

AD-A115 460

TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA

F/G 9/2

A STUDY OF EMBEDDED COMPUTER SYSTEMS SUPPORT. VOLUME VIII. ECS --ETC(U)

SEP 80

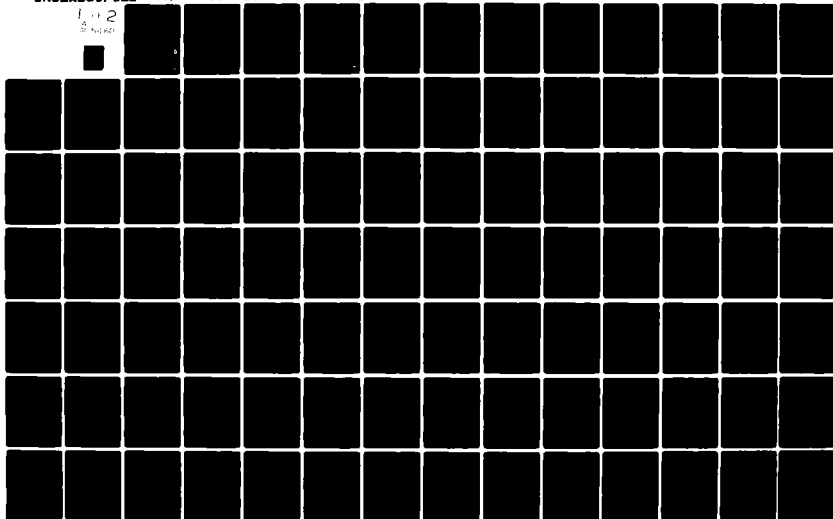
F33600-79-C-0540

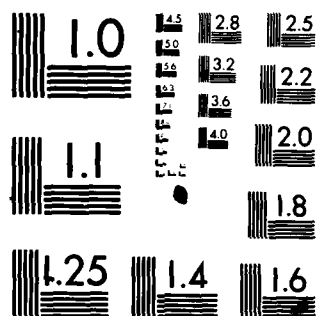
UNCLASSIFIED

TRW-34330-6003-UT-00-VOL-

ML

1-12
AUG 81





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

D

*A STUDY OF
EMBEDDED COMPUTER SYSTEMS SUPPORT
VOLUME VIII
ECS TECHNOLOGY FORECAST*

September 1980

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

CDRL 03
Contract Number F33600-79-C-0540

Prepared for
Air Force Logistics Command AFLC/LOEC
Wright Patterson AFB, Ohio 45433

UNCLASSIFIED/UNLIMITED

DTIC
ELECTE
JUN 11 1982
S
D
H

ONE SPACE PARK • REDONDO BEACH • CALIFORNIA 90278

TRW.
DEFENSE AND SPACE SYSTEMS GROUP

82 06 11 010

AD A115460

DTIC FILE COPY



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM												
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-4115460	3. RECIPIENT'S CATALOG NUMBER												
4. TITLE (and Subtitle) Embedded Computer System Support Volume 8-ECS Technology Forecast		5. TYPE OF REPORT & PERIOD COVERED Final												
7. AUTHOR(s)		6. PERFORMING ORG. REPORT NUMBER 47169H												
9. PERFORMING ORGANIZATION NAME AND ADDRESS TRW Systems Inc One Space Park Redondo Beach, CA 90278		8. CONTRACT OR GRANT NUMBER(s) F33600-79-C-0540												
11. CONTROLLING OFFICE NAME AND ADDRESS HQ AFLC/LOEE Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS												
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE Sept 1980												
		13. NUMBER OF PAGES 124												
		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)														
18. SUPPLEMENTARY NOTES Recommendations pertaining to administrative and programmatic initiatives are presented in this volume. Companion documents dated October 1981, September 1980, same contract number.														
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <table border="0"> <tr> <td>Materiel Management</td> <td>ECS</td> <td>Computerized Systems</td> </tr> <tr> <td>Logistics Support</td> <td>Weapons Systems</td> <td>Avionics</td> </tr> <tr> <td>Logistics Management</td> <td>Technology Forecasts</td> <td></td> </tr> <tr> <td>Planning</td> <td>Program Management</td> <td></td> </tr> </table>			Materiel Management	ECS	Computerized Systems	Logistics Support	Weapons Systems	Avionics	Logistics Management	Technology Forecasts		Planning	Program Management	
Materiel Management	ECS	Computerized Systems												
Logistics Support	Weapons Systems	Avionics												
Logistics Management	Technology Forecasts													
Planning	Program Management													
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Abstract: This portion of the study focuses on technological forecasts in the support of Embedded Computer Systems. It evaluates such concepts as intercenter networks, computer nets, fiber optics communications, high order languages, standardization, built-in test, and operator-computer interaction. (See related volumes: LD 47169A, B, C, D, E, F, G, J, K, and L). Conclusions: See volume 1 (LD 47169A).														

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ITEM 20 (con't)

Recommendations: . See volume 1 (LD 47169A).

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



***A STUDY OF
EMBEDDED COMPUTER SYSTEMS SUPPORT
VOLUME VIII
ECS TECHNOLOGY FORECAST***

September 1980

CDRL 03
Contract Number F33600-79-C-0540

Prepared for
Air Force Logistics Command AFLC/LOEC
Wright Patterson AFB, Ohio 45433



ONE SPACE PARK • REDONDO BEACH • CALIFORNIA 90278

TRW
DEFENSE AND SPACE SYSTEMS GROUP

FOREWORD

This volume is one of nine individually bound volumes that constitute the Phase II Final Report "Study of Embedded Computer Systems Support" for Contract F33600-79-C-0540. The efforts and analyses reported in these volumes were sponsored by AFLC/LOEC and cover a reporting period from September 1979 through September 1980.

The nine volumes are

<u>Volume</u>	<u>Title</u>
I	Executive Overview (CDRL 05)
II	Selected ECS Support Issues: Recommendations/ Alternatives (CDRL 02A)
III	Requirements Baseline: Aircrew Training Devices (CDRL 02A)
IV	Requirements Baseline: Automatic Test Equipment (CDRL 02A)
V	Requirements Baseline: Communications- Electronics (CDRL 02A)
VI	Requirements Baseline: Electronic Warfare (CDRL 02A)
VII	Requirements Baseline: Operational Flight Programs (CDRL 02A)
VIII	ECS Technology Forecast (CDRL 03)
IX	National Software Works Investigation (CDRL 04)



Accession For	☒ ☐ ☐
NTIS GRA&I	
DTIC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	
Special	
A	

CONTENTS

FOREWORD	ii
ABBREVIATIONS AND ACRONYMS	vii
1. INTRODUCTION	1-1
1.1 Productivity Increases	1-1
1.2 AFLC Interaction With DSARC and AFSC	1-3
1.3 Skill Centers	1-3
1.4 Historical Data	1-4
1.5 Categories of Embedded Computers	1-5
2. INTERCENTER NETWORKS	2-1
2.1 AFLC Requirements for Networking	2-1
2.1.1 Closed-Circuit Television (CCTV)	2-1
2.1.2 Distributed ISF	2-2
2.1.3 Operator-interactive Data Exchange	2-3
2.2 Present ALC Networks	2-4
2.3 Suggested AFLC-Wide Networks	2-4
2.3.1 Telephone Line System	2-4
2.3.2 Leased Land-lines	2-5
2.3.3 Video Network	2-5
2.3.4 Encryption	2-5
2.4 Communication Costs	2-5
3. DISTRIBUTED INTEGRATION SUPPORT FACILITIES	3-1
3.1 Advantages/Disadvantages of Distributed ISF's	3-1
3.2 Example of Distributed AISF	3-2
3.3 Conceptual Requirements	3-3
3.4 Recommended Feasibility Analysis	3-7
4. COMPUTER NETS	4-1
4.1 Distributed Computer Nets	4-2
4.1.1 Fixed Tasks in Independent Computers	4-2
4.1.2 Fixed Tasks in Interconnected Computers	4-2
4.1.3 Variable Tasking	4-2
4.1.4 Fault-tolerant Nets	4-4
4.2 Fault Tolerant Nets	4-4

CONTENTS (Continued)

4.3	Fiber Optics Communications	4-6
4.4	Effect on AFLC	4-6
5.	LARGE SCALE INTEGRATION-VERY HIGH SPEED INTEGRATED CIRCUITS	5-1
5.1	1980 State of the Art	5-1
5.2	VHSIC	5-1
5.3	LSI-VHSIC in Airborne Radars	5-2
5.4	Microprocessors in Electro-Optics	5-3
5.5	LSI-VHSIC in Electronic Warfare Systems	5-6
5.6	NAV-COMM Systems	5-6
5.7	Standard Microcomputer	5-7
5.8	Effect on AFLC	5-8
6.	HIGH ORDER LANGUAGES	6-1
6.1	Jovial: Present Language for ECS Software	6-1
6.2	Ada: Future Language for ECS Software	6-1
6.3	Atlas: Language for ATE Software	6-3
6.4	USAF Usage of HOL	6-4
6.5	Schedules	6-5
6.6	Effect on AFLC Operations	6-5
6.7	Recommendations to AFLC	6-8
7.	EMULATION OF ECS	7-1
7.1	Current Emulation Hardware	7-2
7.2	Current Emulation Software	7-3
7.3	Future Emulation Hardware	7-4
7.4	Future Software Support Systems	7-6
7.5	Recommendations to AFLC	7-7
8.	STANDARDIZATION	8-1
8.1	Standard Instruction Set Architecture	8-1
8.2	Standard Data Bus	8-3
8.3	Standard Software Tools	8-4
8.4	Standard High Order Language	8-5
8.5	Standard ATE	8-6

CONTENTS (Concluded)

8.6	Standard Host Computers	8-6
8.7	Standard Mathematical Models	8-6
8.8	Standard Operator-Interactive Software	8-7
9.	BUILT-IN TEST	9-1
9.1	Technology	9-1
9.2	Economics of Bit	9-2
9.3	Recommended AFLC Action	9-3
10.	OPERATOR-COMPUTER INTERACTION	10-1
10.1	The Problem	10-1
10.2	State of the Art	10-2
10.2.1	Word Processing	10-3
10.2.2	Pictorial	10-3
10.3	Suggested AFLC Actions	10-4
APPENDIX A: TYPICAL DATA RATES FOR DISTRIBUTED AISF		A-1
APPENDIX B: DATA COMMUNICATION COST ESTIMATES		B-1
APPENDIX C: STANDARD MICROCOMPUTERS		C-1
APPENDIX D: SIGGRAPH STANDARD		D-1
APPENDIX E: HIGH ORDER LANGUAGES		E-1
APPENDIX F: COMTEC - 2000 SUMMARY		F-1
REFERENCES		R-1

ILLUSTRATIONS

3-1.	Hypothetical Example of a Distributed ISF	3-3
3-2.	Example of Distributed ISF for JTIDS	3-6
5-1.	Growth in Electro-optical Computer Requirements	5-5
6-1.	Schedule for New USAF High Order Languages	6-2
6-2.	Block Diagram of Standardized Support System	6-7
7-1.	Example of User-Controlled Parallel Computer Emulation	7-5

ABBREVIATIONS AND ACRONYMS

AFIT	Air Force Institute of Technology
AISF	Avionics Integration Support Facility
ATD	Aircrew Training Device
ATE	Automatic Test Equipment
BIT	Built-In-Test
C-E	Communications-Electronics
DSARC	Defense System Acquisition Review Council
EW	Electronic Warfare
HOL	High Order Language
ICS	Interpretive Computer Simulation
IFF	Identification Friend/Foe
IM	Item Manager
ISF	Integration Support Facility
LRU	Line Replaceable Unit
LSI-VHSIC	Large Scale Integration-Very High Speed Integrated Circuits
MATE	Modular Automatic Test Equipment
MCF	Military Computer Family
MDS	Microprocessor Development System
MTBF	Mean Time Between Failure
OFP	Operational Flight Program
SM	System Manager

1. INTRODUCTION

The effect on AFLC of technological advances that will occur during the next ten years is driven by two key trends.

1. TRW envisions accelerated proliferation of microprocessors, avionics computers, and special-purpose processors within both flight and test equipment during the next ten years. Many subsystems will be equipped with internal circuit-board computers, thus displacing many stand-alone ECS's. The numerical availability of systems with internal ECS's will increase because of the reduction in connectors, power supplies, and electronic parts count.
2. Demographers expect the college-age population to be 2.5 million students less in 1990 than during the 1970's peak. The American Council on Education projects the 1985 population of 16-year olds as falling to 3.6 million from the 4.2 million peak. Recognizing AFLC's need to train 2000 new programmers (Reference 1-1) and an unknown number of computer technicians during the next ten years, TRW suggests that an increase in productivity will become essential if AFLC is to maintain the readiness of Air Force equipment.

As a result of the TRW technology forecast analysis, a set of five recommendations that can be implemented by management directives or administrative action are presented in Sections 1.1 through 1.5. In addition, detailed suggestions relating to nine technical areas are presented in Sections 2 through 10. These technical areas are introduced in Table 1-1 which also summarizes the recommended actions that stem from the technology analysis. The table also presents a statement of expected benefits to AFLC and a qualitative estimate of the cost.

Supporting and supplementary data on several of the technical areas discussed are presented in the references and in Appendices A through E.

1.1 PRODUCTIVITY INCREASES

AFLC can anticipate a reduction in the number of skilled people available at the same time as the Air Force usage of embedded computers increases. As a result, AFLC will be under pressure to increase the productivity of programmers and technicians. Productivity can be increased by standardizing instruction sets, high order languages,

Table 1-1. Impact of Embedded Computer Technology

Technology	AFLC Benefit	Recommended AFLC Action	AFLC Implementation Cost
Intercenter Networks	- Increased productivity - Low-cost training - Reduced duplication of software development - Permits central skill centers - Permits distributed ISF - Reduced paperwork	- Requirements analysis of teaching, software exchange, management information throughout AFLC - Implementation of multi-purpose, AFLC-wide net	Low
Distributed Integrated Support Facilities	- Low-cost software maintenance for distributed avionics - Concentrates specialists	Analyze economics of distributed versus collocated ISF, cost, availability, usage at each ALC, inter-center traffic	Low
Computer Nets	Avoids duplication in maintenance and software support	- Enforce AFLC needs in design - Create AFLC-wide plan for support of distributed avionics	Low
LSI-VHSIC	Reduced cost of testing	- Increased use of general-purpose test equipment - Standardization of embedded microcomputers	None
High Order Language and Emulation of ECS	- Lower maintenance cost for support software - High programmer productivity	- Assign Software Manager to maintain HOL, emulators, standard tools - Develop J73 and Ada tools, support system, emulators	Low
Standardization	- Large savings in software, ATE, and maintenance costs - Increased productivity of test technicians and programmers	- Develop problem-oriented dialects of J73 and Ada for simulation, emulation, and system test - Enforce use of standard HOLs in AFSC - Prepare AFLC-wide emulation plan	Medium
Built-in-test	Simplifies maintenance of potentially complex circuitry	- Enforce standardized instruction set, HOL, data bus - Develop standard tools, HOL dialects, models, operator-interface software, ATE	Low
Operator-Computer Interaction	- Increased programmer productivity - Increased user productivity	- Analyze and publish historical-ECS logistics cost data - Determine optimum mix of BIT/ATE for LSI VHSIC - Design ATE to verify BIT circuits - Establish skill center for operator-interactive software	Low

software tools, mathematical models, and test equipment. Productivity can be increased by the inter-ALC network that allows skill centers to communicate voice, digital data, and video to users throughout AFLC.

1.2 AFLC INTERACTION WITH DSARC AND AFSC

Standardization of hardware, software, and interfaces is in AFLC's best interest. Therefore AFLC should work with the Systems Command to analyze alternative designs early in the development cycle of those systems that use embedded computers. AFLC should develop its own life-cycle cost models of embedded computer hardware and software and should analyze the cost impact of design decisions on future AFLC costs. Where the Systems Command chooses alternatives that reduce their own cost at AFLC's expense, AFLC should be prepared to present cost data to the DSARC beginning at DSARC-1. The AFLC should also be prepared to fund alternative designs whose initial cost may be high but whose projected life-cycle cost, as calculated by AFLC, is lower. AFLC should rigorously identify departures from standardization in computer architecture, data bus, high order languages, models, etc. at DSARC reviews. Early pre-PMRT action is probably the most effective way of reducing long-term costs of logistic support of embedded computer systems. It is suggested that AFLC follow commercial practice and invest 2 to 5 percent of projected support cost in analyzing the economics and ability to support alternate designs from the total AFLC viewpoint prior to DSARC I and II.

1.3 SKILL CENTERS

AFLC should establish skill centers for several emerging new technologies in order to make maximum use of increasingly scarce hardware and software specialists. Skill centers would probably be beneficial in at least the following technologies:

1. Integrated Support Facilities, for development and verification of changes to operational software, especially in distributed avionics networks;
2. Support of distributed avionics networks and ECS applications to engines, IFF, radar, optics, air data, navigation, flight control, weapon delivery, electronic warfare, and cockpit displays (productivity would be improved if specialists in each application were concentrated at an ALC);

3. High order language;
4. Emulators and other software tools;
5. Mathematical models of aircraft, weapons, environment, sensors and actuators, mission models, data-reduction programs, etc;
6. Operator-computer interactive systems;
7. Management of the intercenter communication network; and
8. AFLC-initiated standardization.

Computer program exchange and coaching by experts are two of the important benefits to be derived by concentration of specialists at skill centers. Other direct benefits are cost savings from the minimization of delays, elimination of duplicate developments, and less travel.

AFLC should prepare formal plans for the development of these skill centers and supporting facilities. Schedules and budgets should be prepared and responsibility assigned to centers. These plans should support known production aircraft and probable modifications expected during the next 10 years. In most cases, the selection of the most productive choices is not obvious without a quantitative analysis of the alternatives. It is suggested that AFLC follow commercial practice and invest 2 to 5 percent of projected system cost to analyze the economics and productivity of AFLC internal systems and facilities (e.g., intercenter networks, distributed ISF, new test dialects) from the total AFLC viewpoint prior to selecting systems to be used from 1985 to the end of the century.

1.4 HISTORICAL DATA

AFLC should analyze and publish historical data on the cost of maintaining and supporting ECS hardware and software, ATE, built-in test, and trainers. AFLC should analyze and publish software and hardware failure records (MTBF and MTTR). The results should be in the form of predictive models that could be used in life-cycle cost studies. The models' results could be compared to industrial costs as a means of assessing the difference between the costs of organic and contract maintenance.

1.5 CATEGORIES OF EMBEDDED COMPUTERS

As digital flight computers proliferate into flight control, weapon delivery, display generation, engine control, IFF fusion, etc., the boundaries between OFP, EW, and C-E are becoming imprecise. Are digital flight control, digital radio sets, and engine controllers OFP? Are pre-flight test programs loaded into avionics computers or EW computers OFP? The artificial distinction between EW and other OFP is vanishing as IFF, radar, and optics software also become mission-dependent. A more realistic categorization of EW and OFP might distinguish:

- Unsupported ECS (such as air data computers) that nevertheless must be simulated in ISF's,
- Supported ECS whose software is mission-independent (such as radar signal processors),
- Supported ECS whose software is mission-dependent (such as radar data processors, and
- Supported ECS whose software is secure (such as EW and IFF fusion).

A more realistic categorization would probably influence AFLC's administrative planning during the next 10 years.

2. INTERCENTER NETWORKS

Networking technology makes it possible to interconnect the ALC's, AFLC headquarters, and other key sites to allow technical personnel to assist each other at all centers and to permit interchange of data and software (References 2-1 through 2-4). The described communication network emphasizes geographic dispersion and contains computers distributed throughout the ALC's. This communication network should not be confused with the computer nets described in Section 4, which are an ensemble of computers in an aircraft, a trainer, a C³ center, or a test set that are linked to perform a specific mission.

To implement the communication network, AFLC has the choice of purchasing equipment and forming a private network, leasing lines from a common carrier such as AT&T, SBS, Xerox, RCA, Western Union, etc., or using an existing network. Depending on the data rate, priority, and security requirements, the Autodin II, the ARPANET, or the DSCS-III satellite would be candidate communication links.

An existing experimental communications network and its usefulness to AFLC for transferring computer programs among centers are described in Volume VII; the National Software Works Investigation. That volume also addresses a postulated network application to future AFLC software support requirements.

2.1 AFLC REQUIREMENTS FOR NETWORKING

The AFLC requirements for networking are in the areas of closed-circuit television, distributed ISF, and operator-interactive data exchange as described in Sections 2.1.1 through 2.1.3.

2.1.1 Closed-Circuit Television (CCTV)

The problem of training more than 200 programmers per year plus perhaps as many technicians, (Reference 1-1), could be greatly simplified by installing video classrooms at each ALC. Classes could be taught, for example, by senior personnel at an ALC or by professors at the Air Force Institute of Technology (AFIT). Each classroom should be equipped with receive-only video and 2-way voice. One classroom at each ALC should be equipped as a studio for originating programs, and one or more

video tape recorders could be used for videotaping and playback of classes. In addition to classroom use, the network could also be used as a "picture-phone" for conferences and as a remote blackboard. The wide-band communication channel could be used for high-speed document and data exchange, and video discs might well become an archival storage medium for AFLC.

AFLC would have to determine the number of nodes required, the number of classrooms needed at each node, and the number of classrooms to be operated simultaneously. Then, discussions with equipment vendors and lessors of lines will define the costs. As a preliminary estimate, for three classrooms (one of which serves as a studio) at each of five ALC's and a studio at Dayton, which can be used by HQ AFLC and AFIT, purchase cost of electronics equipment is approximately \$330,000 plus \$15,000/year maintenance. These estimates assume commercial TV with no customizing to transmit simultaneous data on sidebands. Commercial TV error rates are acceptable, and there is no time-delay constraint.

2.1.2 Distributed ISF

Section 3 describes the advantages and characteristics of distributed ISF's, whose embedded and host computers are located at several different ALC's and the calculation of intercenter data rates and time delays. The estimated data rate is 56 kbps each way with a round-trip delay not to exceed 20 milliseconds for most real-time ISF's. The net can be implemented by land-line because the time-delay of satellite links is too long for ISF's. Full-time lease costs are estimated to be \$11,000/month between center pairs (Appendix B). The requirements analysis suggested in Section 3.4 should yield a more exact estimate of the costs of just the distributed ISF service and that service combined with other network services. Although some common components and software may be used in ISF's and trainers, it is unlikely that ISF elements will be linked to trainer elements in real time before 1990.

2. 1. 3 Operator-interactive Data Exchange

An operator-interactive network connecting the ALC's could store and transmit the following information:

- AFLC management information: schedules, budgets, forecasts of various projects, items, and weapon systems; and DTC cost data, and cost models.
- Configuration management: status, contractors, approved configuration of hardware and software items, and computer-stored drawings with a history of changes.
- Software: computer-stored listings and users manuals for software tools, approved operating systems for embedded computers, ATE programs, graphics programs and data, stimulus generators, mission models, weapon models (e.g., bombs, guns, missiles), airplane models, environment (gravity and atmospheric) models, library of commonly-used flight programs, algorithms, EW mission updates (linked to operational bases), and servicing instructions for embedded computers and other ALC equipment.
- Inventory control: parts quantities at each location, reordering instructions, usage rate, projected usage rate, etc.
- Scientific computing: service to the ALC's for scientific computation.

A data network will reduce the incentive at each ALC to duplicate developments by permitting more rapid transfer of urgent information than would the US mail, permitting both parties to discuss material during the transfer process (i.e., allow a few senior experts to instruct many junior people), and transferring models and software tools with explanations. Shared use of tools will be especially valuable as languages (on-board languages and test languages) are standardized. Even if different host computers are used at different centers, the tool logic can be standardized and verified. AFLC should analyze the need for an extension of this network to the operational airbases to be used for distribution of flight and test software and for the distribution of training (ATD) software.

