

AD-A150 583

PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16

1/6

MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV

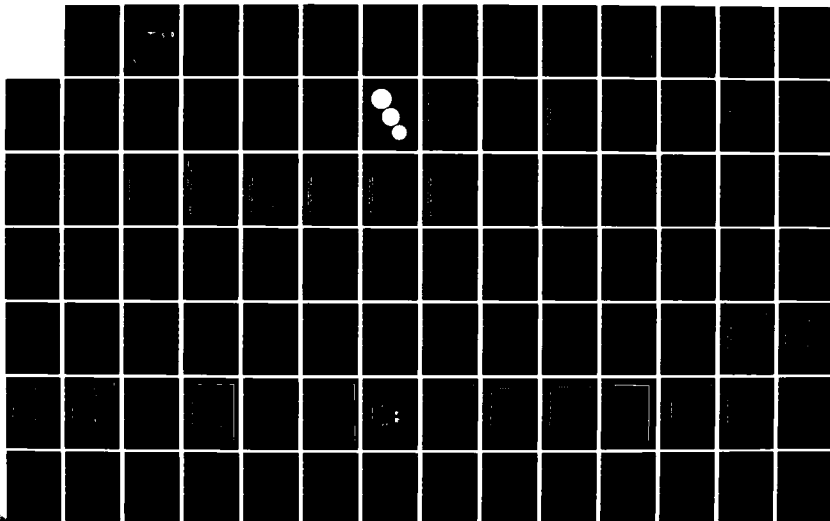
WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 06 MAY 82

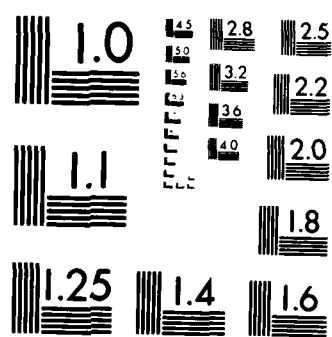
UNCLASSIFIED

ASD-TR-82-5011-VOL-1

F/G 9/2

NL

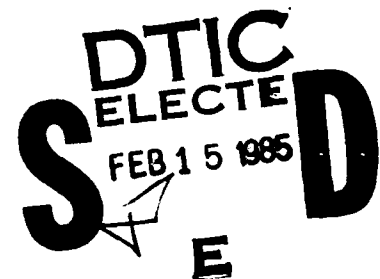




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

**THE THIRD TECHNICAL FORUM
ON THE
F-16 MIL-STD-1750A MICROPROCESSOR
AND THE
F-16 MIL-STD-1589B COMPILER**

AD-A150 583



5 - 6 MAY 1982

**AIR FORCE MUSEUM AUDITORIUM ,
AREA B
WRIGHT-PATTERSON AFB OHIO 45433**

PROCEEDINGS



VOL I OF II: PAPERS



**APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED**

DTIC FILE COPY

85 01 30 043

NOTICE

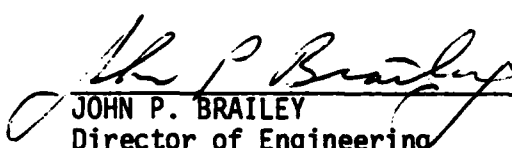
When Government drawings, specifications or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated,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

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

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


JEFFERY L. PESLER
Project Engineer
Deputy for F-16


FELIX O. GLOVER
Dep Chief, Avionics Division
Deputy for F-16


JOHN P. BRAILEY
Director of Engineering
Deputy for F-16


GEORGE L. MONAHAN, JR.
Brigadier General, USAF
System Program Director
Deputy for F-16

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

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ASD-TR-82-5011	2. GOVT ACCESSION NO. AD-A150583	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proceedings; Papers Presented at the Third Technical Forum		5. TYPE OF REPORT & PERIOD COVERED Vol I of II 5-6 May 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Editors: Mr Jeffery L. Pesler Lt Dennis Shoulders		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Hq ASD/YPEA Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Hq ASD/YP Wright-Patterson AFB, Ohio 45433		12. REPORT DATE 5-6 May 1982
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same as above.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) N/A		
18. SUPPLEMENTARY NOTES N/A		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) MIL-STD-1750A; MIL-STD-1589B; Microprocessor; Computer Instruction Set Architecture; Compilers; Support Software; Mnemonics; Digital Avionics; Standardization.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This is a collection of unclassified papers to be distributed initially at the Third Technical Forum on the F-16 MIL-STD-1750A microprocessor and support software. The purpose of this forum is to keep interested organizations informed of the progress of the F-16 embedded computer standardization efforts.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

The Deputy for F-16 is sponsoring the development of a MIL-STD-1750A microprocessor and MIL-STD-1589B support software in conjunction with the Multinational Staged Improvement Program (MSIP). For the next two days we will provide a comprehensive technical summary of the F-16 embedded computer standardization efforts and a forum of papers from other "users". Significant interest in the F-16 MIL-STD-1750A/MIL-STD-1589B programs have prompted this technical forum.

The purpose of the Third Technical Forum is to present the technical details of the F-16 embedded computer standardization efforts, to exchange ideas on lessons learned from other system applications and present other hardware/software developments in support of these standards.

This is the PAPERS Volume, Volume I of the Third Technical Forum Proceedings. Volume II contains the preliminary specifications handed out during the forum.

Many thanks to Mr. Jeffery L. Pesler for his key role and leadership in the adoption of the standards within the F-16 MSIP as well as organizing and directing the Third Technical Forum. Also to the ASD/ENA and ASD/AX staff who have assisted in organizing the technical program and proceedings, Mr. Jerry L. Duchene and Mr. Ronald S. Vokits. Special thanks to Mr. Randal K. Moore of General Dynamics for his leadership and support.

Thanks also to the moderators and all the speakers, who responded with outstanding presentations and papers in a timely manner, despite such short notice. And, finally, to the secretaries Mrs. Marie Jankovich, Mrs. Sharlene Thompson, Mrs. Brenda Harris and Mrs. Connie Castin for their expert administrative help in handling the conference correspondence and registration.


GEORGE L. MONAHAN, JR.
Brigadier General, USAF
System Program Director
Deputy for F-16

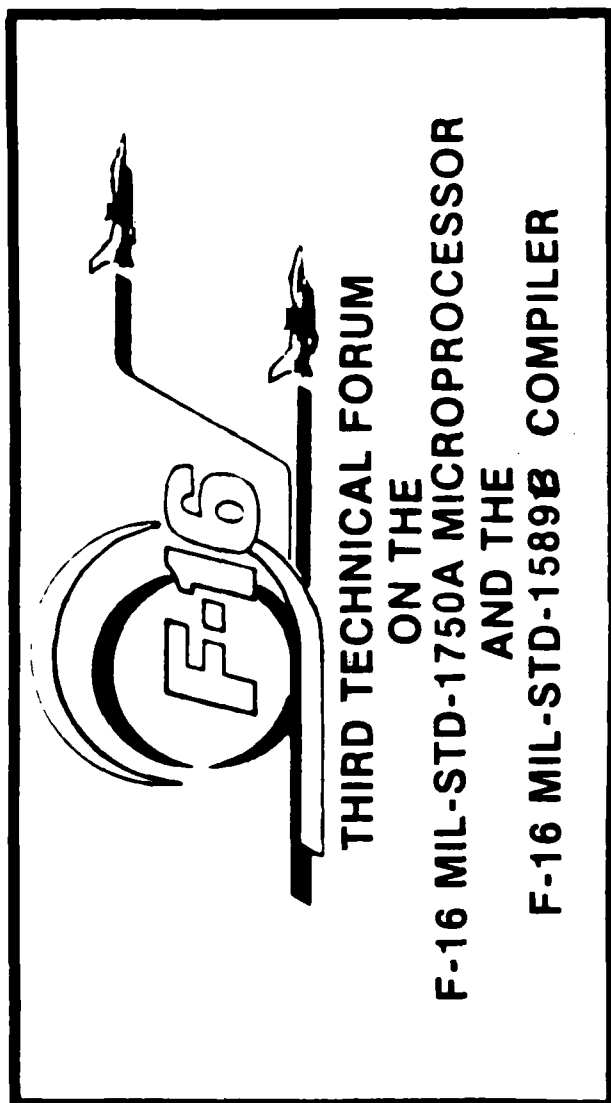
Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



TABLE OF CONTENTS

5 May		Page
Welcome	Mr Jeffery L. Pesler ASD/YPEA	1
Keynote Address	Mr D Burt Newlin OUSD/DMSSO	7
Introduction	Col David J Teal ASD/YP	57
MIL-STD-1750A Microprocessor .		
Management Overview	Mr Randal K Moore GD/FW	81
Program Overview	Mr Dan Wilnai Fairchild	87
Functional and Technical Description	Mr Shai Mor Fairchild	97
Applications and Hardware	Mr Henry Lynn GD/FW	171
Test and Evaluation Equipment	Mr David M Scott GD/FW	238
Interim Processor Transition	Mr Henry Lynn GD/FW	247
F-16 MIL-STD-1589B Compiler	Dr Judy Edwards GD/FW	249
Fairchild Marketing Approach	Mr Ron Byrne Fairchild	273
6 May		
Opening Remarks	Col George W Botbyl ASD/AX	293
Life Cycle Cost Analysis	Ms Janice L Lyons GD/FW	303
MIL-STD-1750A; Managing the Standard	Mr Ronald S Vokits ASD/AXT	323
Transition Plan to Upgrade the Digital Integrating Subsystem to MIL-STD-1750A for MRASM	Mr J B Reed GD/C	339

		Page
Wasp Weapon System/Digital Processing	Lt Gene J Dylewski HQAD/SD-8	355
Delco Electronics MIL-STD-1750A Architecture Computer Family	Dr Clive D Leedham Delco Electronics	379
A New Silicon-On-Sapphire MIL-STD-1750A Microprocessor	Mr Henry Silcock Mikros Systems	411
Texas Instruements VHSIC MIL-STD-1750A Microprocessor	Mr Dennis R Best Texas Instruments	437
Testability Support System for 1750A: In-Circuit Fault Simulator	Mr Constantin Timoc NASA/JPL	449
Embedded Computer Standardization Program Office (ECSP0)	Maj Dan Burton ASD/AXS	465
An Implementation of a 1750 Computer and its Working Environment	Mr Israel E Koren AEL Israel Ltd	477
Support of Software Tools	Mr Joel Fleiss Proprietary Software Systems	485
Use of the 1750A Support Software on an IBM 3033 Under MV/CMS	Mr John Riordan Mikros Systems	501
The AFWAL MIL-STD-1589B Jovial Compiler	Mr Mike Burlakoff AFWAL/AAAF	528



CHAIRMAN: MR. JEFFERY L. PESLER ASD/YPEA
VICE CHAIRMAN: MR. RANDAL K. MOORE GENERAL DYNAMICS CORP
PAPERS CHAIRMAN: MR. JERRY L. DUCHENE ASD/ENA
PAPERS COMMITTEE: MR. RONALD S. VOKITS ASD/AXT

5-6 MAY 1982

THE THIRD TECHNICAL FORUM
ON THE
F-16 MIL-STD-1750A MICROPROCESSOR
AND THE
F-16 MIL-STD-1589B COMPILER

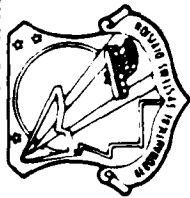
1. Welcome to Dayton, Ohio, the home of the F-16 System Program Office (SPO) and the Air Force Museum. My name is Jeff Pesler and I will be your host for the next two days. This is the third in a series of technical forums sponsored by the F-16 SPO. The purpose of these forums is to keep you - the user community - informed about new developments in the F-16 embedded computer standardization program. The first Technical Forum was held in May of 1981. It was for "government only" and there were only thirty attendees. The second, in October, grew to seventy and then the October presentation was taken public at the 4th Digital Avionic Systems Conference in November. Ninety attendees came to the tutorial session and over two hundred were at the forum that followed. Today over two hundred fifty people are expected at the Third Technical Forum. As you can see, there is a large interest in our program and we have grown considerably.

2. I hope you will enjoy and benefit from the presentations over the next two days. Every effort was made to assemble material on the F-16 Standardization Program that would be useful to others. The General Dynamics and Fairchild briefings will contain in depth technical and management details on the F-16 microprocessor and compiler development programs. On the second day, papers on other on-going standardization efforts will be presented. Thank you again for your participation and interest in the F-16 program.



JEFFERY L. PESLER
Project Engineer
Deputy for F-16

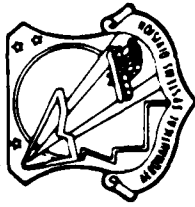
F-16 PROGRAM



AGENDA

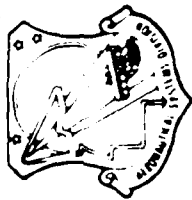
5 MAY 1982	
0900 WELCOME	MR JEFFREY L. PESLER ASD/YPEA
0910 KEY NOTE ADDRESS	MR BURT NEWLIN OUSD/DMSSO
0930 INTRODUCTION	COL DAVID J. TEAL ASD/YP
0945 MANAGEMENT OVERVIEW BACKGROUND F-16 APPLICATIONS PHASED APPROACH MANAGEMENT PLAN	MR RANDAL K. MOORE GENERAL DYNAMICS
1005 PROGRAM OVERVIEW PROGRAM GOALS PROGRAM APPROACH SCHEDULE	MR DAN WILNAI FAIRCHILD CAMERA & INSTRUMENT CORP
1015 BREAK	
1030 FUNCTIONAL & TECHNICAL DESCRIPTION OVERVIEW FUNCTIONAL DESCRIPTION INTERFACES PERFORMANCE	MR SHAI MOR FAIRCHILD CAMERA & INSTRUMENT CORP

F-16 PROGRAM



AGENDA (CONTD)

5 MAY 1982	
1200 LUNCH	
1300 FUNCTIONAL & TECHNICAL DESCRIPTION (CONTD)	
1330 APPLICATIONS & HARDWARE	MR HENRY LYNN
MEMORY SYSTEM DESIGN	GENERAL DYNAMICS
I/O SYSTEM DESIGN	
1400 ENGINEERING TEST & EVALUATION EQUIPMENT	MR DAVID SCOTT
"OFF THE SHELF" EQUIPMENT	GENERAL DYNAMICS
MODIFICATIONS FOR TEST/EVALUATION	
APPLICABILITY	
1415 INTERIM PROCESSOR TRANSITION	MR HENRY LYNN
DESIGN GUIDELINES	GENERAL DYNAMICS
INSTRUCTION SET COMPARISON	
1445 BREAK	
1500 F-16 COMPILER DEVELOPMENT PROGRAM	DR JUDY EDWARDS
BACKGROUND	GENERAL DYNAMICS
TECHNICAL DESCRIPTION	
STATUS	
1600 FAIRCHILD MARKETING APPROACH	MR RON BYRNE
	FAIRCHILD CAMERA & INSTRUMENT
	CORP



AGENDA (CONTD)

D29

21E2

6 MAY 1982

0800 OPENING REMARKS

0815 LIFE CYCLE COST ANALYSIS

0845 MIL-STD-1750A, MANAGING THE STANDARD

0915 DIS TRANSITION TO MIL-STD-1750A

0945 APPLICATION OF MIL-STD-1750A TO WASP MILLIMETER WAVE
MINIMISSILE

1015 BREAK

1030 DELCO MIL-STD-1750A ARCHITECTURE COMPUTER FAMILY

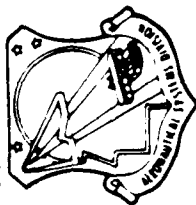
1100 A NEW SILICON-ON-SAPPHIRE MIL-STD-1750A
MICROPROCESSOR

1130 VHSIC MIL-STD-1750A DATA PROCESSOR

1200 LUNCH

MS JAN LYONS
GENERAL DYNAMICSMR RON VOKITS
ASD/AXTMR JIM REED
GENERAL DYNAMICSLT GENE DYLEWSKI
HQ AD/SD-8DR CLIVE LEEDNAM
DELCOMR HENRY SILCOCK
MIKROSMR DENNIS BEST
TEXAS INSTRUMENTS

F-16 PROGRAM



AGENDA (CONTD)

6 MAY 1982

1300 TESTABILITY SUPPORT SYSTEMS FOR 1750A,
IN CIRCUIT FAULT SIMULATION

1330 MIL-STD-1589B, THE ECSP0 VIEW

1400 MAINTENANCE AND DEVELOPMENT TOOLS

1430 BREAK

1445 SUPPORT OF SOFTWARE TOOLS

1515 USE OF 1750A SUPPORT SOFTWARE ON
IBM 3033

1545 AFWAL MIL-STD-1589B COMPILER

1615 ADJOURN

MR CONSTANTIN TIMOC
NASA/JPL

MAJ DAN BURTON
ASD/AXS

MR ISRAEL KOREN
A.E.L. ISRAEL LTD

MR JOEL FLEISS
PROPRIETARY SOFTWARE SYSTEMS

MR JOHN RIORDAN
MIKROS

MR MIKE BURLAKOFF
AFWAL/AAAF-2

STANDARDIZATION INITIATIVES TO REDUCE SOFTWARE COSTS

By D. Burton Newlin, Jr.

Office of the Under Secretary of Defense (Research and Engineering)
Defense Materiel Specifications and Standards Office

The Department of Defense has become increasingly dependent on computer technology to accomplish its mission. Computer costs are consuming an ever increasing percentage of the DoD's budget. Within the DoD the majority of the computers are embedded within weapon systems to perform command and control, communications, navigation, surveillance, target detection intelligence and other functions critical to their strategic and tactical missions. With the accelerating advances in electronic technology hardware has become relatively inexpensive, and the development of microprocessors and microcomputers have permitted substantial increases in both the amount of main storage which could be supported and the associated software to be written. As a result the majority of DoD's computer resources now involve the computer software associated with weapon systems.

Management attention has been directed towards standardization as one approach to control and reduce these software costs. The two major efforts towards standardization involve the development of a common high order programming language and the development of a limited number of instruction set architectures for computer software applications embedded within defense systems.

Increased Management Attention

Legislative changes to the computer acquisition process have increased the emphasis management must place on DoD's Mission Critical Computer Resources. Recent OSD internal organizational changes have further emphasized the importance of software management and need to strengthen the decision-making management organization to oversee defense-wide standardization efforts. Dr. Edith Martin has been recently appointed Deputy Under Secretary of Defense for Research and Engineering (Research and Advanced Technology). Dr. Martin has a computer science background and will be overseeing the management of the VHSIC and Ada programs.

Mr. Mark Grove is the new Assistant Deputy Under Secretary (Research and Advanced Technology). In this new position he continues to be responsible for the research, development and acquisition policies for computer resources. Mr. Grove is also responsible for embedded computer or mission critical computer standardization activities currently centered on controlling and reducing the proliferation of high order languages and instruction set architectures.

Legislative Changes to the Computer Acquisition Process

When the Brooks Act, Public Law 89-306, was passed in October 1965, computer technology was at a stage of development where hardware considerations were the major concern in the acquisition process. The intent of this law was to provide for the economic and efficient purchase, lease, maintenance, operation and utilization of automatic data processing equipment by federal departments and agencies. As technology advanced, hardware cost declined, while software cost escalated, and this has required that increased management attention be devoted to the software acquisition process. Within the Department of Defense, the acquisition of mission critical computer resources is differentiated from general purpose computers used for routine business applications and is therefore managed under separate policies.

The distinction between general purpose ADP equipment and DoD's embedded computer systems was recognized in Section 908 of the Department of Defense Authorization Act of 1982, (The Warner Amendment) Public Law 97-86. This law, codified as Title 10 U.S.C. Section 2315, provides the DoD with an incentive to seek improved and streamlined methods and practices for the acquisition of computer resources used in mission critical applications. Under this law, we have coined the term "Mission Critical Computer Resources (MCCR)" to refer to any automatic data processing equipment or services if the function, operation, or use involves intelligence, cryptologic activities related to national security, command and control, equipment which is part of a weapon system or which is critical to the direct fulfillment of military or intelligence missions.

The Establishment of a Standardization Area for Embedded Computer Resources

Under the Defense Standardization Program, a new standardization area has been established to help reduce software cost and improve the management of Embedded Computer Resources or, now more accurately, "Mission Critical Computer Resources" within DoD. An Embedded Computer Resources standardization program plan has been developed that identifies ongoing government and industry standards activities in this technology area as well as related areas where standards will be needed in the future as the technology evolves. This management plan outlines objectives and establishes priorities, milestones and resource allocations consistent with the identified standardization documents that need to be developed, revised or cancelled. The standardization initiatives addressed in the plan are concentrated primarily on the standardization of software high order languages, architectures, software documentation, quality control, configuration management and their related data item descriptions. These standards are needed to encourage competition and reduce the downstream logistics and maintenance costs during the out years when there will be a shortage of qualified software personnel. Since software will continue to be labor intensive, we need to do everything possible to encourage standardization and encourage open competition in these cost driver areas.

DoD's Common Language Effort

A first step in reducing the number of programming languages for weapon applications was to adopt an interim list of approved languages. This list

is being revised to contain 8 specific languages which shall be used on all major defense systems unless it is demonstrated that none of the approved languages are technically practical or cost effective over the system life cycle. The second step towards the standardization of high order languages is the implementation and adoption of the Ada* programming language specified in MIL-STD-1815. Ada is now being coordinated as both a national and international standard through the ANSI and ISO committees.

Standardization of Computer Architectures

The Department of Defense has been convinced, because of advances in hardware technology, that the standardization should be technology transparent and that computer standardization requirements should be expressed through a limited number of instruction set architectures (ISA's). This policy has been expressed in a draft DoD Instruction 5000.5X. The primary objective of Instruction 5000.5X is curtailment of high costs resulting from hardware and software proliferation, and in particular, logistics support, personnel and training costs in the field. It is also believed that the effects of limiting the number of ISA's will result in improved software development; it has been shown to yield improved competitiveness of hardware procurements. MIL-STD-1750 is a shining example.

While there is general agreement that standardization has merits, there have been concerns expressed by the Congress and industry regarding the DoD's approach and the level at which standards should be applied such as the computer interface level. The standardization of ISA's has become a highly controversial issue, since it may well affect competition for hardware procurements.

Opponents of the DoD strategy claim that some DoD standards programs are so limiting that that may decrease competition and in some instances act as a deterrent to innovation. We do not agree with these claims. Again, MIL-STD-1750 has demonstrated that truth lies to the contrary. This controversy resulted in a special Defense Science Board Task Force Study to review the Instruction Set Architecture Standards issue.

Defense Science Board on Embedded Computer Acquisition and Management

A special Defense Science Board Task Force on Embedded Computer Resources Acquisition and Management was chartered in August 1981 to review the ISA and other embedded computer standardization issues. The task force met between September 1981 and January 1982 to advise the Secretary of Defense and Under Secretary of Defense for Research and Engineering on overall research, engineering, acquisition and management issues and to provide long-range guidance in these areas. The task force provided an evaluation of the current and proposed policies of the Department relative to management, standardization, current and planned research and development programs, and recommended improvements which appear possible and practical. They also reviewed the effect of recent changes to Public Law and the opportunities for improvement in the internal management process which these changes offer.

* Ada is a trademark of the U.S. Department of Defense.

The Defense Science Board Task Force final report lists several major findings and specific recommendations to the Under Secretary of the Defense for Research and Engineering. Although this report has not been officially released at this writing, its major findings were that continued development and application of computer standards are needed to gain and maintain control over DoD's accelerating computer acquisition, logistics and maintenance costs. They recommended that the DoD adopt a limited number of Instruction Set Architecture Standards to control the life cycle costs associated with the long life of major systems. The Task Force recommended the policy on High Order Languages be updated to add Ada to the approved list and to remove the outdated languages. Finally, it was recommended that the policy on management of computer resources be revised to emphasize the "software first" approach to system design and development.

General Accounting Office Letter Report Entitled "DoD Instruction 5000.5X Standard Instruction Set Architecture For Embedded Computers"

At the request of Congressman Jack Brooks, the General Accounting Office conducted an investigation of the Defense Department's plans to implement DoD Instruction 5000.5X. The GAO report (MASAD-82-16) dated January 27, 1982 stated that if this policy were implemented within the Department, it would (1) inhibit DoD's use of technical innovations by private companies, (2) lock DoD into obsolete technology, and (3) severely restrict competition to those companies willing to implement obsolete technology within DoD. The report recommended that the Secretary of Defense not implement DoD Instruction 5000.5X. Our official response rebutted the GAO assertions in some detail. In general, the Secretary disagreed with GAO's findings and recommendations.

An Assessment of DoD's Software Standardization Initiatives

The major objectives of standardization within the DoD are to improve operational readiness, enhance interchangeability, reliability and maintainability of military equipments and reduce acquisition, logistics and life cycle costs. Through the standardization of high order languages, instruction set architectures and improved computer software documentation, DoD's software costs should remain under manageable control. Although it is recognized that DoD's software costs will continue to increase in the near term, it is anticipated that through our standardization efforts this rate of increase will be controlled and even reduced.

Standardization based on a selected instruction set architecture is a concept used by computer manufacturers to develop families of compatible computers. In an article by Bhandarkar in IEEE Computer Magazine it states:

"The goals of architecture management . . . are twofold; to ensure the consistent implementation of . . . architecture and to provide a rational process for the evolution of that architecture. The emphasis is clearly on maintaining architectural stability by eliminating gratuitous differences. At the same time . . . future enhancements will be incorporated into the architectures in a carefully controlled manner." [1]

- [1] IEEE Computer Magazine, February 1982 "Architecture Management for insuring Software Compatibility in the VAX Family of Computers," Dileep Bhandarkar.

If this is what industry does, then why is it bad for the DoD?

The standardization of instruction set architectures continues to be a politically sensitive issue. DoD's position is that it continues to support a limited number of instruction set architectures such as MIL-STD-1750, to help reduce and control the software costs.

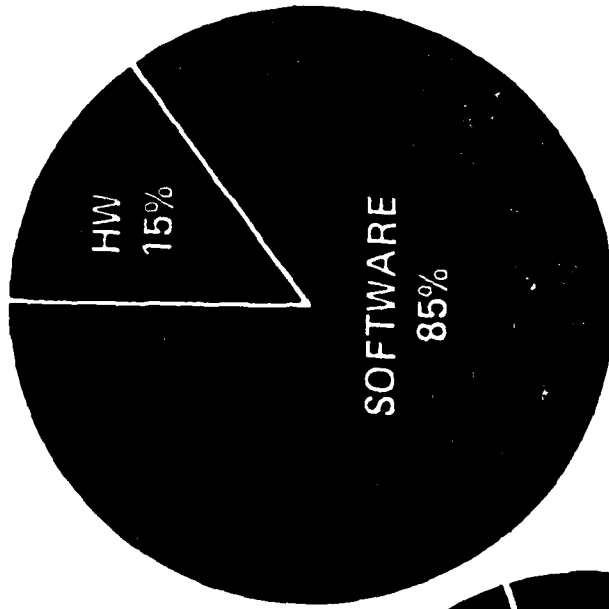
**EMBEDDED COMPUTER RESOURCES
ACQUISITION AND STANDARDIZATION POLICY**

D. BURTON NEWLIN

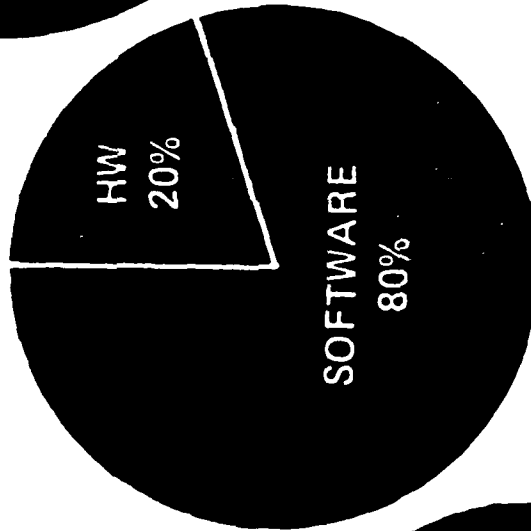
**OFFICE OF THE DEPUTY UNDER SECRETARY OF DEFENSE
(ACQUISITION POLICY)**

**DEFENSE MATERIEL SPECIFICATIONS AND
STANDARDS OFFICE**

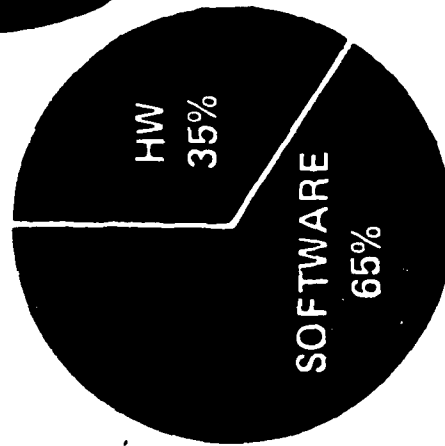
EMBEDDED COMPUTERS HARDWARE vs SOFTWARE



1990 \$37,990M



1985 \$13,920M



1980 \$4,100M

ACQUISITION POLICY DIRECTIVES

- **DODD 5000.29, MANAGEMENT OF COMPUTER RESOURCES IN MAJOR DEFENSE SYSTEMS**
- **DODI 5000.31, INTERIM LIST OF DOD APPROVED HIGH ORDER PROGRAMMING LANGUAGE (HOL)**
- **DODI 5000.5X, INSTRUCTION SET ARCHITECTURE (ISA) STANDARDIZATION POLICY FOR EMBEDDED COMPUTERS**
- **DODD 4120.3, DEFENSE STANDARDIZATION AND SPECIFICATION PROGRAM**

STANDARDIZATION PUBLIC LAW

**TITLE 10 UNITED STATES CODE CHAPTER 145,
CATALOGING AND STANDARDIZATION,
SEC. 2451-2455, CHARGES SECRETARY OF DEFENSE
WITH DEVELOPING AND MAINTAINING A
STANDARDIZATION PROGRAM TO:**

- **DEVELOP AND USE SINGLE SPECIFICATIONS**
- **ELIMINATE OVERLAPPING AND DUPLICATE SPECIFICATIONS**
- **REDUCE NUMBER OF ITEMS**
- **MAKE FINAL DECISIONS RE CATALOGING AND STANDARDIZATION**
- **REPORT TO CONGRESS ANNUALLY**

PUBLIC LAWS DIRECTING COMPUTER ACQUISITION POLICY

- **THE CATALOGING AND STANDARDIZATION ACT
10 USC 2451-56 (1976)**
- **THE BROOK'S BILL P.L. 89-306**
- **THE PAPERWORK REDUCTION ACT OF 1980
P.L. 96-511**
- **THE WARNER AMENDMENT TO THE DEPARTMENT
OF DEFENSE AUTHORIZATION ACT OF 1982
P.L. 97-86**

COMPARISON BETWEEN GENERAL PURPOSE AUTOMATIC DATA PROCESSING EQUIPMENT AND COMPUTERS USED IN DEFENSE CRITICAL SYSTEMS

	ADPE	MCCR/ECR
• MISSION:	GENERAL PURPOSE STATIC FEW IN NUMBER	DEFENSE CRITICAL SYSTEM DYNAMIC NUMBERS VARY WIDELY
• MANAGEMENT:	OASD(C) OVERSIGHT HIERARCHY WELL ESTABLISHED	OUSD (RE) OVERSIGHT BY PROGRAM CURRENTLY FUTURE TBD
• PROCUREMENT:	BROOKS ACT PUBLIC LAW 89-306	DEFENSE AUTHORIZATION ACT 1982 PUBLIC LAW 97-86
• LOGISTICS:	CENTRALIZED	VARIABLE BY SERVICE AND APPLICATION
• LONGEVITY:	FOREVER	MODERATE TO LONG
• TECHNOLOGY CHANGE:	MODERATE	VERY HIGH
• INTEROPERABILITY:	LOW	MODERATE TO HIGH
• COMPETITION:	HIGH	MAXIMUM PRACTICAL

*ADPE - AUTOMATIC DATA PROCESSING EQUIPMENT

*ECR - EMBEDDED COMPUTER RESOURCES

*MCCR - MISSION-CRITICAL COMPUTER RESOURCES

EMBEDDED COMPUTER RESOURCES STANDARDS (ECRS)*

This area encompasses standards used in mission critical defense computer systems, related software (development environments, support software, application programs, firmware, and data bases) and associated documentation.

* As defined in Standardization Directory, SD-1

HIGH ORDER PROGRAMMING LANGUAGE (HOL)

A. APPROVED HOLs

- ADA; MIL-STD-1815
- CMS-2Y
- CMS-2M
- JOVIAL J73; MIL-STD-1589
- FORTRAN; MIL-STD-1753; FIPS 69

B. APPROVED SPECIAL PURPOSE LANGUAGES

- ATLAS; ANSI/IEEE 416-1978
- SPL/I
- COBOL; ANSI X3.23-1974

INSTRUCTION SET ARCHITECTURE (ISA)

<u>ISA TITLE</u>	<u>DOCUMENT NO.</u>
• MIL-STD-1862 (NEBULA)	MIL-STD-1862
• NAVY STANDARD 16-BIT ISA	ELEX-P351
• NAVY STANDARD 32-BIT ISA	PD-PMS 408-1
• NAVY STANDARD SIGNAL PROCESSOR ISA	MIL-A-85232
• AIR FORCE STANDARD 16-BIT ISA	MIL-STD-1750

DEFENCE SCIENCE BOARD

Track Force

ON

EMBEDDED COMPUTING

RESOURCES (ECR)

ACQUISITION AND MAINTENANCE

Field Briefing

10/10/70, 10/10/70

10/10/70, 10/10/70



CHARTER

"...review, evaluate and make recommendations concerning the acquisition, management and utilization of digital computers and associated technology to support the military mission of the Department of Defense."

- Are management policies appropriate?
- Are key programs properly constituted?
- Is management and oversight adequate?
- What is the effect of legislative environment?



MAJOR ISSUES

- 1. Should DoDI 5000.5x be issued?**
- 2. Are the proposed implementations of 5000.5x reasonable?**
- 3. Should the AdaTM Program be modified?**
- 4. Should Ada and the Military Computer Family (MCF) be used for the WWMCCS Information System (WIS)?**
- 5. Should the general acquisition process be modified?**
- 6. Are changes desirable to management of DoD's Computer acquisition process?**

Ada is a registered Trademark of the U.S. Department of Defense.

1428.2



GAO LETTER REPORT B-199008

JANUARY 27, 1982

1. Ada will handle software problems. . .
 - But not real-time, e.g., operating systems
2. Technology will solve logistics problems. . .
 - But never-fail computers not here yet
3. Militarized commercial hardware provides competition. . .
 - But costs time, money, logistics chaos
4. DoD can't keep up with technology. . .
 - Objective is to ensure adequate performance
5. Can't utilize Ada efficiently. . .
 - Not true, but HLL machines may come



ON REVISION OF DoD INSTRUCTION 5000.31

Findings:

- DoDI 5000.31 has had positive effect, however its implementation has been spotty.
- Ada Program is proceeding well.

Recommendations:

- Issue updated Instruction
- Assure adequate continuing support to AJPO
- OSD review Air Force JOVIAL Program
 - Strengthen centralized support
- Absent contrary evidence based on detailed studies,
 - Army increase Ada portion of MCF Program
 - Navy increase funding of Ada effort



IMPLEMENTATION OF DoD INSTRUCTION 5000.31

- Approved list of High Order Languages (HOLs)
- The AdaTM Program
 - Purpose

Provide a single High Order Language for writing software for DoD embedded computer applications.
 - Approach
 - Ada Joint Program Office (AJPO)
 - Provide common products
 - Coordinate individual efforts
 - National and international standards
 - Ada Compiler Validation Facility
 - Service-unique efforts
 - Commercial activities

Ada is a registered Trademark of the U.S. Department of Defense.



ON PROCEEDING WITH DoD INSTRUCTION 5000.5x

Findings:

- Arguments in favor of ISA standardization compelling.
- Effects on competition are slight.
- Follow-on actions are necessary.

Recommendation:

- Issue the proposed Instruction
 - Review for consistency with acquisition regulations
 - Clarify scope (e.g., "hardware-intensive" exemption)
- Actively manage implementation
 - Control waiver process
 - Control changes to approved ISA list



PROPOSED POLICY, DoD INSTRUCTION 5000.5x

- DoD adopt small number of ISAs for broad use
- Each ISA assigned to a MILDEP for management
- Published waiver process
 - Criteria established in advance of action
 - Technically not practical or not economical over life-cycle
- Approved list continuously monitored
 - Assure policy has positive effect
 - Add or delete based upon experience and analysis



COMPTROLLER GENERAL OF THE UNITED STATES
WASHINGTON D.C. 20548

B-199008

JANUARY 27, 1982

The Honorable Jack Brooks
Chairman, Committee on Government
Operations
House of Representatives

Dear Mr. Chairman:

Subject: DOD Instruction 5000.5X, Standard Instruction Set
Architectures For Embedded Computers (MASAD-82-16)

On July 30, 1981, you requested that we review the Department of Defense (DOD) plans to implement proposed DOD Instruction 5000.5X. Your request raised a number of questions about the potential impact of implementing the proposed instruction. We have responded to your five specific questions in enclosure I.

As we understand it, the primary objective of Instruction 5000.5X is curtailment of high costs resulting from hardware and software proliferation, and in particular, logistics support costs in the field. To accomplish this, DOD has chosen to limit the number of architectures that could be used for the design and development of computer hardware and software for the tactical environment. Moreover, DOD would require ownership of standard architectures for military-embedded computers. Although this proposed instruction has merit when considered in context of the hardware/software environment that existed during the mid-1970s, our evaluation raises some serious issues that challenge its validity in the time frame of the 1980s. Some of the more salient points for consideration are:

--Dramatic advances have been made in software technology. DOD has recognized that a lack of a standard programming language is a major contributor to the high cost of developing and maintaining software for military applications. DOD is to be commended for its initiative to fill that void by developing a common high-order programming language called Ada. Ada very specifically aims to readily adapt a very wide variety of DOD applications to most present (and future) computer architectures. Ada can potentially encompass the particularly useful aspects of future architectural advances and make their gains available to users, without their having to learn and worry about how the gains were

(954029)

realized. In other words, aggressive pursuit of a standard high-order language, such as Ada, could alleviate the software proliferation problem and at the same time permit the Government to fully capitalize on architectural advances.

- Likewise there have been many advances in computer technology. These advances are the result of demands made by the civilian sector for more reliable and rugged computers. And indeed, the civilian sector is starting to impose much stiffer reliability requirements on integrated circuits. These advances will be realized probably at little or no cost penalty because all integrated circuits will be made to the same high standards. There are computer companies already marketing highly reliable computers through the use of innovative architectures. These modern computers have substantially fewer parts and in many cases are a computer on a single board thereby reducing the need for extensive logistics support.
- Improved competition using militarized versions of commercial computers will open up competition to many firms that would not bid on specifications with DOD-owned architectures. The resulting unit prices will be less because DOD will not pay for duplicating hardware development and control and utility software development as it proposed to do under Instruction 5000.5X. Lower hardware unit costs and high hardware quality are in fact available in the commercial market because of the technology and broader market base.
- DOD ownership of architectures would seriously inhibit competition by a significant portion of the computer industry, and therefore DOD would not have the flexibility to capitalize on advances in computer architectural technology in a timely fashion. The ultimate impact would result in DOD very likely running the risk of getting locked into obsolete architectures.
- DOD would not be able to efficiently utilize the new DOD programming language Ada and will not be able to fully capitalize on the anticipated software cost savings Ada was designed to yield.
- The three services have initiated efforts commensurate with Instruction 5000.5X; for example, the Army's Military Computer Family, the Navy's AN/UYK-43 and 44, and the Air Force's 1750 programs. In a previous report entitled "The Department of Defense's Standardization Program for Military Computers--a More Unified Effort Is Needed" (LCD-80-69, June 18, 1980), we were critical of both the Army and Navy efforts. We made the following statements in that earlier report and believe they are even more valid today:

"* * * some computer manufacturers are already designing computers with modern architectures that will have Ada compilers available. These or other manufacturers will probably offer follow-on computers that will directly carry out Ada instructions and substantially improve performance reliability. Because these changes provide better support options, such as building more redundancy into systems, they should compel the Department to further evaluate the level of standardization to be achieved before allowing the Army and Navy to commit themselves * * * for the long term. (Emphasis added.)"

"We view the need for architecture standardization as a function, in part, of the availability of Ada as the standard computer programming language. Because Ada is being developed to be a machine-transportable language with a relatively low life cycle maintenance cost, the need for standard architectures may be diminished when it is available * * *." (Emphasis added.)

CONCLUSIONS

We believe that DOD can accomplish its objectives more effectively through exploitation of advances made with high-order language standardization and related hardware technology. Further, we believe implementation of Instruction 5000.5X would preclude DOD's ability to make use of current and anticipated advances in software and related hardware technology.

RECOMMENDATIONS

We recommend that the Secretary of Defense not implement Instruction 5000.5X.

We also recommend that the Secretary of Defense direct the services to reevaluate their ongoing efforts and demonstrate why they are more cost effective than standardizing on a high-order language such as Ada and relying on the computer industry to provide the stimulus for computer architectural innovations.

SCOPE AND METHODOLOGY

During our review, we contacted officials representing 16 computer manufacturers, 3 system contractors who incorporate embedded computers in the systems they develop, and 4 industry associations representing manufacturers of computers and electronic equipment. We reviewed position statements and correspondence regarding Instruction 5000.5X. We also contacted program officials and reviewed program documentation regarding Ada, the Army's Military Computer Family, the Air Force's 1750, and the Navy's AN/UYK-43 and 44 efforts.

Our review was performed in accordance with our standards for audits of governmental organizations, programs, activities, and functions.

As arranged with your office, we did not obtain official agency comments on this report and we plan no further distribution of this report until 30 days from the date of the report, unless you publicly announce its contents earlier. Then, we will send copies to interested parties and make copies available to others upon request.

Sincerely yours,

A handwritten signature in cursive script, reading "Milton J. Fowler".

Acting Comptroller General
of the United States

Enclosure

RESPONSES TO SPECIFIC QUESTIONSREGARDING INSTRUCTION 5000.5XWHAT EFFECT WOULD INSTRUCTION 5000.5X
HAVE ON THE USE OF COMPETITION IN THE
DEPARTMENT OF DEFENSE?

Instruction 5000.5X would effectively preclude any commercial architecture from the Department of Defense (DOD)-approved list because DOD must have full and clear data rights to all architectures listed. Currently, only DOD developed and funded architectures have been approved. The Army's unsuccessful attempt to use a commercial architecture in its Military Computer Family program is an example of how this policy will inhibit commercial architectures from competing for embedded computer systems.

The Army's unsuccessful attempt was caused by (1) the reluctance of the commercial firm to accept Army assurance that its proprietary architecture would not be remarketed commercially and (2) the lack of industry interest in providing hardware based on a competitor's design. It is fortunate that this attempt was unsuccessful because the commercial firm involved is now marketing a new architecture due to the prior one's limitations.

A majority of the industry officials interviewed assured us that they would not compete on DOD-embedded computer procurements if they had to use DOD-approved architectures. These officials were concerned that their key personnel would be diverted from current work to meet the production needs of the DOD-embedded computers, which have obsolete architectures. Therefore, the key personnel would lose their current technological expertise. A smaller number of industry officials felt that Instruction 5000.5X would encourage competition in DOD procurements. However, these officials generally represented companies currently under DOD contracts implementing approved architectures.

WHAT EFFECT WOULD INSTRUCTION 5000.5X
HAVE ON THE CURRENT COMPUTER INDUSTRY?

Most industry officials and some military officials stated that Instruction 5000.5X would effectively eliminate many competent computer companies from the militarized embedded computer market. Very few companies are willing to compete on procurements mandating obsolete architectures. Under the current Navy program for the development of the AN/UYK-43 and 44 computers only two companies responded.

WOULD INSTRUCTION 5000.5X LOCK
DOD INTO OBSOLETE TECHNOLOGY?

Instruction 5000.5X will minimize DOD's opportunities to capitalize on new architecture developments in the commercial marketplace. Computer architecture is a rapidly evolving

technology and has a profound effect on the application of computer technology. As the computer industry improves the application of high-order languages, such as Ada, it also needs improvements and innovations in computer architectures to better support the use of high-order languages.

Most of the industry officials stated that DOD would definitely have obsolete equipment due primarily to the fact that the approved architectures in Instruction 5000.5X are or will be obsolete by the time they are implemented. We were also told that these architectures do not lend themselves to efficiently utilize the new DOD programming language Ada and will not be able to fully capitalize on the anticipated software cost savings Ada was designed to yield.

The architectures listed in Instruction 5000.5X do not include many modern concepts such as stack-oriented architectures, memory-to-memory architectures, efficient multiprocessing support, multiple concurrent tasking support, pipelined architectures, signal processing architectures, image processing, and array computers. Therefore, technological advances in computer architectures will be ruled out because of Instruction 5000.5X.

ARE COMMERCIAL OFF-THE-SHELF COMPUTERS
CURRENTLY AVAILABLE THAT COULD SATISFY
DOD'S MAJOR NEEDS FOR EMBEDDED COMPUTERS?

Militarized versions of off-the-shelf commercial computers are available, work very effectively with modern software, and can offer current computer technology at reasonable costs. Commercial computers have the advantage of giving program managers the latest technology and the most effective and efficient architecture for the particular job. Commercial computers also (1) have lower life-cycle costs, (2) can be militarized to the point where they are rugged enough for combat, (3) can help ease the logistics support burden, and (4) provide more competition.

Today, nearly all of the research and development in electronics is funded by the commercial sector (particularly in computer technology) and is available to DOD through the purchase of militarized commercial products. Military use of commercial technology would significantly reduce applications development and software life-cycle costs. For example, if DOD utilized commercial architectures, most of the associated research and development costs of the architecture and systems software would be borne by the manufacturer and not DOD. Although DOD is spending hundreds of millions of dollars for customized architectures, it has not offered adequate justifications for its dominant reliance on noncommercial architectures.

The market for embedded computers represents about 5 percent of the total computer market. This means that all embedded computer activities must compete for technical resources with a market that is about 20 times larger. We believe that DOD would do better

by utilizing the resources of the entire market by opening the procurements to all the computer industry. Industry could then offer its best architectures, technology, and software support on a system-by-system basis.

Militarized versions of commercial computers already exist. For example, militarized versions of Data General Corporation and Digital Equipment Corporation computers are used in a variety of weapons, communications, and electronic warfare projects within all three services. Ruggedized IBM minicomputers are used extensively in the Marine Corps' Source Data Automation Program. Ruggedized, shock mounted, and straight commercial versions of other vendors' hardware are widely used in command and control and intelligence applications through the AN/GYQ-21(V) program.

Commercial hardware itself is becoming more and more rugged because ruggedness is being required in laboratory, manufacturing, control, vehicle, airborne, and shipboard environments. More dense circuitry and improved packaging have contributed to this trend. As a result, today's computers operate successfully under more adverse conditions than yesterday's and will perform even better in the future.

It is also argued that commercial technology will help solve DOD's problem of logistical support and wartime survivability. Because computers are using less circuit cards than before, the problem of maintenance or logistic support is diminishing. Past computers, in the 1970s, had up to 200 circuit cards compared to today's equivalent that uses only 13 circuit cards and is smaller and faster. Today's AN/UYK-19 (a military version of a commercial computer) uses only 13 circuit cards and the new "B" model only 7 circuit cards.

Adherents of commercial technology argue that it is less expensive and more practical to stock entire computer spares in the field (as the field replaceable unit) than to have more maintenance people in the field to diagnose and swap out an individual problem. Their logistics remedy is to place whole units in the field and ship them back to a central depot for repair. This reduces costs because an individual technician can service a greater number of machines. Also, as computers become smaller, it will be more cost effective to stock entire computer spares in the field.

It is easier to diagnose hardware problems due to advances in commercial technology. New hardware design allows relatively unskilled personnel to isolate problems. Field maintenance will consist of replacing the field unit. This will be facilitated by self-testing logic in the unit and fault-isolating diagnostics.

Throwaway computers are becoming possible because many field units will be a single circuit board, not several cabinets of electronics. Commercial industry is already at the point where several boards are throwaway units. With the high cost associated

with maintaining equipment in the field, DOD could probably justify throwing away an even higher percentage of circuit boards. Units that can and should be repaired will be shipped back to a central depot where commercial vendors will repair the failed component. This will free critical military personnel from learning skills that are more readily available from private industry. While this service could be provided by the hardware manufacturer, it is also possible to compete this separately.

Survivability and the ability to maintain continuity of operations will be enhanced not only by simplified logistics but also by distributive processing. In the future, as computers become smaller and smaller, we will see individual computers designed to implement individual functions or to support an individual commander or operator in the field. These individual computers will then be connected together in a network, both locally within a building and more remotely over wider areas. When one of these computers is down, for whatever reason, it can be quickly disconnected from the network and a spare unit plugged into its place.

Lastly, opponents of standard architectures argue that using commercial hardware will increase competition. Instead of using standard architectures for all programs, there would be competitive selection of a computer system for each major new program. The competition would be based on technology as well as price. The pressure to win new programs would encourage the suppliers to introduce new technology without added cost to DOD.

SHOULD STANDARDIZATION OCCUR AT THE
INSTRUCTION SET ARCHITECTURE LEVEL OR AT
THE HIGHER LEVEL LANGUAGES, SUCH AS ADA?

Cost effective standardization for DOD-embedded computers requires standardization at the high-order language level, such as Ada. The proliferation of languages contributes significantly to the high cost and poor quality of software used in military computer applications. According to the DOD High-Order Language Working Group, none of the many programming languages used in the military is suitable as a standard language for military applications--including the Navy's CMS-2, the Air Force's JOVIAL, and the Army's TACPOL which have been established as interim standard languages. Four of the five architectures currently listed under Instruction 5000.5X are oriented to using these languages.

Studies made by the Defense Advanced Research Projects Agency and by Decisions and Designs, Inc., predicted that as the standard DOD language, Ada will result in substantial cost savings DOD-wide through common software, improved programmer productivity, and new technical features. According to these studies, DOD could save as much as \$24 billion from 1983 to 1999.

Because DOD is implementing Ada as its standard programming language for military applications, computer manufacturers are

currently developing Ada-oriented computers and Ada compilers using the latest technology with substantially improved performance and reliability. As a result, the need for standard computers and instruction set architectures has diminished.

The consensus of opinion from most of those we interviewed during this review was that DOD should standardize at a high-order language, such as Ada, and let the computer industry innovate at the architectural and hardware levels.



THE SECRETARY OF DEFENSE

WASHINGTON, D.C. 20301

2 APR 1982

Honorable Jack Brooks
Chairman
Committee on Government Operations
House of Representatives
Washington, D. C. 20515

Dear Mr. Chairman:

This is in reply to your recent letter which provided a copy of GAO letter report, "DoD Instruction 5000.5X, Standardization Set Architectures for Embedded Computers," (MASAD-82-16), dated January 27, 1982, (OSD Case #5889).

I must respectfully disagree with your conclusions that:

- "...this (proposed) policy would lock DoD into the use of inferior technology."
- "...DoD would not be able to take advantage of private industry's technical innovations."
- "...it would severely restrict competition to those companies willing to help DoD implement obsolete technology."

Our experience to date has demonstrated the opposite effect. Hence, we believe the proposed policy is an established success, even before formal issuance.

The GAO was invited to observe meetings of the recent Defense Science Board review of this subject. However, the two groups reached quite different conclusions. Further, the conclusions and recommendations of the GAO report do not track with their earlier reviews of this same issue ("The Department of Defense's Standardization Program for Military Computers--A More Unified Effort is Needed," June 18, 1980, LCD-80-69).

One recommendation from that earlier report, and echoed in the current letter report, is that Ada should be implemented as DoD's standard programming language. We agree. We followed GAO's recommendation to establish a tri-Service program office for that purpose.

Experience with the principles of 5000.5X, although it has not been formally issued, are:

- There were 12 significant bidders for the Army Military Computer Family (MCF) Program based upon the Government MIL-STD-1862 Instruction Set Architecture (ISA). Earlier, when the program was based upon a commercial ISA, there were only two.
- There are over 20 suppliers of computers built to MIL-STD-1750 including four from England (Enclosure 1).
- Significant cost avoidances have been demonstrated on the F-111 upgrade and the F-5GII aircraft as a direct result of this available competition.
- The Army is using MIL-STD-1750 equipment in at least two systems.
- There is a formal agreement for the Air Force to use the Army's MIL-STD-1862 ISA when available.

As we read the current report, there are two salient differences in viewpoint between our position, which has evolved over the past six years, and that of the GAO and a small segment of the computer industry:

- Defense must be concerned with the acquisition of adequate equipment (minimum-essential with provision for realistic growth) and the life-cycle support of that equipment. Post-acquisition costs normally run from three to ten times original acquisition costs for hardware. Software costs for a system over its life also range to several times the hardware costs and the proportion dedicated to software is growing.
- We do not agree with the GAO assessment of "commercial" versus "military" technology in this field.

More specific discussion of the report and responses to the specific questions voiced in its Enclosure I are given in Enclosures 2 and 3, respectively.

The Defense Science Board Task Force on Embedded Computer Resources Acquisition and Management has reported out to the full Defense Science Board. Their findings and recommendations with respect to proposed Instruction 5000.5X are summarized in Enclosure 4. The Board, in accepting the Task Force results, recommends that we proceed with 5000.5X and further recommends that we work with industry to seek a means to add "commercial" Instruction Set Architecture to its list. We will aggressively pursue that added recommendation.

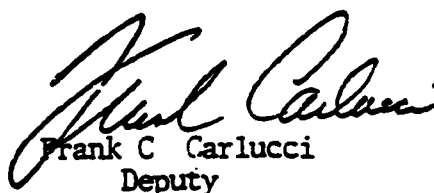
The Task Force's final report is in preparation and should be available by mid-April. I will have a copy provided to you as soon as possible. In the interim, if you so desire, I would be pleased to arrange a briefing on this topic by the Task Force or by my staff.

GAO's second recommendation to limit our standardization efforts to programming language alone is not practical at this time. We do not agree that significant architectural innovations are nearby. We do not believe that a technological filibuster while awaiting breakthrough and production availability is sound. We will follow closely the results of industry-sponsored research as well as that sponsored by DoD and make every honest effort to select the most appropriate equipment for our needs.

We have explicitly designated the Under Secretary of Defense, Research and Engineering, as the Senior Official to be responsible for all computer acquisitions covered by 10 U.S.C. 2315. We issued interim guidelines for these acquisitions and will revise those guidelines by July 31, 1982, in consultation with your Committee and Staff and with the Office of Management and Budget.

Thank you, Mr. Chairman, for your continued interest. I trust that we may work closely together to resolve any remaining concerns.

Sincerely,



Frank C. Carlucci
Deputy

Enclosures
a/s

COMPANIES CURRENTLY MARKETING MIL-STD-1750A PROCESSORS

AIRBORNE PROCESSORS

Bendix
Collins Division, Rockwell
Control Data Corporation
Delco Electronics
Ferranti, Ltd. (U.K.)
Hughes Aircraft Co.
IBM
Litton Industries
Marconi (U.K.)
Martin-Marietta Corp.
McDonnell-Douglas Electronics
Norden Division, United Technologies
Plessey (U.K.)
Singer-Kearfott
Sperry UNIVAC
Teledyne Systems
Westinghouse Electric

SINGLE-CARD 1750A PROCESSORS

Mikros/McDonnell-Douglas
Tracor/General Electric

VLSI MICROPROCESSORS

Fairchild/Honeywell
National Semiconductor
Smith Industries (U.K.)

VHSIC PROCESSORS

Texas Instruments, Inc.
Westinghouse Electric

Enclosure 1

Discussion of GAO Letter Report
"DoD Instruction 5000.5X, Standard Instruction Set
Architectures for Embedded Computers," (MASAD-82-16)

"--Dramatic advances have been made in software technology. DoD has recognized that a lack of a standard programming language is a major contributor to the high cost of developing and maintaining software for military applications. DoD is to be commended for its initiative to fill that void by developing a common high order programming language called Ada. Ada very specifically aims to readily adapt a very wide variety of DoD applications to most present (and future) computer architectures. Ada can potentially encompass the particularly useful aspects of future architectural advances and make their gains available to users, without their having to learn and worry about how the gains were realized. In other words, aggressive pursuit of a standard high order language, such as Ada, could alleviate the software proliferation problem and at the same time permit the Government to fully capitalize on architectural advances."

Response: Ada is indeed coming and, when implemented, will have major impact on the software process. There have not been dramatic advances in software technology and the inefficiencies of continuing the proliferation of the past will live on for decades in specific systems. What improvements have been made in commercial and military software development and support processes have been both slow and inconsistent.

"--Likewise there have been many advances in computer technology. These advances are the result of demands made by the civilian sector for more reliable and rugged computers. And indeed, the civilian sector is starting to impose much stiffer reliability requirements on integrated circuits. These advances will be realized probably at little or no cost penalty because all integrated circuits will be made to the same high standards. There are computer companies already marketing highly reliable computers through the use of innovative architectures. These modern computers have substantially fewer parts and in many cases are a computer on a single board thereby reducing the need for extensive logistics support."

Response: By standardizing at the interface between software and the hardware upon which it runs, DoD can gain access to the best commercial processes without being locked to single suppliers. Although the commercial market does demand better reliability, their needs do not match those of the military environment. The "innovative" architectures used in the commercial world are at a system level and have little or nothing to do with ISAs. The computer-on-a-board is, in fact, an argument favoring ISA standardization because any computer instantiation must have a native ISA and for the logistics argument to hold, standard hardware is implied and hence a standard ISA is demanded, de facto. Working from hardware up implies restricted competition, by any practical measure; starting with the ISA first allows open and equitable competition so as long as the actual selection does not itself force restriction; i.e., so long as the Government has unlimited rights in that ISA.

--Improved competition using militarized versions of commercial computers will open up competition to many firms that would not bid on specifications with DoD-owned architectures. The resulting unit prices will be less because DoD will not pay for duplicating hardware development and control and utility software development as it proposed to do under Instruction 5000.5X. Lower hardware unit costs and high hardware quality are in fact available in the commercial market because of the technology and broader market base."

Response: It has been demonstrated that basing competition on a commercial computer restricts competition. Far more companies have bid on MIL-STD-1750 and MIL-STD-1862 based procurements than have ever been experienced with other approaches. "Commercial computers" is at best a shibboleth for restricted competition. Attachment 1 compares performance and cost of a representative "equivalent" commercial product to the Navy AN/UYK-43 and 44. The facts are contrary to the assertion.

--DoD ownership of architectures would seriously inhibit competition by a significant portion of the computer industry, and therefore DoD would not have the flexibility to capitalize on advances in computer architectural technology in a timely fashion. The ultimate impact would result in DoD very likely running the risk of getting locked into obsolete architectures."

Response: Direct experience has demonstrated the converse of this assertion. Those who have not competed have made a conscious choice not to do so. Far more qualified bidders have competed when ISAs for which the Government has unlimited rights were used as the basis for the procurement.

--DoD would not be able to efficiently utilize the new DoD programming language Ada and will not be able to fully capitalize on the anticipated software cost savings Ada was designed to yield."

Response: The ability to utilize Ada is not dependent upon the ISA, per se. The question is compiler and support software (environment) availability. It is more efficient and economical if the scarce national resources (personnel) are engaged in producing the best possible environments for a few ISAs than to partially optimize a large number of them. NEBULA is an efficient Ada target, despite claims of certain companies to the contrary.

The advantages of Ada will be muted if sound management of the environment-to-hardware interface (ISA) is not achieved.

--The three Services have initiated efforts commensurate with Instruction 5000.5X; for example, the Army's Military Computer Family, the Navy's AN/UYK-43 and 44, and the Air Force's 1750 programs. In a previous report entitled "The Department of Defense's Standardization Program for Military Computers--a More Unified Effort is Needed" (LCD-80-69, June 18, 1980), we were critical of both the

Army and Navy efforts. We made the following statements in that report and believe they are even more valid today:

"***some computer manufacturers are already designing computers with modern architectures that will have Ada compilers available. These or other manufacturers will probably offer follow-on computers that will directly carry out Ada instructions and substantially improve performance reliability. Because these changes provide better support options, such as building more redundancy into systems, they should compel the Department to further evaluate the level of standardization to be achieved before allowing the Army and Navy to commit themselves *** for the long term. (Emphasis added.)"

"We view the need for architecture standardization as a function, in part, of the availability of Ada as the standard computer programming language. Because Ada is being developed to be a machine-transportable language with a relatively low life cycle maintenance cost, the need for standard architectures may be diminished when it is available ***. (Emphasis added.)"

Response: Some computer manufacturers are indeed producing modern architectures for which Ada will be available. The direct-execution machine is not available and practical feasibility is yet in question.

Had GAO completed the quotation it would have included:

"...In the long term, Ada could become the standard architecture when computers that can directly execute the language are developed. However, in the short term, standard computer architectures are needed to reduce life-cycle costs. We believe that those architectures should be common across Service lines, compatible with Ada and the minimum number required to meet common functional requirements and to retain the older languages, such as CMS-2 and JOVIAL, until they are phased out. These measures are necessary to minimize life cycle costs and to facilitate the transition to Ada."

Truncating the quotation completely changed the context and leads to misunderstanding.

Attachment
a/s

Rollm MSE/800 vs AN/UYK-43 (A and B Versions)

	<u>Rollm MSE/800</u>	<u>AN/UYK-43 (A Version)</u>	<u>AN/UYK-43 (B Version)</u>
Max Program Size (using virtual memory)	4.3 Billion	16 Billion	16 Billion
Max Physical Size	8.0 M Bytes	5.0 M Bytes <u>1/</u>	10 M Bytes <u>1/</u>
Word Size (bits)	32	32	32
Package Size	46" x 17.5" wide 24 " deep	48" x 15.8" wide x 22.3" deep AN/UYK-7 Footprint <u>2/</u>	Same as AN/UYK-7 Foot-print only 72" Height <u>2/</u>
I/O Bandwidth (total)	18.20 M Bytes/Sec	>12.0 M Bytes/Sec	>24.0 M Bytes/Sec
Performance/Throughput	1900 KOPS	2500 KOPS	4540 KOPS
Cache Size (K bits)	16 K Bytes	16 K Bytes	32 K Bytes
Power (watts) (max)	3800 Watts	2500 Watts	5500 Watts
Cooling (type)	Air Only	Air/Water	Air/Water
MTBF (hours)	Undetermined <u>3/</u>	>6000 <u>4/</u>	>6000 <u>4/</u>
MTTR (minutes)	<30 Minutes	<15 Minutes	<15 Minutes
Instruction Set (superset of)	Data General NV 8000	AN/UYK-7	AN/UYK-7
Cost	Approx. \$500K	\$400K <u>5/</u>	\$750K
Software Issues	DG operating sys Many HOL compilers Ada in 1983	MTASS-L support Ada in 1985	MTASS-L <u>4/</u> Ada in 1985
Production Deliveries	June 1982	Mid-1984	Mid-1984

- 1/ The capacity will grow as dynamic RAM density increases
- 2/ Fits through submarine hatch (circular)
- 3/ Rollm is now calculating MTBF, available in September 1981
- 4/ Although 6000 hours have been specified, >10000 hours can be expected (according to PMS-408 representatives).
- 5/ AN/UYK-7 with DDMFMs (1981 contract) 512K
- 6/ A multi-tasking operating system should be developed

DoD Responses to Specific Questions
Regarding Proposed DoD Instruction 5000.5X

What effect would Instruction 5000.5X have on the use of competition in the Department of Defense?

Answer: The objective of proposed DoD Instruction 5000.5X is to control and maintain the interface between MIL-SPEC (see attached definitions) computer hardware and the software which runs on it. This interface is termed the "instruction set architecture" or ISA of the computer (a more rigorous definition is given in Enclosure 5). An ISA may be implemented in hardware using whatever design and components, processes, etc., which the producer chooses. DoDI 5000.5X lists a small set of approved ISAs, in all of which the Government has unlimited rights (i.e., can freely use).

When recent procurements have been based upon these ISAs, there was a demonstrated increase in the numbers of qualified bidders participating. In the recent competition for the Army's Military Computer Family (MCF) based upon MIL-STD-1862 or the NEBULA ISA, there were 12 bidders--more than had been experienced on any similar procurement not based on ISAs in which the Government had unlimited rights.

The Air Force has had similar, but more extensive, experience with their procurements based upon MIL-STD-1750, another ISA on the proposed 5000.5X list and in which the Government has unlimited rights. There are currently 23 suppliers of equipment built to the MIL-STD-1750 ISA, four of which are from England. Enclosure 1 lists these companies. Further, when MIL-STD-1750 was used as the basis for the computers for the F-111 upgrade program, the resulting competition allowed the opportunity for insertion of new technology and reduced the price of the computers by almost 50%. Prior to invoking MIL-STD-1750, procurements were sole source in many instances, and otherwise restricted competition at best.

The net of it is that application of the principles of DoDI 5000.5X has invariably enhanced competition and reduced cost--this is demonstrated experience and not speculation on what the effects might be.

What effect would Instruction 5000.5X have on the current computer industry?

Answer: As noted above, the instruction encourages and opens up competition. On any given procurement, a broader section of the industry can compete on a more equitable basis than was possible under earlier acquisition strategies. Of course, any given computer manufacturer may choose not to participate because of a corporate

philosophy which permits it only to market in an environment which minimizes competition, or if it perceived that the competitive environment could negatively effect its eventual market share. How this will balance out over time is yet to be determined. Clearly, the proposed instruction will reduce the present extent of sole-source acquisitions of proprietary products, and that may be interpreted by some of the industry as negative; from the Government's perspective and that of the broader industry sector, it must be viewed as positive.

Would Instruction 5000.5X lock DoD into obsolete technology?

Answer: Recall that the ISA is the interface between software and hardware. Standardization at that interface has been demonstrated to allow and even encourage the injection of new technology. Although there are some performance parameters to be specified at that interface, both within and in addition to the ISA, there is essentially transparency at the ISA to the implementing technology. The expert opinion, upheld by experience, is that breaking the sole-source procurement syndrome or its handmaiden, weak competition, will accelerate the injection of technology--quite contrary to locking the Department in to obsolescence. That lock-in not only to technology but also to supplier, is precisely the lesson to be learned from earlier approaches.

Are commercial off-the-shelf computers currently available that could satisfy DoD's major needs for embedded computers?

Answer: In some cases the answer must be "yes." And, in those cases where the logistics implications are tractable, where there is no need for militarization, and where there can be a fair and open competition, then some preference should be given to the off-the-shelf approach. Generally, "off-the-shelf" means a common, commercial product intended for the civilian marketplace and one designed and produced in a proprietary way. The need for any change in the physical or electrical characteristics of the machine will break the "off-the-shelf" model. In some instances, several manufacturers could start with their proprietary "soft" product and through redesign and modification provide acceptable militarized products--that would not meet the essential "off-the-shelf" test. Further, it is often true that existing software must, for reasons of economy, schedule or risk, be accommodated. Except in rare cases, the implication is sole-source procurement and the well-known consequences. Where those negative aspects are acceptable in the balance, off-the-shelf is a viable acquisition-strategy alternative and should be so considered. In no way can it be considered the predominant method.

Should standardization occur at the instruction set architecture (ISA) level or at the higher level languages, such as Ada?

Answer: Standardization is required at both the high order language (HOL) and instruction set architecture levels. HOL standardization improves the efficiency of applications code and the transportability. ISA standardization is necessary to allow reuse of systems and support software from system-to-system. Together they can provide significant productivity and economics. Either alone is, at best, a partial solution to these problems. Until the non-stop, never-fail true High Level Language (HLL) machine is practical and available, ISA standardization is the only way to enable application and system software reuse. It should also accelerate the availability of Ada, per se, and, hence, perhaps hasten the reality of the HLL machine.

MAJOR FINDINGS AND RECOMMENDATIONS OF THE DEFENSE SCIENCE BOARD
TASK FORCE ON EMBEDDED COMPUTER RESOURCES (ECR) ACQUISITION AND
MANAGEMENT

1. Major Findings

The controversy over proposed DoD Instruction 5000.5X is largely between interests which have as primary concerns the procurement of military embedded computers and interests which have as primary concerns the operation, maintenance, deployment and post-acquisition support of these same computers. The rhetoric from both sides often neglects the complex issues inherent in these concerns. The situation is further complicated by the fact that procurement issues are immediate and measurable in dollars soon, while providing a cost-effective deployment and support environment for the rest of the century is harder to quantify. In other words, the time-frame foci of the two sides in this controversy are quite different. The arguments are further confused over the questions:

1. "Should there be a standard set of ISAs at all?"
2. "If there are to be standards, should they be Government-owned or Commercial ones?"

Embedded Computer Instruction Set Architectures (ISAs) should be standardized. This action is necessary if the Department of Defense is to improve its position with respect to (1) software development costs and, further, it is (2) mandatory to improve the logistic support and, above all, the battlefield sustainability of automated systems.

Given our finding of support for ISA standardization, Government control over the ISAs is required. It is unwise to commit for a 20-year period to an ISA which can be changed or differently supported at the whim of a commercial entity. Too many considerations of competitive follow-on procurement, overseas manufacture and ongoing support are at stake.

ISA Standardization will affect the competition for hardware procurements and this effect may be either positive or negative in specific cases. The Task Force did not find the arguments against standardizing this software/hardware interface convincing. With standardization of a "good" set of Government-owned ISAs, industry has the opportunity to enter competition on an equal basis.

Current Service programs based on ISA standardization are well founded and in accord with general policy intent; there are, however, areas wherein they could be strengthened. The issue of multiple producers for logistically-identical hardware is still an open question and one which deserves more study.

High Order Language (HOL) standardization has had a positive effect on software development and support costs. Implementation of the policy to use HOLs has, however, been spotty.

DoD's Ada Program is well based and is proceeding well. The Army's portion, the Ada Language System (ALS) should be strengthened in view of its criticality to the MCF Program and to efforts of the other Services.

DoD Directive 5000.29 is the basic policy document for automated defense systems and we feel its revision and update are urgent matters for action. Problems still exist with regard to computers and software and it is too early to force this area into the routine oversight pattern afforded to more mature technologies. Work is needed to rationalize DoD's inconsistent set of specifications and standards for computer resources. Further, recent legislative changes must be promulgated and 5000.29 is the proper vehicle for this.

There is no consistent management process across the OSD Staff and the Military Departments relating to computer (automation) technology. OSD has been relying on a committee management approach which was born of necessity in 1976. The scope of automation in systems and in their support has far outstripped this approach.

2. Recommendations

The proposed DoD Instruction 5000.5X be issued. It should be reviewed for consistency with acquisition regulations and the scope should be clarified, e.g., where does it apply and where does it not. OSD should actively manage implementation of the policy and control both the waiver process and changes to the approved ISA list.

The Military Departments proceed with current programs—the Army Military Computer Family (MCF), the Navy Tactical Embedded Computer Resources (TECR) Program and the Air Force MIL-STD-1750 Program— but with some modifications:

- The Army should increase the quantity of Advanced Development Models (ADMs) and/or Full Scale Development Models (FSDMs) of the MCF hardware to support early software development experience.
- Both the Army and Navy reconsider their acquisition strategies to make use of multiple producers of logistically-identical computer hardware if economically feasible.
- The Air Force should bound the number of unique implementations of MIL-STD-1750 with an eye toward achieving logistic cost avoidance.
- The Air Force should consider adding Input/Output specifications and other hardware-independent details to MIL-STD-1750.

A Senior Policy Official (USDRE) be designated and a more consistent management approach be established across the OSD staff and within the Military Departments. The first action has been taken concurrently with our deliberations. The Under Secretary should now:

- Determine the level and mode of operation and provide revised guidance to the Components regarding implementation of P.L. 97-86.
- Establish a more uniform set of R&D objectives and acquisition policies.
- Provide for OSD management of selected generic automation programs, such as has been done for Ada and the Very High Speed Integrated Circuits (VHSIC) Programs.
- Task the official to whom this new policy designation is delegated to be a principal advisor to the DSARC on computer and software (automation) matters.
- Strengthen OSD's oversight of Components' computer and software R&D and acquisition activities.
- DoD Directive 5000.29 be revised expeditiously. It should incorporate the changes in acquisition approach enabled by 10 U.S.C. 2315. It should emphasize the "software-first" approach to system

design and development and should encourage use of rapid software prototyping and competitive concept definition. Source selection procedures should formally include software considerations (this may require DAR revisions).

- DoD develop and apply a consistent set of computer hardware and software specifications and standards across the Department. Current work by the Joint Logistics Commanders Joint Policy Coordination Group for Computer Resource Management (JPCG-CRM) may be an adequate basis but DoD should assure uniform implementation of the resulting product, whatever its genesis.
- Intersystem and local network protocols should be developed from the existing DoD standards and emerging international standards. Standards for data elements should also be developed.
- OSD conduct a careful review of the Defense Acquisition Regulations (DARs) to assure that they are in step with computer acquisition policy. In particular, care should be taken that time limitations in the DARs do not drive technology insertion schedules through misinterpretation. The DARs must not attenuate sharing of Government-developed applications or support software.
- DoD Instruction 5000.31 be updated quickly to remove outdated languages and to add Ada.
- The Under Secretary should assure adequate and continuing support for the Ada Joint Program Office (AJPO).
- The Service Ada Programs be strongly supported and that they continue to receive adequate staff and funds. The various components of this program should be better coordinated.

DEFINITIONS PERTINENT TO DISCUSSION OF DoDI 5000.5x

The Instruction Set Architecture (ISA) of a computer is the set of instructions (e.g., add, store-into-memory) used to program a computer, augmented by other minimal information available to a programmer (e.g., interrupt capability).

Each computer has an ISA which is the logical basis for its physical structure.

An ISA may be implemented in many different ways since its specification is independent of hardware.

An alternative definition of Instruction Set Architecture is:

"An ISA is the specification of the interface between software and hardware. It includes the attributes of a computer as may be seen by a machine [assembly] language programmer or the target code generator of a compiler for a high-order language (HOL). It describes the conceptual structure and functional behavior of a computer as distinct from the organization of the data flow and controls, logic design or physical implementation."

MIL-SPEC Computers are specially designed for the military environment. The performance required is delineated, usually, in a Military Specification (MIL-SPEC) or Military Standard (MIL-STD) which form an integral part of the contract for the acquisition of the specific material. MIL-SPEC equipment is generally not available "off-the-shelf" and must be designed and fabricated "to order" to meet a set of performance and/or environmental requirements, including the form factor of the equipment. Examples are space-borne equipment; airborne weapons-delivery, navigation or flight-control computers.

Embedded Computers are those computers incorporated as an integral part of, dedicated to, or required for the direct support of, or for the upgrading or modification of, major or less-than-major systems. Thus this term refers not only to those computing devices buried deeply within subsystems as radars, radios, missiles and the like but more generally to computers which are used to perform a portion of a larger task such as fire-control, automatic testing, navigation, and threat warning. The key discriminator is whether the application is computation alone or whether computation is merely a subtask to be performed as a part of a larger activity. In the industrial realm, "embedded" computers would be found managing process control in a steel mill or a chemical plant or as the automation element in an automobile. In a hospital or research laboratory, computers are embedded in CAT scanners, in scintillation counters, in gas chromatographs, in EKG or EEG equipment — they perform specialized and dedicated tasks and are not, in general, available to support the general computational or data processing needs of the organization and hence are subject to a more specialized selection process than classical

• Automatic Data Processing Equipment (ADPE).

Embedded Computer Resources include the totality of resources required for the support and operation of "embedded computers." Thus, the term includes, but is not necessarily limited to:

- Computer Data
- Computer Hardware
- Computer Programs
- Documentation
- Personnel
- Supplies
- Services
- Training
- Software

- * Support Software
- * Utility Programs
- * Test Software
- * Operational (Applications) Software
- * Training Software

Tactical Computers are those used in the tactical, strategic, intelligence, cryptologic or command and control environments of military operations. Generally this would mean deployed equipment on ships, aircraft or with the fielded army or dedicated to the training and support of military personnel as a part of their military assignments as contrasted to the CONUS administrative support of the Department, the Services or the Agencies. The term is even less precise than "embedded" and, so, we prefer not to use it.

Ruggedized Computers are those which are specially designed and tested to ensure resistance to such environmental hazards as shock, vibration, humidity, sand, salt, temperature, operational and storage extremes, altitude and explosive hazards without the requirement for redesign or change of the computer itself. That is the protection is provided through shock mounting, enclosures such as transit cases, or other means to isolate a fundamentally commercial instrument from an environment it was not basically designed and produced to withstand. Some consider that a complete mechanical redesign holding the electrical design constant constitutes "ruggedized" equipment;

however that would be the extreme case and would require different production lines and techniques so one is talking about a completely separate product in this case and the "learning curve" is broken, support requires differing parts and training, etc.

Commercially Available: Computers are those available to the general public from the equivalent of a published catalog at preestablished prices and requiring essentially no design effort. Clearly, options may be required and adaption or configuration to the customers needs may be required but this tailoring should be a small portion of the final selling price. It would also be expected that the equipment would be supported by existing field service personnel and commercial logistics systems. This class would generally exclude "ruggedized" equipment although equipment for steel mills, chemical plants and petroleum fields may be both commercially available and ruggedized. Commercially available computers are frequently delivered with software which may have application to common business, technical or educational needs. Software associated with MIL-SPEC, MIL-STD or special purpose computers designed for tactical, strategic, intelligence, cryptologic or command and control environments is generally unique to the special military application.

Computers to Support the Military Mission of the Department of Defense could include all of the above categories. The test here is the application and whether it is on the critical path to the fulfillment of the military mission of the Department, either defensive or offensive. The test is the application of the equipment and not its source. All computers and related equipment destined for the "excluded" applications of 10 U.S.C. 2315 are included in this general description, i.e., "...if the function, operation, or use of the equipment or services—

1. "involves intelligence activities;
2. "involves cryptologic activities related to national security;
3. "involves the command and control of military forces;
4. "involves equipment that is an integral part of a weapon or weapons system; or
5. "...is critical to the direct fulfillment of military or intelligence missions."

Not included in the fifth excluded application area is equipment and services "...for routine administrative and business applications..." Interim guidelines for acquisitions under 10 U.S.C. 2315 maintain this gray area under case-by-case oversight to assure that the legislative intent is preserved. The definition will be modified as we gain experience in this area.

Computers in Direct Support of Systems Excluded by the FY1982 Defense Authorization Bill which excludes acquisition of certain systems from the Brooks Act (PL89-306) may include all of the above described categories. More particularly, they are those computers which remain under exemption five when

the ... "routine administrative and business applications including payroll, finance, logistics, and personnel management applications)" are excised. There are systems in these applications areas which fail the "routine" test as they are on the critical path toward the fulfillment of the "military mission." Automatic test equipment, deployable logistics systems such as the Combat Support System, NALCOMIS, Global Weather Service systems, the Satellite Control System, the NORAD system, Avionics Intermediate Shop equipment, and shipboard machinery monitoring systems are examples of this area of application.

F-16 MULTINATIONAL STAGED IMPROVEMENT PROGRAM

COL DAVID J. TEAL
DEPUTY SYSTEM PROGRAM DIRECTOR
F-16 SYSTEM PROGRAM OFFICE

1. Welcome to everyone! It is a great pleasure for me to be able to tell you a little bit about what we are doing on the F-16. I'll show you some of the things coming up on the F-16 aircraft which I think exhibits a new way of doing business. We are structuring ourselves so that we can insert technology rapidly at the field level, and improve our combat capability quickly at a reasonable cost. That's why we are excited about it. I am also very grateful to Jeff for all the time he has put into organizing this conference so that you could be here today to hear the significant investment we are making in the implementation of the architectural standards. We also must plan for the support and follow-on effort necessary to sustain the implementation and modernization of all the standards as appropriate in the future. We would like to make all the tools being developed available within the Department of Defense. Hopefully a great savings in cost and time will be realized to you. That's the way I see it from my position. That's why it's important that we get together to exchange this information, because it's good for us and it's good for you.

2. I would like to show you what we are doing within the F-16 and why standards are a key part of this activity. We are structuring an architecture for the F-16 Multi-National Staged Improvement Program (MSIP), which will allow us to insert many new systems on the F-16 over the next few years. We have already implemented Stage 1, delivered in November 1981. Stage 1 consists of the Group A provisions, the structure, wiring and hard-point changes. The cockpit has also been wired. The idea being to reduce future retrofit costs when we replace our existing avionics.

3. Stage 2 consists of the "core avionics" which are the essential equipment needed to support the growth systems. This stage consists of a new cockpit including a wide angle raster HUD, up-front communications-navigation-identification system, a data transfer unit, two multifunction displays, improved radar, radar altimeter, an improved cooling turbine, enhanced fire control computer, and advanced central interface unit. This aircraft is scheduled for delivery in July 1984.

4. Stage 3 is a continued integration of systems which enable us to fight air-to-air and air-to-surface, day-night, as well as in weather. It includes internal ECM, provisions for JTIDs, GPS, LANTIRN, and BVR missile. A lot of thought has gone into the structure and the capability to incorporate new technology without constantly taking the aircraft back to the depot, which reduces the number of aircraft ready for combat.

5. Jovial J-73 higher order language is being used to program the Stage 2 core avionics, and MIL-STD-1750A instruction architecture is being used as the software/hardware interface. The core avionics are tied together with a 1553 data bus. Our 1553 data bus has dual protocol 1553 (USAF/F-16) as well as 1553B capability. MIL-STD-1760 is being adopted as the aircraft-to-weapon interface.

6. Now, the good news! "They haven't yet standardized their weapons."

7. The F-16 application of the standards started late in 1980/early 1981. However, once the ball got rolling things happened very quickly, thanks to Jeff and a lot of other folks from the program office and General Dynamics who got together to make this very ambitious and somewhat risky program work. In this particular case, we are talking about the implementation of MIL-STD-1750. One of the primary efforts is the implementation of 1750 in a microprocessor. Jeff organized the first two forums early and late in 1981 with the idea of sharing our tools with other organizations within the Department of Defense. We have progressed in the development of these standards through PDR's and CDR's and we are starting now in the printed circuit board developments.

8. The standards are being applied to the central processing units in the core avionics which will be programmed in J-73 HOL. The fire control computer, advanced central interface, multi-function display set, data entry cockpit interface set, and data transfer unit will all be designed to implement 1750, as well as operate on the 1553 data bus.

9. All of the new LRU's will either have a multi-card CPU or use the 1750 microprocessor. Also, some of the new government furnished equipment, both the radar and HUD, will use 1750 and MIL-STD-1589. The existing equipment on the F-16 will remain the same, i.e., the INS. However, as we upgrade and recompute some of these systems, there will be opportunities to insert the standards.

10. I won't speak for the managers of programs outside the control of the F-16 SPO, but will simply state that they are targets of opportunity. AMRAAM represents a very large market for the microprocessor. LANTIRN will use the Delco multiscard 1750 CPU. The electronic warfare community is a big supporter of the standards and are looking very closely at them as new systems come along. Both Seek Talk and Combined Altitude Radar Altimeter are potential candidates for the standards. Perhaps we can find a way for PLSS, CPS, and JTIDS to see some practical applications, if they still have the lead time. Hopefully there are representatives here from those programs and contractors who may see the light - get the religion - this morning, if you have not already done so.

11. This is how we understand 1750A. It is an 'interface standard' and not a computer or an F³ specification, or a sole source environment. It is important that everyone understand so that what we do is in line with Department of Defense policy. Once 1750 is adopted, an upgrade in hardware technology will not force you to throw out all the old software.

12. This is the way we are managing the 1750 development program. General Dynamics is managing the overall program for the SPO. They have defined all the specifications per SPO direction and have held a competitive procurement. Fairchild was selected as the best over all solution for the 1750 program.

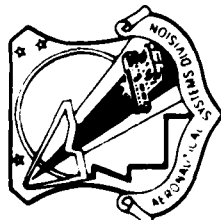
13. Our program is a parallel development effort. Since the 1750 microprocessor will not be ready for FSD, the Z8002 was chosen as an interim solution. However, the FCC will have 1750 and all of the core systems will be programmed in Jovial. The current plan is to insert the 1750 CPU in place of the Z8002 prior to production.

14. As far as the HOL goes, the current F-16 uses J3B. It was a very successful program. It is a very good HOL and very beneficial. We look forward to continued benefit with J73.

15. The two J73 compiler vendors under contract are Advanced Computer Techniques and Proprietary Software Systems. ACT is targeting to Z8002 initially in support of full scale development, and PSS is targeting to 1750. Later, both companies will develop the alternate target generator. The initial deliveries have been made, look good and are on schedule.

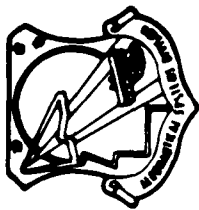
16. By the end of 1982, in conjunction with the first flight of the MSIP flight test program, both the ACT & PSS compilers will be completed. They will be duly targeted to Z8002 as well as 1750A.

17. Thanks for listening. I hope this overview gave you an understanding of the application of avionics interface standards on the F-16. Let us know if we can be of assistance to you. We welcome your ideas and suggestions.



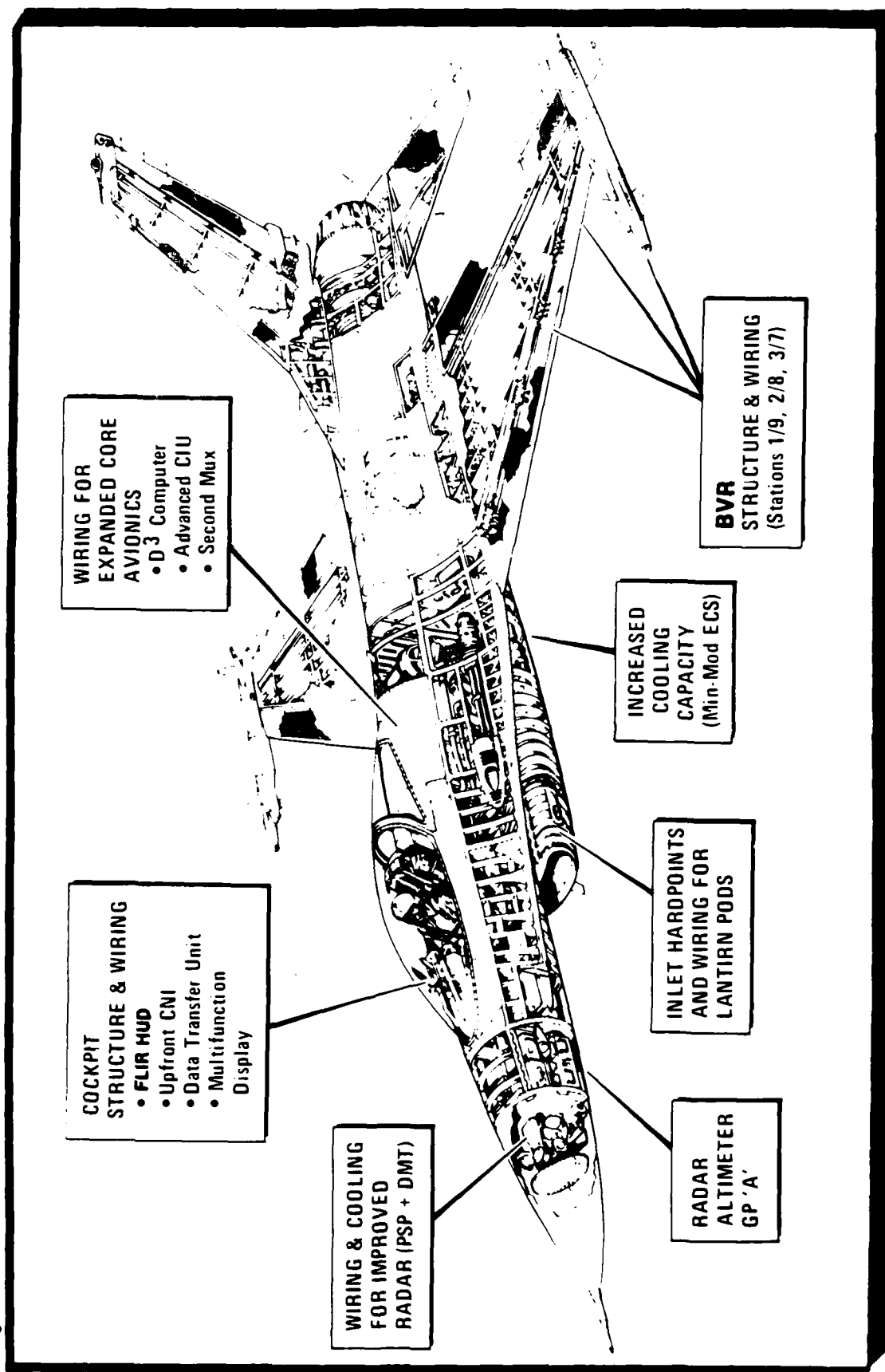
MULTINATIONAL STAGED IMPROVEMENT PROGRAM

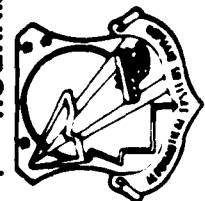
- STAGE I PROVISIONS FOR GROWTH SYSTEMS
MINIMUM ESSENTIAL STRUCTURE, WIRING, AND COOLING
PROVISIONS TO PRECLUDE COSTLY AIRCRAFT RETROFIT
- STAGE II AIRCRAFT AVIONICS TO SUPPORT GROWTH
DEVELOPMENT AND PRODUCTION OF ESSENTIAL AIRCRAFT
AVIONICS EQUIPMENT TO SUPPORT GROWTH SYSTEMS
- STAGE III GROWTH SYSTEMS INTEGRATION
INSTALLATION OF GROWTH SYSTEMS TO MEET MISSION
REQUIREMENTS



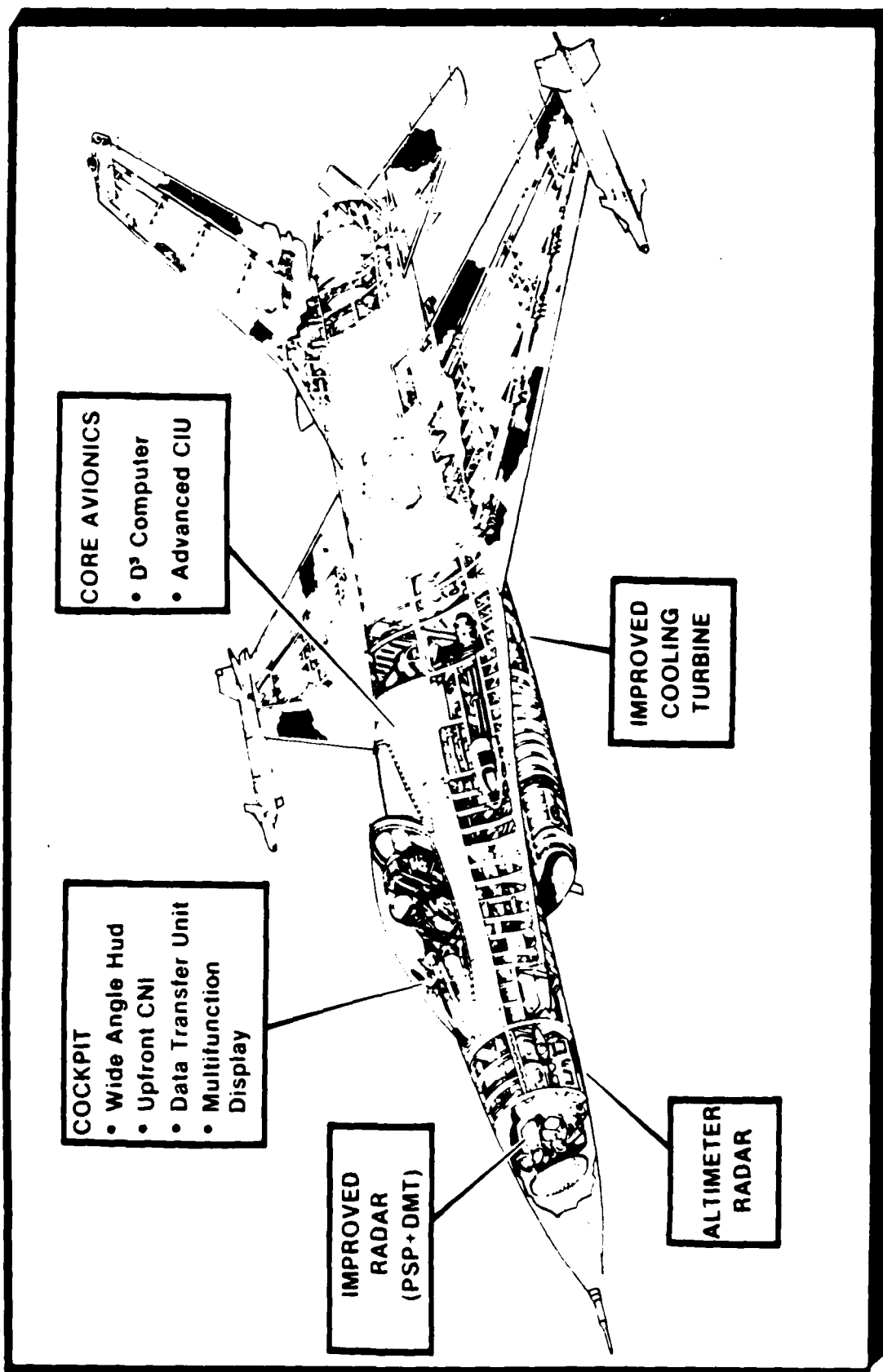
MSIP STAGE I

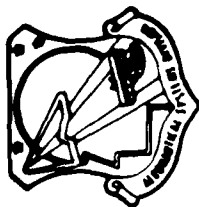
50A1



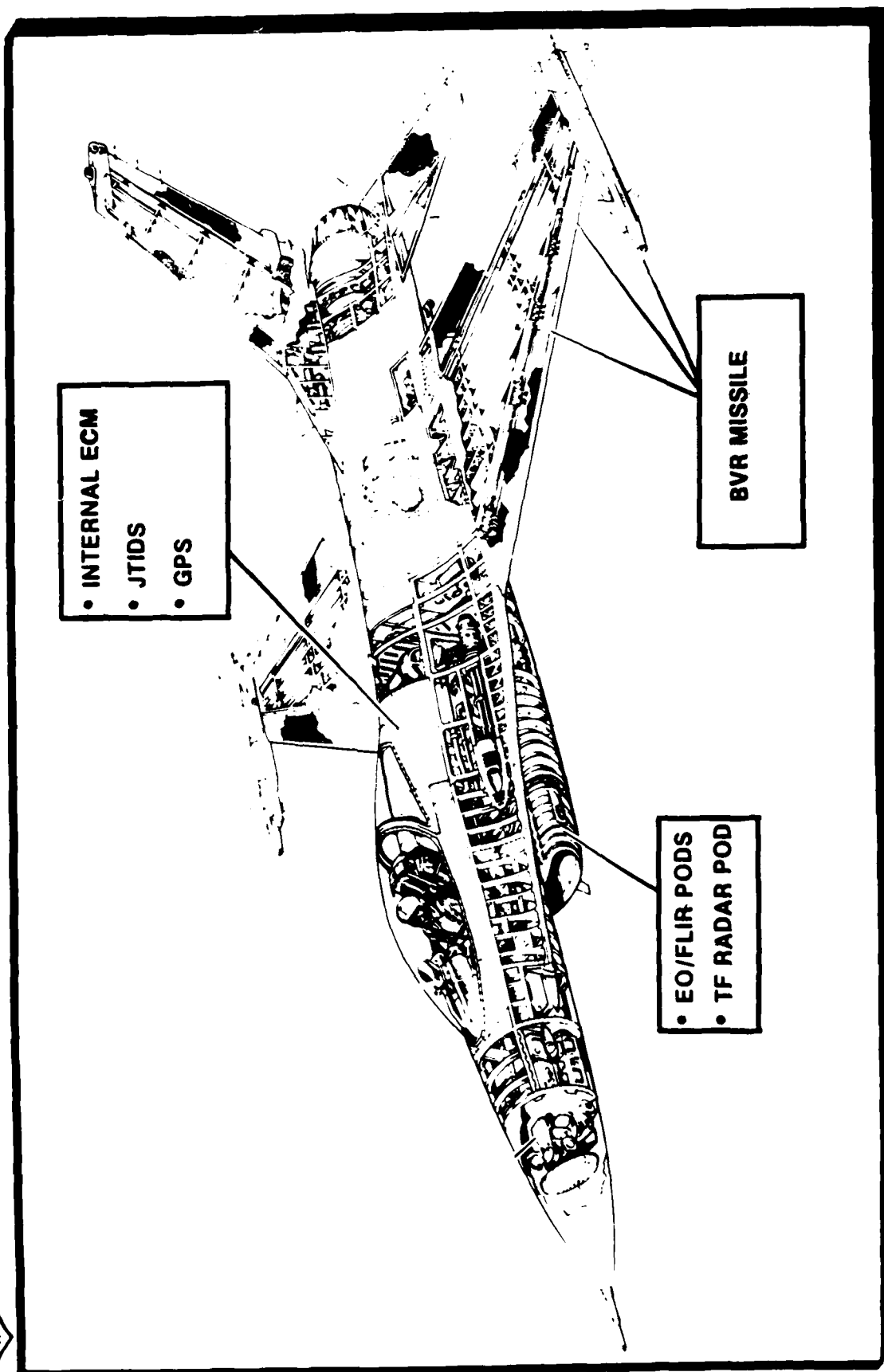


MSIP STAGE II





MSIP STAGE III



- INTERNAL ECM
- JTIDS
- GPS

- EO/FLIR PODS
- TF RADAR POD

BVR MISSILE

RATIONAL STANDARDIZATION IS INTEGRAL PART OF MSIP II

BACKGROUND

- BASIC F-16 SYSTEM USED AVAILABLE STANDARDS
 - ✓ MIL-STD-1553
 - ✓ JOVIAL H01
- BOTH PROVIDED BENEFIT TO GENERAL DYNAMICS AND USAF

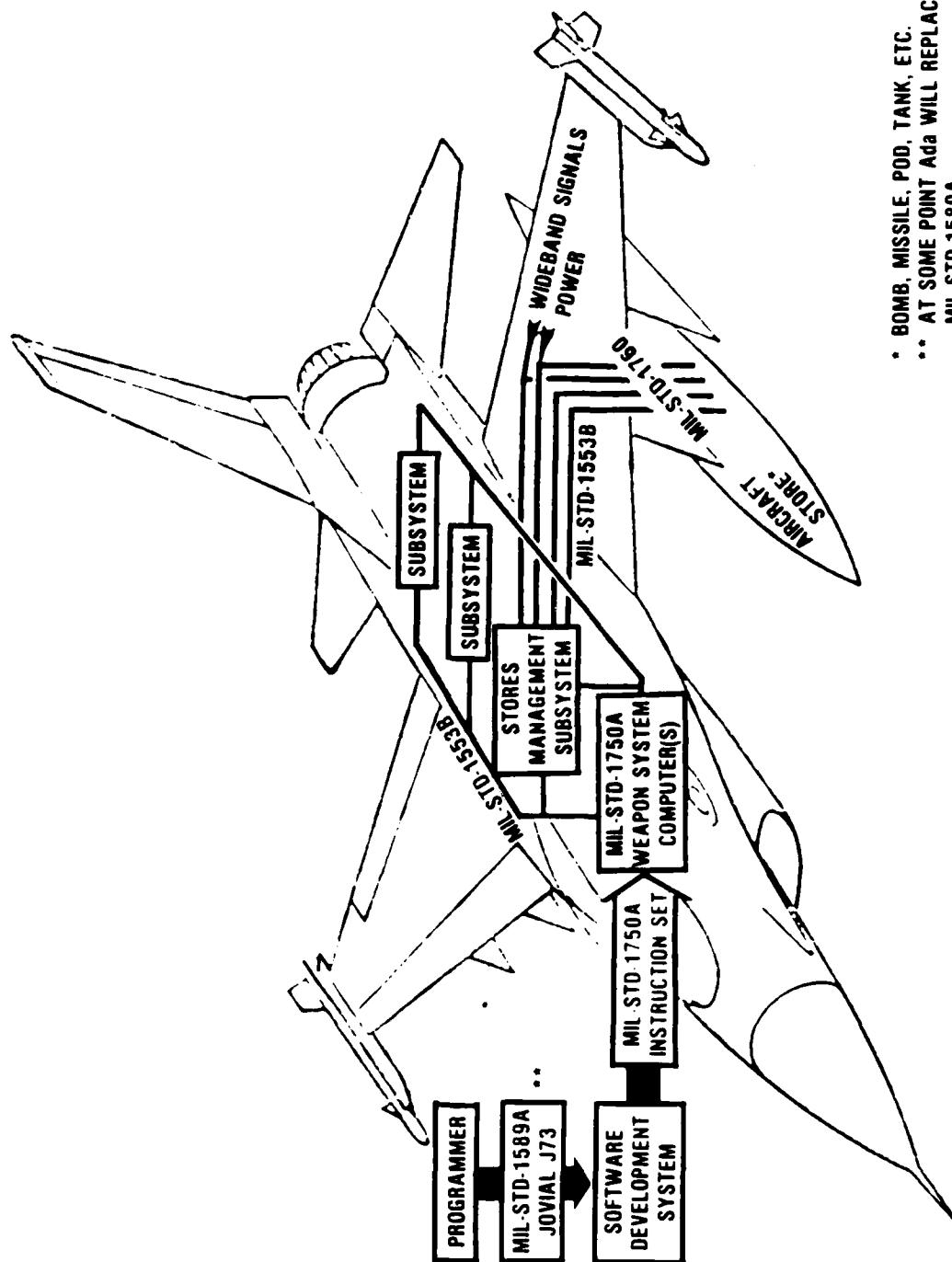
ADDITIONAL BENEFITS EXPECTED FROM:

- ✓ MIL-STD-1553B
- ✓ MIL-STD-1760
- ✓ MIL-STD-1589B
- ✓ MIL-STD-1750A

MSIP II PHILOSOPHY

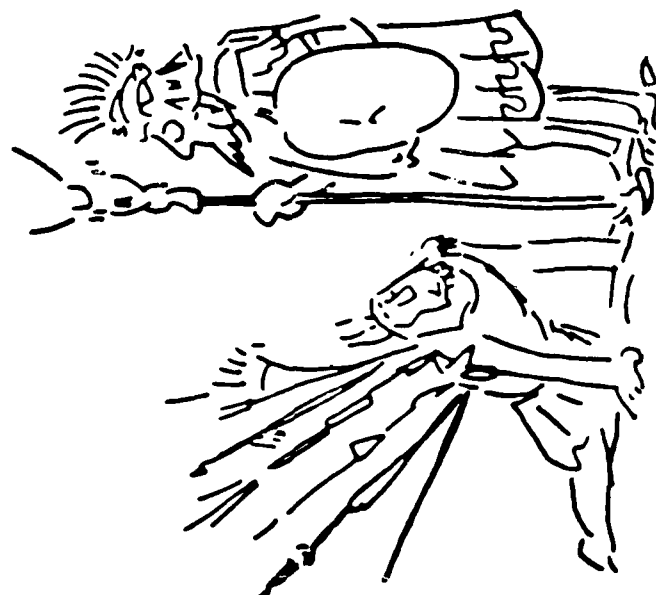
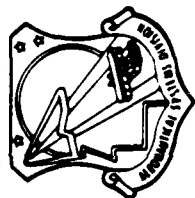
- USE STANDARDS UNLESS
THERE ARE COMPELLING
REASONS NOT TO
- RATIONAL STANDARDIZATION

BASELINE ARCHITECTURE

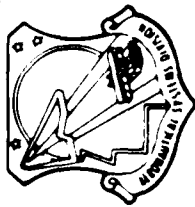


* BOMB, MISSILE, POD, TANK, ETC.
 ** AT SOME POINT Ada WILL REPLACE
 MIL-STD-1589A

F-16 PROGRAM



**"FIRST THE GOOD NEWS:
THEY HAVEN'T YET STANDARDIZED THEIR WEAPONS"**

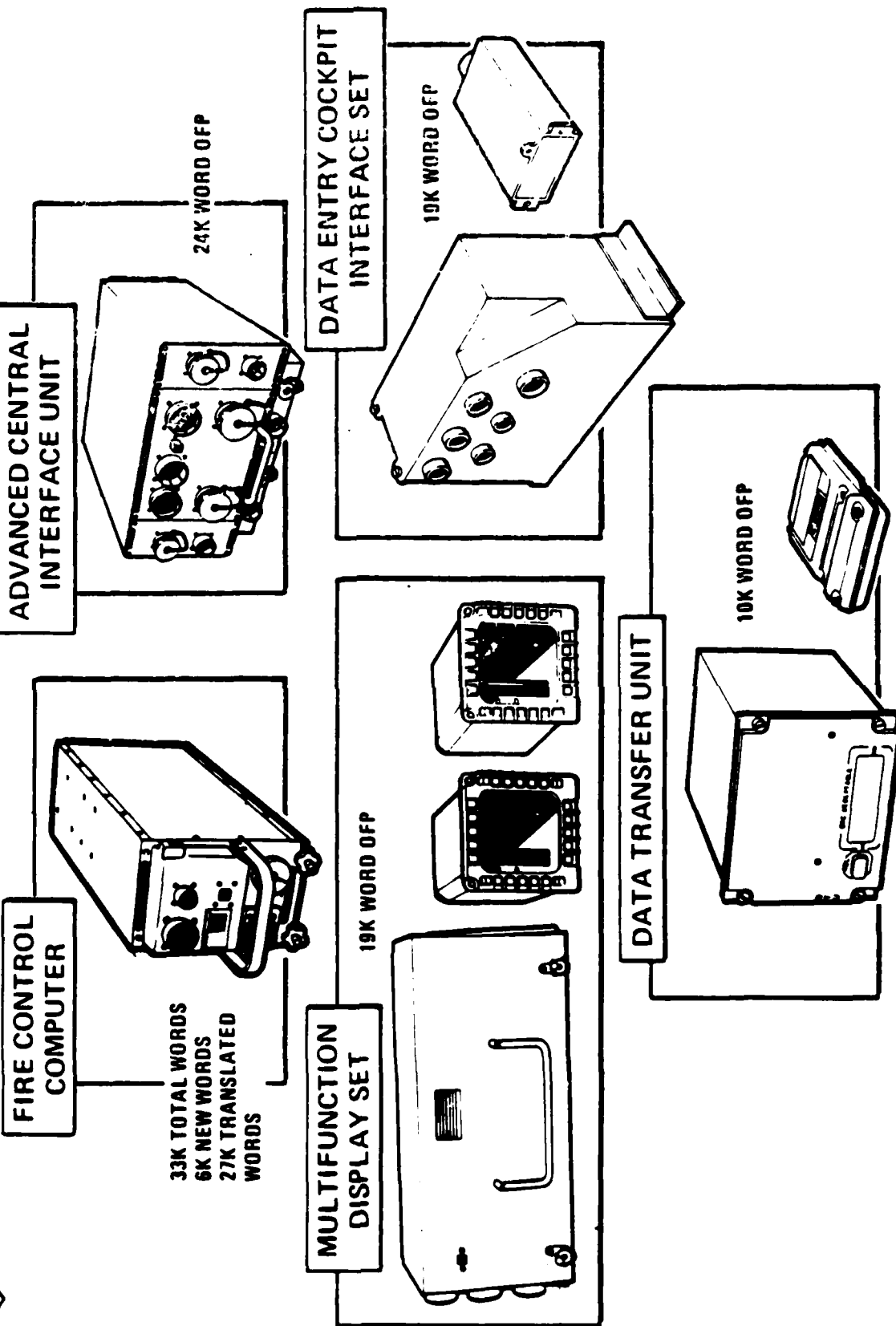


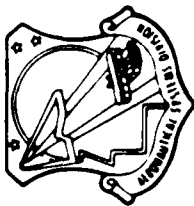
POSITIVE STEPS TO 1750 & 1589

AFSC STANDARDIZATION CONFERENCE	18-20 NOV 80
INDUSTRY SURVEY	DEC 80
F-16/AX COORDINATED EFFORT	JAN-MAR 81
F-16 RFP TO GD (MICROPROCESSOR DEVELOPMENT)	3 APR 81
FIRST TECH FORUM (1750/1589)	28 MAY 81
SOURCE SELECTION	JUN-JUL 81
F-16 GO AHEAD	10 SEP 81
SECOND TECH FORUM (1750/1589)	20 OCT 81
PDR	2-3 NOV 81
CDR	17-18 FEB 82
THIRD TECH FORUM (1750/1589)	5-6 MAY 82
PCB RFP APPROVED	12 FEB 82



CORE AVIONICS





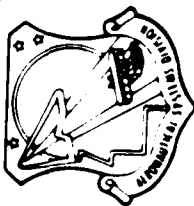
APPLICATION OF MIL-STDs IS A CENTRAL FEATURE

- ALL PROCESSORS PROGRAMMED IN JOVIAL/J-73 (MIL-STD-1589B)
- ALL PROCESSORS USE MIL-STD-1750A INSTRUCTION SET ARCHITECTURE
- STORE INTERFACE WILL CONFORM TO MIL-STD-1760
- NEW DATA BUS CONFORMS TO MIL-STD-1553B



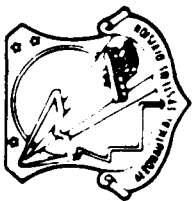
F-16 MSIP IMBEDDED COMPUTER SYSTEMS

LRU	F-16 STATUS	CFE/GFAE	CPU
FIRE CONTROL COMPUTER	NEW	CFE	1750A MULTICARD
DATA ENTRY ELECTRONICS UNIT	NEW	CFE	1750A MICROPROCESSOR
PROGRAMMABLE DISPLAY GENERATOR	NEW	CFE	1750A MICROPROCESSOR
DATA TRANSFER UNIT	NEW	CFE	1750A MICROPROCESSOR
CENTRAL INTERFACE UNIT	NEW	CFE	1750A MICROPROCESSOR
FIRE CONTROL RADAR	NEW	GFAE	1750A MULTICARD
HEAD UP DISPLAY	NEW	GFAE	1750A MULTICARD
INERTIAL NAVIGATION UNIT	CURRENT	GFAE	UNIQUE
PAVE PENNY	ON HOLD	GFAE	UNIQUE
CENTRAL AIR DATA COMPUTER	CURRENT	CFE	UNIQUE
FLIGHT CONTROL COMPUTER	CURRENT	CFE	ANALOG



F-16 MSIP PHASE III COMPUTER SYSTEMS

LRU	F-16 STATUS	CFE/GFAE	CPU
AMRAAM	NEW	GFAE	Z8002/UNIQUE
LANTIRN TARGET POD	NEW	GFAE	1750A MULTICARD
LANTIRN NAVIGATION POD	NEW	GFAE	1750A MULTICARD
ASPJ	NEW	GFAE	UNIQUE
ALR-67/69	NEW	GFAE	
ALR-69	CURRENT	GFAE	
SEEK TALK	NEW	GFAE	Z8002 (1750 POT)
GPS	NEW	GFAE	UNIQUE
JTIDS	NEW	GFAE	UNIQUE
PLSS	NEW	GFAE	UNIQUE
CARA	NEW	GFAE	UNIQUE (1750 POT)



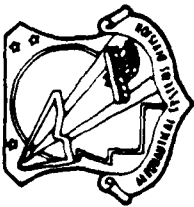
WHAT IS MIL-STD-1750A?

