics, radar, weather, and security projects to major programs such as SACDIN, JTIDS, AABNCP, and AWACS.

Mr. Straub is a graduate of the U.S. Naval Academy and holds M.S. degrees in business administration and electrical engineering.

ABSTRACT

This paper reports on ARINC Research Corporation's work in developing and evaluating software acquisition alternatives for the USAF's Multi-Mode Radar program (since renamed the Multi-Role Radar (MRR) Program). Although the paper reflects work accomplished for that program, the approach taken could be used for any software-intensive avionics program where several aircraft are involved and for which most of the software and hardware might be common. The work was sponsored by Air Force Systems Command's Deputy for Reconnaissance and Electronic Warfare, Aeronautical Systems Division (ASD/RW).

The paper assesses the applicability of current radar technology and production programs to an MRR; discusses guidance provided by existing proposed policies, Directives and Standards; examines the operation, cost, schedule, risk, supportability, and management aspects of three software development alternatives and addresses the use of the ASD/ACCX software cost estimating model to analyze software development costs. Software acquisition alternative results are presented.

BACKGROUND

Over the past two decades, aircraft radar systems have evolved from single function designs to equipment capable of operating in many different modes. In conjunction with this increase in capability has been a reduction in both system size and weight, and the number of line replaceable units (LRUs). This evolution in radar system design has resulted in a wide variation of equipments presently installed in Air Force tactical aircraft, ranging from the limited mode, tube-type MA-1 in the F-106 to the multi-mode, all-digital-processing radar of the F-16. Naturally, this spectrum of radar systems and lack of hardware and software commonality among existing radars has added to maintenance and support costs because each new black box developed for a radar set creates a requirement for different spares, support equipment, documentation, and training.

The need for a detailed review of attack aircraft radar programs was highlighted at the First and Second Air Force Avionics Planning Conferences held during 1978, where the development of a Common Multi-Mode Radar (CMMR) (since renamed the Multi-Role Radar (MRR)) was proposed as a candidate for standardization. The conferees recommended that a commonality and life-cycle-cost (LCC) study be performed as soon as possible to determine the feasibility of a common radar approach to solve existing supportability problems and meet future operational requirements. This Government study, completed late in 1978 and known as the "ASD Common Radar Study," concluded that the use of a common radar for multiple aircraft applications was technically feasible and might provide sizable development, procurement, and support-cost benefits. As a result of the study and recommendations from the planning conferees, the Air Force established a new program element, and ARINC Research Corporation became involved in the development of CMMR hardware and software acquisition studies*.

At this juncture, one might ask, "what in-the-world is a common multi-role radar?" Figure 1 depicts the architecture and LRUs of a hypothetical MRR and highlights where its common hardware elements might be located. Note that two of the radar LRUs, the Programmable Signal Processor (PSP) and the radar computer -- both software-intensive devices -- would be common from a hardware viewpoint. Any aircraft-unique requirements would be handled by software (rather than hardware as in the past). It should be noted that the ASD Common Radar Study projected that more than half of the cost of such a radar would be attributed to these two LRUs.

*See ARINC Research Corporation Publications 1564-11-1-2122, "Development of Acquisition Strategies for the Common Multi-Mode Radar Program," January 1980 and 2362-01-1-2291, "Common Modular Multi-Mode Radar (CMMR) Software Acquisition Study," March 1981.

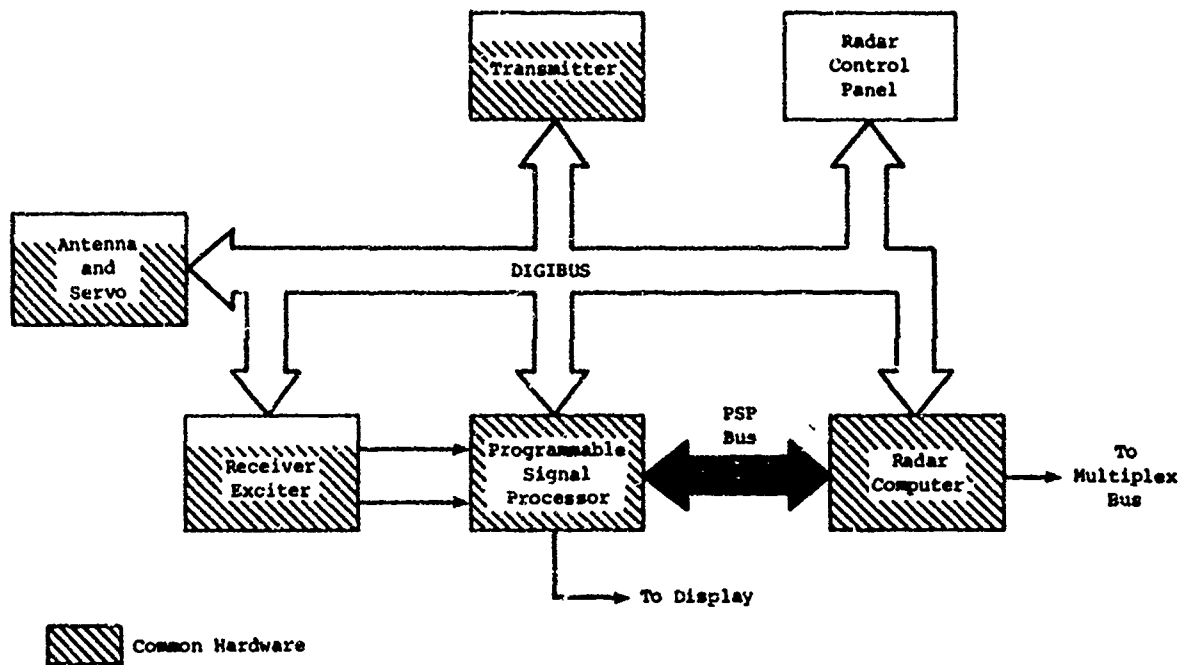


Figure 1. HYPOTHETICAL MRR LRU CONFIGURATION

Existing Radar Modes

There is no precise definition for the term "radar mode". There are major modes, submodes, selectable modes, and automatically occurring modes in a modern tactical radar. A mode that is manually selectable in one aircraft may be automatic in another. In one aircraft the weapon selected determines the radar mode of operation; in another, the target detection range allows the radar to automatically select the weapon required.

Figure 2 indicates the trend in today's projected radar growth requirements. As shown, the F-111D is the most advanced air-to-ground aircraft in the Air Force today, while the F-15 is the most advanced air-to-air one (from a radar viewpoint).

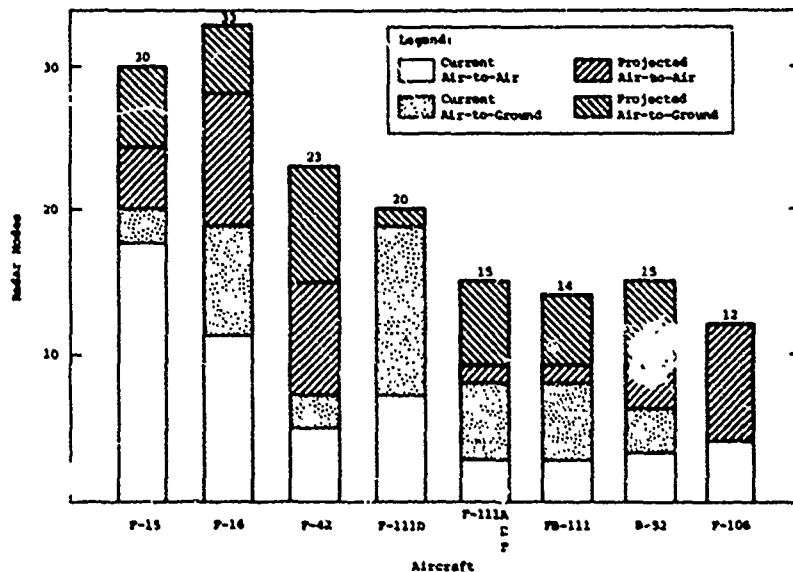


Figure 2. PRESENT AND PROJECTED AIRCRAFT RADAR MODE REQUIREMENTS

MRR SOFTWARE STUDY APPROACH

The approach used during our second (software) study is depicted in Figure 3. First, we reviewed nine present and planned radar technology efforts and assessed their relevance to a MRR.

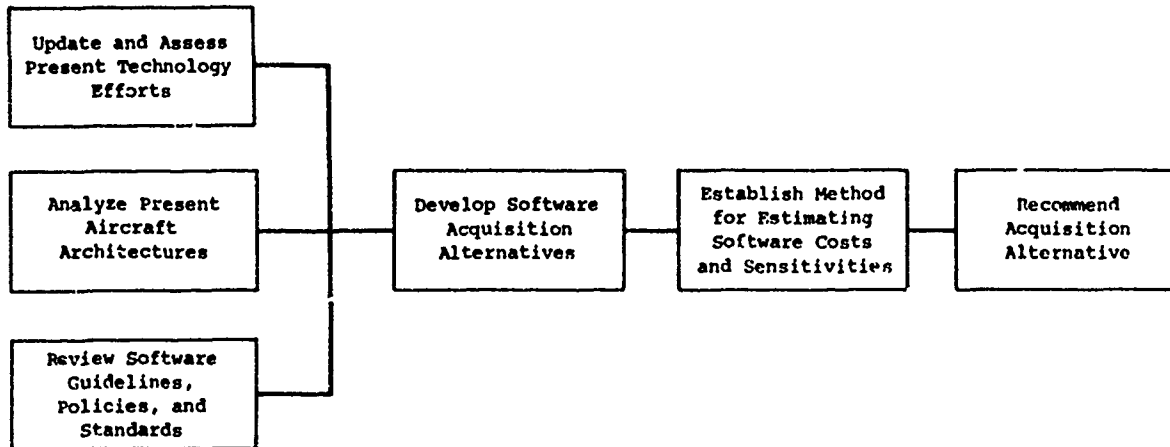


Figure 3. MRR SOFTWARE STUDY APPROACH

We also studied all available Air Force software Guidelines, Policies and Standards. Finally, we analyzed present modern aircraft architectures to determine how their existing radars interfaced and were integrated with the remainder of their avionics. These three efforts were performed in parallel. Next, we developed three software acquisition alternatives and established methods for estimating software development costs and sensitivities. We then used the results of these efforts to present our recommended acquisition alternative.

An underlying assumption used throughout our work was that there would be a dual competitive fly-off between the existing F/A-18 APG-65 radar and an advanced version of the F-16 APG-66 radar because of the development costs already invested. This assumption was used to estimate HOL vs. Assembly language costs.

ASSESSMENT OF PRESENT TECHNOLOGY EFFORTS

Table 1 presents a summary and current status of four radar programs applicable to the MRR program. Our technology review included an analysis of both the software algorithms and applicable Standards.

Table 1. APPLICABLE TECHNOLOGY PROGRAMS REVIEWED

Title	Program Element/Project	Status	Potential Impact on MRR
Radar Program- mable Signal Pro- cessor (PPSP)	64201/2519	Continuing	Software algorithms transferable to F-16 if done ASAP
Non-Cooperative Target Recogni- TION (NCTR)	63742/1177	Continuing	Algorithms transferable to MRR
ECCM Radar Im- provement (ERIP)	64201/2259	Cancelled- Work Ab- sorbed by 2519	Terrain following radar portions applicable to MRR
Electronically Agile Radar (EAR)	63241/1206	Project Com- plete	Technology source for some MRR algorithms and software

REVIEW OF SOFTWARE GUIDANCE, POLICIES, AND STANDARDS

Current acquisition policies and directives require that Program Managers adhere to certain avionics acquisition Standards unless compelling reasons (cost, schedule, etc.) provide for exemption. The major Standards with potential for significant impact on the MRR program were MIL-STD-1553B, -1750A, -1589A and the Atlas test HOL. Figure 4 indicates where and how the key Standards might impact the MRR program. MIL-STD-1760 (Aircraft Stores/Electrical Interface), not shown in the Figure, was in draft form and may also be applied to an MRR.

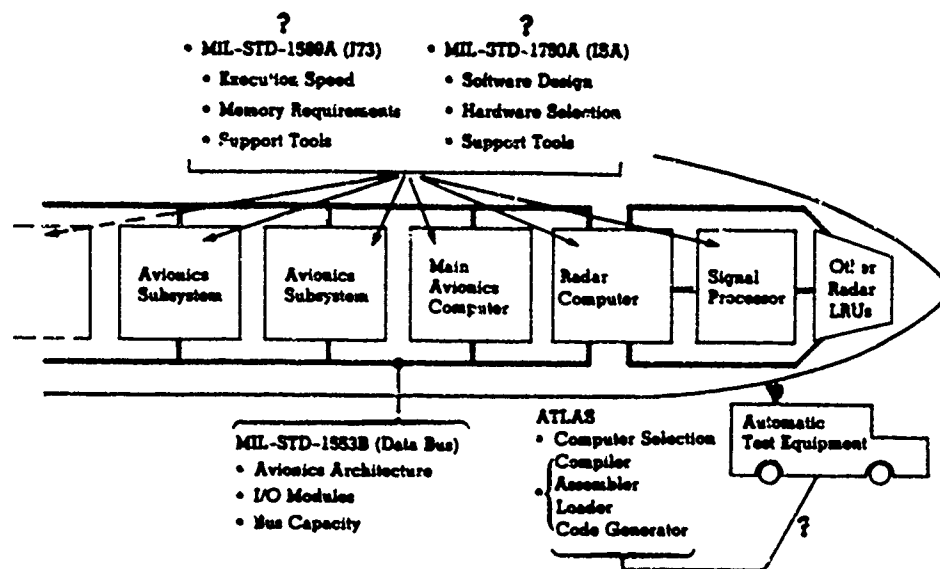


Figure 4. KEY SOFTWARE STANDARDS AND ISSUES IMPACTING MRR

With respect to the use of MIL-STD-1750A, we perceived an emerging competitive hardware base suitable for the radar computer; there were more than 20 models currently under development in the industry. The practicability of implementing a -1750A instruction set architecture for the radar PSP was still in contention both in the Government and industry, because a very fast moving technology was involved; further study would be required before use of the new ISA could be established for the PSP.

A conservative analysis of support savings by partially using the J73 Higher Order Language (HOL) for the F-16 Operational Flight Program (OFF) was performed and is presented later on. From a schedule consideration, it appeared likely that sufficient development and support tools for J73 would be available by the time of any competitive fly-off.

Table 2 presents a summary of the relevance of key software acquisition guidelines, policies, and Standards reviewed during our investigation and our assessment of their intent, impact on the MRR program, when to address them during acquisition, and the resolution of any issues they might create.

Table 2. SUMMARY OF SOFTWARE GUIDELINES, POLICIES, AND STANDARDS

Document	Intent	Impact on Program	When to Address	Implication on Radar		Issues	Resolution
				APG-65	APG-66		
DOD D 5000.29 AFR-800-14 DOD I 5000.31 AFR-300-10	Embedded Computer System management and control	Must address software management and development in PMP; requires HOL	Now	N/C*	N/C	None	Form CRWG; write CRISP; develop CPDP
MIL-STD-1589A	Defines J73	Directed to use	Now	N/C	N/C	Not used now; compiler availability; reduced speed; increased cost	Make partial use of HOL
DOD 5000.5x MIL STD-1750A	Defines ISA	Directed to use	Now	N/C	N/C	Not used now; hardware not available	Identify computers
MIL-STD-1521A	Technical reviews and audits	Must establish DID/CDRL items	1981	OK	OK	Documentation status; No standard set of acceptance criteria	Under analysis by JLC
MIL-STD-480 -433 -490 AFR-65-3 AFSCP-800-7	Configuration management and control; CPC: Format	Requires formal control of software configuration	1981	OK	OK	ICD status for candidates unknown	Form ICWG; develop CMP
MIL-S-52779	Software QA	Dictates contractor reqts for QA pgm	1981	OK	OK	No DID available; difficult to enforce	Under Analysis by JLC
MIL-STD-1553B	Defines Mux bus	Directed to use	1982	N/C (1553A)	N/C (1553A)	Not all candidate aircraft have bus	Use additional interface modules
MIL-STD-1760	Defines aircraft stores interface	Directed to use	When finalized	?	?	In draft form; interoperability of 1760/1553B	Use additional interface modules
MIL-STD-1679	Define software development	Requires structured programming, etc.	If adopted by USAF	N/C	N/C	USAF does not have an equivalent	Under analysis by JLC

* Non-compliant

ANALYSIS OF PRESENT AIRCRAFT ARCHITECTURES

We also analyzed the avionics architectures of three current aircraft programs to gain insight into precedents and lessons learned about software development for advanced-aircraft radars. It is significant that the radar hardware and software designs for all three aircraft examined (F-15, F-16, F/A-18) were still being updated and probably would not mature for several years.

The F-15 and F/A-18 employ some HOL source coding in their OPPs, but the F/A-18 radar OPP was written entirely in assembly language and there were no plans to convert to a HOL. The F-16 Advanced Radar OPP was scheduled to be written in J73. Software support for the radar OPP for all three aircraft was being accomplished contractually under a prime/subcontractor arrangement. The DoD had not committed itself to organic support for the software of these aircraft radars, primarily because of the required detailed knowledge of the software and hardware involved and the in-house skills required to provide the support needed.

Table 3 summarizes the radar development approach used by the F-15, F-16, and F/A-18 aircraft, their current status, and their compliance with several of the key software Standards.

Table 3. ASSESSMENT OF CURRENT PRODUCTION PROGRAMS

<u>Weapon System</u>	<u>Development Approach</u>	<u>Current Status</u>	<u>Current Compliance with Key Software Standards</u>
F-15	<ul style="list-style-type: none"> • Prime Integrating Contractor • Centralized Architecture • First digital bus • Extensive use of solid state memories • Assembly language source code 	<ul style="list-style-type: none"> • New radar computer & PSP 	None
F-16	<ul style="list-style-type: none"> • Prime Integrating Contractor • Decentralized architecture • First 1553 bus • Extensive use of solid state memories • Partial use of a HOL 	<ul style="list-style-type: none"> • New radar under development 	Uses: -1553 -1589 HOL(FCC) -ATLAS
F/A-18	<ul style="list-style-type: none"> • Prime Integrating Contractor • Distributed architecture • Extensive use of core memories • Partial use of a HOL 	<ul style="list-style-type: none"> • Baseline not yet frozen 	Uses: -1553A -CMS-2M HOL (Mission Computers)

DEVELOPMENT OF SOFTWARE ACQUISITION ALTERNATIVES

After our technology, production and Standards reviews, we developed and analyzed three software acquisition alternatives: responsibility to be assigned to (1) prime integrating contractors (PICs), (2) radar integrating contractors (RICs), or (3) a single operational flight program developer and integrating contractor (SIC).

The PIC alternative assumed that there would be five individual contractors (normally the aircraft manufacturers) responsible for the development of the radar OFPs required for the five candidate aircraft. These contractors were assumed to have a better overall knowledge of individual aircraft avionics, but would require assistance from the radar manufacturer(s) to develop and integrate the radar hardware and its software into an aircraft's avionics suite. The radars would be supplied either as GFE or CFE.

The RIC alternative assigned responsibility to two or more radar manufacturers for the radar OFP development, and integration of the radar and its software into the avionics suites of each aircraft. (Because of the quantities of units potentially involved, this approach would require two or more radar manufacturers.) Assistance from the developer of the avionics main computer hardware and software and other subsystems manufacturers would be required.

In the SIC alternative, one contractor would develop and integrate all radar OFPs into the avionics suite. We would very likely require contractual assistance from both the avionics software and the radar hardware manufacturers. The SIC might work with the radar, aircraft or other avionics developers.

Derinition of Criteria

Each of the MRR software development alternatives selected displayed certain advantages and disadvantages. In our study, we specifically paid attention to six criteria:

- . Operational Capability
- . Cost
- . Schedule
- . Risk
- . Supportability
- . Management

Operational Capability

Operational capability was keyed to the ability of a contractor to meet the radar software OFP requirements for the five aircraft. It did not address the larger issue of overall aircraft operational requirements. The ability of the PIC, RIC, or SIC to meet the radar's OFP acquisition needs, including the aircraft avionics interface requirements, was paramount in establishing how well this criterion was met. The prime factor in evaluating this criterion was the ability of a contractor to build growth and flexibility into the "baseline" OFP design to accommodate changes in a timely manner.

Cost

Cost was a measure of the economic impact of the radar software acquisition approach selected. Our evaluation did not include costs for independent verification and validation and integrated avionics flight test time. We made an assumption that these costs would apply equally to all three software acquisition alternatives.

Schedule

Schedule was a measure of the relative likelihood that the PIC, RIC, or SIC could meet the schedule demands of the five candidate aircraft. The integrating contractor selected would need to have an overall understanding and appreciation of the difficulties associated with developing radar OFPs that matched the radar production hardware schedules with the overall aircraft test and integration schedules, leading to a timely aircraft Initial Operational Capability (IOC).

Risk

In establishing the overall risk associated with the MRR program, schedule and cost were excluded because they were addressed in other criteria. Risk evaluation was a subjective judgment of the relative difficulty of integrating a radar OFP into the remainder of the avionics software under the PIC, RIC, and SIC approaches. For example, one radar hardware manufacturer alone could not meet the production quantity demand for the radars required for all five candidate aircraft; a teaming or leader-follower approach would be necessary. With more than one radar manufacturer, development and configuration control of more than one MRR OFP would become more difficult.

One of the key factors considered was the software status of the F-16 APG-66 Advanced Radar and the F/A-18 APG-65 OFPs at the time of selection of the MRR candidate. Another factor considered was the ease with which an OFP could be updated to include additional modes for the F-16 and other candidate aircraft in a timely manner.

Supportability

Supportability was a measure of how well a contractor could meet the demands of supporting the radar OFPs initially, and at the same time, set the stage to transition to organic support.

In our analysis of this criteria, we assumed that software support would transition to organic as soon as possible.

Management

The Management criterion concerned the relative ranking of the management complexities in acquiring MRR software. For each of the three acquisition alternatives, there was an assessment of how difficult it might be to define the responsibilities of various contractors and DoD organizations involved, the ability to control costs, and the degree of involvement to which the MRR Program Manager needed to commit Government resources.

ESTABLISHMENT OF METHOD FOR ESTIMATING SOFTWARE COSTS AND SENSITIVITIES

Approach

There was very little actual acquisition data that distinguished hardware from software costs in production aircraft. Cost data for development of OFPs that use HOL source code were even more rare. The only HOL applications

were in the F-16 Fire Control Computer which was under contractual Reliability Improvement Warranty (RIW), and the F/A-18 Mission Computers, for which the OFF baseline had not been frozen. We reviewed several Engineering Change Proposals (ECPs) for the existing F-16 avionics, and these provided valuable insight into its radar software mechanization. We were not able, however, to obtain any useful cost data for our MRR cost estimating.

Since we could not locate comparable manufacturer's estimates or actual values for use in our examination of the cost software acquisition impacts, we turned to the cost-estimating tools employed within the DoD to assist planners in developing software acquisition costs. One of the cost models, RCA's PRICE S, is used frequently in software cost estimating for major DoD programs. A derivative version employing the basic principals of PRICE S had been implemented by ASD/ACCX.* We selected this model because of its ease of implementation and use for parametric analyses. Previous to this time, the primary use of this model had been to assist ASD in source selection.

Assumptions

Use of the ASD model required an estimate of the projected memory allocations for the MRR OFF instruction and computer-associated radar software. We based our program estimates on the existing F-16 APG-66 Advanced Radar and F/A-18 APG-65 designs.

We made the assumption that the "baseline" MRR OFF would have a 180,000, 16-bit instruction/word modular program; this would increase to 205K after the competitive selection to accommodate the needs of the candidate aircraft. This compared closely with the 191K projected for the F-16 APG-66 Advanced Radar and 178K projected for the F/A-18 APG-65. It was assumed that the radar OFF would be written in assembly language, since the ACCX cost model uses machine instructions and assumes that the number of assembly and machine instructions are the same.

Our hypothetical MRR baseline OFF instruction distribution is indicated in Table 4. By using this baseline and analyzing the existing and desired radar modes of each of the candidate aircraft, we were able to develop instruction estimates for seven OFFs for the five candidate aircraft. These are listed in Table 5. The estimates were developed by using approximately 75 percent of the baseline modules and some new design and code for the cost model inputs. Note the assumption that the B-52 had two radar OFFs and the F-111D model had an OFF which was different than the other F-111s because of its unique air/ground requirements. Specific radar modes required for each aircraft were analyzed to establish the OFF sizes. For example, in the case of the F-111D, its larger program size (relative to that of the other F-111s) was attributed to its additional air-to-air and air-to-ground modes. In the case of the F-106, its small program size reflected a lack of any requirement for air-to-ground modes.

*TI-59 Handheld Calculator Software Cost Estimating Model, published by the Comptroller's Office, Aeronautical Systems Division (ASD/ACCX), Wright-Patterson AFB, Ohio, June 1980.

Table 4. BASELINE MRR OFP SIZE
(Modular program, written in assembly language)

Radar Computer	
Air-to-air Modes	35k
Air-to-ground Modes	23k
Built-in-test	17k
Subtotal	75k
Signal Processor	
Air-to-air Modes	43k
Air-to-ground Modes	62k
Built-in-test	12k
Miscellaneous	3k
Subtotal	120k
Miscellaneous	10k
Total	205k

Table 5. OFP STRUCTURES OF CANDIDATE AIRCRAFT

	F-16	FB/F-111	F-111D	B-52		F-4E	F-106
		A/E/F		Air/Air	Air/Grnd		
Final PGM size (k)	205	140	160	80	100	150	40
Initial PGM size (k)	160	105	125	50	75	115	30
New Code (k)	50	35	35	20	25	35	10
New Design (k)	25	5	5	5	10	5	<1

In addition to basic OFP instruction input, the model required other inputs: G&A, profit, labor rate and specific avionics parameters.

Model Use and Results

For the comparison of acquisition approaches, the G&A, profit, and labor-rate inputs were assumed to be the same for all OFPs. The G&A and profit parameters were those normally used by ASD; the labor rate was escalated for inflation.

In the ASD model, the analyst also has the flexibility of adding costs for OFP integration into the avionics software. This additional integration factor was used to indicate the differences in the alternatives selected. By applying engineering judgments, we selected the variable parameters for the seven OFPs, and then developed costs for the three alternatives.

Our results of the development costs (FY 1981 \$M) for the seven OFPs are shown in Figure 5. Totals ranged from \$26.8M for the PIC to between \$32.4 and 35.0M for the RIC, depending on the winner of the competition. (The dotted line indicates the RIC costs of recoding the existing F/A-18 radar OFP for the F-16.) The SIC costs totaled \$33.9M.

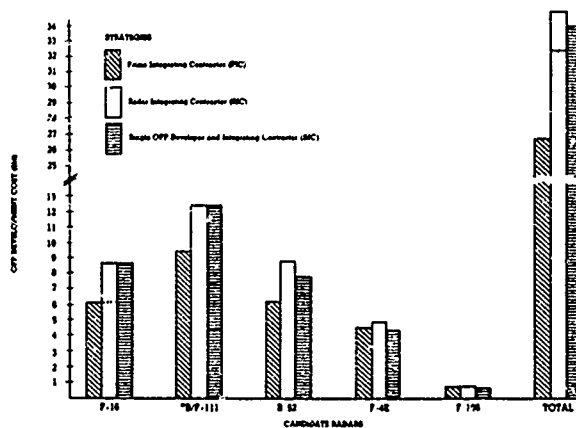


Figure 5. OFP DEVELOPMENT COSTS FOR THREE ACQUISITION STRATEGIES

SENSITIVITY ANALYSIS

We made a number of excursions to develop insight into the affects of various input parameters on software development cost, although we did not try to relate our results directly to our alternatives. RIC cost model F-16 parameters were used for the sensitivity excursions.

Effect of Program Size on OFP Development Cost

An unresolved issue was the size of the OFP required to achieve the capabilities demanded of the MRR. There is no doubt that the program software development cost could be affected significantly by the extent to which a developer required more or less machine instructions to establish the "standard" MRR capabilities.

We varied the OFP size from 155K to 355K. The results of developing a program requiring 100 percent new code and 100 percent new design is shown in Figure 6. The relationship indicates that the cost of developing the F-16 Advanced Radar OFP would be more than \$22M using 180K instructions; an increase of approximately \$1M would occur for each 10K of instruction growth input to the model.

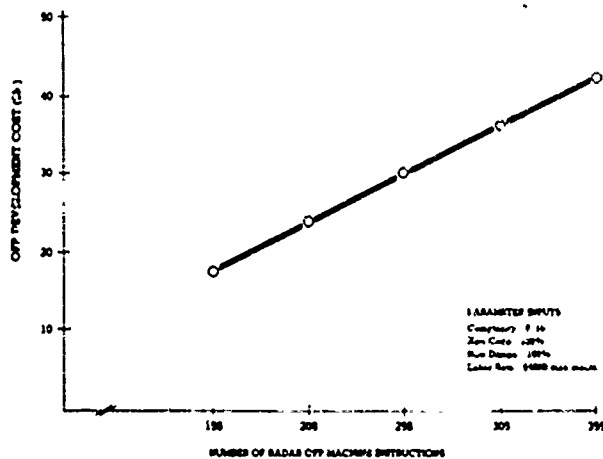


Figure 6. EFFECT OF PROGRAM SIZE ON OFF DEVELOPMENT COST
(Not including flight test and verification and validation)

Effect of Labor Rates and Code and Design Parameters on OFF Development Cost

Another factor considered in MRR software acquisition was the rapidly escalating cost of software labor. In this excursion, the RIC F-16 parameters and a 180K program size were used again. Labor rates were varied from \$4500 to \$6500 per man-month. Three code and design parameters were assumed to gain insight into the sensitivities: 100 percent new code, 50 percent new design; 100 percent new code, 0 percent new design (highly unlikely); 50 percent new code, 50 percent new design.

The results, depicted in Figure 7, show that cost was more sensitive to the amount of the new code than to the amount of new design. The MRR Program Manager was alerted to changes in labor rate in developing his OFF estimates. The cost difference between \$5000/man-month and \$5500/man-month is \$1.8M for a 180K program with 100 percent new code and 50 percent new design. These parameters might be similar to those required to reprogram the MRR OFF from assembly language to J73. These inputs resulted in a cost of \$19.2M. A similar cost might be expected to recode any assembly language MRR OFF.

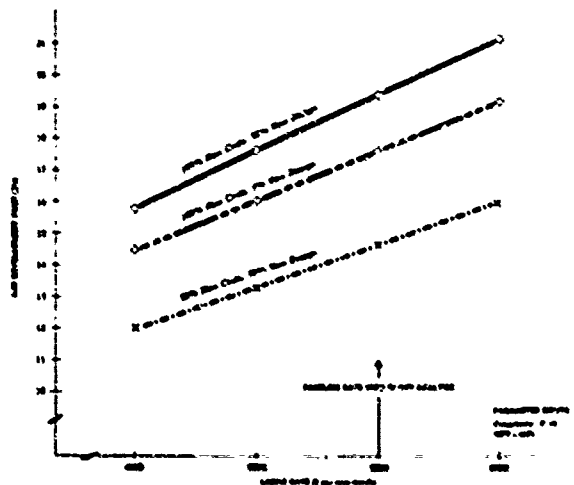


Figure 7. EFFECT OF LABOR RATE AND CODE AND DESIGN PARAMETERS ON DEVELOPMENT COST

Effect of Personnel Resources on OFF Development Cost

One of the concerns of the MRR Program Manager in selecting any software acquisition alternative would certainly be the experience of the software developer, as manifested in the degree of "learning" required to develop the MRR software. In the ASD model, this aspect is considered quantitatively in the Resource parameter. Our baseline value (3.5), was considered by ASD to be "average." We varied this input from 3.3 (above average) to 3.7 (below average). We also varied the new-code and new-design parameters from 20-percent new code, 10 percent new design to 40-percent new code and 20-percent new design, while holding the program size constant at 205K and the labor rate constant at \$5500/man-month.

Our results are depicted in Figure 8, which shows two plots ranging from \$16M to \$9.5M for the 40-percent new code, 20-percent new design case and \$13.5M to \$8M for the 20-percent new code, 10-percent new design case. Although the average skill resource parameter was used to develop the basic costs of Figure 8, the MRR Program Manager was alerted to the wide range of costs associated with the effort, based on the capabilities of the personnel involved.

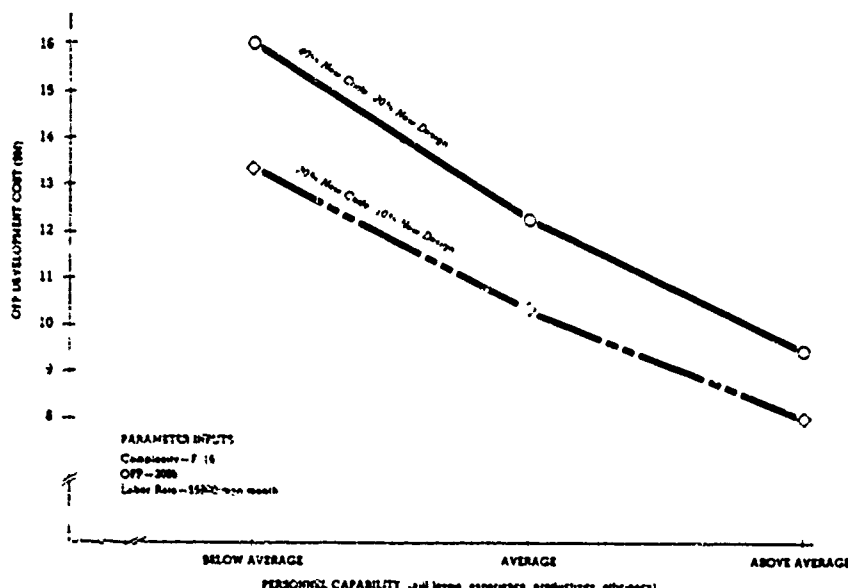


Figure 8. EFFECT OF PERSONNEL RESOURCES ON DEVELOPMENT COST

Estimate of Annual MMR Baseline OFF Support Cost

As analyzed by LOGICON,* the software life-cycle can be defined in terms of three major phases: subsystems software development, weapon-system integration, and software maintenance. Each of these phases includes three general cost elements: OFF engineering, support hardware and software, and quality assurance. OFF engineering includes all technical manpower directly involved in designing, coding, testing, documenting, and managing an OFF.

The maintenance phase includes two types of OFP engineering costs: corrective maintenance and modification maintenance. Corrective maintenance removes latent errors from an OFP, while modification maintenance involves both adapting an OFP to accommodate changes in avionics and hardware and enhancing an OFP to improve system performance. Support hardware and software cover the wherewithall for automatic-testing, aircrew-training, and direct OFP support. The latter includes: host-computer facilities, simulators, language translators, PROM loaders, ROM burners, etc. Quality assurance includes independent verification and validation and flight testing of the operational software.

Our review of other software cost analyses indicated that the cost of support ranged from 40 percent to 400 percent of the development cost. Systems that were threat-sensitive (such as electronic warfare systems) tend to fall at the high end of this range; systems with stabilized OFP requirements (such as flight control systems) tend to fall at the low end. A radar system falls in the higher end of the range.

For our support cost analysis, we assumed that the F-16 205K radar OFP was "baselined" after competition and maintained by the winning contractor for a three year period. The software would then transition to organic support with the contractor providing assistance between the third and fourth years. For the remaining six years, the Air Force would support the OFP organically. We changed the model Resource parameters accordingly.

The resulting curves in Figure 9 depict estimates of the annual support cost for the baseline MRR OFP. They do not include costs for the other six OFPs, since these would vary considerably depending on the organic support concept selected.

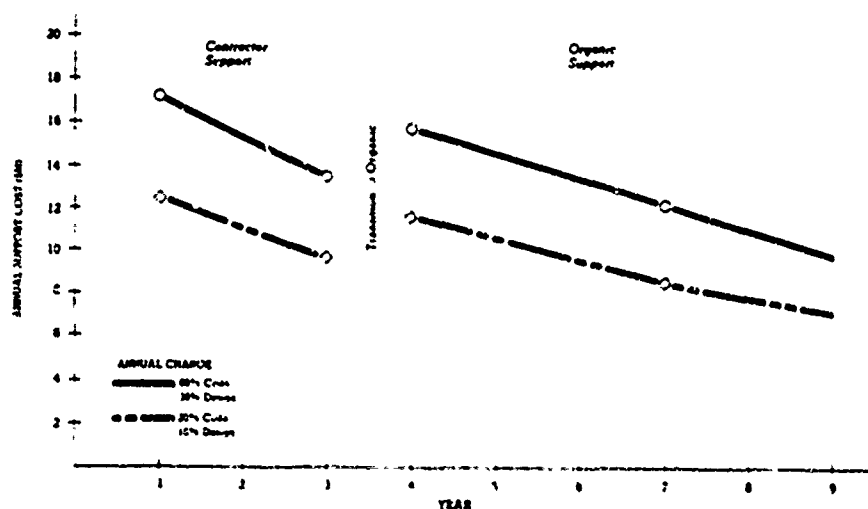


Figure 9. ESTIMATE OF ANNUAL F-16 OFP SUPPORT COST (10-year span)

*LOGICOM Report, "Potential Effects of Standardization on Avionics Software Life-Cycle Costs," 29 June 1979.

Effect of HOL Program Conversion Efficiency on OFP Development Cost

One of the most difficult assessments to confront the MRR Program Manager would be a judgment as to the maturity and modularity of the software design and code when a winner was selected. For the F-16, the radar OFP was planned to be written primarily in J73, ready to accept new code and new design for additional growth. For the F/A-18, the OFP would also be modular, but might be in assembly language.

To gain insight into some of the problems of reprogramming a previously designed assembly language OFP into a HOL, we applied the RIC parameters to our assumed 205K OFP assembly code program and developed new source code based on program efficiencies ranging from 100 percent to 50 percent. The resultant cost did not include any HOL support tools that might be needed. At least a compiler targeted to the radar computer would be necessary unless this was already available or to be developed and funded separately.

For the purposes of this analysis, we defined a 100 percent efficient program as one that was previously written in assembly language source code, was entirely rewritten in a HOL, required 25 percent new design, and did not require additional memory due to recoding. We did not attempt to analyze the problems associated with execution speed, although we recognized that this would be the primary reason for continuing to use assembly or machine language. For the purposes of our analysis, inefficiency resulted from the compiler only.

Even though the ratio of HOL instructions to machine-language instructions might vary from 1:1 to 1:30, we used the average 1:4 conversion for J73, as suggested in the ASD cost model. We held any new design at 25 percent and varied the new code's compiler efficiency from 1.0 to 0.5 (highly unlikely). This caused the number of program instructions to grow from 205K to 308K.

The results, depicted in Figure 10, show a spread of additional OFP development cost ranging from 0 to \$18.5M for a variation of efficiency from 100 percent to 50 percent. The numbers would have been higher if the 25 percent new design figure had been increased.

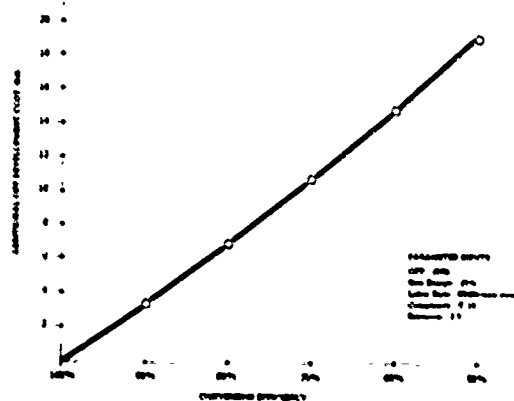


Figure 10. EFFECT ON DEVELOPMENT COSTS OF CONVERTING ASSEMBLY LANGUAGE TO HOL

Any penalty in development cost due to HOL conversion may be offset by a savings in support cost in the long run. As an indication of the order of magnitude of these savings, we used an average yearly support cost of \$9.5M from Figure 9 and applied it for a ten-year period. A total value of \$95M represents a yearly change of 20 percent new code and 10 percent new design in OFP alone, under the assumptions we used. RADC had indicated that programmer productivity (number of lines of code produced per month) could nearly double when a HOL rather than assembly language was used. We used this estimation in our calculations. The potential support cost saving resulting from use of a program written in 80 percent HOL was \$7.6M for the assumed ten-year period support phase.

Assembly Language Translation to J73

For this analysis, we assumed that the existing 180K OFP grew to 205K as before. We also assumed that the radar computer (RDP) portion (75K) of the OFP was rewritten in 80 percent HOL, the PSP portion (120K) was rewritten in 50 percent HOL and the "Misc" portion (10K) remained written in assembly language source code. We again used the \$5500/man-month, the 100 percent new code, and 25 percent new design parameters.

The results were a plot of program size vs. cost, which varied by 24K and \$2.5M respectively for each 20 percent of HOL inefficiency. For example, with a 80 percent HOL efficiency for the assumed assembly language OFP, the size of the program is 24K more (229K) and showed a cost growth of \$2.5M. Table 6 illustrates this program. The \$2.5M did not include potential programmer man-hour saving resulting from HOL coding or additional costs for compiler development, etc., as discussed before. It also did not include a review of assembler efficiencies, etc.

Table 6. EFFICIENCY

	205k OFP * (Ass. Lang.)	205k HOL/MIX OFP (100% Efficiency)	229k HOL/MIX OFP (80% Efficiency)
RDP	75k	60k HOL (80%) 15k AL	72k 15k
PSP	120k	60k HOL (50%) 60k AL	72k 60k
MISC	10k	10k AL	10k
	205k	205k	229k

*In Thousands of Machine Instructions.

The MRR Program Manager would need to continually monitor the problems and cost variances associated with using a HOL - especially if the program initially exists in assembly language and rewriting of its source code is required. For this excursion, we only analyzed the growth in memory size and costs due to compiler inefficiency, but there is also a major concern associated with program execution speed: a thorough review of the present radar OFPs would be required if the radar OFP was to be written primarily in J73.

PRESENTATION OF ACQUISITION ALTERNATIVE RESULTS

Table 7 ranks the three MRR software acquisition alternatives according to the criteria identified. Our judgments were based on the assumption that the MRR market would be sufficiently large to sustain more than one radar manufacturer. If this assumption is not valid, the difference between the RIC and SIC alternatives is almost indistinguishable.

Table 7. RANKING OF MRR SOFTWARE ACQUISITION STRATEGIES

Ranking Criteria	PIC	RIC	SIC
Operational Capability (Threat Accommodation)	-	0	+
Cost			
Acquisition	+	0	0
Initial Support	-	0	-
Schedule			
Radar OFF	-	+	0
Weapon System IOC	+	-	0
Risk	+	-	0
Supportability	0	-	+
Management (Ability to control costs and flexibility to accommodate more than one radar manufacturer)	+	-	0
+ Most Attractive			
0 Moderately Attractive			
- Least Attractive			

For the Operational Capability criterion, we ranked the SIC alternative as the most attractive. The contractors for all three approaches could meet the radar OFF acquisition needs, given sufficient funding and time. However, we believed that the SIC would be in a better position to control and maintain the configuration of seven OFFs, build growth and flexibility in the OFF design, and accommodate changes to the radar OFFs in a timely manner.

For the Cost criterion, the PIC was the most attractive acquisition alternative. The RIC and SIC were ranked equally, although the RIC cost could be lower or higher than the SIC cost, depending on the winner of the competition. For the initial support cost, a SIC was the most attractive because of the contractor's "learning" experience gained in developing seven OFFs.

We split the Schedule ranking into two parts. The RIC alternative was the most attractive for meeting the radar OFF schedule because of the contractor's unique experience with the radar hardware and software. For the weapon system IOC schedules, the PIC alternative was the most attractive because of the contractor's knowledge of his avionics software, his responsibility for existing radar OFFs, and his overall responsibility in meeting the aircraft IOC.

For the Risk criterion, the PIC alternative was the most attractive because of the difficulty in integrating the radar OFFs into the rest of the complex avionics software suites. This ranking also recognized the possibility that a second radar manufacturer would be required if the market size were large enough, compounding the risk with the RIC alternative.

For the Supportability criterion, we ranked the SIC alternative as the most attractive. The SIC developed an overall understanding of each of the candidate aircraft's weapon system requirements and how the radar OFP was to meet these requirements. The SIC would learn as the integration problems encountered in each aircraft were understood and resolved; he would be able to maintain an unbiased perspective with regard to both the avionics and radar software. The SIC would be in a better position to assist the Air Force in establishing organic MRR software support because his responsibilities would be exclusive of the aircraft or radar manufacturer's.

The PIC alternative was ranked as the most attractive for the Management criterion, because less Government management, engineering, and contracting resources would be required. For example the alternative introduced "third party involvement" in the problem software development and integration.

All three acquisition alternatives would require prime and subcontracting arrangements.

SUMMARY

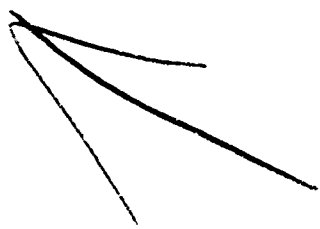
This paper has presented the approach that we used to define and select software acquisition alternatives for development of a MRR and the results that we obtained. Our recommended approach to another software intensive avionics program as complex as MRR would be to:

(a) Review all relevant on-going technology efforts with the thought of using common algorithm and other technology outputs wherever possible and practical.

(b) Review both aircraft and avionics Standards to ensure compliance as required and to avoid troublesome interface compatibility, integration, and interoperability problems later on.

(c) Assess the status of existing hardware and software production programs to determine if and where they might be affected, modified and/or used in the most cost-effective manner.

As a result of our software acquisition analysis, we found that the preferred software acquisition strategy implied by the ranking of our alternatives was a single OFP developer and integrating contractor (SIC).



THE APPLICATION OF STANDARD SYSTEM SPECIFICATION
TECHNIQUES TO THE DESIGN OF VERY LARGE SCALE INTEGRATED CIRCUITS

Dave Jordan
Marconi Avionics

BIOGRAPHY

Dave Jordan obtained his honors degree in Mathematics from the University of London in 1974. After some years in the Telecommunications Industry, he joined the Airborne Software Division of Marconi Avionics Limited where he is now the senior member of a team which has developed the Mentor System Specification System.

ABSTRACT

The design of large scale integrated circuits has traditionally been the province of Silicon Real-Estate Artists. Standardization has meant keeping the same artist for each design iteration. The lack of feasibility of this approach for circuits with gate counts between 10,000 and 100,000 is apparent. The use of modeling programs using hardware description languages such as HARTAN, leads to the use of standard circuit modules with their descriptions held in a model library. At Marconi Avionics this approach has been extended through integration with Mentor, a system specification system language which can be applied at all levels of design and which allows the chip designer to both differentiate and correlate the behavioral and functional description of circuits and modules. Advanced language analysis techniques enable detailed design verification and the construction of a perfectly consistent design database.

ADVANCED STANDARDIZED SYSTEMS/SUBSYSTEM

**SESSION CHAIRMAN: Major L Caro
AFWAL/AAAF**

**MODERATOR: Colonel Darrold D. Garrison
AVRADA/DAVAA-D**

LANTIRN - TOMORROW'S SOFTWARE DEVELOPMENT TODAY!

Kenneth B. Hawks

LANTIRN Software Manager, ASD/RWNM, WPAFB OH, (513) 255-6642

BIOGRAPHICAL SKETCH

Kenneth B. Hawks was born in Utica, New York. He received his Bachelor of Science Degree in Physics from Syracuse University in 1964, and his Master of Science Degree in Public Administration from Troy State University in 1975. During his 18 years with the United States Air Force, he has been associated with all phases of embedded computers in weapon systems. Prior to his present assignment as the LANTIRN Software Manager, he was Chief of the Embedded Computer Resources Branch for the Deputy for Reconnaissance and Electronic Warfare Systems.

ABSTRACT

The Low Altitude Navigation and Targeting InfraRed System for Night program is a complex system consisting of a wide field-of-view, holographic Heads-Up Display; a navigation pod featuring a wide field-of-view forward looking infrared (FLIR) and terrain following radar subsystems; and a targeting pod whose initial capability will provide a narrow field-of-view FLIR, a laser designator/ranger, and missile boresight correlation.

The airborne computer resources consists of 22 computers and 32 computer programs. LANTIRN is the first truly distributed processing system with autonomous processors developed for aircraft. Additionally, LANTIRN was the first program with JOVIAL J-73, MIL-STD-1750A architectures, and MIL-STD-1553 busses directed upon it.

The LANTIRN software development has successfully demonstrated the practicality of language and architecture standardization. Savings in schedule and resources directly resulted from the transportability of code. While there were (and still are) challenges to standardization, LANTIRN has proven its effectiveness on large, complex, state-of-the-art systems.

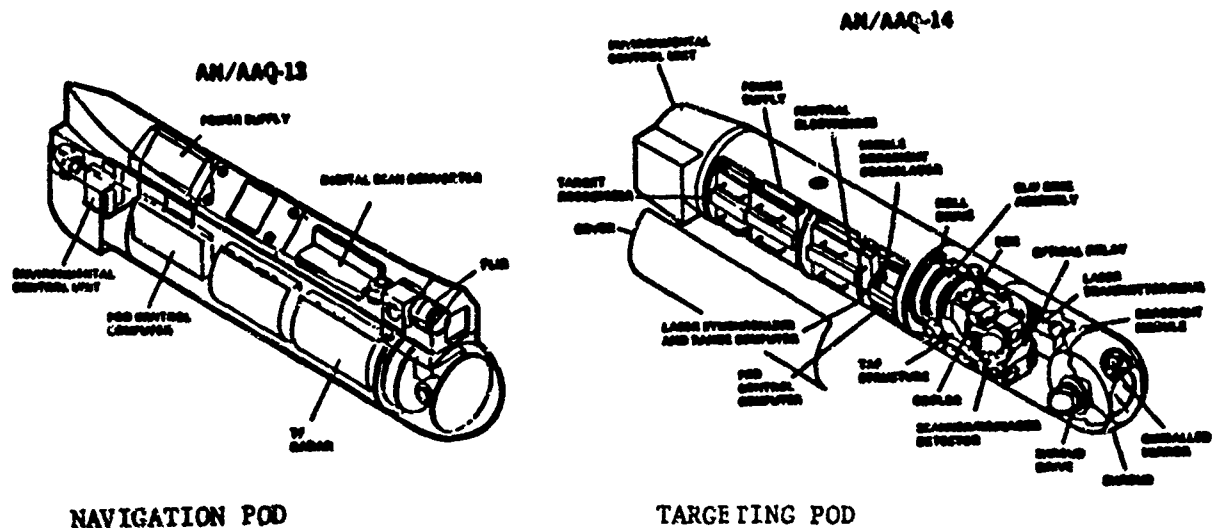
LANTIRN SOFTWARE DEVELOPMENT

The LANTIRN system is designed to provide ingress to the target area at night or under the weather, deliver laser guided bombs (LGB), conventional munitions or automatic IR Maverick launch and then egress at low level. The system is designed to enhance the F-16 and A-10 aircraft with potential application on the F-15.

The Head-Up Display (HUD) features a unique stroke-written-raster fly-back technique which allows all mission symbology to be superimposed on the raster picture derived from the night vision sensor.

The LANTIRN Fire Control System consists of two pods, a navigation pod, and a targeting pod.

Figure 1

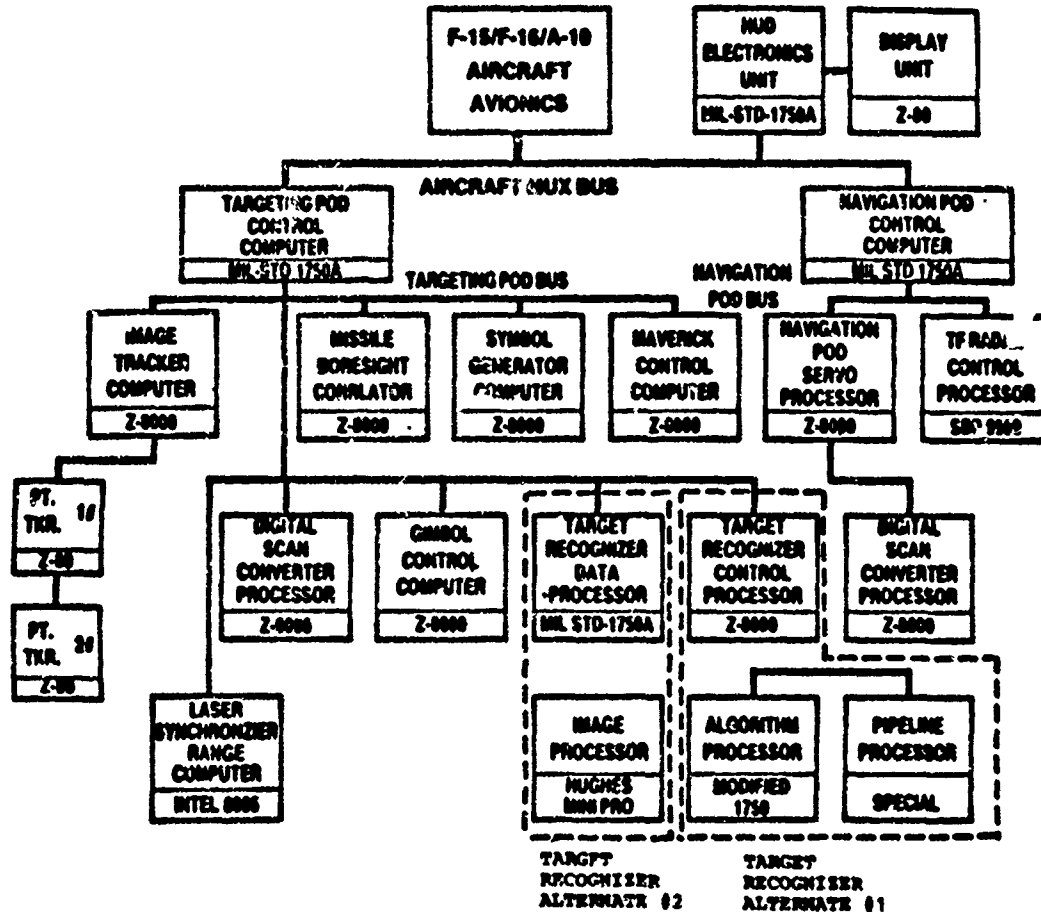


The navigation pod uses a terrain avoidance radar and Wide Field of View (WFOV) FLIR to provide day or night low level operations under the weather. The Ku-band radar has high ECM resistance; a look-into-turn capability and high resolution. The WFOV FLIR provides a 28 by 21 degree field of view; snap look look-into-turn, and automatic gain, level and brightness control.