The data rate requirements depend on the information to be transmitted and on the number of terminals at each ALC, both of which must be determined by, or on behalf of, AFLC.

2.2 PRESENT ALC NETWORKS

The following networks are known to interconnect some, or all, ALC's.

- RJET: telephone-line terminals using an IBM 360 computer at Ogden ALC (transmission of ATE utility programs).
- CREATE: telephone-line terminals using a Honeywell host computer at WPAFB (scientific computation).
- INFOCEN: telephone-line system using an IBM 370 computer at WPAFB (configuration control of software).

These three networks might be the basis for an expanded operator-interactive network, as discussed in Section 2.1. Some of these networks might be retained, others might be included in a larger system, and still others might be closed, depending on the economics.

2.3 SUGGESTED AFLC-WIDE NETWORKS

Networks suggested for incorporation into an AFLC-wide network are a telephone line system, leased land-lines, and a video network. These alternate implementations are reviewed in Sections 2.3.1, 2.3.2, and 2.3.3.

2.3.1 Telephone Line System

The telephone line system suggested is an expansion of RJET, CREATE, or INFOCEN in order to perform more of the functions described in Section 2.1.3. The equipment at each node can be of common manufacture, as in the existing systems, or can be different, with computer switches used at each node as communication buffers. The requirements would include the number of nodes (ALC's, AFLC, HQ, AFSC terminals, Pentagon, other), the actual number of users at each node in 1980, and the number of users forecast for 1985, 1990, and 1995. The traffic could be analyzed with a simulation of internodal traffic to determine the average delay, the delay dispersion, the average data rate, and the data rate dispersion for each user forecast.

The cost analysis would include a calculation of the cost of hardware purchased for each node (in addition to existing equipment), the cost of software development to meet the 1985, 1990, and 1995 traffic loads, and the cost of maintenance. The cost analysis would also include an

estimate of cost savings and cost avoidance that would result from the network and a comparison with existing methods of transferring information.

2.3.2 Leased Land-lines

The leased land-lines concept is envisioned as a 50 kbps system, perhaps using some equipment from the existing networks. Expanded service would be possible, including distributed ISF (Sections 2.1.2 and 3) and slow-scan television for classroom use. The traffic and cost analyses are as described in Section 2.3.1.

2.3.3 Video Network

The proposed video network would be a 6-plus mbps link which would permit real-time television concurrently with data transfer. Several variations would be analyzed.

1. Data and video sequentially, allowing use of commercial CCTV equipment.
2. Data modulated onto video sidebands, permitting simultaneous TV and data, but requiring custom equipment.
3. Leased-lines in an all-up system with phased-growth and use of DSC-III or commercial leased services.
4. Private-line system, all-up and phased growth.

The corresponding traffic and cost analysis are described in Section 2.3.1.

2.3.4 Encryption

TRW recognizes that certain intercenter data must be secure. A cursory investigation of available encryptors shows that KG and KT devices are available at 56 kbps, 1 mbps, and 20-plus mbps. System costs increase as data rates rise. The communication analysis should determine specific requirements for encryption and the availability of NSA devices. It is possible that encryptor costs will not be chargeable to AFLC.

2.4 COMMUNICATION COSTS

Many services are available, ranging from leased telephone lines, at the one extreme, to private-line microwave or satellite, at the other.

Tariffs are complex and should be analyzed in detail as design alternatives mature; approximate rates are given in Appendix B. At 56 kbps, leased AT&T digital land-lines will cost \$11,000/month from WR-ALC to SM-ALC including local terminations and connections. This line would transmit very slow-scan TV (2 minutes) or digital data. Lower tariffs are available, depending on daily usage. At 1.25 mbps, dedicated satellite service costs approximately \$48,000/month between the same points, including leased terminals. The satellite line would transmit slow-scan TV (8 scans/second) and data, or wide-band data. Lower tariffs are possible depending on daily usage. For less than three hours of usage per day, a toll-tariff may be cheaper than a dedicated line. A detailed analysis of the service required and the tariff alternatives should be performed.

3. DISTRIBUTED INTEGRATION SUPPORT FACILITIES

The technology of the 1980-1990 period will permit distributed Integration Support Facilities (ISF), with embedded computers and host computers located at several ALC's, interconnected by digital communication links (Section 2). Thus, the simulation of each item (e.g., the radar, weapon delivery, engine-control, IFF-fusion, navigation, cockpit display or ground-control-center computers) can be located at the site of the item manager and the simulation can be directed from a host computer at the site of the weapon system manager. Simulation runs will be made in support of hardware integration, software maintenance, and hardware maintenance.

3.1 ADVANTAGES/DISADVANTAGES OF DISTRIBUTED ISF's

The advantages of distributed ISF's are as follows:

1. Distributed ISF's allow shared use of expensive simulation facilities that involve embedded computers, sensors, actuators, and drivers. One or more item simulations can be built, depending on the anticipated ISF schedules for each weapon system. Shared usage will be especially valuable when MIL-STD-1750/MIL-STD-1862 computers and J73/Ada language become standardized. Few changes will be needed to convert an item simulation from one weapon system to another and, even if several item simulations are built, the same personnel and test equipment can support them all.
2. Hardware problems that are difficult to diagnose with test sets can be debugged in a dynamic environment that can reproduce the failure symptoms.
3. Development cost of identical ISF and ATD elements can be reduced by developing devices and software common to both. Most of the elements of an ATD exist in an ISF except for the operator's station, out-of-the-window display, failure driver software, and proficiency analysis software. Despite the use of common elements, it is unlikely that a single ISF would interact in real time with one or more trainers.
4. The distributed ISF is administratively convenient because the item managers control only the simulation of their devices whereas the weapon systems manager controls the simulation of the total weapon system. The latter need not be expert in the subsystem technology.

The disadvantages of distributed ISF's include:

1. Cost and numerical availability of the communication system.
2. Management complexity of scheduling and configuration control between SM and IM's.
3. Availability of the item laboratories to support extensive integration testing (increased workload).

3.2 EXAMPLE OF DISTRIBUTED AISF

Figure 3-1 shows an example of a distributed ISF that might be constructed in the 1980's. An embedded radar computer is installed at WR-ALC with associated live and simulated sensors, and an embedded EW computer is installed at the same ALC with associated sensors. Each host computer stimulates its sensors in accordance with a mission scenario and collects test results. The embedded computers (and perhaps some sensors) within an ALC are interconnected by a real or simulated MIL-STD-1553B data bus. The outputs of each embedded computer are distributed to the displays, to weapons, and to other computers on the bus.

The embedded navigation computer with real and simulated sensors and the embedded jet engine computer are installed at OC-ALC. A host computer drives both and collects results. Both embedded computers are connected to a MIL-STD-1553B data bus.

The embedded display computer and weapon delivery computer test beds are installed at SM-ALC (the site of the Weapon Systems Manager in this example) and interconnected by a MIL-STD-1553B data bus. The host computer directs the entire simulation, including the remote test beds and drives the two embedded computers at SM-ALC. Outputs of the embedded computers (such as weapon-launch discretes) are passed to other computers and to the host.

A digital communication link interconnects the three centers. The only data transmitted between centers are the data that normally flow (on the aircraft data bus) from one computer cluster to the other, plus the interhost synchronization data. The latter are the data required to synchronize the mission scenarios at each ALC. The simulation results are stored in each host. After each run, the raw or partly-reduced data are

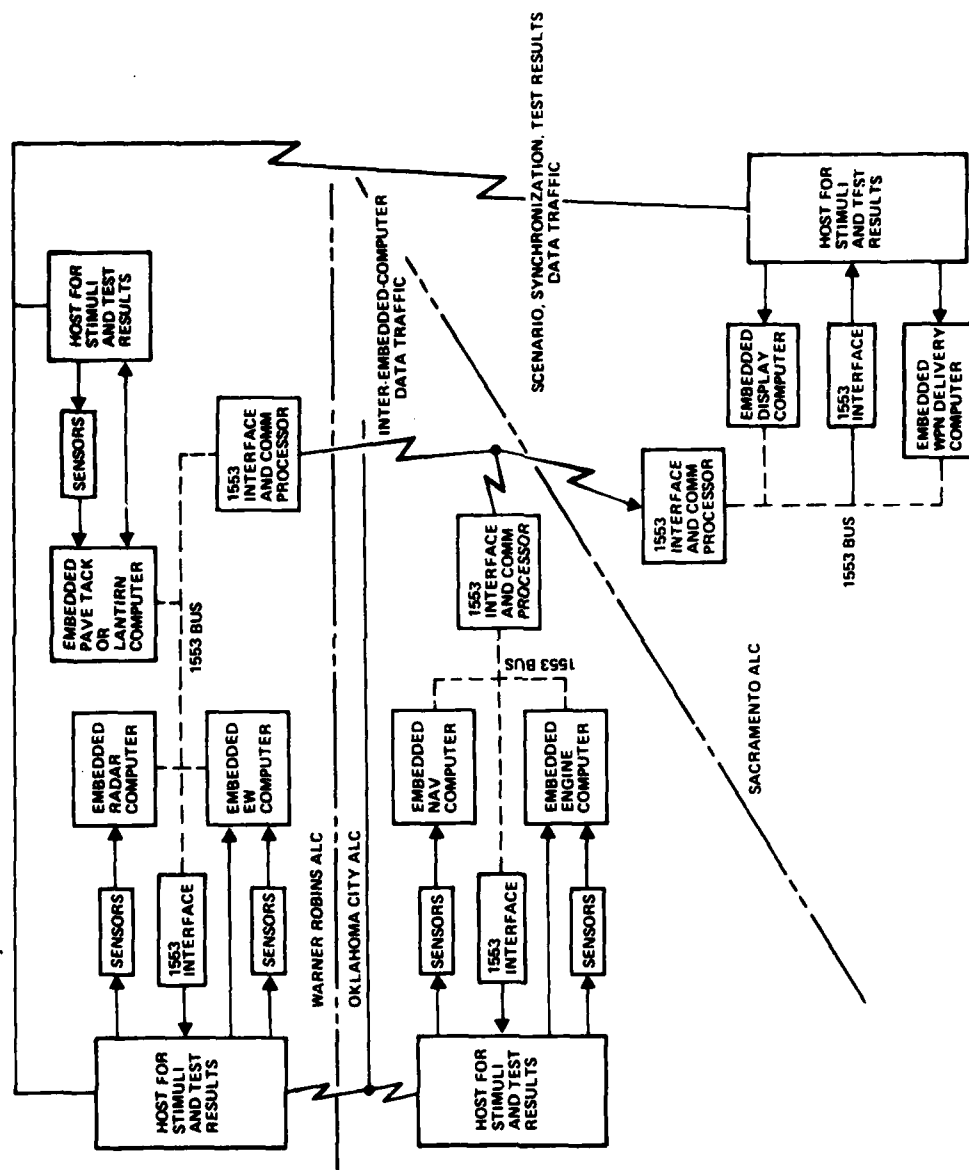


Figure 3-1. Hypothetical Example of a Distributed ISF

transmitted to the weapon-system host for final analysis. All sensor display data would not be transmitted to a single cockpit location. Instead, the sensor displays would be retained at each ALC (e.g., radar display and operator at WRALC), thus greatly reducing the inter-center data flow. Most simulation runs will utilize the ECS and hosts at a single ALC. Occasionally, when the full weapon system must be simulated, runs will be made with all centers intercommunicating. A full demonstration should be made of the feasibility of this distributed ISF concept including the effects of the different compromises.

A feasibility analysis of the bandwidth adequacy of currently available telecommunication links is presented in Section 3.4. Appendix A shows the parameters passing from the navigation computer to the surveillance radar computer for a hypothetical distributed E-3A Integration Support Facility (ISF). The intercenter data rates for this example are calculated in Appendix A based on a typical update cycle. Each parameter is assumed to be transmitted in a 32-bit word including label, parity, etc. The maximum burst rate from these data is 18.7 kbps. The shortest allowable delays occur when a parameter, calculated near the end of a computing cycle, is needed early in the next cycle. Some high data rate parameters cannot tolerate communication delays greater than 10 msec. Therefore, in order to use commercial land-line communications, it is suggested that the Nth computing cycle use some of the results from the (N-2) cycle for those parameters calculated near the end of a cycle. The remaining parameters can come from the (N-1) cycle. The delay-time simulation at one ALC (described in Section 3.4, Item 4) will determine if this approximation is acceptable. Appendix A also includes the data rates between a PAVE TACK pod and an F-111 or an F-4 AISF. It concludes that 56 kbps each way is satisfactory for data and communication protocols if displays based on remote sensor data are not required at any location. Appendix A also presents the calculation of the data rate between a SRAM AISF and an FB-111 AISF showing that 56 kbps is satisfactory. These rates should be verified by AFLC, using specific embedded computers at specific ALC's, if a distributed ISF is to be considered.

Because data rates and the delays are important to the economic feasibility of distributed ISF's, an experiment should be performed at one ALC by monitoring and delaying intermodule traffic (Section 3.4, Item 4).

The feasibility analysis does not include the effect of possible error rates in the communication link. This problem would be resolved during the feasibility demonstration recommended in Section 3.4. For JTIDS, planning has begun for sharing resources at three different locations as shown in Figure 3-2. Further explanation can be found in Volume V, Communications-Electronics.

3.3 CONCEPTUAL REQUIREMENTS

In summary, the following are some of the conceptual requirements for a distributed ISF.

1. Bandwidth necessary for real-time simulations. Based on the described provisional analysis, a 56 kbps channel is satisfactory for the simulated intercenter portion of the MIL-STD-1553B data bus, including overhead traffic and including 4 kbps for the interhost data.
2. Shortest allowed time delay. On the simulated intercenter portion of the MIL-STD-1553B communication channels the shortest allowed time delay is 8 milliseconds, although most data can tolerate a 23 millisecond delay. The allowable time delay on the interhost data link is not critical.
3. Timing requirements among the hosts. One millisecond is probably adequate and can be done with local time sources.
4. Post-run data transmitted between centers for analysis.
5. Usage times. Estimates should be made of the usage of each ISF element and of the usage of the entire network. Most software preparation, debugging, and hardware interface testing will probably be done at a single ISF without need for the network. Multi-center runs will test inter-computer interfaces and will be infrequent. If ISF elements are to be shared with an ATD, the time allocations for each should be defined.

The detailed functional analysis must be determined by AFLC prior to the feasibility analysis described in Section 3.4.

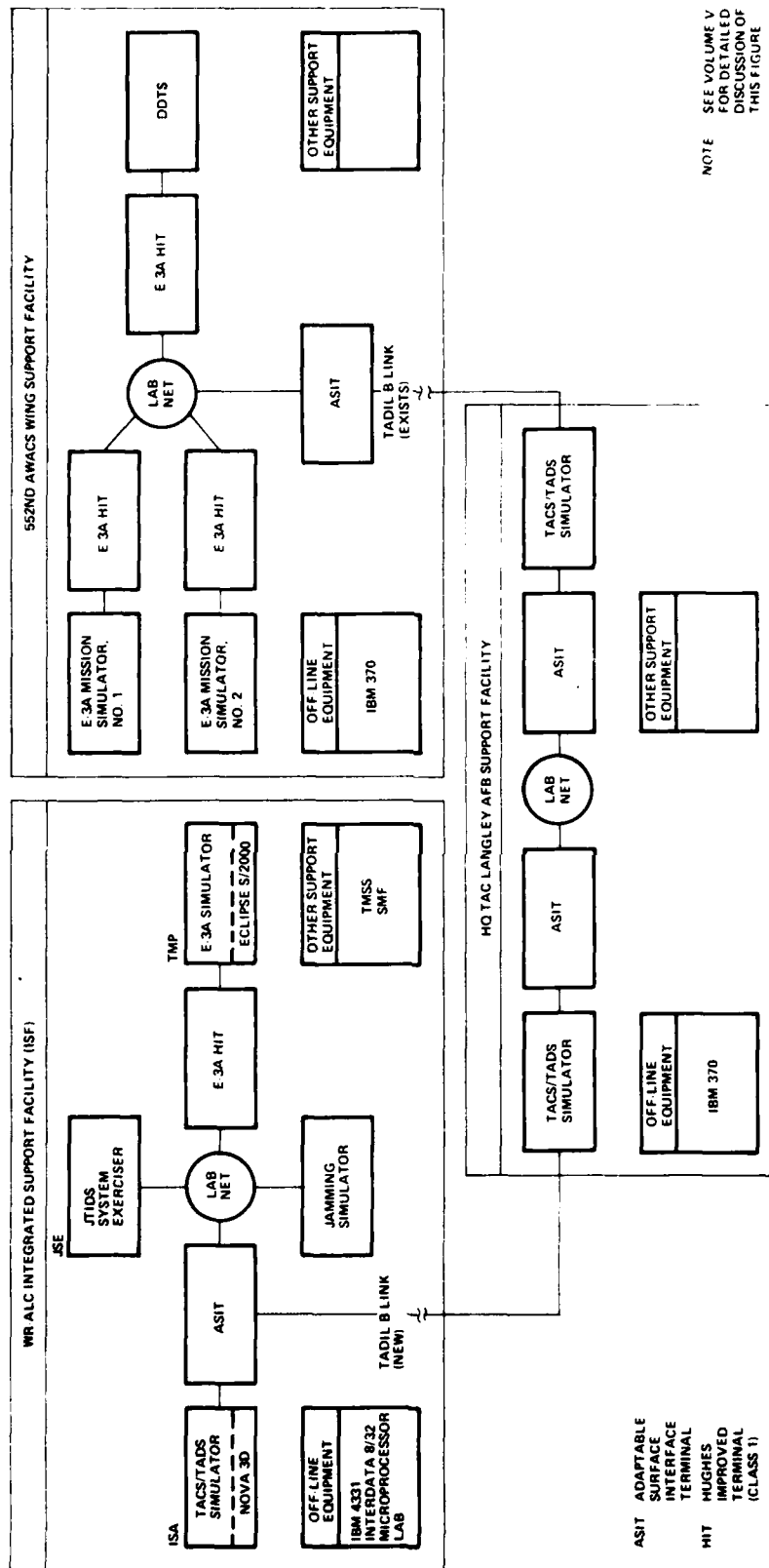


Figure 3-1. Example of Distributed ISF for JTIDS

3.4 RECOMMENDED FEASIBILITY ANALYSIS

A feasibility analysis of several possible configurations using several ALC's will determine if distributed ISF's are cost effective. The analysis might proceed as follows:

1. Define candidate configurations of distributed ISF's as in Figure 3-1. Show the actual weapon systems and their complements of items containing embedded computers.
2. Estimate the simultaneity of operations of the ISF's and the hours per week of operations for each weapon system.
3. Calculate data rates to and from each ALC in each configuration, considering expected error rates and the possible need for error-correcting coding.
4. Measure the time delay in transmission and compare to allowable delay for each configuration and each communication system.
5. Validate the intercenter data rates and delays on an existing ISF, located at one ALC, by subdividing the single-host ISF into software modules that correspond to the geographic location of probable ISF sites. An experiment would determine the intermodule bandwidth and the performance of the ISF while introducing time delays into the intermodule traffic. The resulting performance, with delays expected from long-distance lines, could then be determined.
6. Determine the cost of the distributed ISF, including any communication chargeable costs.
7. Determine the cost of several collocated ISF's with duplicated hardware and software.
8. Compare the distributed ISF with the collocated ISF with respect to cost, performance, availability, hardware maintenance, and software maintenance.

These analytical results should then be used in the network analysis (Section 2) to determine the cost of communication services for the ISF and the feasibility of time-sharing ISF with other inter-ALC services.

4. COMPUTER NETS

The computer nets that perform a mission on an aircraft, in a trainer, in a C³ center, or in a test set should not be confused with communication networks, described in Section 2, which contain computers that are geographically dispersed over several ALC's (References 4-1 and 4-2).

The AFLC is already maintaining embedded computers that are part of nets in flight systems, in ground communication systems, and in aircrew trainers. AFLC will also construct ground computer nets to do complex testing jobs. The designer motivations in creating a net are

- Increase the computation that can be done, using small computers associated with each subsystem, and understood by the vendor; and
- Increase the availability of the ECS by providing fault tolerance.

Nets will prove more difficult to service than individual computers because

- Fault detection and switching logic must be tested, and
- Software changes in one computer can involve others and may require reverification of the software for the entire net. This situation is especially true if variable tasking (Section 4.1) is employed in one or more of the computers in the net.

An example of an airborne net is the interconnection of an air data computer, inertial navigation computer, weapon delivery computer, flight control computer, radar processor, engine controller, electronic warfare computer, IFF-fusion computer, and display-drive computer. Each of these computers performs fixed calculations and may transmit data to and from some of the other computers on a MIL-STD-1553B data bus. Additional computers may be in the net, such as computers within an optical sensor pod. The ensemble of computers is called a computer net (sometimes called an array). Sections 4-1 through 4-4 describe the profound effect of the increased use of nets on AFLC, in hardware, software, and administration.

4.1 DISTRIBUTED COMPUTER NETS

The distributed computer net is the most general class of net and consists of several interconnected computers. The range of complexity of the net is described in Sections 4.1.1 through 4.1.4.

4.1.1 Fixed Tasks in Independent Computers

This net is the simplest possible. Each computer collects data from its sensors and displays the result without any interaction with the other computers. Such nets are in widespread use in older aircraft (e.g., F-111, F-4, B-52) and in ground test sets. The repair of any one computer does not affect any others.

4.1.2 Fixed Tasks in Interconnected Computers

In this net, each computer performs fixed functions (e.g., navigation or engine control), passes results to other computers, and receives inputs from other computers. The functions do not change according to hardware or software failures. This net is typical of many aircraft entering service, in which the air data computer outputs feed the navigation system, the engine computer, and the automatic flight control computer while a radar computer may drive a display computer and a weapon delivery computer. The Boeing 767, CX (if authorized), and F-16 AFTI (if authorized) are examples of this type of net. Hardware maintenance is simple if all computers retain a MIL-STD-1553B interface. Software maintenance is simple, to the extent that input-output routines are separated from the mission algorithms and from the data that are likely to be changed by AFLC. Verification on an ISF (Section 3) may be necessary after hardware or software changes. The ISF must have the ability to stimulate all computers simultaneously.

4.1.3 Variable Tasking

In this type of net, each computer changes its tasks in response to the loading on itself and on the other machines. Variable-tasking is used in large ground computers. A special type of variable tasking can be called "fallback tasking," in which only a predetermined reduced set of tasks is executed in response to increased load; i.e., the number of tasks executed does not decrease continuously as loading increases. Software for fallback tasking is less expensive to write and is vastly simpler to

verify than is the software for the more general variable tasking. Fallback tasking has occasionally been used in an avionics application (e.g., DAIS test bed, F-18 mission computers). It is possible that complex ground control centers, ATE, or ATD (that use large host processors) might adopt some form of variable tasking.

Maintenance of variable-tasking hardware would be no more complicated than for independent computers if a MIL-STD-1553B bus interface is used; however, maintenance and testing of variable-tasking software will be exceptionally complicated and will require high programmer skills and many new software tools. Where several computers divide a task, tools will be needed to convert a single application listing into code for the individual computers. Because of software complexity and the need for graceful degradation, TRW suggests that AFLC resist attempts to design the most general type of variable tasking into flight computers that will be PMRT'd before 1990, when the concept may be operational. Meantime, fallback tasking appears to be the state of the art in real-time systems that must be verified.

An example of fallback tasking is the net of two AYK-14 computers used as mission computers on the F-18. The two AYK-14's normally perform different functions: display and weapon delivery. Both are monitored by BIT and by self-test software. When a failure is detected in either one, the other changes function and backs-up the pair. The automatic switching logic is still being designed by the Control Data Corporation as of mid-1980.

Maintenance of the F-18 mission computer hardware is still being decided upon as of mid-1980. After automatic switchover, a BIT panel shows which computer has failed. The USN has not decided whether to provide special test sets on the carrier, whether Naval Avionics Repair Facilities (NARF) will repair and inventory the computers, or whether each aircraft carrier will ship failed units directly to the manufacturer. Software maintenance using an ISF will be done at the China Lake NAS starting in 1984.