- IS

A GOVERNMENT-OWNED INSTRUCTION SET
ARCHITECTURE STANDARD IN PUBLIC DOMAIN

- IS NOT

A COMPUTER OR COMPUTER DESIGN
F3 SPECIFICATION
OR ONE VENDOR'S "BLACK BOX"



MIL-STD-1750A MICROPROCESSOR PROGRAM MANAGEMENT

GD/FW WILL MANAGE THE PROTOTYPE/PREPRODUCTION PHASES

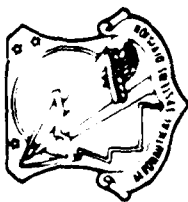
SPECIFICATION DEFINES USER REQUIREMENTS

FAIRCHILD SELECTED VIA OPEN COMPETITION

APPLICABLE DOCUMENTATION

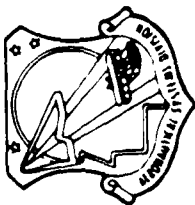
MIL-STD-1750A 2 JUL 80 ALL OPTIONS

NOTICE 1 (24 JUL 81, CONTROL BOARD)



STAGED IMPLEMENTATION APPROACH

- FIRST DEVELOPMENT ACIU, PDG, DEEU AND DTU WILL UTILIZE Z8002 PROCESSORS AND J-73 HOL
- FCC UTILIZE 4 CARD 1750A APPROACH FOR FSD
- MICROPROCESSOR PARALLEL DEVELOPMENT
 - PROTOTYPE 1750A REPLACEMENT MID 83
 - FLYABLE 1750A REPLACEMENT LATE 83
 - FIRST PRODUCTION ACIU, PDG, DECIS, AND DTU WILL INCORPORATE 1750A PROCESSORS AND J-73 HOL



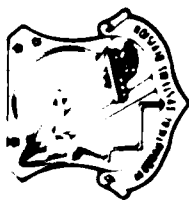
INTEGRATED SUPPORT SOFTWARE PRODUCT

D32

19E1

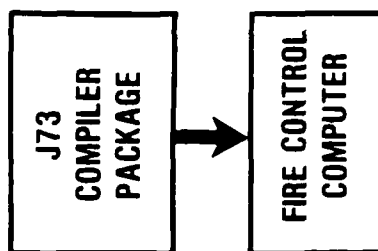
- APPROACH
 - INITIAL
 - USE EXISTING GFE JOVIAL J73 COMPILERS (WITH GD MODS) TO ACCOMPLISH TRANSLATION TASK, INITIAL OFP DEVELOPMENT
 - DEVELOPMENT
 - ACQUIRE AND USE INTEGRATED PRODUCTS TARGETED TO 1750A AND Z8002 (TWO SUPPLIERS, ACT AND PSS, IN WORK)
 - MATURE AND OPTIMIZE THE PRODUCTS
 - PRODUCTION
 - QUALIFY THE 1750A SYSTEM (AND Z8002 IF REQUIRED)

F-16 PROGRAM

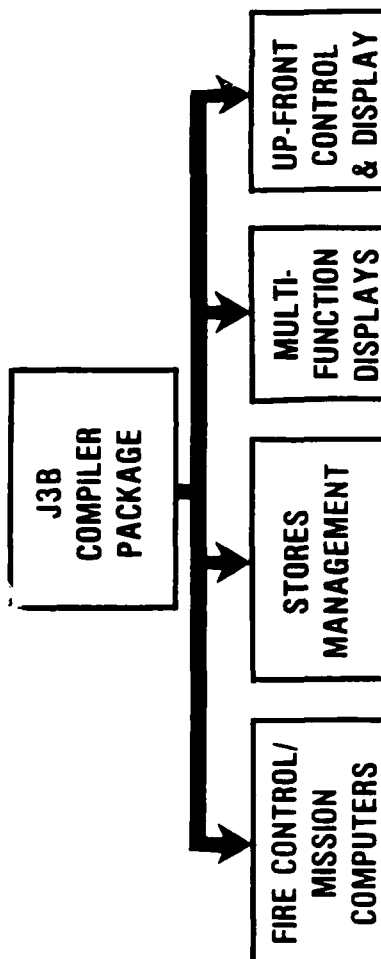


HIGHER ORDER LANGUAGE (HOL) BENEFITS DEVELOPMENT

- HOL PROVED IN INITIAL F-16 DEVELOPMENT

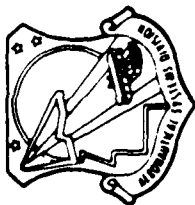


- HOL HAS A CENTRAL ROLE IN MSIP II DEVELOPMENT



- PROVEN BENEFITS
 - IMPROVED STRUCTURE, CLARITY
 - EFFICIENT CODE
 - SIMPLIFIED TROUBLE-SHOOTING, AUDITING, VERIFICATION
 - EASIER TO MAINTAIN
- GENERAL DYNAMICS AND USAF ARE WORKING TO EXTEND THIS COMMON PACKAGE TO THE BROADER AVIONICS COMMUNITY:
 - ✓ DATA TRANSFER UNIT
 - ✓ ADVANCED RADAR
 - ✓ OTHER GFE
 - ✓ LANTIRN
 - ✓ TFR

F-16 PROGRAM



SUPPLIER STATUS

ADVANCED COMPUTER TECHNIQUES

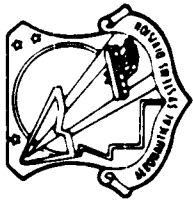
- FIRST WORK ON Z8002 (1750)
- RECEIVED PRELIMINARY COMPUTER PROGRAM DEVELOPMENT PLAN, SOFTWARE QUALITY ASSURANCE PLAN, CONFIGURATION MANAGEMENT PLAN, AND LANGUAGE SPECIFICATION
- COMPLETED CDR, PDR - GOOD DESIGN DETAIL. COMPLETED SQA PR. PRELIMINARY DEVELOPMENT SPEC AND DRAFT PRODUCT SPEC
- DETAIL DESIGN PROCEEDING AND SCHEDULE LOOKING GOOD. INITIAL DELIVERY VERY GOOD.

PROPRIETARY SOFTWARE SYSTEMS

- FIRST WORKING ON 1750A TARGET. INITIAL DELIVERY UNDER EVALUATION
- RECEIVED PRELIMINARY DEVELOPMENT AND PRODUCT SPEC, LANGUAGE SPECIFICATION, AND PLANS. COMPLETED PDR & CDR & SQA PR.
- PRELIMINARY ANALYSIS OF BASELINE COMPILER AND ERRORS AND OPTIMIZATIONS. AFWAL NEW RELEASE CORRECTED ALL BUT ONE OF OUR ERRORS.
- NEW AFWAL VERSION CAN MAKE OFPs, BUT CODE IS NOT OPTIMAL.
- STUDY IN PROGRESS ON INCORPORATING BUILT-IN FUNCTION USER-LIBRARY INTO SEA COMPILER
- EXPECTING OPTIMIZATIONS TO BE ADDED AND DELIVERED IN JUNE 1982.

J-73 SCHEDULES

F-16 PROGRAM



ITEM	1981			1982			1983			1984		
	J	F	M	J	F	M	J	F	M	J	F	M
CONTRACT AWARD MSIP MILESTONES												
ADVANCED COMPUTER TECHNIQUES												
• SUPPORT SOFTWARE DESIGN REVIEWS												
• DESIGN												
• INCREMENTAL DELIVERIES												
Z8002 TARGET												
MIL-STD-1750A TARGET												
• TEST												
PROPRIETARY SOFTWARE SYSTEMS												
• SUPPORT SOFTWARE DESIGN REVIEWS												
• DESIGN												
• INCREMENTAL DELIVERIES												
MIL-STD-1750A TARGET												
Z8002 TARGET												
• TEST												

F-16 MIL-STD-1750A MICROPROCESSOR DEVELOPMENT PROGRAM

➡ **MANAGEMENT OVERVIEW**

RANDAL MOORE

PROGRAM OVERVIEW

DAN WILNAI

FUNCTIONAL AND TECHNICAL DESCRIPTION

SHAI MOR

APPLICATIONS HARDWARE

HENRY LYNN

ENGINEERING TEST/EVALUATION EQUIPMENT

DAVID SCOTT

**THIRD TECHNICAL FORUM
WRIGHT-PATTERSON AFB
5 MAY 1982**

**R. K. Moore
Third Technical Forum
5 May 1982**

Chart # 0

R. K. MOORE

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #1

MIL-STD-1750A does not standardize hardware, it standardizes computer instruction set architecture. We are developing hardware that may become a defacto standard, but that should not be confused with the larger issue of 1750 standardization. The existing 1750 computers on the F-16 use a multicard 1750 processor--the LANTIRN processor as well as the Fire Control Computer that Delco is supplying. The chip set that Fairchild is developing for General Dynamics and the Air Force will form the basis of a single-card central processor. We are undertaking the chip set development because the F-16 SPO requested it. The benefit for us is in software development downstream, but the big payoff is for the Air Force.

CHART #2

Four MSIP systems are targeted to use the 1750 microprocessor: the advanced central interface unit, the data entry cockpit interface set, the data transfer unit, and the multifunction display set. We are at the point in the MSIP program where we are beginning to make provisions and go through the proposal actions required to actually put the chips sets into the equipment.

CHART #3

This is a little more detail of the phased approach to introducing the 1750. We are going through a full equipment development with our Z8000 microprocessors as the core. Parallel with the final stages of development and testing of the Z8000-based equipment we will be developing a single-card 1750 processor that's meant to replace the card that houses the Z8000 processor. We intend to have full production capability for both 1750- and Z8000-based equipment.

For software development we will have dual-target Jovial software tools that are being developed for us by two suppliers. We haven't decided how we are going to handle assembly language software--whether to write a translator to go from Z8000 to 1750 assembly code, or start from scratch and do the whole thing over. The machine-specific test programs will have to be written from scratch.

The testing plan is to go through full safety-of flight, qual and flight test with the Z8000-based equipment, then phase the

R. K. MOORE

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

1750 into the testing on a card basis. Our philosophy is that we must test the equipment, and that the type of CPU is not critical for most of that testing. Therefore, we intend to demonstrate that the 28000 cards and the 1750 cards are equivalent as far as the equipment is concerned. We intend to go through a limited flight test to verify that equivalency does exist.

Provisions have been implemented within all equipments that will allow use of either the 28000 or the 1750 chips. Mr. Henry Lynn will give more details on what we've done along those lines. We expect to make a decision this year as to whether to go forward with full development of the 1750 card. When I say decision that means GD writing a proposal and the Air Force making a decision. We are in the early stages of that proposal effort now. Production commitment to 1750 will be made when we have demonstrated that the two processor cards are equivalent.

CHART #4

This next slide is the schedule for the microprocessor program. I would like to show you a few things on this schedule. Notice that two CDRs are shown. We had an incremental CDR. At the 1st increment the design looked very good. We are quite satisfied, quite pleased with the work that Fairchild had done. There was an open item at the conclusion of that increment: the type of package.

CHART #5

We decided that we did not want to use a 64 pin package on one-hundred mil pin centers. Fairchild has developed a package that is a 64 pin package with 50 mil centers. We did a lot of research and decided that most wiring board manufacturers were not comfortable with 50 mil technology right now. The package that we ultimately agreed upon is designated as a "gull-wing" package. The gull-wing is a dual-in-line package modified by Fairchild to be surface mounted. The pins are on 50 mil centers. We expect the printed wiring board technology to be the same as for a flat pack. We plan to go forward with the qualification of this package, but the details have not been worked out.

CHART #4 (Continued)

The deliveries of the prototype chip sets and the engineering evaluation equipment are shown with a one month delay. That was a result of some extended time taken earlier in the program to redefine aspects of the architecture. The slip did not affect the end result of our program, the preproduction chip sets. However, as Fairchild progressed further in the design and layout and actual figures on the chip die sizes were available, we became

R. K. MOORE

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

dissatisfied with the size of the CPU chip. We saw that if we took some more time we could probably bring the dimensions down, and end up with a more producible design, and feel much more comfortable with the rest of the program. So we are taking some time to do more analysis on the design and make some changes. We expect that we will extend the schedule once again. The schedule that Dan Wilnai will present will indicate Fairchild's ideas as to what the program should look like in the future. The formal program schedules have not yet been updated. I wanted to alert you so that when you see Dan's presentation, you will understand why his schedule and mine are different.

BACKGROUND

- o MIL-STD-1750A STANDARDIZES ARCHITECTURE - NOT HARDWARE

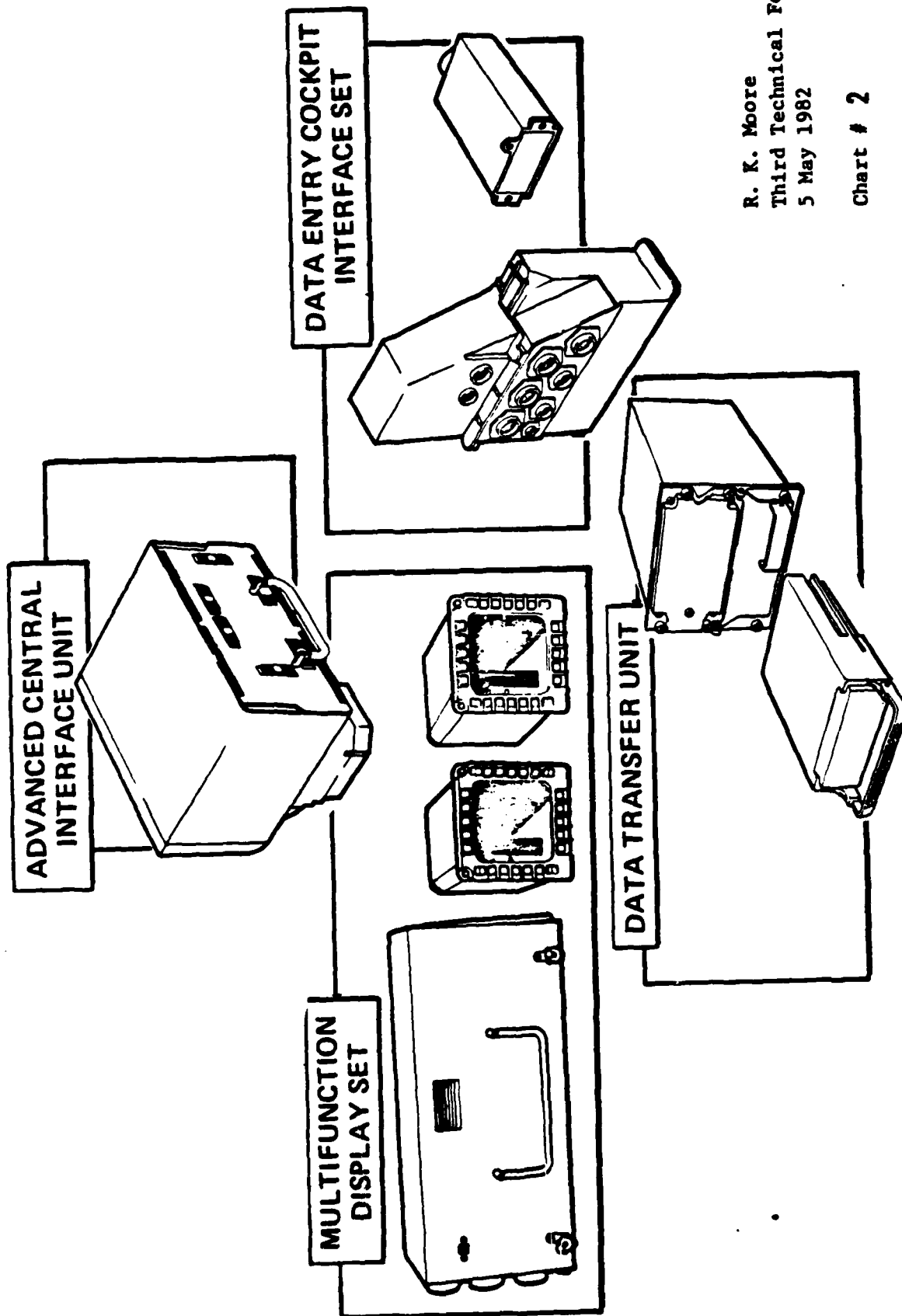
DESIGN

- o EXISTING 1750A COMPUTERS HAVE MULTICARD CPU
- o 1750A CHIP SET FORMS BASIS OF SINGLE-CARD CPU
- o PROCESSOR DEVELOPMENT REQUESTED BY F-16 SPO.

R. K. Moore
Third Technical Forum
5 May 1982

Chart # 1

FOUR F-16 MSIP II SUBSYSTEMS ARE TARGETED FOR POTENTIAL USE OF 1750A MICROPROCESSOR



R. K. Moore
Third Technical Forum
5 May 1982
Chart # 2

007002

1750A IMPLEMENTATION - PHASED APPROACH

(ACIU, DEEU, DTU & PDG)

- o HARDWARE DEVELOPMENT
 - oo Z8002 - BASED EQUIPMENT
 - oo 1750A - BASED REPLACEMENT CPU CARD
 - oo PRODUCTION CAPABILITY FOR BOTH
- o SOFTWARE DEVELOPMENT
 - oo DUAL-TARGET JOVIAL SOFTWARE TOOLS FOR OFF'S
 - oo ASSEMBLY LANGUAGE TRANSITION FOR BURN-IN PROGRAMS TO BE DEFINED
 - oo MACHINE SPECIFIC SOFTWARE MUST BE REWRITTEN FOR 1750A
- o TESTING PLAN
 - oo SOF, QUAL, AND FLIGHT TEST WITH Z8002
 - oo PHASE 1750A INTO TESTING ON A CARD REPLACEMENT BASIS
 - oo LABORATORY TESTING TO DEMONSTRATE Z8002/1750A EQUIVALENCE
 - oo LIMITED FLIGHT TEST TO VERIFY TRANSITION
- o 1750A PRODUCTION PLAN
 - oo PROVISIONS FOR Z8002/1750A TRANSITION BUILT INTO EQUIPMENTS
 - oo DEVELOPMENT DECISION - MID 1982
 - oo PRODUCTION COMMITMENT - WHEN Z8002/1750A EQUIVALENCE IS DEMONSTRATED
(EST 1Q 84)
 - oo RETROFIT AS REQUIRED

R. K. Moore
Third Technical Forum
5 May 1982

Chart # 3

Chart # 5



F-16 MIL-STD-1750A MICROPROCESSOR DEVELOPMENT PROGRAM

MANAGEMENT OVERVIEW

RANDAL MOORE

▶ PROGRAM OVERVIEW

DAN WILNAI

FUNCTIONAL AND TECHNICAL DESCRIPTION

SHAI MOR

APPLICATIONS HARDWARE

HENRY LYNN

ENGINEERING TEST/EVALUATION EQUIPMENT

DAVID SCOTT

INTERIM PROCESSOR TRANSITION

HENRY LYNN

**THIRD TECHNICAL FORUM
WRIGHT-PATTERSON AFB
5 MAY 1982**

**D. Wilnai
Third Technical Forum
5 May 1982**

Chart # 0

D. WILNAI

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #1

I would like to give you a brief overview of the program at Fairchild. You will hear much more detail about the microprocessor chip set from Shai Mor, and you will hear about the test equipment that we are developing from Dave Scott of GD. Ron Byrne will describe our marketing approach later in the day. All I will do is give you a general idea of the program as a whole, and you will hear the details later.

CHART #2

Our objective in the program is to develop a microprocessor chip set comprised of three VLSI devices. One of them, the main device, is a single chip that implements in a sixty-four pin package the complete 1750 instruction set architecture. It is the CPU in the system. The other two devices are support devices. One of them implements the MMU functions of the 1750; the other one implements the block protect RAM function. In addition to that, we have the task of developing test and evaluation equipment for the program. The first target for the development program is to deliver fifteen prototype chip sets and five units of the test and evaluation equipment. Our current estimate is that we deliver that by February 1, 1983. The next step is to deliver 100 units of fully qualified preproduction chip sets by September 1, 1983. These will be roughly four months behind the original schedule.

CHART #3

Fairchild started the logic design of the 9450 back in June 1981. That was long before we had any indication that we had won the 1750A chip set development contract. The contract was awarded in mid-October 1981. The reason we started then (and we started on Fairchild funds) was because of the tightness of the schedule. The only way we saw of getting to the end of the program in time was to start early. Logic design started in June 1981 and finished by the end of January 1982. We have taken here a somewhat aggressive approach, non-conventional in a sense. Normally in a microprocessor development you will finish the logic design and only then will you start the circuit design and layout. In this program, because of schedule pressure, we started the circuit design and layout early. As you can see, we have finished the circuit design just recently and the first design is going into fab now. As mentioned earlier we realized the design was not the design that we wanted, and that it didn't meet our goals; therefore, we have started an optimization effort. The optimization is centered around logic design and layout. We have decided to take this approach to reduce the risk in the program. What you see now is five iterations of silicon processing. Basically what we did was add another iteration of silicon processing. The first device, which is now going into fab, will be a test device. It will allow us to test the logic as well as a few

D. WILNAI
TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #3 (Cont'd)

things about the architecture. The second iteration will be a result of the circuit optimization phase of the program, and we expect that to come out of fab in mid-October 1982. At 2½ month intervals after that we get the third, fourth, and fifth iterations. You may want to ask why we have elected to have five iterations. Maybe it will work the first time, but nobody knows how many iterations it will take. Our experience has been that it takes between three and five iterations on a chip of this complexity to get fully functional parts. What you see is what we consider to be a realistic schedule, maybe on the pessimistic side. Preproduction deliveries will be September 1, 1983.

CHART #4

So where are we? Again we have completed the design of the 9450, are going into fab with a testing device, and are currently into a circuit optimization effort. At the same time, the two support circuits, the 9451 and 9452 are completed and in fab now. The test equipment development is on target. We have completed the top level design and we don't expect any problems.

AD-A150 583

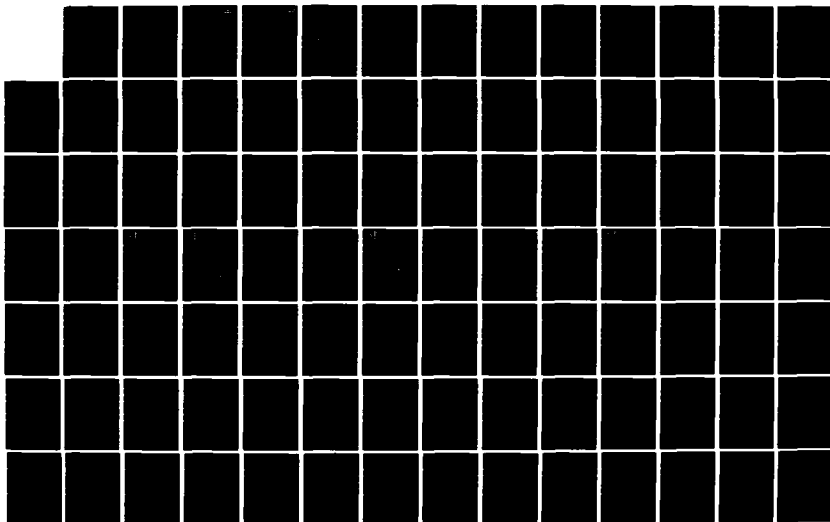
PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16
MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV
WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 86 MAY 82
ASD-TR-82-5011-VOL-1

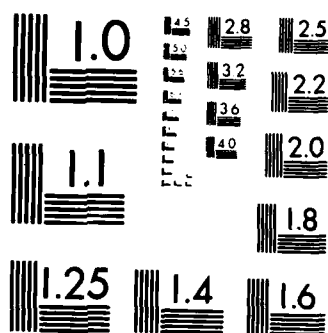
2/6

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

1750A MICROPROCESSOR

PROGRAM DEVELOPMENT

DAN WILNAI

FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

D. Wilnai
Third Technical Forum
5 May 1982

Chart # 1

2

PROGRAM OBJECTIVES

- DEVELOP A HIGH PERFORMANCE MILITARIZED, SINGLE-CHIP MICROPROCESSOR THAT FULLY IMPLEMENTS MIL-STD 1750A INSTRUCTION SET ARCHITECTURE.
- DEVELOP SUPPORT CIRCUITS FOR EXPANDING THE ADDRESSING RANGE TO 1M WORDS WHILE PROVIDING MEMORY PROTECTION:
 - 1) MMU - MEMORY MANAGEMENT UNIT
 - 2) BPR - BLOCK PROTECT RAM
- DEVELOP TEST AND EVALUATION SUPPORT SYSTEM BOTH HARDWARE AND SOFTWARE.
- DELIVER 15 UNITS OF PROTOTYPE CHIPSETS AND 5 UNITS OF TEST SYSTEMS BY FEBRUARY 1, 1983.
- DELIVER 100 UNITS OF PREPRODUCTION CHIPSETS FULLY QUALIFIED BY SEPTEMBER 1, 1983.

D. Wilnai
Third Technical Forum
5 May 1982

Chart # 2

3

PROGRAM STATUS - MAY 82

- DESIGN OF THE F9450 (1750A CPU) STARTED IN JUNE 1981;

ARCHITECTURE DESIGN	- COMPLETE.
DETAILED DESIGN	- COMPLETE.
TEST DEVICE	- IN FAB.
CIRCUIT OPTIMIZATION	- IN PROGRESS.

- DESIGN OF F9451 (1750A MMU) AND F9452 (1750A BPR) STARTED IN OCTOBER 1981;

ARCHITECTURE AND LOGIC DESIGN	- COMPLETE.
CIRCUIT LAYOUT	- COMPLETE.
1st SILICON	- IN FAB.

- ENGINEERING TEST/EVALUATION EQUIPMENT

TOP LEVEL DESIGN	- COMPLETE.

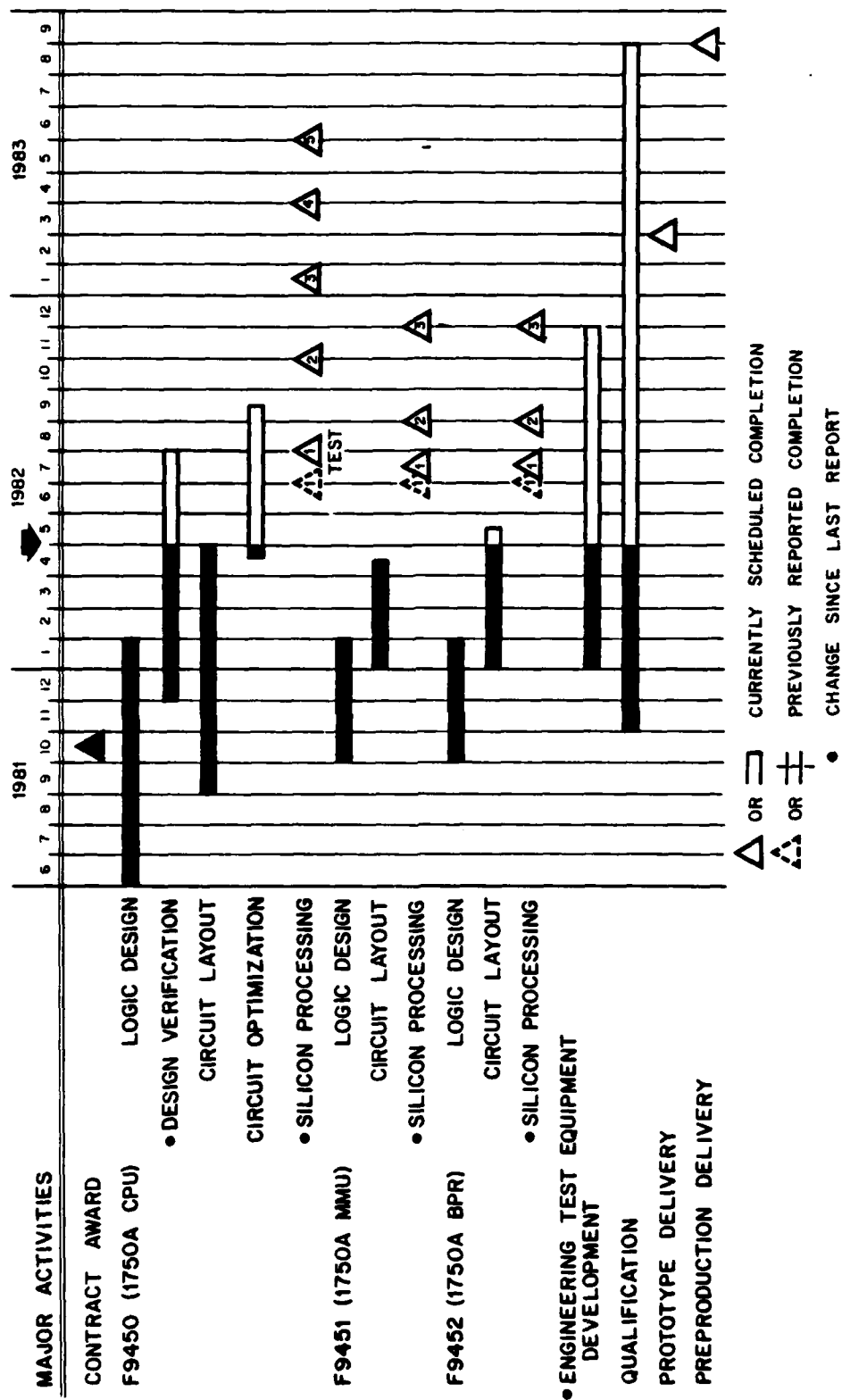
D. Wilnai
Third Technical Forum
5 May 1982

Chart # 3

MIL-STD 1750A MICROPROCESSOR DEVELOPMENT PROGRAM

MAJOR ACTIVITIES

STATUS AS OF MAY 1, 1982



D. Wilnia
Third Technical Forum
5 May 1982

Chart # 4

F-16 MIL-STD-1750A MICROPROCESSOR DEVELOPMENT PROGRAM

MANAGEMENT OVERVIEW

RANDAL MOORE

PROGRAM OVERVIEW

DAN WILNAI

➤ FUNCTIONAL AND TECHNICAL DESCRIPTION

SHAI MOR

APPLICATIONS HARDWARE

HENRY LYNN

ENGINEERING TEST/EVALUATION EQUIPMENT

DAVID SCOTT

INTERIM PROCESSOR TRANSITION

HENRY LYNN

**THIRD TECHNICAL FORUM
WRIGHT-PATTERSON AFB
5 MAY 1982**

**S. Mor
Third Technical Forum
5 May 1982**

Chart # 0

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #1

What I'll describe is the 1750 chip set as it is being developed by Fairchild. First I'll cover the general 1750A chip set characteristics, then some basic chip set configurations, and then details on the three devices that constitute the chip set; namely, the 9450 which is the main CPU, the 9451 which is the MMU or Memory Management Unit, and the 9452 which is the BPR or Block Protect RAM.

CHART #2

No text given

CHART #3

This is the artist conception of the full chip set. There are three 64 pin packages and some other small standard packages. The CPU is a single chip, 64 pin package device. The MMU consists of the F9451 which is a 64 pin device plus four standard RAM chips which contain the two maps plus two bi-directional buffers. The BPR consists of the F9452 which is a 64 pin package device, plus one standard RAM chip which contains the map for the protection bits.

CHART #4

A smaller system which contains only the CPU and the BPR is shown here.

CHART #5

Now let me go through the characteristics of the system starting with the CPU. The CPU is a single 64 pin microprocessor which implements MIL-STD-1750A. It requires a single five volt supply plus an injector current. The technology used is I3L-II which is basically a bi-polar technology. I3L has the advantages of being bi-polar--in other words high speed--without the disadvantages of high power and low density. Another feature of the technology which is inherent in bi-polar is high radiation tolerance. The chip is fully static with a maximum clock frequency of 20 Megahertz. All of the inputs and outputs are

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

compatible with low power Schottky. We have on-chip multi-processing capabilities as will be explained later.

CHART #6

A basic goal of the program has been to develop a high performance chip that is fast over the entire military temperature range. For the F9450 we are talking about 200 nanoseconds for a register to register add and 1.85 microseconds for a 16 x 16 multiply. The F9450 handles single and double precision arithmetic and provides 16 general purpose registers as defined in MIL-STD-1750A. The full complement of 32 and 48-bit floating point instructions as defined in the standard have been implemented on chip. The chip is designed for real time processing environments and has 16 levels of interrupt vectors, also as defined in the standard. The chip can directly access 64 thousand words of memory; with the MMU the system can access up to one mega-word. Two programmable timers are also implemented on-chip.

CHART #7

The BPR is providing the function of write protection in physical memory for the CPU and DMA in 1K blocks. The BPR is implemented using the F9480, a gate array used internally within Fairchild. The gate array was used to reduce development time. The other chip required to implement the BPR is a standard RAM chip which will contain the protection bits.

CHART #8

The MMU provides a translation from logical to physical addresses for both instructions and operands. It increases the addressing space to one mega-word. Protection is accomplished in blocks of 4K words in basically three modes: (1) access key-- matching the lock to the key; (2) write protect-- applies for data accesses; and (3) execute protect-- applies for instruction fetches. The F9480 gate array was again used for the main block of the logic to shorten the design cycle. Four RAM chips contain the address maps. Two bi-directional buffers were necessary to allow use of a gate array package with only 64 pins, as I will explain later.

CHART #9

This is the logic symbol for the F9450--the CPU. The F9450 is a 64-pin chip. Shown are the 16-bit multiplexed bus and all the

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

status lines and handshake signals necessary for bus and I/O cycles, the bus control and arbitration signals, abort and normal power-up discretes, reset, the CPU clock and the timer clock, the trigger-go reset discrete, the eight status lines-- access key and address state-- which are necessary for MMU operation, the external fault inputs, and the nine external interrupts. There are five pins for supply: Vcc, two injector pins, and two ground pins. One pin is not connected.

CHART #10

This is the logic symbol for the F9451-- the main element of the MMU. The left side of the diagram illustrates the signals that interface with the CPU: the information bus (IB), the four AK's, and the four AS's, and all the status and handshaking signals. On the right side most of the signals interface with the four local RAM chips that contain the maps for the instruction and data spaces. The RAL and E/W lines are the output that comes back from the rams-- both the access lock and the execute- or write-protect bit. BOE controls the bi-directional buffers that are part of the MMU system, but physically reside outside the 9451. the memory protect error goes back to the CPU fault inputs. the ADD VAL output signals the memory subsystem and the BPR that the extended address coming from the MMU system is valid. The SPEED input allows the user to select the response speed of the MMU, i.e. the number of wait states inserted by the MMU.

CHART #11

This is the logic symbol for the F9452-- the major element of the BPR. the signals on the left side of the symbol are basically those signals that interface to the CPU and the MMU if present. The signals in the upper right interface to and control the local RAM that contain the protection bits. In the case of a memory write protect violation, the memory protect error output will signal the CPU to report the fault and the memory system to inhibit the write. Global protect enable is an output that tells the rest of the system that no memory location can be written into since the BPR will not have been initialized. After the BPR has been initialized by the CPU, and the bits in the local RAM have been loaded, global protect enable will be deactivated. The last set of pins are the ABORT input coming from the CPU; DMA acknowledge, since the BPR must distinguish between memory accesses by the CPU and DMA; the address valid input coming from the MMU; and two speed pins that do a job similar to the speed pins in the MMU, letting the BPR know how many wait states to insert depending upon the speed of the CPU clock.

CHART #12

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

This is a system configuration consisting of a CPU and an MMU. The MMU consists of all the logic contained within the broken line.

CHART #13

This is a system configuration consisting of a CPU and a BPR. Note that if an MMU is not present, four bits of the extended address must be tied to ground and four bits must come from the information bus.

CHART #14

This is a system that contains all the elements of the chip set: the CPU, the MMU, and the BPR. Note that the memory protect error signals are tied in a WIRE-OR so that either the MMU or the BPR can report an error.

CHART #15

This chart shows a few features of I3L-II technology. I3L is a bipolar technology. As opposed to TTL which is based upon a multiple-input, single-output transistor, I3L transistors are single-input, multiple-output. Minimum gate delays are 2.5 nanoseconds or less, with typical gate delays of 4 nanoseconds for the four collector configuration at 100 microamps of injector current per cell. Packing density is 500 to 600 gates/mm² including power and interconnect. The intrinsic density-- active devices only-- is about double. The technology is directly interfaceable to TTL. This feature is used on the F9450 since part of the chip is TTL. As mentioned the technology is fast and dense, and it makes implementation of standard cells such as PLA's and ROM's relatively easy. Being bipolar, the technology allows operation over the full military temperature range. Also the technology is scalable for improved performance and density.

CHART #16

The indicated subjects will be covered in the following discussion of the CPU.

CHART #17

The data types are as defined in MIL-STD-1750A.

CHART #18

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

The addressing modes are as defined in MIL-STD-1750A.

CHART #19

This is an internal block diagram of the CPU. There are five major blocks: (1) the Address Processor, (2) the Data Processor, (3) the Interrupt Processor, (4) the Microprogrammed Control Section, and (5) the Timing and Bus Arbitration Unit. The Address Processor is responsible for generating all of the addresses that go out from the CPU during bus cycles, either instruction addresses or operand addresses. The two basic registers included in the Address Processor are the IC, or Program Counter, and the MAR, or Memory Address Register. An incrementer/decrementer implied within the block allows the next operand address or instruction address to be prepared while executing some other operation in the Data Processor. The Data Processor block is the part of the chip that does the data processing. The main part of the block is the ALU. A 17-bit ALU was required because of the multiply algorithm used. The ALU is fast with carry look-ahead. The shifter shifts left and right and is also capable of byte-shifting, sign-extend, and things of that nature. The register file which is contained within the Data Processor block consists of the 16 general purpose registers that are user accessible, plus another seven scratchpad registers, some of which are also shift-registers. The Data processor also includes the status register and the two timers as defined in MIL-STD-1750A.

I mentioned two timers are working on separate clocks. These are real time timers and obviously they shouldn't rely on the main CPU clock. On the other hand to be able to read the counter while counting, the two clocks have to be synchronized. On chip we synchronize the timer clock with the main CPU clock. We can do a clean read of the timer while the timer is still counting. So, while the timer is independent as far as counting the time from the CPU clock, it is still synchronized to allow a read while counting. We put a lot of effort in trying to multiply as fast as possible, and what we use is a modified Booth algorithm which is doing two shifts past that and cutting in half the time that a normal add/shift algorithm produces. To implement the algorithm we needed another shifter. If you are familiar with the Booth algorithm, you have to add the multiplicand shifted by two, and some extra logic to cover the subtle cases of overflow that result in this type of operation. As you can see the data processor besides getting two operands and producing results is also generating the different ALU conditions necessary for the control section to do branches.

The next block is the interrupt processor, and this block receives the interrupt and faults, and is responsible for processing these. To do that it contains three major subblocks and these are the three registers: the PIR register, which records all the interrupts coming in whether from the inside the

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

chip or outside the chip. The Mask register which is deciding which of the interrupt are masked and which is not. The fault register which receives the external and internal faults. Some of the external faults are synchronized, like the memory protect error and some of them are non-synchronized. So some of them had to be implemented as an edge sensitive fault. Also, included in that block is the priority logic that looks at the PIR and the mask. In case of several interrupts happening at the same time, it determines which one is going to get serviced according to the priority that I will present in a few minutes. Also included in that block is the enable logic controlled by executing an XID instruction that says interrupts enable or disable. The other block is the microprogram control section. It's a two level pipeline microprogram section generating the control bits for the rest of the processor, and it also includes the source and destination logic. As you know the two operands selected from the data processor could come either from the instruction body (the way they're defined in the instruction format) or from the microprogram. If you want to work between two scratchpad registers obviously it would not come from the instruction format. The logic that looks at that part of the instruction register and selects where the source for the two operands being worked on in this particular cycle is in that block. The last block is the timing and bus arbitration unit. This is the block that gets the CPU clock, which is up to 20MHz, generating all the internal clocks and strobes necessary to activate any of these blocks and it is also responsible for the handshaking with the bus. Two things which are obvious to the user are the STRBA and the STRBD which come out of that section. STRBA is covering the time that the address is on the bus and STRBD covers the time the data is on the bus. The bus is the multiplex bus. Part of the status register is those eight bits of address state and address key that go into the MMU.

CHART #24

This is the F9450 register set

CHART #27

In the system there are three types of timing cycles: a three state cycle for a pure ALU cycle. The four state cycle is the minimum memory bus cycle to I/O and memory, if no wait states are inserted at 20 Mhz it needs 200 nanoseconds. Wait states are used

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

as required for slower memory to access data. The five state cycle means that conditions during the cycle require the next microaddress to be contingent upon conditions in the machine. This cycle is also used in the abort cases. So there are three, four and five state cycles which at 20 Mhz mean 150, 200 and 250 nanoseconds.

CHART #28

This is what happens during the minimum read bus cycle. The CPU clock is shown along with the internal states S0, S1, S2 and so forth. Bus request is the first thing that is generated when you want to start a bus cycle. If the bus grant is granted at the same time then the CPU has control over the bus and begins it's bus cycle. Once it gets the bus it pulls down the bus busy line to signal that this is when the cycle starts. Once the CPU gets the bus it will output the three status lines: mem/io, read/write, data/instruction. As long as the CPU does not have the bus these lines and most others are tri-stated. Bus lock is used if the CPU does not need to lock the bus then it follows exactly the bus busy. You lock the bus when you want to for example, read and then write to a certain location. STRBA indicates when the bus holds an address, and the trailing edge is used to latch the address into a 373 type device. RDYA is pulled low when you want to insert wait states, in this case none are inserted since this is a minimum cycle. RDYD acts the same as RDYA to insert wait states when the data is not ready. the bus has two phases: address phase and then data phase. The times shown are preliminary, since chips have not been fabricated yet.

CHART #29

Similar to the previous slide is the minimum write cycle, the only difference is the extended time when the data is on the bus which is used to accomodate the slower CMOS memories that are available. To insert wait states onto the data phase of the bus cycle you pull this low and another wait state is inserted.

CHART #30

The same thing occurs during the minimum write bus cycle, similar to the previous slide. To extend the address the RDYA signal must be used and an S1 state is inserted.

CHART #31

This simplifies explanation of the RDYD signal in a bus cycle.

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #32

The RDYA signal polarity and setup times are shown.

CHART #33

This shows what happens during a bus arbitration cycle. The CPU pulls the bus request low telling the arbiter it needs the bus. If the bus arbiter grants the bus to the CPU it pulls the bus grant low. As long as the CPU does not have the bus it is tri-stated. The arbitration is done on a CPU clock cycle basis.

CHART #34

Some multi-processor configurations are shown. First the CPU and a DMA device are on the bus with the DMA device having the highest priority. You need a flip-flop and a gate that receives the DMA Enable from the CPU. The bus grant is synchronized with the CPU clock.

CHART #35

With two CPUs the scheme is very similar to the previous slide. We arbitrarily assign one of the CPUs a higher priority. It issues the request and receives the grant, and the other one gets the grant bar. It is again synchronized to the CPU clock.

CHART #36

With up to eight bus masters, we have a priority encoder. It is a direct extension of the two CPU scheme.

CHART #37

There are six fault inputs to the CPU, three are synchronous and are latched into the fault register on the trailing edge of bus busy. The other three are asynchronous and edge sensitive and labeled SF0, SF1, and SF2.

CHART #38

The fault register is a sixteen bit register defined by the

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

standard. Any bit set in the fault register causes the level 1 interrupt to occur.

CHART #39

There are two sub-classifications of error conditions that are called major error and unrecoverable error. Unrecoverable errors have to do with instructions, and when one occurs we can not continue. It causes an abort and forces an interrupt level 0 to occur. Major errors will cause an abort also, but these do not cause a level 0 interrupt.

CHART #41

This is the priority scheme organization with a linkage and service pointer for each interrupt.

CHART #42

The console operations are built in on chip. The console request input signals when we want to do a console operation. By reading an I/O address the particular console operation is decoded. The console represents to the CPU three I/O addresses: one command address and two data addresses.

S. MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #47

The self test is part of the initialization routine that is invoked by the reset pin. The self test reads and writes to the registers, verifies the various ALU and pre-ALU functions, checks the booth hardware, shifter, the ROM constants, IC and MAR are checked. Then a synthetic instruction is generated and executed to test the IR and associated logic. If an error is encountered bit 13 in the fault register is set and the normal power up will not be set.

CHART #48

This is the initialization routine that occurs on power-up.

CHART #49

This is the rest of the initialization routine, part of which is the self-test. After that we read the configuration register which tells us the chip set configuration for initialization purposes.

CHART #50

Performance is calculated by a program written by Jack Herlin. The total throughput is 683 KOPS for the DAIS avionics mix. Using only fixed point we get 1.5 MIPS, floating point only is 180 KOPS. If an MMU is attached we get about 600 KOPS.

SHAI MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #51

What I want to describe now is the way we plan to implement the built-in functions. To refresh the memory of people who might have forgotten, the 1750A standard definition of a built-in function opcode basically serves as an escape code. If we allow user defined types of operations, for example, if someone wants to implement the sine, cosine or any other transcendental functions, this is what he's going to use. Now the way we propose to do it is using the 9450 together with another chip that is in an advanced stage of development at Fairchild, the 9443, which is an IEEE floating point standard coprocessor. To accommodate the 9450, the 9443 has now two modes of operation: IEEE mode and 1750 mode. I will go over very briefly how the 43 works. The 43 is basically a slave processor. It's not a real coprocessor looking at the IB and the instruction stream and intercepting its own instructions. The way it works is as a slave. It has to receive from the CPU, the command and the starting address for its operand, if it happens to be a memory type operation. So 9443 is what we call a coprocessor or slave processor. We are presenting to the CPU four I/O addresses, and it is treated by the CPU as four I/O devices. One I/O address presents the starting operand address. The CPU is calculating the starting address of the operand for the particular floating point operation we want to execute; for example, sine or cosine. If that particular operation does not involve a memory operation then this is not calculated by the CPU, and obviously not sent out to the coprocessor. Then you have a control register which presents another I/O address to the CPU, and it specifies things like rounding mode. If you are familiar with IEEE floating point, you know there is a lot to specify. What is important in our case is that it tells the 1750 and 9443 whether its going to be IEEE mode or 1750 floating point mode; so that's the control register. Then the instruction register presents another I/O address to the CPU which is where the instruction originates, and it also tells the 43 what operation to do; sine, cosine or whatever. The status register is what the CPU is reading from. The coprocessor and for example, faults are registered in the status register. The 43 has eight internal registers in the single precision or four extended precision. All operations for the 43 either register to memory or register to register. There is no memory to memory in the 43. If its a memory operation, then the 43 gets the starting address from the CPU and then goes ahead and fetches its operand from memory. In other words, it becomes another bus contender. It has to request for the bus like any other CPU or DMA device or whoever is hung on the bus. Once it gets its starting address, its control register data and instruction register, the 43 goes ahead and executes independently the instruction. When the execution is completed, the two lines from the 43 signal back to the CPU either completion or fault. These two can be interrogated by I/O polling or the user can generate an interrupt using them. The 43 maybe further interrogated by executing I/O instructions. For example, if one of the BIF instructions that was sent to the coprocessor was illegal, it will be reported in the status register.

SHAI MOR
TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #52

This is an illustration of what the 9443 represents to the CPU. Actually the 43 has 32 bit addressing capabilities but in our case we don't use it, we use only the AL register. Shown here are the control register, instruction register, and status register.

CHART #53

This is the way the BIF instruction looks in our 1750 implementation. As you can see its a three word instruction. And if you ignore the last word, it looks like a normal direct memory or indexed memory instruction. The 4F is defined in the MIL-STD, and considering the way the 9443 works, only the indicated addressing modes make sense. Since the 9443 is expecting addresses there is no point in using immediate type of addressing. Bit 8 specifies indexed or non-indexed operation, and the register is specified in Bits 12 through 15. This is the same way the specification is done in the normal 1750 instructions. Actually the same DODA routines are used in the microcode. IR is the instruction that we want to execute on the coprocessor. The instruction is read into the CPU and sent back to the coprocessor; thus there is a three word instruction format. Another feature that we have added here is the ability to talk to four coprocessors with one 1750 instruction. The two bits 10 and 11 specify which coprocessor the instruction refers to.

CHART #54

These are the configurations in which the 9443 will work with 9450. The first configuration is without an MMU. As you can see the 9443 is one of the bus contenders in the system, and it has to be arbitrated just like any other bus contender. In the case of an MMU system the 9443 has to supply its own AS&AK to be able to operate through the MMU. These are not included as part of the 9443; therefore, we need an external device that is addressed as an I/O device by the 9450. Other aspects of the operation are basically the same.

CHART #55

This is the sequence of events that occur when a BIF instruction is occurring. The CPU is calculating the effective address according to the normal 1750 procedure. If the coprocessor is not present than an illegal operation is signaled and the operation is terminated. If the coprocessor is present, then the CPU will use the three I/O addresses to send to the 9443 the starting address, and two other addresses: the instruction that you want to execute and the control register. In the case of 1750 the major thing the control register will tell the 9443 is to go to the 1750 mode, rather than the IEEE mode. The note explains once again that the control register and the coprocessor AS/AK latch are loaded separately using

SHAI MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #55 (Cont'd)

XIO instructions. There was no point making these operations part of the built-in function instruction format because it would make a four word format and the operations are seldom used. The 9443 is a very high performance device with floating point operations on the order of two or three microseconds for a multiply or divide in single precision.

CHART #56

The last two sections of the presentation have to do with the MMU and the BPR. I will go over the functions, how the functions were implemented, some word about timing, the pinout, and system configuration.

CHART #57

The function of the MMU is to translate logical addresses to physical addresses and extend the addressing space to one million words. There are two address maps, one for the instruction space and another for the data space. Each is 256 x 16 registers. Another feature of the MMU is memory protection. There are three modes of protection; the first is access key protect in which the AK coming from the CPU is matched against the lock four bits in the appropriate register and if a mismatch occurs a memory protect error is generated. If we are in the data mode there is one bit that tells us whether we are allowed to write into the associated 4K block of memory. The same bit is used in the instruction fetch mode to tell us whether we are allowed to fetch from that 4K block.

CHART #58

This is a familiar slide showing AS&AK coming from the CPU. The AS is selecting a page group. The four most significant bits of the IB select one register from the page group and the PPA coming out is concatenated with the twelve least significant bits from the IB to generate the twenty bits of physical address.

CHART #59

The MMU was implemented using what looks like a one register cache. The assumption is, using the principle of locality of accessing, that if we accessed a 4K block it would be more than reasonable to assume that the next access to memory would be from the same 4K block. So we have on chip two register caches; one for the data section and one for the instruction section. If we get an address coming out of the CPU that is identical to the previous address contained in the latch, then we go ahead to the data register and use it. If we get a miss, we access the external RAM maps. Statistically, we will get hits probably 90% of the time.

SHAI MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #60

This chart indicates the MMU timing. If we ignore the time it takes to execute an XIO instruction to and from a register in the map, we look at this. If we get a hit then we incur two additional wait states. If we get a miss then the MMU is inserting four additional wait states because it has to access the map, bring back the access lock and the protection bits in the PPA. So statiscally there should be something like 2.2 wait states. That is what we use in our performance calculations.

CHART #63

This slide illustrates that the CPU/MMU combination. Let me emphasize that when we talk about MMU we are talking about seven chips. The main chip is in a 64 pin package which is called the 9451. There are six external chips. Four of the chips are commercially available RAM chips which contain the two maps, instruction and data. Two are bi-directional buffers which are also commercially available products. The MMU also outputs the three reserve bits which currently are not used for anything.

CHART #64

The final section addresses the BPR: functions, implementation, timing, pinout and system configuration.

SHAI MOR

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #65

The BPR functions are to provide write protection in the physical address space for the CPU and the DMA. The protection is done in 1K pages as opposed to the 4K pages as done in the MMU. The BPR also provides global write protection, in other words, nothing can be written. This is done when the system is powered up because before the RAM is initialized and the protection bits are set, everything must be globally protected. After the BPR is initialized then global protection goes away and protection is accomplished according to the contents of the RAM.

CHART #66

This chart shows what the RAM looks like. In the case where we have an MMU we have two sets of 64 registers; one set for the CPU and one for the DMA. Each bit in the register represents 1K of physical memory. So if a particular bit is set to one, it means that this 1K is write protected. In the case where we do not have an MMU then the addressing space is significantly smaller and it contains actually two sets of four registers each, and again each bit is representing 1K of physical memory.

CHART #67

In the BPR implementation we used a similar concept to what we did in the MMU and it actually applies even more here. Because here if we read in one register, we have protection for 16K so the likelihood of getting a page boundary crossing is even smaller. Again we have a comparator and if the current address matches the previous one, then we don't have to go to the external RAM. If we get a miss then, similar to the MMU, we go ahead and read the appropriate 16 bit register from the RAM and do the protection only then. Obviously this will take longer. To minimize the number of external chips and considering that in the time that we do a BPR XIO, mainly loading a register in the BPR, reading one from it is not critical at all. What we chose to do is use only one RAM chip and doing the loading and reading in two bytes of eight bits. Therefore, we don't need two external RAMS; we can only use one.

CHART #68

This diagram shows how the RAM is connected to the BPR. All the control is coming from the BPR.

CHART #69

This is the impact of the BPR on timing. The critical figure that we have to look at is this one here. As you can see its again two wait states inserted by the BPR if we get a miss. The BPR and MMU are working at the same time, so only if we get a miss will the wait states be serial. If we get a hit, the wait states will be in parallel. So there are no extra wait states in 90% of the cases when

SHAI MOR
TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #69 (Cont'd)

we connect the BPR and MMU at the same time. The rest of the cases are not critical. Again we have the two pins into the BPR that control the speed; namely determining how many wait states are going to be added by the BPR, and it depends on the CPU clock.

CHART #72

This slide has already been shown. Shown are the interface of the CPU, the RAM controls, the memory protect error that goes back to start the fault sequence if we get a protect error. This is the global protect enable which is a separate pin that disables any writing to the memory before the BPR is initialized.

CHART #73

This is a system consisting of the CPU, MMU and BPR. There is nothing new to say in addition to what I've already said except that as you can see the memory protect error is a tie of the signals from the BPR in the MMU, and if either device is signaling a memory protect error the fault register will be loaded and the fault sequence will start.

1750A CHIP SET

- * 1750A CHIPSET CHARACTERISTICS
- * 1750A CHIPSET CONFIGURATIONS
- * F9450 - 1750A CPU - TECHNICAL DESCRIPTION
- * F9451 - 1750A MMU - TECHNICAL DESCRIPTION
- * F9452 - 1750A BPR - TECHNICAL DESCRIPTION

S. Mor
Third Technical Forum
5 May 1982

Chart # 1

1750A CHIPSET

F9450: 1750A CPU

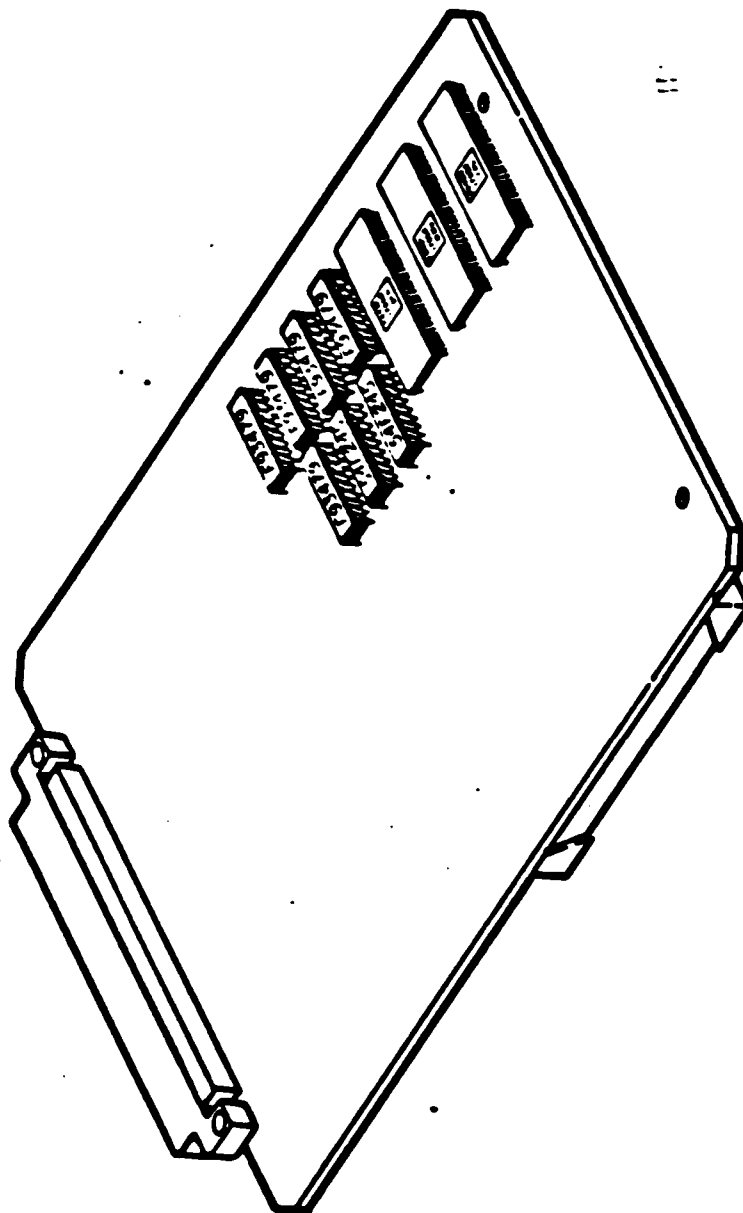
F9451: 1750A MEMORY MANAGEMENT UNIT (MMU)

F9452: 1750A BLOCK PROTECT RAM (BPR)

S. Mor
Third Technical um
5 May 1982

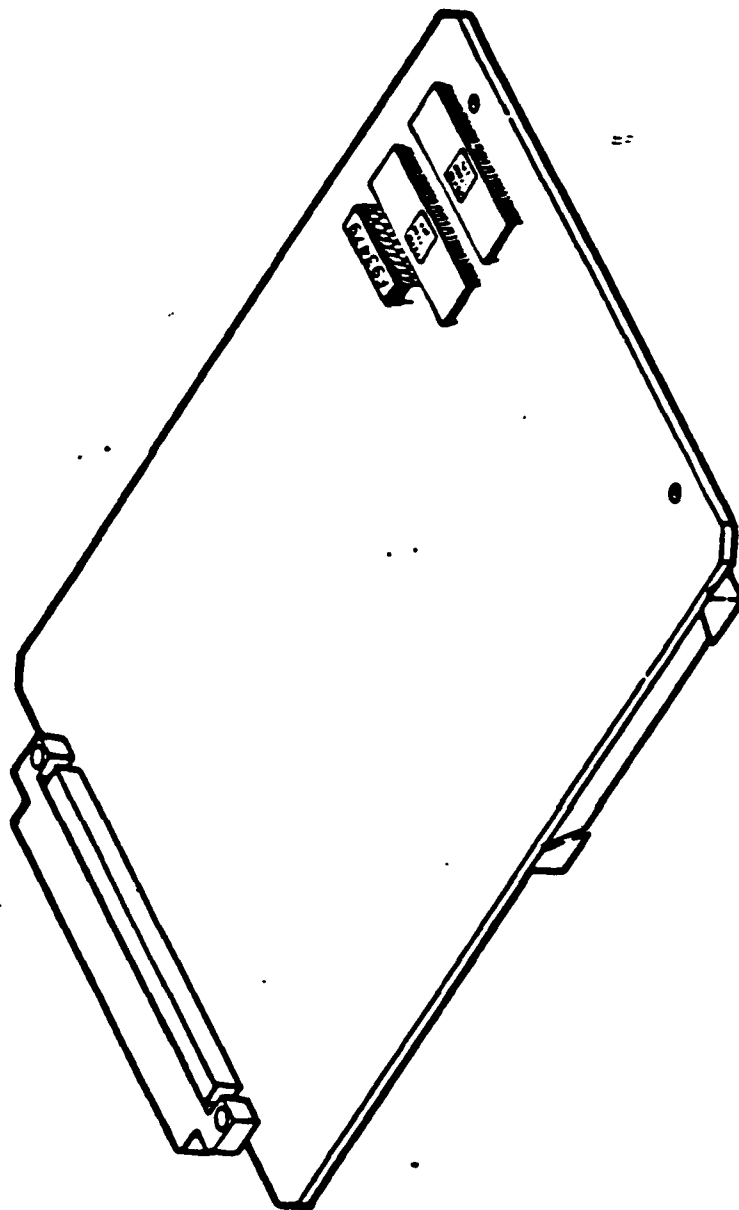
Chart # 2

1750A MICROPROCESSOR CHIP-SET



S. Mor
Third Technical Forum
5 May 1982

1750A MICROPROCESSOR CHIP-SET



S. Mor
Third Technical Forum
5 May 1982

Chart # 4

F9450 (1750A CPU)

- SINGLE 64-PIN MICROPROCESSOR. IMPLEMENTS MIL-STD 1750A INSTRUCTION SET ARCHITECTURE.
- SINGLE 5V SUPPLY (ADDITIONAL INJECTOR CURRENT SOURCE REQUIRED) - POWER DISSIPATION LESS THAN 3 WATTS
- 1³L-II (3 MICRON) TECHNOLOGY WITH 10**5 RADIATION TOLERANCE.
- STATIC OPERATION WITH CLOCK FREQ. 0-20 MHZ.
- LOWER POWER SCHOTTKY INPUT/OUTPUT.
- MULTIPROCESSING CAPABILITIES.

S. Mor
Third Technical Forum
5 May 1982

Chart # 5

FEATURES (CONTINUED)

- HIGH-PERFORMANCE OVER MILITARY TEMPERATURE RANGE (200 ns ADD; 1.85 μ s 16 x 16 MULTIPLY).
- SINGLE AND DOUBLE PRECISION ARITHMETIC (16 AND 32 BITS).
- 16 GENERAL PURPOSE REGISTERS.
- 32 AND 48-BIT FLOATING POINT ARITHMETIC IMPLEMENTED ON CHIP.
- REAL-TIME PROCESSING WITH 16 LEVELS OF INTERRUPT VECTORS.
- DIRECTLY ADDRESS 64K WORDS, EXTENDABLE TO 1M WORDS WITH MMU.
- TWO PROGRAMMABLE TIMERS.

S. Mor
Third Technical Form
5 May 1982

Chart # 6

BPR FEATURES

- PROVIDES WRITE PROTECTION IN PHYSICAL MEMORY FOR CPU AND DMA IN PAGES OF 1K WORDS
- UTILIZES:
 - F9480 GATE ARRAY CHIP
 - 93479 RAM CHIP

S. Mor
Third Technical Forum
5 May 1982

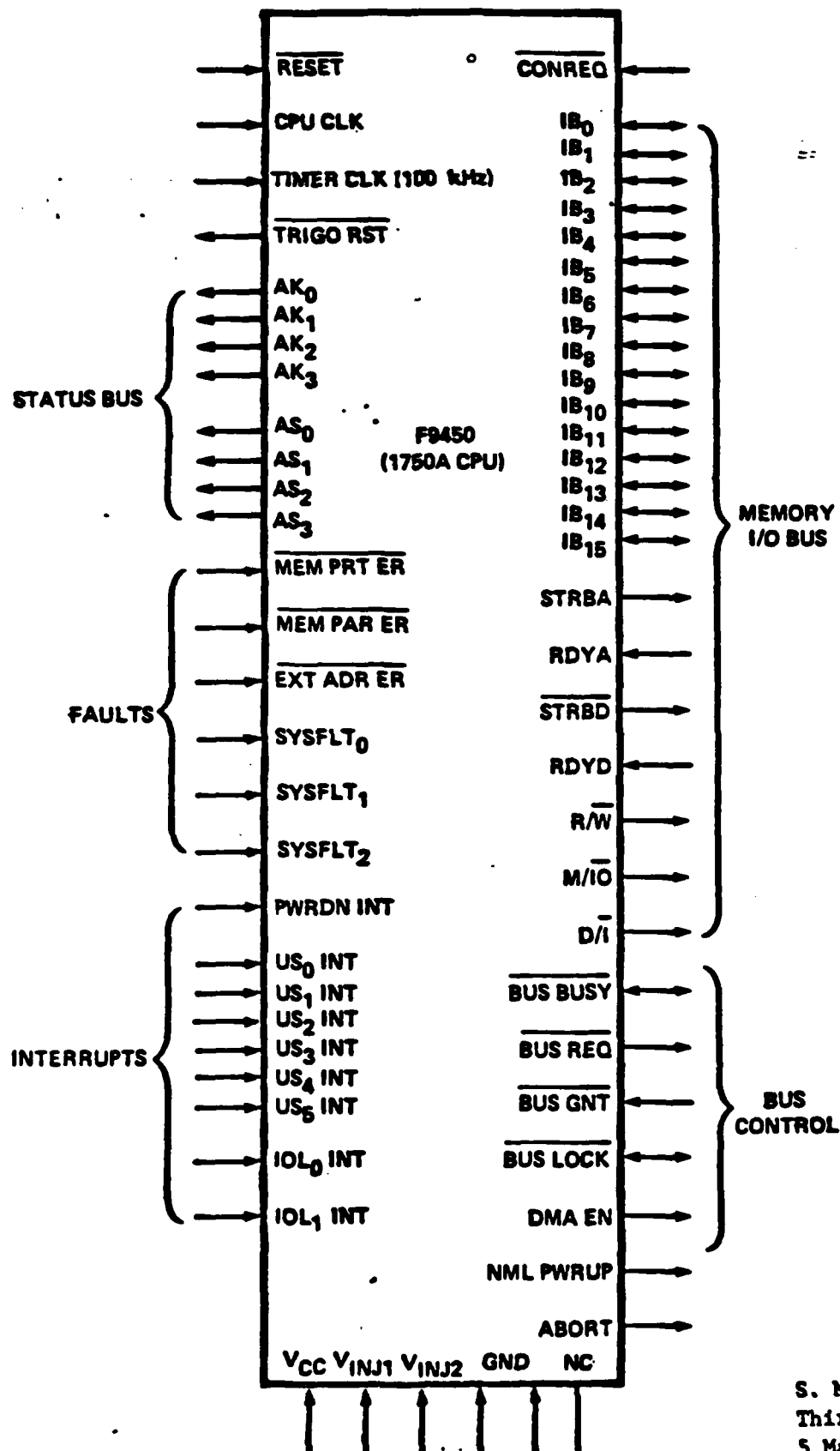
Chart # 7

MMU FEATURES

- PROVIDES LOGICAL TO PHYSICAL ADDRESS TRANSLATION FOR INSTRUCTIONS AND OPERANDS •
- 1M WORD ADDRESSING SPACE
- PROTECTION IN LOGICAL SPACE OF 4K WORD PAGES FOR:
 - ACCESS KEY
 - WRITE
 - EXECUTE
- UTILIZES:
 - F9480 GATE ARRAY CHIP
 - 4 93479 RAM CHIPS
 - 2 54F245 BIDIRECTIONAL BUFFERS

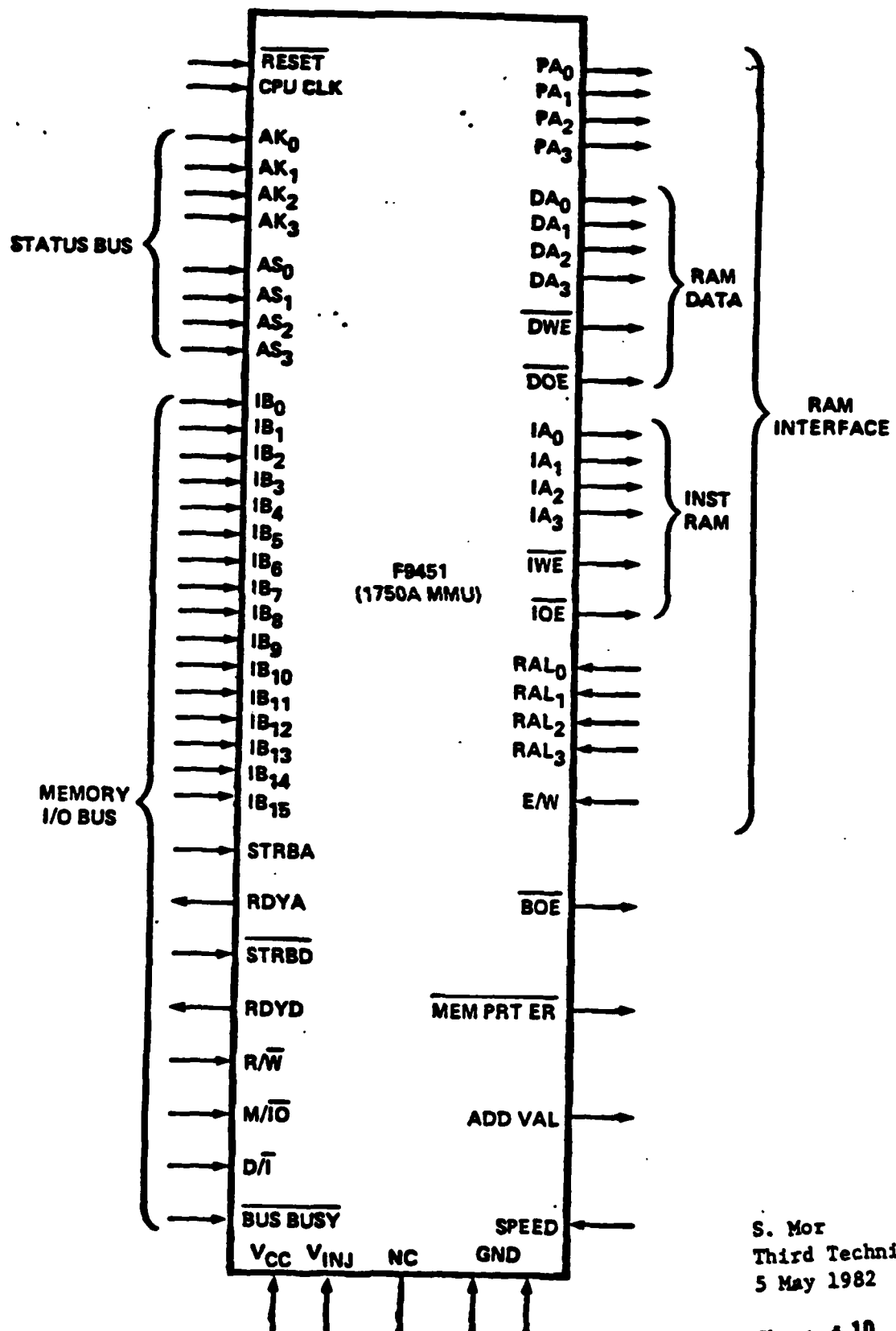
S. Mor
Third Technical Fo.
5 May 1982

Chart # 8



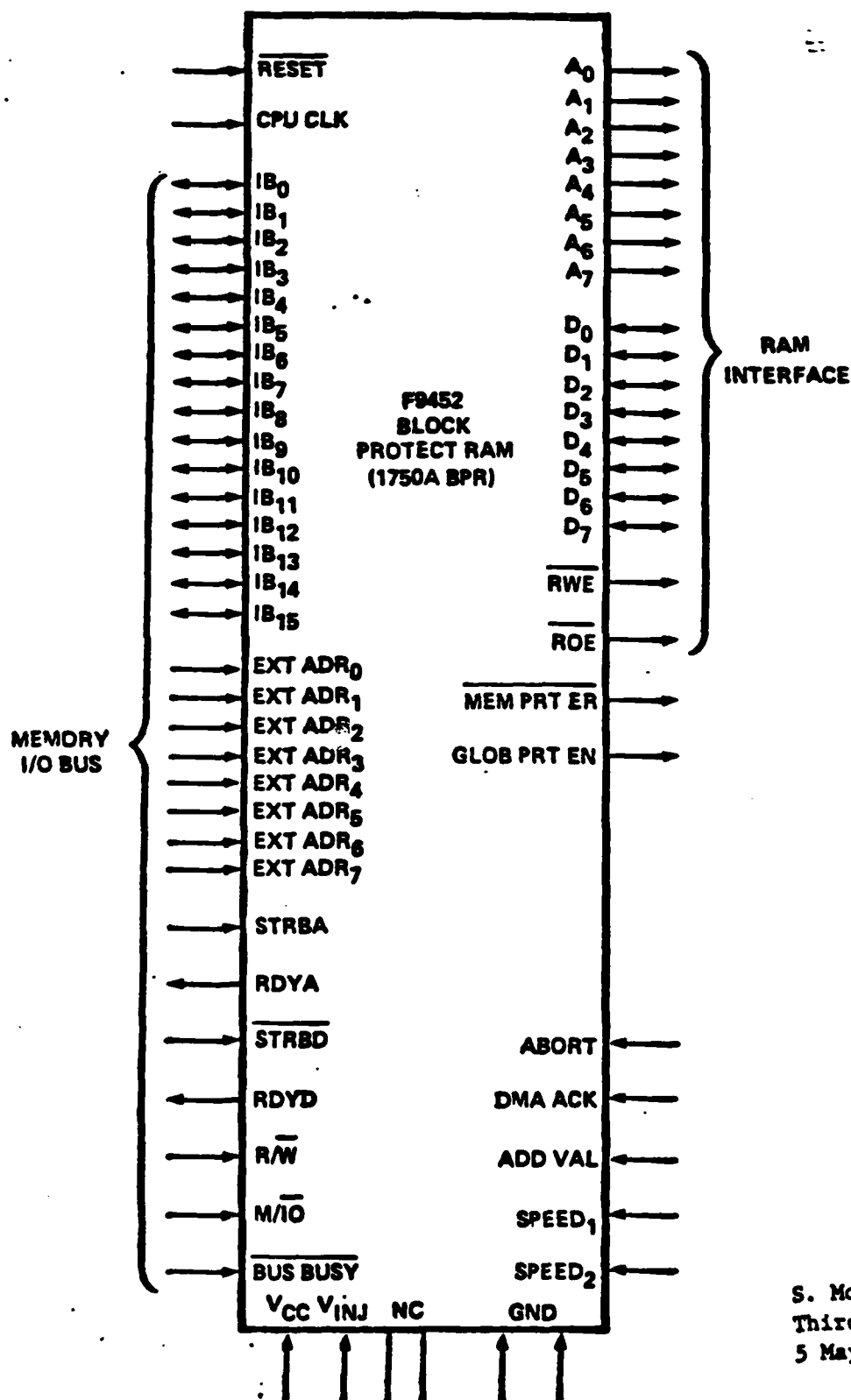
S. Mor
Third Technical Forum
5 May 1982

Chart # 9



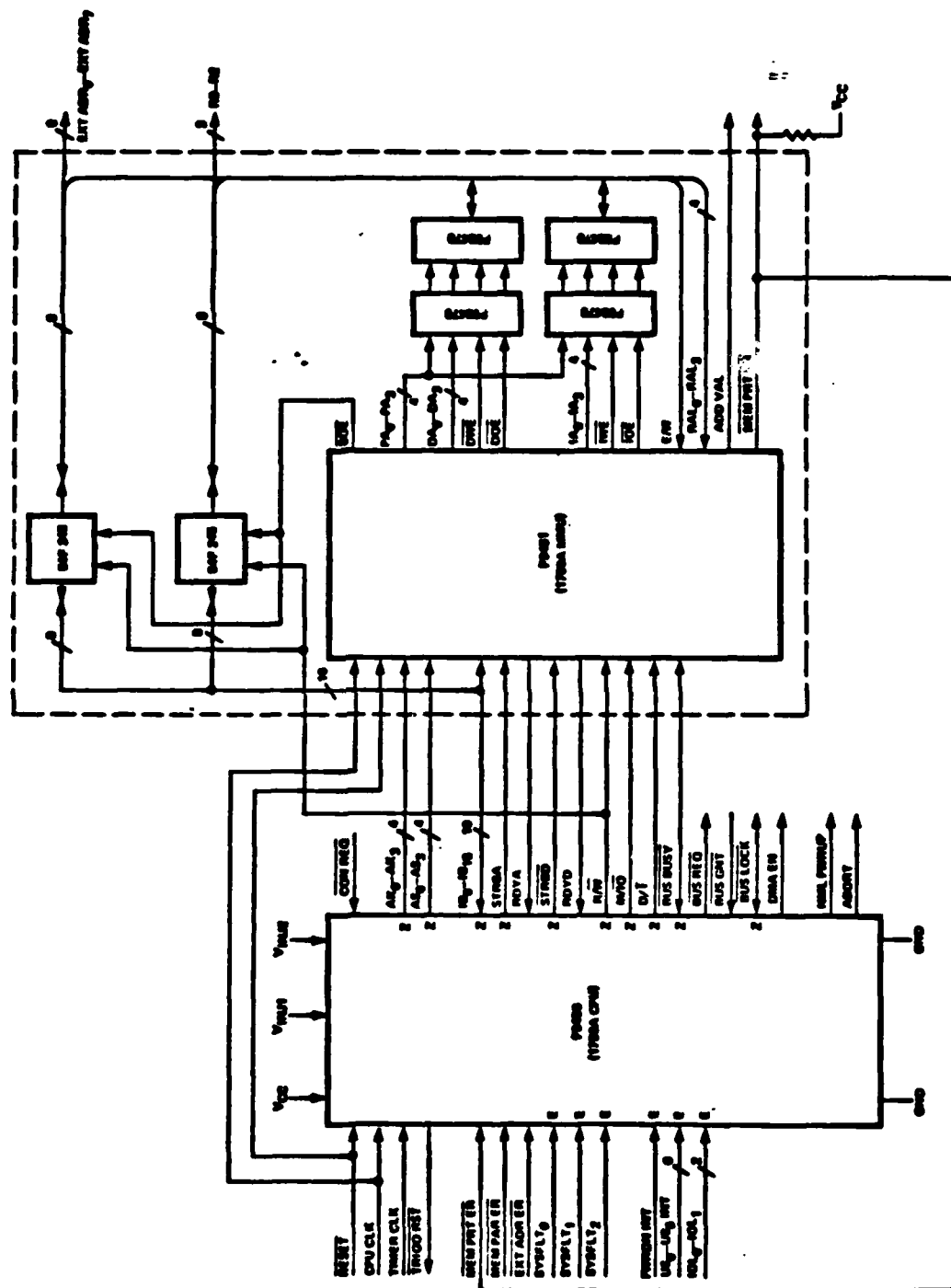
S. Mor
Third Technical Forum
5 May 1982

Chart # 10



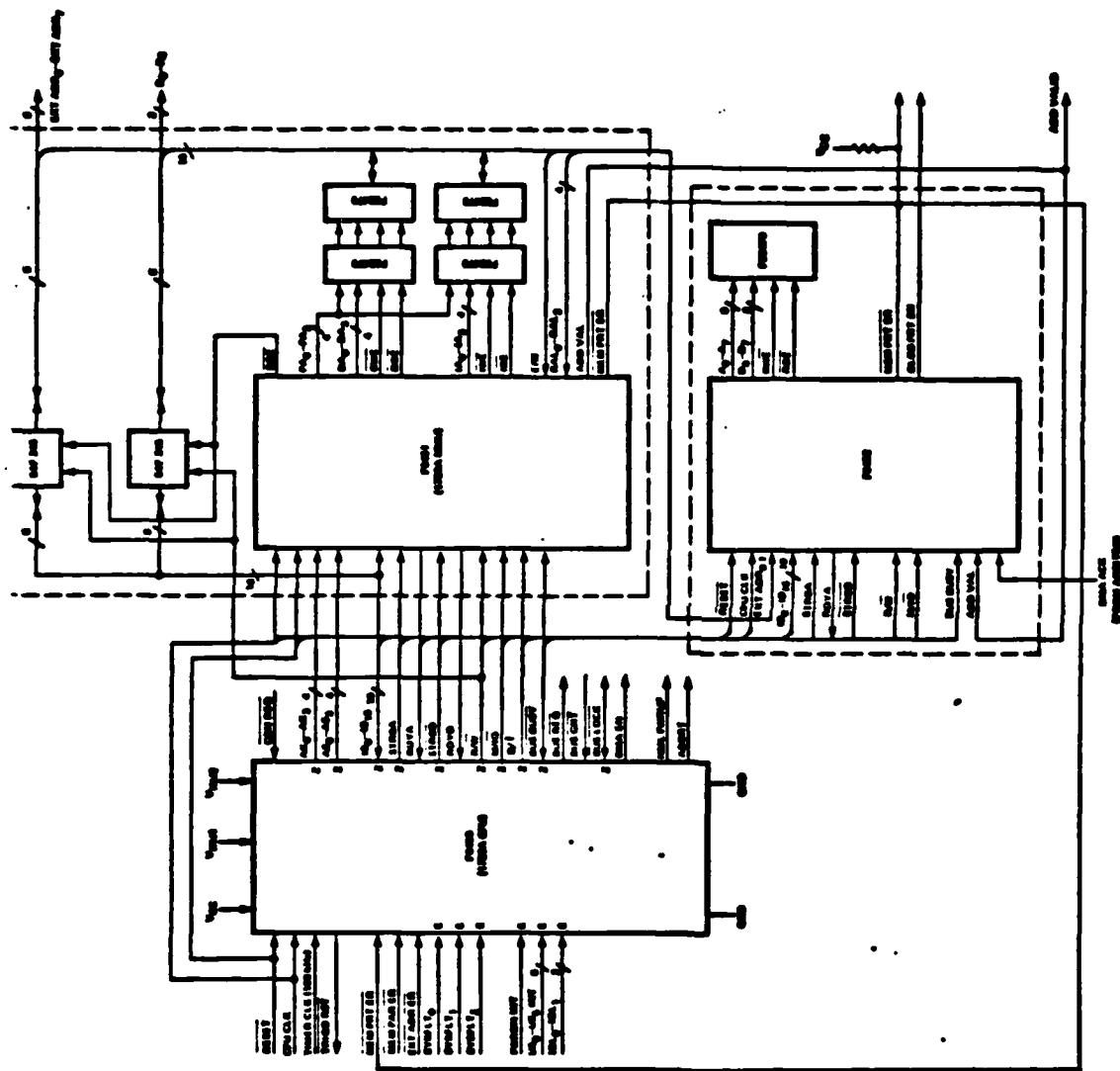
S. Mor
Third Technical Forum
5 May 1982

Chart # 11



S. Mor
Third Technical Forum
5 May 1982

Chart # 12



S. Mor
 Third Technical For...
 5 May 1982
 Chart # 14

I³L II TECHNOLOGY FEATURES

- MINIMUM GATE DELAY OF 2.5NS AND BETTER
- AVG. GATE DELAY 4NS (MIN. SIZE F.O=4) AT 100 μ A CELL CURRENT
- PACKING DENSITY 500-600 GATES/MM² INCLUDING POWER AND INTERCONNECTS AND 1200 GATES 1mm² INTRINSIC DENSITY
- DIRECT TTL INTERFACE CAPABILITY
- FAST AND DENSE CONVENTIONAL CIRCUITRY I.E. PLA's, ROM, RAMS ON SAME CHIP
- MIL. GRADE TEMP. OPERATION
- SCALABLE FOR IMPROVED PERFORMANCE AND DENSITY.

S. Mor
Third Technical Forum
5 May 1982

Chart # 15

F9450 - 1750A CPU

- * DATA TYPES
- * ADDRESSING MODES
- * BLOCK DIAGRAM
- * REGISTER MODEL
- * PIN-OUT
- * TIMING DIAGRAMS
- * MULTIPROCESSING (BUS ARBITRATION)
- * INTERRUPTS AND FAULTS HANDLING
- * CONSOLE OPERATIONS
- * INITIALIZATION AND SELF TEST
- * PERFORMANCE
- * BIF IMPLEMENTATION

S. Mor
Third Technical Forum
5 May 1982

Chart # 16

DATA TYPES

- 1) BITS
- 2) BYTES (8 BITS)
- 3) WORD (16 BITS)
- 4) DOUBLE WORDS (32 BITS)
- 5) SINGLE PRECISION FLOATING POINT (32 BITS)
- 6) EXTENDED PRECISION FLOATING POINT (48 BITS)

S. Mor
Third Technical Forum
5 May 1982

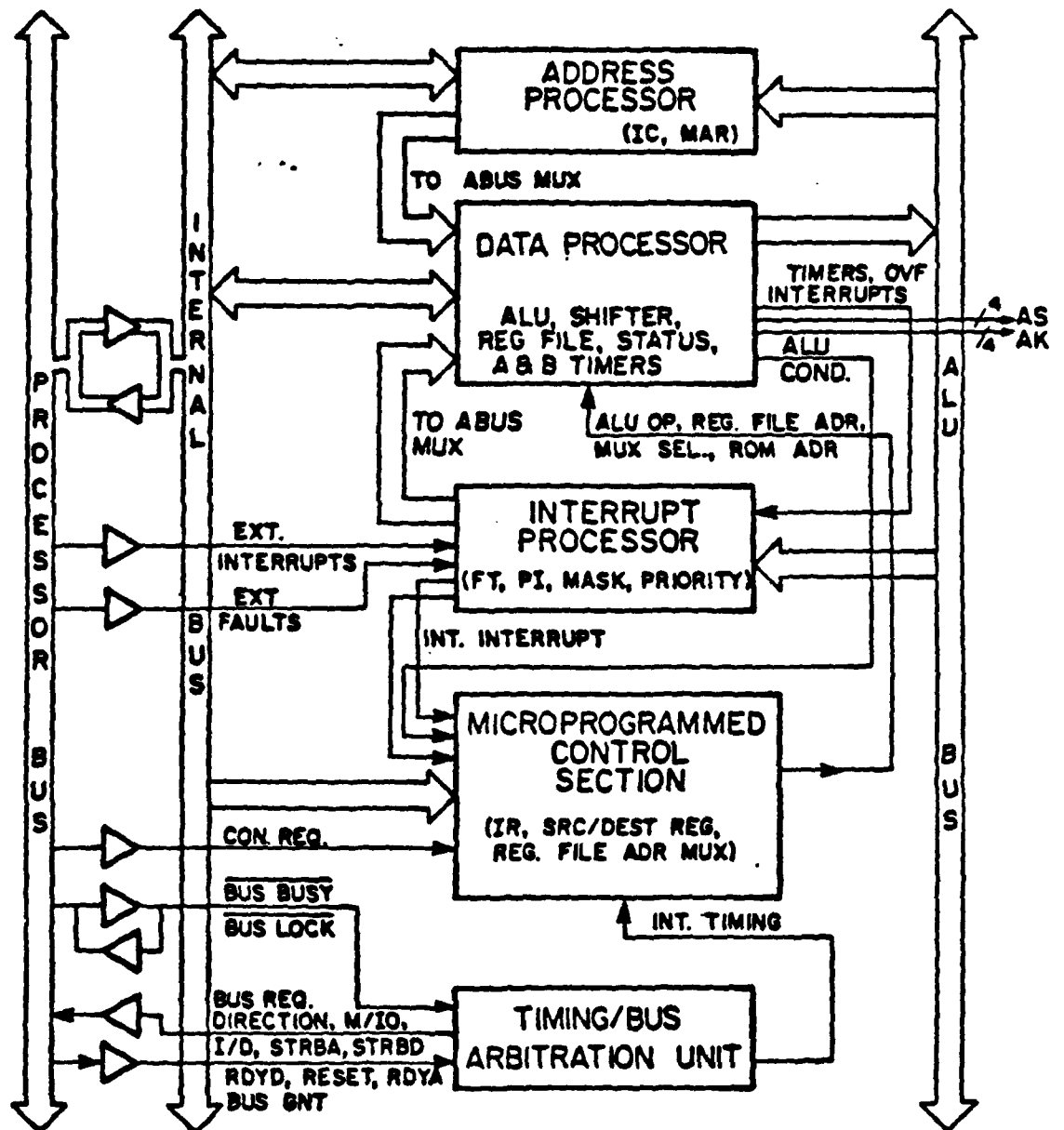
ADDRESSING MODES

- REGISTER DIRECT
- MEMORY DIRECT
- MEMORY DIRECT INDEXED
- MEMORY INDIRECT
- MEMORY INDIRECT WITH PREINDEXING
- IMMEDIATE LONG
- IMMEDIATE SHORT
- IMMEDIATE SHORT POSITIVE
- IMMEDIATE SHORT NEGATIVE
- IC RELATIVE
- BASE RELATIVE
- BASE RELATIVE INDEXED

S. Mor
Third Technical For
5 May 1982

Chart # 18

F9450 BLOCK DIAGRAM
5 MAIN BLOCKS W/INTERFACE SIGNALS



S. Mor
 Third Technical Forum
 5 May 1982

Chart # 19

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15

TIMER A
TIMER B

PENDING INTERRUPT REGISTER (PIR)
MASK REGISTER (MR)

FAULT REGISTER (FR)

INSTRUCTION COUNTER (IC)

STATUS WORD (SW)

SYSTEM CONFIGURATION REGISTER (SCR)

F9450 REGISTER SET

S. Mor
Third Technical Forum
5 May 1982

Chart # 24

TIMING

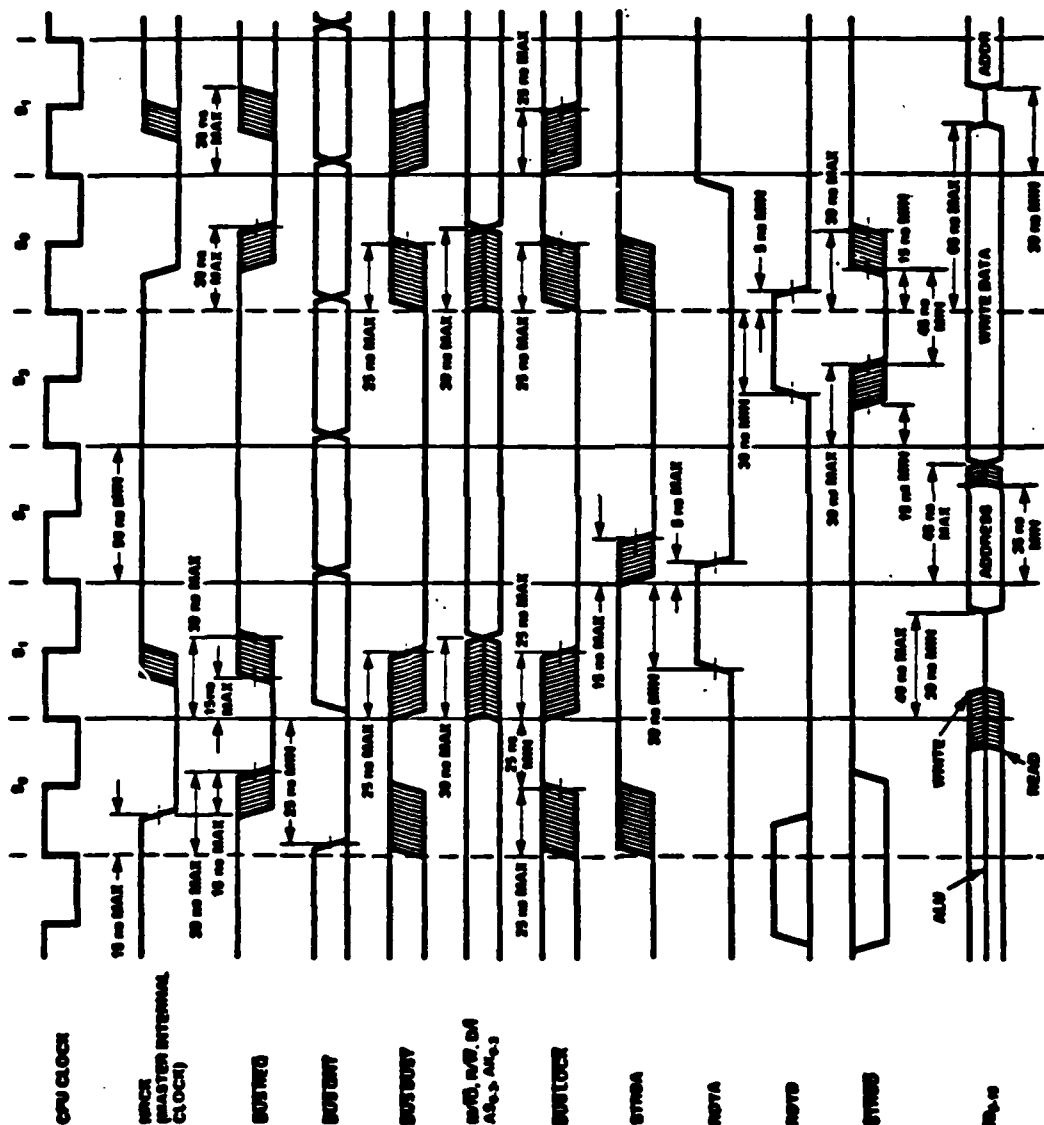
The Timing Unit generates three types of nano/micro cycles according to the following:

- * a) A 3-state* cycle for pure ALU nano/micro control words.
- b) A 4-state cycle for a nano/micro control word with a bus request.
- c) A 5-state cycle. This applies to those nano/micro words where the address of the next nano control word is conditionally determined by the result conditions of the current ALU cycle.

The 5-state cycle also applies to those bus cycles that follow any cycle where a fault causing an Abort Condition has occurred.

* A state is equal to one CPU clock period.

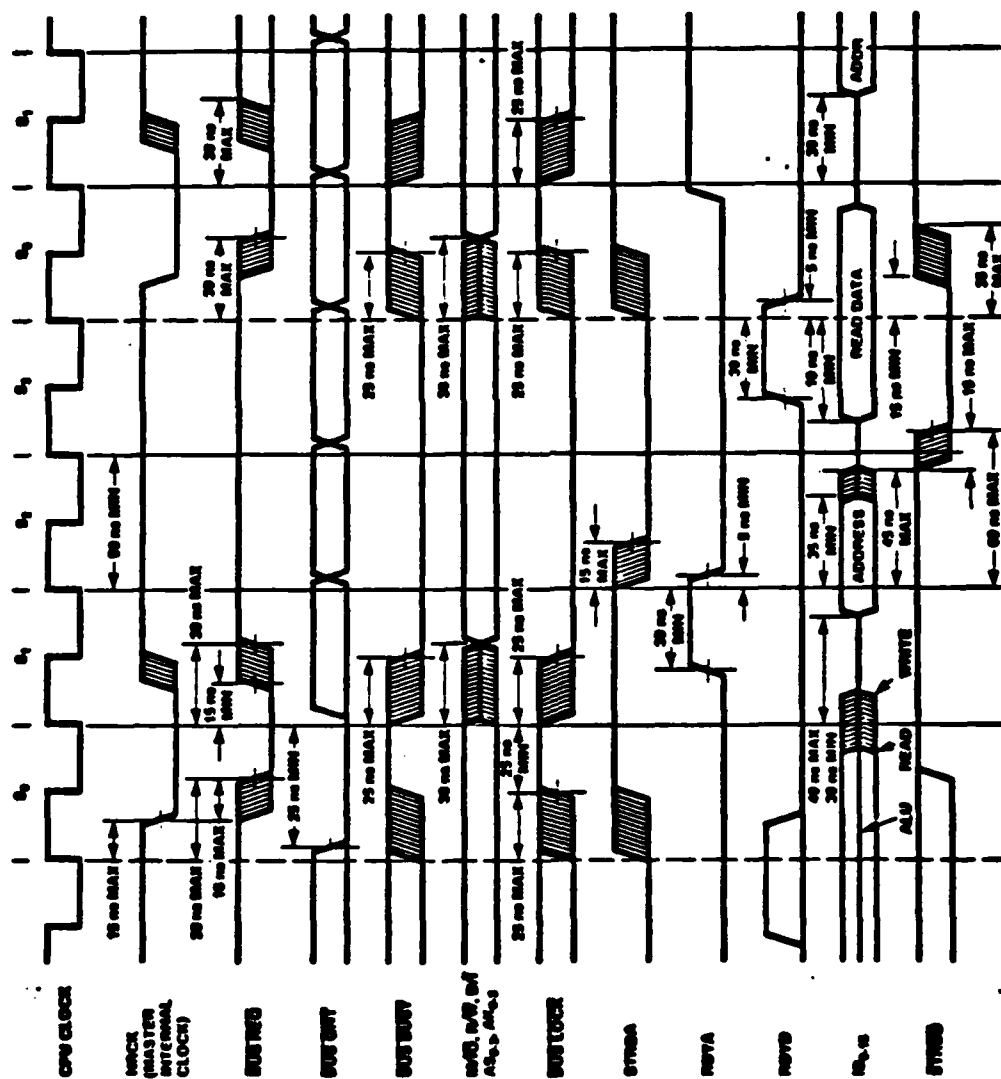
1974 (11-11-11) MINIMUM
MINIMUM WRITE BUS CYCLE



S. Mor
Third Technical Forum
5 May 1982

Chart # 28

PM555 (1770A) TIMING
MINIMUM READ BUS CYCLE



S. Mor
Third Technical Forum
5 May 1982

Chart # 29

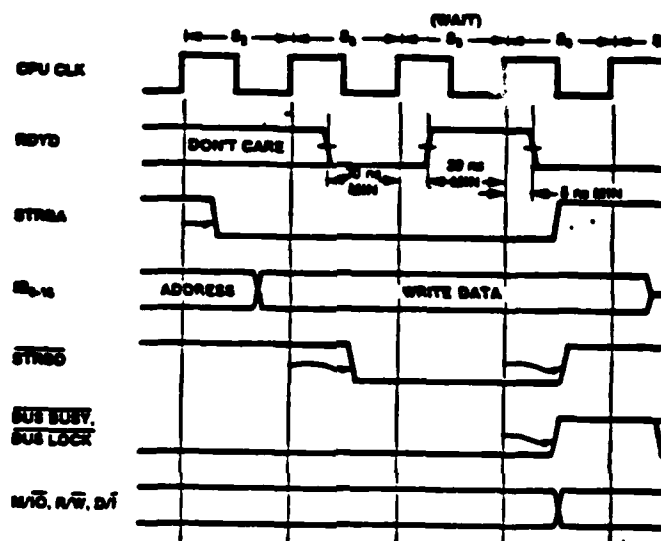


FIGURE 10 THE USE OF "RDYD" SIGNAL WHEN PERFORMING A WRITE BUS CYCLE

S. Mor
Third Technical Forum
5 May 1982

Chart # 30

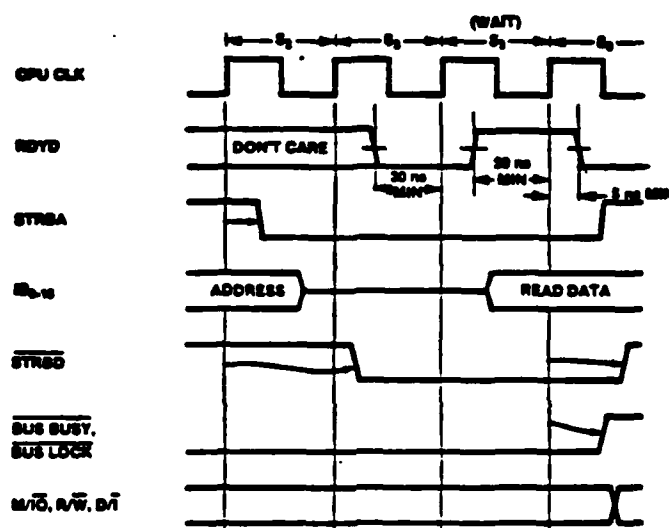
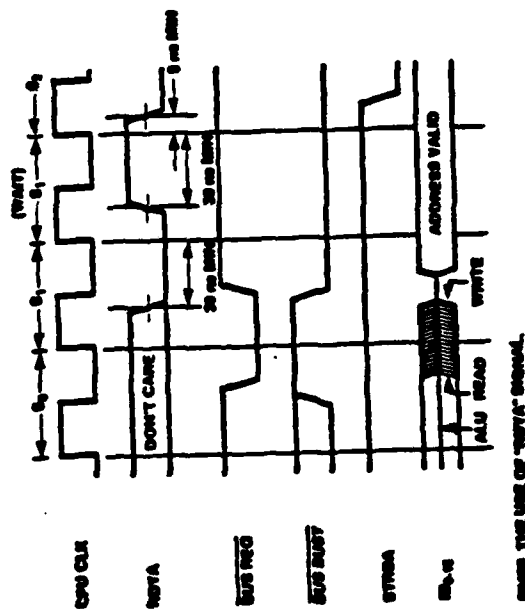


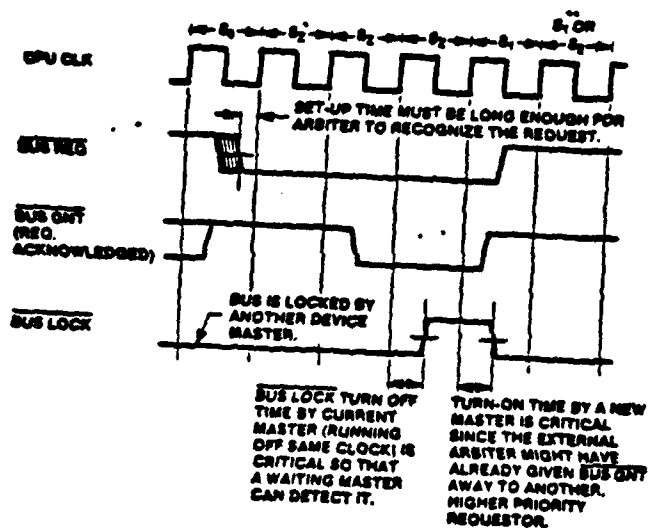
FIGURE 1. THE USE OF "RDYD" SIGNAL WHEN PERFORMING A READ BUS CYCLE



FROM THE USE OF "WAIT" SIGNAL.

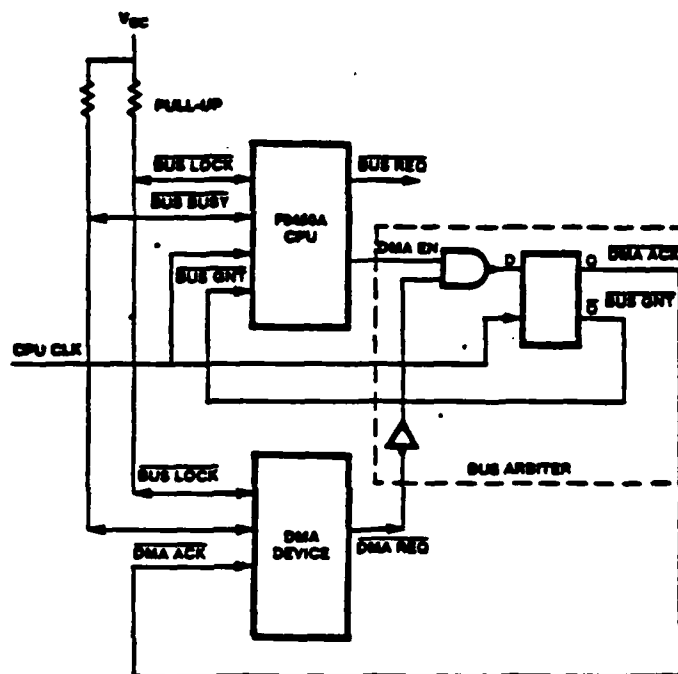
S. Mor
Third Technical Fo
5 May 1982

Chart # 32



P440. SIGNALS REQUIREMENTS FOR ACQUIRING BUS MASTERSHIP

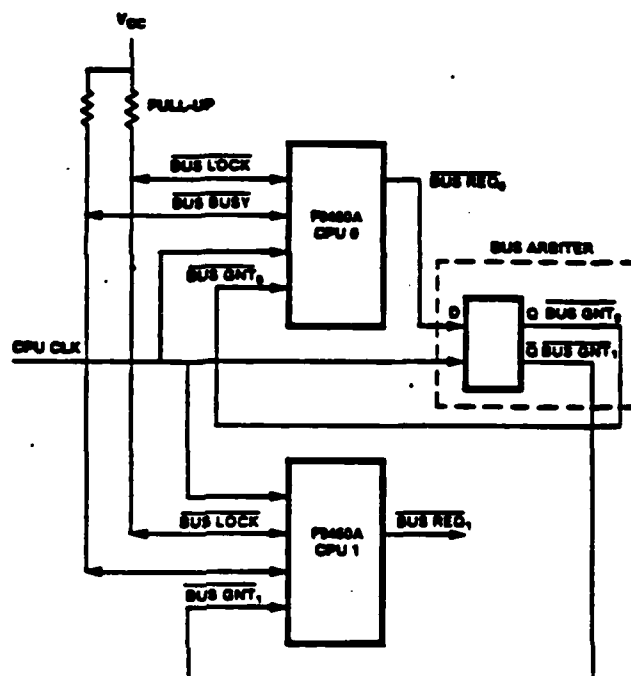
Notes:
 All device masters are P440s or P440-compatible devices
 Bus masters and external arbiters are running off same clock
 S_1 is the high impedance state in which all CPU drivers are 3-stated
 S_2 is the high impedance state in which all CPU drivers are 3-stated
 S_3 Depending on the status of RDYA input.