The targeting pod uses a Narrow Field of View (NFOV) FLIR, a laser designator/ranger, a missile boresight correlator, dual digital image trackers, and provisions for automatic target recognition. The target pod provides the capability to acquire, track, and attack mobile or fixed targets.

The computer resources are as shown in Figure 2. The LANTIRN system is a distributed and federated network of "smart" components. All of the major components contain microprocessors, with the key central computers being MIL-STD-1750A. The Navigation Pod control computer has a throughput rating in excess of 1 megaops utilizing the DIAS instruction mix.

Figure 2



LANTIRN was directed to implement three key standards that had an influence on software development. MIL-STDs-1750A, 1589B, and 1553B all helped shape the software design. Early in the program, it was decided to provide all of the support software as government furnished property (GFP). This eliminated any unknowns from the support tools for software maintenance. Most of the host computers were also furnished as GFP.

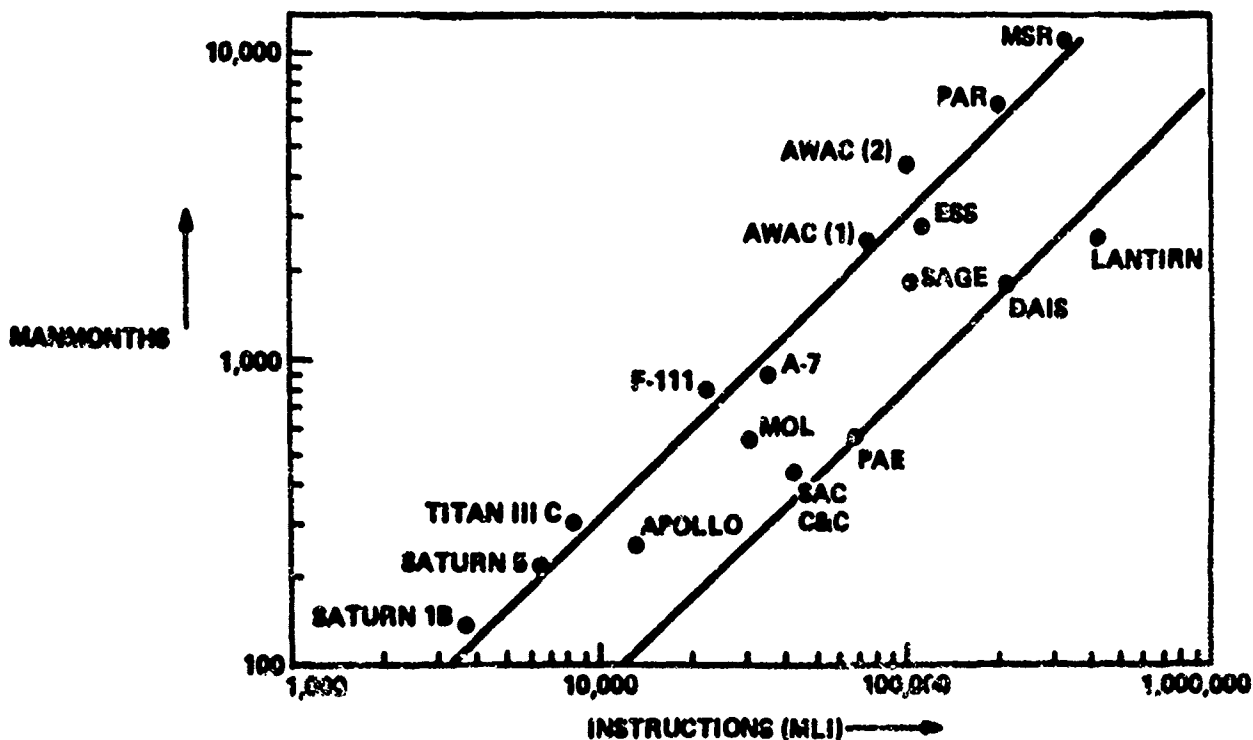
Structured programming techniques assured a logical, methodic design evolution. Government participation in the detailed design walkthroughs assured an indepth understanding of the design by program engineers, while the using and testing community benefited from the briefing style design reviews.

Standardization has made the LANTIRN software effort possible. A major factor has been the interchange of standard library routines among development teams and even between contractors. Terrain following radar code developed by Texas Instruments (TI) executes in the pod control computer developed by Delco utilizing the Martin Marietta executive common to both pod control computers. Likewise, the Z8000 microprocessors share a common, universal executive. In LANTIRN, the same source code could execute in a 1750A, TI 9989, or Z8000 architecture.

The application of standard computer architectures throughout LANTIRN has allowed transportability of source and object code among subsystems. The vast size of the LANTIRN airborne software and its complexity dictated the need for a higher order language to insure schedule and budget constraints would be met. To put LANTIRN perspective, Figure 3 shows its relative size.

Figure 3

LANTIRN SOFTWARE SIZE COMPARISON



The application of JOVIAL J-73 (MIL-STD-1589B) was the first in an airborne system. While the immaturity of the language and support tools presented numerous challenges, JOVIAL resulted in more productive programming. Two features of JOVIAL, namely COORDINATES and TABLE/MATRIX manipulation, are especially noteworthy for airborne applications.

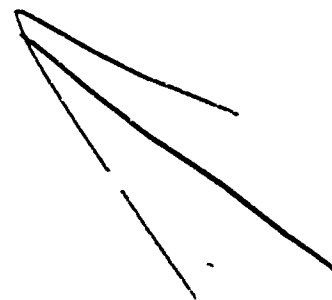
The software development approach also has resulted in the depot requirements being known very early. Additionally, the facilities, training, and support tools required are greatly reduced by commonality.

The LANTIRN executive is the first real-time multitasking executive. It is designed to be transportable, reduce risk, and improve timing and capability. The standard for its design was the Digital Avionics Integrated Suite (DAIS) executive with high overhead features such as multiprocessor voting deleted. It is flexible in that it controls numerous, divergent tasks such as Controls and Displays, Fire Control, Navigation, Stores Management/ Weapons Launch, and Ambiguity Resolution in a totally interrupt driven environment.

The overall software design of the pods is transparent to the host aircraft but for an interface program that connects aircraft inputs into common pod data structures. This allows expansion to numerous airframes with minimum effort.

SUMMARY

The LANTIRN program is a preview of weapon systems to come. More microcomputers will be appearing doing more functions. The application of MIL-STD-1750A and JOVIAL J-73 have significantly affected the LANTIRN software development in a positive manner.



AFTI/F-16 DIGITAL FLIGHT CONTROL SYSTEM DEVELOPMENT

James K. Ramage
Air Force Wright Aeronautical Laboratories
Wright-Patterson AFB, Ohio

ABSTRACT

The Advanced Fighter Technology Integration (AFTI)/F-16 Advanced Development Program is a joint USAF, Navy, and NASA effort aimed at the development, integration, and flight test evaluation of emerging technologies for improving fighter aircraft mission effectiveness. Major development thrusts include Digital Flight Control System (DFCS), Direct Force and Weapon Line Control, Pilot Vehicle Interface and Automatic Maneuvering Attack System (AMAS). Development of an advanced highly reliable multimode primary digital flight control system is the core technology building block for integrating mission and flight critical systems. This paper summarizes the overall AFTI/F-16 Phase I DFCS development effort with emphasis on key design parameters, flight test results, and lessons learned.

Application of digital flight control technology offers the opportunity to integrate advanced concepts in a multirole, high performance fighter aircraft to achieve operational versatility, improved overall mission effectiveness and decreased cost of ownership without sacrificing system reliability and safety. The DFCS portion of the program encompasses the complete development and integration of a full authority multimode triplex digital FBW flight control system employing decoupled 6DOF flight path control capabilities. Major technical developments include: (a) task-tailored multimode control laws incorporating direct force and weapon line pointing features, (b) triply redundant digital flight control computer complex using the newly developed BDx-930 processor, (c) advanced redundancy management techniques, which provide essentially two fail-operate capability and meets or exceeds a loss of control reliability of 1×10^{-7} failures/flight hour, (d) integrated crew station using multipurpose controls and displays, and (e) compatible interface for integration with other subsystems, including fire control, mission avionics and associated multipurpose displays, through a common digital data bus. Organizing primary flight control modes and associated control laws based on specific mission/weapon delivery requirements offers the opportunity for enhancing overall mission effectiveness, while at the same time emphasizing the pilot's role as a mission manager rather than a subsystem operator. In consonance with this design philosophy, the following basic mission tailored control modes have been implemented in the AFTI/F-16 test bed: (a) Normal Mode with ancillary functions for take-off/landing, refueling, formation, cruise and pilot relief, (b) Air-to-Air Gunnery Mode, (c) Air-to-Ground Gunnery Mode, and (d) Air-to-Ground Bombing Mode. Control law/sensor reconfiguration schemes are also employed to provide acceptable flight characteristics based

on available functioning system elements. In each of the primary modes, the flight control system provides the necessary flight path decoupling, and desired vehicle response characteristics, specifically tailored and optimized for the appropriate mission segment. In addition, the DFCS architecture, control laws and redundancy management algorithms are designed to accommodate the Phase II automated maneuvering attack system (integrated flight/fire control), which will demonstrate automatic weapon delivery capabilities during low altitude dynamic maneuvering flight conditions.

Digital control technology is essential for properly integrating the basic AMAS capability along with other sophisticated target acquisition aids such as voice command, helmet mounted sight, automatic terrain following/avoidance, automatic navigation, etc. Realizing mission effectiveness benefits in a practical, reliable and affordable design requires consideration of a number of important design trade-offs. This paper will highlight some of the more critical building block design considerations which pave the way for the development of integrated flight/fire control systems capable of demonstrating significant fighter mission effectiveness improvements.

JAMES K. RAMAGE

Biographical Sketch

Mr Ramage, Age 38, BS Aerospace Engineer 1967, University of Tulsa, MS Engr Mgt 1971, University of Dayton. Mr. Ramage was initially assigned to the Flight Dynamics Laboratory as a military officer in the U.S. Air Force in October 1967. During the past 15 years, Mr Ramage has been actively involved in a broad range of advanced flight control system development activities. Early in his career, Mr Ramage was responsible for conducting in-house analyses, and simulations in support of a number of flight control development programs, including the XV-4B V/STOL, C-130 Gunship and the B-47 FBW Demonstrator. From 1972 to 1977, Mr Ramage served as Technical Manager for the Survivable Flight Control System and the Control Configured Vehicles Advanced Development Programs. Mr Ramage has also served on many ad hoc committees for reviewing design, flight safety and survivability aspects associated with the F-16, B-1, YF-16 and YF-17 vehicles. From 1977 to 1979, Mr Ramage was the Program Manager for the Digital Flight Control System (DFCS) Pre-Design Study and is currently serving as Chief Engineer for the Advanced Fighter Technology Integration (AFTI/F-16) Advanced Development Program.

QUANTUM LEAP IN AVIONICS

W. E. Cantrell

General Dynamics Corporation
Fort Worth Division
P. O. Box 748
Fort Worth, Texas 76101
(817) 732-4811

W. E. Cantrell is Manager of Advanced Avionics at the Fort Worth Division, General Dynamics. A graduate of Auburn University, Mr. Cantrell joined General Dynamics in 1952 after Air Force assignments at Keesler AFB and WPAFB. He has been associated with avionic systems on the B-58, F-111 and F-16 series aircraft. He is currently responsible for advanced avionics suites on the F-16XL and derivative aircraft and manages the Avionic Systems Department's discretionary funds and lab programs.

ABSTRACT

Current standardization levels in such programs as the F-16 are providing benefits of productivity and growth that have been significant in the success of that program. The ever-increasing drive to performance, multi-use systems and diverse weapons has heavily taxed current avionic resources. In addition, the data transfer requirement is complicated by the high speed data flow that modern computers both feed on and produce; by multiple source-multiple destination video distribution requirement; the need to self-test the system to lower levels; and the desire to dynamically reconfigure from a failure. Fortunately, the technology to achieve solutions to these new problems is evolving in the VHSIC and fiber optics programs, so that it is possible to rearchitecture the system at the module level as opposed to the LRU level. Module level standardization around a small number of types allows a large number of system level combinations while achieving economies of scale at the module level. The usual objection to standardization, that it freezes innovation, is avoided by technology transparency provisions; while at the same time the objection that standardization obsolesces the present is avoided by downward compatibility provisions. Candidates for standardization in this approach include bus interfaces, the system network, modules and racks.

"Copyright © 1982 by General Dynamics Corporation
All Rights Reserved"

BACKGROUND

In two iterations of the F-16 avionics suite, the objectives of the DAIS program have been incorporated in a production program. Initially, the F-16 adopted the architecture, interface and software standards of DAIS (Figs. 1 and 2). More recently, in the MSIP II program, the DAIS architecture standard was extended to a multibus structure, the 1750A instruction set was dictated for processor functions, the current HOLL (J73) was adopted, programmable interactive multifunctions displays were incorporated and the weapons inter-

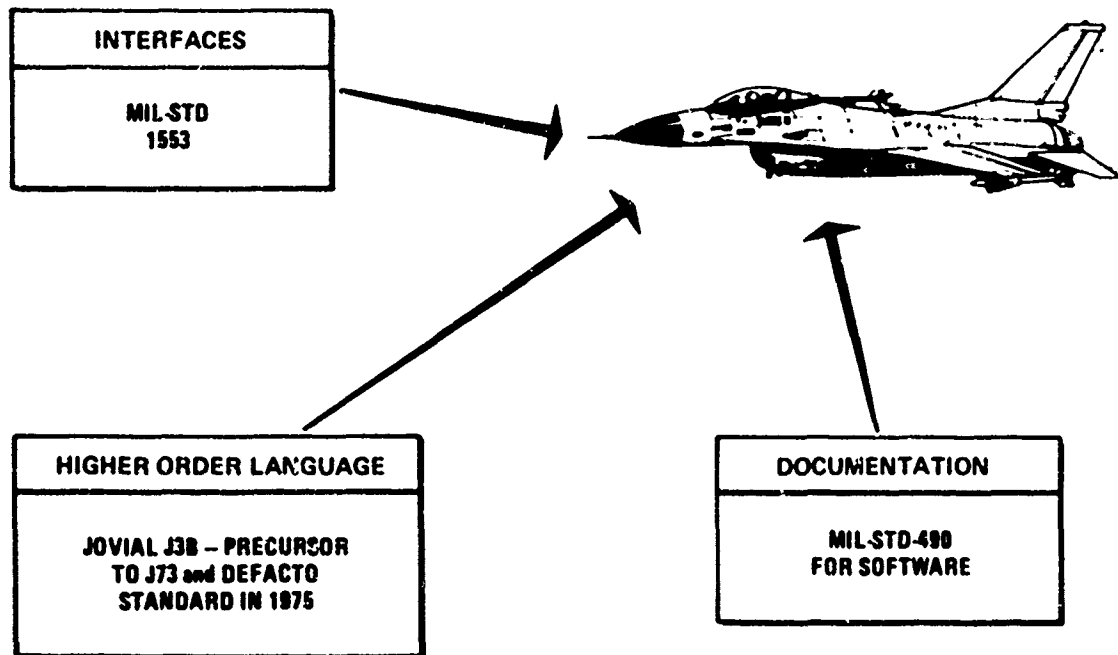


Fig. 1 STANDARDS AVAILABLE IN 1975 WERE APPLIED TO F-16

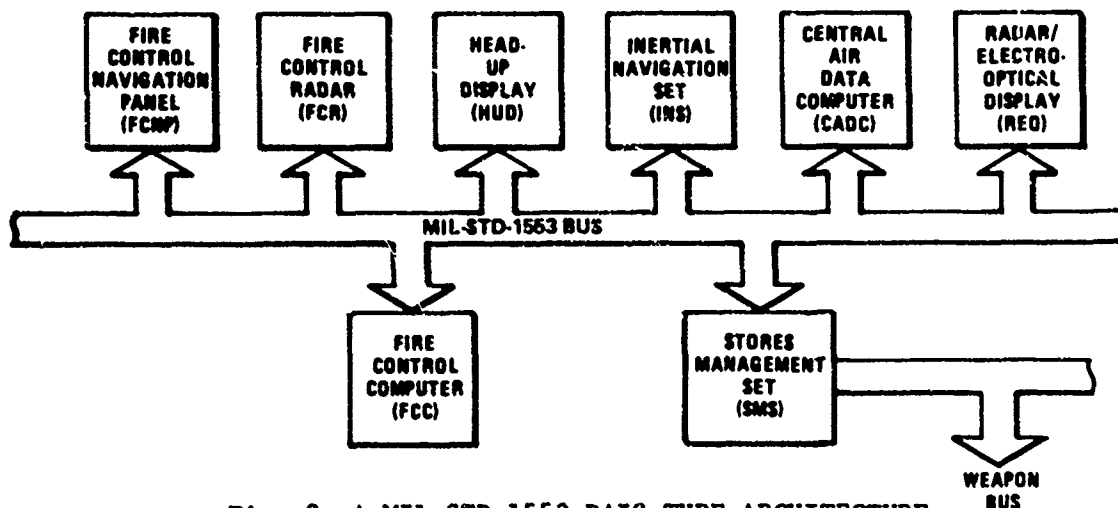


Fig. 2 A MIL-STD-1553 DAIS-TYPE ARCHITECTURE WAS IMPLEMENTED

face standard (MIL-STD-1760) was adopted (Fig. 3). This puts the F-16 in the forefront of standardization for USAF programs (Fig. 4). At the same time,

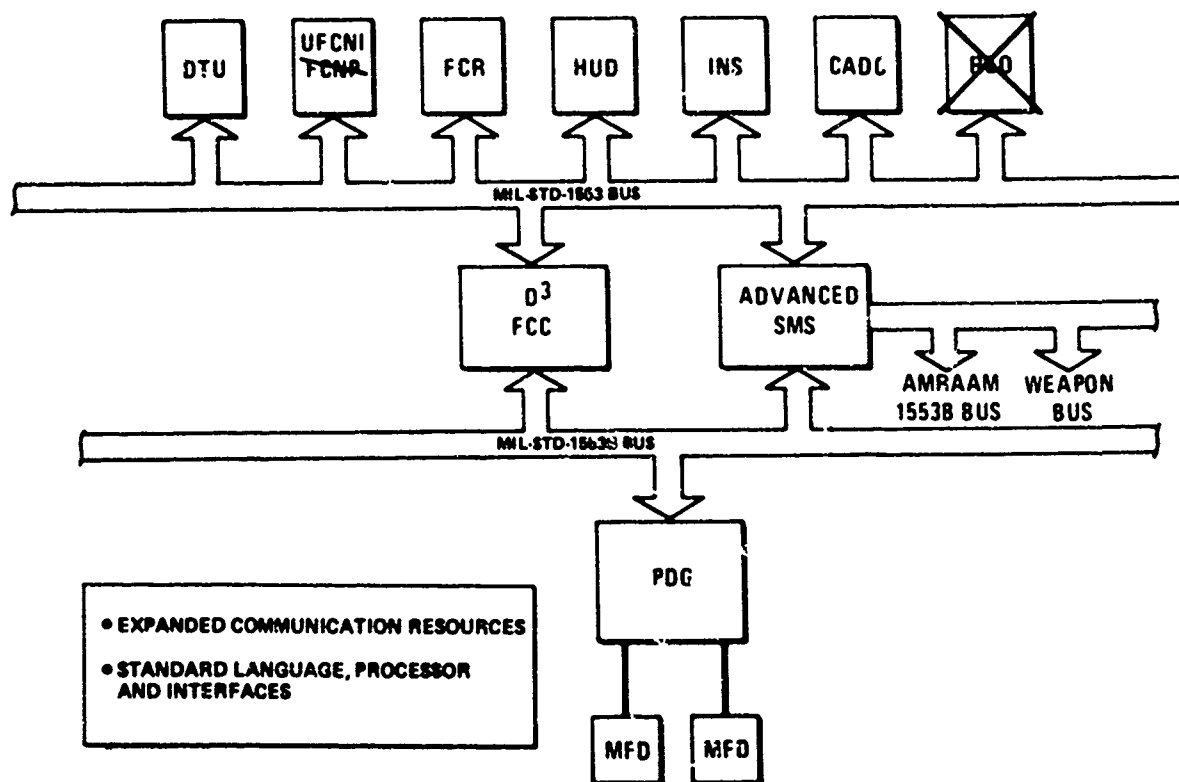


Fig. 3 MSIP: LOGICAL EXTENSION OF F-16 ARCHITECTURE

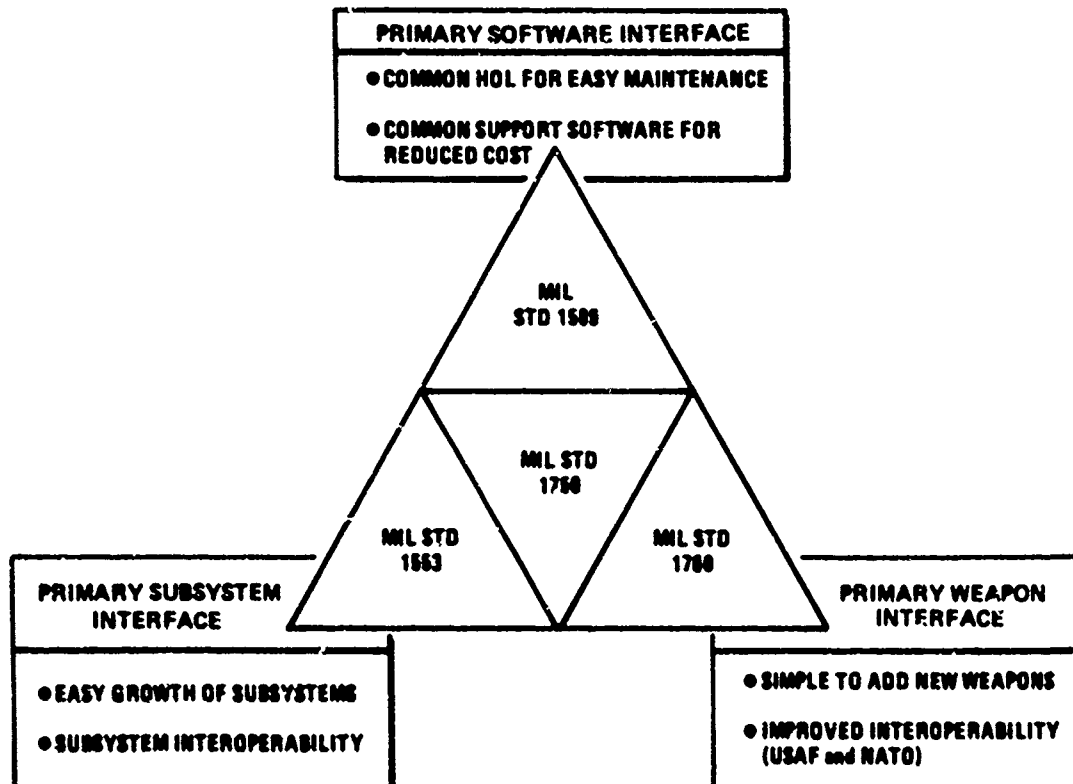


Fig. 4 SYSTEM LEVEL STANDARDIZATION WAS FULLY EMBRACED

however, (Fig. 5), it was becoming apparent that to meet the system demands of the future, a new approach would be needed. First, a system level

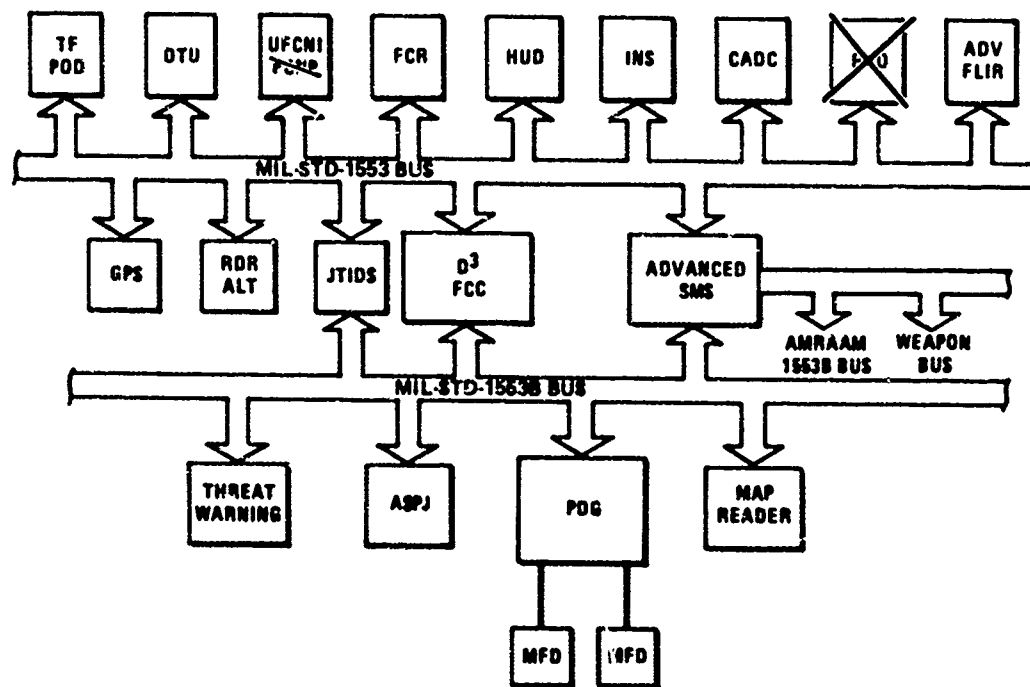


Fig. 5 MSIP ARCHITECTURE EXTENDED FOR FUTURE GROWTH

architecture and interface was needed to accommodate the non-avionics as well as the avionics systems to accomplish centralized control, display, and fault detection and reporting. Second, as more and more elements were interfaced to the bus, duty cycle limitations forced first a multi-bus, then a hierarchical bus, bringing with it the overhead and data latency problems of pass-through. Third, because the modern processor feeds on and generates high speed data and because modern sensors and weapons generate multiple video with multiple destinations, a high speed distribution system was overlaid on the MIL-STD-1553 information transfer system. With all of this complexity came complex wiring, many connectors, costly mother boards, customized cards and LRUs and support problems (Fig. 6). Fortunately, new technologies are evolving that promise new solutions to these problems, at the same time fostering standardization objectives.

- CURRENT STANDARDS HAVE NOT LED TO INTERCHANGEABILITY
- DATA PATHS INDUCE TROUBLESOME LATENCY
- HIGHER SPEED DATA IS REQUIRED FOR NEW SYSTEM FUNCTIONS
- MULTIPLE SOURCE-MULTIPLE DESTINATION VIDEO IS REQUIRED
 - MIL-STD-1780 REQUIREMENT
 - SENSOR FUSION REQUIREMENT
- THE WIRING AND CONNECTOR PROBLEMS ARE STILL WITH US
- LRU LEVEL STANDARDIZATION IS INFEASIBLE, EXPENSIVE

APPLICATION OF NEW TECHNOLOGY TO NEW STANDARDS WILL AVOID CURRENT LIMITATIONS

Fig. 6 A NEW APPROACH IS REQUIRED TO GROW BEYOND MSIP

NEW TECHNOLOGIES

The most significant new technology to be exploited are the single chip implementation of complex functions in VLSI/VHSIC (Fig. 7). The power of this

- DIGITAL PROCESSORS
 - POWERFUL, SINGLE-CHIP PROCESSORS
 - LOW COST
- DIGITAL MULTIPLEX
 - MAJOR INTERCONNECT REDUCTIONS
 - SINGLE-CHIP MULTIPLEX TERMINALS
- GENERAL VLSI/VHSIC TECHNOLOGY
 - SINGLE-CHIP COMPUTER MEMORIES
 - SINGLE-CHIP CUSTOM FUNCTIONS
 - MICRO-ANALOGS
 - "FRONT-END" APPLICATIONS

Fig. 7 SIGNIFICANT SINGLE CHIP FUNCTIONS

technology makes it possible to economically encompass on single standard modules not only the normal processing tasks, but also the overhead functions of interface, I/O, self-test and parallel bus management. The latter makes it possible to configure a "least case" processor with access through other standard modules to memory and special functions to configure a variety of processing tasks and architectures.

These technologies must be applied in concert and in a revolutionary manner, but the payoffs can radically change the USAF standardization posture. Instead of being faced with the simplicity or complexity choices of today, we will be able to implement complex functions in simple standard hardware. Instead of selecting between quantity and quality, we will be able to build large quantities of standard high quality modules. Undesirable choices will not have to be made.

Current line replaceable units (LRUs) will disappear in favor of on-airplane replaceable modules housed in integrated module racks that remain on the airplane. The avionic spares crib will contain a small number of multiple-use standard modules instead of a large number of diverse electronic SRU and LRU types. Avionic functions will self-test before, during, and after flight to determine correct operation and, in critical cases, will heal themselves by substitution of on-line, hot spares. Failures will be automatically isolated to a single, on-aircraft replaceable electronic module, thereby eliminating the need for most of the Avionic Intermediate Shop (AIS). Along with these changes, the cadre of avionic and test equipment technicians needed to support the aircraft will be greatly reduced both in numbers and skill levels.

These sweeping avionic standardization changes will also extend to the depot level. LRU maintenance will disappear and many of the standard modules used in the avionics will be low-cost, high reliability, throwaway items. Consequently, an avionic depot repair facility for these modules will not be required. Since the modules will be transparent to technology and will be purchased to a form, fit, and function interface standard, replacement modules will be built with the then-current technology rather than with that of the original buy, thus keeping pace with advancing technology. As a result, the present problems of providing SRU repair parts in a constantly changing technology environment will disappear.

The promises of the future for improved standardization are not utopian. They are achievable with current technology. Three of the key advances that can enable new avionic designs to obtain the desired logistic benefits are:

1. Low-Cost, Single-Chip Digital Processors
2. High-Speed, Single-Chip Digital Multiplex Terminals
3. Single-Chip VLSI/VHSIC Technology
 - Computer Memories
 - Standard Interface Test Chips
 - Standard Functions

The key element of these technologies is size reduction. As size shrinks, bringing reductions in cooling and less requirements for power, it becomes evident that the opportunities for implementation of common hardware can become a reality. For example, the size of a MIL-STD-1553 digital multiplex terminal has shrunk from three 5" x 7" electronic cards in 1976 to a single 5" x 7" card today and will shrink to a single 4" x 5" card by 1984. The next step will reduce the size of such a terminal to a pair of VLSI integrated circuit chips (Fig. 8). Given a standard module package and standard casings and fittings, all avionic equipment could then utilize the same multiplex terminal hardware.

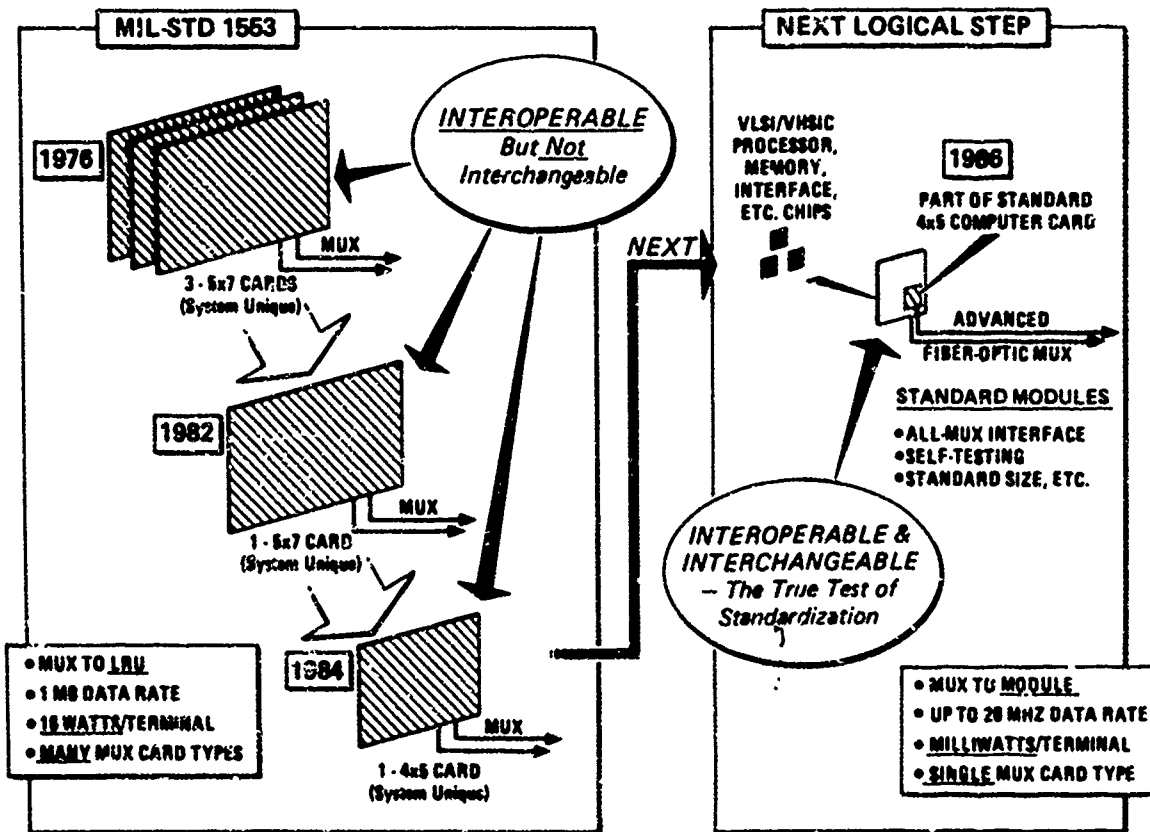


Fig. 8 MULTIPLEX TERMINAL TRENDS

Most important, however, is the application of this technology on a broad front (Fig. 9). This will result in meeting the decisive needs and promises of future systems rather than making only small incremental improvements.

Independent applications of all of these technologies in the normal manner cannot produce the order-of-magnitude gains that are achievable through a concerted application.

- STANDARD MODULES
- ADVANCED MULTIFLEX
- ARCHITECTURE
- EXTENSIVE ON-BOARD SELF-TEST
- INTEGRATED RACKS

Fig. 9 NEW TECHNOLOGY IS APPLIED ON A BROAD FRONT

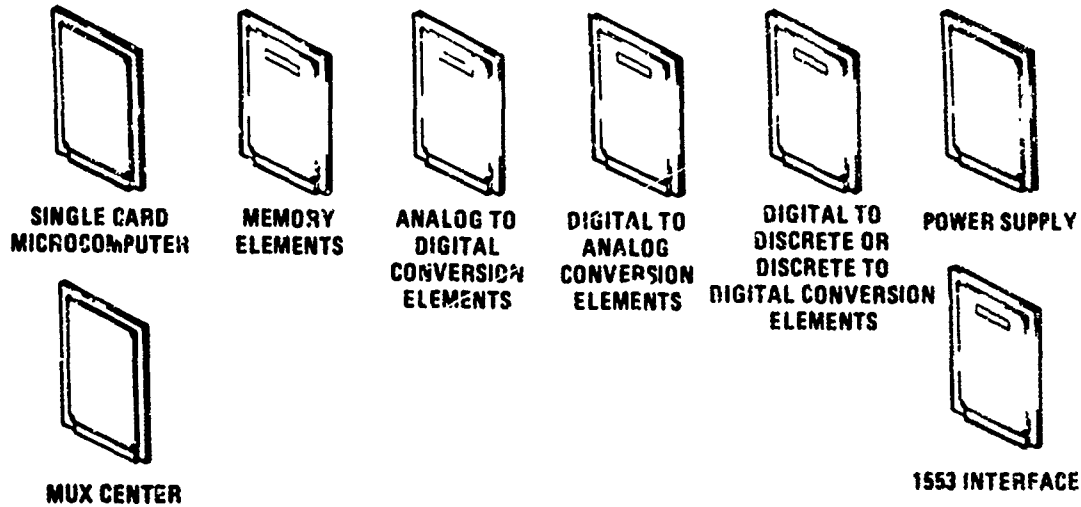
Analysis of various types of avionics systems has shown that identical types of functions are performed in many different systems and in different parts of the same systems. Figure 10 shows how this commonality of functions is shared between a group of five aircraft systems. An unusual combination of systems has been selected to dramatize the commonality of functions even among diverse systems. If the more conventional avionic systems are added to the list, the same sharing of function types is also observed.

COMMON FUNCTIONS AVIONIC SUBSYSTEMS	Computing	Analog/ Digital Interface	Digital/ Analog Interface	Digital/ Analog Control	Discrete Interface	Frequency- Digital interface	Mass Memory	Power Supply	Serial Digital Multiplex
FLIGHT CONTROL	✓	✓	-	✓	-	-	✓	✓	✓
ENGINE CONTROL	✓	✓	✓	✓	✓	✓	✓	✓	✓
ENVIRONMENTAL CONTROL	✓	✓	✓	✓	✓	-	✓	✓	✓
AIR INLET CONTROL	✓	✓	-	✓	-	-	-	✓	✓
AIR DATA & MOTION SENSORS	✓	✓	-	-	-	-	-	✓	✓

Fig. 10 AVIONIC SUBSYSTEMS CAN BE PARTITIONED INTO COMMON FUNCTIONS

APPLICATION OF NEW TECHNOLOGIES TO NEW STANDARDS

In today's avionic designs, each of these common functions is performed by a unique hardware design. Typically, different vendors will provide different hardware even though the functions are identical. This situation exists because current designs emphasize LRUs (circa World War II) rather than functions. On the other hand, if standard interfaces and packaging are adopted (as is possible with a unified systems architecture), it becomes practical to design standard functional modules for multi-use applications (Fig. 11). These modules, plus unique sensor and effector interface modules, plus unique sensor and effector interface modules, then become the building blocks for a new type of system architecture. Virtually any type of system function can be built from these modules together with suitable software. Because the common module types will be used in many different applications, it will be cost-effective to develop special VLSI circuits and production methods that will permit such modules to be manufactured in large quantities at low cost.



ALTOGETHER, ABOUT 20 STANDARD MODULES ARE IN AN F-16 MSIP CLASS SYSTEM. THESE SAME MODULES HAVE APPLICABILITY TO LAND, SEA AND AIR SYSTEMS AND SUPPORT EQUIPMENT

Fig. 11 COMMON FUNCTIONS CAN BE PERFORMED BY STANDARD MODULES

Figure 12 contains a general description of one such module and lists some of the more important features. Such a computer module is currently feasible using the MIL-STD-1750A processor chip set being developed by the F-16 program. Other modules of the family would be of similar construction.

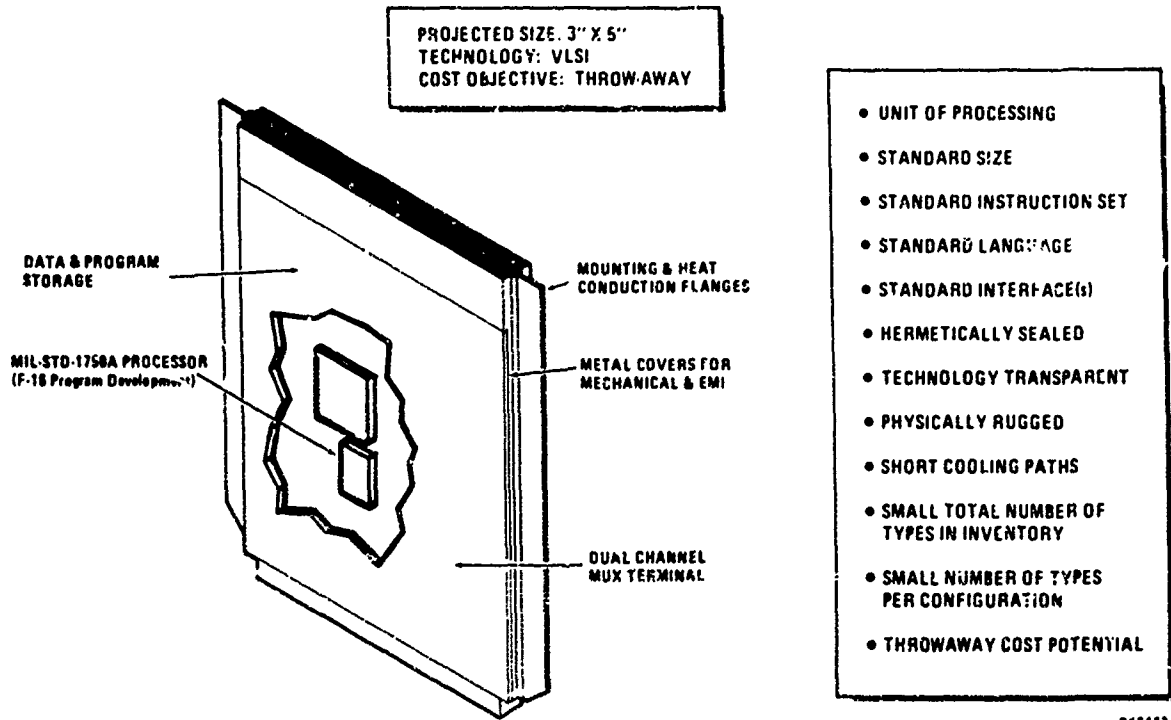


Fig. 12 TYPICAL STANDARD COMPUTER MODULE

Modules of the type shown in Figure 12 will be physically protected from flight line environment to which they will be exposed. For this reason, hermetic sealing will be employed. The modules will become the line replaceable units and therefore must be designed accordingly. Current module or card design approaches will not suffice.

A new type of modular architecture will be necessary to utilize standard modules of the types discussed (Fig. 13). Multiplex communication will be used between modules and functional clusters of modules and not just between LRUs as in existing designs. This approach will largely eliminate many thousands of mechanical electrical connections that are used in current avionic equipment. It is ironic that, while these connectors facilitate rapid field replacement of defective elements, they also contribute failures that increase the number of maintenance actions. In modern digital equipment, even a momentary break in a connection tends to register as a hard failure. Evidence indicates that connection related problems may be responsible for a large segment of the could-not-duplicate (CND) and re-test-OK (RTOK) problems that (1) tax maintenance resources and (2) tend to repeat in flight and reduce combat effectiveness. The same standard, digital multiplex communications interface is used at all levels to simplify design and permit necessary data interchange at all levels of the system (Fig. 14).

Advanced multiplex networks of the type needed for such applications have already been designed and breadboarded. These networks employ advanced data switching techniques to provide the necessary data transfer rates to

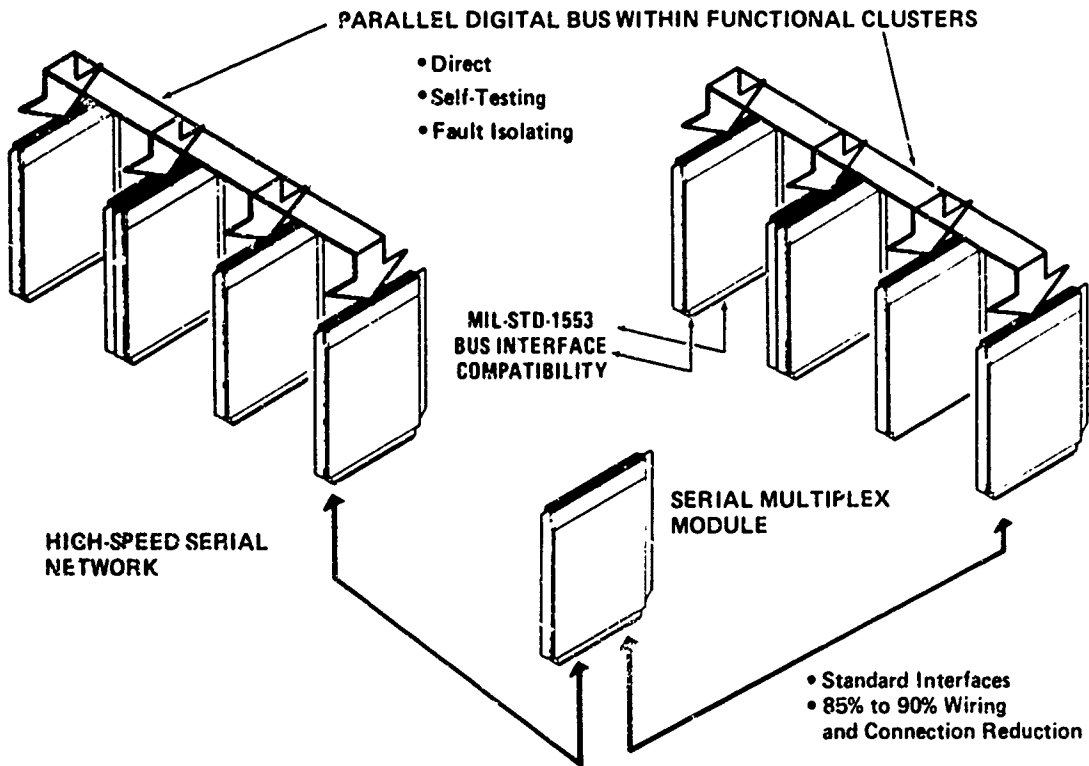


Fig. 13 ADVANCED MULTIPLEX COMMUNICATIONS

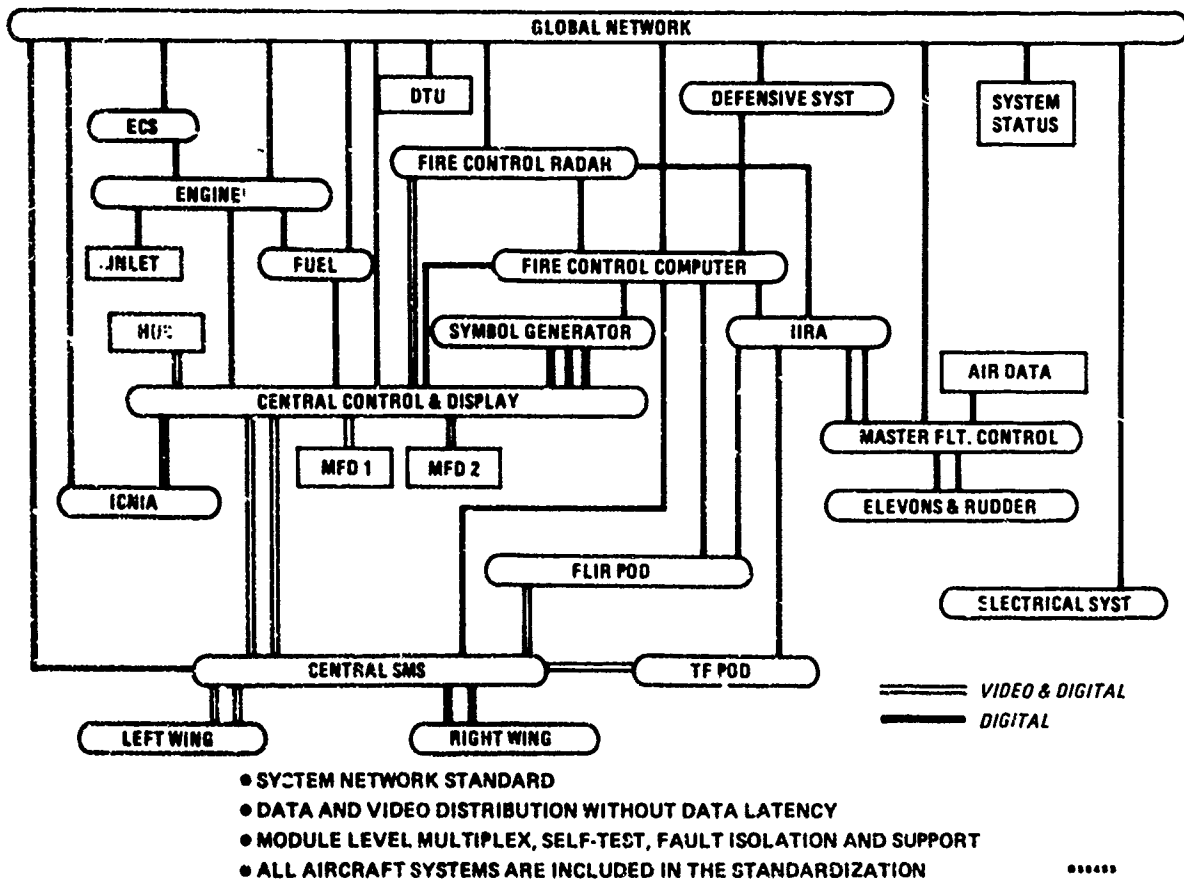


Fig. 14 A NEW ARCHITECTURE RESULTS

handle both high-speed digital and wide-band video type data. The terminals transmit less than one-quarter watt of power and can be constructed entirely with VLSI chip technology. The only remaining step is to reduce the hardware to VLSI integrated circuit chips suitable for use in small, standard modules.

This architecture has the added benefit that with the proper interface module, an early implementation of an advanced architecture can be achieved at the LRU level (Fig. 15, left picture). The LRUs internally use the new architecture (and, of course, the current processor instruction set and software standards), but interface the existing system on a MIL-STD-1553B bus. Conversely, a system configured for the advanced architecture can communicate through a MIL-STD-1553B interface to a current inventory LRU. Thus, bad actors can be replaced, good LRUs retained, and advanced configurations incrementally implemented. Both front-end and life cycle economics are benefitted.

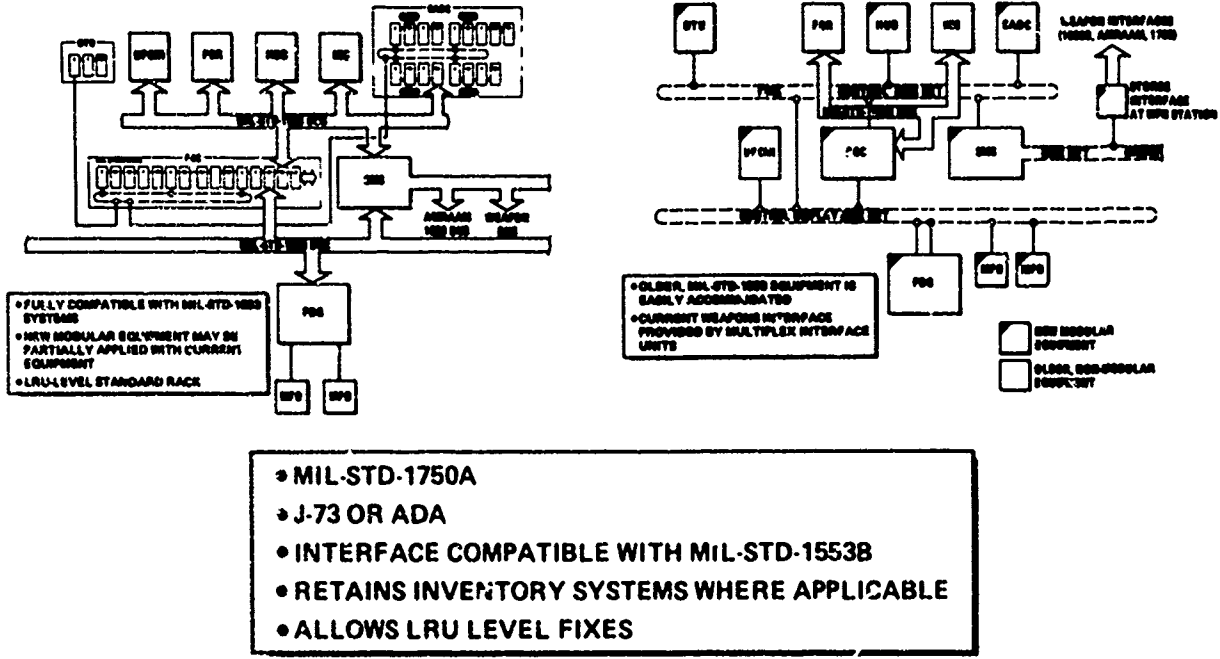
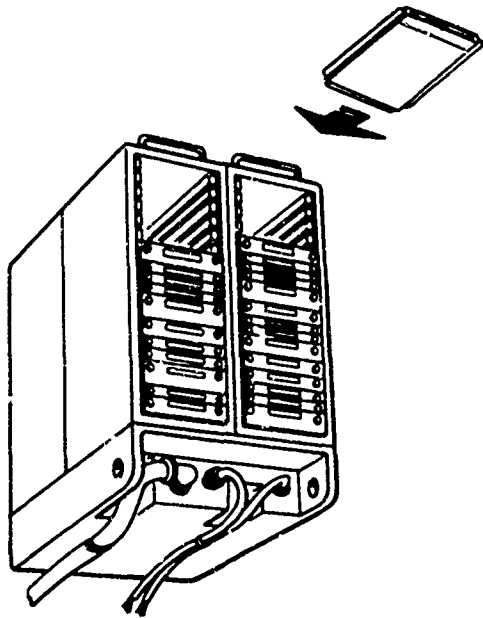


Fig. 15 AND...OLD STANDARDS ARE PRESERVED

Direct module replacement at the airplane level will be a major logistic benefit of the new technology avionics. To achieve this goal, an integrated rack packaging will be used in place of existing LRUs. Racks similar to that shown in Figure 16 will permit ready access to individual modules. Many of these common integrated racks will be used throughout the airplane and can be larger or smaller depending on application and installation constraints. The rack sections will be separately removable from the aircraft to permit back-plane repairs or modifications. Compared to current avionics, these repairs should be very infrequent, since all racks will utilize back-plane wiring that is reduced by approximately two orders of magnitude from that of current

avionics. Individual modules will be enclosed in sealed metal cases to provide complete mechanical and EMI/EMP protection. These rugged, sealed modules will permit flight line replacement. All modules will be cooled by conduction to cold plates in the integrated racks. Either forced air or liquid cooled versions of the rack may be used.