4.1.4 Fault-tolerant Nets

This type of net continues to function after certain failures of the hardware elements. Fault tolerant nets may also be variable tasking. In 1980, however, the state of the art was in designing fault-tolerant fixed-task nets, in which each computer performs a fixed task and in which the hardware and software are redundant. These fault tolerant nets are discussed in Section 4.2.

4.2 FAULT TOLERANT NETS

There are many ways to design fault-tolerant computers and fault-tolerant computer nets. All involve redundant hardware and software elements at various levels of the design. This section considers the simplest such net in which two or more computers perform the same tasks; tasks that do not change as loading changes or as failures occur. The net can detect certain hardware or software failures (either with BIT or by voting some of the outputs) and can switch or vote the redundant computers. The design of fault-tolerant systems is governed by the type of faults to be tolerated. "Hard" faults are those that fail permanently; "transient" faults are those that occur momentarily, such as noise picked up on a communication line. The earliest application of fault-tolerant nets have been in analog, fly-by-wire autopilots which were used operationally on the DC-10 and L-1011. More recently, digital flight control systems are in use for the Space Shuttle Orbiter and are being designed for DC-9 and DC-10 modifications, the Boeing 767, the AMST, CX, and F-16 AFTI. Other types of ECS fault tolerance are included in the B-52 Offensive Avionics System (a backup, automatically-switched computer) and on the F-18 (the dual AYK-14s described above). The commercial "Tandem" computer is a dual-computer, automatically-switched fault-tolerant "net". Many of these will enter the logistics pipeline in the 1980's. These systems will make a profound impact on AFLC's support of embedded computers.

1. There may be a need for live flight control and engine control actuators (possibly hydraulic laboratories) that are coupled into the ECS support systems because of the traditional difficulty of modeling actuators analytically. Because of the expense and expertise involved, AFLC may choose to keep the flight control simulations at one specialized center where the life actuators can become part of a distributed ISF.

2. A need for test equipment that can verify the net's ability to detect hardware failures (Section 9, "Built-in Test").
3. A need for test equipment that can verify proper switchover among the multiple redundant embedded computers, given a correct measurement of their status. AFLC will have to measure transient switching behavior and steady-state behavior.

An example of a fault-tolerant, fixed-task net is the four-channel, redundant digital flight control system for the US Navy's F-18. The embedded digital computer is a microcomputer whose program and data are in PROM's on circuit cards. The flight control software is not maintained by the USN. When changes are desired, new PROM's will be ordered from McDonnell-Douglas who will write new equations. The equations will be sent to GE for coding and PROM burn-in. GE will then verify the new circuit cards on an Avionic Integration Bench (like an ISF).

Maintenance of the F-18 flight control hardware, especially the redundancy management aspects, is still being decided. The following plan is evolving.

1. On-board continuous checkout is designed to detect end-to-end faults using a self-test routine in the software that also compares channel pairs.
2. A crew-initiated preflight test is run using a routine in the flight software. It stimulates sensors and checks the 4-channel failure detection and switching. The result of this 50-second test is a "go" or "no-go" indication to the crew.
3. A crew-initiated maintenance test is run using a routine in the flight software. It stimulates sensors and requires pilot manipulation of the aircraft controls. The test takes 6 minutes and isolates the failure to an LRU, displaying the result on a maintenance panel in the wheel-well and on a cockpit display.
4. Defective LRU's will be repaired in the intermediate shop in the aircraft carrier. Test equipment has not yet been bought but will isolate failures to a circuit board and will often identify one or two possible component failures. LRU's that are not repairable in the intermediate shop will probably be sent back to GE; organic maintenance at a Naval Aviation Rework Facility (NARF) is also possible.

4.3 FIBER OPTICS COMMUNICATIONS

Fiber optics may be used for computer-to-computer communications in high electrical noise environments. A draft fiber optics version of MIL-STD-1553B has been proposed by the Society of Automotive Engineers. It is not clear whether fiber optics will be maintained by AFLC during the 1980 to 1990 period of this study. If this is the case, AFLC will have to develop an understanding of fiber optics and the facilities for the repair and maintenance of interconnecting cables and transmitter/receiver modules. Corresponding weapon system ISF's would also require fiber optic interconnections to maintain system fidelity during integration testing.

4.4 EFFECT ON AFLC

The following are the impacts on AFLC.

1. AFLC should press for extensive built-in test (Section 9), even in fixed task nets, that detect failures and identify which stand-alone embedded computer has failed. The need to isolate failures to circuit board-level embedded computers is not yet established. Fault-tolerant nets will certainly include considerable BIT, because of the need to switch redundant elements. AFLC may wish more extensive BIT and should, in any case, press for circuits to induce faults and test the BIT.
2. An ISF will contain several embedded computers, all working simultaneously and interchanging data, stimulated by synchronized drivers. Embedded computers that are not supported by AFLC can probably be simulated, thereby avoiding interface hardware, sensors, actuators, and sensor-stimulus generators in ISF. Embedded computers whose software is supported by AFLC may require these devices; e.g., digital flight control computers may require live actuators ("iron birds").
3. Complex software tools capable of writing programs for variable-tasking computer nets will be needed if such nets are in inventory.
4. AFLC should persuade AFSC to confine software changes to a few computers in the net.

- Hardware-related software changes can occur in any computer in the net. Where hardware-related changes are predictable (such as accelerometer calibration constants or radar modulation characteristics), the software is usually designed to accept them and is verified over the full range of anticipated values. Therefore, changes of this type can be permitted in several computers of the net.
 - Mission-related algorithmic changes and complex data changes that will be done by AFLC should be confined to one computer to simplify software development and reverification.
5. AFLC should suggest guidelines to AFSC for computer nets. For example, the flight computer guidelines might require:
- Built-in test, as in (1).
 - Confined software changes, as in (4).
 - Allow fallback tasking but avoid the more general variable tasking, in order to simplify maintenance and reverification.
 - A simple means should be designed into the net that will disable any of its constituent computers in order to permit the dispatch of an aircraft with an embedded computer known to have failed and to permit the net to be tested with a failed computer.
 - Put all non-mission-dependent instructions and data in ROM. The added cost of infrequent changes may be offset by the resulting ease of maintenance because inadvertent erasures are eliminated.
6. Distributed nets have found their way into aircrew training devices. Multiple computers offer the ability to perform more computations in real time. As flight systems themselves become nets, the ATD will have to become a net. Eventually, the ATD might be constructed as an interconnected net of MIL-1750A standard instruction set chips, many of them emulating airborne computers while the rest contain environmental and airplane data.
- The separation of flight software from the environment software in the trainer will greatly simplify updates; aerodynamic updates will then be confined to one computer, flight program updates will be confined to an emulation of one of the flight computers (or to the computer itself).
7. AFLC should ensure that ISF's are capable of inducing faults, measuring the correct response of BIT, and measuring the correct operation of switching circuits (Section 9).

5. LARGE SCALE INTEGRATION - VERY HIGH SPEED INTEGRATED CIRCUITS

5.1 1980 STATE OF THE ART

Numerous large-scale integration (LSI) devices are on the market (References 5-1 through 5-8). Line widths are down to 2 to 4 microns in the most advanced devices and RAM's are available with 16K and 64K bits of storage. Devices are available with densities of 1000 gates/chip or with 200 MHz speeds. LSI devices will continue to be sold in dual in-line packages (DIP) that are mounted on printed circuit boards. Thus, AFLC hardware maintenance will be affected only by the increased complexity of the boards and the need to program many elaborate test cases and diagnostics. Built-in test functions are becoming widespread in medium-scale integration (MSI) airborne avionics and will spread to LSI and very high speed integrated circuit (VHSIC) airborne and test equipment, and EW and C-E systems (Section 9).

5.2 VHSIC

The VHSIC program will develop a family of high-speed devices uniquely tailored to military equipment. Some chips will perform functions that are not available in the commercial marketplace (e.g., anti-jam and smart weapons processing). VHSIC Phase I circuits will be available in the marketplace in approximately 1984 with densities of 20,000 gates/chip or more and 25 MHz RAM/ROM speeds (compared to 4 to 6 MHz in 1980). Phase II circuits will be available by about 1988 at densities of 80,000 gates/chip and speeds of 100 MHz. It is therefore likely that Phase I circuits will be maintained by AFLC, both in flight and in test equipment, before 1990. It is likely that Phase II circuits will enter AFLC only in test equipment before 1990. In either case, VHSIC circuits will represent a continued evolution of LSI circuits and will not produce a qualitatively different impact on AFLC. The principal effect of the high-density, fast VHSIC chips (25,000 gates/chip at 0.5 nanosec delay) will be the difficulty of verifying hardware failures. Built-in self-test circuits on the chips themselves will help. One VHSIC chip will probably be a 32-bit microprocessor operating at a clock speed above 20 MHz. From an AFLC viewpoint, maintenance of VHSIC boards will not differ from maintenance of LSI boards. VHSIC software will undoubtedly be written in a high order language, resident on the board or in a

host computer. AFLC should ensure that the language used by VHSIC is a standard AFLC-supported language such as J 73 or Ada.

Test devices for VHSIC will be computer-driven and will have provisions for inducing failures and measuring the response.

5.3 LSI-VHSIC IN AIRBORNE RADARS

Generally two types of processing can be mechanized by microprocessors in airborne radars, regardless of the radar classification.[†] The two types of processing are signal processing and general-purpose processing. The functions normally performed by a signal processor are video processing, timing generation, smoothing, filtering, etc. The signal processing microprocessor is typically high-speed and is in either a pipelined or array configuration. The signal processor is normally dedicated to a function or set of functions and is either fixed-function or microprogrammable. The timing in these processors is normally very critical and difficult to test without a hot-bench, especially if other processors are interfaced.

The general-purpose processor is typically an off-the-shelf machine that is programmed in a high order language and has floating point arithmetic capability. Typical functions accomplished by the general-purpose microprocessor are track filter/file, mode control, and operator interface. Increasingly, the crew interface is being accomplished by a dedicated general-purpose microprocessor.

Functions in ground support facilities are also being implemented through microprocessors. Interfaces within an Integration Support Facility (ISF) may be implemented through microprocessors for ease of alteration. Microcomputer Development Systems (MDS) are used along with other tools (themselves implemented by microprocessors) to provide for microprocessor development and support.

As for projected implementation, microprocessors are currently being used in airborne radar systems for general-purpose applications and will soon be fielded as "programmable signal processors." These

[†]A typical class of airborne radars is the fire control class (FSG 1270); others are surveillance, ground mapping, etc.

signal processor-type applications are made possible through breakthroughs in high-speed technology and can be expected to provide for further versatility in tactical and strategic radars. AFLC can be expected to support radar data processors and programmable signal processors (PSP) by the fall of 1982. The radar data processor appears to pose no special support problems; however, the PSP will require additional equipment, testing techniques, flight test data, etc.

Microprocessors within ground support systems are becoming more evident in the ALC support complexes. The F-111 and F-15 AISF's at McClellan and Warner Robins currently employ microprocessors in several applications and the currently planned MDS at McClellan will provide for multiple support for a variety of systems outside the radar area. Another highly impacted area is self-test, which is currently expected to expand with added fault isolation techniques (Section 9).

5.4 MICROPROCESSORS IN ELECTRO-OPTICS

To provide night and all-weather capability, the Air Force is developing forward looking infrared receivers (FLIR) and low light-level television (LLTV). These target location aids are typically pod-mounted and contain their own embedded computer systems that interface with the aircraft host computers. The combined system gives the aircraft the capability of target location/detection, target designation, and navigation update from known landmarks and from velocity matching.

As an example of the development of an electro-optical tracking system, consider the evolution of PAVE TACK, PAVE TACK/VATS, and LANTIRN. The PAVE TACK pod provides full-hemisphere coverage using a stabilized line-of-sight with a precision laser boresighted with an Infrared Detecting Set. A controllable field-of-view allows wide-angle viewing for target acquisition with a narrow field-of-view used to aid in target identification. A cockpit hand controller allows the Weapons Officer to manually direct the line-of-sight, then revert to open-loop target tracking during evasive and attack maneuvers, meanwhile making vernier corrections with the hand controller. The laser designator/range finder can be used to pinpoint the target for guided munitions. The PAVE TACK computer contains 16K words of memory. It also interfaces to the aircraft computer.

The objective of the PAVE TACK Video Augmented Tracking System (VATS) was to add an automatic video tracking unit which would perform automatic video scene correlation to maintain lock on target despite evasive/attack maneuvers of the aircraft or the target. A microcomputer was added that could digitize and process selected video data using contrast detection, discrimination, and image encoding. The microcomputer contains 8K words of PROM and 6K words of RAM. In addition, 8K words of memory were added to the main computer. The automatic video tracking (with no aircrew assistance) was significantly improved when compared to tests using the standard PAVE TACK pod. Thus, a significant performance improvement was realized from the increased application of digital computation with no substantial change in size and weight of the pod.

The newest venture in electro-optical systems is LANTIRN (Low Altitude Navigation and Targeting Infra-Red System for Night). Planned for deployment on the A-10 and F-16, LANTIRN is intended to lessen the night/weather workload for the single-seat attack aircraft pilot by creating flying cues that are the same as daylight cues. It uses two forward-looking infrared (FLIR) sensors, one with wide and the other with narrow field-of-view. The wide-view FLIR displays the obscured outside world to the pilot on a wide-view Head-Up Display (HUD) called a video raster HUD. It shows the pilot a representation of the world ahead of his airplane that is about 27 degrees wide in the horizontal axis and about twenty degrees high vertically. After classifying targets, setting them into priority order, and matching them with Maverick air-to-subject missiles, the system waits for pilot consent to fire. When he consents, the system locks-on and fires a Maverick at each target in priority sequence. Specifications call for the LANTIRN system to be able to handle multiple Mavericks per pass and have a kill probability at least double the present level.

The LANTIRN pod will contain substantial computer capacity (Figure 5-1). Some of the increase over PAVE TACK/VATS can be attributed to the use of HOL - in this case JOVIAL J 73. Some of the increase is due to increased functions.

These examples of optical pods illustrate the proliferation of embedded computers and the increase in size of those computers. The

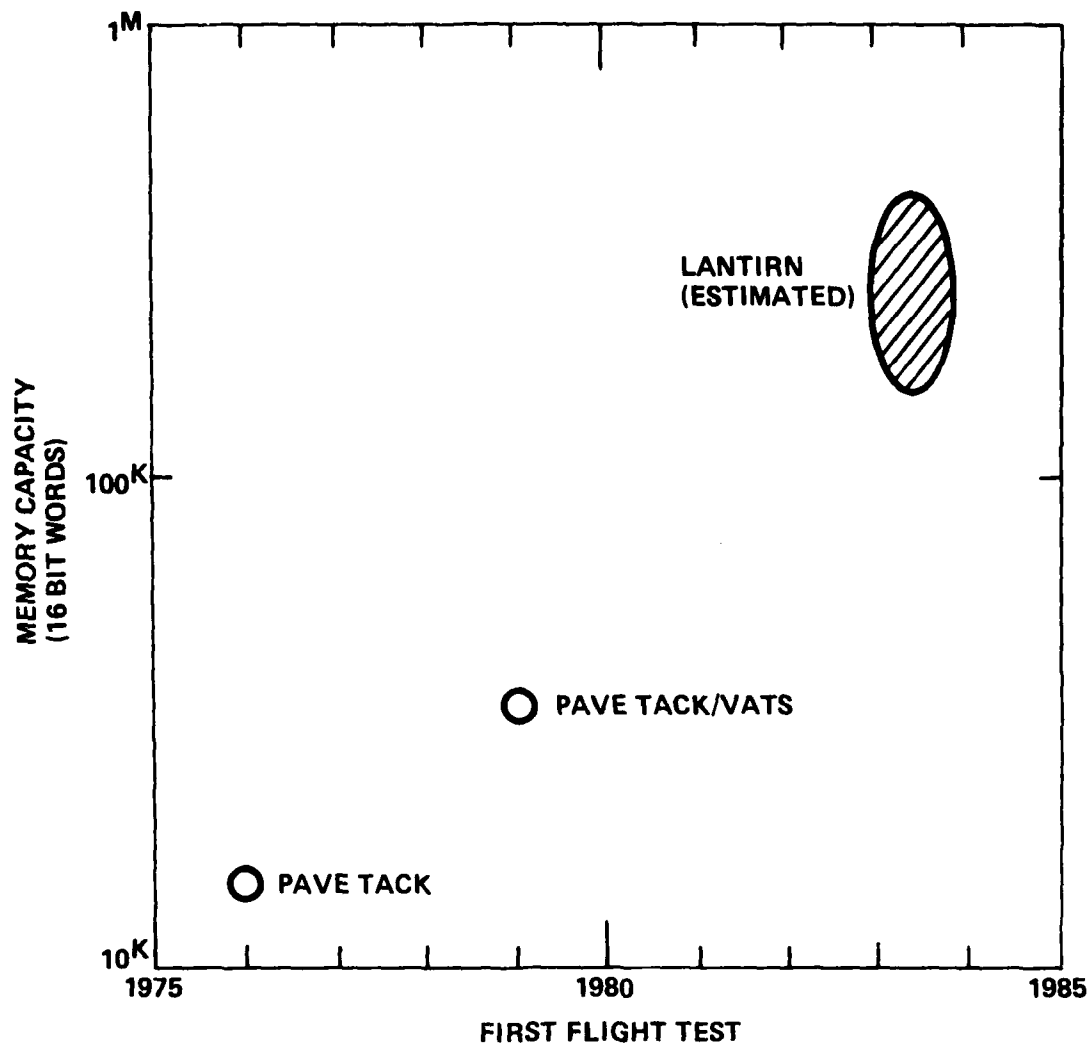


Figure 5-1. Growth in Electro-optical
Computer Requirements

pod's embedded computers interact with the aircraft's computers to form a net, as described in Section 4. AFLC will be required to maintain LANTIRN beginning in approximately 1985. Modifications will undoubtedly follow during the late 1980's with a still newer system transitioning to AFLC before 1990. The modifications and the newer system will undoubtedly contain VHSIC chips.

VHSIC processors will provide the capability to develop real time optical correlators, digital map displays and multi mode sensor displays for tactical aircraft. VHSIC processors and components will provide high throughput and complex data manipulation in weapon systems developed in the 1980's and 90's. These weapons will not be maintained by AFLC during the 1980-90 period of this study.

5.5 LSI-VHSIC IN ELECTRONIC WARFARE SYSTEMS

To provide real time signal processing for electronic warfare, the Air Force is building advanced radar receivers for tactical and reconnaissance systems which use dedicated computers to process raw data. These data consist of hundreds of complex pulse trains that must be de-interleaved, identified, and displayed to the operator or used to activate countermeasures. The use of VHSIC Phase I components should be expected in EW systems developed in the mid-1980's and maintained by AFLC in the late 1980's.

5.6 NAV-COMM SYSTEMS

VHSIC technology will permit a radical change in the design of NAV-COMM equipment, which may be cheaper than the traditional design using LRU's, each of which performs a specific function (e.g., HF Comm, ILS, Tacan). The radical change is to design a chip that performs all functions within a frequency band; for example, an L-band chip could perform the DME, JTIDS, and IFF functions while a VHF-UHF chip could perform ILS, VOR, and communication functions. A NAV-COMM set would consist of several such chips on boards, mounted in a single LRU and connected to the display and computer (which may simply be another circuit card) by a MIL-STD-1553B data bus. Each chip and board would have its own BIT and the LRU would have an internal power supply. Each board would connect to the appropriate antenna. If this new design concept

proves to be cheaper than the conventional designs, AFLC can expect to service the first such devices in the late 1980's. The principal effects on AFLC will be to reduce the amount of flight-line test equipment and to require that the repair shop tests boards in the same manner as LRU's are now tested. It is expected that there will be far fewer spare circuit boards to inventory.

5.7 STANDARD MICROCOMPUTER

A potential cost saving opportunity exists with the standardization of whole microcomputer boards in airborne and ground avionics and in test sets, as LSI and VHSIC devices enter service. Such standard boards would be specified by system designers whenever an embedded circuit-board microcomputer was needed. The standard board would have the following advantage:

1. Several procurement sources would exist for spare parts,
2. Operating system software might be interchangeable if MIL-STD-1750A instruction sets were used,
3. Inventory of replacement boards could be reduced, and
4. Maturity of circuit design is more likely than at present since production runs will be longer.

Several levels of standardization are possible.

1. Standard circuit board size, mounting, and cooling; standard backplane connectors, standard power, and standard input-output circuitry (buses) and format. Probably implementable in ATE, C-E, and ATD embedded-circuit board computers. Not likely to be useful for airborne equipment because of differences in vibration and thermal requirements and frequent need to conform to the shape of aircraft.
2. Standard backplane connectors, standard power, standard input-output circuitry. Useful for all embedded circuit-board computers. Although size would not be standardized, such boards would represent a large benefit to AFLC in simplifying the design of test equipment and probably in raising reliability.

Appendix C describes the standardization of interfaces among microcomputer circuit boards in more detail. AFLC should decide if the economics of maintenance would justify standard microcomputers at either

level. AFLC could then sponsor the development of the standard, perhaps via IEEE/ANSI, and enforce its use in AFSC.

5.8 EFFECT ON AFLC

The impact of support for microprocessor-based systems ranges from negligible to significant, depending on the application and the method of implementation. The following describe some of the more pertinent impacts facing AFLC.

1. A net set of tools is emerging for the ISF complex.
 - HOL's for microprocessors
 - In-circuit emulators
 - Logic analyzers
 - Microcomputer development systems
 - PROM programmers
 - ROM simulators
2. Testing techniques for microprocessors are immature and will require more experience and standards to become more efficient.
3. Methods of standardization without limiting technology growth must be found because of the massive proliferation of these devices.
4. Microcomputers embedded in avionic systems will probably require a larger fraction of input-output drivers relative to logical and algorithmic software than do stand-alone computers. Therefore, ECS that use embedded circuit boards will require a higher ratio of electrical engineers to programmers than exists now in AFLC.
5. The use of embedded microcomputers of different brands requires the inventorying of many types of chips at the operating locations. The use of standard circuit boards (Section 5.5) might reduce logistics costs.
6. Configuration control procedures embodying microprocessor devices will be needed to baseline and manage the configuration of the emerging systems.
7. Documentation requirements satisfying the needs of microprocessors must be defined and adhered to.

8. A difference exists between USAF and USN logistics practices: the former is supporting radar processors whereas the latter is not, (in the case of F-18), expecting that all changes will be hardware-related and therefore infrequent.
9. Many interface and timing problems may not be detectable without a hot bench.

6. HIGH ORDER LANGUAGES

In 1978 the Department of Defense had in its inventory software written in about 150 different programming languages (Reference 6-1). This linguistic proliferation increased maintenance problems due to programmer training requirements and lack of support tools for many of the languages (References 6-1 through 6-12). Consequently, DOD and Air Force have attempted to standardize on a few high order programming languages (HOL). DOD Instruction 5000.31, "Interim List of DOD Approved High Order Programming Languages" (Reference 6-2), states that only approved HOL's may be used for new defense system software. The programming languages approved in that report are JOVIAL J73, the language specified for Air Force embedded computer systems, Ada, the new DOD language to be used for all military operational software at some point in the future, and ATLAS, the language used for ATE software.

6.1 JOVIAL: PRESENT LANGUAGE FOR ECS SOFTWARE

JOVIAL, the language used by the Air Force since 1959, is a derivative of ALGOL and was specifically modified to support command and control systems. As an ALGOL derivative, JOVIAL provides the block structures necessary for implementation of structured programming software designs.

The J73 dialect will be used to develop operational software and support software for the next several years. At some point in the future, Ada will replace J73 for new software. Because Ada will not be mature enough for developing operational or support software for several years (Figure 6-1), the Air Force will continue to develop new software in J73 for many years (to achieve the cost saving, reliability, and portability that will result from a standard language) and will continue to maintain J73 programs. AFLC should develop J73 tools and skills now.