PM88A
A SINGLE CPU/DMA SYSTEM WITH
AN EXTERNAL ARBITER

S. Mor
Third Technical Forum
5 May 1982

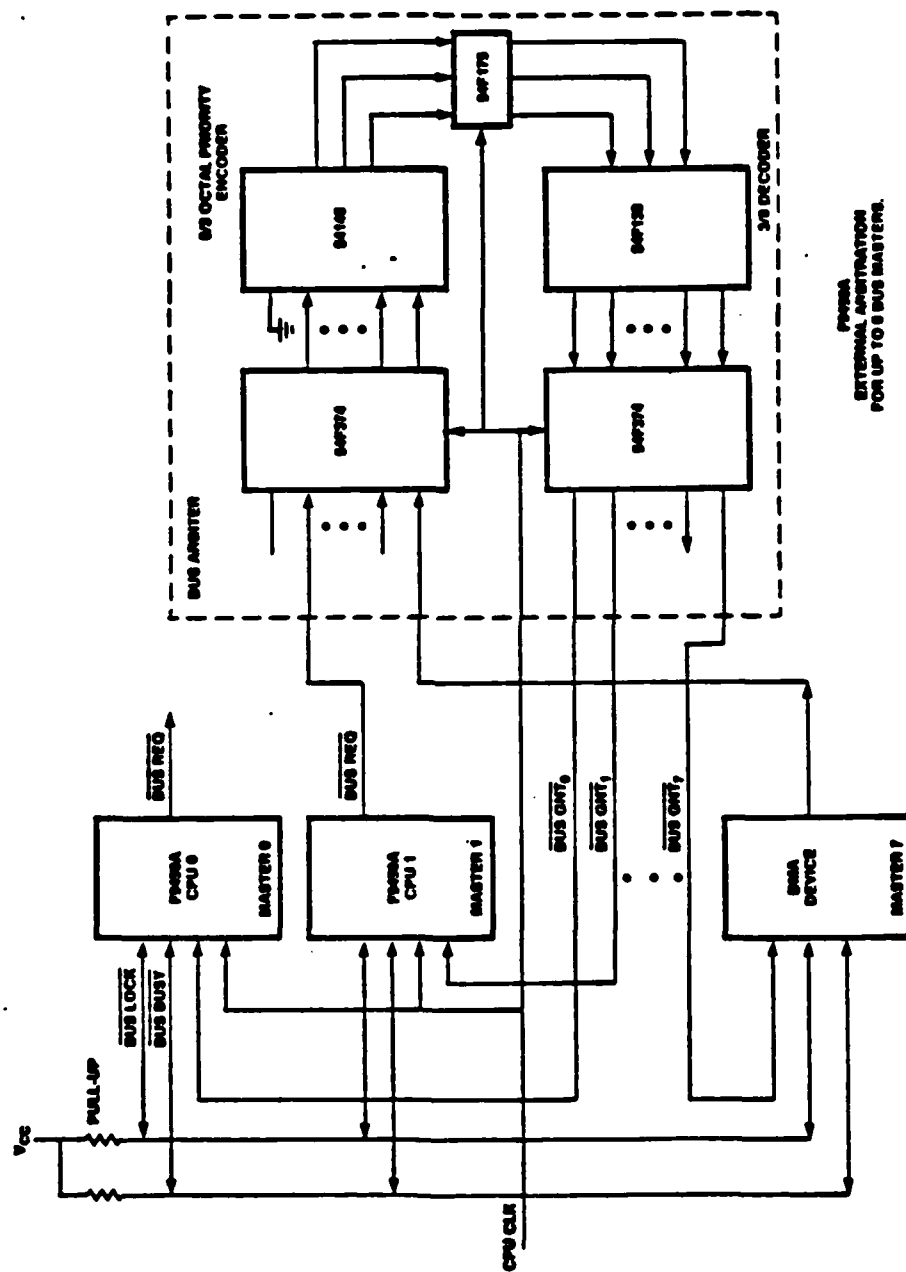
Chart # 34



A DUAL PM80A CPU WITH
EXTERNAL ARBITER

S. Mor
Third Technical Forum
5 May 1982

Chart # 35



S. Mor
Third Technical Forum
5 May 1982

Chart # 36

EXTERNAL FAULTS:

SYNCHRONOUS FAULT INPUTS (LATCHED BY TRAILING EDGE
OF BUS BUSY SIGNAL)

- MEMORY PROTECT ERROR
- MEMORY PARITY ERROR
- EXTERNAL ADDRESS ERROR

ASYNCHRONOUS FAULT INPUTS (EDGE SENSITIVE)

- SYSTEM FAULT 0
- SYSTEM FAULT 1
- SYSTEM FAULT 2

S. Mor
Third Technical Forum
5 May 1982

Chart # 37

FAULT REGISTER (FT)

THE FAULT REGISTER IS 16 BITS WIDE.

- BIT 0 CPU MEMORY RPROTECT ERROR
- BIT 1 NON-CPU MEMORY PROTECT ERROR
- BIT 2 MEMORY PARITY ERROR
- BIT 3 SPARE
- BIT 4 SPARE
- BIT 5 ILLEGAL I/O ADDRESS
- BIT 6 SPARE
- BIT 7 SYSTEM FAULT 0
- BIT 8 ILLEGAL MEMORY ADDRESS
- BIT 9 ILLEGAL INSTRUCTION
- BIT 10 PRIVILEGED INSTRUCTION
- BIT 11 ADDRESS STATE ERROR
- BIT 12 SPARE
- BIT 13 BITE (BUILT-IN TEST) OR SYSTEM FAULT 1 OR 2
- BIT 14 SYSTEM FAULT 1
- BIT 15 SYSTEM FAULT 2

ANY BIT SET IN THE FAULT REGISTER, WILL CAUSE A
MACHINE ERROR (PIR, BIT 1).

S. Mor
Third Technical Forum
5 May 1982

Chart # 38

ABORT CONDITIONS:

UNRECOVERABLE ERRORS:

- * ILLEGAL INSTRUCTION
- * INSTRUCTION PROTECT FAULT
- * INSTRUCTION PARITY
- * INSTRUCTION ILLEGAL ADDRESS
- * ADDRESS STATE FAULT

MAJOR ERRORS:

- * PRIVILEGED INSTRUCTION
- * CPU ACCESS MODE
- * CPU WRITE PROTECT
- * CPU DATA ILLEGAL ADDRESS
- * ILLEGAL I/O ADDRESS

S. Mor
Third Technical Forum
5 May 1982

Chart # 39

INTERRUPT DEFINITIONS

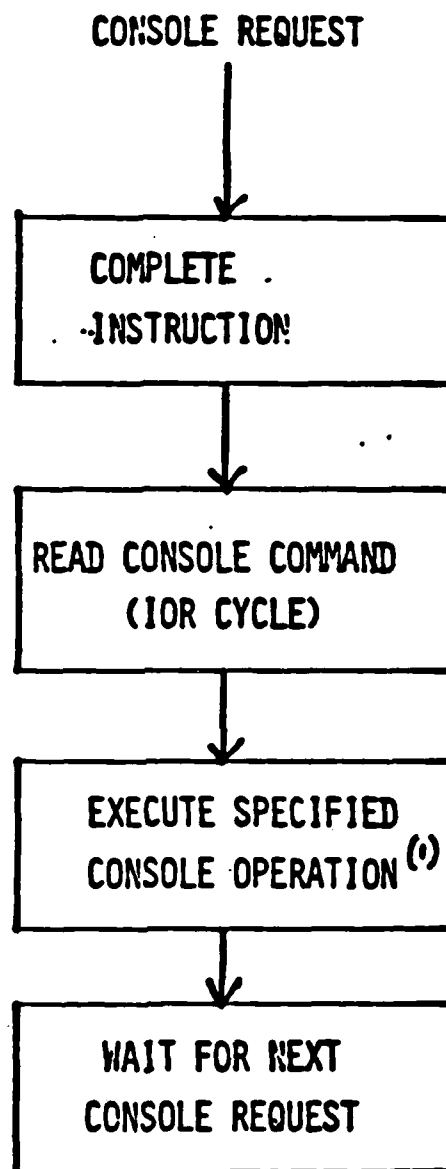
Interrupt Number	MX/PI Register Bit Number	Interrupt Linkage Pointer Address (Hex)	Interrupt Service Address (Hex)	
0	0	20	21	Power Down (cannot be masked or disabled)
1	1	22	23	Machine Error (cannot be disabled)
2	2	24	25	User 0
3	3	26	27	Floating Point Overflow
4	4	28	29	Fixed Point Overflow
5	5	2A	2B	Executive Call (cannot be masked or disabled)
6	6	2C	2D	Floating Point Underflow
7	7	2E	2F	Timer A
8	8	30	31	User 1
9	9	32	33	Timer B
10	10	34	35	User 2
11	11	36	37	User 3
12	12	38	39	Input/Output Level 1
13	13	3A	3B	User 4
14	14	3C	3D	Input/Output Level 2
15	15	3E	3F	User 5

NOTE: Interrupt number 0 has the highest priority. Priority decrease with increasing interrupt number.

S. Mor
Third Technical Forum
5 May 1982

Chart # 41

CONSOLE SEQUENCE



(1) READING AND WRITING TO THE CONSOLE IS DONE USING I/O CYCLES.
ADDRESSES ARE:

- CONSOLE COMMAND : 8400
- CONSOLE READ ADDRESS : 8401
- CONSOLE WRITE ADDRESS : 0400

S. Mor
Third Technical Forum
5 May 1982

Chart # 42

SELF TEST (BIT)

INVOKED BY:

- The power-up (initialization) routine

SELF TEST FUNCTIONS:

- Reads/writes all registers in register file
- Causes fault and verifies that the fault bit is set and a machine error interrupt is pending
- Verifies ALU and pre-ALU Functions
- Checks booth hardware by performing a multiply
- Checks divide hardware by performing a divide
- ALU shifter is checked right/left by booth/divide
- Verified ROM constants can be accessed
- Verified IC/MAR can be accessed and incremented
- Generates an instruction (using a ROM constant). Loads it into IR and executes it, to check a full path through the control section.

ERRORS IN BIT:

- Set bit 13 in the fault register
- Normal power up will not be set

S. Mor
Third Technical Forum
5 May 1982

Chart # 47

INITIALIZATION

INITIALIZATION FUNCTIONS

READS THE CONFIGURATION FROM AN I/O DEVICE TO DETERMINE
THE PRESENCE OR ABSENCE OF:

MMU BIT 15

BPR BIT 14

CONSOLE BIT 13

CO-PROCESSOR BIT 12

INITIALIZES MMU (IF PRESENT)

INITIALIZES BPR (IF PRESENT)

RESETS REGISTERS

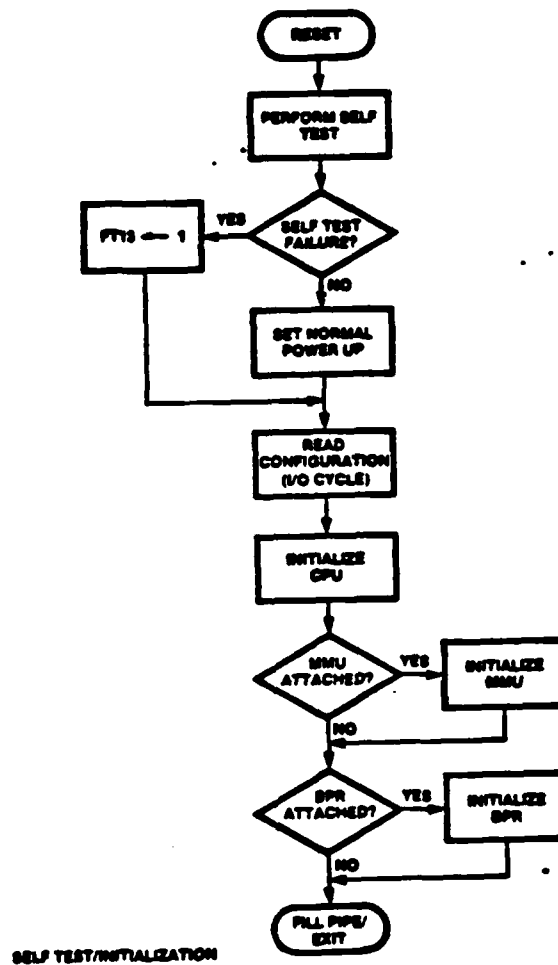
STARTS TIMERS

DISABLES INTERRUPTS/DMA

RESETS TRIGGER GO

FILL THE PIPE

S. Mor
Third Technical Forum
5 May 1982



S. Mor
Third Technical Forum
5 May 1982

Chart # 49

1750A THROUGHPUT CALCULATION SUMMARY

SUM OF WET-SINGLE PRECISION FIXED PT. INSTRUCTIONS- 43.673
 SUM OF WET-DOUBLE PRECISION FIXED PT. INSTRUCTIONS- 14.215
 TOTAL WET FOR FIXED PT. INSTRUCTIONS- 57.888

SUM OF WET-FLOATING PT. INSTRUCTIONS- 80.365
 SUM OF WET-EXTENDED FLOATING PT. INSTRUCTIONS- 8.160
 TOTAL WET FOR FLOATING PT. INSTRUCTIONS- 88.525

TOTAL WET FOR ALL INSTRUCTIONS-
 146.413

SUMMATION OF WET/MNU OVER ALL INSTRUCTIONS-
 SUMMATION OF WBU1 OVER ALL INSTRUCTIONS-

168.688
 44.630

FIXED POINT THROUGHPUT-

1451.090 KOPS

FLOATING POINT THROUGHPUT-

180.740 KOPS

TOTAL THROUGHPUT-

683.002 KOPS

THROUGHPUT/MNU-

592.812 KOPS

BUS UTILIZATION-

30.482 %

INPUTS-- CLK. FREQ.- 20.0

MEMORY ACCESS TIME- 30

S. Mor
 Third Technical Forum
 5 May 1982

Chart # 50

9443 OPERATION

- represents 4 I/O addresses to the CPU
 - Starting operand address (Address register)
 - control register (specifies rounding mode, IEEE/1750A format and so forth)
 - Instruction register
 - Status register (contains exceptions and faults, one of them is illegal 9443 OP)
- There are 8 internal registers in single precision or 4 registers in double (extended) precision.
- the 9443 operations are memory to register (destination may be memory or register), or register to register.
- In memory operations, the 9443 will fetch its operands from memory (using the starting address supplied by the CPU). The number of memory accesses depends on the instruction and precision mode.
- Once the 9443 gets its AL, CR, IR registers loaded it will bid for the bus (if one of the operands is in memory) using the general arbitration scheme.
- Upon completion of its operation, the 9443 will activate one of its 2 pins that indicate completion. One indicates completion without faults, and the other indicates faults.
- These pins may be polled (using IO instructions) or used for interrupts to the CPU.
- the 9443 may be further interrogated^{as} to the nature of the faults, by reading (using IO) its status register.

S. Mor
Third Technical Forum
5 May 1982

Chart # 51

9443 REGISTERS REPRESENTATION IN I/O SPACE

AL: LOW ORDER OPERAND ADDRESS

AH: HIGH ORDER OPERAND ADDRESS

CR: CONTROL REGISTER (IEEE/
1750A FORMAT)

IR: INSTRUCTION REGISTER

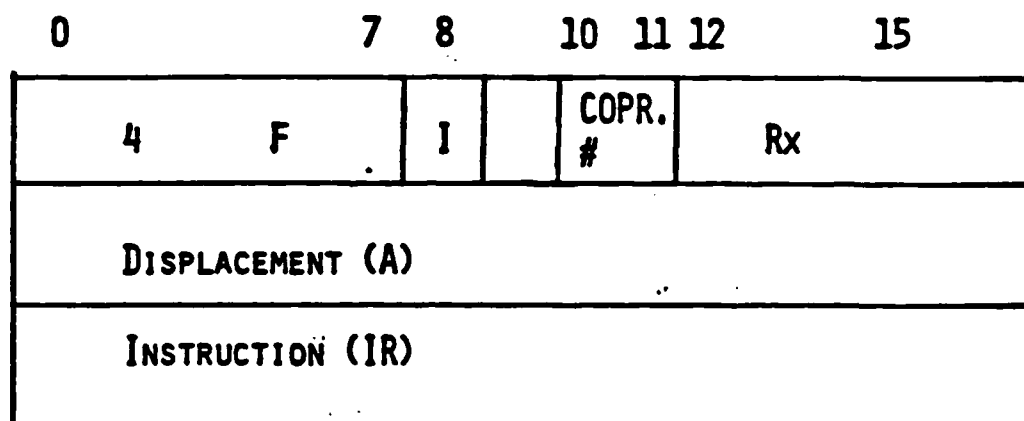
SR: STATUS REGISTER (EXCEPTIONS
AND FAULTS).

9443

S. Mor
Third Technical Forum
5 May 1982

Chart # 52

BIF INSTRUCTION FORMAT



AVAILABLE 1750A ADDRESSING MODES:

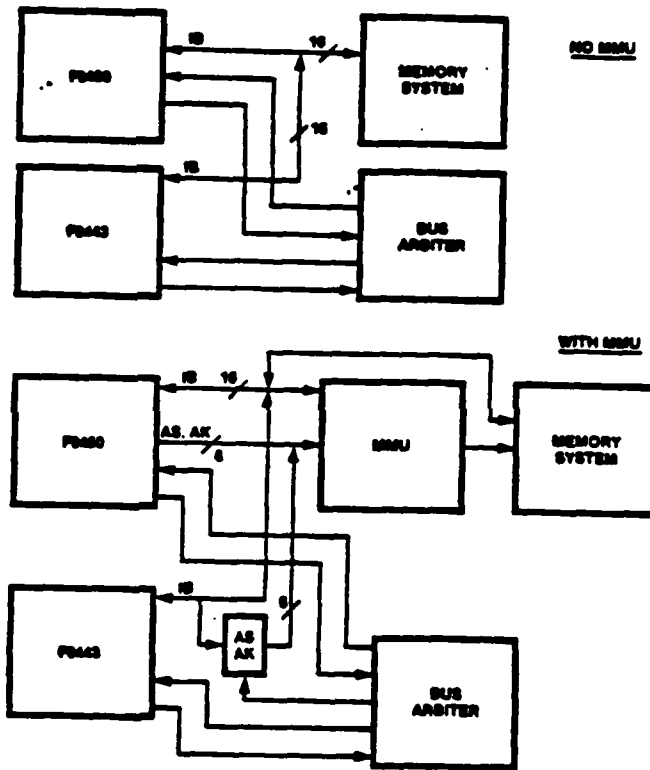
1) MEMORY DIRECT (D):	[A]	}	I = 0
2) MEMORY DIRECT INDEXED (Dx):	[A + (Rx)]		
3) MEMORY INDIRECT (I):	[(A)]	}	I = 1
4) MEMORY INDIRECT INDEXED (Ix):	[[A + (Rx)]]		

COPR. #: SELECTS ONE OF FOUR POSSIBLE COPROCESSORS

S. Mor
Third Technical For
5 May 1982

Chart # 53

PM400/PM443 CONFIGURATION



BIF SEQUENCE

BEGIN

CALCULATE EFFECTIVE ADDRESS (EA) USING THE NORMAL BODA;

IF COPROCESSOR NOT PRESENT THEN SIGNAL ILLEGAL OP AND
TERMINATE

ELSE

BEGIN

CALCULATE I/O ADDRESS ACCORDING TO COPROCESSOR
NUMBER; SEND (IOW) EA TO COPROCESSOR AL
REGISTER (I/O ADDRESS 0800 FOR COPROCESSOR #0);
READ NEXT WORD (IR);
SEND (IOW) IR TO COPROCESSOR INSTRUCTION
REGISTER (I/O ADDRESS 0801 FOR COPROCESSOR #0)

END

END;

NOTE:

THE COPROCESSOR CONTROL REGISTER (CR) THAT IS SELECTING
1750A FORMAT IN THE 9443, AND THE COPROCESSOR AS/AK
REGISTER (EXTERNAL LATCH) ARE LOADED SEPARATELY FROM
THE 9450 USING X10 INSTRUCTIONS.

F9451 - 1750A MMU

o

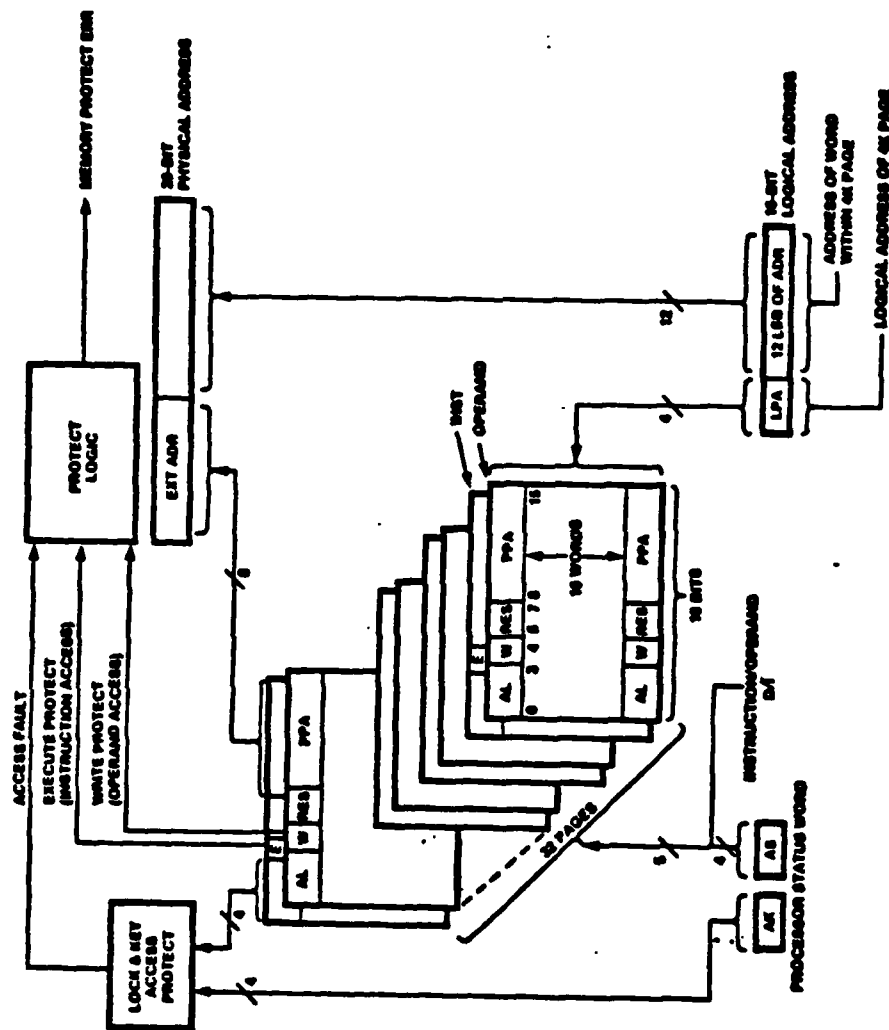
- * FUNCTIONS
- * IMPLEMENTATION
- * TIMING
- * PIN-OUT
- * SYSTEM CONFIGURATION

S. Mor
Third Technical Forum
5 May 1982

MMU FEATURES

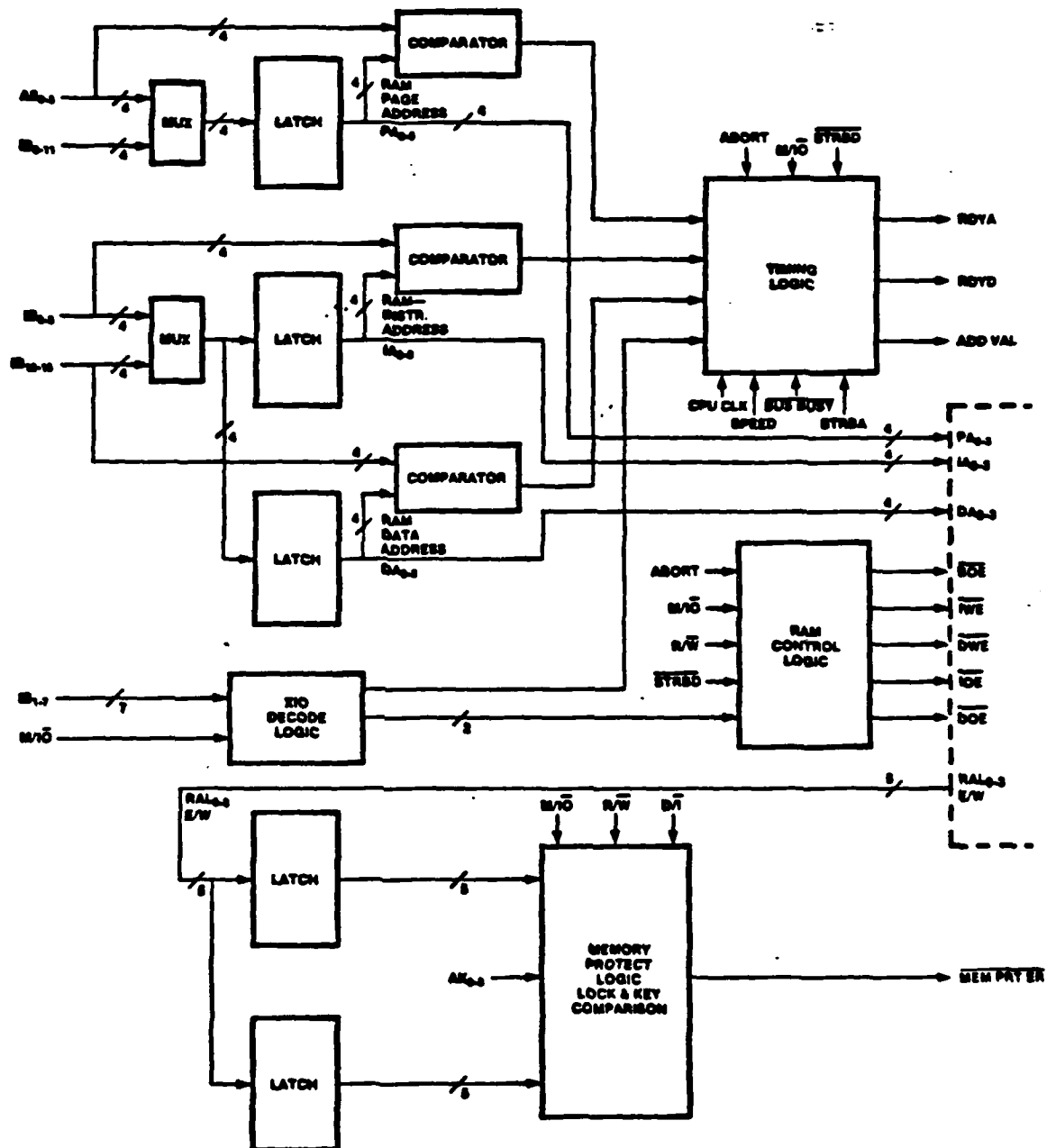
- LOGICAL TO PHYSICAL ADDRESS TRANSLATION
- ADDRESSING SPACE OF 1M WORDS (20 BIT ADDR)
- 2 MAPS:
 - 1) INSTRUCTION MAP (256 x 16)
 - 2) OPERAND MAP (256 x 16)
- PROTECTION IN LOGICAL SPACE OF 4K PAGES FOR THE FOLLOWING:
 - 1) ACCESS KEY PROTECT
 - 2) WRITE PROTECT
 - 3) EXECUTE PROTECT

S. Mor
Third Technical Forum
5 May 1982



S. Mor
Third Technical Forum
5 May 1982

Chart # 58



P441 MMU BLOCK DIAGRAM

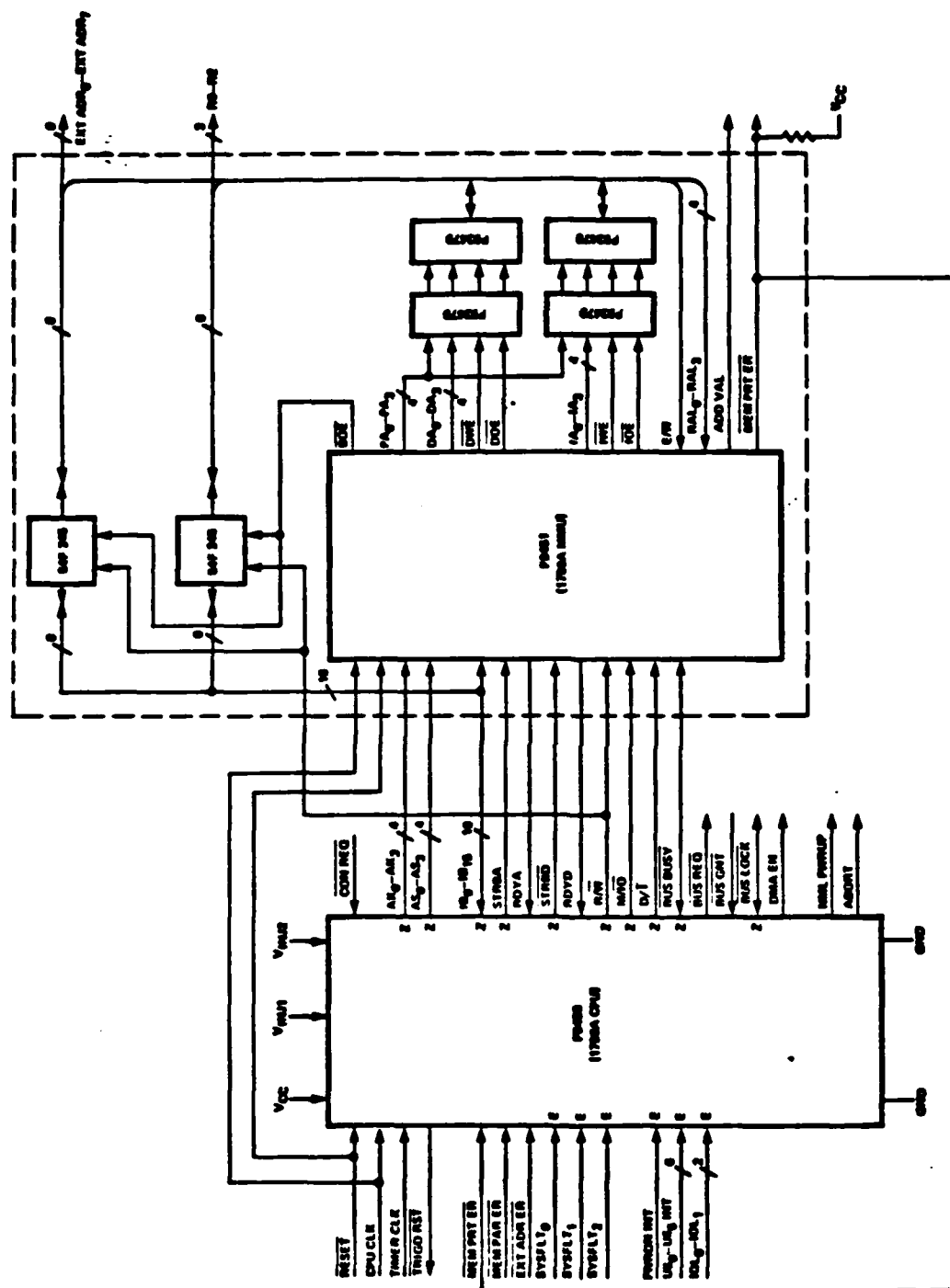
S. Mor
Third Technical Forum
5 May 1982

MMU WAIT STATES

CLOCK RATE

		20 MHZ		10 MHZ	
TYPE OF CYCLE		# S1	#S3	#S1	#S3
I/O	MMU	3	1	1	0
	MMU	2	0	1	0
MEMORY	PAGE BOUNDARY CROSSED	4	0	2	0
	NO PAGE BOUNDARY CROSSED	2	0	1	0

S. Mor
Third Technical Forum
5 May 1982



S. Mor
Third Technical Forum
5 May 1982

Chart # 63

F9452 - 1750A BPR

- * FUNCTIONS
- * IMPLEMENTATION
- * TIMING
- * PIN-OUT
- * SYSTEM CONFIGURATION

S. Mor
Third Technical Forum
5 May 1982

Chart # 64

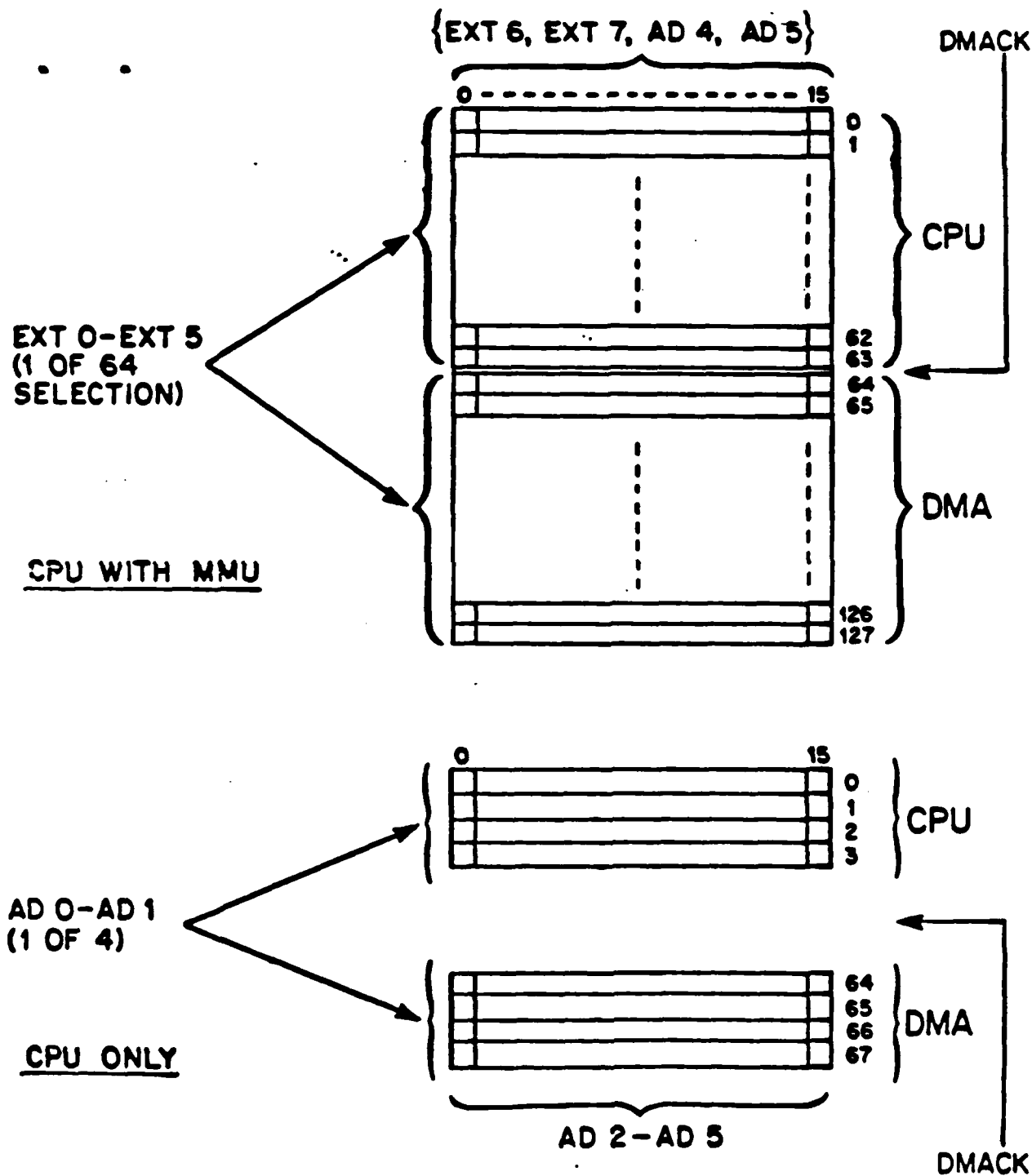
BPR FUNCTIONS

- WRITE PROTECT OF PHYSICAL CPU MEMORY
- WRITE PROTECT OF PHYSICAL DMA MEMORY
- PROTECTION IS DONE ON 1K PAGES
- PROVIDES GLOBAL WRITE PROTECT FROM
INITIALIZATION UNTIL IT IS ENABLED
(BY AN XIO INSTRUCTION)

S. Mor
Third Technical Forum
5 May 1982

Chart # 65

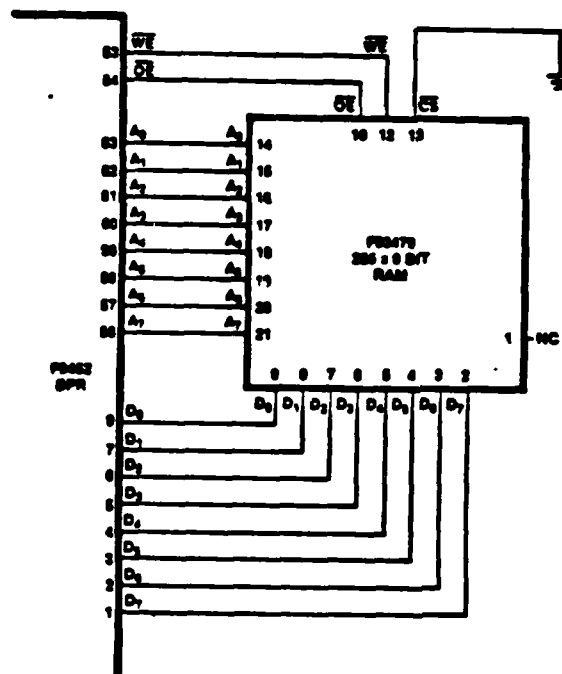
BPR STRUCTURE



S. Mor
Third Technical Forum
5 May 1982

Chart # 66

PM452 SPR INTERCONNECTION



S. Mor
Third Technical Forum
5 May 1982

Chart # 68

WAIT STATES

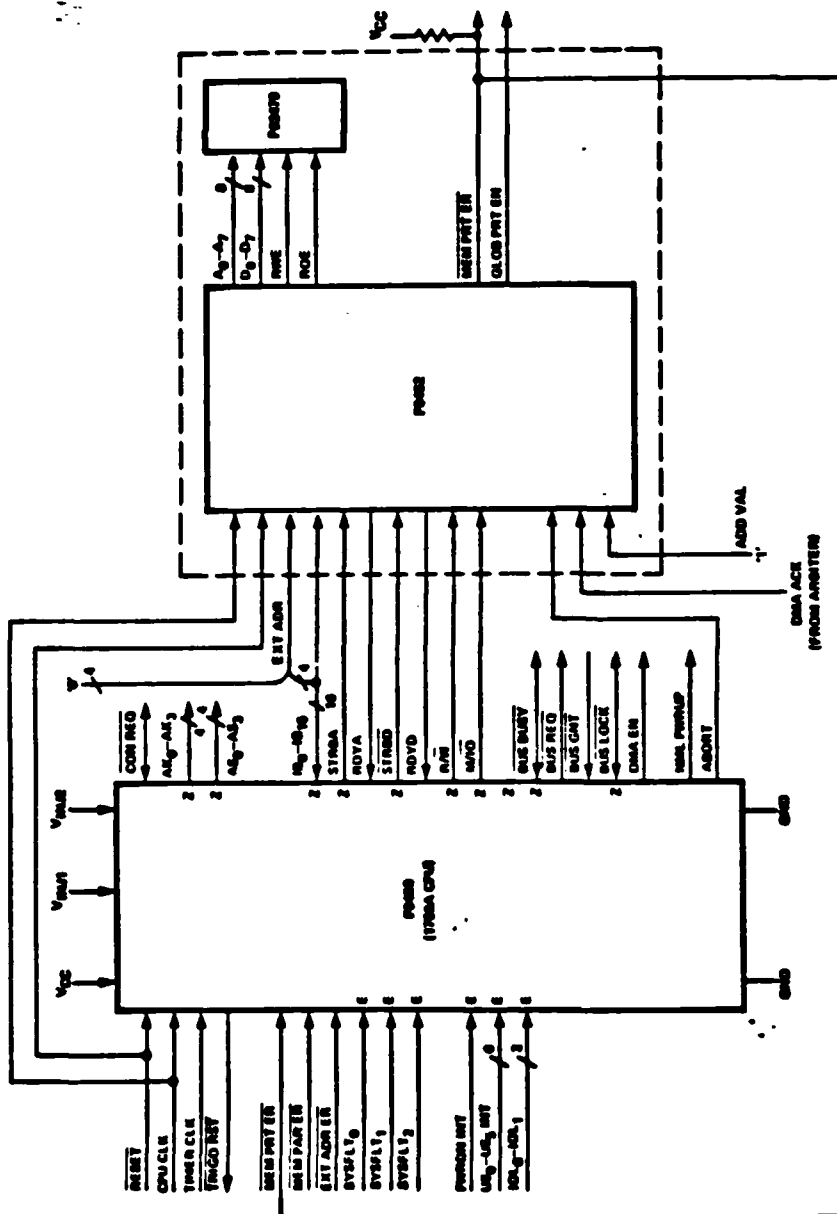
NEEDED IN ADDRESS PHASE:-
(VALID STRBA TIME)

CPU CLOCK FREQ RANGE (MHZ)	READ	MEMORY		READ ¹⁰	WRITE
		HIT	WRITE Miss		
15-20	1	2	3	2	2
12-15	1	1	2	2	2
8-12	1		1	1	1
LESS THAN 8	0		0	1	1

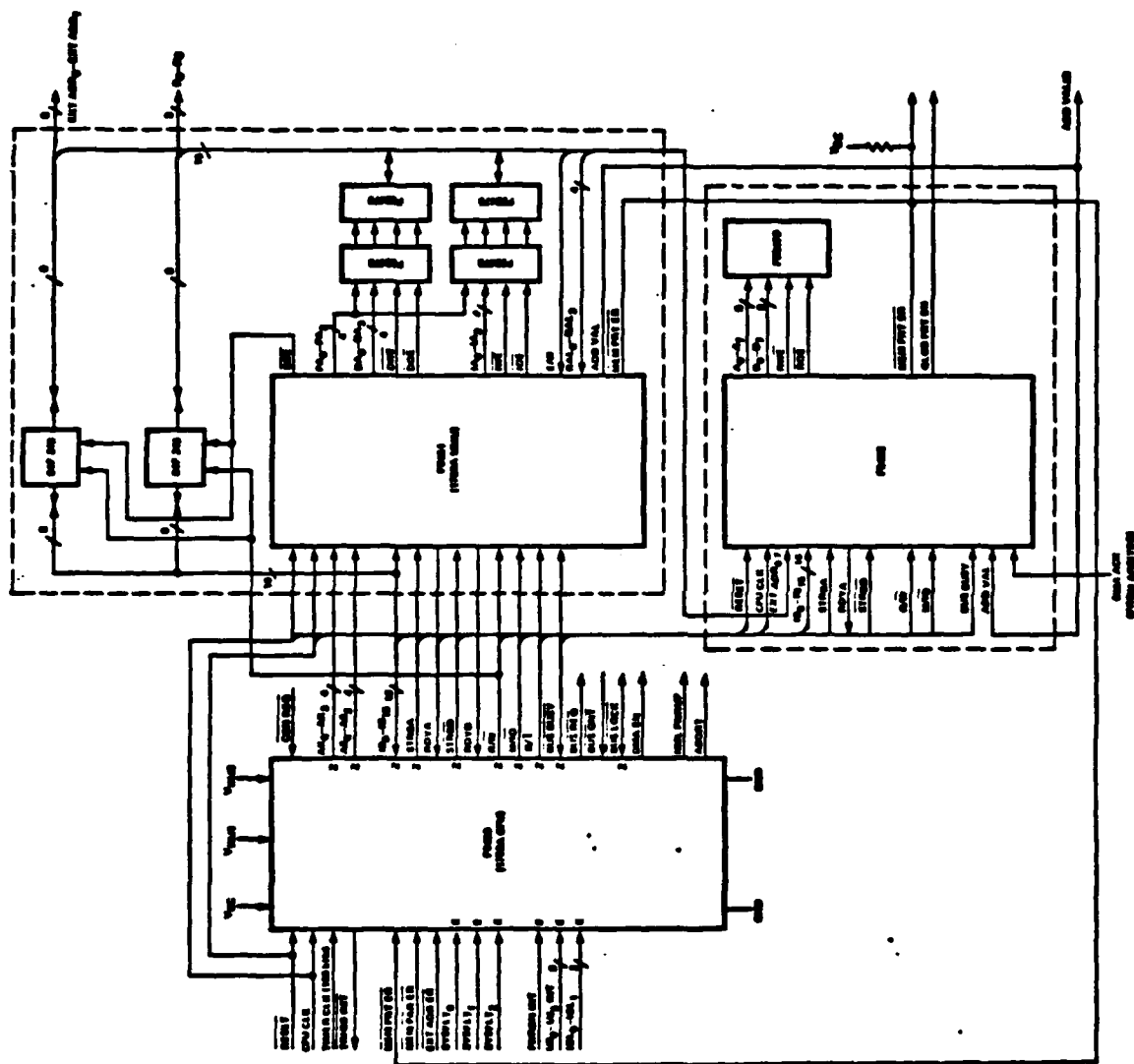
NEEDED IN DATA PHASE:-
(VALID STRBD TIME)

FREQ RANGE (MHZ)	READ	MEMORY		OTHERS	IO BPR
		WRITE			
15-20	0	0		0	6
12-15	0	0		0	4
8-12	0	0		0	4
LESS THAN 8	0	0		0	2

S. Mor
Third Technical Forum
5 May 1982



S. Mor
Third Technical Forum
5 May 1982
Chart # 72



S. Mor
Third Technical Forum
5 May 1982

Chart # 73

F-16 MIL-STD-1750A MICROPROCESSOR DEVELOPMENT PROGRAM

MANAGEMENT OVERVIEW

RANDAL MOORE

PROGRAM OVERVIEW

DAN WILNAI

FUNCTIONAL AND TECHNICAL DESCRIPTION

SHAI MOR

➤ APPLICATIONS HARDWARE

HENRY LYNN

ENGINEERING TEST/EVALUATION EQUIPMENT

DAVID SCOTT

**THIRD TECHNICAL FORUM
WRIGHT-PATTERSON AFB**

5 MAY 1982

**H. C. Lynn
Third Technical Forum
5 May 1982**

Chart # 0

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #1

I have tried to anticipate some of the questions that you might have about how you can apply this microprocessor in your own system. Consequently, I have broken it down like this: I will go through the overview, the electrical interface, clocks and timers, the faults, interrupts, bus control philosophy, specification of the memory I/O bus, memory system design, and the I/O system design.

CHART #2

As an overview, you've noticed that the CPU clock goes to all the components of the chip set and must go to the bus arbitor as well. The CPU clock synchronizes all the signals in the chip set that are local to the CPU card. Because of the multiplexed memory I/O bus, there is a choice of memory interface systems that can be designed using this microprocessor. You can keep the multiplex memory I/O and extend the address phase of the cycle. Or you can demultiplex the bus as it comes off the CPU and transmit the address bus in parallel with the data bus. A change in the specification 16ZE181, which will be in the new specification, will take the bus arbitor out of the CPU and place it in the external system. We do not believe this presents an undue burden to the whole system and we show you ways to solve that.

We define the interrupts to be edge triggered and we'll talk about the faults that generate the machine error interrupts.

CHART #3

Our view of the microprocessor chip set is as three functional groups of components - the CPU, the memory management plus its peripheral support and the block-protect RAM. These components were specified to interface in this manner with a minimum amount of additional logic needed to make the chip set work. The bus arbitor, which was formerly part of the chip set, is brought out. We found out that in architecturing different types of buses, it was hard to get one type of bus arbitor to do all the functions. The bus arbitor is now out in the system where the user can tailor it for his specific application.

CHART #4

When the block-protect RAM operates with the CPU as is the case in the F-16, the extended address bus inputs to the block-protect RAM are split into two groups. The four least significant bits go down to the memory I/O bus where the block-protect RAM will latch the most significant bits of the address bus as they come out of the CPU. Zeroes are inserted on the four most significant bits.

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #5

This is a revised block diagram of the CPU. In this I've split off the system faults which are the asynchronous inputs to the fault register and chosen to call those machine error interrupts. That's because functionally they do act like interrupts. They're edge triggered, like the normal interrupts and they all OR their bits together to make a single machine error interrupt. The bus control signals in the memory I/O bus will be discussed in turn.

CHART #6

On the MMU there are two special signals - the ready address which goes to the CPU and the memory protect error which goes to the CPU and the memory system. Both are open collector type functions for OR tying.

CHART #7

Similarly the block-protect RAM has the same type of open collector outputs on it. These outputs are designed to work with 1K pull-up resistors. Next,

CHART 8

The question came up, what about the power requirements and what about signal interface requirements?

CHART #9

These are power requirements expressed in watts. The 5 volt TTL part of the CPU at CDR looked like 1.3 watts total power. There was 1 amp of injector current at a nominal 1.3 volts. That is, a current type input and not a voltage type input, so this 1.3 watts must be supplied by another power source. You can bleed current from the 5 volt power supply if you can stand the 3.7 watt loss in the external resistors. Normally, you would provide a current source to drive that input. The MMU has a very high power dissipation because of the bipolar RAMs and TTL that interface with it. The gate array part of the chip set is very low in power and was less than .2 watt total injector power at 1.3 volts. The block-protect RAM, because it doesn't have as many TTL parts associated with it, is expected to be about 1.1 watts out of the 5 volt supply and less than .2 watt out of the 1.3 volt supply. These specifications cover operation from -55 to 125° case temperature.

CHART #10

What about interface drive? It's bipolar, it's TTL compatible. You can break the signals down into three groups - the three state signals, the uni-directional inputs and outputs, and the open collector signals. This tends to

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #10 (CONT'D)

tie all those signals together on one interface chart. The inputs are specified as 800 microamps max load for logic zero, 40 microamps for logic one, using true logic. Output sink 8 milliamps and supply 400 milliamps at 2.4 volts. The leakage of the tri-state in the open collectors is expected to be about the same as normal TTL gates.

CHART #11

OK, about the clocks and timers used in the chip set.

The CPU clock frequency is selected by the application.

CHART #12

The CPU design as Shai pointed out is completely static. It will run from DC and is designed to go to 20 megahertz. Internal state changes with the CPU take place on the low to high transition of the clock. 20 megahertz is the design maximum operating frequency and at that frequency you must be very careful about your clock symmetry requirements because the components in the chip set will require a minimum pulse width high and low of 20 nanoseconds.

CHART #13

Shai told you that the timers are counting asynchronously with a 100 kilohertz clock. The 100 kilohertz must be supplied by your system. Because of power, we changed from the 4 megahertz input which is in 16ZE181 down to 100 kilohertz. The 100 kilohertz is sampled by a signal internal to the microprocessors. This is to make sure that the timer will have settled out by the time it needs to be read if the microprocessor is doing an I/O operation with one of the timers. That way we can assure data consistency.

CHART #14

The external scale to divide down 20 megahertz to the 100 kilohertz is just a couple of components in conventional TTL and was not considered to be an undue burden to the implementor.

CHART #15

Ok, the trigger go counter is a general purpose counter in 1750 which is not specified to have any period length or any particular count format. What 1750 does provide is an XIO command to reset the timer. That command is explicitly decoded by the CPU and comes out as a pin, and may be used to reset the trigger go counter. Now in 1750 it says it also sets an output discrete. On F-16 we use the trigger go counter output as a software timing fault generator, and take it back to the system zero fault input on the central processing unit.

CHART #16

Which leads right into faults.

CHART #17

Some faults are detected in the chip set and some of them must be detected cooperatively by the CPU with the whole system. The CPU on its own can detect illegal instructions, privileged type instruction errors, built-in test errors at power on, or address state faults. All of those are explicit and well defined in the specification. The MMU will detect the memory protection violations which are two types - the access violations and the execute write protect violations. Block-protect RAM if used will detect memory write protect violations using separate maps for the CPU and the DMA.

CHART #18

Because it is impossible for the microprocessor to know in advance how much memory will be in any particular application, we ask the whole system to do the illegal address decoding for memory addresses. And also, using the same rationale we don't know which XIO commands will be implemented in the user system. We ask the host system to figure out which commands are legal for the system and which commands are illegal for the system.

Optional fault detection is built into the fault register with hooks for memory parity error. The software fault time out in the F-16 turns out to be system zero machine error interrupt. The two system BITE interrupts set BITS 14 and 15, respectively, of the fault register and also set BIT 13.

CHART #19

There are exactly three machine error interrupts feeding the fault register. There are other internal detected events which show up in the fault register that also cause machine errors. Six spare inputs in 1750 are allocated to the user to define, as he will, and two are left for interrupt expansion.

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #20

All the interrupts have similar characteristics, that is, they are all asynchronous to events that are going on inside the CPU. We have elected to make them edge triggered and they are detected on the low to high transition of the signal input. And also because we recognize that edge-triggered things are susceptible to noise, they are received on the CPU by hysteresis gates for extra noise immunity.

CHART #21

Machine interrupts or faults, if you will, and the BITS that they affect in the fault register are defined by this table. BIT 7, the system fault zero, is a software fault time out on the F-16. System fault 1 and system fault 2 set BITS 14 and 15 and according to the standard also set BIT 13. This is interrupt level one. It is maskable but cannot be disabled.

CHART #22

The user-defined interrupts and their levels are specified by 1750.

CHART #23

The expansion interrupts in the standard are called I/O level 1 and I/O level 2 and come in the interrupt priority on fixed levels, namely, 12 and 14. The interrupt pins on the CPU are there. The rest of the hardware that must be added to make it conform to MIL-STD-1750 is shown on the next chart.

CHART #24

We have a pseudo-schematic of very large priority encoder taking up to 256 inputs, generating an 8 bit code. Any expansion interrupt sets an interrupt level to the CPU. The CPU in turn responds to that by going out with the 16 bit address which is the input/output code register, and gets those bits onto the bus which correspond to highest priority interrupt.

CHART #25

Ok, a little bit about bus control. Bus control has been generalized a little bit. One of the things bus control does here that it wouldn't normally do is control synchronization. We'll show you some simple bus arbiters.

CHART #26

The bus arbitor's function of course is to control access to the global memory I/O bus. Another function is to synchronize control signals for use by the MMU and the BPR, because when the DMA is going on, the CPU is tri-state and essentially not active. It is sitting there recording faults occurring on the bus. We have made the function of synthesizing the address strobe for non-multiplexed bus applications part of the generalized bus arbitor.

CHART #27

Looks fearsome, it has lots of inputs and outputs and this is the memory interface over here. On the CPU side is the DMA enable/disable 1 CPU Busy, Bus Lock. The CPU request, as Shai pointed out, is only used in multiprocessor applications. In a normal one CPU system, the CPU always has its request on and the DMA can always come and steal the bus anytime DMA is enabled and the bus is not locked and not busy. It provides a bus grant to the CPU which is sampled synchronously at 20 MHz clock. The bus arbitor itself must be synchronous in order to ensure that the setup and hold times for the CPU grant are observed.

Now in order to synthesize the strobe address for the DMA functions (the plus after the name indicating a resynchronized signal by the bus arbitor), the bus arbitor uses the ready address which is being generated jointly by the MMU and the BPR. When the bus goes ready, the bus arbitor will synthesize bus ready. DMA acknowledge goes to the BPR so that it will know which memory map to use for write protection. On this side we have the generalized memory interface form of the signals.

CHART #28

Ok, this is a method of control synchronization which was discussed with Fairchild two weeks ago, which uses the CPU clock and the ready address coming from the BPR and/or the MMU. Suffice it to say that when the bus is not busy, the strobe address flip-flop is preset (active high). When ready address comes along, it is sampled low at the end of the bus cycle of the CPU or the DMA device. The bus goes not busy and resets the resynchronized strobe address. In order to combine the CPU busy with the DMA busy, you may have to use an open collector buffer, or a tri-state buffer if this line has to go any significant distance in your system.

CHART #29

This is the simplified bus control part of the generalized bus arbitor. It uses the two external gates and a single flip-flop to give the CPU grant and the DMA grant to the subsystems.

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #30

Ok, this is what it all looks like when it is pulled together. The ready address, being open collector, is pulled up by a 1K resistor. The Memory/not IO and the Data/not Instruction status lines are such that they may be left in the high state or floating and pulled up by 1K resistors when DMA is going on. These signals including the strobe address resynchronized go to the MMU and BPR. The other part of the DMA interface is on the lower part of the diagram.

CHART #31

We'll talk about the Memory/IO bus interface as it leaves the chip set.

CHART #32

The three status signals that tell you what is going on are Memory/not IO, Read/not Write, and Data/not Instruction. They were chosen so that you would be able to pull these lines to the default (high) state meaning Memory and Data (not Instruction) when DMA is going on. A very straightforward encoding of the status lines down to the point where we get to the XIOs. Now XIOs are divided into two groups: those which are implemented inside of the CPU and those which are implemented within the host system. For those that are inside of the CPU, as part of 16ZE181 (the microprocessor specification) we said we want to be able to monitor those things as they are being loaded or read by the CPU. We want to do that for our software test stations and also for the capability to diagnose faults within the CPU. Consequently when we read or write anything internal to the CPU, the CPU acts like it's a "write" to the external equipment, except that read data is not sampled. For example, if it's reading Timer A, the CPU will do an XIO type bus cycle with "write" status and it will send the contents of Timer A out over the bus at the same time it is being copied into the register file. The external read/write XIO cycles behave normally. The DMA read/writes show that the two status lines can be pulled high.

CHART #33

Ok, Shai told you that you can hold up the address portion of the transfer cycle by withholding the ready address.

CHART #34

The design goals for the setup and hold of ready address are 30 nanoseconds. Status becomes valid 30 nanoseconds after the beginning of the cycle as the bus goes busy. The MMU uses the status along with address information to access the instruction registers or data registers to perform the memory protection algorithm required by 1750 and eventually gets a mapped memory physical address out here. We would not expect that the host system would use the ready address.

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #35

This is the way it all fits together with the bus arbitor, the ready address and resynchronized address strobe being drawn in explicitly. The BPR and MMU cooperatively pull down on the ready address meaning "not ready" at the beginning of the data transfer angle. Once the MMU determines the extended memory address is valid, then it tells the BPR to lock in its map. The BPR, once it determines the write protect status, releases its ready address. The bus arbitor in turn generates the resynchronized address strobe to the MMU and/or the BPR. It uses the bus busy to do that and therefore does not require any use of the CPU busy or address strobe.

CHART #36

This is the timing diagram for the read bus cycle. On a read cycle the address for the memory (or XIO command) comes out first. As the address strobe drops, the address is latched by the host subsystem. The bus must then be turned around and the external bus must be gated into the CPU. Now Fairchild has designed their chip so that the data strobe will not start before the internal bus has gone tri-state so there should be no bus contention right here. The read data comes on to the bus and requires a setup time of 10 nanoseconds setup time with respect to the CPU clock.

CHART #39

But there is a warning. If you turn around the bus to read data, you must turn it around again to get the next address out. From the time strobe data goes away, if you are using strobe data to control your bus buffers, you have 45 nanoseconds to get your bus buffer tri-stated before the CPU begins to send the next address.

CHART #40

Ok, where do you get all your address lines for the system? XIO commands are easiest since they always come from the 16 address bits multiplexed out at the beginning of the cycle. Memory address lines (most significant bits) must be taken from the memory management unit (if present) and the 12 least significant bits come from the multiplexed address/data bus. Address errors are decoded in the host subsystem and reported to the CPU by the illegal address discrete where it sets the bits in the fault register.

CHART #41

This is a block diagram of a non-multiplexed CPU card with a generalized bus arbitor illustrating a serial DMA device bus acquisition scheme. The memory/IO bus is on the left. There is only one gate delay between the transmit/not receive input and strobe data to prevent bus contention at the end of the bus cycle. The 373 latches demultiplex the address lines for the memory/IO subsystem.

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #41 (CONT'D)

Now for DMA to get in and access the memory protect maps in the memory management unit and the block-protect RAM, it must gate its address onto the internal bus. That's the only way the MMU and BPR can get this information. This is done via the address buffers shown in the lower part of the diagram. The bus arbitor will synthesize an addressstrobe for the MMU and BPR using the CPU clock for the DMA access cycle.

The read/write line is shown as being bi-directional. When the CPU is in control, the line is driving out and when the DMA is in control, the line is driving onto the internal bus. The Memory/not IO and Data/not Instruction status lines are simply pulled up at the host subsystem interface.

CHART #42

Ok, this set of notes is about memory system design--access/cycle times or "what do I have to do to make the CPU run without wait states?"

CHART #43

First the CPU alone. If we cut off the processor bus at its pins, not taking into account any external data buffers that are required to interface with memory, then we can talk about access times with respect to a memory system which is everything outside of the chip set. Basically the processor gets the address out and valid 40 nanoseconds after the beginning of the S1 and it required data 10 nanoseconds before the end of S3. Using that, you can write the equations for the access and cycle times where N is the number of wait states and f is the frequency in megahertz.

You have a minimum of three states plus the number of wait states minus 50 nanoseconds. That is summarized in a table on a following chart. The cycle time for an external memory reference without wait states is four states minimum plus the number of wait states.

CHART #44

Now when you have an MMU, the MMU inserts a minimum of two wait states in any memory access time. As Shai pointed out, if you get a miss on the instruction address or operand address, then it adds another two wait states. For the MMU then, the minimum access time becomes five states plus a factor which is expressed here as a Boolean vairable whose value is two if a page boundary is cross on zero otherwise.

H. C. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #44 (CONT'D)

Because the MMU is using the status information generated by the processor, it cannot begin to enable its extended address bus until that gets stable. Beyond that it takes an additional 50 nanoseconds to get the extended address bus stable out here. So if you're counting from the beginning of the first S1, it takes 90 nanoseconds to get the extended address bus valid, with the same requirement of getting data valid 10 nanoseconds before the end of S3.

CHART #45

This chart gives the access/cycle times to operate with a fixed number of wait states for three CPU clock frequencies. At 20 megahertz in order to operate with no wait states requires a 100 nanosecond system memory access time. Now system memory includes everything from the CPU data buffers through the address latches, the address decoders, the access time in the memory and through its buffers (if any). One wait state will get you 150 nanoseconds access time while four wait states will get you 300 nanoseconds. Now with an MMU, assuming no page boundaries are crossed, the access time extends to 150 nanoseconds with a 300 nanosecond cycle time.

This chart includes all delays in the memory system outside of the processor. You can see that it is a very tight memory system which is required.

CHART #46

Another point about the memory system is that the CPU discovers too late in its data transfer cycle that the memory should be write protected. It goes ahead and generates a data strobe and relies on the memory system to use the memory protect error being generated by the MMU or Block-Protect RAM to pass or not pass the data strobe along to the final memory element. One additional note is that with no wait states, the data strobe is specified to be 45 nanoseconds minimum in duration. Also the data strobe is well centered in the middle of the memory protect error signal so that simple gating can take care of that.

CHART #49

Memory error control is all done all by the memory system external to the chip set. It is all reported to the CPU by a single memory parity error discrete which sets bits in the fault register. The line is sampled at the conclusion of each memory cycle on the trailing edge of the bus busy. If it is low at that time, then bit 2 of the fault register is set which triggers a machine error interrupt (if not masked). You can implement any memory error detection scheme you want, e.g., Byte Parity, Word Parity, Hamming Codes. However, all those error signals must get down to one line for fault reporting to the CPU.

H. G. LYNN

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #50

The memory fault status register is an option in 1750 which is not implemented within the chip set but is easy to do if needed. The AF control board met and decided that the memory fault status register should be loaded on illegal address errors, parity errors, and memory protect errors. All these errors 'OR' down conceptually to a single clock line to a register. The contents of the register are shown at the bottom of the chart. The four address state bits of the processor, the memory/not IO status line (inverted) and the four most significant bits of the page address on the memory address bus.

CHART #51

The start-up ROM is a feature that is very easy to implement via XIO commands. You decode the start-up ROM enable and disable commands and use a cross-coupled latch flip-flop cleared by reset so that is always enabled at power on in accordance with the standard.

CHART #52

A few more notes about IO system design are included here.

CHART #53

An example of an IO word that will be in every microprocessor system designed with this chip set is the configuration word. The configuration word tells the CPU whether the MMU, the BPR, the user console and/or a co-processor are present. It must be there. The configuration word is assigned to XIO command 8410.

CHART #54

On the F-16 we use the following XIO command spaces. A lot of commands in the 1750 specification are in the range of 2000-200F, etc. The Block Protect RAM uses 256 locations. The avionics 1553 multiplex data bus is allocated 512 locations, and the chip set reserves 4 locations.

CHART #55

In summary, we believe that we are compatible with all the issues resolved by the 1750A control board on 20 July 1981. We've met all our F-16 requirements described in 16PP379A, dated 7 July 1981. And we think we have a good technical solution to providing a microprocessor chip set for general application.

**MIL-STD-1750A MICROPROCESSOR:
INTERFACE AND APPLICATIONS**

- o OVERVIEW**
- o ELECTRICAL INTERFACE**
- o CLOCKS & TIMERS**
- o FAULTS**
- o INTERRUPTS**
- o BUS CONTROL**
- o MEMORY/IO BUS INTERFACE**
- o MEMORY SYSTEM DESIGN**
- o IO SYSTEM DESIGN**

**H. C. Lynn
Third Technical Forum
5 May 1982**

Chart # 1

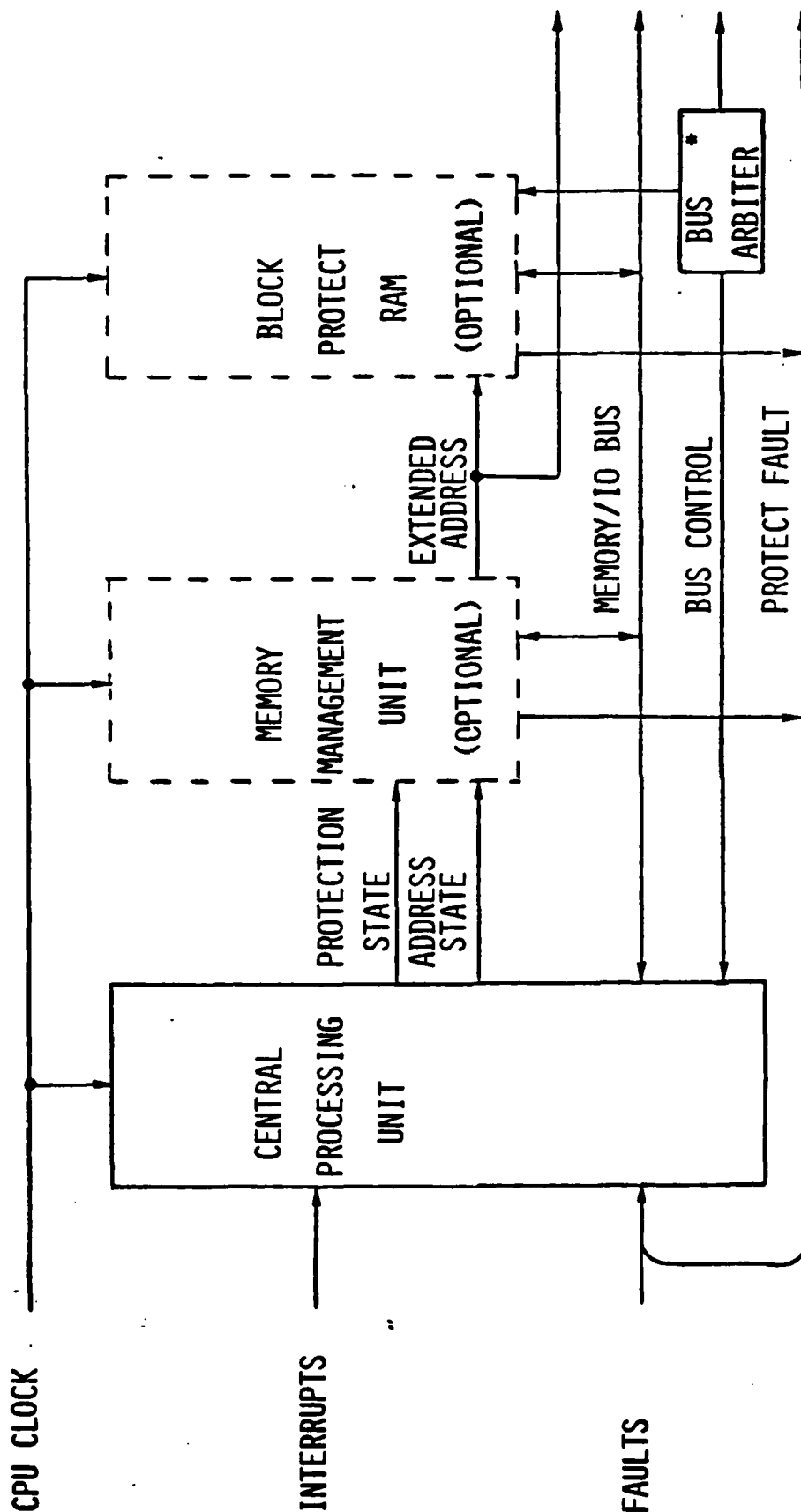
CHIP-SET INTERFACE OVERVIEW

- o CPU CLOCK SYNCHRONIZES SIGNALS IN CHIP-SET
- o MULTIPLEXED MEMORY/IO BUS TRANSFERS DATA
- o EXTERNAL BUS ARBITER CONTROLS BUS ACCESSES
- o EDGE TRIGGERED INTERRUPTS ARE ASYNCHRONOUS
- o FAULTS GENERATE MACHINE ERROR INTERRUPTS

H. C. Lynn
Third Technical Forum
5 May 1982

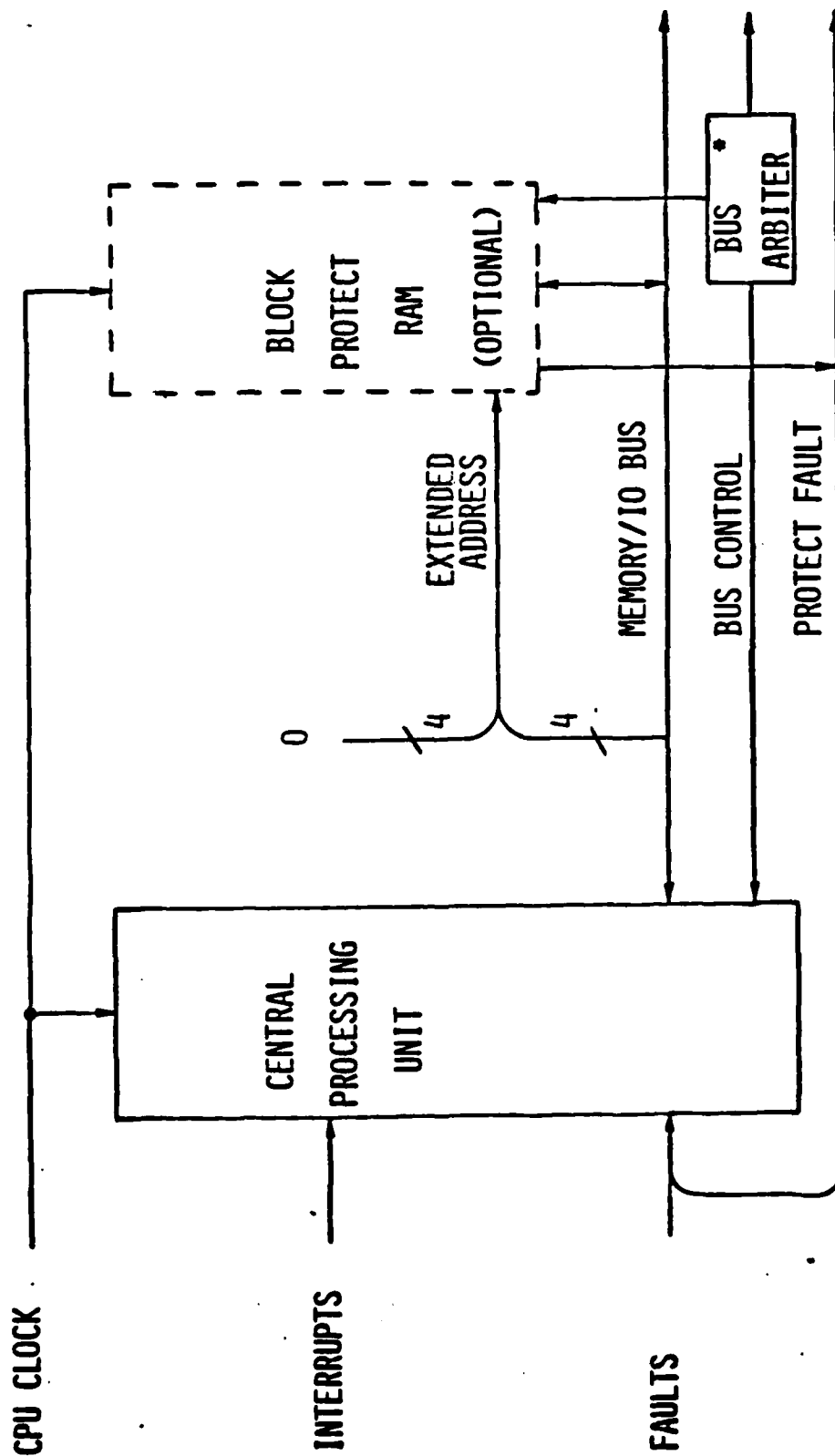
Chart # 2

CPU WITH OPTIONAL MMU AND BPR



* NOT INCLUDED IN CHIP SET

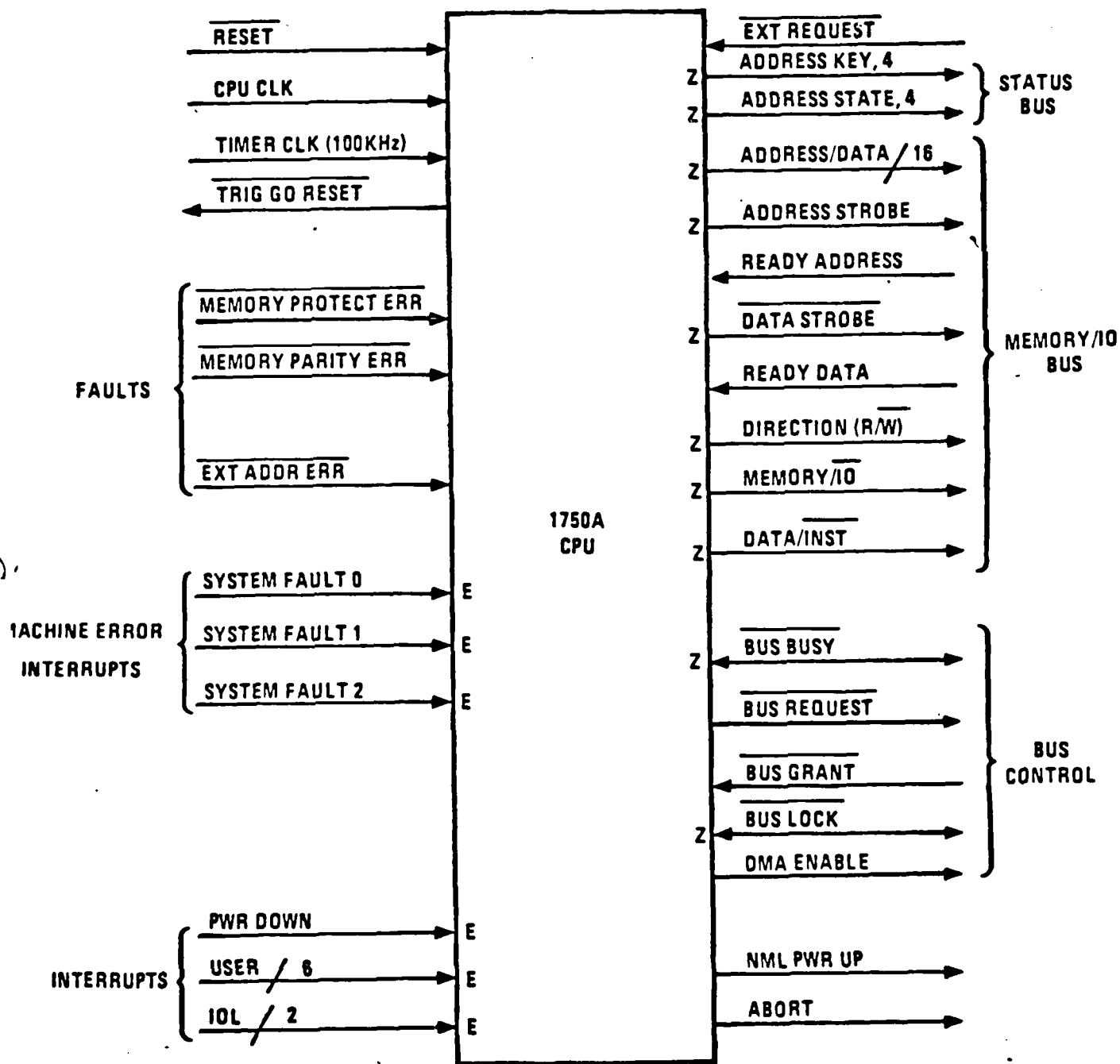
CPU WITH OPTIONAL BPR



• NOT INCLUDED IN CHIP SET

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 4



E = POSITIVE EDGE TRIGGERED
Z = 3-STATE

Figure 3 Central Processing Unit

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 5

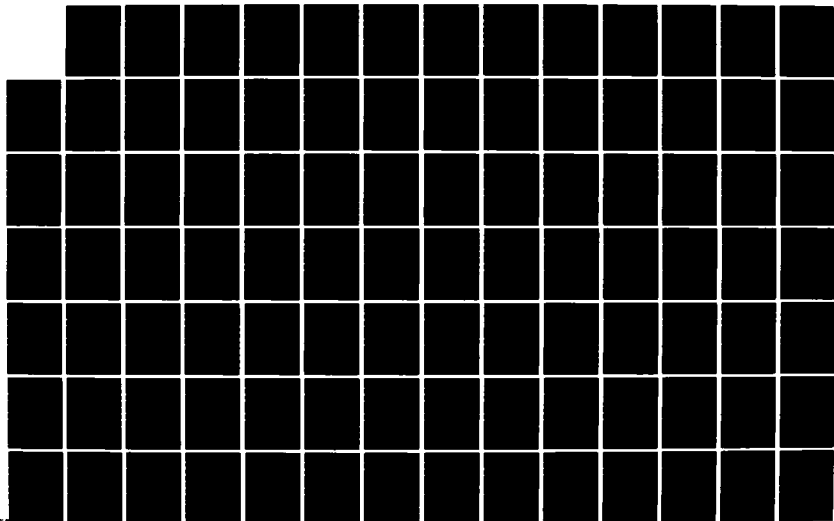
AD-A150 583

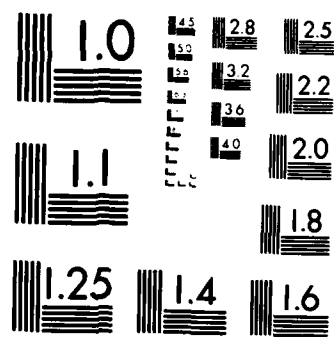
PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16
MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV
WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 06 MAY 82
ASD-TR-82-5011-VOL-1 F/G 9/2

3/6

UNCLASSIFIED

NL





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

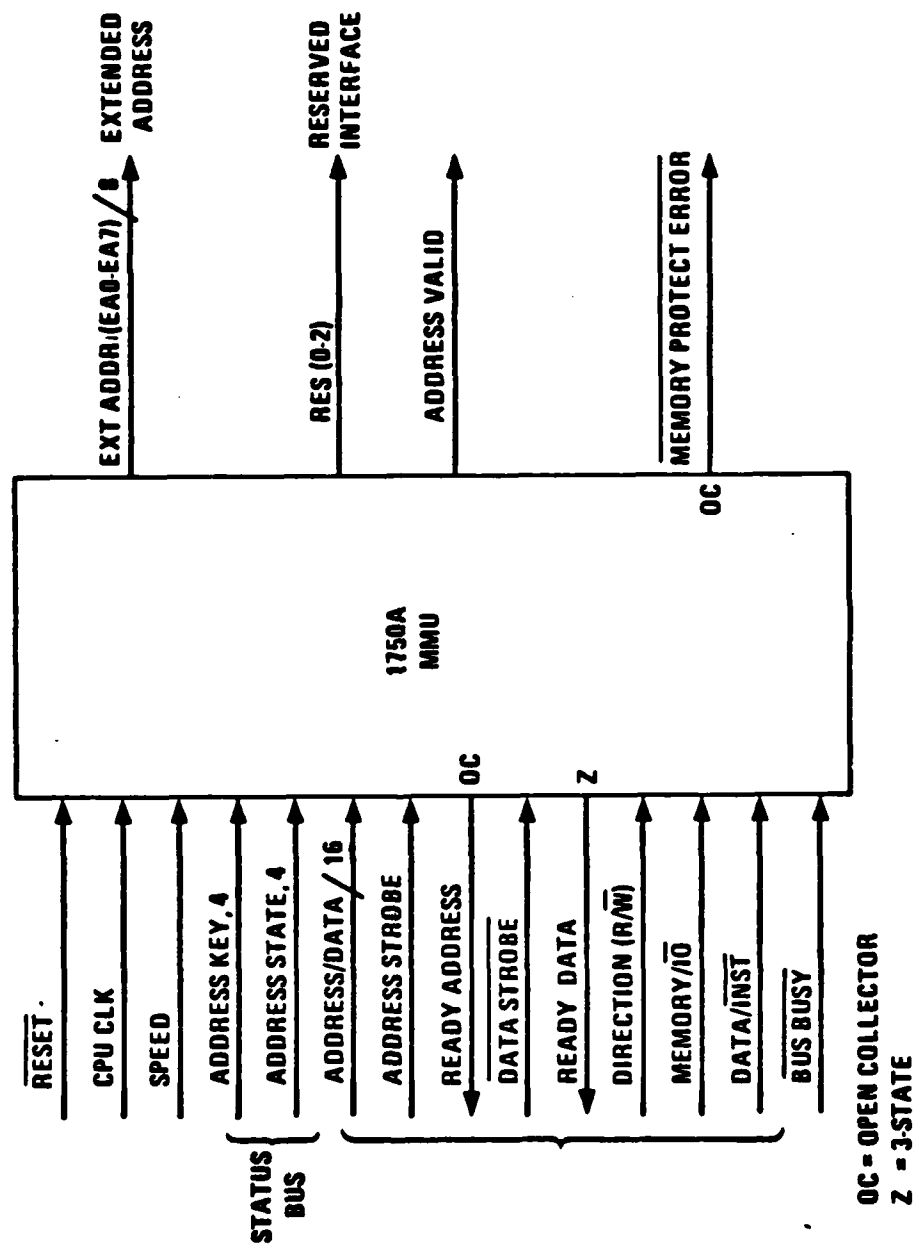


Figure 4 Memory Management Unit

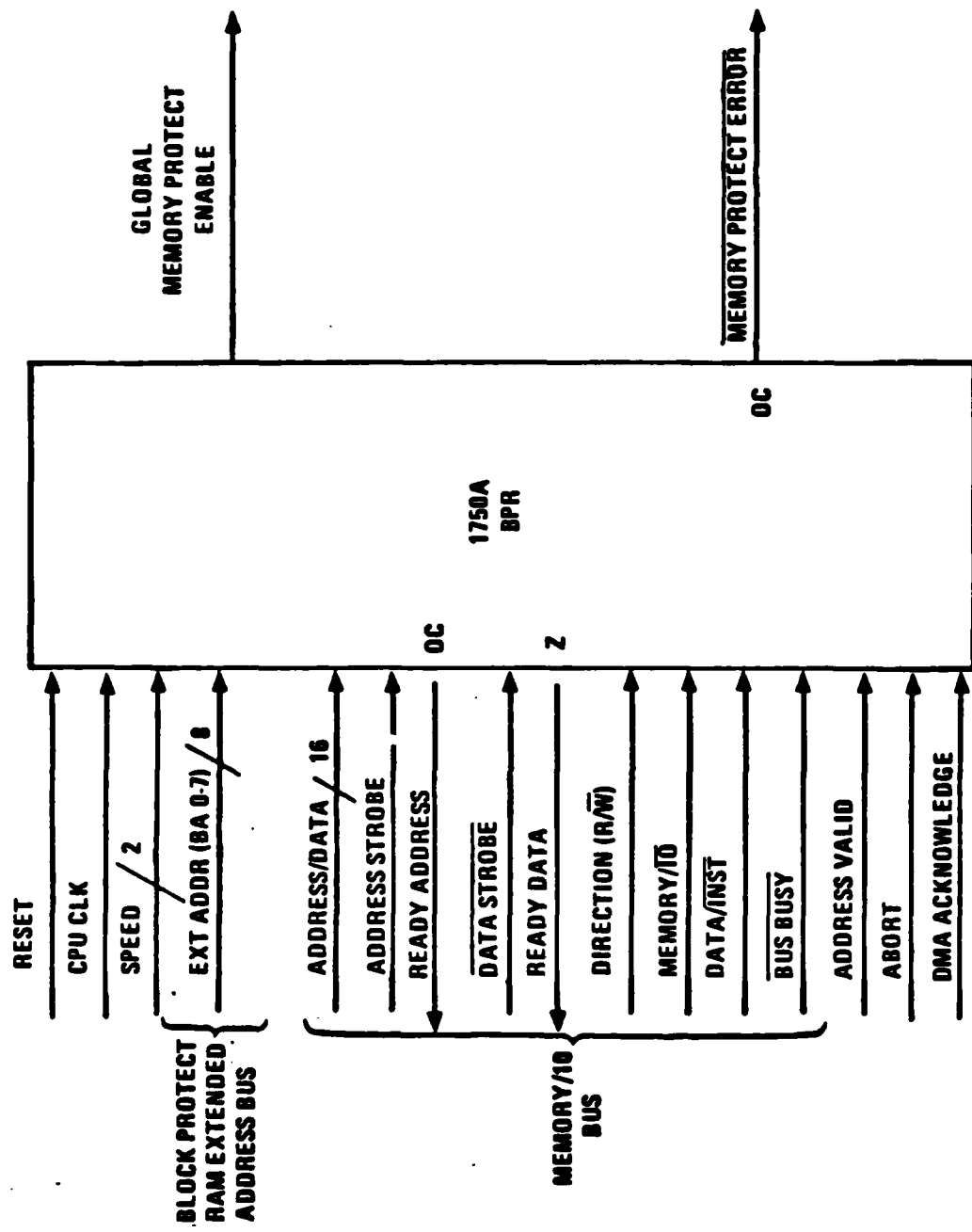


Figure 5 Block Protect RAM

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 7

ELECTRICAL INTERFACE
POWER REQUIREMENTS
SIGNAL CHARACTERISTICS

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 8

POWER REQUIREMENTS

	CPU	MMU ¹	BPR ²
+ 5 VOLTS	1.3 W	4.0 W	1.1 W
1.3 VOLTS	1.3 W	0.2 W	0.2 W

-55° C TO +125° C CASE TEMPERATURE

NOTES:

1. INCLUDES BIPOLAR RAMS AND BUFFERS
2. INCLUDES BIPOLAR RAM

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 9

ELECTRICAL SIGNAL CHARACTERISTICS

TYPE		INPUT	MICROAMPERES	VOLTS
3-STATE	UDI	LOGIC '0'	800	< 0.8
		LOGIC '1'	40	> 2.0
	UDO	OUTPUT		
		LOGIC '0'	8000	< 0.5
		LOGIC '1'	400	@ 2.4
		OFF	20	0 < V < 5

UDI = UNIDIRECTIONAL INPUT

UDO = UNIDIRECTION

OC = OPEN COLLECTOR

H. C. Lynn
Third Technical Forum
5 May 1982

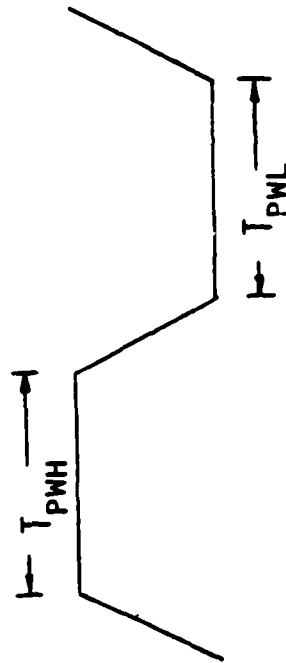
Chart # 10

CLOCKS & TIMERS

- o CPU MASTER CLOCK
- o TIMER REFERENCE
- o TRIGGER GO COUNTER

CPU CLOCK

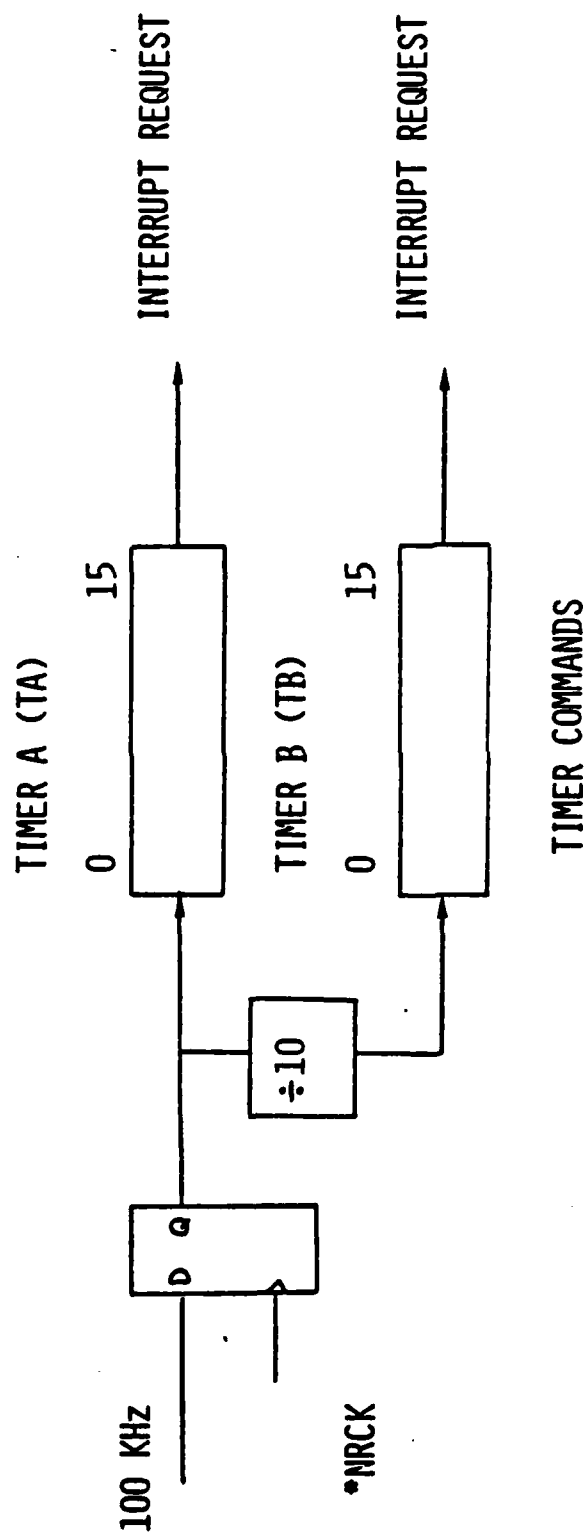
- o CPU LOGIC DESIGN IS STATIC
- o INTERNAL STATE CHANGES OCCUR ON LOW TO HIGH TRANSITIONS ONLY
- o 20 MHZ MAX OPERATING FREQUENCY



- o $T_{PWH} = T_{PWL} = 20 \text{ NANOSECONDS}$

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 12

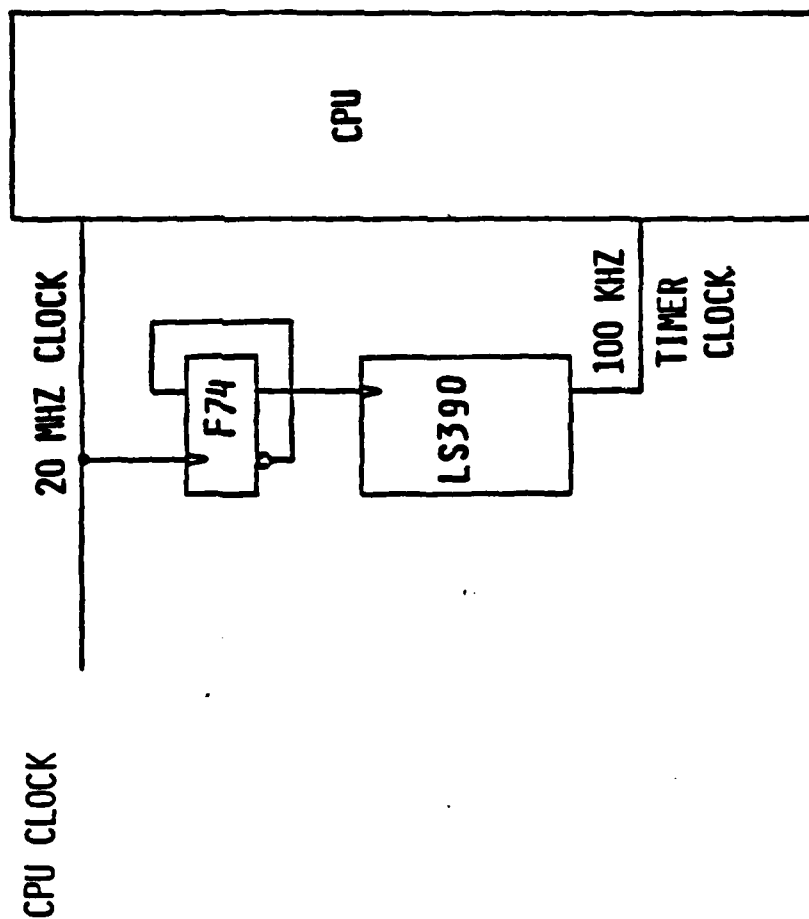


*NOTE: TIMER CLOCK IS SAMPLED AT THE BEGINNING OF CPU CYCLE TO INSURE THAT CONTENTS OF TIMER REGISTERS WILL BE SETTLED BY THE TIME THEY ARE GATED ONTO THE INTERNAL BUS

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 13

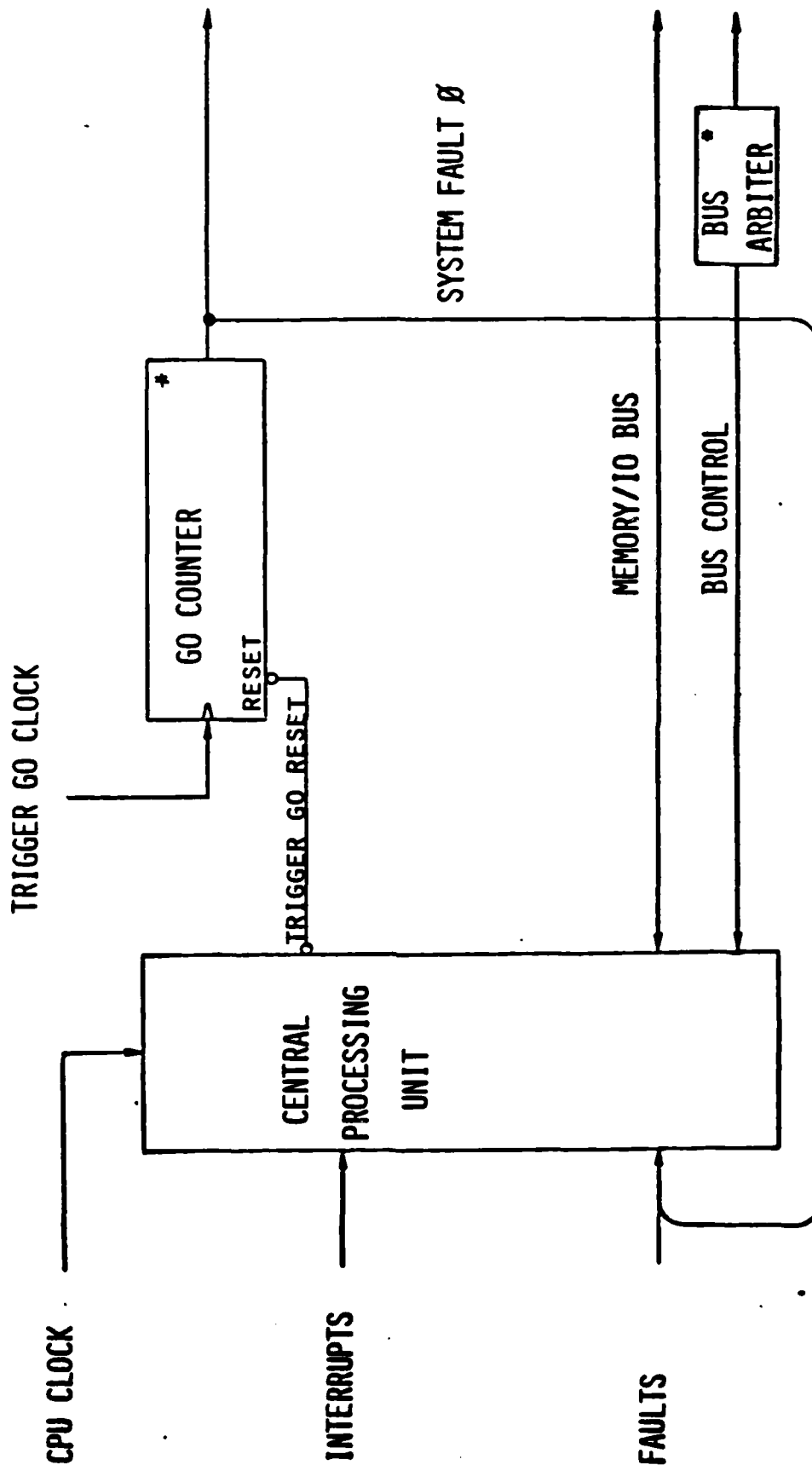
TIMER CLOCK



H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 14

GO COUNTER (OPTIONAL)



• NOT INCLUDED IN CHIP SET

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 15

FAULTS

CHIP SET FAULT DETECTION

HOST SYSTEM FAULT DETECTION

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 16

CHIP SET FAULT DETECTION

CPU WILL DETECT

ILLEGAL INSTRUCTIONS

PRIVILEGED INSTRUCTION ERRORS

BUILT-IN TEST ERRORS

ADDRESS STATE FAULTS

MMU (IF USED) WILL DETECT

MEMORY PROTECTION VIOLATIONS

BPR (IF USED) WILL DETECT

MEMORY WRITE PROTECT VIOLATIONS

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 17

HOST SUBSYSTEM FAULT DETECTION

MANDATORY FAULT DETECTION

ILLEGAL MEMORY ADDRESSES

ILLEGAL XIO COMMANDS

OPTIONAL FAULT DETECTION

MEMORY PARITY ERROR

SOFTWARE FAULT TIMEOUT

SYSTEM BITE ERRORS

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 18

INTERRUPTS (EXTERNAL)

0 MACHINE ERROR (3)

0 USER DEFINED (6)

0 EXPANSION (2)

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 19

INTERRUPT CHARACTERISTICS

- o ASYNCHRONOUS EVENTS OCCURRING OUTSIDE OF CPU
- o DETECTED ON LOW-TO-HIGH TRANSITION OF SIGNAL INPUT
- o RECEIVED BY HYSTERISIS INPUTS FOR EXTRA NOISE IMMUNITY

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 20

MACHINE ERROR INTERRUPTS

- o THREE ASYNCHRONOUS EDGE-TRIGGERED INPUTS TO FAULT REGISTER
- o INTERRUPTS 'OR' WITH OTHER ERROR CONDITIONS TO FORM MACHINE ERROR INTERRUPT

INTERRUPT NAME	FAULT REGISTER BIT NUMBER	INTERRUPT LEVEL
SYSTEM FAULT 0	7	1
SYSTEM FAULT 1	13, 14	1
SYSTEM FAULT 2	13, 15	1

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 21

USER DEFINED INTERRUPTS

- o SIX ASYNCHRONOUS EDGE-TRIGGERED INPUTS TO PENDING INTERRUPT REGISTER

INTERRUPT NAME	INTERRUPT LEVEL
USER 0	2
USER 1	8
USER 2	10
USER 3	11
USER 4	13
USER 5	15

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 22

EXPANSION INTERRUPTS

- o TWO ASYNCHRONOUS EDGE-TRIGGERED INPUTS TO PENDING INTERRUPT REGISTER
- o USAGE SHOULD CONFORM TO MIL-STD-1750 PARAGRAPH 4.4.2.6 (INPUT/OUTPUT INTERRUPT CODE REGISTERS)

INTERRUPT NAME	INTERRUPT LEVEL
IOL 1	12
IOL 2	14

BUS CONTROL

GENERALIZED BUS ARBITER

CONTROL SYNCHRONIZATION

SIMPLE BUS ARBITER

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 25

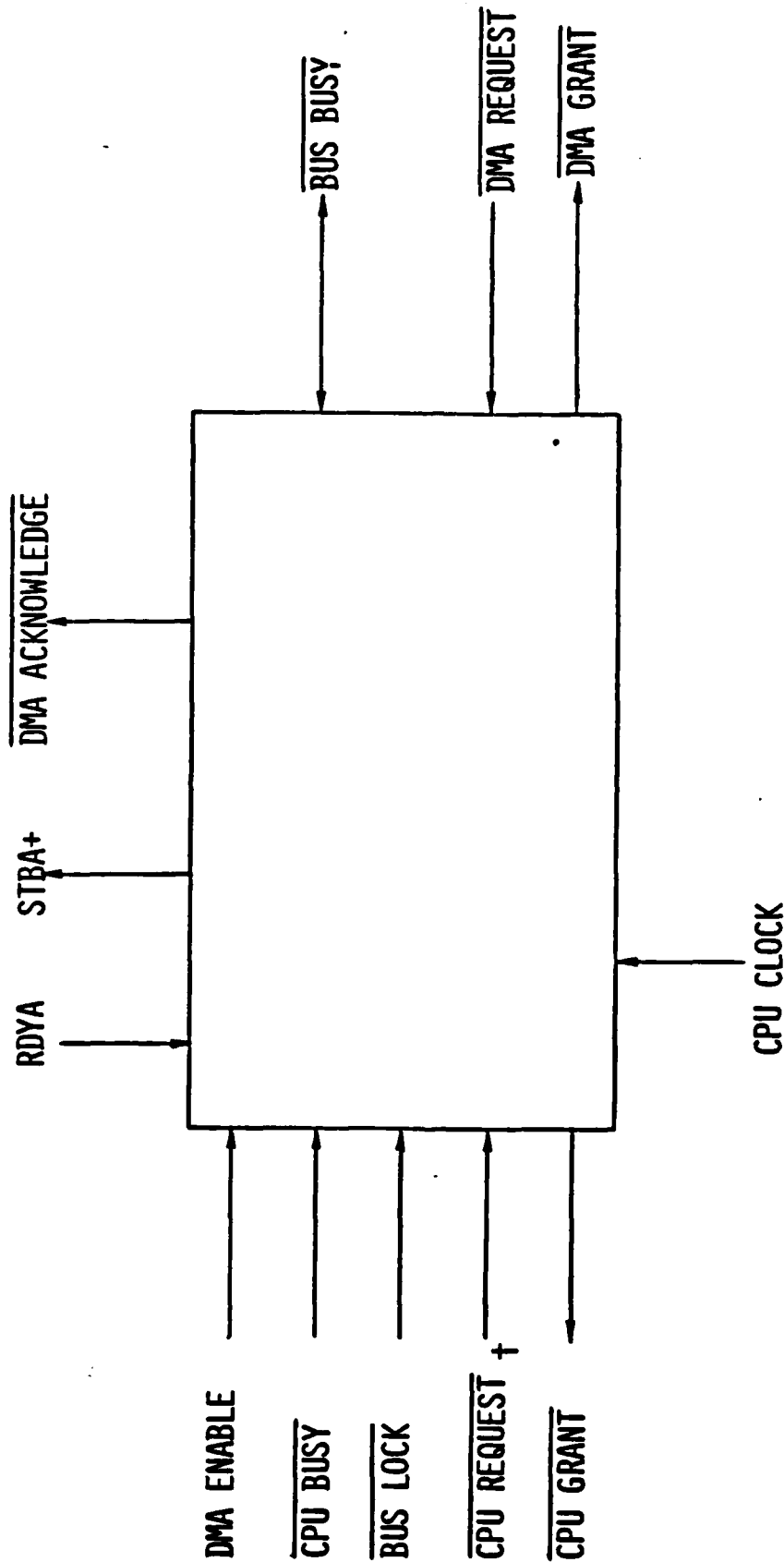
BUS ARBITER FUNCTIONS

- o CONTROL ACCESS TO GLOBAL MEMORY/IO BUS
- o SYNCHRONIZE CONTROL SIGNALS FOR USE BY MMU & BPR
- o SYNTHESIZE ADDRESS STROBE FOR NON-MULTIPLEXED BUS APPLICATIONS

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 26

GENERALIZED BUS ARBITER
(NOT INCLUDED IN CHIP SET)

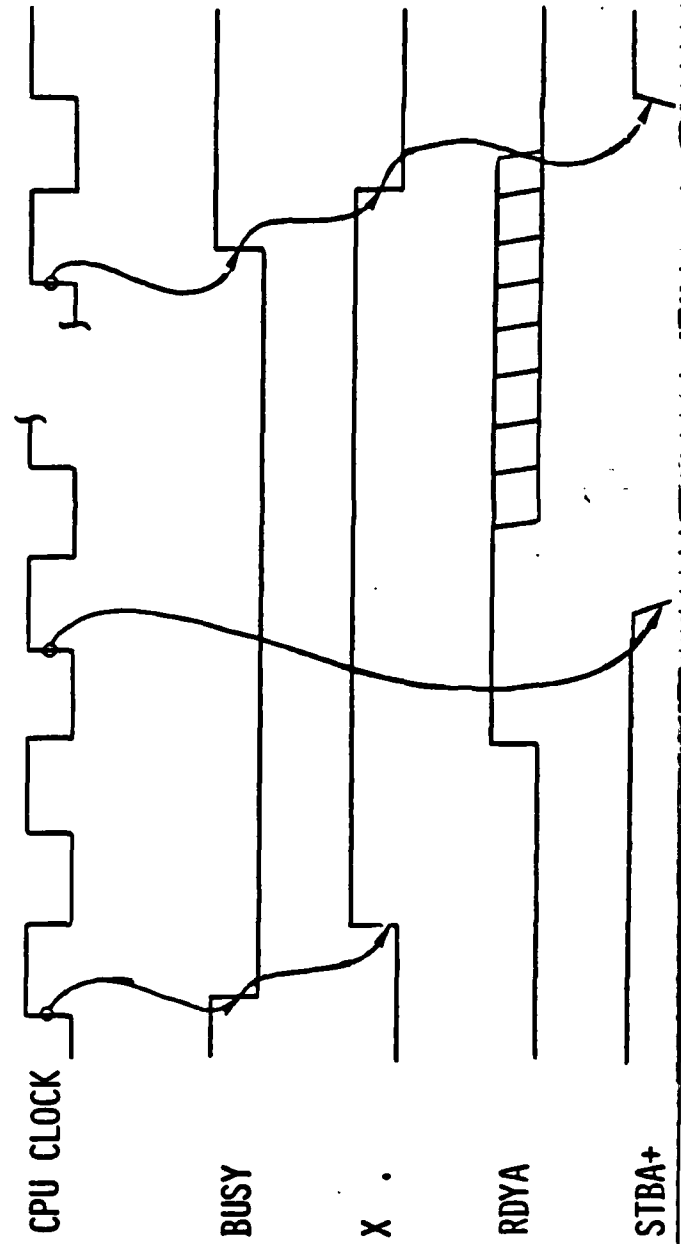
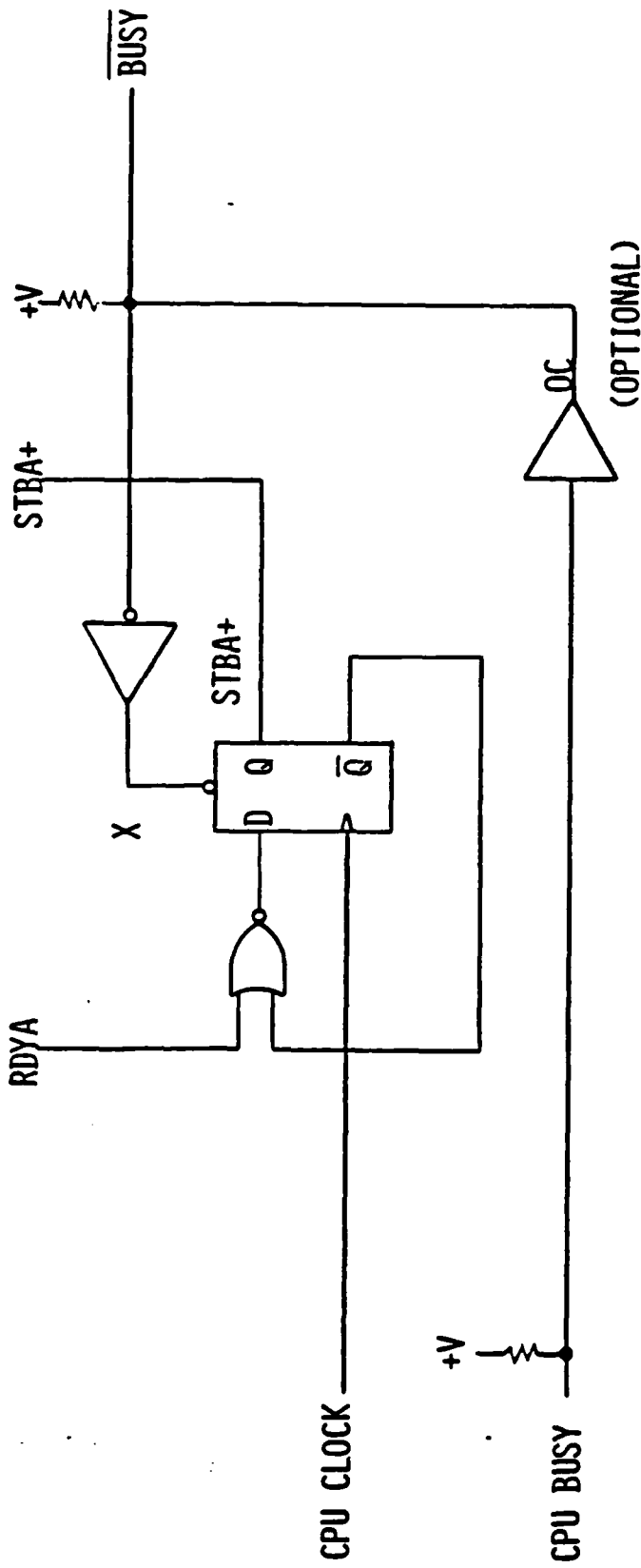


- + MULTI-PROCESSOR OPERATION ONLY
- + SYNCHRONIZED BY CPU CLOCK

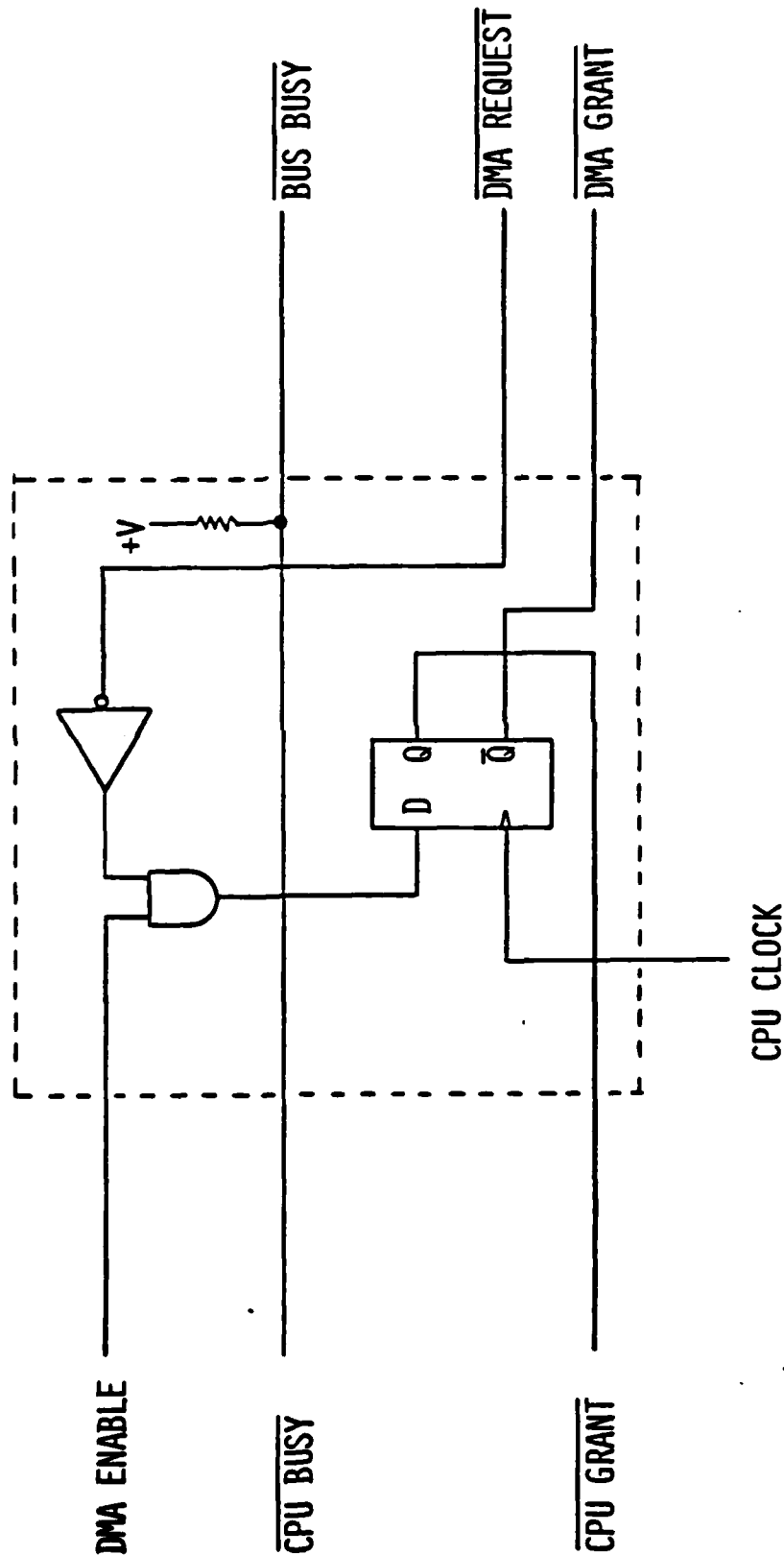
H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 27

EXTERNAL CONTROL SYNCHRONIZATION (NOT INCLUDED IN CHIP SET)



SIMPLE BUS ARBITRATOR (CPU ONLY)
(NOT INCLUDED IN CHIP SET)



H. C. Lynn
Third Technical Forum
5 May 1982

Chart 129

The diagram illustrates the internal logic of a DMA controller. It features several key components and signal paths:

- Inputs:** CPU CLOCK, INTERRUPTS, FAULTS, and a $\overline{\text{BUSY}}$ signal from the MMU/BPR.
- Logic Elements:** The circuit includes two D-type flip-flops (labeled P and Q), a D flip-flop (labeled D), and various combinational logic gates (AND, OR, NOT).
- Control Signals:**
 - STBA+:** A strobe signal derived from the CPU clock and $\overline{\text{BUSY}}$.
 - ADDRESS/DATA BUS (00-15):** A 16-bit bus connected to the MMU/BPR.
 - RDYA, ADDRESS STROBE, RDYD, DATA STROBE:** Signals for address and data bus operations.
 - DIR (R/W), MEMORY/IO, DATA/INST:** Direction and data type signals.
 - ABORT, DMA ENABLE:** Control signals for aborting or enabling DMA.
- Outputs:**
 - DMA ACK:** Acknowledgment signal to the MMU/BPR.
 - DMA BUS ENABLE, DMA BUS REQUEST:** Signals for DMA bus operations.
 - LOCK, BUSY:** Status signals for bus locking and busy state.
- Connections:** The controller is connected to the MMU/BPR (via TO and DMA lines) and the DMA device (via TO, DMA, and DEVICE lines).

