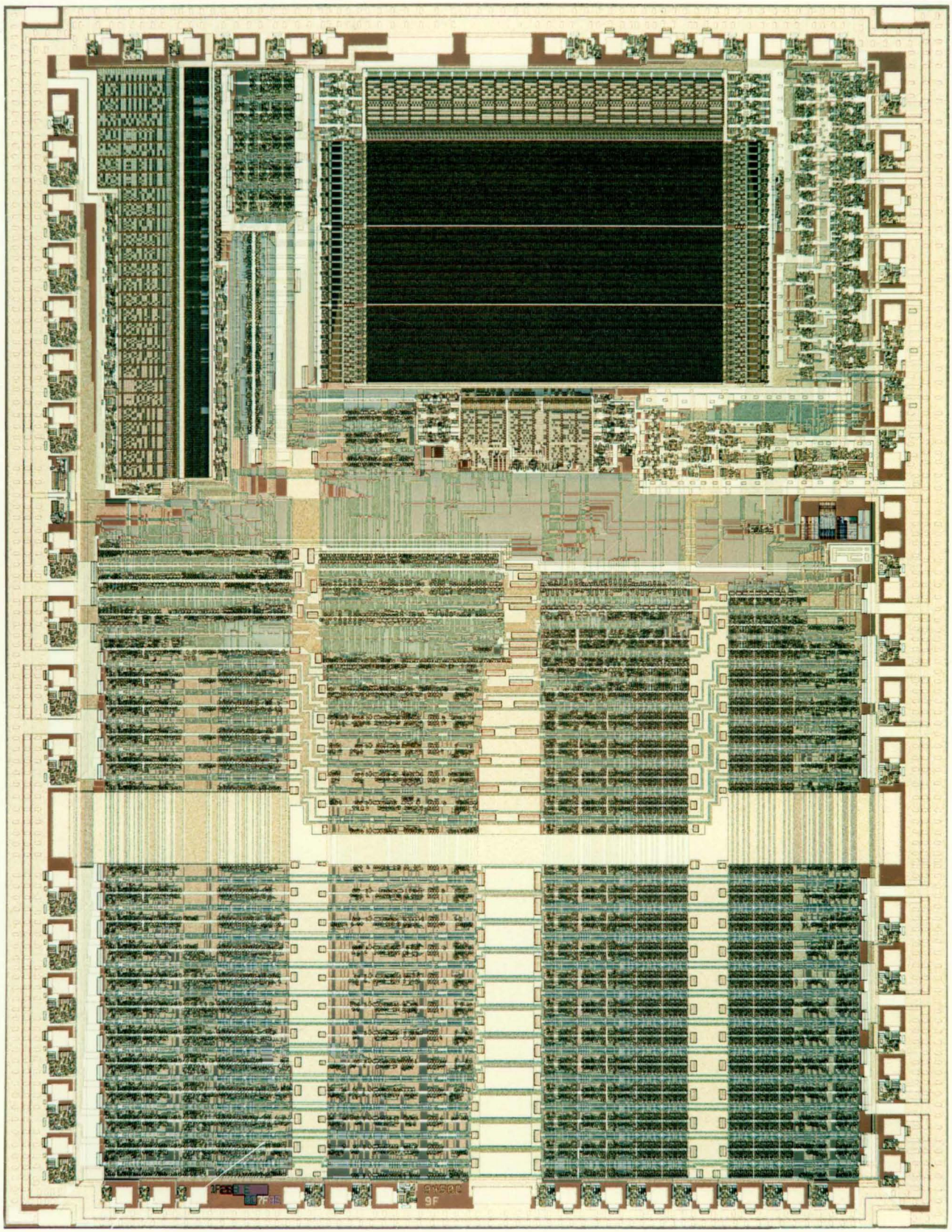
FAIRCHILD

A Schlumberger Company

TECHNICAL SEMINAR

F9450



FAIRCHILD F9450

16-BIT MICROPROCESSOR WITH COMPLETE MIL-STD-1750A ISA

FAIRCHILD
A Schlumberger Company

F9450
SEMINAR

FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

MICROPROCESSOR DIVISION

450 NATIONAL AVENUE

MT. VIEW, CA 94043

(415) 962-3888

TWX 910 379 5091

THE FAIRCHILD F9450 WAS DEVELOPED IN RESPONSE TO THE NEED FOR

HIGH SPEED
REAL TIME
SOPHISTICATED
SIXTEEN BIT
STANDARDIZED (Mil.Std.1750A)

MICROPROCESSOR ENVIRONMENT

THE FAIRCHILD F9450 MEETS ALL DESIGN GOALS, UTILIZING LATEST ISOPLANAR INTEGRATED INJECTION LOGIC TECHNOLOGY & IMPLEMENTING COMPLETE Mil.Std.1750A(notice 1, 5/82) INSTRUCTION SET ON A SINGLE VLSI CHIP.

---F9450 MICROPROCESSOR FAMILY---

☆F9450 SINGLE CHIP CPU

☆F9450 SUPPORT CHIPS

F9451 Memory Management Unit
F9452 Block Protect Unit

☆DEVELOPMENT ENVIRONMENT

VAX/VMS based assembler/linker/loader
SBC50 IEEE796 based F9450 CPU board
Third Party development equip. support
Third Party PDL & Operating System support

☆F9450

ON CHIP FACILITIES INCLUDE:

- *COMPLETE FLOATING POINT MATH*
- *TWO PROGRAMMABLE TIMERS*
- *16 GENERAL PURPOSE REGISTERS*
- *16 LEVEL INTERRUPT PROCESSOR*
- *COMPREHENSIVE FAULT HANDLER*
- *MULTIPROCESSING SUPPORT*
- *PROCESS SYNCHRONIZATION*
- *2 Meg. WORD ADDRESSING*
- *THRUPUT UP TO 1.5 M.I.P.S.*

BY UTILIZING ADDITIONAL MEMBERS OF THE F9450 PRODUCT FAMILY, ADDITIONAL FACILITIES INCLUDE:

☆*F9451*

MEMORY MANAGEMENT UNIT

16 Meg.Word Addressability

Logical to Physical Translation

Protection of Logical Space

On Chip Page Descriptor Storage

Two Translation Maps

☆F9452

BLOCK PROTECT UNIT

Write protect of CPU memory space

Write protect of DMA memory space

Protection of 1K page space

(Higher granularity protection)

Global protection of memory space until ENABLE

F9450

DEVELOPMENT SUPPORT

F9450
THIRD PARTY SUPPORT PRODUCTS

☆DEVELOPMENT SYSTEMS

TEK, HP

will have full in-system emulator support

☆SOFTWARE TOOLS

JOVIAL

AVAILABLE USAF & SEA, PSS, SOFTECH & ACT. (VAX/VMS)

FORTRAN

AVAILABLE ACT. (VAX/VMS)

MACRO ASSEMBLER & LINKER

AVAILABLE USAF & ACT, PSS, INTERMETRICS, MDC.

SIMULATORS & DEBUGGERS

AVAILABLE USAF, MDC, & PSS.

ADA

UNDER DEVELOPMENT @ BOEING, SOFTECH, WESTINGHOUSE (VAX/VMS)

F9450
THIRD PARTY SUPPORT PRODUCTS
(CONTINUED)

☆OPERATING SYSTEMS

EMBEDDED 'REAL-TIME' & MULTITASKING

VRTX from HUNTER and READY

ESCAPE from PSS

DIGITAL FILTER BENCHMARK

DESCRIPTION

THIS BENCHMARK USES A 16-BIT PROCESSOR TO IMPLEMENT A SPECIFIC EIGHTH-ORDER CASCADED FILTER. ORIGINALLY PUBLISHED IN FEB.'81 IEEE MICRO, IT HAS BEEN USED BY VARIOUS MFG. TO DENOTE PERFORMANCE IN A REAL-TIME ENVIRONMENT.

INPUT	PERFORMS WRITES TO I/O DEVICE, & POLLS DEVICE UNTIL ACKNOWLEDGED
OUTP-LD	MULT. SAMPLES BY CONSTANT, THEN ADDS ANOTHER CONSTANT TO RESULT
DELAY-LD	SHIFTS MEMORY ARRAY TO ANOTHER LOCATION
PRE-LD	PERFORMS MULTIPLE MULTIPLYS AND ADDS
TOTAL	TIME REQUIRED FOR FILTER FROM INPUT TO OUTPUT, (1 sample).