6.2 Ada: FUTURE LANGUAGE FOR ECS SOFTWARE

To promote reliability and maintainability, the designers of Ada emphasized program readability over ease of writing Ada programs. Thus, Ada programs may take slightly more effort to compose properly

JOVIAL J73	CY											
	80	81	82	83	84	85	86	87	88	89	90	
LANGUAGE SPECIFICATION	_____											
COMPILER DEVELOPMENT	_____											
AFSC DEVELOPMENT OF NEW SW	_____											
AFLC SW SUPPORT	_____											

Ada	CY											
	80	81	82	83	84	85	86	87	88	89	90	
LANGUAGE SPECIFICATION	_____											
COMPILER DEVELOPMENT	_____											
AFSC DEVELOPMENT OF NEW SW	_____											
AFLC SW SUPPORT	_____											

Figure 6-1. Schedule for New USAF High Order Languages

but once composed, are generally more likely to be correct and also easier to change. Useful forms of redundancy are required in Ada programs and Ada compilers must check for consistency. Error-prone notations in other languages have been avoided and English-like constructs are generally used in preference to obscure abbreviations. To support economical program development and maintenance, Ada programs may be composed of many separately-compiled parts without sacrificing the consistency-checking properties.

The quality of the Ada design, the strong support of Ada by DOD, and its growing acceptance in industry make Ada increasingly likely to succeed as a common language for both general-purpose and military applications. An Ada compiler for the VAX 11/780 is currently under development for the U.S. Army. The current challenge is to develop a standardized language and programming environment that will allow the advantages of Ada to be realized in practice.

6.3 ATLAS: LANGUAGE FOR ATE SOFTWARE

ATLAS is a standardized test language for expressing test specifications and procedures. It is a test-oriented language independent of test equipment, and provides a standard abbreviated English language used in the preparation and documentation of test procedures which can be implemented either manually or with automatic or semi-automatic equipment.

ATLAS programs are written in much the same way as test specifications. ATLAS programs are self-documenting; that is, test engineers and technicians can read a program and understand it.

Both the OSD memo establishing ATLAS as the ATE standard language within DOD and the AFLC memo identifying ATLAS waiver justification policy acknowledge the limitations of ATLAS as a language for test programming and the need for development of a test language which is compatible with the ATLAS test specification. Thus, there is a need for development of standard ATLAS compilers with validation and support facilities similar to those intended for Ada, such that the proliferation of ATLAS dialects, as test programming languages under the Adapted ATLAS

umbrella may be halted. There is also some expression of interest within USAF for an interactive version that would permit real-time preparation and change of test programs.

6.4 USAF USAGE OF HOL

There are several DOD programs that are currently slated to use JOVIAL J73. Examples are

- | | | |
|------------|--------------------|----------------------|
| • MX | • Pershing | • ATF |
| • CX | • CHOL development | • Wild Weasel Update |
| • DIS | • AMRAAM | • DSM |
| • DAIS/IDA | • LANTIRN | |

There are several other programs that have a possibility of using J73, such as

- | | |
|-----------------------------------|---------------|
| • NATO Air Defense System | • E-4 update |
| • F-106 fire control radar update | • SCF upgrade |
| • EW applications | • MATE |
| • KC-135 upgrade | • F-111 |

It is expected that all new Air Force ECS programs will be required to use J73. JOVIAL J73/I has been used successfully on DAIS and on the Inertial Upper Stage of the Space Transportation System. The two versions are basically the same; therefore, we should not expect to find any problems inherent to the language itself.

The Air Force plan for Ada implementation consists of four principal steps.

1. An initial Ada compiler, to be hosted on and targeted for the IBM 360/370, will be developed in FY 80 to 83 and maintained through FY 84.
2. The initial compiler will be rehosted and retargeted for the CDC 6600, DEC 10 and Interdata 7/32 computers in FY 84.

3. Ada common support environment will be developed in FY 81 to 83, and maintained indefinitely.
4. The compilers and support environment will be integrated, validated and demonstrated on the IBM 360/370 system in FY 83 and 84.

The Rome Air Development Center will coordinate the Air Force Ada development effort. The combined efforts of all the services will be coordinated by the Ada Joint Program Office (AJPO) which has been the technical overseer of Ada development so far and will replace the High Order Language Working Group (HOLWG). AJPO is a special committee serving the office of the Under-Secretary of Defense for Research and Engineering.

6.5 SCHEDULES

A schedule that shows the projected development and use of each language is shown in Figure 6-1. AFLC can expect to begin supporting J73-based systems in 1983. AFLC support of JOVIAL will probably continue past 1990 because of the volume of maintenance and change activities in support of JOVIAL-based weapons. AFLC support of Ada-based weapons is unlikely before 1987. AFLC-sponsored development of new Ada software is unlikely before 1990. During the 1990's, Ada and J73 will probably coexist at AFLC facilities, as will Ada technology HOL's now in use. ATLAS programs for ATE will continue to be used in the 1980's and 1990's. Although not presently planned, an interactive version of ATLAS (perhaps a dialect of Ada) may enter AFLC service in the late 1980's.

6.6 EFFECT ON AFLC OPERATIONS

Currently, most operational software is written in assembly language. The Air Force is moving toward the time when all operational software will be written in a HOL (as directed by DODD 5000.29). In the near term, this HOL will be JOVIAL J73 (as directed by DODI 5000.31); several years hence the HOL will be Ada.

The use of a common HOL will benefit AFLC. The job of the software engineer and programmer will be easier as HOL's are easier to understand than assembly language and are self-documenting to some

extent. Users will have to learn JOVIAL and Ada but will not have to learn a multitude of different assembly languages.

Tools to test, debug and modify software at the source level will be needed. Much of the support software will be reusable from system to system. Simulations will be required to exercise the operational software on the flight computer, an interpretive computer simulation, and an emulation on the host computer. These simulations should output diagnostic data that relate to the source code rather than to the machine code as is common today.

The compilers will have to generate a set of tables that contain data needed by the simulations to allow them to accept source level inputs and to generate source level diagnostics.

A standard interface between JOVIAL J73 compilers and simulations is needed to specify the format and content of the tables to be passed from the compiler to the simulations. The starting point for this exists today in the form of an Interface Control Document (ICD) that specifies Internal Symbol Dictionary (ISD) tables generated by AFWAL J73/I compiler for use by the Software Design and Verification System (SDVS). These tables would be used by a test case generator that would be controlled by inputs written in a test case language as in SVDS. A standard for this language should be written. It may also be feasible to have a standard interface between the simulations and the post-processor, making use of a standard post-processor possible in conjunction with all simulations and flight tests

The introduction of operational software written in a standard HOL provides the opportunity to develop a standardized support system as shown in Figure 6-2. Compilers and assemblers will generate a set of interface tables that could be used by a test case generator to locate statements and data names within the operational software. The format and content of these tables will be standard for all compiler implementations. A standard test case language would control the simulation run, specify the desired outputs, and specify the post-processing to be performed on the output data. The test case generator would produce a set

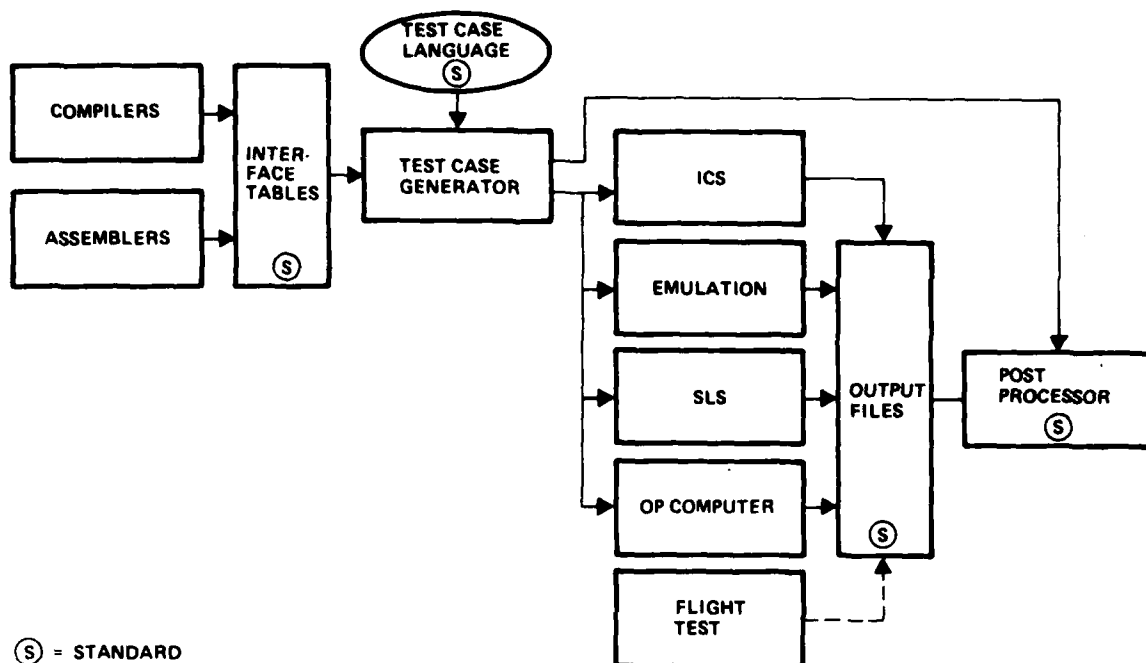


Figure 6-2. Block Diagram of Standardized Support System

of tables that are then used by the simulation to control its operation. These tables are different for each type of simulation but serve the same purpose.

Figure 6-2 shows four types of simulations: an interpretive computer simulation, an emulation using a microprogrammable computer, a statement-level simulation, and use of the operational computer within a simulated environment. The outputs of these simulations will be in a standard format such that they can be processed by a standard post-processor. Flight test data could also be transformed into the standard format.

The support system of Figure 6-2 would consist of standard interface definitions, standard software modules, and some pieces of software that are tailored for each specific system. Modules that are peculiar to a single operational system are added to the standard system. All support software should be written in J73 with the possible exception of scientific models which could be written in FORTRAN. The development of a standard support system will result in reduced support software development cost, improved capability and reliability, and better personnel utilization.

6.7 RECOMMENDATIONS TO AFLC

Support for operational software written in standard HOL's can be imposed by specific AFLC actions.

- Initiate development of J73- and Ada-based programmer aids, such as emulation-writing tools, improved source-language editors, and efficient static-testing tools.
- Begin the design and development of a standardized support system, such as shown in Figure 6-2. Establish detailed requirements and the necessary standards. Rehost existing J73 compilers to the host computers at the ALC's.
- Evaluate the need for J73- or Ada-based problem-oriented languages for simulation development and as an interactive test language.
- Investigate the use of existing support software and concepts such as SDVS, ISSS and the Ada support environment.
- Take an active role in the Ada development process.

7. EMULATION OF ECS

An emulation of an embedded computer is a simulation of the embedded computer (sometimes called the "target" computer) on another "host" computer on which the embedded computer's software can be executed directly. That is, the computer program intended to be loaded into the embedded computer (flight computer, C³ computer, trainer, or other ECS) can be executed in the host computer without recoding (References 7-1 through 7-5). For the following reasons, emulations are useful to AFLC

- A host computer can be made to emulate several different target computers by merely changing software, thus reducing hardware costs.
- A common emulator for several static test stands would reduce the AFLC inventory of embedded computers and would also reduce maintenance and technician-training costs.
- A common emulator could be used in several dynamic simulation areas in place of embedded computers.
- Emulations allow extensive diagnostics and recording of intermediate results that are not always permitted by the embedded computers.
- Emulations aid hardware/software tradeoff analysis for proposed modifications.
- Emulations allow verification of proposed hardware and software modifications before the change is implemented.
- Emulations for embedded computers supported by AFLC in aircrew training devices would permit direct loading of the OFP's.

Emulations are executed on various types of computers

- Scientific host computers in which many host-machine instructions simulate an embedded computer. Such simulations are called interpretive computer simulations (ICS) and can be accurate to the bit level or word level (an algorithmic-level simulation is a scientific simulation). Input-output instructions are also simulated. Bit-level ICSs may execute hundreds of times slower than real time.

- Specialized emulation computers, such as Nanodata Corporation's QM-1, whose microcode can be made to execute instructions bit-by-bit like the embedded target computer. Input-output instructions are also emulated, these emulators can run in near-real time.
- Newly developed LSI chips that execute the same instruction set as the embedded computer. These chips may soon be available for the MIL-STD-1750A standard instruction set. The fidelity of the emulation depends on the ability to emulate the input-output. These emulations would run in real time.
- Specialized computer arrays; whose microcode emulates the embedded computer or which uses chips that execute the same instruction set. Parallel computers in the array emulate the CPU and input-output in real time, at low cost, and with high reliability due to the low parts count. Such emulations are now in development (Section 7.1).

Future trends in emulation will take two paths:

1. For non-MIL-STD-1750A embedded computers, parallel computers as described in the last bullet above, and
2. For MIL-STD-1750A embedded computers, AFLC should expect the use of LSI chips that execute the same instructions, leaving only the input-output to be emulated.

7.1 CURRENT EMULATION HARDWARE

Hardware must be microprogrammable to be useful in emulation. Although a number of systems are microprogrammable, notably the large scale IBM series, only a few manufacturers encourage users to alter or add microcode.

The most widely used, commercially available microprogrammable system at the present time is the Nanodata QM-1. The QM-1 is a general-purpose emulation system with two levels of microprogram control. The second level is called nanostore and is used to interpret the microinstruction set. Therefore, the QM-1 can be made to emulate totally any single-CPU ECS. Cycle time at the micro level is 75 nanoseconds.

This two-level system allows the QM-1 to have a high degree of parallelism, thus better emulating the target machine. Because of its wide use, the QM-1 has the most extensive set of support software of

the commercially available emulation systems. The QM-1's main problem, from the AFLC point of view, is the difficulty of programming microcode and in many cases the QM-1 is unable to emulate an embedded computer in real time.

The Burroughs B1700 is a general-purpose emulation system with a one-level read-write control storage which limits versatility. The system has a cycle time at the micro level of 166 nanoseconds. It will not, however, meet the real time requirement for most embedded computer systems.

The Burroughs D machine has two control levels, but the nanostorage (second level) is read-only and cannot be changed dynamically. Thus, only fixed interpretations of microcode are available. It does have two predetermined sets of microinstructions. The D machine has the same two problems as the B1700: lack of software support and inability to meet real time.

The three computer systems described are a cross-section of the machines designed for microprogramming and emulation. Many other microprogrammable systems are beginning to be supported and used for emulation, such as the VAX 11/780 system which is used for large ballistic missile emulation in Huntsville, Alabama.

The second side of hardware emulation systems is the computer-on-a-chip. Representative of this group is the Advanced Micro Devices (AMD) AM 9080A chip, for which AMD has an emulation system. In most cases, these emulations are only applicable to the manufacturer's devices. General-purpose emulators built from these devices are described in Section 7.3.

7.2 CURRENT EMULATION SOFTWARE

For the development of emulations, a large amount of microcode must be written. Programmer productivity would be greatly improved with a HOL that would allow the user to code descriptive statements which are then translated to microcode. Unfortunately, such compilers do not exist for emulation computers. The few languages that exist write

microcode, not the nanocode required as machine instructions by these two-level machines. There are currently only two such HOL's: SMITE (software machine implementation tool using emulation), developed at TRW, and ISP (instruction set processor), developed at Carnegie-Mellon University. Both of these languages allow the symbolic description of computer architectures. Their biggest drawback is that the microcode is not efficient enough to reach real-time emulation speeds. The majority of work and documentation using these HOL's is in conjunction with the Nanodata QM-1 or with special-purpose research systems.

Another method of emulating a target system is via simulation (e.g., an ICS). The host system uses software to translate the target instruction set to the appropriate sequence of actions needed to emulate machine instructions, thus giving an environment which will allow code written for the target system to run. Slow response time is an inherent disadvantage of the ICS.

7.3 FUTURE EMULATION HARDWARE

Improving the performance of a single-CPU microprogrammable emulation computer is very attractive because of the existing software support and large amount of experience available. Two design improvements are possible: single-CPU computer, or a new machine which is able to run existing software support tools with minimal changes. The single-CPU upgrade can increase throughput by off-loading input-output to another mini, adding a subroutine capability, emitter-coupled logic (ECL), larger word length and a programmable logic array. Developing a new machine would increase the number of parallel CPU's, e.g., by off-loading input-output and adding parallel multipliers.

Parallelism can be achieved by a manufacturer who makes a special-purpose emulation computer or by a user who can assemble chips and microcomputers. An example of user-constructed parallelism is shown in Figure 7-1. Bit-slice architecture is used and implemented with high-speed bipolar components, to execute a series of microinstructions which emulate an ECS-computer instruction. Each ECS-computer instruction is

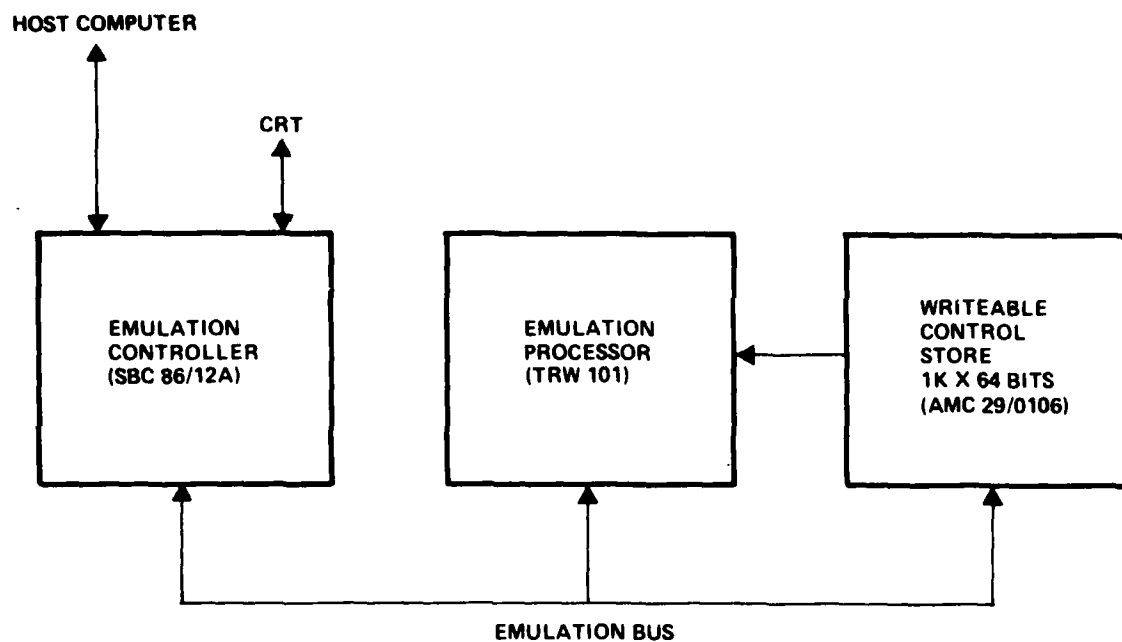


Figure 7-1. Example of User-Controlled Parallel Computer Emulation

represented by a sequence of microinstructions. With state-of-the-art bit-slice technology, a microinstruction requires 125 nanoseconds for execution. On the average, six microinstructions are needed to represent an OFP instruction. Hence, the emulation of an ECS instruction would require, on the average, 0.75 microseconds. This speed is usually faster than the original ECS. Appendix C describes a typical parallel-computer emulator in detail.

Another method would be to build a high-speed general-purpose emulation system based on one of the two HOL's (SMITE or ISP). With this method, the process of building an emulation for a new architecture is a straightforward translation. The method has already been applied to *software for ISP* at Carnegie-Mellon University.

High-speed emulation research at Stanford University could contribute significantly to operational systems. Stanford has a system called EMMY which reaches IBM 360/145 speeds.

Perhaps even more important than speed is the inherent flexibility of the emulation technology. The microinstructions can be changed from representing one embedded computer to another by reading a different set of microinstructions into the writable memory from a disk.

7.4 FUTURE SOFTWARE SUPPORT SYSTEMS

Future upgrades to existing high order languages (such as SMITE and ISP) will produce more efficient code to assist in the task of reaching the real-time emulation goal. These languages will be modified to handle newly-developed hardware.

An acceptance of a standard high order language for microprocessor-based equipment (perhaps Ada) would help emulation development. Commercial microprocessors are regularly sold with BASIC and PASCAL. The USAF is trying to standardize on the PASCAL-like language, Ada (Section 6); however, the Ada language may be incapable of generating efficient microprocessor code without the addition of microprocessor functions.

7.5 RECOMMENDATIONS TO AFLC

The following recommendations are suggested to AFLC.

1. AFLC should analyze its own emulation requirements and create a plan for emulations to be used in static test stands, dynamic simulation areas, test equipment support facilities, and other areas. The emphasis should be on real-time emulations. It appears that a system based on the bit-slice, multi-board microprocessor technology would give AFLC a low cost, highly reliable, fast system.
2. AFLC should participate in HOL-development groups to sponsor development of a HOL that would write efficient microcode and nanocode for emulators of various types, especially the parallel-CPU type emulator.
3. Because USAF electronics equipment will contain nets of embedded computers, the trainer requirements for developing and upgrading software will become much more complex and costly. A solution would be for AFLC to sponsor development of a common general-purpose emulator which could be used to emulate each embedded computer. Having done so, checked-out OFP's could be loaded directly into the trainer emulators without reprogramming, and trainer costs would be reduced.
4. AFLC emulation specialists should be concentrated in one skill center. These people would do the programming or modifications of the host emulation system for each target system. They would then send the new target machine emulation to each user (static test stand, dynamic simulation area) or trainer group whose only requirement would be to load their local, common emulation system.
5. Emulation on a scientific computer may run a few hundred times slower than real time, while emulations on computers designed specifically for emulation will run only two to ten times slower than real time. These considerations must be examined in planning for support capabilities.

8. STANDARDIZATION

The Air Force Logistics Command is the principal beneficiary of the standardization of hardware and software in embedded computers (References 8-1 through 8-4). The most cost-effective action for an AFSC SPO would be to procure custom equipment optimized for its own weapon system. In some cases, a custom development is justified for added performance; in these cases, however, the AFLC 25-year support costs are increased with respect to:

- Purchasing and inventorying small-lot spares at many bases;
- Training technicians to maintain complex and/or unique flight, and test equipment;
- Developing and maintaining software development facilities (ISF's and other support tools);
- Training programmers to write applications programs in seldom used high order languages; and
- Training programmers to maintain software (especially operating systems) in seldom used machine languages.

Successive avionic system acquisitions have produced a continued proliferation of unique avionic systems and subsystems leading to ever increased support and retrofit costs. Support costs can be significantly reduced for some subsystems by standardization. To enforce standardization, AFLC should participate closely in the early phases of design and identify deviations from standardization to the DSARC boards, with the cost impact of nonstandardization. It is expected that DSARC would disapprove deviations where the Systems Command could not justify the added expenditure during the system's deployment phase.

8.1 STANDARD INSTRUCTION SET ARCHITECTURE

Two computer architecture standards have been defined.

1. MIL-STD-1750. Military Standard Airborne Computer Instruction Set Architecture (Reference 8-2).
2. MIL-STD-1862. Military Standard Instruction Set Architecture for the Military Computer Family (Reference 8-1).

MIL-STD-1750 defines a standard 16-bit word Instruction Set Architecture (ISA) primarily for avionic application; MIL-STD-1862 defines a standard 32-bit word ISA primarily for tactical C³ applications. These standardized interfaces do not dictate the internal design or implementation; e.g., neither standard specifies the execution time, throughput requirements, or input-output structure. These standards do define the instruction set such that any program written in the standard computer language will execute on any computer conforming to the standard with only minor changes, provided the input-output subroutines (usually written in assembly language) are isolated from the application programs.