Chart # 30

MEMORY/IO BUS INTERFACE

- 0 STATUS
- 0 CONTROL
- 0 ADDRESS DECODING
- 0 MEMORY INTERFACE INCLUDING DMA

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 31

STATUS LINE CONVENTIONS

CPU		<u>M/I</u>	<u>R/W</u>	<u>D/I</u>
	INSTRUCTION FETCH	H	H	L
	DATA READ	H	H	H
	DATA WRITE	H	L	H
	XIO READ (INTERNAL) *	L	L	H
	XIO WRITE (INTERNAL)	L	L	H
	XIO READ (EXTERNAL)	L	H	H
	XIO WRITE (EXTERNAL)	L	L	H
DMA	DATA READ	H	H	H
	DATA WRITE	H	L	H

* WRITE STATUS

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 32

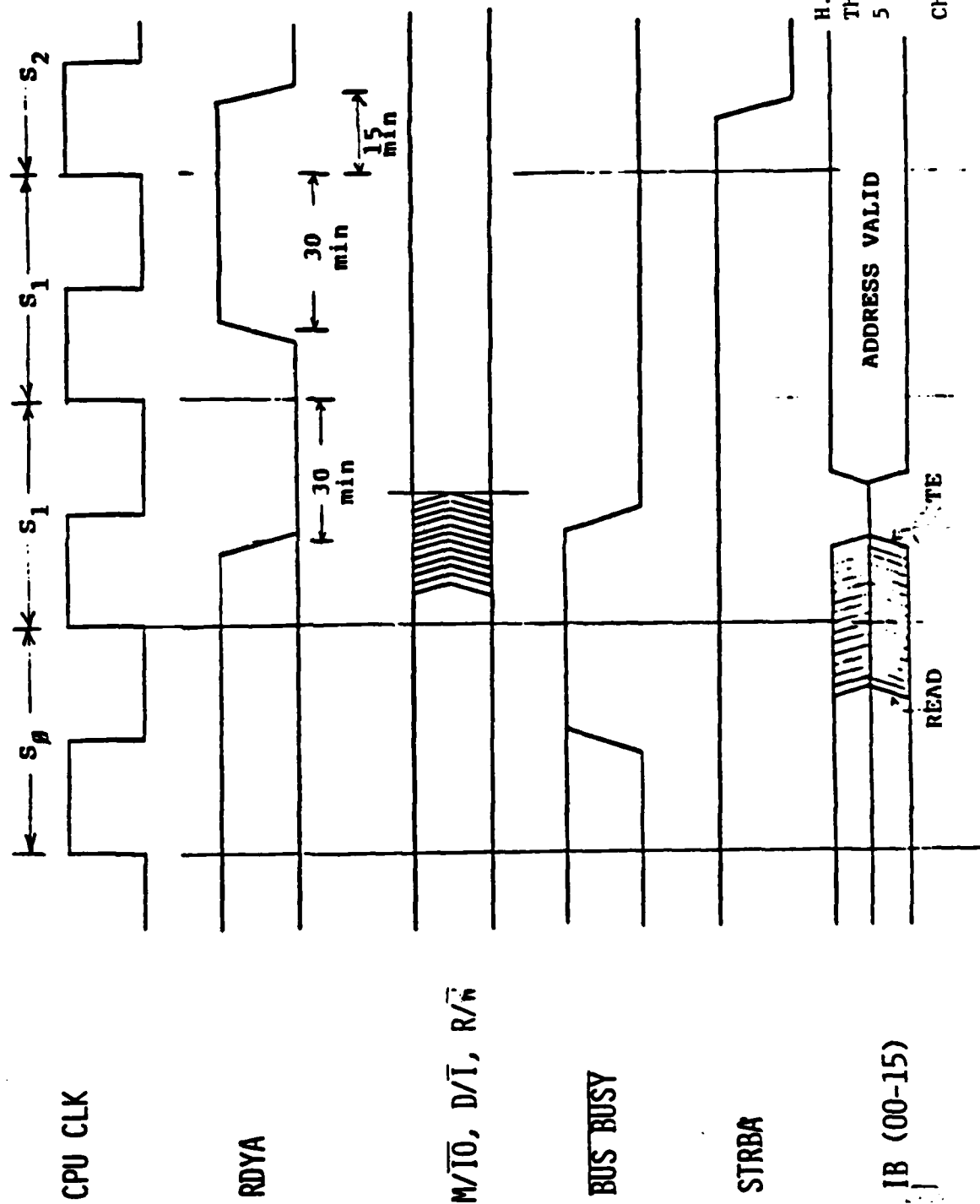
F9450A CPU

CONTROL

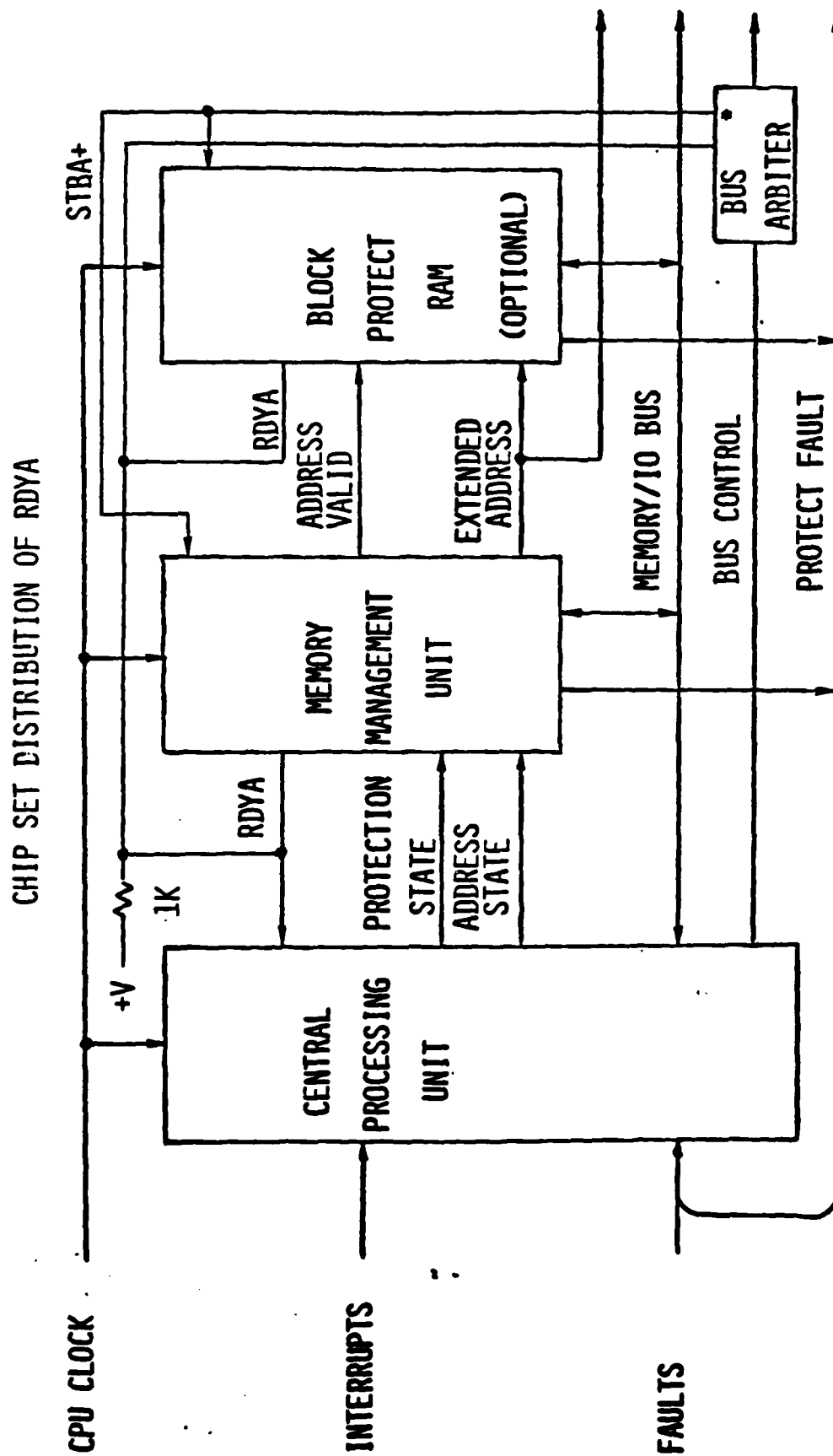
- * THE USE OF "RDYA" SIGNAL.
- * MIN. READ BUS CYCLE.
- * THE USE OF "RDYD" SIGNAL.
- * MIN. WRITE BUS CYCLE.
- * WARNING !

H. C. Lynn
Third Technical Forum
5 May 1982

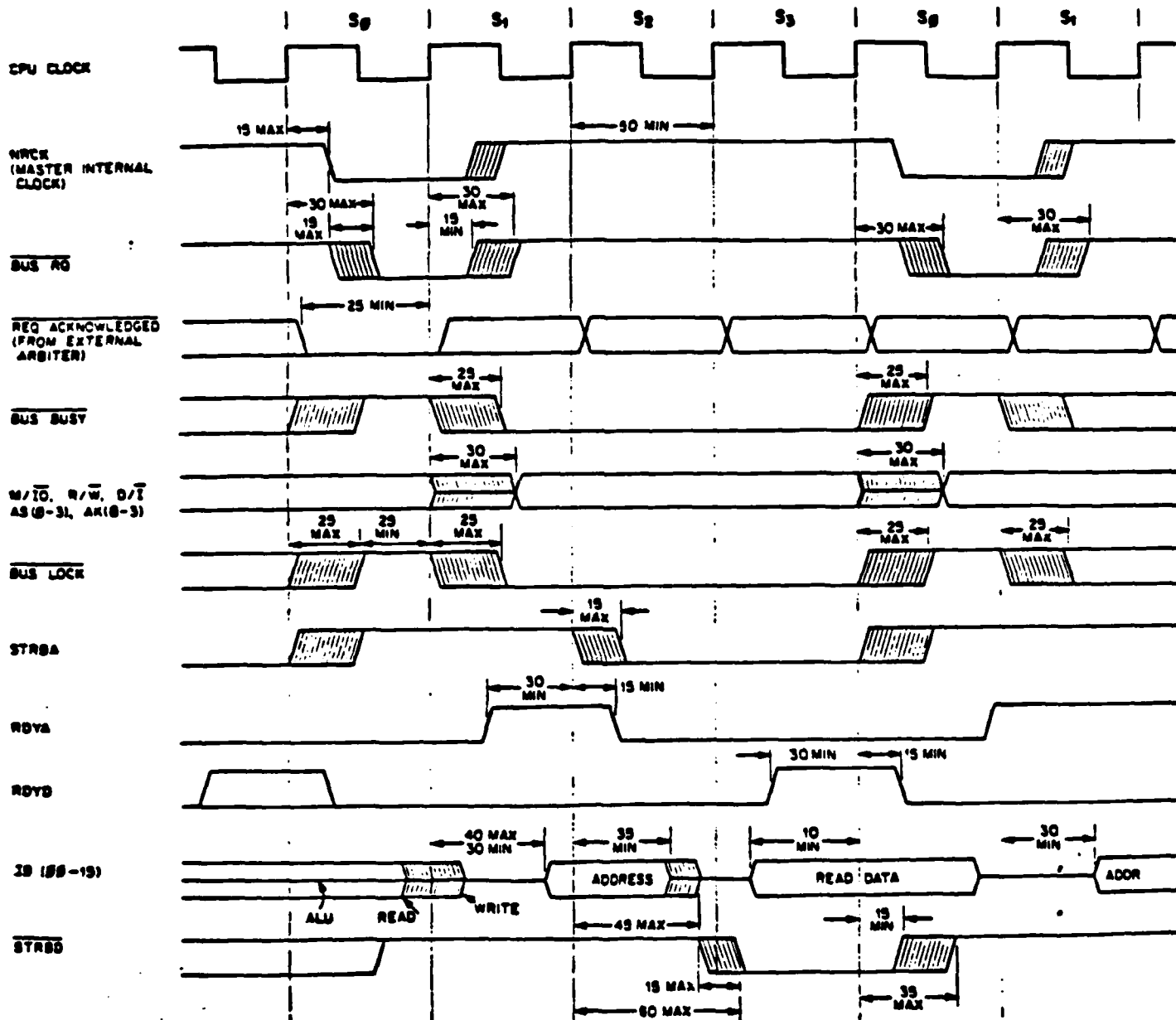
Chart # 33



H. C. Lynn
Third Technical Forum
5 May 1982
Chart # 34

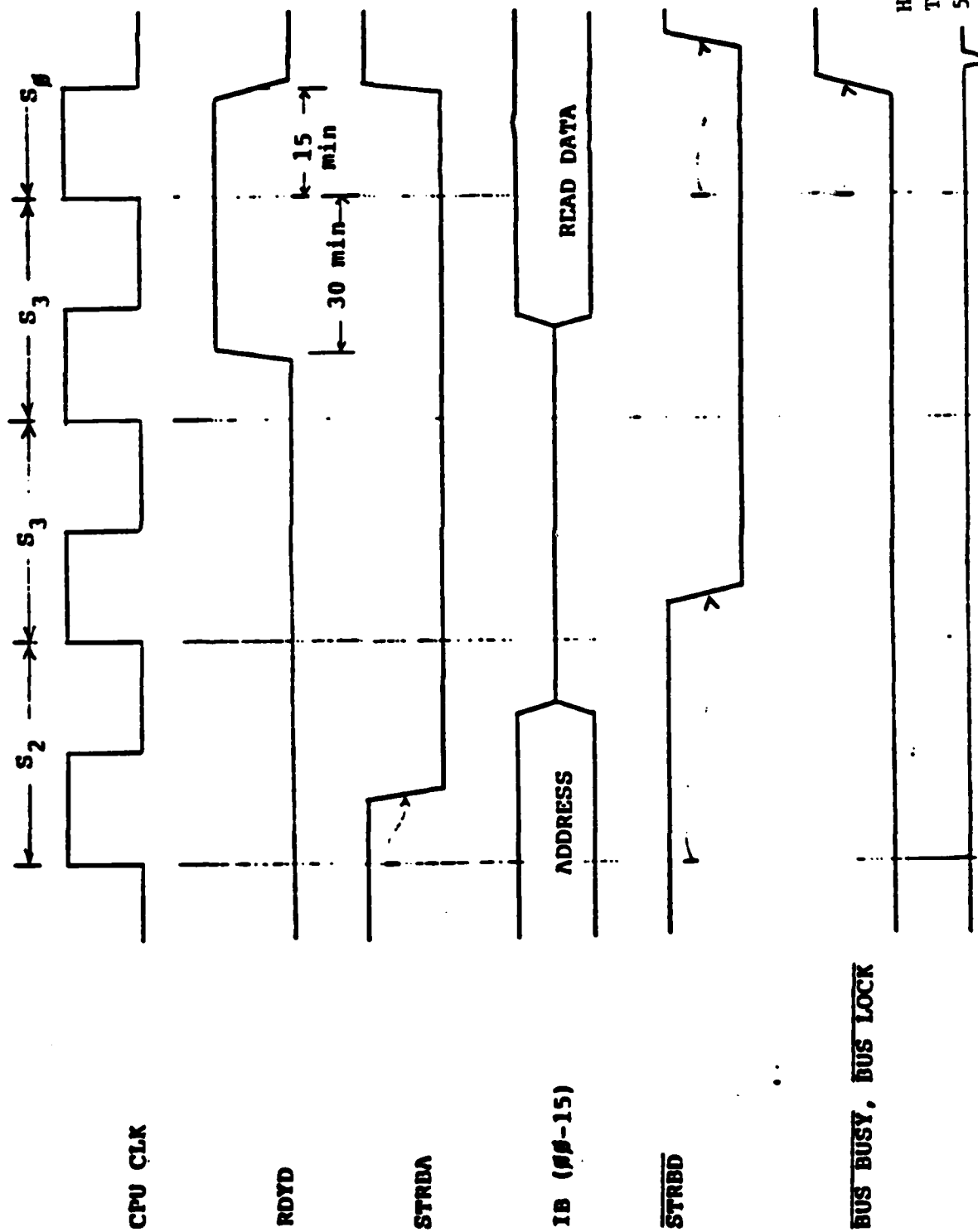


F8450 (1750A) TIMING
MINIMUM READ BUS CYCLE



H. C. Lynn
Third Technical Forum
5 May 1982

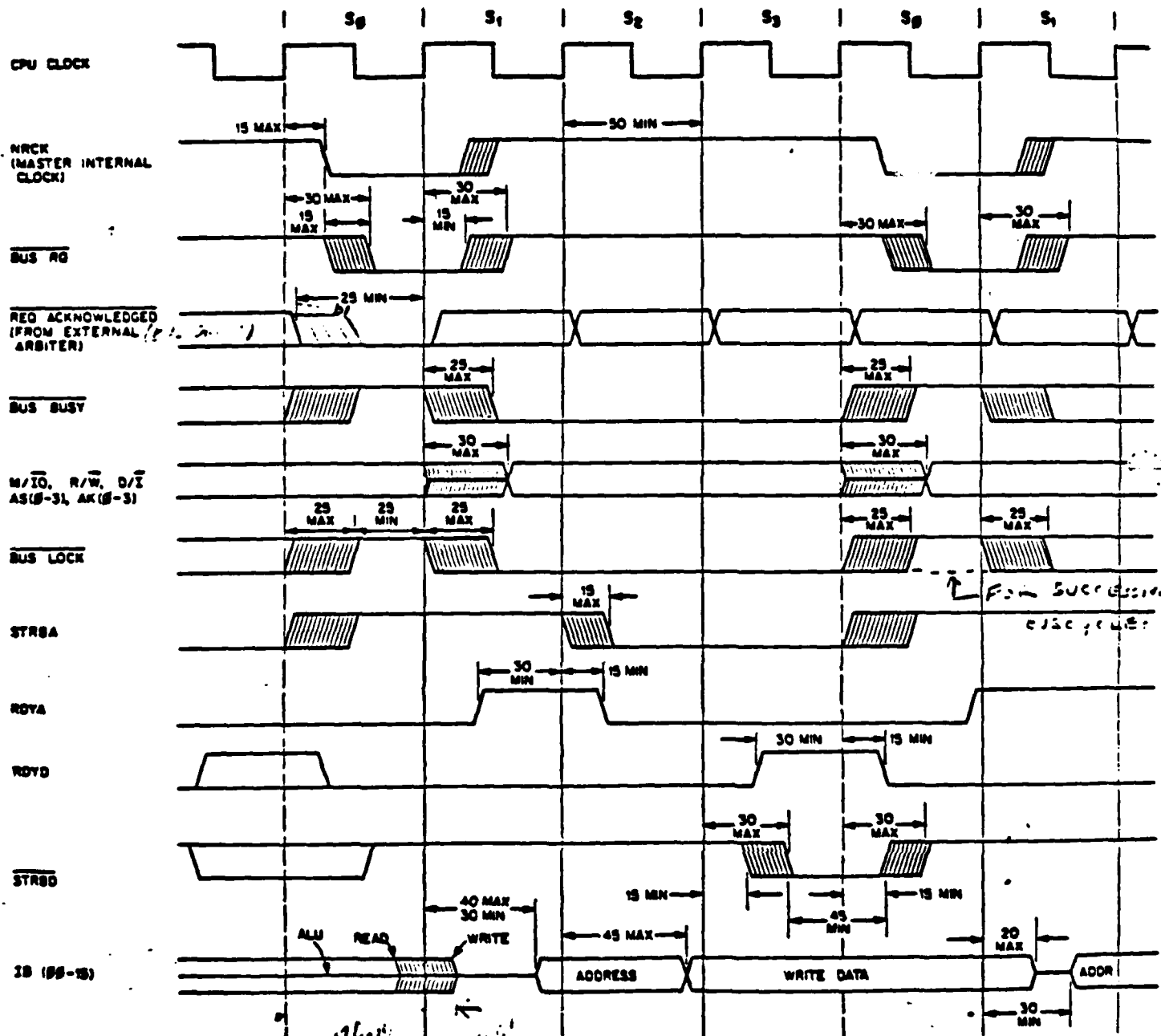
Chart # 36



H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 37

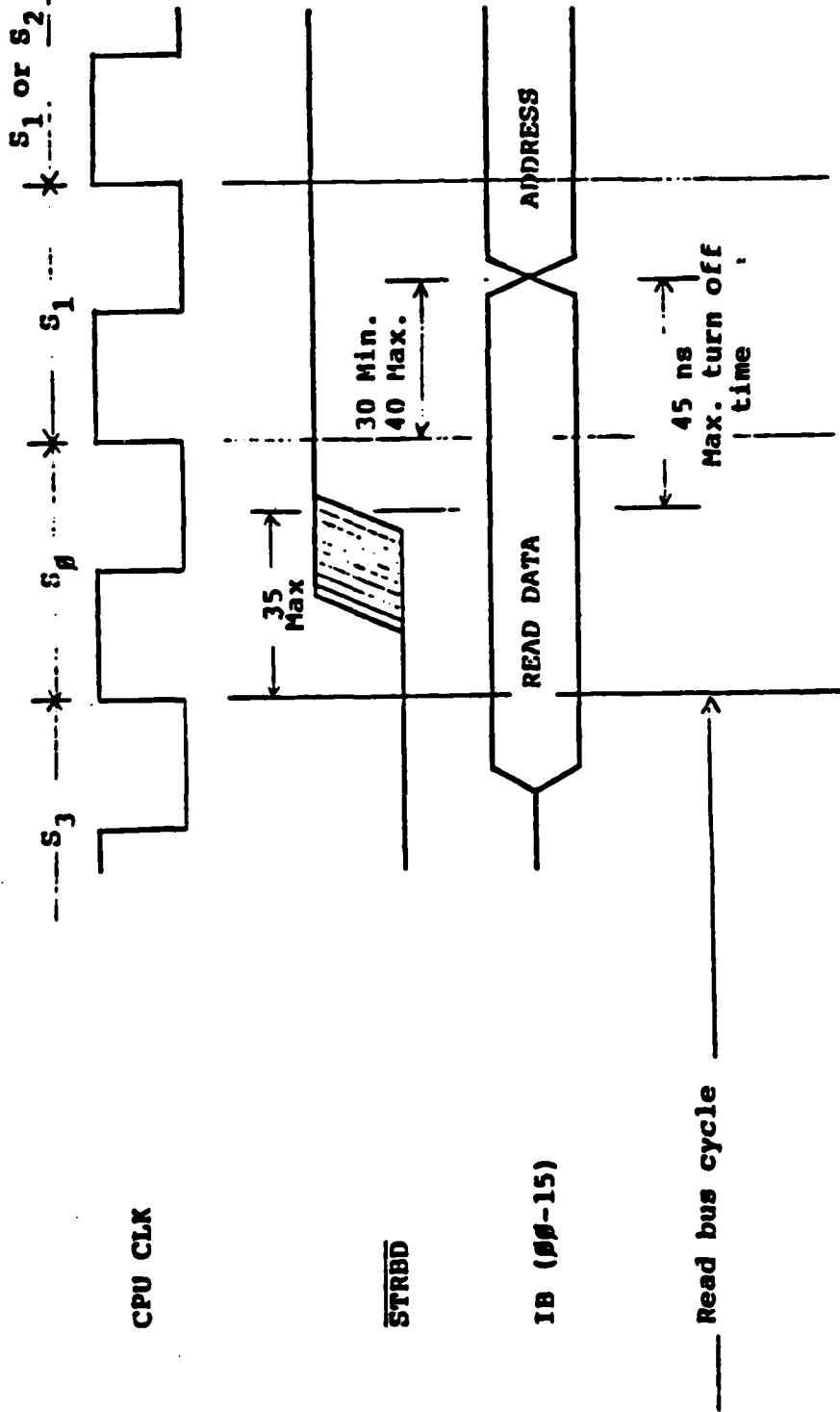
**F9450 (1750A) TIMING
MINIMUM WRITE BUS CYCLE**



H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 38

WARNING :



F9450A

A SINGLE CPU DOING A READ BUS CYCLE

FOLLOWED BY A SECOND BUS CYCLE

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 39

ADDRESS DECODING

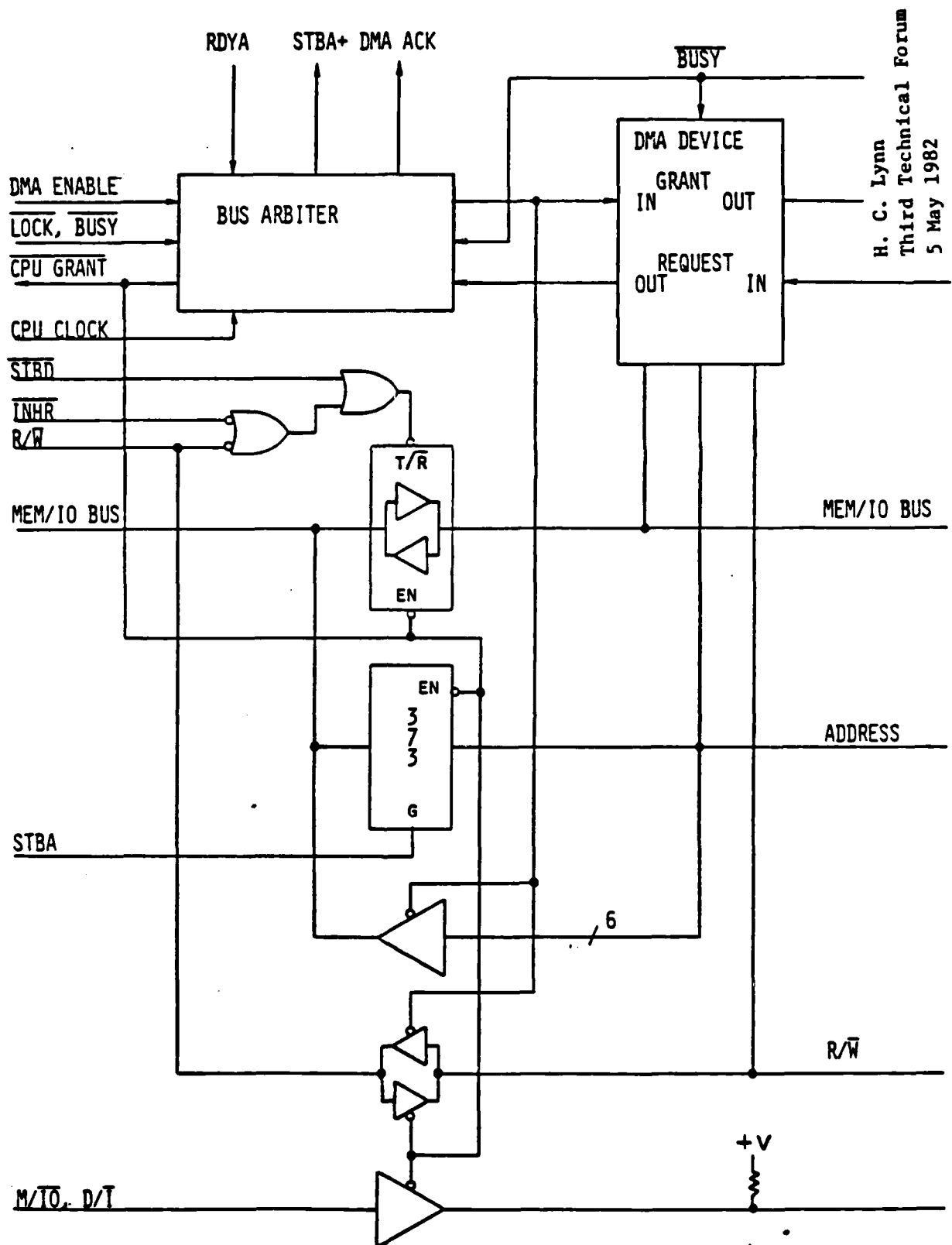
- o XIO COMMANDS ALWAYS TAKEN FROM ADDRESS BROADCAST AT BEGINNING OF DATA TRANSFER CYCLE

- o MEMORY ADDRESS MOST SIGNIFICANT BITS MUST BE TAKEN FROM MEMORY MANAGEMENT UNIT (IF PRESENT)

- o ADDRESS ERRORS REPORTED TO CPU VIA ILLEGAL ADDRESS FAULT DISCRETE

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 40



H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 41

MEMORY INTERFACE INCLUDING DMA

MEMORY SYSTEM DESIGN

ACCESS/CYCLE TIMES

WRITE PROTECTION

ERROR CONTROL

MEMORY FAULT STATUS REGISTER

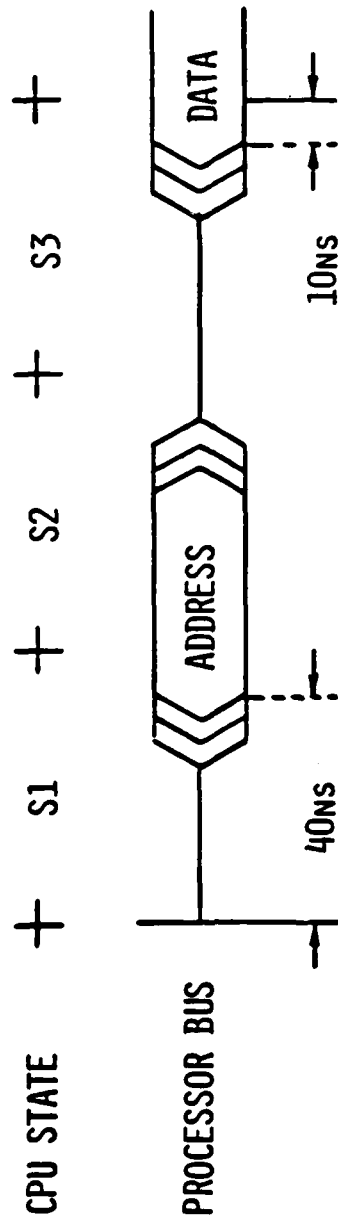
START-UP ROM

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 42

* SYSTEM MEMORY ACCESS/CYCLE TIMES

WITHOUT MMU



$$T_{ACCESS}^{**}(NS) = \frac{(3 + N) \times 10^3}{F (MHZ)} - 50$$

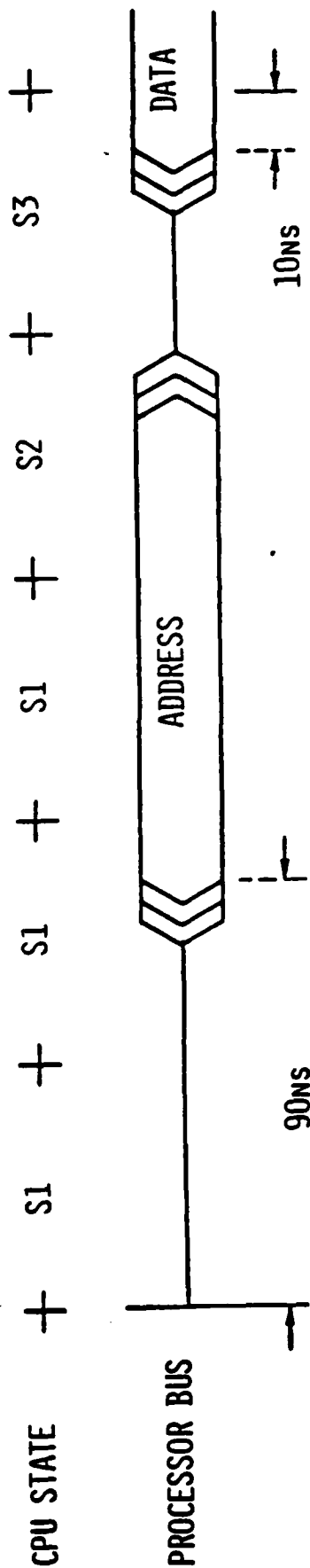
$$T_{CYCLE}^{**}(NS) = \frac{(4 + N) \times 10^3}{F (MHZ)}$$

* INCLUDES DELAYS OF ADDRESS LATCHES, DECODERS AND BUFFERS AS WELL AS MEMORY ACCESS TIME.

** N IS THE NUMBER OF WAIT STATES DUE TO RDYA OR RDYD; F (MHZ) IS THE CPU CLOCK FREQUENCY IN MEGAHERTZ.

* SYSTEM MEMORY ACCESS TIMES

WITH MMU



$$T_{ACCESS\_MMU} = \frac{(5 + 2xB + N) \times 10^3}{F \text{ (MHz)}} \sim 100$$

$$T_{CYCLE\_MMU} = \frac{(6 + 2xB + N) \times 10^3}{F \text{ (MHz)}}$$

- * INCLUDES DELAYS OF ADDRESS LATCHES, DECODERS AND BUFFERS AS WELL AS MEMORY ACCESS TIME
- ** B IS A BOOLEAN VARIABLE WHICH IS TRUE IF A 4K PAGE BOUNDARY IS CROSSED; N IS THE NUMBER OF WAIT STATES DUE TO RDYA OR RDYD; F (MHz) IS THE CPU CLOCK FREQUENCY IN MEGAHERTZ.

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 44

CPU CLOCK (MHZ)	CONFIG- URATION	SYSTEM MEMORY ADDRESS ACCESS/CYCLE TIME ** (NS)				
		NO WAIT*	ONE WAIT*	TWO WAIT*	THREE* WAIT	FOUR* WAIT
20	CPU	100/200	150/250	200/300	250/350	300/400
	CPU + MMU	150/300	200/350	250/400	300/450	350/500
	CPU	138/250	200/313	263/375	325/438	388/500
16	CPU + MMU	213/375	275/438	338/500	400/563	463/625
	CPU	250/400	350/500	450/600	550/700	650/800
	CPU + MMU	400/600	500/700	600/800	700/900	800/1000

F9450A

SYSTEM MEMORY ACCESS/CYCLE TIME REQUIREMENTS

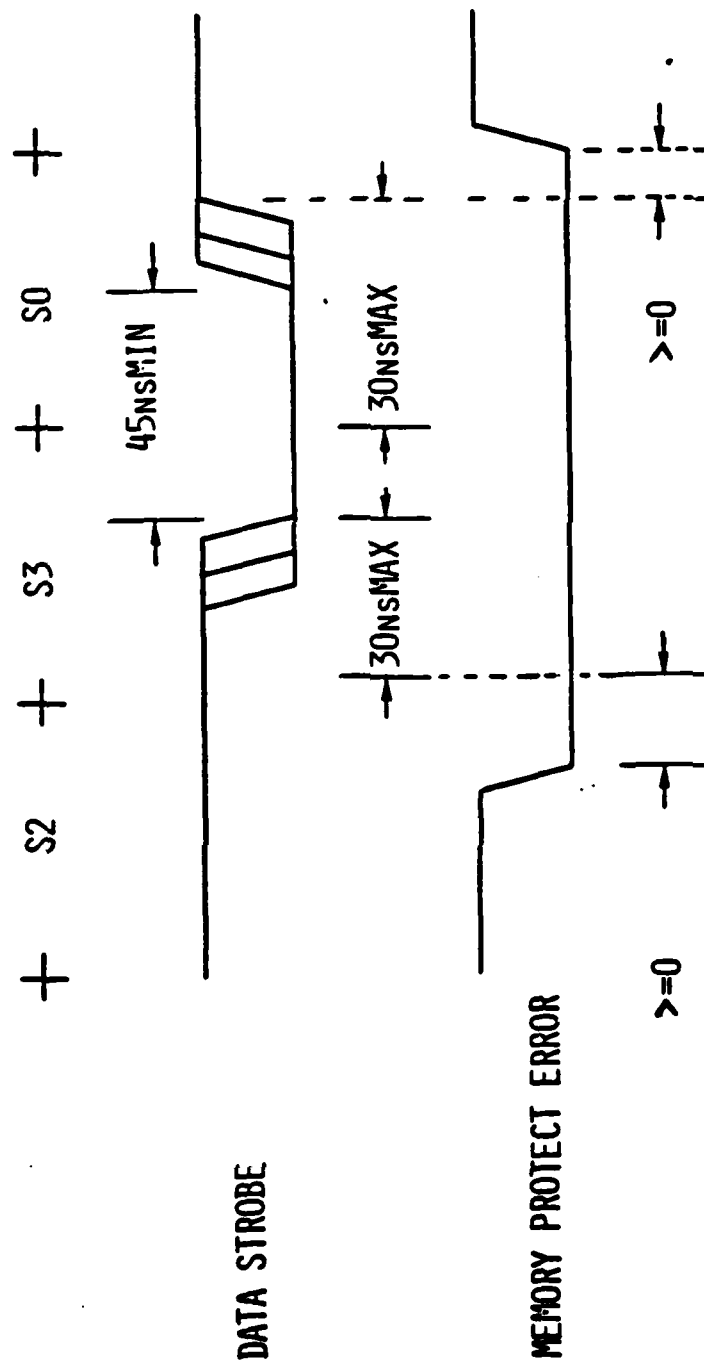
* A WAIT STATE IS INSERTED DUE TO EITHER RDYA OR RDYD NOT BEING ACTIVE WHEN SAMPLED BY THE CPU AT THE PROPER TIME.

** SYSTEM MEMORY ADDRESS ACCESS TIME INCLUDES ADDRESS LATCHES AND ADDRESS DECODERS DELAYS.

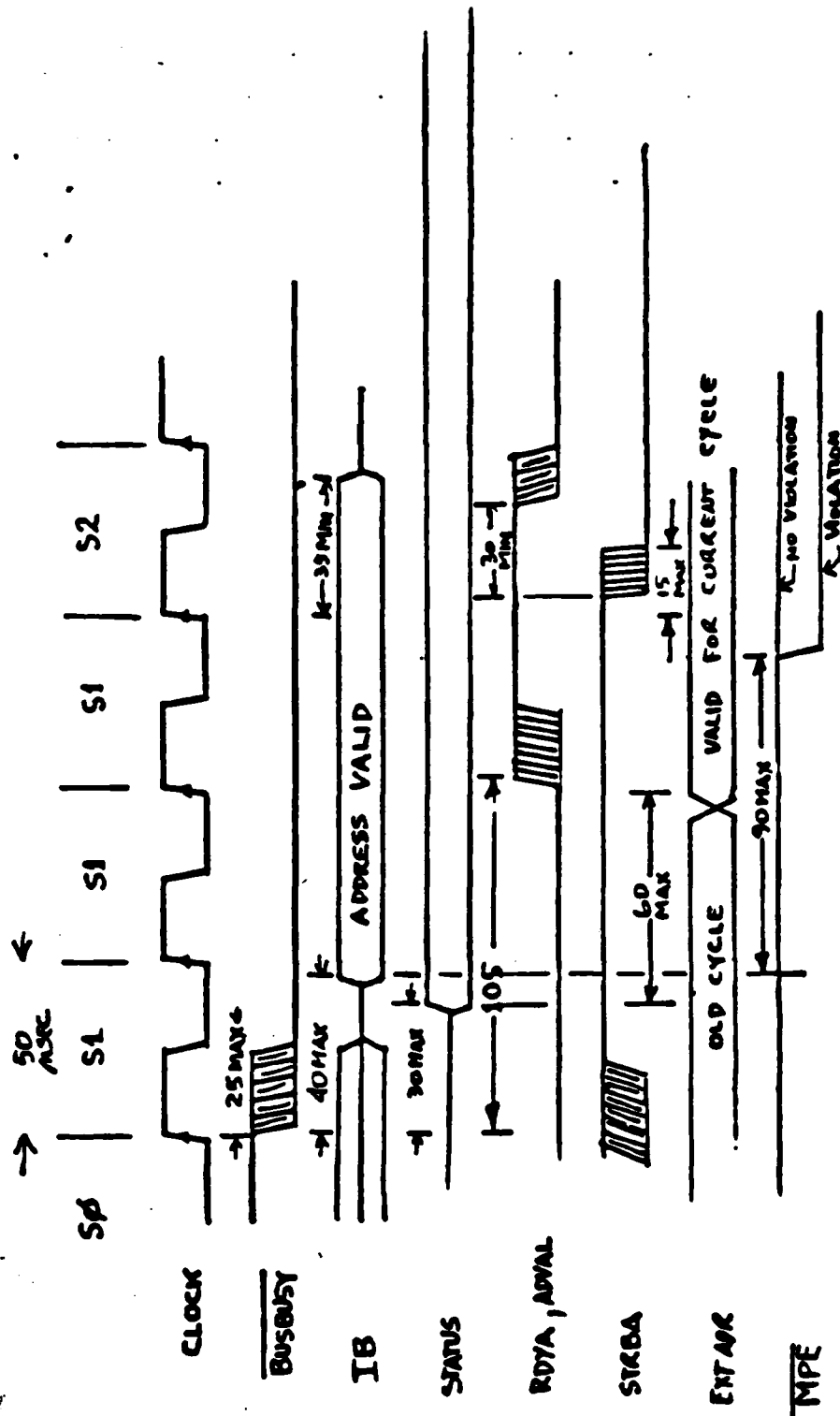
H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 45

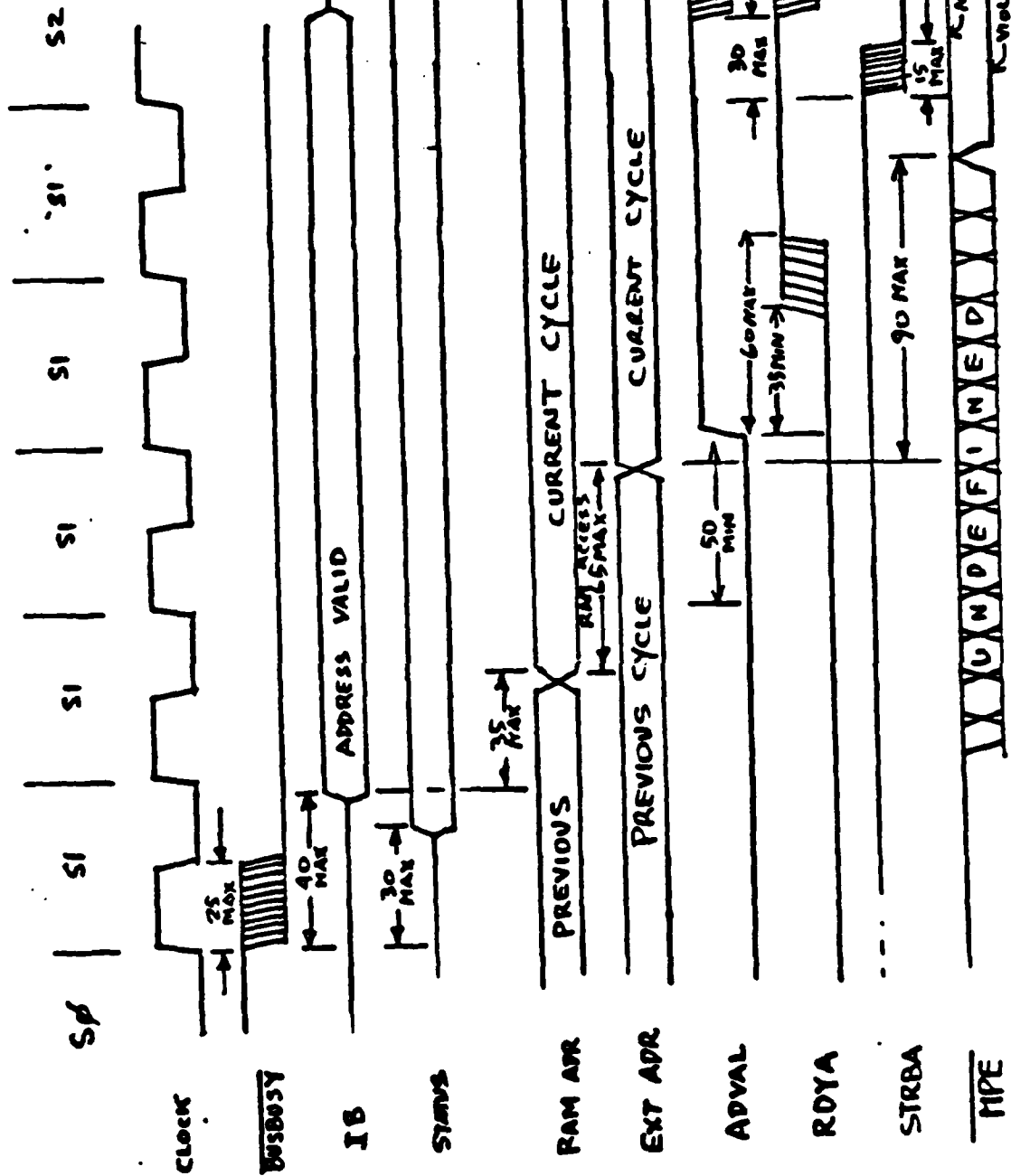
MEMORY WRITE PROTECTION



WRITE DATA STROBES MUST BE QUALIFIED BY MEMORY PROTECT ERROR TO
INSURE DATA INTEGRITY



F9451 MMU MIN. TIMING (PAGE BOUNDARY NOT CROSSED)
AT 20 MHz, CLOCK RATE



F9451 MMU TIMING (PAGE BOUNDARY CROSSED)

H. C. Lynn
Third Techn... 1 Forum
5 May 1982

AT 20 MHz

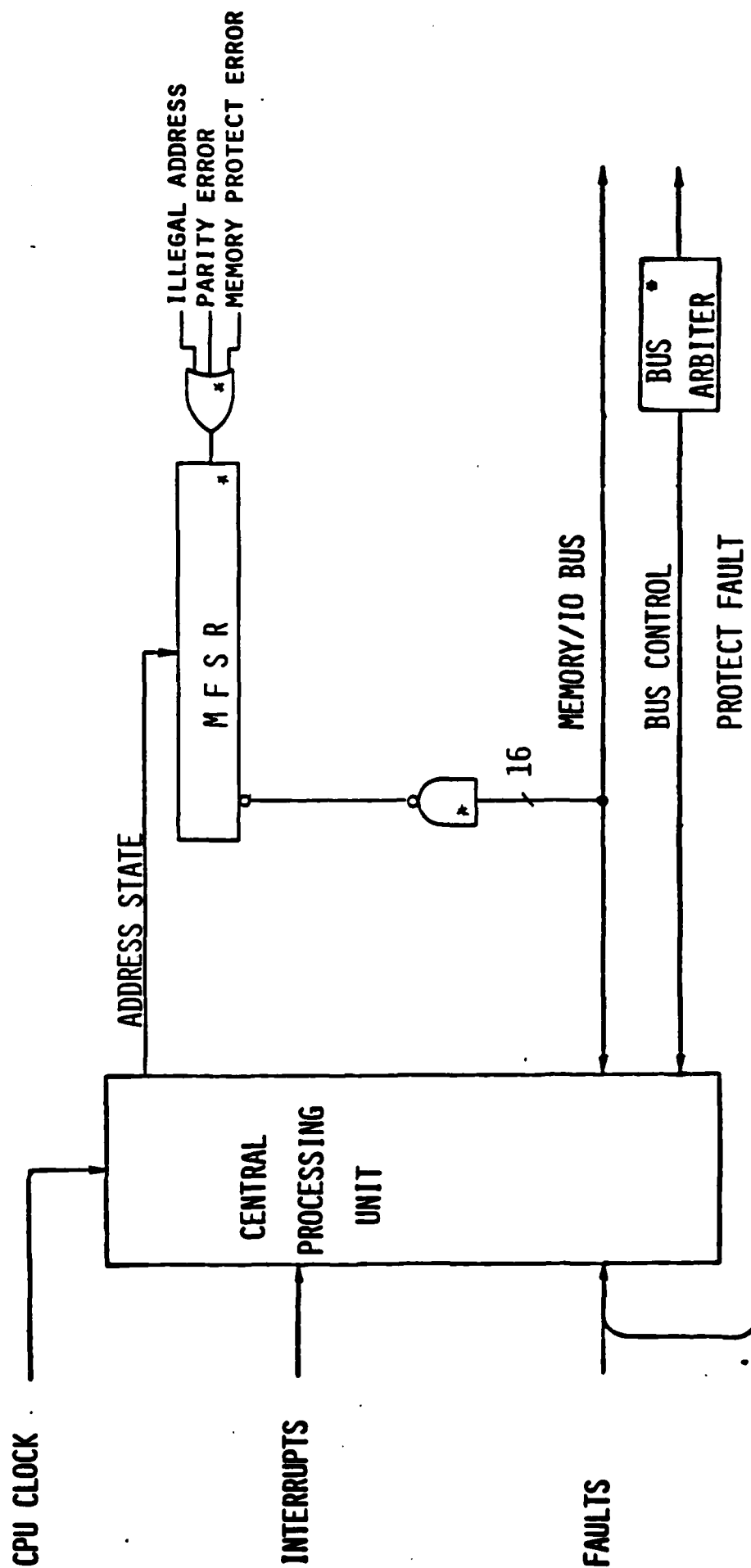
MEMORY SYSTEM ERROR CONTROL

- o ERROR DETECTION IS DONE EXTERNAL TO CHIP-SET
- o CPU SAMPLES MEMORY PARITY ERROR FAULT DISCRETE AT CONCLUSION OF EACH MEMORY READ CYCLE
- o IF DISCRETE IS ACTIVE LOW AT LOW TO HIGH TRANSITION OF BUS BUSY, BIT 2 OF FAULT REGISTER IS SET TRIGGERING MACHINE ERROR INTERRUPT (IF NOT MASKED)
- o ERROR DETECTION SCHEMES INVOLVING MULTIPLE PARITY BITS PER WORD (EG. BYTE PARITY, HAMMING CODES) MUST COMBINE ERROR SIGNALS FOR FAULT REPORTING

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 49

MEMORY FAULT STATUS REGISTER (OPTIONAL)

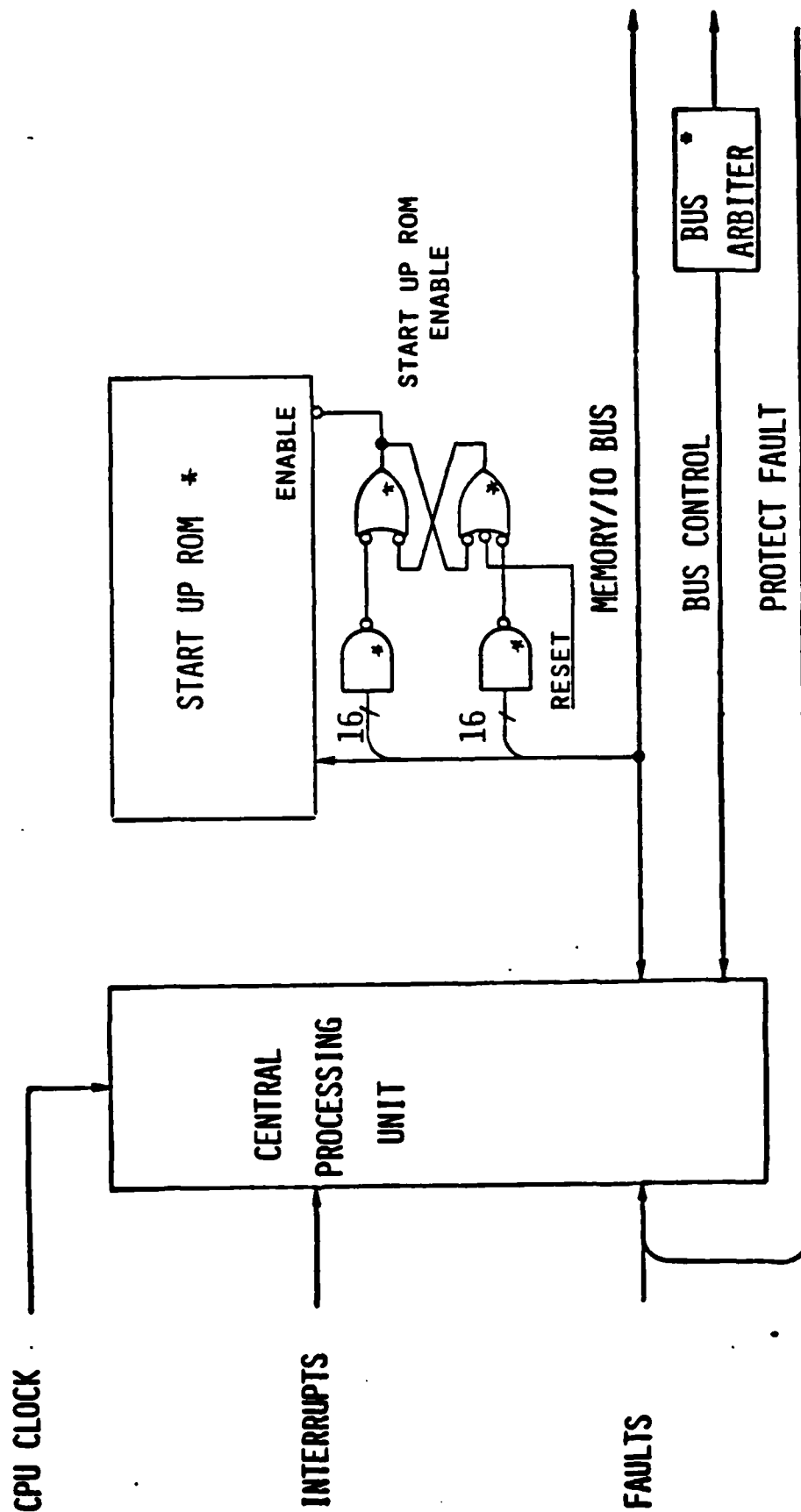


• NOT INCLUDED IN CHIP SET

H. C. Lynn
Third Technical Forum
5 May 1982
Chart # 50

0	3	4	10	11	12	15
LPA	RESERVED	IO	AS			

START UP ROM (OPTIONAL)



• NOT INCLUDED IN CHIP SET

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 51

IO SYSTEM DESIGN

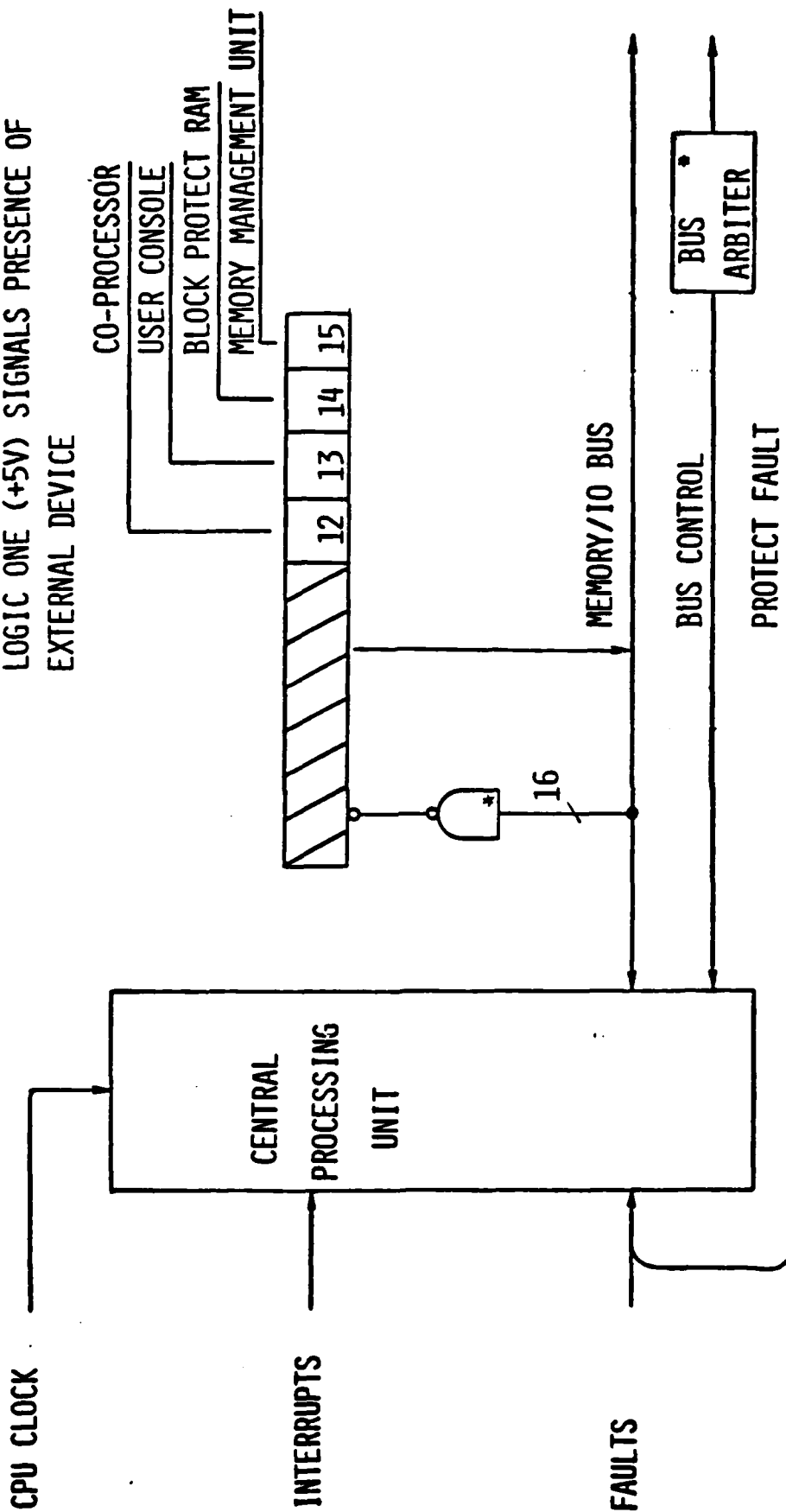
H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 52

INITIALIZATION WORD *

XIO ADDRESS: 8410

LOGIC ONE (+5V) SIGNALS PRESENCE OF
EXTERNAL DEVICE



* NOT INCLUDED IN CHIP SET

H. C. Lynn
Third Technical Forum
5 May 1982

Chart #53

F - 16 X10 PREFIX CODES

<u>DEVICE</u>	<u>ADDRESS RANGE</u>
CPU	2000 - 200F A000 - A00F 4000 - 400F C000 - C00F
BPR	5000 - 50FF D000 - D0FF
AMUX	3000 - 30FF B000 - B0FF
CONSOLE	0400 8400 - 8401
INIT	8410

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 54

SUMMARY

THE GD 16ZE181/MIL-STD-1750A MICROPROCESSOR

- o IS COMPATIBLE WITH ISSUES RESOLVED BY AF 1750A
CONTROL BOARD ON 20 JULY 1981
- o MEETS ALL F-16 REQUIREMENTS DESCRIBED IN
16PP379A, DATED 7 JULY 1981
- o FULFILLS THE OBJECTIVES OF ACSN ON PERFORMANCE,
SIZE, AND COST

H. C. Lynn
Third Technical Forum
5 May 1982

Chart # 55

F-16 MIL-STD-1750A MICROPROCESSOR DEVELOPMENT PROGRAM

MANAGEMENT OVERVIEW

RANDAL MOORE

PROGRAM OVERVIEW

DAN WILNAI

FUNCTIONAL AND TECHNICAL DESCRIPTION

SHAI MOR

APPLICATIONS HARDWARE

HENRY LYNN

► ENGINEERING TEST/EVALUATION EQUIPMENT

DAVID SCOTT

INTERIM PROCESSOR TRANSITION

HENRY LYNN

**THIRD TECHNICAL FORUM
WRIGHT-PATTERSON AFB**

5 MAY 1982

D. M. Scott

Third Technical Forum

5 May 1982

Chart # 0

D. M. SCOTT

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #1

We are going to examine the evaluation units that Fairchild will provide to General Dynamics under the microprocessor development contract. The units will be used to test the chip set, and evaluate it's potential for integration into the core avionics applications.

CHART #2

The test equipment will support these four tasks, and possibly a few more. It will support the console control that Shai discussed earlier. For instance, reading and writing to memory, reading and writing to the internal registers and the ability to run and halt the processor. The equipment will also be used to support SEAFAC validation, which I will describe in a little more detail. Validation will be accomplished by hosting the SEAFAC test load modules on a mainframe computer, and then downloading them to the FS1 discs. The load modules will then be run on the chip set one at a time to obtain the test results, and finally validate the microprocessor. I should mention that complying with the SEAFAC ATP is a part of our specification requirements. The FS1 will also be used to perform other testing under chip set contract--namely acceptance testing and qualification testing.

CHART #3

Fairchild will provide to Honeywell Avionics of Minneapolis, one of their subcontractors, a standard FS1 Fairchild System One which is a commercial minicomputer. It has a 9445 bit slice emulator, 64K of memory, two floppy disk drives, and an integrated CRT keyboard unit. It will be provided to General Dynamics with disk operating system software and other support software packages. There are several I/O ports for a link to a mainframe, to a printer and to the CRT keyboard.

CHART #4

Honeywell will design and build a test card that will plug into the motherboard of the FS1. This card will contain a bus interface to the FS1 Nova bus, and an interface to the chip set information bus, which contains the handshaking and protocol

D. M. SCOTT

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

signals. The card will contain some zero insertion force sockets, so we can easily test various chip sets. There will be jumper headers that we can use to bypass the MMU and BPR to allow testing any of the four configurations. The test card will also have 64K words of 16 bit memory, and the console interface I talked of earlier. The card will allow us to simulate the external processor I/O, such as DMA simulation, interrupt simulation and things of that nature. There will be test points for logic analyzers and scopes to aid in debugging any problems with the chip set.

CHART #5

Randal mentioned earlier that we are still defining the requirements for this piece of equipment. We haven't conducted a formal design review on the equipment so it is not in it's final form at this time. On the card that Honeywell is building will be a hardware control panel that will be used with the early runs of silicon, which might not be fully functional. The control panel will allow us to run and halt the processor and to single step through instructions, and if necessary individual clock cycles. The address/data bus will be displayed so that we can see what's happening in the CPU as it communicates with the other devices and with memory. There will be hardware breakpoint capability, which is still be formulated because of the pipelined architecture. There will not be any interrupt or DMA simulation, because on the early parts we will be concentrating on the internal workings of the chip set.

MIL-STD-1750A MICROPROCESSOR CHIP SET
TESTING PROGRAM AND EQUIPMENT

DAVID SCOTT
THIRD TECHNICAL FORUM, WPAFB
5 MAY 1982

D. M. Scott
Third Technical Forum
5 May 1982

Chart # 1

2

ENGINEERING TEST/EVALUATION EQUIPMENT

USED FOR:

- 1750A CHIP SET CONSOLE CONTROL
- TEST AND EVALUATION TOOL
- SUPPORT SEAFAC VALIDATION
- SUPPORT GD REQUIRED TESTING

D. M. Scott
Third Technical Forum
5 May 1982

Chart # 2

3

DESIGN APPROACH:

USE FAIRCHILD SYSTEM 1 DEVELOPMENT SYSTEM

- F9445 CPU EMULATOR
- 128K BYTES MEMORY
- 2 DOUBLE DENSITY 8" FLOPPY DISK DRIVES
- OPERATING SYSTEM SOFTWARE
- CRT/KEYBOARD
- I/O PORTS

D. M. Scott
Third Technical Forum
5 May 1982

Chart # 3

4

MODIFY FS-1 BY ADDING A 1750A TEST CARD

- NOVA BUS INTERFACE
- 1750A PROCESSOR BUS INTERFACE
- ZIF SOCKETS
- 128K BYTES RAM
- CONSOLE INTERFACE
- PROCESSOR I/O SIMULATION
- TEST POINTS

D. M. Scott
Third Technical Forum
5 May 1982

Chart # 4

HARDWARE CONTROL PANEL

- BASIC CONTROL OF CHIP SET OPERATION
 - RUN/STOP
 - SINGLE INSTRUCTION STEP
 - SINGLE CLOCK PULSE
 - ADDRESS AND BUS DISPLAY
- HARDWARE BREAKPOINT OPERATION
- INHIBIT ALL INTERRUPTS AND DMA/IO

D. M. Scott
Third Technical Forum
5 May 1982

Chart # 5

INTERIM PROCESSOR TRANSITION

Mr Henry Lynn

GD/FW

For a copy of this presentation please contact Mr Henry Lynn

General Dynamics Corporation
P O Box 748
Fort Worth, Texas 76101

(817)-732-4811

F-16 INTEGRATED SUPPORT
SOFTWARE SYSTEM

J73 BACKGROUND

- ALL F-16 MSIP SOFTWARE WILL BE IN J73 HOL
- CURRENT JOCIT AND SEA COMPILERS DO NOT MEET ALL MSIP REQUIREMENTS
- AFVAL HAS DEC '81 RELEASE THAT WE ARE USING IN INTERIM. IT STILL HAS ERRORS, BUT WILL MAKE PATCHABLE OFF
- GENERAL DYNAMICS HAS TWO SUPPLIERS UNDER CONTRACT (PSS AND ACT) TO PROVIDE FULL PRODUCTION J73 SUPPORT SOFTWARE PACKAGES.

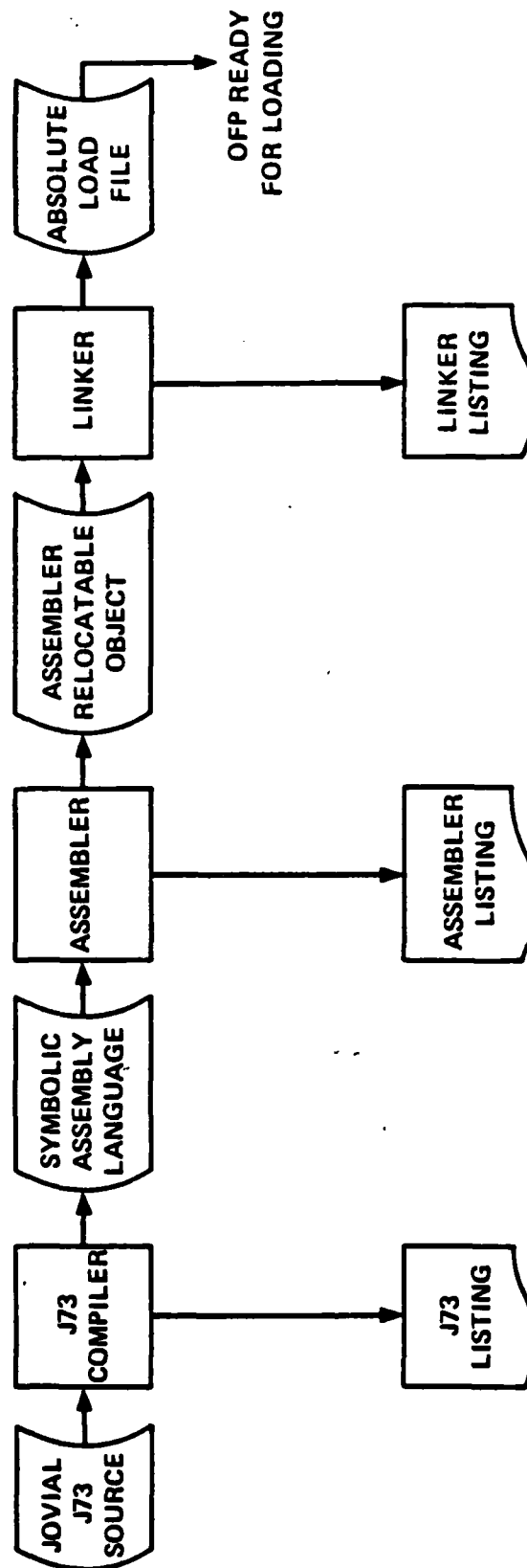
GOVERNMENT REQUIREMENTS
FOR SUPPORT SOFTWARE
DELIVERABLES

- JOVIAL J73 MUST REFLECT THE LATEST MIL-STD-1589B
- MIL-STD DOCUMENTATION FOR NEW CLASS A SOFTWARE
- ACCEPTANCE TESTING MUST INCLUDE VALIDATION TEST SET
- FORMAL DESIGN REVIEWS
- SOFTWARE MANAGEMENT AND QUALITY ASSURANCE PLAN

WHAT WE ARE BUYING

- JOVIAL J73 SUPPORT SOFTWARE SYSTEM WRITTEN IN ANSI FORTRAN '78 AND HOSTED ON THE IBM 370
- FEATURES
 - MIL-STD-1750A AND Z8002 TARGETS
 - COMMON "FRONT-END" COMPILERS, ASSEMBLERS AND LINKAGE EDITORS
 - FLEXIBLE COMPILER/HOST INTERFACE TO SUPPORT BOTH SINGLE MODULE COMPILATIONS AND OFP GENERATION
 - USER SPECIFIED MACHINE SPECIFIC FUNCTIONS
 - USER SPECIFIED LITERAL POOL TO SHARE LITERALS AMONG MULTIPLE COMPILATIONS
 - SYMBOL ATTRIBUTE AND SET/USED LISTING FOR EACH J73 MODULE
 - IEEE ASSEMBLY LANGUAGE AS COMPILER/ASSEMBLER INTERFACE
 - COMPLETE OFP LISTING GENERATED BY LINKER
 - IEEE ASSEMBLY LANGUAGE
 - J73 STATEMENTS INTERSPERSED AS COMMENTS
 - ABSOLUTE ADDRESSES THROUGHOUT
 - SYMBOLIC DEBUG INFORMATION PASSED TO LAB SUPPORT EQUIPMENT
 - FULL MIL-STD DOCUMENTATION
 - OPTIMIZATION TO 15% LEVEL

SUPPORT SOFTWARE PACKAGE HAS THREE MAJOR SECTIONS



ADVANCED COMPUTER TECHNIQUES

- 20 YEARS EXPERIENCE IN SUPPLYING SUPPORT SOFTWARE TO OEM MANUFACTURES
- PRODUCING J73 COMPILER, ASSEMBLER, AND LINKER FOR Z8002 AND MIL-STD-1750A
- COMPLETELY NEW IMPLEMENTATION IN FORTRAN 78
- DESIGNED TO MEET FULL F-16 SPECIFICATION
- FULL MIL-STD DOCUMENTATION
- DELIVERIES
 - Z8002 FIRST
 - 1750A SECOND

PROPRIETARY SOFTWARE SYSTEMS

- EXPERIENCED IN REHOSTING/RETARGETING AFVAL JOVIAL CROSS COMPILER
- PRODUCING J73 COMPILER, ASSEMBLER, AND LINKER FOR MIL-STD-1750A AND Z8002
- IMPLEMENTATION IS A MODIFICATION OF THE AFVAL JOVIAL COMPILER
- ASSEMBLER AND LINKER BASED ON THE PSS "DUAL" COMMERCIALY AVAILABLE SYSTEM
- FULL MIL-STD DOCUMENTATION ONLY ON NEW WORK
- DELIVERIES
 - 1750A FIRST
 - Z8002 SECOND

OTHER F-16 SUPPORT TOOLS

- SYSTEM DEVELOPMENT INTERFACE MANAGER (SYDIM)
 - DOCUMENTS COMPLETE OFF INTERMODULE INTERFACE
 - AUTOMATICALLY GENERATES J73 COMPOOLS
- MULTIPLE COMPIATION DRIVER
 - COMPILES AND ASSEMBLES COMPLETE OFF IN ONE IBM JOB STEP
 - ALL INPUT/OUTPUT VIA IBM PARTITIONED DATA SETS
- JOVIAL REFORMATTER
 - REFORMATS JOVIAL SOURCE ACCORDING TO F-16 STANDARDS
- MASTER ATTRIBUTE/SET/USED
 - COMPOSITE OF ALL COMPIATION ATTRIBUTE/SET/USED LISTINGS
- SOFTWARE TEST STATIONS
 - SIMULATES ACTUAL AIRBORNE ENVIRONMENT
 - REAL TIME COMPUTER MONITOR AND CONTROL INCLUDING JOVIAL SYMBOLIC DEBUG FOR TARGET COMPUTER

COMMON ASSEMBLY LANGUAGE

- o BECAUSE THE MNEMONICS IN 1750A ARE VERY DIFFICULT TO READ, EVEN FOR AN EXPERIENCED SOFTWARE ENGINEER, WE ARE ADOPTING A COMMON ASSEMBLY LANGUAGE FOR THE MIL-STD 1750A AND Z8002 THAT IS DIFFERENT FROM 1750 MNEMONICS.
- o THIS APPROACH GREATLY SAVES TIME IN THE LAB AND IN OUR TRANSITION TO MIL-STD 1750A BY REDUCING THE NUMBER OF MNEMONICS BY A FACTOR OF 8.
- o THE COMMON ASSEMBLY LANGUAGE WILL BE TAKEN FROM THE DRAFT IEEE STANDARD.

- o ADVANTAGES
 - o EASILY READABLE CODE
 - o EASY TO LEARN
 - o SIMPLE NAMING CONVENTIONS
 - o MANY FEWER MNEMONICS SINCE ADDRESSING MODE IS NOT IN MNEMONIC
 - o SPECIFIC OPERAND SYNTAX
 - o EASY TO PROGRAM
 - o EASILY TRANSLATED TO OTHER IEEE STANDARD MACHINES (Z8002)

OTHER ADVANTAGES TO IEEE

STANDARD

- o MINIMIZED TARGET-DEPENDENT CODE IN THE COMPILER
- o APPEARS MORE LIKE COMMERCIALY AVAILABLE SYSTEMS
 - oo HOL PROGRAMMERS CAN DEBUG PROGRAMS WITHOUT BEING EXPERIENCED ASSEMBLY LANGUAGE PROGRAMMERS
 - oo PEOPLE CAN BE TRANSITIONED AMONG PROJECTS WITHOUT HAVING TO BE RETRAINED IN SUPPORT SOFTWARE

CHARACTERISTICS OF MIL-STD-1750A IEEE MNEMONICS

- o FIRST CHARACTER OF MNEMONIC IS FIRST CHARACTER OF THE ACTION VERB DESCRIBING THE OPERATION. THE ENTIRE VERB IS INCLUDED WHEN PRACTICAL.
- o THE ENTIRE 1750A INSTRUCTION SET DECOMPOSES INTO 32 BASIC OPERATIONS:

ABS	- Absolute value
ADD	- Add
AND	- Logical AND
B	- Branch
BRK	- Break (Branch to Executive)
CALL	- Branch to subroutine
CLR	- Clear (set to zero)
CMP	- Compare
DEC	- Decrement
DIV	- Divide
FLOAT	- Convert to floating point
HALT	- Halt
INTGR	- Convert to integer
LD	- Load
MOV	- Move

MIL-STD-1750A BASIC OPERATIONS (CONT.)

MUL	- Multiply
NAND	- Logical NAND
NEG	- Arithmetic negate
NOP	- No operation
OR	- Logical OR
POP	- Pop from stack
PUSH	- Push onto stack
RET	- Return from subroutine
RO	- Rotate
SET	- Set (to logic 1)
SH	- Shift
ST	- Store
SUB	- Subtract
TEST	- Test (for logic 1)
XCH	- Exchange
XIO	- Execute I/O
XOR	- Logical exclusivd OR

0 BASIC MNEMONICS ARE MODIFIED BY SUFFIXES TO
SPECIFY OPERAND TYPE, SHIFT DIRECTION, ETC.

No Suffix	- Single word operation
F	- Floating point
L	- Long (2 words) -(Shift) left
X	- Extended precision floating point
M	- Multiple (words, registers, bits or operations)
I	- Individual Bit
S	- Product or dividend is same precision as other operands
B	- Byte
4	- Four bits
BK	- Block of words
R	- (Shift) right
A	- (Shift) Arithmetic
LR	- (Shift) left or right (according to sign of count)
RNG	- (Compare to) upper and lower range

o SUFFIXES ARE ALSO APPLIED TO THE BASIC BRANCH
MNEMONIC "B" TO SPECIFY CONDITIONAL BRANCHING

General Branch Suffixes:

R	- BR = unconditional branch
C	- Carry flag set (may be used alone or with any other conditional suffix)

Compare to Data Branch Suffixes

LT	- Less than
LE	- Less than or equal to
E	- Equal to
NE	- Not equal to
GE	- Greater than or equal to
GT	- Greater than

Compare to Zero Branch Suffixes

N	- Negative
NP	- Not positive (negative or zero)
Z	- Zero
NZ	- Not zero
NN	- Not negative (positive or zero)
P	- Positive

- o APPLYING THESE RULES TO MIL-STD-1750A ADD INSTRUCTIONS YIELDS:

<u>MIL-STD-1750</u>	<u>IEE STANDARD</u>
INCM	ADD
AIM	ADD
AISP	ADD
ABX	ADD
A	ADD
AB	ADD
AR	ADD
FA	ADDF
FAB	ADDF
FABX	ADDF
DA	ADDL
DAR	ADDL
EFA	ADDX
EFAR	ADDX

14 MNEMONICS COMPRESSED INTO 4 ALL EASILY RECOGNIZABLE
AS ADD INSTRUCTIONS

- o ADDRESSING MODES ARE REMOVED FROM THE MNEMONIC AND
PLACED IN THE OPERAND FIELD

<u>ADDRESSING MODE</u>	<u>FORMAT</u>
(R) REGISTER-REGISTER	R _B , R _A
(D) MEMORY DIRECT	ADDR
(DX) MEMORY DIRECT INDEXED	ADDR(R _x)
(I) MEMORY INDIRECT	@ ADDR
(IX) MEMORY INDIRECT INDEXED	ADDR(R _x)@
(IM) IMMEDIATE LONG	#DATA
(ISP) IMMEDIATE SHORT POSITIVE	#N (1 ≤ N ≤ 16)
(ISN) IMMEDIATE SHORT NEGATIVE	#N (-1 ≥ N ≥ -16)
(ICR) INSTRUCTION COUNTER RELATIVE	\$DSPL
(B) BASE RELATIVE	DSPL(Br)
(Bx) BASE RELATIVE INDEXED	(Br, R _x)
(S) SPECIAL	VARIES

- o OPERANDS ARE CODED IN THE ORDER OF SOURCE
FOLLOWED BY DESTINATION

ADD	 RB, RA
MOV BK	 @RB, @RA
LD	 ADDRESS (Rx), RA
ST	 RA, ADDRESS (Rx)

0 DATA STORAGE DIRECTIVE FOLLOWS GENERAL IEEE

RULES:

Single byte	DATAB	integer literal bit literal fixed point literal
Single word	DATA	integer literal bit literal fixed point literal
Double word	DATAL	integer literal bit literal fixed point literal
Multiple bytes	DATA C	character literal
Floating point	DATA F	real literal
Extended floating point	DATA X	real literal

- o LITERALS FOLLOW J73 RULES
- o FIXED POINT FORMAT FOLLOWS J3B-2 CONVENTION TO ALLOW
CONTEXT FREE SPECIFICATION

TYPE	FORMAT
Decimal integer	J73 integer literal 25 -14
Bit string	J73 bit literal 1B'1' 3B'377' 4B'AB5F'
Character string	J73 character literal 'ABC' 'SAY' 'HELLO''''
Fixed point	J3B-2 fixed point literal 102A8 -0.25A0
Floating point	J73 real literal 69.3 785.23E-2

0 FINAL RESULT IS A ONE-FOR-ONE TRANSLATION
OF 1750A INSTRUCTIONS TO IEEE FORMAT

ORIGINAL MIL-STD-1750A		MSIP IEEE	
MNEMONIC	OPERAND	MNEMONIC	OPERAND
XIO	RA, CMD	XIO	Ra, #data
XIO	RA, CMD, RX	XIO	Ra, #data(Rx)
VIO	RA, ADDR	XIOM	Ra, address
VIO	RA, ADDR, RX	XIOM	Ra, address(Rx)
SBR	N, RB	SETI	#data, Rb
SB	N, ADDR	SETI	#data, address
SB	N, ADDR, RX	SETI	#data, address(Rx)
SBI	N, ADDR	SETI	#data, @address
SBI	N, ADDR, RX	SETI	#data, address(Rx)@
RBR	N, RB	CLRI	#data, Rb
RB	N, ADDR	CLRI	#data, address
RB	N, ADDR, RX	CLRI	#data, address(Rx)
RBI	N, ADDR	CLRI	#data, @address
RBI	N, ADDR, RX	CLRI	#data, address(Rx)@
TBR	N, RB	TESTI	#data, Rb
TB	N, ADDR	TESTI	#data, address
TB	N, ADDR, RX	TESTI	#data, address(Rx)
TBI	N, ADDR	TESTI	#data, @address
TBI	N, ADDR, RX	TESTI	#data, address(Rx)@
TSB	N, ADDR	TESTSETI	#data, address
TSB	N, ADDR, RX	TESTSETI	#data, address(Rx)

CURRENT AFWAL J73/1750A LISTING

STATEMENT NUMBER		RELOCATABLE OFFSET		MIL-STD-1750A MNEMONICS	
9.	REGION 2 *	86 4 0 0004 L	DL	R4, zz	
	0004 H	C9 4 4	FMR	R4, R4	
	0006 H	86 6 0 0002 L	DL	R6, yy	
	0007 H	C9 6 6	FMR	R6, R6	
	0009 H	86 8 0 0000 L	DL	R8, xx	
	000A H	C9 8 8	FMR	R8, R8	
	000C H	A9 6 8	FAR	R6, R8	
	000D H	A9 4 6	FAR	R4, R6	
	000E H	96 4 C 0008	DST	R4, 8, R12	
	000F H	85 E C 000A	LIM	R14, A, R12	
	0011 H	85 2 C 0008	LIM	R2, 8, R12	
	0013 H	02 2 00	STB	R2, 0, R14	
	0015 H	85 2 C 0001	LIM	R2, 1, R12	
	0016 H	02 2 01	STB	R2, +1, R14	
	0018 H	7E F 0 0000*	SJS	R15, sqrt	
	0019 H	01 0 01	DLB	R0, 1, R12	
	001B H	96 0 0 0006 L	DST	R0, rss	
	001C H				

TYPICAL F-16 J73/1750A LISTING

Absolute address (following Link)		Statement Number	IEEE Mnemonics	Actual Jovial Statement	Margin Comments
6FC2	8600 0000	11.	BEGIN	RSS = SQRT((XX*XX) + (YY*YY) + (ZZ*ZZ));	
6FC4	C900	12.	LDL	XX,RO	
6FC5	8620 0000	12.	MULF	RO,RO	
6FC7	C922		LDL	YY,R2	
6FC8	A902		MULF	R2,R2	
6FC9	8620 0000		ADDF	R2,RO	
6FCB	C922		LDL	ZZ,R2	
6FCC	A902		MULF	R2,R2	
6FCD	0F00		ADDF	R2,RO	
6FCE	7EF0 0000		STL	RO,0(B15)	
6FDO	0702		CALL	@-R15,SQRT	
6FD1	9600 0000		LDL	2(B15),RO	
			STL	RO,RSS	
					; RSS

CHART #1

We are very briefly going to discuss and look ahead to the ultimate availability of the chip set, and we are going to discuss a little bit about how it will be available, and where it will be manufactured and those issues. We are also going to very briefly talk about long-term support to users of the chip set that will be available.

CHART #2

First thing I want to cover here is production facilities, JAN qualification, estimate of delivery availabilities, rough order of magnitude estimate of cost, and something we call a demand entry system, which is an early way of getting on the list. And then later on we will talk a little bit about our user support system.

CHART #3

I put this foil of Shai's back up here in case there exists any confusion as to what is meant by the term "chip set". The chip set, as you know from earlier discussion, consists of really four elements; the CPU, the block protect RAM, the MMU, the 93479 RAMs and the 54F273 buffers. Our discussion will involve all of these parts, so it should be recognized that the buffers and RAMs are standard devices that are part of Fairchild's standard product line.

CHART #4

Initially the 9450, which is a CPU, as well as the 9451 and 52 will be coming out of our facility in Palo Alto, California, and later from Wappinger Falls, New York. The long-term production facility will be Wappinger Falls, New York. The 93479, which is the 256 x 9 RAM will be coming out of Mountain View, in San Jose and later Puyallup, Washington, and the 54F245 buffer will always come out of South Portland, Maine.

CHART #5

What we are looking for on this chip set is satisfying all the requirements for a military-grade integrated circuit. We will meet and satisfy the MIL-M-38510 Rev. E requirements. And we will qualify our facilities to 38510 as well as 883 as well as MIL-STD-976.

In the area of device specifications, or slash sheets, we have a requirement as part of the contract to deliver a draft of a slash sheet for the 50, 51, and 52. The slash sheets, that is the JAN slash sheets, for the 245 should be released through RADC and DESC in 1983 and the 93479 slash sheet is an area of discussion between the program team and the DESC people at this point in time. That product is similar to a slash 233, which is a 4K bipolar RAM which is already in the JAN system. Estimate time for JAN qualifications Class B - second half of 1985; Class S - probably second half of 1986.

R. BYRNE

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #6

JAN qualifications are way out, by the nature of the system, so we will be looking for a DESC item drawing to become a standard procurement document. The release of that will be triggered by the delivery of the 100 preproduction units to GD. All procurement actions are triggered by that delivery. Meantime, there will be Fairchild preliminary information sheet, and advanced data sheet, out in September 1982.

CHART #7

Estimate of availabilities are based on delivery of 100 preproduction units to GD. Shipment to GD/Fort Worth - September of 1983. We are allowing a couple of months in there for a GD/FW approval cycle. It may go faster than that and shipments could then start to others roughly two months after approval by GD or January 1984. An estimate on volume - the ramping up of volume will start in the first quarter of 1984, roughly 100 units per month to 500 units per month in the second half of 1984, and a demand level of production by the first quarter of 1985. By demand level I mean we will scale the production to the demand that is coming in. Now ROM by the way in this presentation stands for "Rough Order of Magnitude" not Read Only Memory.

CHART #8

Current Rough Order of Magnitude estimates of cost - \$650 for the CPU, roughly \$100 each for the block protect RAM and the MMU, \$12 for the buffer and \$30 for the RAM. These are -55 to +125°C parts. Firm prices again will be available not later than the shipment to General Dynamics of the first 100 units, and tied to the release of a DESC-selected item drawing. We would like to get to one source control drawing, not get a source control drawing in from every possible user.

CHART #9

All the normal procurement requirements will be complied with, and these are 1981 dollars for 1984 delivery. Now this is in lieu of a formal order entry system. We have a number of requests from people who would want to enter orders, and the actual creation of a formal contract is a little premature at this point in time. So we have created something we'll call a "demand entry system". We are asking for a letter from your procurement authority, and the reason for that is so that out of one company, we end up getting one letter for one requirement, and not ten letters from ten different levels of the same company for the same requirements. Who you are, what's the quantity requirement, what's the schedule requirement, program identification, contact point, things like that.

Priority ratings are quite important in this case. We will respond to that, and we will acknowledge it. Acknowledgment will essentially say we received the letter and we have scheduled and logged the requirements, and you will be provided a periodic update on the availability and schedule situation.

CHART #10

As to mailing address, there are three things of importance:

1. Copies of your requirement letter as well as our logs are available to the government.
2. Please, no classified information.
3. Please, no business sensitive information within the meaning of the freedom of information act.

CHART #11

Now briefly, what we've talked about so far in this presentation is when parts will be available and roughly what they will cost. The second portion of this presentation is going to deal with some preliminary information on our plans for providing some support to the users of the chip set beyond that which might be available as a result of the effort that is being undertaken by GD, but consistent with, and in consort with, that work.

CHART #12

We want to or will work within the framework of utilizing the tools that were just discussed in the presentation here just a few minutes ago. Secondly, we look at the potential of providing a local in-circuit emulation and debugging support utilizing our existing Fiarchild FSI system as well as our existing EMUTRAC system.

CHART #13

Briefly, EMUTRAC is an in-circuit emulation and tracing system. We call it "Emulation and Tracing". It is used to support our existing line of microprocessors. It is coming on and we look forward to supporting at least a portion of the 9450 with this system.

CHART #14

Just a very brief picture of a set-up. The EMUTRAC system currently resides in the FSI chassis. It has a module or board which targets the EMUTRAC to the processor that is being used. The rest of this is pretty basic; there are files, printers, terminals, those kind of things.

CHART #15

We have in place microprocessor resource centers, which are application engineering centers. They are located currently at these locations both in the U.S. and in Europe. We plan to bring one aboard in Dayton as this program matures. And our staff in these programs, in these centers, will be trained to provide user level application support on the MIL-STD-1750 chip set.

R. BYRNE

TEXT FROM THE THIRD TECHNICAL FORUM, 5 MAY 1982

CHART #15 (Cont'd)

One last thing, we will provide the normal reliability and QA support that you people require, utilizing this type of device in a high-performance, high-reliability application. We will provide continual life testing. We will meet and continue to meet the MIL-STD-883B, MIL-M-38510 requirements as they are modified, interpreted and published by DESC and RADC. And we will support appropriate government source inspection requirements.

FAIRCHILD

A Schlumberger Company

MIL-STD-1750A

SINGLE CHIP BIPOLAR MICROPROCESSOR

F9450

DEVICE AVAILABILITY AND SUPPORT

R. Byrne
Third Technical Forum
5 May 1982

Chart # 1

Microprocessors
JRB20501

FAIRCHILD

A Schlumberger Company

● PRODUCT AVAILABILITY

PRODUCTION FACILITIES

JAN QUALIFICATION

ROM DELIVERY ESTIMATE

ROM COST ESTIMATE

DEMAND ENTRY SYSTEM

● PLANNED USER SUPPORT SYSTEM

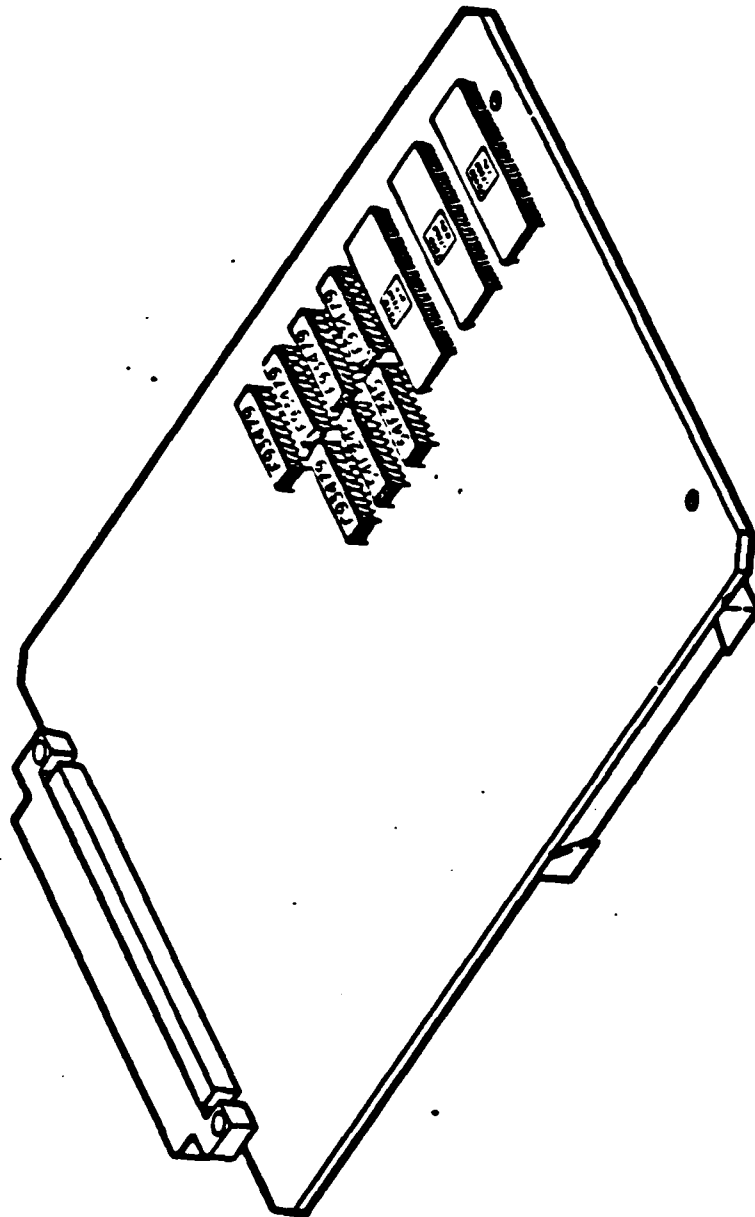
R. Byrne
Third Technical Forum
5 May 1982

Chart # 2

Microprocessors

JRB20501

1750A MICROPROCESSOR CHIP-SET



R. Byrne
Third Technical Forum
5 May 1982

Chart # 3

FAIRCHILD

A Schlumberger Company

MIL-STD-1750 CHIP SET

PRODUCTION FACILITIES

DEVICE

F9450

F9451

F9452

F93479

F54F245

FAIRCHILD PLANT

PALO ALTO, CA

AND

WAPPINGERS FALLS, NY

MOUNTAIN VIEW, CA

SAN JOSE, CA

PUYALLUP, WA (FUTURE)

SO. PORTLAND, ME

R. Byrne
Third Technical Forum
5 May 1982

Chart # 4

NOTE: 1³L FACILITIES - PALO ALTO, WAPPINGERS FALLS, MOUNTAIN VIEW, SO. PORTLAND

Microprocessors
JRB20501

FAIRCHILD

A Schlumberger Company

JAN QUALIFICATION

MIL-STD-1750 CHIP SET

REQUIREMENT

- MIL-M-38510 REV E

DEVICES DESIGNED TO MEET M38510 REQUIREMENTS

- MIL-M-38510 REV E, MIL-STD-833B MIL-STD-976

ALL FACILITIES ARE QUALIFIED

SO. PORTLAND AND MOUNTAIN VIEW NOW QUALIFIED

WAPPINGERS FALL AND PUYALLUP WILL BE QUALIFIED

- MIL-M-38510 REV E - DEVICE SPECIFICATIONS (SLASH SHEETS)

F9450, 51 AND 52

PRELIMINARY DOCUMENTS - DELIVERABLE 2ND HALF 1983

F54F245

EST RADC/DESC RELEASE IN 1983

F93479

NO SCHEDULE - DESC DETERMINATION OF NEED

DEVICE SIMILAR TO /233

R. Byrne
Third Technical Forum
5 May 1982

Chart # 5

Microprocessors

JRB20501

PROCUREMENT CONTROL DOCUMENTATION

● JAN QUALIFICATION

TODAY'S ESTIMATE - 2ND HALF 1985 - CLASS B
2ND HALF 1986 - CLASS S

● DESC SELECTED ITEM DRAWING

DESC SELECTED ITEM DRAWING NEEDED TO PROVIDE STANDARD SCD UNTIL
JAN SPECIFICATION AVAILABLE.

DRAWING RELEASE "TRIGGERED" BY GD-FW

APPROVAL OF 100 PREPRODUCTION UNITS

● FAIRCHILD ADVANCE INFORMATION DATA SHEET

RELEASED - SEPTEMBER 1982

R. Byrne
Third Technical Forum
5 May 1982

Chart # 6

Microprocessors

JRB20501

FAIRCHILD

A Schlumberger Company

ROUGH ORDER OF MAGNITUDE

DELIVERY ESTIMATE

F9450-MIL-STD-1750A P

INITIAL SHIPMENTS - 60 DAYS AFTER GD-FW APPROVAL OF 100 PREPRODUCTION UNITS

SHIP TO GD-FW - SEP 1983

GD-FW APPROVAL - NOV 1983

SHIP TO OTHERS - JAN 1984

AVAILABILITY - VOLUME - APPROXIMATELY

100 PER MONTH	1ST QTR 1984
300 PER MONTH	2ND QTR 1984
500 PER MONTH	2ND HALF 1984
DEMAND LEVEL	1ST QTR 1985

NOTE: ALL DATES KEYED TO DELIVERY OF 100 EACH TO GD-FW

R. Byrne
Third Technical Forum
5 May 1982

Chart # 7

Microprocessors

JRB20501

FAIRCHILD

A Schlumberger Company

ROUGH ORDER OF MAGNITUDE COST

	<u>ROM COST</u>
F9450	\$650 EA
~~54F274~~ 54LS215	12
F9451	100
F9452	100
93479	30

- NOTE (1) ESTIMATED COST IS FOR -550 TO +1250 C TC DEVICES.
- (2) FIRM PRICES WILL BE AVAILABLE NOT LATER THAN SHIPMENT TO GD-FW OF FIRST 100 CHIP SETS AND RELEASE OF DESC SELECTED ITEM DRAWING.
- (3) FIRM PRICES WILL BE BASED ON DAR REQUIREMENTS WITH DD633 DATA PROVIDED WHEN PROCUREMENT CIRCUMSTANCES REQUIRE.
- (4) ROM COSTS ARE 1981 DOLLARS FOR 1984 DELIVERY.

R. Byrne
Third Technical Forum
5 May 1982

Chart # 8

Microprocessors

JRB20501

MIL-STD-1750 CPU CHIP SET

- DEMAND ENTRY SYSTEM MAINTAINED AT MICROPROCESSOR DIVISION, FAIRCHILD
 - DEMAND INPUT
 - RECEIPT OF REQUIREMENT LETTER FROM USER'S PROCUREMENT AUTHORITY
 - USER LOCATION
 - QUANTITY
 - SCHEDULE REQUIREMENTS
 - PROGRAM IDENTIFICATION
 - USER CONTACT POINT
 - PRIME CONTRACT NO. (IF AVAILABLE)
 - PRIORITY RATING - IF APPLICABLE (DX-DO)
 - ACKNOWLEDGEMENT DOCUMENT MAILED TO USER'S PROCUREMENT AUTHORITY
 - RECEIPT OF LETTER OF INTENT
 - CURRENT ESTIMATE OF POTENTIAL AVAILABILITY AGAINST USER'S
 - SCHEDULED REQUIREMENTS
 - REQUIREMENT ENTERED IN DEMAND REGISTER
 - PERIODIC AVAILABILITY UPDATE
- RECEIPT OF DEMAND LETTER AND/OR ISSUANCE OF ACKNOWLEDGEMENT DOCUMENT WILL NOT RESULT IN A CONTRACTUAL AGREEMENT.

R. Byrne
Third Technical Forum
5 May 1982

Chart # 9

Microprocessors
JRB20501

FAIRCHILD

A Schlumberger Company

MIL-STD-1750 CPU CHIP SET

DEMAND ENTRY SYSTEM

MAILING ADDRESS FOR REQUIREMENT LETTER:

MIL-STD-1750 CPU REQUIREMENTS
BOX 40-1750
3420 CENTRAL EXPRESSWAY
SANTA CLARA
CALIFORNIA
95051

NOTES: (1) COPIES OF ALL REQUIREMENT LETTERS AS WELL AS REQUIREMENT INPUT LOG
WILL BE AVAILABLE TO AUTHORIZED U.S. GOVERNMENT PERSONNEL

(2) NO CLASSIFIED INFORMATION

(3) NO BUSINESS SENSITIVE INFORMATION

R. Byrne
Third Technical Forum
5 May 1982

Chart # 10

Microprocessors

JRB20501

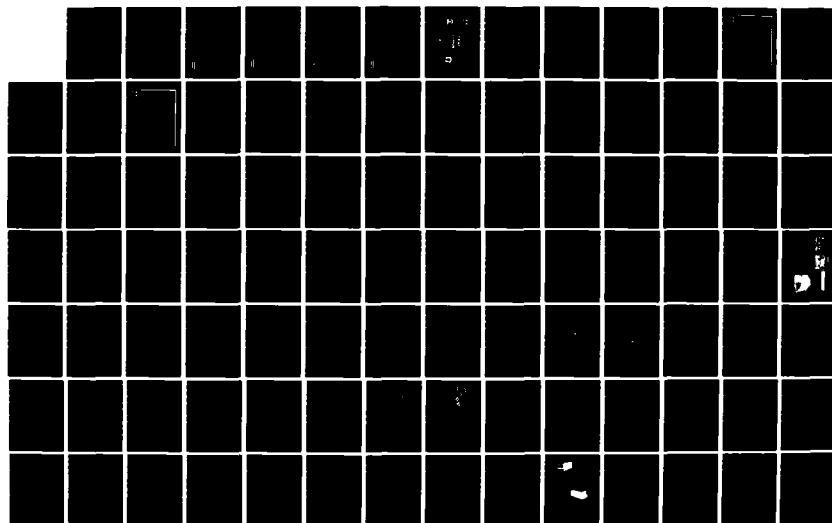
AD-A150 583

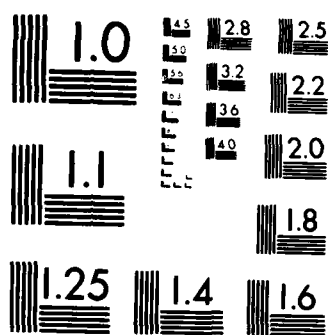
PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16
MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV
WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 06 MAY 82
ASD-TR-82-5011-VOL-1 F/G 9/2

4/6

UNCLASSIFIED

NL





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

FAIRCHILD

A Schlumberger Company

MIL-STD-1750 CPU CHIP SET

- FAIRCHILD PLANNED USER SUPPORT SYSTEM

DEVELOPMENT SYSTEM AND TOOLS

LOCAL MRC CENTER APPLICATION SUPPORT

FACTORY RELIABILITY AND QA SUPPORT

R. Byrne
Third Technical Forum
5 May 1982

Chart # 11

Microprocessors
JRB20501

MIL-STD-1750 CPU CHIP SET
DEVELOPMENT SYSTEM AND SOFTWARE TOOLS

- UTILIZE MAINFRAME HOISTED (360 OR VAX) GOVERNMENT PROVIDED SOFTWARE TOOLS

MIL-STD-1589B CROSS COMPILER

MIL-STD-1750 CROSS ASSEMBLER

MIL-STD-1750 CROSS LINKER, SIMULATOR

- POTENTIAL OF PROVIDING LOCAL IN CIRCUIT EMULATION AND "DEBUGGING" SUPPORT WITH FAIRCHILD FS-1 AND EMUTRAC SYSTEM

R. Byrne
Third Technical Forum
5 May 1982

Chart # 12

Microprocessors
JRB20501

FAIRCHILD

A Schlumberger Company

EMUTRACTM

EMULATION AND TRACING SYSTEM

SOME FEATURES ARE:

- PROVIDES OPTIONAL SUBSTITUTION FOR CPU AND I/O PERIPHERALS AND MEMORY IN PROTOTYPE SYSTEMS
- 16KB OF MAPPABLE SUBSTITUTION-RAM PROVIDED FOR 2148 + 93475 RAM
- 4BIT TAGS - BREAKPOINT MARKING USER ASSIGNABLE
- BREAKPOINT COMPARATOR - CAN DETECT EIGHT SIMULTANEOUS BREAKPOINT CONDITIONS
- PROGRAMMABLE MICRO-SEQUENCER
- FULLY INTERACTIVE HARDWARE/SOFTWARE DEBUGGING

R. Byrne
Third Technical Forum
5 May 1982

TM-EMUTRAC is trademark of FAIRCHILD CAMERA & INSTRUMENT CORP.