Note:assume no 'wait-states' & immediate acknowledge on query

DIGITAL FILTER BENCHMARK

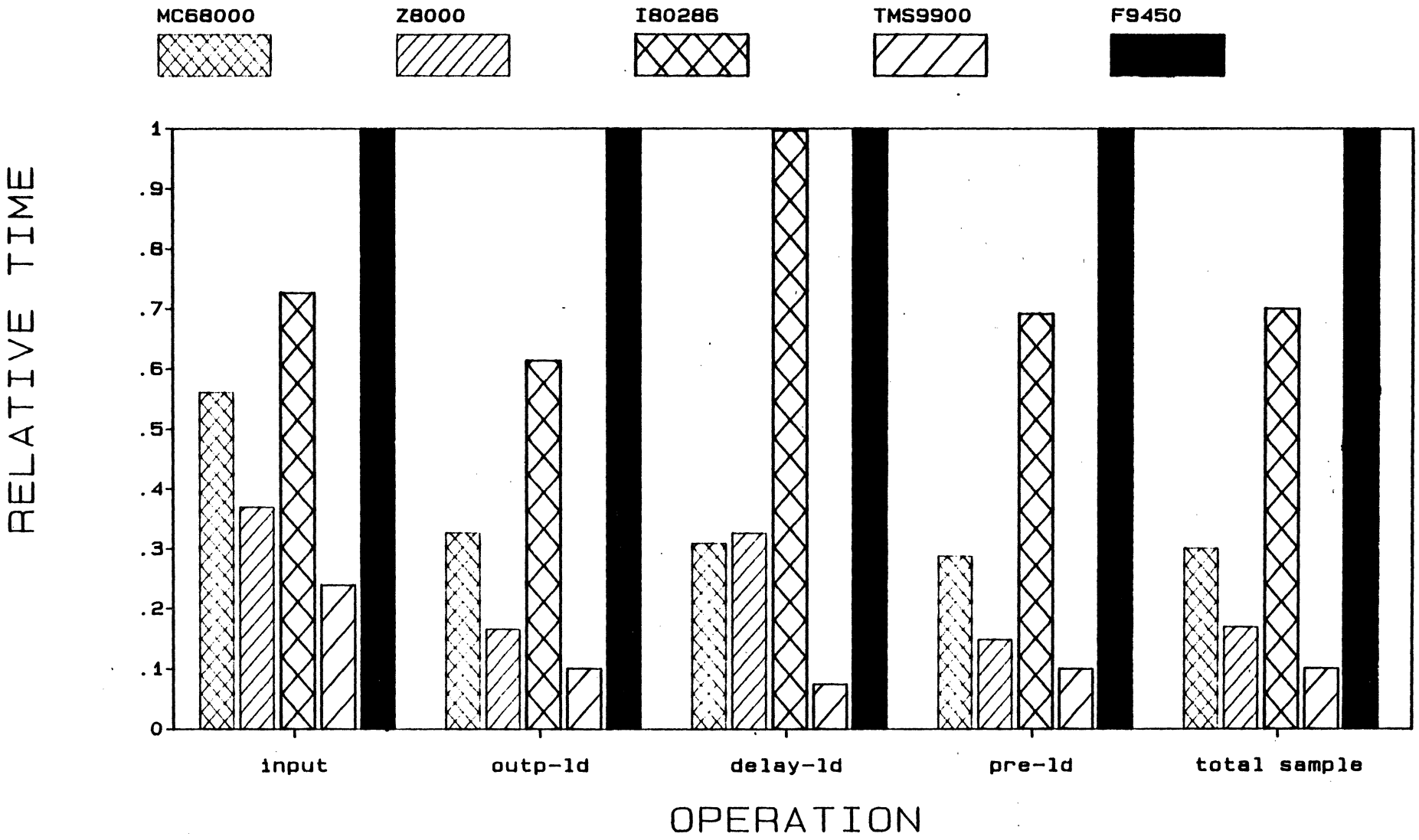
Procedure	mc68000	z8000	i80286	tms9900	F9450
input	10.25	15.50	7.90	24.20	5.75
outp_ld	65.50	127.50	34.60	211.50	21.25
delay_ld	32.00	30.25	9.90	135.80	9.85
pre_ld	194.50	380.25	80.40	559.40	55.15
TOTAL :	327.00	584.00	139.90	1000.00	98.05

NOTE: all times in microseconds

data from: Nagle & Nelson, IEEE MICRO, 2/81, pp23-41

INTEL Corp., iAPX 186, 286 BENCHMARK REPORT, 1982

DIGITAL FILTER
BENCHMARK



Note: data from IEEE MICRO, 2/81 & Intel Corp.
relative time = Normalized, F9450 = 1

NEW PRODUCT PLANS

☆ *SBC50 Complete F9450 system for evaluation & development*

☆ *'C' Compiler*

FUTURE OPTIONS INCLUDE

☆ *DMA Controller*

☆ *System Support Module*

☆ *Enhanced F9450 (F9450E)*

☆ *Other Languages*

FAIRCHILD F9450
16 BIT BIPOLAR
MICROPROCESSOR FAMILY

I^3L TECHNOLOGY

F9450 MICROPROCESSOR

INSTRUCTION SET & PROGRAMMING

SUPPORT DEVICES & APPLICATIONS

DEVELOPMENT ENVIRONMENT

I^3L TECHNOLOGY & THE F9450 MICROPROCESSOR

What is I^3L?

Comparison with Other Technologies

Features of the F9450 Family

F9450 Functional Block Diagram

Processor State Diagram

Fault Handling

Interrupt Handling

Pin Functions

Bus Transactions

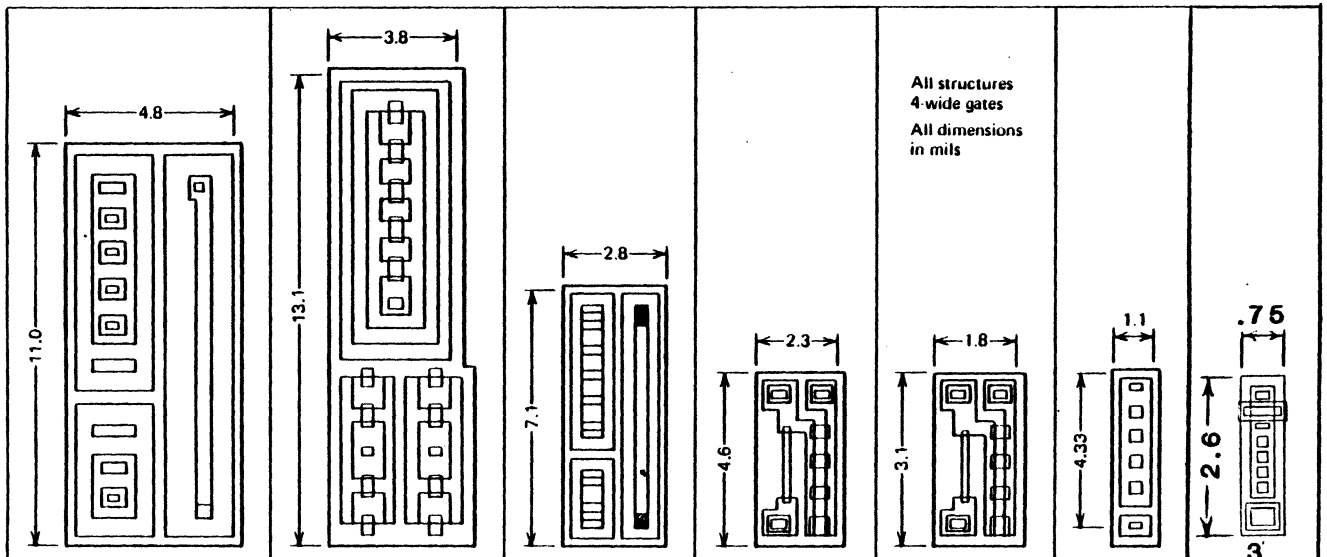
I^3L TECHNOLOGY EVOLUTION

	1974-78	1979-81	1982-84
Technology	I^3L-I	I^3L-Is	I^3L-II
Process Features	5-6u Projection Print Mixed Wet-Dry Etch	4.5u Projection Print Wet-Dry Etch	3u Projection Print Dry Etch Full Implant
Gate Delay	10ns	8ns	5ns
Major Products	9440 9408 9414 9423 9441 9442	9445 9447/8/9 9451 9452 9423	9450
Complexity	10,000	20,000	150,000