- EASES SUPPORTABILITY
- DIRECT MODULE ACCESS
- FLIGHT LINE REPLACEMENT OF MODULES
- ELIMINATES HIGH DOLLAR LRU'S
- INTEGRATED RACKS REMOVABLE
- REDUCTION IN SKILL LEVELS
- BASIS OF AVIONIC COMMONALTY BETWEEN AIRPLANE TYPES

Fig. 16 INTEGRATED STANDARD RACKS REPLACE TRADITIONAL LRUs

Standardization doesn't just happen (Fig. 17). Typically, many road-blocks are thrown up because new standards break with the past (inertia, cost, schedule risk) and old standards are perceived to be anti-progressive. There are, however, many technical and management approaches to achieving acceptable and rational standardization. The benefits outweigh the objections. Probably the most difficult objection is the one that standards do not foster innovation. The answer to that is technology transparency.

Technology transparency guarantees that the standard is constructed to allow innovation, flexibility and growth (Fig. 18). Since our design philosophy is a "least case" baseline processor augmented with other standard modules, interfaced on a standard bus to other modules in a functional cluster and to the system on a standard network, technology advances simply upgrade the performance and shrink the augmenting modules onto the parent module without affecting the F^3 of the module or the racks (by contrast, LRU level standardization tends toward the "worst case" design). In fact, as the shrink under-populates the rack, new functions can be accommodated without physical growth and reconfiguration.

In a business sense, technology allows F^3 procurement of modules from competitive sources. The technology in these modules allows a choice between cost and/or performance.

TECHNICAL

- TECHNOLOGY TRANSPARENCY
- SUBSETS OF WORST CASE
- DOWNWARD COMPATIBILITY
- FLEXIBILITY, BUT NOT AT THE EXPENSE OF INTEROPERABILITY
- GRACEFUL DEMISE WHEN SUPERCEDED
- EXTENDABILITY AND GROWTH

MANAGEMENT

- JOINT USERS GROUP TO ACHIEVE CONSENSUS
- INFORMED RESOLUTION OF ISSUES AT THE HIGHEST LEVEL
- LIFE CYCLE COST ADVANTAGES
- COMPLIANCE DIRECTIVES AND FUNDING
- BUSINESS INCENTIVES

Fig. 17 REQUIREMENTS FOR RATIONAL STANDARDIZATION

- MODULE IS TRANSPARENT TO TECHNOLOGY WITHIN THE CASE
 - ✓RETAINS PHYSICAL AND ELECTRICAL (F³) INTERFACE
 - ✓ELIMINATES PROCUREMENT OF OLDER TECHNOLOGY SPARES
- USER BENEFITS FROM TECHNOLOGY GAINS
 - ✓PERFORMANCE UPGRADES
- ALLOWS FOR INNOVATIONS
 - ✓INTEGRATION BETWEEN MODULES
 - ✓CHANGES WITHIN MODULES
 - ✓REDUCE MODULE TYPES BY COLLAPSING BACKPACKS INTO PARENT MODULES
 - ✓FLEXIBILITY WITH SOFTWARE
- MULTI-SOURCE PROCUREMENTS POSSIBLE

Fig. 18 TECHNOLOGY TRANSPARENCY

While these standardization benefits will largely solve most of the problems being experienced today and probably make an affordable Air Force possible in the future, there is no assurance that this will occur. Although large, decisive improvements are critical for survival and victory, they are culturally difficult to implement. A revolutionary application of new technology will require a revolutionary change in both the USAF and industry (Fig. 19). Procurement of avionic systems and spares will undergo a dramatic change. Industry product lines and alignments will change. USAF procurement policies will be altered. Standard modules will be procured directly by the military from module sources and will be provided as GFE to avionic vendors.

- AF AND INDUSTRY -
- MAJOR PORTIONS OF ALL SYSTEMS WILL BE BUILT FROM A SMALL SET OF STANDARD, MULTI-USE MODULES
 - ▼ Changes Company Product Lines and Alignments

 - SYSTEM FUNCTIONAL DIFFERENCES WILL CONSIST PRIMARILY OF ALGORITHM/SOFTWARE IMPLEMENTATIONS AND CUSTOM SENSOR/EFFECTOR MODULES
 - ▼ New Development Approaches and Software Management

 - STANDARD, MULTI-USE MODULES MAY BE MULTI-SOURCED BY A SINGLE AGENCY FOR MULTI-SYSTEM USE
 - ▼ New Organization Charters and Procurement Policies

 - SOME TRADITIONAL FUNCTIONS WILL WITHER OR DISAPPEAR
 - ▼ Intermediate Shop (Sources, Personnel, Training, Deployment)
 - ▼ Maintenance Training and Complex T.O.s
 - ▼ Depot Repair

Fig. 19 MODULAR AVIONICS WILL ALTER AF AND INDUSTRY ORGANIZATIONS

Avionic system developers will find themselves creating special sensor and effector modules and function-unique software to be used with standard modules common to many other uses. Because most functions of the Avionic Intermediate Shop will disappear, the large organizations now associated with this function will be greatly reduced. With large numbers of throwaway modules, the depot repair facilities and organizations will shrink, or the function will revert to the original manufacturer. These changes can provide far more Air Force fighting power per dollar. The task is technically achievable. The challenge is to break free of the comfortable post World War II path of avionic design and support. Instead of incremental applications of advanced technologies with incrementally small improvements, a revolutionary and concerted technology application to gain a decisive advantage should be made.

SUMMARY

In summary (Fig. 20), standardization is right. It has been demonstrated to be right on the F-16. To make that "rightness" practicable requires the capability to grow on the past, at the same time incorporating the best that technology can provide without disruptive redevelopment. Module level standardization is the right level for these objectives, but business and management incentives are required for palatable acceptance.

- STANDARDIZATION HAS COME A LONG WAY AT THE SYSTEM LEVEL
- NEW TECHNOLOGIES OFFER NEW STANDARDIZATION OPPORTUNITIES
- DOWNWARD COMPATIBILITY IS IMPORTANT
- TECHNOLOGY TRANSPARENCY ALLOWS INNOVATIONS
- MODULE LEVEL STANDARDIZATION IS "RIGHT"
 - DEVICE LEVEL IS TOO LOW FOR TECHNOLOGY TRANSPARENCY
 - LRU LEVEL IS TOO HIGH - TOO CUSTOMIZED/TOO COSTLY
 - MODULE LEVEL PROVIDES LARGE COMBINATIONS OF SYSTEMS CAPABILITIES
- KEY TO ADOPTION IS THE BUSINESS INCENTIVE
- STANDARDIZATION DOESN'T MEAN LESS BUSINESS, JUST A NEW WAY OF DOING BUSINESS

Fig. 20 SUMMARY

INTEGRATED DIGITAL AVIONICS SYSTEM (IDAS)

Peter Boxman

US Army Avionics R&D Activity
4300 Goodfellow Blvd.
St. Louis, MO 63120
314-263-1634

ABSTRACT

IDAS is a continuous level of effort, engineering development program geared to a systems application approach to accommodate the following:

NEED:

- Limit avionics hardware proliferation
- Limit multiplex bus proliferation in terms of bus software, protocol, and architectures, specifically to promote standardization
- Arrest repetitive multiplex architecture development by aircraft manufacturers during Full Scale Engineering Development (FSED)
- Reduce Operational and Support (O&S) cost of avionics systems
- Optimize and standardize cockpit control design for multiple aircraft application
- Provide a vehicle for channeling and implementing Army and tri-service tech base efforts and products

FORMULATED TO MEET REQUIREMENTS OF:

- TRADOC Letter of Agreement (LOA) to implement a standard IDAS for new and modernized aircraft
- Army Aviation Mission Area Analysis (AAMAA)
- Carlucci Initiative/Recommendation 21
- Army Science Board
- Air-Land Battle 2000 Report

TECHNICAL APPROACH, 1983-85:

- Simulate and Validate
 - Standard core architecture (i.e., those avionics systems with their associated controls and displays that can be implemented across the Army fleet independent of type or mission of aircraft)
 - Flexibility for technology updates
 - Accommodation of mission peculiar avionics (i.e., ASE, Weapons Electronics) without affecting software integrity of core system

PRODUCT:

- Standard set of multiplex compatible GFE
- Validated Interface Control Documents (ICDs) and competitive procurement data (F³) for non-GFE avionics
- Validated ICDs for mission peculiar avionics
- Handoff to each aircraft FSED and Product Improvement Program (PIP)

BIOGRAPHICAL SKETCH

AUTHOR: PETER BOYMAN

TITLE: CHIEF, AIRCRAFT SUPPORT BRANCH
AVIONICS SYSTEMS INTEGRATION DIVISION
US ARMY AVIONICS RESEARCH & DEVELOPMENT ACTIVITY
HQ US ARMY AVIATION RESEARCH & DEVELOPMENT COMMAND
ST. LOUIS, MO

PREVIOUS ASSIGNMENTS WITHIN THE AVIONICS R&D ACTIVITY HAVE BEEN:
AVIONICS PROJECT LEADER FOR COBRA MODERNIZATION AND BLACK HAWK.

EDUCATION: BSEE, 1969, NEW JERSEY INSTITUTE OF TECHNOLOGY
MSEE, 1973, FAIRLEIGH DICKINSON UNIVERSITY

PROFESSIONAL SOCIETIES: INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE)
ARMY AVIATION ASSOCIATION

Abstract

EMBEDDED COMPUTER STANDARDIZATION IN THE SUBMARINE ADVANCED COMBAT SYSTEM (SUBACS)

The Submarine Advanced Combat System is a modular, distributed combat system for 688 class attack submarines (SSNs), currently in concept development. SUBACS will be among the first weapon systems to employ the AN/UYK-44(V) computer and the AN/UYS-2 Enhanced Modular Signal Processor (EMSP). The SUBACS architecture utilizes commonality and distributed processing to provide flexibility in reconfiguration and growth. To achieve this, standardization is exhibited at many levels within the system. This talk will focus on the development and evolutionary plans of the SUBACS and the utilization of standard computing devices in those plans.

Biography

Mr. Ronald L. Ticker
Systems Engineer
Submarine Combat Systems Project (PMS-409)
Naval Sea Systems Command
Washington, D. C. 20362

BS University of Maryland
Graduate Studies - George Washington University

1979

Current Assignment:

Mr. Ticker's system engineering responsibilities include:

- o AN/BQQ-5C software
- o AN/UYS-1 TRIASP implementations in AN/BQQ-5 & SUBACS
- o AN/UYS-2 EMSP, AN/UYK-44(V) & RASS implementation in SUBACS

STANDARDIZED COMPUTING SYSTEM SDS80

J. Olsson

Ericsson Defense and Space Systems Division
Box 1001
S-43126 Molndal, Sweden

BIOGRAPHY

Mr. Olsson received the M.Sc. Degree at Chalmers University of Technology, Gothenburg, in 1977. He is currently working as project manager for Computer Systems in Command, Control and Communications Systems. From 1977 to 1980, he worked with high order languages and program development environments for standardized computer systems.

ABSTRACT

The Standardized Computing System SDS80 is a complete hardware/software package. Through its modularity and versatility, SDS80 is well suited for various military applications e.g. in aircraft and command and control centres as well as non-military applications.

The main design goal has been the reduction of life cycle costs, especially the cost for software development and maintenance.

SDS80 consists of:

- Computers D80 and D80M (Micro)
- High order languages Ada and PASCAL/D80
- Program Development System AIDS (Ada Integrated Development System) and FUS80 (Program Development System for Pascal/D80)

The D80 Computer is a modular design using the latest technology (Ceramic Chip Carrier) to reduce weight and size. High Order Language support and 32-bit word length are other features. When using Ada, world-wide software support is available.

Responsible for its design and development are Telefonaktiebolaget LM Ericsson and its Defense and Space Systems Division, Ericsson Information Systems and SRA Communications, all members of the ERICSSON Group. ERICSSON is one of the world's leading telecommunication companies and Sweden's largest producer of digital computers.

Test and qualification of airborne and groundbased versions of D80 will be completed during 1982. For the new Swedish multirole aircraft JAS, the D80 Computer has been selected as systems computer as well as the computer for the radar and display systems. Production of D80 modules will thus continue until beyond the year 2000. The Ada implementation is depending on a continued research program ordered by the Swedish Air Materiel Department.

BACKGROUND

By the mid-1970's, the Swedish Air Materiel Department experienced the situation common to all major embedded computer system procurement agencies, that of:

- More hardware subsystems being enhanced by embedded computer systems.
- Software, as opposed to hardware, rapidly becoming the dominating development and maintenance cost of computer systems (see Figure 1).

The JA37 fighter aircraft was taken as the case in point. Here, no fewer than six major subsystems (Air Data, Inertial Navigation, Central Computer, Display, Automatic Flight Control and Radar) contained embedded computer systems. Each computer system was independently developed and contained no Line Replaceable Unit (LRU) common with any other subsystem. Each computer system was programmed in its own native assembly language requiring individual software development tools in conjunction with different host computers. The governmental maintenance facilities had to match this development with individual test equipment, spare parts and specialized personnel.

The Air Materiel Department therefore, initiated discussions involving the major Swedish companies responsible during JA37 development in order to develop a standardized computing system concept which would reduce subsystem development costs and maintenance resources. Related activities, notably with the U.S. Department of Defense (DoD), were also monitored during this time. The SDS80 concept then evolved to encompass three main spheres of interest:

- The adaptation of a High Order Language (HOL)
- The specification of a Software Development System
- The design of a standardized computer supporting the HOL

The cost benefits envisioned by such a concept are outlined in Table 1.

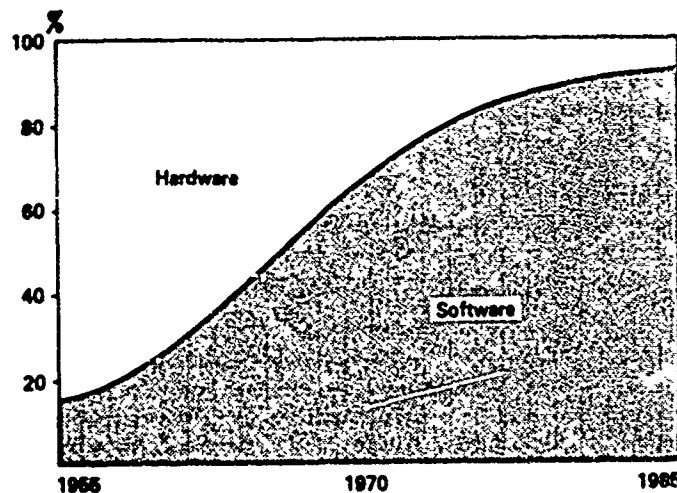


FIGURE 1. HARDWARE/SOFTWARE COST RELATIONS

Qualifying the above, SDS80 provides an integrated hardware-to-software concept which primarily reduces individual subsystem software development and maintenance costs. Further savings are then made if SDS80 is applied to other subsystems contained within a common (e.g. aircraft) system (see Figure 2).

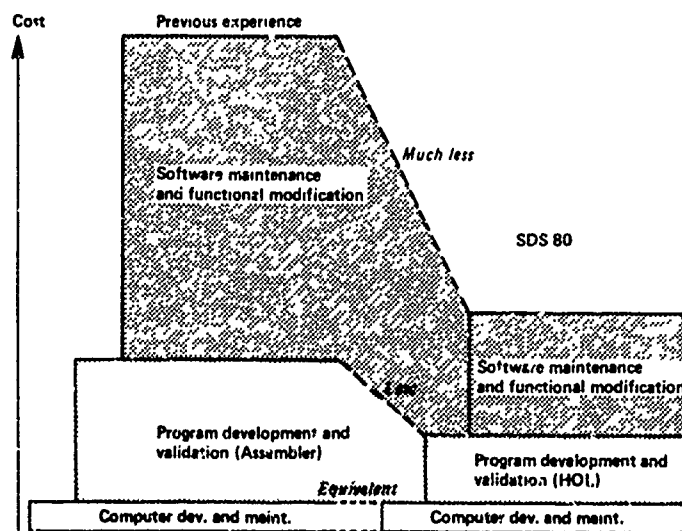


FIGURE 2. RELATIVE LIFE-CYCLE COSTS FOR INDIVIDUAL SUBSYSTEMS EMBEDDED COMPUTER SYSTEM

<u>PHASE</u>	<u>SOFTWARE</u>	<u>HARDWARE</u>
Development	Increased Programmer Productivity Increased Program Reliability Reduced Checkout Time Uniform Documentation Common Subroutine Libraries	Reduced Parallel and Overlapping Activities Effective HOL Code
Production	Better Configuration/Control Easier Program Edition Generation	Lower Manufacturing Cost Due to Larger Prod. Runs
Life Cycle Maintenance	Facilitated Personnel Trng Common Test Programs/Procedures	Reduced Specialized Test Equipment Spare Part Commonality Fewer Different LRUs
Functional Modification	Less Lead-In Time More Secure Changes	Easier System-Wide Up-grading

TABLE 1. SDS80 COST BENEFITS

HIGH ORDER LANGUAGES (HOL)

Within the SDS80 concept, there are two high order languages available:

- Ada
- PASCAL/D80

Both languages are based on the PASCAL language.

Ada is well-known by now and should not need any further presentation. Consideration concerning language subsets, availability and direct micro-program support are covered in other documents.

The name given to SDS80's original HOL is PASCAL/D80. It is now fully implemented in the SDS80 system. It may be considered a superset dialect of standard PASCAL where often direct support for extra features are given in the D80 microprogram. Major differences between PASCAL and PASCAL/D80 are the constructs which are non-existent in PASCAL but are extremely necessary for embedded computer systems.

Experience has shown that a programmer educated in PASCAL has not only little problems in learning PASCAL/D80, but that the programmer can readily understand PASCAL/D80 code at first glance. The majority of programmers will work at the module level (which corresponds to the PASCAL construct "PROGRAM"). A more elite group would be responsible to organize the MODULES into PROCESS'es run on certain PROCESSORS within a SUBSYSTEM. Thus, the modular structure of the language follows the programming team organization of sophisticated software projects.

The PASCAL/D80 compiler can be viewed to have two modes of operation. First, that it will compile and then link (with aid of a LINKER) the defined statements within the modular structure of the language and produce executable code for a D80 computer system. This is referred to as the Target Compiler. Secondly, executable code can be produced for the software development system's Host Computer in order that non-real-time, algorithmic tests may be made on the code. This is referred to as the Host Compiler.

SOFTWARE DEVELOPMENT SYSTEMS

The SDS80 Software Development Systems are referred to as AIDS (for Ada) and PUS80 (for PASCAL/D80). They are integrated collections of software tools with a recommended set of host hardware units designed to:

- Aid in the development of Ada and PASCAL/D80 Programs
- Maintain good documentation
- Assist in configuration and control of projects
- reduce verification and validation costs

The host hardware units are:

- Host Computer - Digital Equipment Corporation's (DEC) VAX 11/780 with a minimal VMS configuration
- Operator's Control Panel (OCP) with key-board display

The OCP has a number of unique features normally not offered with embedded computer systems:

- One OCP may service up to eight subsystems simultaneously. Each operator has his own keyboard display and "logs-in" to one of the subsystems in the data net.
- The OCP contains a floppy disk of 256 kbyte capacity, which is globally available to all operators (i.e. subsystems).
- The OCP has a direct data link with the Host Computer thus permitting direct Host to Target loading plus access to symbolic data files created by the Compiler/Linker. It is via the latter that an operator may validate a program using symbolic names as opposed to absolute memory addresses.

See Figure 3.

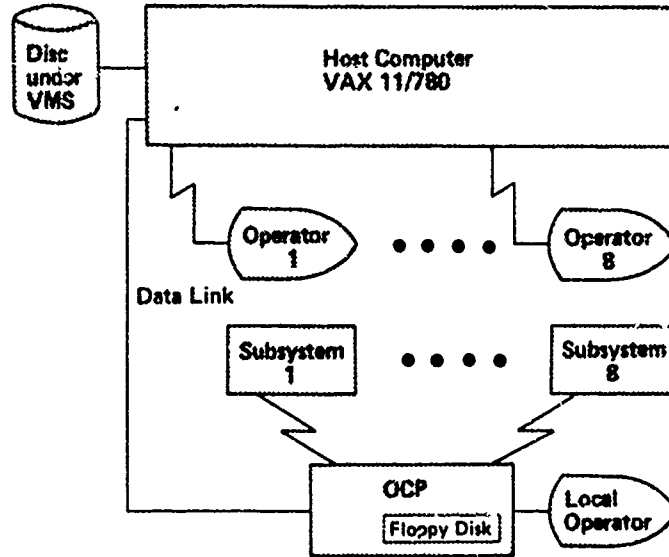


FIGURE 3. SOFTWARE DEVELOPMENT SYSTEM - HARDWARE INTERFACES

The software supported by AIDS and PUS80 has been organized into three groups, referred to as General, Host and Target Utilities. The software components making up each utility group plus the groups themselves are linked together via a dialog "thread of control". (See Figure 4). The philosophy is to lead the user through a controlled path ensuring that the correct tool be properly used at the appropriate time. As the user becomes more experienced, prompt messages in the dialog may be bypassed.

Integrated with the dialog is a help information facility (invoked by "?") so that a user may check the options available at any time. As supported by the Host operating system's help information facility, information is requested in the form of subtopics within topics thus offering a nearly self-educating capability for the user.

Since users will most likely develop certain specialized tools for their individual applications, the dialog supports a controlled use of DEC Command Language (DCL), so that full Host operating system facilities may be used at the different utility group levels. This is also so that commands for printing lists, filing texts, cleaning up directories, etc. may be executed within the framework of the dialog.

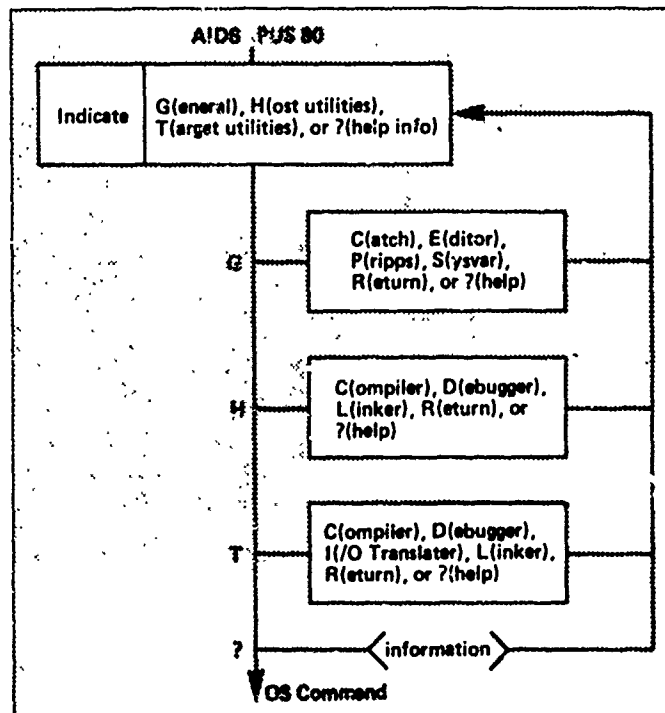


FIGURE 4. SOFTWARE DEVELOPMENT SYSTEM-INTERACTIVE DIALOG

STANDARDIZED PROCESSOR

To specify one standard computer (or family of computers) was viewed at the outset as a compromise situation. Instead, SDS80 supports a standardized processor where its architecture supplies the building blocks needed for relatively simple to extremely high performance embedded computer systems. Each such computer system is referred generically as a D80 and would appear in a minimum configuration as shown in Figure 5.

The D80 computer system is configured in accordance with the modular language structure of HOLs. The realtime, input/output and operating system HOL constructs are single machine instructions or natural hardware data types. The D80 instruction repertoire is designed for HOL implied data access and statements.

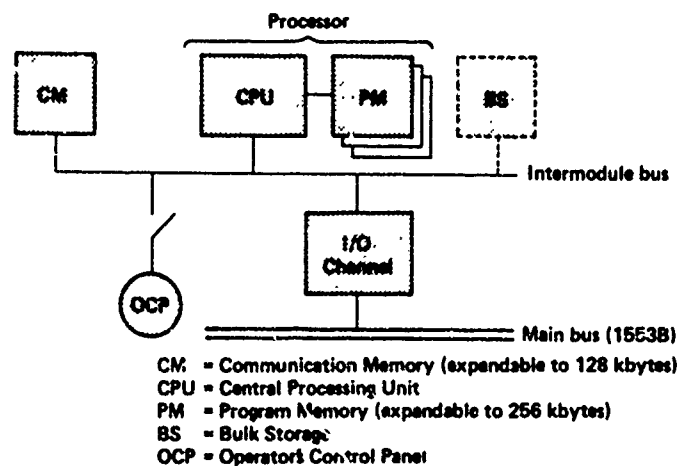


FIGURE 5.

In order to increase computing capacity and performance, D80 may contain up to 15 processing units (Processors, I/O channels, Bulk Storage) connected to the intermodule bus. This means that the D80 hardware supports expansion and flexibility from a single to a multi-processor computer system.

CM is the common (global) variable memory shared by all processors connected to the intermodule bus. Additionally CM contains a processing unit which controls the traffic on this bus by polling all units and building directive/message queues to/from the various units. Communication with OCP is also done via CM. The updating of the subsystem clock ($\approx 64 \mu s$ resolution) is also a CM function.

Input/Output Channels contain 30 data boxes of 64 bytes each. The actual configuration is programmable and data access from any processor is done by normal load/store instructions from/to the logical continuation of CM's address room. Software tools are available to perform this configuration and integration. Because of SDS80's aircraft system parentage, I/O channels have initially been designed to interface with a US MIL-STD-1553B data bus.

BS is a non-volatile storage device intended to contain the software for one or more D80 subsystems plus to be used for registration purposes. BS is implemented as a 4 Mbyte (expandable to 8 Mbyte) unit, divided into 128 (32-bit word) sectors. BS is accessed as a continuator of CM's address room.

Each D80 processor is made up of a CPU with its own PM where PM is a memory containing the executable program code and constants. Each D80 CPU can be described as in Figure 6.

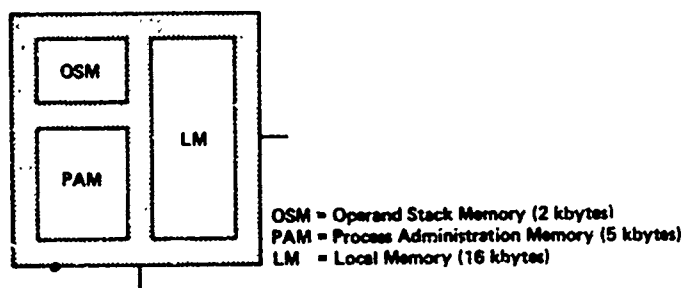


FIGURE 6.

Direct hardware support is given to the administration of up to 15 parallel processes per processor where each process has a built in priority. OSM contains (during program execution) intermediate computational results, parameters to subroutines, instruction operands and is sectioned into equal parts, one section for each process. Similarly PAM contains, individually for each process, process status indication (i.e., active, stopped, interrupted) a program counter (i.e., where in PM the process is executing) and subroutine call/return information (up to 15 nested levels) all automatically updated by the hardware during program execution. LM contains local procedure variables (lexically linked by information in F*) forming up to 15 activation data areas plus a heap, precisely following the structure implied by HOLs. Unlike OSM and PAM, LM memory allocation may vary in size for each process as specified by the software. Extremely efficient local data access/allocation plus low parallel process/multi-processor operating system software overhead is thus obtained via the above.

D80's data word length (e.g. integer) is 32-bits and machine instructions are provided for real (i.e. floating point) arithmetic. Depending on the instruction mix, a single D80 CPU has been nominally rated at 1 Mop/s.

A low-cost microprocessor complement (but not alternative to D80 is also provided within SDS80, referred to as D80M. D80M is a single-processor system with parallel-process support. Computing performance is reduced to about 1/5 compared to a D80 CPU due to a lack of single machine instructions for HOL constructs and real arithmetic. D80M is nevertheless attractive for reducing subsystem costs especially when an additional D80 CPU results in wasted overall computing capacity.

VALIDATION TEST PACKAGES

SDS80 contains verification and validation software for various hardware and software elements. Additionally, there are currently several individual test program packages for the hardware integration and test of all elements making up a D80 computer system. The latter is supported by associated test program tools and special test equipment.

RELATED EXPERIENCE

The SDS80 program was initiated by the Swedish Air Materiel Department. The responsibility to carry out the program was given to three Swedish companies, active in the military electronics field, who have formed the consortium DY 80. These companies are Telefonaktiebolaget L M Ericsson and its Defense and Space Systems Division, Ericsson Information Systems and SPA Communications, all members of the ERICSSON Group.

The parent company of ERICSSON is one of the world's leading telecommunications companies. The main product area covers public switching equipment. Development of computerized telephone exchanges started in the early 60's. Now, the computer-controlled AXE system is ordered by or delivered to customers in some 40 countries all over the world.

The Defense and Space Systems Division of ERICSSON is mainly active in the defense electronics field. Radar systems for air, sea and ground applications are developed and manufactured at this Division. For these radar systems high performance computers, mainly dedicated to signal processing, are developed.

Software and hardware development and production of computerized operation and maintenance systems, supporting analog and digital telephone exchange equipment, are also carried out at the Division.

Ericsson Information Systems, formerly DATASAAB AB, is active in a number of fields, where the products are based on computers, developed and manufactured by the company. Realtime systems for administrative applications, based on minicomputers is a rapidly growing product group of the company. Bank systems and computer terminals (Alfaskop) are also well established products.

SRA Communications are active mainly in the communications field. Mobile telephone systems and radio equipment are the largest contributors to the company's turnover. In the military electronics field, SRA Communications have been responsible for the display systems in the different versions of the 37 Viggen aircraft. SRA also develop and produce jamming equipment, airborne as well as groundbased. Air Traffic Control systems for military and non-military use are also within the company's product range.

TECHNICAL DATA

Software Development System

- AIDS or PUS80
- Host Computer VAX 11/780

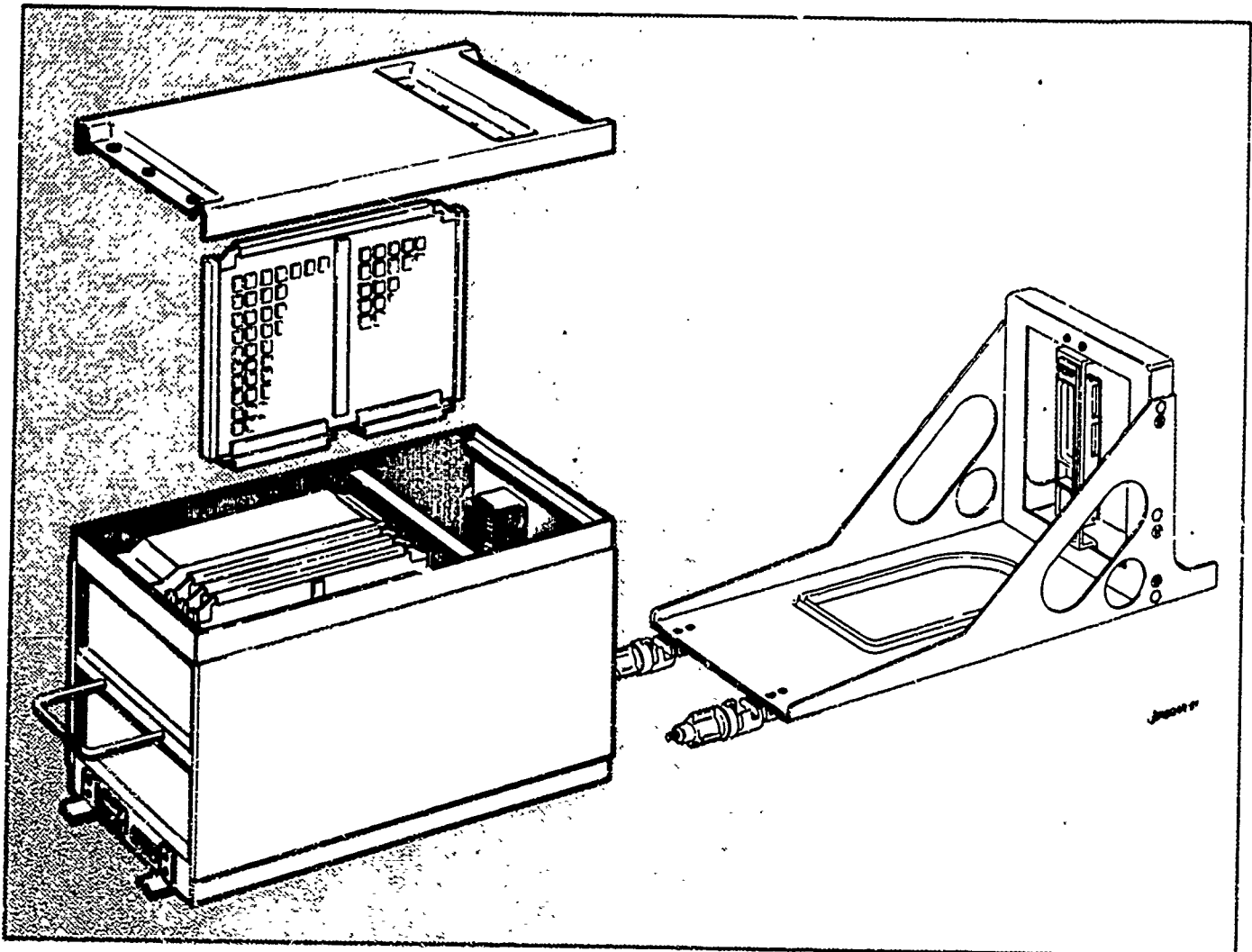
Programming Language

- Ada or PASCAL/D80

COMPUTER D80

- Word Length 32 bit
- Speed 1 Mop/s
- Instruction Time from 0.125 μ s
- Multiprocessing Capability
- Interface MIL-STD-1553B
- Power 56 W, AC or DC
- Weight 6.5 kg
- Dimensions (LxHxD) 125 x 193 x 318 mm
- Cooling Forced air or Fan
- Built-In Test

FIGURE 7.



Abstract

NAVY REAL TIME SIGNAL PROCESSOR DEVELOPMENT: SECOND GENERATION PLANNED SERVICE STANDARD

Planned growth in the coming decade of Navy combat systems will generate signal processing performance requirements that far exceed the capability of the current Navy standard signal processor. There is a further need to improve the programming environment of Navy standard signal processors to increase programmer productivity. The Navy has initiated development of a second generation standard signal processor, the Enhanced Modular Signal Processor (EMSP), nomenclatured as the AN/UYS-2. This paper describes the Navy program to develop the EMSP as a multi-processor signal processing system. The approach to specifying system performance and programming environment along with an acquisition approach meant to encourage vigorous competition for the engineering development contract award is discussed. The commodity management concept for EMSP's in-service lifetime involves interface management within the system and controlled technology infusion. This important plan to stay abreast of technology and to meet user community requirements for product stability is described.

Biography

CAPT C. B. Robbins
Deputy Project Manager,
Navy Shipboard Tactical Embedded Computer Resources Project
Naval Sea Systems Command
Washington, D. C. 20362

BS	U. S. Naval Academy	1961
MS	Naval Postgraduate School	1972
EE	Naval Postgraduate School	1972
	Graduate, Naval Nuclear Power Program	1965

Experience

22 years with the Navy, including the following assignments:
USS Hammerberg DE 1015
USS Somers DD 947
USS Detector MSO 415
Undersea Surveillance Project (PME-124)
Commanding Officer, U. S. Naval Facility Centerville Beach, Ca.

Current Assignment: Deputy Project Manager, Navy Shipboard Tactical Embedded Computer Resources Project (PMS-408)

Responsible for development and acquisition of Enhanced Modular Signal Processor (EMSP) AN/UYS-2

TRI-SERVICE COMBINED ALTITUDE RADAR ALTIMETER (CARA)
THE ARMY PERSPECTIVE

WILLIAM M. GILL
US Army Avionics R&D Activity
Fort Monmouth, New Jersey
201-544-4403

Biographical Sketch

Mr. Gill is Project Leader for Radar Altimetry the at USA Avionics R&D Activity. Current projects include the Army Standard Radar Altimeter, the AN/APN-209A(V) and the Tri-Service Combined Altitude Radar Altimeter (CARA).

Mr. Gill received a B.S. Degree in Engineering Physics from South Dakota State University in 1967 and an MSEE from Fairleigh Dickinson University in 1975. He is a member of the IEEE and the Association of the U.S. Army.

The CARA is being developed under an Air Force contract as a replacement for thirteen different types of radar altimeters now in the Air Force inventory. These thirteen types of radar altimeters range in age from seven to thirty eight years and in MTBF, from 39 to 570 hours.

The Army requirements for CARA are for fixed wing aircraft; the OV-1D Mohawk and the new JVX Joint Services Advanced Vertical Lift Aircraft for which the Army is the executive agency.

The OV-1D requirement provides a replacement for the AN/APN-171A(V).

The CARA requirement planned for 1,086 JVX aircraft was dictated by:

-high altitude requirement. CARA provides radar altitude from 0 to 50,000 feet; JVX mission altitudes go to 30,000 feet. The standard Army altimeter, the AN/APN-209A(V), optimized for helicopter usage has a range of 0-1,500 feet; the standard Navy altimeter, the AN/APN-194(V), has a range of 0-5,000 feet.

-MIL-STD-1553B Bus Compatibility. The CARA system being designed for the F-16 aircraft will have 1553B compatibility. Since the JVX will utilize a 1553B bus, the Army will attempt to standardize on the CARA system, including the 1553B bus, Interface Adapter, being designed for the F-16.

-Nuclear Hardening. The Army is coordinating the nuclear hardening requirements for CARA with the Air Force, in an attempt to standardize on requirements and also to fully satisfy the JVX requirements.

The Army has been participating in the development of the CARA indicator, required for the OV-1D, to assure ANVIS-Compatibility. An Infrared suppression filter under development by the Army for the AN/APN-209A(V) Radar Altimeter will provide a standardized solution to ANVIS-Compatibility problems, and will have tri-service applicability.

TRI-SERVICE COMBINED ALTITUDE RADAR ALTIMETER (CARA)

THE ARMY PERSPECTIVE

Contents

- I The CARA Program
 - A. Program Initiation
 - B. CARA Technical Description
- II The Army Requirement
 - A. Retrofit Applications
 - B. The JVX Aircraft
 - 1. MIL-STD-1553B
 - 2. Nuclear Hardening
 - 3. Compatibility with Night Vision Goggles

THE CARA PROGRAM

Program Initiation

The CARA program began in FY-77 as an Air Force investigation into the low reliability and high support costs associated with Air Force radar altimeters. The Air Force had thirteen different types of radar altimeters in inventory, ranging in age from seven to thirty eight years and ranging in MTBF from thirty nine to 570 hours. Annual support costs for these altimeters exceeded \$10 million per year.

A Class IV C modification program¹, with Warner-Robins Air Logistics Center as manager, was authorized in September of 1980. This program was to replace approximately 4000 existing Air Force altimeters with the CARA, using interface adapters for each type of aircraft. No aircraft wiring changes were to be required. Warner-Robins contracted ARINC to write a specification² for the CARA. This specification was coordinated with the Army and the Navy. Subsequently, a fully coordinated RFP was let to industry and a contract was awarded to Gould, Inc. in Jan 82.

CARA Technical Description

The CARA (Figure 1) is an all solid state FM/CW radar altimeter system which provides accurate altitude (+/-2 feet +/-2 percent) data from zero to 50,000 feet AGL. Each CARA system consists of a receiver-transmitter (R/T), two antennas, height indicators as required, an interface adapter as required, a mount for the R/T, antenna mounting plate adapters as required, and adapter cable assemblies as required. All of these various subsystems are divided into Group 1 items, where design is independent of aircraft type, and Group 2 items which are peculiar to the aircraft type or series.

The R/T consists of a one-watt maximum power output transmitter with power management circuitry that will reduce the power output depending on terrain and altitude. The RF sweep is segmented into seven segments for resolution and accuracy and is swept both up and down to compensate for doppler effects. The receiver utilizes low-frequency band-pass tracking filters to track the leading edge of the return signal, allowing measurement of altitude to the nearest object. Built-in test circuits provide fault detection on a continuous, interruptive, and manual self-test basis.

The height indicator, when used in the given aircraft configuration, controls all of the functions of the CARA system, such as power, manual self test, low altitude warning set, and display dimming. The display provides analog altitude from zero to 5,000 feet, digital altitude from zero to 50,000 feet, visual R/T fail warning and low altitude warning. The indicator utilizes two microprocessors to format the altitude data received from the R/T and to also control the self test functions.

The interface adapters are used to process the R/T altitude data and provide outputs to make the CARA compatible with other onboard avionics systems that previously received this data from the replaced altimeter. The interface

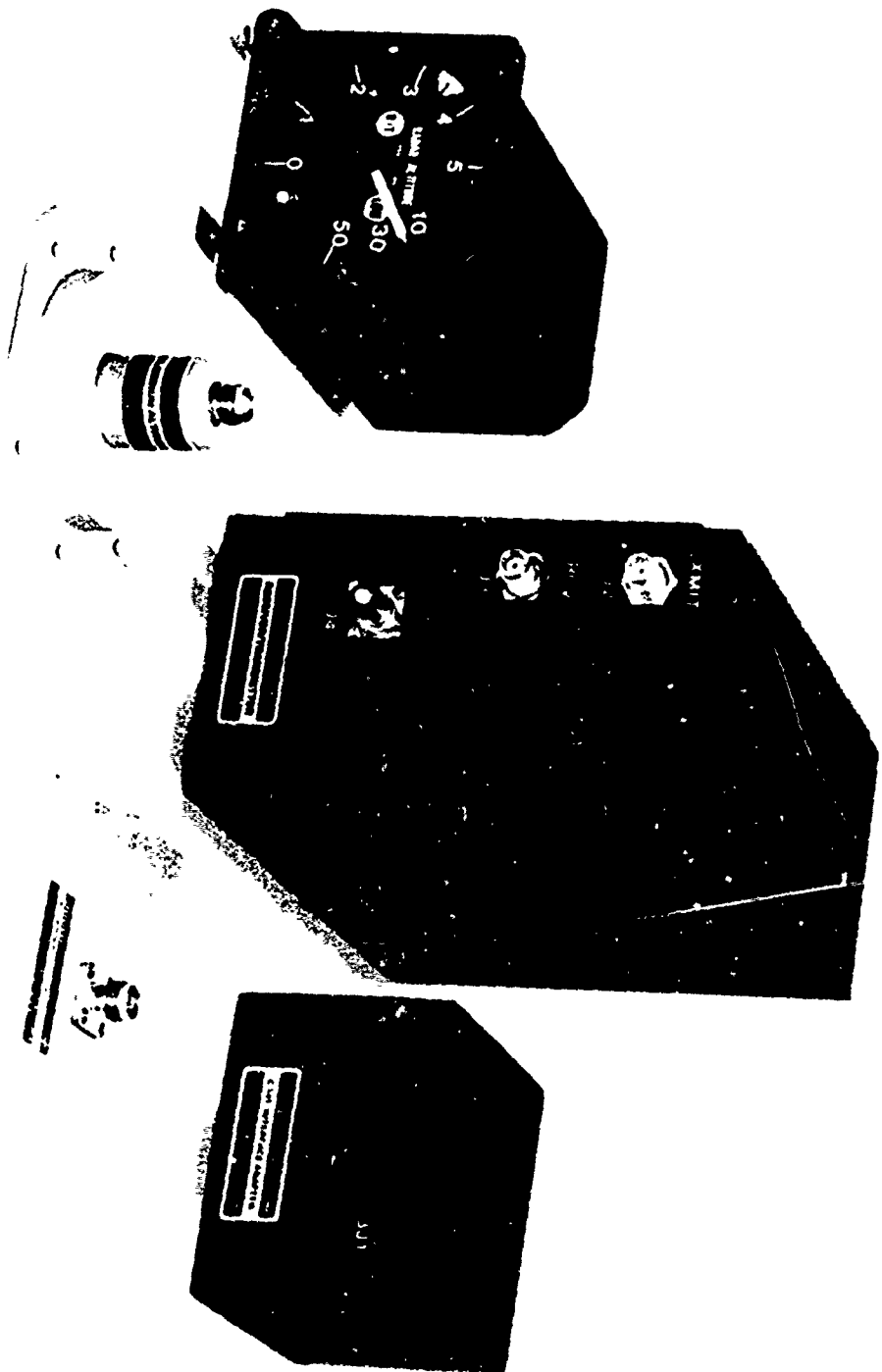


FIGURE 1
CARA COMMON ITEMS

adapters provide the electrical signals peculiar to any aircraft to allow the CARA to be a standard altimeter and yet be compatible with the many retrofit and forward-fit installations.

Three antenna configurations are available with CARA: (1) Fixed Wing aircraft with a 40 by 60 degree three dB beamwidth, (2) Rotary-Wing aircraft with a 35 by 35 degree three dB beamwidth and (3) A-7 aircraft antennas with a 65 by 65 degree beamwidth. In some aircraft installations, the CARA antennas are placed on adapter plates so they will exactly replace the existing antennas, with no aircraft changes.

THE ARMY REQUIREMENT

Retrofit Applications

The Army currently has two types of aircraft that utilize the aging AN/APN-171A; the OV-1D Mohawk and the EH-1H Quickfix. The APN-171 has a terminal logistics date of 1988, after which it will no longer be supportable. The cost to retrofit these old altimeters exceeds the CARA acquisition cost. Also, the APN-171 has only a 5,000 foot range, whereas the CARA provides a 50,000 foot range. An aircraft product improvement is planned for FY-85 to retrofit the fleet of 170 OV-1D aircraft with the CARA. The APN-171 altimeters released by this retrofit could be utilized to support the in the ten EH-1H aircraft.

The JVX Aircraft

The main thrust of Army interest in the CARA program is to provide a high altitude radar altimeter for the Joint Services Advanced Vertical Lift Aircraft (JVX), for which the Army is the executive agency. Since the JVX mission package requires a radar altimeter that operates to 30,000 feet, neither the Army Standard AN/APN-209A(V) with a range of 1500 feet, nor the Navy Standard AN/APN-194(V) with a range of 5,000 feet would suffice.

The JVX schedule includes a preliminary design study phase of twenty three months duration to be awarded in 1QFY83. This will be followed by a full scale engineering development phase commencing in 4QFY84. Production release is planned for 1QFY88 to meet the Marine Corps Initial Operational Capability (IOC) of Jan 1991. Present program plans call for production of 1086 aircraft over a ten year period.

The JVX is planned⁴ as a tilt rotor aircraft that will perform a wide variety of missions: efficient hover and terrain flight; external load and rough field operations; efficient long range, high speed cruise; high altitude loiter; and high load factor maneuvering. The Army requires JVX to perform its Corps and Division SEMA mission replacing the OV-1D, RV-1D, RU-21, RC-12D, EH-1 and EH-60 aircraft.

The Air Force JVX will perform missions of combat search and rescue (CSAR) and special operations replacing or supplementing the H-53, the HH-60D and the MC-130.

The Navy requires a replacement for its HH-3A CSAR aircraft.

The Marine Corps, with the largest JVX requirement, will utilize the aircraft for amphibious force projection and land assault, replacing the CH-46 and the CH-53.

The JVX established unique requirements which can not wholly be met by any existing radar altimeter. Of those available, the CARA most nearly matches the requirements. These requirements include: high altitude, zero to 30,000 feet; MIL-STD-1553B bus compatibility; a nuclear hardening specification; and ANVIS (AN/AVS-6) compatibility.

MIL-STD-1553B

The CARA design requires an interface adapter to provide the embedded 1553 compatibility. For standardization and interoperability, the Army plans to utilize the interface adapter being designed for the F-16 CARA system to meet the JVX 1553B requirement. The F-16 interface adapter is being designed to accommodate both the F-16 version of 1553 as well as 1553B. The choice between these two formats will be available by pin selection on the interface adapter. In addition, the interface adapter has address select pins and an address select parity pin that allows the CARA terminal address to be selected externally. These factors are necessary for Army applications.

There are some minor format differences between the CARA data word codings and the draft word codings of the SAE A2K Task Group, which was tasked to develop standard 1553 word coding formats. For example, the CARA data word coding shows a least significant bit (LSB) of 2.5 feet, as the draft A2K format guidelines specify an LSB of two feet.

Since the CARA interface adapter is being designed with a microprocessor controller, it will be possible to program the adapter with different scale factors to make the unit fully meet Tri-Service/Army requirements. A suggested approach is to design an interface adapter that is standard so no changes, not even software changes, would be required. This is the Army's objective for its participation in the CARA interface design process.

Nuclear Hardening

There are two general types of damaging radiation to be considered in nuclear hardening. The electromagnetic pulse (EMP) is caused by a rapid expansion of ionized gasses that displace the earth's magnetic fields. The effect of the EMP on electronics could be compared to the effects of a lightning strike. Hardening against EMP, at least for military systems, is an established set of design criteria and is well in hand. The CARA EMP design generally meets Army requirements.

The second type of damaging radiation is collectively referred to as TREE (Transient Radiation Effects on Electronics). TREE consists of total dose, high transient dose rates and neutron fluence.

Total dose includes neutrons and gamma rays which, when they pass through the crystal lattice structure of semiconductors, cause release of charged carriers. These charged carriers then cause a high level of leakage, which

either destroys the semiconductor or as a minimum causes a system malfunction. Of the three TREE effects, total dose, as a result of fallout could cause the most problems on the battlefield, while dose rate and neutron damage effects would occur only close to a nuclear detonation.

Dose-rate effects are most damaging to computer memories, especially of the NMOS variety. Other effects are latchup, which is a high current, low voltage condition that can rapidly cause component burnout.

Neutron fluence damage tends to be inversely proportional to the gain-bandwidth product of the semiconductor component. For switching and power transistors, which have a low gain-bandwidth product, damage can occur at a level well below that which would seriously harm a person.

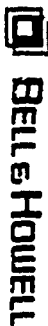
The CARA has a relatively severe nuclear hardening requirement, compared with other Air Force avionics systems. But as severe as these requirements are, the JVK has nuclear hardening requirements considerably more severe. The Army standard AN/APN-209(V) Radar Altimeter meets the JVK hardening requirements.

In summary, it appears that the tactical mission of Army aircraft with their battlefield operations, establishes nuclear hardening requirements that are quite different than those imposed on Air Force equipments. These differences present a standardization issue which must be addressed to provide the systems developers adequate guidance to allow standardization.

Compatibility with Night Vision Goggles

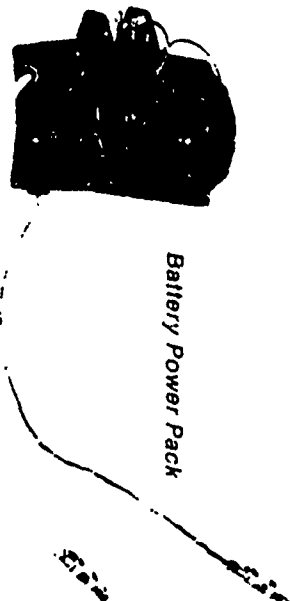
A particularly difficult design problem encountered with the CARA was to make the indicator compatible with the Army's new Aviator's Night Vision Imaging System (ANVIS, AN/AVS-6), (Figure 2). This is a third generation night vision goggle system that is extremely sensitive in the red visible and infrared portion of the spectrum (wavelength 600-1100 nanometers), (See Figure 3). The CARA, and other recent radar altimeters such as the Army AN/APN-209A(V), utilize three separate lighting systems, namely, the light-emitting diode (LED) digital readout, the low altitude warning light and the integral edge lighting that lights the dial face. For existing designs, such as the APN-209 which utilizes red LED's and incandescent lamps (with their inherent high infrared emissions) for the warning and edge lighting, the retrofit problem is very difficult to resolve. The Army has solved the problem by changing the red LED's to green LED's and developing a compound optical filter that is laminated to the altimeter cover glass to filter out the red and infrared from the incandescent lamps. (See Figure 4). This filter utilizes both absorptive and "hot mirror" reflective techniques and can be made extremely thin (.020 inch). Filters that were available, with the proper filtering characteristics, tended to crystallize under certain environmental conditions and hence were unacceptable for this application. Also, available filters were relatively thick and would not fit the existing altimeter bezel.

For a new design such as CARA, the problem is not as difficult. The CARA will utilize green LED's for the digital readout. The edge lighting may be either filtered incandescent lamps, in an externally replaceable lamp module, or electroluminescent (EL) lighting, which is ANVIS compatible. The warning lamps could be a dual light configuration using incandescent lamps for daytime visibility and green LED's for night goggle compatibility.



Ordering Information
 ANVIS system
 Part No. 022753
 Soft cloth carrying case
 Part No. 022791
 Shipping and storage container
 Part No. 022790
 Daylight training accessories
 Part No. 022780
 Special purpose test set
 Part No. 022430
 Maintenance manual
 Part No. 263-20 and 263-30
 Aircraft power converter
 Part No. 022749

ANVIS System consists of binocular assembly, image intensifier tubes, helmet visor interface and battery power pack.