In the Digital Avionics Integration System (DAIS) program, two implementations of MIL-STD-1750 have been undertaken by Westinghouse and by Sperry Univac in the AN/AYK-15A processors. Extensive tests of software and support tools have been made. An operational flight program is being developed to perform a representative close air support mission in this implementation. The executive and application software are being written in JOVIAL J73. This development will demonstrate the interoperability of the two implementations.

As a result of the testing of these two implementations, a number of deficiencies were discovered in the MIL-STD-1750 standard. Correction of these deficiencies and an expanded memory option required for the AF LANTIRN Program resulted in the issuance of MIL-STD-1750A. An extensive set of software support tools is being developed for MIL-STD-1750A. User group meetings are held periodically. Full test and certification of the standard is scheduled to be performed by the ASD/SEAFAC facility in late CY 81.

The MIL-STD-1862 ISA for the Army's Military Computer Family (MCF) was also intended to reduce proliferation in hardware and software tools. Current plans are to initiate a competitive procurement in early CY 81, followed by dual awards for full scale development in CY 83. Earliest deployment will be in CY 86; thus, AFLC will probably not be required to support MIL-STD-1862 ECS prior to 1990. Nevertheless, AFLC should prepare support facilities and tools for MIL-STD-1862 during the mid-1980's.

It is expected that LSI circuit boards and later, VHSIC chips (Section 5), will be designed to execute these standard instruction set architectures. If these computer architectures are standardized, the benefit to AFLC will be that operating systems and support software tools will be similar or identical for many computers, thus simplifying maintenance. Compilers to generate machine code will be similar or identical, no matter who manufactures the computer. By confining future embedded microcomputers to those that use the MIL instruction sets, the proliferation of microcomputer languages can be avoided in AFLC. In addition, standard instruction sets and HOL's will allow the development of better software tools than are currently possible, thereby making possible a large increase in programmer productivity.

8.2 STANDARD DATA BUS

The MIL-STD-1553B data bus is designed to interconnect all airborne LRU's, including stand-alone embedded computers and LRU's that contain embedded-computer circuit-boards. The majority of digital LRU's (except rare LRU's whose data rates exceed the bus capability) will attach to the data bus, no matter what their function or design (neither connectors nor power characteristics are standardized). It is expected that VHSIC chips will be developed to provide computer interfaces to a MIL-STD-1553B bus. The benefits of bus standardization to AFLC are many.

1. Ground test facilities can use one bus to support many subsystems because driver logic and hardware are common.
2. All ISF's that are used to test arrays of computers can use the 1553B bus or a simulated bus to connect the devices at a single ALC.
3. A standard interface could be defined from embedded microcomputers to the MIL-STD-1553B chip, further standardizing the embedded microcomputers and simplifying the logistics of bus driver chips.

8.3 STANDARD SOFTWARE TOOLS

There are two technological trends in AFLC support software:

1. Standardization of high order languages and instruction sets, which permits the same tools to be used throughout AFLC; and
2. Longer-range development of integrated tool sets (also called "integrated software support environments") that meet the needs of AFLC users.

The most significant advantage of standard instruction sets is common support software tools. An extensive set of support software tools is being developed for the MIL-STD-1750A ISA as follows:

- MIL-STD-1750A ISA simulator,
- MIL-STD-1750A Assembler/Cross Assembler,
- J73 Compiler with MIL-STD-1750A ISA code generator,
- Linker/Loader programs, and
- MIL-STD-1750A Acceptance Test Program.

AFLC should insist on their use, where applicable, and should maintain them in order to achieve the full benefits of standardization.

Integrated tools are being developed for standard languages, such as DOD's Ada and JOVIAL. The Ada Programming Support Environment (APSE) will provide a common data base, which will contain all data relevant to a given project and which will be portable from one implementation of an APSE to another. The APSE will also provide for a common user interface which will allow a user to communicate with the APSE in the same "language", independent of the host computer on which the APSE is resident. Another feature of the APSE is a minimal set of tools which will be present in all implemented APSE's. APSE is intended to be independent of the host processor.

A second example of an integrated tool is the research project, GANDALF, which is being developed at Carnegie-Mellon University. This project defines two tools: a syntax-directed editor and an incremental program constructor. The syntax-directed editor provides syntactical and semantical error elimination by allowing only correct

constructs to be entered via the editing process. The incremental program constructor provides the function of the current compiler, linker, loader, and symbolic debugger tools. These two tools allow the user to pass from one to another with minimum user effort. The heart of the GANDALF concept is the single representation of a program, in which the traditional representation of source code, object code, listing, and absolute code are eliminated and replaced by a single syntax tree from which all other representations are derived.

Other examples of integrated tools are TRW's Software Design and Verification System (SDVS), Boeing's Support System (BSS), GTE Sylvania's PHOENIX System, and the Navy's Facility for Automatic Software Production (FASP). The Air Force Wright Aeronautical Laboratories (AFWAL) have recently published a draft for a JOVIAL-based software development environment called Integrated Support Software System (ISSS). A commercial integrated tool set (the Programmer's Workbench) is being developed for UNIX by Bell Telephone Laboratory.

AFLC would improve the productivity of AFLC software engineers by assigning a "Software Manager" to develop and maintain each tool, much as item managers are responsible for maintenance of hardware. Standardization of languages and computer instruction sets will make it possible to use the same tools throughout AFLC. Tools can be readily transferred among centers using the proposed intercenter network. Integrated tools, developed by AFSC, should reach AFLC with the systems they support during the 1980's. Continued support and promotion of these standards by AFLC will encourage the implementation and development of standard environments which will reduce AFLC's maintenance and support costs.

8.4 STANDARD HIGH ORDER LANGUAGE

Section 6 contains a discussion of the standardization of high order languages used for writing application programs for embedded weapon systems computers and for embedded test equipment computers.

8.5 STANDARD ATE

Present practice is for Air Force Systems Command to develop custom test sets for each embedded computer (indeed, almost for each LRU developed). For the future, AFSC is developing a standardized approach for acquiring general-purpose testers through Project MATE (Modular Automatic Test Equipment). MATE will encourage production of test capabilities which are hardware and software modular in accordance with proposed standards. As standard languages, buses, and instruction sets come into use, AFLC should persuade AFSC to adhere to the modular developments. These modules will allow cheaper maintenance and minimized technician training.

8.6 STANDARD HOST COMPUTERS

Test tools, such as ISF's and integrated tools, reside on mainframe or mini-computers called hosts. Each ISF has one or more hosts. Aircrew Training Devices (ATD) also use hosts with computing characteristics and input-output similar to those of the ISF hosts. Thus, AFLC should undertake an economic analysis of standardization of ISF and ATD hosts. Even if AFLC-wide standardization is not possible, standardization of hosts may be possible within any one center. The economic value of standardization would be weakened if all applications software were written in a standard high order language because all software would be transportable among hosts, standardized or not. The benefits of standardization would also diminish if the computer and operating systems were maintained under service contracts.

8.7 STANDARD MATHEMATICAL MODELS

At present, most AFLC organizations develop their own vehicle models, weapon models, and environmental models needed for ISF testing or for aircrew training. Each unique math model must be developed, coded, tested, debugged, integrated, and maintained. As standard HOL comes into use and as skilled software manpower becomes more scarce, it will become important to develop a library of models to be shared by ISF's and ATD's. The intercenter network and a potential network extending to flight bases will make that practical.

AFLC should prepare an inventory of existing models and those expected to be used in the 1980 to 1990 period. AFLC should determine whether or not these existing models can serve as the basis for development of new models. If not, AFLC should plan the orderly development of common models by assigned ALC software managers.

8.8 STANDARD OPERATOR-INTERACTIVE SOFTWARE

As described in Section 10, some degree of standardization is occurring in operator-interactive software with regard to CRT and plotter interfaces. However, standardization of operator interaction (e.g., creating displays and calling sequences of displays) and programmer interaction (e.g., creating terminal driver software and writing operator-interactive logic) is unlikely to occur in the next 10 years because each manufacturer has his own alphanumeric keyboard, function keys, light pens, cursors, etc.

Interactive displays have increased programmer productivity relative to their productivity using batch processing. Interactive displays have made real-time ISF's and trainers possible. Lack of standardization of terminals and software, however, raises maintenance costs and limits the increase of programmer productivity by making it necessary for programmers to learn several different operating systems. Nevertheless, TRW is unable to recommend extensive standardization of user interfaces because a monopoly on hardware and software of any one manufacturer at each ALC is unwise.

Suggested AFLC actions during the next few years in this technical area are described in Section 10.3.

9. BUILT-IN TEST

AFSC incorporates test circuits and software into ECS LRU's for any of three reasons:

- To verify the operational status of the LRU,
- To detect failures for repair reasons (these test provisions are in support of anticipated logistics command needs), and
- To detect failures for purpose of switching redundant elements.

These circuits and software are required for normal operation of redundant ECS systems.

Furthermore, the ECS may be used to test other LRU's in a weapon system or test set, e.g., by controlling the insertion of artificial inputs or by comparing redundant information during normal operation of the weapon system.

9.1 TECHNOLOGY

The relationship between built-in test (BIT) in computer circuits and the supporting ATE will change markedly under the influence of LSI and VHSIC and as the failure modes of ECS's are better understood (References 9-1, 9-2, and 9-3). TRW expects that the BIT cost will decrease and reliability increase, partly due to the inclusion of self-test logic in the chips.

AFLC can expect the need for flight-line ATE to decrease whereas the role of shop ATE will be (1) to test BIT circuits, (2) to isolate BIT-detected failures in operating circuitry to the replaceable element (module, board, or component), and (3) to verify correct operation of an ECS (BIT and operating circuits) prior to releasing an LRU to the field. In ground equipment (ATD and CE), the distinction between BIT and ATE will lessen as more self-test circuits are included in operating equipment. ATE will, more and more, take the form of general-purpose test equipment such as traditional oscilloscopes and signal generators as well as programmable testers with hardware adapters for modules and circuit boards.

The design of BIT hardware or software depends on the failures to be detected. BIT can be designed to detect certain permanent failures or transient failures, for CPU, memory or input-output failures, or for software "failures".

If failures are to be localized to an ECS, system-level tests are usually best.

- Comparison of the outputs of redundant computers.
- Use of an error-detecting protocol on communication lines, in the memory and, in extreme cases, in the CPU using codes that remain valid under arithmetic operations.
- Use of reasonability tests on incoming data to protect against sensor failures.
- Loop-back tests of input-output circuits, in which a few output channels are fed back to input channels, thus testing virtually all input-output circuits and software.

These tests will identify most failures of stand-alone or circuit-board ECS. Built-in test provisions often include the monitoring of such key parameters as:

- power supply voltage or current;
- clock frequency, measured with request to an independent oscillator; and
- software time-out discrete, to verify that instructions are being executed.

The number of faults that can be detected with BIT depends on the complexity of the circuitry. As the BIT circuitry increases, the probability increases that BIT failures will mask an ECS failure, and that BIT failures will induce an ECS failure.

9.2 ECONOMICS OF BIT

AFLC should define requirements for BIT based on AFLC historical data and projected requirements. At present, the development contractors specify how equipment is tested and the mix of BIT and ATE, using some data from AFLC. AFLC could assist in the specification of economically optimum BIT by making AFLC data available in the

form of maintenance models (development costs and operations costs) and by conducting analysis of ECS maintenance in airborne equipment, C-E, and ATD.

For example, a BIT analysis of airborne ECS would include:

1. AFSC out-of-pocket costs, e.g.,
 - development of BIT,
 - development of ATE,
 - recurring cost of BIT and ATE;
2. AFLC out-of-pocket costs, e.g.,
 - maintenance (flight line and shop),
 - spares,
 - development of ATE modifications; and
3. USAF Mission Costs, e.g.,
 - additional aircraft needed to replace those unavailable due to ECS failure,
 - aircraft lost due to in-flight failures of ECS,
 - unsuccessful missions due to in-flight failures or unavailable aircraft.

The optimum amount of ECS BIT that would minimize either out-of-pocket costs or the total USAF costs then could be readily calculated. If mission costs are included, the optimum amount of BIT will be considerably higher than if only out-of-pocket costs are included. Section 9.3 includes the suggestion that AFLC quantify this analysis for flight equipment, C-E, and ATD.

9.3 RECOMMENDED AFLC ACTION

The three actions will exert "design-to-test" pressure on AFSC.

1. Analyze historical failure data on hardware and software to determine the incidence of hard failures and transient failures of various types so AFSC may improve the design of ECS BIT circuits that simplify maintenance.

2. Conduct economic analysis of the amount and type of BIT and the mix of BIT and ATE that will minimize AFLC costs. AFLC could then present quantitative results to AFSC and to DSARC early in the system design cycle.
3. Ensure that redundant, fault-tolerant, and BIT-equipped ECS contain provisions to induce hardware and software faults in order to verify correct operation of BIT.

10. OPERATOR-COMPUTER INTERACTION

A wide variety of operator-interactive terminals is appearing and will continue to appear in AFLC. They employ different hardware and software and require different documents, tending, and repair.

10.1 THE PROBLEM

Computer terminals are used throughout AFLC and will be used ever more widely for:

- ISF: initialization, diagnostics, read-out of results, control of simulations.
- ATD: initialization, control, test case definition and selection, performance monitoring.
- ATE: test control, read-out of results, failure diagnostics.
- Text processing: prepare, edit, file, index, merge, transmit and format documents.
- Management information systems: collect, organize, index, file and format documents, budgets, schedules, drawings, etc.
- Computer programming: prepare, edit, file, index and transmit listings and documentation.
- Inventory: catalog, track, and issue assemblies, sub-assemblies, parts, and documentation.
- Computing: process requests, monitor and display results.

Operators most often receive information from computers or their terminals visually (volatile display e.g., CRT). They most often input via the terminals using such devices as keyboards, function keys, numeric pads, foot switches, joysticks, mice, lightpens, trackballs, tablets, wands, or touch surfaces. The operator-computer dialogue follows a language whose syntax and lexical units are enforced by the terminal, communication channel, host hardware, operating systems, and application program.

In 1980, terminals of many types are available all of which contain processors and memory, but there the commonality ends. Terminals fall into three logical groups:

1. limited host control (teletypewriter replacements plus simple line editing),
2. host controlled (most editing terminals),
3. host programmed (highly interactive systems).

This diversity will continue throughout the 1980's, causing a significant support problem within AFLC. First, programmers and engineers must learn the display characteristics of each terminal. Second, each terminal type requires unique support software. These characteristics tie the terminals, host, and software together, often forcing the customer into a long-term commitment to specific vendors regardless of cost escalation or support deterioration. Therefore, although interactive systems raise the productivity of programmers and engineers who use the system, the productivity increase is reduced as terminal diversity increases.

Further, the availability of LSI technology and the use of firmware have led to rapid obsolescence of terminal hardware. Today, a product may have a 2-year product life and a 5-year guarantee of replacement parts if the vendor is a major manufacturer. Smaller companies usually offer no guarantee, with 2 years after displacement of the product being the practical limit on specialized replacement parts such as ROM's and custom LSI chips.

10.2 STATE OF THE ART

AFLC purchases of operating system software usually include simple display format subroutines typically written in FORTRAN, BASIC, or COBOL. These subroutines permit the operator to display tabular data (e.g., arrays of numbers in rows and columns on the terminal) and edit text. The manufacturer-furnished operating system software includes device drivers that transmit data between the computer and terminals made by the same manufacturer. Where AFLC wishes to mix terminals and computers of different manufacturers, AFLC must usually write device drivers for the computer (in assembly language) and for the

terminal (if it is intelligent) and modify the application program. Many commercial operating systems contain several display logic systems, each different.

A variety of terminals are sold, whose internal intelligence varies from a single RAM to a complete microprocessor. As chip costs decrease, terminals will incorporate 16- and 32-bit ALU's, thousands of words of ROM, and hundreds of thousands of words of RAM. The proliferation of terminals in an ALC raises maintenance costs and decreases the productivity of operators when transferring from one terminal to another. Proliferation also increases the number of device drivers to be included in the host operating system and the diversity of application program interfaces.

The variety of software packages that can be purchased by AFLC will increase during the next 10 years. The corresponding operator-interactive packages can be divided into word processing and pictorial capabilities.

10.2.1 Word Processing

Various word-processing packages are sold that range from typewriter replacements to elaborate text editors. The operator actions, such as special typewriter characters, are unique to each manufacturer. No standardization existed in 1980 and none was foreseen due to competitive forces. With proper programming, data from one manufacturer's terminal could be displayed on another's. Insofar as operators (such as inventory clerks or word-processing typists) spend most of their time on one terminal, operator productivity is not lost from proliferation. However, when engineers must operate several different systems in a period of days or weeks, productivity is lost due to learning-time and errors.

10.2.2 Pictorial

Various software packages are available and more are in development that will display images such as photos and maps (Reference 10-1). AFLC will undoubtedly be procuring such software during the 1980's. The software in each package is unique; in many cases, the technology for creating the pictorials is not yet firm and several approaches can be

purchased. The first attempt at standardization has begun: Siggraph has created a standard for the format of messages to a subroutine that creates line drawings and accepts user inputs (Appendix D). The standard will probably be adopted by ANSI in 1981, after which AFLC should specify ANSI-Standard subroutines. Although the total software will be unique, the applications programmer will be able to call graphics subroutines that are addressed in a standard manner and are independent of the terminal. No Federal Information Processing Standards exist for computer graphics.

10.3 SUGGESTED AFLC ACTIONS

Three actions can be undertaken by AFLC to assist in resolving these problems.

1. Understanding AFLC needs.

- Create an inventory of computer terminals and graphic software.
- Determine the dialogue requirements for each user group to provide a base for analyzing terminal-software combinations.
- Determine which personnel use several different terminals or hosts in the normal course of their job to identify areas for standardization. Upgrading terminals for some applications may reduce costs by reducing the diversity of training, software, and spares requirements.

2. Establishing a Central Skill Center.

- Establish a central graphics-software skill center that supplies all of AFLC with device drivers and application-independent graphics software and maintains a central library of working graphics software. The skill center should maintain a testbed of terminals for software development. A central skill center would increase programmer productivity, increase the quality of the programs, and increase the commonality of software among terminals of different manufacturers at different centers.

3. Adopting appropriate standards.

- Participate in standardization committees because of the potential value of graphics and communications standards in AFLC centers.
- Specify the ANSI Core Graphics Standard where appropriate, once it is approved. Specify the ACM Siggraph interim standard.
- Specify, where appropriate, the Federal Information Processing Standards, when they exist.
- Consider developing internal standards among the ALC's for user dialogue, media compatibility, and message formats.
- In areas where operators interact with several terminals, confine purchases to as few manufacturers as possible.

APPENDIX A

TYPICAL DATA RATES FOR DISTRIBUTED AISF

If a networking capability could be made available between the five AFLC Air Logistics Centers, a capability would then exist for performing integration testing with complementary functions being performed simultaneously in separate locations. The typical support configurations from currently existing weapon systems discussed in this appendix are not to be interpreted as proposals to change existing support postures. They are configured strictly for the purpose of deriving expected typical maximum data burst rates. In addition, a basic assumption is that the support simulation is to be performed in real time. The three examples selected are

- E-3A support application,
- PAVE TACK AISF application, and
- SRAM AISF application.

A.1 E-3A SUPPORT APPLICATION

For the E-3A application, the support configuration examined is shown in Figure A-1 where the navigation/guidance subsystem support and the surveillance radar subsystem support are separated physically from the central E-3A processing function. The configuration is designed to perform integration testing of the avionic interfaces existing in the weapon system. Each of the subsystem support modules is assumed to have its own environmental simulation. Only timing signals would be transmitted between the subsystem modules.

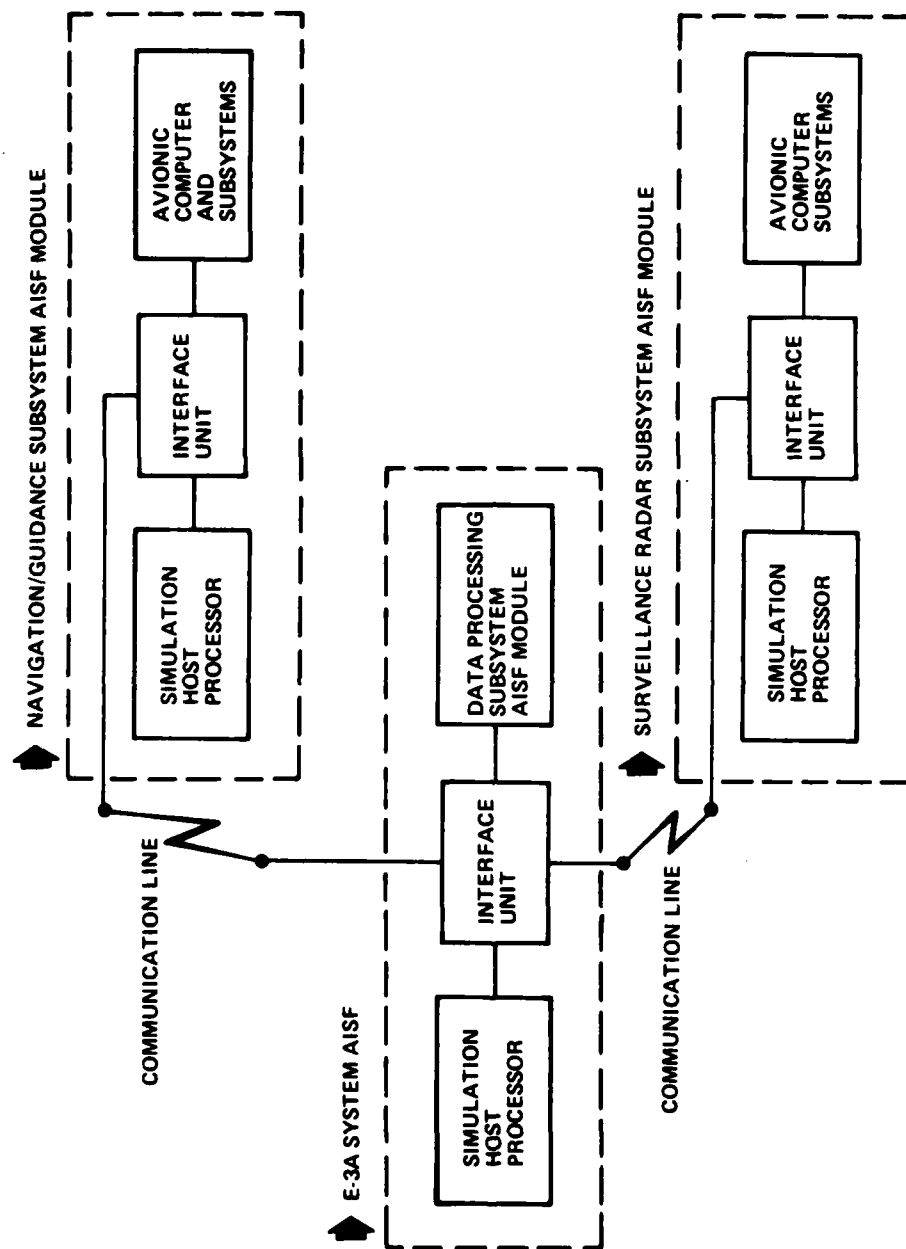


Figure A-1. E-3A Support Application

In the E-3A weapon system, the navigation data are one-way from the navigation/guidance subsystem to the Interface Adapter Unit (IAU) in the data processing subsystem. The data consist of

Parameter	Update Cycle, msec	Staleness, msec
Altitude	200	175.5 to 178.8
Latitude	100	22.8 to 23.9
Longitude	100	22.8 to 23.9
N. Velocity	50	22.8 to 23.9
E. Velocity	50	22.8 to 23.9
Ground Speed	100	22.8 to 23.9
Drift Angle	50	38.3 to 39.9
Heading	50	38.3 to 39.9
Pitch	25	8.0 to 11.2
Roll	25	5.5 to 8.8

Each parameter is transmitted in a 32-bit word including label, parity, etc. The derived maximum data burst rate from this data is 18.7 Kbps.