JH:kt 1/85

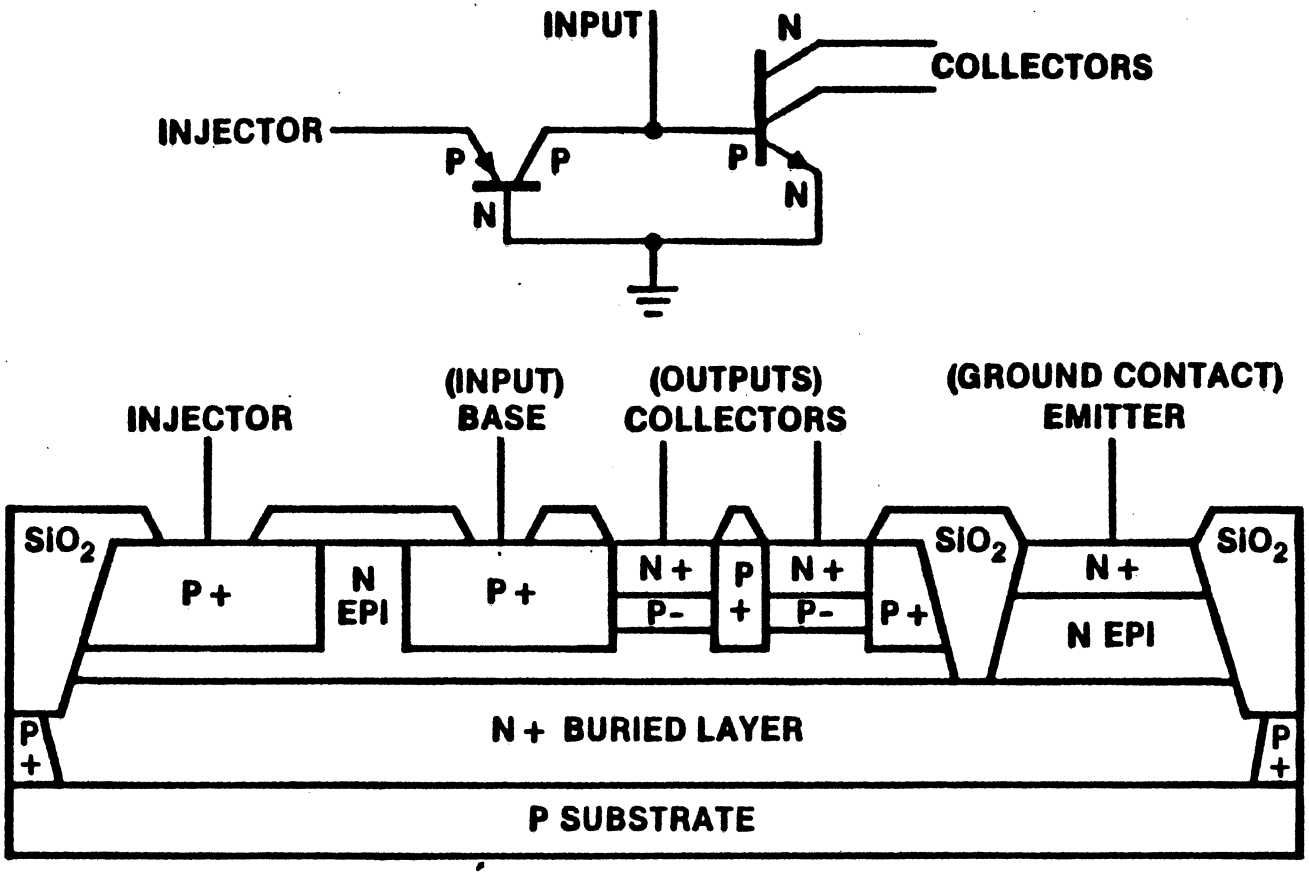
COMPARISON WITH OTHER TECHNOLOGIES

GATE AREAS



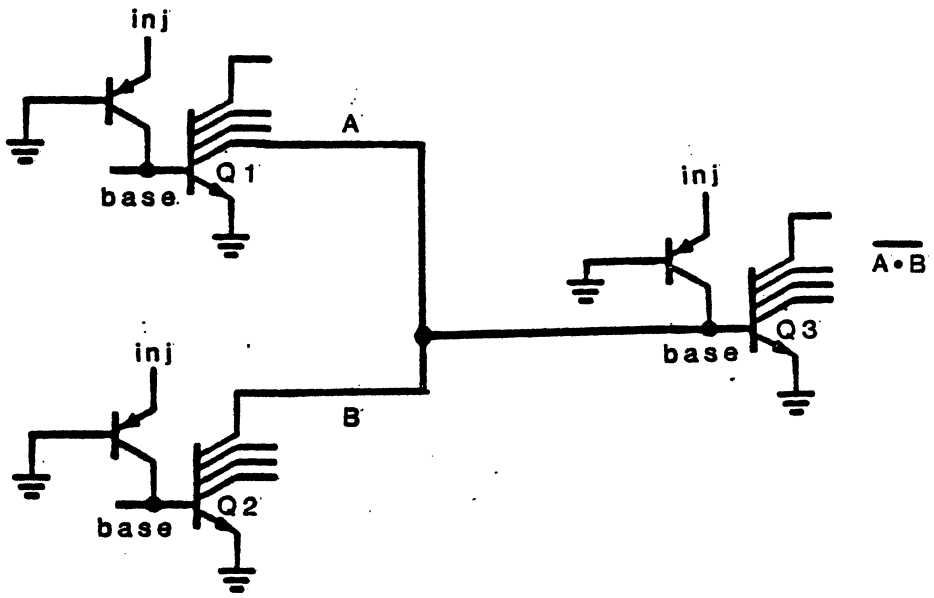
Gate area (sq mils)	52.8	49.8	19.9	10.6	5.6	4.8	2.0

I^3L Cell

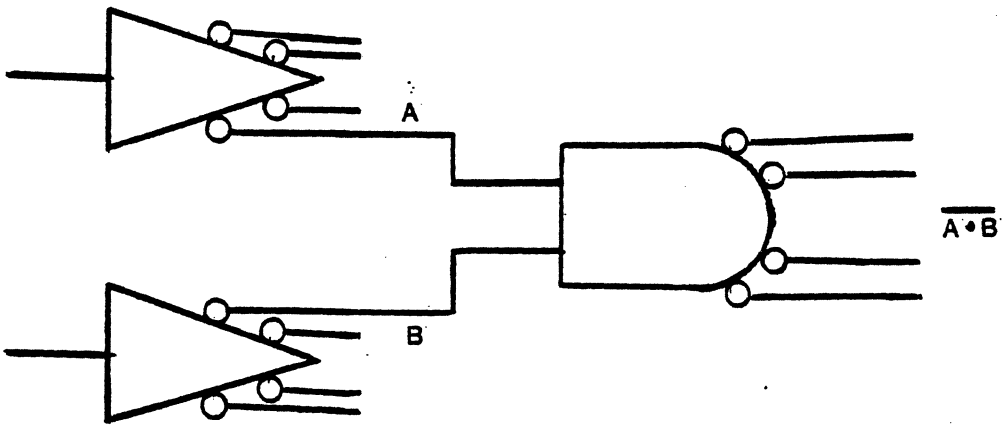

INPUT
N
COLLECTORS
INJECTOR
P
P
P
N
N
INJECTOR
(INPUT)
BASE
(OUTPUTS)
COLLECTORS
(GROUND CONTACT)
EMITTER
SiO2
P+
N
EPI
P+
N+
P-
P
+
N+
P-
P+
SiO2
N+
N EPI
SiO2
P
+
N+ BURIED LAYER
P
+
P SUBSTRATE

I^3L LOGIC IMPLEMENTATION

NAND FUNCTION



CIRCUIT SCHEMATIC



LOGIC SCHEMATIC

JH:11/84

F9450 Microprocessor Family

- F9450 - SINGLE CHIP CPU
  - IMPLEMENTS MIL-STD 1750A INSTRUCTION SET ARCHITECTURE (ISA)
- F9451 - MEMORY MANAGEMENT UNIT (MMU)
  - LOGICAL TO PHYSICAL ADDRESS TRANSLATION FOR INSTRUCTION AND DATA SPACES
- F9452 - BLOCK PROTECT UNIT (BPU)
  - WRITE PROTECTION OF PHYSICAL MEMORY SPACE FOR CPU AND DMA DATA SPACES

F9450

PROCESSOR FEATURES

☆*Single-chip 16-bit microprocessor w/32 & 48-bit floating point math on chip; .2microS add, 1.85microS integer multiply.*

☆*Real-time processing; 2 programmable timers, 16 levels of vectored interrupt.*

☆*Address space of up to 2M words, expandable to 16M words.*

☆*Optimized instruction set for real time applications.*

☆*Built in self-test, fault-handler, and abort.*

☆*24 user-accessible registers.*

☆*Full development support available from Fairchild & 3rd party vendors.*

☆*Single and Double precision integer math on chip.*

☆*Static operation with single clock : operation from 0 to 20MHz.*

☆*TTL I/O with 8Ma drive capability.*

☆*Small size 64 pin dip, optional surface mount devices.*

☆*Full performance over -55C to +125C temp. range.*

F9450 SYSTEM THROUGHPUT (DAIS MIX)