Battery Power Pack

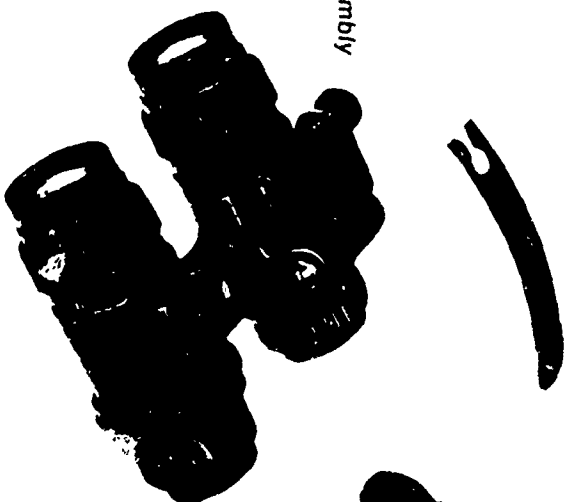


Nominal Specifications (Applies to both 2nd and 3rd Generation unless otherwise indicated)

	3rd Gen.	2nd Gen.
Resolution	.86cy/mr	.72cy/mr
Moonlight	55cy/mr	40cy/mr
Photocathode Type	Gallium Arsenide (1000 micro amps per lumen)	S-25 (300 micro amps per lumen)

- Focus Range 25cm to Infinity
- FOV 40°
- Magnification Unity ± 5%
- 1 stop 1.2
- Fit Adjustments**
- Line of Sight 8°
- Eye Relief 16mm travel
- Vertical Adjustment 16mm travel
- Interpupillary Distance 52mm to 72mm
- Image Tube 18mm prox Focus MCP
- Distortion 1% max.
- Diopter Range +2 to -6 diopter
- Weight of Binocular 463 grams (Total system weight varies with helmet interface)
- Power Source 2.9 VDC (Battery or aircraft power)
- Environmental**
- Altitude Sea level to 15,000 ft.
- Shock 48-in. drop in ship, case
- Vibration 2G (5-42-500HZ)
- Humidity 95% (min.)
- Operating Temp. -32° C to +52° C

Binocular Assembly



Helmet Mount



1000

FIGURE 2

EYE EFFICIENCY/ PHOTOCATHODE RESPONSE: 3rd GENERATION

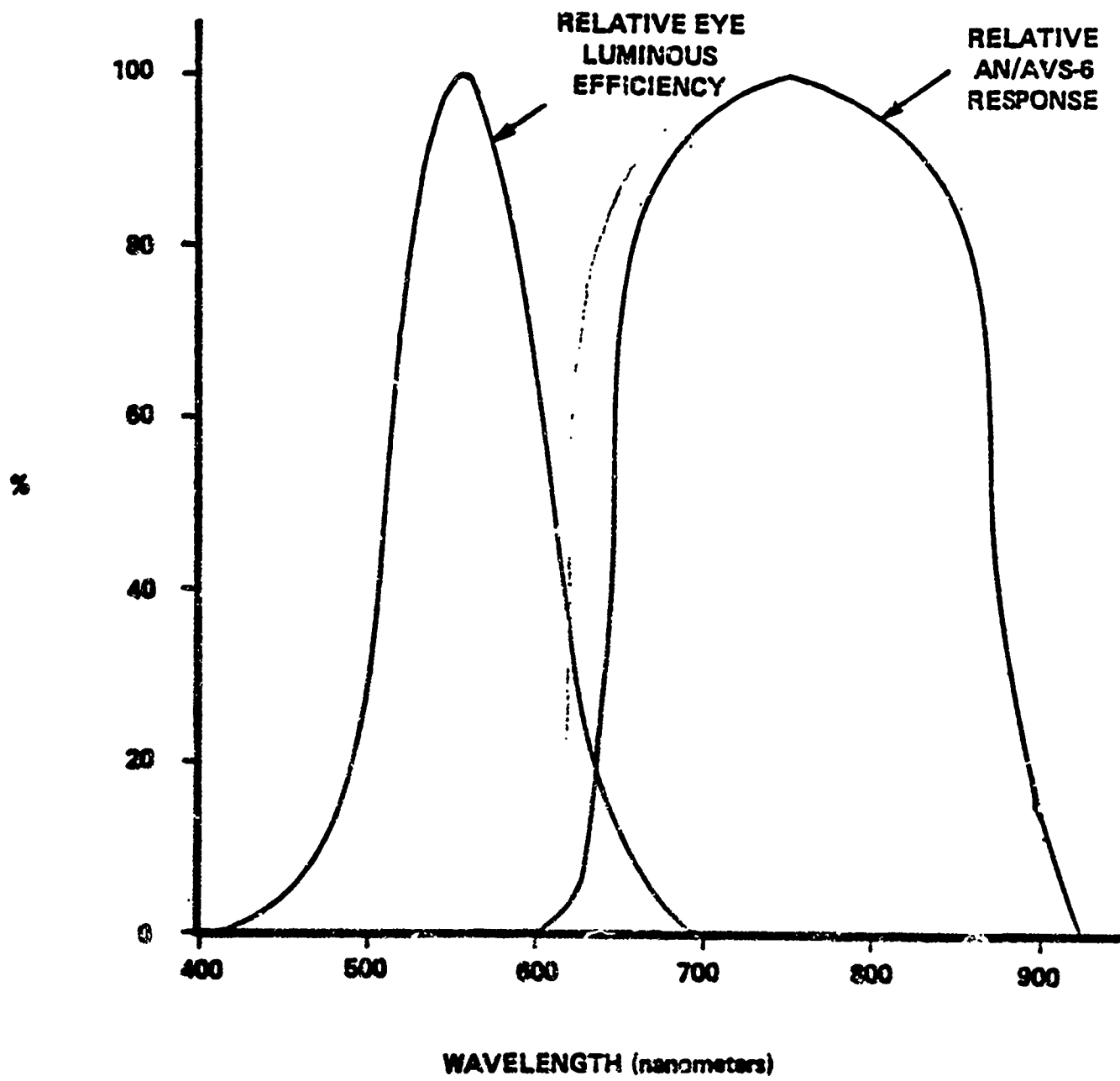


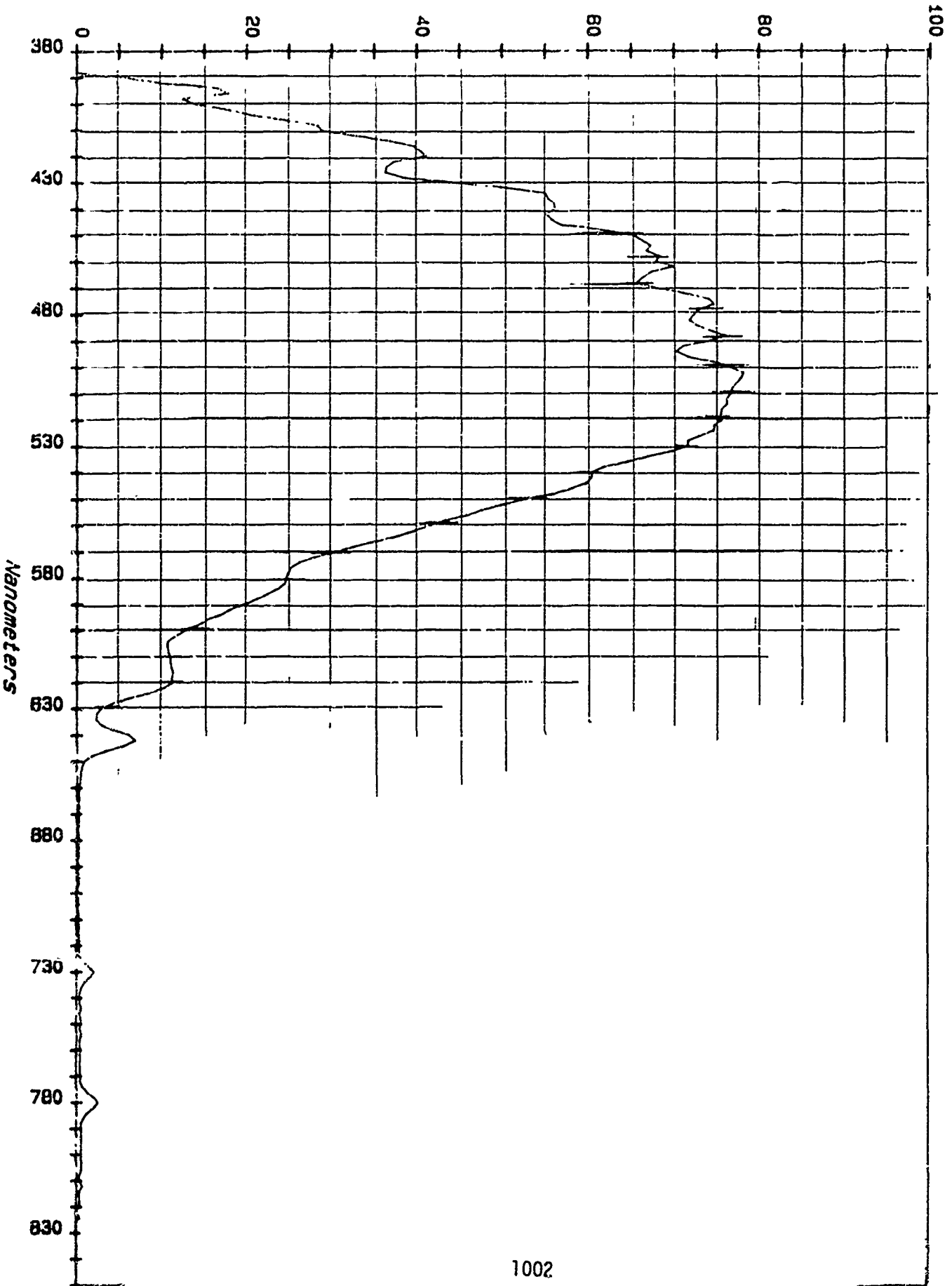
FIGURE 3

Transmittance %

Date: 8/10/82
Title: IR Resp

Name: Genfit
Max: 1.000e 00
PREC: 4.44

FIGURE 4



Green cockpit lighting, in addition to being necessary for ANVIS compatibility, is also essential for reducing the external signature of the aircraft as seen through night vision goggles, samples of which are contained in the referenced document.

In conclusion, I have attempted to highlight some of the difficulties encountered in the Tri-Service standardization process. I am sure that the situation depicted above, for the CARA altimeter, is not unique and probably represents the norm as opposed to the exception. My purpose is to surface these issues both in the present forum, as well as through appropriate program/product managers, to stimulate the kind of thinking, planning, and meaningful Tri-Service coordination that must take place to succeed. Since we are all challenged, in these economic times, to make the best use out of every dollar spent by the DOD, it behooves us all to provide the level of cooperation necessary to insure the success of programs like CARA and others presented at this conference.

References:

1. Class IV Modification Program Management Directive No. L .-1036(1), 19 Feb 81.
2. Military Specification, 791RCR-CARA-523A, October 1981; ARINC Research Corporation Publication 2751-01-1-2367.
3. Contract F09603-82-C-0231 issued by Warner-Robins ALC/PMWCB, Robins AFB, GA 31098.
4. Joint Services Operational Requirement for Advanced Vertical Lift Aircraft (JVX JSOR), 24 Jun 82.
5. MIL-STD-1553 Multiplex Applications Handbook, 1 May 1980, AFSC. Draft Addendum, Chapter 11, August 1982.
6. IEEE Spectrum october 1982, "Electronics in the Nuclear Battlefield".
7. U.S. Army Night Vision & Electro-Optics Laboratory, Technical Report, Night Vision Goggle Compatible Lighting by Major Mickey Potter, Feb 1981.

MATE STANDARDIZATION

Capt Richard E. Farmer
Program Manager, Program Development
MATE Program Office ASD/AEGB WPAFB, Ohio 45433 X53596

Capt Farmer was assigned to the MATE Program Office in June of 1982 as a Program Manager in the Program Development Branch, following a five year tour of duty as an ICBM launch officer with the 91st Strategic Missile Wing, Strategic Air Command, Minot, North Dakota. He received a Bachelor of Arts Degree in Political Science from Miami University, Oxford, Ohio. Capt Farmer served for five years as a marketing representative with Champion International and is a veteran of eight years of enlisted service in the United States Army.

ABSTRACT

The MATE (Modular Automatic Test Equipment) program was developed to combat the proliferation of unique, expensive ATE within the Air Force. MATE incorporates a standard management approach and a standard architecture designed to implement a cradle-to-grave approach to the acquisition of ATE and to significantly reduce the life cycle cost of weapons systems support. These standards are detailed in the MATE Guides. The MATE Guides assist both the Air Force and Industry in implementing the MATE concept, and provide the necessary tools and guidance required for successful acquisition of ATE. The guides also provide the necessary specifications for industry to build "MATE-qualifiable equipment". The MATE architecture provides standards for all key interfaces of an ATE system.

The MATE approach to the acquisition and management of ATE has been jointly endorsed by the commanders of Air Force Systems Command and Air Force Logistics Command as "The way of doing business in the future."

MATE STANDARDIZATION TEXT

1.0 Background - The electronics in modern weapon systems have reached a level of complexity where we are unable to deploy major avionics systems without fielding unique, complex, and expensive automatic test equipment (ATE) to provide effective support. In fact, the Air Force has become so dependent on ATE that about 75% of its support equipment budget is devoted to developing and acquiring automatic test systems. The operation and support costs alone for these systems are as much as three times the already high acquisition cost. The combination of acquisition and operational support costs for Air Force ATE is over one billion dollars per year. There are

several factors contributing to this large drain of financial resources. They include today's rapidly advancing technology, the skyrocketing costs of the equipment itself, and the ever-increasing life expectancy of our major weapon systems, which have in turn forced an increase in the life requirements of our test equipment.

Looking at this drain from another perspective, more than 70% of the life cycle cost (LCC) of any system is "locked in" quite early in the conceptual phase when the system concept is defined. This locking in of LCC becomes an even more serious culprit when considering ATE. Traditionally, these Air Force systems have been developed and acquired as part of the prime mission equipment acquisition process and were designed to support only one weapon system. Further, they were frequently the last item of consideration in the weapon system acquisition process and were given only limited attention. Little or no thought was given to developing ATE systems with applicability to more than one weapon system, much less to ATE that was designed to keep pace with advancing technology. Consequently, the Air Force has been forced to spend higher and higher percentages of its acquisition budget to totally replace the older automatic test systems and a larger percentage of its support budget to supply and maintain the proliferation of ATE which has resulted. This does not even address the logistics nightmares that have been created for field commanders who must deploy with exorbitant spare requirements just to support the ATE.

To combat this problem, the Air Force established the MATE Program in 1976 to develop a systematic, disciplined approach to the definition, acquisition, and support of automatic testing capabilities within the Air Force and to investigate the feasibility of developing interface standards for ATE architecture. The program represents a cradle-to-grave approach to the acquisition of ATE and will significantly reduce the LCC of weapon systems support.

The MATE Full Scale Development (FSD) contract was awarded to Sperry Systems Management Division, Great Neck, NY in July 1981. This contract requires (1) Full Scale Development of the MATE System; (2) first MATE System application to the Intermediate automatic test system for the A-10 Inertial Navigation System; and (3) Technical Assistance to the Air Force for MATE System application to the separately competed Depot Automatic Test System for Avionics (DATSA) program. This separate contract was awarded to Emerson Electronics and Space Division, Florissant, Missouri, in May 1982.

2.0 MATE System. The MATE System encompasses all hardware and software required to provide the Air Force with the procedures, technical tools, and facilities required to acquire and technically support Automatic Test Systems. It is in effect a management system supported by personnel and technical tools and consists of the following major elements:

a. MATE System Guides

- (1) Introduction to the MATE Guides
- (2) MATE Acquisition Guide
- (3) MATE Development Guide
- (4) Testability Design Guide

- (5) Production/Operational Guide
- (6) Test Program Set Acquisition Guide

- b. Control & Support Software
- c. Automated Acquisition Tools
- d. MATE Support Center

Each one of these will be discussed in turn.

2.1 MATE System Guides.

a. Introduction to the MATE Guides. This guide provides the top level entry point into the MATE Guides. It has been prepared to introduce the MATE System, as embodied in the MATE Guides, and to enhance the useability of the MATE Guides. It also contains an overall index for all the guides.

b. MATE Acquisition Guide. The MATE Acquisition Guide provides the process, procedures and tools necessary for the acquisition and management of Automatic Test Equipment (ATE). This guide treats the ATE acquisition process as an integral part of the Weapon System acquisition process and guides the user from the formulation of the Statement of Need (SON) through Full Scale Development. The guide is the primary tool for ATE acquisition, directing the user at the appropriate point in the process to the other MATE guides.

The guide contains the flow diagrams which identify functions and iterative procedures that are performed during the ATE acquisition process. These iterations are performed as additional or updated data becomes available, with each iteration yielding a more refined definition of the avionic support equipment requirements. This process provides for the establishment and selection of cost effective support approaches by using Life Cycle Cost Models to perform required trade-offs. This process is also employed for the generation of maintenance plans and the development of intermediate and depot test station configurations.

c. MATE Development Guide. The MATE Development Guide is the primary MATE tool for implementation of the MATE standard architecture. It provides all of the procedures, standards, specifications and technical tools required to develop ATE systems, stations, individual test modules, and test programs. This guide provides guidance in the three major disciplines of ATE development (1) Hardware (2) Software and (3) Human Factors Engineering.

Procedures are provided to cover all phases of ATE development starting with the preparation of the ATE System Specification through ATE and TPS design, fabrication and DT&E testing, and ending with ATE delivery, installation and test at the IOT&E site. Tasks are identified and detailed, covering functional areas such as: test station design; hardware and software development and product specification; technical reviews and audits; DT&E and OT&E test planning; Programmer and Test Station Operator/Maintainer Manuals; test station self-test and

calibration programs; Test Program Set design, development, Verification and Validation and Acceptance; Test Station Fabrication Integration and Test; Personnel Training and Training Aid Design.

In order to implement the MATE architectural concepts, the key interfaces of an ATE system have been standardized. Standards are provided for the (1) Controller Translator Card which is the interface between the test station computer and the communication bus, (2) the bus itself, which is the IEEE-488 industry standard, (3) the Control Interface Intermediate Language (CIIL) which specifies the command and data language formats required on the 488 bus for all communications between the computer and test modules, (4) the Test Module Adapter (TMA) used when necessary, to interface off-the-shelf test modules with the standardized 488/CIIL bus, (5) Higher Order Language standard (IEEE-STD-716 ATLAS) for writing all test programs, (6) High Order Language standard (MIL-STD-1589B, JOVIAL) for writing all control and support software, and (7) MIL-STD-1750A Instruction Set Architecture for the central computer.

Standards are also provided for the station input/output interfaces which specify the mechanical and electrical requirements, the UUT Adapter with the Test Station, and finally, the format and data to be displayed to operators and programmers.

Design guides and manuals are provided in the key ATE design areas. These include (1) Control, Support and Test Software, (2) Human Factors Engineering, and (3) ATE hardware development areas such as architecture, switching, grounding, packaging, thermal design, interfaces, self-test and calibration, controls and displays, timing and synchronization and reconfigurability.

d. Testability Design Guide. Testability is the design characteristic of a Unit Under Test (UUT) that allows determination of the UUT's operational status and, in the case of malfunction, isolation to the faulty lower level replaceable item. The Testability Design Guide provides the procedures and technical tools required by avionic and ATE developers to produce equipment that is testable at each of the required maintenance levels. The guide defines procedures for determining the relative cost and benefits associated with avionics Built In Test (BIT); identification of preferred test techniques and generic test equipment as well as the methods to translate them into MATE preferred test (software) templates; and the design concepts, principles and techniques required for the implementation of testability features into electronic equipment. In addition, the guide contains a description of the requirements to manage an avionics equipment Design for Testability (DFT) Program.

e. Production/Operational Guide. The Production/Operational Guide defines the process and procedures to be utilized for the management of ATE during the production and operation phases of the Weapon System Life Cycle. Activities from ATS IOT&E and production authorization until equipment is returned to inventory for reuse are described. As a companion guide to the MATE Acquisition Guide it completes the coverage of the ATE Life Cycle. Similarly, it directs the user at appropriate times to the Development, Test Program Set Acquisition and Testability Guides for specific required procedures and information.

Procedural information is provided for program office guidance in development implementation of turnover and transition activities, planning and implementation of Program Management Responsibility Transfer (PMRT), planning and implementing the procurement of production quantities of test sets and MATE equipment, use of the MATE System, development of technical orders, test set acceptance testing, and development and implementation of QA plans.

f. Test Program Set (TPS) Acquisition Guide. The MATE TPS Acquisition Guide provides the TPS Acquisition manager with the tools and guidance required for the successful acquisition of MATE TPSs that are performance-effective, cost-effective and supportable, within the appropriate program schedule. The Guide provides a framework with all of the management and technical methods, tools, procedures, documentation and guidance required by the Air Force to effect a successful MATE TPS Acquisition. The framework is based on a sound definition of the TPS Life Cycle, which is cross-referenced to each phase of the Weapon System Life Cycle. Guidance is provided for the following acquisition scenarios:

- o TPSs and the Weapon System are contracted for at the same time.
- o TPSs are contracted for after the weapon system contract award.
- o TPSs are contracted for existing UUTs.
- o TPSs are contracted for as a result of a major redesign in the UUTs.
- o TPSs are contracted for as a result of a requirement to upgrade the performance of the TPSs.

The guide provides information, guidance, methods and tools for each of the various TPS acquisition situations cited.

2.2 MATE Control & Support Software

a. MATE Operating System (MOS). The MOS provides the environment for the development and execution of test programs and supports a self sustaining maintenance capability. It performs the following functions: Task scheduling, Input/Output scheduling, Interface to Peripherals, Process Interrupts, File System Management, Main Memory Management and Service System calls.

b. MATE On-Line Editor (MOLE). The MOLE provides a general purpose text creation/alteration capability. It performs the following functions: Accepts Operator Commands, Manipulation of test files, provide screen oriented operations for CRT type terminals.

c. MATE ATLAS Compiler (MAC). The MAC provides the ATLAS compilation capability. It provides the following functions: Accept MATE ATLAS subset, verify Syntax and Semantics, Analyze Signal Situations, perform automatic resource allocation, generate MATE Intermediate code, and provide formatted output listings.

d. MATE Test Executive (MTE). The MTE provides the primary interface between the test operator, the executing ATLAS program, the test station test programmer, and provides a convenient means of entering and correcting an

ATLAS program. It performs the following functions: Acceptance of operator commands, Interpretation of ATLAS intermediate code, Interface to instruments via IEEE-STD-488 bus, and Interface to peripheral devices via the MOS.

e. Test Module Adapter (TMA). The TMA consists of all device drivers and logic to control test instruments. It performs the following functions: Translation of MATE Intermediate language commands to test instrument control commands, acceptance and processing of raw instrument data, and control of test instrument confidence tests.

2.3 Automated Acquisition Tools. These MATE tools encompass the software models, programs, and reference data banks provided as aids for ATE and TPS acquisition, management, and configuration control. The MATE Support Center and ASD Computer Center facilities are employed, as required, for the operational use of these tools.

a. ATE Acquisition Tools

(1) Life Cycle Cost Model - The Life Cycle Cost Model selected for the MATE program provides estimates of development, production, operation and support costs for avionics and ATE. The model provides the capability of developing support system costs, through all acquisition phases (conceptual through operational). The estimates are developed from resource requirements and lead to management decisions regarding; level of repair, test station loading, shop configuration at both the intermediate and depot level, BIT versus ATE selection, Interim Contractor Support, and schedule and employment changes.

(2) MATE Data System. The MATE Data System provides historical cost and maintenance data on existing avionics and their related support equipments. The Data System also provides configuration status accounting data on the avionics, support equipment, MATE hardware modules, test program sets, and MATE system control and support software. A unique feature of the data system is its ability to permit selection of test modules or the test systems containing these test modules by matching their characteristics against inputted test requirements for units to be tested.

(3) Software Module Data System. This system provides an automated library for the storage and retrieval of MATE source programs. Programs are normally stored and retrieved using Computer Program Identification Numbers (CPINs). Provisions have been made, however, to store and recall programs using contractor assigned identification numbers until such time as CPINs are assigned.

b. TPS Acquisition Tools

(1) TPS Cost Predictor Model. This program provides the capability to store and retrieve historical cost data from a data base of credible TPS development programs. The cost prediction model accepts UUT characteristic data, labor rates, support and other fixed cost parameters as input to a cost algorithm and computes a predicted TPS development cost for the UUT (LRUs or SRUs). It can provide budgetary costs from limited UUT data, available early in the weapon system development cycle, which is useful

for weapon system support planning purposes. As the UUT design becomes more definitive, more accurate TPS development costs can be predicted by the model.

(2) TPS Acquisition Quality Assurance Computer Programs. These programs provide the capability to determine and monitor the quality of the TPS through its various stages of development, implementation, and acceptance. These computer programs shall insure that the design of the TPS meets the requirements of a prescribed fault detection/fault isolation criteria for any given set of avionics be it at the LRU or SRU level.

The programs will be capable of supporting the quality assurance function in the general areas of inspection, analysis, audit, and demonstration for the validation and verification of any given TPS.

2.4 MATE Support Center.¹ The MATE Support Center is composed of two primary subsystems: MATE Software Development and Support Subsystem, and the MATE Module Qualification Subsystem.

The MATE Software Development and Support Subsystem provides the computer, peripheral and software capabilities required to develop and maintain all retargetable MATE software. The retargetable system software packages are part of this subsystem and include the ATE control and support software (MATE Operating System, MATE Test Executive, MATE ATLAS Compiler, MATE On Line Editor).

The MATE Module Qualification Subsystem provides the computer, peripherals, software, and test module capabilities required to qualify MATE hardware and software modules for use on MATE application programs. The subsystem configuration is flexible and provides for ease of change or growth to satisfy the testing needs for initial module qualification, for technical support of all fielded MATE test station modules, and for qualifying new test modules to satisfy future test requirements.

3.0 MATE Benefits.

MATE Standardization ensures Air Force benefits through the following:

a. A continuity of lessons learned. Because of standard software and architecture, a problem corrected on one system will also be fixed for others. The advantage here is that this will reduce the number of problems new test systems will face.

b. A re-use of system assets and spares. MATE gives the capability to share common assets and possible reductions in the number of hardware inventory items by a factor of 10.

c. Common personnel training and support. The Air Force now trains people in 42 software programming languages. MATE will use only two. Standardized human interfaces mean shorter training time and better cross station mobility.

1 - Currently a Sperry Capital Facility at Great Neck, NY.

d. Software transportability. 95% of control and support software will be common to all MATE stations.

e. Operational flexibility. Modifications to the system to accommodate technical advances can be performed with relative ease.

f. Software maintainability. Standard control and support software will be under organic control of the USAF.

g. Competition in industry. Through open competition of MATE module suppliers, the marketplace will be used to the best advantage for the government.

h. Reduced life cycle costs. The bottom line is total savings to the government over the life of ATE systems fielded.

3.1 MATE Implementation - In recognition of the many problems which face the Air Force in the acquisition and management of Automatic Test Equipment, and of the disciplined and rational approach MATE uses to address these problems, the commanders of Air Force Logistics Command and Air Force Systems Command issued a joint endorsement of the MATE Program on 21 July 1982. The AFLC and AFSC commanders commitment to the MATE Standardization is best illustrated by this quote from their endorsement letter, "We expect each major organization in our commands to support a thorough and timely implementation of MATE on Air Force programs."

3.2 Summary - MATE is a standardized approach to modular ATS. It is a well organized, flexible system keyed to standard interfaces, as well as a "cradle-to-grave" management system. This rational approach to standardization provides flexibility for technology improvement while eliminating costly government ATE acquisition inefficiencies. Standardization is not just desirable; it is essential.



DIGITAL AUDIO DISTRIBUTION SYSTEM

Capt J. Darian Ross, Jr.
ASD/AEAC
Wright-Patterson AFB OH

Biographical Sketch:

Capt Ross is the US Air Force Program Manager for the tri-service Digital Audio Distribution System (DADS) Program.

Capt Ross graduated from the US Air Force Academy in 1972 and holds a BS degree in Engineering Management.

From 1974 to 1979, Capt Ross was stationed at Charleston AFB SC and performed as a C-141 Navigator and Instructor Navigator in the Military Airlift Command. In 1979 he was selected for the Education With Industry (EWI) Program and was assigned to the Re-Entry Systems Division (RSD) of General Electric at Philadelphia PA from 1979 to 1980.

Following his tour with GE/RSD, Capt Ross was assigned to the Aeronautical Systems Division (ASD). Prior to his selection as DADS Program Manager, Capt Ross managed the APX-101 Transponder Program and functioned as the Test Manager on the Standard Central Air Data Computer (SCADC) Program.

Capt Ross will attend the Program Managers Course, Defense Systems Management College (DSMC), at Fort Belvoir VA in January of next year.

Abstract:

In January 1982, the Standard Aircraft Intercommunication System (ICS) Feasibility Study findings were released. One of the primary objectives of this study was to determine existing and future aircraft performance requirements for intercom systems and to ascertain the degree to which present systems satisfy these requirements.

The ICS Study revealed that the military inventory presently includes over seventeen (17) different systems consisting of over (40) Line Replaceable Units (LRUs). Problems were identified with the majority of these systems. The primary problem results from inadequate channel isolation which adversely impacts the intercom system's performance. In addition, concern was expressed as to the survivability of existing intercom systems in a nuclear environment. There are also problems of supportability, Electromagnetic Interference (EMI), and intelligibility in high noise environments.

These findings combined with Deputy Secretary of Defense Frank Carlucci's initiative #21, which calls for the development and use of standard operational and support systems gave rise to the Tri-Service Digital Audio Distribution System Working Group (TDADSWG). The TDADSWG, composed of Air Force, Army and Navy representation, has been chartered to pursue the development of a standard Digital Audio Distribution System (DADS) to rectify these problems. Efforts are presently under way to provide DADS by the FY88 timeframe.

STANDARDIZED SOFTWARE DEVELOPMENT

SESSION CHAIRMAN: David J. Krite
ASD/ENAMR

MODERATOR: Colonel Harold C. Falk
AFWAL/AA
Special Assistant for Simulators

AD-P003 588

SYSTEM PLANNING TOOL TO MEASURE
COST AVOIDANCE RESULTING FROM
INDEPENDENT ASSESSMENT TECHNIQUES
APPLIED TO TEST PROGRAM
SOFTWARE DEVELOPMENT

P. D. KIDD

TECHNOLOGY DEVELOPMENT OF CALIFORNIA
624 SIX FLAGS DRIVE
ARLINGTON, TEXAS 76011
(817) 461-1242

ABSTRACT

Test Program (TP) software errors result in avoidable costs. This paper describes a mathematical concept, based on principles of probability theory, to determine system cost savings resulting from the application of Independent Assessment (IA) techniques to TP software development. Independent Assessment embodies both an engineering evaluation of hardware as well as the evaluation of software. The development of a cost savings model from this concept would result in a MATE tool which would allow system planners to better evaluate and minimize system life cycle costs. Such a model would allow planners to quantify the cost impact of TP software errors on Unit Under Test (UUT) life cycle cost and to define and study specific means to minimize those costs.

BACKGROUND

Most Test Program (TP) software errors result in costs incurred by the user or customers that could have been avoided (i.e., avoidable costs). TP software, in conjunction with automatic test equipment (ATE), is used to verify function, detect and diagnose electronic equipment during scheduled and unscheduled maintenance. The test item is referred to as the Unit Under Test (UUT). Maintenance decisions are based on test results. If the TP indicates that the UUT is "bad," the UUT is either sent to the next level of maintenance or repaired at that level of maintenance. For example, test results at the intermediate level (I-level) where the UUT is tested may cause the UUT or a replaceable part of the UUT to be sent to the depot (D-level) for maintenance. For any number of reasons, the UUT may actually be "good," even though the test indicates "bad."

Such incorrect maintenance decisions cause costs to be incurred which are unnecessary and may be avoidable. If the incorrect test result occurred due to an error in the TP software, then the false test result and associated

extra costs were avoidable. Avoidable costs include not only logistics support costs arising from incorrect maintenance decisions but also the cost of possible hardware damage or destruction arising from faulty TP software.

These costs can be very significant and represent a portion of the total life cycle cost or ownership cost of the UUT. For any specific TP software fault, the impact on UUT ownership cost will be a function of several variables, including how quickly the software error is identified and corrected, the effect of the error on maintenance diagnosis, and others. Some software errors may have no significant maintenance impact. Others may have a cost impact but the effect is unnoticed. The error may never be discovered during the life of the UUT.

Perhaps a more comprehensive quality assurance program applied during TP software development could prevent many of the errors and reduce the avoidable UUT logistics support costs. Historical data (Ref. 1) indicate that over 40% of all TP software errors are not discovered and corrected during the course of Test Program Set (TPS) development processes (see Figure 1). These undiscovered errors are deployed with the TPS and corresponding UUTs, where they contribute to increased UUT life cycle costs. If only 50-60% of software errors are found during the development process, there remains considerable room for improvement.

An improved and more thorough software quality assurance (SQA) program is one way to significantly reduce the percentage of undiscovered software errors. Implementation of an improved SQA program requires modest additional costs during TPS development. When viewed from an overall software/hardware systems standpoint and considering the ownership cost for that system, the additional SQA cost will be a cost-effective investment in most programs. Also, many qualitative benefits can be derived for maintenance personnel through the improved TP quality. This paper is limited to systems cost benefits.

Systems planners need the means to compute cost benefits of software quality assurance efforts. Development of this cost savings model would allow the cost effectiveness of a SQA effort to be quantified. A model to relate TP software errors and consequences/effects on UUT logistics costs and other systems costs must be developed in order to attain this capability. Furthermore, such a model must relate specific SQA activities during TPS development to reduction of software errors that otherwise escape detection during TPS development. Clearly, such a model would permit system planners to better evaluate and minimize system life cycle costs. Planners would be able to quantify the cost impact of TP software errors on total system life cycle cost and define and study specific means to minimize those costs.

Technology Development of California (TDC) has derived a mathematical expression, based on fundamental principles of probability theory, which expresses the cost avoidance resulting from a third party Independent Assessment (IA) of a TPS during its development. An IA comprises software evaluation and quality assurance as well as an engineering evaluation of the test program effect on the hardware. TDC intends to develop from this concept a specific model to evaluate the cost benefits of TPS independent assessment

TPS DEVELOPMENT PHASE

	<u>Source Data</u>	<u>TPS Design</u>	<u>Fabrication</u>	<u>Integration</u>	<u>TPS Acceptance</u>
Percentage of all TPS software errors originated in this phase	5%	51%	4%	36%	4%
Percentage of errors originated in this phase which are not corrected during TPS development	41%	47%	66%	37%	21%
Percentage of all TPS software errors which are originated in this phase and not corrected during TPS development	2%	24%	2.6%	13.3%	1%

Figure 1 SOFTWARE ERRORS NOT DISCOVERED DURING TPS DEVELOPMENT (Breakdown by Phase Where Originated)

when used in lieu of standard SQA techniques. The remainder of this paper is devoted to the steps involved in the development of this model concept.

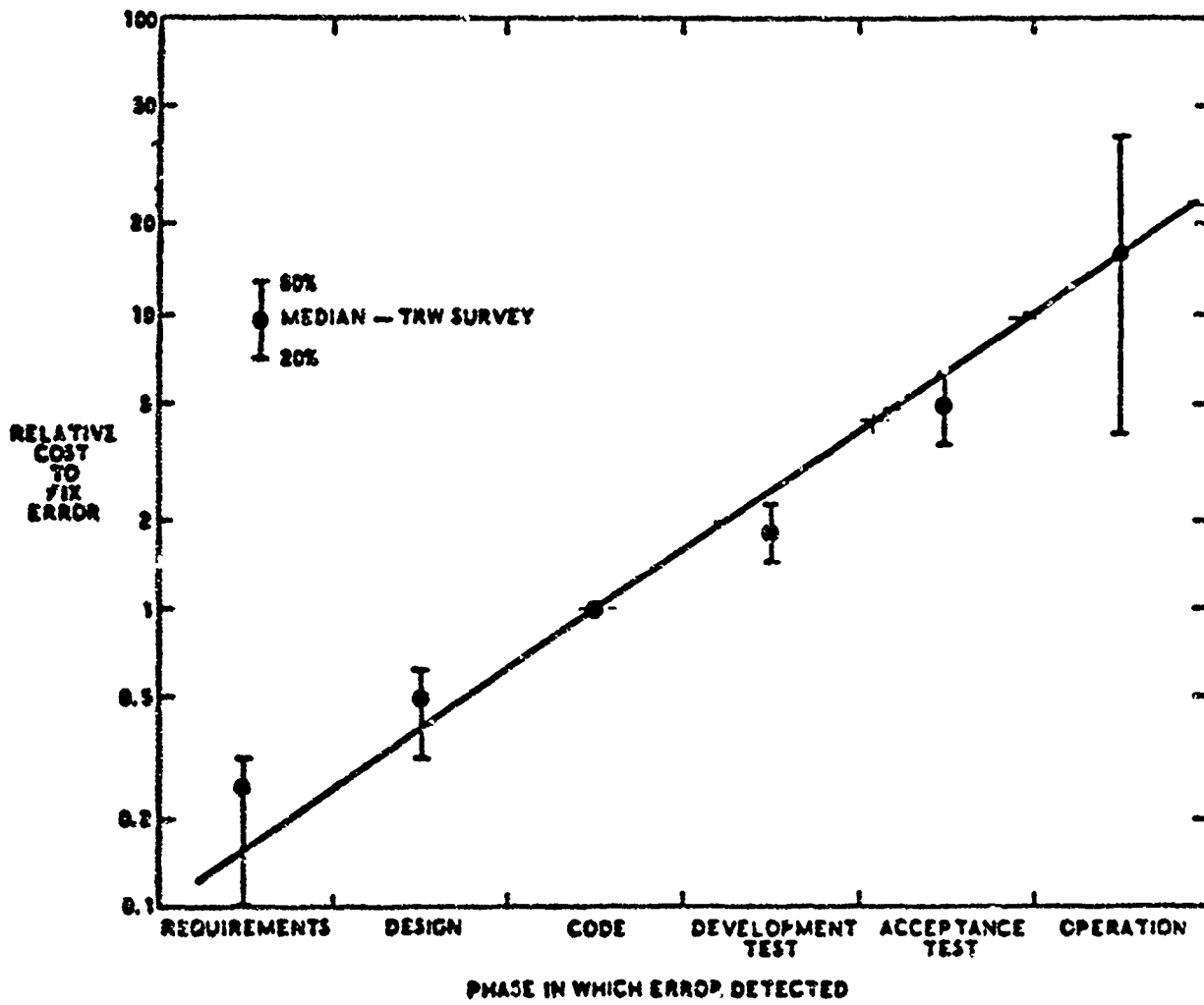
MODEL CONCEPT

The expected software-related costs avoided as a result of an IA must be determined in order to quantify the cost effectiveness of such efforts. The cost categories most directly influenced by IA activity are: (1) error correction costs, and (2) logistics support costs.

Once a software error is detected and identified, there are several possible costs that will be incurred to correct the error. These costs include: software modification/correction, preparation of Engineering Change Proposals (ECPs), and other administrative costs. These costs have been shown to be a function of time (Ref. 2). The magnitude of the cost depends on the point in the TPS development schedule (Figure 2) that correction of the error occurs. The cost to correct the error increases with time. Thus, the benefit to be derived will be a function of the difference between that time when the error is detected during an IA and the expected later time when the error would have been detected if no IA were done. A cost to correct is associated with each of these times and the cost difference between the two represents the cost avoided. The larger the difference, the larger the cost benefit. Time of error discovery by means other than IA is a stochastic problem. Expected time and cost to correct must be expressed in terms of probability distributions describing the distribution of error discovery times.

The second area of cost, UUT logistics support costs, are those costs which are incurred during the operations phase of a UUT and result directly or indirectly from software errors not detected during the TPS development phase. Clearly, TPS software errors at any maintenance level (O, I, D) can result in incorrect maintenance decisions that lead to unnecessary costs. For example, a given software error may cause a good UUT to test "bad" or a bad unit may test "good." As a result, an incorrect maintenance decision ensues and additional and unnecessary logistics support costs are incurred which could have been avoided had the software error been found and corrected at an earlier time. Avoidable logistics support costs include not only costs incurred as a result of incorrect maintenance decisions resulting from faulty TPS software but also include costs due to possible hardware damage or elimination resulting from faulty TPS software.

Cost relationships and probability distributions will be functions of several variables. Among these variables are time, τ , software error type/category, η , and UUT complexity level, κ . Clearly, both cost and probability expressions are time-dependent. The accumulative probability of discovery of a given error type will increase with time. Probabilities and costs also depend on the software error type. Probability of discovery of a given error type depends on the nature of the error. Some error types may be easily discovered while others may be difficult due to their nature and the effects of the errors on a UUT. A given error type may manifest different effects on UUTs of different complexities. Thus, the nature of the UUT itself and its complexity must be taken into account.



Software validation: the price of procrastination
 (Taken from paper by Barry W. Boehm, TRW Systems, 1977)

Figure 2 RELATIVE COST TO CORRECT SOFTWARE ERROR VS. TIME

To find the value of an independent assessment in reducing avoidable costs, one must consider two scenarios. One scenario represents the case where no IA is ever done. The other scenario represents the case where an IA is done. A quantitative and systematic comparison of these two scenarios will provide help in deriving an expression for the benefit of an IA.

Let us consider one UUT of complexity level κ and one software error belonging to the η^{th} error type or category. For any given software error originated at some time τ_0 , there is a probability that the error will have been found by some later time, τ , through means other than independent assessment. Let $p_1(\tau, \eta, \kappa)$ represent the probability for discovering the η^{th} software error type associated with hardware of complexity level κ during a small time interval between τ and $\tau + \Delta\tau$. Then the expected time, τ_E , to discovery of the η^{th} error type for a UUT of complexity level κ is given by

$$(1) \quad \tau_0(\eta) = \int_{\tau_0}^{\infty} \tau p_1(\tau, \eta, \kappa) d\tau,$$

where $\int_{\tau_0}^{\infty} p_1(\tau, \eta, \kappa) d\tau = 1$. The term $\tau_0(\eta)$ represents the fact that

different error types or categories are generally originated at different points in the TPS development process (Ref. 1). In any case, the term may represent the mean of a distribution of origination times for a given software error type.

If we represent time-dependent avoidable logistics support costs incurred as a result of this error type by $C^*(\tau, \eta, \kappa)$, then the avoidable logistics supports costs expected to be accumulated before discovery of the η^{th}

software error are given by $\int_{\tau_1}^{\tau_E} C^*(\tau, \eta, \kappa) d\tau$, where τ_E is the expected

time to discovery of the error, given by expression (1), and τ_1 represents the beginning of the TPS-UUT operational stage (see figure 2). Let τ_L represent the useful economic life of the UUT. It is possible that $\tau_E > \tau_L$ for some errors. Thus, a dummy variable may be substituted as the upper limit of the integral, and we may write

$$(2) \quad C_L = \int_{\tau_1}^{\tau_U} C^*(\tau, \eta, \kappa) d\tau,$$

where C_L represents accumulated avoidable logistics costs incurred as a result of the η^{th} software error type and associated with a UUT of complexity level κ . If $\tau_E > \tau_L$, then $\tau_U = \tau_L$; otherwise, $\tau_U = \tau_E$.

We must now explore the scenario in which specific software quality assurance (SQA) techniques are instituted during development of the TPS. And we must relate the use of those techniques to specific reductions in the numbers and types of software errors, and subsequent reductions in cost. In this paper we concentrate on the implementation of independent assessment in lieu of standard SQA techniques, as TDC applies IA to Test Programs.

Let $p_2(\tau, \eta, \kappa)$ represent the conditional probability that the η^{th} software error type associated with hardware of complexity level κ will be discovered as a result of independent assessment activities conducted during a small interval of time between τ and $\tau + \Delta\tau$, given that the η^{th} error type exists at time τ . The probability that the η^{th} error type exists at time τ is given by $1 - \int_{\tau_0}^{\tau} p_1(\tau, \eta, \kappa) d\tau$, where τ_0 is a dummy variable. We can represent this

probability or survival function as $P'_1(\tau, \eta, \kappa)$. Employing Bayes' theory on conditional probabilities, it can be shown that the probability that the error both exists at time τ and will be discovered as a result of independent assessment activity conducted during the interval between τ and $\tau + \Delta\tau$ is given by

$$(3) \quad p_2(\tau, \eta, \kappa) P'_1(\tau, \eta, \kappa).$$

This expression represents the probability of success of an independent assessment in finding an existing error. It can be shown that the probability of failure in finding the error is given by

$$(4) \quad [1 - p_2(\tau, \eta, \kappa)] P'_1(\tau, \eta, \kappa).$$

The distributions describing these probabilities must be determined in the process of developing a model. Only the conceptual approach is developed in this paper.

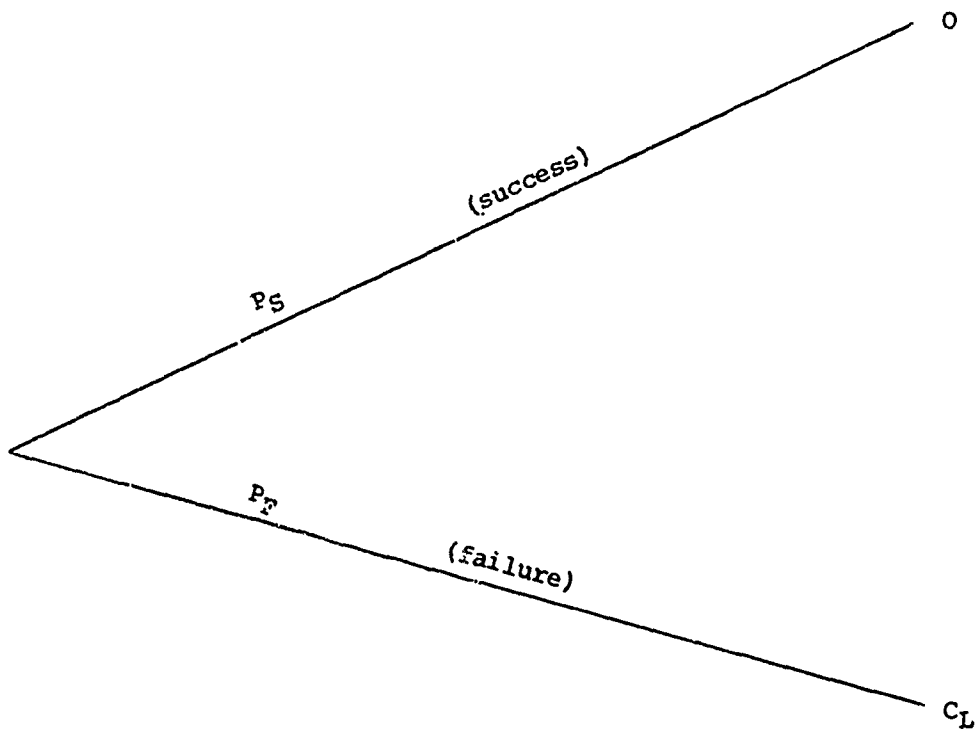
Let the probability of success (expression 3) be represented by P_S and the probability of failure (expression 4) be represented by P_F . The result of an IA is either success or failure. If the IA is restricted to TPS development stages only, then no avoidable logistics support costs will be incurred for the η^{th} software error type if the IA is successful. Costs given by expression (2) will be incurred if the IA is not successful (see Figure 3). Thus, the expected avoidable logistics support costs incurred for the η^{th} software error type when an IA is conducted at some time, τ_a , during TPS development is given by $(P_S)(0) + (P_F)(C_L)$, where P_S and P_F are evaluated at τ_a . This expression may be written as

$$(5) \quad [1 - p_2(\tau_a, \eta, \kappa)] P'_1(\tau_a, \eta, \kappa) \int_{\tau_1}^{\tau_a} C''(\tau, \eta, \kappa) d\tau.$$

τ_a represents some finite interval of time which is very small compared to the complete time interval for TPS development. Also, τ_a can vary in value according to the error type. The summation of τ_a over all error types would substantially represent the time spent on the SQA portion of an independent assessment for one UUT.

The development to this point has been concerned with logistics support costs. We shall develop similar expressions for the cost to correct an error. The cost to correct a software error has been shown to increase with time in the TPS development process (Reference 2). Let the cost to correct a software error (includes programmer time, administrative costs, ECP-related

<u>EVENT</u>	<u>RESULT</u>	<u>EFFECT</u>
(IA)	(success/failure)	(avoidable costs incurred)



expected avoidable logistics support costs incurred - $(P_S)(0) + (P_F)(C_L)$

Figure 3 INDEPENDENT ASSESSMENT SUCCESS/FAILURE
DIAGRAM - AVOIDABLE LOGISTICS SUPPORT COST

costs) be represented by $C'(\tau, \eta, \kappa)$. The probability of success and failure in finding a software error as a result of an independent assessment is given by expressions (3) and (4), respectively. If the IA, conducted at time τ_a during TPS development, is successful, then the cost incurred is $C'(\tau_a, \eta, \kappa)$. If the IA is not successful, the expected cost to correct is given by $C'(\tau_E, \eta, \kappa)$. This situation is represented in Figure 4. The expected cost to correct when an IA is conducted at time τ_a is given by

$$(6) \quad (P_S) [C'(\tau_a, \eta, \kappa)] + (P_F) [C'(\tau_E, \eta, \kappa)].$$

When no IA is conducted, the expected cost to correct is given by $C'(\tau_E, \eta, \kappa)$.

In order to find the benefit of the IA, we must quantitatively compare the scenario when an IA is conducted with the scenario when no IA is conducted. When no IA is conducted, the expected cost to correct, $C'(\tau_E, \eta, \kappa)$, plus the expected logistics support cost, given by expression (2), provides the total expected costs to be incurred. These costs may be written as

$$(7) \quad C'(\tau_E, \eta, \kappa) + \int_{\tau_1}^{\tau_U} C''(\tau, \eta, \kappa) d\tau.$$

If an IA is conducted at time τ_a during the TPS development process, the expected cost to correct plus the expected avoidable logistics support costs provide the total expected cost to be incurred, which may be written as

$$(8) \quad (P_S) [C'(\tau_a, \eta, \kappa)] + (P_F) [C'(\tau_E, \eta, \kappa)] + (P_F) \int_{\tau_1}^{\tau_U} C''(\tau, \eta, \kappa) d\tau,$$

where P_S and P_F are evaluated at τ_a .

The benefit of the IA is represented by the difference between expected total avoidable costs when no IA is conducted and those costs expected when an IA is conducted. If we subtract from this difference the cost of the IA itself, $K_{IA}(\eta, \kappa)$, then we have the net benefit to be derived from doing the IA. This net benefit may be written as

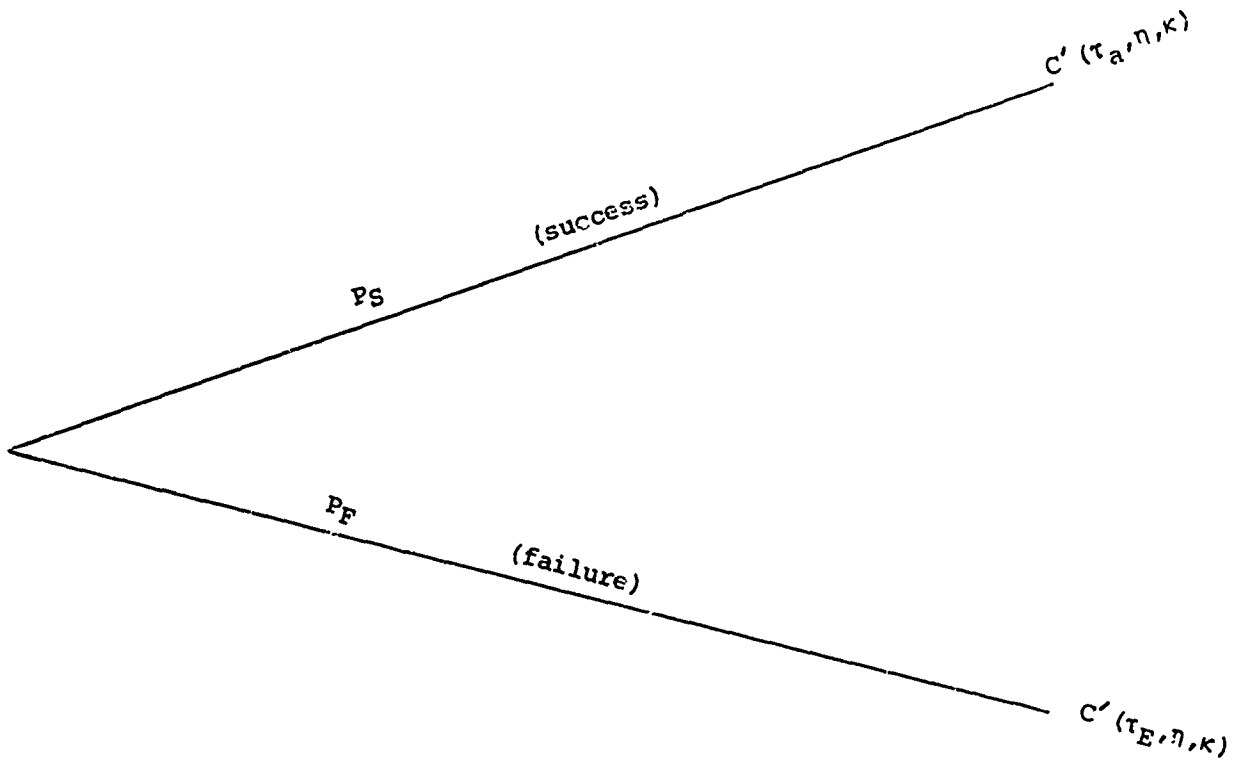
$$(9) \quad [1 - P_F(\tau_a, \eta, \kappa)] \left[\int_{\tau_1}^{\tau_U} C''(\tau, \eta, \kappa) d\tau + C'(\tau_E, \eta, \kappa) \right] \\ - [P_S(\tau_a, \eta, \kappa)] [C'(\tau_a, \eta, \kappa)] - K_{IA}(\eta, \kappa),$$

$$\text{where } P_F(\tau_a, \eta, \kappa) = \left[1 - \int_{\tau_0}^{\tau_a} p_1(\tau, \eta, \kappa) d\tau \right] [1 - p_2(\tau_a, \eta, \kappa)]$$

$$\text{and } P_S(\tau_a, \eta, \kappa) = \left[1 - \int_{\tau_0}^{\tau_a} p_1(\tau, \eta, \kappa) d\tau \right] [p_2(\tau_a, \eta, \kappa)].$$

This concept can be extended through multiple summations to apply to any number of software error types, any number of errors originated for each type,

<u>EVENT</u>	<u>RESULT</u>	<u>EFFECT</u>
(IA)	(success/failure)	(expected cost incurred)



expected cost to correct an error = $(P_S) [C'(\tau_a, \eta, \kappa)] + (P_F) [C'(\tau_E, \eta, \kappa)]$

Figure 4 INDEPENDENT ASSESSMENT SUCCESS/FAILURE DIAGRAM · COST TO CORRECT ERROR

and any number of UUTs of varying complexities. Thus, cost benefits for an entire avionics system or several systems can be found in this manner.