Chart # 13

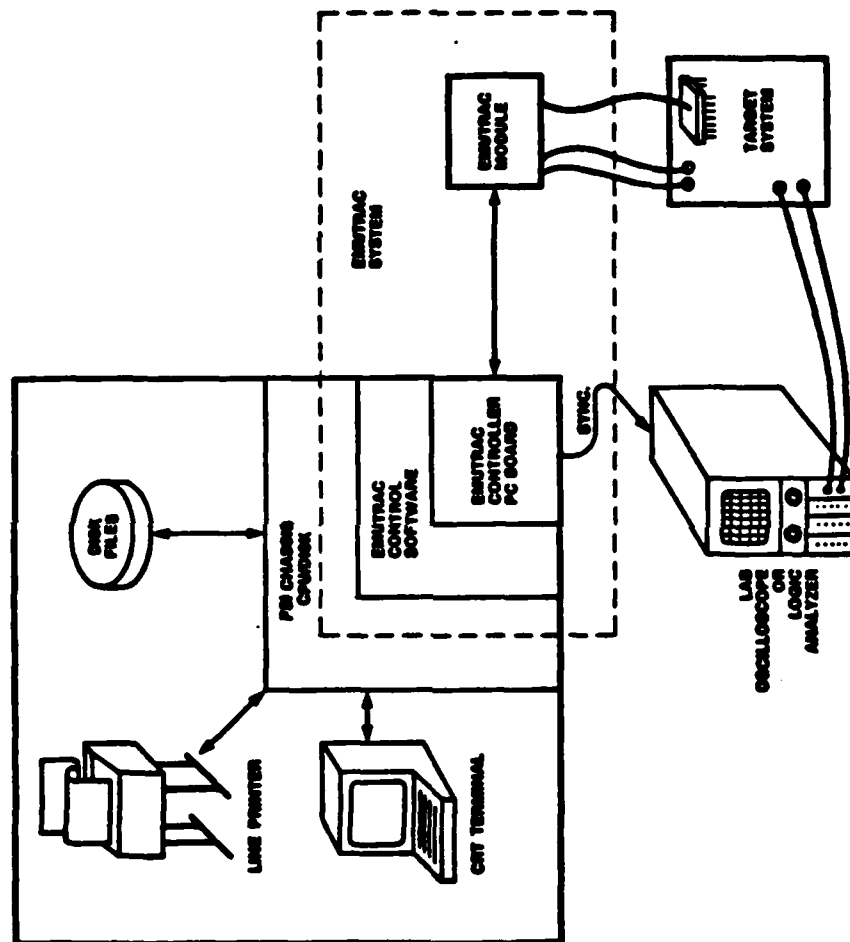
Microprocessors

JRB20501

FAIRCHILD

A Schlumberger Company

EMUTRAC System Set-up



TODAY, EMUTRAC SUPPORTS F9445, F3870, F6800

F9450 SUPPORT - 1984

R. Byrne
Third Technical Forum
5 May 1982

Chart # 14

Microprocessors

JRB20501

FAIRCHILD

A Schlumberger Company

MICROPROCESSOR RESOURCE CENTER

MIL-STD-1750 CPU CHIP SET

LOCAL APPLICATION SUPPORT

US LOCATIONS

BOSTON

CHICAGO

DALLAS

DAYTON (FUTURE)

MINNEAPOLIS

SANTA ANA

SANTA CLARA

EUROPE

LONDON

PARIS

MUNICH

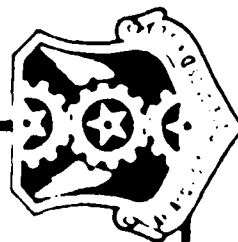
MRC CENTERS WILL HAVE FS-1 SYSTEMS WITH EMUTRAC
SET UP FOR MIL-STD-1750 CPU CHIP SET USER SUPPORT

R. Byrne
Third Technical Forum
5 May 1982

Chart # 15

Microprocessors

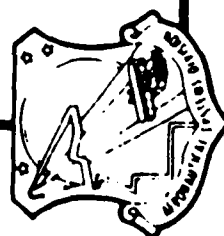
JRB20501

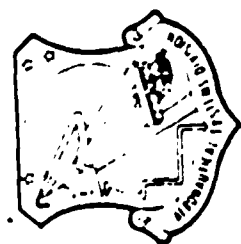


AVIONICS CONTROL

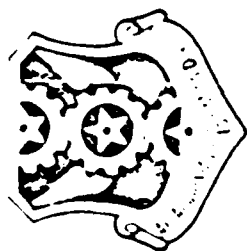
BRIEFER : COLONEL GEORGE W. BOTBYL

DEPUTY FOR AVIONICS CONTROL

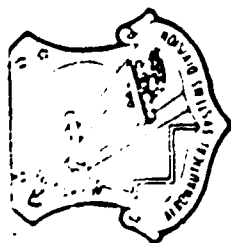




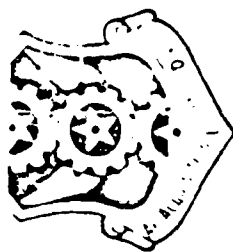
AVIONICS CONTROL AVIONICS BUSINESS



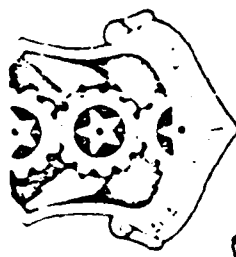
- USAF AIRCRAFT INVENTORY
 - 10,000 AIRCRAFT
- AVIONICS
 - 200,000 SUBSYSTEMS - \$18 BILLION INVESTMENT
 - DEVELOPMENT BUDGET - \$1.1 BILLION/YEAR
 - CLASS IV AND V MODS - \$1.2 BILLION/YEAR
 - OPERATIONS AND MAINTENANCE - \$3 BILLION/YEAR



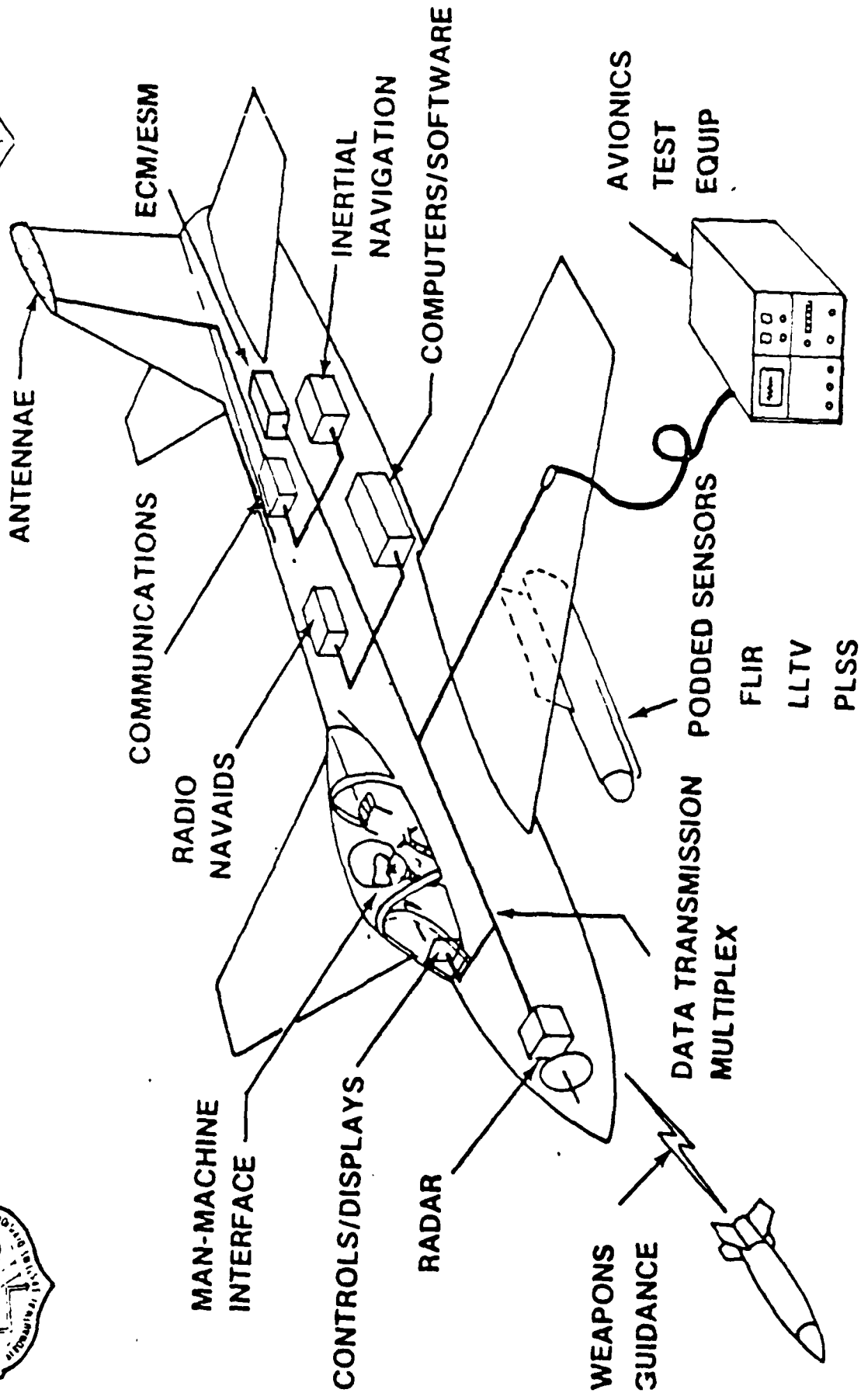
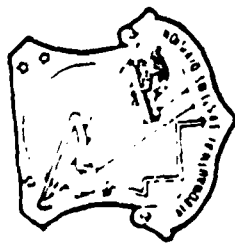
AVIONICS CONTROL CHARTER

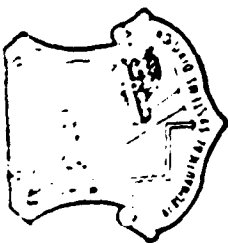


- ASAF (RD&L) - APRIL 1978:
 - REVIEW AVIONICS PROGRAMS
 - DEVELOP AVIONICS INVESTMENT STRATEGY
 - DEVELOP AVIONICS PLANS AND DATA BASE
- AFSC/CC - DECEMBER 1980:
 - MANAGE EMBEDDED COMPUTER RESOURCE STANDARDS
 - JOINT AFSC/AFLC CHARTER

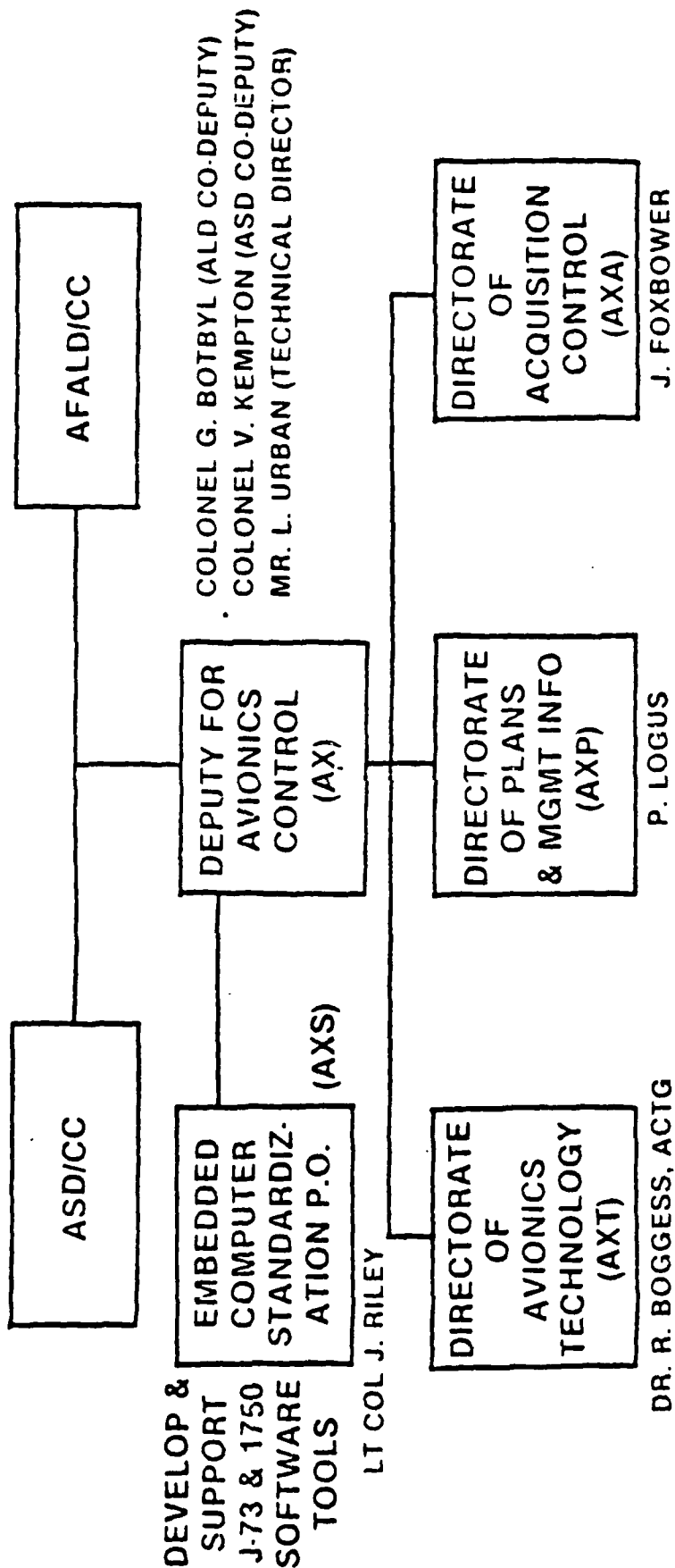
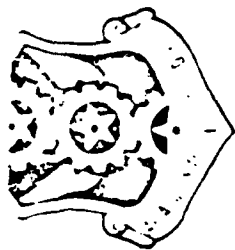


AVIONICS





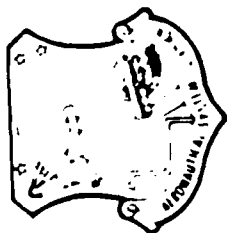
AVIONICS CONTROL ORGANIZATION



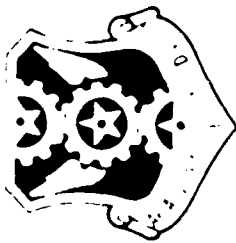
ENGINEERING CONSULTANTS
TECHNOLOGY TRANSITION
NEW STANDARDS

AVIONICS MASTER PLAN
AVIONICS PLNG BASELINE
LIFE CYCLE COST

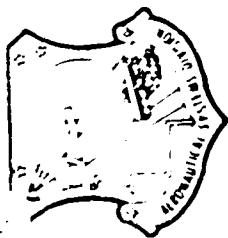
INTERFACE WITH SPOs & ALCs
NEW DEVELOPMENTS
MODIFICATION PROGRAMS



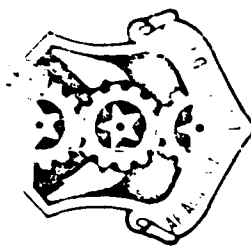
AVIONICS CONTROL ORGANIZATIONAL PHILOSOPHY



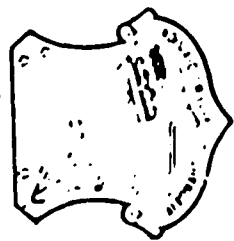
- NOT A PROGRAM OFFICE
- INSURES USAF/AFSC/AFLC POLICIES RATIONALLY CONSIDERED/IMPLEMENTED
- PROMOTES COMBAT EFFECTIVENESS
- PROMOTES RELIABLE AND LOGISTICALLY SUPPORTABLE AVIONICS
- PROMOTES INTEROPERABILITY AND RATIONAL STANDARDIZATION
- SERVES AS CATALYST - INDUSTRY USER GROUPS



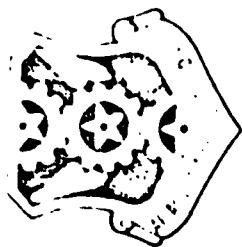
AVIONICS CONTROL CURRENT STANDARDS



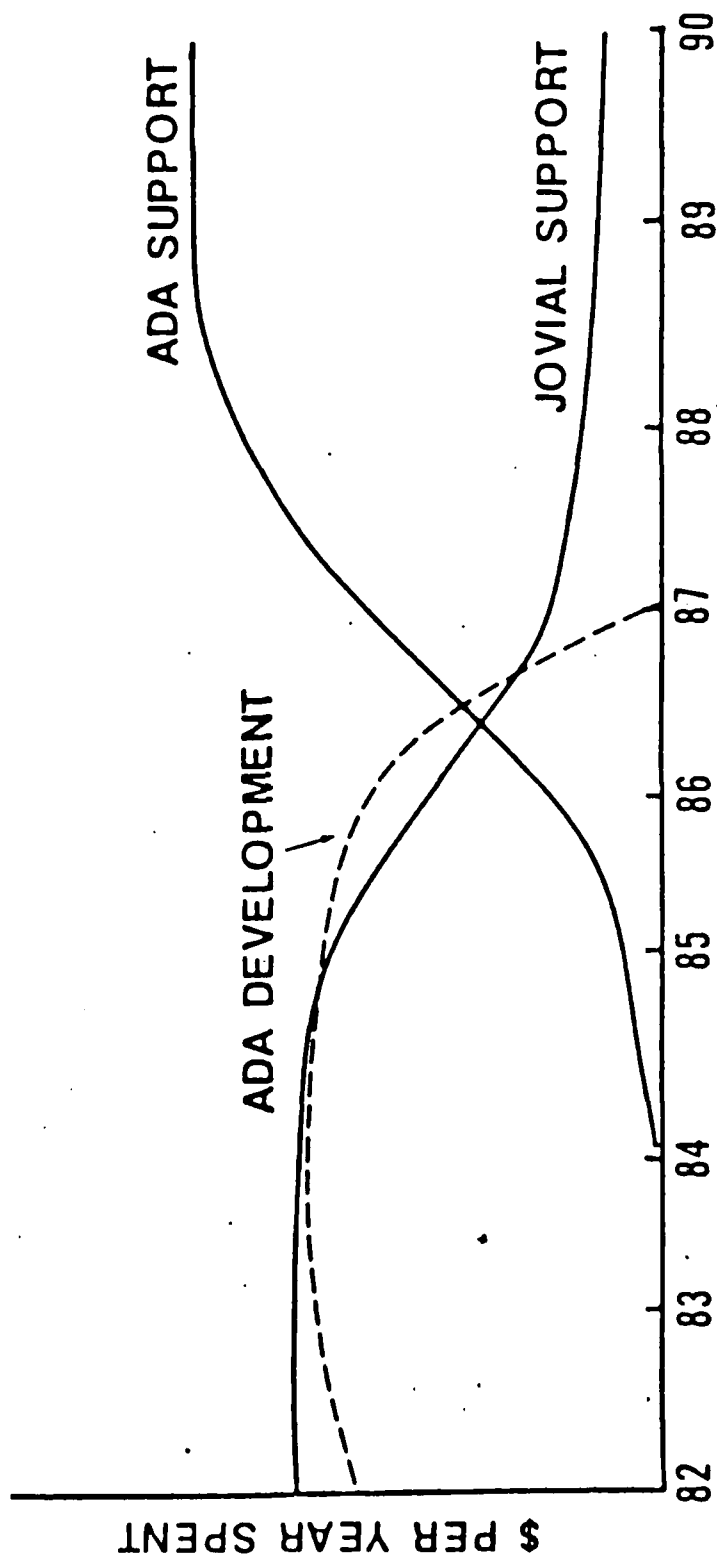
- ARCHITECTURAL MIL-STDs:
 - 1553B MULTIPLEX BUS
 - 1589B HIGHER ORDER LANGUAGE
 - 1750A INSTRUCTION SET ARCHITECTURE
 - 1760 ARMAMENTS INTERFACE
- SUBSYSTEMS:
 - STANDARD/PREFERRED ITEMS (ARC-164, ARN-118)
 - F3 STANDARDS (INS, CARA, SCADC)

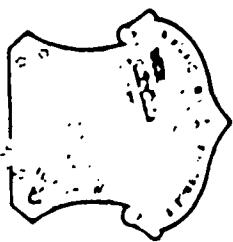


HIGHER ORDER LANGUAGES

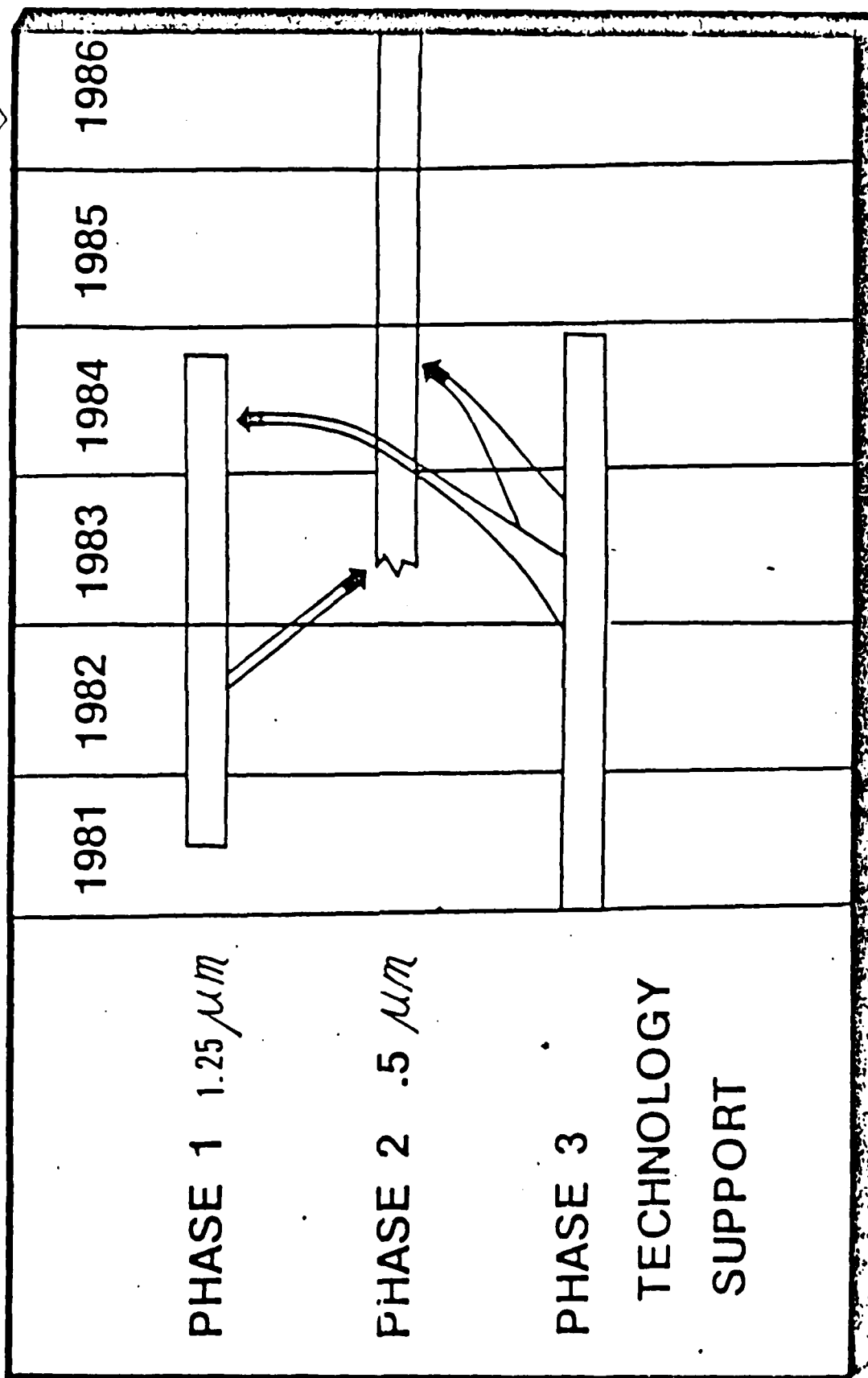
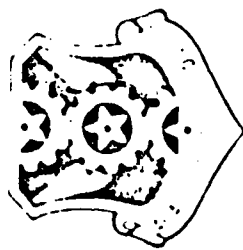


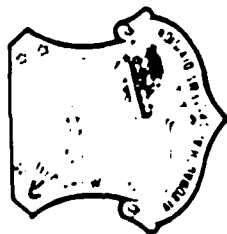
INVESTMENT STRATEGY



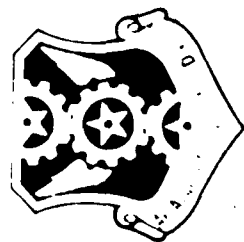


VHSIC SCHEDULE





AVIONICS CONTROL ACCOMPLISHMENTS



- FIRM COMMITMENT/FUNDED
 - A-10 (1553B F3 INS, CDU, CARA AND MATE)
 - F-16 MSIP AND F-16E (1553B, 1589B, 1750A, 1760A, CARA AND MRR)
 - F-111D/F AND FB (1533B, 1589B, 1750A, CARA AND SCADC)
 - B-1B LRCA (1553B, MRR)*
 - F-5G (1553B, 1589B, 1750A, F3 INS)
 - C/KC-135 (1553B, 1589B, 1750A, SCADC AND CDU)
 - LANTIRN (1553B, 1589B, 1750A)
 - MATE (ATLAS, 1589B, 1750A)
 - MRR (1553B, 1589B, 1750A)
 - CARA, SCADC, GPS, CNPI, JTIDS, A/J COMM
- COMMITTED BUT FUNDS NOT FIRM
 - F-15E (1553B, 1589B, 1750A)
 - C-17 (1553B, 1589B, 1750A F3 INS AND CARA)
 - HH-60D, (1553B, 1589B, 1750A)
 - F-111A/E AND EF (1553B, 1589B, 1750A, F3 INS, CARA, SCADC AND MATE)

*PARALLEL EFFORT (1589B, 1750A)

MICROPROCESSOR LIFE CYCLE COSTING FOR MIL-STD 1750A

Janice L. Lyons

General Dynamics

The U.S. Air Force is interested in installing a standard microprocessor, as soon as possible, in current and next generation fighter/attack aircraft and other defense systems. A standard microprocessor would eliminate the need for maintaining multiple software packages and multiple chips and chip sets.

This paper presents an approach for analyzing the life cycle costs of such a standard microprocessor, specifically that defined in MIL-STD-1750A. The costing approach was formulated to highlight the operational and design differences found between military standard and conventional commercial microprocessors as employed in a military environment. Both hardware and software life cycle cost implications are addressed.

INTRODUCTION

The MIL-STD-1750A microprocessor specification presents a revolutionary concept for microprocessor design. The standard does not specify a specific hardware configuration, but rather defines a standard for processing functions and chip performance. That is, the microprocessor can be designed as either a highly integrated chip or as a chip set. This approach allows the microprocessor instruction set and software to remain constant as the hardware incorporates technology advances. The chip/chip set can then evolve to remain responsive to changing user needs.

Such an innovative approach to chip design and standardization presents a problem in assessing the impact on life cycle cost. At present, there is very little cost data and related information on microprocessors, much less the information required to project the cost of a design such as 1750A. This paper does not attempt to address the specific costs of the standard microprocessor; instead the sections that follow present the approach, assumptions and methodology used to analyze MIL-STD-1750A life cycle costs.

AREAS OF CONSIDERATION

In studying the 1750A microprocessor costs, it was discovered that the differences between a military standard processor and a conventional commercial processor were most apparent in the operations and support phase of the life cycle. These O&S differences can be attributed to such design-related characteristics as the MIL-standard processor being an instruction/operating standard as opposed to a hardware standard, as well as a general design capable of effective operation in more than one type of operating environment.

As a microprocessor operating standard, the 1750A processor has a standard instruction set which impacts both hardware and software revisions and maintenance. A standard higher order language (HOL) becomes feasible as the programming language is no longer dictated by the compiler within the individual microprocessor. The processor software, developed to be compatible with the instruction set, becomes independent of system hardware changes and modifications. As technology advances and new microprocessor chips and chip sets are brought into service, minimal software changes are required to successfully integrate the revised hardware into the operating system. Also, no massive software upgrades are needed to match hardware technology advances; software revisions should be minimal and should

occur on a continuing basis.¹

With changes in chip technology transparent to the software, hardware revisions can occur as justified to meet changing user needs. This allows the microprocessor system to realize the benefits from continuing improvements in such areas as reliability, which has historically improved 14% annually.²

The compatibility of the 1750A microprocessor with multiple operating environments encourages its wide-spread application in military weapon systems. By not being weapon or function specific, the microprocessor may be manufactured in significant quantities and thus realize cost reductions due to economies of scale. Additional cost reductions can be realized through competitive bidding since larger production quantities will make multiple source contracting feasible.

Multiple weapon system usage will also affect the quantity of spares required. Spares provisioning further impacts inventory management and technical data requirements. As microprocessor maintenance becomes less weapon-specific, support equipment will also benefit from economies of scale since universal designs can be utilized as opposed to multiple specialized equipment. Additionally, support equipment, like the processor, will benefit from being more widely procured and operated.

COSTING APPROACH

Microprocessor life cycle costs (LCC) were divided into categories as shown in Table 1. These divisions of LCC by hardware and software are believed to best represent the cost impacts of the available performance, operations and support information. In defining the approach to costing, a ten-year life-cycle was assumed; electronics systems are typically updated every five years,¹ thus a ten-year life cycle takes into account one complete hardware/software revision of the microprocessor.

The hardware costing task was approached as a standard procurement costing problem. Estimates were received from microprocessor developers as to the expected development costs for a military standard processor. A representative development cost figure was chosen with the help of the Air Force to represent the 1750A chip development effort.³

A survey approach was also taken to determine unit costs for a specific quantity buy. The information received indicated a 70% "learning curve" effect as microprocessor chip quantities

TABLE 1

ELEMENTS OF MICROPROCESSOR LIFE CYCLE COSTS

<u>HARDWARE LCC</u>	<u>SOFTWARE LCC</u>
Development	Development & Acquisition
Production	Operations & Support
Operations & Support	System Upgrades
Hardware Upgrades	Software Maintenance
Spares & Inventory	Support Maintenance
Support Hardware	Personnel Requirements
Personnel Requirements	

increase from 5000 to 500,000 production units per year (Figure 1).¹ For a relatively small annual procurement of 1000 chips, unit costs could be expected to be in the \$4000 range, while for a multiprogram procurement of 5 million chips annually, costs are expected to be less than \$100 per unit. The large variation in microprocessor costs due to changes in annual production rates reflects the relatively large fixed to variable production and development costs in the microprocessor industry. This further emphasizes the production economies of scale available in chip manufacture for widely used microprocessor chips.

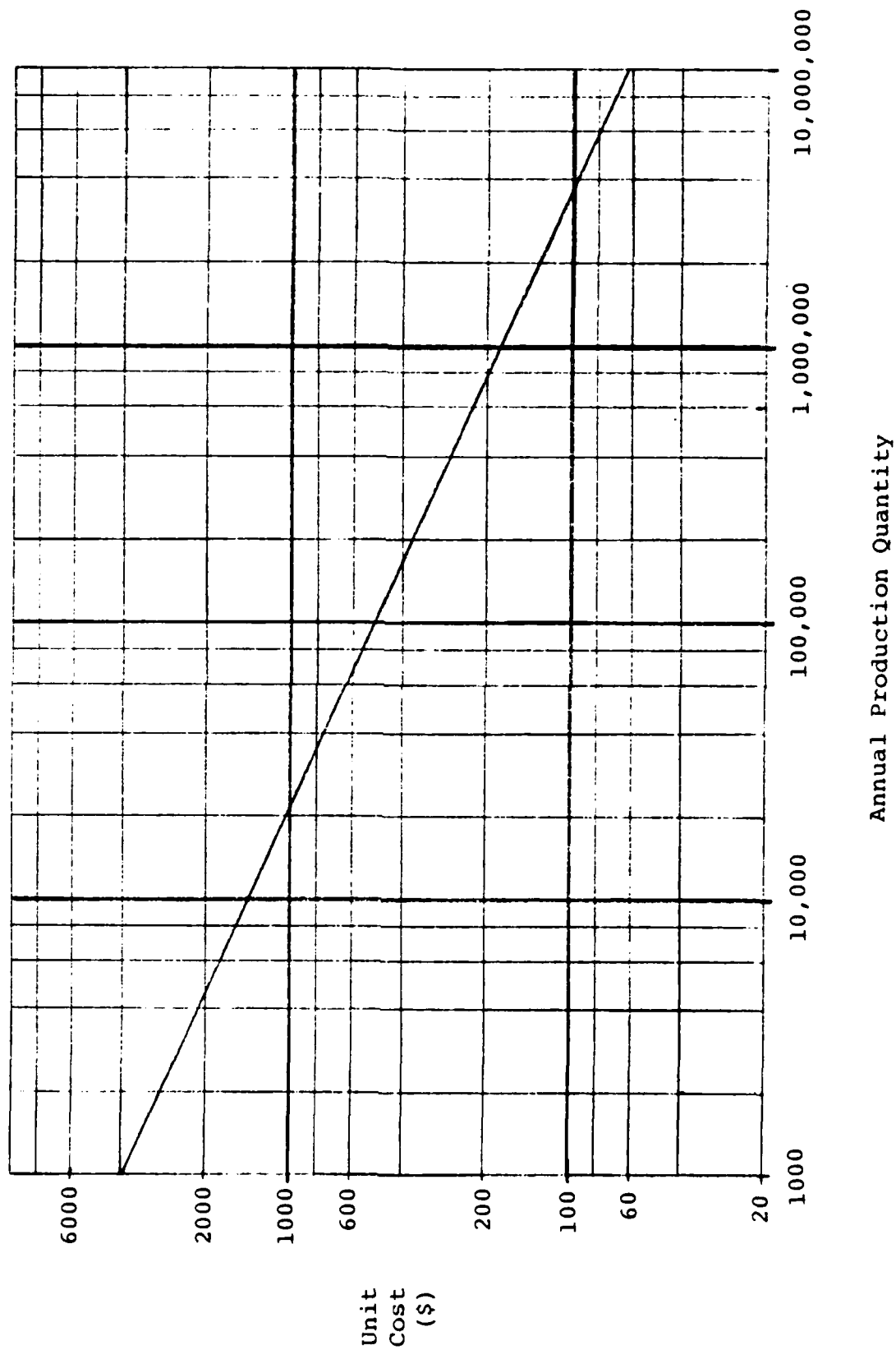
Industry rules-of-thumb and insights obtained from related study efforts were the basis of hardware operations and support costs estimates.

Since system upgrades to hardware are continuous for the 1750A, no significant cost for hardware redesign is anticipated; design changes evolve with technology updates.

The spares quantity required over a ten-year period was assumed to be equivalent to the initial purchase quantity.¹ The processor design, whether it is an integrated chip or a chip set, determines the processor's requirements for inventory management and technical data. Also, the number of inventory items is affected as a general use microprocessor replaces multiple specialty processors.

Figure 1

Learning Curve Effect for Microprocessor Chip Production



Support hardware was costed at 30% of hardware procurement costs. Additional charges for specific support equipment were not determined since essentially the same equipment is necessary to support the standard processor or a commercially available equivalent system.⁴

Personnel for 1750A hardware maintenance and support is initially expected to be equivalent to that required for any commercial microprocessor system. For subsequent programs using the standard processor, less personnel training and technical data should be required as a result of parts standardization. The manpower requirements are not expected to be altered significantly since the reliability and maintainability of electronics systems will not be significantly impacted by the standard processor.

Software development and acquisition costs were determined by analogy with other software efforts. To initially write code for the 1750A would be equivalent to writing code for any other new, untried processor.⁵ Once the code is written, however, costs to upgrade the code to incorporate hardware changes should be minimal. Normally, a processor requires 20% of its initial code to be upgraded annually due to hardware revisions reflecting technology improvements.¹ In the case of the standard processor, changes in chip technology should be transparent with respect to the software.

Software maintenance costs average 90% of software development and acquisition expenses. This is typically 27% of software life cycle costs where system upgrades are made on an annual basis. (Table 2.)¹

Support software costs depend on the quantity of maintenance tests required to fault-isolate a chip set to the desired level. Development of maintenance tests requires an average of six man-months per chip fault-isolated.⁴

The impact of the standard processor on software maintenance personnel costs is realized predominantly in the area of training. With a standard instruction set, less training is required to keep a programmer competent in compiler assembler language since the language in use is not constantly changing.⁵ Other personnel cost savings are inherent in the reduction of the system upgrading task.

CONCLUSION

The U.S. Air Force anticipates significant long-range economic benefits to be derived from the adoption of MIL-STD-

TABLE 2

TYPICAL SOFTWARE LCC BREAKOUT

<u>FUNCTION</u>	<u>% OF LCC</u>	
Development		
Design	12	} 30%
Code	6	
Test	12	
Maintenance		
Emergency Correction	9	} 27%
Routine Debugging	6	
Input Data/Files Revision	12	
Updates		
Hardware Changes	4	} 43%
User Enhancement	30	
Documentation Improvement	4	
Code Efficiency Improvement	3	
Other	2	
Life Cycle Cost	100	

Source: Ernie Currey (Aeronautical Systems Division, Wright-Patterson Air Force Base). Interview, 4 September 1981

1750A. The most significant benefits are expected to be realized in the operations and support phase of the life cycle in the areas of personnel training, hardware and software upgrades, and support software.

There are also potential cost savings to be gained from large-scale procurement of a standard microprocessor. When initially brought into use, the 1750A processor will be more expensive than its commercial equivalent due to the investment required for research and development and due to its production as a specialty item. With widespread use of the standard, unit costs are expected to decline as yearly production quantities increase, and as competitors enter the market.

Whether the incorporation of a standard microprocessor results in a cost savings or cost increase over the life cycle is determined by the relationship between the initial investment cost and the anticipated long-term unit and operating cost savings.

REFERENCES

1. Ernie Curry (Operations Research Analyst, Wright-Patterson Air Force Base). Interview, 4 September 1981.
2. Ajmel S. Dulai (Software Engineering Chief, Wright-Patterson Air Force Base). Interview, 4 September 1981.
3. Jeffrey L. Pesler (1750A Project Engineer, Wright-Patterson Air Force Base). Personal communication, 31 August 1981.
4. Clifford W. Douglas (Senior Engineer, General Dynamics/Fort Worth). Interview, 13 August 1981.
5. Judy A. Edwards (Engineering Specialist Senior, General Dynamics/Fort Worth). Interview, 11 August 1981.

Janice L. Lyons is an operations analyst at General Dynamics/Fort Worth Division. She received her BA in economics from The College of William and Mary in 1976 and MSSE in systems engineering from Clemson University in 1979. Prior to joining General Dynamics in 1980, she was employed in industrial and software engineering by Milliken and Clemson. Her current areas of research are the economic impacts of advanced electronics design and alternative support concepts as applied to aircraft avionics.

**MICROPROCESSOR
LIFE CYCLE COSTING
FOR MIL-STD 1750A**

Presented By

JANICE L. LYONS

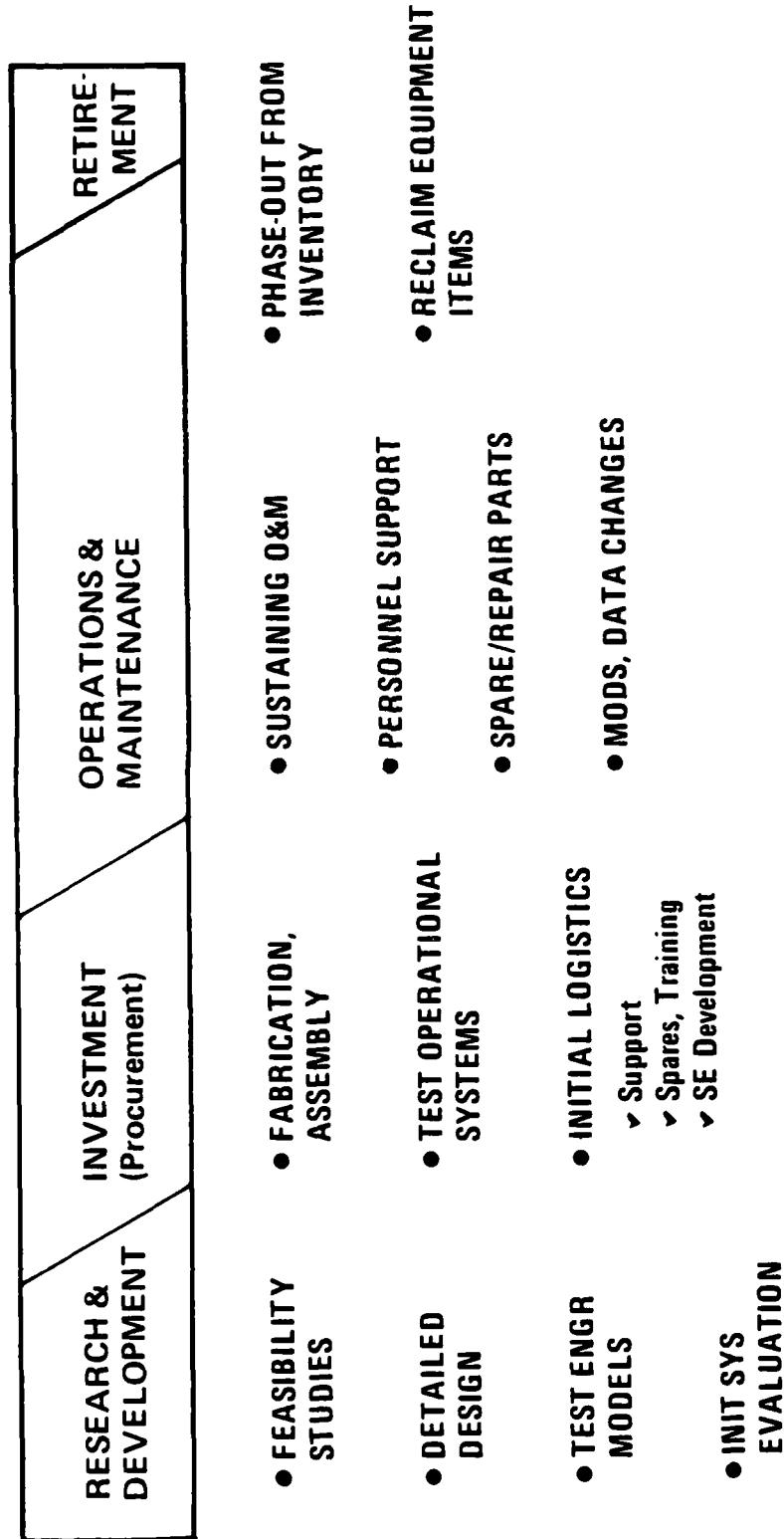
GENERAL DYNAMICS

Fort Worth Division

853098

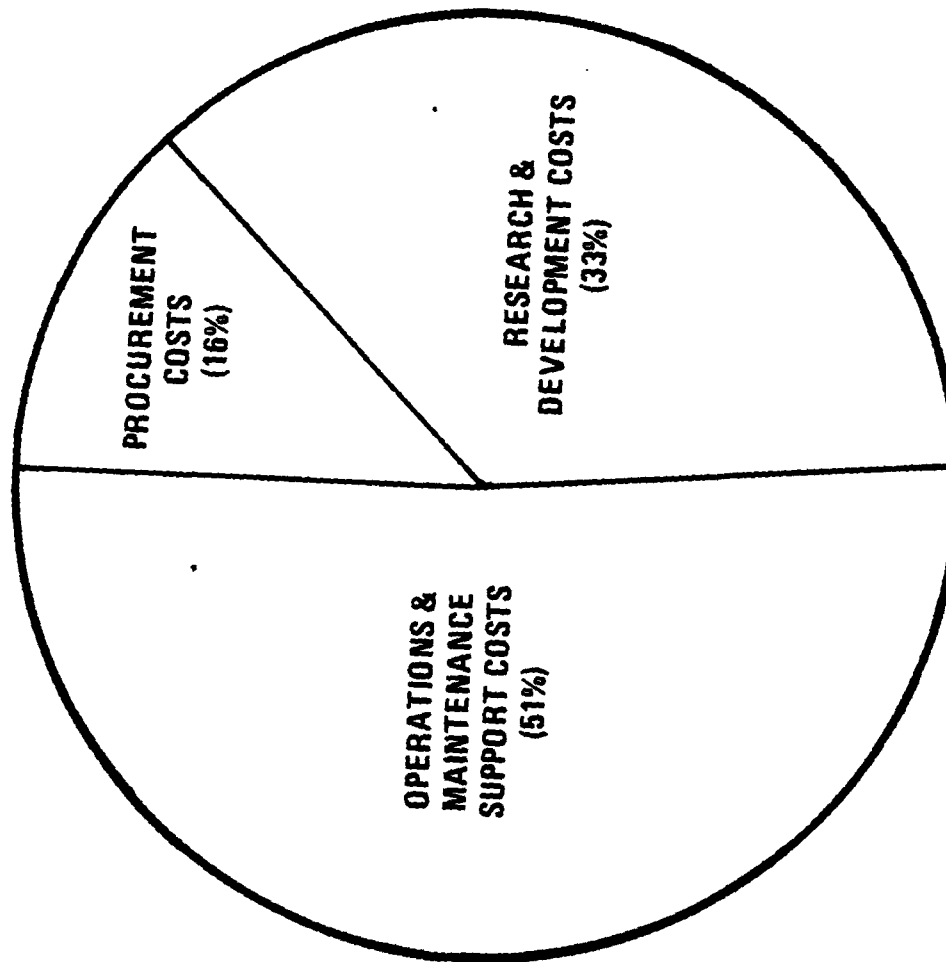
ELEMENTS OF LIFE CYCLE COST

THROUGH THE PROGRAM LIFE CYCLE



MICROPROCESSOR LIFE CYCLE COST DISTRIBUTION

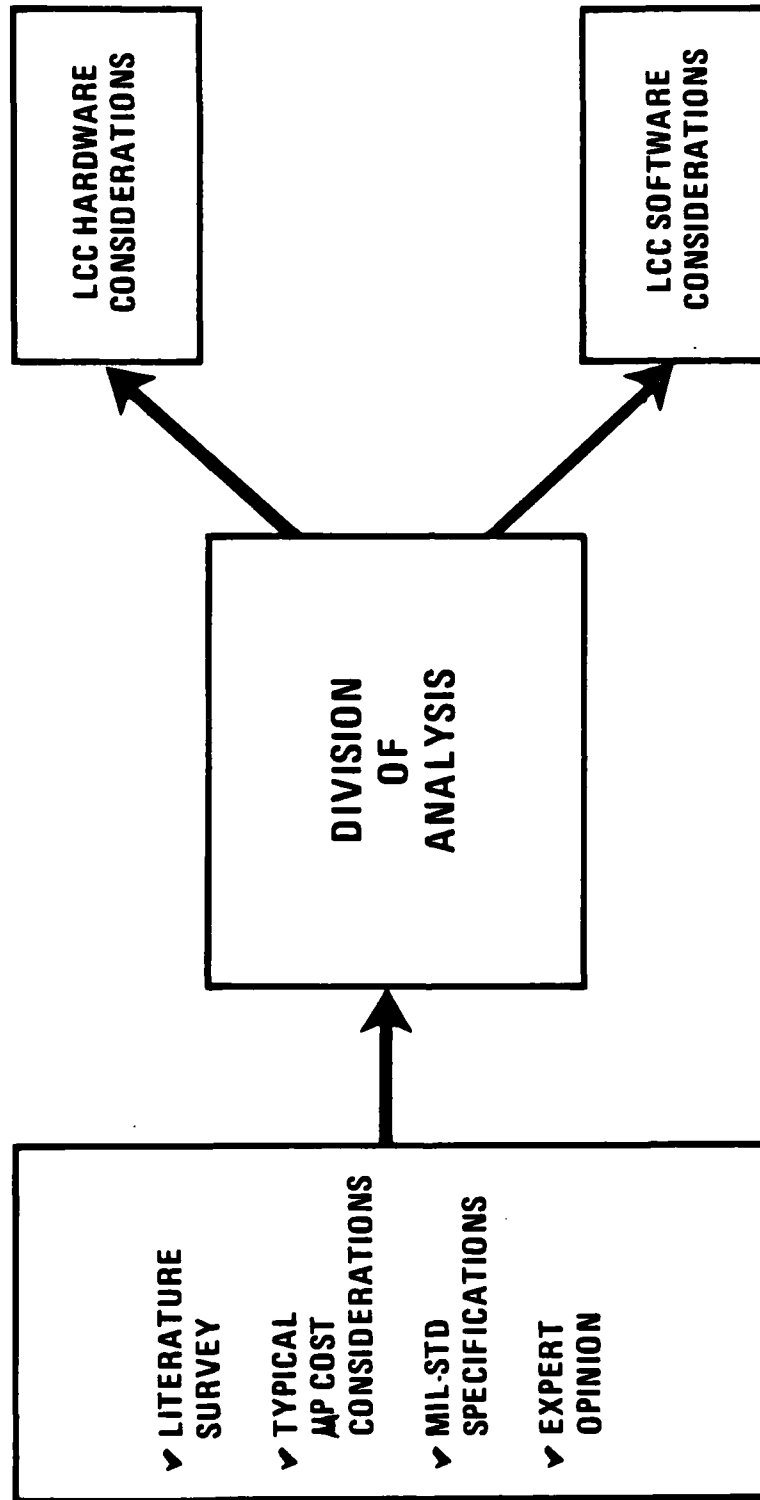
EXAMPLE OF MILITARY STANDARD PROCESSOR



* Retirement Costs Assumed Negligible

853089

LCC STUDY APPROACH



053102

MAJOR ELEMENTS OF LCC CONSIDERATION

HARDWARE LCC

- RESEARCH & DEVELOPMENT

- PRODUCTION

- OPERATIONS & SUPPORT

- ✓ Hardware Upgrades
- ✓ Spares & Inventory
- ✓ Support Hardware
- ✓ Personnel Requirements

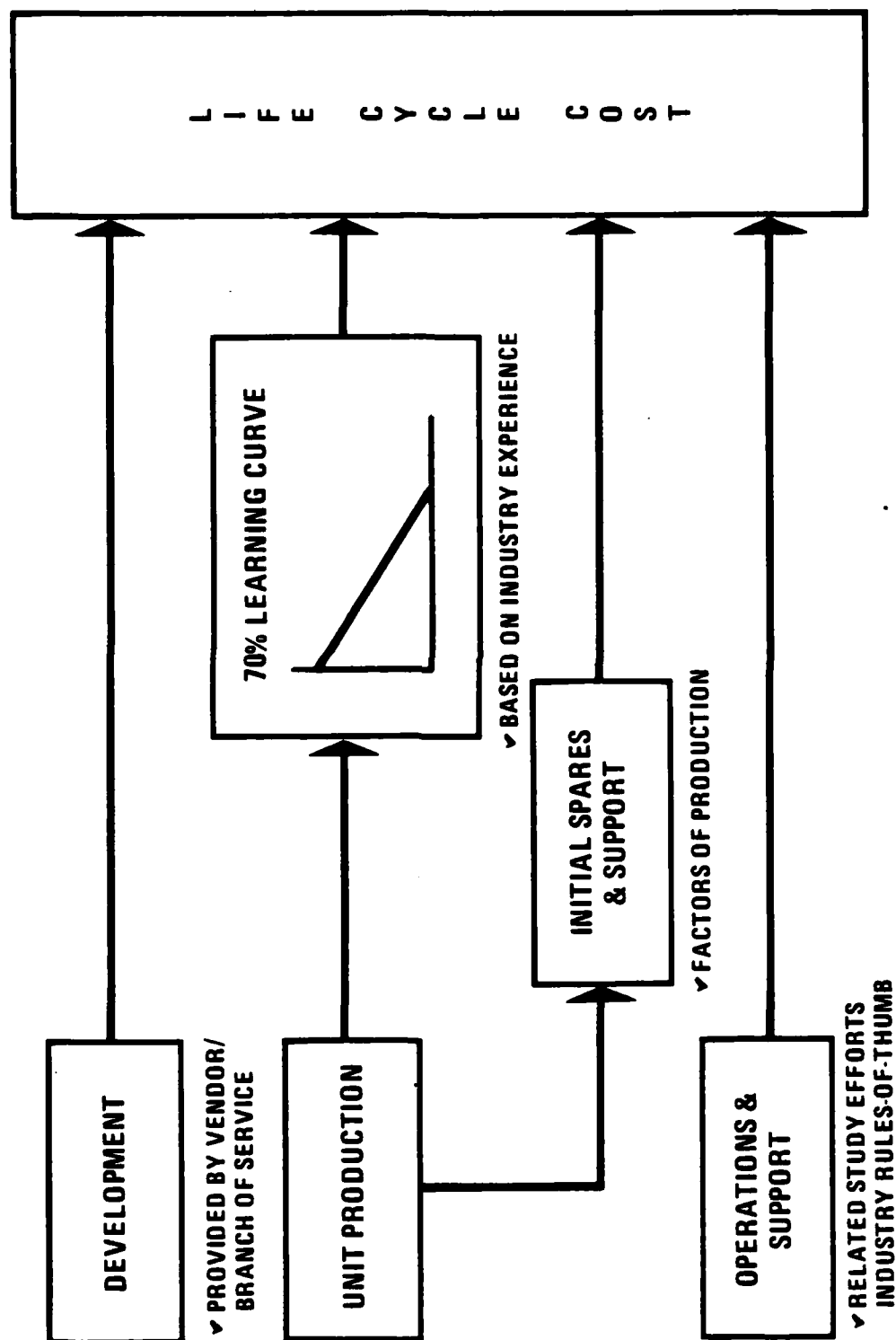
SOFTWARE LCC

- DEVELOPMENT & ACQUISITION

- OPERATIONS & SUPPORT

- ✓ System Upgrades
- ✓ Software Maintenance
- ✓ Support Maintenance
- ✓ Personnel Requirements

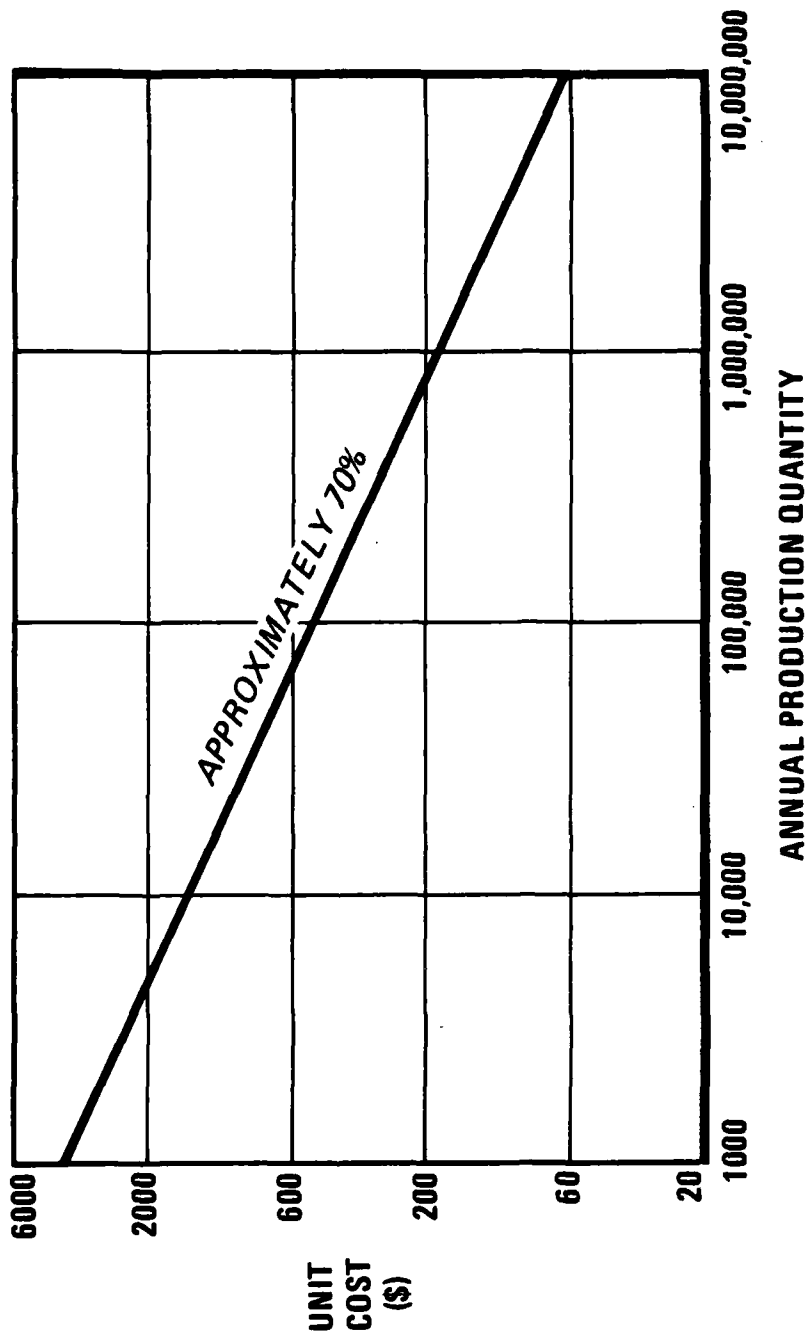
HARDWARE LCC METHODOLOGY



853100

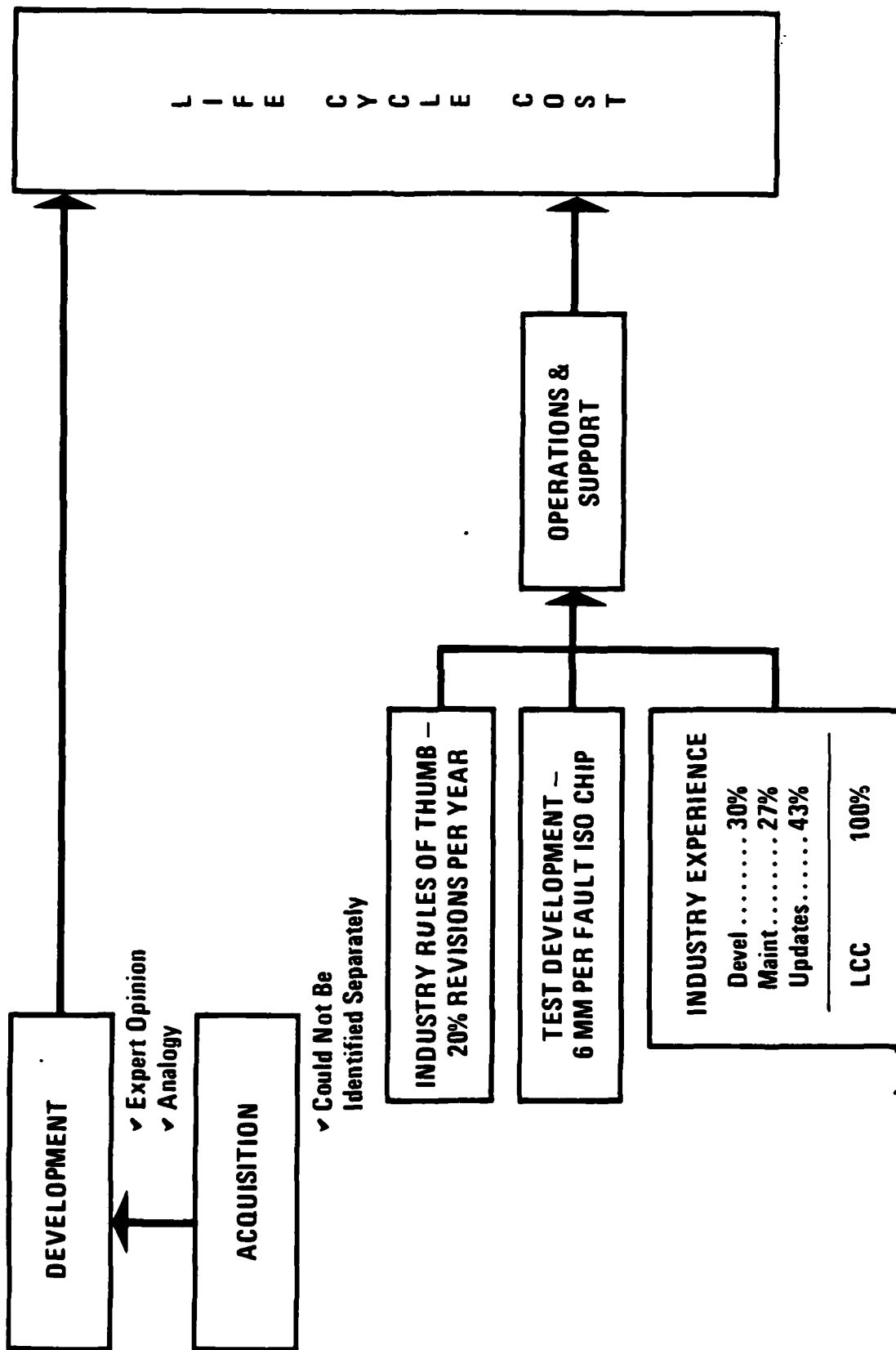
PRODUCTION "LEARNING CURVE" EXPERIENCE

(ECONOMIES OF SCALE)



MS3101

SOFTWARE LCC METHODOLOGY



BS3103

SUMMARY

MICROPROCESSOR LCC

	DEVELOPMENT	PROCUREMENT	O&S	LCC	TOTAL LCC
HARDWARE	27%	32%	41%	51%	100%
SOFTWARE	52%		48%	43%	

053105

OBSERVATIONS

● UTILITY OF METHOD

- ✓ Logical Division Along Hardware and Software Lines
- ✓ Expert Opinion for State-of-the-Art Considerations
- ✓ Industry Rules-of-Thumb/Analogy to Related Experience

● CHANGING COST DATA

- ✓ Need Exists for Documenting Cost Actuals
- ✓ Research to Identify Elements Contributing to LCC
- ✓ Data Analysis Required to Isolate Cost Trends

● POTENTIAL BENEFITS OF A STANDARD MICROPROCESSOR

- ✓ Continuous Upgrades
- ✓ Parts Standardization
- ✓ Economies of Scale

Jeff Pester

MIL-STD-1750A

MANAGING THE STANDARD

by Ronald S. Vokits/ASD/AXT

Abstract

This paper will cover the aspects of managing MIL-STD-1750A from the Control Agent's point of view. The regulations governing this activity and the responsibilities of the organizations involved will be discussed.

Introduction

MIL-STD-1750A defines the Air Force's 16 bit computer instruction set architecture (ISA). This standard describes how the computer will operate from the assembly language programmers' point of view. It describes the data formats, instruction operations, addressing modes and interrupt operation. This standard does not specify the technology, speed, or packaging. It defines an interface.

Regulations

The application and control of MIL-STD-1750A falls under three regulations: DOD 5000.5X, AFR 800-14, and AFR 800-28. The Department of Defense' draft regulation (DOD 5000.5X) provides guidance for standardization of ISA's. Its primary purpose is to reduce the life cycle cost of military systems by reducing proliferation of embedded computer systems. It requires that only approved ISA's be used in defense systems and subsystems except for three areas:

- a. Non-militarized general purpose and commercial automatic data processing computers.
- b. Automated test equipment and ground training equipment (simulators).
- c. Commercial products.

The policy covers the ISA's approved for each service. The Army will use MIL-STD-1862 (Nebula ISA) for 32 bit computers. The Navy has three approved ISA's and the Air Force has MIL-STD-1750A. The Government must have full and clearly defined rights to the ISA before it can be approved. It is then reviewed every two years. Each DOD component is directed to set up an office and procedures to process waivers on the use of standard ISA's. AFR 800-14 establishes AFSC/XRF as the single focal point for planning and policy related to the development and acquisition of computer resources. AFR 800-28 covers Air Force policy on avionics acquisition and support. Under this regulation, the Deputy for Avionics Control is established as the single Air Force organization for focusing and controlling all Air Force avionics efforts.

Control Process

The MIL-STD-1750A Control Process was established to ensure conformance with DOD and Air Force policy. The control process:

- a. Promotes the use of the ISA.
- b. Controls changes to the standard.
- c. Provides a mechanism for making clarifications and interpretations to ambiguous or conflicting parts of the standard.
- d. Establishes a means to verify that a particular company has correctly implemented the standard.
- e. Incorporates a means of granting waivers where the use of MIL-STD-1750A is not technically practical or cost effective.

The organizations involved and their responsibilities are as follows:

1. AFSC/XRF is the Designated Control Agent and must ensure that Air Force policies are followed. The DCA controls all changes to the policy and reviews and approves all AFSC waiver requests.
2. ASD/AXT is the MIL-STD-1750A Control Agent. His job is to assure a stable standard. He serves as Chairman of the Control Board and establishes and maintains the Control Facility. The Control Agent is the main interface between the Users Group and the Air Force. He does not grant waivers, but he reviews the requests and makes recommendations to AFSC/XRF.
3. The Control Facility (part of the Systems Engineering and Analysis Facility (SEAFAC) provides technical support to the Control Agent. This facility is manned by ASD/ENASF. Their job includes performing the architectural verification test, reviewing and evaluating all changes recommended by the Users Group and providing interpretation and clarification to the standard. Technical questions on the standard and its implementation are answered by this group.
4. The Control Board consists of representatives from the Air Force's user community. Its membership includes the DCA, CA, and the AFSC and AFLC organizational focal points. This group operates using published control procedures. The main function is to review and vote on changes suggested by the Users Group.
5. The Users Group is composed of industry and DOD representatives interested in promoting the use of this standard ISA. It is an open forum working group. The DOD membership participates but does not vote on issues.

THE CHANGE PROCESS

The change process begins when a member of the Users Group submits a Change Proposal to the Chairman of the User Group. The proposal is given a number for tracking purposes, and assigned to one of the four committees (Architecture, Standards, Tools, and Verification) for review and recommendations. If the proposal is approved by a two-thirds majority of the Users Group, it is sent to the Control Board. The Control Facility evaluates the proposed change and prepares a report for the Control Board. The Control Board reviews the change, hears the report and then votes. If rejected, the proposal goes back to the Users Group for reconsideration. If approved, the proposed change becomes a candidate for the next revision to the standard subject to the final approval or disapproval of the Designated Control Agent.

The primary criterion for accepting a change is maintaining upward compatibility with the hardware. Some changes have been judged to be of the type required to correct a problem. These will diminish as the standard matures. Some changes are approved on the basis of clarification or interpretation. These changes are usually initiated because of misunderstanding or misinterpreting the standard. Caution now must be used as extensions are proposed to give added capabilities or increase performance. The Air Force will not benefit from ISA standardization unless the standard is stable enough for manufacturers to design to, set up production and be competitive.

A Mature Standard

MIL-STD-1750A was derived from the DAIS AYK-15 ISA. Industry was later invited to review and comment on the standard. The basic standard was published on 21 February 1979, after three reviews. The Users Group was established in August 1979 with over 40 systems, computer and software companies participating. The "A" version was published in July 1980 after four meetings of the Users Group. The "A" version added the expanded memory options and other clarifications. Since then the Users Group has continued to meet and further refine the standard. Change Notice 1, which corrects some problems in the expanded memory update and incorporates the built-in-function options, will be formally published in July 1982. The Control Board decided that the standard would be frozen for three years. Additional changes will be considered, but will not be incorporated formally until this stabilizing period has lapsed.

An indication to the maturity of the standard is the number of Air Force systems now incorporating MIL-STD-1750A. These are:

F-16 MSIP Fire Control Computer	Delco Electronics
LANTIRN POD	Delco Electronics
F-16 MSIP Equipment	Fairchild
Wide Field-of-View HUD for F-16 and A-10	Marconi
F-111 Computer	Singer-Kearfott
F-5G Computer	Teledyne
F-5G Radar (G.E)	Tracor
MATE	Sperry Univac
B-1 Radar	Westinghouse

Verification

Verification is the process the Air Force uses to ensure that a vendor's computer complies with MIL-STD-1750A. The Architectural Test Procedure (ATP) is conducted by SEAFAC personnel. The test checks only the ISA. It does not test for through-out performance or specification compliance. It is not an acceptance test. The verification test is conducted by hand-loading some communication routines that enable loading the verification test software. Once this is done, the machine tests itself by performing instruction according to the software and comparing the results. The results are downloaded and used to write a test report. The report describes the areas where the computer meets or fails to meet the standard. The test report is given only to the organization requesting the verification test.

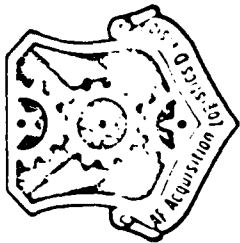
Arrangements for the verification test are made by letter to Mr. Ron Vokits, ASD/AXT, Wright-Patterson AFB, OH 45433. The contractor must include the following within the processor to be tested:

- a. 64K of memory (test can be performed with 32K)
- b. Front panel capability
- c. RS-232 I/O channel controlled by XIO commands
- d. Software subroutines to allow communication over the
RS 232 channel.

The government provides all other software involved in the architectural test procedure.

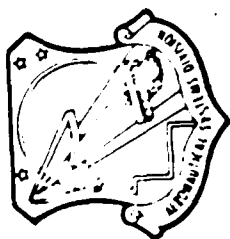
Summary

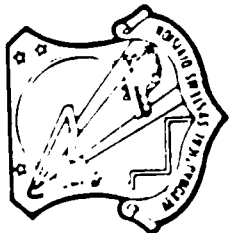
It is the job of the MIL-STD-1750A Control Agent to manage and control the standard. The Control Process has been documented and is functioning. The standard is mature and is frozen for three years. Another part of the job is to ensure that MIL-STD-1750A is correctly implemented. This job is done by the Control Facility by using the Architectural Test Procedure. A similar control process has been set up for the MIL-STD-1589B (JOVIAL).



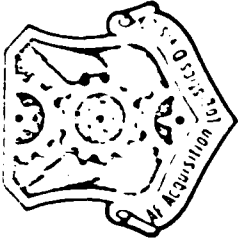
MIL-STD-1750A

MANAGING THE STANDARD

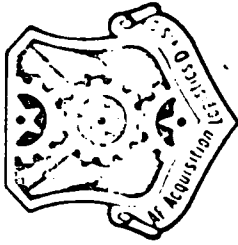
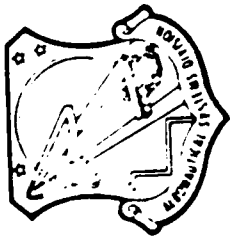




OUTLINE



- REGULATIONS
- ORGANIZATIONS INVOLVED
- A MATURE STANDARD
- VERIFICATION



DOD 5000.5X

PURPOSE: GUIDANCE FOR STANDARDIZATION OF ISA'S

SCOPE: EMBEDDED DIGITAL COMPUTERS & PROCESSOR

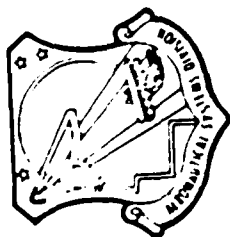
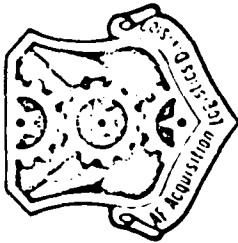
EXCEPTIONS: A. NON-MILITARIZED GEN PURPOSE
& COMMERCIAL ADP
B. ATE & SIMULATORS
C. COMMERCIAL PRODUCTS

POLICY: APPROVED DOD ISA

ARMY - NEBULA (MIL-STD-1862)
NAVY - (3 ISA'S)
AF - MIL-STD-1750A

REVIEWED EVERY TWO YEARS

DOD HAS FULL AND CLEARLY DEFINED RIGHTS
DOD COMPONENT ORGANIZATIONS SET UP AN OFFICE
AND PROCEDURES TO PROCESS WAIVERS

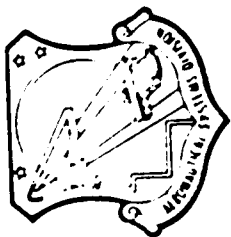


AFR 800-14

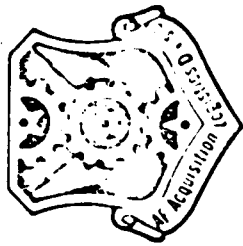
- MANAGEMENT OF COMPUTER RESOURCES IN SYSTEMS
- AFSC/XRF IS THE SINGLE FOCAL POINT FOR PLANNING & POLICY RELATED TO THE DEVELOPMENT AND ACQUISITION OF COMPUTER RESOURCES (AFSC SUPPLEMENT).

AFR 800-28

- AIR FORCE POLICY ON AVIONICS ACQUISITION AND SUPPORT
- AX IS THE SINGLE AIR FORCE ORGANIZATION RESPONSIBLE FOR FOCUSING AND CONTROLLING ALL AIR FORCE AVIONICS EFFORTS.

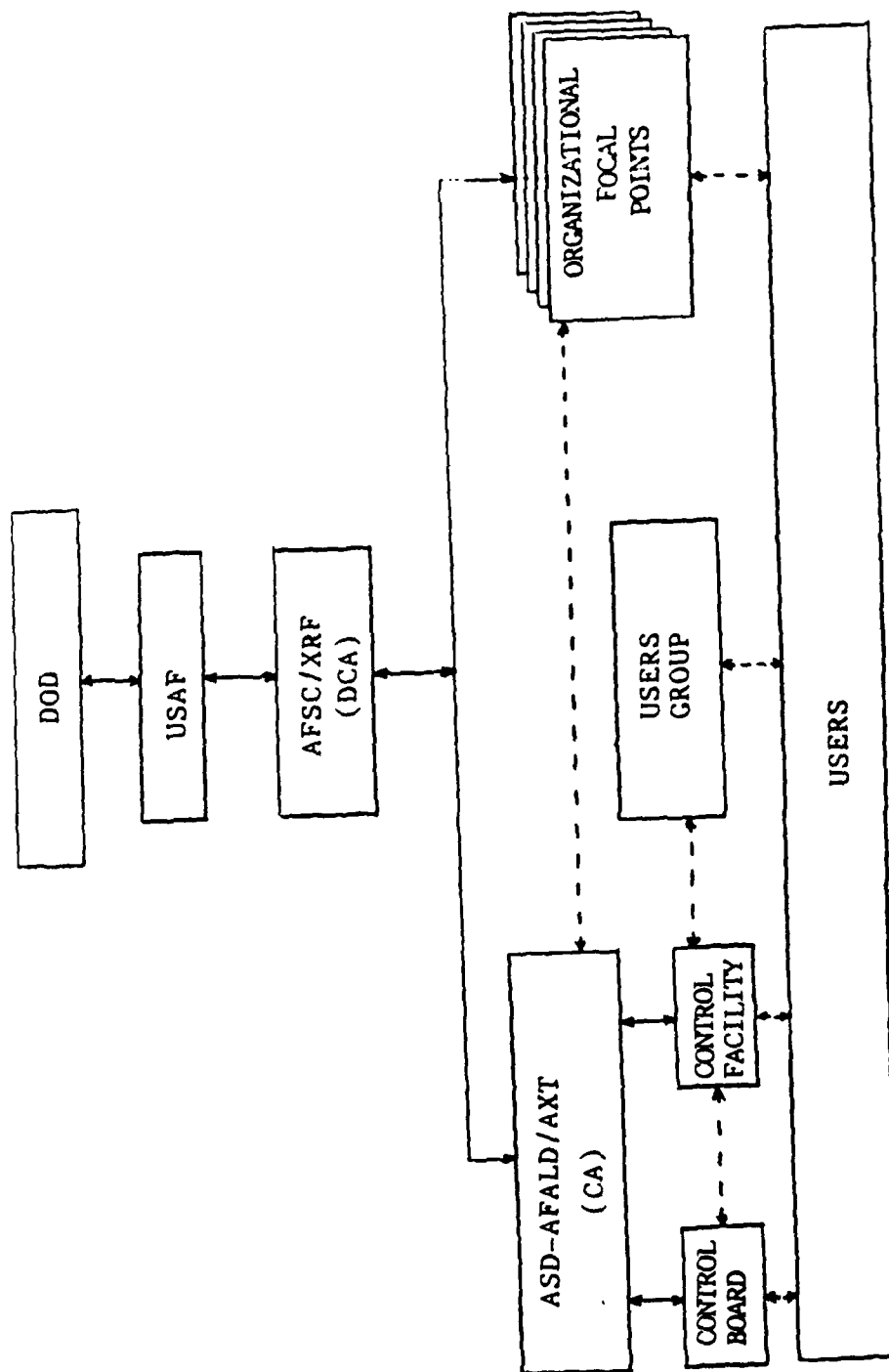


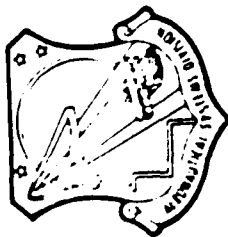
MIL-STD-1750A CONTROL PROCESS



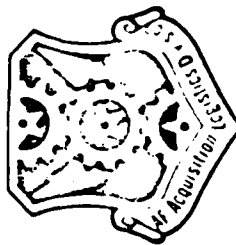
- CONFORMANCE WITH DOD POLICY
- PROMOTE USE OF THE ISA
- CONTROL CHANGES
- CLARIFICATIONS, INTERPRETATIONS
- VERIFICATION
- WAIVER PROCESS

ORGANIZATIONS INVOLVED

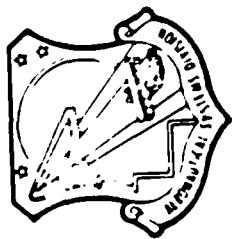




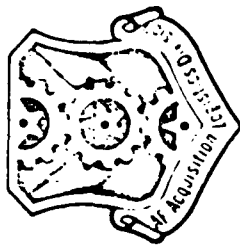
MATURE STANDARD



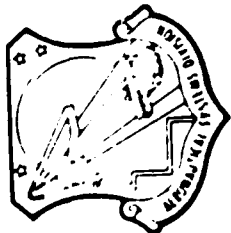
- DAIS AN AYK-15 ISA WAS THE BASELINE
15 SYSTEMS BUILT
- INDUSTRY WAS INVITED TO REVIEW AND COMMENT ON STANDARD
THREE REVIEW CYCLES - JAN 78, JUN 78, OCT 78
OFFICIAL STANDARD - 21 FEB 79
- USERS GROUP WAS ESTABLISHED AUG 79
OVER 40 SYSTEMS, COMPUTER, AND SOFTWARE COMPANIES PARTICIPATING
FOUR MEETINGS TO DISCUSS & RECOMMEND CHANGES TO MIL-STD-1750
- MIL-STD-1750A DATED 2 JULY 1980
- CHANGE NOTICE 1 DATED 1 JULY 1982



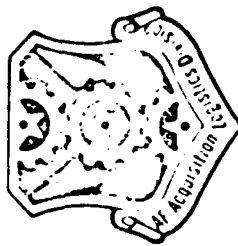
REAL PROGRAM APPLICATIONS



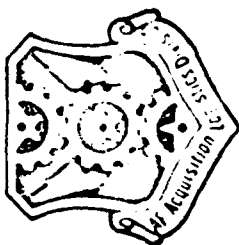
F-16 FIRE CONTROL COMPUTER	DELCO ELECTRONICS
LANTIRN POD	DELCO ELECTRONICS
F-16 MSIP EQUIPMENT	FAIRCHILD CAMERA & INS CORP
WIDE FIELD-OF-VIEW HUD FOR F-16 & A-10	MARCONI
F-111 COMPUTER	SINGER-KEARFOTT
F-5G COMPUTER	TELEDYNE
F-5G RADAR (G.E.)	TRACOR



VERIFICATION



- ARRANGEMENTS MADE THROUGH ASD/AXT (CONTROL AGENT)
- TEST CONDUCTED BY ASD/ENASF (SEAFAC)
- COMPLIANCE WITH ISA STANDARD
- NOT AN ACCEPTANCE TEST
- METHOD - LOAD ATP - MACHINE TESTS SELF
- TEST REPORT

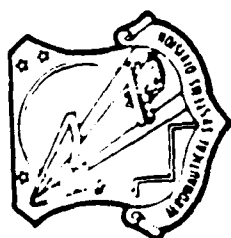


VERIFICATION REQUIREMENTS

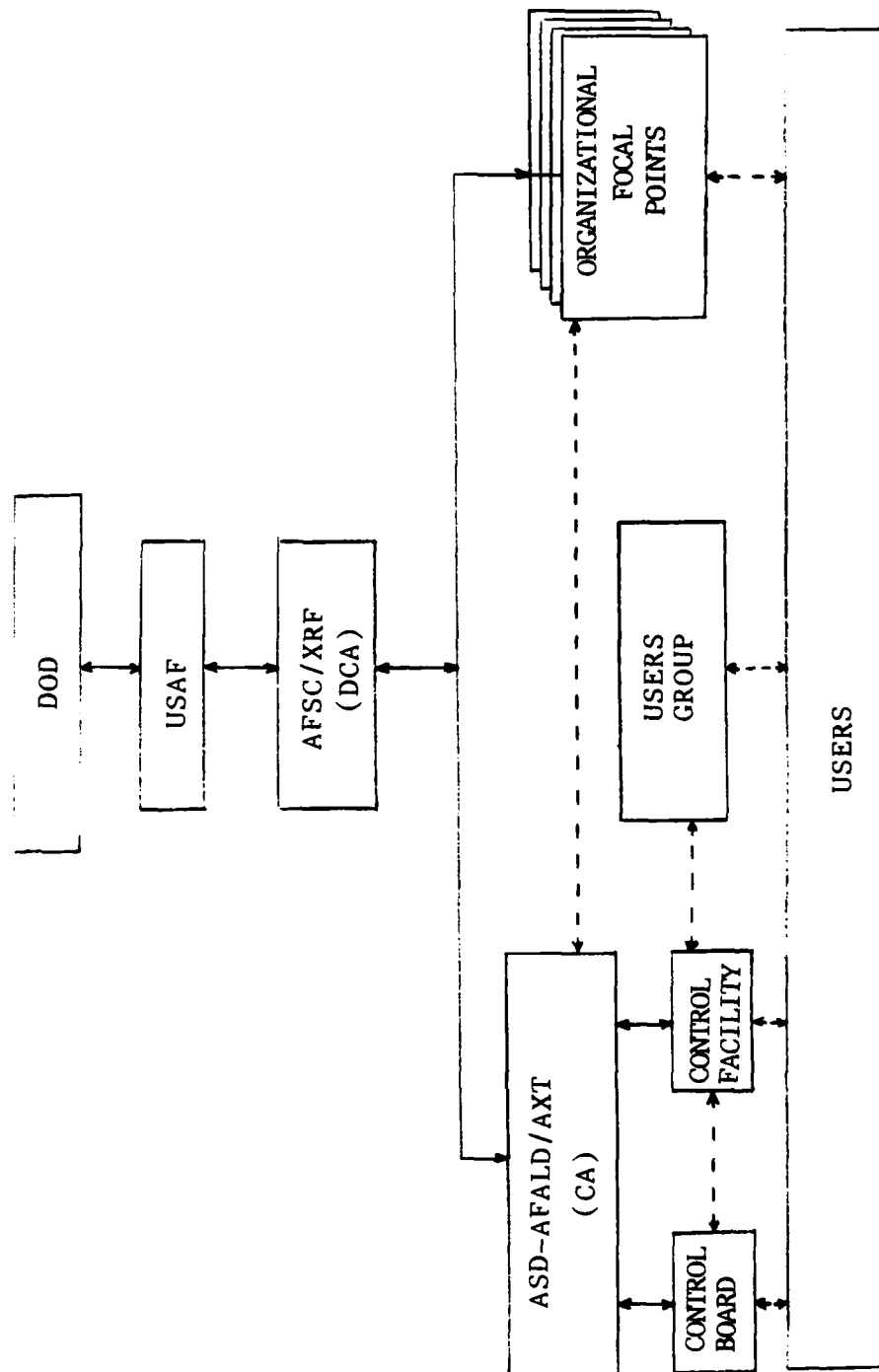
HARDWARE: PROCESSOR WITH AT LEAST 32K OF MEMORY
 FRONT PANEL CAPABILITY
 RS-232 I/O CHANNEL CONTROLLED BY XIO COMMANDS

SOFTWARE: I/O DRIVER TO ALLOW COMMUNICATION OVER RS-232
 I/O CHANNEL

(GOVERNMENT PROVIDES ALL OTHER SOFTWARE)



ORGANIZATIONS INVOLVED



TRANSITION PLAN TO UPGRADE THE DIGITAL INTEGRATING SUBSYSTEM
TO MIL-STD-1750A FOR MRASM

ABSTRACT

J. B. Reed
MRASM Chief Engineer

F. W. Purdy
MRASM Chief Product Software

The Medium Range Air to Surface Missile (MRASM), currently in development for joint Air Force and Navy usage, employs an array of micro processors to perform on-board computation. Because of program schedule, which calls for IOC in 1985, and because of severe size, weight, environmental, and power constraints, the architecture uses two Z8000 based Digital Integrating Subsystem (DIS) computers developed in 1978-1980 for the AF Armament Division. The evolution of the F-16 MSIP LSI chip processor and support software, MIL-STD-1750A, and DIS hardware allows an orderly, minimum cost, transition development to MIL-STD-1750A. The result of the DIS transition effort will be a state-of-the-art processor and support software, which will be directly applicable to MRASM, and may find broader future application. This paper addresses the transition plan to upgrade DIS to 1750A.

BACKGROUND

The Digital Integration Subsystem (DIS) Computer has been developed and flight test demonstrated over the past several years to fill the Air Force's need for a small, lightweight, low power, missile-borne processor system. During the initial phases of its evolution, the best available single chip microprocessor for that application was the Z8000, and it was used as the core element in the system. A great deal of flight hardware, (especially missile tailored I/O support), test software, and development support software has been created in support of its application both in the Air Force Midcourse Guidance Demonstration (MGD) program, and for the Medium Range Air to Surface Missile (MRASM) program.

During the development of DIS, MIL-Standard 1750 matured to Mil-Standard 1750A, the value of the standard became widely understood and accepted and the F-16 MSIP program undertook the development of both a low cost, single chip processor that can execute the full instruction set and an elegant set of development support software. But the need for the low cost, low power standardized missile-borne processor remains a real one, and the MRASM program is committed to upgrade its processors to MIL-STD-1750A.

The first five figures depict the DIS 1A hardware, its environmental requirements and implementation groundrules, the DIS 1A hardware provided to MRASM as legacy, the MRASM hardware development, the DIS support software provided as legacy, and the MRASM developed support software.

TRANSITION PLAN OVERVIEW

A management assessment of MRASM requirements, existing hardware and software, and MSIP program plans and status was made. This assessment resulted in the following groundrules:

- Full Compliance with MIL-STD-1750
 - Ensures maximum utilization of software tools
 - Allows use of future S/W tool enhancements
 - Provides floating point and other instruction set advantages
- Minimum Program Nonrecurring Cost through:
 - Maximum retention of MRASM/DIS I/O hardware
 - Black box functional identity
 - Maximum utilization of F-16 MSIP support software
 - Maximum use of MRASM J-73 application software through recompilation

TRANSITION PLAN OVERVIEW (Continued)

- Use MSIP VLSI CPU chips to retain low cost advantage of single chip microprocessor
- Schedule to emplace all hardware and support software paced by VLSI development
- Flight test qualify 1750 computers and S/W MRASM variant

Figure 6 pictorially depicts the overall hardware/software transition elements and strategy broken down into the three major elements; flight hardware, flight and support software, and test hardware and software.

As can be seen, DIS provides a basic flight qualified package of the packaging standards and the high technology memory elements which can all be reused directly in the 1750 DIS. MRASM provides all of the system peculiar I/O. The VLSI chips from MSIP will be used to provide the core processing elements. The transition plan modifies the I/O controller as well as the memory controller to conform to the split I/O and memory buses, and develops the CPU board.

MSIP will provide the core elements for the 1750 DIS support software: the JOVIAL cross-compiler and 1750A cross assembler. Modifications to the existing DIS linker loader and DIS interpretive computer simulator will interface the MSIP compiler and assembler to the software debug and test tools implemented on MRASM. Transition of the MRASM Operational Flight Software (OFS) to 1750 DIS will require recoding of the DIS operating system only; applications tasks can be recompiled from the JOVIAL source.

In the area of hardware test, since only three cards are changing, and black box functional identity can be enforced, only a minimum set of new hardware and software is required as shown.

HARDWARE DEVELOPMENT PLAN

Figure 7 details the current estimate of hardware design parameters for the three circuit boards requiring change.

Figure 8 shows the hardware development schedule currently in work to develop the 1750 DIS boards and units.

FLIGHT AND SUPPORT SOFTWARE DEVELOPMENT

The MRASM transition to 1750 requires rehosting the existing OFS developed for DIS 1A and revalidating the rehosted OFS. For the majority of the OFS, written in MIL-STD-1589 A and B compatible JOVIAL, this is accomplished by recompiling the source code with the MSIP compiler. As shown in Figure 9, the common DIS operating system and certain routines in the guidance and navigation computer are written in assembly language. These will be translated to the MSIP/IEEE Standard 1750A assembly language and reassembled.

The support software suite required for the MRASM OFS rehost is produced by a marriage of MSIP and DIS 1A tools. The MSIP JOVIAL compiler and 1750A assembler, depicted in Figure 10, will be used unmodified. Modifying the MRASM DIS Linker Loader (DLL) provides the interface to existing DIS software load, debug, and test tools. Figure 11 shows the DLL modified to accept the relocatable object and relocatable debug files as produced by the MSIP programs. DLL builds an absolute load file that includes debug data supporting interactive symbolic (JOVIAL level) debug using the DIS test tools.

DIS Control-Monitor-Load (CML) is one of the programs that accepts the DLL loadfile. As described in Figure 11, CML executes on a DIS Diagnostic Station (DDS) providing interactive download and software debug capabilities with a DIS target computer. CML will require minor modification to download to a 1750 DIS.

MRASM is developing a DIS interpretive computer simulator to provide flexibility and high visibility during unit/subprogram checkout in an environment that decouples early hardware and software testing. The DIS ICS, shown in Figure 12, accepts the DLL loadfile and has the same interactive user interface as CML. Several approaches to developing a 1750 DIS ICS are being traded.

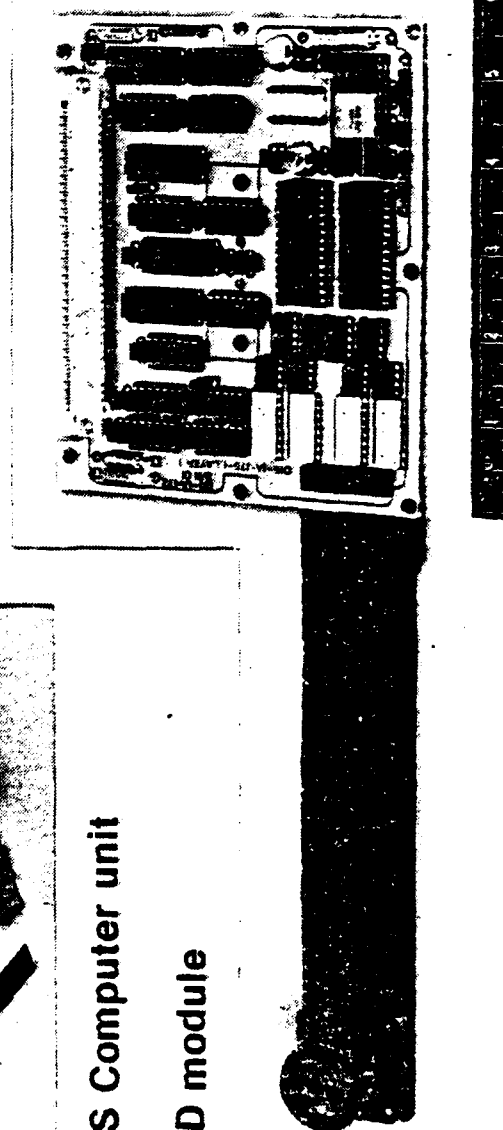
Figure 8 shows the plan for developing the 1750 DIS support software tools.

DIS-1A COMPUTER PACKAGED FOR CRUISE MISSILE ENVIRONMENT

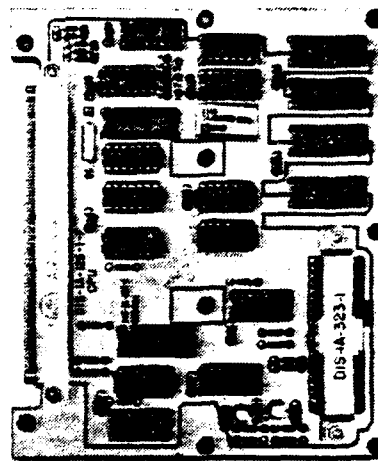
- DIS Block 1 A design is complete
- Environmental qual tests in 1982
 - CPU
 - 64k E² PROM/RAM memory
 - New power supply
 - Discrete
- IOC
- A/D
- D/A
- BIU



DIS Computer unit
A/D module



CPU module



290120318

FIGURE 1

DIS 1A PROJECT REPACKAGED DIS I FOR MRASM REQUIREMENTS

PHYSICAL DESIGN REQUIREMENTS

- TEMPERATURE: -55°C TO +75°C COLD PLATE TEMP
- VIBRATION: 4.0 GRMS SINE, 250 TO 2000 Hz PLUS 5.1 GRMS EQUIVALENT RANDOM 10 TO 2000 Hz
- SHOCK: 9000 G'S @ 8000 Hz
- PRESSURE: 1.33 PSIA TO 120 PSIA
- DESIGN TO COST
 - TARGET COST: MCM \$23K, GNC \$16K
 - CHALLENGE COST: MCM \$16K, GNC \$11K

PHYSICAL DESIGN IMPLEMENTATION

- MULTI SOURCE MIL SPEC ELECTRONIC PIECE PARTS
- STRICT DERATING POLICY - 110°C MAX JUNCTION TEMP
- STATE-OF-THE ART PACKAGING TECHNIQUES
 - MULTI-LAYER CIRCUIT BOARDS
 - RIGI-FLEX CIRCUIT BOARDS
 - 10 MIL MINIMUM LINES AND LINE SPACING (10% COPPER)
- REPACKAGED POWER SUPPLY
 - DIRECT THERMAL PATH FROM COMPONENTS TO COLD PLATE
 - NO INTERNAL WIRING
 - EASILY EXPANDABLE

FIGURE 2

DIS 1 AND 1A PROJECT DEVELOPED A BASIC COMPUTER FOR MISSILE-BOUND APPLICATION

- 0 Z8000 BASED CPU
- 0 I/O CONTROLLER (SUPPORTS UP TO FOUR STANDARD I/O MODULES)
- 0 DIS MUX BUS INTERFACE UNIT
- 0 1553 BUS INTERFACE UNIT
- 0 DIRECT MEMORY ACCESS
- 0 A/D INPUT MODULE
- 0 D/A OUTPUT MODULE
- 0 DISCRETE OUTPUT MODULE
- 0 64K E²/RAM MEMORY
- 0 SERIAL I/O MODULE
- 0 PARALLEL I/O MODULE
- 0 POWER SUPPLY
- 0 HOUSING
- 0 STANDARDIZED BACKPLANE

FIGURE 3

MRASM PECULIAR REQUIREMENTS INTRODUCED ADDITIONAL CHANGE TO

DIS IA

CHANGES: 96K MEMORY AND ADDRESS CAPABILITY FOR GNC (10/81)

POWER SUPPLY CAPABILITY INCREASED TO HIGHER LOAD FOR MISSION CONTROL MODULE (MCM)

EXPANDED I/O CAPABILITY IN MCM (I/O EXPANDER)

NEW MRASM PECULIAR I/O MODULES

- PYRO CNT
- POWER SWITCHING
- BIT
- COMBINED SERIAL/DISCRETE
- RALT

MGD I/O MODULES PACKAGED FOR SOFTWARE TESTING

- RS 232
- PARALLEL I/O

FIGURE 4

SUPPORT SOFTWARE SYSTEM DEVELOPED BY DIS BEING UPGRADED FOR MRASM

- o DIS PROJECT DEVELOPED TOOLS
 - JOVIAL COMPILER (1589A) TARGETED FOR DIS
 - DIS CROSS ASSEMBLER
 - REAL TIME DIS OPERATING SYSTEM
 - DIS COMPUTER CONTROL, MONITOR AND MEMORY LOADER
- o MRASM PROGRAM UPGRADES
 - JOVIAL COMPILER INCORPORATES SYMBOL TABLE INSTRUMENTATION FOR SYMBOLIC DEBUG AND TARGET DEPENDENT OPTIMIZATIONS
 - DIS LINKING LOADER UPDATED TO SUPPORT SYMBOLIC DEBUG
 - DIS COMPUTER CONTROL, MONITOR AND MEMORY LOADER UPDATED TO PROVIDE INTERACTIVE JOVIAL LEVEL SYMBOLIC DEBUGGING ON DIS TARGET COMPUTER
 - REAL TIME OPERATING SYSTEM STREAMLINED TO MINIMIZE CPU OVERHEAD AXEL MEMORY UTILIZATION
 - MICROTEC Z8002 INSTRUCTION LEVEL SIMULATOR UPGRADED TO FULL DIS INTERPRETIVE COMPUTER SIMULATION WITH JOVIAL SYMBOLIC DEBUGGING CAPABILITY

FIGURE 5

DIS-1750 TRANSITION IS STRAIGHT FORWARD APPLICATION OF EXISTING AND ADMITTED TECHNOLOGY DEVELOPMENT

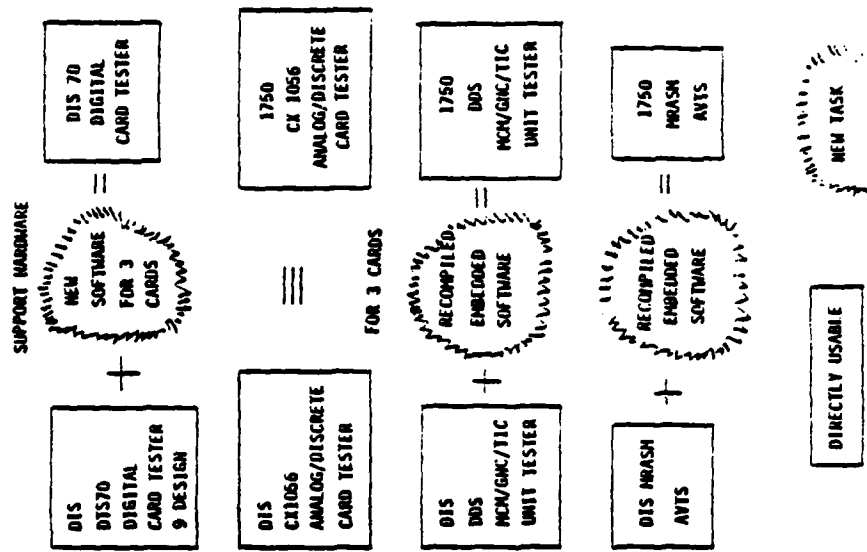
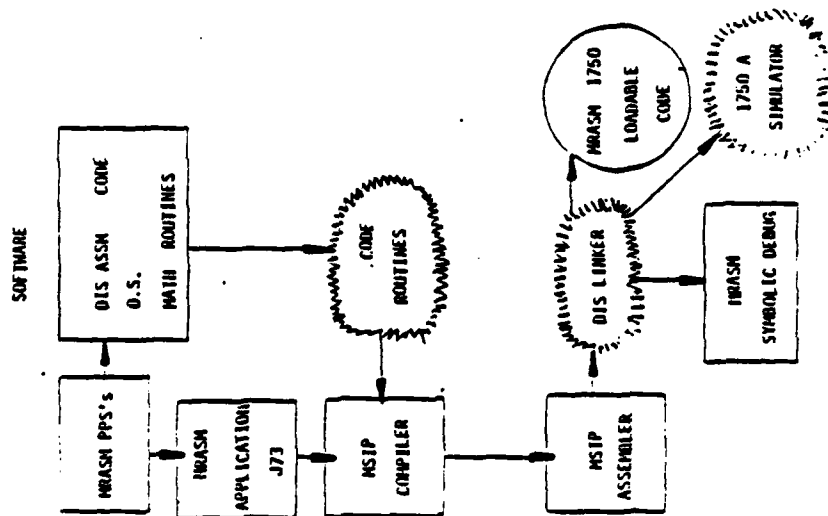
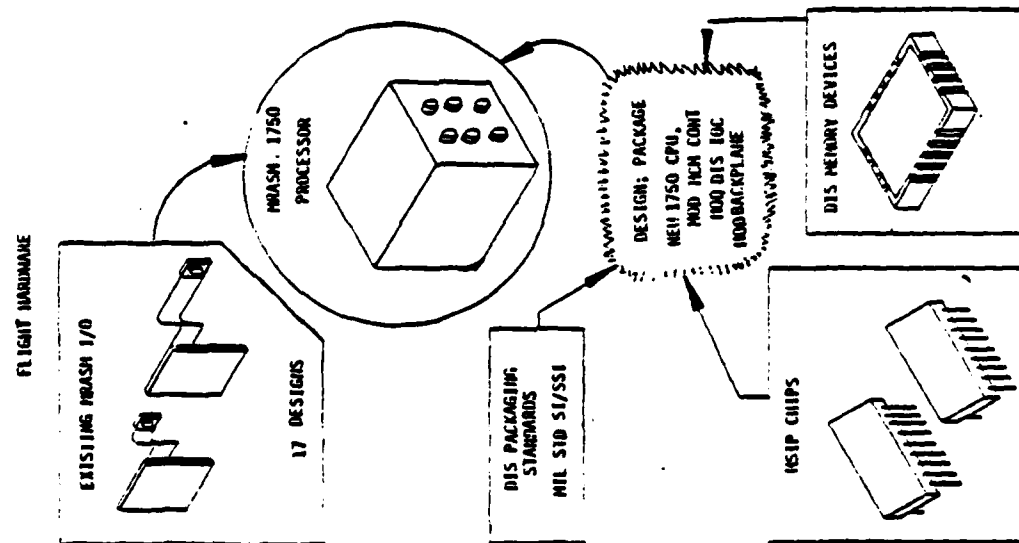


FIGURE 6

1750 DIS

CONVERSION REQUIRES MODIFICATION OF ONLY THREE CARDS

	BLOCK 1A	1750A
CPU	1 23 23 DIPS 7 WATTS Z8002 INSTRUCTION SET % CIRCUIT CHANGE*	1 30 30 DIPS 10 WATTS MIL-STD-1750A 100%
IOC	1 DOUBLE BOARD 95 13 DIPS 82 FLAT PACK 12.5 WATTS 4 I/O CHANNEL % CIRCUIT CHANGE	1 DOUBLE BOARD 50 50 DIPS 6.6 WATTS 16 30%
MEMORY	1 DOUBLE BOARD** 99 35 DIP 64 LCC 10 W 64 K WORDS MEMORY SIZE % CIRCUIT CHANGE	1 DOUBLE BOARD 99 35 DIP 64 LCC 10 W 64 K WORDS 10%***

* MECHANICAL DESIGN BOARD SIZE AND PACKAGE APPROACH REMAIN THE SAME.

** ONE HALF CONTAINS ALL BUS INTERFACE, DECODING AND PROGRAMMING CIRCUITS. THE OTHER HALF CONTAINS ONLY THE MEMORY ARRAY.