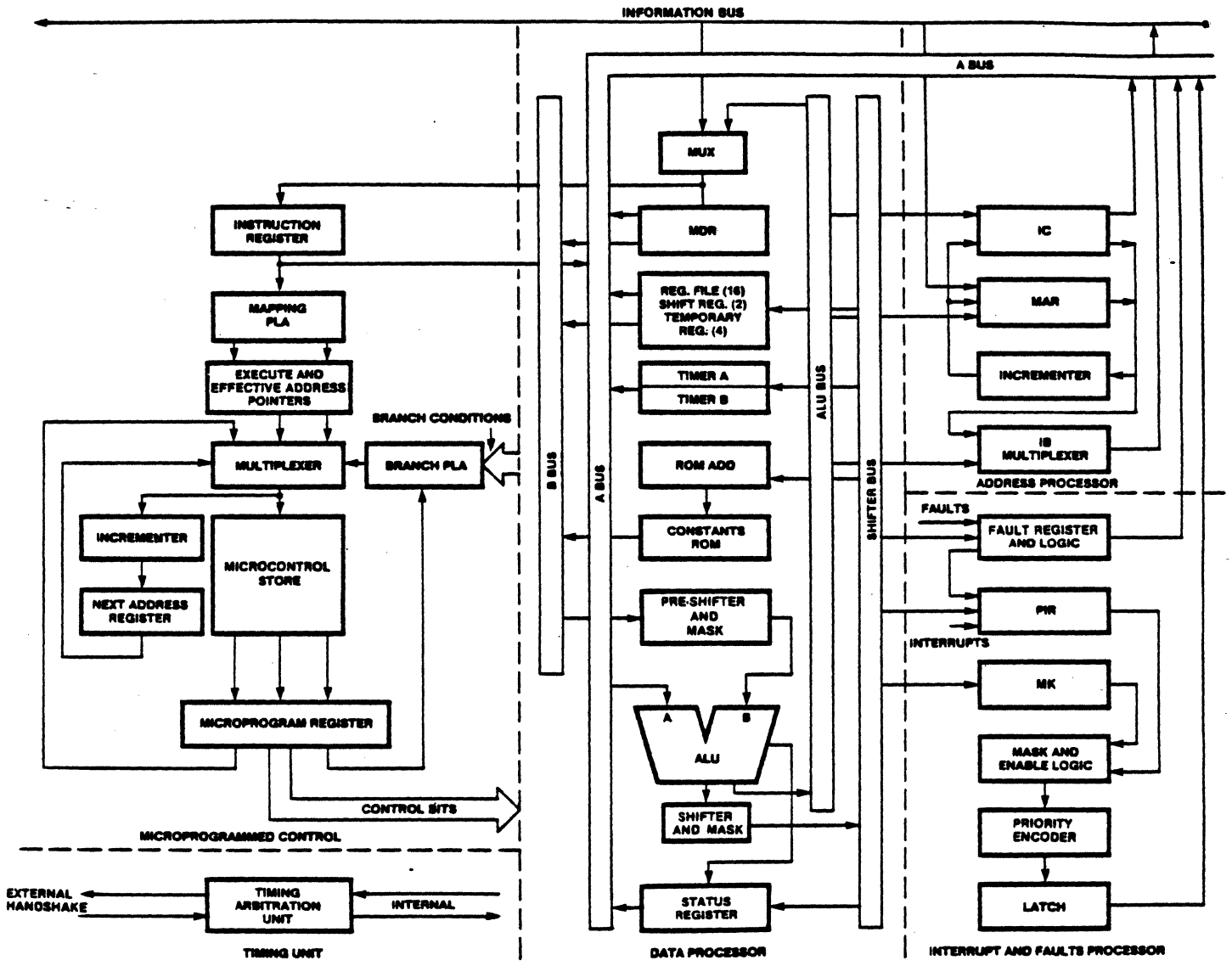
CPU CLOCK (MHz)	F9450 SYSTEM THROUGHPUT (KOPS - DAIS MIX)				
	NO WAIT*	ONE WAIT*	TWO WAIT*	THREE WAIT*	FOUR WAIT*
20	690	638	594	556	522
18	621	574	535	500	470
16	552	510	475	445	418
10	345	319	297	278	261

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

DAIS INSTRUCTION MIX:

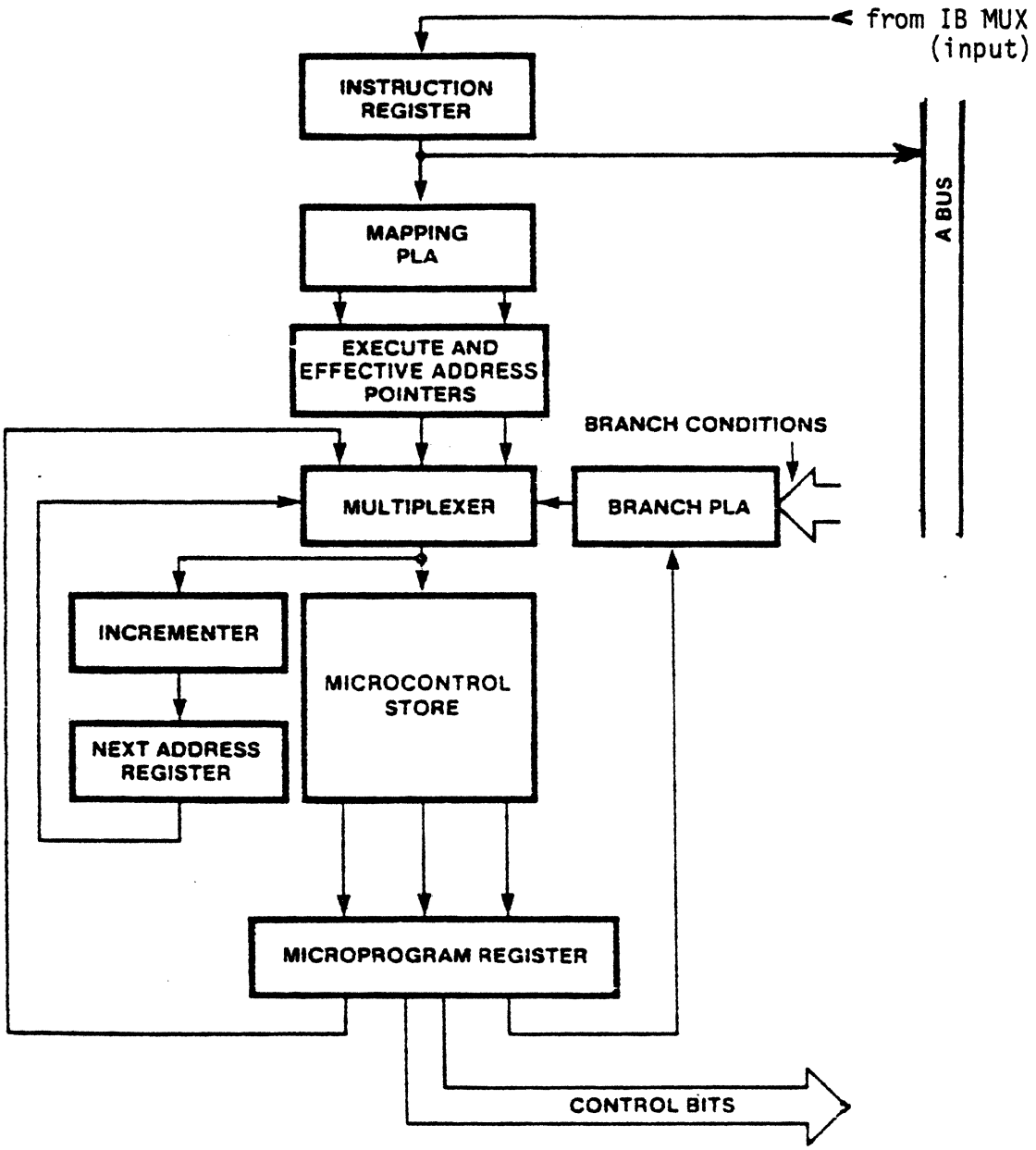
SINGLE PRECISION	74%
DOUBLE PRECISION	10%
FLOATING POINT	15%
EXTENDED FLOATING POINT	1%