For the surveillance radar subsystem (SRS) interface, the data consist of radar commands, azimuth data, and navigation data from the IAU to the SRS and radar reports from the SRS to the IAU. The radar command data rate is minimal. The navigation data rate is a subset of that from the navigation/guidance subsystem to the IAU. This subset consists of roll, pitch, N. velocity, E. velocity, heading, and wrap-around test bytes. The words are 32 bits with a block of data transferred every 25 msec (N. velocity, E. velocity and heading are updated every other cycle.) The navigation data to the SRS require 15.1 Kbps. The azimuth data are encoded in 14 bits. One least significant-bit change can be transmitted in 100 μ sec, which corresponds to a maximum data rate for azimuth of 10 Kbps. The target reports, assuming 100 reports/sec and with each report consisting of three 32-bit words, correspond to 9.6 Kbps. Thus, the derived maximum data rate between the SRS AISF module and the E-3A AISF is 34.7 Kbps.

A.2 PAVE TACK AISF APPLICATION

For the PAVE TACK AISF application, the support configuration examined is shown in Figure A-2. The PAVE TACK AISF is at WR-ALC, the F-111F AISF is at SM-ALC, and the F/RF-4 AISF is at OO-ALC. The PAVE TACK AISF would be linked with only one of the two aircraft AISF's at a given time. The separate support capabilities are interconnected to permit performing integration testing between the PAVE TACK equipment and the F-111F or the F/RF-4 aircraft equipment. An alternative would be to duplicate much of the PAVE TACK equipment at OO-ALC and SM-ALC.

The mode of operation assumed in this analysis of the interaction of the PAVE TACK AISF with the F-4 or the F-111 AISF is that the aircraft AISF will provide the flight dynamics simulation, the PAVE TACK AISF will provide the simulated pod responses, and the weapons officer position would be active at the PAVE TACK AISF.

The data link between the aircraft computers and the PAVE TACK computer consists of a 100 Kbps duplex data link and a 50 Kbps display data link. Analog and discrete control and status indicators operate between the cockpit and the PAVE TACK pod. These signals and associated data rates are summarized in Table A-1.

The data list selected for this study provides for a flexible, closed-loop, simulation capability for the aircraft and the pod, with all cockpit operations relating to aircraft and weapons originating at the aircraft AISF, and PAVE TACK cockpit operations data simulated at the PAVE TACK AISF. This approach requires a dedicated channel for voice link communications between pilot and weapons officer, as well as detailed precoordination of test scenarios. A net reduction in the data link is possible, however, because the control and status signals, the pod video, and the 50 Kbps display data do not have to be transmitted to the aircraft AISF.

The principal disadvantage of this simplified configuration is that the PAVE TACK AISF does not have simulated flight displays showing the aircraft situation, but is restricted to the displays and controls

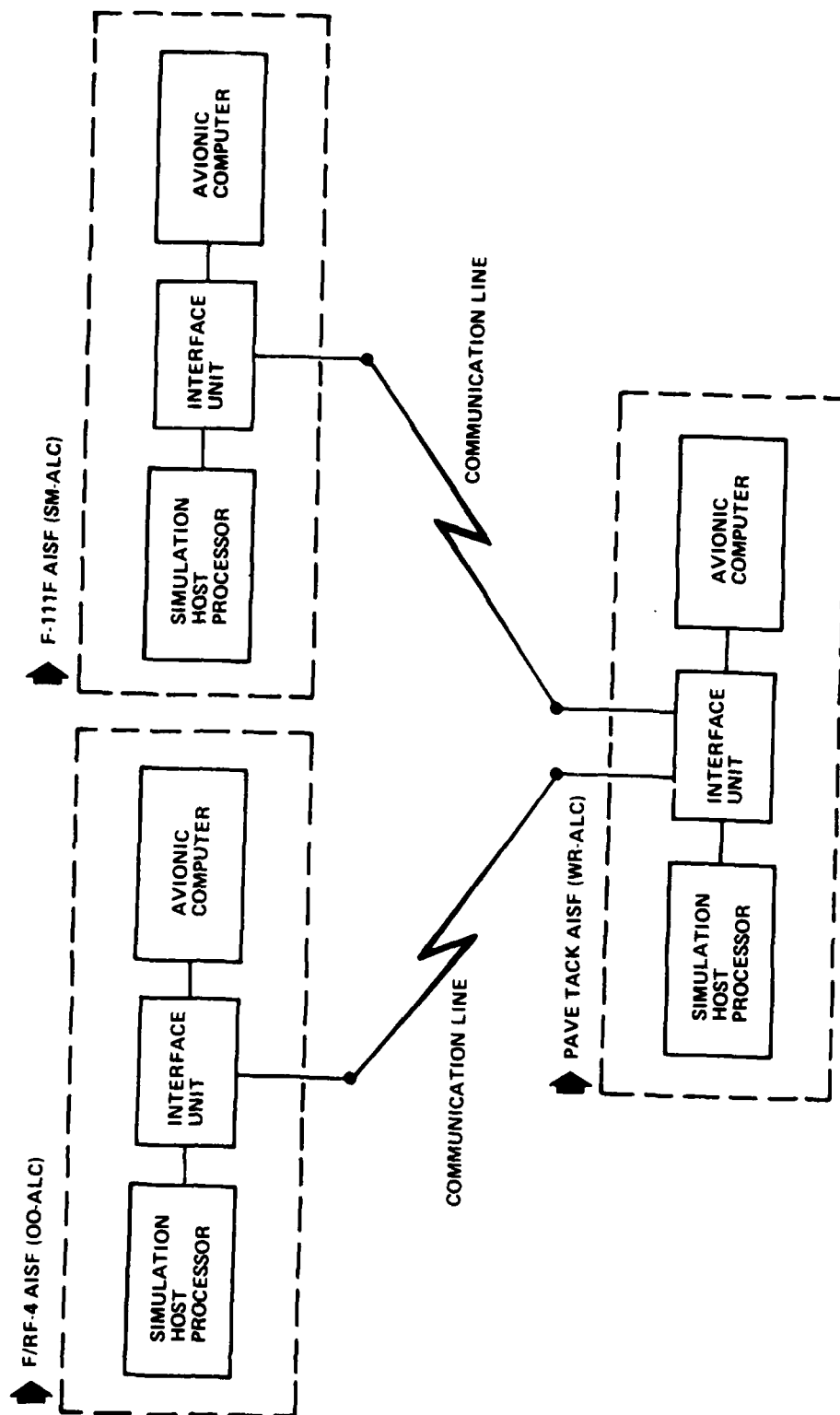


Figure A-2. PAVE TACK AISF Application

Table A-1. Aircraft/PAVE TACK Pod Interface

- Aircraft to Pod (100 KHz Channel)

Aircraft Position X	Double Precision
Aircraft Position Y	Double Precision
Aircraft Position Z	Double Precision
Aircraft Velocity X	
Aircraft Velocity Y	
Aircraft Velocity Z	
Aircraft Attitude Roll	
Aircraft Attitude Pitch	
Aircraft Attitude Yaw	
- Pod to Aircraft (100 KHz Channel)

Position X
Position Y
Position Z
Velocity X
Velocity Y
Velocity Z
Range
Range Rate
Altitude Rate
- Pod Video (Pod to Aircraft)

EIA standard RS 170 signal

- 50 KHz Display Data (Pod to Aircraft)

directly associated with the pod management. Thus, operation of the pod from the PAVE TACK AISF would not provide a high degree of fidelity of pod operation in a simulated operational environment. However, the fidelity is considered adequate for part-task simulation and interface verification.

The required AISF data rate for the aircraft-to-pod and pod-to-aircraft links is substantially lower than the 100 Kbps utilized in an operational aircraft/PAVE TACK system. The pod/aircraft data links are asynchronous to the pod computer and to the aircraft computer. These two data channels can be refreshed at the 40 Hz pod computer cycle rate with no loss of fidelity, and at a reduction to approximately 50 Kbps total channel rate, not counting pod video and the 50 Kbps display data link.

A.3 SRAM AISF APPLICATION

For the SRAM AISF application, the support configuration examined is shown in Figure A-3 with the SRAM AISF located at OC-ALC and the FB-111 AISF located at SM-ALC. The separate support capabilities are interconnected to permit integration testing between the SRAM equipment and the FB-111 carrier aircraft equipment. The alternative would be to duplicate many of the avionics subsystems at each of the Air Logistics Centers. If the SRAM AISF were to be moved to SM-ALC, a similar communication link to the B-52 AISF at OC-ALC would be required.

Several serial digital channels exist between the FB-111 carrier aircraft and the SRAM (Table A-2). The MK-IIB inertial navigation system (INS) data which needs updating periodically (serial channel V, 0_{15} , and I_{05}) adds up to a data rate of 20 Kbps. Allowing for overhead, a 50 Kbps line should be able to accommodate the data transfer. Transmission from the FB-111 carrier aircraft computer to the SRAM is performed on a 100 Kbps line. However, it is a one-time update and the only time-critical data are the fine-align update data which total less than 16 words. Thus, a 50 Kbps line should accommodate the total data transfer.

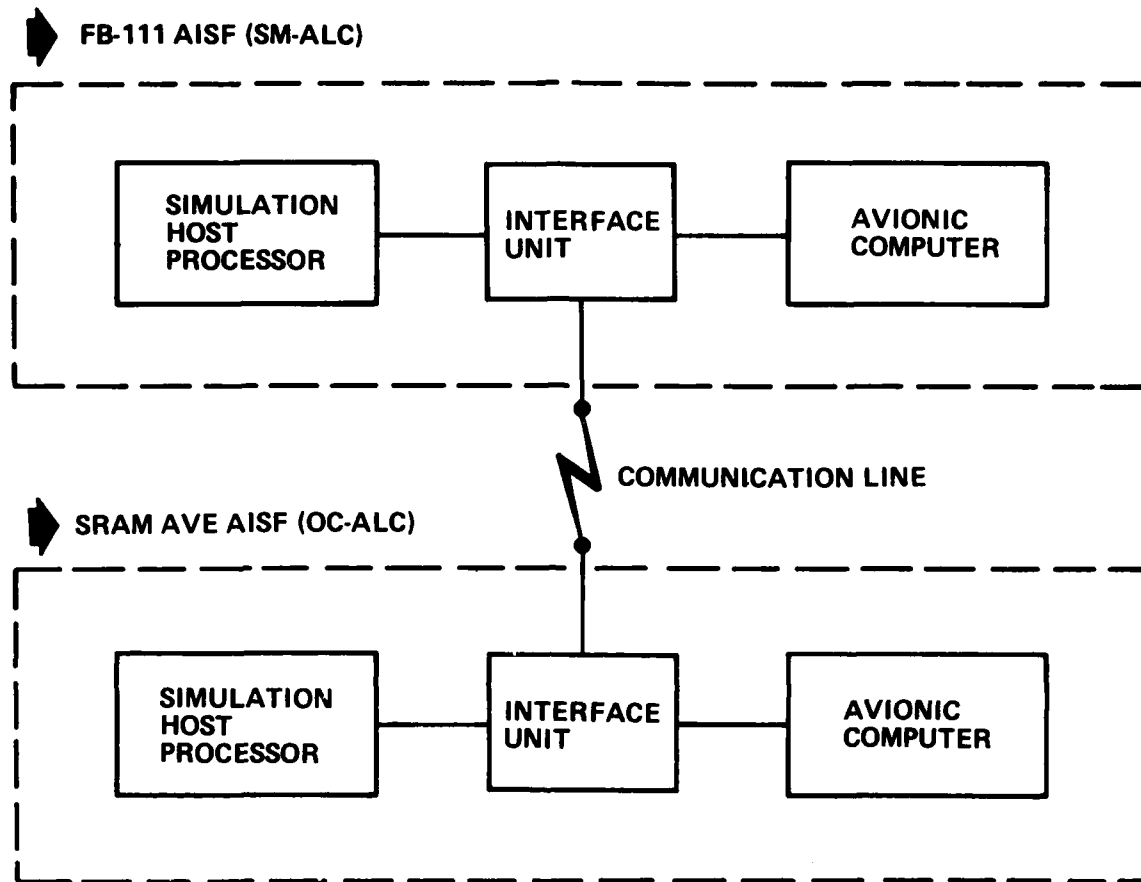


Figure A-3. SRAM AISF Application

Table A-2. Aircraft/SRAM Interface

● MK-IIB INS to SRAM (serial Channel V, 26-bit words)		
		<u>Rate</u>
Direction Cosine CX _x		32/sec
Direction Cosine CX _y	D.P. †	32/sec
Direction Cosine CX _z	D.P.	32/sec
Present Position Longitude	D.P.	32/sec
Present Position Latitude	D.P.	32/sec
Velocity X	D.P.	32/sec
Velocity Y	D.P.	32/sec
Velocity Z	D.P.	32/sec
INS Channel Status		32/sec
● MK-IIB INS Converter Set to SRAM (serial Channel 0 ₁₅ , 26-bits words)		
Radar Altitude		8/sec
Barometric Altitude		8/sec
Wander Angle		8/sec
Pitch Angle		8/sec
Roll Angle		8/sec
True Heading		8/sec
True FAT		1/sec
Static Pressure		1/sec
Latitude, Target	D.P.	1/sec
Longitude, Target	D.P.	1/sec
Target Latitude (Manual)	D.P.	1/sec
Target Longitude (Manual)	D.P.	1/sec
Target Elevation (Manual)	D.P.	1/sec
SRAM Command Word		1/sec
● SRAM to MK-IIB INS Converter Set (serial Channel I ₀₅ , 26-bit words)		
Channel Status		32/sec
Target ID		32/sec
Target Longitude	D.P.	32/sec
Target Latitude	D.P.	32/sec
Target Elevation		32/sec
● Carrier Computer to SRAM		
Course Align Initialization Data		13 blocks ‡
Level Inertial Capture Data		1 block
First Order Leveling Data		1 block
Fine Align Update Data		1 block
Targeting Data		11 blocks
Carrier to SRAM 100 KHz		
SRAM to carrier 75 KHz		‡

† Double precision.

‡ Each block 16 word maximum.

‡ Each block transmitted back for verification.

APPENDIX B

DATA COMMUNICATION COST ESTIMATES

The costs for transmitting data at a rate of 50 Kbps from McClellan AFB, California to Robins AFB, Georgia have been estimated for several situations. The cases include the transmission of both digital and analog data over both dedicated and switched AT&T facilities. In addition, cost estimates have been prepared for transmitting a 1.544 Mbps video channel and a 50 Kbps data channel via a satellite and via dedicated ground stations. Costs for handling 50 Kbps have been included based on the use of 12 satellite voice channels and AT&T facilities from San Francisco to McClellan Air Force Base and from Atlanta to Warner Robins. Data Pro reports were used to establish AT&T tariffs. Cost estimates include Category A/B service and rental expenses.

The most desirable method to transmit the 50 Kbps signal is via the wide-band DDS at a cost of approximately \$11,277 per month. Although six 9.6 Kbps analog channels appear to cost less than the digital service, the final system cost will probably be greater due to the required handling of six channels. Also, the quality of the digital DDS is much better than the analog lines and is therefore preferred.

The most practical method for handling the 1.544 Mbps video data is via satellite. The cost estimate for a satellite 1.5 Mbps channel and a 50 Kbps channel, including dedicated ground stations, is \$47,800 per month. This is considerably less costly than a 1.544 Mbps terrestrial channel. A 3.0 Mbps channel, instead of the 1.544 Mbps channel, could be obtained for an approximate additional \$15,000 per month. The satellite system costs are based on data quoted by American Satellite Corporation and are for a full duplex operation. Costs include ground station and digitizer expenses.

If a 50 Kbps capability (i. e., without a 1.544 Mbps channel) is adequate for handling the data requirements, the 50 Kbps broadband DDS Service is recommended.

It should be pointed out that data circuits from an AT&T Category A city (such as Atlanta) to a Category B city (such as Warner Robins), a distance of about 80 miles, costs half again as much as service from Sacramento to Atlanta. Also, switched circuits are only practical for very brief periods. For 4 hours per day, 22 days a month, they are much more expensive to use than dedicated circuits.

Table B-1 presents a summary of these estimated costs. Figure B-1 illustrates the data communication paths.

Table B-1. Data Communication Cost Estimates
(McClellan AFB to Robins AFB)

Data Type	Dedicated Circuit [†]	Dollars per Month	Switched Circuit [†]	Dollars per Month [‡]
Digital	DDS @ 50 Kbps	\$11,277	DSDS @ 50 Kbps	\$15,424
Analog	8800 B @ 50 Kbps	\$15,811	DP @ 50 Kbps	\$24,352
Digital	DDS @ 9.6 Kbps	\$12,335	Not offered	
Analog	3002 @ 9.6 Kbps	\$10,075	DDS @ 4.8 Kbps	\$27,918
Satellite (Dedicated Terminal) @ 1.544 Mbps and 50 Kbps				
Satellite (AT&T Long Lines)	3002 @ 50 Kbps			\$47,800
				\$21,830

[†] DDS Digital Dial Service
(Digital Dedicated Line)
DDD Direct Distance Dial
(Analog Switched Voice Time)
DSDS Digital Switched Dedicated Service
(Digital Switched Time)
DP Dataphone
8800 B Analog Dedicated Time
3002 Analog Voice Grade Time

[‡] Cost is for 9-hour day,
22 days per month.

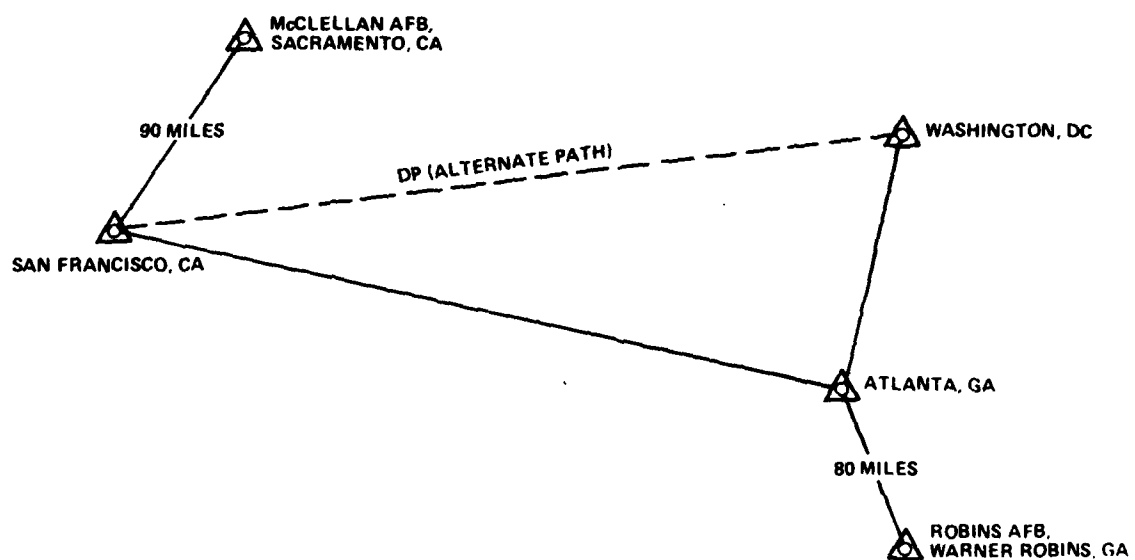


Figure B-1. Data Communication Paths

APPENDIX C

STANDARD MICROCOMPUTERS

The key to developing and integrating operational and support system hardware and software is the standardization of interfaces to single board microcomputers.

The standardization criteria should channel the development of these interfaces in the proper direction but not constrain the innovation and creativity of the engineers and programmers. Major standardization areas are

- high speed bus and card size;
- I/O and controller interfaces;
- multitasking multiprocessor operating systems;
- subroutine construction, data communication, and documentation protocols;
- software development tools and debugging facilities; and
- programming languages.

In this discussion, the computational power is assumed to reside in a microcomputer board, special purpose processors, and boards which communicate through a common highspeed bus. Figure C-1 shows a two-bus architecture of this type proposed for a ground support system which is used to reprogram and verify changes to electronic warfare applications programs. This architecture is typical of a multi-processor microcomputer system designed to satisfy a particular application.

The first level of standardization is with the bus and the board size. The bus and the board size must be standardized to accommodate many applications. The strategy is to make it advantageous for various

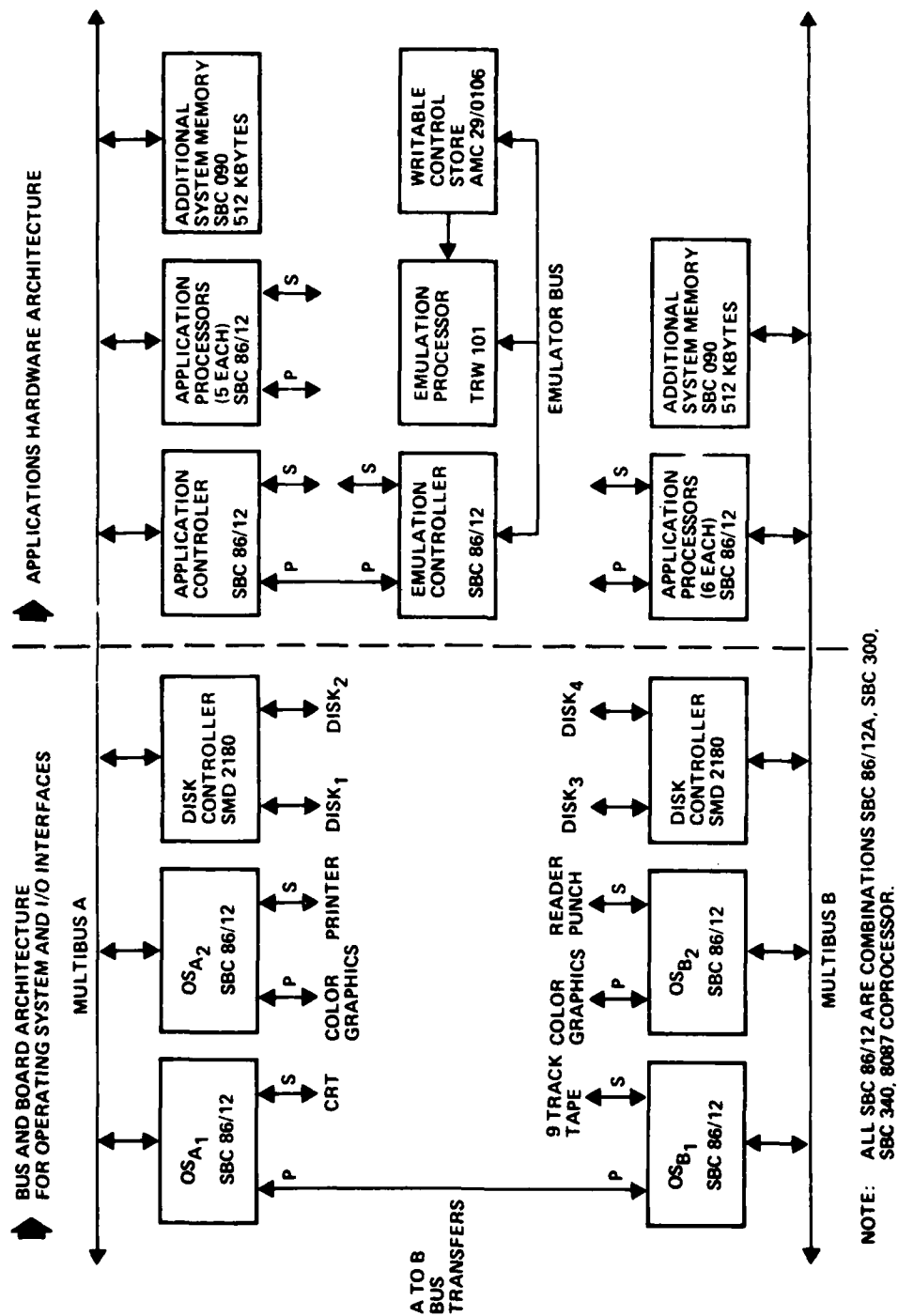


Figure C-1. ARC 2 Bus Multiprocessor Hardware Design

manufactures to design products which will interface with the bus and thus create a competing situation, reduce cost, and provide the system designer with a rich variety of boards from which to choose. A prime example of this phenomenon has occurred in the commercial marketplace around the INTEL Corporation MULTIBUS and board size. More than 30 manufacturers produce boards to interface to this standard bus. A designer selecting this bus can most likely integrate his system by selecting off-the-shelf boards without any need for expensive custom designed components.