In the development of this concept, several explicit assumptions have been made. There are some implicit assumptions that should be pointed out. We have assumed that errors are corrected as they are found. Failure to promptly correct software errors could lead to overstatement of the net benefit of an independent assessment using this concept. We have also implicitly assumed independency of error types; i.e., we have assumed that the errors considered and/or corrected do not induce other types of errors. Thus, the composition of our set of error types to be considered when implementing this concept must conform to this criterion. If our set of error types are not all independent, we will overstate the net benefit of an independent assessment.

MODEL APPLICATIONS

The amount of unnecessary expenditures induced by TP software errors can be reduced via the application of third party independent assessment techniques during TPS development. The development of a model from the concept derived herein can provide a quantitative assessment of the benefits of IA techniques applied to TP software development. The cost impact of TP software errors on UUT logistics support can also be estimated by a model developed from this concept. The costs considered in such a model could be expanded beyond those discussed in this paper to include items such as ATE capital expenditures and support, where appropriate. Reduction in TP software-induced UUT maintenance errors may lead to some reduction of ATE capital requirements, expenditures, and support.

As an application example, the development of this model for the Modular Automatic Test Equipment (MATE) program could result in a tool that would allow system planners to better evaluate and minimize system life cycle cost. Planners would be able to quantify the cost impact of TP software errors on total hardware, software system life cycle cost and be able to define and implement specific means for minimizing those costs.

REFERENCES

- (1) TPS Development Model, MATE Guide G5V3P2.
- (2) Barry Boehm, "Software Engineering", IEEE Transaction on Computers, 1976, pp 1226-1228.

BIOGRAPHICAL SKETCH - PAUL D. KIDD

After spending several years as a member of the USAF, Mr. Kidd graduated from the University of Tennessee in 1969 with a B.S. in Physics. He obtained his M.S. in Physics in 1970 and completed studies toward a doctorate. He worked in industry for several years, studying upper atmosphere rocket exhaust plume radiative mechanisms.

Since obtaining an MBA degree in Operations Research and Management, Mr. Kidd has contributed in the areas of systems engineering, fault tree analyses, R & M predictions, life cycle cost and logistics studies, and the development of various mathematical models. Mr. Kidd has developed simulation models, life cycle cost models, availability models, and various cost tradeoff and logistics support models.

AD-P003 589

A CORPORATE APPROACH TO AN EMBEDDED SOFTWARE
DEVELOPMENT AND SUPPORT STANDARD

By

D.D. Doe	Boeing Aerospace Company
D.E. Hilt	Boeing Aerospace Company
G.B. Wigle	Boeing Aerospace Company
J.P. Bateman	Boeing Commercial Airplane Company
L.L. Tripp	Boeing Computer Services Company
W.F. Jackson	Boeing Military Airplane Company

ABSTRACT

During the last decade, embedded software has become an important part of the major products of The Boeing Company, which range from commercial jetliners to weapon systems for the Department of Defense (DoD). During this time, standards and practices for embedded software development were not uniformly applied by each project/program, resulting in some projects/programs meeting goals more effectively than others. In late 1980, the upper-management of The Boeing Company launched a corporate-wide program to achieve a uniform and effective methodology for embedded software development and support.

The program consists of four parts; 1) a comprehensive development and support standard, 2) a set of guidelines to help implement the standard, 3) the development and implementation of a plan to obtain a set of automated tools which support the standard and guidelines, and 4) an associated training program. This paper focuses on the Boeing Embedded Software Standard, and presents the results of some comparisons to the proposed MIL-STD-SDS and related Data Item Descriptions (DIDs).

The Standards project was accomplished in three phases; a) conceptual, b) planning, and c) development. The project was guided by a steering committee of multi-company experienced software representatives appointed by upper-management. The core project development team consisted of experienced software representatives from each of the participating companies.

WHY A SOFTWARE STANDARDS PROJECT

During recent years, embedded software has increasingly become an important part of the major products of The Boeing Company. These products range from commercial airliners to commercial software to weapon systems for DoD. Over the years, standards and practices for embedded software development were not uniformly applied by each project/program. This resulted in different levels of success among the projects/programs.

The upper-management at The Boeing Company decided that software standards could help provide consistent high quality products, reduce schedule risk, and reduce costs.

BEGINNING THE STANDARDS PROJECT

In December of 1980, participation in a corporate level software standards project was requested of Boeing Commercial Airplane Company (BCAC), Boeing Computer Services Company (BCS), Boeing Engineering and Construction Company (BEC), and Boeing Military Airplane Company (BMAC) by Boeing Aerospace Company (BAC). Each company responded, naming key personnel to support the effort. In early 1981, the Software Standards Development Committee (SSDC) was established with concurrence by the various company executives. Members included experienced software and system managers from each company.

The SSDC first had to define the bounds of the problem, and then develop a planned solution to the problem. They concluded that the nature of the software problem at Boeing was primarily management oriented, not technical. The approach taken was to start with successful software and system engineering management experience, software technical expertise, and existing standards and practices. These proven management methodologies and technical practices were then tailored to the business environment of the Boeing operating companies. To accomplish this goal, the SSDC established the Software Standards System (S^3) Project in August 1981.

The S^3 Project required manning, funding, and direction. Personnel and funding requirements for each company were based on the percentage of embedded software engineers within each operating company. By February 1982, personnel and funding had been provided by each company, with guidance, direction, and approval coming from the SSDC. The charter of the S^3 Project is to develop management and engineering standards, productivity tools and aids, prepare related training classes, and maintain the standards after they have been released. Figure 1 illustrates the S^3 project organization, funding, and direction.

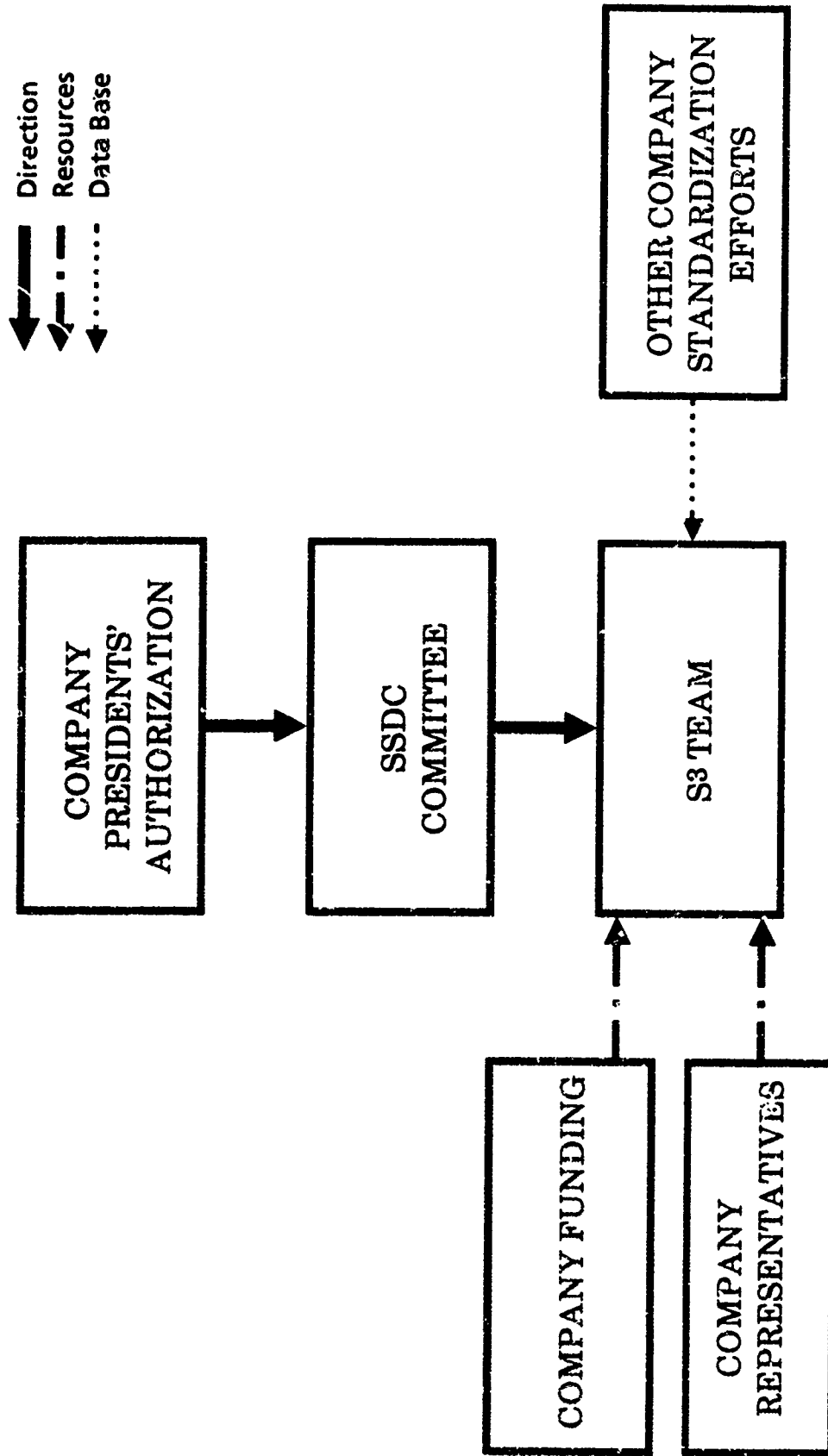


Figure 1 Boeing S3 Project Direction, Resources, And Data Base

THE S³ PROJECT

The scope of the S³ Project is to perform four tasks. Task one is to develop a Standard which provides direction for the development and support of embedded software systems. Task two is to develop guidebooks covering software management, cost estimating, procurement, requirements, design, test, and product assurance. Task three is to perform studies of software productivity tools and aids, and recommend solutions. This task includes recommendations concerning software engineer workstations. The final task is to develop a training program to introduce the Corporate Standards System. It identifies the different groups of people to be introduced to the Standards, and the class requirements for each of these groups. The S³ Project is developing the course content, and training organizations within each company are responsible for conduct of the classes.

DEVELOPMENT OF THE SOFTWARE STANDARD

A team of experienced software people was formed to develop the Boeing Embedded Software Standard. The team developed an outline of the Standard and assigned individual responsibilities. Each individual developed drafts which were reviewed at Peer Review meetings. Each section was then finalized. The process of putting every page of the Standard through the Peer Reviews resulted in an integrated approach to the development of the Standard. Each person on the team was able to ensure that each section fits together with the other sections of the document.

THE BOEING EMBEDDED SOFTWARE STANDARD

The Boeing Embedded Software Standard is a manual which specifies the requirements for the development of embedded software throughout the entire software life cycle. It includes management standards, development standards, and product standards. Upon release, it is to be located at all Boeing operating companies engineering standards stations. The relationship of the Corporate Software Standard to Company Software Standards is shown in Figure 2. The structure of the document is discussed in the following paragraphs.

Many variations of the software life cycle were studied and discussed before adopting the Boeing Embedded Software Development and Maintenance Process shown in Figure 3. The bold box named Computer Program Development includes eight phases which are to be applied to each software item developed. These eight phases are shown in Figure 4. A decision was made to address the software activities in each of the other phases shown in Figure 3 to emphasize the complete software life cycle. The Software Standards System emphasizes that software engineering involvement must begin early in the system development process for the software development process to be successful.

Standards are grouped into three general categories: management, development, and product. The document is structured so that a section is provided for each category. Management Standards define activities, reviews, and products for planning, organizing,

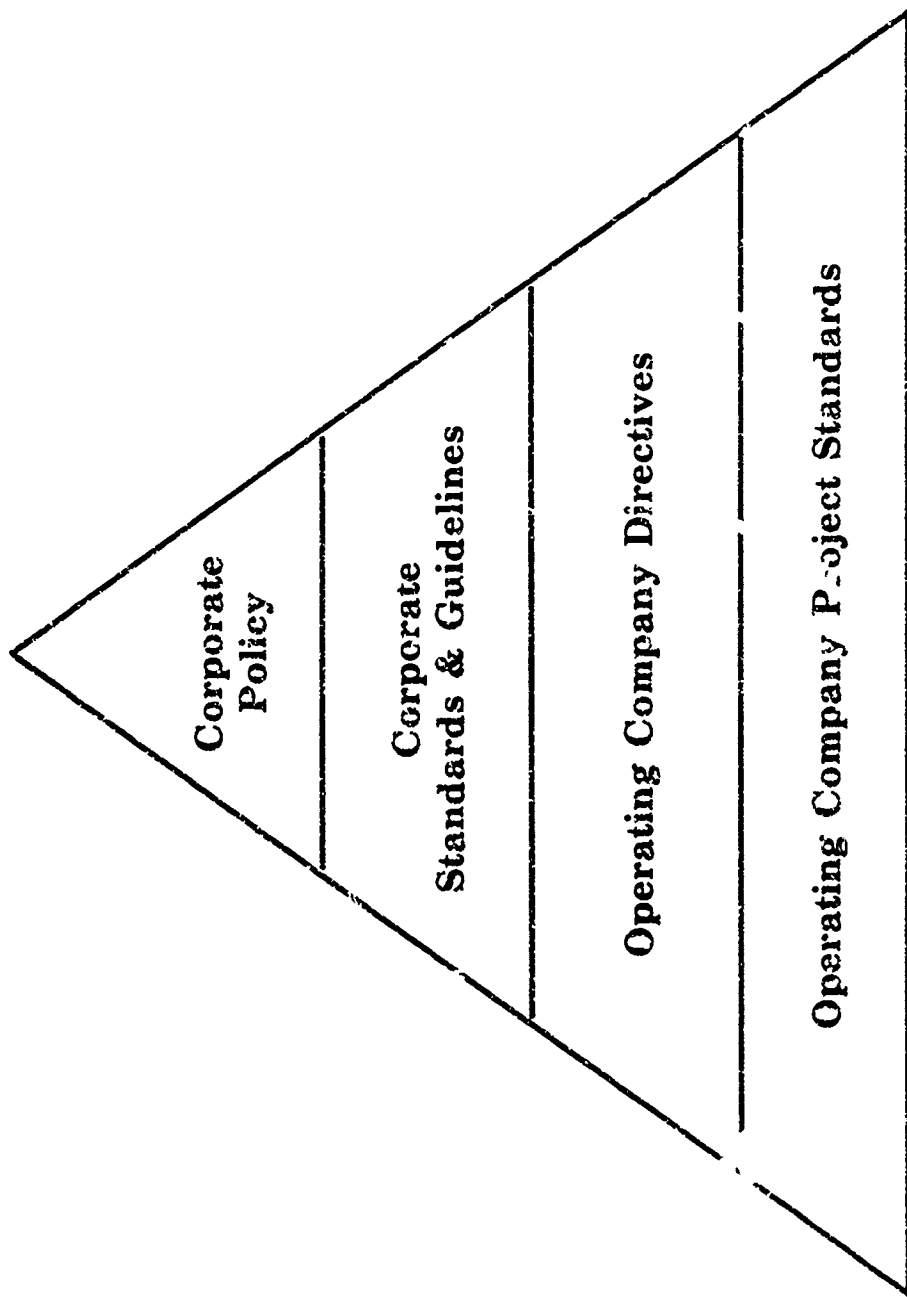


Figure 2 Corporate & Company S/W Standards Relationships

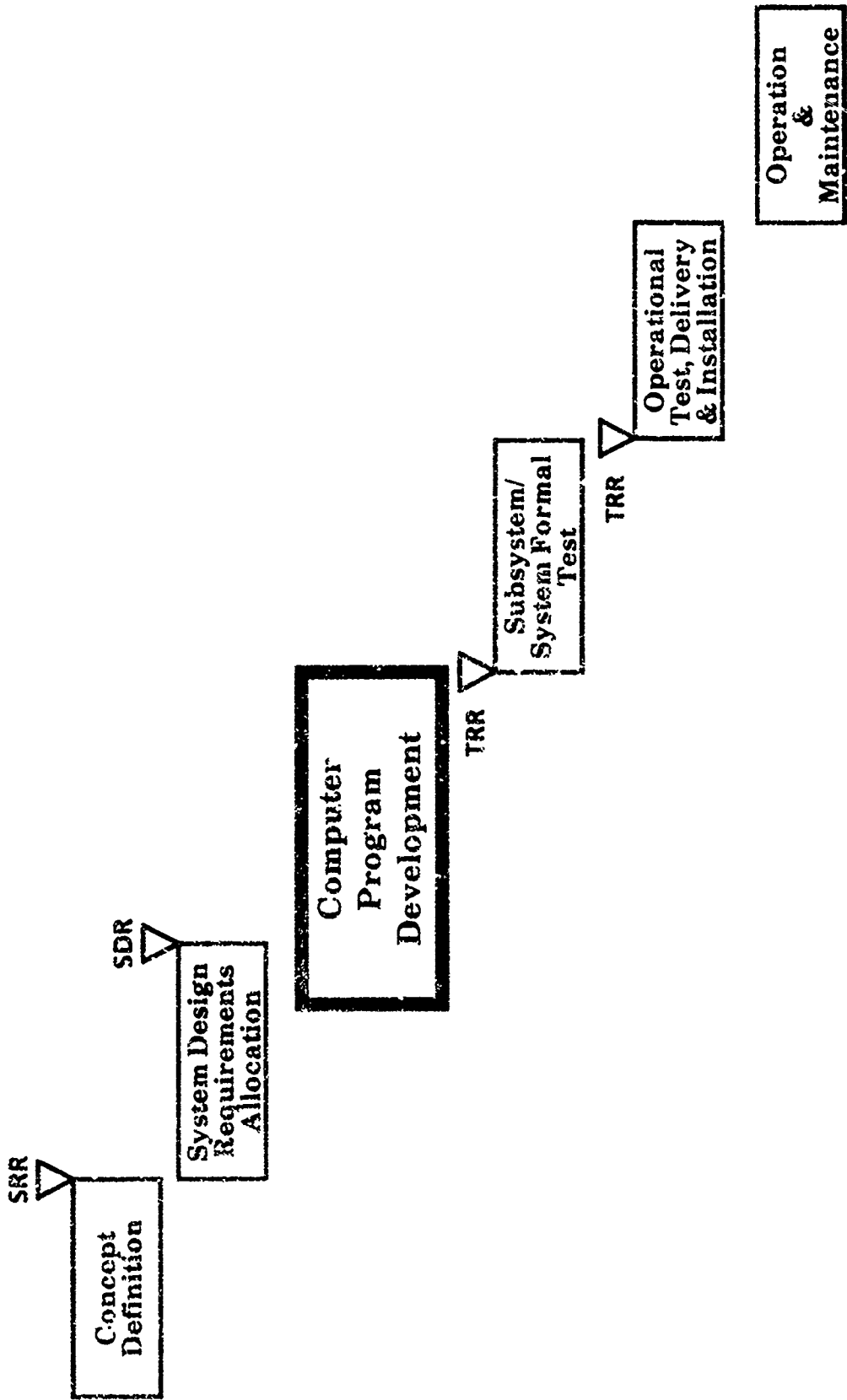


Figure 3 Boeing Embedded Software Development And Maintenance Process

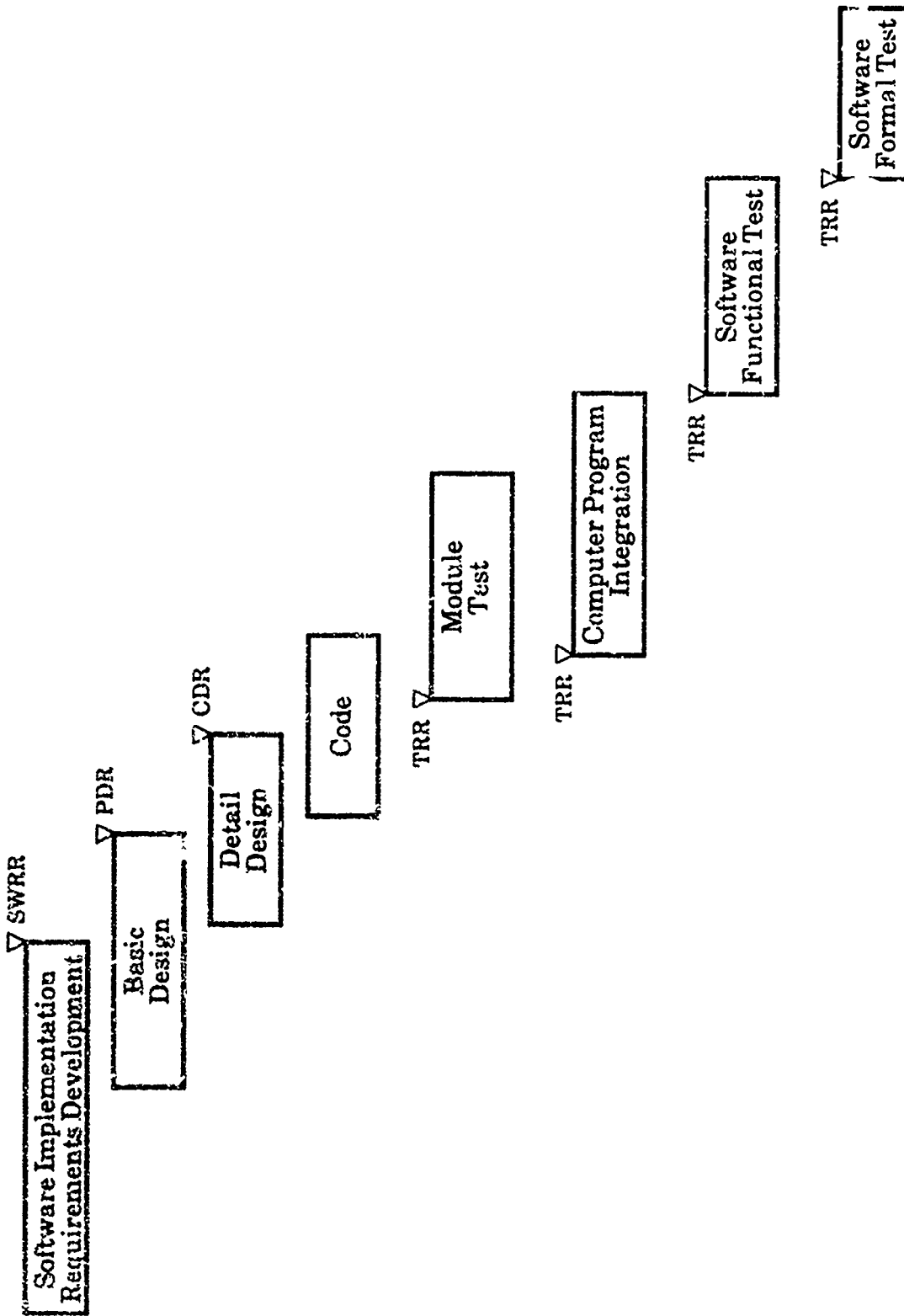


Figure 4 Boeing Computer Program Development Process

controlling, and directing embedded software development. They generally apply to the whole development process, and provide visibility into it. Development Standards are technically oriented. They define the software life cycle and the data requirements (inputs), tasks, and products (outputs) associated with each phase. The Product Standards define the purpose, content, and organization of each product identified in the other sections. They are compatible with Data Item Descriptions for military acquisitions.

The structure of the Standard is a two page format for each topic in the management and development sections, and a general structure for the product standards. The two page format for the management and development sections consists of right-hand and a left-hand pages. This format is shown in Figure 5. The right hand page is a Task/Data Matrix for that topic, showing the standards for data requirements, tasks, and products required. The columns of the matrix correspond to functional activities. Letters are used in the columns to recommend which functional activities provide data, perform tasks, and provide the products. The left-hand page contains further standards in narrative form for the topic. These standards cover timing, quantitative, and other subjects not conveniently expressible in the matrix. The structure of the product standards section includes a Description/Purpose section, and a Requirements section, which defines the content and organization of the product.

THE REVIEW PROCESS

The review process for the Standard is shown in Figure 6. The draft standard was released in July 1982. Copies were provided to the SSDC representatives who served as focal points within each operating company. These focal points distributed the copies to engineering, management, quality assurance, configuration management, test, and other computer related organizations for review. This review by experienced personnel in many fields provided a wide range of detailed comments covering nearly every aspect of the software development life cycle.

Each comment was acknowledged, logged, and tracked. The response of the S³ team to each comment, was returned to the reviewer who provided the comment to close out the comment. If a reviewer was not satisfied with the S³ decision, the reviewer could appeal it to the SSDC for final resolution. Upon completion of comment resolution, the standard was finalized and released.

IMPLEMENTATION

The Boeing Embedded Software Standard will be effective upon its formal release through the Boeing Corporate Standards Engineering Organization. It will be enforced at the company level with each company identifying an organization responsible for company implementation. The key, however, to successful implementation is training.

The underlying concept is that all personnel involved with software development must be trained in the use of this Standard, in order for it to become an integral part of the software development process in all Boeing operating companies. Therefore, executives, software managers, non-software managers, and software engineers will be taught the application and use of this standard. Training of each of these groups is an important element of the S³ Project.

4.10.2 _____

(1) _____

(2) _____

(3) _____

Left Page

Matrix						
	Function					
	Inputs					
Tasks						
Products						
Notes						

Right Page

Figure 5 Format Of The Boeing Embedded Software Standard

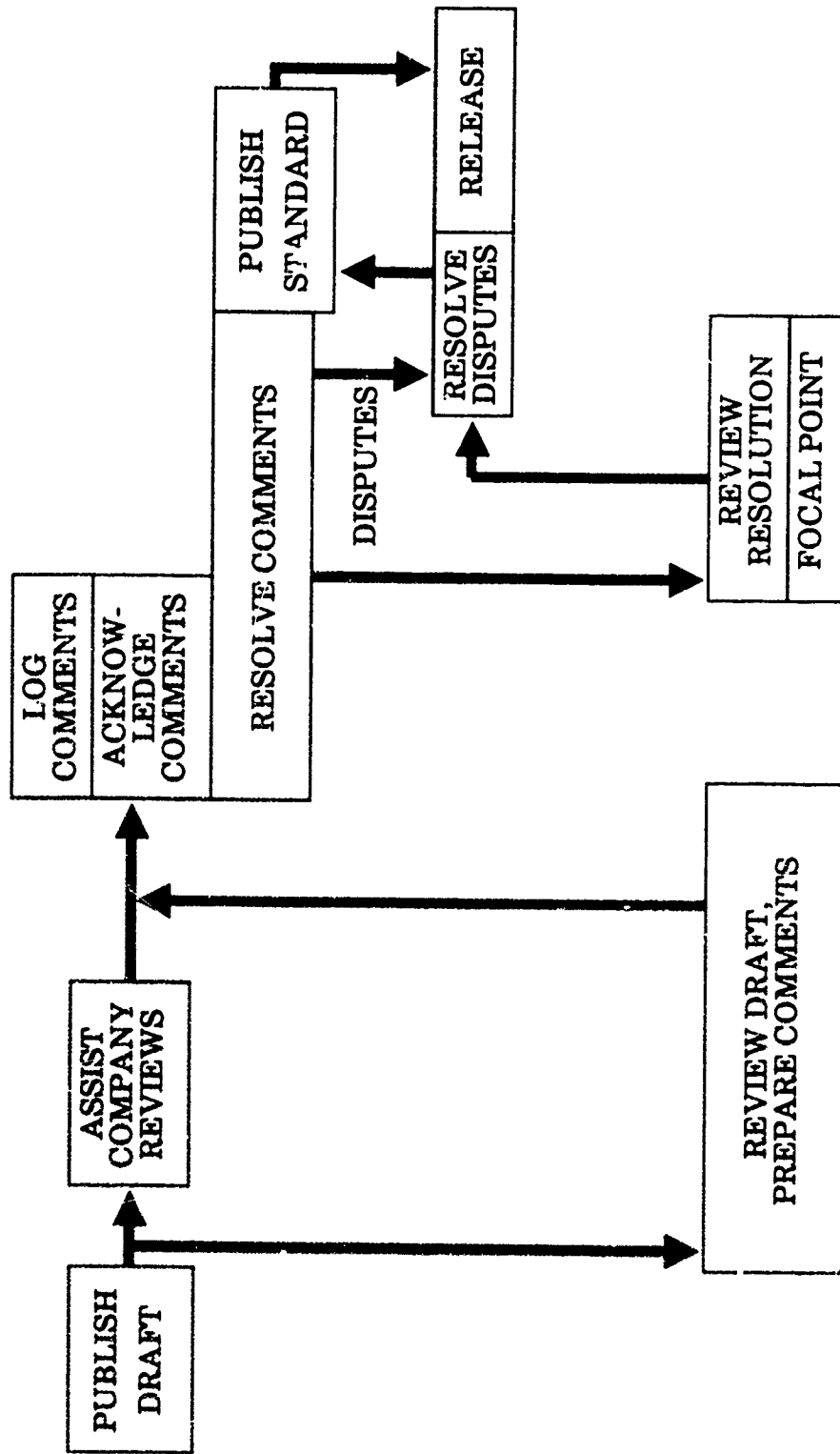


Figure 6 Review Process For The Standard

BOEING VS. THE PROPOSED MIL-STD-SDS APPROACH

The Boeing Embedded Software Standard was recently compared to the proposed MIL-STD-SDS and the related Data Item Descriptions (DIDs). The comparison indicated that the two approaches are functionally equivalent. The Boeing Company is continuing to coordinate with the Computer Software Management (CSM) subgroup of the Joint Policy Coordinating Group on Computer Resource Management (JPCG-CRM) in order to ensure that the Boeing Standard continues to remain in full compliance with the MIL-STD-SDS and the related DIDs. As a result of this effort, changes have been made to the Boeing Corporate Standard and changes recommended to the military standards. The proposed changes to the military standards were sent to the appropriate IEEE, EIA, and AIA committees.

The key differences between the Boeing and the military standards are 1) The Boeing Software Life Cycle begins earlier in the total system development cycle than does MIL-STD-SDS life cycle, 2) the Boeing Standard provides for the allocated baseline to be established as a result of the Software Preliminary Design Review, while MIL-STD-SDS requires it to be established as a result of The Software Specification Review (SSR), and 3) there is a significant difference in the level of control of detailed functional and interface requirements required by MIL-STD-SDS and the Boeing Standard.

In the above three cases, Boeing has recommended changes to the military standards which, in our opinion, should result in improved schedule and cost performance.

CONCLUSIONS

As a result of the increased use and importance of software in its products, Boeing has recognized the need to standardize the embedded software development and maintenance process. Through this standardization, the company intends to increase its productivity and provide a consistently high level of product quality and customer satisfaction.

BIOGRAPHICAL SKETCHES

Dennis D. Doe is presently the manager of Boeing Aerospace Company's Software Technology organization. His assignments have included management of the Corporate Software Standards Development and Implementation effort, Software and System Test Manager for Boeing Engineering Company, Test Manager of the Panama Canal Marine Traffic Control System, Systems Engineering Manager for B-1 Avionics, Design and Lead Engineer on Lunar Orbiter, SRAM, and STA. He has a BSEE degree from the University of Washington.

David E. Hilt is presently the technical lead engineer for the Corporate Boeing Embedded Software Standard project. Since joining the Boeing Aerospace Company in 1978, he has had assignments on the Space Defense Operations Center (SPADOC) proposal, and as Mission Control Lead Engineer for the Anti-Satellite System (ASAT). Prior to joining Boeing, he worked for the Jet Propulsion Laboratory after serving there as a captain in the U.S. Army. He held various technical and supervisory positions on such programs as Mariner Mars, Viking, and Voyager. His specialty was orbit determination software and navigation systems. He has B.A. and M.S. degrees in mathematics from the University of Delaware and the University of New Hampshire, respectively.

Gary B. Wigle is a member of the Software Technology organization with Boeing Aerospace Company, presently assigned to the Corporate Embedded Software Standard project. Prior to joining Boeing in 1981, he served as a captain in the U.S. Air Force. His assignments included being a field systems/software analyst for the Semi-Automatic Ground Environment (SAGE) air defense system, and a software engineer in System Program Offices (SPOs) for the Precision Location Strike System (PLSS) and the Low Altitude Navigation and Targeting by Infrared at Night (LANTIRN) system. He has a B.S. degree in physics from the U.S. Air Force Academy and an M.S. degree in Systems Management from the Air Force Institute of Technology.

Joan P. Bateman is a principal engineer for Boeing Commercial Airplane Company, (BCAC), specializing in software standards and project management methodologies. She is currently the BCAC representative to the Corporate Embedded Software Standard project. Past assignments have included proposal and project manager for software methodology, tools, and documentation contracts, and lead engineer for the E-3A operational program validation, verification, and certification. Ms. Bateman is a charter member of the IEEE Software Standards Subcommittee, and has been active on the terminology, quality assurance, and test working groups. She has a B.S. in mathematics from the University of Pittsburgh and is a MBA master's candidate at Pacific Lutheran University.

Leonard L. Tripp is a senior computer scientist at Boeing Computer Services Company with specialty in the field of system/software development methodologies. His recent assignments have included being a technical leader on projects to prepare software validation and verification guidelines and software engineering standards. He is the chairperson for the IEEE Software Standards Taxonomy Working Group. He is the author of several books and over twenty papers. He obtained M.S. and B.S. degrees in mathematics from Brigham Young University. He is a member of the Mathematical Association of America and the Association of Computing Machinery.

Weldon F. Jackson is a Software Engineering supervisor in the Boeing Military Airplane Company. Since joining Boeing in 1959, he has worked on a variety of software development tasks. These include applications software for one-dimensional heat transfer solutions, utility and data base software for Lunar Orbiter, software development of automated maintenance scheduling for Minuteman missile bases, preparation of SRAM flight test data processing specifications, preliminary design of satellite on-board computer software and long range patrol aircraft acoustical data processing software, supervisor of software development for real-time aircraft and weapon simulators for B-1 Avionics and E-52 OAS projects, and is currently Software Engineering Manager for SRAM. He has a B.S. degree in mathematics from Brigham Young University.



MIL-STD

DEFENSE SYSTEM SOFTWARE DEVELOPMENT

Deane F. Bergstrom
Rome Air Development Center (COEE)
Griffiss AFB NY 13441

Mr. Bergstrom is a Supervisory Computer Scientist in the Rome Air Development Center (RADC), Command and Control Division. As Chief of the Software Engineering Section he is responsible for planning, implementing, and managing a broad spectrum research and development program which addresses critical Air Force needs in the area of embedded computer systems for C2 applications. The program includes efforts in Automated Requirements and Design Technology, Software Engineering Tools and Methods, and Software and System Quality. He received a BA in Physics from Syracuse University and has completed graduate studies in Systems Management with the University of Southern California.

ABSTRACT

The acquisition manager has a difficult task in determining the type(s) and amount of software and documentation to meet mission requirements. In April of 1979 the Computer Software Management Subgroup of the Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resource Management (JLC-JPCG-CRM) sponsored a joint government-industry workshop in Monterey, California to address selected problem areas in the acquisition and development of defense system software. As a result of this workshop's recommendations, the JLC initiated the development of uniform Military Standards.

Rome Air Development Center, in response to the JLC initiative, has produced a draft MIL-STD for Defense System Software Development with the active cooperation and participation of the Army and Navy. The baseline documents used in this effort were Navy MIL-STD 1679 and the RADC Software Development Specification (CP 0787796100B). In addition, updates and improvements have been made to MIL-STD 1521A, "Reviews and Audits"; MIL-STD 483, "Configuration Management"; and MIL-STD 490, "Specification Practices" to be consistent with the new Standard.

MIL-STD

DEFENSE SYSTEM SOFTWARE DEVELOPMENT

INTRODUCTION

In April of 1979 representatives of the three services and industry met at the US Navy Postgraduate School in Monterey, CA to address significant problems in the acquisition, development, and maintenance of defense system software. The ever increasing investment in software and the spiraling costs of maintaining delivered systems focused the attention of this group on both technological and management initiatives which would alleviate the problems and which could be applied across the services in a consistent manner. As a result of the workshop, four initiatives were proposed and subsequently acted upon. The first was to develop a joint-service regulation and policy which would enable the services to view the software acquisition problem from the same frame of reference. This was important if all services were to adopt the products of the remaining initiatives dealing with standards, documentation, and software quality assurance. The development of a tri-service MIL-STD for Defense System Software Development would be structured in accordance with the policy and would contain provisions for the application of a minimum set of software practices aimed at improving the overall quality of the finished products. The documentation of these products was also to be accomplished in a consistent manner while at the same time reducing the proliferation of documentation requirements which has plagued both industry and the services. Software quality assurance was to be addressed as a separate issue although there are provisions in the proposed MIL-STD which specifically deal with software quality in terms of the delivered products. It was believed that the entire concept of software quality assurance had not been given sufficient attention and that there was no consistent methodology for providing the necessary management visibility into the software development process (See Figures 1, 2, 3).

RELATIONSHIPS TO STANDARD

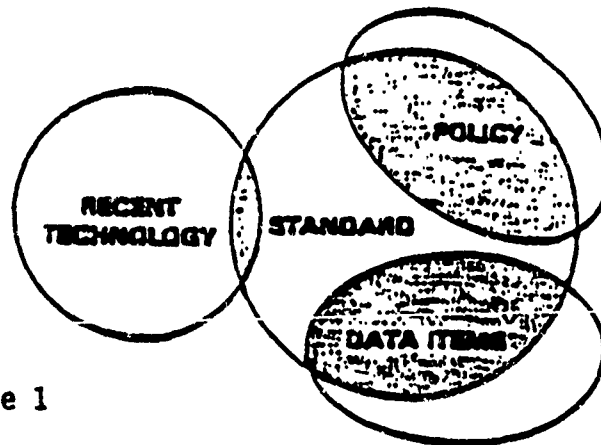


Figure 1

MIL-STD-SDS DEVELOPMENT GUIDELINES

- REFLECT JLC POLICY/ACQUISITION FRAMEWORK (OUR BASELINE)
- COMPLIANCE EVIDENT IN CODE OR DOCUMENTS
- ADHERE TO MIL-STD-962 REQUIREMENTS
- MUST BE ACCEPTABLE TO ALL THREE SERVICES
- AVOID SYSTEM SPECIFIC REQUIREMENTS
- INCORPORATE LATEST TECHNOLOGY
- DRAW UPON MIL-STD-1679 AND RADCS SOFTWARE DEVELOPMENT SPECIFICATION WHERE APPROPRIATE

Figure 2

MIL STD SDS CONTENTS

- COVERS SOFTWARE DEVELOPMENT
 - ACTIVITIES AND MILESTONES
 - TYPES OF INFORMATION
 - APPROACHES/METHODS USED
 - PROJECT PLANNING/CONTROLS
- EMPHASIZES DISCIPLINED APPROACH
 - SOFTWARE DESIGN/IMPLEMENTATION
 - TOP-DOWN
 - STRUCTURED
 - MODULAR

Figure 3

MIL-STD DEFENSE SYSTEM SOFTWARE DEVELOPMENT APPROACH

The approach taken in the development of MIL-STD-SDS was to baseline the document on the latest service unique standards for software and to produce one which properly reflected major service concerns and requirements while removing any system specific requirements which would preclude the application of SDS to a wide variety of software development activities. Thus, Navy MIL-STD 1679 and the RADC Software Development Specification were selected as the appropriate documents. The new standard was also structured to accommodate software practices and technology which had gained acceptance (in use). To insure this a series of six mini-studies was completed and documented. The process for developing the new standard included compliance with MIL-STD 962 for preparing MIL-STD's and the JLC policy which contained a new view of the software development cycle. The JLC software acquisition model does not vary substantially from the commonly accepted version, but does contain new documentation requirements captured in the MIL-STD-SDS and tracked by the new documentation (DID) package.

It was recognized that other military standards would be impacted by SDS. Thus, updates and meaningful improvements were made to MIL-STD 483, "Configuration Management"; MIL-STD 490, "Specification Practices"; and MIL-STD 1521A, "Reviews and Audits." Although these documents are not discussed in this paper, they are part of the final set of draft documents presently being reviewed by the services and industry (See Figures 4, 5).

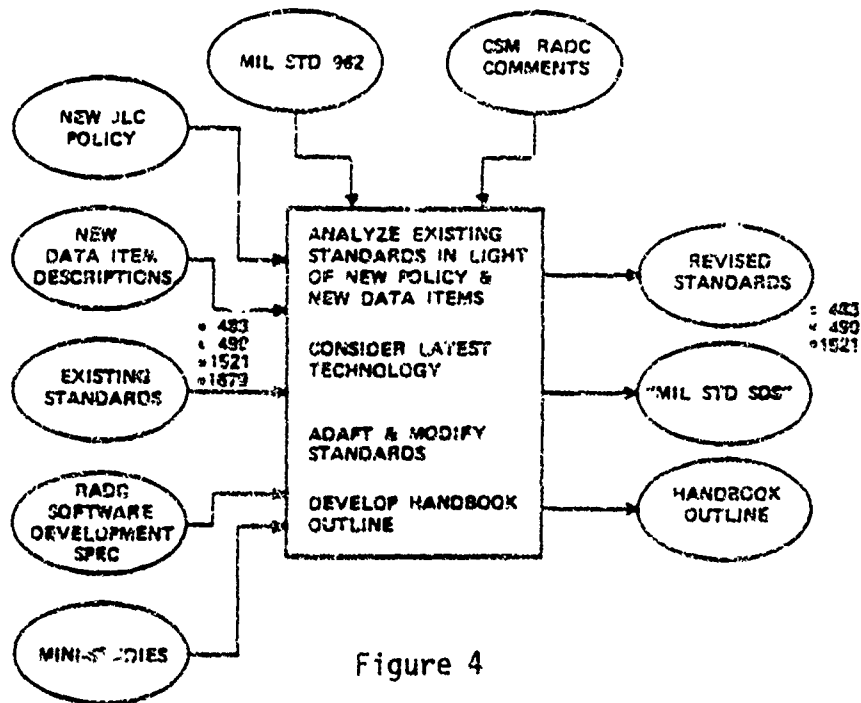


Figure 4

GENERAL REQUIREMENTS

- SOFTWARE REQUIREMENTS ANALYSIS
- SOFTWARE DESIGN REQUIREMENTS
- CODING REQUIREMENTS
- SOFTWARE INTEGRATION REQUIREMENTS
- SOFTWARE TEST REQUIREMENTS
- SOFTWARE CONFIGURATION MANAGEMENT
- SOFTWARE QUALITY PROGRAM
- SOFTWARE PROJECT PLANNING/CONTROL
- SUBCONTRACTOR CONTROL
- DEVIATIONS AND WAIVERS

Figure 5

MIL-STD-SDS, DETAILED SOFTWARE REQUIREMENTS

Software Requirements Analysis - Perhaps the most critical aspect of software development is stating, analyzing, and documenting the system and software requirements to meet mission and development objectives. SDS contains requirements for this phase of the life cycle and also specifically addresses the issues of interface requirements definition, functional requirements specification, and input, processing, and output requirements. The documentation of these requirements includes provisions for software qualification, the analysis approach, the allocation of functions to software components, processing and storage budgets, and the relationship of the requirements

analysis phase to subsequent design, code, integration, and test. A further requirement is to use a structured requirements analysis tool or technique to accomplish the analysis (See Figure 6).

Software Design - The software hierarchy selected for SDS consists of Computer Software Configuration Items (CSCI), Computer Software Components (CSC), Units, and Modules. Software functional requirements are allocated to this structure which decomposes the software into the hierarchical form with CSCI's at the top level. The preliminary software design process translates software requirements into a top-level design for each CSCI using a modular architecture and defines a framework for the detailed design phase. The allocation of CSCI functions to CSCs and below follows this stepwise procedure until all software functions have been accounted for. Also, external interfaces to the CSCIs are allocated and specified. Detailed requirements are stated to insure that complete interface definition and control are considered from both a functional and performance (timing) basis.

A Program Design Language is to be employed by the contractor to develop the top level design. The purpose of this requirement is to use a formalized design approach which provides the benefit of "self-documenting" design detail. The output PDL can then be used for code development and updated to reflect new or changing software requirements.

DETAILED REQUIREMENTS: HIGHLIGHTS

- **STRUCTURED REQUIREMENTS ANALYSIS TOG'S/
TECHNIQUES**
- **TOP-LEVEL SOFTWARE DESIGN**
- **TOP-DOWN DESIGN USING PROGRAM DESIGN
LANGUAGE**
- **TOP-DOWN IMPLEMENTATION**
- **UNIT DEVELOPMENT FOLDER**
- **QUALITY REQUIREMENTS SPECIFICATION,
MEASUREMENT, AND ASSESSMENT**
- **PROCESSING RESOURCE ESTIMATION AND
MONITORING**
- **PROGRAM SUPPORT LIBRARY**

Figure 6

Although a top-down approach is to be followed, exceptions are allowed when the need for early design of critical software has been demonstrated. This mechanism is used here and in other paragraphs of SDS to provide sufficient flexibility for mission and software contingencies and to allow for contractor innovation.

Coding Requirements - Code production under the auspices of SDS requires the contractor to comply with a minimum set of coding standards and conventions based on a structured code methodology. In addition, the coding standards and conventions require the use of a high order language, modularity, naming conventions, symbolic parameterization, commenting, and error and diagnostic messages which are uniform. Other coding standards are required which address numerical, Boolean, and mixed mode operations. Although the set of coding standards and conventions contained in SDS is meant to be applied to defense system software development, the contractor may have adopted an analogous set of standards for internal use or in response to prior government efforts. If his coding techniques provide the same level of control, apply top-down methodology, and are consistent with SDS, then such a set could be proposed in response to government RFP's. It is not the intent of SDS to constrain the software development process unnecessarily nor is it desirable to limit competition. Corporate tools and methods which provide analogous capabilities are encouraged (See Figure 7).

HIGHLIGHTS (CONT.)

CODING STANDARDS:

- **USE OF HIGHER ORDER LANGUAGE**
 - **RESTRICTION ON CONTROL CONSTRUCTS (SEQUENCE, IF THEN ELSE, DO WHILE, DO UNTIL, CASE)**
 - **PRECOMPILER/SIMULATION ALLOWED**
 - **MODULARITY**
 - **SIZE**
 - **SINGLE FUNCTION, ENTRY, EXIT**
 - **SYMBOLIC PARAMETERIZATION**
 - **NAMING CONVENTIONS**
 - **AVOIDANCE OF MIXED-MODE OPERATIONS**
 - **RESTRICTION ON LOOP INDEX MODIFICATION**
 - **PARAGRAPHING, BLOCKING, INDENTING**
 - **AVOIDANCE OF COMPLICATED EXPRESSIONS**
 - **SINGLE STATEMENT PER LINE**
 - **COMMENTS.**
 - **ERROR AND DIAGNOSTIC MESSAGES**

Figure 7

Software Integration Requirements - SDS requirements for the integration of software components calls for the early demonstration of software capabilities and the documentation of the software integration and testing approach. In addition, the contractor is required to present the results of the above process in a Test Readiness Review. In the past, both the contractor and the government have scheduled tests prematurely which leads to "wheel spinning" and cost growth. The Test Readiness Review should eliminate or alleviate the situation. Software performance testing requirements in SDS establish software performance in accordance with CSCI requirements and approved software test plans, descriptions, and procedures.

Configuration Management - Internal contractor procedures to provide administrative and technical direction and surveillance are required by SDS to identify and document the functional and physical characteristics of CSCIs, control the changes to those characteristics, and record and report on the processing of change and the status of the implementation. Specific SDS paragraphs cite requirements for implementing adequate software change control procedures for formally controlling all changes to baselined documents and program materials. Problem management is required and a means to identify, track, and resolve software problems must be established by the contractor. The problem management approach must describe the problem in sufficient detail to permit analysis and a recommended solution or disposition, describe the affected documentation, and describe the impact on baseline(s), cost schedule, interfaces, or any external system requirements.

Software Quality Program - SDS contains provisions for establishing a software quality program to define the requirements for, achieve, and evaluate the quality of the implemented software and associated documentation. At the discretion of the procuring agency, the contractor is required to identify the degree of excellence which the composite attributes of the software must exhibit to fulfill operational requirements and he must do so in quantifiable and measurable terms. Attributes include compliance with all specified functional and performance specifications, characteristics such as adaptability, maintainability, reliability, portability, interoperability, etc., and other software attributes which provide a determinant of excellence. Recent advances in software metric data collection and analysis support this SDS requirement and may, in the future, be used as the basis for incentive fee awards to contractors that demonstrate superior performance. Conversely, the software quality program could provide leverage when the government takes the position that the contractor has failed to meet established quality goals.

Software Project Planning and Control - The contractor is required by SDS to implement procedures for planning and controlling the software development. Sizing and timing budgets

set at the initial phase of the effort are to be monitored during the remainder of the program and presented at the major milestone points. A comparison is made to the baseline and appropriate action taken whenever new information warrants an update to CSCI sizing and timing records. Also, the contractor is required to maintain status and cost forecasts, analyses, and reports for each unit, CSC, and CSCI. The use of a program support library is required and may be specified as a deliverable entity. The library system used will depend on the nature of the implementation and could range from a manual technique to an automated system complete with management reporting and documentation capabilities. Unit development folders are to be used by contractor programming personnel to document each unit of software produced which completely portrays the design, code, and test information to a level of detail to allow other technical personnel to fully understand the status and characteristics of the software described in the folder.

Subcontractor Control - SDS contains a general provision for subcontractor control which requires compliance with the contents of SDS which have been specified by the government in the contents of the procurement package. Thus, if the prime contractor decides to employ a subcontractor for software development, compliance with SDS is assured.

Deviations and Waivers - Exact compliance with SDS is required. In the event that the contractor proposes to deviate from the standard and applicable Data Item Descriptions, he must do so in accordance with the provisions of DOD-STD-480; "Configuration Control" for Engineering Changes, Deviations, and Waivers or the short form in MIL-STD-481.

CURRENT STATUS AND FUTURE PLANS FOR MIL-STD-SDS

The draft versions of SDS and the updates to MIL-STD 483, 490, and 1521A have been delivered to the government. At present, these documents are being reviewed and commented on by the three services and industry. The standards are accompanied by the draft set of Data Item Descriptions.

A contractual effort is underway to analyze the results of the review and produce a final (coordination) version of the document set. The analysis will be documented in a report which will contain the source comments, their origin, impact on the proposed documents, and their disposition. Also, professional development materials will be produced which will enable the development of DOD software acquisition and management personnel skills necessary to apply the new standards in a comprehensive and consistent manner. Case studies and a formal course will be available at the conclusion of the effort to conduct pilot training sessions for DOD personnel and will be accompanied by a guidebook for applying and tailoring the documents. This effort

is expected to be completed during the fourth quarter of FY83 (See Figure 8).

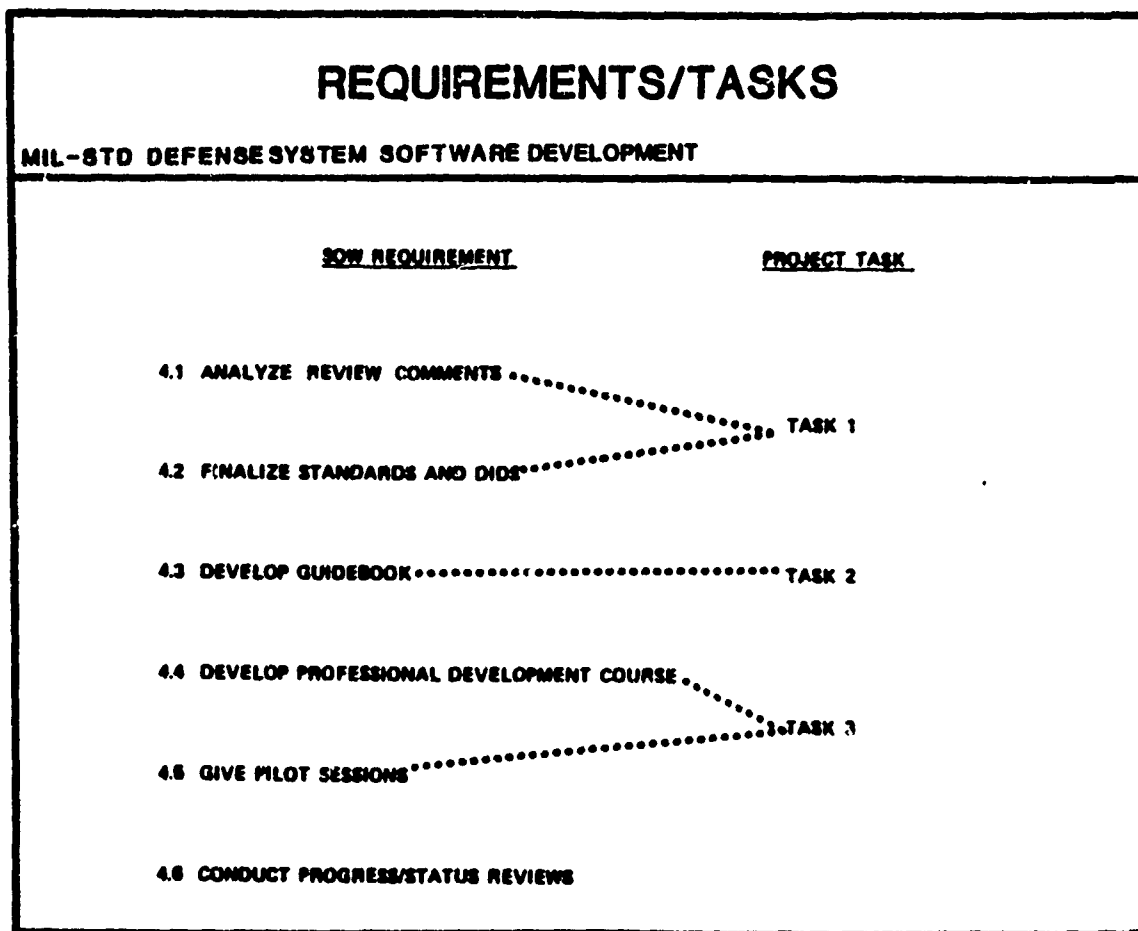
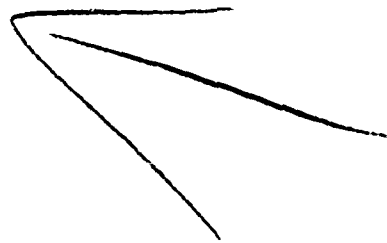


Figure 8



COST/SCHEDULE MANAGEMENT FOR SOFTWARE DEVELOPMENT

Major H. Wendt, USAF and M. W. Evans

Major Howard W. Wendt, USAF is Commander, DCASPRO-Ford, Palo Alto, California. He previously held positions in NATO Logistics Plans at HQ USAF and at the AGM Joint Program Office, Crystal City, Virginia. He has attended AFITs' Education with Industry Program at Westinghouse Corporation, Maryland and graduated from DSMC's Program Management Course. He holds a B.A. Degree in Economics from the University of Nebraska and a M.S. in Computer Systems Administration from George Washington University. Major Wendt's current emphasis at Ford is specifically in the area of monitoring in-house software development. The software projects under contract are from the Air Force and the Navy. Extensive work has also been done by Major Wendt to develop a reference library of DoD software guidance and standards for application assistance in the software monitoring process.