RM:5/84

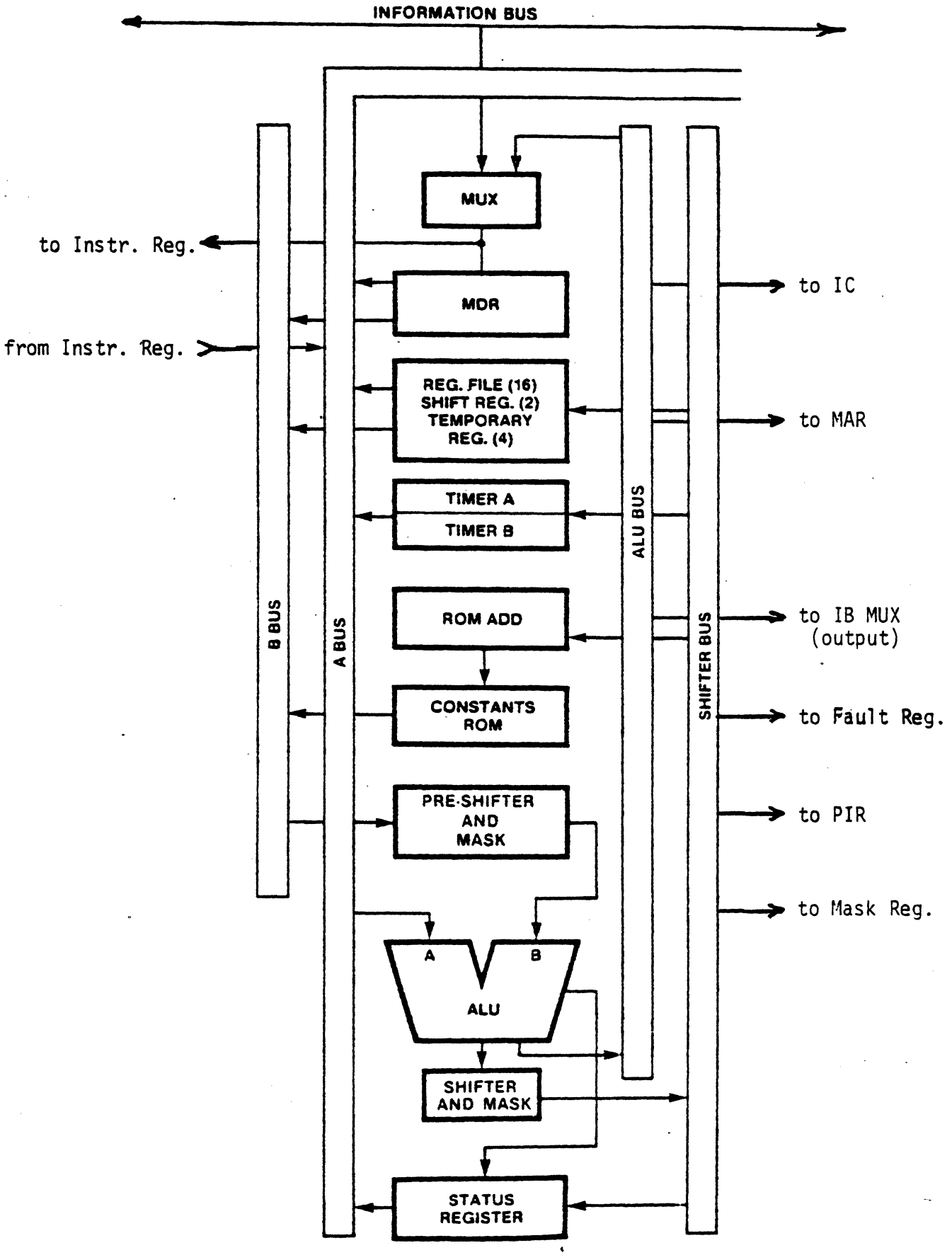


F9450 BLOCK DIAGRAM

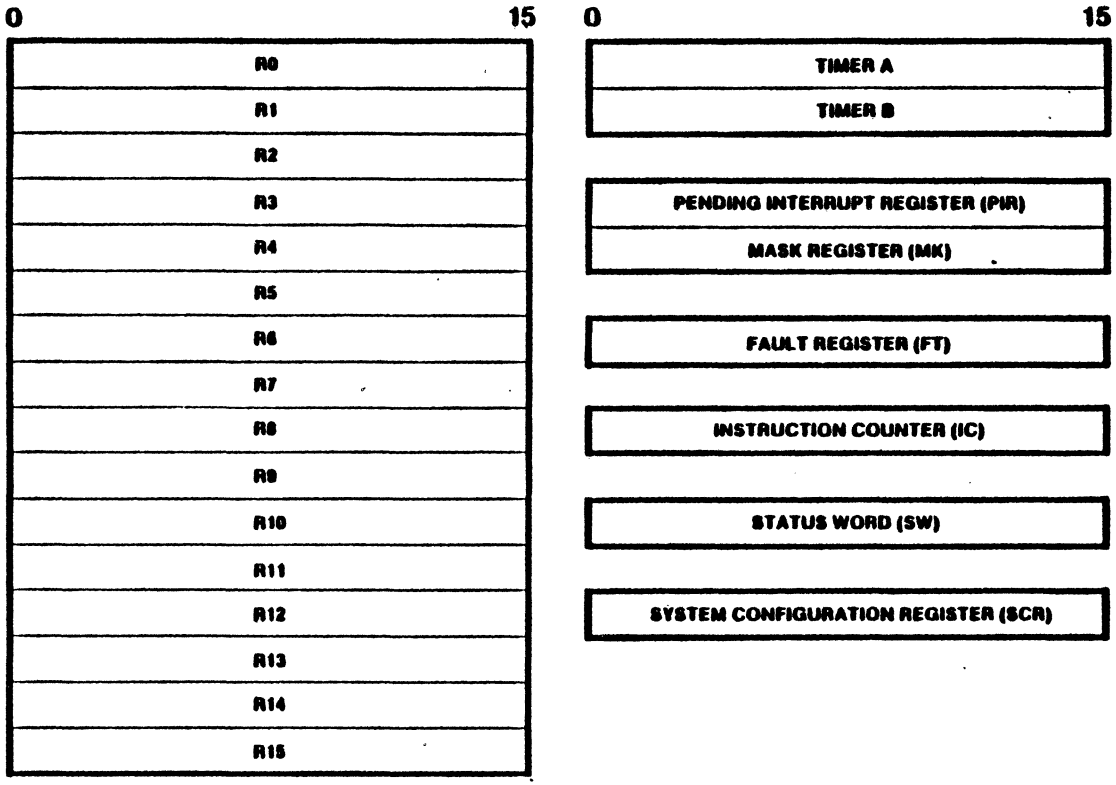
MICROPROGRAMMED CONTROL



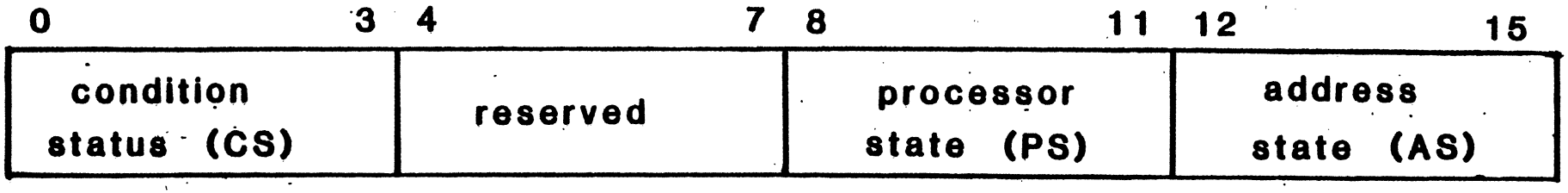
DATA PROCESSOR



F9450 Register Model



STATUS WORD (SW)



BIT 0 : CARRY - SET TO 1 IF A CARRY FROM AN ADDITION OR BORROW FROM A SUBTRACTION
BIT 1 : POSITIVE - SET TO 1 IF RESULT IS GREATER THAN ZERO
BIT 2 : ZERO - SET TO 1 IF RESULT IS EQUAL TO ZERO
BIT 3 : NEGATIVE - SET TO 1 IF RESULT IS LESS THAN ZERO

BITS 4 - 7 : RESERVED

BITS 8 - 11 : ACCESS KEY AND PROCESSOR STATE BITS SERVE TWO FUNCTIONS:

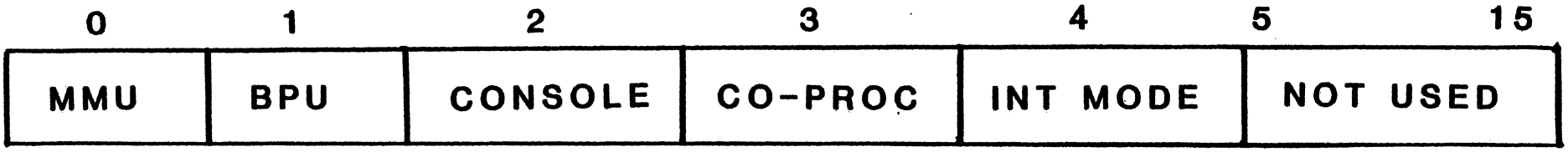
(1) DETERMINE THE LEGAL/ILLEGAL CRITERIA FOR PRIVILEGED INSTRUCTIONS. IF PS ≠ 0 AND AN ATTEMPT IS MADE TO EXECUTE A PRIVILEGED INSTRUCTION, BIT 10 IS SET IN THE FAULT REGISTER AND THE INSTRUCTION IS ABORTED.

(2) DEFINES THE ACCESS KEY THAT IS USED IN SYSTEMS WITH AN MMU TO MATCH WITH AN ACCESS LOCK.

BITS 12 - 15 : ADDRESS STATE

IF AN MMU IS PRESENT, THE AS BITS DEFINE A PAGE REGISTER GROUP IN THE MMU. OTHERWISE, AN ADDRESS STATE FAULT IS GENERATE (BIT 11 IN THE FAULT REGISTER) FOR ANY OPERATION ATTEMPTING TO SET THE AS BITS TO A NONZERO VALUE.

System Configuration Register (SCR)