A number of standard I/O interfaces have received acceptance for serial, parallel, and high-speed disk communication. To remain competitive, I/O equipment manufacturers have engineered their products to interface to these standards. In turn, controller manufacturers must design so that their interface to the I/O gear meet these same standards. For example, all of the I/O equipment shown in the system described on Figure C-1 uses standard serial, parallel, and disk interfaces and can be integrated from a rich selection of commercially available products. The I/O equipment shown was tailored specifically for the application that this system is to address. The disks each have 96 megabytes of storage capacity, the printer operates at 600 lines per minute, and the color graphics devices are high resolution image processors. The P, S, and SMD represent the standard parallel, serial, and disk interface, respectively.

The adoption of a standardized real time event driven multitasking multiprocessor operating system has the effect of imposing engineering disciplines on software designs and establishing standard data structures, software system interface protocols, and architectures that guide and channel the organization of application program modules. The applications programmer is thus provided with simple and concise interfaces with other application modules and with the I/O and priority structure embedding

in the operating system. The level of expertise needed to develop application modules is reduced and hence, the more skilled personnel can dedicate effort to providing the applications programmers with interfaces that most affectively and efficiently utilize the system hardware. Within the application modules, standardized subrouting structure, subroutine data communication protocols, and internal subrouting documentation can easily be adopted.

Most of the software development support tools used to produce programs for military applications are commercially available products which perform essentially the same function for various microcomputers. Each computer vendor has a software debugging monitor, a disk operating system, an editor, an assembler, and various high order language compilers. Problems develop because the human interface to the support tools developed by various manufacturers varies. Because most military systems will have a variety of embedded microcomputers, the applications programmers must be familiar with a number of human interfaces, although each of the development systems performs the same functions. A standardization of these human interfaces to the software development tools would result in the more efficient production of high quality applications software.

The final level of standardization is programming languages. At the assembly language level, each microprocessor has its own instruction set. However, the various assemblies can be standardized with respect to definition of macros and memory usage and integration of a common set of assembler controlling pseudo-operations. At the high-order-language level, most manufacturers of software development systems are adopting standard syntax (e.g., FORTRAN, COBOL and PASCAL) as proposed by various standardization committees. Various dialects of these languages should be discouraged because the slight advantage that might be gained in a particular application is offset by the degradation of the standard.

APPENDIX D

SIGGRAPH STANDARD

Much of the power of computer graphics lies in its capability to present information in a variety of forms. A computer graphics questionnaire distributed by the National Computer Graphics Association contains a comprehensive list of various graphic capabilities. It is expected that over a 10-year period, most of these capabilities will be needed by various groups within the AFLC.

The wide diversity among graphic hardware devices has been a basic problem in computer graphics from its beginning in the early 1960's. This diversity will be even more prevalent in the 1980's. Software has become the key issue (i. e., major cost item) in computer graphics because of both this diversity, and the sophisticated nature of the graphic devices themselves. Interactive computer graphics is growing too rapidly and becoming too sophisticated to allow each such facility to develop its own software.

To help solve this software problem and achieve computer graphics program portability, a standarization effort has been taking place over the last few years. As a result of this activity, a new ANSI technical committee X3H3 was formed in June 1979 to develop an American National Standard for Computer Graphics Programming Languages. The basis for this standard is to be the Core system developed by the Graphics Standard Planning Committee of the ACM Special Interest Group on Graphics (SIGGRAPH). It is currently projected by X3H3 that a draft proposed graphics standard will be completed by December 1981.

The Core system is primarily oriented towards interactive vector graphics. The new ANSI standard will include both stroke and raster capabilities. The Core system was designed to be useable by a wide range of applications, ranging from those requiring static plotting to

those requiring dynamic motion and real time interactions. Consequently, four orthogonal classes of upward compatible levels are specified: output, input, dimensionality of the coordinate space (2D or 3D), and hidden surfaces. Each class contains specifications for two to four levels of capability. For example, the output levels are basic using only temporary segments, buffered using temporary and retained segments, and dynamic offering three levels of image transformation.

In addition, the X3H3 committee is defining a small Core which will be a relatively small set of useful graphic functions accessible to the application program. The various partitionings of the Core system will allow implementations with higher performance and less resource demands for those applications needing real time response, and not a full set of graphic functions. This, in general, will counter the claim that such standards compromise speed of response.

Currently there are approximately 50 graphic software packages in various stages of development which implement the Core system. These have been developed by both industry (e.g., TRW DSSG) and universities (e.g., George Washington University). Several of these packages are sold as commercial products. This includes DIGRAF from the University of Colorado, and DI-3000 from Precision Visuals. License fees range from \$4000 to \$8000. Various graphic vendors (e.g., Aydin, Tektronix, Vector General) also have software packages which are similar to the Core system. The number of commercially available software packages which conform to the Core, and include raster capabilities will rapidly increase in the next few years. It is recommended that AFLC procure a current Core-type package, and also continue to keep abreast of the ANSI graphics standards development activity.

The ANSI standard, like the Core standard, does not include all the graphic software needed by an installation. Higher level functions such as high-quality text generation, curve display, map making, and

texturing routines would sit on top of the Core. It is expected that these functions will be provided in Core supported packages in the future. Other graphics support may include aids for training and education. This particular area is being investigated at NASA Johnson Space Center.

A current research area that will impact graphics software in the 1980's is the development of data base management systems employing graphic data structures. The user would employ spatial location and visual appearance of information in order to find it. Current systems include the Spatial Data Base Management System of Computer Corporation of America and the Picture Building System of IBM Research Lab.

Another important trend that will impact the type of graphics software in the AFLC environment is the use of customized intelligent graphic terminals. Each user could define his own high-level graphics presentation language appropriate to his task. He could specify his own function keys and means for manipulating the displayed objects of interest. To support this capability it would be necessary to design a general purpose graphics command interpreter to be used on the top of, or together with a Core supported system. The ability to interactively design the user dialogue is under development at TRW DSSG. Research into new methods of graphics interaction are currently being explored by the Architecture Machine Group at MIT. VLSI technology will also impact the development of intelligent graphic terminal software. Specialized algorithms needing high performance such as hidden surface removal may be implemented into VLSI chips.

Networking will play a key role in the AFLC environment. The graphics system software should support incorporation of images that may have been generated at other AFLC nodes. Attention is currently being focused on developing computer graphic protocols to support this capability. Other research efforts include work in pictorial data encoding of raster images, and segmentation schemes. The ability to move

images around a network will be needed in display conferencing systems. Display conferencing will permit the conferees at different locations to simultaneously view common information displays, and possibly interact with these images with a pointing device in a real time exchange of information.

Rand Corporation is currently building a network-oriented color graphical conferencing system under the ACCAT program. Users share a common display, which is divided into sections: a individual section for each user and a common graphical "blackboard" window where the users may take turns sketching out, and modifying a graphical display in full color.

Computer graphics technology will undergo a tremendous growth in the 1980's. It is difficult to know at this time which capabilities will move out of the research environment, and be viable for the AFLC.

AD-A115 460

TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA

F/8 9/2

A STUDY OF EMBEDDED COMPUTER SYSTEMS SUPPORT. VOLUME VIII. ECS —ETC(U)

SEP 80

F33600-79-C-0540

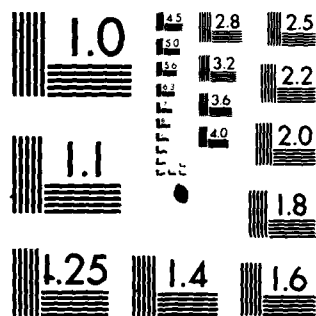
UNCLASSIFIED

TRW-34330-6003-UT-00-VOL-

ML

2-2
PAGE

END
DATE
FILMED
7-82
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

APPENDIX E

HIGH ORDER LANGUAGES

A programming language is a formal grammar which facilitates the expression of actions to be performed by a digital computer system. The earliest programming languages, assembly/machine languages, provided only for specifying directed action in terms of computer system attributes (registers, memory addresses, etc.). Alternatively, subsequent development of high order languages (HOL's) permitted expression of desired computer system action in terms of problem domain (files, records, data items, etc.).

The use of any particular HOL is made possible by the existence of a compiler for that language. A compiler is a computer program which translates HOL language statements into machine code. A compiler is, therefore, language specific and machine dependent, in that the compiler has knowledge of the language grammar for a specific HOL and produces code for a specific machine.

Since the earliest high order languages were developed about a quarter of a century ago, the proliferation of such languages has been quite remarkable. Sammet (1978) has identified 166 programming languages that are in common use (i. e., users other than the language developer).

The use of high order languages leads to a substantial increase in programmer productivity. Programs written in a HOL require about the same effort to design, half the effort to code and test, and one-third the effort to maintain as the same program written in assembly language. Several properties of HOL's lead to this increase in productivity. High order languages are easier to learn than assembly languages, largely because the notation is more natural to intended problem areas. Therefore, less specialized training is required of programmers, and writing

programs for different machines requires little additional learning. The more natural notation of HOL's also makes programs easier to write. Because the programmer can concentrate on the problem to be solved, the need to address low-level design problems is reduced. Furthermore, because each statement in a HOL normally translates into many machine language instructions, a program written in a HOL is usually much shorter than an equivalent program written in assembly language. The effort to produce a given number of lines of code is independent of language has a large effect on productivity.

Because high order languages are much more machine independent than assembly languages, programs written in a HOL are potentially portable; i. e., such programs theoretically can be run on completely different machines, with only minor changes. Programmer time spent on conversion is thus reduced, and the amount of code which can be used in more than one application is increased.

Programs written in a HOL are likewise easier to read. The natural notation contributes to self-documentation, and the problem-oriented notation promotes modularity and structured design. These factors increase programmer productivity in the sense that programs written in a HOL are easier to debug and to modify.

High order languages provide better software tools than assembly languages, which simplifies the debugging of programs written in HOL's. The most important of these tools is the compiler. In addition to automating part of the software development process, a compiler can detect errors in intermodule data interfaces and can aid in auditing conformance to programming standards. A compiler can also provide a convenient interface between the program under development and other software tools such as debuggers.

The time saved in the design, coding, and debugging stages of software development can profitably be used in increasing pre-delivery testing time, thus leading to more reliable software. This is particularly important because operation and maintenance costs currently are estimated as 30 to 70 percent of total life-cycle costs.

Reliability is increased by the use of HOL's in other ways as well. The number of errors in a program is, in many cases, proportional to the number of lines of code; thus the shorter HOL programs will have fewer errors. In addition, a prime source of errors, data type conversions, may be controlled by the compiler, while other errors are easier to find because the natural expression characteristics of the HOL makes the program easier to understand.

It has often been argued that compilers cannot generate machine code which is as efficient as that written by an assembly language programmer. While this statement has had some historical validity, modern compilers can produce optimized code which is as acceptable as that produced by experienced assembly language programmers.

The three programming languages discussed in the Sections E.1, E.2, and E.3 are JOVIAL, Ada, and ATLAS, respectively.

E.1 JOVIAL, CURRENT LANGUAGE FOR ECS SOFTWARE

The JOVIAL language has been used by the Air Force since 1959. JOVIAL is a derivative of ALGOL and was specifically modified to support command and control systems. As an ALGOL derivative, JOVIAL provides the block structures necessary for implementation of structured programming software designs.

JOVIAL has suffered from a proliferation of dialects and minor differences between implementations. In 1967, a version of JOVIAL, J3, was established by the Air Force as its standard programming language for command and control systems. In 1972, a committee report was accepted to modernize J3. The new dialect, J73/I, was adopted as the official Air Force standard, but a JOVIAL implementation called J3B was developed based upon a preliminary report from the modernization committee. Due to schedule considerations, J3B was used on several operational flight programs (F-16 and B-1) and underwent further modifications picking up strong typing rules and tighter control of inter-compilation unit interfaces.

DOD Instruction 5000.31, "Interim List of DOD Approved High Order Programming Languages," was issued on November 24, 1976. The intent of this DOD instruction was to reduce the proliferation of high order languages in defense systems and to ensure that languages which are approved are properly controlled. The Instruction states that only DOD approved high order programming languages may be used for new defense system software, specifies the list of approved languages, and assigns the control agent for each language.

The initial list of approved languages consists of CMS-2 (dialects CMS-2Y and CMS-2M), SPL-1, TACPOL, JOVIAL (dialects J3 and J73), COBOL, and FORTRAN. Control responsibility for each language was assigned to the primary user of the language, if one could be identified. The Navy was given control responsibility for CMS-2 and SPL-1, the Army control of TACPOL, and the Air Force control of JOVIAL. FORTRAN and COBOL were to be controlled jointly by the Office of the Assistant Secretary of Defense, the National Bureau of Standards, and the American National Standards Institute.

Air Force Regulation 300-10, issued on December 15, 1976, further limits languages which may be used in new Air Force systems to FORTRAN, COBOL, and JOVIAL J73/I and J3. As of early 1980, JOVIAL J3 is no longer supported; thus JOVIAL J73 is the only acceptable version. A compiler for JOVIAL J73, however, has not yet been validated.

Two dialects of JOVIAL were included in the list of approved languages provided by DOD Instruction 5000.31 and AFR 300-10: J3 and J73, as defined by MIL-STD 1588 and MIL-STD 1589A, respectively. The choice of J3 was relatively non-controversial: it was well-established and had been used in several major Air Force systems. The choice of the second dialect of JOVIAL, J73/I, was not as clear. J73/I was a relatively new language, not in particularly wide use, and many users felt that another JOVIAL dialect, J3B, used in the B-52 program,

would have been a better choice. In late 1978, the AirForce initiated an effort to standardize on a single dialect of JOVIAL by incorporating the proven capabilities of J3B into the otherwise more modern J73/I. This upgrade incorporated features from JOVIAL J3B Version 2, and remedied JOVIAL J73/I deficiencies. The revised language is known as JOVIAL J73. To date, J73 compilers have not been formally validated and accepted; however, some are running and several more are being developed. Table E-1 summarizes the status of J73 compilers.

J73 will be used to develop operational software and support software for the next several years. At some point in the future, Ada (Section E.2) will replace J73 for new software, but programs written in J73 will be used for many years to come. Because Ada will not be mature enough to use for developing operational or support software for several years, the use of J73 as a standard language during these interim years will be reflected in increased cost saving, reliability, and portability. AFLC should prepare to support operational software written in J73 by developing associated tools and skills now.

E.2 Ada: FUTURE LANGUAGE FOR ECS SOFTWARE

In addition to reducing the number of approved high order programming languages via DOD Instruction 5000.31, the Department of Defense is conducting an extensive development effort aimed at achieving a new common HOL, now called Ada.

Ada is primarily intended as the principal programming language for embedded computer system software in a new Department of Defense projects. This type of software is currently done largely in assembly language. Ada is not restricted to embedded computer applications, however, and will undoubtedly be used in a wide range of applications, including command and control, avionics, space systems, telecommunications, intelligence, surveillance, and simulation. Ada is not only expected to become a truly common HOL in the Department of Defense

Table E-1. J73 Compilers

Developer	Host	Target Computer	Date Available
SEA (AFWAL)	Dec-10	Dec-10	now
		1750	now
		362-F	now
		AYK-15	now
		1750A	late 1980
	360/370	360/370	late 1980
		1750	late 1980
		362-F	late 1980
		AYK-15	late 1980
		1750A	late 1980
Softech (RADC)	306/370	360/370	late 1980
		1750	late 1980
		175A	late 1980
	6180	6180	late 1980
	8/32	8/32	early 1981
Softech (Army) Pershing	370	Bendix 920	now
Softech (BMO) MX	360/370	MECA	late 1980
(AD) DIS	3033	PDP 11/34 Z8000	Dec 1980
SEA (TI)	360/370	TI 990/9900	late 1980
SEA	VAX 11/780	VAX 11/780	?

it is also expected to be a medium for technological interchange among the NATO military establishments and a common HOL in the European Defense community as well.

The Ada programming language was designed in accordance with the Steelman requirements of the Department of Defense (Steelman 78). Three general Steelman criteria were considered most important in the design of Ada: (1) to encourage the construction of reliable and easily maintainable Ada programs, (2) to be as simple as possible, in deference to the very real difficulty of programming as a human activity, and (3) allow for the construction of efficiently executable Ada programs.

To promote reliability and maintainability, the Ada designers emphasized program readability over the ease of writing programs. Thus, Ada programs may require slightly more effort to compose properly, but once composed, they may be (1) more likely to be correct and (2) easier to change. Useful forms of redundancy are required in Ada programs, and Ada compilers must check for consistency. Error prone notations in other languages have been avoided, and English-like constructs are generally used in preference to obscure abbreviations. To support economical program development and maintenance, Ada programs may be composed of many separately compiled parts without sacrificing the consistency checking properties. Thus, revisions of Ada programs can be inexpensive without allowing certain insidious errors to creep in undetected. The Ada language design is clearly within the state of the art of compiler technology.

The following observations can be made

- Ada will be well suited to nearly all applications handled today effectively by FORTRAN or JOVIAL; in fact, Ada should be an improvement in most cases.
- Ada will be well suited to the development of ECS and support software.

- Ada has potential in operating system, communication, and data base applications; however, some refinements to the language may be necessary following test and evaluation in order to realize this potential.
- In a recent study sponsored by the Air Force Avionics Laboratory, Ada emerged with the top score of 95 percent compared to 74 percent for the nearest competitor (JOVIAL) in an assessment of 52 HOL features covering seven different categories.
- Ada may be ideally designed to interface with program design support tools, and the integration of compilers and support systems will be a Department of Defense goal of potentially great benefit to ECS applications.

The ultimate success of Ada in becoming an accepted production quality programming language in the Department of Defense depends upon an effective environment for Ada support. Such an environment has organizational and administrative aspects, as well as technical and software aspects. Thus, the Department of Defense document, "Requirements for the Programming Environment for the Common High Order Language (Stoneman revised)", describes not only the Department of Defense policy and administration requirements, but also software development and maintenance tools requirements for a support environment.

The technical and software requirements are many, including,

- a firm and clear definition of Ada;
- numerous efficient compilers producing efficient object code, hosted on and targeted for a variety of machines;
- technical means for confirming that these compilers all satisfy the definition of Ada;
- reliable, easily maintainable software tools to assist in the design, testing, and debugging of application programs;
- additional tools to assist management in the development and testing of such application software; and
- a means of adding to the set of tools at any installation, including the ability to transport tools from other installations; thus, there must be a basic common structure of the underlying interfaces between compilers and other tools.

Several administrative and organizational requirements have been imposed on the environment. In particular, the Department of Defense has identified that there must be:

- an Ada Control Board (ACB) which will be responsible for maintenance and control of the definition of the Ada language;
- an Ada Validation Facility (AVF) which will certify that translators are complete and correct implementation of Ada and will report this information to the ACB;
- an Ada Support Facility (ASF) which will develop, procure, and maintain a basic set of tools for Ada environment, as well as disseminate information about the language and tools.

The Department of Defense will also encourage industry to produce additional tools which support the language and will fund research on new techniques and methods. User groups with interest in particular applications will be encouraged and supported by the ASF.

The quality of the Ada design as well as the strong support of Ada by the Department of Defense and the growing acceptance of Ada by industry, increase the likelihood that Ada will succeed as a common language for both general purpose and military applications. The current challenge is to develop a programming environment that will allow the potential advantages of Ada for program development and maintenance to be realized in practice. The volume of Ada usage is predicted to exceed that of FORTRAN usage by 1995. An Ada compiler for the VAX 11/780 is currently under development by Softech for the U.S. Army.

E.3 ATLAS: LANGUAGE FOR ATE SOFTWARE

Approximately three-fourths of all Air Force aircraft-related software today is Automatic Test Equipment (ATE) software. This situation indicates the relevance of high order languages to ATE software development.

An OSD memorandum of 14 July 1978 declared the Abbreviated Test Language for All Systems (ATLAS), as defined in IEEE Standards 416-1976 and 416A-1976, to be the Department of Defense ATE language standard for test specification and test procedures.

ATLAS is a standardized test language for expressing test specification and procedures. It is a test-oriented language independent of test equipment and provides a standard abbreviated English language used in the preparation and documentation of test procedures which can be implemented either manually or with automatic or semi-automatic test equipment.

ATLAS was developed originally for avionics applications under the auspices of Aeronautical Radio, Inc. (ARINC) and under the direction of the Airlines Electronic Engineering Committee (AEEC), which approved the original version on October 10, 1968.

ATLAS programs are written in much the same way as test specifications. ATLAS programs are self-documenting (i.e., a program can be read and understood without the background of a computer science expert).

The ATLAS language has the following general characteristics:

- The language is dedicated to defining the test requirements of the Unit Under Test (UUT) with neither reference to nor dependence upon the test equipment which may be used. The test equipment may be an automatic, a manual, or a hybrid design.
- The selective use of English language terms which are compatible with the description of test requirements and of a formal structure for their use constitute an environment to ensure an unambiguous description of the requirements of a test procedure for the UUT designers, developers, users, and maintenance personnel.
- The specification of test requirements in terms of the UUT by ATLAS facilitates the transportability of those test specifications from implementation on one set of test equipment to another, providing all of the test requirements can be satisfied by the test equipment.

The three levels of ATLAS (Standard ATLAS, Subset ATLAS, and Adapted ATLAS) are distinguished as follows:

- Standard ATLAS is a complete language, including vocabulary, syntax, and rules, as described in ANSI/IEEE Std 416-1976, ATLAS Test Language, IEEE Std 416 and IEEE Std 416a.
- Subset ATLAS is a language where every constituent construct is included within Standard ATLAS (i. e., no extensions) except that, for commercial or technical reasons, Subset ATLAS does not include all the vocabulary or statement and syntactic options of Standard ATLAS. It is anticipated that each Subset ATLAS is unique; thus specific forms of Subset ATLAS have not been defined.
- Adapted ATLAS is a family of languages which conform closely to ANSI/IEEE Std 416 but which have modified vocabularies and may have syntax distortions.

The different versions of Adapted ATLAS are normally closely associated with the specific hardware employed in various test systems currently in use or being designed.

While ATLAS is used extensively and significant effort has not been initiated to supplant the language, it is not without limitations. Reservations regarding the use of ATLAS are focused in these general areas:

- Most important, ATLAS only encodes a test program. ATLAS neither generates a test program nor has any facility for modeling a circuit and checking the veracity of a test program.
- Very few ATE manufacturers adhere precisely to ATLAS. Many subsets, or extended versions, of ATLAS exist because it is not flexible enough to meet all the requirements of the real world.
- ATLAS is mostly digital and does not sufficiently address analog testing.

- The transferrability of programs, while important to the military, is not significant to industrial users. Industrial ATE users have better trained personnel and maintain much less ATE than does the military.
- ATLAS is oriented toward board-level field testing. Manufacturers have found extensive component-level testing to be more cost effective than detailed board-level testing. Manufacturers need to get a product tested and out the door and are not as interested in maintaining that product for 15 years as is the military.
- ATLAS is useful only for large computer-driven ATE and is not applicable to smaller, benchtop ATE because such equipment is very simple and very hardware specific.