*** THE MEMORY ARRAY WILL REMAIN EXACTLY THE SAME.

FIGURE 7

MRASM - FSED 1750 DIS DEVELOPMENT SCHEDULE

15 APR 1982

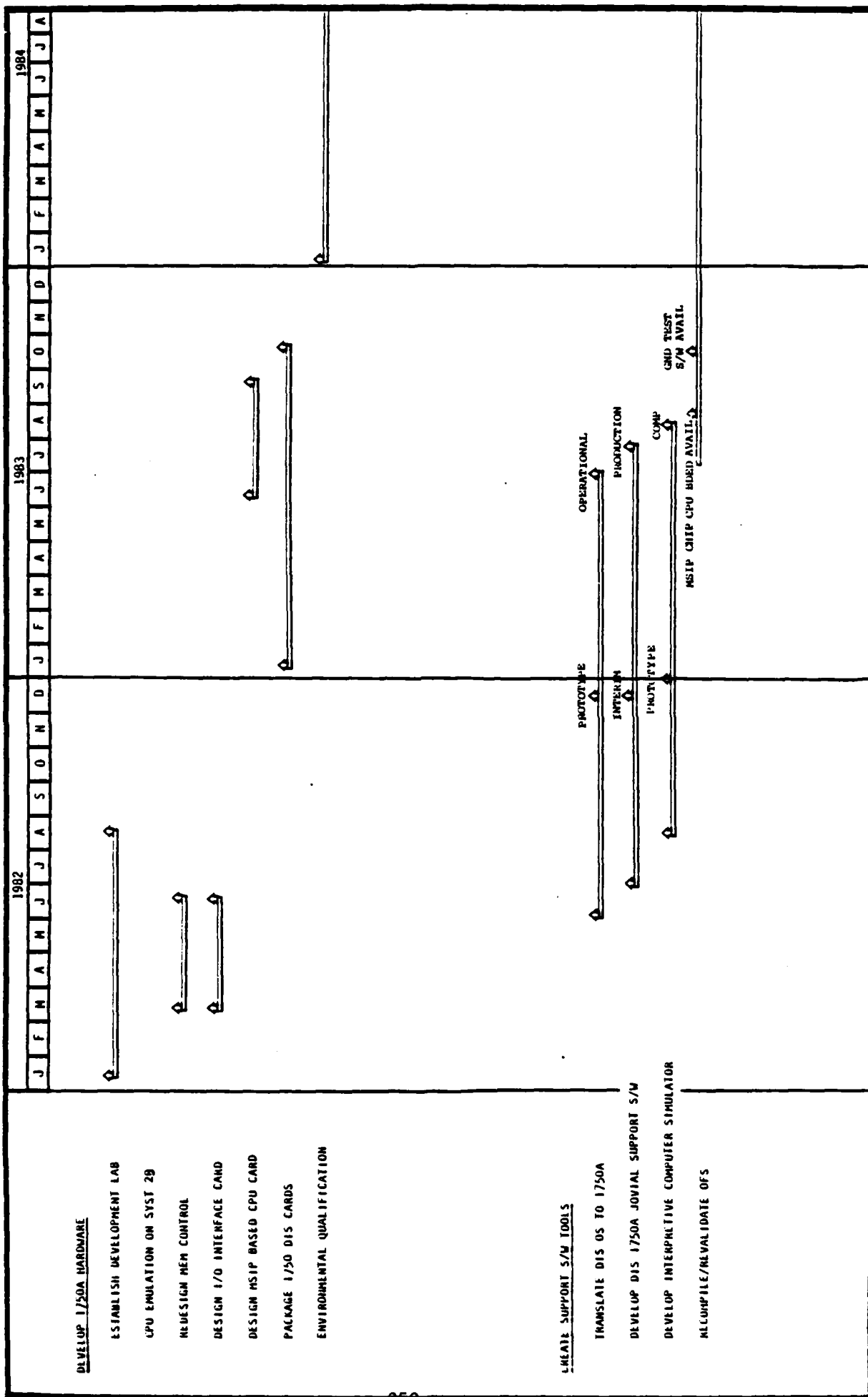
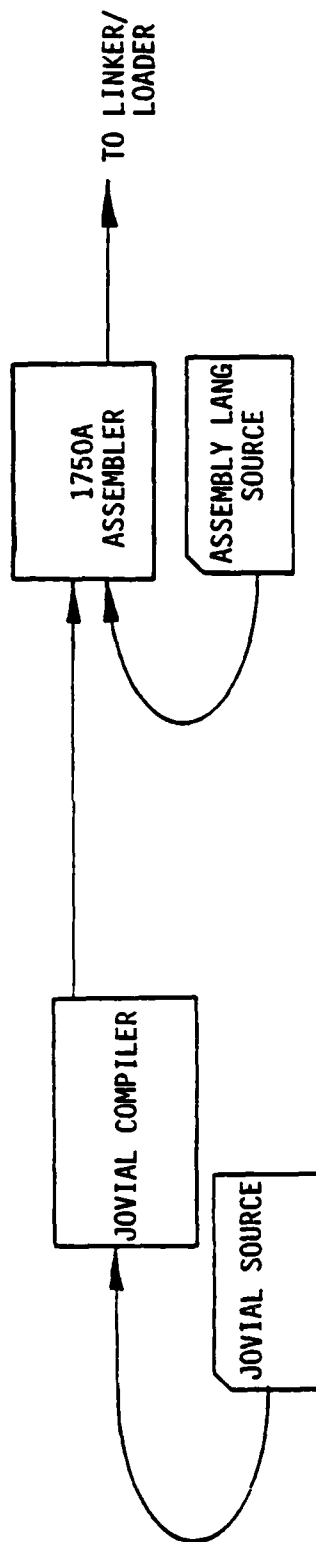


FIGURE 8

SOURCE CODE IN COMPATIBLE SUBSETS MINIMIZE MRASM OFS REHOST EFFORT



• DIS OPERATING SYSTEM

• GNC MATH ROUTINES

• TRANSLATE TO 1750A SYMBOLIC ASSEMBLY LANGUAGE (IEEE STANDARD)

• RERUN VALIDATION/ACCEPTANCE TESTS

• FASTER CPU WOULD ALLOW RECODING MATH ROUTINES IN JOVIAL

• MACHINE ARCHITECTURE DEPENDENT FUNCTIONS OF OS REQUIRE MODIFICATION

• EXTENDED MEMORY

• I/O DRIVERS

• INTERRUPT PROCESSOR

• INTERFACE BETWEEN OPERATING SYSTEM AND APPLICATION TASKS (SYSTEM CALLS) REMAIN UNCHANGED

• FLIGHT SOFTWARE

• TEST SOFTWARE

• RECOMPILE FOR 1750A TARGET

• RERUN VALIDATION/ACCEPTANCE TESTS

• WRITTEN IN MIL-STD-1589 A & B COMPATIBLE JOVIAL

FIGURE 9.

MSIP JOVIAL COMPILER AND ASSEMBLER REQUIRE NO MODIFICATIONS

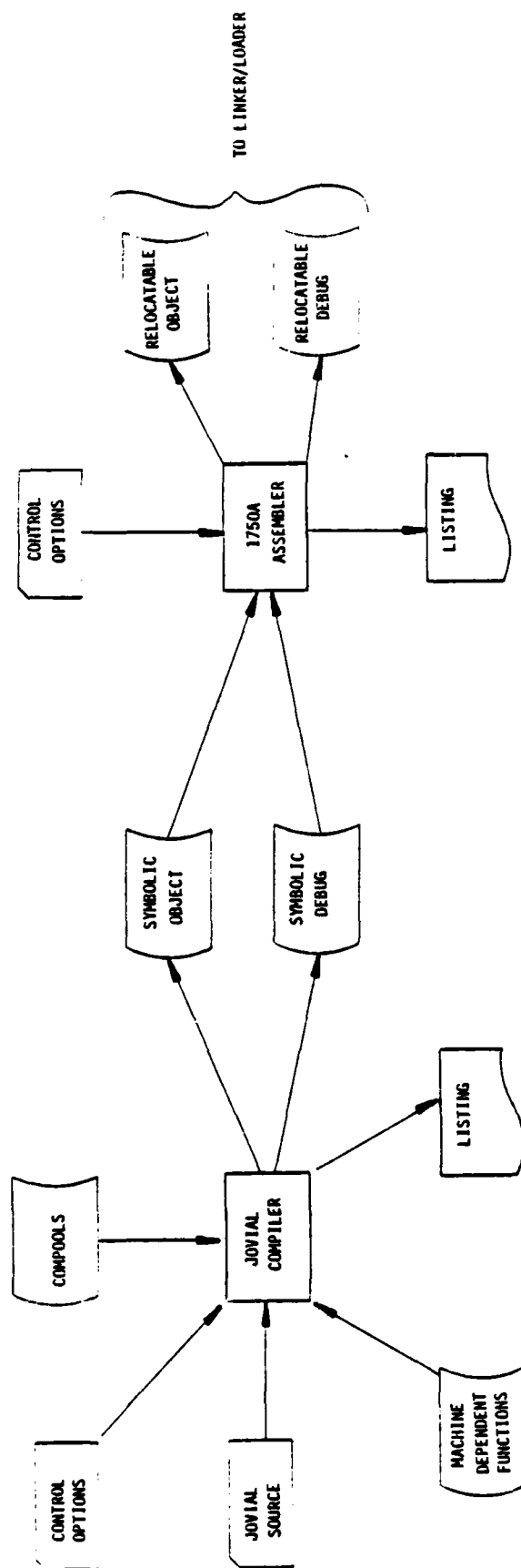
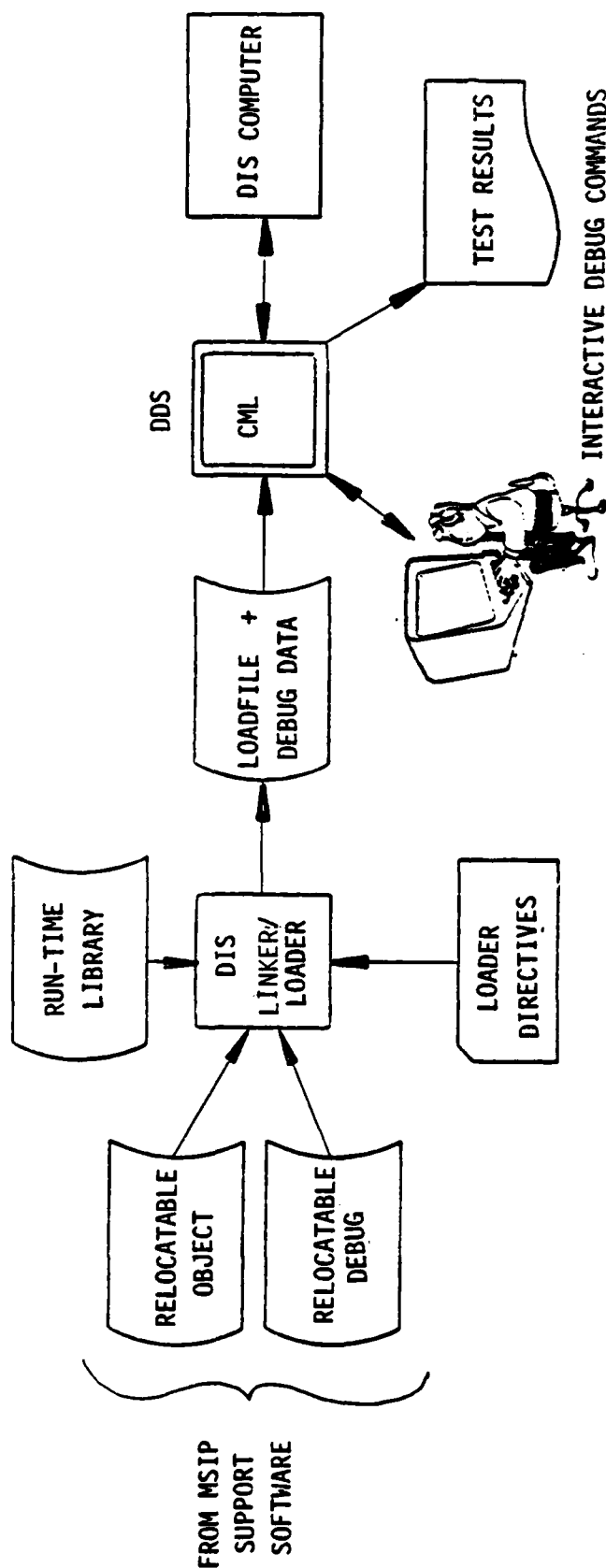


FIGURE 10

DIS LINKER/LOADER AND CONTROL-MONITOR-LOAD MODIFICATIONS INTERFACE MSIP SUPPORT SOFTWARE TO DDS SOFTWARE LOAD/DEBUG CAPABILITY



DIS LINKER/LOADER FUNCTIONS

- BUILDS LOADFILE FROM RELOCATABLE OBJECT PROGRAMS AND RUN-TIME LIBRARY
- CONFIGURES/INITIALIZES OPERATING SYSTEM
- COMPLETES DEBUG DATA FILE

DLL MODIFICATIONS REQUIRED

- NEW OBJECT AND DEBUG DATA FILE FORMAT
- APPROX 10-15% Δ, CAN BE HANDLED PRIMARILY BY BY PREPROCESSOR/REFORMATER
- DIFFERENT TARGET MEMORY PARTITIONING

DIS CONTROL-MONITOR-LOAD FUNCTIONS

- COMMUNICATES WITH DIS TARGET COMPUTER VIA DDS I/O INTERFACE
- CONTROLS BASIC COMPUTER FUNCTIONS
- PROVIDE INERACTIVE SYMBOLIC (JOVIAL LEVEL) DEBUG USING DEBUG DATA GENERATED BY COMPILER/ASSEMBLER/LINIGER

CML MODIFICATIONS REQUIRED

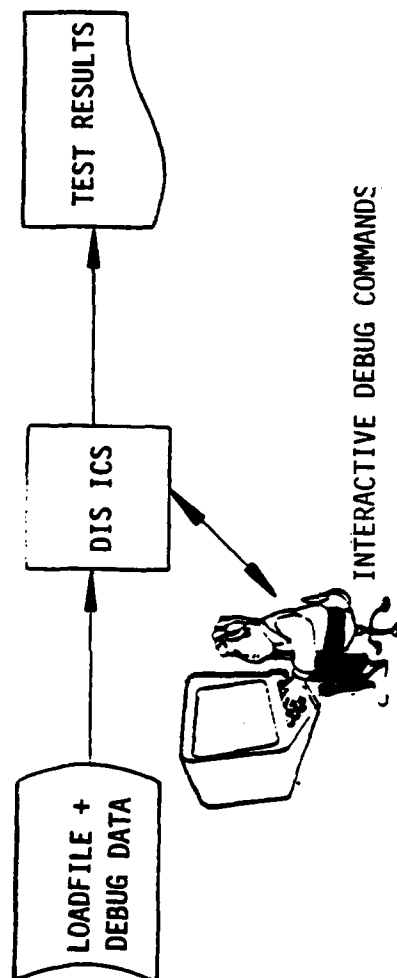
- NEW LOADFILE AND DEBUG DATA FILE FORMATS
- APPROX 5% Δ

FIGURE 11

INTERPRETIVE COMPUTER SIMULATOR MODIFICATIONS REFLECT DIS/Z8002 TO DIS/1750A HARDWARE CHANGES

- DIS ICS EXECUTES FLIGHT SOFTWARE BY SIMULATING ALL COMPUTER ELEMENTS

- CPU
- MEMORY
- INPUT/OUTPUT
- CLOCK
- INTERRUPTS



- DIS ICS PROVIDES HIGH VISIBILITY REQUIRED FOR UNIT/SUBPROGRAM CHECKOUT IN AN ENVIRONMENT THAT DECOUPLES EARLY HARDWARE AND SOFTWARE TESTING
- INTERACTIVE SYMBOLIC (JOVIAL LEVEL) DEBUG USING DEBUG DATA GENERATED BY COMPILER/ASSEMBLER/LINKER
- SOFTWARE TESTING IN THE DIS HARDWARE VIA THE DDS IS LIMITED BY THE INCREASED COMPUTER/BUS LOADING AND EXECUTION TIME AFFECTS
- SEVERAL APPROACHES TO DEVELOPING DIS/1750A SIMULATOR ARE BEING TRADED

FIGURE 12

WASP WEAPON SYSTEM/DIGITAL PROCESSING

Lt Gene J. Dylewski
Lt Don R. Huckle
Air Force Armament Division
Eglin AFB, Florida

ABSTRACT

This paper provides a summary of the Wasp weapon system and discusses the inherent digital processing electronics and requirements. It addresses the applicability of MIL-STD-1750A (Airborne Computer Instruction Set Architecture) and MIL-STD-1589B (Military Standard Jovial J73) to the Wasp system and invites replies from members of the microprocessor technology community as to alternative digital processing solutions.

WASP WEAPON SYSTEM/DIGITAL PROCESSING

The Wasp program goal is to provide the Air Force with an air-to-ground missile for use against massed armor in a Battlefield Interdiction role. The Wasp system is being designed to provide multiple armor kills from a single aircraft pass against any massed armor configuration. Missiles can be launched singularly or in selected salvos from high or low altitudes without exposing the aircraft to terminal defenses. Missiles are capable of being launched within the entire flight envelope of the current and advanced tactical fighter aircraft. Primary Air Force aircraft are shown in Figure 1 and NATO aircraft considered for Wasp employment are also listed.

The primary mode of operation for Wasp is the standoff blind launch mode where the aircraft does not visually acquire the targets. Illustrations of the blind launch timeline and the Wasp delivery are shown in Figures 2 and 3. As seen in Figure 2, the missile seeks its cruise altitude after launch and flies towards the target array. The millimeter wave (MMW) seeker and its associated signal processing provide each missile's digital processor with guidance and targeting information. The digital processor combines seeker data with flight sensor information to provide for autonomous lock-on-after-launch capability in

adverse weather and battlefield conditions. Flexibility of launch modes allow for single missile launches in a Point and Shoot mode for Close Air Support missions.

The Wasp program began the validation phase in Nov 1979 with the award of competitive 42-month contracts to Boeing Aerospace Co. and Hughes Aircraft Co. Main emphasis throughout the validation phase has been on the MMW seeker development and testing. After a thorough evaluation in Feb 1982, the Air Force System Program Office elected to continue the validation program with Hughes. The Hughes seeker has undergone static testing and is currently in captive flight testing. The validation phase will end with the launching of eight single free-flight missiles from an F-16 aircraft at Eglin AFB, beginning in Nov 1982. Full Scale Development (FSD) will begin immediately after completion of validation.

The Hughes missile configuration is shown in Figure 4. Each pod is envisioned to contain twelve missiles which are packaged tandem in six launch tubes. The pod is roughly the size of an F-16 370 gallon fuel tank. This All Up Round is a 2000 pound class store. Each missile weighs slightly over 100 pounds and is approximately five feet in length. The four wings and four control fins fold to allow high packaging density. The wings and fins open after missile launch.

Flexibility in deployment tactics will be required to provide the best possibility for success on every mission. Launch aircraft will carry two or four Wasp pods on each mission. For the massed armor scenario, the aircraft typically ingresses at a low altitude for maximum survivability, fires all its missiles, jettisons the empty pods, and egresses as the missiles fly on to their assigned targets. Jettisoning the empty pods rather than returning them for refurbishment is part of a "wooden round" maintenance approach being pursued for its low total system life cycle cost. The Wasp missile system must be available when needed and reliable when deployed. Additionally, the system must be economical to support and deploy. The wooden round concept requires no field checkout other than that associated with the Life Cycle Surveillance Test Program. Cost and supportability are major factors influencing the eventual Wasp configuration.

Wasp faces several technology challenges during development. Although the major emphasis is now on the MMW seeker, other technologies being demonstrated include folding wings/fins, high boost to sustain propulsion ratios, warhead penetration, and high speed digital processing. High speed launch imposes great demands on the missile during the early phase of the mission. Control surfaces must be quickly deployed and boost must be sufficient to clear the aircraft. Launch requirements for aircraft separation are contradictory to the requirement for a moderate cruise velocity during seeker search. The Hughes missile design employs separate boost and sustain motors to meet this requirement. The warhead under development is a hit-to-kill shaped charge. The conflicting requirements for small size and weight vs the requirements for armor penetration are driving the design.

Another important subsystem of the Wasp missile is the guidance electronics. Navigation performance must be matched with the requirements to build an affordable missile with acceptable performance. The relatively short flight of the missile allows some relaxation in the midcourse guidance requirements. Several other factors influence midcourse guidance requirements, among these are: seeker search area, missile attack pattern and angle, and target location errors. Hughes' design incorporates rate gyros and accelerometers to provide an attitude hold system. This approach matches the missile attack pattern, target allocation scheme, and seeker search areas. Analytical demonstrations of the required kills per pass have been accomplished for the expected nominal launch conditions as well as for realistic battlefield variations from nominal.

The key to success of the Wasp program is the ability of the MMW seeker/signal processor and associated electronics to detect, acquire and guide the missile to a valid target impact. A large part of this ability depends on state-of-the-art high speed microprocessor technology. The digital processing unit must coordinate large amounts of guidance and targeting information, perform data computations and manipulations, and control all of the functional tasks from pre-launch to impact. The electronics must perform with flawless accuracy and nearly 100% reliability to meet system requirements. More importantly, the Hughes' design places stringent requirements on the digital processing design in terms of processing speed, memory capacity, size, weight and power dissipation.

In Wasp as in most other modern tactical missiles, the central processing unit (CPU) or data processor is one of three major components of the digital processor. The other two components are the input/output (I/O) and signal processor (referred to as the video processor in Wasp). The CPU performs most of the logical and medium speed computations involved in missile guidance. In Wasp these data processing functions are:

- a. Missile Guidance
- b. Seeker Control
- c. Signal Processing
- d. System Control

All of these processes are characterized by data rates that are dependent upon missile time constraints.

Digital signal processing of high speed data (at sensor rates) is processed by the video processor. This is currently a block of high speed dedicated logic that performs functions such as digital filtering, Discrete Fourier Transformation calculations, detection, post detection integration, and range gate buffering. The I/O subsystem performs various conversions, timing and logic functions such as range gate and transmitter control, A/D and D/A conversion.

The Wasp missile processor operates in a multi-tasking mode taking data from a number of different sources and responding within restricted timing constraints. The Wasp processor must be capable of high speed iterated numerical operations on arrays of data and also be capable of rapid decision functions for mode control and detection algorithms. This type of application requires a number of architectural attributes:

a. HIGH SPEED. The Wasp processor requirements are in excess of 2.0 million instructions per second (MIPS). A slower processor could be used, but this would require the off loading of a number of software functions to the video processor and I/O hardware, and may exceed volume constraints.

b. VECTORED INTERRUPT CAPABILITY. The Wasp processor must quickly respond to many different asynchronous data sources. Task scheduling functions must be performed in hardware. The time required for saving and restoring system logic states must be minimal.

c. EFFECTIVE MEMORY ADDRESSING. Transfers of large arrays of information dictate a high speed addressing structure.

d. HIGH SPEED DECISION LOGIC. The large number of Wasp system flight modes and complex detection logic require a conditional branch mechanism which is the speed equivalent of a simple add instruction.

e. RAPID ARITHMETIC. High speed multiplication and addition are essential to the numerical tasks in missile guidance. Presently, floating point operations induce speed and hardware penalties. However, from a programming standpoint, floating point capability would be very useful.

Table 1. is a summary of the Wasp missile processor requirements:

Table 1. Missile Processor Physical Constraints

Size: 24 in² (PC Board Area)
Power Dissipation: 10 Watts
Temperature Range: -55°C to 125°C Ambient
Qualification Levels: Mil-Spec-883 Level B.
Throughput: 2 MIPS (Fixed Point, "Missile Mix"
Instruction Mix)

Existing software support tools are essential for software development. Missile applications require programming and debugging aids that are used in other computer applications with minor additions. The Wasp processor must be supported by a symbolic assembler, a linking loader, hardware diagnostic programs

and an emulation program hosted on peripheral computers. The emulator must keep track of system timing in a very exact manner and be capable of simultaneous access by a number of programmers. Additionally, the Wasp processor development system requires a monitoring system for the missile processor. This system must be capable of multiple breakpoints, bus monitoring and software trace capability. The monitoring equipment must not interfere with program execution speed.

The throughput figure identified in Table 1. needs further elaboration. The speed of the processor is dependent upon the frequency and the types of instructions to be executed. In the case of Wasp, the Radar Signal Processing software presents the greatest amount of the time loading. Design repartitioning to offload a number of software functions to the video processor and I/O hardware would reduce the timeloading but would result in a major redesign effort. This may or may not be possible. Table 2 presents the Wasp instruction mix/throughput for the critical path software - Radar Signal Processing:

Table 2. Wasp Instruction Mix/Throughput
Critical Path: Radar Signal Processing

Static Program Memory: 5000 Instructions

Sample Period: 4.8 msec

% Time Loading/Execution Time: 46%/2.2 msec

Mix:

Load and Store	40%
Add and Subtract	10%
Test and Branch	15%
Logical Compare	15%
Shift (3 Bit Average)	5%
Multiply	2%
Divide	1%
Double Precision	1%
Miscellaneous (i.e., Jumps, Go Subs)	11%

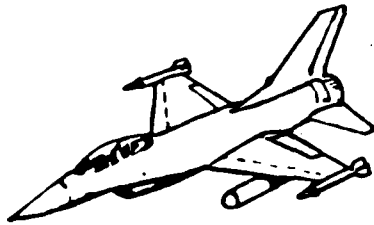
The Hughes' validation missile processor design incorporates their company developed General Missile Processor (GMP). The GMP is a high speed programmable minicomputer composed of Schottky TTL hybrids. It was first designed in 1974, specifically for missile signal processing and guidance functions. The GMP has been adapted and used on several other missile programs, among them are: the Phoenix missile, the Advanced Medium Range Air-to-Air Missile (AMRAAM), and Tank-Breaker. The GMP has been modified into the Wasp Digital Processor (WDP) processing 2.1 MIPS of the missile/guidance instruction mix. It is this single characteristic which sets the WDP apart from other available commercial signal processors or federated microprocessor schemes. In its hybrid form, the WDP dissipates 101 watts of power. It has been form factored for use in the free-flight missiles. Further reduction in size is necessary for the Wasp FSD configuration in order to meet size, weight and power constraints.

Air Force Systems Command requirements for Embedded Computer Resources (ECR) specify that all ECR shall be developed in accordance with MIL-STD-1750A (Airborne Computer Instruct Set Architecture) and MIL-STD-1589B (Military Standard Jovial J73). In addition to the above standards, the Wasp FSD Statement of Work will require that:

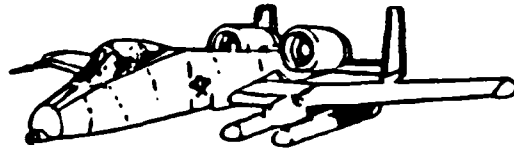
- a. Embedded computer hardware shall include a minimum of 30 percent spare capacity in terms of memory and throughput.
- b. Preassembled, off-the-shelf microcomputers or minicomputers shall be used unless it can be shown that no such technology exists which satisfy appropriate form-fit and function requirements.
- c. Support tools shall include an efficient compiler for an AFSC approved high order language.
- d. The computer, its memories and peripherals, shall be modular in design and shall be designed for FSD ease of maintenance.
- e. In no case will the development of an assembler, compiler, or operating system be undertaken.

Ideally, the Wasp design must utilize a processor that meets all of the previous standards and requirements. To our knowledge, no commercial microprocessor scheme and support software/peripheral equipment exists that meets all of these requirements. It is the intent of this paper to invite replies from members of the computer technology community as to the present and future availability of microprocessor technology which will meet Wasp unique digital processing requirements and military standards.

Wasp poses several technological challenges. It is the first tactical missile weapon to use a millimeter wave sensor. The warhead, constrained by the physical missile size, must be capable of lethal penetration of the latest threat. Controlling the missile are sophisticated signal processing electronics which autonomously detect, acquire, track and guide the missile to a direct hit. Inherent in Wasp is the need for state-of-the-art digital processing technology which meets system requirements. Any questions concerning this paper should be addressed to Headquarters Armament Division, Deputy for Antiarmor Munitions, Eglin AFB, FL, Attention Lt Gene Dylewski, (904) 882-5868, AV 872-5868.

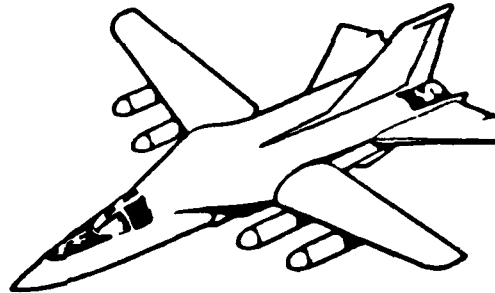


F-16
2 PODS - 24 WASPS



A-10
4 PODS - 48 WASPS

MIRAGE V
ALPHA JET
JAGUAR
TORNADO
HARRIER



F-111
4 PODS - 48 WASPS

FIGURE 1. WASP LAUNCH AIRCRAFT

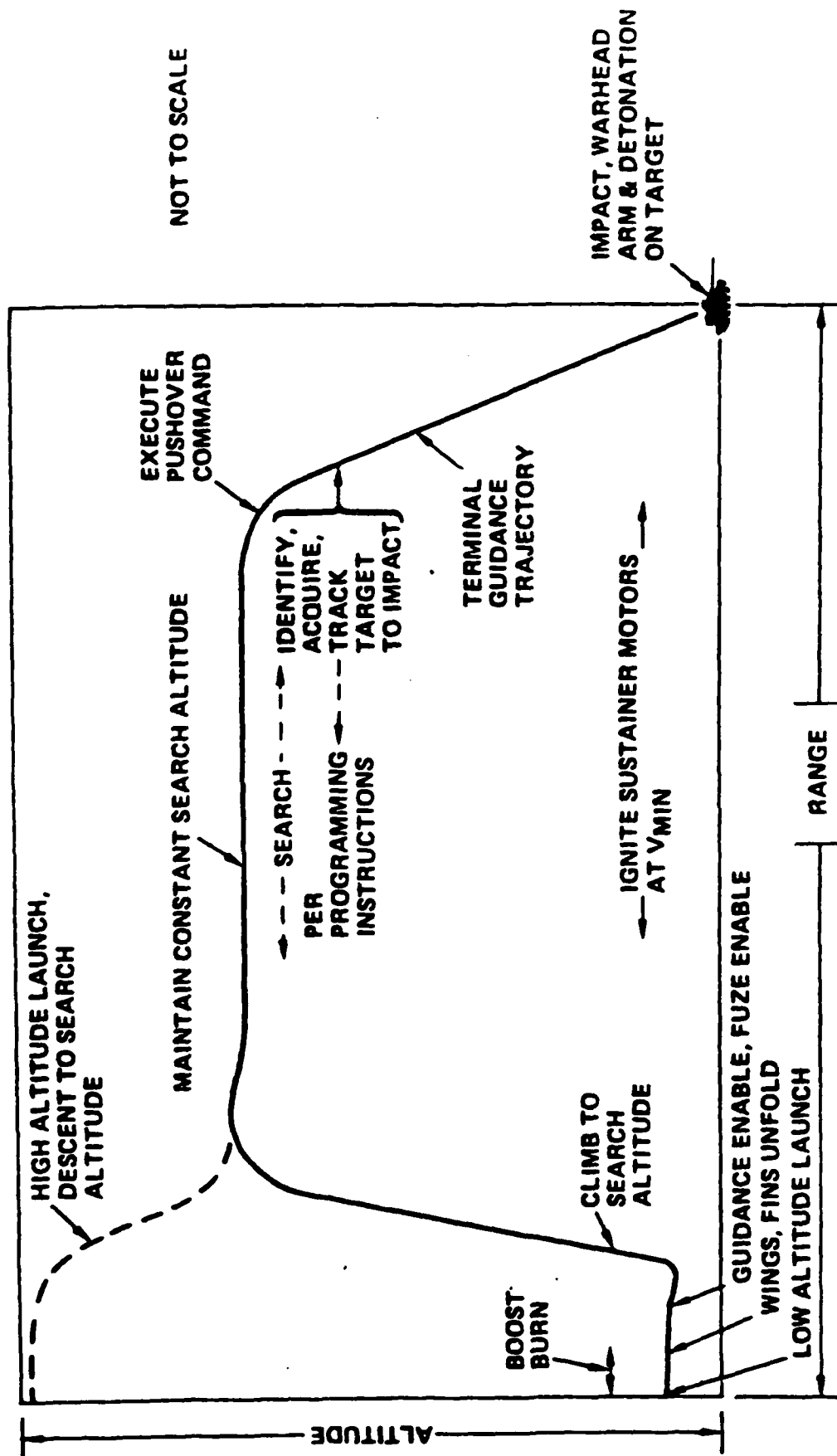


FIGURE 2. WASP BASIC MODE TIMELINE

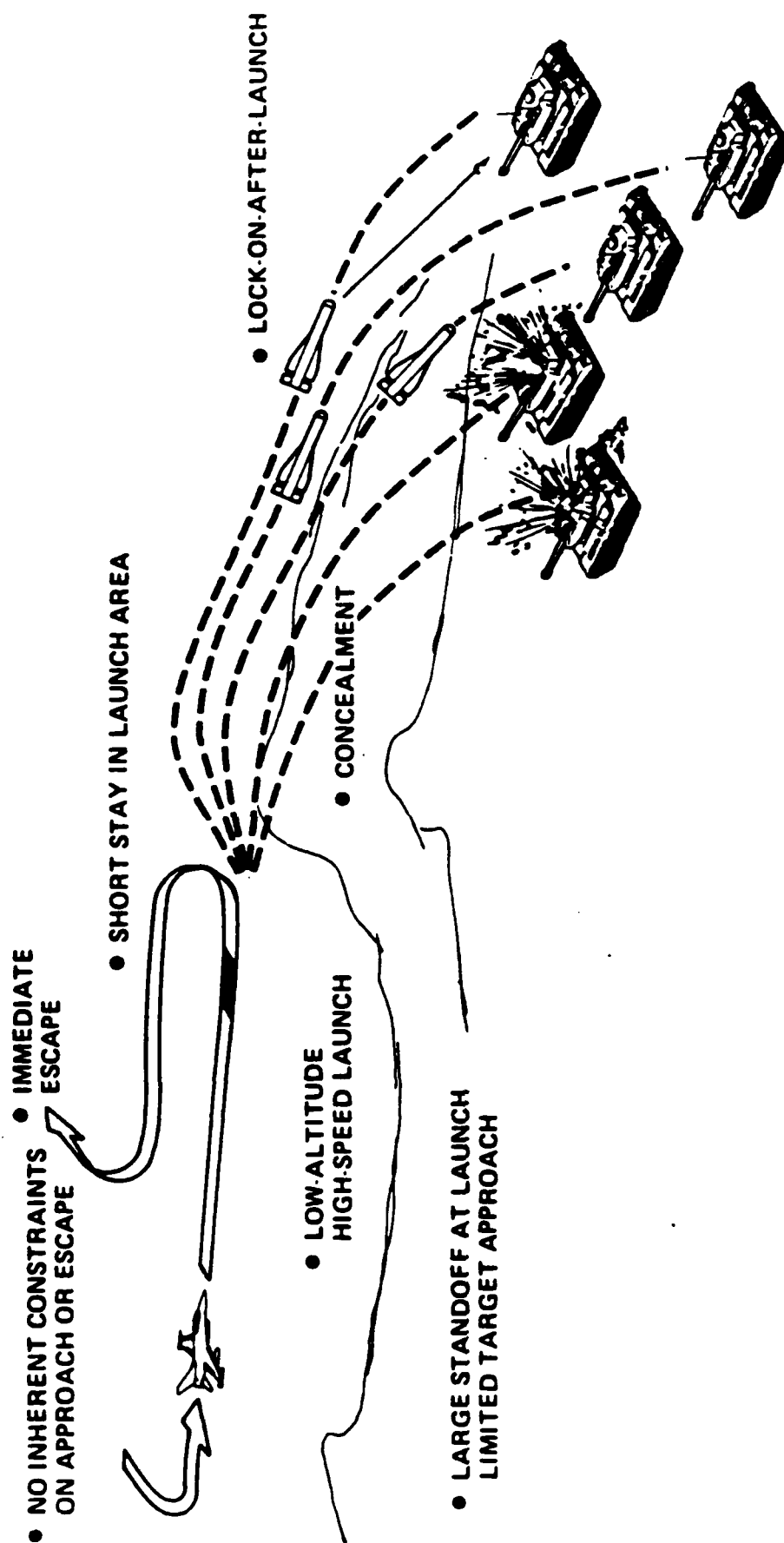


FIGURE 3. WASP BASIC MODE SCENARIO

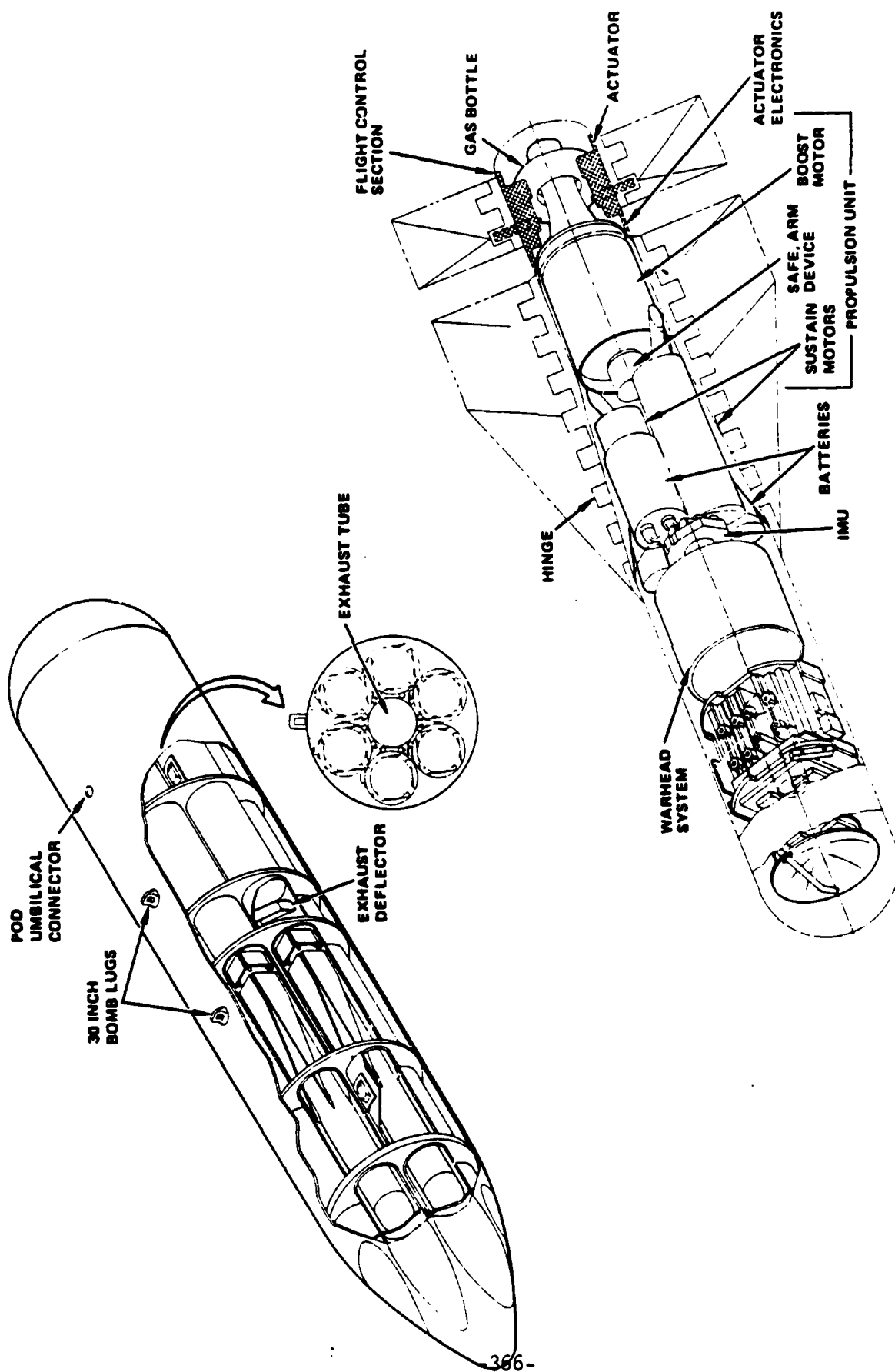


FIGURE 4. HUGHES WASP MISSILE/POD CONFIGURATION

WASP WEAPON SYSTEM/DIGITAL PROCESSING

LT GENE J. DYLEWSKI

AIR FORCE ARMAMENT DIVISION

EGLIN AFB, FLORIDA

PURPOSE

- DISCUSS ISSUES RELATED TO AFSC STANDARDS/REQUIREMENTS
AND THEIR EFFECT ON THE DEVELOPMENT OF A SOPHISTICATED
WEAPON SYSTEM

- EXAMPLE: WASP WEAPON SYSTEM

WASP WEAPON SYSTEM/DIGITAL PROCESSING

- SYSTEM OVERVIEW
- TECHNOLOGICAL CHALLENGES
- DIGITAL PROCESSING
- SUMMARY

PROGRAM OVERVIEW

- AIR TO GROUND CONVENTIONAL MUNITION
- TARGET - SECOND ECHELON ARMOR
- REQUIREMENTS
 - MULTIPLE KILLS/PASS
 - ALL WEATHER
 - STANDOFF
 - OFF BORESIGHT
 - LOCK ON AFTER LAUNCH
 - BLIND LAUNCH

PROGRAM OVERVIEW (CONTD)

- PROGRAM STATUS:
 - CURRENTLY IN VALIDATION PHASE WITH HUGHES
 - FULL SCALE DEVELOPMENT TO START IN OCT 83
- FLEXIBLE DEPLOYMENT TACTICS
- CURRENT DESIGN
- COST, RELIABILITY AND SUPPORTABILITY
 - "WOODEN ROUND" MAINTENANCE

TECHNOLOGICAL CHALLENGES

- MILLIMETER WAVE SEEKER
- ROCKET MOTORS
- FOLDING WINGS AND FINS
- WARHEAD
- HIGH SPEED DIGITAL PROCESSING

WASP DIGITAL PROCESSING

- KEY TO WASP SUCCESS IS AUTONOMOUS CAPABILITY
 - MISSILE GUIDANCE
 - SEEKER CONTROL
 - SIGNAL PROCESSING
 - SYSTEM CONTROL
- HIGH PROCESSING SPEED DICTATES STATE-OF-THE-ART MICROPROCESSOR TECHNOLOGY
 - EFFICIENT INTERRUPT SCHEME
 - HIGH SPEED MEMORY ADDRESSING
 - RAPID ARITHMETIC OPERATIONS
 - HIGH SPEED DECISION LOGIC

PROCESSOR REQUIREMENTS

- PHYSICAL
 - SIZE - < 24 IN² (PC BOARD AREA)
 - POWER DISSIPATION - < 10 WATTS
 - MIL-STD-883B LEVEL B QUALIFICATION
 - THROUGHPUT - > 2 MIPS
- NAVIGATION ACCURACY DICTATES 16 BIT MACHINE
- SUPPORT TOOLS
- AFFORDABILITY

CRITICAL PROCESSING PATH

- RADAR SIGNAL PROCESSING
- PROGRAM SIZE - 5000 WORDS
- EXECUTION TIME 2.2 MSEC
- INSTRUCTION MIX
 - MEMORY TRANSFER 40%
 - ARITHMETIC 10%
 - LOGICAL 15%
 - TEST AND BRANCH 15%
 - MULTIPLY AND DIVIDE 4%
 - MISCELLANEOUS 16%

COMPUTER RESOURCE REQUIREMENTS

- STANDARDS:
 - MIL-STD-1750A
 - MIL-STD-1589B
- FULL SCALE DEVELOPMENT STATEMENT OF WORK
 - 30% SPARE MEMORY AND THROUGHPUT
 - OFF THE SHELF HARDWARE
 - EFFICIENT COMPILER
 - MODULAR DESIGN
 - NO COMPILER DEVELOPMENT

SUMMARY

- TECHNOLOGICAL CHALLENGES
- DIGITAL PROCESSING
 - AVAILABLE HARDWARE
 - DESIGN CONSTRAINTS
 - STANDARDS
- AFFORDABILITY

**DELCO ELECTRONICS MIL-STD-1750A ARCHITECTURE
COMPUTER FAMILY**

S82-66
5 May 1982

Presented by

CLIVE D. LEEDHAM

**DELCO ELECTRONICS DIVISION
GENERAL MOTORS CORPORATION
Santa Barbara Operations
6767 Hollister Avenue
Goleta, California 93117**

ABSTRACT

Delco Electronics early acceptance of, and participation in, Air Force activities with standardized Military computers is reviewed. The early acceptance and in-house development of MIL-STD-1750A architecture computers enabled Delco Electronics to capture Air Force programs requiring this new standard. These early programs are detailed, and Delco Electronics plans for the evolution of computers operating to this architecture standard are discussed.

INTRODUCTION

The move toward computer-oriented standards began at Wright-Patterson Air Force Base with the MIL-STD-1553 Multiplex Data Bus Conference in 1976. Milestones that followed included the introduction of MIL-STD-1750 itself, the creation of the User's Group in August 1979, and the establishment of the Embedded Computer Standardization Program Office.

Although Delco Electronics monitored these activities, new standards (1553, 1750, etc) were initially met with resistance from the Delco Electronics working groups, no doubt because of the not-invented-here attitude that is so often

encountered. The standards idea generated interest among key people, however, so that we were able to muster sufficient support within Delco Electronics to begin studies of both the hardware and software implications. These studies eventually led to the conclusion that hardware incorporating this instruction set architecture (ISA) could be built, and, perhaps more importantly, this ISA would solve avionics problems as we knew them. Finally, it appeared that standardization at the ISA level would allow introduction of new devices and, hence, new computers as time progressed. Furthermore, it appeared that this approach would not inhibit future computer development plans. Nothing we have experienced so far suggests that our analysis was wrong.

CURRENT MIL-STD-1750A PROGRAMS

Delco Electronics competed for and secured the first two Air Force programs requiring computers executing the MIL-STD-1750A instruction set architecture. About 18 months ago, the computers for the Low Altitude Navigation Targeting Infrared Night (LANTIRN) program was competed, and Delco Electronics was selected by Martin Marietta, Orlando Division (Contract 745528), to supply its M372

MIL-STD-1750A computer with solid state memory. Production options for 300 shipsets (600 computers) have been proposed. Some 5 months later, a decision was made to introduce MIL-STD-1750A into the F-16 aircraft as part of the General Dynamics Multi-national Staged Improvement Program (MSIP). Again, the requirement was competed, and Delco Electronics was selected. Negotiations are now complete (General Dynamics Purchase Order 149) for production options of up to 1,800 shipsets (1,800 computers) with core memory. The principal characteristics of these two programs and their computer configurations are shown in Figure 1.

DELCO'S M372 MIL-STD-1750A COMPUTER

Architecturally Enhanced CPU

The M372 computer utilizes a compact, high performance processor mechanized with advanced LSI circuits to achieve an optimal balance of performance, power, and size. The organization is implemented in a highly efficient, pipelined structure that allows each instruction to be executed in a minimum of time.

The CPU is implemented using the 2900 family of commercially available LSI devices and is optimized to efficiently execute the MIL-STD-1750A instruction set. To achieve the highest possible throughput consistent with cost and power, the design is based on the use of:

- 24-bit arithmetic unit and multiplier
- Instruction lookahead buffer to reduce instruction/operand memory contention
- Concurrent memory and I/O data flow.

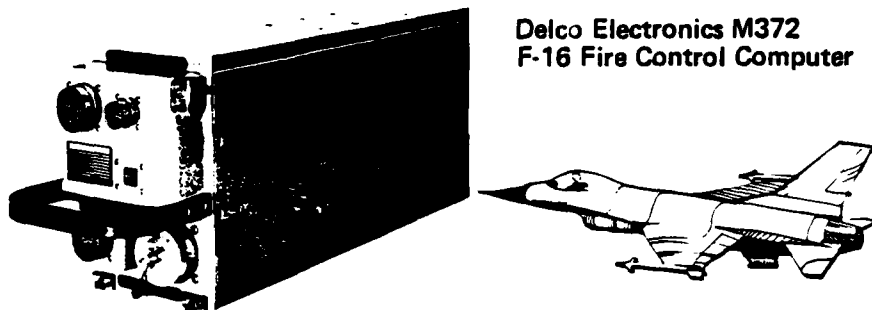
High throughput is achieved primarily through architectural sophistication.

However, some increase in speed comes from the use of high speed, bipolar, bit-slice LSI devices and Schottky transistor-transistor logic. The CPU uses pipelining to fetch an instruction while executing the previous one; buffering is provided for up to sixteen 16-bit instruction words. The effect of instruction pipelining is a submicrosecond execution rate for fixed-point, short instructions. The microcode cycle (clock time to execute logic control for one micro-store word) is 187 nanoseconds. The processor optimizes internal bus traffic by:

- Providing separate address and data memory interface (MI) buses to provide the fastest memory access and instruction and data fetches.
- Providing a dedicated programmed I/O (PIO) bus, which minimizes loading on the critical memory interface bus and permits separation of programmed I/O activity from the memory interface, thereby preventing asynchronous input/output from degrading instruction throughput.

For configurations employing a core memory, the use of memory interleaving minimizes the transport time required to fetch data from the 16-bit memory subsystem. Multiplexing memory modules on the same basic memory bus achieves economies in the electronics and increases the utility of the bus system. The basic bus organization allows the use of single- or dual-interleaved memories as well as a variety of semiconductor memory types.

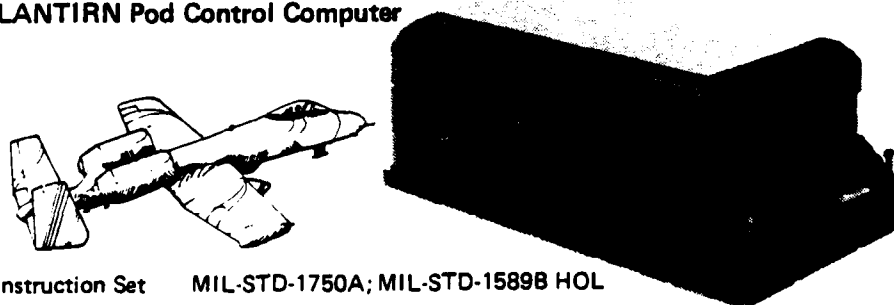
The CPU is microprogrammed, with approximately 25 percent spare control store for future growth. The control store is organized to perform both instruction mapping and sequencing and control. Instruction mapping translates instruction code



**Delco Electronics M372
F-16 Fire Control Computer**

Instruction Set	MIL-STD-1750A; MIL-STD-1589B HOL
Size	1/2-ATR by 18.6 inches
Weight	31.3 pounds
Power	280 watts (3-phase, 400-hertz, 115-volt source)
Throughput	587 KOPS (DAIS instruction mix)
Memory	Two 32k core modules; 750ns cycle time; 16 bits plus parity
Input/Output	Discretes (26 input; 8 output) Analog (6 dc input; 2 synchro output) MIL-STD-1553B data bus (two channel dual-redundant, multiprotocol) DMA (greater than 1 million words per second)
Reliability	2,403 hours mean time between failures

**Delco Electronics M372
LANTIRN Pod Control Computer**



Instruction Set	MIL-STD-1750A; MIL-STD-1589B HOL
Size	5.0 x 7.1 x 13.1 inches
Weight	17.9 pounds
Power	271 watts
Throughput	666 KOPS (DAIS instruction mix)
Memory	Nonvolatile program (65,536 words) Volatile RAM (65,536 words) Power-transient protected RAM (1,024 words) Electrically alterable ROM (1,024 words) Word size 16 bits plus parity
Input/Output	Discretes (25 input; 32 output) Parallel bus (DMA: 880k words/s; Program I/O: 200k words/s) MIL-STD-1553B data bus (remote terminal)
Reliability	2,400 hours (projected) mean time between failures

Figure 1. MIL-STD-1750A Program Characteristics

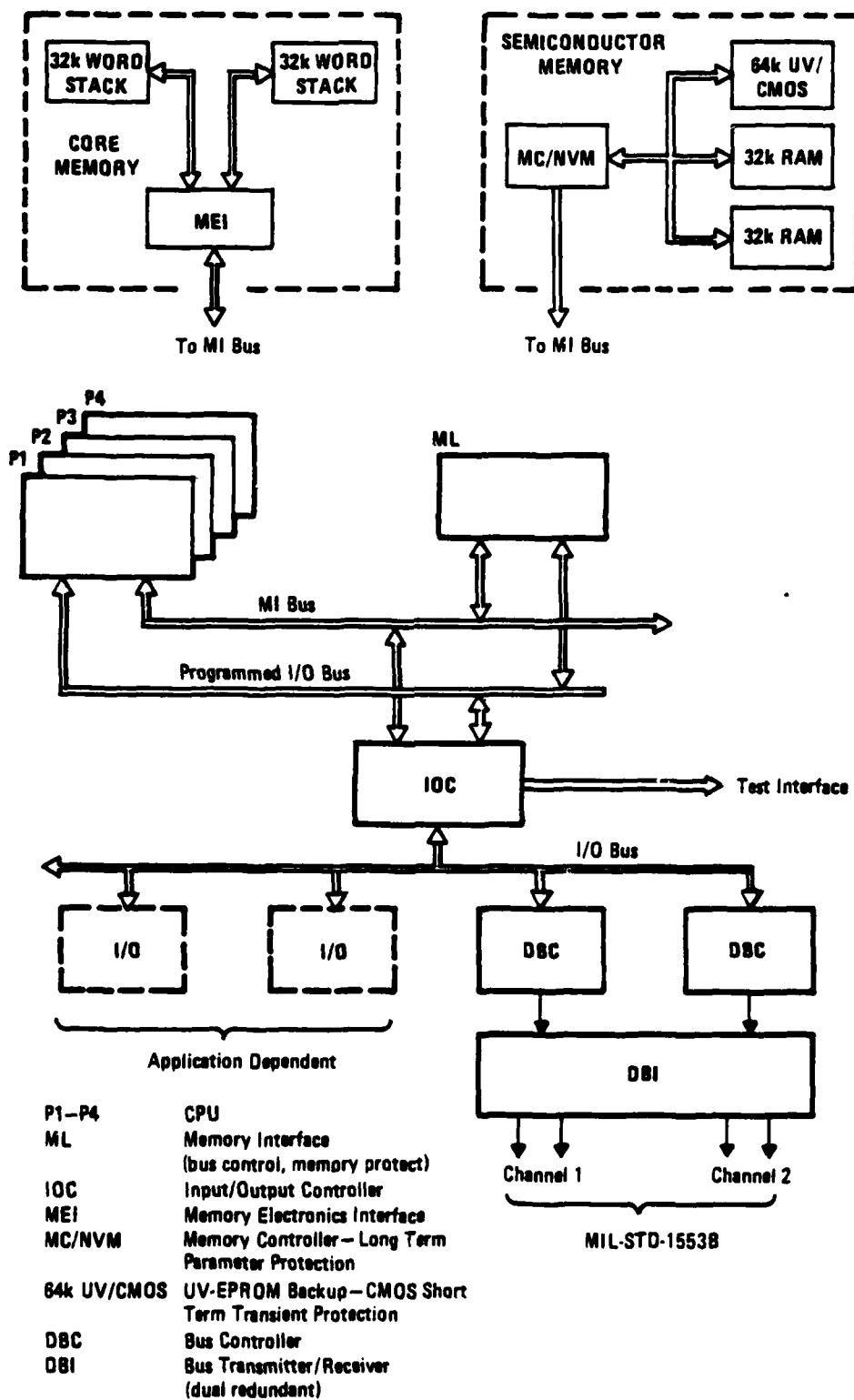


Figure 2. M372 Computer Partitioning

into microprogram starting addresses. All 16 bits of an instruction, or multiple words of multiword instruction, can be mapped into microprogram addresses. The sequencing and control section provides the address linkage between microinstructions, and it controls the CPU processing elements.

Memory System

Two types of memory systems are currently in use on Delco's M372 computer family. The F-16 Enhanced Fire Control Computer uses a core memory, while the LANTIRN Pod Control Computer is a combination of several semiconductor technologies. Either option is compatible with the CPU organization previously described.

Semiconductor Memory. The semiconductor memory is implemented with devices of several technologies, provides 64k words of storage, and features nonvolatility and reprogrammability, coupled with fast system level access/cycle times for high memory bandwidths. This system incorporates 64k words of high speed NMOS RAM, backed up by 64k words of UV EPROM, 2k words of EEPROM, and 1k words of CMOS RAM (See Figure 3.)

In operation, the memory subsystem user's interface with the NMOS RAM allows for fast access (375 nanosecond system level access times) to instruction and operand data. The operand write data address to the EEPROM-backed-up section of memory is written into the NMOS RAM and is transferred to the EEPROM. Write operations to the CMOS RAM follow a similar path. During power turnon, the contents of the UV EPROM, EEPROM, and CMOS RAM backup memories are loaded into the NMOS RAM.

Write operations into NMOS RAM, EEPROM, or CMOS RAM exhibit cycle times of 375 nanoseconds. Writing into the CMOS RAM utilizes a pipelined address/data transfer that

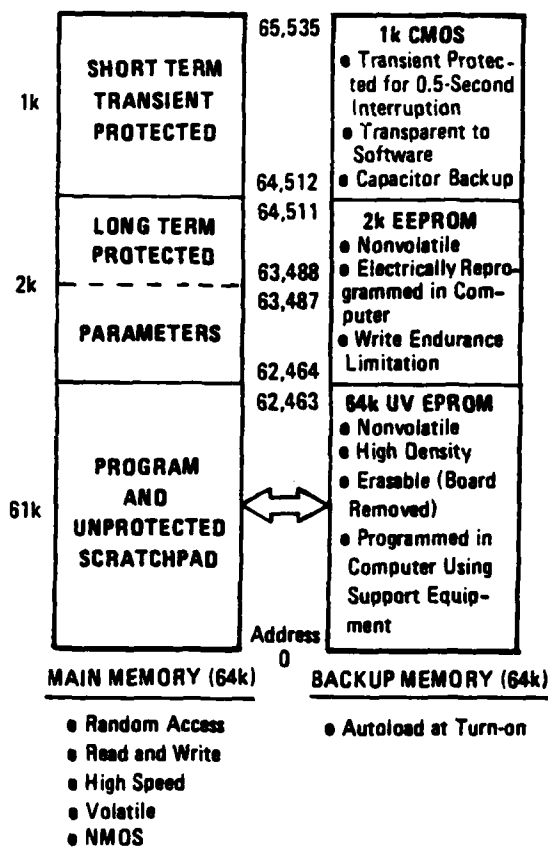


Figure 3. Semiconductor Memory Map

allows maintenance of consecutive write operations at the 375-nanosecond rate. The EEPROM write operations require 20 milliseconds per write, which limits the rate of consecutive write operations to this area of memory. A discrete input bit is made available to the software, indicating when the EEPROM has completed its write cycle. By proper utilization of this interlock feature, the available memory bandwidth of 2.667 million 16-bit words per second is maintained.

Any area within memory address space can be utilized for variable storage, but only those sections backed up by CMOS RAM or EEPROM are power-interruption protected or nonvolatile-memory protected. The size of the EEPROM-backed-up address space can be varied to suit the system

requirements from 0 to 256, to 512, to 1,024, or to 2,048 words. This is accomplished by programming the memory module connector whereby a section that is deselected from EEPROM backup becomes backed up by UV EPROM. The CMOS RAM protected area can also be deselected by connector programming to allow this section of memory to be backed up to the UV EPROM.

Programming of the UV EPROM is accomplished by means of a computer support equipment interface with the memory subsystem installed in the computer enclosure. The NMOS RAM is loadable from the support equipment independently of the UV EPROM, providing an easily alterable memory for software development activities.

Core Memory. The basic core memory system is 64k words by 18 bits, including parity and a storage-protection bit. The memory system consists of two 32k-word Memory Magnetics Assemblies (MMA), with a Memory Electronics Interface (MEI) circuit card assembly to facilitate memory interleaving. The interleaved memory bus architecture maximizes memory subsystem throughput while minimizing components and power consumption. Features of this architecture are:

- Optimized cost/performance ratio
- Advantageous use of synchronisms between memory bus subscribers while operating asynchronously
- No boundary restrictions on operand or instruction storage
- Worst case access time of 1.5 microseconds with maximum contention
- Memory bus system access time of 562 nanoseconds for single words and 937 nanoseconds for word pairs
- Read, write, and read-modify-write memory modes.

CORE MEMORY MODULE CHARACTERISTICS

<u>ITEM</u>	<u>DESCRIPTION</u>
ORGANIZATION	2-1/2D, coincident current, wide temperature range, low drive current, 13-mil OD cores
WORD SIZE	18 bits
CAPACITY	32,768 words per module, expandable to 131,072 words
CYCLE TIME	0.75 microsecond
ACCESS TIME	0.375 microsecond read; 0.937 microsecond write
POWER	MEI = 14.7 watts; 32k MMA = 32.5 watts
SIZE	4.3 x 6.4 x 3.0 inches (MEI plus MMA)
WEIGHT	3.5 pounds (MEI plus MMA)

The memory system is supplemented with a memory logic (ML) circuit card assembly, which provides basic internal processor bus control. Each core memory module exhibits access times of 375 nanoseconds and cycle times of 750 nanoseconds. By combining these into an interleaved memory system, an effective cycle time of 375 nanoseconds is achieved. The ML circuit also contains the memory write protect logic.

Word-by-word memory write protection is implemented by employing the optional 18th memory bit as a write-protect bit and using this bit for control in conjunction with the read-modify-write feature of the memory. This technique results in a one-clock-extended memory write cycle when compared to block protection techniques. Attempts to write into protected memory result in setting the appropriate fault register bits.

Direct memory access (DMA) users are buffered from the memory interface bus through the Input/Output Controller (IOC). Consequently, memory bus loading is minimized, and the processor and memory subsystem can run at the fastest possible speed. Incorporation of the IOC also allows DMA users to enjoy a shared data and address I/O bus, merged from the memory address and data buses, and provides simpler control by the IOC. As a result, the computer memory interface and I/O bus system provide a carefully chosen and optimized blend of performance and efficiency.

Bubble Memory. Delco Electronics has recently completed the design of a bubble memory system. This type of memory system has been selected by the McDonnell Douglas Aircraft Company for use in Delco's Flight Management Systems being supplied for versions of the DC-10 and DC-9 aircraft. System testing of bubble memory systems for the preproduction computers has recently been completed. These tests included operation over the temperature range of -40 to +85°C. This memory type is also available for M372 computers.

Input/Output Subsystem

A versatile and efficient input/output subsystem is a core element of the computer. An Input/Output Controller (IOC) services the interrupt processor, the various digital input/output channels and discretes, the multiplex buses, and the external support equipment port. The IOC interfaces both the memory interface (MI) and the programmed I/O (PIO) buses with the I/O bus. The Input/Output Controller supports direct memory access (DMA), programmed I/O, interrupt I/O, and the instruction insert operations. The I/O bus consists of a parallel, bidirectional, shared data and address bus with control signals dedicated to each of the

above operations. General purpose functions provided by the I/O system include the following:

- Sixteen levels of vectored, prioritized interrupts
- Two programmable 16-bit interval timers
- Time-out or watchdog timer
- Decode of I/O device addresses and function codes (microprogram outputs from CPU)
- Interface with signal conditioner modules.

MIL-STD-1553B Data Bus. Delco's new multi-bus system allows two or more MIL-STD-1553B multiplex bus channels to be used in an avionics suite with minimum hardware penalties. Bus systems can be dedicated to functions such as display or flight control, with a passthrough feature provided to transfer data from one bus system to another. The bus control system allows the computer to simultaneously operate as a bus controller or remote terminal on single or multiple independent multiplex data bus systems. Each bus channel consists of a dual redundant serial data interface capable of communicating concurrently with subsystems operating under MIL-STD-1553, 1553A, and/or 1553B protocols.

Two dual-redundant channels are presently provided on three 1/2-ATR circuit card assemblies, a single Data Bus Controller (DBC) for each channel and a shared transmitter/receiver module, referred to as the Data Bus Interface (DBI). The controllers interface via the I/O bus to the Input/Output Controller (IOC), which contains circuitry necessary to interface the DBC channels to the programmed I/O and DMA functions of the processor.

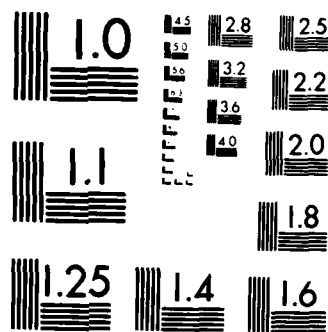
AD-A150 583	PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16 MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 06 MAY 82	5/6
UNCLASSIFIED	ASD-TR-82-5011-VOL-1	F/G 9/2 NL

PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16
MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV
WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 06 MAY 82
ASD-TR-82-5011-VOL-1 F/G 9/2

5/6

UNCLASSIFIED

NL



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

The DBC controls the operation of the multiplex bus, each channel operating independently so that the multiplex bus interface operates simultaneously, requiring little intervention from the CPU. The DBC receives control commands and address information from the CPU through the programmed I/O via the IOC, which initiates the execution of a microprogrammed algorithm to operate in a given mode by sequentially accessing commands and/or data from the memory via DMA.

Data flow between the DBC and IOC is word by word, controlled by "hand-shaking" signals provided by the DBC for multiplex bus transactions (DMA instructions and data) and by the processor for DBC control and status (programmed I/O). The DBC tests all multiplex bus transactions for invalid sync, invalid biphasic Manchester code, incorrect bit count (total transmission time), and parity error and indicates errors to the IOC.

Operator or software activation of the MIL-STD-1553 system has been minimized to assure proper sequencing without significant programmer involvement.

Additional I/O Functions. Depending on the application, various additional I/O functions are provided. In the F-16 Fire Control Computer application, two additional circuit card assemblies are provided, a Digital I/O (DIO) and an A/D Converter (ADC). The Digital I/O includes 6 differential input, 8 differential output, and 12 high level input discretes along with the real time clock and COP circuitry (real time program loop error detection). The 12-bit A/D Converter accepts six dc analog inputs and two three-wire synchro inputs.

The LANTIRN Pod Control Computer has a Discrete/Interrupt Timer (DIT) circuit card assembly that includes 16 input and 16 output external differ-

ential discretes, 9 input and 16 output internal discretes, 6 external interrupts, programmable timers per MIL-STD-1750A, and COP circuitry.

Power Supply

The F-16 Fire Control Computer power converter is composed of an ac/dc converter and a dc/dc converter. This mechanization combines the reliability of a low risk, well proven design with high efficiency, low cost, and minimum size. The converter employs 400-hertz, three-phase, 115-volt rms primary power and provides all necessary system operating voltages. The converter will shut down and not be damaged if one or more phases of the source voltage is missing or degraded. The converter also provides moding for turn-on and turn-off sequencing. The power converter exhibits the following features:

- Operates from a 400-hertz, three-phase, 115-volt rms source and meets the requirements of MIL-STD-704 and MIL-STD-461, including the emergency condition requirements of MIL-STD-704
- Turn-on soft-start prevents input surge currents
- Programmed turn-off and restart for power interruptions
- Self-contained overvoltage protection
- Input protection for high voltage transients
- Internal high energy transient generation reduced by controlled turn-on and virtual elimination of simultaneous conduction of power transistor pairs
- Maximized reliability from use of switching transistors with high reverse energy rating and liberal power derating of semiconductors.

A memory protect moding section provides the delays and timing necessary to assure the orderly turn-on and turn-off of memory. It allows data to be maintained in core through a power interruption. This section includes a level detector, memory turn-on delay, input/output isolator, and the memory protect logic. The input level detector also provides for dc/dc converter turn-on and shutdown at a predetermined level of the input to aid in the moding function.

Output voltage regulation is ± 5 percent (except for the 10-volt core memory voltage, which is ± 1 percent). A regulation of ± 2 percent is achieved considering line variations and temperature of operation. When considering load variations, ± 5 percent regulation is standard on all Delco Electronics computer power supplies.

The LANTIRN Pod Control Computer receives its voltages from an external source.

Mechanical Design

The MIL-STD-1750A F-16 Fire Control Computer is packaged in a 1/2-ATR enclosure. Its fully modularized design and construction is similar to that of Delco's M362 series computers (including the original F-16 Fire Control Computer, Inertial Upper Stage Guidance Computer, and the Airborne Instrumentation Labs computers) and the highly reliable Magic III computers of which more than 7,000 have been produced.

In the F-16 configuration, the enclosure design employs a bolted and bonded assembly technique first used by Delco on the Apollo program to form a tight, light weight, low cost, rigid assembly. The top and bottom of the enclosure contain heat exchangers to which the computer circuit card assemblies are thermally coupled.

A top cover allows access to the circuit card assemblies for maintenance. The unit contains plug-in circuit card assemblies plus a connectorized power supply. Delco-designed circuit card guides (heat transfer clips) are used for mounting circuit cards in the enclosure. Flatpack devices provide high density functional packaging. Each circuit card contains a center core for conducting the heat from circuit devices to the card edges. Strain relief is inherent in the flatpack mounting scheme. The 1/2-ATR card size is optimum for the thermal and vibration environments encountered.

The circuit card assemblies are plugged into a multilayer interconnect board using metal-shell, two-piece, pin/socket, twist-pin, gold-plated connectors. The packaging is simple, reliable, and eliminates adverse heat density conditions. The power conditioner structure is fabricated from plate stock and is screwed together with a sheet metal cover, providing an EMI-tight assembly. All surfaces are coated to prevent corrosion.

Because of the packaging flexibility available through the use of 1/2-ATR modules, Delco has been able to accommodate unusual form factor requirements with no sacrifice in performance in a wide variety of applications, as in the case of the LANTIRN Pod Control Computer where coldplate cooling and extremely tight packaging constraints imposed by the LANTIRN pod size and form factor resulted in the configuration shown in Figure 1.

Support Equipment

User's Control Console. The M372 User's Control Console (UCC) provides the capability for computer hardware maintenance, operational software development and debug, and system

integration support. Consisting entirely of commercial equipment, and based on the Computer Automation, Incorporated, LSI-4 minicomputer, it includes the following items:

- 64k-word semiconductor RAM
- Distributed I/O system with intelligent cables
- Dual floppy disk subsystem with 486k bytes of storage
- CRT/keyboard
- 30-character-per-second printer
- Autoload ROM
- High speed controller-processor interface.

The UCC is packaged in a small, upright cabinet on casters, has a self-contained power supply, and is sufficiently portable for use in highly remote field locations. A complete software package is provided, including a user-oriented operating system, symbolic debug package, and maintenance programs. The UCC provides complete control of the computer, including:

- Stop, halt, run, and reset
- Instruction step and instruction insert/execute
- DMA read/write
- Real time interrupt and operator controlled interrupt
- Parity detect, illegal address protect, and stop on fault
- Match on operand address, DMA address, instruction address, and DMA data.

By adding an optional I/O exerciser, the UCC can be converted into a full-

up tester. The I/O exerciser consists of a dual-redundant multiplex data bus interface and digital input/output electronics capable of fully exercising the I/O subsystem. The I/O exerciser is in the UCC cabinet and interconnected to the computer through operational cable interfaces.

Software Development Station. The Software Development Station (SDS) is an extension of the UCC and is also built around a Computer Automation, Incorporated, LSI-4/90, 16-bit minicomputer with a 64k-word memory. The SDS is supported by sufficient peripheral equipment and software to provide a sophisticated standalone program development station for M372 application programs.

The software to support the facility includes the Delco M372 assembler system and the LSI-4 OS operating system. The assembler system contains a relocatable assembler, linkage editor, and cross reference/edit routines. Outputs include absolute listings after link with extensive error and warning message support, memory maps, external symbol tables, global and module-level cross reference tables, and files to support a symbolic debug capability. Utility programs provide nine-track magnetic tape communication between the SDS and a central data processing facility. This capability allows J73 computer object modules generated by the central data processing facility to be linked with assembly modules prepared on the SDS.

The LSI-4 OS operating system provides file management, LSI-4 program generation, editor programs, and job control to support M372 generation and modification. Utility commands allow the user to assign logical units, execute job files, and manage disks. The user may generate LSI-4 programs that run under the OS operating system and use OS service routines.

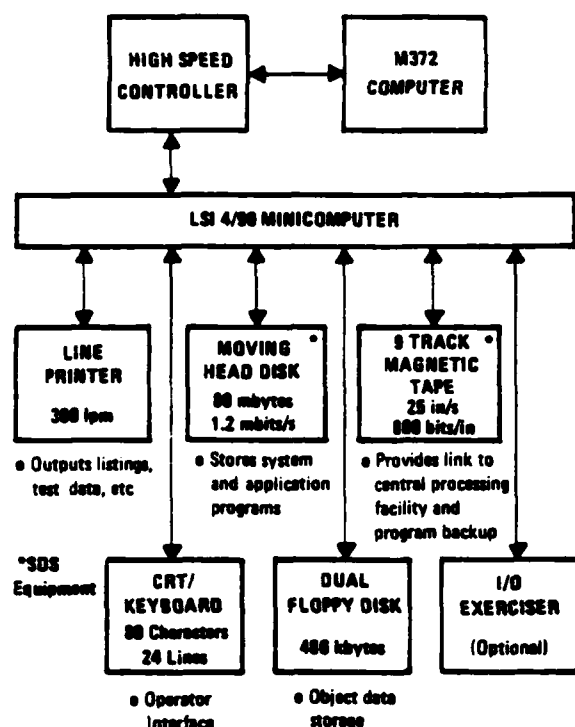


Figure 4. UCC/SDS Block Diagram

SUPPORT SOFTWARE PROGRAMS

PROGRAM	HOST
SYMBOLIC DEBUG PACKAGE	LSI-4
ACCEPTANCE TEST PROGRAM	M372/LSI-4
• Instruction Exerciser • Memory Exerciser • I/O Controller • Controller	
J73 COMPILER (GFE)	IBM370
• MIL-STD-1750A • IBM370	
INTERPRETIVE COMPUTER SIMULATION (GFE)	DEC-10
• MIL-STD-1750A	
MIL-STD-1750A ASSEMBLER SYSTEM (OPTIONAL)	LSI-4
• Assembler • Link Editor • Loader	

An 80-megabit disk provides adequate storage to support the system and the user's application programs. A floppy disk provides a convenient means to move object programs for loading into the M372 computer.

M372 Computer Support Software

Available M372 computer support software is listed below. Government sponsored development is currently underway for a MIL-STD-1750A simulator and a J73 compiler hosted on the DEC-10 computer.

MAGIC V COMPUTER (M572)

The Magic V computer program is a logical extension of Delco's M372 computer development activity. Whereas the M372 is designed to meet early-1980 high performance avionics mission computer applications (with updates, as necessary, consistent with new requirements and available technology), the Magic V computer is targeted for the mid-1980's and beyond, with performance goals matching the anticipated requirements of this time frame. The broad outlines of the Magic V computer began to emerge approximately a year and a half ago when it became evident that the key to meeting the next generation computer requirements was use of VLSI circuit technology. All major computer characteristics -- weight, power, speed, size, cost, reliability -- have the potential for enhancement through VLSI.

Because of Delco's commitment to standards, the first Magic V computer (M572) will implement the two major hardware standards, MIL-STD-1750A instruction set architecture and MIL-STD-1553B multiplex bus. In addition to these standards, a baseline set of performance goals was established, based on an assessment of the requirements of the mid-1980's. The imposition of standards and the performance goals listed below defined

the M572. The M572 is intended to be a total computer system, including a CPU, memory, I/O, power supply, and structure. Production hardware will be available in 1984, and a higher performance version will be available approximately a year later.

The overall M572 program objectives are to:

- Implement the MIL-STD-1750A instruction set architecture with an all-VLSI solution.
- Select a technology that reduces power consumption, thereby minimizing concerns over cooling, packaging, and heat dissipation.
- Minimize the technology risk.
- Establish a reliable manufacturing source, with Delco controlling the design and thus guaranteeing the future availability of the VLSI devices.
- Provide the hardware on a schedule that assures a timely transition from the M372 and its upgraded versions to that of the M572.

Among the more significant specific M572 performance objectives are:

- Throughput of 750 KOPS in 1984 to 1985, increasing to 1.6 KOPS by 1986.
- A 70- to 80-percent reduction in recurring cost compared with a comparable 1981 time frame.
- Reliability of at least 40,000 hours MTBF.
- Power consumption of less than 20 watts, including power supply.
- Fault isolation to a plug-in module, with coverage approaching 100 percent

- A complete computer system, including power supply and enclosure, which occupies 0.1 cubic foot and weighs 6 pounds.

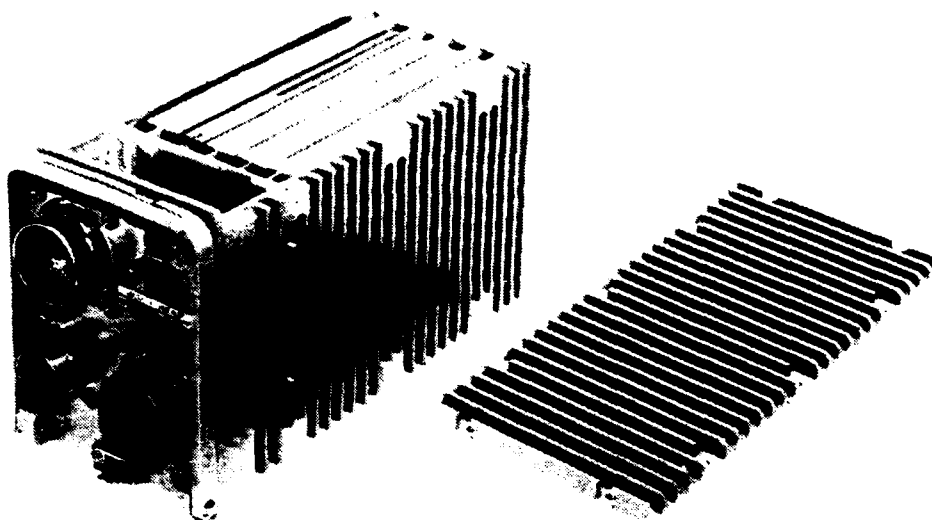
These objectives are based on a blend of an assessment of the mid-1980's requirements with a judgement as to what the technology will allow. If experience is any judge, these goals will probably be exceeded.

A summary of the characteristics of the M572 along with a comparison with the current technology as applied to the F-16 Enhanced Fire Control Computer is presented in Table 1. A mockup of an M572 equivalent to the F-16 Fire Control Computer is also shown.

The Magic V computer is being developed on company funds as a part of Delco's Independent Research and Development effort. This allows Delco the freedom to develop the computer in strict accordance with the above objectives on a schedule consistent with Delco's resources and realistic future program needs. Another factor in this decision is Delco's commitment to developing computer systems, rather than just components, to assure an optimized solution to the needs of its customers.

The development of a custom VLSI computer solution by Delco has several important advantages for the customer, the most significant of which is the guaranteed availability of the VLSI chips for as long as required for a given program. This is possible because Delco will own the masks and have a guaranteed source of processing. Use of custom devices also reduces the number of part types, thereby simplifying procurement and sparing and reducing life cycle cost.

Delco believes that the Magic V program and schedule are low risk, realistic, and timely and address themselves to the requirements of the



PARAMETER	CURRENT TECHNOLOGY M372 F-16 FIRE CONTROL	MAGIC V OBJECTIVES (Compared with F-16 Requirements)
PHYSICAL Size, Weight, Volume	5.4 by 7.8 by 18 inches; 31 pounds; 0.43 cubic foot.	4.1 by 5.3 by 8.7 inches; 6.0 pounds; 0.11 cubic foot.
Power	290 watts.	<20 watts, including power supply.
Speed (DAIS Mix)	587 KOPS (core Memory); 608 KOPS (semiconductor memory).	> 750 KOPS (1984) 1,800 KOPS (1986)
Reliability	1,900 hours.	40,000 hours.
Relative Cost	\$P per system.	\$0.2 to 0.3P per system.
SUBSYSTEMS CPU	MIL-STD-1750A ISA.	MIL-STD-1750A ISA
Memory	Two 32k by 18 bits core modules.	64k by 16-bit plus error detection and correction main memory (single CCA). Nonvolatile main memory backup. Available options include: battery, EEPROM, and Bubble.
Input/Output	• Two channels, dual-redundant MIL-STD-1553B (1 CCA/channel) • Discretes • Analog (dc and synchro) • Direct Memory Access (DMA).	• Single chip MIL-STD-1553B bus • Single chip standard I/O (serial, parallel, discretes, DMA, RS-232 . . . , exact complement - TBD).
Power Supply	8.0 pounds; 210 watts output; and > 70 percent efficiency.	0.7 pound; 20 to 30 watts output; and 75 percent efficiency.
TECHNOLOGY Packaging	• Integrated circuits in flatpacks • Conventional double and multilayer circuit boards • Forced air cooling with heat exchangers.	• Custom VLSI chips in leadless chip carriers (LCC's) • LCC's on ceramic modules (MAM's) mounted on printed circuit cards, or LCC's on metal core circuit cards • Passively cooled.
VLSI	Standard MSI and SSI chips.	• Custom VLSI circuits • Three-micron bulk CMOS.
Fault Management	• Self-test fault isolation to a single replaceable module (LRU) • Coverage - 95%.	• Self-test fault isolation to a single replaceable module (LRU) • Coverage approaching 100%.

Table 1. M572 Computer Characteristics

high performance avionics and space computer markets of the next decade.

CONCLUSION

Delco Electronics supported the Air Force move to a standard architecture relatively soon after the idea was introduced, but not without some indecision in-house. In fact, the acceptance and implementation of these standards forced a costly change in development plans within Delco Electronics. This early decision to produce MIL-STD-1750A computers allowed Delco to capture the first two MIL-STD-1750A Air Force production programs -- LANTIRN and F-16 MSIP.

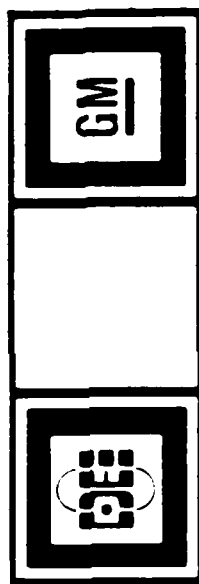
Standardization at the ISA level allows introduction of new devices that lead to new computer systems

that are faster, smaller, lighter, and consume less power. The M572, to be introduced in 1984 as a result of active, current IR&D effort, will be a MIL-STD-1750A machine. This machine may then be followed by the M582 executing the MIL-STD-1862, NEBULA architecture.

There is a complementary requirement, however, that the Military control the new standards properly. With the proper control of standards from the Department of Defense and with development from the private sector, these standards could become a valuable asset.

Delco Electronics believes in the concept, and, if the concept is maintained, Delco Electronics may be manufacturing only Military standard computers 5 years from now.

Delco Electronics



Division of General Motors

DELCO ELECTRONICS DIVISION



R. Schultz
General Manager

KOKOMO OPERATIONS



SANTA BARBARA OPERATIONS

Dr. E. Dale Wilmoth, Manager



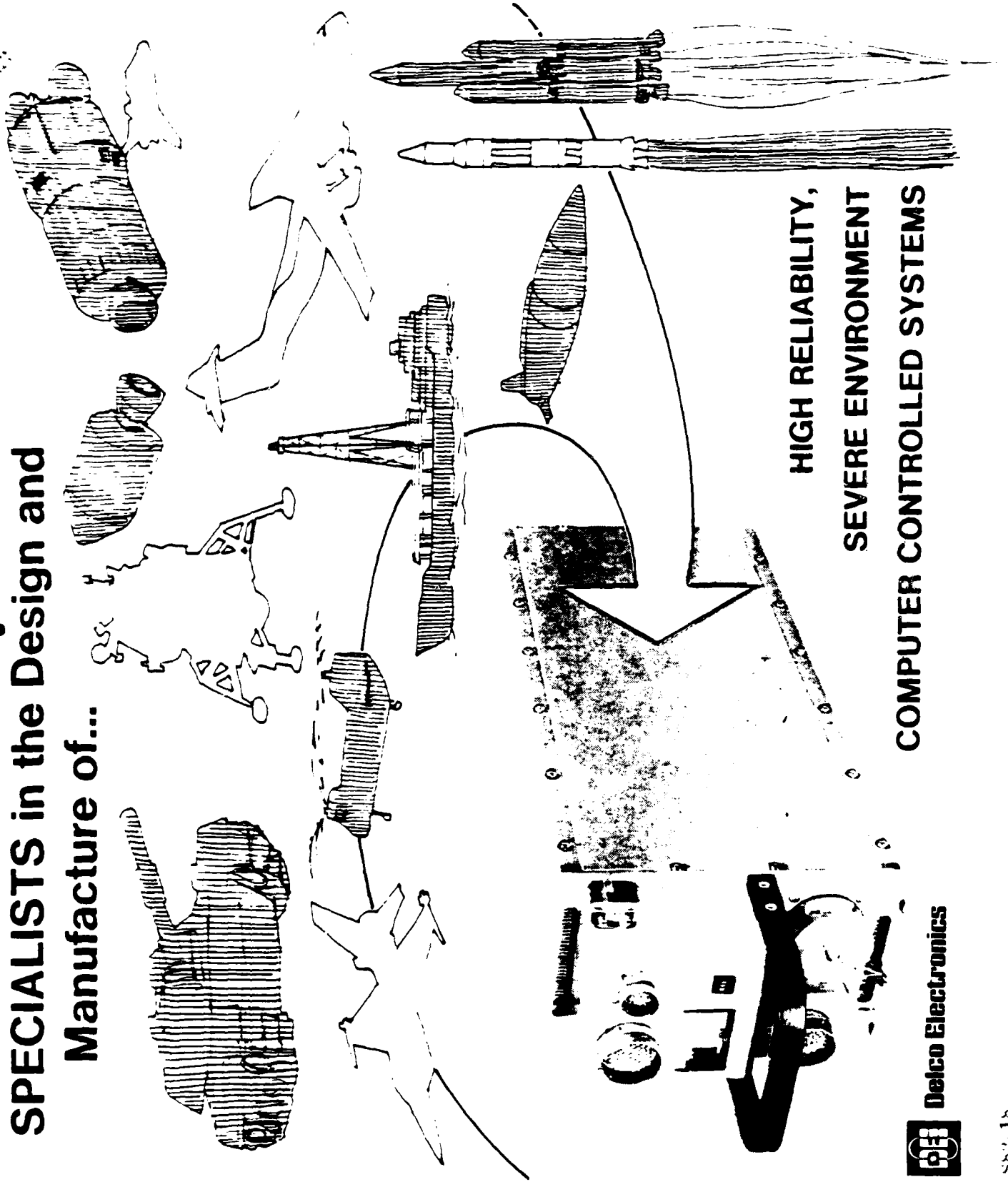
MILWAUKEE OPERATIONS

W. Don Helm, Manager



- 4¼ million square foot plant area
- Fully equipped, modern electronics manufacturing capability
- Offshore manufacturing in Singapore, Mexico, and Europe
- Stable work force of 14,000

SPECIALISTS in the Design and Manufacture of...



**HIGH RELIABILITY,
SEVERE ENVIRONMENT
COMPUTER CONTROLLED SYSTEMS**

Delco Electronics

SEE 18

LOW ALTITUDE NAVIGATION TARGETING INFRARED NIGHT (LANTIRN) PROGRAM

LANTIRN requires small size, high performance equipment built to military standards.

Delco's M372 Computer matches these requirements and was selected for the Martin Marietta LANTIRN application.

Delco's M372 Computer embodies MIL-STD-1750A ISA, MIL-STD-1553 Data Bus, and MIL-STD-1589E HOL.



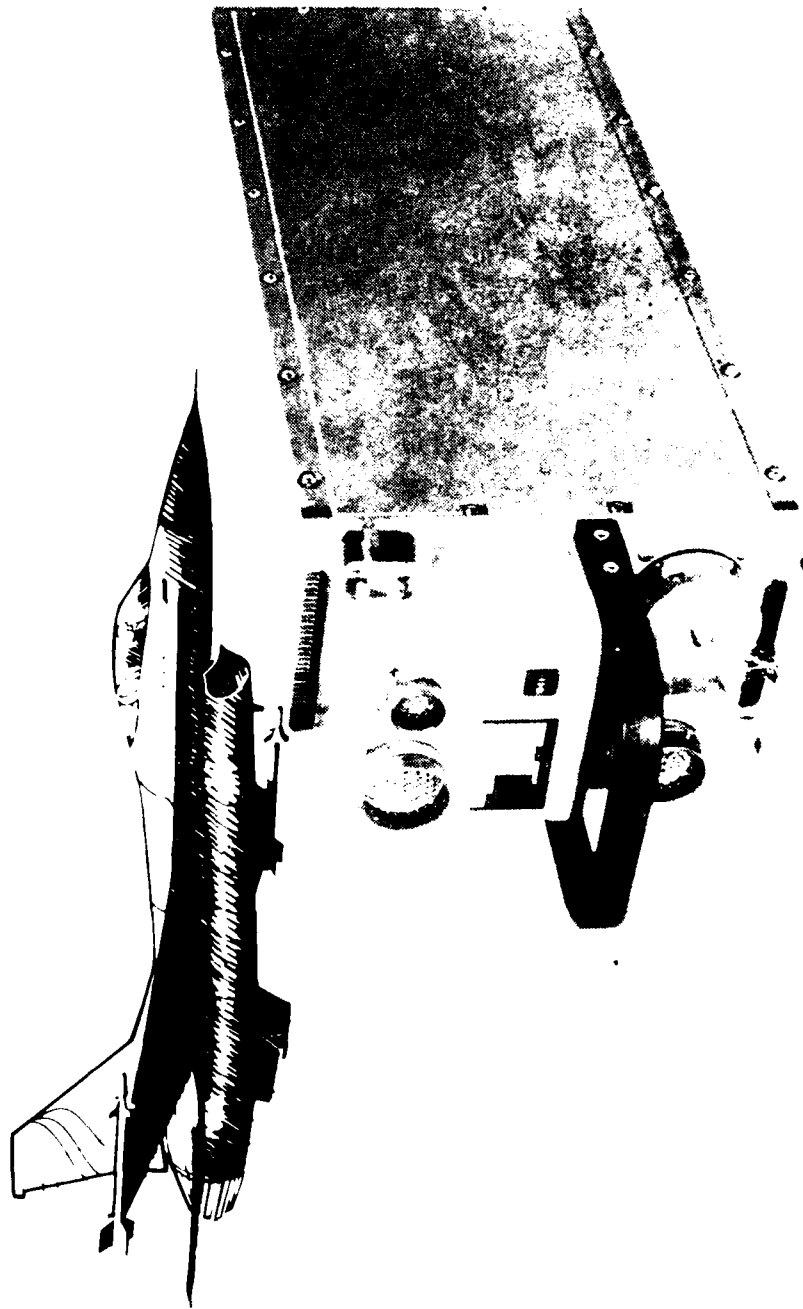
Delco Electronics

F-16 ENHANCED FIRE CONTROL COMPUTER

F 16 Enhanced FCC required small size, high performance equipment built to military standards, 'F3' compatible with existing M362F.

Delco's M372 computer matched these requirements and was selected.

Delco's M372 computer embodies MIL-STD 1750A ISA, MIL-STD-1553 Data Bus, and MIL-STD-1589B HOL.



Delco Electronics

M372 ENHANCEMENTS

MEMORY MANAGEMENT

- Achieve full compliance with MIL-STD-1750A memory management option.
- Increase addressing from 64 thousand to 1 million words.
- Provide this option on single 1/2-ATR circuit card assembly.

MULTIPLEX BUS

- Reduce current hardware by at least 35%.
- Provide single 1/2-ATR circuit card implementation.
- Achieve easier maintainability at SRU level, reduced cost and power, and increased reliability.

PARTS REDUCTION

- Reduce parts by 45 to 50% for processor and associated functions, including memory and I/O control.
- Achieve reduced cost and power, increased reliability, and easier SRU isolation.



Delco Electronics

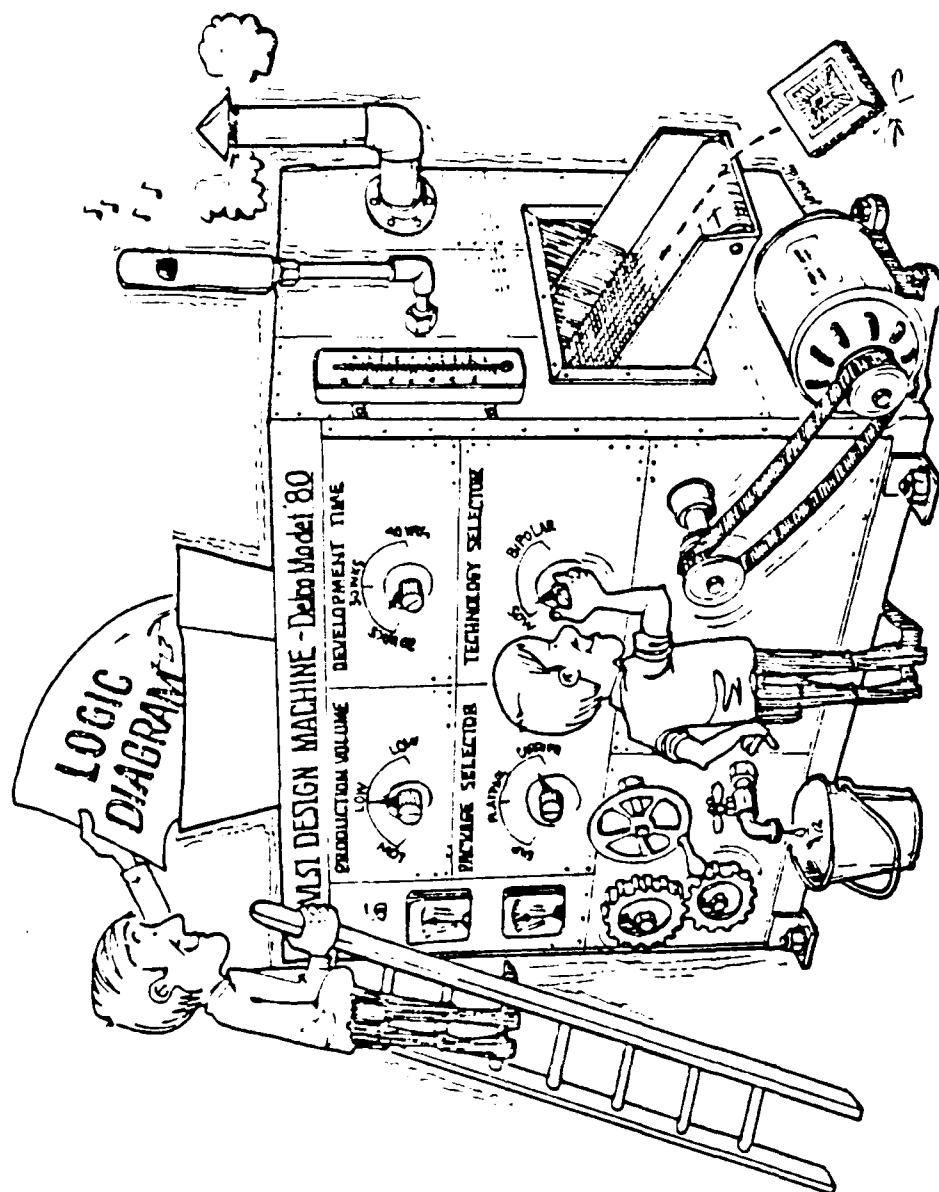
F-16 FIRE CONTROL COMPUTER

(M372)

INSTRUCTION SET	MIL-STD-1750A
SIZE	1/2-ATR by 18.6 inches
WEIGHT	31.3 pounds
POWER	280 watts (3-phase, 100-hertz, 115-volt source)
THROUGHPUT	587 KOPS (DAIS instruction mix)
MEMORY	Two 32k core modules; 750 ns cycle time; 16 bits plus parity
INPUT / OUTPUT	Discretes (26 input; 8 output) Analog (6 dc input; 2 synchro output) 1553B bus (two channels dual-redundant, multiprotocol) DMA (greater than 1 million words per second)
MTBF	2,403 hours



Delco Electronics



Delco Electronics

MIL-STD-1750A COMPUTER (M372)



SR2-18



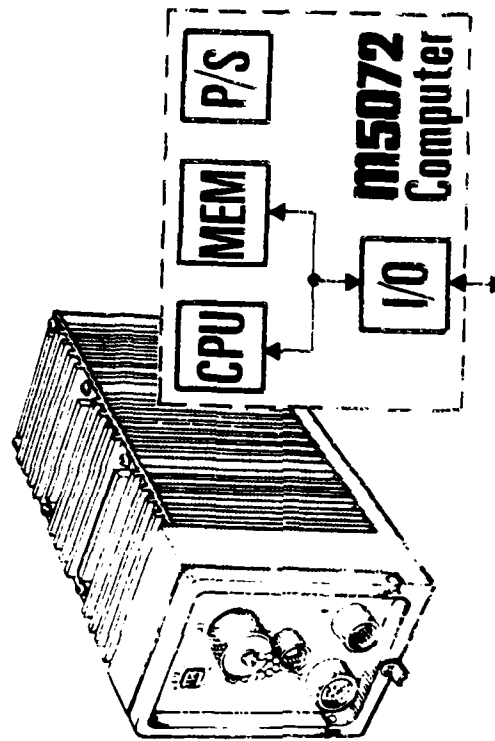
LANTIRN PGD CONTROL COMPUTER
(M372)

INSTRUCTION SET	MIL-STD-1750A
SIZE	5.0 inches by 7.1 inches by 13.1 inches
WEIGHT	17.9 pounds
POWER	271 watts
THROUGHPUT	666 KOPS (DAIS instruction mix)
MEMORY	Nonvolatile program (65, 536 words) Volatile RAM (65, 536 words) Power transient protected RAM (1, 024 words) Electrically alterable ROM (1, 024 words) Word size 16 bits plus parity
INPUT /OUTPUT	Discretes (25 input; 32 output) Parallel bus (DMA: 880k words / s; Program I/O: 200k words / s) 1553B bus (remote terminal)
MTBF	2, 400 hours (projected)



Delco Electronics

**MAGIC V
ALL-VLSI
MIL-STD-1750
MICROCOMPUTER
SYSTEM
DEVELOPMENT**



Delco Electronics

S82-13

WHAT IS MAGIC V?

- Magic V is a total computer system which
 - is keyed to standards
 - is targeted for mid-1980's production
 - maximizes the use of custom VLSI technology
 - is designed for easy performance upgrades
- VLSI is the key to enhancements in
 - Speed
 - Power
 - Size/weight/volume
 - Cost
 - Reliability



Delco Electronics

SR2-18

OVERALL MAGIC V PROGRAM OBJECTIVES

- Implement the MIL-STD-1750A instruction set architecture with an all-VLSI solution.
- Select a technology that minimizes power consumption, thereby minimizing concerns over:
 - Cooling
 - Packaging
 - Heat dissipation.
- Minimize technology risk.
- Establish a reliable VLSI source with design controlled by Delco, thus assuring future availability of the VLSI devices.
- Provide hardware on a timely schedule transition from M372



Delco Electronics

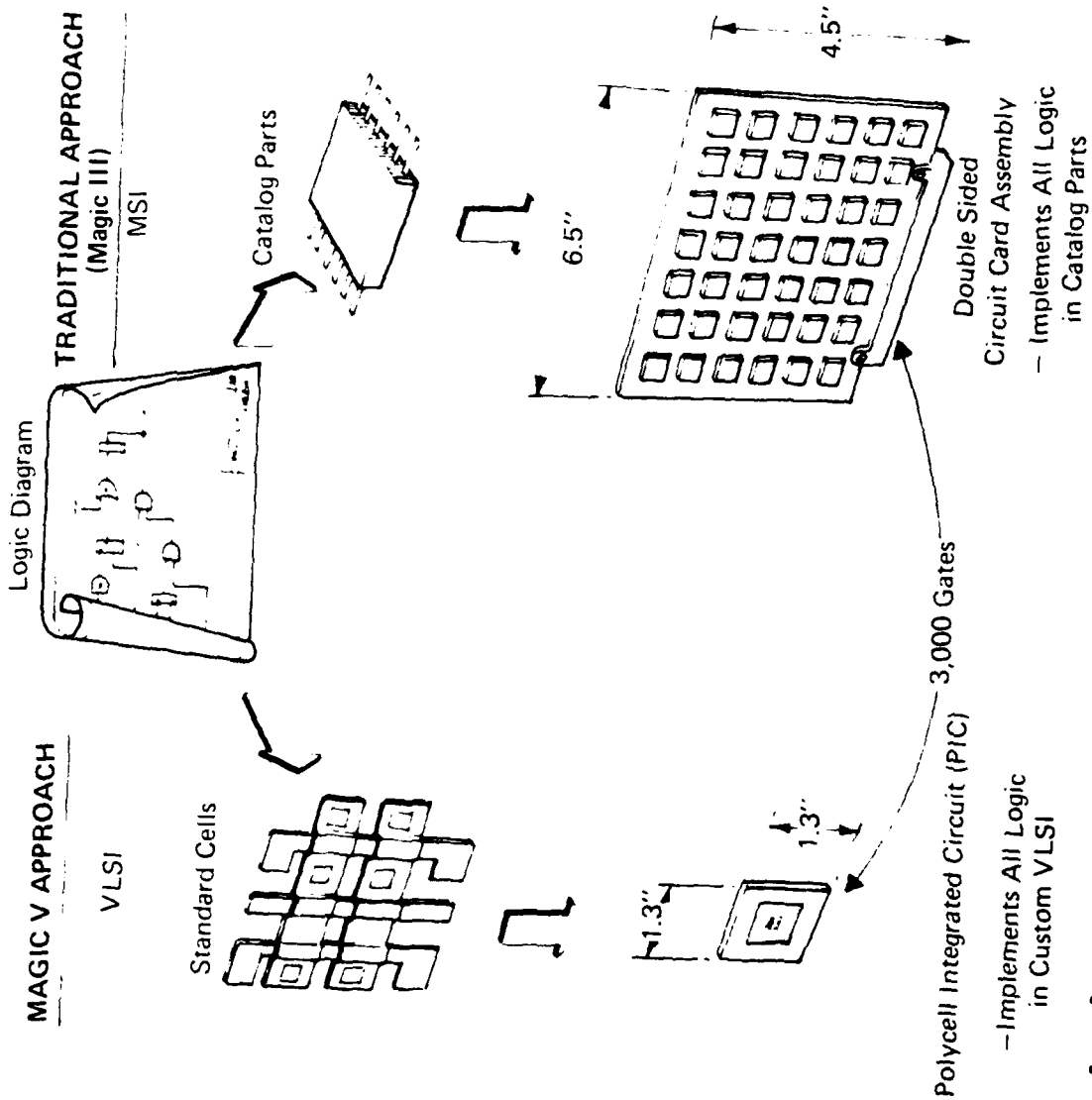
SPECIFIC MAGIC V OBJECTIVES

- Throughput of 750 KOPS (1984 / 1985), increasing to 1.5 MOPS by 1986.
- 70 to 80% cost reduction over comparable 1981 M372 system.
- MTBF of at least 40,000 hours.
- Less than 20 watts power consumption, including power supply.
- Fault isolation to plug-in module, with coverage approaching 100%.
- Complete computer system (including power supply and enclosure) in 0.1 cubic foot with a weight of 6 pounds.

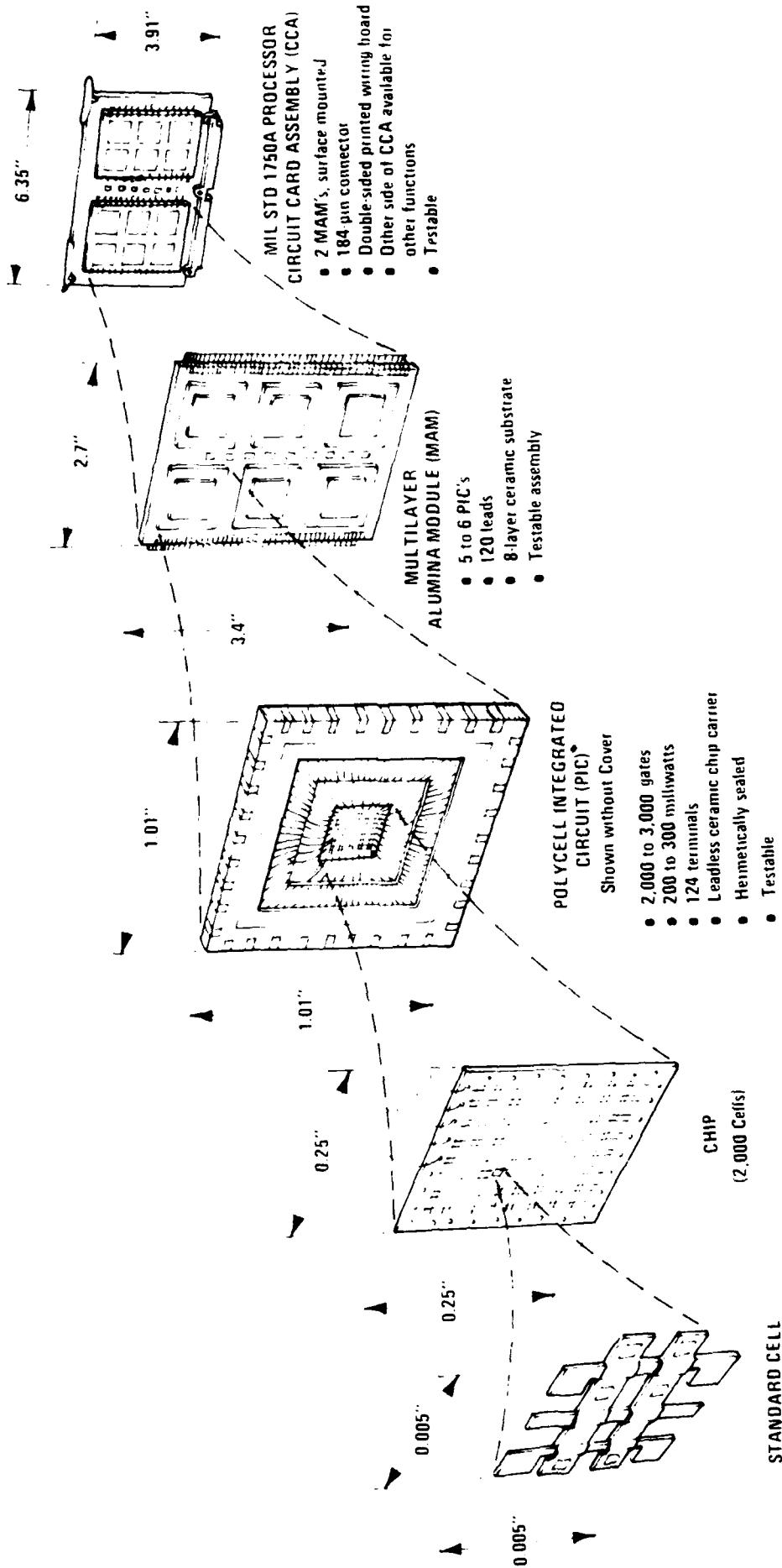


Delco Electronics

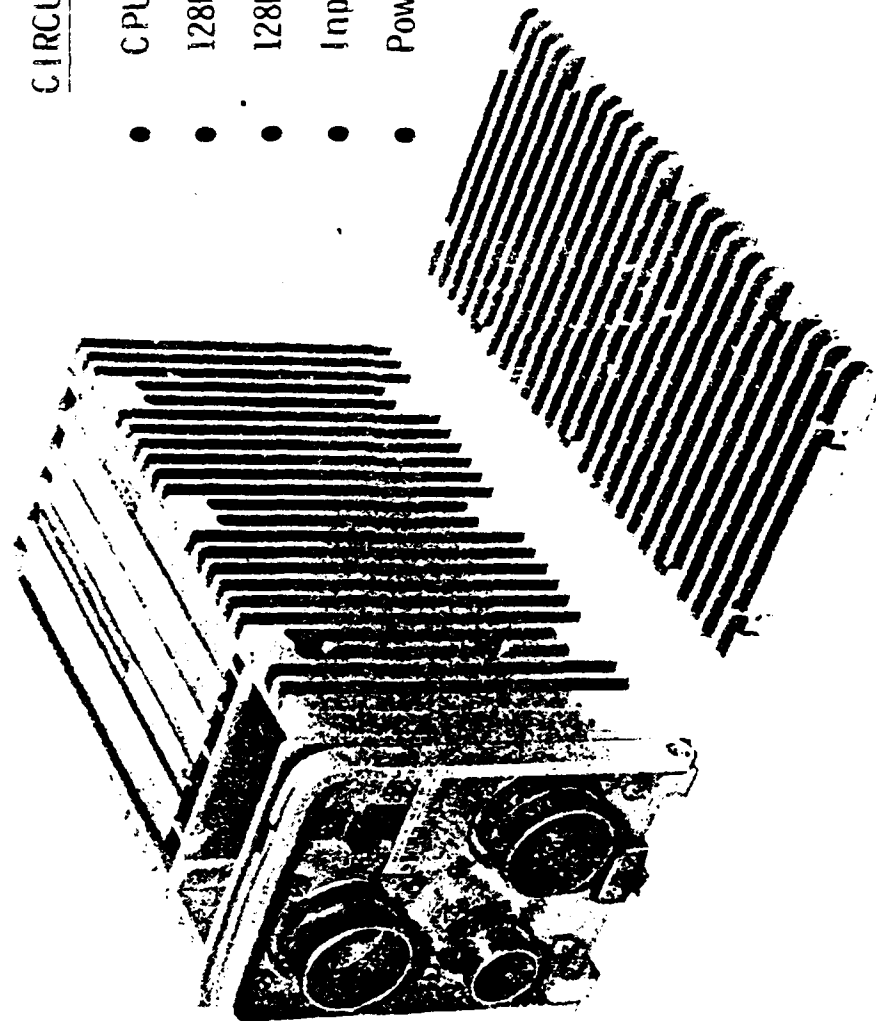
NEW DESIGN APPROACH



MAGIC V PACKAGING APPROACH



TYPICAL MAGIC V COMPUTER SYSTEM



CIRCUIT CARD ASSEMBLIES

- CPU plus Spare Card Side
- 128k CMOS Main Memory
- 128k Nonvolatile S/C Memory
- Input/Output
- Power Supply



Delco Electronics

5802-18

SUPPORT SOFTWARE PROGRAMS

PROGRAM	HOST	SOURCE
SYMBOLIC DEBUG PACKAGE	LSI-4	Delco Electronics
ACCEPTANCE TEST PROGRAM • Instruction Exerciser • Memory Exerciser • I/O Controller • Controller	M372/LSI-4	Delco Electronics
J73 COMPILER • MIL-STD-1750A • IBM 370	IBM 370	U.S. Air Force
INTERPRETIVE COMPUTER SIMULATION • (MIL-STD-1750A)	DEC-10	U.S. Air Force
MIL-STD-1750A ASSEMBLER SYSTEM (OPTIONAL) • Assembler • Link Editor • Loader	LSI-4	Delco Electronics

A NEW SILICON-ON-SAPPHIRE MIL-STD-1750A MICROPROCESSOR

Joseph R. Burns, William Duane, Henry Silcock,
Gregory Portanova, Steven Nicholas,
John Riordan, Richard Meckstroth

Mikros Systems Corporation
Mercerville, New Jersey

ABSTRACT

This paper describes the MKS1750, a new implementation of the United States Air Force MIL-STD-1750A architecture fabricated entirely in SOS/CMOS technology. The MKS1750 is a chip set based on a proprietary microprogrammable 16-bit microprocessor (the MKS16) supported by additional logic implemented as semi-custom gate arrays and ROMs for control store. A total of eleven chips is required. Power consumption of the MKS1750 is less than 1W and throughput is greater than 200 KIPS, measured using the USAF DAIS mix.