BIT 0 : '1' IF MMU IS CONNECTED IN THE SYSTEM

BIT 1 : '1' IF BPU IS CONNECTED IN THE SYSTEM

BIT 2 : '1' IF CONSOLE IS CONNECTED IN THE SYSTEM

BIT 3 : '1' IF CO-PROCESSOR IS CONNECTED IN THE SYSTEM

BIT 4 : INTERRUPT MODE FOR POWER DOWN AND USER INTERRUPTS

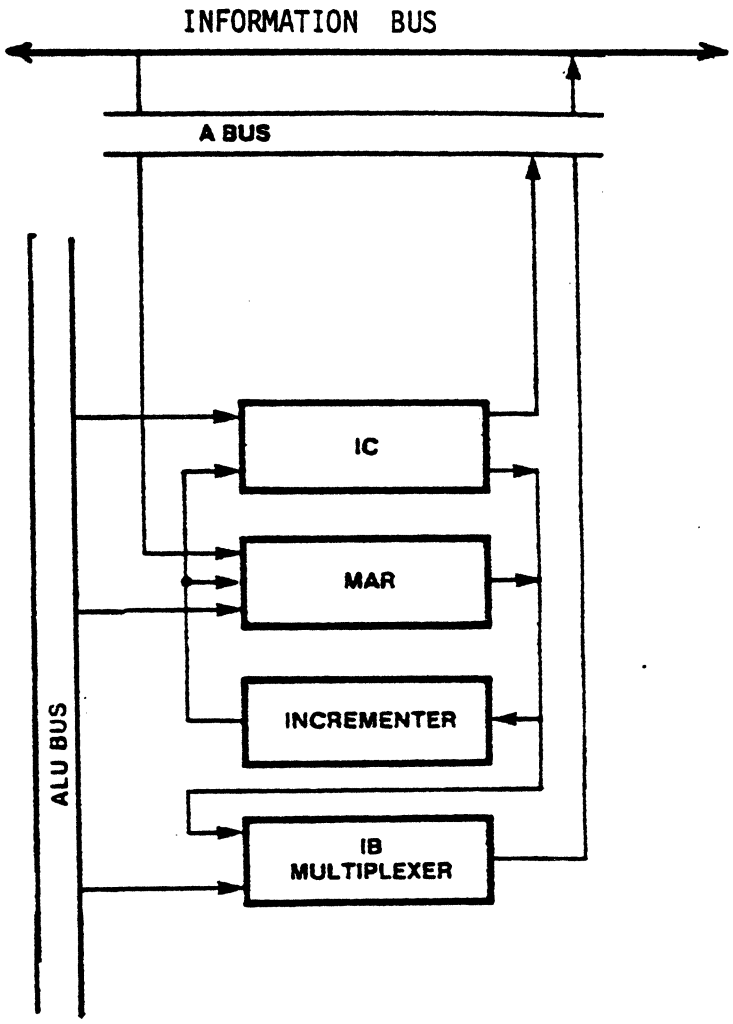
'1' IF LEVEL SENSITIVE

'0' IF EDGE SENSITIVE (LOW TO HIGH)

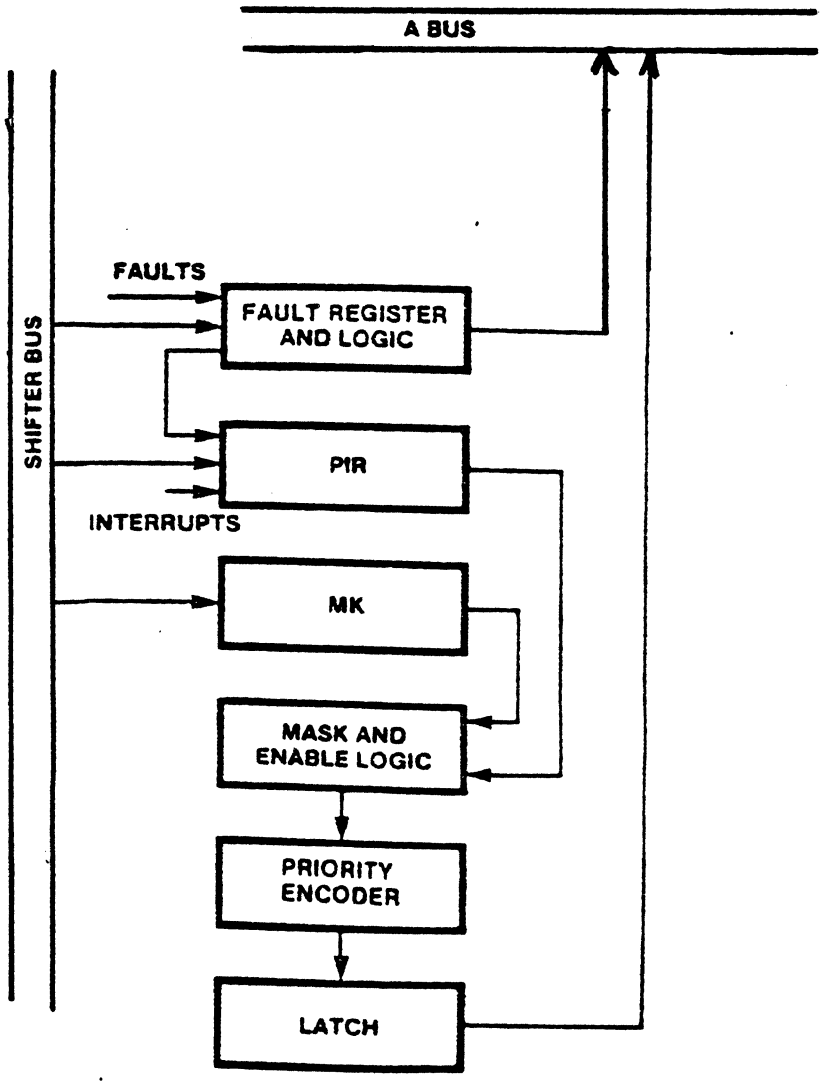
THE SCR IS AUTOMATICALLY LOADED ON RESET OR A BPT.

THE SCR IS NOT A PROGRAM ACCESSIBLE REGISTER; IT IS AN I/O DEVICE AT LOCATION 8410.

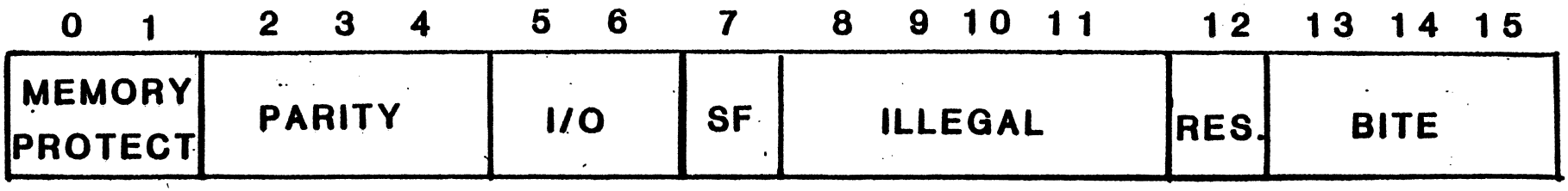
ADDRESS PROCESSOR



INTERRUPT & FAULTS PROCESSOR



FAULT REGISTER (FT)



BIT 0 : CPU MEMORY PROTECT 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 : RESERVED

BIT 13 : BITE (BUILT-IN TEST) OR SYSTEM FAULT 1
BIT 14 : SPARE
BIT 15 : SYSTEM FAULT 1

INTERRUPT PRIORITIES

	PRIORITY *** (PIR/MK bit number)	Interrupt Linkage Pointer Address	Interrupt Service Pointer Address
POWER DOWN*	0	20	21
MACHINE ERROR**	1	22	23
USER 0	2	24	25
F.P. OVERFLOW	3	26	27
FIXED PT. OVERFLOW	4	28	29
EXECUTIVE CALL*	5	2A	2B
F.P. UNDERFLOW	6	2C	2D
TIMER A	7	2E	2F
USER 1	8	30	31
TIMER B	9	32	33
USER 2	10	34	35
USER 3	11	36	37
I/O LEVEL 1	12	38	39
USER 4	13	3A	3B
I/O LEVEL 2	14	3C	3D
USER 5	15	3E	3F

* CANNOT BE MASKED OR DISABLED

** CANNOT BE DISABLED

*** INTERRUPT LEVEL 0 HAS THE HIGHEST PRIORITY

RM: 5/84

Interrupt Handling

- THE 16 LEVELS OF INTERRUPT IN THE MIL-STD-1750A ARE IMPLEMENTED AND PRIORITIZED ON THE F9450.

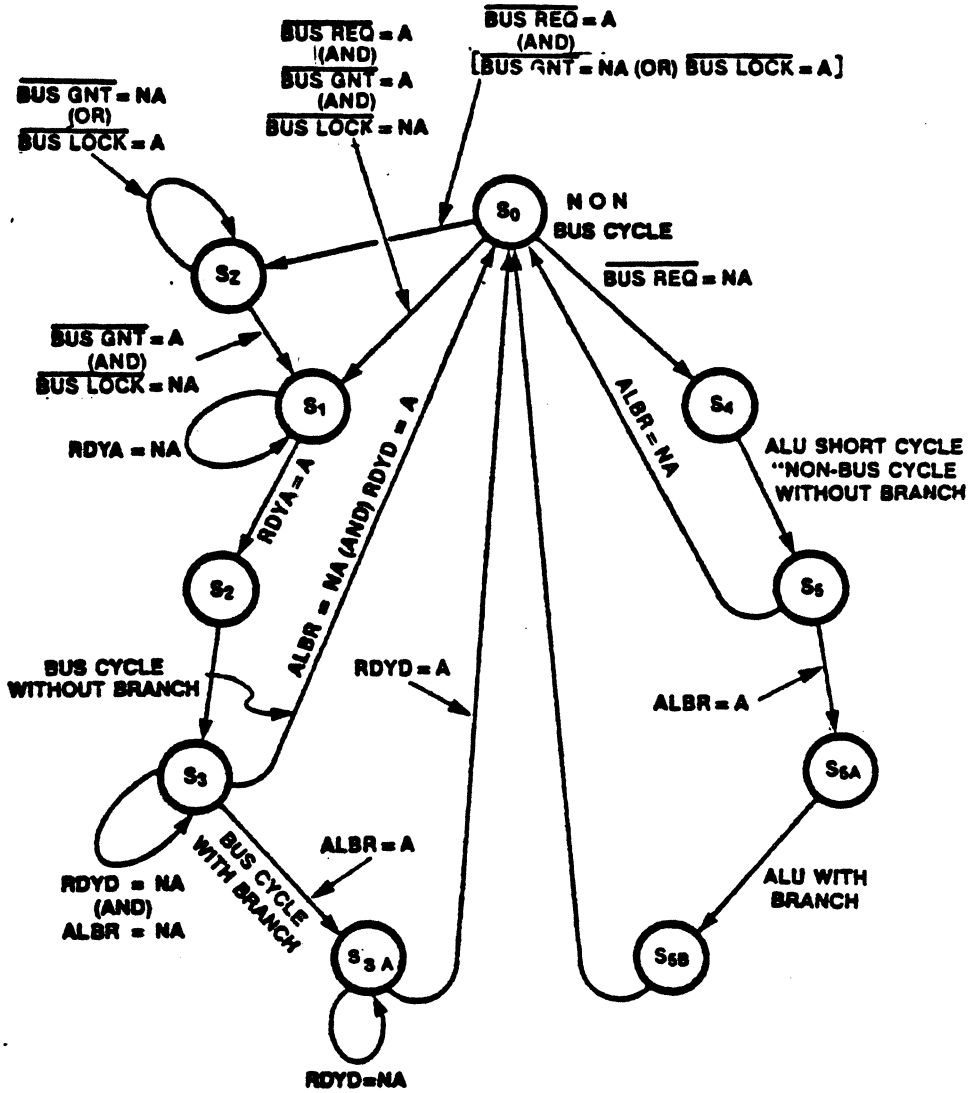
- THE F9450 SAMPLES PENDING INTERRUPTS ONLY DURING THE LAST MACHINE CYCLE OF THE CURRENTLY EXECUTED INSTRUCTION.