Michael W. Evans is Vice President for Management Sciences and is responsible for managing the services portion of our business. Mr. Evans has over 20 years experience in managing software personnel in a variety of development environments. He has worked for Univac, IBM, Litton Industries, ITT, and most recently has been manager of the Software Standards, Procedures and Control Department for Ford Aerospace's Western Development Laboratories (WDL).

At Ford Aerospace, Mr. Evans reported to the Director of Software Engineering and was responsible for the specification and implementation of all software development, management and quality assurance standards.

Mr. Evans is a recognized expert in software project recovery techniques, software methodology, software test and integration, and configuration management and has published many articles successfully applying development techniques to a number of different project environments. He is currently under contract to John Wiley and Sons to produce a book entitled Management of the Software Development Process which will be published in early 1983.

Supporting this staff are technical personnel with specific experience in the design, implementation, and management of software support in a variety of project and application environments.

ABSTRACT

The importance of good cost/schedule analysis in the Program Management environment is taken for granted. Methods and procedures for WBS have been structured, taught, and generally are accepted as good management techniques. In a hardware environment, this WBS structure is applied to a proposed design or prototype. The basic hardware elements and interfaces are known; cost schedule estimates can be applied to the specific design elements. Software programs do not.

have a structure or design at contract award. A preliminary design compatible with current WBS criteria is not available until PDR. The pre-PDR software development is comparable to hardware R & D; what design does exist is based on a supposed prototype or model and not a suitable basis for cost/schedule forecasts.

This paper discusses the need for a well planned Software WBS and the numerous management and technical factors which must be planned for and developed to insure the WBS adequately represents the program and provides the needed controls.

INTRODUCTION

The planning and implementation of software financial schedule, administrative practices, and early definition of project monitoring practices and procedures is an essential component of project success. In order to maintain accurate visibility into the technical and administrative health and status of the project, the software manager must incorporate into the project structure requirements for project reviews and audits which evaluate progress against plans. These provide data to evaluate the effectiveness of the software project organization, assess project administrative controls and practices and measure the technical integrity of the project.

The problem with planning and implementation of these controls in a project environment is that they are often looked on by software management as an adjunct to hardware project development, ancillary to the technical activities of the project and, as a result, of secondary importance. The cost and schedule monitoring requirements are often documented through generic plans and procedures based on non-project specific criteria and then implemented by personnel with neither project or software development experience. The monitoring approaches implemented are often not tailored to specific project characteristics, and do not integrate the diverse and often conflicting areas of software development. Even when they do, they may not be accepted by software management or project personnel as a constructive, positive source of data.

INITIAL PLANNING FOR SUCCESS

The initial planning phase of a software project are often the most crucial phase, for it provides the framework under which the project will operate and establishes the cost and schedule criteria upon which project success will be evaluated. The quality of the planning phase determines the success of the project, not merely with regard to meeting cost and schedule requirements, but also with regard to the integrity of the products and the level of staff morale and commitment throughout the duration of the project.

The planning of all segments of a software development project before the start of work or allocation of resources is an essential prerequisite to project success. Software cost control planning is one of particular importance and one of the most difficult to develop.

Projects which are not well planned are characterized by software development events and activities initiated by haphazard estimation of software size, unrealistic projection of resources requirements, and scheduling done too quickly. All of these factors result in lack of management and staff commitment, essential for success. A pattern of unanticipated project activities and frequent unplanned development catastrophes and crises is an unfortunate symptom of an inadequately planned project. The pattern is compounded by continued management difficulty controlling the many products of software development, allocation of resources to development tasks and general inability to smoothly move the project from phase to phase.

These diverse planning steps have a common foundation, the Work Breakdown Structure (WBS). The Work Breakdown Structure is the prime means by which the program office organizes and allocates work throughout the organization.

The WBS is a product-oriented division of hardware, software, services, and other work tasks which organize, define, and graphically display the work to be accomplished, in order to develop a specified product. The WBS subdivides the work into manageable units through the various levels of hierarchy.

A WBS is a graphic illustration of the determination of program objectives; and the progressive passing down of these objectives from higher to lower levels of management. Its configuration, content, and detail will vary and will depend upon:

1. The status of the program.
2. The size and complexity of the program.
3. The organizational structure of participating organizations.
4. The arrangement of responsibility for the work to be performed according to the judgment of management.

The initial determination and definition of program objectives through the WBS should assure that the project objectives are fully supported by lower level objectives, that the program structure is fully integrated, that each part of the project is consistent as a whole, and that resources, schedules, and development requirements are properly defined and allocated throughout the program organization.

SOFTWARE PROJECT RELATIONSHIP

The software project is most often a segment of a larger system development and as such, must be implemented within program constraints and in accordance with program financial and schedule control systems. The software development is initiated to make a profit for the company, enhance company effectiveness or productivity, not advance the state of the art or solve a difficult technical problem. This pragmatic reality is often lost sight of by software managers limiting the success potential of the software project and greatly increasing the development risk. The software manager often forgets the program profit commitment and critical financial and schedule constraints in the pursuit of technical excellence.

The relationship to the financial and schedule controls have several parts all of which require support from the software organization.

There are many parts to the planning and implementation of the program cost and schedule controls requiring participation, interaction, and negotiation between the many program, engineering, and support organizations, and most especially, the software project organization.

Within the program environment, the software project is only one of many program elements, any one of which has the potential for delaying or impacting program success. In a typical program, these elements include systems engineering, hardware development, integration and test, installation, logistics support, training, and others which must be effectively managed at the program level. The program manager must orchestrate the activities of each functional area of the program, manage the interactions, interfaces, and overlaps between them, and control the flow of work and allocation of resources between organizational elements of the program. The degree to which the manager is successful in these roles has a direct relation to how successful the program is, and as a result, how successful the software development was in the context of the system objectives.

From the program managers standpoint, the overriding constraint is to be sure that the activities of these diffuse project elements work together in an effective and controlled manner, and that the program is developed within preallocated schedule and budget constraints. To the program a smooth running project is more important than technical excellence.

THE WBS PROGRAM RELATIONSHIP TO THE SOFTWARE DEVELOPMENT

The WBS forms the basis for planning and validating the program structure and organization.

The basic program objectives which form the core of the program planning process, provide, when properly expressed and translated into a WBS structure, the focus and organization to the program activities. These objectives are to:

1. Provide a framework to identify project requirements separately from the performing organizations, cost estimating and accounting systems, funding sources, etc.
2. Provide insight into project element inter-relationships and overlaps.
3. Identify specific work packages for time, cost labor and material estimating, pricing, budgeting, work assignment and authorization, accounting, and reporting.
4. Allocate system reliability requirements to subsystems and components.
5. Establish a specification tree for defining documentation hierarchies and production requirements.

The WBS then is the basis for planning and defining the structure for the program, allocating the work and tasking individual organizational elements in a coordinated and focussed manner.

From the WBS, each of the functional organizations within the program establish and define a cost schedule and development structure compatible with the other segments of the program.

WBS DEVELOPMENT CONCEPTS

The software WBS develops through several related forms as the software project is estimated, planned, designed and coded. Each of these phases represents a specific application relative to the WBS and each contributes to the final software WBS which will monitor the status and progress of the software cost and schedule. There are numerous considerations that the program manager must take into account as he begins to build the WBS.

1. A projected schedule of the tasks which must be done, how long each will take, and in what order the tasks will be accomplished.

2. An estimate of the resources needed to accomplish project objectives, including manpower, facilities, equipment, and other direct costs, such as trips, training and supplies.

3. A cost estimate for the entire project based upon the schedule and resource estimates.

4. Detailed, written plans projecting how the software project is to be managed, monitored, and controlled. The plans should answer:

- What are the responsibilities of each organizational element and staff members within each.

- What are the reporting relationships and responsibilities between organizational components and between staff members.

- What are the development standards to be applied to the project and how are project adherence and quality to be monitored and assessed.

- What are the components of the development environment and what experience may be anticipated when implementing the project under the environment.

- What project procedures development methodologies and controls will be applied and how will success be measured.

- How will costs and schedules be monitored.

The software project must initially translate these basic parameters into a structure which is tailored to the project characteristics and anticipates development requirements and the flow of work as the implementation proceeds. Basically this tailoring must be done in two specific areas:

1. The organizational resource, i.e. people, dollars, time.

2. The tasks, what is to be done by what schedule, for what cost.

ORGANIZATION DEVELOPMENT FOR WBS

From the organizational base, the various software project organizations implement each of the documented project parameters; organizational and administrative controls and practices, budget, schedules and monitoring practices and procedures, and the technical, development, and proper control methodologies (Figure #1).

Cost/Schedule planning and control for software present a particular challenge to the software manager. While the initial planning during Pre-Award and Post-Award is most crucial, it is also the most difficult because of the uncertainty of the cost and schedule information.

The software dilemma is particularly critical due to the uncertain nature of initial software estimates. This is not to say that some software cost estimates are not accurate; many are and they are becoming more accurate. The problem stems from the way these estimates are made, the supporting documentation and the program structure during early program phase.

In trying to devise a method of estimating schedules and costs, the software industry still uses the number of lines of code produced as a measure of which to base for length of time it will take to produce a software system.

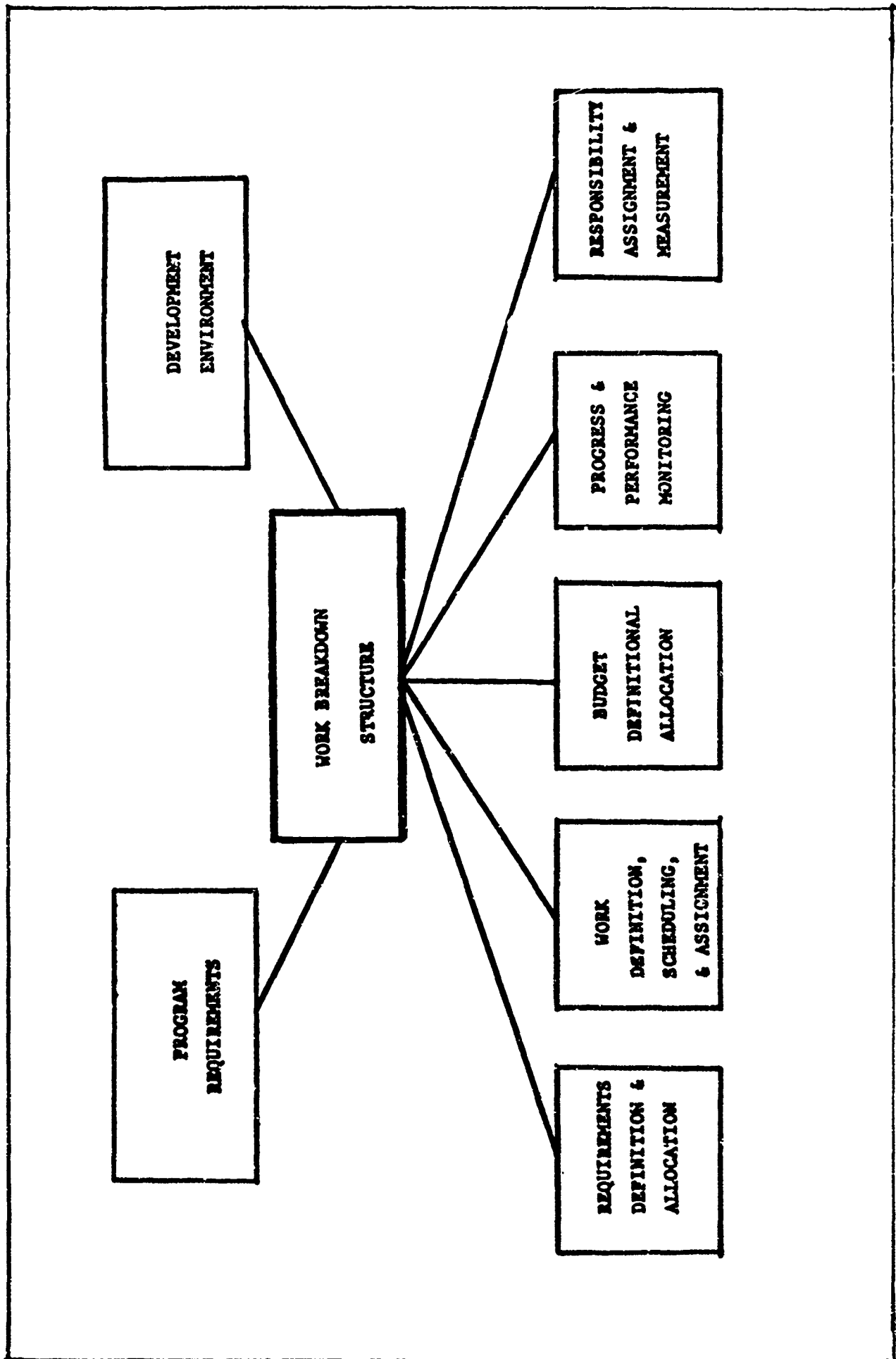
This measurement is not a very good one, and is somewhat similar to estimating how long it will take to produce a painting by estimating the number of brush-strokes that will have been applied when the painting is complete. First of all, most programmers do not know how to estimate lines of code, and wonder what it has to do with anything, anyway. Most programmers, unlike typists, have not done the same work over and over again, and do not have a reference point from which to measure. Programmers do not want to code the same application over and over again, and before they get to the point where they could tell you how many lines of code a particular application will take, they have requested to change areas or even moved to another company looking for a different software challenge.

Unfortunately there is not a convenient measure available to measure software size and, as a result, the use of the parameter persists as a means for sizing software applications.

A second problem impacting the accuracy of the estimation process is the poor specifications of user, system, and design requirements early in the project. Software managers are forced to derive complete sizing of software systems based on incomplete requirements and system specifications. This lack of a firm technical basis for software costing makes all costing suspect and subject to inaccuracy, and not a valid projection of software development requirements.

Organizational planning provides the software manager with the tools to structure his people and time against the functional software development tasks; design, code, test, etc. It should provide him with a rough idea of how many people will be needed in each particular skill and provides some ideal of timing.

The next problem, however, is to assemble this vast research and planning base and apply it to the specific product to be produced, the Computer Program Configuration Item (CPCI), the modules, the units, and the various documentation products.



Assembling the specific production tasks into a WBS is the heart of the structure which will be used for cost and schedule control. The product tasks, as we will discuss later, cannot be identified in detail at the beginning of the program and must rely on the organizational structure planning for early estimates of cost and schedule.

DEFINING THE ORGANIZATIONAL STRUCTURE

In a nominal project, the major software development task categories (Figure #2) are:

1. Software Planning - Those tasks which result in development implementation. Tasks required to plan, schedule, and budget the application of support resources to project activities design, document, and implement software planning parameters, and activities required to maintain and modify software planning parameters adequately in terms of project and development realities.

2. Document Production - These tasks which are required to develop, document, produce, publish, review, and maintain all project documentation.

3. Software Design - Those technical tasks which translate stated user performance operational and interface requirements into a software system design which is translatable into code.

4. Code and Coding Control and Support - Those tasks which result in development of software code, control, review, and document the coding process, and test and integrate software modules into units which may then be integrated into a software system configuration.

5. Test Integration, and Demonstration - Those tasks which plan development, implement, execute, evaluate and document development, integration, and functional testing in accordance with project and contract requirements.

6. Software Configuration Management and Support Tools - Those tasks which implement the software configuration management function control software baselines and data, and assure audibility and traceability between project baselines.

7. Management, Customer Support and Project Review and Overhead - Those tasks which are required to implement management techniques, procedures, and disciplines, customer control and interface requirements, and project review and evaluation requirements.

For the purpose of defining the mix of manpower, those seven areas are further broken into a lower level of tasks which must be completed in order to effectively support project requirements although the task areas identified in Figure #2 are representative of those required to complete the development of a software product, the actual tasks must be tailored to the technical and environmental parameters of the project. This tailoring is accomplished by analyzing how the project is to be developed; what controls are to be placed on the development, reviews, audits, and management overhead anticipated for the project; the technical steps necessary to implement, integrate, and demonstrate the project; and, delivery requirements implicit in the development effort.

Software Development Organizational Tasks

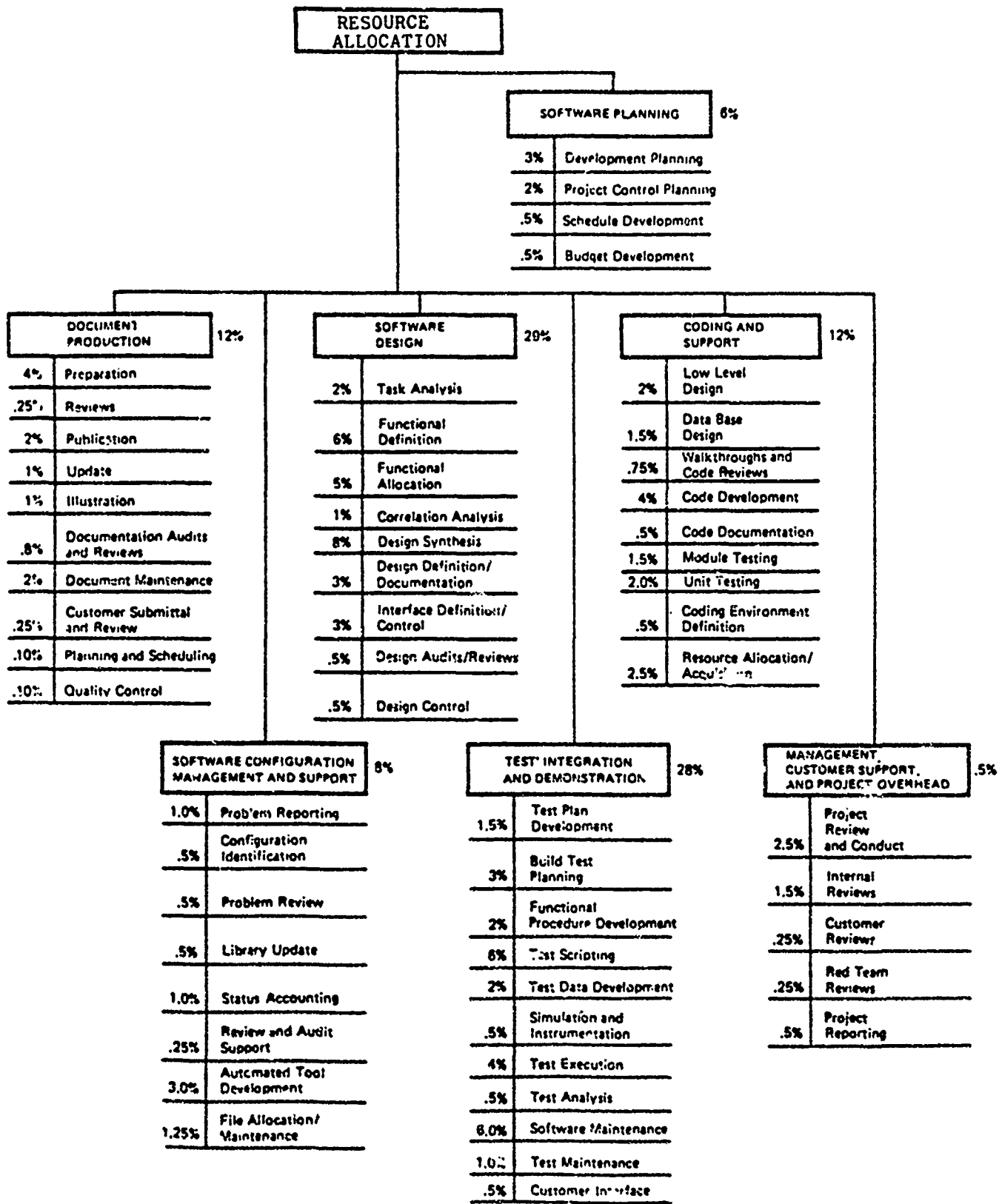


Figure #2

Those functional task descriptions are further decomposed into the specific project task areas which are needed to develop, document, test and control the software product.

This decomposition is a detailed breakout of each of the tasks which support the major tasks and should bear a direct, traceable correspondence to the specific tasks identified in the software development plan. This categorization is an essential planning component in that it allocates and/or gauges the many diverse software development tasks into a structure compliant and consistent with the characteristics of the software development project and tailored to the specific program and contract requirements.

The planning must not stop here. It must now be translated into specific development tasks which can be measured, and traced for cost accountability. A temptation is to stop the software planning at this point and manage cost and schedule from a functional view point. For example, we might consider how complete the overall coding tasks or design tasks are.

These areas can in fact be measured and do provide the manager with an idea of his expenditures and schedule for his resources. They do not provide the customer, however, with a measure of his product development. He wants to know what components are on schedule, which are behind. Where are the high cost variances? What are the projections for getting the development task back on schedule, within cost? To do this, the organizational planning must now be translated into a product environment.

SOFTWARE PRODUCT ENVIRONMENT

Defining a software design during the initial contract and planning phases is not practical. Developing the design is the purpose of the contract and should not be anticipated at the beginning of the program in enough detail to base cost and schedule planning on it.

By definition, Embedded Computer Resources (ECR) are tied to some form of hardware development. When WBS is applied by DoD in the form of C/SSR or C/SCSC, it is applied uniformly to the hardware and the software. A WBS is reported to the third, maybe fourth level of the system. The system at this time (Figure #3) is sufficiently developed to demonstrate a breakout of the software system into several distinct parts as they support and evolve from the basic hardware components.

Defining software at this level is comparable to defining an aircraft as needing a fuselage, wings, and an engine. This is not to say that more is not known about software projects but little more is asked and the software WBS from this point can take whatever form or basis the contractor desires. Without the detail breakout expected from hardware, software development is treated as a subset of the hardware program. In this instance, reporting of Computer Program Configuration Items (CPCIs) would begin at the fifth level. This WBS approach, without further lower level reporting, does not provide control, visibility into problems, an audit trail, or a means for specific cost and schedule projections. Usually fourth and fifth level items are reported on an exception basis. Many contractors will maintain only internal records below this level.

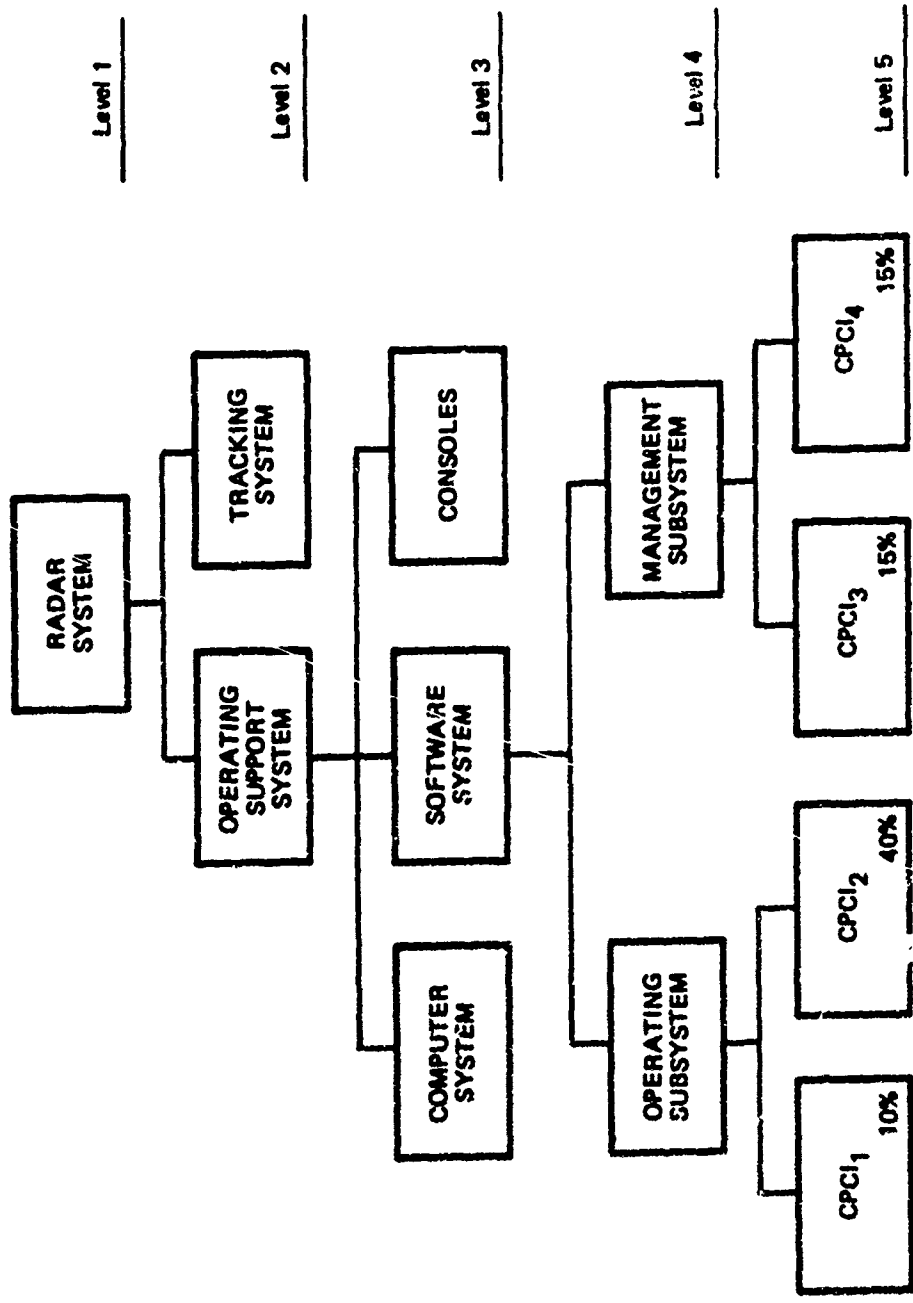


Figure #3

Without specific guidance, the contractor will breakout the software tasks to best suit his purpose, one of which is identification and allocation of personnel resources. It is a helpful planning tool as we have discussed, but it is not product oriented. That is, it cannot be translated into a milestone display over time without a breakout against a product and the proposed design.

REQUIREMENTS WBS

An item which can be used consistently to measure cost and schedule are the "requirements" which are specified for development of the software. The majority of these requirements are known during the proposal and negotiation phases of the contract. They also are the basis for the subsequent design process. This continuity factor through the life of the contract makes the requirements an ideal source for cost and schedule development. At the start of the program, a requirements WBS can be developed (Figure #4) which outlines the relationships of the estimated costs to the known requirements. Some funds will still be undistributed and held in accounts for Level of Effort (LOE) tasks and management reserve. However, a significant amount of the contract can be budgeted against the requirements.

As we have stated, this use of requirements and measurable tasks forces subsequent program planning into a task oriented environment. The organizational planning is still a good planning tool to estimate the cost to complete the work for each requirement. As we can see, (Figure #5) each requirement can be budgeted for the design, code and test efforts necessary for their completion. The separation of functional tasks for each requirement begins to provide an idea of the schedule of events for developing that requirement.

DEVELOPMENT CYCLE INTERFACE

As we can see (Figure #6), the requirements analysis and preliminary design consideration take place prior to PDR. These early phases are critical planning times but they do not offer the insight into the software design that is needed for a true WBS. The requirements are established and the personnel planning completed prior to PDR and we do have a preliminary software WBS based on the requirements and personnel plans.

The PDR becomes a crucial point for conversion of the requirements WBS into a design oriented WBS. The purpose of a software PDR is to obtain early mutual agreement on the very basic structure of the software design.

Figure #7 now shows these requirements applied to the more traditional "design WBS". The requirements are still individually identified and are now used to form the basis for the CPCs. Actually, the requirements will take the form of an individual module or unit to be coded and could be broken out for the next WBS level. The requirements cost are now additive within the CPC to the CPCIs and will still be representative of the program cost breakdown.

The audit trail now begins to show the initial budget translated into tasks based on the requirements. This audit trail then continues through the entire design process as the requirements are incorporated into the design and design budget is then correlated to the previous requirements budget. There is an

inherent fear on the part of most customers to allow any major rebaselining once the program has started; although it happens as a regular occurrence when the program slips or overruns. To schedule a rebaselining is thought to invite cost escalations, changes to the "scope" of the contract and numerous other arguments to recoup costs which were neglected in the original proposal.

By aligning the budget with the requirements at the outset, the budgeted cost of requirement "A" should be rebaselined and traceable to the new budgeted cost of it's associated CPC. Any increases should be traceable to a specific requirement and circumstances causing the growth.

Other management accounts, reserve accounts and Level of Effort (LOE) accounts that may be outside the requirements structure should be manageable and not subject to unexplained growth as the design elements.

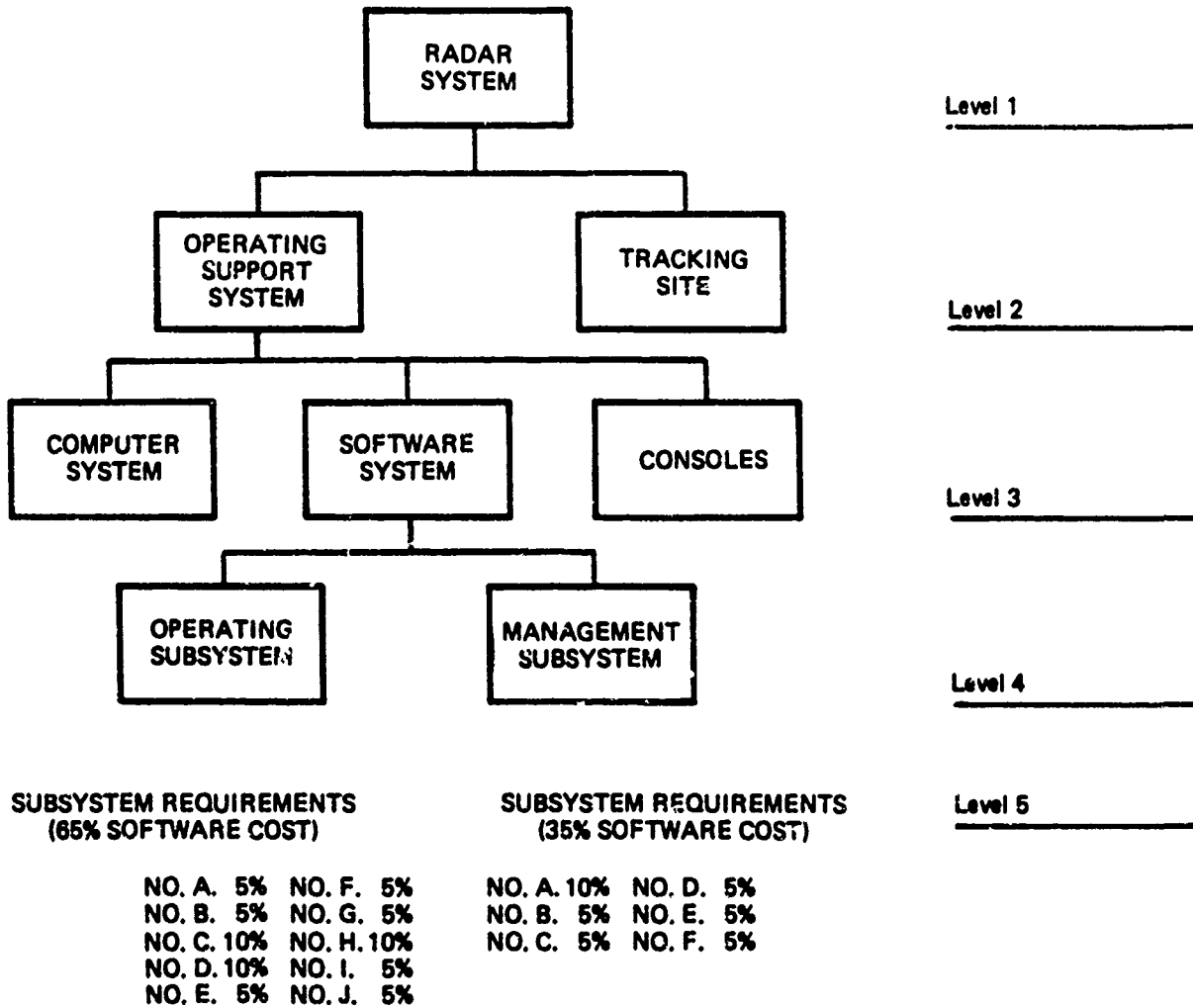
POST-PDR REVIEW

A rebaselining and a formally scheduled meeting Post-PDR meeting become major management elements in this approach. The rebaselining will be successful to the degree of emphasis placed on it by the customer. With forceful customer involvement, the rebaselining will be planned from the start and the transition to the design phase will be smooth. To further insure success, a meeting should be planned and contracted for as are the PDR or CDR. Otherwise, management attention on the PDR is quickly transferred after PDR to the design tasks. By scheduling the WBS review shortly after PDR, the management focus will continue. Planning for PDR will also take into account the effort needed to satisfy the WBS review. Questions and action items at the PDR can address the cost and schedule planning issues in preparation for a subsequent review. An approved design at PDR should be readily transferable to the WBS environment. If it is not, the issues can be resolved during the PDR thus expediting the subsequent WBS meeting. With that type of emphasis, the WBS review will go smoothly and successfully accomplish the realignment of the WBS.

SUMMARY

Management of software development is becoming recognized as being as an important part of software development as the software itself. There is little visibility for management or the customer. The more general in nature or unstructured we allow the reports to be, the less we really understand about the product. "Trust Me" becomes the by-word and management has little choice.

Planning for meaningful cost and schedule control of software will not be an easy process. It should not be assumed to conform to the same planning rules and principles used for hardware. The techniques discussed in this paper are not difficult nor are they new to program planning. They are, however, necessary elements which are all too often overlooked or ignored in software development planning. They must be instituted and used to insure adequate visibility and tracking in the software development environment.



REQUIREMENTS WBS

REQUIREMENTS/ORGANIZATIONAL INTEGRATION

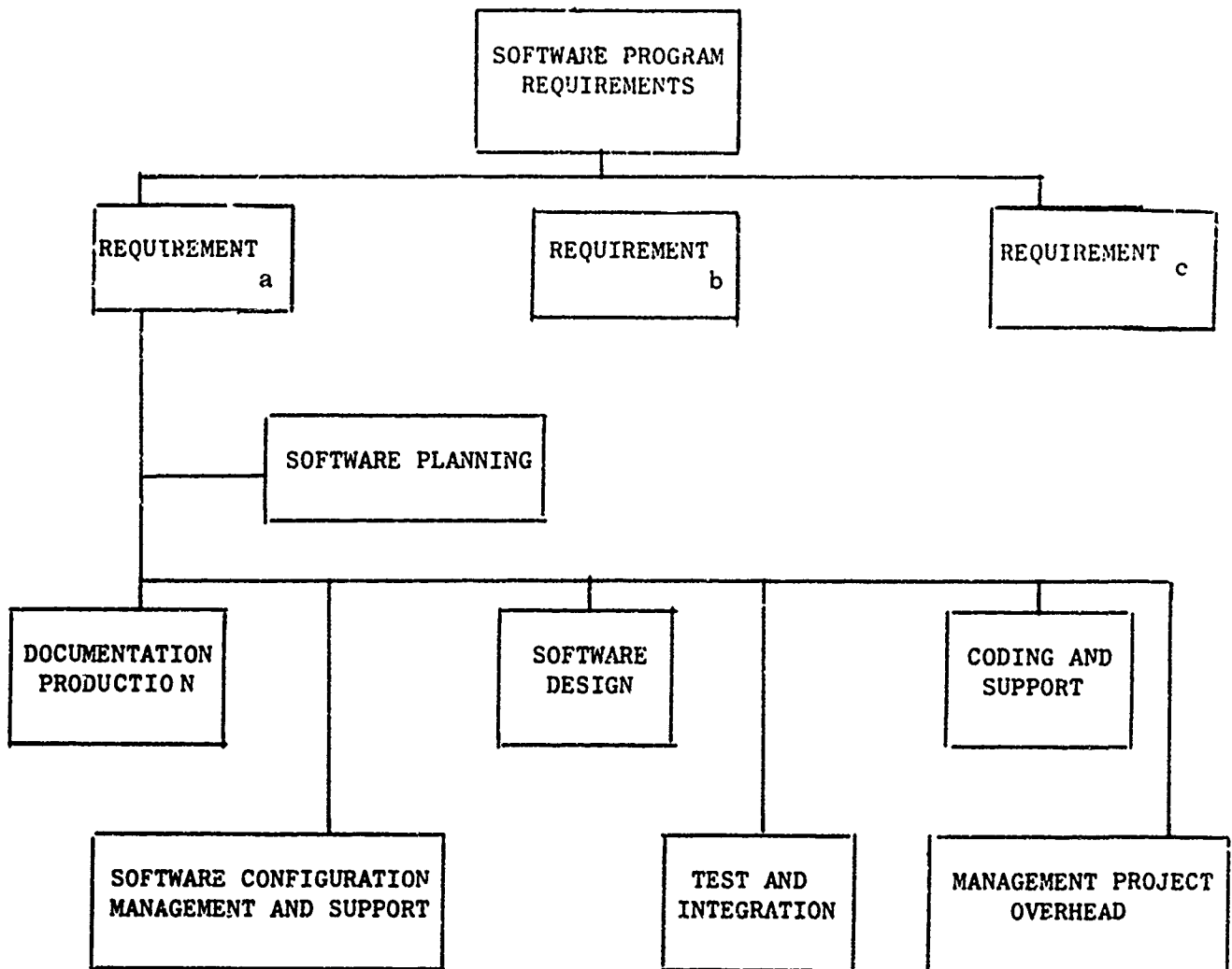
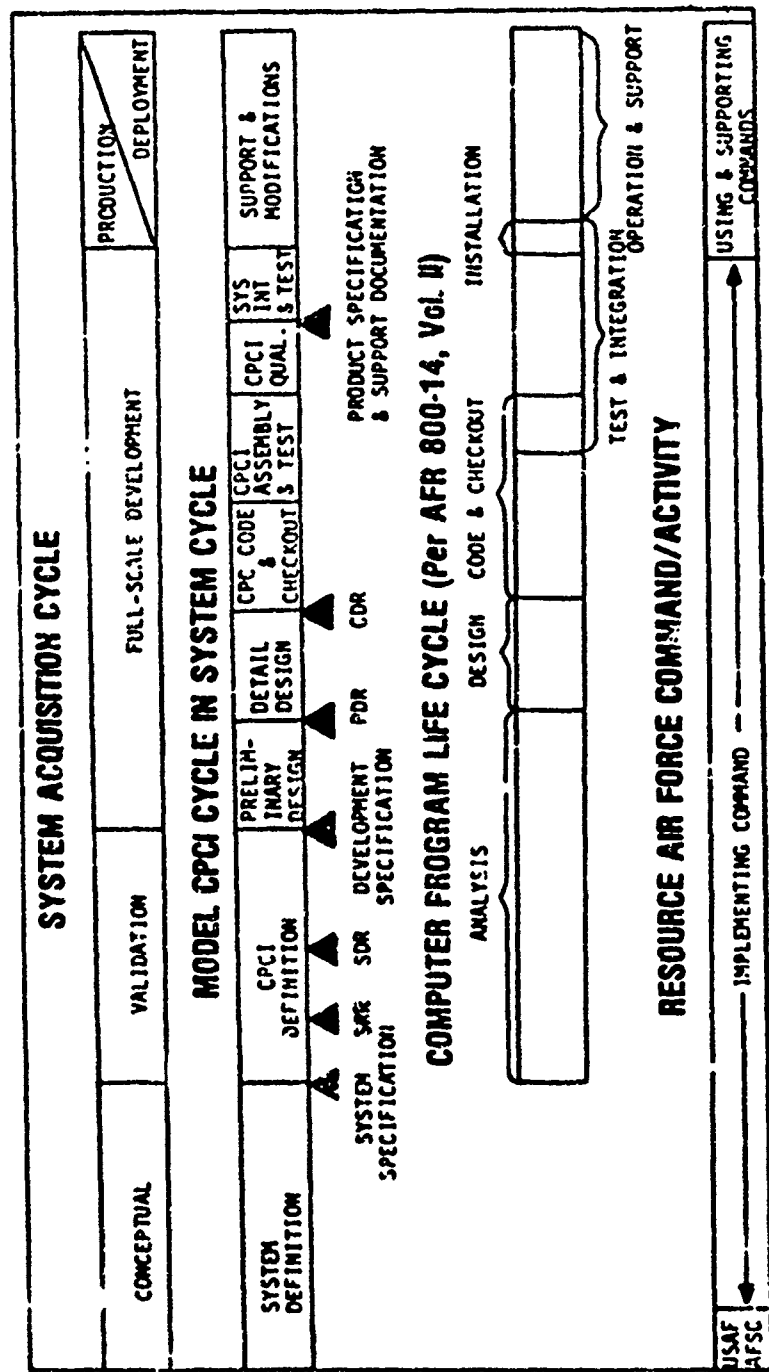
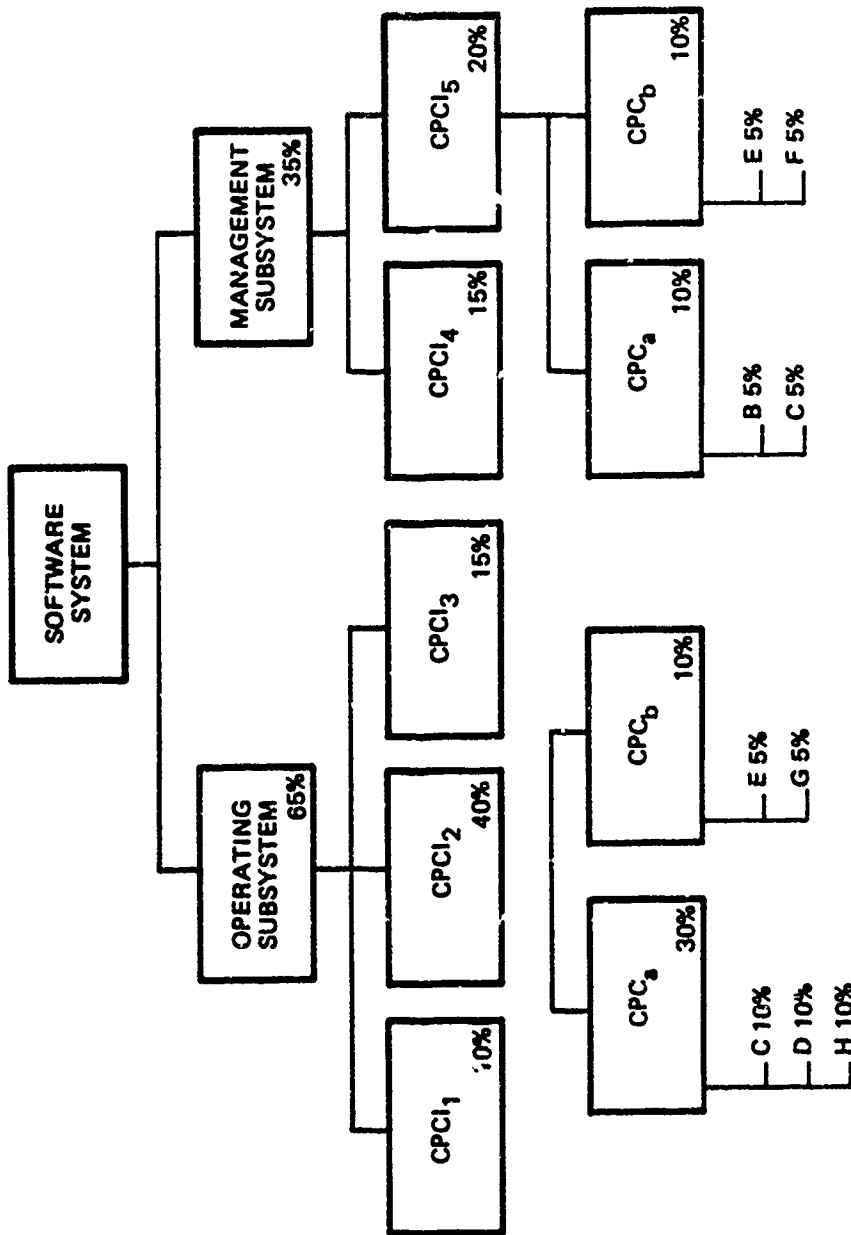


Figure #5





SOFTWARE CONFIGURATION MANAGEMENT IN A PROJECT ENVIRONMENT

Major H. Wendt, USAF and M. W. Evans

Major Howard W. Wendt, USAF is Commander, DCASPRO-Ford, Palo Alto, California. He previously held positions in NATO Logistics Plans at HQ USAF and at the AGM Joint Program Office, Crystal City, Virginia. He has attended AFITs' Education with Industry Program at Westinghouse Corporation, Maryland and graduated from DSMC's Program Management Course. He holds a B.A. Degree in Economics from the University of Nebraska and a M.S. in Computer Systems Administration from George Washington University. Major Wendt's current emphasis at Ford is specifically in the area of monitoring in-house software development. The software projects under contract are from the Air Force, Army and Navy. Extensive work has also been done by Major Wendt to develop a cross-reference library of DoD software guidance and standards for application and assistance in the software monitoring process.

Michael W. Evans is Vice President of the Integrated Computer Engineering Corporation (ICE) and is responsible for managing the services portion of the business. Mr. Evans has over 20 years experience in managing software personnel in a variety of development environments. He has worked for Univac, IBM, Litton Industries, ITT, and most recently has been manager of the Software Standards, Procedures and Control Department for Ford Aerospace's Western Development Laboratories (WDL).

At Ford Aerospace, Mr. Evans reported to the Director of Software Engineering and was responsible for the specification and implementation of all software development, management, and quality assurance standards.

Mr. Evans is a recognized expert in software project recovery techniques, software methodology, software test and integration, and configuration management and has published many articles successfully applying development techniques to a number of different project environments. He is currently under contract to John Wiley and Sons to produce a book entitled Management of the Software Development Process which will be published in early 1983.

ABSTRACT

The smooth transition of responsibility and flow of work and data within the software project is a prerequisite to project success and development productivity. Establishing a structure characterized by effective data management and control practices is the responsibility of the software manager and requires careful planning, an understanding of the specific data requirements each of the methodologies applied to the project, and definition of the data flow and reviews, audits, and milestones which support the development process. This paper describes how in a project environment data management is integrated under the software configuration discipline and how the program support library (PSL) concept may be used to centralize data control responsibility within the organization.

INTRODUCTION

Software Configuration Management is the systematic application of technical and administrative practices to insure that system and software requirements are properly identified, evaluated, controlled, and ultimately transformed into an operational software configuration.

There has been much industry emphasis in recent years concerned with describing what must be done in a project to adequately support the configuration requirement and how, in an administrative sense, the discipline should be organized and supported. Unfortunately, the software configuration management techniques, as described in the literature and implemented as part of the management of software are based on hardware and systems acquisition practices and break down when used to manage the day-by-day development of software data products.

This systems orientation causes software configuration management practices to be effective when dealing with formally baselined and controlled items but ineffective when dealing with the management and control of the many pieces of data developed or controlled by a software project during the period of implementation. Because of the management and administrative tasks associated with classical software configuration management, the overhead associated implementation of configuration management in the dynamic, tightly structured environments characteristic of the software project often preclude the effective application of the disciplines in an ongoing development project. As a result, the software project tries to force fit modified versions of the software configuration management discipline to the project and finds that, very rapidly, control of the development process is lost.

A second problem is rooted in the very nature of the software development process. As illustrated, early in the software project, the configuration management function must control a small number of requirements and data items as expressed in requirements and design specifications. The standard methods of software configuration are more than adequate.

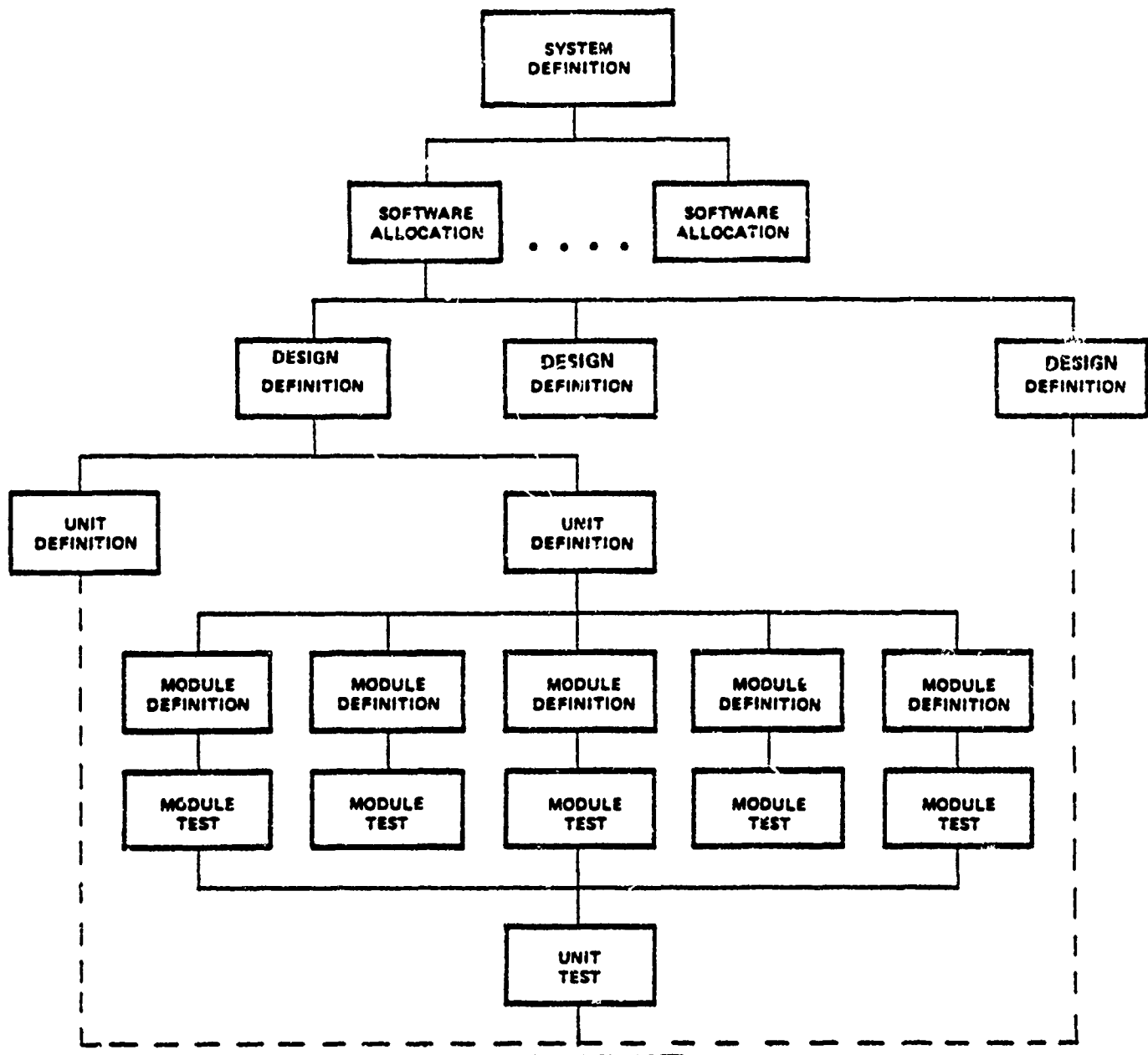
As the development proceeds and reaches the design stage, the number of data items to be controlled, although still manageable expands dramatically. When the project reaches the detailed design and implementation stage, however, the data requirements explodes and the configuration management policies and practices are unable to sustain the level of support required.

Avoidance of these problems by a software manager requires early recognition of the data management problem and the planning and structuring project data and control flow from the outset of the development.

PROJECT DATA FLOW

From the beginning of the software development the software manager should select, specify, and monitor the effectiveness of software methodologies, project standards and conventions, and reviews and audits used to evaluate the integrity of baseline and software technical data as it is being developed.

These data relationships and review requirements define and establish a logical organization and flow for the project which controls and organizes data developed or used by the project.



TESTING AND INTEGRATION
BUILD TESTING
EVALUATION OF BUILD RESULTS
PRELIMINARY & QUALIFICATION TESTING (FUNCTIONAL)
EVALUATION OF TEST RESULTS
FORMAL QUALIFICATION TESTING (FUNCTIONAL)
EVALUATION OF TEST RESULTS
FUNCTIONAL CONFIGURATION AUDIT

PROJECT DATA EXPANSION

The objective of this planning is to describe the software at any point in the development cycle, and control the many software products, including documentation, source code, and executable code, developed during the design, implementation and test of a software product.

The process of identifying the integral parts of a software system at discrete points in the implementation life cycle allows management to control changes to this system and to maintain the integrity and traceability of the system documentation.

The following sections detail the methodology for the identification of the software products, control of changes to these products, and the record-keeping activities which must take place in order to provide detailed project status information throughout the software development cycle.

SOFTWARE CONFIGURATION MANAGEMENT ORGANIZATION

The software manager has responsibility for the planning and implementation of the configuration management function. This includes execution of configuration management procedures, making necessary resources available to the staff in order that they may execute these procedures, and resolving any problems that the staff may have in interfacing to other project organizations.

Configuration management of the software subsystems should be structured to minimize project overhead and responsiveness. The configuration management function is centralized in a Program Support Library (PSL).

The PSL will maintain established functional, allocated, and interim product baselines and monitor and control the development of informal project baselines and project data products.

The PSL responsibility consists of:

1. Technical control and project monitoring of baseline content and quality
2. Organizational facilities for baselining and controlling the content of structure of software products
3. Reporting procedures and structure for standardized reporting and processing of software design or implementation issues and documentation of library contents and data products

The software products to be controlled consist of a set of documentation, source code, executable code, and status records for all major software elements.