A complete description of the chip set is given, including the architecture of the MKS16, gate array partitioning and the functional specification of each array. The paper also discusses the structure of the MKS1750 microprogram, available software support and packaging considerations.

1. INTRODUCTION

The advantages of SOS/CMOS technology in computer applications are well known: high speed, low power consumption, high packaging density, and radiation resistance. While the cost of the sapphire substrate precludes its use in high-volume commercial applications, the aforementioned attributes of SOS/CMOS are overwhelmingly important (and in some cases, imperative) in military computers (1,2).

The Mikros strategy has been to use SOS/CMOS for the design and development of a full 16-bit CPU chip. Additionally, the basic CPU is microprogrammable using external control store to allow emulation of a variety of instruction set architectures.

This paper describes the emulation of the USAF MIL-STD-1750A architecture using the Mikros MKS16 processor chip supported by

SOS/CMOS gate arrays and control store. The chip set executes the full 1750A instruction set (including floating-point) at a throughput of 265 KIPS (DAIS instruction mix) and dissipates less than 1 Watt.

2. MKS1750 ARCHITECTURE

2.1 OVERVIEW

The MKS1750 is a microprogrammed processor which executes the USAF MIL-STD-1750A instruction set (3). It has the following features:

- o MULTIBUS™-compatible interface (4)
- o 16-bit microprogrammable processor
- o Eleven chip LSI implementation

The architecture of the MKS1750 is based on a 16-bit internal bus (the I-bus, see Fig. 1). Sequential execution of 1750A instructions is controlled by the MKS16 processor, which is responsible for instruction fetch and decode, effective address calculations, operand fetch and store, and arithmetic operations. External logic is provided to support 1750A emulation in the following areas:

- o additional microsequencing features
- o MULTIBUS™ interface
- o 1750A interrupts and faults
- o additional register file and addressing logic

The control ROM (2Kx48) provides synchronous control signals to both the external logic and the MKS16.

2.2 THE MKS16 MICROPROGRAMMABLE PROCESSOR

The MKS16 has a register file/ALU/sequencer architecture with internal status, instruction and shift registers (see Fig. 2) (5). The processor communicates with a bidirectional data bus through two multiplexers (IBX, OBX). Internal storage is provided by the scratchpad (SPAD), a 16x16 register file. The A and X registers (AR, XR) are used as accumulators and for shift operations. The status register (SR) contains condition codes and status flags. The 16-bit parallel ALU provides two's-complement arithmetic and boolean operations. The architecture supports a limited degree of parallelism at the micro-operation level.

The MKS16 is driven by a 27-bit microinstruction word. The microinstruction is time-multiplexed into PRIMARY and SECONDARY words which are fetched sequentially from control store during one microcycle. The format of the standard microinstruction word is shown in Fig. 3. In this case, bits C15-17, C31, C32 are not used by the MKS16 and may be used as control bits for external logic.

[illegible]

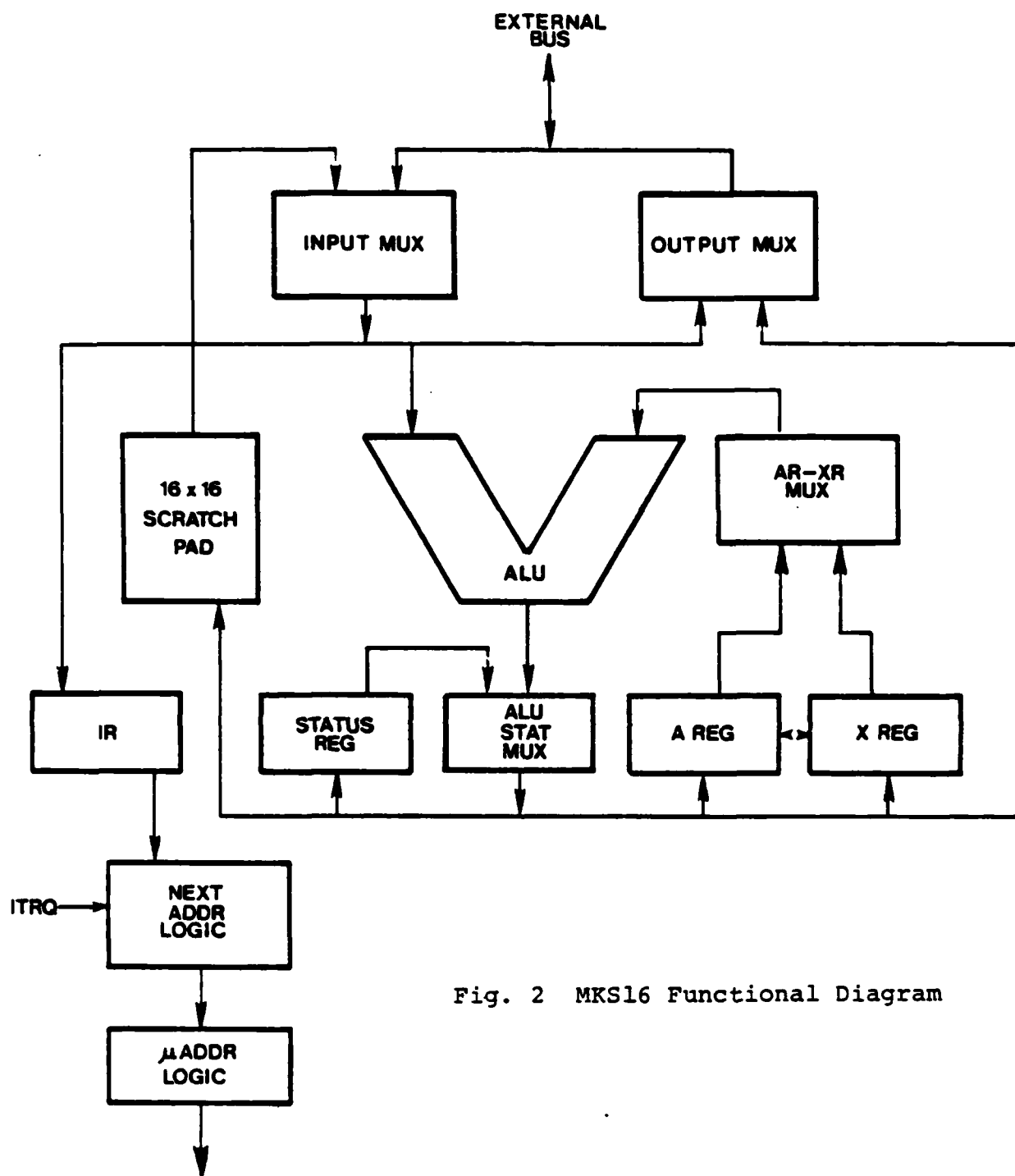


Fig. 2 MKS16 Functional Diagram

The MKS16 microsequencer does not use a default flow-of-control convention; each microinstruction specifies the next microaddress, explicitly. The low-order four bits of the address (the WORD ADDRESS) are given by C27-C30. The high-order four bits (the PAGE ADDRESS) are determined by a LINK MODE specifying the source of the page address. The sequencer allows conditional branching based on the condition codes.

The MKS16 requires three clock signals, and can run at either TTL or CMOS levels. The maximum clock rate is 5.5 MHz, giving a microcycle time of 180 nsec.

2.3 REGISTER STRUCTURE

The MKS1750 uses an external 16x16 register file to provide the 1750A registers. The MKS1750 contains dedicated logic to support register concatenation and register addressing using macroinstruction fields. This logic uses the external instruction register (XIR), which is loaded with a 1750A instruction during the fetch cycle. The XIR S and D fields are used by most 1750A instructions as register addresses, and may be used directly to access the specified register. Alternatively, a literal 4-bit address may be specified directly by the microinstruction word. The S and D fields are implemented as four-bit up/down counters, to provide access to adjacent registers as required by 1750A multiple-precision instructions. The S and D fields may also be concatenated to form an eight-bit up/down counter. Zero-detect logic is provided for all three counters. This logic is used extensively in implementing indexed addressing and shift instructions.

2.4 THE MKS1750 MICROSEQUENCER

The MKS16 internal sequencer is oriented towards instruction decoding. Typically, four bits of the instruction register (IR) are used as the page address, providing a sixteen-way branch in control store. The MKS1750 microsequencer provides logic for expanded microaddress space, microsubroutining, instruction decoding and testing of external conditions.

Three extra microaddress bits are taken directly from the microinstruction word, and are appended to the MKS16-generated address as the high-order bits. The resulting 11-bit address is known as the NORMAL ADDRESS (N-address) and is one of four inputs to the NEXT ADDRESS MULTIPLEXER in the microsequencer (see Fig. 4). Another input is the LONG ADDRESS (L-address) which is a literal 11-bit address specified in the microinstruction. The next address may also be obtained from the return address stack (3x11). The final address mux input is obtained by concatenating the three high-order extension bits with an 8-bit value from the I-bus.

N-addressing is required for a microsubroutine call operation. In this case, the return address is not specified explicitly by the microinstruction, but is determined by incrementing the high-order three bits of the current microaddress. N-addressing is also required by microinstructions which test internal or external conditions.

The MKS16 internal link modes may also be combined with the external modes to provide extra sequencing features. The microsequencer also allows the execution of an instruction to be aborted if certain machine faults occur during execution.

2.5 THE BUS INTERFACE

The MKS1750 is logically compatible with the MULTIBUS™ standard. Address, data, command, and status signals are supported.

The 20 address lines are TTL-compatible signals which provide 1 Mword of physical address space. The 16 bidirectional data lines are TTL-compatible signals which provide a 16-bit data bus. To communicate with slave devices, several control signals are provided: memory read/write (MRDC, MWTC) and I/O read/write (IORC, IOWC). Slave devices acknowledge transfers using the XACK control line.

A multi-master environment is supported using serial priority arbitration. BREQ and BUSY are polled by each requesting device to determine whether a bus access may occur.

There are 8 user-defined interrupt signals which are processed by the interrupt controller. A bus transfer example is shown in Fig. 8.

2.6 INTERRUPTS AND FAULTS

The MKS1750 contains three registers dedicated to the 1750A pending interrupt (PI), interrupt mask (MK), and fault (FT) registers. The hardware determines from the values of PI and MK if there is a valid interrupt which requires service by the MKS16. If so, this logic asserts the MKS16 interrupt request line, and this in turn is detected by the MKS16 sequencer. The MKS1750 normally tests for the presence of a valid interrupt at the end of each instruction. If one is detected, the processor determines its priority by reading PI and MK, calculates the address of the new context, and performs the context-switch operation. Certain 1750A interrupts (executive call and arithmetic exceptions) are not asynchronous events as they occur only as the result of instruction execution. In these cases the microprogram must set the correct PI bit.

Machine faults set the appropriate FT bit to cause a machine error interrupt (provided it is not masked). Certain machine faults cause the current instruction to be aborted (see 3.7).

2.7 SYSTEM TIMING

The MKS1750 uses a four-phase clock. $\phi 1$ is used to latch the primary microinstruction word into the processor and gate arrays. $\phi 2$ is used to latch and output the contents of the MKS16 scratchpad during a read cycle. $\phi 3$ (WAIT) is used to synchronize processor-bus transactions. The processor waits during this phase for a bus access and for the accessed slave device to indicate transfer completion. $\phi 4$ (CLK) is used to latch the secondary microinstruction, the registers, the next address and the condition codes. A system timing example is shown in Fig. 8.

3. PRINCIPLES OF EMULATION

3.1 THE CONTROL WORD

The MKS1750 uses a 48-bit microinstruction word, shown in Fig. 5. Bits C1 - C14 and C18 - C30 are used to control the MKS16 processor.

Several microinstruction fields are used to determine the next microaddress. The high-order three bits (the BLOCK ADDRESS) are given by C17, C31, C32. C15, C24 - 30 are used to form the L-address. C35 - 36 are used to control the next address multiplexer and return address stack.

The interpretation of bits C33 - 48 depends on the value of C16. If C16 = 1 then C33 - 48 is a 16-bit constant which is read by the MKS16 from the I-bus. This provides arbitrary constants without the use of a separate constant ROM. If C16 = 0 then C33 - 48 are external logic control signals. C33 - 34 control XIR reformatting. C37 - 38, C48 control the MULTIBUS[™] interface. C39 - 40 control latches for the XIR, the memory address register (MAR) and the register file. Register file addressing and the XIR counters are controlled by C41 - 45. C41, C46 - 47 control the external condition multiplexer for testing the values of the XIR counters.

3.2 CONTROL STORE ORGANIZATION

Control ROM is organized as 2Kx48. The physical micro-address format is shown in Fig. 6. The distinction between page and word address is valid only for N-addressing. The return address for a subroutine call is obtained by incrementing the block address of the calling microinstruction. The microprogram is functionally partitioned so that floating-point and diagnostic microcode resides in blocks 4 - 7, while the rest of the microcode occupies blocks 0 - 3, and for this reason the return address generator increments the block address modulo 4 rather than modulo 8.

Fig. 3 - MKS16 Microinstruction word

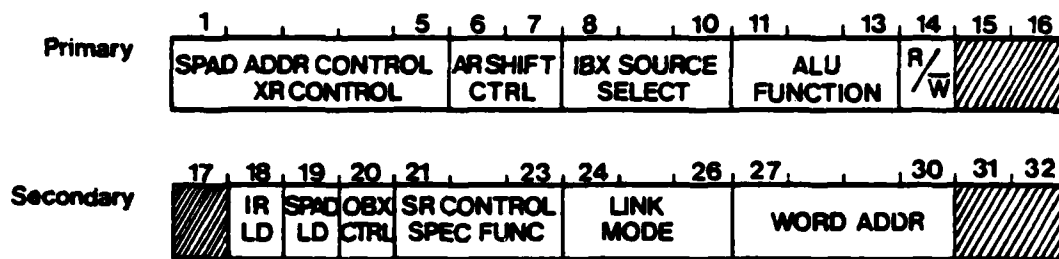


Fig. 4 - MKS1750 Microaddressing

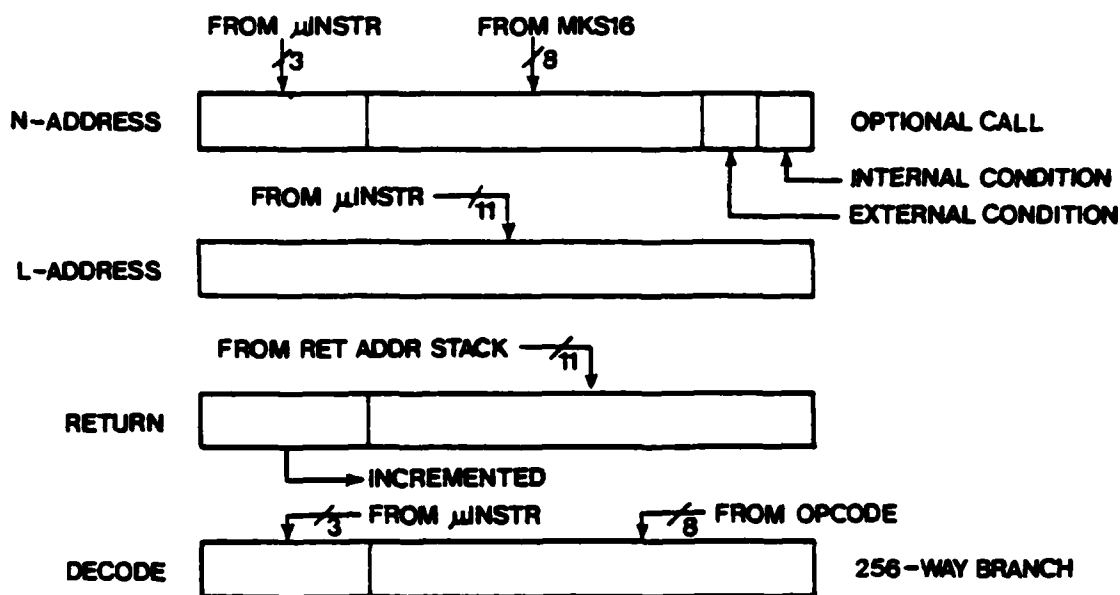


Fig. 5 - MKS1750 Microinstruction Word

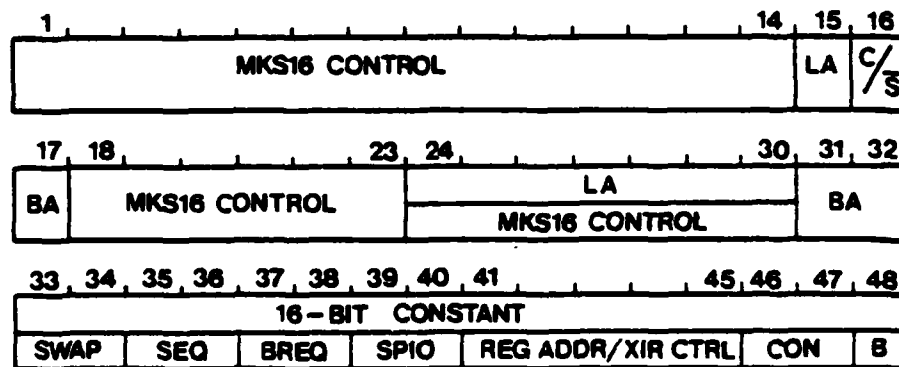
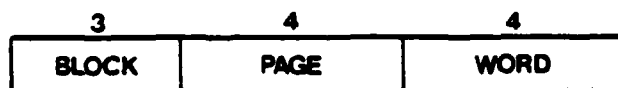


Fig. 6 - Physical Microaddress Format



3.3 MICROPROGRAM STRUCTURE

The MKS1750 microprogram makes extensive use of micro-subroutines; three levels are supported. The functions performed by the subroutines are determined by analyzing the 1750A instruction set in the following areas:

- o instruction format
- o address mode
- o length of operand(s)
- o precision of operand(s)
- o effect on condition codes

The execution of a 1750A instruction is divided into several phases, each of which requires only a subset of the above information. For example, effective address calculation only requires knowledge of the instruction's address mode. All effective address calculation routines return with the address loaded to the MAR and all execution routines leave their results in prespecified internal MKS16 registers. As a result, there are three "operand store" routines, for 16, 32 and 48-bit operands. Each arithmetic operation has one routine for each applicable data type; there are four ADD routines, for single/double precision fixed, floating and extended floating-point numbers. Effective address calculation is combined with the operand fetch phase, so that each 1750A address mode has one routine for each applicable operand length. Each of these routines is responsible for both calculating the effective address and fetching the correct number of operands.

3.4 REGISTER ALLOCATION

The MKS16 internal 16x16 register file (scratchpad) is used for working registers and to maintain the status of the 1750A machine. It is possible to perform a read-modify-write operation on an internal register during one microcycle. In addition, the MKS16 instruction register (IR) is used to contain the 1750A instruction word and the MKS16 status register (SR) is used to maintain the 1750A condition codes. (See Fig. 7)

Registers 0 - 9 are available as temporary registers for any instruction. Registers B - F are used to contain the 1750A status. Register F is dedicated to the instruction counter; register C contains a copy of the instruction word. Register B is used by the BEX instruction. Registers D and E are used in conjunction with the MKS16 SR to maintain the 1750A status word.

3.5 INSTRUCTION EXECUTION

Instruction execution starts with the fetch/decode phase. During this phase the IC is incremented, the next instruction is fetched, and the MKS16 tests for a valid pending interrupt. If there is a valid interrupt the fetch sequence is aborted.

The next microaddress is determined by the high byte of the instruction fetched, providing a 256-way branch. The effective address is then calculated and the operand(s) are fetched and loaded to internal working register(s). For 1750A immediate short address modes, one operand is extracted from the instruction word itself; for IC-relative mode (branch instructions) the jump address is calculated and loaded to a working register.

Final instruction decoding is performed at this point for base-relative indexed and immediate long modes, and the operation specified by the instruction is performed. The result is left in internal register(s). Conditional branch instructions update the IC if necessary, and arithmetic and logical instructions update the condition codes as required. The operand store phase transfers the results to the appropriate destination, usually the register file. The number of words transferred is determined by the precision of the instruction.

3.6 I/O

The MKS1750 implements I/O instructions as follows:

- o the command word is loaded to the MAR, and the processor issues an I/O bus request.
- o the data transfer takes place during the next cycle.

However, certain XIO commands require the microprogram to manipulate the status word or interrupt enable flag explicitly. The MKS1750 implements all mandatory 1750A XIO commands.

3.7 ABORT FAULTS

The 1750A Standard defines an "Instruction Set Architecture" to be the programmer's view of the machine. The standard specifies the state of the machine "between instructions" and does not address what happens during the execution of an instruction. As a result, there are certain "gray areas" in the standard which must be resolved in an implementation-dependent manner.

In particular, certain machine faults may occur as the result of a memory access during the execution of an instruction. If the execution of the instruction is allowed to proceed, the results are unpredictable (e.g. executing an instruction when the fetch caused a parity error). In the MKS1750, this situation is resolved by terminating the instruction, if any of the following "abort faults" occur: CPU memory protect fault, memory parity fault, or illegal address fault. When such a fault occurs, the microprogram jumps immediately to an abort fault handler. This microroutine uses an instruction-length flag to ensure that the IC saved during the subsequent machine error interrupt has a consistent value.

3.8 FLOATING-POINT ARITHMETIC

1750A floating-point instructions are implemented in firmware. The MKS16 contains shift control logic which permits detection and automatic correction of partial product overflow during multiplication, and of mantissa overflow during floating-point addition. The microprograms also use some MKS1750 features in non-standard ways. For example, the MKS1750 may perform multiple operations and test multiple conditions simultaneously. This is used in floating-point normalization, shown below. AR and XR contain the un-normalized mantissa, and XIR contains the exponent plus one, internally represented in excess 128 form:

```
708:  SLLX DECSD TESTSD TOV      :7,CI,8
709:  RRCX TOV                   :(exit)
70A:  XIR = IRCOPY;              :(underflow)
70B:  RRCX TOV                   :(exit)
```

The instruction at 708 is the single-instruction normalize loop. It shifts the entire mantissa left, decrements the exponent, and tests two conditions: overflow of the mantissa and zero of the exponent. If both are false, instruction 708 repeats. If the exponent reaches zero before the mantissa overflows, the next instruction logic selects 70A, which is the first instruction of the exponent underflow handler. Mantissa overflow indicates that normalization is complete, and the next instruction logic selects either 709 or 70B, depending upon whether the exponent has reached zero. Either case is acceptable, and so both instructions perform a right shift to correct for the mantissa having been shifted one bit too far.

3.9 ARCHITECTURAL INSTRUCTIONS

Several 1750A instructions require special attention. For example:

- o TSB - the microprogram inhibits external memory accesses during a TSB instruction by issuing a bus request every cycle, which causes the arbitration logic to lock the bus.
- o MOV - the microprogram checks for the presence of a valid interrupt between each single-word transfer. In addition, for the MOV instruction the microprogram does not increment the value of the instruction counter until all transfers are completed. Thus, if an interrupt occurs and is serviced, on return the execution of the MOV instruction will start over again. This scheme produces the desired result since the registers used by the MOV instructions always contain the correct values.

Fig. 7 MKS16 Register Allocation

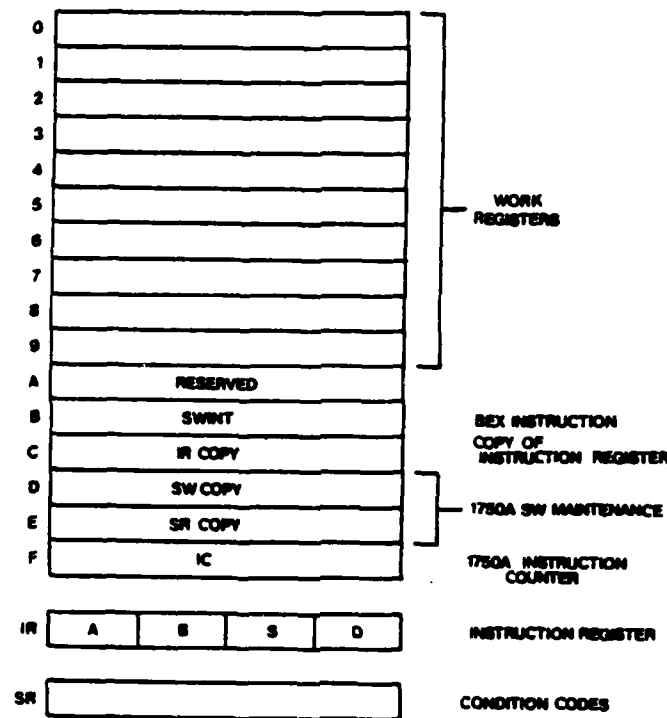
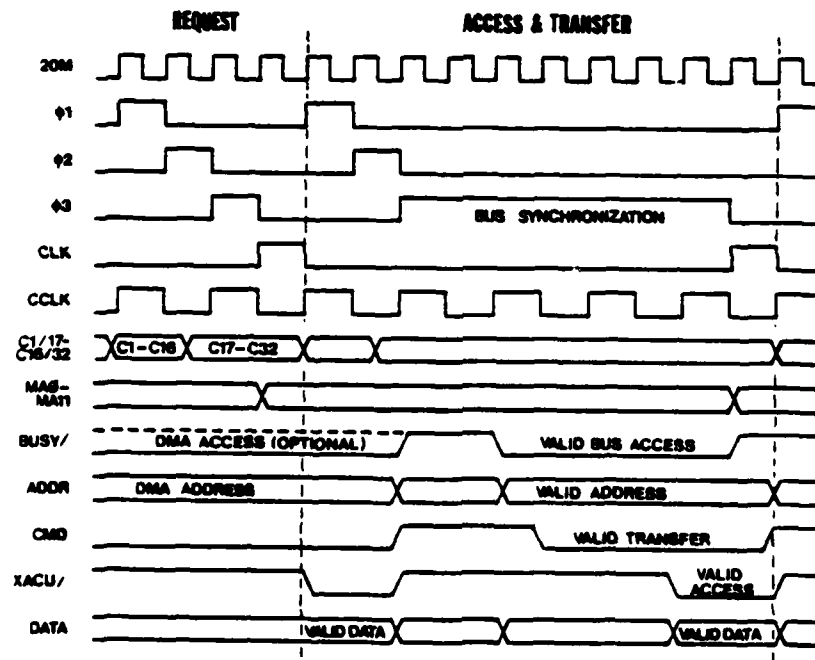


Fig. 8 MKS1750 System Timing



4. MKS1750 IMPLEMENTATION

4.1 FUNCTIONAL PARTITIONING

The partitioning is based on a 1400-gate 102-pin SOS/CMOS universal array manufactured by RCA. The MKS1750 logic is partitioned into three arrays. The boundaries of each array are shown as broken lines in Fig. 1. The emulating controller unit (ECU) contains the logic required for microsequencing, XIR manipulation, and external register control. The bus controller unit (BCU) handles bus transfers, system timing, logical addressing, and bus arbitration. The interrupt controller unit (ICU) processes interrupt requests and executes mandatory XIO instructions for P1, MK and FT. The external 16x16 register file is implemented using a second MKS16 chip. Six 2Kx8 ROMs are used for control store.

4.2 EMULATING CONTROLLER UNIT (ECU)

The ECU contains the logic required for instruction decode, register control, and microsequencing. The XIR provides instruction fields for register addressing. A 3x11 microaddress stack allows microsubroutining.

The XIR may be read into the MKS16 from the I-bus. The value may be modified to facilitate instruction decoding, as follows:

<u>C16</u>	<u>C33</u>	<u>C34</u>	<u>OPERATION</u>
0	0	0	NOP: no operation
0	0	1	SWAB: swap bytes
0	1	0	SWSD: swap S and D fields
0	1	1	SEXT: extend sign of low order byte
1	X	X	read 16-bit constant

The register file may be addressed by the XIR S or D fields, or by a four-bit literal address. The counters are controlled as follows:

<u>C42</u>	<u>C44</u>	<u>C45</u>	<u>OPERATION</u>
0	0	0	NOP
0	0	1	DEC(SD): 8-bit decrement
0	1	0	NOP
0	1	1	INC(SD): 8-bit increment
1	0	0	DEC(S): 4-bit
1	0	1	INC(S): 4-bit
1	1	0	DEC(D): 4-bit
1	1	1	INC(D): 4-bit

Bits C16, C35 - 36, C46 - 47 are used to control the microsequencer.

4.3 INTERRUPT CONTROLLER UNIT (ICU)

The ICU contains the logic required for the MIL-STD-1750A PI, MK and FT. In addition, a valid interrupt request to the processor is generated according to the following equation:

$$\text{valid interrupt} = \text{PI}_0 + \text{PI}_5 + (\text{PI}_n \cdot \text{MK}_n) \cdot \text{ENB} + \text{PI}_1 \cdot \text{MK}_1$$

where $n = 2, 3, 4, 6-15$ and ENB is the interrupt enable flag.

The Pending Interrupt Register is a set of sixteen flip-flops which are set and reset using the RPI, SPI and RPIR XIO commands. The Interrupt Mask Register is a set of sixteen flip-flops which are set and reset using the SMK and RMK XIO commands. MK is also saved and restored as part of the processor context by the interrupt microprogram.

When an XIO instruction is executed, the 16-bit command field is loaded to the MAR, and an I/O request is issued. For mandatory XIO commands, no system bus access is required. If the XIO command is not local (such as a programmed I/O channel), then bus arbitration is required. A timeout will occur if the I/O device is not present, and the appropriate FT bit is set.

The Fault Register is a set of sixteen RS flip-flops used for indicating machine faults. Setting any FT bit causes bit 1 of PI to be set. The FT is controlled by the RCFR XIO command.

4.4 BUS CONTROLLER UNIT (BCU)

The BCU provides logic to address memory and I/O devices, synchronize bus transfer cycles, supply processor clocks, and interface to the MULTIBUS™. A 16-bit logical address is provided by loading the MAR prior to a bus transfer. For memory management, the AS and PS fields of the SW are loaded to the 8-bit External Status Register (XSR).

During the request cycle, the address of the slave device (I/O or memory) is latched into the MAR if the device is not local, a bus request is issued, and the transfer type is latched. During the transfer cycle, the processor can access the bus if no higher-priority device is requesting it. The processor delays the CLK phase until XACK is received. CLK is then used to latch the slave data into a register, XIR, or MAR.

4.5 PACKAGING CONSIDERATIONS

The MKS1750 microprocessor is packaged as a set of eleven hermetically-sealed leadless chip carriers, which are standard JEDEC types. This includes:

- o three 132-pin type A carriers, for SOS gate arrays
- o two 48-pin type C carriers, for the MKS16 and register file

- o six 32-pin type C carriers, for 2Kx8 control ROMs

Leadless carriers provide greater packaging density as they are smaller than conventional DIPs and the shorter lead lengths improve the switching characteristics of the processor. The chip set may be mounted on a small ceramic card to accomodate the particular application form factor required (6).

4.6 SOFTWARE SUPPORT

Software support for MKS1750 development is based on the MIL-STD-1750A Support Software Package which was developed by McDonnell-Douglas under USAF contract. This package includes a macropreprocessor, assembler, linker, 1750A simulator and utilities for formatting and library maintenance.

This package, installed on a local IBM 3033 VM/CMS time-sharing system, was used to develop a resident monitor/debugger for the MKS1750. This monitor is compatible with the support software package, and allows users to develop and debug 1750A software during system prototyping. The monitor supports "quick look and change" commands for memory and 1750A registers, down-line loading of 1750A programs developed using the support package and program execution under breakpoint control. The monitor also includes I/O utilities to support two serial RS232C lines (for a terminal and downline loading) and optional dual floppy disk drives.

4.7 MKS1750 SUMMARY SPECIFICATIONS

ARCHITECTURE	full implementation of MIL-STD-1750A (Notice 1), based on 16-bit microprogrammable processor
TECHNOLOGY	SOS/CMOS - eleven chips
POWER DISSIPATION	less than 1W at 5.5 MHz, 10V
TEMPERATURE	-55°C to +125°C
SIZE	4 x 4 x 0.5 in. (typical)
RADIATION RESISTANCE	10^{10} rad (Si) transient, 10^4 rad total dose
MICROCYCLE TIME	180 nsec
PERFORMANCE	265 KIPS (100 nsec memory) 240 KIPS (250 nsec memory) using DAIS mix (16% floating-point) 450 - 540 KIPS (fixed point only)

ADDRESS SPACE 64K (optional memory management)

EXTERNAL
INTERFACE MULTIBUS™-compatible

5. SUMMARY

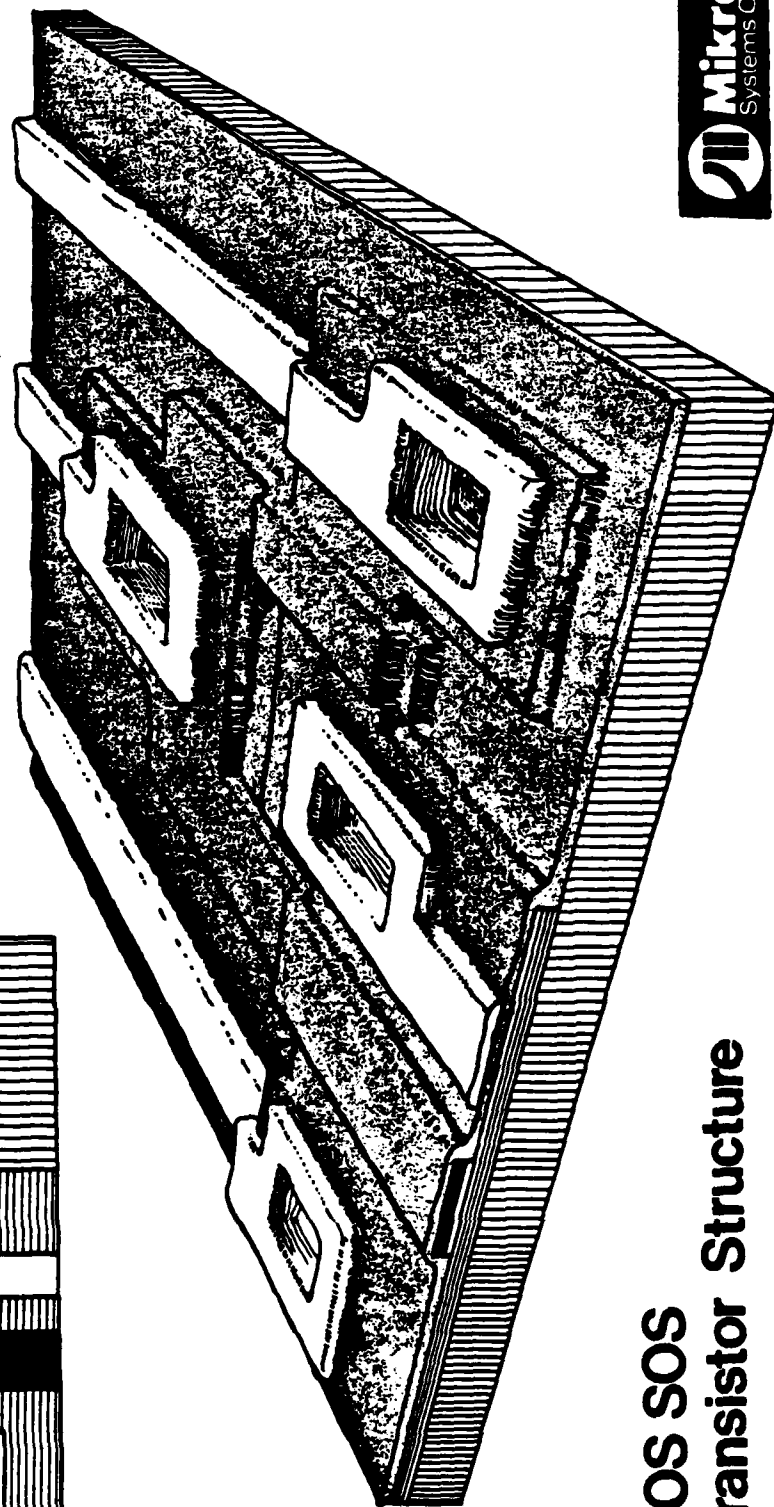
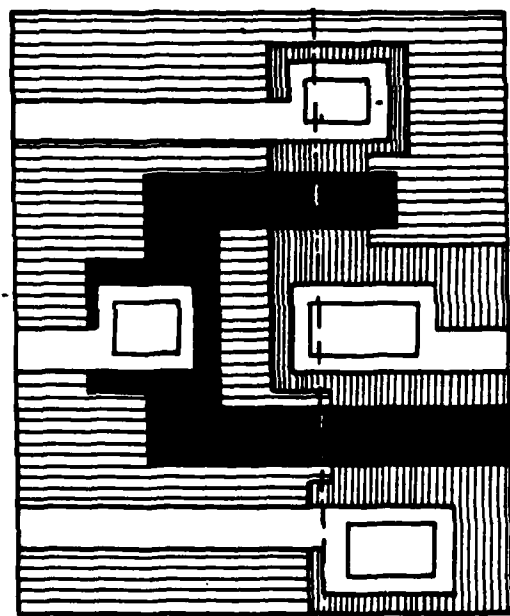
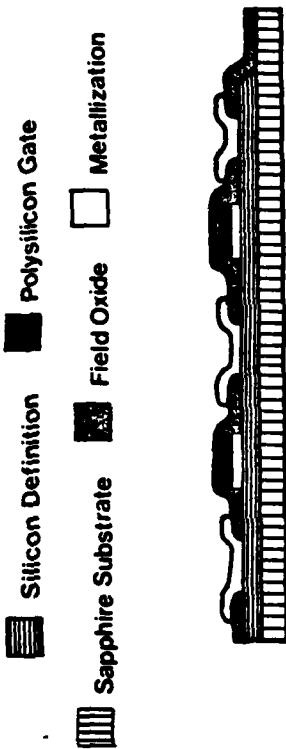
The MKS1750 is a new hardware/firmware realization of the standard 1750A architecture. The eleven-chip set is an all SOS/CMOS microprocessor which is high performance, low power, small in size and radiation resistant. These features make the MKS1750 attractive for a variety of Air Force and other embedded computer applications.

REFERENCES

1. J. Rogers, 'Nuclear Survivability Concepts for Airborne Computers', Trans. AIAA Conf. on Aerospace Computers, Oct. 1977.
2. E.M. Reiss, J.P. McCarthy and A. Bishop, 'A Versatile CMOS/SOS LSI Chip Set for Military Communications Applications', RCA Solid State Division, Somerville, NJ.
3. U.S. Dept. of Defense, "Military Standard Sixteen-Bit Computer Instruction Set Architecture", MIL-STD-1750A (USAF), July 1980 (Notice 1, Nov. 1981).
4. Intel Corporation, "Intel Multibus Specification", Santa Clara, California, 1979.
5. Henry Silcock, 'A User-Microprogrammable Sixteen-Bit Microprocessor for Military/Aerospace Applications', Proc. IEEE National Aerospace and Electronics Conference (NAECON), May 1981.
6. Darrell Barker, '16-Bit Radiation-Hard Emulating Computer', Proc. IEEE NAECON, May 1981.

MULTIBUS™ is a trademark of Intel Corporation.

This paper copyright Mikros Systems Corp. 1982.

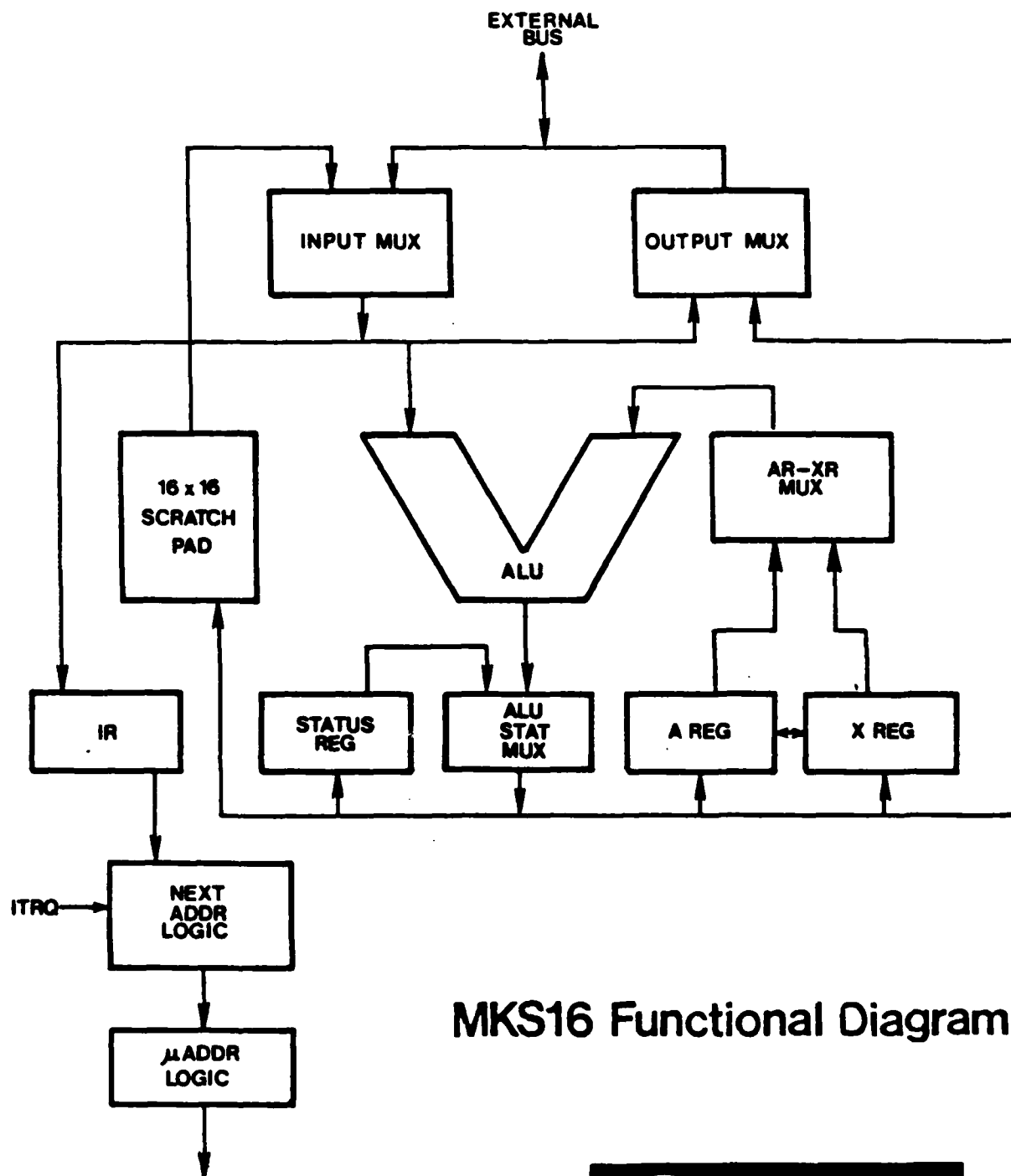


CMOS SOS Transistor Structure

MKS16 Pinout

D ₀	1	48	C ₁₄ / C ₃₀
D ₁	2	47	MA ₇
D ₂	3	46	MA ₅
D ₃	4	45	C ₁₂ / C ₂₈
D ₄	5	44	MA ₆
D ₅	6	43	C ₁₃ / C ₂₉
D ₆	7	42	NC
D ₇	8	41	C ₅ / C ₂₁
D ₈	9	40	MA ₄
D ₉	10	39	C ₁₁ / C ₂₇
D ₁₀	11	38	C ₆ / C ₂₂
D ₁₁	12	37	C ₇ / C ₂₃
D ₁₂	13	36	C ₄ / C ₂₀
D ₁₃	14	35	C ₁ / C ₁₇
D ₁₄	15	34	C ₂ / C ₁₈
D ₁₅	16	33	C ₁₀ / C ₂₆
V _{CC}	17	32	C ₈ / C ₂₄
V _{SS}	18	31	C ₉ / C ₂₅
V _{DD}	19	30	C ₃ / C ₁₉
CLK	20	29	MA ₀
Φ ₂	21	28	MA ₁
MR	22	27	MA ₂
Φ ₂	23	26	MA ₃
SR1	24	25	SR0

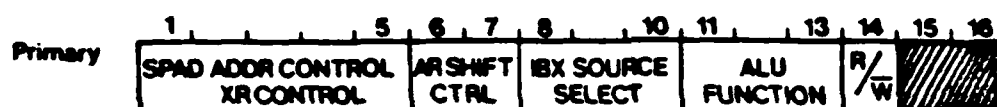


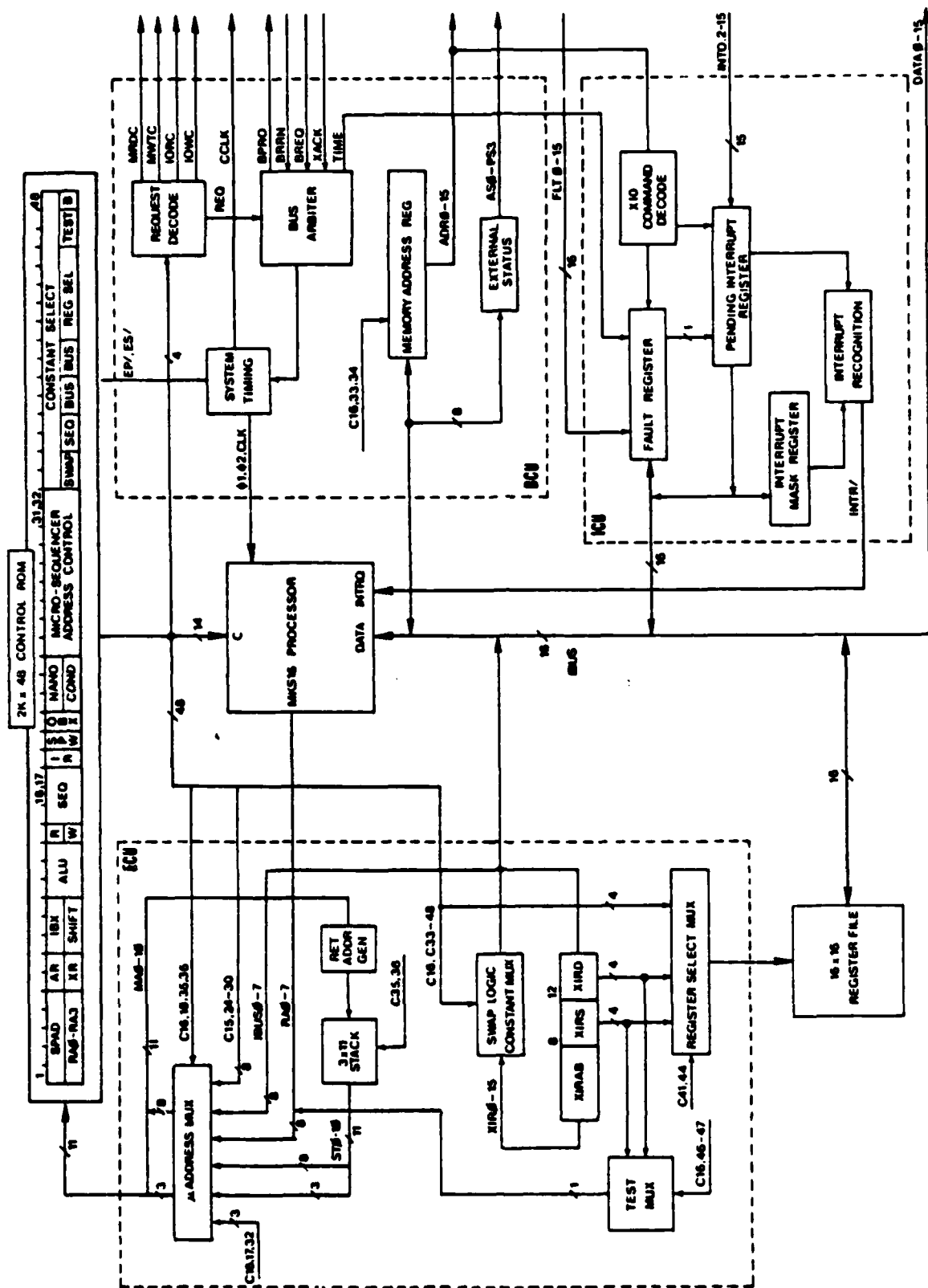


MKS16 Functional Diagram



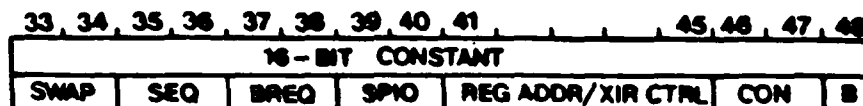
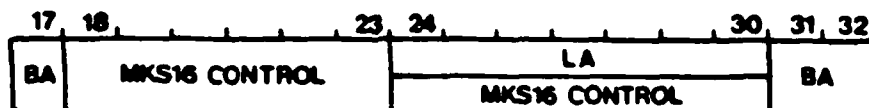
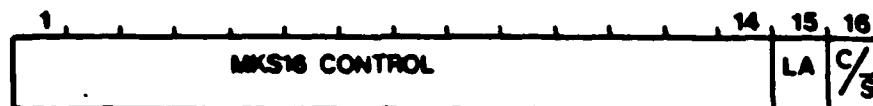
MKS16 Microinstruction Word





MKS1750 Functional Diagram

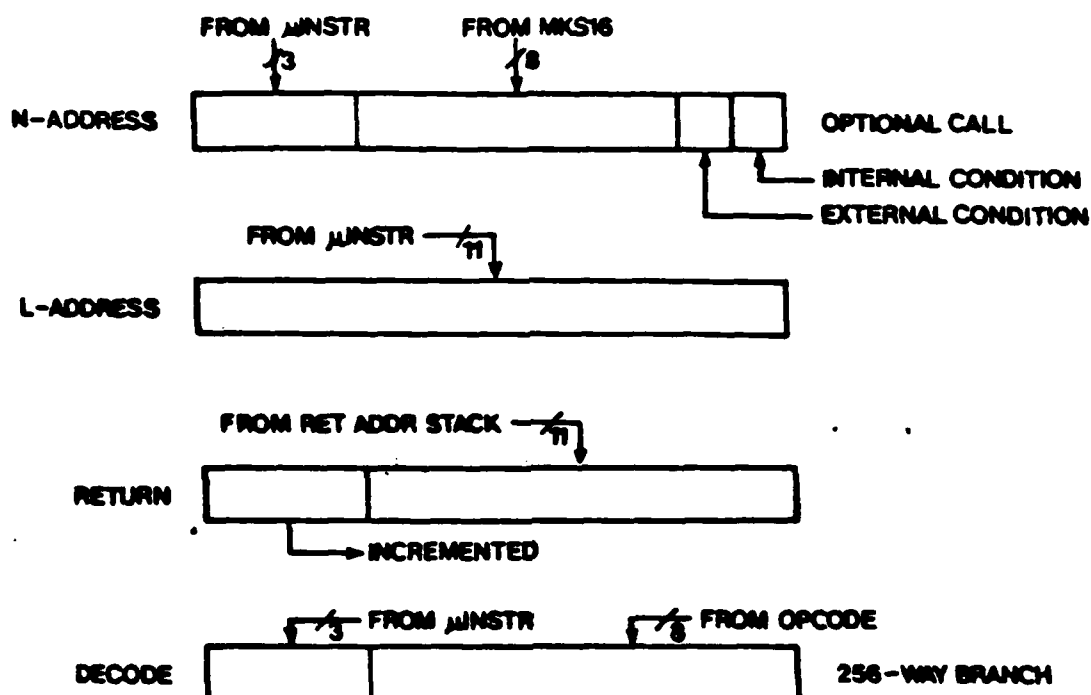
MKS1750 Microinstruction Word

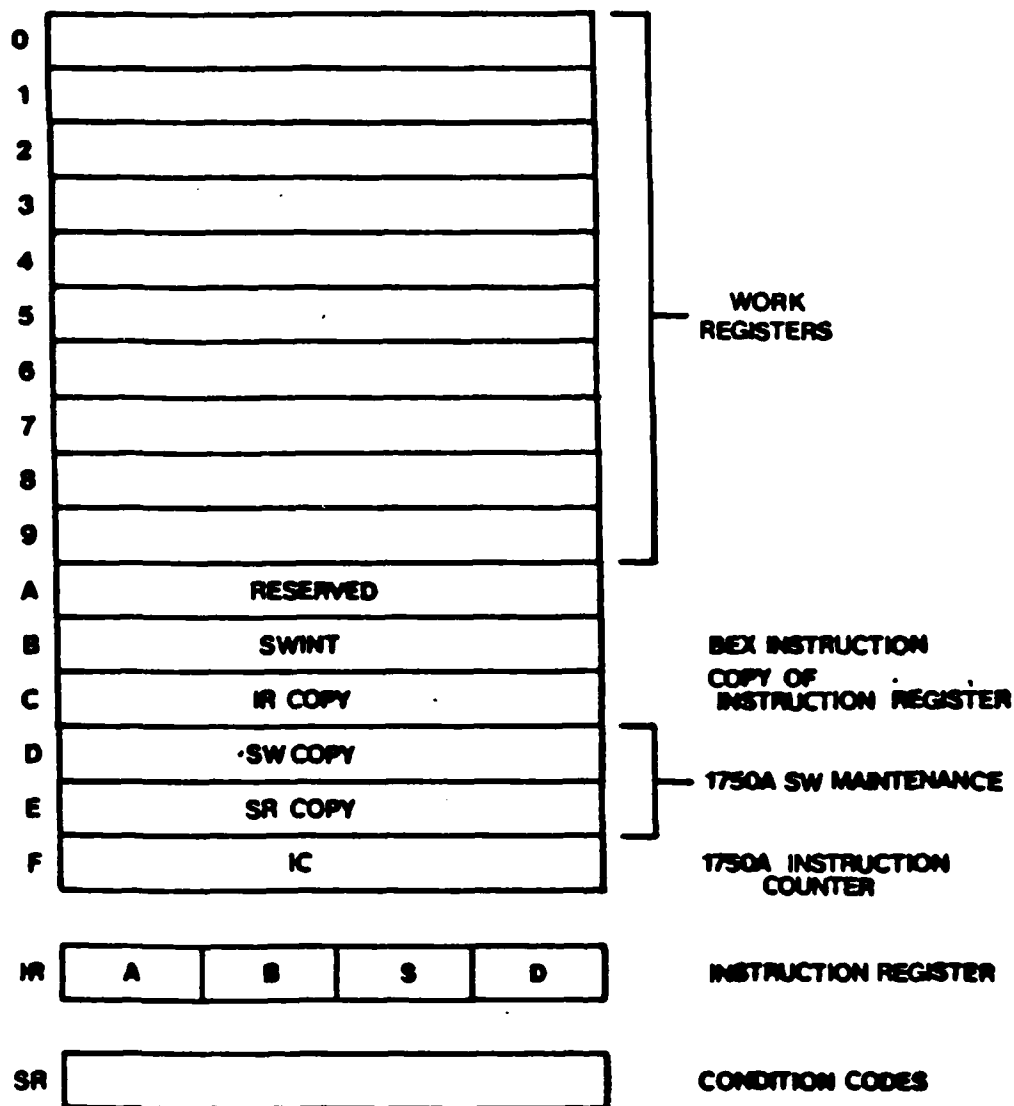


Physical Microaddress Format



MKS1750 Microaddressing

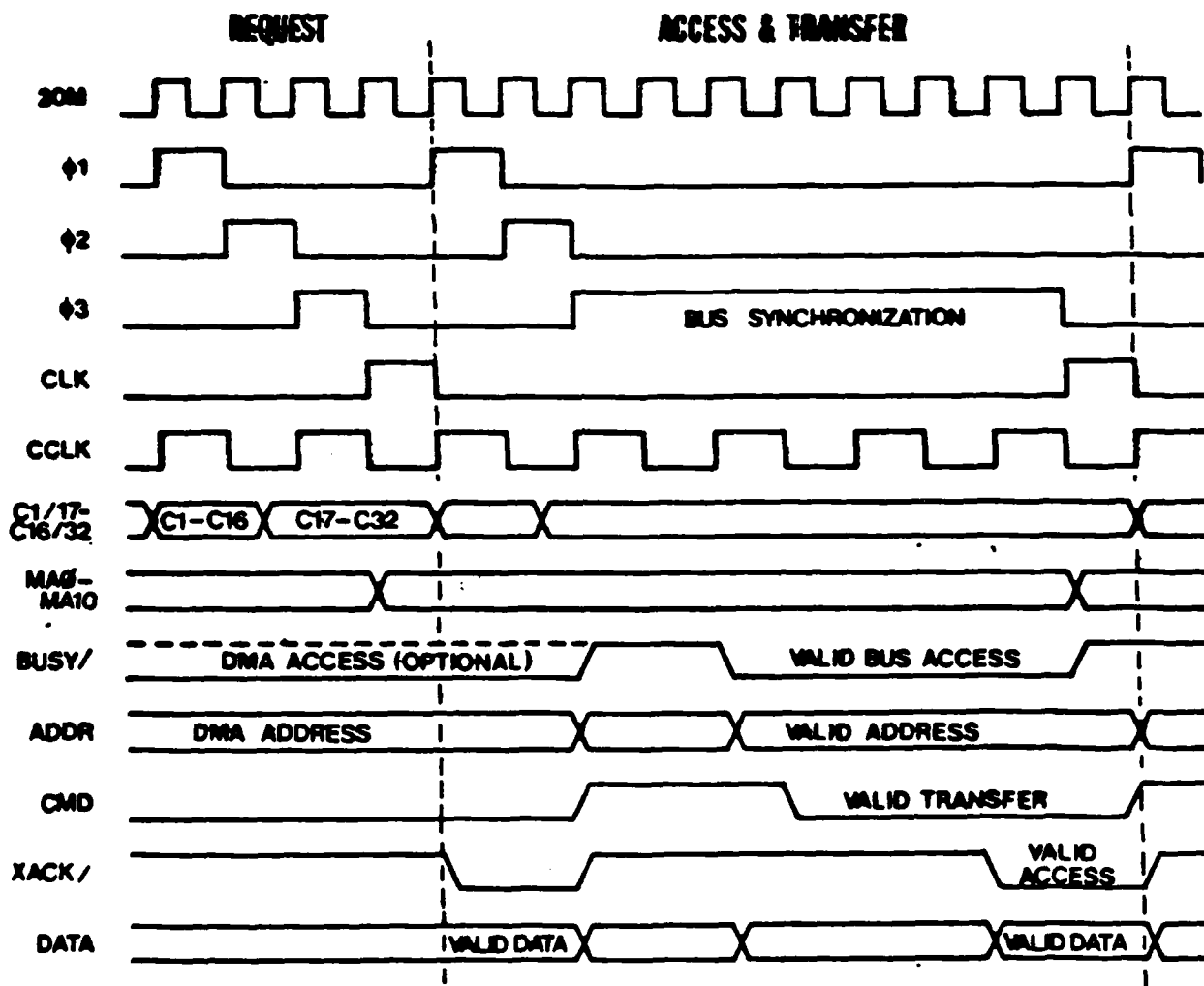




Register
Allocation



MKS1750 System Timing



TEXAS INSTRUMENTS VHSIC MIL-STD-1750A DATA PROCESSOR

**Dennis R. Best and David M. Chastain
Texas Instruments Incorporated**

ABSTRACT

Texas Instruments, as part of its very-high-speed integrated circuit (VHSIC) contract with the Army, is developing a 4-million instruction-per-second (MIPS) MIL-STD-1750A data processor. This paper describes the design goals, implementation approach, and key features of the design.

The 1750A processor is composed of four device types. A 25-MHz data processor unit (DPU) and device interface unit (DIU) implement all 1750A instructions, including floating point and approved extensions to aid operating system functions, and the interfaces to support the interrupt structure, memory extension and I/O command structure defined by MIL-STD-1750A. One or more general buffer units (GBUs) provide interfaces to other processors and/or I/O devices via a network of multimaster buses. These logic devices are supported by a 25-nanosecond 64K-bit ($8K \times 9$) static RAM to achieve an expected 4-MIPS performance on the DAIS mix.

The paper concentrates on the architectural approach to the implementation and the hardware and instruction extension features that support operating system functions, higher order languages, and maintenance and test functions and interfaces to peripheral devices.

INTRODUCTION

Advanced USAF aircraft weapon systems will have many requirements for high-speed, high-density 1750A processors. These processors must provide the required high throughput, low-cost maintenance operations, very-high reliability and, in critical applications, fault-tolerant computing capabilities of future avionics systems.

An IC chip set is being developed under the Texas Instruments VHSIC Phase 1 contract to meet these advanced 1750A needs. At the center of this chip set is a

monolithic microprocessor, the DPU, which executes the full MIL-STD-1750A instruction set plus several legally implemented extensions. The performance of this microprocessor will be in the 4- to 6-MIPS range against an integer mix, and between 2 and 4 MIPS against a DAIS mix. A second coprocessor, the DIU, supports the 1750A microprocessor by handling all interfacing to the outside world. The referenced instruction set extensions, in part, provide direct microcode support for prioritized task scheduling and other operating system and multiprocessor communications functions.

Performance is not the only driving factor guiding the development of the Texas Instruments VHSIC chips. Maintainability, testability, and fault-tolerant capabilities are also important. Built-in test features are being provided on all ICs to allow self-testing at the processor level. These features include a 4-bit maintenance port for interfacing to a serial bus that can be tied to an external maintenance controller. Fault-tolerant computing is provided for via on-chip memory error correction and a 2-bit voting port for triply redundant processor operations.

PROCESSOR DESIGN PHILOSOPHY

The processor design is primarily targeted toward small embedded signal processing applications where size, power, and throughput are the most important considerations. However, the design provides all of the necessary "hooks" to fully support a large-machine, laboratory-development type environment in such a way that the small embedded applications are not impacted. Careful studies of current embedded computer requirements produced the following design guidelines:

- 16- and 32-bit integer operations were most prevalent. However, DoD requirements for HOL programming added emphasis to the need for fast, single-precision, floating-point arithmetic.
- Special hardware support of HOL and OS functions would be required to encourage systems designers to migrate to an HOL environment.
- The majority of the size and power in microprocessor-based systems was a result of random TTL support logic, not the microprocessor device. Therefore, the microprocessor should absorb as many support functions as possible.
- The constraining performance factor would be memory access time. Therefore, the memory path must be short, well controlled, and free of all unnecessary capacitive loading.
- A memory size of 64K words would be adequate for most embedded systems. Larger memories, however, must also be supported with some reasonable penalty in performance. Memory caching for large system memories was a logical assumption.
- Direct hardware support for tightly coupled multiprocessor systems was a requirement.

- Fail-soft capability was required in some systems. The capability should be provided with minimum impact on the simple processor case, however.
- I/O operations (channel-type DMA and messages) should be supported in parallel with normal processing for increased system throughput.
- Device pinout should be restricted to a maximum of 84 pins to facilitate device fabrication, improve reliability, and decrease PWB footprint area.

The above guidelines dictated the processor partitioning and I/O techniques to be used, while MIL-STD-1750A, Notice 1, dictated the instruction set architecture and allowable extensions.

1750A PROCESSOR DESCRIPTION

The 1750A processor, illustrated in Figure 1, is composed of two primary device types: the data processing unit (DPU) and the device interface unit (DIU), supported by static random-access memories (SRAMs) and general buffer units (GBUs). The DPU and DIU perform all processing and I/O related functions. The SRAMs provide all storage for instructions and data, with the exception of the kernel operating system located in ROM on the DIU. The GBUs provide all interfaces to other processors and I/O devices via a network of multimaster buses called system buses (SBUSs).

Each 1750A processor is completely self-contained. Construction of a multiprocessor system consists of simply strapping the appropriate SBUS connections to establish the system bus configuration. Programming of a multiprocessor system is independent of the bus configuration or the number of processors in the system. This configuration information is supplied in table form to the "binder," the last step in system software configuration. The system bus configuration can also be altered dynamically to compensate for failed buses or nodes by changing a single software pointer; no restructuring of programs is required.

Special instruction extensions have been added to the processor in accordance with MIL-STD-1750A, Notice 1, to allow software to view a multiprocessor system as a single processor with a single address space.

The Data Processing Unit

The DPU is a high-performance microprocessor that conforms to the instruction set architecture established by MIL-STD-1750A, Notice 1, plus approved extensions. The processor implements all 1750A instructions, including floating-point arithmetic, and all required interfaces to support the interrupt structure, memory extension, and I/O command structure defined by MIL-STD-1750A.

The processor architecture provides autonomous, multilevel instruction lookahead as well as autonomous lookahead address development. The major arithmetic path,

DATA PROCESSOR NODE

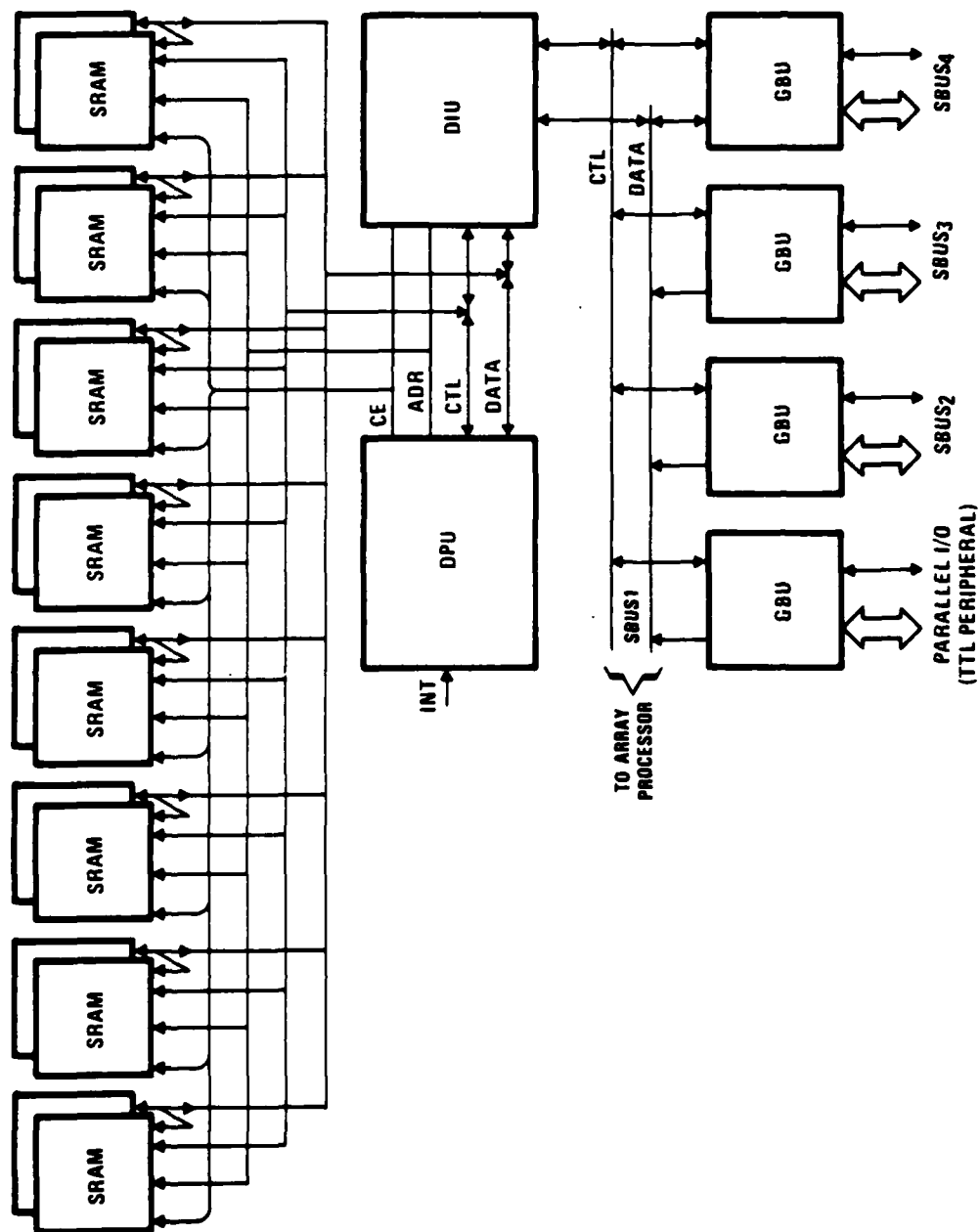


Figure 1. Data Processor Node

composed of the 16-word general register file, 10-word accumulator register file, operand select logic, 32-bit arithmetic logic unit (ALU), and 32-bit barrel shift/align/normalize logic, provides 32-bit-wide arithmetic paths for increased throughput on double-precision and floating-point operations.

Memory interfacing is simplified by built-in chip enable decoding, memory device-compatible interface signals, built-in memory error correction, and onboard support for memory extension to 1 million words.

Approved extensions, in the form of a universal user/executive/hardware procedure call, automatic hardware task-scheduling, semaphore operations, message operations, interprocessor communications, and other operating system type functions, have been added via the BIF opcode, provided by MIL-STD-1750A, Notice 1, to fully support Pascal, JOVIAL, Ada and other high-order languages. Several registers, including the pending task (PT) register, and the extended fault (EFT) register, have been added to support these instruction extensions. A special configuration register (CFR) has also been added to guarantee object code compatibility with user code generated per the basic MIL-STD-1750A specification, without the new extensions. These registers, and an extended definition of the status register, are defined in Table 1.

Interfaces are provided for maintenance and test, and for a unique, triply redundant, voting technique to fully support high-reliability systems requirements. A four-line interface, called the system maintenance bus (SMBUS), provides self-test initiation/reporting, conventional "front panel" functions such as examine/modify registers, halt/step/break-point, etc., and error reporting for the voting logic. The two-line voting bus allows two processing units (two DUPs and two DIUs) to be strapped pin-for-pin as "worker" and "watcher" to provide clock-by-clock voting and automatic shutdown on any discrepancy. Three processing units (three DPU's and three DIUs) designated as "worker," "watcher," and "surrogate" may also be strapped pin-for-pin for "fail-soft" operation wherein the watcher and surrogate monitor and vote on the operation of the worker on a clock-by-clock basis. In this mode, the surrogate will override and replace the worker on any discrepancy while the watcher continues to monitor the surrogate.

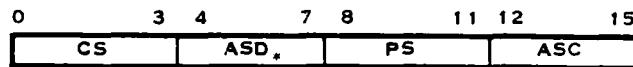
The Device Interface Unit

The DIU is a special-purpose co-processor designed as a companion for the data processing unit to provide all outside interfaces other than memory. The DIU performs programmable I/O, channel I/O, message transactions, interprocessor interlock (semaphore) operations, and other functions in parallel with the data processing unit.

Interfaces to peripherals and other processors are provided by the DIU's system bus (SBUS), a high-speed, multiuser, synchronous bus that supports I/O commands, messages, and bulk data transfers at a sustained rate of one word per clock. The majority of the DIU is dedicated to the support of SBUS communication.

Table 1. Register Extensions/Additions

EXTENDED STATUS WORD



* SPARE IN NATIVE 1750A MODE, SEPARATE ADDRESS AND DATA STATES IN TASK CONTROL BLOCK AND CALLS DATA STRUCTURES ONLY FOR USE WITH INSTRUCTION EXTENSIONS.

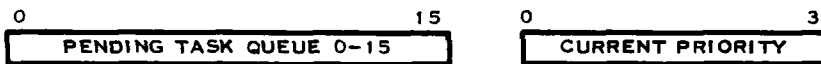
EXTENDED FAULT REGISTER



DEVICE ERRORS, SEMAPHORES/STACK OVERFLOWS AND SOFTWARE TIME-OUT

ERROR-CORRECTED MEMORY FAULTS

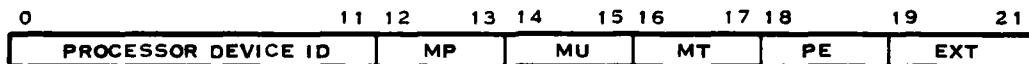
PENDING TASK REGISTER/CURRENT PRIORITY REGISTER



PRIORITY OF EXECUTING TASK

ACTIVE TASK IN PENDING TASK QUEUES 0-15
QUEUE 0 = HIGHEST PRIORITY

CONFIGURATION REGISTER



EXTENDED INSTRUCTION ENABLES

DATA FLOW ENABLE

MAPPING MODE

MEMORY INITIALIZATION SEQUENCE

MEMORY PARITY ORGANIZATION

PROCESSOR ID (FOR MULTIPROCESSOR COMMUNICATIONS)

Functionally, the DIU operates as a microcoded interpreter that is driven by descriptors from a series of linked-list queue structures in memory. These descriptors are "cached" in the channel control and address register files during interpretation. The actual interpreter is composed of the main ROM controller, the channel status register (CST) that maintains queue activity, and a limited function 16-bit ALU.

The memory data and address paths and associated registers are identical to those in the DPU, providing a simple memory interface with memory error correction. A second data path is provided for FIFO buffering of SBUS communications by a separate autonomous controller.

The DIU also provides three programmable timers, per the MIL-STD-1750A specification, and a 2K by 16 ROM that contains a complete kernel operating system, sufficient for small embedded computer systems, plus bootstrap loaders and self-test routines. The primitives provided by this operating system are universally applicable across all languages. Furthermore, the various routines are accessed indirectly through a tables structure in SRAM, allowing any or all of the routines to be replaced by user-written routines. Additional entries have been provided in the table structures to allow extension of the operating system for more general-purpose central avionics-type applications and for laboratory-type multiuser, virtual-memory type environments.

The DIU also supports the system maintenance bus as a slave to the data processing unit to facilitate node level maintenance as well as the triply redundant voting bus for high reliability system applications.

The General Buffer Unit

The GBU provides communication between array processors, data processors, and peripheral devices within a network of processor nodes. The network can be configured with several bus levels, each of which is a multimaster bus with a separate, possibly asynchronous, clock. Each level in the network can support up to 16 master devices; also, additional slave devices and multiple redundant paths may be provided between devices for increased bandwidth and/or fault tolerance. The GBU operates in one of two major modes (coupler or PIO) with many minor modes, all of which are software programmable.

In the coupler mode, the GBU waits for commands or messages on one bus, and if a bus relay is required, vies and becomes master on the other bus to relay the command or message to the next level device. Once a path has been established between an originating device and its target device, information is transferred as a stream of 16-bit words. The transfer of information is buffered by a two-level FIFO at each bus level, creating the equivalent of a single FIFO buffer of arbitrary length between the originating device and the target device. This technique allows information to be transferred at a rate of one 16-bit word per bus clock at the clock rate of the slowest bus clock encountered in the selected path.

Each bus transaction is either a command or a message. All transactions begin with a 16-bit header word that specifies the type of transfer (input or output, command or message, etc.) and an 8-bit path name that uniquely specifies the network path from the originator to the target device. This header word is followed by a variable-length data stream whose length and content are dependent on the header-type fields.

In PIO mode, the GBU interfaces the SBUS to a simple 16-bit parallel I/O device. Each bit may be independently programmed as input or output to support 16-bit single-word transfers or discrete control bit requirements. An "interrupt" capability is also provided so that the GBU can issue a device service request message to any system processor in response to a change in one of the input discretes.

The GBU also supports the SMBUS for self-test and maintenance operations, in addition to the normal operational interfaces.

Static RAM

The SRAM is an 8K by 9 static NMOS memory with special peripheral logic to support the processor devices. The SRAM provides the write-protect bit storage and protection logic necessary to implement the block write-protect option specified by MIL-STD-1750A. Each 1K block of memory can be individually protected by software I/O commands so that stored data cannot be modified. Any attempt to modify protected data will result in a processor interrupt.

The SRAM also provides simple byte parity generation and checking. Or, the data processor may utilize all 18-bits (two SRAMs wide) with parity generation and checking performed by the DPU and DIU devices to achieve maximum throughput and to provide additional software fault-isolation capabilities. Alternatively, three SRAMs may be used to provide a 22-bit memory with full single-correct/double-detect error correction provided by both the DPU and DIU.

All address and control signals are pipelined by registers on the SRAM device so that only one bus propagation delay is added to the base memory access time in determining the maximum processor clock rate. The write pulse is internally generated, freeing the processor (or system designer) from the burden of generating precise timing strobes and allowing the SRAM to be accessed as one would normally access a register file.

PROCESSOR OPERATION

Task Scheduling

The DPU maintains 16 prioritized, linked-list queues of tasks to be executed. Tasks are automatically scheduled by the DPU during the execution of certain OS support instruction extensions, such as semaphore instructions. This mechanism eliminates the majority of the overhead normally associated with OS calls for task interlocks and task

scheduling, allowing separately compiled tasks to execute and communicate with an efficiency similar to a single, larger task. The DPU also supports special user-task and executive-task procedure call mechanisms that eliminate the majority of the overhead normally associated with these functions. This allows compiler-generated code to execute with an efficiency similar to hand-generated assembly language. These features provide the functionality and efficiency necessary to encourage the system designer to migrate to a high-level language environment.

The majority of the communications between the DPU and DIU are accomplished through manipulation of linked-list queue structures in memory. This technique provides FIFO buffering of communications, releasing the DPU to continue processing on the same task or another task while the DIU services a channel I/O request, message request, semaphore operation, etc., to promote concurrent DPU and DIU operation and optimize the use of available memory bandwidth. Tasks are automatically suspended during such operations and automatically rescheduled upon completion of the operation.

I/O Operations

The 1750A supports five basic modes of device interfacing: (1) an intelligent device may be interfaced as an SBUS master/slave, or (2) an SBUS slave only, (3) a simple I/O device may be interfaced to a GBU (operating in PIO mode) on the SBUS, (4) a "special" device may be interfaced directly to the DIU's SBUS using reserved 1750A I/O commands that are not recognized by the GBUs, (5) I/O devices may be interfaced as "memory-mapped" functions, although this type of interfacing is discouraged because of the additional capacitive loading on the memory bus.

All SBUS devices, whether master/slave, slave only, or PIO, interface to the SBUS through one or more levels of GBUs that provide for the routing of commands or messages and synchronization of different clock rates. Up to 254 devices may be accessed by a given processor with any given programming of the GBU bounds registers. This restriction applies only to a single processor, not to a multiprocessor system. Likewise, a processor can dynamically reconfigure the bound registers of the GBUs in a system in order to reach other devices in the system. This interfacing method is the preferred method since it provides the most flexibility, fault tolerance, and bandwidth (for multiword transfers).

The special I/O commands can only be used to access devices attached directly to the DIU's SBUS. These commands execute with less overhead than the normal SBUS commands (PICn, POCn) because no priority vies or routing comparisons are required. The special commands are supported by the Pascal compiler and other support software in the same manner as the normal SBUS commands and are covered by the same protection mechanism.

The system designer is advised to provide external data concentration circuitry, or buffer memories, for sensors that input data sporadically in small quantities. These

buffer memories can then be transferred into the 1750A processor using a high-bandwidth channel transfer to use the available memory bandwidth to best advantage.

Multiprocessor Support

The 1750A has been extended to encourage the use of multiple-processor systems, in lieu of single, special-purpose processors, and to promote standard, modular hardware and software for future system designs. Processor modules can be connected as a multiprocessor system simply by connecting one or more SBUS ports from each processor in parallel. Special instruction extensions have been added to the processor to (1) forward results or operands from a task in one processor to a task in a different address state of a different processor, (2) to interlock the execution of tasks within a processor or between processors, (3) to send messages between the ROM-based operating systems of different processors, and (4) to transfer blocks of data between the local memories of different processors, and other operations necessary to allow the system software to view a multiprocessor system as a single entity, thereby simplifying the design and programming of a multiprocessor system.

INSTRUCTION EXTENSIONS

The 1750A data processor incorporates several architectural extensions to minimize operating system overhead, promote the use of high-level languages, and facilitate the design and programming of multiprocessor systems. These extensions are embodied in a set of new instructions added via the BIF opcode-approved MIL-STD-1750A, Notice 1.

All extended operations instructions follow a single format that is compiler-generated as an in-line procedure/function call. The extended operation instructions consist of three basic classes: noninterrupt task procedure calls, executive procedure calls, and hardware-implemented function calls. This singular format for all compiler-generated procedure calls simplifies code compilation and significantly increases execution efficiency. All parameters for hardware function calls or executive calls and the first six words of parameters for user task calls are passed through dedicated registers consistent with normal compiler usage.

User task procedure calls, or CALL\$ instructions with operation codes 4F00-4F7F, have a procedure number, ranging from 0 to 127, that is used as an index into the Procedure Entry Table (PET) specified by a pointer in the Task Control Block of the active task.

The CALL\$ saves the instruction counter and general registers of the calling procedure on the process stack, allocates a new stack frame for the calling procedure, adjusts the map registers if necessary, and begins execution of the called procedure. The CALL\$ does not attempt to pass parameters (other than those passed in the general registers), allocate local variables, perform type checking, or any other function that might be language-dependent.

The remaining extended operation codes (4F80-4FFF) are shared by the executive call instruction and hardware-implemented functions. Hardware-implemented functions are assigned opcodes in descending order, beginning with opcode 4FFE with opcode 4FFF reserved as an escape to a two-word instruction format or to a coprocessor-implemented function. Executive call functions, called XCALLs, are assigned in ascending order, beginning at opcode 4F80. This scheme allows maximum flexibility in transitioning operating system functions into hardware as device technology advances.

When the processor encounters an opcode in the range 4F80-4FFF, it first checks to see if the function is implemented in hardware. If not, the seven LSBs of the opcode are used as an index into the XCALL Entry Table (XET), which is referenced through a pointer in dedicated memory. The executive task is scheduled on the Pending Task Queue selected by the priority field in the selected Task Control Block, just as any other task is scheduled.

The hardware-implemented functions currently defined are grouped into six major categories: (1) function returns that pass results back from user functions, (2) data forwarding instructions that allow an operand to be stored directly into the address space of another task in any address state of any processor and the SYNC instruction that performs the data flow update operation when the processor is in data flow mode, (3) semaphore instructions that can reference a semaphore in any address state of any processor, (4) queue manipulation instructions that can operate on queue structures in any address state of a processor, (5) CORDIC primitive instructions that allow efficient calculation of all transcendental functions with user selection of number base and user optimization of accuracy versus execution time, and (6) a group of miscellaneous functions for moving data across address state boundaries, tagging memory for software error detection, changing task priorities, and initiating software messages. The registers added to support the instruction extensions are detailed in Table 1 and the hardware-implemented instructions are described in Table 2.

This paper is intended only to give the reader a preview of the characteristics of Texas Instruments VHSIC 1750A processor; a complete, user-oriented specification will be released upon completion of the processor design validation effort.

Table 2. Special Procedure Instructions

Class	Mnemonics	Op Codes		Operation
		OC	OCX	
Special Procedure	RETS	4F	FE	Return with no result
	RETSI	4F	FD	Return with integer result
	RETS D	4F	FC	Return with double integer or floating result
	RETSE	4F	FB	Return with extended floating result
	CALLSF	4F	FA	Call with ID in register
	GETSID	4F	F9	Get re-time ID of function
Data Forwarding	FWD	4F	F6	Forward integer
	DFWD	4F	F5	Forward double-precision integer or floating point
	EFWD	4F	F4	Forward extended-precision floating point
	SYNC	4F	F7	Forward synchronization signal
Semaphore Instructions	V	4F	F0	Signal semaphore
	P	4F	F1	Wait on semaphore
	SP	4F	E0	Surrogate wait on semaphore
	CP	4F	F3	Conditional wait on semaphore
	CV	4F	F2	Conditional signal semaphore
Queue Manipulation Instructions	AQ	4F	EC	Add to tail of queue
	IQ	4F	ED	Insert after old queue entry
	RQ	4F	EE	Remove from head of queue
	DQ	4F	EF	Delete after old queue entry
	CQ	4F	E1	Concatenate queues
CORDIC Instructions	TRIG	4F	E3	Sine, cosine, tangent
	ISIN	4F	E4	Arc-sine, arc-cosine
	ITAN	4F	E5	Arctangent, rectangular to polar
	HYPE	4F	E6	Hyperbolic sine, cosine, tangent and exponential
	IHYP	4F	E7	Inverse hyperbolic tangent, log
Miscellaneous Instructions	EXIT	4F	F8	Exit task
	EPRI	4F	EA	Equate priority of task
	ARI	4F	EB	Assign priority to task
	SND	4F	E2	Send message
	RMOV	4F	E8	Remote move
	TAGM	4F	E9	Tag memory

TESTABILITY SUPPORT SYSTEM FOR 1750A:

IN-CIRCUIT FAULT SIMULATOR

Constantin Timoc

Jet Propulsion Laboratory

M/S T-1180

4800 Oak Grove Drive

Pasadena, CA 91109

(213) 354-4588/5751

ABSTRACT

Testability of avionics has a direct impact on the cost of equipment maintenance. The use of LSI/VLSI components in avionics has many advantages in terms of reliability, power consumption, performance, etc. Testing the hardware, however, is difficult because the thousands of gates packaged in the LSI/VLSI chips are physically inaccessible. One solution to this problem is to provide new support equipment that helps avionics designers and maintenance personnel to develop powerful test and maintenance programs capable of detecting and locating defective components in a cost effective manner.

A system comprising microprocessors cannot be easily designed unless support hardware and software such as development systems, in-circuit emulators, programming languages, operating systems are readily available. One area in which there is presently very little support, is testability. As a result, this paper presents a proposal for a support system called in-circuit fault simulator (ICFS).

An in-circuit fault simulator is a piece of equipment capable of simulating technology dependent faults in a microprocessor in addition to performing the function of a fault-free microprocessor. It is used as support equipment in the process of developing test and maintenance software programs.

The paper presents the design, implementation, and utilization of an ICFS at Jet Propulsion Laboratory for a microprocessor used in spacecraft. The ICFS was first used to develop test vectors, for qualifying the microprocessor for Class S applications. The test vectors developed with the help of the hardware fault simulator are running on a commercial tester and are detecting 99.7% of the faults in the microprocessor. The second utilization of the ICFS was in the development of in-flight self-test software to detect and locate the faults in the flight hardware.

In summary, this paper describes a support system, called in-circuit fault simulator, for developing test and maintenance software for avionics systems incorporating 1750A microprocessors.

TESTABILITY SUPPORT EQUIPMENT FOR MIL-STD-1750A:

IN-CIRCUIT FAULT SIMULATOR

Constantin Timoc
M/S T-1180

Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
(213)354-4588/354-5751

ABSTRACT

The in-circuit fault simulator (ICFS) described in this paper is a piece of support equipment used to measure the fault coverage of test programs at component, board, and system levels. An ICFS performs in real-time the same function as a software fault simulator running on large computers. The ICFS is portable, easy to operate, and can be used to simulate faults in mixed analog and digital boards and systems.

1. INTRODUCTION

The use of LSI/VLSI components in avionics has many advantages in terms of reliability, power consumption, and performance. Testing the hardware, however, is difficult because the thousands of gates in the LSI/VLSI devices are inaccessible for probing. A solution to this problem is to develop test programs (manually or automatically) and to measure their effectiveness to detecting faults with a software fault simulator. In practice, it has been found that software fault simulators have limitations in performing this function for boards and systems comprising LSI/VLSI components. Moreover, software fault simulation of mixed analog and digital boards or systems is presently impractical. In order to solve the problem of fault simulation for a spacecraft system comprising microprocessors, an in-circuit fault simulator (ICFS) was developed and used under contract from JPL/NASA.

2. A HARDWARE FAULT SIMULATOR FOR A GATE

A hardware fault simulator for an AND gate with primary inputs X and Y is shown in Figure 1. The only faults necessary to be injected are input X stuck-at-one (X-1), input Y stuck-at-one (Y-1), and output Z stuck-at-zero (Z-0). To simulate the behavior of a fault-free two-input AND gate, the serial-in and parallel-out auxiliary shift register should contain all "0"s in its three shift register latches (SRL1 through SRL3). To simulate X-1, a "1" is stored in SRL1 and "0"s are stored in SRL2 and SRL3. As can be seen, a "1" from SRL1 will permanently force a stuck-at-one at input X. In the same manner, Y-1 and Z-0 can be simulated by shifting a "1" preceded and followed by "0"s in the appropriate SRL as shown at the bottom of Figure 1. An implementation in CMOS of such a fault simulator for a gate, called microsimulator, is depicted in Figure 2.

3. IMPLEMENTATION OF AN IN-CIRCUIT FAULT SIMULATOR (ICFS) FOR A MICROPROCESSOR

The implementation an ICFS for the RCA 1802 microprocessor was accomplished in two phases. In the first phase, a breadboard that performs the fault-free function of the RCA 1802 was built using standard CMOS chips. The registers and the register array were implemented with MSI chips while the rest of the microprocessor logic was implemented using only SSI chips. The fault simulation capability was not provided in this phase in order to simplify the implementation of a correct fault-free system. In the second phase, automatic fault injection capabilities were added to the fault-free breadboard. This was accomplished by

replacing each SSI chip with its corresponding microsimulator. When all the gates were replaced by the appropriate microsimulator, then the auxiliary shift registers of the microsimulators were cascaded, thus forming a long shift register comprising 1078 SRLs (Figure 3). Therefore, by storing a "1" in the appropriate position of the long shift register, a particular gate was selected and the appropriate fault was simulated. The registers and the register array were tested with a checkboard test pattern; therefore, no faults were injected in them.

4. DEVELOPMENT AND OPTIMIZATION OF TEST VECTORS USING AN ICFS

A test set comprising approximately 13,000 vectors was first developed on a tester based only on the functional specification of the microprocessor. Fault simulation was performed with the hardware fault simulator plugged into the tester and the fault detection effectiveness of the functional vectors was measured at 95%. At the time of writing, only stuck-at faults were simulated, however, work is in progress for simulating stuck-open and bridging faults. Additional test vectors were developed for the faults not detected by the functional vectors, based on the structural information provided by the logic diagram of the microprocessor. From a total of 1078 faults, eight were found to be undetectable due to logic redundancies or test sequences that are impossible to generate. Three more faults may actually be undetectable; however, owing to the difficulty of determining whether or not they are truly undetectable, the worst case was considered, and thus they were classified as undetected. The frequency distribution of the final fault simulation is shown in Figure 4. On the horizontal axis are listed the pattern numbers, while on the vertical axis is the occurrence of detected faults. One test pattern contains about 1024 vectors. Most of the faults are detected for the first time in pattern 0, while patterns 4 and 7 detect very few faults in the sequential and combinational logic; however, those two patterns are directed toward testing the registers and the register array.

In summary, the final test set contains 14,043 vectors that detect 99.7% of the 1070 detectable single stuck-at faults injected in the combinational and sequential logic of the microprocessor. The registers and register array are fully tested for single stuck-at faults by applying checkerboard type test vectors.

5. DEVELOPMENT OF SYSTEM TEST AND DIAGNOSIS PROGRAMS WITH AN ICFS

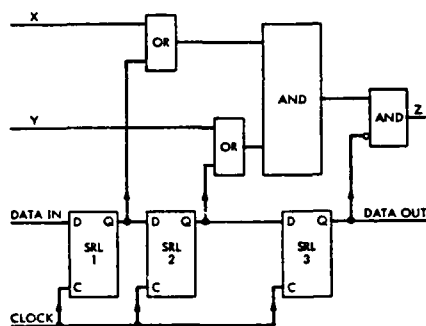
Test and diagnosis programs for boards and systems are developed to detect and locate faults in the hardware. A test program should detect a large percentage of the faults in a very short time. An ICFS is used to measure the fault coverage of a test program by simply replacing a microprocessor device from a system. Faults are being injected automatically and the test program is run on the system to find whether or not they are detected. Finally, the fault coverage of the test program is calculated by dividing the number of faults detected by the total number of faults injected. Such a test program was developed at JPL with the 1802 ICFS for a spacecraft. The usage of the ICFS made possible the optimization of the test program - maximum fault coverage with minimum test instructions.

Fault location in the system is also facilitated by the ICFS. During the fault simulation process, a cross reference between the test program and the faults detected is generated. This information is used to isolate a defective component either by running only the diagnosis program or by combining it with a guided probe.

Experimental results with the 1802 ICFS show that a test and diagnosis program detecting and locating more than 90% of the faults in a spacecraft system resides in main storage requiring about 800 bytes.

6. CONCLUSIONS

An ICFS for a microprocessor is a cost-effective testability support piece of equipment; its feasibility was demonstrated at JPL/NASA for the RCA 1802 microprocessor. An ICFS for the 1750A chip, under development by Fairchild, can be easily implemented with already available microsimulators and made available to the contractors building systems based on this microprocessor chip.



Pat. 4,308,616

FUNCTION	SRL 1	SRL 2	SRL 3
GOOD MACHINE	0	0	0
X=1	1	0	0
Y=1	0	1	0
Z=0	0	0	1

Fig. 1. A microsimulator for a 2-input AND gate.

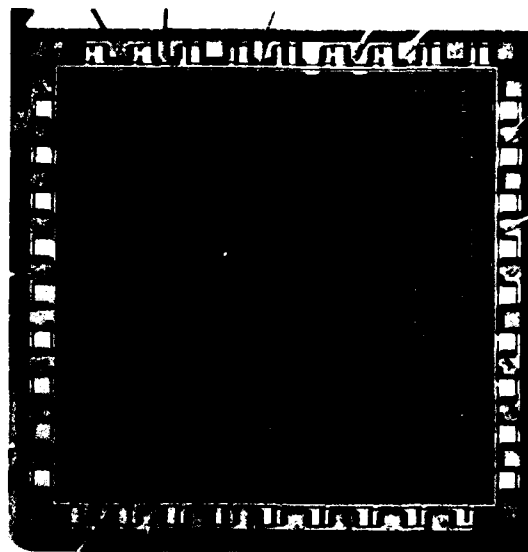


Fig. 2. The implementation of a microsimulator in a CMOS masterslice.

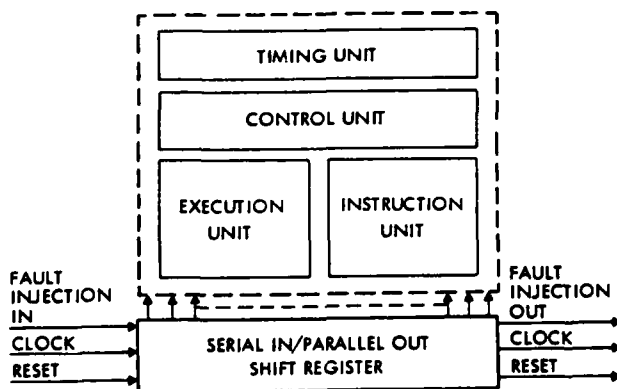


Fig. 3. Block diagram of a physical fault simulator of a microprocessor.

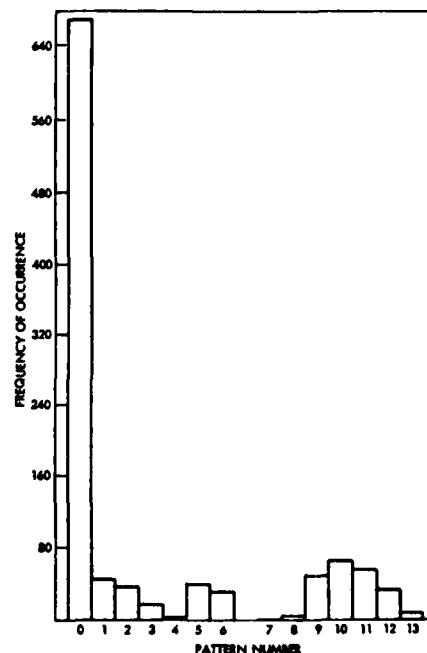


Fig. 4. Frequency distribution of the faults detected vs. pattern number for the test vectors developed for the RCA 1802.