- EXTERNAL INTERRUPTS CAN BE PROGRAMMED AS EITHER LEVEL OR EDGE-SENSITIVE, ACCORDING TO THE INTERRRUPT MODE BIT IN THE CONFIGURATION REGISTER.

- THE INTERNAL INTERRUPT MICROCODE ROUTINE CONSISTS OF:
  - 12 ALU SHORT CYCLES (3 CLOCKS/CYCLE)
  - 2 ALU LONG CYCLES (5 CLOCKS/CYCLE)
  - 5 DATA MEMORY READ CYCLES
  - 3 DATA MEMORY WRITE CYCLES
  - 1 I/O WRITE CYCLE @I/O LOCATION 1000H (INTERRUPT ACK)
  - 2 INSTRUCTION FETCH CYCLES

  TOTAL - 90 CLOCKS, 4.50 μSEC AT 20 MHZ (ASSUMING NO WAIT STATES ADDED)

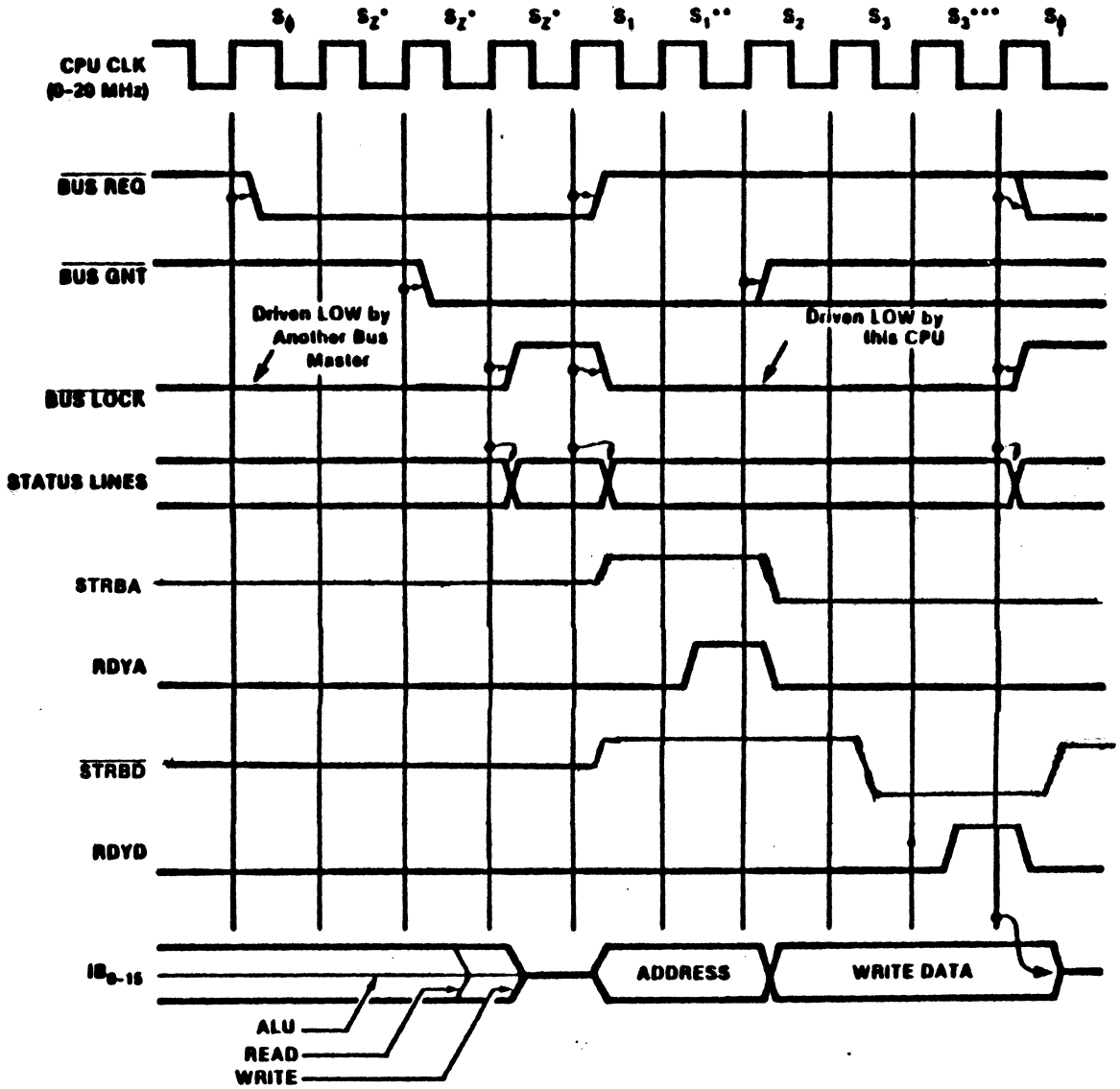
TIMING UNIT



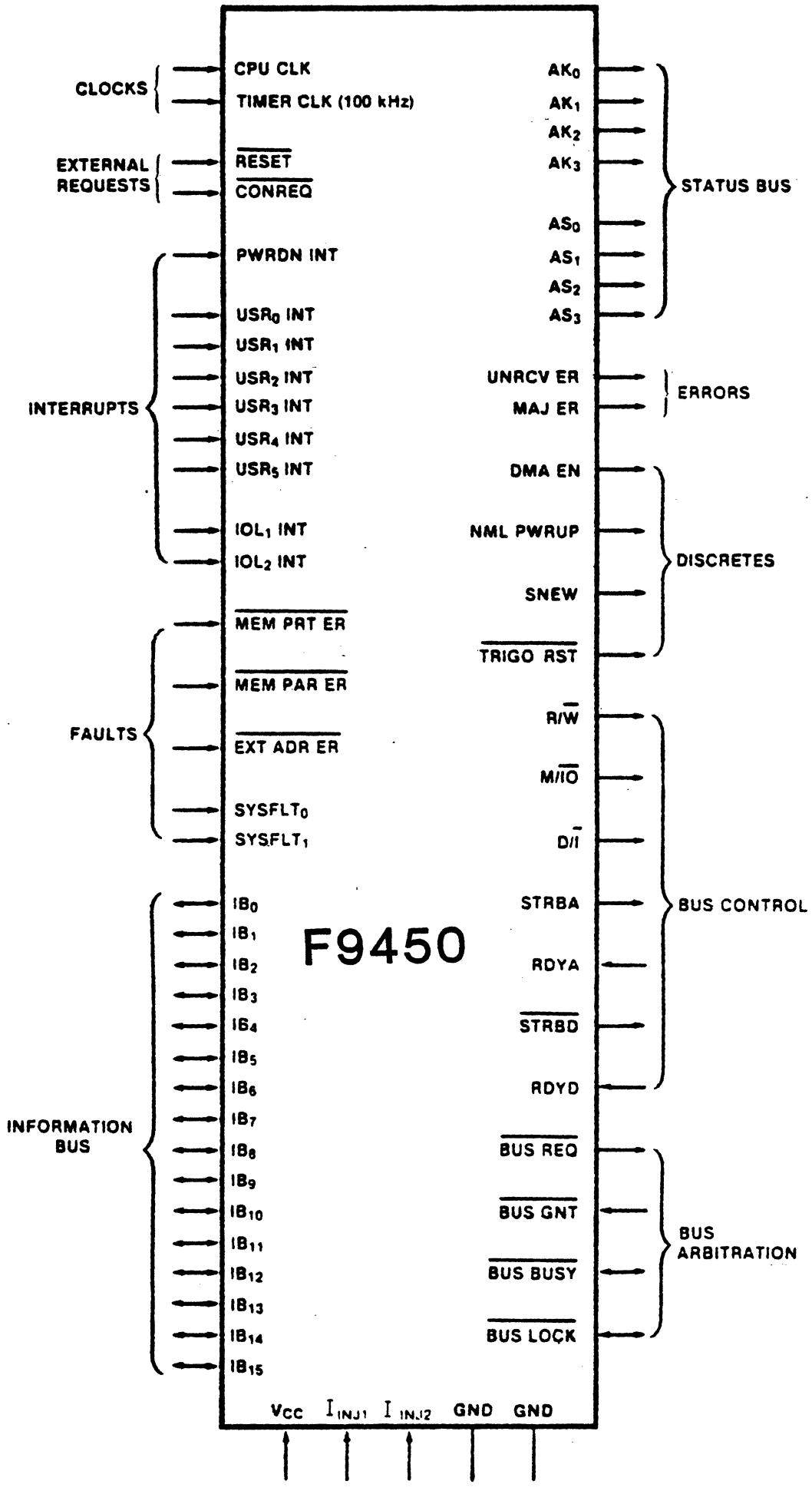
Legend

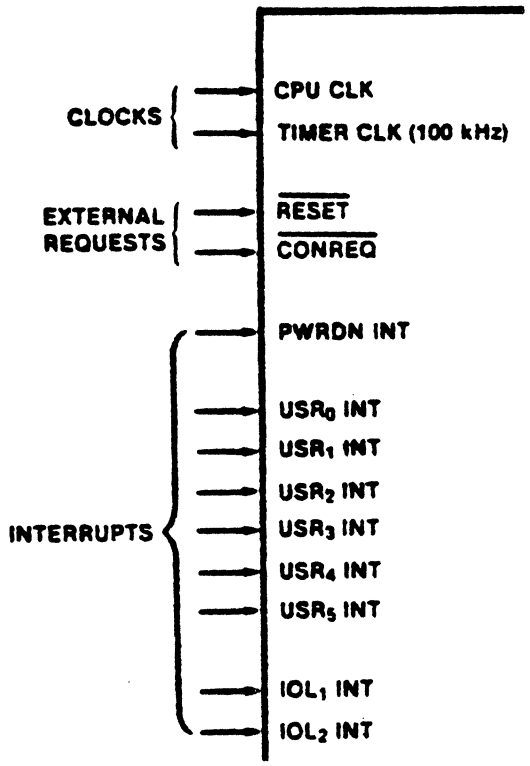
A	=	Asserted (active)
NA	=	Not Asserted
ALBR	=	ALU Branch cycle (5 states) - internal signal
$\overline{\text{BUS REQ}}$	=	Bus Request
$\overline{\text{BUS GNT}}$	=	Bus Grant input
$\overline{\text{BUS LOCK}}$	=	Bus Lock from External Source
RDYA	=	RDYA input
RDYD	=	RDYD input
S_Z	=	High impedance state - CPU drivers are 3-state

F9450 CPU Bus Access Cycle



* High Impedence State. All CPU drivers are 3-state
** Address Wait State
*** Data Wait State


CLOCKS
CPU CLK
TIMER CLK (100 kHz)
EXTERNAL REQUESTS
RESET
CONREQ
PWRDN INT
USR0 INT
USR1 INT
USR2 INT
USR3 INT
USR4 INT
USR5 INT
IOL1 INT
IOL2 INT
INTERRUPTS
MEM PRT ER
MEM PAR ER
EXT ADR ER
SYSFLT0
SYSFLT1
FAULTS
IB0
IB1
IB2
IB3
IB4
IB5
IB6
IB7
IB8
IB9
IB10
IB11
IB12
IB13
IB14
IB15
INFORMATION BUS
F9450
AK0
AK1
AK2
AK3
AS0
AS1
AS2
AS3
STATUS BUS
UNRCV ER
MAJ ER
ERRORS
DMA EN
NML PWRUP
SNEW
TRIGO RST
DISCRETES
R/W
M/IO
D/I
STRBA
RDYA
STRBD
RDYD
BUS CONTROL
BUS REQ
BUS GNT
BUS BUSY
BUS LOCK
BUS ARBITRATION
VCC
IINJ1
IINJ2
GND
GND



Clocks

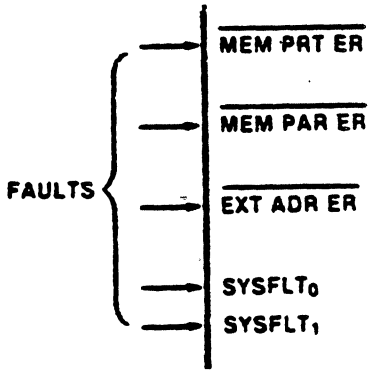
CPU Clock	Single-phase input clock signal (0-20 MHz, 40% to 60% duty cycle).
Timer Clock	A 100 kHz input that, after synchronization with CPU CLK, provides the clock for Timer A and Timer B. If timers are used, the CPU CLK signal frequency must be >500 kHz.

External Requests

Reset	An active-low input that initializes the F9450.
Console Request	An active-low input that initiates console operations after the current instruction.

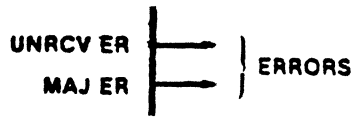
Interrupts