These software elements are categorized and compartmentalized within the PSL. The PSL is organized into five logical areas:

1. User Area - all developing software and documentation are contained within this area
2. Test Area - all executable versions of the software system builds and associated documentation are contained within this area

3. Controlled Project Area - all configuration controlled source software and associated documentation are contained within this area
4. Configuration Management Area - all executable versions of the software which are contained within the controlled project areas and which are to be built into the software system
5. Documentation Area - all versions of evolving and released software documentation as well as all problem reports, action items, and discrepancy reports

These areas are logical groupings of information each having different project control and support requirements. Access to the data, control of information within the PSL and the flow of data throughout the organization is the responsibility of the PSL librarian. The data controlled or supported by the PSL may be resident on automated project files, or retained as hardcopy files, indexed and controlled through the PSL.

Two forms of control are integral to the approach and should be provided for configuration management control:

1. Informal development control and code, build and test parameters will be maintained and controlled by each project area
2. Formal configuration management files which are under project library control which contain released version of all software documentation, software, and test results. These files may be maintained manually, in the development facility on a computer system dedicated to the CM function.

WORKING PSL AREA

Access to the working area is assigned to individual programmers through the software development organization.

The working area will be allocated by the librarian to support the following:

1. Transfer files for building systems for test
2. Source files of software not released to the librarian
3. Source files of test data base structure
4. Test data, tools, and simulation facilities
5. Executable test versions of the software

Informally controlled data items will be used as work space for development of project documentation and interim specifications of system design parameters.

The structure of the PSL working areas, although the responsibility of the task leader, will track the system development structure and architecture defined for the controlled area.

This area is working space supporting those preliminary design code, and testing activities not normally reviewed or controlled through project procedures.

All working data is supported within the PSL through a standard data organization and structure. Each software subsystem will have allocated a hierarchical data organization with each level of the hierarchy, acting as a repository for those data elements defined and developed in support of the level. At the subsystem level, the PSL working area provides for storage of system functional requirements, interfaces, and software testing information unique to the subsystem.

The next hierarchical level of the working area of the PSL provides for the storage and control of user controlled implementation and test data. Within this level, the software designer stores working versions of the data used for the software systems design, inter process interface information and data dictionary information required for development, test or integration of the process.

The lowest level of the hierarchy, the unit and module level, provides for the storage and control of unit and programmer notebook information.

The working area also provides for storage of working copies of data. The area contains a copy of the controlled data base for working modification and update.

The working area provides for the storage and project dissemination of various project utility software developed during the implementation. These utilities are treated as tools, not maintainable or supportable by the project but available through the PSL for programmer application.

The final component is the build area which is an area within the PSL for storing working copies of software builds and build records for informal version testing.

The working area is the domain of the individual programmer and is not subject to project control. All items contained in the working area may be modified by the programmer responsible directly through the PSL.

Completion of the design walkthrough authorizes the start of coding and initiates transfer of the unit design to the Program Support Library (PSL). When changes and corrections are completed, the PSL librarian will review the design material in the working area (unit design, PDL and test Spec.) for format and adherence to established standards. If adequate, the librarian will copy the design material from the specified working area. The unit(s) design is now under configuration control and requires SCRIB action via the SPR to change. At this point, the module have successfully completed the coding phase and, after completion of the unit test procedures, is ready for walkthrough.

The Test Walkthrough will occur upon successful completion of unit testing. When completed, the PSL librarian will review the module code materials in the working area and the test material for proper format and incorporation of all required changes. After completion of the test walkthrough, the librarian will transfer the module code materials from the working area to the controlled area of the PSL. At this point the module has successfully completed the unit schedule milestone and the module code is under configuration control. All further changes to the module code require SCRIB action via Library elements which have been approved through project review or structured walkthroughs are transferred to the controlled area from user areas, by the project librarian and placed under configuration control.

Controlling software products for use by the project is an integration of manual, procedural, organizational, and automated techniques as follows:

1. Library Transfer. As previously described, software will undergo basic walkthroughs: Design walkthrough, Code Review and Test walkthrough which authorize and transfer to the controlled PSL area. These reviews will serve to confirm adherence to requirements, validate software design and to reduce software integration time through identification of design and coding errors early in the development process. These reviews are tied to the software schedule milestones and successful completion of the reviews is required to accomplish schedule milestones and authorize transfer of data from the user area of the PSL to the controlled area.
2. Changes to Controlled Software. The second method of transferring software from informal to controlled status is through Software Configuration Review Board (SCRB) action resulting from processing of approved SPRs. The SCRБ is the means by which the PSL will evaluate all processing/disposition of Software Problem Reports. This board will review and evaluate all Software Reports, authorize appropriate corrective action to be taken, track status of all changes in progress and close the Software Performance Reports after the changes have been made. If it is determined that the recommended change affects approved specification, cost, or schedule, it will be forwarded to the program for resolution. When the SCRБ asks a member(s) of the software development organization to implement an approved change, the member(s) obtains the correct version of the affected software and the related documentation from the controlled Program Support Library. The software and/or documentation will be moved from the controlled area library to the user area. The software development member(s) will do all of their work in the user area until they have completed their work and the software and/or documentation is ready for release to the controlled release library for SCRБ approval. When an SPR is reported closed, the SCRБ examines the correction package submitted for adequacy and completeness. Upon approval of this package, the SCRБ notifies the librarian to accept the changed software, if necessary complete a Version Description Document (as outlined below), distribute a necessary materials to all affected sites, log the SPR as closed, and incorporate the changes into the software library.

When data has been approved for transfer either through SCRБ action or project reviews or walkthroughs the librarian will transfer the information from the user area to the hold segment of the configuration management area. While the data is resident in the area the librarian will review it for correspondence in both form and content to the requirements of the data transfer, will assemble and review all the supporting documentation required by the transfer, and process all SPRs closed by the transfer. The librarian will then generate either a new or interim system release of required from data included in the transfer and will document the release in a Version Description Document (VDD).

When processed the data is transferred from the user area to controlled test area of the PSL where it serves as the new software baseline.

CONTROLLED AREA OF PSL

The controlled area of the PSL is the project center of formal software configuration management. The formal configuration management process supports four functions:

1. Configuration Identification. Identification and baseline management of the functional, allocated, and product baselines throughout the development, acceptance, installation, and operational phases of the program.
2. Configuration Control. Control of changes to baselined and non-baselined software products, performance and functional requirements, test products and parameters, drivers, and results, and internal and external interface formats and operational requirements.
3. Configuration Status Accounting: Control and monitoring facilities for accessing development and operational status of all software.
4. Audits and Reviews. Provide a single data source for supporting internal and customer reviews.

The controlled area of the PSL is the repository for three categories of software information.

1. Requirements and Functional Allocation - this segment of the PSL are the agreed to baselines which serves as the basis for the software development
2. Design and Implementation Information - this PSL segment contains controlled software design and coding data which has been released to the PSL through completion of specific software development milestones
3. Software Test and Support Facilities - controlled software test data, tools and supporting utility software used by more than one programmer in developing testing or integrating software

The controlled PSL area is organized in a hierarchical structure which corresponds to development requirements and phasing of the software production. The controlled structure shadows the organization of the working area facilitating the smooth flow of data as it is transferred under project control. Under this structure, all software data items developed through or controlled by the project have an identifier which uniquely identifies each data item included as part of the software inventory and also documents the relationship of each data item to the overall architecture and organization of the project.

The PSL integrates identification of the data elements with the development functions. Definition of the system requirements establishes a structured, traceable, definition of system requirements allocated to specific software

subsystems controlled by the PSL. Designers of the system define system flow, processing, and test requirements linked to the system requirements. This definition is controlled through the PSL. The software system design translates the system design requirements into specific software structures, data and test requirements linked to the user requirements. The outputs of this phase document the specific software design and test requirements filling the process and unit segments of the controlled PSL. Through the PSL library structure this definition is linked to the system design and requirements levels of the hierarchy. It is during the phase that the detailed software design and testing structure is placed under formal configuration management.

The final development stage, implementation and integration results in PSL production and modification of software systems to support integration activities. This data is provided to the project through the PSL and documented in Version Description Documents (VDDs) which document the specific contents of each software release as well as the status of system and software discrepancies.

This data is extracted from the controlled areas of the PSL and is closely integrated with the configuration control activities.

Controlled builds will be generated through the PSL to support software systems testing. These builds will utilize the automated facilities of the CM support facility automatically selecting the latest version (or selected versions) of each module required in the build from the controlled area. Modules not available in the controlled area will be flagged and the librarian has the option of selecting user replacements. The tools will then automatically initiate compilation and linking of an executable object system storing the system in the test area.

All library builds will be documented through version description document (VDD). The VDD is automatically generated through the PSL and the VDD will document specific requirements for system builds, each module by release level which is included in the system release, all SPRs closed by this release and discrepancies which remain open.

Formal software builds will be generated at regular, preplanned intervals which track either the build testing schedule or the system release schedule. These software versions will be assigned unique version numbers and the contents will be formally maintained as the current supported software system. The PSL will be responsible for assuring correctness of the software documentation and correspondence between documentation.

In addition, the PSL will release interim versions of the system software which are based on the current controlled version. These releases are documented through VDD procedures, however, the releases are considered informal and the same level of documentation reviews is not applied as required of a major release.

CONFIGURATION MANAGEMENT AREA OF THE PSL

The configuration management area of the PSL, is the working area accessible by the PSL personnel. The function of this segment is to allow storage of configuration management data essential for PSL functioning, storage and control of user supplied data which describes documentation production requirements or system data configuration, and a holding area for transitioning software project information from the user area of the PSL to PSL control.

The only level of control for information contained in this PSL segment is the PSL librarian. PSL supported configuration management tools and data are stored in this area and librarian may approve modification without additional review or approval.

DOCUMENTATION

The documentation area is the segment of the PSL used to develop and store released versions of software documentation and baseline data items which support the development of software. This PSL segment is an extension of the controlled area requiring the same level of configuration management and control.

The production of project documentation is an integration of documentation production requirements with the development methodology and PSL support requirements. This integration has several key components:

- a. An early definition of documentation content format requirements
- b. Assignment of a centralized organizational structure responsible for document development and production
- c. Application of PSL control techniques, procedures, and support capabilities towards the levels

The production of software documentation is coupled with the development of the software product.

When the project structure is defined in the planning phase of the project the documentation skeletons are defined and stored in the configuration management area of the PSL skeleton.

The specific outline for each document to be developed and the section by section content requirements of each. This document summarization is rigorously controlled by the PSL librarian. This serves as the basis for the development, production, and evaluation of all format project documents. As well as identifying data control and format requirements to be collected as the development proceeds.

As data is developed and transferred into the controlled area of the PSL and is incorporated into a prespecified PSL architecture and structure. This structure is characterized by assignment of a unique identifier for each data component, controlled through formalized procedures.

This data is not developed specifically for production of each document but is, developed as part of the implementation process. The PSL collects this information from the controlled PSL area, formats the data in accordance with the prespecified document formats, and flags data not available in the PSL.

This process is facilitated through the documentation methodology and through the structure and support facilities of the PSL. These areas collect data contained within controlled PSL version of the Unit Development Folders, Programmers Notebook, Test Data, Data Dictionary, and Interface Information and other required supporting data compiled and for what the data in accordance with the requirements of the document skeleton in the configuration management area of the PSL and output documentation consistent in item project requirements.

Production responsibility for all documentation is centralized in a single entity, the documentation coordinator. This organizational entity works in parallel with development and test and integration activities directly interfacing with these organizations only when the raw data required to produce the document is not available with the controlled PSL or when the data in the library is inaccurate, inadequate, or not in sufficient detail to support the project.

SUMMARY

The software configuration management approach described in this paper is a unique application of automated and manual techniques to the problem of data management in a project environment. The various segments of the library may be stored on a computer system supported through an integrated set of tools and techniques tailored to the project.

The techniques are highly responsive to the needs of the project, provide clear traceability as the requirements, design, and implementation data move through the library structure.

Implementation of this technique off-loads much of the administrative burden from the technical staff freeing them to perform the tasks required on the project.

The only prerequisite to establishing an orderly structure for data control is a management commitment to plan and implement a structure in spite of project pressures and personnel resistance.

Effective data management is a fundamental attribute of productive software development.

REFERENCES

1. Boehm, Barry. "Trends in Software", IEEE Transaction on Computers, Vol C-25 (12), December 1976.
2. Evans, Michael, Dolkas, James. "Improving Productivity through Acceptance of Software Methodology with the Project Organization", IKD Congress, Berlin, October 1980.
3. Aron, Joel D. "The Program Development Process.", Addison-Wesley, 1974.
4. Metzger, Harold. "Managing the Software Development Project" Hempstead Press, 1975.
5. Evans, Michael, and Dolkas James. "Defining Tools and Methodology for the Development Environment," Computer and Government Twenty-first Annual Symposium, ACM, June, 1982.
6. Evans, Michael, Piazza, Pam, Sonnenblick, Paul, "Large Scale Software Development Management Techniques," NAECON, 1982 Dayton, Ohio.
7. Evans, Michael, Piazza, Pam, Sonnenblick, Paul, "How to Salvage A Faltering Software Project," Proceedings of 26th Annual Meeting of Society for General Systems Research with American Association for Advancement of Science, January 5, 1982, Washington.



Abstract

REVISING MIL-STD-1679

MIL-STD-1679 (NAVY) contains requirements for the design and development of software which are applicable in Government contracts. Initial release of the Standard was made in December 1978. Revision of the standard has become necessary for two reasons. First, many comments and suggested changes have been received citing areas where improvements or clarifications could be made. Also, since the promulgation of MIL-STD-1679, DoD and Navy policy related to embedded computer resources and software development has undergone significant change. This makes issuance of the "A" revision a necessity for the Naval Material Command. This talk will deal with the problems of revising MIL-STD-1679 and will directly address the issue of an apparent conflict with the planned release of the Joint Logistic Commanders tri-service standard for military system software development (MIL-STD-SDS).

Biography

William J. Egan
Tactical Embedded Computer Program Office
Naval Material Command
Washington, D. C. 20360

BSEE	Northrop Institute of Technology	1963
MS	University of Southern California	1979
Graduate,	Naval War College	1980

Experience

25 years of both U. S. Navy and private sector experience including:
U. S. Navy Active Duty - Fire Control Technician
North American Rockwell - B-70 Project
General Electric - Supersonic Transport Engine Project
General Dynamics - Standard ARM Missile Project
Pacific Missile Test Center - Software Support of F-14 & EA-6B

Current Assignment: Tactical Embedded Computer Program Office, Headquarters,
Naval Material Command

AD-P003 593

APSE DATABASE USER SCENARIO

Elizabeth S. Kean

Rome Air Development Center

RADC/COES

Griffiss AFB NY 13441

(315) 330-4325

→ The Database facilities provided in the Ada (1) Integrated Environment (AIE) are an integral part of the KAPSE/MAPSE design described in the STONEMAN document. Project management and configuration control facilities are essential elements in the design. They are provided through primitives in the database management system incorporated in the AIE KAPSE/Database. Large and small scale system development can be easily managed by manipulating the primitives to meet user requirements. The users may range from Project Managers, software designers, programmers, and quality assurance personnel to administrative support personnel. These users have very different functional requirements and make various demands on the database that will require a versatile and efficient database system. This paper describes the users interfaces and capabilities provided to AIE users during the development cycle of a software system. ←

The Ada Integrated Environment is designed for all types of software development, the most important being the development of embedded computer software. During the design, development, and maintenance of a software system, it is anticipated that a variety of users will be accessing the AIE KAPSE/Database. A typical system development team will contain relatively the same categories of personnel. The following scenario will demonstrate the AIE database facilities for project management and configuration control for a software development effort. There is a software requirement for cross compiler facilities for a target computer system. The Project Manager for the effort has been assigned the responsibility to develop a cross compiler, run-time system, and debugging system for the target computer. The project configuration of the database will depend on how he plans to manage the development.

(1) Ada is a trademark of the Department of Defense

The Project Manager chooses to set up the project with three team leaders who will report directly to him. Team number 1 is responsible for the cross compiler and linking/loading functions. Team number 2 is responsible for the debugging and run-time system functions. Team number 3 will provide quality assurance over the development. The quality assurance functions include unit testing and integration testing. Each of these teams may be broken into subgroups which report to the team leaders. Figure 1 describes the hierarchy of the development team structure for this scenario.

The Project Manager will oversee the entire development. He is responsible for the successful completion of the project. He will need to ensure that each member of the team has the required resources (computer time, manpower, etc) to perform the effort. Each team has a different set of functions to perform, however, communication between the members of the various teams and team members is essential during the development cycle. For this effort, the Project Manager sets up rules and standards that the teams must follow. For example, the design documentation is initially produced and submitted to the QA personnel for evaluation. The documentation is then passed to the coders, who produce the software according to the specifications. There is a continuous flow of communication between the designers and coders until the code and documentation match and perform the required function. Upon completion of the coding phase, the software is passed to the unit testers who perform code audits and validation/verification tests. There is a continuous communication flow between the QA testers and the coders until the software is determined to be valid. The Project Manager must be kept informed of major trouble reports and overall project progress. The team leaders must be aware of all trouble reports for their team and provide the necessary information to the Project Manager. Figure 2 represents the development cycle for the project.

The Project Manager is considered a privileged user on the AIE. He is allowed to use many of the database primitives that the ordinary user (i.e., programmer, designer, tester, etc) is not. The KAPSE provides the primitive TRANSFER_BUDGETS which allows the Project Manager to allocate disk block and processing unit budgets. Associated with each object is the current number of blocks that make up the object and associated with each user is the total number of CPU seconds used. If either of these budgets are exceeded, access to or processing in the object is denied. The Project Manager will be able to maintain strict control over CPU and disk block usage during project development.

Once the Project Manager decides on a configuration, he may have an employee (Program Librarian) set up the configuration on the AIE. The Librarian can set up the configuration in the following manner. Since the overall structure of the AIE database is hierarchically relational, many primitives and

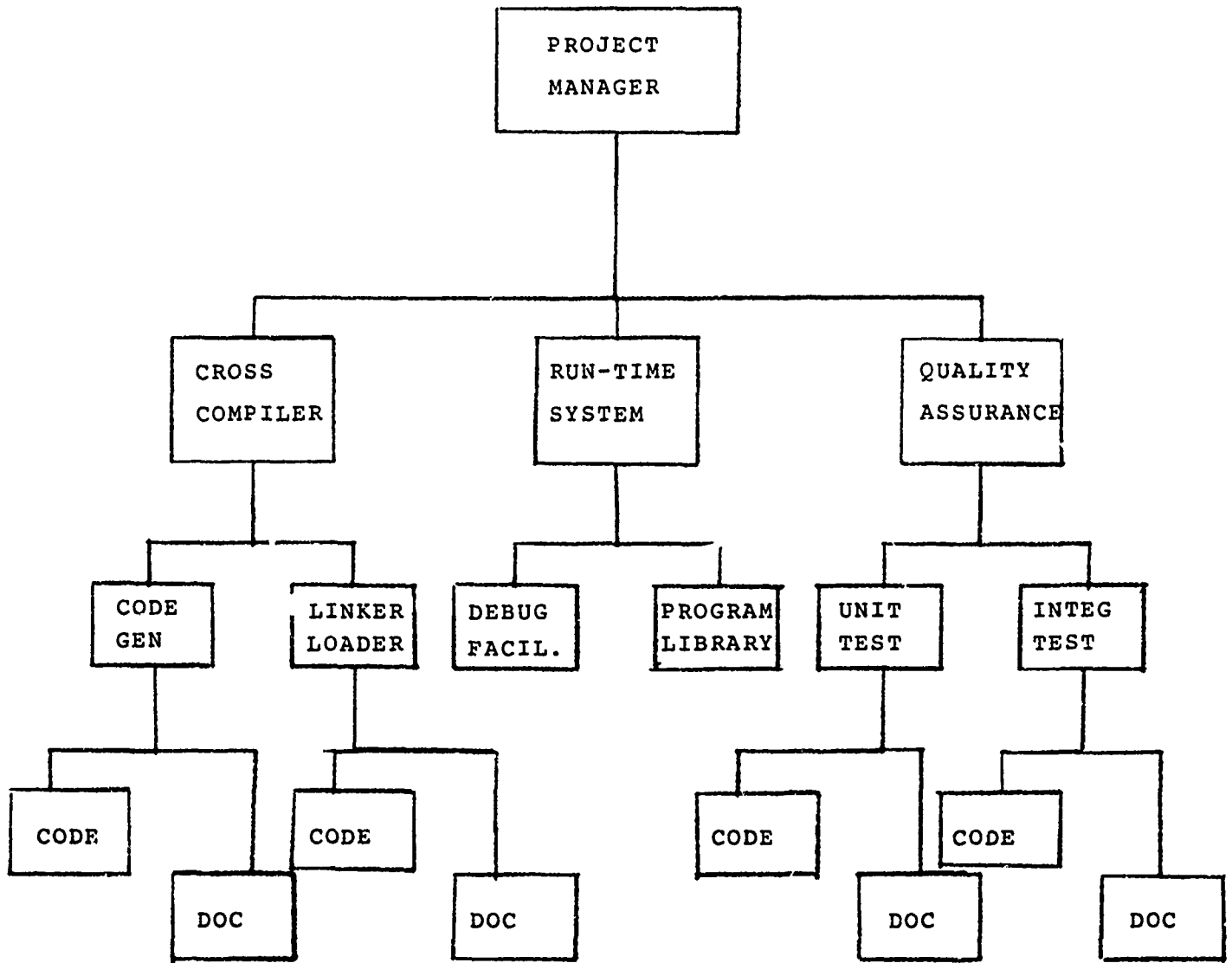


FIGURE 1
PROJECT HIERARCHY

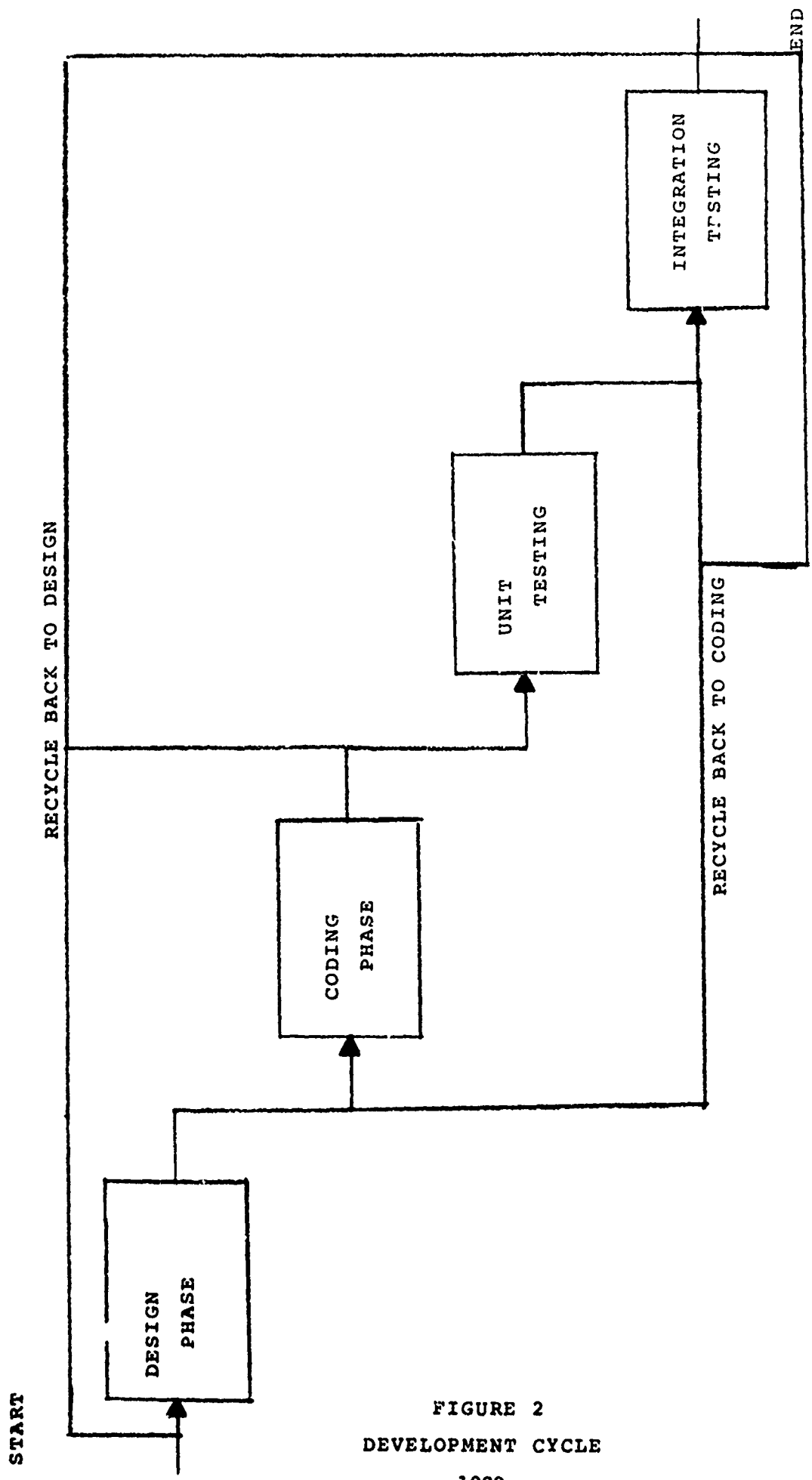


FIGURE 2
DEVELOPMENT CYCLE

operations are available for the Librarian's use. The database contains a collection of objects that have attributes and content. The attributes of an object distinguish it from other objects in the database. There are three classes of objects: simple, composite and window. The content of a simple object is an Ada external file. The content of a composite object is a collection of component objects which may be simple, composite or window objects. The content of a window object is a cross reference to a partition of another object in the database. It is the mechanism through which access to and responsibility for specific parts of the database is permitted.

The attributes of an object are the most important means of partitioning and building the database. They describe the purpose, content, and access of an object. In the AIE database, there are system and user defined attributes. The system defined attributes are category, access control, and history attributes as defined in the STONEMAN document. There are two kinds of user defined attributes, distinguishing and non-distinguishing. The user defined distinguishing attributes are the mechanism for the distinction between objects in the database. They are later used to select the various components from the composite object.

The Librarian in the example project will be able to partition the database by setting up attributes specific to his needs. Then, for the various teams, he can specify the access to these objects. The database primitive for creating a composite object is as follows:

```
CREATE_COMPOSITE("TARGET_COMPILER", COMPONENT_DA=>
                "FUNCTION_AREA_MOD");
```

The FUNCTIONS in this project are CROSS_COMPILER, RUN_TIME, and QA. Each function has its own AREA and MOD (module). The Librarian can create the components as follows:

- (1) (FUNCTION=>CROSS_COMPILER, AREA=>CODE_GEN, MOD=>CODE)
- (2) (FUNCTION=>CROSS_COMPILER, AREA=>CODE_GEN, MOD=>DOC)
- (3) (FUNCTION=>RUN_TIME, AREA=>DEBUGGER, MOD=>CODE)
- (4) (FUNCTION=>CROSS_COMPILER, AREA=>LOADER, MOD=>CODE)
- (5) (FUNCTION=>QA, AREA=>UNIT_TEST, MOD=>CODE)

The positions of the above list of components is immaterial, since the attribute name is mentioned. Another way of describing the above data is:

- (1) CROSS_COMPILER.CODE_GEN.CODE
- (2) CROSS_COMPILER.CODE_.DOCUMENTATION
- (3) RUN_TIME.DEBUGGER.CODE
- (4) CROSS_COMPILER.LOADER.CODE
- (5) QA.UNIT_TEST.CODE

The Project Manager can easily access the above information in various ways. Since the database is relational,

(FUNCTION=>CROSS_COMPILER)

would include (1), (2) and (4) above, whereas,

(MOD=>CODE)

would include (1), (3), (4) and (5).

The Project Manager will want to keep track of the partition just created. By using the database primitive, GET_PARTITION_INFO, he can ascertain the number of components in the partition and use LIST_PARTITION to print out the attributes of all the components within the given partition. The primitive LIST_PARTITION is a combination of other database primitives such as OPEN_PARTITION, GET_NEXT_COMPONENT, and GET_ALL_ATTRIBUTES and can print out a configuration, some subset of components, or a simple database object. For example, the following is an example that sets up attributes and attribute values for the three files, TEST1, TEST2, and TEST3.

- (1) SET_ATTRIBUTE ("TEST1", "FUNCTION", "CROSS_COMPILER");
- (2) SET_ATTRIBUTE ("TEST1", "AREA", "COGE_GEN");
- (3) SET_ATTRIBUTE ("TEST1", "MOD", "CODE");
- (4) SET_ATTRIBUTE ("TEST2", "AREA", "DEBUGGER");
- (5) SET_ATTRIBUTE ("TEST2", "MOD", "CODE");
- (6) SET_ATTRIBUTE ("TEST3", "MOD", "DOCUMENTATION");

The following primitive print out partitions:

- (1) LIST_PARTITION ("MOD => CODE", "AREA");


```
(2) LIST_PARTITION ();
```

The first instruction will list the partition in the following manner:

Partition (MOD => CODE)	Attributes AREA
TEST1	AREA => CODE GEN
TEST2	AREA => DEBUGGER

The second instruction defaults to the partition .CURRENT_DATA and will list all the partitions contained in it.

Every database object has an access control attribute. The various classes have different types of access rights. For example, a simple object can have read, write, append, and consume access rights while a window object has only GO THROUGH access rights. Access control is handled through the window object. As mentioned before, the window is a cross reference to another object. In other words, this cross reference is a path from the window to the required object through a node called a common ancestor composite object. In order to easily trace a window, there is a window cross reference table located at every node which is used as a common ancestor. The Project Manager and/or team leaders will have to set up the access control using the database window operations. The AIE database facilities provide operations and primitives that allow the Project Manager and team leaders to maintain the level of control they wish over the users of their configurations. When new windows are created for the users, the access rights will be a subset of the access rights the creator possesses. Some operations that can be used are:

- (1) CREATE_WINDOW("CROSS_CODE_GEN", "CROSS_COMPILER.CODE_GEN");
- (2) CREATE_WINDOW("QAVIEW", "RUN_TIME.DEBUGGER.CODE", CAPACITY=>"QA_PERSON");
- (3) OPEN(FILENAME, ".QAVIEW.PROG1");

The first operation is the creation of a window with the same access rights, only the name is shorter and easier to use. The second operation is the creation of a window limited to access rights given to QA personnel. The third operation opens the file PROG1, unless, .RUN_TIME.DEBUGGER.CODE.PROG1 does not permit read/write access to QA_PERSON. The team leaders may not want the coders to be allowed to modify the source code when the code has reached unit testing. Therefore, the primitive CREATE_WINDOW

can be used to deallocate access to the coding team during unit testing.

In order to maintain project and configuration management, the Project Manager and team leaders must be kept up to date on the progress of the development. The software programmers and designers will report their progress to the team leaders, who in turn report the progress to the Project Manager. The AIE provides an inter-user mail system which can be an efficient medium for reporting the problems/progress of the development as well as provide communication between the members of the groups.

The team leaders will have to organize the configuration of their respective teams. They will need to partition the database such that each member of the group has his/her own directory or working space. They will require access to each working space in order to manage the development of his function. The database primitives for the window and access control attribute satisfy the team leader's needs. Each group will require a different type of database access and have to interact with each other in the performance of their duties. The designers will be producing specifications of the new software system. They may develop new tools to automatically produce the specifications and therefore the KAPSE/Database must provide the primitives to ensure extensibility. The programmers will be editing, compiling, debugging, testing and executing Ada source programs. They will make the majority of the queries on the database.

These users must interact with each other and the Ada Integrated Environment. They must share certain portions of the database. The database facilities must provide the necessary primitives to satisfy each user and be time/space efficient since an AIE may not be the sole host on a computer system. The specification designers and software programmers must communicate with each other (if they are different people) in order to ensure correlation between documentation and source code. The programmers working in the same group, as well as different groups, will require communication links since many of the subroutines will contain interdependencies. Another aspect of the problem is that the documentation and source code will be stored in different areas of the database, but must be logically connected together. The AIE database system must provide the communication links between the programs and documentation.

The software programmers/designers are the most important users of the system. Since software development is the purpose of the AIE, primitives to aid the programmer/designer are an integral aspect. Programmers work on a critical time path. They require an efficient system for development. First and foremost, they require a good compiler. If the compiler is not adequate (i.e., too slow, produces inefficient code, etc) then the rest of the system reflects this. Secondly, the AIE must provide sufficient backup and recovery routines to ensure integrity of

the software produced during the effort. If the programmer is required to continually fight the system, deadlines will be missed and the accomplishments will be few. Assuming there is a good (validated) Ada compiler on the AIE, the programmer will use many database primitives during the coding phase of the development. The AIE KAPSE/database provides the facilities for program development in an efficient manner.

Many of the programmers will depend on the subroutines developed by other programmers. Therefore, window and access control attributes will play an important role in the software programmer's configuration. In this scenario, each programmer is responsible for one subroutine of the entire software development. Figure 3 is a diagram describing the interdependencies of three subroutines designed and developed by different programmers. Programmer B uses the subroutine developed by programmer A. Therefore programmer B will need to see the specification portion of programmer A's subroutine. Programmer A depends on programmer C's subroutine and therefore needs to see that specification portion. The links between these programmers will be provided through the primitives SET ATTRIBUTE and SET_CAPACITY_ACCESS, which set the access rights of the programmer (read, copy, execute, etc.) for the specified file.

The programmers will be continually accessing the database during a typical computer session. Compilation, editing, debugging, linking, and executing source programs require many database queries that will be transparent to the ordinary user. Upon logging onto the AIE, Programmer A will have a window on the current partition called .CURRENT_DATA which is the default name set up for .CROSS_COMPILER.CODE_GEN.PROG A. The text file OPT TECH is a component of this partition. It contains the source code for machine dependent optimizations to be performed in the back_end of the target compiler. During this session, Programmer A will make a revision to the source code file, recompile and execute the program. When editing this file, access to the MAPSE editor and text file are checked. The file is then opened, read, written, and finally closed. These database operations are performed through the execution of a command language script that is an Ada-like subprogram call.

The compilation of the Ada program requires many file openings and closings, access checking, and database input and output operations. The complexity of the Ada language necessitate an efficient data management system. Separate compilation results in the symbol table for a program to be located in different areas of the database. The database must provide the facilities to retrieve the information from the symbol table in a manner such that the various portions of the symbol table will not require unnecessary paging in and out of memory. Since the database is relational, paths are easily cross referenced. The hierarchical method required much tree searching and therefore processing time. The AIE database can simply

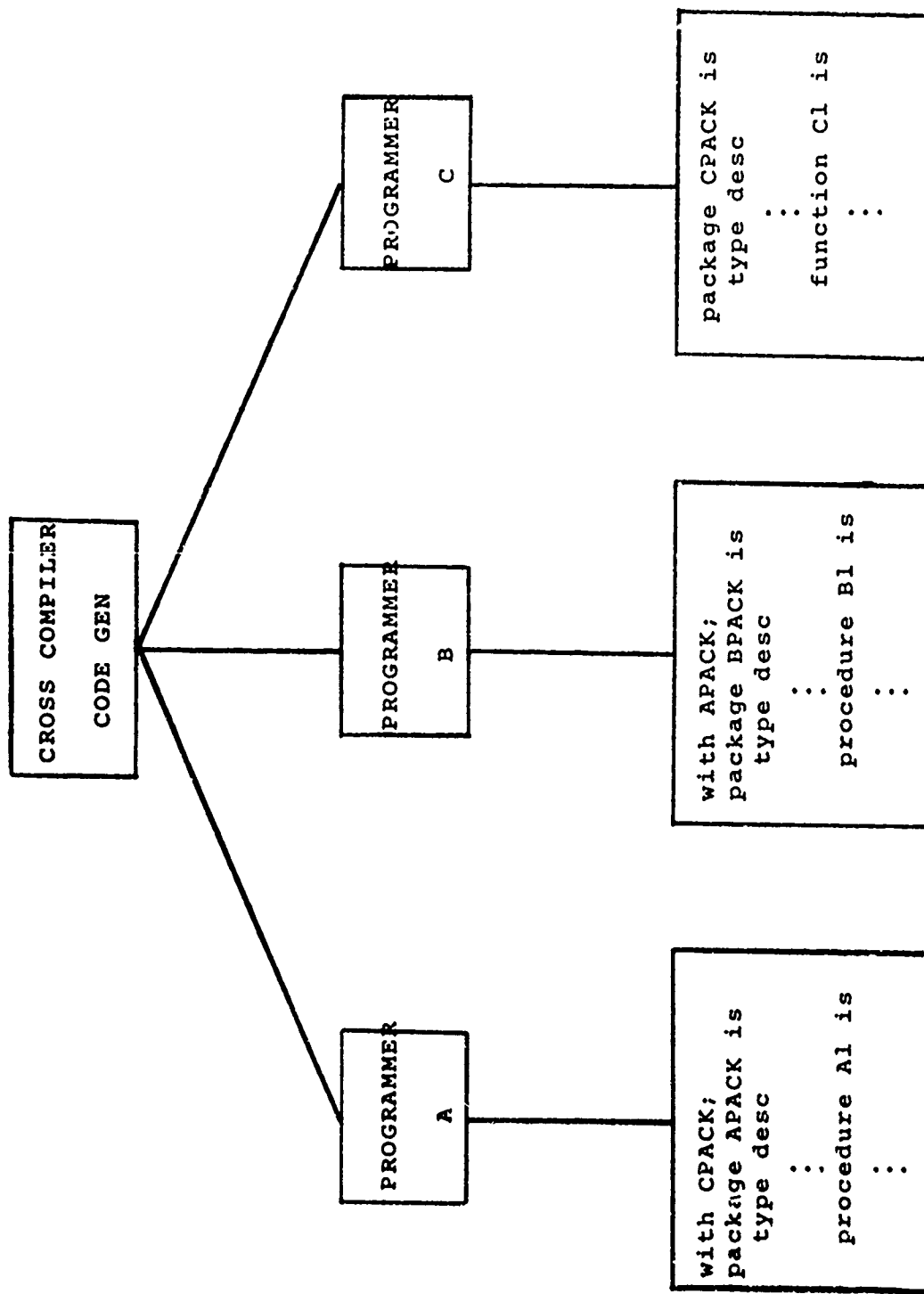


FIGURE 3
PROGRAMMER INTERDEPENDENCIES

create paths to objects in different areas of the database using window objects and attributes. However, this method may require an excessive amount of space to be entirely feasible. Since, the AIE database has an inherent hierarchical structure, it may be used in the traditional manner or a combination of both and therefore provide the best of both worlds.

Quality Assurance (QA) personnel will require access to the development team's specifications, source and executable code. They independently evaluate and test the progress and quality of the system under development. They require access to the programmer's code upon completion of the coding phase. Before completion they do not have access and the programmers should not be able to modify any coding files during unit testing. The KAPSE/Database must provide the necessary facilities to satisfy the QA needs and also protect the programmer/QA tester from unnecessary intervention. The AIE database satisfies these requirements through the access control attribute. The primitive to set the access rights for read and execute only would be:

```
SET_CAPACITY_ACCESS("CODE GEN 1750", CAPACITY=>"QA PERSON",  
    ACCESS_RTS=>"READ,EXECUTE");
```

In order to protect the QA tester from programmer intervention, an attribute UNIT TEST can be set such that when the attribute value is true, the programmer no longer has access to the source code. If the code is passed back to the programmer, then Unit TEST=>FALSE and the QA tester no longer has access to the file. In testing the produced code, the QA user will be using the same database operations as the programmer.

The above scenario represents one method for configuring the database for a project development team. Obviously, different project managers will configure their development teams in various ways, however, the AIE KAPSE/Database contains the necessary facilities to promote good project and configuration management for large and small scale software development. The database primitives are sufficiently flexible to satisfy the requirements of the project managers as well as the classes of users involved in the design and development of software systems.

REFERENCES

1. Computer Program Development Specification for Ada Integrated Environment: KAPSE/Database Type B-5, IR-678-1, Intermetrics, Inc., 30 June 1982.
2. Computer Program Development Specification for Ada Integrated Environment: MAPSE Command Processor Type B-5, IR-679, Intermetrics, Inc., 13 March 1981.
3. Requirements for Ada Programming Support Environments, "STONEMAN", Department of Defense, February 1980.

BIOGRAPHY

Elizabeth S. Kean

Elizabeth Kean joined RADC in June 1979 as a Computer Scientist. She works in the Software Sciences Section of the Computer and Software Engineering Branch. As a member of the High Order Language Group she performs research and development in technology related to the Ada programming language and Ada language programming support environments. She has a Bachelor's degree in Mathematics from Utica College and is pursuing studies to obtain a Masters degree in Computer Engineering.

AD-P003 594

ARCHITECTURAL AND CONTROL CONSIDERATIONS FOR A HIGH SPEED
SIGNAL PROCESSOR IMPLEMENTED WITH AN Ada EXECUTIVE

Steven E. Adams
Intermetrics, Inc.
5162 Springfield Pike
Dayton, OH 45377
(513) 258-1271

Thomas R. Butler
Magnavox Co.
1313 Production Road
Fort Wayne, IN 46808
(219) 429-5957

ABSTRACT. A common assumption in the design of digital signal processors is that the critical system resource is the raw multiplication rate of the Arithmetic Element (AE). Currently, the architecture of these processors is based upon a distributed control network with the number of AEs required to meet the computational load of the application. Simulation of this type of system has shown that the actual bottleneck is the depth and complexity of the control network. This paper examines a signal processor architecture and an Ada executive control structure which supports a dataflow language. This architecture and control structure have been tested in a component level simulation.

1.0 INTRODUCTION

In April of 1981 the Department of the Navy released a Request for Proposal for the design and implementation of the Enhanced Modular Signal Processor (EMSP). This processor is required to handle sonar data rates of up to 800 megabits per second (approximately 40 million multiplies per second), and fit one six-foot water-cooled cabinet. An architecture was developed by Magnavox and an executive was developed by Intermetrics to meet these requirements. To evaluate both the hardware and executive designs, a simulator was built which models all functional elements of the hardware and which incorporates an executable version of the executive. The sections below describe the dataflow programming environment, the derived hardware, and the Ada implementation of the scheduler portion of the executive.

2.0 DATAFLOW PROGRAMMING

The programming methodology designated for use in the final EMSP architecture was the "EMSP Common Operational Software Methodology" (ECOS) as described in Appendix C of the EMSP RFP [1]. The stated purpose of ECOS is to provide the EMSP applications programmer with a convenient method for

specifying signal processing graphs. This representation is intended to allow the programmer to directly address signal processing problems without explicit knowledge of the underlying hardware peculiarities. This section presents an analysis of ECOS, provides a brief summary of a revised ECOS and an Ada structure to hold the graph characterization.

2.1 ECOS ANALYSIS

One obvious area for critical examination in ECOS is repetition parameters. First, it must be noted that repetition parameters serve two different purposes. At pre-runtime they are macros which allow multiple specification of certain graph components, such as nodes or queues. The ECOS translator expands the definition of the graph and the repetition parameters "disappear". At runtime, the repetition parameter is used to indicate which of some number of graph components are to be used in a particular instantiation. The two uses are distinct and must be examined separately.

In the pre-runtime situation, there is no overwhelming argument for either the retention or elimination of repetition parameters. If the user feels the need for repetition parameters, then the ECOS translator should provide the capability to the user. However, this capability must be restricted so that the translator may elaborate the graph at "compile-time" and create local names for those graph elements which are automatically generated. The graph which is then emitted by the translator is completely elaborated. This clarifies the definition of repetition parameters to be merely a mechanism for shortening the description of the graph.

At runtime, repetition parameters are used in the scheduling of nodes. This is because a fully elaborated graph may contain components which will never be used in this specific instantiation. This is true regardless of whether the graph is constructed from graph fragments or whether the graph exists fully expanded in memory. In both cases the exact desired number of the replicated items must be given before execution can begin. The case where the complete graph is described in memory is very expensive in control overhead, since for every scheduling action the executive must evaluate each replicated item for the critical node to determine if the node may run. For example, if a node accepts as input a queue which may be replicated up to forty times, then each time the attempt is made to schedule it, it must evaluate the sizes and thresholds of all forty queues, even if they are not actually present in this configuration.

One possible solution is that valves could be used to serve the function of repetition parameters. Under this scheme, each "repetition parameter" in each processor would be initialized to its valve value upon graph instantiation. In the case where a queue is used and therefore must have its size tested against its threshold, two tests are required. If valves are used instead of repetition parameters, no additional checking is required since a test on valves must be done in any case, and this will be done as a power set comparison.

One issue which is not clear from the ECOS definition is the synchronization mechanism for graph execution. If the movement of data is this mechanism, then at how many points in time does this synchronization

occur? In a system which follows the description in Appendix C [1], there are two places. The first is when the data is actually produced onto the queue. This is inherently obvious since this is the point at which data "arrives" in the system for processing. The other point is when data is actually consumed from the queue. In most systems, however, this point will immediately precede the production of output queues for the same node, since the arithmetic element will not be interrupted for the queue consumption message. This type of synchronization, where the queue consumption is calculated by the node based upon data dependencies, is very expensive in terms of control processor overhead since actual data, the data size, must be sent to the control processor for each queue referenced. In contrast, a method which is based upon constant produce and consume amounts for queues does not have this problem since the synchronization action becomes "Node Complete", requiring only one message. The sizes of all productions and consumptions are known and handled locally based upon this single message. For this reason, the values for the produce and consume amounts for queues are fixed at compile time.

Similar arguments may be advanced for the threshold, read and capacity amounts. In the case of capacity, memory may be statically allocated for the entire graph upon instantiation if all of these values are known. Additionally, no runtime garbage collection is required in this case. The scheduling algorithm is simplified since the capacity and threshold fields are now static. Overall, this single change significantly reduces the control overhead for a single node execution.

The last area of examination in ECOS is trigger queues. These are intended to serve as explicit synchronization mechanisms for primitive programs. This is in direct conflict with the notion of "data-flow" since the only synchronization action that occurs is in the quantity of data produced. Two interpretations exist for the semantics of trigger queues as they are defined in ECOS. The first is that they are weakly bound to nodes, that is, that the primitives which constitute the nodes have no knowledge of trigger queues. The queues are produced and consumed only by the activation and termination of the nodes. This weak binding isolates the primitive programmer from trigger queues, but still requires the executive to move data for these queues, and more importantly, differentiate between a normal data queue and a trigger queue. If the trigger queue production/consumption is under the control of the primitive, then most advantages of trigger queues have vanished. The user is now aware of a difference in queue "types" as well as the executive. This is going to slow down both the control code and the applications algorithm.

An alternate to trigger queues is the normal mechanism of data queues. In a static production/consumption environment, this will not require that the primitives be aware of "trigger-queues" since they are now purely for synchronization. If no data is actually generated, the queue size will still be altered. This allows for a uniform treatment of all queue types and remove the burden from the primitive programmer.

This mechanism is transparent to the ECOS programmer. If an overwhelming need is felt for the TRIGGER clause, then it should be left in the language definition. However, ECOS is intended to represent a dataflow language and therefore should have all explicit control structures removed.

After these changes are applied to ECOS, the resultant language is easier to compile and to manage at runtime. A brief description of the modified ECOS is described in the section below.

2.2 REVISED ECOS

Graphs are treated as single compilation units. In contrast to a conventional programming language, such as Pascal, they are the analog to procedures and functions. If the graph generates values for use by another graph, or is used as a subgraph (i.e. it appears as a single node in another graph declaration), then it is a function. Otherwise, the graph is a procedure. As in conventional languages, graphs are named entities and they consist of a set of components: queues, primitives and variables.

Graph variables serve as global data in dataflow programs. Although graph variables are accessible to both command programs and nodes, they cannot be used for synchronization since copies exist in multiple processors[2]. Scaling factors and special coefficients are some of the uses for graph variables. They are the analog to system scope variables.

Variable declarations are unique within a graph context, and have an explicit compile-time type. The data types provided by the ECOS should match the underlying implementation language, which must at least cover the types of data found in the signal processing problem space (integer, real, and complex numbers). Various precisions should be allowed to match computational needs of the specific application.

Queue declarations serve to define the connectivity of the graph. Each queue may be considered as an analog to an operand in a procedural programming language. As in the case of a conventional variable, a queue must have some static attributes - an underlying arithmetic (or logical) type and a length. In this context, length refers to the maximum amount of data that the queue can hold as opposed to the size of the base data type. If the graph were to reside in a processing system which was a single monolithic processor, then these two attributes would be sufficient to describe the storage requirements of the queue. This is obvious since the existence of the queue can be tied to one specific instance in time, the moment when the producing node has finished and before the consuming node has begun. This assumption also holds true in a distributed multi-processor environment when the time order relationship of the data is preserved. Simply stated, data produced which has a time binding to other data, either serially or concurrently, has this binding preserved throughout the lifetime of the data.

As to the practical matter of implementation, only the produce amount must be specified for a given queue. All other queue parameters will default to the produce amount if no specific values are given. This is because the degenerate case of a queue is that it exists temporarily, only from the time that it is produced by node N and consumed by node N+1.

Several constraints are placed on queue parameters to prevent the production of data which will not fit into an output queue, and the consumption of data not yet present. These constraints are shown below.

```
CAPACITY >= PRODUCE
CAPACITY >= READ
CONSUME <= READ
THRESHOLD <= CAPACITY
THRESHOLD >= READ
```

In ECOS, nodes serve as operators on predefined aggregates of data: queues. The semantics of nodes are inherently those of operators in procedural programming languages. Each distinct node is an operator with its own set rules, defined by the attached queues, graph variables, and the node's internal logic. The attached queues place the node at a unique point in the graph solely by the single connectivity rule. They also serve to define the type and order of the data upon which the node operates. The order of processing of queues is as important in ECOS as it is in the overloaded dyadic operators in Ada.

2.3 GRAPH CHARACTERIZATION

The example shown below is extracted from the generic Ada package which contains the scheduler for the ECOS revision described in section 2.2. Pre-runtime manipulation of the graph topology is done using this representation. The runtime data structure is described below in section 4.1.

In this example, `NODE_NAMES`, `QUEUE_NAMES` and `VARIABLE_NAMES` are enumeration types which are parameters to the package[3,4]. They contain the names of all the respective items for this graph (a graph in this context is a complete elaborated, including resolution of subgraphs).

`BASE_TYPES` is a list of all implemented arithmetic data types available for use by the applications programmer. The precision and rounding attributes for these types are contained in the system package `STANDARD`. `QUEUE_SET` and `VARIABLE_SET` are power set treatments of all the queues and variables in this graph. This is for a terse success or failure test, i.e., a bit mask comparison. This assumes that the `PACK` pragma will collapse the boolean array into a bit string.

Note that only those fields in the component record appear which are needed at runtime. There is no need to track the producer of a queue, since the identifies it from its `OUTPUT` list. The `CONSUMER` field is required to lend speed to the scheduling of the consumer node based on the completion of a particular queue. The `DESIRED` and `ACTUAL` power sets are for the checking of schedule conditions. `DESIRED` is the set of all queues which must equal or exceed threshold for the node to run. The `ACTUAL` set is the current state of all attached queues. Graph variables are expressed as a variant record in this example only for ease of reference.

-- NODE_NAMES, QUEUE_NAMES and VARIABLE_NAMES are parametric enumeration
-- types defined at package instantiation.

type BASE_TYPES is (COMPLEX, INT, REAL);

type QUEUE_SET is array (QUEUE_NAMES) of boolean;

type VARIABLE_SET is array (VARIABLE_NAMES) of boolean;

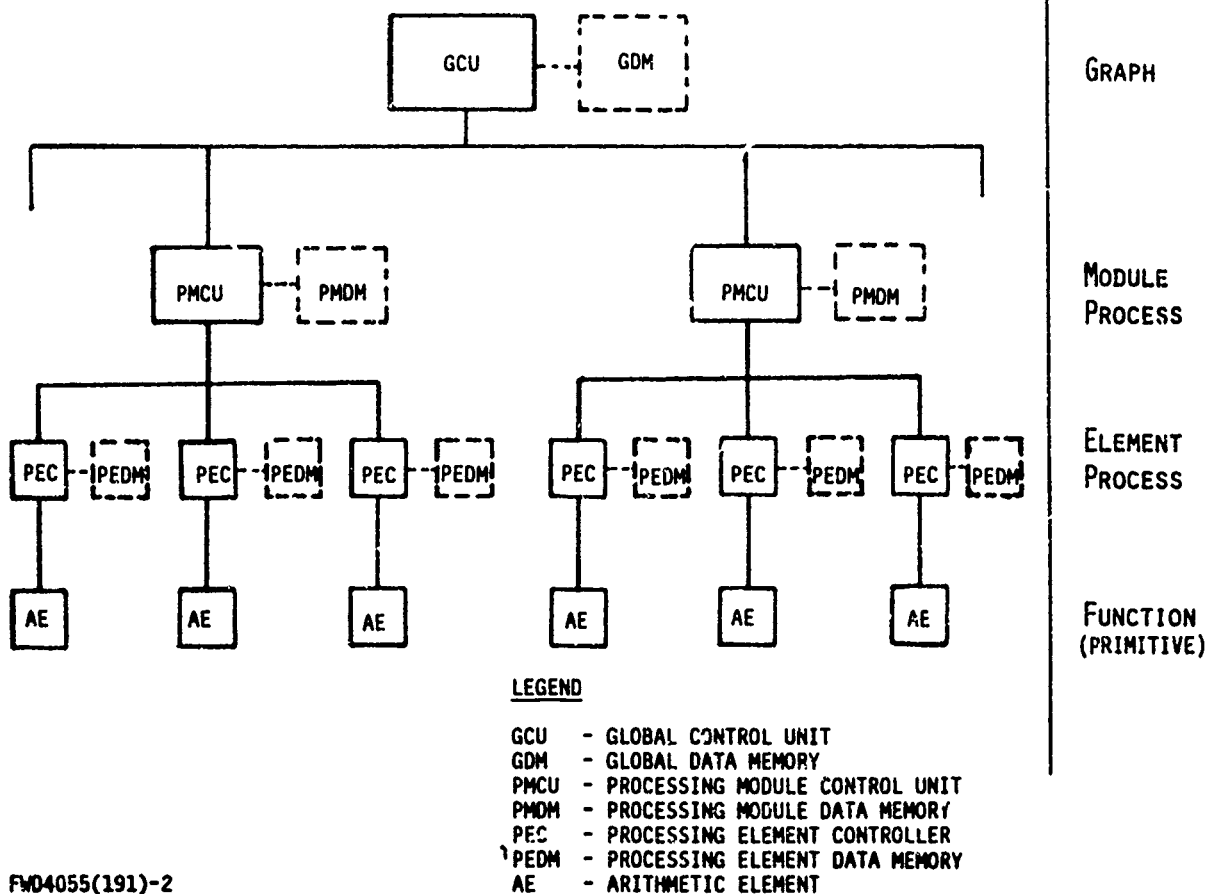
type QUEUE_ENTRY is record
 CAPACITY : integer;
 CONSUME : integer;
 CONSUMER : NODE_NAMES;
 PRODUCE : integer;
 READ : integer;
 SIZE : integer;
 THRESHOLD : integer;
 VALVE : VARIABLE_SET;
end record;

type NODE_ENTRY is record
 ACTUAL : QUEUE_SET;
 CONTROLS : VARIABLE_SET;
 DESIRED : QUEUE_SET;
 INPUT : QUEUE_SET;
 OUTPUT : QUEUE_SET;
end record;

type VARIABLE_ENTRY (VAR_TYPE : BASE_TYPES := INT) is record
 case VAR_TYPE is
 when INT =>
 INT_VALUE : integer;
 when COMPLEX =>
 COMPLEX_VALUE : COMPLEX_NUMBER;
 when REAL =>
 REAL_VALUE : FLOAT;
 end case;
end record;

3.0 HARDWARE DESCRIPTION

The system topology may be described as a distributed hierarchical architecture. The salient attribute of the architecture is three levels of loosely coupled processors. Reference Figure 1.