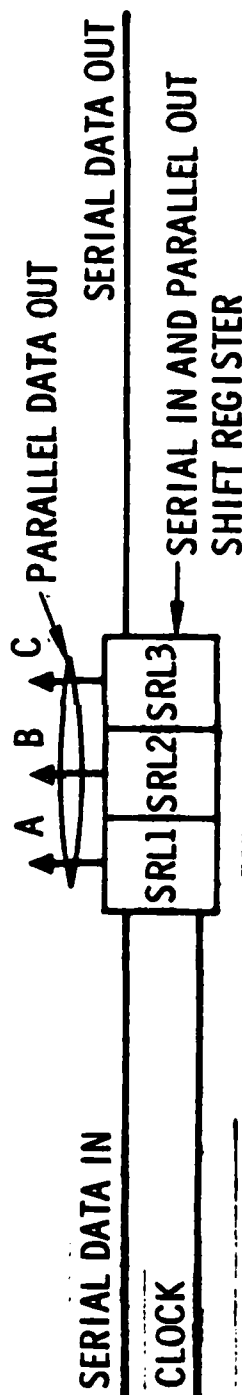
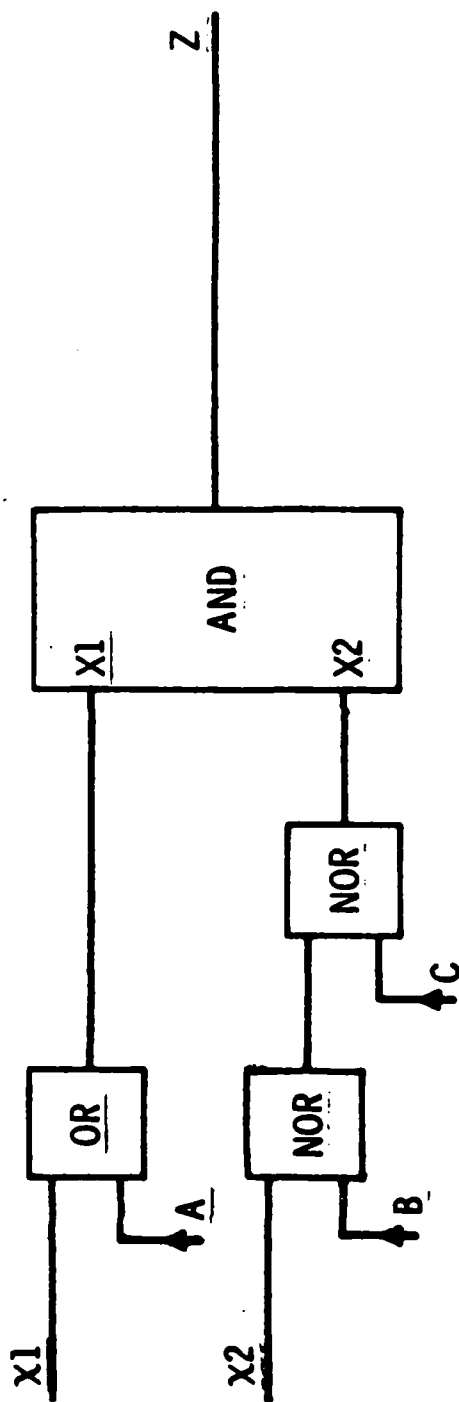
IN-CIRCUIT FAULT SIMULATION

Constantin C. Timoc

OBJECTIVES

- DEVELOPMENT OF TEST VECTORS
- VERIFICATION OF TEST VECTORS
- FAULT DETECTION EFFICIENCY (FDE) MEASUREMENT OF TEST VECTORS
- GENERATION OF FAULT DICTIONARIES

HARDWARE FAULT SIMULATOR FOR AN AND GATE

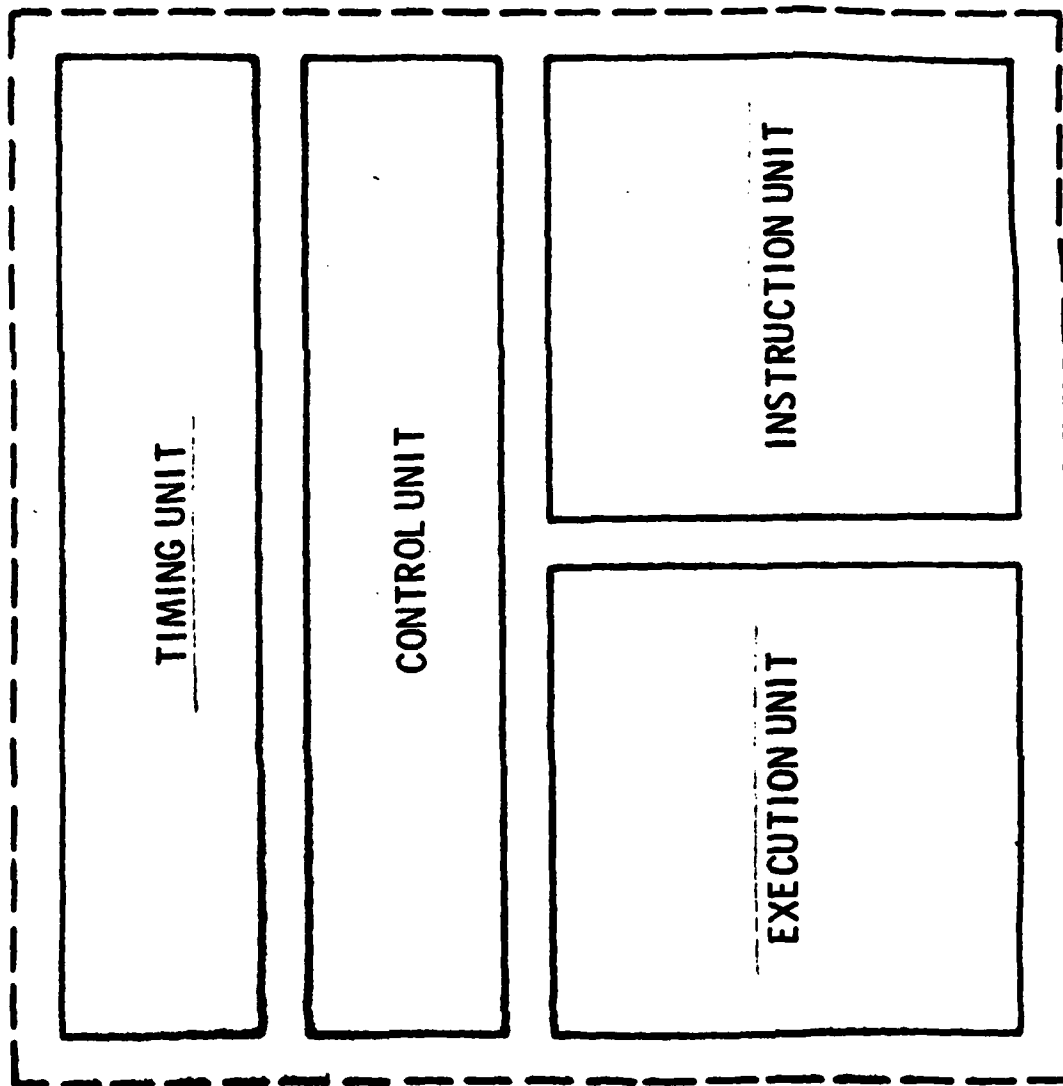


GOOD MACHINE	0	0	0
X1-1 (Z-1)	1	0	0
X2-1 (Z-1)	0	1	0
X2-0 (Z-0)	0	0	1

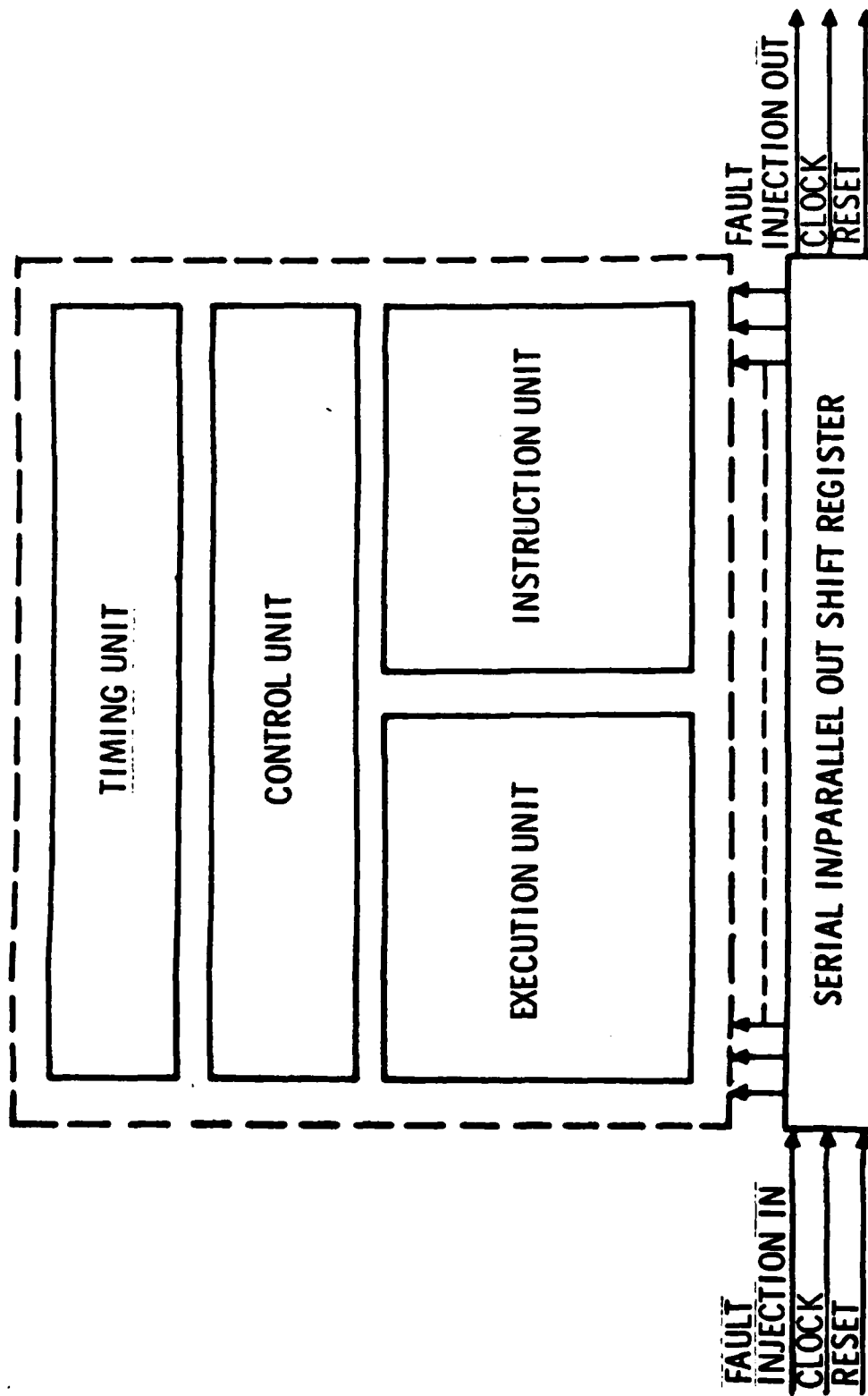
SRL (SHIFT REGISTER LATCH)

PATENT
PENDING

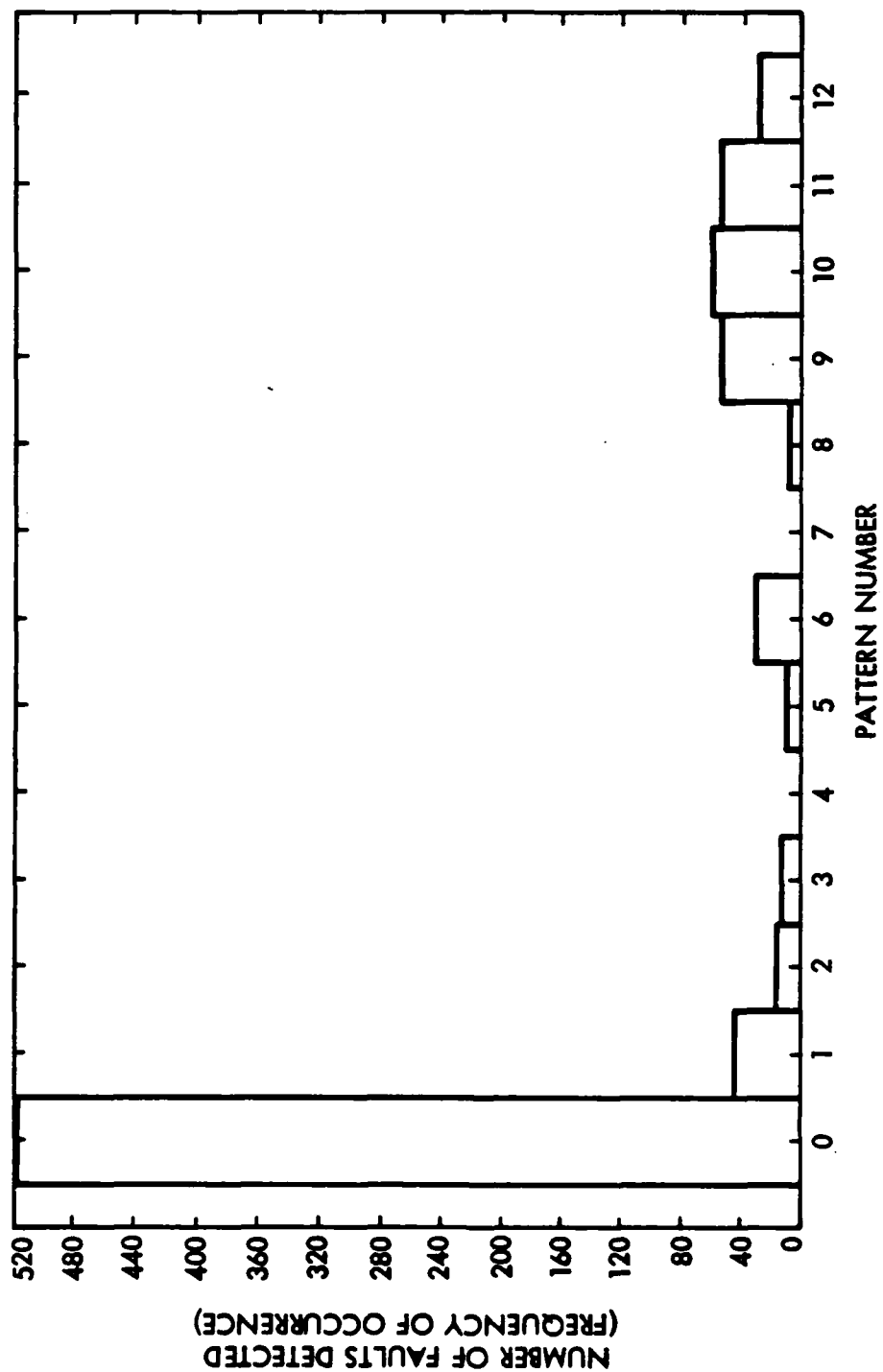
MICROPROCESSOR BREADBOARD



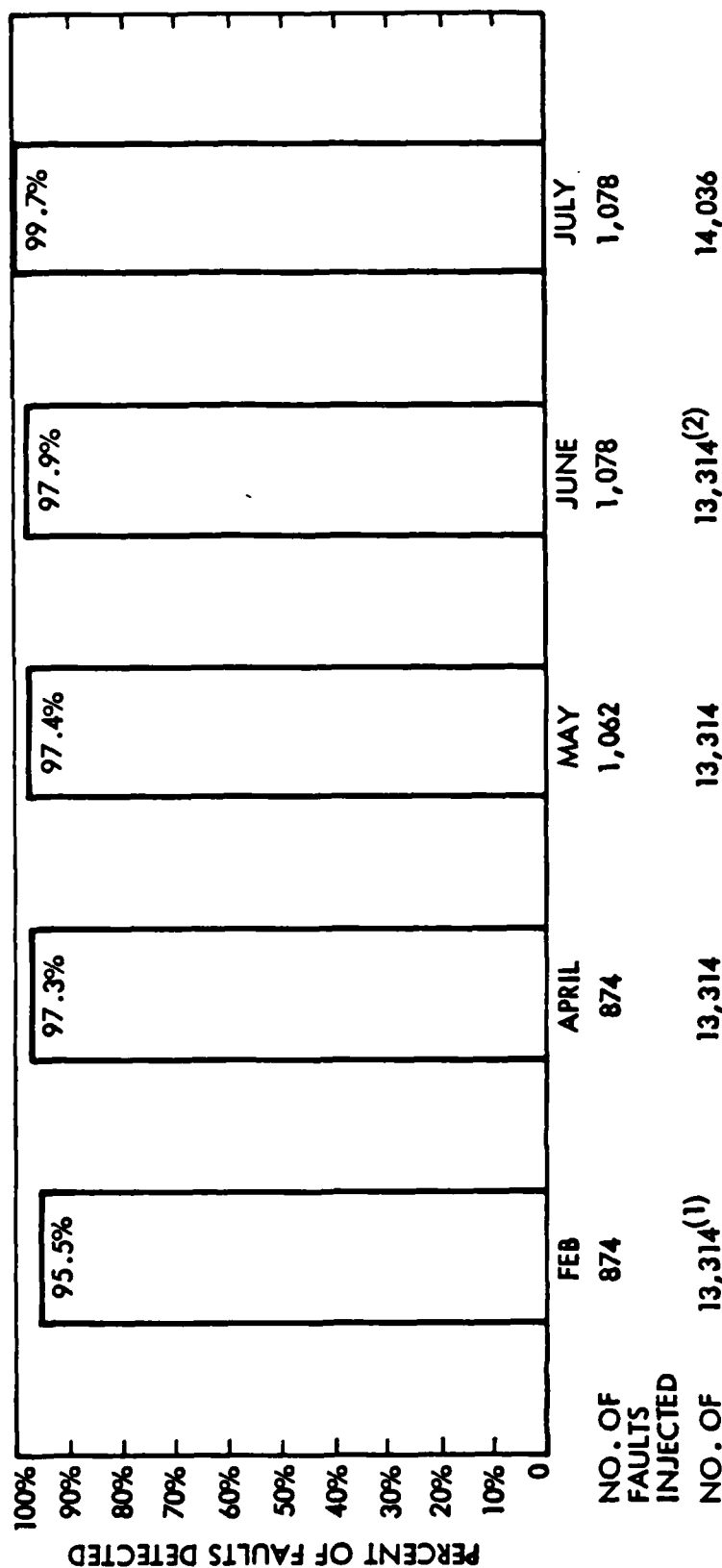
HARDWARE FAULT SIMULATOR FOR A MICROPROCESSOR



FREQUENCY DISTRIBUTION OF DETECTED FAULTS



SUMMARY OF MEASUREMENTS

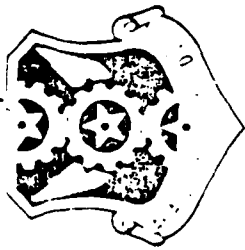
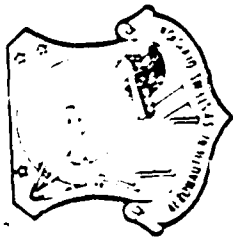


NOTES: (1) IN FEBRUARY, THE TESTER CONTROL SOFTWARE WAS MASKING SOME OF THE OUTPUTS

(2) IN JUNE, SOME OF THE ORIGINAL TEST VECTORS WERE MODIFIED

SUMMARY OF HARDWARE FAULT SIMULATION

- FEATURES
 - SIMPLE TO IMPLEMENT FOR ANY MICROPROCESSOR
 - AID IN DEVELOPMENT OF TEST VECTORS
 - VERIFICATION AND MEASURE SFDE OF TEST VECTORS
 - AID IN GENERATING FAULT DICTIONARIES
- ADVANTAGES
 - HIGH SPEED FAULT SIMULATION
 - AVAILABLE TO THE USER IN REAL TIME
 - SIMPLE TO DEVELOP AND OPERATE
 - PORTABLE
 - LOW COST



EMBEDDED COMPUTER STANDARDIZATION PROGRAM OFFICE (ECSP0)

**LT COL JIM RILEY,
DIRECTOR**

**MAJ. DAN BURTON
DEPUTY DIRECTOR**

**(ASD-ALD/AXS AV 785-5945 OR (513) 255-5945
WRIGHT-PATTERSON AFB, OH 45433**

EMBEDDED COMPUTER STANDARDIZATION PROGRAM OFFICE

- Mission of ECSPPO

— Background to formation of ECSPPO

- Air Force forms standards J73 and MIL-STD-1750
- Compilers and tools developed by Labs: RADC and AFWAL
- Air Force not sure how maintenance should be handled
- General Slay directs study on issue
- Study recommends AF establish a maintenance organization

— ECSPPO established in late CY80 and chartered to:

- Support JOVIAL Compilers and MIL-STD-1750A support software
 - Consult with users that have problems
 - Provide users with current error status reports
 - Correct errors discovered through user problem reports
 - Maintain configuration control over the software tools
 - Periodically release new versions of tools
 - Develop and implement enhancements as required by users
- Foster the development of MIL-STD-1750A microprocessors

- Current Activities of the ECSPPO

— Providing maintenance and configuration management support for:

- The AFWAL 1589B JOVIAL Compiler hosted on the IBM 370 compatible computers and the DEC-10 computer
- The AFWAL developed MIL-STD-1750A Support Tools: ALAP, LINKS, ALINKS, SIM50A

— Established maintenance support contract in April 82 to perform maintenance and configuration management software tools

- From Aug 81 thru Mar 82 maintenance support was provided to the ECSP0 by AFWAL
- AFWAL contract transferred to ECSP0 1 April 82 and level of contract increased
- A number of compiler enhancements are now complete or nearing completion
 - The translation of the IBM Compiler source from J73/I to J73 was completed
 - The IBM code generator was enhanced with a number of optimizations
 - The IBM compiler was self-compiled
 - The DEC-10 compiler was self-compiled
 - Double precision Integer/Fixed point for the 1750A target is now in testing and should be ready for release soon
- A VAX rehost of the AFWAL is in progress with a scheduled completion date of Sep 82
 - Compiler will have a MIL-STD-1750A code generator
 - Compiler will be available for test site use in Oct 82
 - Compiler will be available for full release in spring of 83
- ECSP0 now supporting a number of programs with maintenance service
 - LANTIRN
 - MATE
 - Satellite Control Facility
 - GPS
- ECSP0 Future Plans
 - ECSP0 services will improve and increase due to additional manpower and the direct contractor support
 - Error status reports will be available as requested by the JOVIAL Users Group

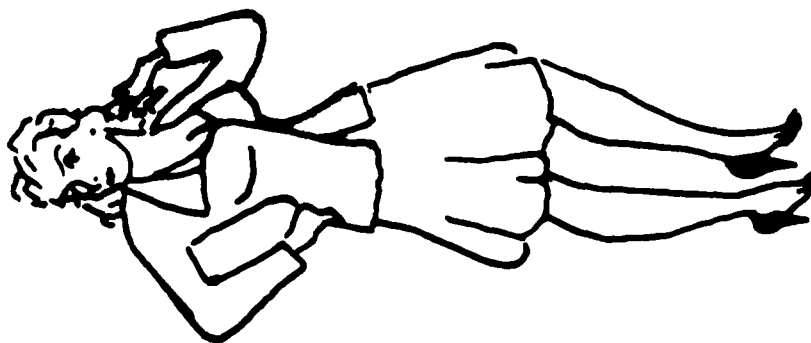
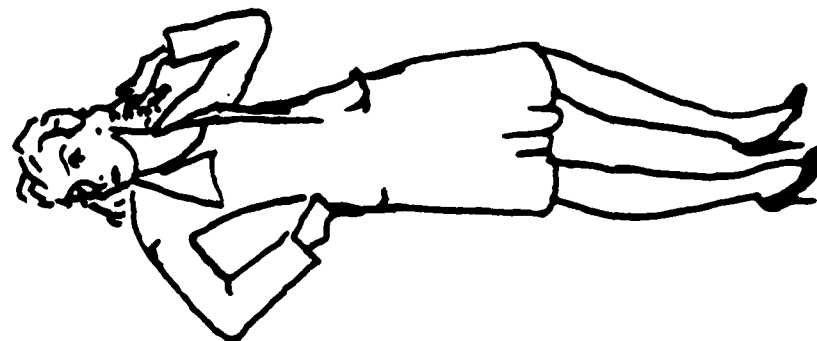
- Error corrections will be handled more responsively
- Resources are available for product improvements, attention will be focused on the 1750A code generator
- A VAX computer will be acquired to support ECSP0 activities
 - All configuration management will be done on VAX computer
 - VAX computer will be the primary maintenance system
 - VAX will be networked to IBM and DEC-10 systems for checkout of those compilers
- The ECSP0 plans to support additional products as conditions warrant and as resources permit
 - The VAX/1750A compiler that is currently under development
 - Code generators for other microprocessors
 - Code generators are already developed and maybe made available to ECSP0
 - Compiler and 1750A Support Tools developed under F-16 MSIP
 - Current plans call for ECSP0 to begin support in FY84 when tools turned over to the Government

PROGRAMMING LANGUAGES

FORTRAN

JOVIAL J-73

ADA



OVERVIEW

• MISSION

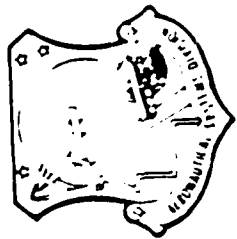
• CURRENT ACTIVITIES

• FUTURE PLANS

ECSP0 MISSION

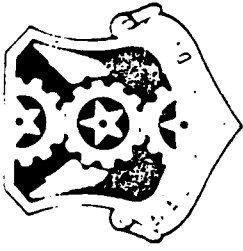
• BACKGROUND

• ECSP0 CHARTERED

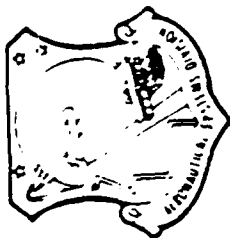


ECSP0

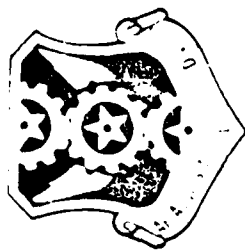
MAINTAINS AND CONFIGURATION MANAGES



- JOVIAL COMPILERS
- 1750A SUPPORT SOFTWARE TOOLS

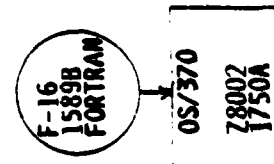
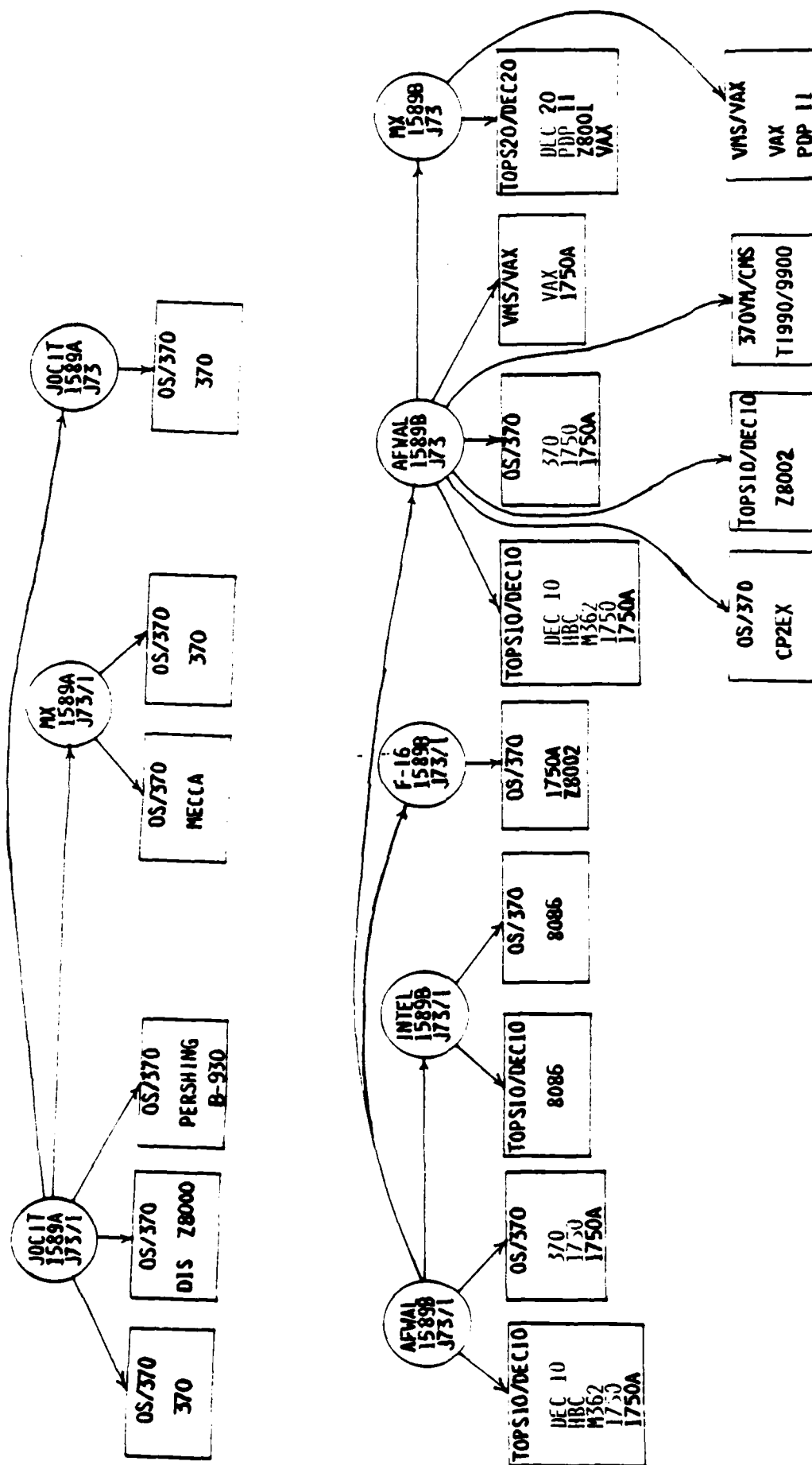


STANDARDIZATION MISSION ON MAINTENANCE OF JOVIAL COMPILERS & 1750A SUPPORT TOOLS



- 1. ERROR CORRECTION**
- 2. ENHANCEMENT**
- 3. CONFIGURATION CONTROL
MANAGEMENT**
- 4. UPDATED VERSION PRODUCTION**
- 5. PROBLEM CONSULTATION**
- 6. STATUS INFORMATION FOR USERS**

JUVIAL COMPILER GENEALOGY



CURRENT ACTIVITIES

- SUPPORTING COMPILERS AND 1750A SUPPORT TOOLS
 - AFWAL 1589B COMPILER ON IBM AND DEC-10 WITH 1750A TARGET
 - AFWAL 1750A SUPPORT TOOLS: ALAP/LINKS/SIM50A/ALINKS
- ESTABLISHED MAINTENANCE SUPPORT CONTRACT
- COMPLETING COMPILER ENHANCEMENTS
 - BOOTSTRAPPING
 - DOUBLE PRECISION INTEGER/FIXED POINT FOR 1750A
- DEVELOPING VAX J73 COMPILER WITH 1750A TARGET

• PROVIDING SUPPORT TO:

LANTIRN

MATE

SATELLITE CONTROL FACILITY

GPS

FUTURE PLANS

- INCREASE IN SERVICE DUE TO MORE PEOPLE AND CONTRACT SUPPORT

ERROR STATUS REPORTS

FASTER ERROR TURNAROUND

PRODUCT IMPROVEMENTS

- ACQUIRE VAX COMPUTER TO SUPPORT CM AND MAINTENANCE

- SUPPORT ADDITIONAL PRODUCTS

VAX 1750A COMPILER

CODE GENERATORS FOR OTHER MICRO'S

F-16 COMPILER AND 1750A SUPPORT TOOLS



תעשיית אלקטרוניקה
A.E.L. ISRAEL LTD. electronics industries

AN IMPLEMENTATION OF A 1750 COMPUTER
AND ITS WORKING ENVIRONMENT

Israel E. Koren
Chief of Computers Center
A.E.L. ISRAEL Electronics Industries
48 Mivtza Kadesh st Bene Berag, 51203, Israel

A.E.L. Israel in response for an RFP by IAI (Israel Aircraft Industries) c/o Israel Air Force has begun a development and implementation of a computer that will emulate the 1750 ISA and various tools. A compact, inexpensive and portable maintenance and development system (MDS) and tools shall be presented. Topics that shall be covered are:

- a. Presentation of a 1750 computer, accessories and tools developed at A.E.L. Israel.
- b. Software tools developed to communicate, monitor and control 1750 computers and to assist in development and maintenance of military embedded computer systems.
- c. Design ideas for special tools that allow efficient integration of hardware with high order language software.



תעשיית אלקטרוניקה
A.E.L. ISRAEL LTD. electronics industries

I. INTRODUCTION

A.E.L. Israel is a science based company dealing in two major areas, one is telephone switching and the other is military electronics. The military electronics is divided into three divisions: Airborne, Marine and Ground Forces Equipment. The company has over 1000 employees of which 150 are engineers and 300 technicians.

The Airborne Division had since 1974 experienced airborne EW with embedded computers which had led the company to invest in a computer design team that purchased a basic computer production file and developed it to a product which is now installed on first line aircraft and field service ground equipment (named CPLN).

During 1981, the airborne division had designed, built and qualified a compact airborne computer to meet a customer's specifications (named CPLR). When the IAI had submitted an RFP for a 1750 computer the design team had found a close match between the CPLR and the requirements of the MIL-STD, the hardware architecture to perform the standards ISA is already there and by means of micro-programming, 80% of the 1750 ISA can be accomplished. The computer design team is currently emulating these instructions and at the same time investigating the needed hardware changes for complete implementation. It is assumed that a complete 1750 computer will be available in the fourth quarter 1982.

II. A GENERAL PURPOSE AVIONIC PROCESSOR

The GPAP-1750 is a family of products that build up a General Purpose Avionic Processor. The processor is composed of a light weight enclosure, a variety of power supplies, memory boards and peripheral interfaces. The



תעשיית אלקטרוניקה
A.E.L. ISRAEL LTD. electronics industries

CPU board comes in two versions, one is implemented using a chip set while the other is built around a bit slice. The advantages of the chip set will be in lower power consumption which leads to lower heat generated and probably a lower price. The bit slice advantages are: extreme speed, availability and most important, flexibility. The flexibility means that one can extend the instruction set of the CPU using the BIF option of the MIL STD at a maximum execution speed. The enclosure is 1/2 ATR short and can be mounted on any side. In the proved CPLN version it dissipates 140 watts at full MIL STD 5400 class II environment with no forced air cooling. The card cage holds eight boards 10 x 7 inches with a 120 pin connector on its 10 inch side. The bus is a separate data/address bus and leaves the user with about 50 pins free. The layered mother board is organized so that the user can route signals to the front panel on separate layers so that no interference can occur with computer signals.

A Triple Option Card - TOC, supports the bit slice CPU board with the discrete Registers, Timers, Trig Go and Start-Up PROM. In the chip set CPU all this support circuitry will become part of the CPU board.

The computed speed of the GPAP-1750 with bit slice CPU and fast memory will be 652 KOPS DAIS MIX. Power consumption for a bit slice CPU, TOC and 64 K words of memory will be around the 25W. Weight of the above configuration is about 12 pounds with room for additional five cards.



תעשיית אלקטרוניקה
A.E.L. ISRAEL LTD. electronics industries

III. MAINTENANCE AND DEVELOPMENT STATION - MDS

The Maintenance and Development Station is in fact a 1750 based computer whose computing performance is identical to the GPAP-1750 computer. An MDC LVL2 or LVL2E, the versatile peripheral equipment available and a laboratory enclosure turn this computer into an easy to access 1750 computer for maintenance and development.

The MDS includes a general purpose power supply to be operated from 220 VAC 50Hz or 110 VAC 60 Hz. The power supply provides the following outputs:

+5 VDC at up to 35A

+12 VDC at up to 1.5A

The card cage in the MDS is wired according to that of the GPAP-1750 bus and can occupy up to fourteen 10 x 7 inch family-standard card assemblies.

Since all card assemblies have standard connections according to the versatile construction method, various aspects of the MDS can be emphasized e.g. additional memory, additional communication channels, etc.

IV. MAINTENANCE AND DEVELOPMENT CONSOLE - MDC 1750

The maintenance and development console (MDC) for the GPAP-1750 is based upon means and methods successfully developed and implemented in A.E.L. Israel computers. The versatile MDC unit features a variety of operating modes and it is designed to assist users of the GPAP-1750 processor family.

The unit provides wide system support particularly in two major areas, development and maintenance.

The MDC unit is, in fact, a small PDP 11 computer station, connected to the controlled GPAP-1750 computer via a special personality module called Computer Monitor and Controller (CMAC).



תעשיית אלקטרוניקה **ישראל בע"מ A.E.L. ISRAEL LTD. electronics industries**

The MDC is also designed to communicate and control a drop-in-board Real Time Bus Analyzer (RTBA) which is very useful in Hardware/Software integration. The MDC is offered in two levels: LVL2 and LVL2E. The basic difference between these two levels is the type of peripheral equipment used and the extent of software support.

LVL2 Maintenance and Development Console. The system includes a VT103 computer terminal equipped with an SBC-11/21 computer, a 64 programmable Input/Output interface to control the CMAC and a dual TU58 tape cassette system. Resident Soft Panel Software (in EPROM) on board the CPU enables the user to:

- Examine/Deposit data in memory
- Examine/Deposit data in General Registers
- Load/Unload programs using tape cassette
- Start/Stop program execution
- Single-Step through program
- Set/Clear Breakpoint on any or all of the following conditions:
 - Fetch instruction from any address (including Read Only Memory)
 - Read/Write from/to any address
 - Read/Write any data
- Control the Real Time Bus Analyzer (RTBA)
- List/Disassembly of memory area on the VT103 screen
- Be used as a console terminal for the computer under test

LVL2E Extended MDC System. This system is in fact an extension of the LVL2 system. The difference between these two systems is in the extent of software support

AD-A150 583

PROCEEDINGS OF THE TECHNICAL FORUM (3RD) ON THE F-16
MIL-STD-1750A MICROP. (U) AERONAUTICAL SYSTEMS DIV
WRIGHT-PATTERSON AFB OH J L PESLER ET AL. 06 MAY 82

6/6

UNCLASSIFIED

ASD-TR-82-5011-VOL-1

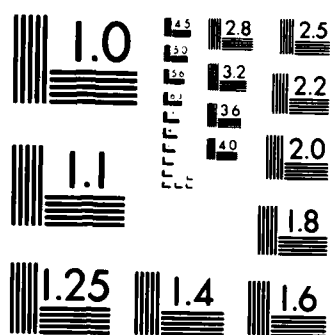
F/G 9/2

NL

END

FILED

DTIC



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



תעשיית אלקטרוניקה **A.E.L. ISRAEL LTD. electronics industries** **א.א.ל. ישראל בע"מ**

and the available extention using an operating system and a disk. The extended LVL2E system comprises a VT103 computer terminal equipped with an LSI, 11-23 CPU, 160 K byte of RAM, Dual TU58 cassette, 10.8 M byte Disk, a 64 programmable Input/Output interface to control the CMAC and a IEEE 488 instrumentation link.

The extra means of the LVL2E MDC provides for running the following programs:

- File System (Disk, Tape cassette)
- Editors (text, core, disk)
- Cross assembler for the 1750 ISA
- Advanced remote symbolic debugger

V. COMPUTER MONITOR AND CONTROLLER - CMAC

The Computer Monitor and Controller - CMAC - is implementation dependent. In the chip set processor it will be a bus controller only, while in the bit slice implementation it is able to control the internal CPU operation (thru special micro-code). The CMAC is actually an interface between a remote console (or computer) and the machine. It allows the user to establish a break whenever a certain condition occurs. The CMAC if used plugs into the first slot adjacent to the CPU card and requires no special wiring on the system bus. (There is special wiring between slots 1, 2.) The control signals to the CMAC are provided via a connector on top of the card opposite the bus connector.

VI. REAL TIME BUS ANALIZER - RTBA

Real time trouble analysis in computer systems is one of the significant advantages of the MDS and MDC systems. The RTBA assembly enables trouble analysis while the program is executed, without any interference or intervention of the test assembly. The RTBA enables



תעשיית אלקטרוניקה
A.E.L. ISRAEL LTD. electronics industries

trouble shooting while the system is operating with a complete set of interrupts and DMA. The main functions performed by the RTBA are:

- Halting program executions on one of pre-programmed conditions,
- Displaying program flow before it was halted.

The RTBA is composed of three main circuits mounted on a single 10 x 7 inch standard card as follows:

- Real time data accumulator
- Real time equality detector
- Control analysis circuits

The RTBA is controlled via the CMAC and the MDC or can be accessed via program control if desired.

VII. SOFTWARE TOOLS

As a system house A.E.L. deals with various types of computers, all of them are mini's and micro's. The experience with these computers led to a conclusion to equip the programmer with a small station that will allow fast program editing compilation and debugging that is the incentive of the MDC.

The MDC LVL2 system includes a DEC VT103 computer terminal equipped with a programmable 64 input-output interface to control the CMAC and a TU-58 dual cassette tape system. A special resident firmware called soft panel within the VT103 allows the user to control the CMAC as a control panel.

The LVL2 MDC can be upgraded to a LVL2E by replacing the CPU adding memory and a disk sub system. The operating system can be either RSX11-M or UNIX since the supporting software is written in "C" language.



תעשיית אלקטרוניקה
A.E.L. ISRAEL LTD. electronics industries

The supporting software includes:

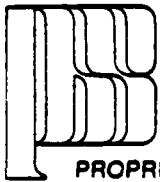
- Cross assembler for 1750 ISA
- Remote symbolic debugger

and all features of LVL2 MDC.

SUPPORT OF SOFTWARE TOOLS

Mr Joel Fleiss

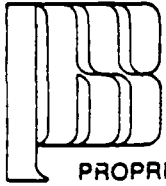
Proprietary Software Systems Inc



PROPRIETARY SOFTWARE SYSTEMS

PROPRIETARY SOFTWARE SYSTEMS

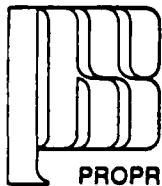
LOCATION	Corporate Headquarters in Los Angeles Offices in Seattle and Ventura County
FOUNDED	October, 1969 State of California
EXPERTISE	• JOVIAL Compilers (J3, J73/I, J73) • Macro/Meta Assemblers (DUAL, etc.) • Link Editors (DUAL, etc.) • Functional Simulators (PDP-11, NOVA, INTEL, MIL-STD 1750A, ZILOG, etc.) • Debug Tools (AIDS, etc.) • Studies: Ada Conversion Transferability Compiler Measurement Code Optimization • Code Auditors (RADC, Boeing) • Support Software Tools (Editors, I/O Handlers, Dictionaries, etc.)



PROPRIETARY SOFTWARE SYSTEMS

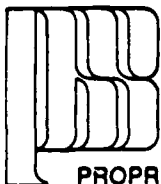
OUTLINE OF PSS PRESENTATION

- 1) Proprietary Software Systems, Inc.
 - A. Corporate Background
 - B. Major Products
 - C. Recent Contracts
- 2) AFWAL VAX Rehost
 - A. 1750A Compiler
 - B. Support Software Tools (McDonnell Douglas)
- 3) General Dynamics
 - A. 1750A Compiler Modifications/Enhancements (AFWAL 1750A Baseline)
 1. Subroutine Linkage
 - *2. Automatic Allocation
 - *3. Literal Pools
 - *4. Machine Specific Subroutines
 - *5. Symbolic Debug
 - *6. Sections
 - *7. Base Register Emphasis
 - *8. Register Copy
 - *9. Binary Operators
 - *10. Fixed Point
 - *11. Bug Corrections
 - *12. IEEE Compatible Assembly Listing
 - *13. Subroutine Register Save
 - B. DUAL IEEE Assembly Syntax
 1. IEEE Syntax
 - a) Base and Index Registers Enclosed in Parentheses
 - b) Extended Branch Mnemonics
 - c) IEEE Mnemonics
 - d) Consistent Argument Specification (Source-Destination)
 - e) Addressing Mode Delimiters
 2. DUAL Features
 - a) Host/Target Independence
 - b) Extensive Macro Facilities
 - c) Minimal User Restrictions
 - d) Multiple Location Counters
 - C. DUAL Link Editor
 1. User Controlled Memory Allocation
 2. Overlay Facilities
 3. Object Libraries
 4. Composite Listing
 5. Debug Facilities



PROPRIETARY SOFTWARE SYSTEMS

- D. Mil-Std 1750A Functional Simulator
 - 1. IBM/DEC-10 Host
 - 2. Interactive Debug Facilities
 - 3. DUAL Compatibility
- E. Load Module Processor
 - 1. General Dynamics Compatible Format
 - 2. Symbolic Debug Facilities
- F. Information Flow



PROPRIETARY SOFTWARE SYSTEMS

PSS PRODUCTS/SERVICES

JOVIAL (J73)

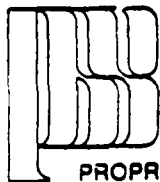
PSS has in-depth knowledge of the existing AFWAL JOVIAL (J73) 1589B compiler. Its staff includes the original designer of the code generator/optimizer along with the optimizer implementor. PSS targets include the MECA-43 16 bit, MECA-43 24 bit, the Delco M362S, the Marconi 920, and the Intel 8086. Targets in progress include the Mil-Std 1750A, the Zilog Z8001 and Zilog Z8002, Vax 780, and the CP2-EX.

DUAL Meta Assembler

DUAL is an easily retargetable macro assembler. PSS has developed over 120 cross assemblers with DUAL, each including DUAL's extensive macro facility. DUAL includes a powerful linkage editor with numerous features designed specifically for the aerospace environment.

AIDS

AIDS provides a simplified method of validating a program. It runs either in a simulated or actual mode and has numerous debugging facilities to simplify program checkout.



PROPRIETARY SOFTWARE SYSTEMS

RECENT CONTRACTS (SINCE 1981)

ARMA	DUAL Cross Assembler
BOEING	Support Software for IUS project, including JOVIAL Compiler, M362S assembler, link editor, code aud- itor, and numerous utility tools. J73 1750A Enhancement Study
COOPERS & LYBRAND	Special Purpose Language
GENERAL DYNAMICS*	JOVIAL J73 1589B compiler, assem- bler, link editor, and functional simulator for Mil-Std 1750A and Zilog Z8002.
HAMILTON STANDARD	JOVIAL J73 1589B Subset Compiler hosted on VAX 780
INTEL	Retargetting of JOVIAL J73 1589B compiler to Intel 8086
LOCKHEED	DUAL Cross Assembler
MOTOROLA*	DUAL Cross Assembler
NASA	DUAL Cross Assembler
NORDEN*	JOVIAL J73 1589B Compiler to Zilog Z8001.
NTEC	DUAL Cross Assembler
RADC	Code Auditor for JOVIAL J73 1589B Language
RADC	JOVIAL-to-Ada Translation Study
SINGER*	J73 1589B Compiler targeted to CP2-EX
WRIGHT-PATTERSON AFB	Rehost of JOVIAL J73 1589B compiler to VAX 780

* Indicates technical effort still in progress.

AFWAL VAX REHOST

- DEC-10 Host with VAX 780 Target

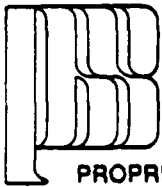
Retargetting DEC-10 AFWAL J73 compiler to generate VAX 780 macro assembly source.

- VAX 780 Host with VAX 780 Target

Operational VAX 780 JOVIAL J73 compiler operating under VMS operating system.

- VAX 780 Host with 1750A Support Software

Rehosting existing AFWAL 1750A code generator to VAX 780 system interfacing with 1750A target, Validating the AFWAL compiler's interface with the re-hosted McDonnell Douglas 1750A support tools.



PROPRIETARY SOFTWARE SYSTEMS

GENERAL DYNAMICS AFWAL J73 COMPILER MODIFICATIONS/ENHANCEMENTS

- Subroutine Linkage (Passing arguments in registers)
Modifying existing technology to pass arguments in registers.
- Automatic Allocation
Current AFWAL compiler allocates local automatic data and local temporaries statically and also assigns a base register to cover this data region. PSS has modified this allocation to allocate local automatic data and temporaries on a dynamic stack.
- Literal Pools
Current AFWAL compiler allocates literals on a module by module basis. PSS modification pools all literals at link time, thus assuring only one copy of each literal. This includes partial literals.
- Machine Specific Subroutines
PSS has modified the AFWAL compiler to automatically recognize a set of 1750A specific functions in order to facilitate generation of efficient 1750A instructions.

These machine specific functions include:

ROTATE
EXEC'CALL
MOV'DATA
OUTPUT
INPUT
CARRY'FLAG
- Symbolic Debug
The compiler object module now includes symbolic information to support simplified debug of 1750A J73 JOVIAL programs.
- User Controlled Allocation
The AFWAL compiler has been modified to include a separation of each compilation unit into four distinct sections. Each of these sections has a section number associated with it and can be modified by the user to control allocation. The sections are CODE, DATA, CONS and LPOOL.
- Base Register Emphasis
PSS has made significant modifications to the AFWAL code generator addressing logic so that it emphasizes base register addressing (short instructions).

COMPILER MODIFICATIONS/ENHANCEMENTS
(CONTINUED)

- **Specific Optimizations**

Several target specific optimizations have been made to minimize the amount of code being generated (minimize unnecessary register copies, pushing and popping only required registers on subroutine entry and exit, fixed point operations to generate inline code, special logic to minimize binary code generation when one of the arguments is a short immediate, etc.)

- **IEEE Assembly Syntax Capability**

Assembly listing reflects new 1750A IEEE assembly syntax.

- **Bug Corrections**

Correction of any known 1750A code generator problems.

PSS JOVIAL J73 1589B 1750A SUPPORT SOFTWARE PROJECTS

- AFWAL VAX Rehost

Existing 1750A compiler and McDonnell Douglas support software hosted on VAX 780 for WPAFB.

- General Dynamics

Modified/Enhanced 1750A compiler with new support software hosted on IBM 370. Part of General Dynamics F-16 flight improvement program.

- Boeing

Modified/Enhanced 1750A JOVIAL J73 compiler coupled with Boeing support software hosted on IBM 370. Part of Boeing B-1B program.

GENERAL DYNAMICS DUAL MIL-STD 1750A CROSS ASSEMBLER

IEEE Assembly Syntax

- Base and Index Registers enclosed in parentheses

<u>OLD</u>	<u>IEEE</u>
LD 1,OMEGA,5	LD R1,OMEGA(R5)

- Register Mnemonics Must Be Used

R0-R15
RL0-RL7
RH0-RH7
RR0-RR14
RQ0-RQ12

- IEEE Mnemonics
- Consistent Argument Specification
Always SOURCE,DESTINATION
- Addressing Mode Delimiters

@.....Indirect
#.....Short Immediate
##.....Long Immediate
+.....Auto Increment
-.....Auto Decrement
%.....Relative Addressing

GENERAL DYNAMICS DUAL MIL-STD 1750A CROSS ASSEMBLER

DUAL Features

- Host Machine Independence

Operational on 22 different hosts including the IBM 370, DEC-10, VAX 780, Univac 1100, Honeywell 6000, CDC 6000, Interdata 7/32, etc.

- Target Machine Independence

Over 120 cross assemblers by PSS and its customers, including numerous built-in cross assemblers (1750A, PDP-11, Intel 8080, Data General NOVA, Zilog Z8001, Zilog Z8002, Motorola 6800, etc.)

- Extensive Macro Facilities

Extremely powerful directives and functions facilitating macro implementation. Features include user definable symbol table, numerous character manipulation functions, conditional directives, macro libraries, etc. DUAL contains over 120 directives and 50 functions.

- Minimal User Restrictions

No limit to number characters in a name, number of macros, etc.

- Multiple Location Counters

Each assembly unit can contain up to 32 distinct location counters. DUAL supports CSECTS, DSECTS, and FORTRAN common.

GENERAL DYNAMICS DUAL MIL-STD 1750A CROSS ASSEMBLER

IEEE Assembly Syntax

- Base and Index Registers enclosed in parentheses

<u>OLD</u>	<u>IEEE</u>
LD 1,OMEGA,5	LD R1,OMEGA(R5)

- Register Mnemonics Must Be Used

R0-R15
RL0-RL7
RH0-RH7
RR0-RR14
RQ0-RQ12

- IEEE Mnemonics
- Consistent Argument Specification
Always SOURCE,DESTINATION
- Addressing Mode Delimiters

@.....Indirect
#.....Short Immediate
##.....Long Immediate
+.....Auto Increment
-.....Auto Decrement
&.....Relative Addressing

GENERAL DYNAMICS DUAL MIL-STD 1750A LINK EDITOR

Link Editor Features

- User Controlled Memory Allocation

User can specify whether modules are to be contiguous or whether sections are to be contiguous, thus potentially separating J73 CODE, CONS and DATA into unique memory sections.

- Overlay Facilities

Link Editor supports overlays via OVERLAY command or FORTRAN common.

- Object Libraries

Link Editor (if option chosen) will automatically search object library to satisfy any undefined externals. Link editor also includes IGNORE and INCLUDE commands.

- Composite Listing

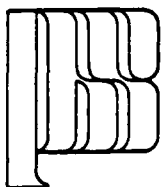
If user passes source in DUAL assemblies (LIST), then, the link editor on option (LIST) will generate a composite absolute assembly listing for all the included object modules.

- Debug Facilities

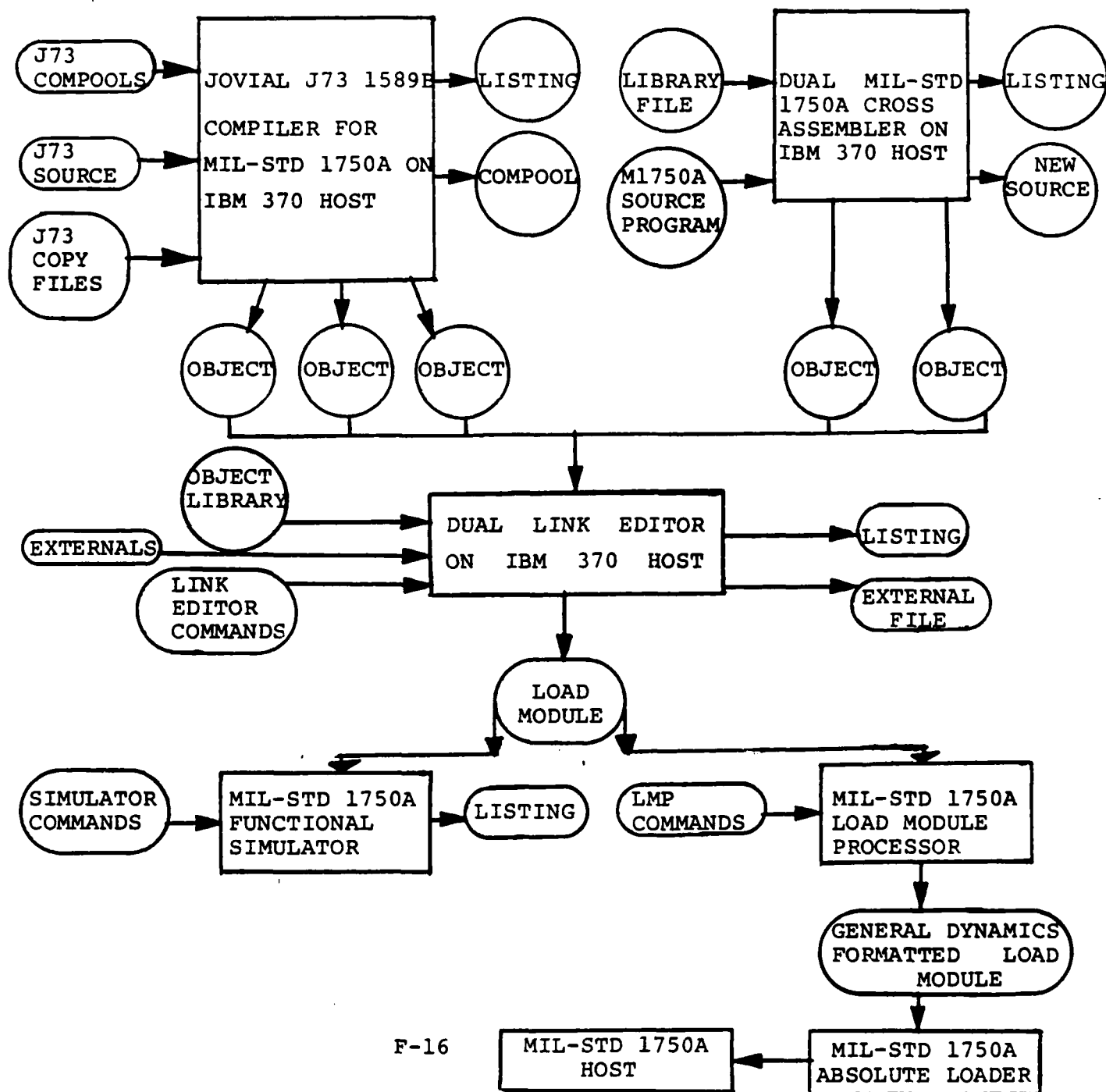
Link Editor generates a file containing all addresses of symbols on a module by module basis to be used by a debugger.

GENERAL DYNAMICS MIL-STD 1750A SUPPORT TOOLS

- Functional Simulator
 - IBM 370 or DEC-10 Host
 - Interactive Debug Facilities
 - DUAL Compatibility
 - Tool used for JCVS Testing
- Load Module Processor
 - Outputs General Dynamics Compatible Format
 - Generates Separate Symbolic Debug File
 - Checksum Facility



GENERAL DYNAMICS MIL-STD 1750A SUPPORT SOFTWARE INFORMATION FLOW



USE OF THE 1750A SUPPORT SOFTWARE ON AN IBM 3033 UNDER MV/CMS

John Riordan

Mikros Systems Corporation
Mercerville, New Jersey 08619

ABSTRACT

The McDonnell Douglas Support Software package is configured to run on a DEC-10 system and an IBM system running TSO. Since the time-sharing system available at Mikros was IBM's VM/CMS, a conversion of the supplied control language was undertaken. Details of this conversion are given.

Simulator IO routines were written to permit interactive use of a 1750A cross-assembled program executing through the simulator. Examples of the control language and IO routines developed will be included.

INTRODUCTION

The MIL-STD-1750A Support Software Package was developed at McDonnell Douglas under contract to the USAF. The Package provides a complete set of software development tools for 1750A assembly language programming, including a Macroprocessor (MPP), Assembler (MSA), Linker (MSL), Simulator (MSS) and utilities for library maintenance and reformatting.

The Package is available from ASD at Wright Patterson AFB, and is distributed for use on an IBM TSO system, or on DEC 10 and VAX 11/780. This paper describes the installation of the Support Software Package on the Princeton University IBM 3033 system under VM/CMS timesharing.

The tape supplied contains 27 files for the installation, testing and use of the support system under IBM's TSO. For a CMS system, the following procedures must be followed:

- a) source code files are copied from the tape and made into proper FORTRAN or ASSEMBLE files
- b) test cases are copied and named as required by the CMS control language
- c) CLIST files are not copied. The equivalent CMS EXEC's must be written
- d) the HELP and ERROR files are copied and named as required

Each program provided must be compiled or assembled, (added to a TXTLIB, if applicable), LOADED, and GENMODed. The program will then be executable (as a module). The proper EXEC will be used to specify parameters, to allocate files and to start execution.

In order to describe the use of the system as configured at MIKROS, the following convention must be detailed:

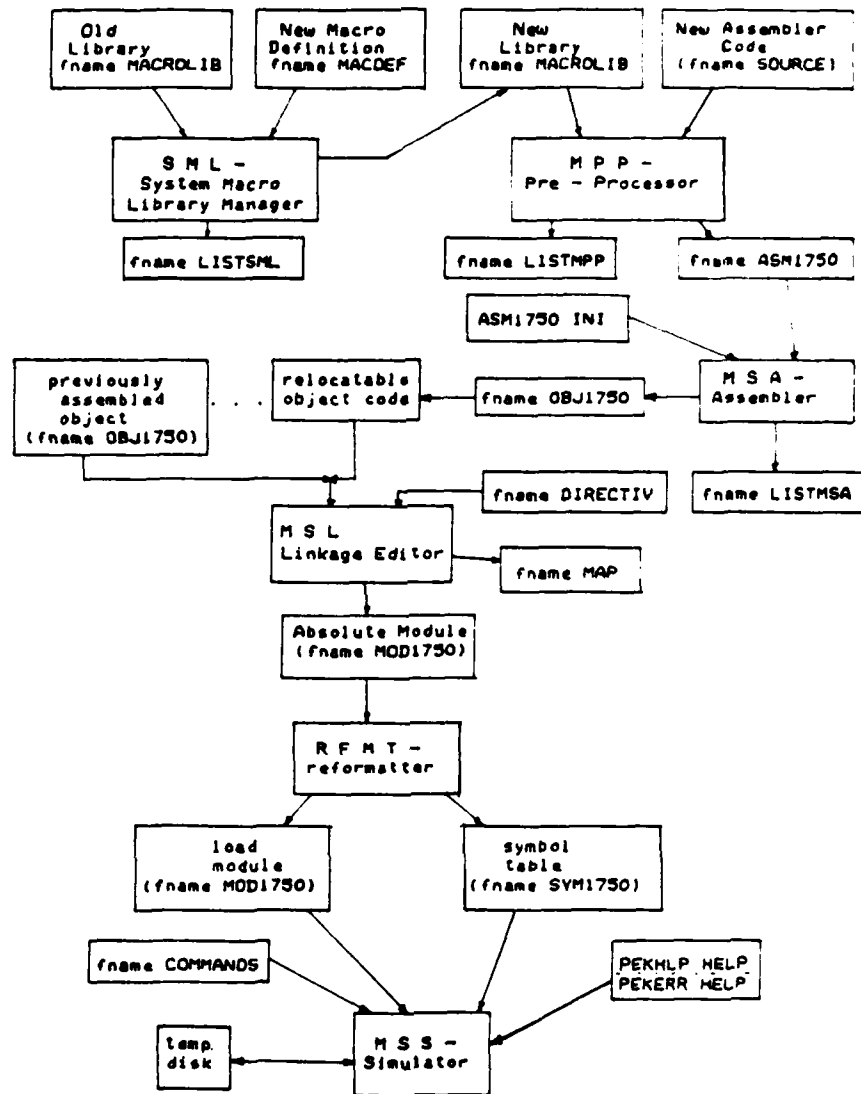
- o Files have three descriptors - filename, filetype, filemode (this is a CMS convention)
- o In all EXEC's used for running the support system, the following are assumed:

Filename	Filetype	Filemode	Use
any	MACROLIB	A	Macro Library
any	MACDEF	A	Macro Definition
any	LISTSML	A	Library Update Listing
any	SOURCE	A	1750A Source Pgm.
any	LISTMPP	A	MPP Listing
any	ASM1750	A	1750 Assembler Code (output of pre-processor)
any	OBJ1750	A	Object file (output of Assembler)
any	LISTMSA	A	Assembly Listing
any	DIRECTIV	A	Linkage Editor Commands
any	MAP	A	Linkage Editor Map
any	ABS1750	A	Absolute Module (output of Link Editor)
any	MOD1750	A	Load Module (output of Reformatter)
any	SYM1750	A	Symbol Table (output of Reformatter)

The EXEC's used (in the included listing) are named:

SML	EXEC	- System Macro Library Manager
MPP	EXEC	- Pre-Processor
MSA	EXEC	- Assembler
MSAX	EXEC	- Assembler (without prompting)
MSL	EXEC	- Linkage Editor
MSLX	EXEC	- Linkage Editor (without prompting)
MSS	EXEC	- Simulator
MSSX	EXEC	- Simulator (without prompting)
ALR	EXEC	- Executes EXEC's, MSA, MSL, and MSS

SIMULATOR FLOW CHART



USE OF FILES ON SUPPLIED TAPE

File No.	Use	New Names
1	JCL to copy files on to TSO	Delete
2	QSHIFT Assembler Prog.	Assemble, Com- pile, Add to SIMULIB
3	CORE	
4	IUNPAK FORTRAN Prog.	
5	QCONV FORTRAN Prog.	
6	SECOND FORTRAN Prog.	SML FORTRAN
7	System Macro Library Manager - SML	
8	Pre Processor - MPP	MPP FORTRAN
9	Hash Table Prog.	Output is
10	Input to Hash	ASM1750 INI A4
11	Assembler - MSA	MSA FORTRAN
12	Linkage Editor - MSL	MSL FORTRAN
13	Reformatter - RFMT	RFMT FORTRAN
14	Assembler Routine (necessary for Simulator)	Assembler, Add to SIMULIB
15	Simulator - MSS	MSS1...7 FORTRAN
16	Help File	PEKHLP HELP
17	Error File	PEKERR HELP
18	Test Case for SML	Used these files to test Simulator
19	MPP	
20	MSA	
21	MSL	
22	MSS	
23	TSO Proc's for SML	Delete
24	MPP	
25	MSA	
26	MSL	
27	MSS	

INPUT/OUTPUT FOR 1750A SIMULATOR

To simulate the I/O processes using the XIO PI/PO and CI/CO instructions, the following assumptions are made:

- 1) The PI/PO instructions read and write from a channel number encoded into the instruction. This number (0-3) is the 'y' of the specification. The 'xx' of the specification is disregarded.
- 2) The channel ↔ FORTRAN Unit No. correspondence is:

	(	channel	0	Unit	20
PI	(		1		21
	(		2		22
	(		3		23
CI	(		4		24

	(		0		30
PO	(		1		31
	(		2		32
	(		3		33
CO	(		4		34

(Note: In code CHNLNO = Channel No. + 1). This correspondence is set in the initialization of the labeled COMMON "QUNITS".

- 3) The (Carriage Return) character is not a valid 1750A input/output character. This limitation results from the fact that FORTRAN demands we read a record at a time, not a character at a time. Therefore, the (CR) serves as the end-of-record delimiter.
- 4) The channel buffer sizes are set in the dimensioning of the labeled COMMON "IOHAND".

Note: The large source program files are too big to edit on CMS in the maximum virtual machine size available to us. To make the changes necessary in the Simulator, the MSS source program was divided into six subset files, using the GETFILE CMS command. Changes to the program's COMMON must be made in each routine, along with code changes made to IN and OUT's COMMON, and the code necessary to implement the IO are shown in the accompanying listing.

CMS EXEC's for Support System

```

*
*           S M L       E X E C
*
*CONTROL ERROR
*TYPE SYSTEM MACRO LIBRARY MANAGER - S M L
*      EXEC created 11-03-81 BY John Riordan OF MIKROS Systems Corp
*
*TYPE NOTE: <CR> in prompts signifies carriage return.
FI * CLEAR
*TYPE INPUT DATA FILE NAME?
*READ VARS *INPUT
*IF *.INPUT = . *GOTO -STAR1
FI 1 DISK *INPUT MACDEF
*GOTO -NXT1
-STAR1
FI 1 TERMINAL
*
-NXT1
*TYPE Listing data file name NAME ( or <CR> for terminal)?
*READ VARS *LIST
*IF *.LIST = . *GOTO -STAR2
FI 2 DISK *LIST LISTSM1 A (RECFM FBA LRECL 80 BLKSIZE 800
*GOTO -NXT2
-STAR2
FI 2 TERMINAL
*
-NXT2
FI 5 TERMINAL
FI 6 TERMINAL
*TYPE Old Macro Library file name?
*READ VARS *OLDLIB
*TYPE New Macro Library file name?
*READ VARS *NEWLIB
*IF *.OLDLIB = . *GOTO -NOLD
FI 3 DISK *NEWLIB MACROLIB A (RECFM VBS LRECL 516 BLKSIZE 2584
*GOTO -NXT3
-NOLD
FI 3 DISK *NEWLIB MACROLIB A (RECFM VBS LRECL 516 RECFM VBS BLKSIZE 2584
*
-NXT3
*IF *.INPUT = . *TYPE SML AWAITING INPUT
SML
*IF *.OLDLIB = . *EXIT
*IF *OLDLIB = *NEWLIB *GOTO -REN
ERASE TEMPOZO MACROLIB
*EXIT
*
-REN
*IF *RETCODE > 0 *EXIT
ERASE *OLDLIB MACROLIB A (NOTYPE
RENAME TEMPOZO MACROLIB A *OLDLIB MACROLIB A (NOTYPE
*EXIT

```

R: T-0.01/0.09 10:31:54

```

*
*           M P P       E X E C
*
&CONTROL ERROR
&TYPE MACRO PRE PROCESS - M P P
*   CREATES ASSEMBLER INPUT FILE FROM
*   SOURCE CODE FILE.
*   CREATED 11-3-81 BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*
F1 * CLEAR
F1 23 DISK TEMP23 ORARY A (LRECL 80 RECFM FB BLKSIZE 800
&TYPE INPUT SOURCE FILE NAME?
&READ VARS &INPUT
&IF &INPUT = . &GOTO -JMP1
F1 5 DISK &INPUT SOURCE A (LRECL 80 RECFM FB BLKSIZE 800
&GOTO -NXT1
-JMP1
F1 5 TERMINAL
*
-NXT1
&TYPE LISTING FILE NAME?
&READ VARS &LIST
&IF &LIST = . &GOTO -JMP2
F1 6 DISK &LIST LISTMPP A (LRECL 137 RECFM VBA BLKSIZE 1414
&GOTO -NXT2
-JMP2
F1 6 TERMINAL
*
-NXT2
&TYPE OUTPUT FILE NAME?
&READ VARS &PUN
&IF &PUN = . &GOTO -JMP3
F1 3 DISK &PUN AS41750 A (RECFM FB LRECL 80 BLKSIZE 800
&GOTO -NXT3
-JMP3
F1 3 TERMINAL
*
-NXT3
&TYPE MACRO LIBRARY FILE NAME (<CR> IF NOT USED)?
&READ VARS &LIB
&IF &LIB = . &GOTO -REST
F1 24 DISK &LIB MACROLIB A4 (LRECL 516 RECFM VBS BLKSIZE 2584
*
-REST
&IF &INPUT = . &TYPE MPP AWAITING INPUT
*
MPP
&IF &RETCODE > 0 &EXIT
ERASE TEMP23 ORARY
*
&EXIT

```

```

*
*           M S A       E X E C
*
&CONTROL ERROR
&TYPE CROSS ASSEMBLER - M S A
*   PRODUCES AN OBJECT FILE FROM AN ASSEMBLER CODE PROGRAM
*   CREATED 11-3-81 BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*
FI * CLEAR
FI 1 DISK TEMP1 URARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
FI 2 DISK TEMP2 URARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
FI 3 DISK TEMP3 URARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
FI 4 DISK TEMP4 URARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
FI 5 TERMINAL
FI 6 TERMINAL
*
&TYPE INPUT SOURCE FILE NAME (<CR> FOR TERMINAL)?
&READ VARS &INPUT
&IF &INPUT = . &GOTO -TERM1      1750 P8
FI 10 DISK &INPUT ASMI750 A (RECFM FB LRECL 80 BLKSIZE 800
&GOTO -NXT2
-TERM1
FI 10 TERMINAL
*
-NXT2
&TYPE LISTING DATA FILE NAME (<CR> FOR TERMINAL)?
&READ VARS &LIST
&IF &LIST = . &GOTO -TERM2
FI 11 DISK &LIST LISTMSA A (RECFM VBA LRECL 137 BLKSIZE 1414
&GOTO -NXT3
-TERM2
FI 11 TERMINAL
*
-NXT3
&TYPE OUTPUT OBJECT FILE NAME (REQUIRED)?
&READ VARS &OBJ
&IF &OBJ = . &GOTO -ERROR
FI 13 DISK &OBJ OBJ1750 A4 (RECFM VBS LRECL 1028 BLKSIZE 5144
FI 12 DISK TEMP12 URARY A4 (RECFM VBS LRECL 1028 BLKSIZE 5144
FI 8 DISK ASMI750 INI A4 (RECFM VBS LRECL 24000 BLKSIZE 24000
&IF &INPUT = . &TYPE MSA AWAITING INPUT
*
MSA
*
&IF &LIST = . &EXIT
&TYPE YOUR ASSEMBLE LISTING IS ON " &LIST LISTMSA"
&TYPE TO PRINT LISTING, TYPE "TYPE130 &LIST LISTMSA"
ERASE TEMP1 URARY A (NUTYPE
ERASE TEMP2 URARY A (NUTYPE
ERASE TEMP3 URARY A (NUTYPE
ERASE TEMP4 URARY A (NUTYPE
&EXIT
*
-ERROR
&TYPE SORRY, OBJECT FILE NAME IS REQUIRED

```

FILE: MSA

EXEC

A

PRINCETON UNIVERSITY TIME-SHARING SYS

TYPE GOODBYE (BETTER LUCK NEXT TIME)
EXIT

```

*
*           M S A X           E X E C
*
&CONTROL OFF
&TYPE CROSS ASSEMBLER - M S A
*   PRODUCES AN OBJECT FILE FROM AN ASSEMBLER CODE PROGRAM.
*   THIS VERSION PERMITS EXECUTING WITH ONE INPUT NAME PARAMETER.
*   AND NO PROMPTING. THE FILE NAMES (WHICH MUST BE COMMON FOR
*   ALL THE FILES) ARE SET WHEN THE EXEC IS EXECUTED:
*   MSAX FNAME * WHERE FNAME IS THE COMMON NAME.
*   THIS EXEC WAS CREATED 12-7-81 BY JOHN RIORDAN OF MIKROS SYSTEMS
*
&IF &1 = . &GOTO -NOAMPER
&1 * CLEAR
&1 1 DISK TEMP1 ORARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
&1 2 DISK TEMP2 ORARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
&1 3 DISK TEMP3 ORARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
&1 4 DISK TEMP4 ORARY A4 (RECFM VBS LRECL 516 BLKSIZE 2604
&1 5 TERMINAL
&1 6 TERMINAL
*
&1 10 DISK &1 ASML750 A (RECFM FB LRECL 80 BLKSIZE 800
&1 11 DISK &1 LISTMSA A (RECFM VBA LRECL 137 BLKSIZE 1414
&1 13 DISK &1 OBJ1750 A4 (RECFM VBS LRECL 1028 BLKSIZE 5144
&1 12 DISK TEMP12 ORARY A4 (RECFM VBS LRECL 1028 BLKSIZE 5144
&1 8 DISK ASML750 INI A4 (RECFM VBS LRECL 24000 BLKSIZE 24000
MSA
*
&IF &RETCODE > 0 &EXIT
&TYPE YOUR ASSEMBLE LISTING IS ON " &1 LISTMSA"
&TYPE TO PRINT LISTING. TYPE: "TYPE130 &1 LISTMSA"
ERASE TEMP1 ORARY A
ERASE TEMP2 ORARY A
ERASE TEMP3 ORARY A
ERASE TEMP4 ORARY A
ERASE TEMP12 ORARY A
&EXIT
*
-NOAMPER
&TYPE YOU FORGET THE FILE NAMES.
&TYPE DON'T GET NERVOUS, TRY AGAIN

```

*

* TYPE REFORMAT PROGRAM - R F M T

* CREATES A LOAD MODULE AND A SYMBOL TABLE

* FROM AN ABSOLUTE FILE.

*

* TYPE OUTPUT LOAD MODULE FILE NAME (REQUIRED)?

* READ VARS &LDM

* IF &LDM = . GOTO -ERROR

*

* TYPE OUTPUT SYMBOL TABLE FILE NAME (REQUIRED)?

*

* READ VARS &SYM

*

* IF &SYM = . GOTO -ERROR2

*

* FI 9 TERMINAL

* FI 3 DISK &LDM MOD1750 A (RECFM FB LRECL 80 BLKSIZE 800

* FI 4 DISK &SYM SYM1750 A (RECFM F LRECL 1024 BLKSIZE 1024 USORG DA

* FI 4 DUMMY

* FI 2 DISK &ABS ABS1750 A4 (RECFM VBS LRECL 1028 BLKSIZE 5144

*

* NOTE: A LOAD OF THE TEXT FILE IS USED HERE INSTEAD OF

* THE EXECUTION OF A MODULE BECAUSE THE MODULE

* SEEMS TO LOAD FUNNY AND CAUSES ERRORS. I DON'T KNOW WHY

* LOAD RFMT (CLEAR START

*

* RFMT

* IF &RETCODE > 0 GEXIT

* ERASE TEMP5 ORARY A

* ERASE TEMP6 ORARY A

* ERASE TEMP7 ORARY A

* ERASE TEMP8 ORARY A

* GEXIT

*

* -ERROR2

* TYPE = REQUIRED = MEANS "WITHOUT WHICH, NOTHING" J

* TYPE TRY AGAIN

* GEXIT 1

* GEXIT

*

* -ERROR

* TYPE SORRY, ABSOLUTE FILE NAME IS REQUIRED

* TYPE GOODBYE, TRY AGAIN

* GEXIT


```

*
*           M S L       E X E C
*
&CONTROL ERROR
FI * CLEAR
&TYPE LINKAGE EDITOR - M S L
*   PRODUCES ABSOLUTE FILE FROM OBJECT FILE.
*   CREATED 11-3-81 BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*
&TYPE DIRECTIVE FILE NAME (<CR> FOR TERMINAL )?
&READ VARS &DIR
&IF &DIR = . &GOTO -TERM1
FI 1 DISK &DIR DIRECTIV A (RECFM F LRECL 80
&GOTO -JMP1
-TERM1
FI 1 TERMINAL
*
-JMP1
&TYPE MAP FILE NAME (<CR> FOR TERMINAL )
&READ VARS &MAP
&IF &MAP = . &GOTO -TERM2
FI 2 DISK &MAP MAP A (LRECL 30 RECFM FBA BLKSIZE 300
&GOTO -JMP2
-TERM2
FI 2 TERMINAL
*
-JMP2
&TYPE ABSOLUTE FILE NAME (REQUIRED)?
&READ VARS &ABS
&IF &ABS = . &GOTO -ERROR
FI 3 DISK &ABS ABS1750 A4 (LRECL 1028 RECFM VBS BLKSIZE 5144
*
FI 5 DISK TEMP5 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 6 DISK TEMP6 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 7 DISK TEMP7 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 8 DISK TEMP8 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
*
&UNIT = 19
-LAB1
&TYPE OBJECT FILE NAME (<CR> IF NO MORE FILES )?
&READ VARS &OBJ
&IF &OBJ = . &GOTO -DONE
&UNIT = &UNIT + 1
FI &UNIT DISK &OBJ OBJ1750 A4 (LRECL 1028 RECFM VBS BLKSIZE 5144
&GOTO -LAB1
*
-DONE
&IF &DIR = . &TYPE MSL #AITING FOR INPUT
*
MSL
*
*
*
*
*

```

```

*
*           M S L X           E X E C
*
CONTROL ERROR
FI * CLEAR
CTYPE LINKAGE EDITOR - M S L
*   PRODUCES ABSOLUTE FILE FROM OBJECT FILE.
*   CREATED 11-3-81 BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*   X VERSION CREATED 12-14-81
FI 1 DISK 61 DIRECTIV A (RECFM F LRECL 80
FI 2 DISK 61 MAP A (LRECL 80 RECFM FBA BLKSIZE 800
*
FI 3 DISK 61 ABS1750 A4 (LRECL 1028 RECFM VBS BLKSIZE 5144
FI 5 DISK TEMP5 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 6 DISK TEMP6 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 7 DISK TEMP7 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 8 DISK TEMP8 ORARY A4 (LRECL 516 RECFM VBS BLKSIZE 2584
FI 20 DISK 61 OBJ1750 A4 (LRECL 1028 RECFM VBS BLKSIZE 5144
*
MSL
*
CTYPE REFORMAT PROGRAM - R F M T
*   CREATES A LOAD MODULE AND A SYMBOL TABLE
*   FROM AN ABSOLUTE FILE.
*
FI 3 DISK 61 MOD1750 A (RECFM F6 LRECL 80 BLKSIZE 800
* FI 4 DISK 61 SYM1750 A (RECFM F LRECL 1024 BLKSIZE 1024 USORG JA
FI 4 DUMMY
FI 2 DISK 61 ABS1750 A4 (RECFM VBS LRECL 1028 BLKSIZE 5144
LOAD RFMT (CLEAR START
* RFMT
  G1F 6RETCODE > 0 GEXIT
  ERASE TEMP5 ORARY A
  ERASE TEMP6 ORARY A
  ERASE TEMP7 ORARY A
  ERASE TEMP8 ORARY A
  GEXIT
*
-ERROR2
  CTYPE " REQUIRED " MEANS "WITHOUT WHICH, NOTHING" J
  CTYPE TRY AGAIN
  GEXIT 1
  GEXIT
*
-ERROR
  CTYPE SORRY, ABSOLUTE FILE NAME IS REQUIRED
  CTYPE GOODBYE, TRY AGAIN
  GEXIT

```

```

*
*           M S S           E X E C
*
*  &CONTROL ERROR
*  CP TERMINAL LINESIZE 30
*  &TYPE 1750A SIMULATOR - M S S
*
*  SIMULATES A 1750A MACHINE RUNNING THE
*  PREVIOUSLY CREATED LOAD MODULE AND
*  SYMBOL TABLE
*  CREATED 11-3-81 BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*
*  FI * CLEAR
*  FI 5 TERMINAL (LRECL 80
*  FI 6 TERMINAL (LRECL 80
*  FI 2 TERMINAL
*  FI 3 DISK PEKHLF HELP A (RECFM FB LRECL 80 BLKSIZE 800
*  FI 7 DISK PEKERR HELP A (RECFM FB LRECL 80 BLKSIZE 800
*  FI 10 DISK TEMP10 ORARY A (RECFM F LRECL 1024 BLKSIZE 1024 DSORG DA
*  FI 10 DISK TEMP10 ORARY A (XTENT 1024
*  FI 11 TERMINAL (LRECL 80
*  FI 20 TERMINAL
*  FI 21 TERMINAL
*  FI 22 TERMINAL - PI
*  FI 23 TERMINAL
*  FI 24 TERMINAL - LI
*  FI 30 TERMINAL
*  FI 31 TERMINAL - 20
*  FI 32 TERMINAL
*  FI 33 TERMINAL
*  FI 34 TERMINAL - Co
*
*  &TYPE LOAD MODULE FILE NAME (REQUIRED)?
*  &READ VARS &LDM
*  &IF .&LDM = . &GOTO -ERROR
*
*  &TYPE SYMBOL TABLE FILE NAME (REQUIRED)?
*  &READ VARS &SYM
*  &IF .&SYM = . &GOTO -ERROR
*
*  &TYPE SIMULATOR COMMAND FILE NAME (<CR> IF TERMINAL )?
*  &READ VARS &DATA
*
*  FI 12 DISK &LDM MOD1750 A (RECFM FB LRECL 80 BLKSIZE 800
*  &TYPE LOAD MODULE FILE &LDM MOD1750 ON UNIT-12
*  FI 9 DISK &SYM SYM1750 A (RECFM F LRECL 1024 BLKSIZE 1024 DSORG DA
*  FI 9 DUMMY
*  &TYPE SYMBOL TABLE FILE &SYM SYM1750 ON UNIT 9
*
*  &IF .&DATA = . &GOTO -LD
*  FI 11 DISK &DATA COMMANDS A (RECFM F LRECL 80
*  &TYPE SIMUL. COMMAND FILE &DATA COMMANDS ON UNIT 11
*
*  -LD
*  LOAD MSS1 MSS2 MSS3 MSS4 MSS5 MSS6 MSS7 (CLEAR START
*  ERASE TEMP10 ORARY A (NO&TYPE

```

FILE: MSS

EXEC A

PRINCETON UNIVERSITY TIME-SHARING SY.

*

EXIT

*

ERROR

TYPE REQUIRED FILE NAME NOT SUPPLIED

TYPE TRY AGAIN /

EXIT 1

```

*
*           M S S X           E X E C
*
      &CONTROL ERROR
&TYPE 1750A SIMULATOR - M S S
*       SIMULATES A 1750A MACHINE EXECUTING THE PREVIOUSLY
*       CREATED LOAD MODULE AND SYMBOL TABLE.
*       CREATED 12-7-81 BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*
      &IF &1 = . &GOTO -ERROR
      FI * CLEAR
      FI 5 TERMINAL
      FI 2 TERMINAL
      FI 6 TERMINAL
      FI 9 DISK PEKHLF HELP A (RECFM FB LRECL 80 BLKSIZE 800
      FI 7 DISK PEKERR HELP A (RECFM FB LRECL 80 BLKSIZE 800
      FI 10 DISK TEMP10 ORARY A (RECFM F LRECL 1024 BLKSIZE 1024 DSORG DA
      FI 10 DISK TEMP10 ORARY A (XTENT 1024
      FI 11 DISK &1 COMMANDS A (RECFM F LRECL 80
*
      FI 12 DISK &1 MOD1750 A (RECFM FB LRECL 80 BLKSIZE 800
*       &TYPE LOAD MODULE FILE &1 MOD1750 ON UNIT-12
      FI 9 DISK &1 SYM1750 A (RECFM F LRECL 1024 BLKSIZE 1024 DSORG DA
*       &TYPE SYMBOL TABLE FILE &1 SYM1750 ON UNIT 9
      FI 20 TERMINAL
      FI 21 TERMINAL
      FI 22 TERMINAL
      FI 23 TERMINAL
      FI 24 TERMINAL
      FI 30 TERMINAL
      FI 31 TERMINAL
      FI 32 TERMINAL
      FI 33 TERMINAL
      FI 34 TERMINAL
*
      LOAD MSS1 MSS2 MSS3 MSS4 MSS5 MSS6 MSS7 (CLEAR START
*
      &EXIT
*
      -ERROR
      &TYPE REQUIRED FILE NAME NOT SUPPLIED
      &TYPE TRY AGAIN
      &EXIT 1

```

FILE: ALR EXEC A

PRINCETON UNIVERSITY TIME-SHARING SY

*
* A L R E X E C
*
*TYPE 1750A SIMULATOR - ALR (ASSEMBLE, LINK, RUN)
* EXECUTES EXEC'S DESCRIBED ELSEWHERE.
* CREATED BY JOHN RIORDAN OF MIKROS SYSTEMS CORP.
*
*CONTROL ERROR
*EXEC MSAX &1
*EXEC MSLX &1
*EXEC MSSX &1

FORTRAN Code for Implementation of
PI/PO and CI/CO Instructions

PTIONS:

EFFECT: NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTODEL(NONE)

SOURCE EBCDIC NOLIST NODECK OBJECT NOMAP NOFORMAT GOSTMT NOXREF NUALC

SUBROUTINE INICMD,VAL)

```

C----->
C
C   PURPOSE:      PERFORM INPUT FROM DEVICES
C   ARGUMENTS:    CMD      - INPUT COMMAND
C                 VAL      - VALUE TO BE RECEIVED FROM DEVICE
C   PROGRAMMER:   NC BAKER
C   DATE CREATED:  7 JUL 1980
C   CHANGE HISTORY: *NONE*
C   SIDE EFFECTS:  *NONE*
C   REMARKS:      THIS IS A STUB.
C                 IT IS BEING FILLED IN BY J. RIORDAN OF
C                 MIKROS SYSTEMS CORP.
C                 I AM ASSUMING Q(FOR PI,PU COMMANDS) THAT
C                 THE SECOND HEX DIGIT REFERS TO THE CHANNEL
C                 NUMBER (0-3).
C                 WRITTEN 12-18-81
C
C   CALLS:        *NONE*

```

```

IMPLICIT INTEGER(A-Z)
EXTERNAL AND,OR,SHIFT,XOR
INTEGER CMD,VAL
INTEGER R(16)
INTEGER SW(7)
INTEGER CS(4)
DOUBLE PRECISION TIMVAL,TIMRAT,TIMDR
COMMON/QTIMER/TIMVAL(3),TIMRAT(3),TIMDR(3),TIMFL(3),TIMFLD(3)
COMMON/QQ/LOP,ROP,NBLNXT,ATYPE,OPM
COMMON/QER/QERTP,QENPCB,QERCLT,QERHLT,QENPK,QERLUP,ERRUOC,ERRXPU,
-ERRBPT,ERRINN,ERRFPJ,ERRFPU,ERRUPM,ERRICF,ERRUCF,ERRINT,ERRPRI
COMMON/QMKEY/M,IPR,OPR,RQM,MPRAM
COMMON/EVENT/FLUV,QQQQ1,FLUV,QQQQ2,FXOV,QQQQ3,INTRPT,QQQQ4,RTN,QQQ
-Q5
COMMON/QVAL/QVALS,QQQQ6(2),EC,DEBUB,FT,MK,P1,LSTINT,INTREN,NOMEN,M
-PEN,DMAEN,FC,IC,IOIC1,IOIC2,MFSR,DUB,DIW,PUB,NEXT,GOMAX,R0,R1,R2,R
-3,R4,R5,R6,R7,R8,R9,RA,RE,RC,RD,RE,RF,C,P,Z,N,RESO,PS,AS,QQQQ7(1),
-4L,MU,QQQQ8(1),QQQQ9(1),QQQQ10(1),ROMTOP,QQQQ11(1),TIMER,TIMEB,G
-U,QBASE,QLOOP,FPI,NORMCK,CONSOL,IMA,IOIC,PAGES,TIMERS,SRUM,LUCKEY,
-PRDT,DMA,DISCRT,MFSR,SGO
COMMON/DIALOG/INSTR(1),TOSTOP,TOSNAP,INTQ,FROM,INSTEP
COMMON/QUNITS/QUINU,QUUFU,QLDQU,QSYMU,QDISKU,GRAMU,
-QDOU,QPLTU,QINPU(5),QOUTPU(5)
COMMON/IOHAND/ICUR(5),OCUR(5),INBUF(5,80),OUTBUF(5,80)
COMMON/VALSTK/QVLS12,QVLPTR,QVLSK(7,30)
EQUIVALENCE(R(1),R0)
EQUIVALENCE(SW(1),CS(1))
EQUIVALENCE(CS(1),C)
EQUIVALENCE(R10,RA)
EQUIVALENCE(R11,RB)
EQUIVALENCE(R12,RC)
EQUIVALENCE(R13,RD)
EQUIVALENCE(R14,RE)
EQUIVALENCE(R15,RF)
EQUIVALENCE(QBASE,HC)

```



```

3.0 (01 MAY 80)      IN      SYSTEM/370 FORTRAN H EXTENDED (ENHANCED)      DATE
EQUIVALENCE(BASE1,RO)
EQUIVALENCE(BASE2,RE)
EQUIVALENCE(BASE3,RF)
EQUIVALENCE(SP,RF)
INTEGER HEXRAY(4),N,CHNLNO
INTEGER BLANK
DATA BLANK /' ' /
C      CHNLNO = CHANNEL NUMBER (0-3)
C      PICUR = PROGRAMMED INPUT CURSOR
C
      IF (CMD.EQ.49152) GO TO 101
C      PROGRAMMED INPUT
C      CALL HEX TO CONVERT CMD TO HEX VALUES TO PICK OUT CHANNEL NO.
      CALL HEXIT(CMD,HEXRAY)
      CHNLNO = HEXRAY(3) + 1
      GO TO 102
101  CHNLNO = 5
102  IF((ICUR(CHNLNO).GT.0).AND.(ICUR(CHNLNO).LT.81)) GO TO 103
      UNIT = QINPU(CHNLNO)
      WRITE(UNIT,1001)
1001  FORMAT(/,'> CONSOLE INPUT')
      READ(UNIT,1002,END=99)(INBUF(CHNLNO,N),N=1,80)
1002  FORMAT(80A1)
      ICUR(CHNLNO) = 1
103  FIRST = ICUR(CHNLNO)
      DO 10 N=FIRST,80
          IF (INBUF(CHNLNO,N).EQ.BLANK) GO TO 10
          CALL LOOKUP(INBUF(CHNLNO,N),VAL,1)
          ICUR(CHNLNO) = N + 1
      RETURN
10  CONTINUE
99  VAL = 13
      ICUR(CHNLNO) = 0
      RETURN
C
      END
EFFECT#NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTODBL(NONE)
EFFECT#SOURCE EBCDIC NOLIST NODECK OBJECT NOMAP NUFORMAT GOSTMT NOXREF NOALC
*      SOURCE STATEMENTS =      61. PROGRAM SIZE =      766. SUBPROGRAM NAME =
*      NO DIAGNOSTICS GENERATED
OF COMPI LATION *****                                236K BYTES OF COB

```

PTIONS:

EFFECT: NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTOOBL(NONE)

SOURCE EBCDIC NOLIST NODCK OBJECT NOMAP NOFORMAT GOSTMT NOXREF NCALC

SUBROUTINE OUT(CMD,VAL)

```

C----->
C
C      PURPOSE:      PERFORM OUTPUT TO DEVICES
C      ARGUMENTS:    CMD      - OUTPUT COMMAND
C                   VAL      - VALUE TO BE OUTPUT
C
C      PROGRAMMER:   MC BAKER
C      DATE CREATED:  7 JUL 1980
C      CHANGE HISTORY: *NONE*
C      SIDE EFFECTS:  *NONE*
C      REMARKS:       THIS IS A SUB
C                   IT IS BEING FILLED IN BY J. RIORDAN OF
C                   MIKROS SYSTEMS CORP. WRITTEN 12-18-81
C
C      CALLS:        *NONE*
C-----

```

```

IMPLICIT INTEGER(A-Z)
EXTERNAL AND,OR,SHIFT,XOR
INTEGER CMD,VAL
INTEGER R(16)
INTEGER S(7)
INTEGER CS(4)
DOUBLE PRECISION TIMVAL,TIMRAT,TIMDR
COMMON/QTIMER/TIMVAL(3),TIMRAT(3),TIMDR(3),TIMFL(3),TIMFLD(3)
COMMON/QO/LOP,RUP,NBLNXT,ATYPE,OPM
COMMON/QER/QERTRP,QERPCB,QERCLT,QERHLT,QERPK,QERLOP,ERRUOC,ERRXPU,
-ERRBPT,ERRINN,ERRFPU,ERRFPU,ERRRPM,ERRICF,ERRUCF,ERRINT,ERRPRI
COMMON/QMKEY/M,IPR,OPR,RDM,MPRAM
COMMON/EVENT/FLOV,QQQQ1,FLUV,QQQQ2,FXUV,QQQQ3,INTRPT,QQQQ4,RIN,QQQI
-QQ
COMMON/QVAL/QVALS,QQQQ6(2),EC,DEBUG,FT,MK,PI,LSINT,INTREN,RUMEN,M
-PEN,UMAEN,FC,IC,IOIC1,IOIC2,MFSR,DUB,DIW,PUB,NEXT,QUMAX,R0,R1,R2,R
-3,R4,R5,R6,R7,R8,R9,RA,RB,RC,RD,RE,RF,C,P,Z,N,RESO,PS,AS,QQQQ7(1),
-ML,MU,QQQQ8(1),QQQQ9(1),QQQQ10(1),RUMTOP,QQQQ11(1),TIMERA,TIMERB,G
-Q,QBASE,QLLOOP,FPI,NORMCK,CONSOL,IMA,IOIC,PAGES,TIMERS,SROM,LOCKEY,
-PRUT,DMA,DISCRT,SMFSR,SGU
COMMON/DIALOG/INST(1),TOSTOP,TOSNAP,INTU,FRUM,INSTEP
COMMON/QUITS/QUINU,QUUTU,QLDQU,GSYMU,QDISKU,GRAMU,
-QQUU,QPLTU,QINPU(5),QOUTPU(5)
COMMON/IOHAND/ICUR(5),UCUR(5),INBUF(5,80),OUTBUF(5,80)
COMMON/VALSTK/QVLS1Z,QVLPTR,QVLSTK(7,30)
EQUIVALENCE(R(1),R0)
EQUIVALENCE(S(1),CS(1))
EQUIVALENCE(CS(1),C)
EQUIVALENCE(R10,RA)
EQUIVALENCE(R11,RB)
EQUIVALENCE(R12,RC)
EQUIVALENCE(R13,RD)
EQUIVALENCE(R14,RE)
EQUIVALENCE(R15,RF)
EQUIVALENCE(BASE0,RC)
EQUIVALENCE(BASE1,RD)
EQUIVALENCE(BASE2,RE)
EQUIVALENCE(BASE3,RF)
EQUIVALENCE(SP,RF)

```

3.0 (01 MAY 80) OUT SYSTEM/370 FORTRAN H EXTENDED (ENHANCED) DATE

INTEGER HEXRAY(4),N,CHNLNO
INTEGER BLANK
DATA BLANK/' '/

C*

IF(CMD.EQ.16384) GO TO 101
CALL HEXIT(CMD,HEXRAY)
CHNLNO = HEXRAY(3) + 1
GO TO 102

101 CHNLNO = 5

102 IF((VAL.NE.13).AND.(OCUR(CHNLNO).GT.0)
- .AND.(OCUR(CHNLNO).LE.80)) GO TO 9

UNIT = OUTPUT(CHNLNO)
WRITE(UNIT,1001)(OUTBUF(CHNLNO,N),N=1,80)

1001 FORMAT(80A1)

VAL = 0

OCUR(CHNLNO) = 1

DO 7 I = 1,80

7 OUTBUF(CHNLNO,I) = BLANK

RETURN

9 CALL LOOKUP(OUTBUF(CHNLNO,OCUR(CHNLNO)),VAL,2)

OCUR(CHNLNO) = OCUR(CHNLNO) + 1

RETURN

END

EFFECT*NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTOBL(NONE)

EFFECT*SOURCE EBCDIC NOLEST NODECK OBJECT NOMAP NOFORMAT GOSTMT NOXREF NOALC

* SOURCE STATEMENTS = 55 PROGRAM SIZE = 674 SUBPROGRAM NAME =

* NO DIAGNOSTICS GENERATED

OF COMPILE ***

240K BYTES OF COR

PTIONS:

EFFECT: NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTOOBL(NONE)

SOURCE EBCDIC NOLIST NODECK OBJECT NOMAP NOFORMAT GOSTMT NOXREF NOALC

```

C
C      SUBROUTINE HEXIT(DECNUM,ARRAY)
C      THIS ROUTINE PRODUCES A PSEUDO-HEX ARRAY FOR A DECIMAL
C      INPUT VALUE. FOR EXAMPLE, THE DECIMAL VALUE 1023, WHICH
C      IN HEX IS 3FF IS PRODUCED AS "3 15 15" BUT IN REVERSE
C      ORDER (IE. ARRAY(1)=15,ARRAY(2)=15,ARRAY(3)=3, AND
C      ARRAY(4)=0).

```

```

      INTEGER DECNUM,ARRAY(4),NUMB,1,LSTNUM

```

```

C      THE INPUT VALUE DECNUM IS NOT ALTERED.
C      A COPY IS MADE TO ISOLATE IT.

```

```

      LSTNUM = DECNUM

```

```

      NUMB = LSTNUM

```

```

      DO 10 I = 1,4

```

```

        NUMB = NUMB/16

```

```

        ARRAY(I) = LSTNUM - (16*NUMB)

```

```

10      LSTNUM = NUMB

```

```

      RETURN

```

```

      END

```

EFFECT*NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTOOBL(NONE)

EFFECT*SOURCE EBCDIC NOLIST NODECK OBJECT NOMAP NOFORMAT GOSTMT NOXREF NOALC

* SOURCE STATEMENTS = 10 PROGRAM SIZE = 302. SUBPROGRAM NAME =

* NO DIAGNOSTICS GENERATED

OF COMPILATION *****

252K BYTES OF COB

DATE _____

ACTIONS:

```

EFFECT: NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTOOBL(NONE)
SOURCE EBCDIC NOLIST NODECK OBJECT NUMAP NOFORMAT GOSTMT NOXREF NOALLC

```

```

C      SUBROUTINE LOOKUP(CHAR,NUMB,WHICH)
C
C      THIS ROUTINE SEARCHES ONE TABLE FOR THE INPUT VALUE
C      AND RETURNS ITS ASSOCIATED VALUE. IF WHICH = 1, THEN THE
C      THE TRANSFORM IS CHAR > HEX; IF WHICH = 2, THE TRANSFORM
C      IS HEX > CHAR. DEFAULTS ARE RETURNED IF NO MATCH IS FOUND.
C
C      INTEGER TABLEA(64),TABLEB(64),INDEX,CHAR,NUMB,WHICH
DATA TABLEA /' ','.',':','/','.0','.1','.2','.3','.4','.5','.6','.7','.8','.9',
-          '.',':','<','=','>','.?','.@',
-          'A','B','C','D','E','F','G','H','I','J','K','L','M',
-          'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
-          '-','_','*', '=', '?', '/'
DO 8 INDEX = 1,63
    TABLES(INDEX) = INDEX + 31
CONTINUE
    TABLEB(64) = 0Z
IF (WHICH.EQ.2) GO TO 20

    CHAR > HEX    (INPUT)
DO 10 INDEX = 1,64
    IF (CHAR.NE.TABLEA(INDEX)) GO TO 10
    NUMB = TABLES(INDEX)
    RETURN
10 CONTINUE
    NUMB = -1
    RETURN

C      HEX > CHAR    (OUTPUT)
20 DO 30 INDEX = 1,64
    IF (NUMB.NE.TABLEB(INDEX)) GO TO 30
    CHAR = TABLEA(INDEX)
    RETURN
30 CONTINUE
    NOTE: TABLEA(32) = '*'        DEFAULT VALUE
    CHAR = TABLEA(32)
    RETURN
END
```

```

EFFECT*NAME(MAIN) OPTIMIZE(3) LINECOUNT(60) SIZE(MAX) AUTOOBL(NONE)
EFFECT*SOURCE EBCDIC NOLIST NUDECK OBJECT NUMAP NOFORMAT GOSTMT NUAREF NUALL
* SOURCE STATEMENTS = 26+ PROGRAM SIZE = 1012. SUBPROGRAM NAME = I
* NO DIAGNOSTICS GENERATED
OF COMPILATION *****
248K BYTES OF CODE

```

* NO DIAGNOSTICS THIS STEP

BIBLIOGRAPHY

McDonnell Douglas Astronautics Co. 'MIL-STD-1750A Support Software Maintenance Manual, Contract F33657-74-C-0727' and the 'User Guide - Support Software Package'.

THE AFWAL MIL-STD-1589B JOVIAL J73 COMPILER

Mike Burlakoff

Support Systems Branch, AFWAL/AAAF
Wright-Patterson Air Force Base, Ohio 45433

Introduction

The Air Force Wright Aeronautical Laboratories (AFWAL) JOVIAL J73 compiler was developed for the Avionics Laboratory by Software Engineering Associates. This compiler was developed by modifying an existing JOVIAL J73/I compiler to first compile MIL-STD-1589A code, and then upgraded to MIL-STD-1589B. The compiler was re-hosted from the AFWAL DEC-10 to the Aeronautical Systems Division (ASD) IBM 370 computer. During the development process, MIL-STD-1750 and MIL-STD-1750A code generators were developed. The source code of the compilers has recently been translated from J73/I to J73 for both hosts and all targets.

History of the Development

Between 1974 and 1976, an Air Force project was initiated to develop a JOVIAL compiler for the Avionics Laboratory Digital Avionics Information System (DAIS) program. The contractor for this effort was Computer Sciences Corporation (CSC) and the compiler developed was a JOVIAL J73 level I or J73/I compiler.

The techniques used to develop the J73/I compiler were as follows: 1) A compiler was written in 1108 assembly language, and hosted on a UNIVAC 1108 computer, to compile JOVIAL J73/S code which is a subset of JOVIAL J73/I; 2) The compiler source code was then written in J73/S and compiled with the original 1108 assembly language version; 3) A code generator was developed on the UNIVAC 1108 machine to produce code for the DECsystem-10. This began the retargeting and re-hosting process; 4) The compiler was then re-hosted from the UNIVAC 1108 to the DECsystem-10 and upgraded to the JOVIAL J73/I language. The compiler source was then written in J73/I; 5) Code generators were then added for the AN/AYK-15 and DELCO M362F computers. The AN/AYK-15 computer is considered the predecessor to the MIL-STD-1750 computer and is sometimes referred to as the Hot Bench Computer (HBC).

J73/I Applications

With the J73/I compiler now hosted on and targeted for the DECsystem-10 and also targeted for the AN/AYK-15 computer, the compiler received a great deal of use in large scale avionics support software systems and in Avionics Executive and Mission Software systems using MIL-STD-1553A/B protocol. This extensive use of the J73/I compiler for developing code on a large host and executing on both the host and avionics (AN/AYK-15) processors resulted in a very mature compiler. The compiler was also used by a number of other organizations and was the tool used for the JOCIT compiler development effort.

MIL-STD-1589B Development

Between 1979 and 1980, Software Engineering Associates (SEA) of Torrance, California developed the MIL-STD-1589B version of the compiler for the AFWAL Support Software Group of the System Avionics Division. Gil Weisbord and Mike Littlejohn of SEA were the principal developers. Bob Engimann of TRW has been performing integration, problem analysis and configuration control of the compilers.

The techniques employed to accomplish this MIL-STD-1589B development were: 1) Modify the J73/I compiler to compile JOVIAL J73 (MIL-STD-1589A) programs on the DECsystem-10; 2) Add code generators for the MIL-STD-1750 and MIL-STD-1750A hardware; 3) Re-host from the DECsystem-10 to an IBM 370 with IBM, 1750, 1750A code generators, and 4) Upgrade the MIL-STD-1589A compiler to the MIL-STD-1589B version.

The IBM 370 re-host was done by AFWAL at the request of RADC to serve as an interim or backup compiler to the RADC contracted JOCIT effort. The IBM re-host retained the same high level of efficiency and maturity as the DEC-10 version.

J73 Applications

The JOVIAL J73 compiler is presently being used in the Avionics Laboratory by the System Integration Branch, AAAS (formerly known as DAIS). It is being used in converting support software, the Avionics Executive and Mission Software, from JOVIAL J73/I to JOVIAL J73 on the DECsystem-10 and 1750 hardware. The compiler has also been used in a number of other Avionics Laboratory software development projects such as development of the ALINKS 1750A Linker (TRW) and the JOVIAL Interactive Debugger (TRW).

Other Air Force agencies have used the compiler to develop software systems such as the Programming Support Library, Code Auditor and the JOVIAL Automatic Verification System.

The Avionics Laboratory has distributed the compiler to approximately 63 outside agencies. It is currently being used throughout DOD by developers of JOVIAL software. Some of the larger Air Force programs using the compiler are LANTIRN, F-16 and the KC-135 Update Program.

This product is currently the only available MIL-STD-1589B JOVIAL compiler and is being used as a baseline for a number of other JOVIAL compiler/code generator developments. Code generators for the Z80002, TI990 and TI9900 series have been developed and the INTEL 8086/8087 Code Generator (IBM 370 host) is being developed using this compiler. The J73 compiler is also being re-hosted on a VAX 11/780 by the Embedded Computer Standardization Program Office (ECSP0). Because of its wide distribution and use, it is expected that other developments will evolve from this baseline.

Compiler Hosts/Targets/Features

The compiler is hosted on the DECsystem-10 and targeted for the DEC-10, MIL-STD-1750, MIL-STD-1750A, AN/AYK-15, and the DELCO M362F MAGIC computer. The average CPU memory required on the DEC-10 for the compiler is approximately 75K words and the average compilation speed is approximately 2400 lines/minute and 1100 statements/minute.

The other host is the IBM 370 system. It is targeted for the IBM 370, MIL-STD-1750 and MIL-STD-1750A hardware. The average CPU memory required on the IBM 370 is approximately 500K bytes and the average compilation speed is approximately 2100 lines/minute and 1000 statements/minute.

Compiler Inputs

The compiler supports multiple COMPOOL input files as defined by the language specification.

The compiler supports multiple COPY files under the DEC-10 operating system and the IBM operating system.

The compiler processes multiple source programs with a single compiler invocation. For each input source program, a separate object module is produced.

Compiler Outputs

The compiler produces object modules in the formats defined by the DEC-10 operating system, the DEC-10 JOVIAL Interactive Debugger (JID), the IBM 370 operating system, the AN/AYK-15, DELCO M362F, 1750 and 1750A linkage editors. On option, (grammar checking mode), the object module is not produced.

For a COMPOOL compilation, the compiler produces a COMPOOL output file in a format suitable for subsequent use by the compiler as a COMPOOL input file.

All compiler listable outputs include on each page the version number of the compiler, the time and date of compilation, and the page number.

The compiler produces a source program listing interspersed with diagnostic messages for errors discovered during the compilation. A narrative of the error is given along with the severity level, error number, and point of occurrence. Each line of the source program has appended to it in the source listing, the statement number of the lowest numbered statement which appears on the line. Statement numbering begins at 1. Statements are delimited by semicolon and END.

The source listing can be printed in well structured form with statements that appear at the same BEGIN END level starting in the same column in the listing.

The compiler can produce an edited object code listing containing both octal (or hex, if applicable) and symbolic form for each machine language instruction produced. The first instruction produced for a statement includes in the object code listing the source program statement number from which the instruction was produced. For the IBM 370, an option exists to provide an assembly module which can actually be assembled by the IBM assembler to produce relocatable object code, which can be linked and executed.

The compiler produces a set/used listing containing an entry for each name referred to in the program. Entries are sorted by name. Printed with each name is the following:

1. An indication of the type of the named entry (e.g., label, table)
2. An indication of the scope of definition of the named entity
3. The relative location assigned to the named entity by the compiler
4. The arithmetic type (e.g., integer, character) of the named entity, if applicable.
5. The bit displacement within a word of the named entity, if applicable
6. The size of the named entity, if applicable
7. The sequence number of the source statement in which the named entity is declared
8. The number of each statement which refers to the named entity
9. For references to variables, an indication that the variables are set or used

A source program statistics summary can be produced. The statistics summary includes the number of occurrences in the source program of the following language forms: Characters, lines, symbols, declarations (by declaration type), statements (by statement type), and comments.

The DEC-10 hosted compiler produces a JOVIAL Interactive Debugger Symbol Table, Name Table, Breakpoint Space and other software features needed by the JOVIAL Interactive Debugger (JID). (See Oct 81 Newsletter). JID provides a capability for debugging at the JOVIAL high order language level. JID is available for the DEC-10 host and target only.

Current Status and Future Plans

The MIL-STD-1589B versions (J73/I source) have been formally validated on both the DEC-10 and IBM 370 hosts for all targets.

The source code for the DEC-10 hosted compiler with DEC, 1750, 1750A and AN/AYK-15 code generators has been translated from J73/I to J73. An initial

version of this compiler has been released to ASD/AXS. The compiler and all code generators have been formally validated by the Language Control Facility.

The IBM hosted compiler with IBM, 1750 and 1750A code generators has also been translated from J73/I to J73. Final integration and testing was completed in Apr 82.

Distribution of the compilers has been transitioned from AFWAL/AAAF to the Aeronautical Systems Division, ECSPD (ASD/AXS). ASD/AXS will also assume configuration control and maintenance of the compilers from AFWAL/AAAF in mid-FY82. Further information regarding configuration management planning, problem reporting and compiler enhancements can be obtained by contacting ASD/AXS.

AFWAL

MIL-STD 1589B

JOVIAL J73 COMPILER

MIKE BURLAKOFF

AFWAL/AAAF-2/WPAFB, OH 45433

**AFWAL MIL--STD 1589B JOVIAL J73 COMPILER
EVOLUTION OF THE COMPILER**

- **ORIGINAL AF PROGRAM:**
- **DEVELOP A JOVIAL J73/1 COMPILER FOR THE DIGITAL AVIONICS
INFORMATION SYSTEM (DAIS) PROGRAM (1974-1976)**
- **CONTRACTOR: COMPUTER SCIENCES CORP (CSC)**
- **COST: 1,050K**

AFWAL MIL--STD 1589B JOVIAL J73 COMPILER EVOLUTION OF THE COMPILER (CONT)

• J73/I DEVELOPMENT TECHNIQUES:

- 1) ON UNIVAC 1108, A COMPILER WAS WRITTEN IN 1108 ASSEMBLY LANGUAGE TO
COMPILE A JOVIAL DIALECT J73/S (SUBSET OF J73/I)
- 2) COMPILER SOURCE WAS WRITTEN IN J73/S AND COMPILED WITH ASSEMBLY
LANGUAGE VERSION
- 3) CODE GENERATOR WAS DEVELOPED ON UNIVAC 1108 TO PRODUCE DEC--10
CODE (RE--TARGET)
- 4) REHOSTED FROM UNIVAC TO DEC--10 AND UPGRADED TO J73/I (RESULTING
COMPILER NOW IN J73/I)
- 5) ADDED AN/AYK--15, DELCO M362F CODE GENERATORS

**AFWAL MIL--STD 1589B JOVIAL J73 COMPILER
EVOLUTION OF THE COMPILER (CONT)**

- **J73/I APPLICATIONS: (DEC--10 HOST, DEC--10, AN/AYK--15 TARGETS)**

- **LARGE AVIONICS SUPPORT SOFTWARE SYSTEMS (DAIS)**

- **AVIONICS EXECUTIVE AND MISSION SOFTWARE SYSTEMS USING**

MIL--STD 1553A/B PROTOCOL (DAIS)

- **THEREFORE, A MATURE COMPILER WITH CODE DEVELOPED ON A LARGE HOST,
AND EXECUTED ON HOST AND AVIONICS PROCESSORS**

AFWAL MIL-STD 1589B JOVIAL J73 COMPILER EVOLUTION OF THE COMPILER (CONT)

MIL-STD 1589B DEVELOPMENT:

- FOR AFWAL/AAAF BY SOFTWARE ENGINEERING ASSOCIATES (SEA). (1979-1980)
- COST: 583.3K

MAJOR TASKS:

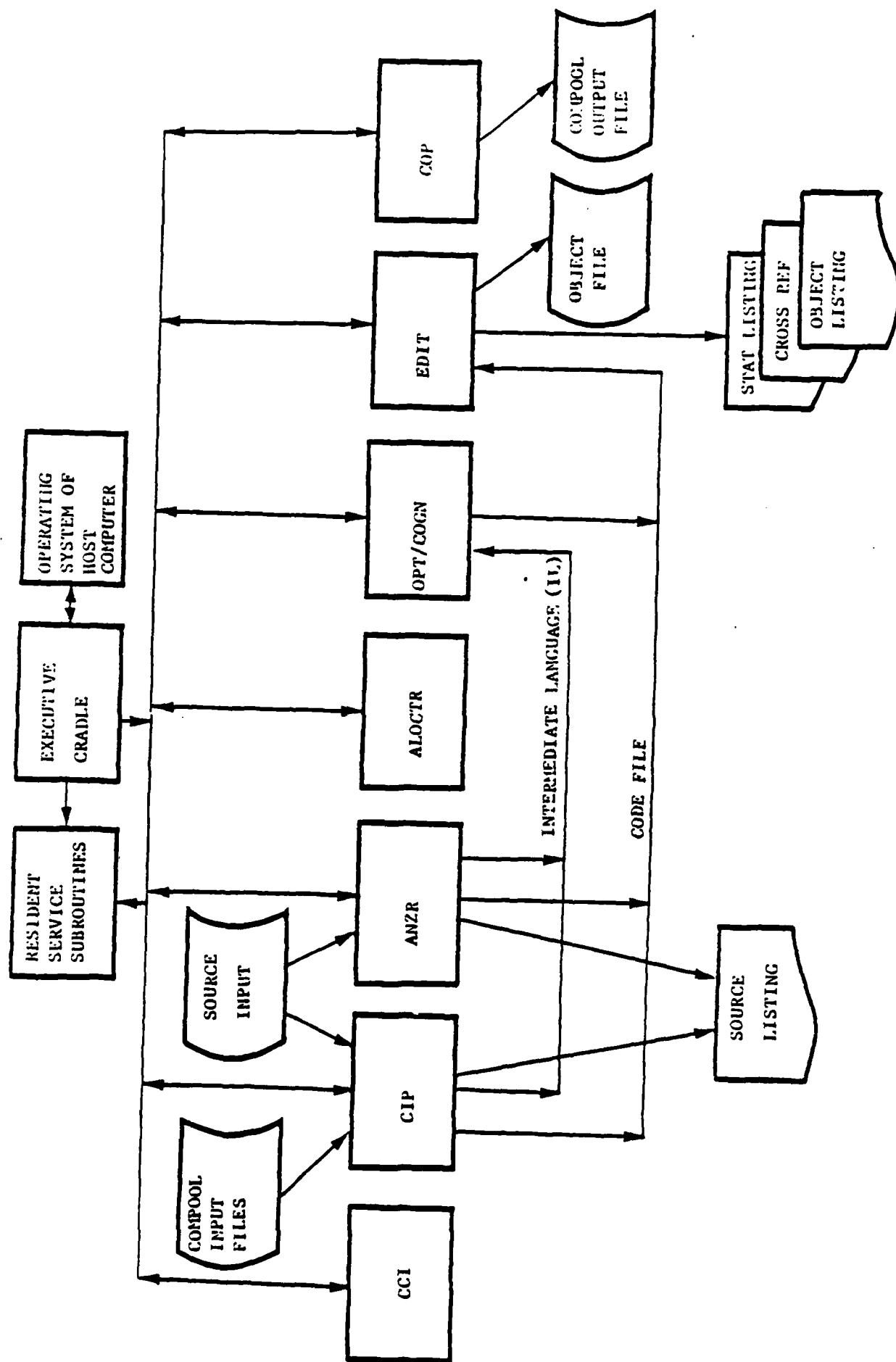
- 1) MODIFY J73/1 COMPILER TO COMPILE J73 PROGRAMS ON THE DEC-10
- 2) ADD MIL-STD 1750, MIL-STD 1750A CODE GENERATORS
- 3) RE-HOST TO ITTEL (IBM 370) WITH IBM, 1750, 1750A CODE GENERATORS
- 4) UPGRADE FROM MIL-STD 1589A TO MIL-STD 1589B.

AFWAL MIL—STD 1589B JOVIAL J73 COMPILER

AFWAL MIL—STD 1589B APPLICATIONS

- **AFWAL/AAAS — SYSTEMS INTEGRATION BRANCH (DAIS) HAS CONVERTED SUPPORT SOFTWARE, EXECUTIVE AND MISSION SOFTWARE FROM J73/I TO J/73 (DEC—10 AND 1750 HARDWARE EXECUTION)**
- **LARGE SUPPORT SOFTWARE SYSTEMS (DEBUGGER, LINKER, ETC)**
- **AIR FORCE PROGRAMS (LANTIRN, F—16, KC—135, ETC)**
- **IBM AND DEC—10 COMPILERS DISTRIBUTED BY AFWAL TO APPROXIMATELY 63 USERS**

AFWAL MIL-STD 1589B JOVIAL I73 COMPILER OPERATING STRUCTURE



AFWAL MIL-STD1589B JOVIAL J73 COMPILER

HOSTS/TARGETS/FEATURES

• DEC-10

- TARGETS: DEC-10, MIL-STD 1750, MIL-STD 1750A, AN/AYK-15, DELCO M362F**
- AVERAGE CPU MEMORY REQUIRED: 75K WORDS**
- AVERAGE COMPILATION SPEED: 2400 LINES/MINUTE**
- 1100 STATEMENTS/MINUTE**

• ITEL (IBM370)

- TARGETS: IBM 370, MIL-STD 1750, MIL-STD 1750A**
- AVERAGE CPU MEMORY REQUIRED: 450K BYTES**
- AVERAGE COMPILATION SPEED: 2100 LINES/MINUTE**
- 1000 STATEMENTS/MINUTE**

AFWAL MIL--STD 1589B JOVIAL J73 COMPILER

STATUS/PLANS

- **MIL--STD--1589B FORMAL VALIDATIONS COMPLETED:**

- **J73/I SOURCE VERSIONS:**

IBM: IBM, 1750, 1750A CODE GENERATORS

DEC: DEC, 1750, 1750A, MAGIC CODE GENERATORS

- **INITIAL J73 SOURCE VERSION:**

DEC: DEC, 1750A

- **TRANSLATION OF COMPILER SOURCE FROM J73/I TO J73 (BOOTSTRAP) COMPLETED FOR BOTH HOSTS**

- **CONFIGURATION CONTROL, MAINTENANCE, ENHANCEMENTS, AND SOFTWARE DISTRIBUTION TRANSITIONED TO ECSP0**

END

FILMED

3-85

DTIC