APPENDIX F

COMTEC - 2000 SUMMARY

F.1 BUILT-IN TEST

F.1.1 Summary of BIT

- Increased BIT in complex equipment, including ECS hardware and software.
 - LSI/VLSI aggravates the need for more BIT. Chips will have increased self-test capabilities.
 - Increased ATE for preflight/inflight/postflight equipment monitoring will require new ATE support systems.
 - Further study required on level of continuous BIT without disrupting normal operation of equipment.
- Need ability in support equipment to introduce hardware faults into hardware and software faults into software.

F.1.2 Excerpts from COMTEC-2000

Several technological issues need to be addressed and analyzed in greater detail. Additional hardware research is required to determine optimal instrumentation for built-in-test and status monitoring without interrupting normal operations. This is particularly true when considering continuous monitoring at the chip or component level. The feasibility of approaches such as wafer-testing points or on-chip diagnostics should be evaluated.

Further study is required to determine the optimum balance between hardware mechanization (i. e., built-in-test) and software mechanization (functionally integrated testing). Such a study should address the requirement satisfaction benefits of both approaches versus the cost to develop and maintain the capabilities.

Implementation of enhanced inflight diagnostics would lower the requirement for and dependence on automatic test equipment. Squadrons could become more autonomous and more readily deployable. Immediate failure identification could compress turnaround times and enhance operational readiness rates. This should be pursued as a high-priority requirement.

Prelaunch/post-mission diagnostics appear technologically feasible. Further research should be conducted to determine the exhaustiveness of testing required to achieve various mission accomplishment confidence levels versus the cost to implement and maintain these capabilities.

An inflight data collection capability should be developed concurrently with enhanced diagnostic development.

Meaningful parameters and techniques for inflight failure prediction should be identified and such techniques should be applied to critical subsystems in next generation weapon systems.

No required system test and evaluation functions appear technologically infeasible. From a software standpoint, nothing proposed appears to be beyond the current state of the art. Unfortunately, implementation of these functions can be expected to be increasingly complex and thus more expensive to engineer. From a systems standpoint, current technologies appear adequate to satisfy the requirements, but further study is required to determine the optimum balance between built-in-test and software mechanization.

The evolution of large scale and very large scale integrated devices will exacerbate the system testing problem.

F.2 FAULT-TOLERANT COMPUTER ARRAYS

F.2.1 Summary

- The use of digital systems for flight safety control functions will push the state of the art in software and hardware reliability. Simplicity in hardware circuitry and software logic may provide the means for adequate testing and inherent reliability.

- Distributed processing should improve the reliability to long-lived systems.
- Distributed avionics architectures promise graceful degradation.
- Software and hardware reliability need further development.

F.2.2 Excerpts from COMTEC-2000

Distributed avionics architectures promise much in subsystem interchangeability, enhanced system reliability, graceful degradation of overall system capability, system reconfiguration, and mission tailoring. Much of the necessary hardware technology is already on the horizon, but the software techniques needed for distributed executive design and the systems analysis tools needed to understand the broad issues of reconfiguration and interaction of peer-coupled systems need considerable emphasis. Fortunately, these software needs appear to be at least as pertinent to commercial distributed systems as they are to military systems, so the Air Force may be able to utilize important independently sponsored work.

The use of digital systems for flight-safety control functions will push the state of the art in software and hardware reliability. Simplicity in hardware circuitry and software logic may provide the means for adequate testing and inherent reliability.

Distributed processing should improve the reliability of long-lived systems but will require greater (specification) detail at the component interfaces.

F.3 MICROPROCESSORS

F.3.1 Summary

- Embedded microprocessors will continue to proliferate, especially in sensors, displays, controls.
- Microprocessors will be increasingly employed at the subsystem level, e.g., buried in sensors and other hardware.

- Support facility requirements will increase in number and complexity as the need to test digital systems with many interfacing processors increases.
- Computer-on-a-chip technology will limit the ability to design computer monitor and control units, since many required signals will not be pin-available.
- Standardized processor requirements may evolve.
- There is no standard and suitable microprocessor language.
- The reference does not discuss VHSIC.

F.3.2 Excerpts from COMTEC-2000

A well-planned functional architecture analysis and simulation of distributed processing should be conducted and standard memory and Control Processing Unit (CPU) processing modules should be developed that can be coupled in a distributed processing system. Programmable processing modules are recommended to allow the same module to be used for different signal processing functions within a particular sensor or allow application to another type of sensor by substituting a different instruction set in Programmable Read-only Memory (PROM). (PROM would apply to the pipeline processing functions and will not satisfy the memory requirements required for comparison, correlation, and formatting functions.)

Two limiting factors associated with the continued usage and expansion of advanced sensors are seen. In software, the development of appropriate near-real time algorithms and data representations for sensor processing is likely to be both a major schedule (time to implement) and cost factor. In hardware, the technology for high-speed processing is likely to be a limiting factor in the employment of sensors, even in distributed architecture approaches.

A hardware concern is the ability to design computer monitor and control systems for microprocessors deeply embedded in systems. Though computer monitor and controls are used in current generation avionic computers, the development of single chip computers may render current technology ineffectual.

Microcomputers will tremendously increase the capabilities of future armament systems at affordable cost. The federated design approach, placing processors at the subsystem level, will result in standard processor requirements of 500 KOPS, 32K Read-only Memory (ROM), and 24K Random Access Memory (RAM) at a 16-bit word length. Commercial processors of this capacity will be available, most likely in metal oxide semiconductor technology.

F.4 NETWORKS

F.4.1 Summary

- A distributed AISF technology demonstration should be pursued.
- Network for sharing of data bases and software is needed.
- Non-secure communications bandwidths will continue to increase; costs will decrease.
- Secure communications will have low bandwidth at high cost.
- Security will progress steadily.

F.4.2 Excerpts from COMTEC-2000

The approach to developing bus protocols now being coordinated for MIL-STD-1553 should serve as a model for similar efforts on high-speed (wideband) bus protocols.

It appears that the technologies required to satisfy each of the support functions will be available. However, current systems applications are constrained by the lack of a viable networking system to enable support, operational, and development commands to share support software. Of particular concern is the need for on-line data "encyclopedias" to permit sharing of programs and data bases.

Communication link bandwidths will greatly increase in peacetime and unit costs for data transmission will decrease significantly. However, a stressed environment will experience a narrow bandwidth at high cost.

Multilevel security will progress steadily, as will man-machine interfaces. A variety of pressures toward distributed processing will be felt. Improvements in such areas as inference, understanding, and fusion will depend on Air Force requirements and the amount of research support they receive.

Communication links will achieve vastly higher bandwidths at substantially lower cost in a benign environment. However, low bandwidth at high cost will characterize communications in a hostile environment. Large amounts of intelligent processing will be needed at senders and receivers in order to reduce bandwidth needs. This is one of several pressures toward distributed processing.

Security, including multilevel computer security, will make steady progress. Successful certification of a multilevel secure operating system for limited transactions will provide for dynamic key management. Cryptographic security on a "per-call" basis, coupled with data-encrypting, will allow the development of distributed, multilevel secure systems (using noncertified hardware, software, and operating systems) with more general purpose capability. Efforts to develop Kernelized Secure Operating Systems (KSOS) may result in a certified multilevel secure operating system for a general purpose, monolithic machine, although certification for each new version will be lengthy and costly, compared with distributed processing models. While these trends (sans NASA certification) will be fostered by the commercial sector, security requirements for controlling classified EM or acoustical emanations of systems and equipment will increasingly add to the cost of terminal procurement.

Intersystem architectures will be affected by a continued significant decrease in the cost of processing power (CPU and memory). This implies that computing resources will proliferate with new environments and application areas, and that the economic incentive to centralize these resources will nearly disappear. Also, communication costs for

a given bandwidth will decrease, but not as rapidly as the cost of computation. This will continue to make it economically advantageous to perform computations at the sources of information as a means of compressing the bandwidth requirements. Customized computer architectures will become increasingly available, thus permitting a wider distribution of specific computer functions. Computing systems will be less constrained by environmental considerations, permitting their inclusion in a wider range of environments. Worldwide commercial communication networks will provide reliable, inexpensive, high bandwidth service. This will tend to promote the advancement of communications-oriented solutions to commercial requirements, and these same solutions can be applied to the military sector in peacetime.

Development of a network capability for avionics integrated support facility computers should be pursued as offering a potentially high rate of return pending demonstration of the technology.

Intersystem architectures will proliferate in the commercial sector for economic and sociological reasons. Additional pressures toward distributed processing will confront the Air Force. It is unlikely that a single intersystems strategy will prevail. More likely, individual network cultures will develop around the offerings of vendors. This commercial trend toward noninteroperability will benefit the Air Force as, inevitably, "gateways" will be incorporated in commercial networks. Their technology will be available for attacking Service and NATO interoperability problems. The predominant connection strategy today is hierarchical-intelligent terminals connected to minis arrayed around large CPU's. This should give way to "peer coupling", as an increase in remote intelligence blurs the distinction between master and slave.

F.5 STANDARD COMPUTERS

F.5.1 Summary

- Computer architecture standardization may be harmful.
- The field life of systems is much longer than it takes them to become obsolescent. Hardware support becomes a problem.

- Reliability of long-term storage of digital components is not well understood and needs to be studied.
- Pin-compatible parts may be an answer to long-term logistics support.
- Standard terminal interface specifications are needed.

F.5.2 Excerpts from COMTEC-2000

Computer architecture standardization is likely to be a troublesome area and has the potential for doing great harm by divorcing Department of Defense from commercial support software and by constraining the use of mass produced microcomputers.

The rapid rate of technological change, coupled with long development times for weapon systems, leads to instant obsolescence. The long field life of systems further compounds the problem.

Hardware technology that is driven by commercial demands will continue to advance at a rapid rate.

Situations arise in which components in useful hardware cannot be replaced because they are no longer manufactured. The useful life of an Air Force weapon system can be 20 years and technology is progressing at a very rapid pace. Investigation of the use of "pin compatible" interfaces is recommended for installed parts that may be replaced at a later date with a new part. The new part would be compatible with the interface even though the system might not be able to take full advantage of the capability of the new part. This strategy has worked with memory; the issue is whether the concept can be extended. Trends toward bus standardization and microprocessor standards should help.

The reliability aspects of long-term storage of parts are not well understood, but are believed to be related to package design and construction, wire bonding, and die attach. A study oriented toward the development of standardized high-reliability military packages should be undertaken. The die-interconnect-package relationships should receive special attention.

Future weapons will require large amounts of data from the aircraft avionics system and will in fact contain their own avionics systems. Flexible data exchange architectures are required to ensure compatibility of the avionics system and the aircraft/weapon systems. Data exchanged between weapons and aircraft can include audio, video, high-rate digital data, and MIL-STD-1553B data. Investigation of data exchange architectures is necessary to provide future weapon systems with maximum capabilities.

An urgent need is to establish interface standards for terminals to permit maximum interchangeability of terminals among systems. Interface standards among subsystems within the terminal would promote modularity. Terminals then could be configured from standard modules to provide a terminal tailored to the specific application. A standard internal bus structure for terminals appears feasible and offers good flexibility and extensibility. The Air Force should take an active role in seeing that such standards are established.

F.6 LANGUAGES (AND SOFTWARE TECHNOLOGY)

F.6.1 Summary

- Programmer productivity is too low, and growing too slowly to keep pace with computer hardware capabilities.
- The development of a standard modern HOL should be supported.
- The development of new configuration control techniques for hardware and software should be supported.
- Static test tool development should be encouraged.
- Standard languages, to be successful, must be accepted by the commercial marketplace.
- There is currently no standard and suitable micro-processor language.
- Development of requirements definition languages and problem definition languages should have a high payoff.

- MIL-STD-483 Part I Specs require too much detail.
- A software engineering support facility should be established to collect and make available current tools and methodologies, and direct new technology.
- More government planning of software support systems is needed.
- Standard software support packages should be developed.

F.6.2 Excerpts from COMTEC-2000

There has recently been strong emphasis in the Department of Defense on adopting a few standard languages and computer instruction sets, in order to reduce the logistic support required for the 150 languages and 700 types of computers currently in the Department of Defense inventory. While it is obvious that something must be done about the present chaos, it is equally clear that standardization will have to occur in a manner that will not isolate Department of Defense from the stream of technology in the commercial market. Standardization on obsolete technology and inadequate provision for admitting improved technology are the Scylla and Charybdis of these efforts.

Software design tools can increase productivity by unambiguously describing a problem and validating the result. The private sector has already initiated software design research upon which the Air Force should build. Two classes of software design tools will have a high payoff: requirements definition languages and problem definition languages.

Problem-Oriented Languages (POL's) can be used to avoid unlimited expansion of HOL in order to keep the HOL compiler at a reasonable size. The POL's should be written in the HOL and implemented in translator/library systems that output the standard HOL.

Syntactic constructs should be added to the standard HOL used to support distributed processing. These constructs should allow a system designer to direct procedures to specific processors in a distributed system and the compiler should then determine input/output schemes and data exchange requirements to relieve the programmer of this burden. In addition, these constructs should allow for timing requirements and should issue errors for timing conflicts.

Testing technology should keep pace with hardware development. This should include considerations for reduced test sequence length, adequacy of tests, procedures for automated test sequence generation, and testability as a design consideration.

Dynamic testing, the mainstay of current computer testing methodology, is not adequate to produce the degree of reliability required in many systems, particularly those involving human safety or mission accomplishment. As systems become more complex, the inadequacy will become even more apparent. The increasing use of multiprocessor systems will pose very serious problems, because varying timing conditions will make errors impossible to recreate under observable conditions. Under these circumstances, dynamic testing will have to be supplemented by testing methods which do not depend on running the system. Logical (static) analysis of designs and programs to prove certain properties, such as correct execution of firmware-based operators and freedom from dead lock, will be necessary.

An HOL for real time simulations should be developed that is portable and can also be executed on time-sharing computers.

The Air Force should encourage putting all information about a development in software on machine-readable media in machine-analyzable form. The software support system used by the Air Force should have the capability of analyzing, editing, and displaying this information in proper form.

The development of new techniques for configuration control of hardware and software should be supported. Configuration control should be based upon the storage of information in magnetic form for easy handling, analysis, correction, and display. These techniques are required in hardware, particularly in electrical circuit design, where drawings are derived automatically from various computer-aided-design data bases. The storage of this digital design information for hardware can utilize the same configuration control techniques as for software.

A significant opportunity exists for the development of efficient static test tools, particularly if the Air Force evolves to a standardized programming environment. With the increasing complexity of software products and the trend toward distributed processing, static testing may become the only practical way to exhaustively test software. Thus, R&D activities associated with the development of effective static test tools should be funded. Currently, significant government funds are invested in project-unique, High-order Language (HOL) unique, and computer-unique test tools. With a standardized programming environment, some of these funds could be redirected toward development of a much more powerful and broadly applicable set of static test tools.

A variety of test editors is desirable. Source language editors are needed which have an understanding of the simple syntax of the source language and can find the components such as identifiers in programs, can do substitution on identifiers and other tokens in the language, and can also do simple syntax checking that does not require more than simple context-free parsing.

Analyzers and transformers are an open set, of which the minimum essential components are the language processors, the configuration generators, the static analysis tools for determining paths through source language, and the set of debugging tools for source language level debugging. Air Force support for more sophisticated tools for verification of the characteristics of software, for analysis of requirements, for test data generation, and for simulation should continue as technological advances make practical systems possible.

The most promising technology to be pursued is in creating and making available software engineering development and maintenance tools, facilities, and methodologies. This seems to be mostly within the state of the art; the additional technological advances required should be given high priority. A project should be initiated to establish a software engineering support facility. Such a facility could be a focal point to collect and make available current tools and methodologies, monitor technology advances, and direct new technology. Government sponsorship is necessary.

The Air Force should aggressively support development of an integrated software support system and should sponsor the development of components which are suitable for inclusion in such a system. This system should be used both for software development and software maintenance.

Because the Air Force has more interest in the maintenance phase of software, and because the software support systems used for developing software are expected to be useful during the maintenance phase, the Air Force should try to guide the creation of adequate software support systems which are integrated and easy to use. The Air Force should sponsor development of a set of requirements for software support systems that will emphasize the integration of the components into a single usable system and will encourage the development of more helpful components in both the software development and maintenance phase.

Computer monitoring and control should be a recognized requirement for all future systems and studies should be conducted to determine the limitations of current design technology.

F.7 EMULATORS

F.7.1 Summary

- Software implementation tools, including code generators and simulator writers will enhance software acquisition and maintenance capabilities.

F.7.2 Excerpts from COMTEC -2000

Software implementation tools that will greatly enhance Air Force acquisition and maintenance of software are a code generator and simulator writing tool, problem-oriented language translators and libraries, and a distributed processing support tool. All three assume that a standard HOL is adopted.

The code generator and simulator writing tool is needed to produce uniformly high-quality code simulators for the standard modular compiler. This tool should be written in the standard HOL and should take as input a model of the target computer. Description of the model would include parameters such as register quantity and capability, word and memory size, and microcoded characteristics. The output would be an optimized code generator for a simulator of the target machine.

F.8 MISCELLANEOUS

F.8.1 Support Systems

The entire area of support effectively has been ignored in the past. The growing costs associated with support, however, are impacting on the ability to introduce new systems and technologies. Support R&D promises an extremely high rate of return on a relatively small investment. It is recommended that standards be developed and support capabilities be enhanced to preclude duplicate developments and lower the manpower-intensive nature of current support. In particular, many attempts to standardize hardware across avionics integration support facilities are constrained by the requirements of competitive procurement of automatic data processing equipment. To achieve the economics associated with standardization, this area warrants further procurement and legal attention.

The adoption of standard support software packages should be pursued as having a high rate of return with a low front-end investment.

F.8.2 Distributed AISE

Development of a network capability for avionics integrated support facility computers should be pursued as offering a potentially high rate of return pending demonstration of the technology.

F.8.3 Graphics

The Air Force can lower the cost and increase the availability of real time surface-shaded displays for flight simulators by standardizing the basic display component of these systems. Large, standardized procurements of basic display systems for flight simulation should be made to obtain maximum economics of scale. Research should be supported in the development of more effective algorithms for real time surface-shaded color display for flight simulation to hasten the reduction of cost for this important technology. The concept of electronic maps should be supported by advanced development funds.

Very large screen, high-resolution displays will only be developed through government funding. Large interactive displays also may require special developments that are not direct extensions of commercial interactive displays.

While a low-voltage, flat-screen, solid-state display is badly needed to replace the Cathode Ray Tube (CRT), no technology shows sufficient promise to justify a large-scale product development. For the present, any potential CRT replacement should only be funded as a research project to determine technical feasibility.

REFERENCES

- 1-1. Vicen, Paul M, The Logistics of Software - AFLC in the 1980s, NAECON, 1980.
- 1-2. Best, Maj, SON for Embedded Computer System Software Support, AFLC/XRX Draft Memo, 15 May 1980.
- 1-3. COMTEC-2000, Volumes I and II, HQ AFSC TR 78-03, December 1978.
- 2-1. White, W. and Holmes, M., "The Future of Commercial Satellite Telecommunications," Quest (Vol. 2, No. 1), TRW Defense and Space Systems Group, Spring 1978.
- 2-2. Wecker, S., "Computer Network Architectures," IEEE Computer, September 1979.
- 2-3. Walden, D.C. and McKenzie, A.A., "The Evolution of Host to Host Protocol Technology," IEEE Computer, September 1979.
- 2-4. Mochoff, N., "The Global Video Conference," IEEE Spectrum, September 1980.
- 4-1. Thurber and Wald, "Associative and Parallel Processors," Computing Surveys, December 1975.
- 4-2. Satyanarayanan, M., "Multiprocessing: An Annotated Bibliography," IEEE Computer, May 1980.
- 5-1. Avionics Master Plan, the Deputy for Avionics Control, 30 April 1979.
- 5-2. Sylvester, Dr. R.J., Revised White Paper on AFSC Microprocessor Policy, ASD/EN, Wright Patterson Air Force Base, 18 January 1980.
- 5-3. Hilbing, LtCol F.J., Forecast of Computer Software Technology, Computer Related Information Systems Symposium, January 1979.
- 5-4. Johnson, R.W., "Forecast of Computer Hardware Technology," Proceedings of Computer Related Information System Symposium (CRISYS), January 1979.
- 5-5. Winslow, T., Airborne Systems Software Acquisition Engineering Guidebook for Microporcessors and Firmware, TRW Defense and Space Systems Group, February 1980.

REFERENCES (Continued)

- 5-6. Buie, J.L., "VLSI, Very Large Scale Integration," Quest, (Vol. 2, No. 1), TRW Defense and Space Systems Group, Spring 1978.
- 5-7. Sumrey, L.W., "VLSI with a Vengeance," IEEE Spectrum, April 1950.
- 5-8. Giles, M. and Nash, J.N., "Applications of LSI to Digital Systems," IEEE Paper, CH 1518-026, 1979.
- 6-1. Computer Programming Languages, Air Force Regulation 300-10, December 15, 1976.
- 6-2. Interim List of DOD Approved High Order Programming Languages (HOL), Department of Defense Instruction 5000.31, November 24, 1976.
- 6-3. Steelman, Department of Defense Requirements for High Order Computer Programming Languages, June 1978.
- 6-4. Pebbleman, Department of Defense Requirements for the Programming Environment for the Common High Order Language, January 1979.
- 6-5. Sammet, J.E., Roster of Programming Languages for 1976-77, SIGPLAN Notices, 13, 11, November 1978.
- 6-6. MIL-STD-1589A Military Standard JOVIAL (J73), United States Air Force, March 1979.
- 6-7. IEEE Standard ATLAS Test Language, ANSI/IEEE Std. 416-1978, Institute of Electrical and Electronics Engineers, Inc., 1978.
- 6-8. Automatic Test Equipment Language Standardization, OSD memorandum, 14 July 1978.
- 6-9. Request for Waiver of Abbreviated Test Language for All Systems (ATLAS) Language Requirements, AFLC memorandum (LOE), 18 January 1980.
- 6-10. Wagner, Peter, "The Ada Language and Environment," Electro Proceedings, May 1980.
- 6-11. Scheer, L.S. and McClimens, M.G., DOD's Ada Compared to Present Military Standard HOL's, NAECON Proceedings, May 1980.

REFERENCES (Concluded)

- 6-12. Management of Computer Resources in Major Defense Systems, Department of Defense Instruction 5000.29, 26 April 1976.
- 7-1. Clark, N.B., The Use of Emulation for Total System Design, Proceedings of 1978 Summer Computer Simulation Conference, July 1978
- 7-2. Flink, C.W., "A Microprogrammed Environment for a Software Development System," Compcon, Fall 1975.
- 7-3. Hayes, J.P., Computer Architecture and Organization, 1978.
- 7-4. Press, B., "Diagnostic Emulation; New Software Tool for the Microprocessor Age," Quest, TRW Defense and Space Systems Group, 1978.
- 7-5. Advanced Microprogramming Development System - System 29/05 Manual, Advanced Micro Computers, 1978.
- 8-1. Instruction Set Architecture for the Military Computer Family, MIL-STD-1862, 28 May 1980.
- 8-2. Airborne Computer Instruction Set Architecture, MIL-STD-1750A, 1 March 1980.
- 8-3. Aircraft Internal Time Division Command/Response Data Bus, MIL-STD-1553B, 21 September 1978.
- 8-4. Teates, H. and Wise, B., "Creditation: A New Embedded Computer Standardization Approach," IEEE Computer, September 1980
- 9-1. "Special Issue on Fault-Tolerant Digital Systems," Proceedings of IEEE, (Vol. 66 No. 10), October 1978.
- 9-2. "Special Issue on Fault-Tolerant Computing," IEEE Computer, March 1980.
- 9-3. Christie, L.D. and Tiffany, P.C., Design Considerations for the Automated Maintenance of Computerized Avionics Systems, IEEE Paper CH1518-150, 1979.
- 10-1. ACM Computing Surveys, Journal of the Association for Computing Machinery, December 1978.