FWD4055(191)-2

FIGURE 1 DISTRIBUTED HIERARCHICAL ARCHITECTURE
FWD4055(191)-2

The system is loosely coupled in the sense that the processors at each level have their own data and program memories with associated control circuitry. Inter processor communication is via global memory and direct functional communication channels. Data and control flow is distributed through the three levels to avoid high bus bandwidths and contention problems. Control flow within the system is top-down with control distributed vertically and horizontally throughout the architecture. Data flow is via queue mechanisms in global memory and control flow is via communication channels.

The processing element is the lowest level in the Distributed Hierarchical Architecture. It is dedicated to signal processing arithmetic computations and characterized as a modular, asynchronous, floating point processor. It is at this level that the application primitives are executed.

3.1 Functional Description

The hierarchical multiprocessor system, shown in Figure 2, has more than one level of functional partitioning. From the highest level viewpoint it is a single signal processing system that is directed by an external platform control interface. Internally, at the Global Control Unit level, it is

- 6) The Processing Module Data Memory is a data retention and transfer resource for the Processing Module Control Unit on multiple Processing Elements.
- 7) The Processing Element is the Signal Processing Computational Unit for the Distributed Hierarchical Architecture and performs all arithmetic processes.
- 8) The Processing Module Control Unit and Data Memory and one to five Processing Elements are grouped together to constitute a Processing Module.

3.2 Analysis, Modeling, And Simulation

The Distributed Hierarchical Architecture was selected for an extended analysis, modeling, and simulation effort using the IBM simulation product, General Purpose Simulation System (GPSS). Two signal processing applications, Adaptive Beam Forming (ABF) and Sonobuoy, were successfully overlaid on the proposed architecture.

The data and control bandwidths within the Distributed Hierarchical Architecture machine were evaluated and shown to be acceptable through paper analyses and computer simulations for the two applications, ABF and Sonobuoy. Bandwidth requirements for the various data paths within the system were established by paper analysis. Simple, analytic models were used to study bandwidth tradeoffs between various processing and memory resources. Simulations were used to verify the analytic models. Examples of some of the GPSS simulation models that were developed is the data bus arbitration and memory bank allocation schemes. The total system modeled by GPSS included five processing modules, each with five processing elements.

The GPSS modeling included statistics on data bus utilization, processing element utilization, and maximum memory demands. The results obtained compared favorably with those of the static analyses.

4.0 OPERATING SYSTEM

The operating system in the EMSP system described above is derived from the requirements of the applications language, i.e. the services that it must provide, and from the nature of the hardware that it must control. This section describes the structure of this operating system, and presents a detailed discussion of the algorithm used in the PM controller to schedule execution of nodes for the PEs.

The types of resources that must be managed by the PM controller are intrinsically bound to the two building blocks of data flow programming: I/O capacity and CPU time. The I/O capacity is divided into three areas, each representing a synchronization action for this processor.

- 1) GCU commands - startup or termination action for graphs.
- 2) inter-PM messages - completion messages from graph fragments in other Processing Modules.
- 3) PE process request - a node has finished in an AE, and a new node must be supplied.
- 4) Data transfer - raw data has arrived from the environment into the EMSP system.

The CPU time is also broken into three areas, each involved in advancing the computation (graph execution). The completion of the computation is ensured by the pre-runtime system.

- 1) Node scheduling - maintaining a list of available nodes for execution.
- 2) Memory management - synchronizing data accesses through local and global memory.
- 3) Error recovery - relates to both the I/O and CPU resource management, but is very CPU intensive.

All aspects of the EMSP executive are derived from the data flow requirements of the ECOS notation. The execution of a graph is based solely on conditioned (value) operand (queue) availability; each operation (node) may begin execution as soon as all of its inputs are present[5]. This places the node scheduling function as the critical element of the executive.

4.1 NODE SCHEDULING

In general, there are two approaches to node scheduling in a data flow systems: synchronize and schedule on the beginning of node execution or on node completion. The model developed for the Magnavox EMSP architecture is based upon the node completion scheduling algorithm. In this type of system, the synchronization action for the system is "NODE COMPLETE". All of the information to advance the computation is the directly accessible from the runtime data structure. This also focuses efficiency considerations on the routines which manipulate these data structures. The figure shown below is the node scheduler for the EMSP PM controller.

```

-----
-- EMSP Distributed Data-flow Executive:
--   Generic Scheduler
--
-- This package definition is a template for a generic (Ada-speak)
-- node scheduler for an EMSP which has the Magnavox heirarchy of
-- control and data flow. The only visible points to the world
-- are the NODE_COMPLETE routines which perform synchronization
-- between nodes, regardless of Processing Module location. When
-- the package is instanced, the enumerations lists of NODE_NAMES,
-----

```



```

-- QUEUE_NAMES, and VARIABLE_NAMES must be provided. There will  --
-- also be another package, <graph-name>_INIT, which will fill in  ---
-- then defined arrays of NODES, QUEUES and VARIABLES. This      ---
-- this single scheduling package to be maintained and optimized  ---
-- for a given architecture.                                     ---
--

```

```
generic
```

```
    type NODE_NAMES      is (<>);
```

```
    type QUEUE_NAMES     is (<>);
```

```
    type VARIABLE_NAMES is (<>);
```

```
package SCHEDULER is
```

```
    subtype ADDRESS      is integer range 0 .. system.max_int;
```

```
    type BASE_TYPES     is (COMPLEX, INT, REAL);
```

```
    type QUEUE_SET      is array (QUEUE_NAMES) of boolean;
```

```
    type SCOPE          is (LOCAL, GLOBAL);
```

```
    type VARIABLE_SET   is array (VARIABLE_NAMES) of boolean;
```

```
    type QUEUE_ENTRY (CONTEXT : SCOPE) is record
```

```
        case CONTEXT is
```

```
            when LOCAL =>
```

```
                BASE      : ADDRESS;
                CAPACITY  : integer;
                CONSUME    : integer;
                CONSUMER   : NODE_NAMES;
                HEAD       : ADDRESS;
                PRODUCE    : integer;
                READ       : integer;
                SIZE       : integer;
                TAIL       : ADDRESS;
                THRESHOLD  : integer;
                VALVE      : VARIABLE_NAMES;
```

```
            when GLOBAL =>
```

```
                null;
```

```
        end case;
```

```
    end record;
```

```
    type NODE_ENTRY (CONTEXT : SCOPE) is record
```

```
        case CONTEXT is
```

```
            when LOCAL =>
```

```
                ACTUAL    : QUEUE_SET;
                CLASS     : integer;
                CONTROLS   : VARIABLE_SET;
                DESIRED    : QUEUE_SET;
                INPUT      : QUEUE_SET;
                LOCKED     : boolean;
```

```

        OUTPUT      : QUEUE_SET;
    when GLOBAL =>
        null;
    end case;
end record;

type VARIABLE_ENTRY (VAR_TYPE : BASE_TYPES := INT) is record
case VAR_TYPE is
when INT =>
    INT_VALUE      : integer;
when COMPLEX =>
    COMPLEX_VALUE  : COMPLEX_NUMBER;
when REAL =>
    REAL_VALUE     : FLOAT;
end case;
end record;

NODES      : array (NODE_NAMES)      of NODE_ENTRY;
QUEUES     : array (QUEUE_NAMES)     of QUEUE_ENTRY;
VARIABLES : array (VARIABLE_NAMES)  of VARIABLE_ENTRY;

procedure GLOBAL_NODE_COMPLETE (N: in NODE_NAMES);
procedure LOCAL_NODE_COMPLETE (N : in NODE_NAMES);

end SCHEDULER;

package body SCHEDULER is
    procedure broadcast (N : in NODE_NAMES) is separate;
    -- map to IAPX-432 I/O processor

```

```

-- The Ada tasking mechanism is used to guarantee that for each
-- graph only one of the CONSUME, PRODUCE or SCHEDULE actions is
-- occurring at any given moment. Multiple graphs cannot affect
-- each other's execution by simultaneous access of the same data
-- (i.e. queue).

```

```

task UPDATE_QUEUE is
    entry CONSUME_QUEUE (Q : in QUEUE_NAMES);
    entry PRODUCE_QUEUE (Q : in QUEUE_NAMES);
    entry SCHEDULE      (N : in NODE_NAMES);
end UPDATE_QUEUE;

task body UPDATE_QUEUE is
    Q : QUEUE_NAMES;
begin
    loop
        select
            accept CONSUME_QUEUE (Q : in QUEUE_NAMES) do

```

```

    QUEUES(Q).SIZE := QUEUES(Q).SIZE - QUEUES(Q).CONSUME;
    QUEUES(Q).HEAD := QUEUES(Q).HEAD + QUEUES(Q).PRODUCE;
    if QUEUES(Q).HEAD >= QUEUES(Q).BASE + QUEUES(Q).CAPACITY then
        QUEUES(Q).HEAD := QUEUES(Q).BASE;
    end if;
    if QUEUES(Q).SIZE < QUEUES(Q).THRESHOLD and then
        NODE(QUEUES(Q).CONSUMER).CONTEXT = LOCAL then
        NODE(QUEUES(Q).CONSUMER).ACTUAL(Q) := false;
    end if;
end CONSUME_QUEUE;

or
accept PRODUCE_QUEUE(Q : in QUEUE_NAMES) do
    -- only do PRODUCE if valve is on, allowing flow
    if VARIABLES(QUEUES(Q).VALVE).INT VALUE /= 0 then
        QUEUES(Q).SIZE := QUEUES(Q).SIZE + QUEUES(Q).PRODUCE;
        if QUEUES(Q).TAIL = QUEUES(Q).BASE + QUEUES(Q).CAPACITY then
            QUEUES(Q).TAIL := QUEUES(Q).BASE;
        else
            QUEUES(Q).TAIL := QUEUES(Q).TAIL + QUEUES(Q).PRODUCE;
        end if;
        if QUEUES(Q).SIZE >= QUEUES(Q).THRESHOLD and then
            NODE(QUEUES(Q).CONSUMER).CONTEXT = LOCAL then
            NODE(QUEUES(Q).CONSUMER).ACTUAL(Q) := true;
        end if;
    end if;
end PRODUCE_QUEUE;

or
accept SCHEDULE(N : in NODE_NAMES) do
    for Q in NODE_NAMES'RANGE loop
        if NODES(N).OUTPUT(Q) and then
            NODES(N).CONTEXT = LOCAL and then
                (QUEUES(Q).SIZE + QUEUES(Q).PRODUCE >= QUEUES(Q).CAPACITY) then
                    return;
                end if;
            end loop;
            NODES(N).LOCKED := true;
            -- place node N on the dispatching port for the PE
        end SCHEDULE;
    end select;
end loop;
end UPDATE_QUEUE;

procedure GLOBAL_NODE_COMPLETE (N : in NODE_NAMES) is
    Q
        : QUEUE_NAMES;
begin
    for Q in QUEUE_NAMES'RANGE loop
        if NODES(N).OUTPUT(Q) and then
            NODES(QUEUES(Q).CONSUMER).CONTEXT = LOCAL then PRODUCE_QUEUE(Q);
            if not NODES(QUEUES(Q).CONSUMER).LOCKED and then
                NODES(QUEUES(Q).CONSUMER).ACTUAL = NODES(QUEUES(Q).CONSUMER).DESIRED
            then
                SCHEDULE(QUEUES(Q).CONSUMER);
            end if;
        end if;
    end loop;
end loop;

```

```

end GLOBAL_NODE_COMPLETE;

procedure LOCAL_NODE_COMPLETE (N : in NODE_NAMES) is
  ALREADY_BROADCAST      : boolean      := false;
  Q                       : QUEUE_NAMES;
begin
  NODES(N).LOCKED := false;           -- node is no longer executing
  for Q in QUEUE_NAMES'RANGE loop
    if NODES(N).INPUT(Q) then        -- for all input queues
      CONSUME_QUEUE(Q);              -- consume in critical region
    end if;
  end loop;

  for Q in QUEUE_NAMES'RANGE loop
    if NODES(N).OUTPUT(Q) then       -- for all output queues
      if NODES(QUEUES(Q).CONSUMER).CONTEXT = LOCAL then
        PRODUCE_QUEUE(Q);            -- consumer in this processor
        if not NODES(QUEUES(Q).CONSUMER).LOCKED and then
          NODES(QUEUES(Q).CONSUMER).ACTUAL = NODES(QUEUES(Q).CONSUMER).DESIRED
        then
          SCHEDULE(QUEUES(Q).CONSUMER);
        end if;
      elsif not ALREADY_BROADCAST then
        BROADCAST(N);
        ALREADY_BROADCAST := true;
      end if;
    end if;
  end loop;

  if not NODES(N).LOCKED and then
    NODES(N).ACTUAL = NODES(N).DESIRED then
    SCHEDULE(N);
  end if;
end LOCAL_NODE_COMPLETE;
end SCHEDULER;

```

The intent of this data structure is to hold the fully elaborated graph in memory in all of the PM. In order to reduce the size of the structure, variant records were used to differentiate between local and non-local entities. Local graph components are used for scheduling, and non-local components merely complete the graph topology.

The variable CONTEXT is used to identify the locality of this graph component. If a node is GLOBAL, then the scheduling algorithm is not run in this PM, but it is triggered in the correct PM by broadcasting the "NODE COMPLETE". In the case of a queue, if either end is used within a PM, then the CONTEXT for the queue is LOCAL. This has no implications on where the queue is stored, but merely states where it is used. In an idealized processor, storage would directly correspond to usage; however, memory constraints will restrict where data may actually be placed.

There are two scheduling routines, one for the scheduling of nodes which have all inputs produced by nodes internal to the PM; and one for the nodes which have one or more inputs generated in another PM or which come from the

EMSP environment. In both cases, there are several constraints which must be met before the actual scheduling occurs. First the node must not be currently scheduled (time ordering of data), and second all of the queues must at least meet their threshold. If these conditions are true, then the actual scheduler is invoked. The scheduler now checks for the only blocking condition which is that the production of an output queue would exceed a queue capacity. If no queue meets this condition, then the node is placed upon a general dispatching port for the appropriate class of PE/AE.

ECOS requires that the time order of the data be preserved across the execution of the graph. Given the general structure of the PM/PE/AE network, the only way to ensure this order is to block the execution of subsequent node firings until the previous firing completes. The LOCKED variable solves this problem by preventing multiple schedulings of the same node. It is set to true in the scheduler just before the node is placed on the dispatching port, and is set to false upon reception of "NODE COMPLETE" from the PE.

Memory is managed by each PM. They manage the local data memory which is directly under their control. They also manage segments of global memory. The GCU allocates blocks of global memory to each PM on a queue-by-queue basis. The determination of which queues reside in global memory is made by the portion of the executive in the GCU based upon information provided by the pre-runtime system. This makes the global memory assignment a startup function only. (It may possibly occur during major system reconfiguration -- such as when an entire PM fails.) All memory is allocated as circular queues, with a HEAD and a TAIL. By making the PRODUCE and CONSUME amounts static, the queue manipulation problem becomes trivial.

5.0 SUMMARY

This system has been implemented in Pascal, MACRO-32 and Ada on a VAX 11/780 at the Magnavox Fort Wayne facility. The system includes a graph analyzer and optimizer which generates statically configured graphs for execution in a reconfigurable hardware level simulator. This simulation system was used to refine the concepts presented in this paper and to test the assumptions of single message synchronization through the "NODE COMPLETE" mechanism. The ECOS compiler has been further refined and been rehosted to a PDP-11 and translated into Edison[5]. This second system is much smaller than the VAX version and provides more refined compilation features and better generated code.

Several versions of the Ada executive have been implemented on an Intel iAPX432. However, the current Intel Ada compiler[7] does not support tasking, which has prevented the testing of the code shown above. The solution was tested using an assembly language routine to perform the function of the Ada task dispatcher. Improved performance is expected with upgrades in the compiler.

BIBLIOGRAPHY

- 1) PD-101 (PMS-4083) For the Enhanced Modular Signal Processor (EMSP), N00024-81-R-7151(Q), May 1982
- 2) P. Brinch Hansen, "Distributed Processes: A Concurrent Programming Concept", CACM, 21(11), pp 934-940, November 1978
- 3) "Reference Manual for the Ada Programming Language", Department of Defense, November 1980
- 4) I.C. Pyle, "The Ada Programming Language", Prentice-Hall, 1981
- 5) J.R. McGraw, "The VAL Language: Description and Analysis", ACM TOPLAS, Vol. 4 No. 1, pp 44-82, Jan. 82
- 6) P. Brinch Hansen, "The Design of Edison", SP&E, 11(4), pp 353-396, April 1981
- 7) 'Engineering Specifications for the iAPX432 Extensions to Ada', Intel, 1982

Mr. Steven E. Adams received the B.S. degree with honors in engineering sciences from the University of Cincinnati, Cincinnati, Ohio, in 1977.

Presently, he is the Technical Director for Intermetrics, Inc., Dayton, Ohio. He has served as Chief Programmer on the DAIS program, the 1750 Verification Software program, and on the Magnavox Enhanced Modular Signal Processor (EMSP) development effort. He has worked in the areas of language, operating system and hardware design. Currently, he is the Principal Investigator on a Research and Development project for Magnavox in the area of signal processor operating systems.

Mr. Thomas R. Butler is a project engineer with Magnavox Electronic Systems Company in Fort Wayne, Indiana. His current assignment is the Advanced Multiprocessor Architecture Development Program. This program focuses on the development of architectures capable of high throughput rates while being supported by High Order Languages. Mr. Butler's previous assignments involved work on the Navy's Enhanced Modular Signal Processor program and a finite state processor for the FAA Tower Cab Digital Display system.

Mr. Butler served as vice president and was a member of the Board of Directors of CPU Inc., a company specializing in custom hardware and software for business and industry. He holds one patent in bandwidth reduction techniques. Mr. Butler received an A.A.S. degree with Distinction from Purdue University in 1977.

Avionic Systems Integration Facilities

Mark van den Broek

Paul M. Vicen

Abstract

Management of Embedded Computer Systems (ECS) after deployment presents a major challenge to the Air Force Logistics Command (AFLC). The organization to accomplish that task is described, along with AFLC's mission and support concepts. The briefing then describes several of the initiatives being pursued by AFLC to improve support for current and next generation systems under the ECS Support Improvement Program (ESIP).

Biographical Sketch - Paul M. Vicen

Mr Vicen has been working on AFLC embedded computer system support since 1975, with efforts ranging from software personnel issues to engineering facility design. He is currently assigned as the program manager of the ECS Support Improvement Program.

Mr Vicen received a B.S. from Ohio State University in 1971, and a M.S. in Management Science from the University of Dayton in 1981.

Biographical Sketch - Mark van den Broek

Mr van den Broek, currently the Chief of Electronics Engineering and Integration Division at HQ AFLC, has been intimately involved in ECS management since 1974. He has been instrumental in developing the support philosophies and capabilities for avionics, communications, and electronic warfare systems.

Mr van den Broek holds an MBA from George Washington University and a B.S. in Business from Juniata College. He presented a paper on ECS support to NAECON in 1978 and has participated in a number of other major forums, including the Digital Avionics Support Conference and the AFSC Standardization Conference.

PLANNING OF OPERATIONAL SOFTWARE IMPLEMENTATION TOOL

Aaron Spinak

Proprietary Software Systems, Inc.
9911 West Pico Boulevard, Penthouse K
Los Angeles, California 90035
(213) 553-2997

Biographical Sketch

Aaron Spinak is a senior member of the technical staff at PSS specializing in the planning and implementation of large scale software developments in the aerospace environment. He was manager of IRAD Software at Litton, DSD where he first developed the POSIT methodology, and has subsequently refined it on numerous assignments.

Abstract

This article describes the Planning of Operational Software Implementation Tool. The plan is based upon the identification of the hierarchical relationship of the individual elements of the software design, the development of a sequence of functionally oriented demonstrable steps, the allocation of subroutines to the specific step where they are first required, and objective status reporting. The results are meaningful determination of milestones, improved managerial visibility, better project control, and ultimately a successful software development.

I. Introduction

This paper addresses the lack of a standardized large scale software planning and implementation methodology. Various software standards are in existence. Generally, they have only standardized programmer oriented activities. These activities range from preparation of a standard set of software related documents (requirements specification, design specification, user manuals, etc.) to standardizing coding techniques (structured programming rules, standardized languages, etc.). This is all necessary and good, and yet there has not been a significant corresponding improvement in the large scale software development picture. Costs are still excessively high, schedule overruns are the norm and program quality and reliability are still questionable.

This paper asserts that the failures of large scale software developments result primarily from the failings of the software and systems implementation planning and management methods. It also concludes that a programming team with average competency and excellent implementation planning and management methods stands a far better chance of success with a large scale software development than does an outstanding group of programmers coupled with a poor implementation planning and management methodology.

An implementation planning and management methodology, utilizing automated management tools is described on the following pages. The main objective of the paper is to spotlight the importance of implementation planning and management and hopefully to inspire the initiation of some standardization efforts in this area.

II. Background

The success of large software development efforts has improved throughout the industry. These improvements are largely attributable to the application of a technological wave of new approaches that have been loosely referred to as structured programming. More explicitly, the new technologies include replacement of flowcharts via usage of design languages, elimination of the GOTO by confinement to a small complete set of logical constructs, increased emphasis and formalization of the role of the programming support librarian, increased emphasis on reviews via usage of either structured walk-throughs or inspection teams, and a reorganization of programming personnel into the chief programmer team formation. Unfortunately, despite the considerable progress that has been made, many projects still fail to meet their schedule, have cost overruns, and the end product never quite operates as reliably as intended. In any event, even for supposedly successful projects, the cost of software is still too high.

The major reason for these continuing software difficulties and continued high costs, despite advances in technique, is that the impact of the aforementioned technical advances is limited when constrained by the effects of traditional management techniques. All of the previously mentioned structured programming techniques deal with the programmer and programming. None deal directly with the issues of planning and managing a large scale software development. The industry is generally using the same planning and management approaches it has always used, and these have frequently proven to be unsuccessful. The result is that the manager continues to have little visibility and little effective control over the developing system. If the manager had a mechanism that permitted him to arrive at a meaningful implementation plan; permitted him to objectively assess the project's status as it developed; provided him clear visibility of the development activity; considered cost, schedule, manpower and the chosen design, then the manager would be in a position to truly manage the project and lead it to a successful conclusion at minimal cost.

The Planning of Operational Software Implementation Tool (POSIT) fills this gap. POSIT is to management as structured programming is to the programmer. As with structured programming, which is complemented by this plan, the plan improves visibility, meaningfulness and orderliness. It allows the manager to start the project off on the right path, closely monitor the software development as it progresses, and ultimately to bring the project to the desired successful conclusion.

III. Goals

When this implementation management and planning methodology was first being created, the following goals were established for the methodology:

- o It should produce a meaningful (as opposed to arbitrary) implementation plan.
- o It should be a detailed working plan that forms the basis for the day-to-day management of the project.
- o It should be the basic tracking mechanism.
- o It should objectively provide project status.
- o It should provide management with total project visibility.
- o It should be compatible with state-of-the-art programming techniques.
- o It should provide objective early reporting of development trends.
- o It should readily allow itself to be maintained and updated utilizing automated mechanisms.
- o It should dramatically improve the probability of completing a large scale software development on schedule, within budget and with exceptional quality.

As fantastic as it may sound, nevertheless it is true, that the planning and management methodology described herein consistently accomplishes all of these goals and in addition provides a large number of side benefits.

IV. Implementation Plan Selection Criteria

The Top Down method appears to have been first espoused by Dr. H. D. Mills of IBM. Though this discussion, and other discussions in the computing literature, advocated top down design and implementation, the focus was on top down design and little direction was presented on how to plan a top down implementation. Simply stating that a computer program should be implemented in a top down sequence is insufficient for a large software development. Due to the complexity of the hierarchical structure for larger software developments, literally thousands of top down implementation sequences may be possible. It is essential for success that a proper top down implementation sequence is selected.

For example, one could implement all the subroutines at a particular hierarchical level for the entire system, followed by all the subroutines at the next level across the system, and so on, until finally the bottom level subroutines are implemented. There are those in the industry who advocate this sequence. This would be top down, but in our opinion, represents an inferior implementation sequence. This is because bottom level subroutines usually are required to provide a demonstration of a completed operational system function. Thus, for most of the system's development, very few

operational functions would be demonstrable. However, early functional demonstrability is one of the main benefits that should be achieved from top down implementation. Alternatively, many other top down sequences might be inferior because they strongly conflict with expected equipment delivery dates for the system being developed.

Selecting the most appropriate top down implementation sequence is an issue of primary importance. POSIT uses a comprehensive methodology that has been developed for creating and maintaining an optimal top down implementation. This technique provides broad benefits to virtually all aspects of the ensuing software development.

V. Top Down Concepts

Top down design refers to a method of designing a computer program wherein higher level or calling subroutines are designed before lower level or called subroutines. This does not mean that all subroutines at one level must be designed, or named, before creating the design, or name, of any subroutines at the next level. It means that if one were to consider the system's Subroutine Hierarchy as a treelike structure, along each branch of the tree subroutines would be defined and chosen for design, starting from the top of the hierarchy and working down.

Top down implementation refers to the development of a computer program in a downward hierarchical sequence along each branch of the program's Subroutine Hierarchy. Design, documentation, coding, integration and testing usually are concurrently performed on different portions of the developing system. In a top down sequence, these are performed along each branch simultaneously under development.

VI. Preparation of the POSIT Plan

A common mistake is to prematurely create the detailed implementation plan. If we face reality, we must admit that the effect of this mistake is a plan with a high level of arbitrariness. This happens because creation of a meaningful implementation plan demands a greater understanding and analysis of the system, and that translates to more time, cost and effort spent in planning. Perhaps even less common is for management and the customer to demonstrate the patience for developing a meaningful plan. Unfortunately, developing large scale software without a good plan is like flying without navigation gear; there's no telling where you will end up.

A viable software implementation plan can only be prepared after a sufficient quantity of system analysis activities have occurred and before the detailed implementation has begun. The plan is then used to launch the implementation phase for a large scale software development. In operation, the POSIT approach is based upon the utilization and interplay of three elements or tools. They are the Subroutine Hierarchy, the Network of Demonstrable Functions (NDF), and the Software Status Report (SSR).

In a nutshell, the Subroutine Hierarchy represents a design abstract for the computer software. The Network of Demonstrable Functions represents a functional abstract of the operational system. The Software Status Report

relates the software design to the NDF system functions for the purpose of scheduling the software and maintaining the status of software development. The main point of this nutshell description is that attentive preparation of these documents results in a meaningful schedule that allows management to have real visibility in areas such as the software's true status, cost to completion, and time to completion.

VII. Subroutine Hierarchy

Top down implementation planning is based on certain premises. One of these premises is that by minimizing or eliminating large unknowns, management has the best chance of accomplishing the project's goals. If there is some large functional area for which management has little basis, other than someone's intuition, for expecting the implementation to take say six months, with a particular size staff, as opposed to say three years, then the project is in a precarious position. A large nebulous function that has only been quantified at its total level by intuition, even though based on experience, is a dangerous unknown. The obvious way to get better control of a big unknown is by reducing it to many small pieces, some of which may be small unknowns. To put it in other terms, analyzing the task and breaking it down into many smaller pieces eliminates the risk of large unknowns. There may still be some unknowns or surprises, but the potential absolute effect of a misjudgement relative to a small task is going to be inherently smaller than for a misjudgement associated with the much larger original task. An important additional aspect is that in the process of decomposing the original function understanding occurs and, for the most part, comprehension replaces intuition.

Also, by decomposing a system into a large number of small pieces, a point is reached where the individual pieces can be treated for planning purposes as statistically equivalent. At the management level, the differences in size or complexity of individual small subroutines is of minimal importance. As subroutines are implemented, actual data should be used to update the estimated statistical characteristics of the average subroutine. For example, suppose an original memory allocation of 128K is made for 2000 subroutines. This averages 64 memory words per subroutine. Suppose, after 200 subroutines have been implemented, 15000 words have been used. This would show an actual average of 75 words per subroutine with the trend total being 150K for all 2000 subroutines. Thus, with only 10% of the subroutines implemented, a reliable danger signal has been raised, and the signal includes the magnitude of the forecasted overrun. With such an early warning, management still has time to take some appropriate effective action to act upon the issue before it becomes an actual problem.

This premise of statistical equivalence leads us to strive towards creating a full Subroutine Hierarchy after the "analysis phase" and as an initial part of the POSIT Plan.

The Subroutine Hierarchy is a high-level representation of the structure of the hierarchical design of the computer program. It readily conveys a high-level image of the design being represented, showing all of the parts constituting the design, their hierarchical relationship to each other, their categories and, to a degree, their functions. All of the subroutines must represent all subroutines, perhaps averaging 25 to 50 higher-order language statements.

The Subroutine Hierarchy evolves as the design and the software evolve. Initially, when the implementation plan is first prepared, the Subroutine Hierarchy represents the intended design structure of the final computer program. At completion of the software development, it represents the actual structure of the final computer program. At all intermediate stages, it is kept current and represents the currently projected structure of the final program.

For a large computer program, with perhaps one thousand or more subroutines, the subroutines may be treated in a statistical manner for the purposes of determining status, and for making estimates, schedules and plans. This is one of the basic principles of this approach. Namely, by partitioning a large computer program into its elemental pieces (subroutines), the effects of isolated misjudgement (e.g. size or complexity) relative to any individual subroutine tend to average out over the total program development, and do not affect the overall outcome. The effects of frequent misjudgement of the same characteristic of many subroutines (e.g., development time) tend to become quickly apparent and serve as a reliable indicator of development trends and ultimate results (if not corrected).

The Subroutine Hierarchy enables the software designers to conveniently conceptualize about the program and its parts, and to visualize the hierarchical organization of the program. It communicates in an overall conceptual manner the structure of the design. It is the essential design element, representing the components of the program's design for the purpose of planning and tracking the implementation of that design. It thus becomes the dominant factor in estimates of cost, manpower and memory size for the computer program.

Figure 1 shows a portion of the Subroutine Hierarchy for an actual project. Due to the large number of subroutines, the hierarchical structure has been automated and is represented in an equivalent horizontal rather than vertical (or treelike) manner. Varying hierarchical levels are represented by varying levels of indentation. The hierarchy identifies both the symbolic name and descriptive name of each subroutine. It identifies the particular step in the implementation plan where the subroutine will be first required. (The next section will elaborate on the definition of steps.) Only the first occurrence of a subroutine in the tree is expanded to the bottom level. Subsequent occurrences of any subroutine use a reference number to identify the line number of the first occurrence. If the subroutine itself invokes other subroutines, an asterisk is used to indicate that the full expansion can be found at the first occurrence. This automated technique considerably reduces the size of the Subroutine Hierarchy document.

VIII. Network of Demonstrable Functions (NDF)

The structure of the software design and the identification of the constituent subroutines have been described as part of preparing the Subroutine Hierarchy. No discussion has yet occurred relative to the development sequence of these subroutines, nor relative to the individual milestones that will be scheduled and tracked during the development. This is where the NDF comes into play.

The NDF is that part of the implementation plan that identifies the individual functional increments, or steps, and the sequence of development for those steps. Related steps are grouped together in a natural functional sequence to form paths for major subfunctions of the system. The steps are scheduled, developed, tracked, integrated, tested and eventually internally demonstrated and accepted. In other words, on the surface it is a "Pert-like network." Beneath the surface there are a number of aspects of the NDF that must be explained before its value can be fully grasped.

First of all, the NDF must be created by personnel that have an in-depth functional understanding of the application, its requirements, and the expected operational characteristics of the system. The personnel assigned to the NDF task will have acquired the necessary knowledge as a result of their prior system analysis activities. If they do not have this knowledge, they must first acquire it before they can hope to create a meaningful, detailed, functionally oriented plan of demonstrable steps. This is an essential point; a considerable comprehension of the requirements and analysis of the system is mandatory preparation for creating an NDF that will truly be able to guide the implementation to a successful conclusion. For various misguided reasons, projects often cut short the effort necessary to analyze and prepare a viable implementation plan. The consequent "savings" is usually given up by the eventual schedule overruns resulting from the original poor plan.

Secondly, the NDF steps are oriented towards functions primarily from the user's standpoint, not from the programmer's standpoint. For example, "output directive/menu index" is a typical step. This is as opposed to "build test configuration table", which would occur internally within the computer and not provide the user direct observation of the step having occurred. On the other hand, a step such as "print test configuration table" could be demonstrated to the user. Successful demonstration of this "printing" would imply successful "building" of the test configuration table.

This leads us into the third important aspect of the NDF. To the maximum extent possible, steps of the NDF should be readily demonstrable to an observer who is not a programmer. Those few steps that are not readily demonstrable to such an observer must, nevertheless, still be demonstrable. This demonstrability is the only basis upon which an objective determination can be made as to the completion of the step.

The principle of demonstrability leads us to a fourth important aspect of the NDF. The development sequence of demonstrable steps must correspond to a natural functional sequence of increasing functional capability. To put it another way, from the user's operational standpoint, it must be a sequence which demonstrates "first things first."

A fifth important aspect of the NDF is that the steps must each add on to an already cycling system. Each new step must be directly integrated into the cycling system, producing a continuously increasing functional capability that is always demonstrable. Steps required to demonstrate a new step must be implemented and integrated prior to the integration of the new step. In terms of subroutines, this means that for a particular step, those subroutines that are required for invoking a particular subroutine of that step must be implemented as part of that step or as part of a prior step. In other words, the design must be implemented in a top down sequence along each branch of the

Subroutine Hierarchy. Subroutines that are referenced but are not required for demonstration of the particular step are to be left as stubs until a step requiring those subroutines is undertaken. Sometimes it is necessary to create an "environmental interface module" (EIM) to simulate some part of the environment that the step is dependent upon, but which is not yet actually available to function as part of the system. In that case, an EIM is created so that the software can continue to be demonstrated as part of a cycling system.

IX. Software Status Report (SSR)

The key basis for planning and tracking the implementation is assigning the implementation of each subroutine to a single step. This is where it all comes together. The correlation between the Subroutine Hierarchy and the Software Status Report must be accurate. When design or plan changes occur, and they will, changes must be made to both documents. Both of these documents should be looked upon as evolving documents, but they must evolve concurrently.

Why does this "single step" premise form the basis to this approach to implementation planning? Because everything is accounted for. Each subroutine appears for implementation on only one step — the first step that requires the subroutine. The effort required for each step can be considered to be a function of the number of subroutines in the step. The programming implementation budget can be spread over the steps in proportion to the number of subroutines in each step. Then, if you are on schedule, you are on budget. Subroutines don't appear redundantly (on more than one step) to confuse the bookkeeping. Everything balances and all subroutines are able to be tracked. A full decomposition of the system into subroutines and a careful and complete assignment of those subroutines to a series of well-defined, demonstrable steps is of fundamental importance to successful usage of the POSIT methodology.

The Software Status Report ties the Subroutine Hierarchy and the NDF together by relating the design elements to the demonstrable steps. This is the fundamental point from which the value of the Software Status Report, and even the POSIT methodology, is derived. The Software Status Report meaningfully relates the design to demonstrable functions and the corresponding schedules.

In concept, the document is very simple. For each step from the NDF, the corresponding required subroutines from the hierarchy are listed. Each subroutine is allocated to a single step, the first step from the NDF that requires the particular subroutine. Consequently, the subroutines listed under a particular step are just those subroutines still required for demonstration of the particular step's function. Other subroutines may also be required for the step, but they would not be listed with the step if some prior step already required the subroutines. For status tracking purposes, columns are provided where design, code, documentation, test, size and other status fields can be checked off for each subroutine. These fields will be recorded as complete or will contain the date set for completion. Each particular detailed task and category (e.g., coding a subroutine) is governed by a well-defined checkpoint. When the task has satisfied the checkpoint criteria, its status is recorded as 100% complete; prior to that point the

task's status is carried as zero percent complete. An example of a checkpoint rule is "a subroutine is not designed until it has been approved by an internal peer design review". No attempt is made to allow intermediate percentage estimates of completion by the programmer. Statistical data provided in the Software Status Report is based on treatment of individual subroutines as statistical equivalents. In addition, the Software Status Report includes a description of each step, i.e., the function that is being demonstrated by the particular step. The report also identifies the qualifications or limitations, if any, that apply to the step's demonstration and the requirements that the step fulfills.

Whereas in concept the Software Status Report is very straightforward, creation of the report requires a thorough functional understanding of the system and of the corresponding design as represented by the Subroutine Hierarchy. Only with such knowledge as a base could the programming staff hope to allocate specific subroutines to each NDF step. Preparing the initial SSR always has a feedback effect that results in further refinement of the Subroutine Hierarchy and the NDF.

To summarize, the Software Status Report contains all of the steps from the NDF and all of the assigned subroutines from the hierarchy, along with the development status for each subroutine and step. With automated support, highly objective status reports are easily generated from this data base. Technical and administrative management are provided accurate visibility into the status of the total software development.

Figure 3 shows the Software Status Report for a typical step from an Implementation Plan. Figure 4 provides a brief description of each of the fields on the report. Figure 5 contains a Management Summary for one of the paths on the NDF. Figure 6 shows an Overall Management Summary covering all paths of the project. Figure 7 describes the fields and columns of the Management Summary.

X. Conclusion

In the 1980's large scale software developments are still too costly, unpredictable and too frequently unsuccessful. It is contended that management, and in particular management's implementation planning and management approach has a greater influence on the project's outcome than any other single factor. Yet, no standardized large scale software detailed planning and implementation methodology is in effect.

An implementation planning and management methodology that has been successful in each of its usages has been briefly described in this paper. The methodology covers the high level and the detailed day-to-day working level of software development. The methodology influences all aspects of a large scale software development, but most importantly it provides competent management with the tools necessary for managing a complex large scale software development. The benefits flowing from the methodology are numerous and include:

- o Improved management visibility
- o Improved customer visibility

- o More objective statusing
- o Reduction of impact of changes
- o Elimination of most test driver costs
- o Improved programmer morale
- o More easily maintained product
- o Elimination of separate software integration phase
- o A meaningful plan
- o Easily automated status reporting
- o The NDF provides the outline for a Test Plan
- o Complete accountability and cross checks
- o Algorithmic assignment of schedule dates and manpower
- o Greatly improved probability of project success

It is perhaps appropriate to highlight the fundamental differences between this methodology and others that may seem similar. These differences are usually in the following areas:

1. The emphasis on demonstrability.
2. The emphasis on meaningfulness as achieved by allocating the design (Subroutine Hierarchy) to the operational requirements (NDF) as the crux of the status report (SSR).
3. The emphasis on decomposition of the system to a very fine level of granularity (all individual subroutines) at an early stage as a component in preparation of the plan.
4. The emphasis on an investment in sufficient analysis prior to preparation of the plan so that a truly meaningful plan can be prepared.
5. The emphasis on "statistical equivalence" resulting from the decomposition granularity and using this equivalence to obtain objective statusing as well as algorithmic assignment of schedule dates and manpower assignments.
6. The emphasis on a single continuously cycling system with small pieces (steps) continuously being integrated with the system under development, thereby effectively eliminating a separate integration phase, as well as most test drivers.
7. The emphasis on utilizing the plan as a working document at multiple levels of the project and to have the plan evolve based upon development feedback.

Other benefits and other differences can be identified, but the above summaries should be sufficient. The overriding conclusions are that standardization in the area of implementation planning and management should exist, and that a candidate methodology purported to be successful is available. It is in the best interest of all large projects to investigate the POSIT methodology in order to establish whether it genuinely improves the success probability of large scale software developments. If it does, it will surely merit consideration as the basis of an implementation planning and management standard.

SYMBOLIC NAME										DESCRIPTIVE NAME	CLASS /STEP	LINE NUM	REFER NUM	
1	2	3	4	5	6	7	8	9	*					
.	CSTMAC	ACTIVE MODE PREDICTOR	C90	287	
.	CDBCOM	DOUBLE INTEGER COMPARE		288	197
.	CSHDGB	HSD BIT ACQUIRE		289	195
.	CDBADD	DOUBLE INTEGER ADD		290	277
.	CSTMEV	EVENT PREDICTOR	C50	291	
.	CDBADD	DOUBLE INTEGER ADD		292	277
.	CSHDGB	HSD BIT ACQUIRE		293	195
.	CDBSUB	DOUBLE INTEGER SUBTRACT		294	284
.	CDBCOM	DOUBLE INTEGER COMPARE		295	197
.	CSRCRP	RECALL QUEUE RESPONSE PROC.	C90	296	
.	CSCHEK	STATUS CHECK		297	258*
.	CSSUPD	SUSPEND RESPONSE PROC.	C130	298	
.	CSTMSU	MODEL RE-ADJUST	C130	299	
.	CDBCOM	DOUBLE INTEGER COMPARE		300	197
.	CDBADD	DOUBLE INTEGER ADD		301	277
.	CDBSUB	DOUBLE INTEGER SUBTRACT		302	284
.	CSOCLO	CONTROL CENTER RESPONSE PROC.	C60	303	
.	CSHDGB	HSD BIT ACQUIRE		304	195
.	CSCHEK	STATUS CHECK		305	258*
.	CSMOUP	MODE CHANGE RESPONSE PROC.	C100	306	
.	CSHDGB	HSD BIT ACQUIRE		307	195
.	CHXASC	HEX TO ASCII CONV.		308	217
.	GEFMSG	DISPLAY EVENT MESSAGE		309	31
.	CSCHEK	STATUS CHECK		310	258*
.	CSTIRM	RADIATION TIME TEST		311	281*
.	CSTWOV	WINDOW OVERRIDE RESPONSE PROC.	C120	312	

Figure 1. Sample Page From Subroutine Hierarchy

STEP: 010 Transfer Operator Entry to SYMBIONT

DESCRIPTION:

This step provides an operator entered directive to the Symbiont

INTENT:

This step demonstrates that an operator entered directive is placed in an SSB and entered in the Symbiont Queue. This step simulates the eventual LMC, LAN Handler, IDM to Symbiont interface. This also provides a basic buffer and queue management mechanism.

SOURCE:

AUTHOR: A. Spinak
 DUE DATE: 03/02/82
 COORDINATOR: T. GREER
 STATUS: DATE TYPE
 DSGN RVU 02/09/82 TEAM
 TEST 03/11/82 ACCEPTED
 ANOMALIES 0

NOTES:

Most or all of this step's software is in the nature of a temporary workaround or Environmental Interface Module (EIM). To prove, the SSB and the Symbiont Queue should be checked. No queue boundary conditions will be demonstrated.

SEGMENTS	DESCRIPTION	CL	DATE	DESIGN PERSON	SI	DATE	CODE PERSON	ST	QA	CMP	LINES
SIOINP	OPERATOR 2 SPT SIMULATION	I	02/01/82	MJB	*	02/11/82	MJB	*	*	*	21
SIOPOM	PARSE OPERATOR MESSAGE	I	02/02/82	MJB	*	02/11/82	MJB	*	*	*	21
SIOPAK	PACK CD SSB	I	02/03/82	TO	*	02/11/82	TO	*	*	*	30
SIOCBB	LOAD TEST COMM BUFF BLOCK	S									0
SIOSSB	LOAD TEST SSB BLOCK	S									0
SIOLNK	CHANGE LINK ID	S									0
ORQENQ	PLACE NODE ON QUEUE	I	02/02/82	TCG	*	02/16/82	MJB	*	*	*	51
ORQINI	INIT FREE Q NODE POOL	I	02/05/82	TCG	*	02/16/82	TCG	*	*	*	14
OPOPQU	POP FREE QUEUE NODE	I	02/05/82	MJB	*	02/16/82	TO	*	*	*	15
GBFOUE	I/F TO QUEUE ENTRY	I	02/02/82	TO	*	02/12/82	TO	*	*	*	35
GQINIT	I/F TO INIT QUEUES	I	02/04/82	TO	*	02/12/82	TO	*	*	*	14
OPSHQU	PUSH FREE QUEUE NODE	I	02/05/82	MJB	*	02/12/82	MJB	*	*	*	16
SPINIT	SPT INITIALIZATION	I	02/10/82	TCG	*	02/15/82	TCG	*	*	*	42
GBINIT	I/F TO INITIALIZE BUFFERS	I	02/22/82	MJB	*	03/05/82	TO	*	*	*	58
GBPOOL	I/F TO RELEASE BUFFER	I	02/22/82	MJB	*	03/05/82	MJB	*	*	*	55
GETBUF	I/F TO GET BUFFER	I	02/22/82	MJB	*	03/05/82	MJB	*	*	*	56

Figure 3. Sample Page From Software Status Report - step 010

STEP: NUMBER AND NAME OF THE STEP

DESCRIPTION:

DESCRIPTION OF THE SYSTEM FUNCTIONS AND CAPABILITIES IMPLEMENTED IN THIS STEP AND A CONCISE SUMMARIZATION OF THE DEMONSTRATIONS TO BE TESTED' (UP TO 3 LINES)

INTENT:

EXPLANATORY INFORMATION THAT IS USEFUL IN INTERPRETATION AND COMPREHENSION OF THE STEP. (UP TO 5 LINES)

SOURCE: REFERENCE TO THE REQUIREMENTS MET BY THIS STEP.

AUTHOR: ORIGINATOR OF THIS STEP INPUT.

DUE DATE: DATE WHEN THE STEP WILL BE COMPLETED AND READY FOR ACCEPTANCE.

COORDINATOR: COORDINATOR OF THE IMPLEMENTATION OF THE STEP AT THE DETAIL LEVEL.

STATUS: DATE TYPE

DSGN RVU: DESIGN REVIEW DATE TYPE OF REVIEW (TEAM, CDE OR PROG)

TEST STATUS DATE TYPE OF TESTING (CHECKOUT OR ACCEPTED)

(IF TYPE IS BLANK, DATE IS PLANNED DATE)

(ELSE DATE IS ACTUAL DATE)

ANOMALIES: NUMBER OF OUTSTANDING ANOMALIES

NOTES:

NOTES WHICH WOULD BE HELPFUL IN EXPLAINING THE STEP AND ITS IMPLEMENTATION. (UP TO 5 LINES)

THE SEGMENT COLUMNS ARE AS FOLLOWS:

STEP NO - STEP NUMBER

SEGMENTS - SEGMENT NAMES

DESCRIPTION - SEGMENT DESCRIPTION

CL - CLASSIFICATION CODE (I-IMPLEMENT, S-STUB, U-UNDEFINED)

DESIGN DATE - PLANNED DATE OF DESIGN COMPLETION

DESIGN PERS - INITIALS OF PERSON RESPONSIBLE FOR THE SEGMENT'S DESIGN

DESIGN ST - DESIGN STATUS (* IF COMPLETED)

CODE DATE - PLANNED DATE OF CODE COMPLETION

CODE PERSON - INITIALS OF PERSON RESPONSIBLE FOR CODING OF THE SEGMENT

CODE ST - CODE STATUS (* IF COMPLETED OR IF REJECTED)

QA - QA STATUS CODE (* IF ACCEPTED)

CMP - COMPILATION CODE (* FOR CLEAN COMPILE)

LINES - NUMBER OF LINES IN ACCEPTED SEGMENT

Figure 4. Software Status Report Description

SOFTWARE STATUS REPORT MANAGEMENT SUMMARY

STEP NUM	STEP NAME	C	UNIQ	CUM	DESGN	CODE	CMP	TEST	ACCEPT	DUE DATE	LINES
C010	Initialize Command Task		14	14	14	14	14	14	14	05/14/82	721
C020	Generate Command File		8	22	8	8	8	8	8	06/18/82	514
C030	Accept & Check Incoming Block		12	34	12	12	12	12	12	07/23/82	480
C040	Process Control & Status Block		6	40	6	4	0	0	0	09/10/82	0
C050	Process Event Block		5	45	0	0	0	0	0	10/22/82	0
C055	Process Monitor & Control Bloc		1	46	0	0	0	0	0	11/05/82	0
C060	Transmit Block to CPA		10	56	0	0	0	0	0	12/03/82	0
C070	Transmit File to CPA		17	73	0	0	0	0	0	01/21/83	0
C080	Recall File Directory		2	75	0	0	0	0	0	03/04/83	0
C090	Attach File to Queue		7	82	0	0	0	0	0	03/18/83	0
C095	Simulate Real Command Files.		1	83	0	0	0	0	0	04/08/83	0
C100	Initiate Command Radiation		2	85	0	0	0	0	0	05/27/83	0
C105	Suppress, Restore Acknowledge		1	86	0	0	0	0	0	06/17/83	0
C110	Modify Standards-and-Limits Ta		20	106	0	0	0	0	0	07/01/83	0
C115	ODR Dump.		1	107	0	0	0	0	0	07/15/83	0
C120	Transmit Additional Directives		16	123	0	0	0	0	0	08/12/83	0
C125	Verify Link Configuration.		1	124	0	0	0	0	0	09/09/83	0
C130	Suspend, Abort, Resume Radiati		9	133	0	0	0	0	0	09/16/83	0
C140	Verify Command Bits		10	143	0	0	0	0	0	09/23/83	0

TOTALS			143	143	40	38	34	34	34		1715
PERCENTS				100	27	26	23	23	23		
PROJECTION											7213

Figure 5. Command Path Summary

GRAND TOTALS FOR EACH PATH

PATH LETTER	UNIQ	CUM	DESGN	CODE	CMP	TEST	ACCEPT	DUE DATE	LINES
O PATH TOTALS	199	199	109	106	105	99	99		4166
A PATH TOTALS	52	261	60	58	55	55	55		2563
C PATH TOTALS	143	404	40	38	34	34	34		1715
D PATH TOTALS	6	410	0	0	0	0	0		0
E PATH TOTALS	11	421	0	0	0	0	0		0
K PATH TOTALS	118	539	36	26	22	21	21		925
L PATH TOTALS	192	731	0	0	0	0	0		0
P PATH TOTALS	3	734	3	3	0	3	3		0
S PATH TOTALS	2	736	0	0	0	0	0		0
T PATH TOTALS	51	787	14	14	14	14	14		521
U PATH TOTALS	21	808	9	7	7	7	7		216
V PATH TOTALS	1	809	0	0	0	0	0		0
Z PATH TOTALS	10	819	10	10	10	10	10		419
TOTALS	819	819	281	262	247	243	243		10525
PERCENTS PROJECTION		100	34	31	30	29	29		35473

Figure 6. Management Summary

- Grand Totals

SOFTWARE STATUS MANAGEMENT SUMMARY
DESCRIPTION OF THE SUMMARY
09/17/82

THE MANAGEMENT SUMMARY STATUS SHOWS THE COMPLETION COUNTS AND PERCENTAGES FOR EACH STEP AND FOR THE OVERALL EFFORT. THE STEPS ARE LISTED IN SEQUENTIAL ORDER ALONG WITH THEIR NAMES. THE TERM SEGMENT USED BELOW IS ESSENTIALLY SYNONOMOUS TO SUBROUTINE.

A DESCRIPTION OF EACH COLUMN IS AS FOLLOWS:

- C - THIS STEP WAS UPDATED BUT NOT OUTPUT, THE SUMMARY MAY BE INCORRECT
- UNIQ - # OF UNIQUE SEGMENTS IN THIS STEP
- CUM - CUMULATIVE COUNT OF THE SEGMENTS UP THROUGH THIS STEP
- DESGN - # OF SEGMENTS THAT HAVE BEEN DESIGNED REVIEWED IN THIS STEP
- CODE - # OF SEGMENTS THAT HAVE BEEN CODED IN THIS STEP
- CMP - # OF SEGMENTS THAT HAVE BEEN CLEAN COMPILED IN THIS STEP
- TEST - # OF SEGMENTS THAT HAVE BEEN TESTED IN THIS STEP
- ACCI - # OF SEGMENTS THAT HAVE BEEN ACCEPTANCE TESTED IN THIS STEP
- DUE DATE - DUE DATE OF THIS STEP
- LINES - # OF LINES IN THE SEGMENTS WHICH HAVE BEEN ACCEPTANCE TESTED

THE TOTALS, PERCENTS AND PROJECTIONS COLUMNS ARE AS FOLLOWS:

- TOTALS - THE SUMMATION OF ALL THE STEP SEGMENTS FOR EACH COLUMN
- PERCENTS - THE PERCENTAGE OF THE TOTAL NUMBER OF SEGMENTS THAT HAVE BEEN COMPLETED
- PROJECTIONS - A PROJECTION OF HOW MANY LINES WILL BE CREATED IN TOTAL

Figure 7. Description of Management Summary