Power Down Interrupt	An input signal that is active on the positive-going edge or the high level, according to the interrupt mode bit in the configuration register. These inputs have hysteresis circuitry for noise immunity.
User Interrupt	Input signals that are active on the positive-going edge or high level, according to the interrupt mode bit in the configuration register. This input has hysteresis circuitry for noise immunity.
I/O Level Interrupts	Active-high inputs that can be used to expand the number of user interrupts.



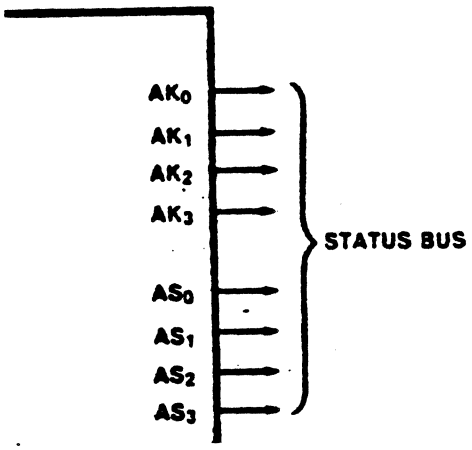
Faults

Memory Protect Error	An active-low input generated by the MMU or BPU, or both, and sampled by the $\overline{\text{BUS BUSY}}$ signal into the Fault Register (bit 0 if CPU bus cycle, bit 1 if non-CPU bus cycle).
Memory Parity Error	An active-low input sampled by the $\overline{\text{BUS BUSY}}$ signal into bit 2 of the Fault Register.
External Address Error	An active-low input sampled by the $\overline{\text{BUS BUSY}}$ signal into the Fault Register (bit 5 or 8), depending on the cycle (memory or I/O).
System Fault 0 System Fault 1	Asynchronous, positive-edge-sensitive inputs to the F9450 that set bit 7 ($SYSFLT_0$) or bits 13 and 15 ($SYSFLT_1$) in the Fault Register. These inputs are protected from system noise through hysteresis circuitry.



Error Control

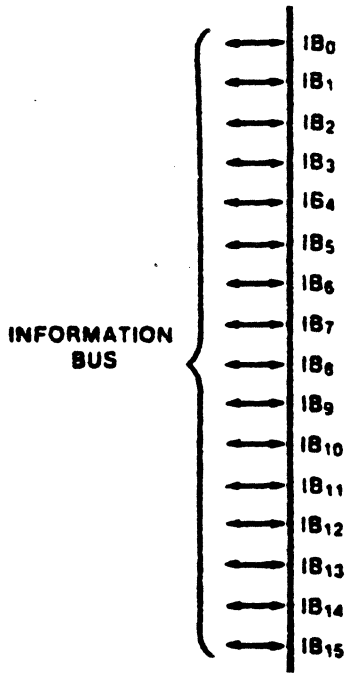
Unrecoverable Error	An active-high output that indicates the occurrence of an error classified as unrecoverable.
Major Error	An active-high output indicating the occurrence of an error classified as major.



Status Bus

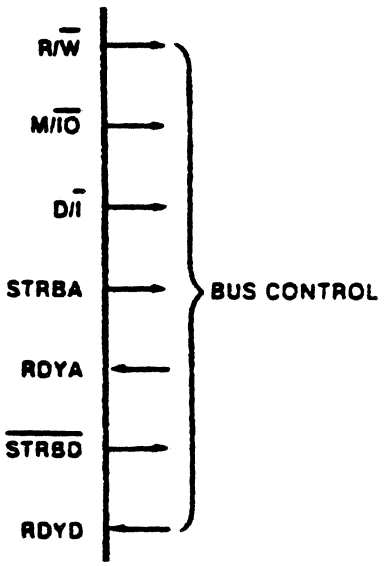
Access Key — Active-high outputs used to match the Access Lock in the MMU for memory accesses (a mismatch is one of several possible situations that cause the MMU to pull the $\overline{\text{MEM PRT ER}}$ signal low). These signals are 3-state during bus cycles not assigned to this CPU.

Address State — Active-high outputs that select the page register group in the MMU; 3-state bus during bus cycles not assigned to this CPU. These outputs can be used to expand the F9450 direct addressing range to 1M words.



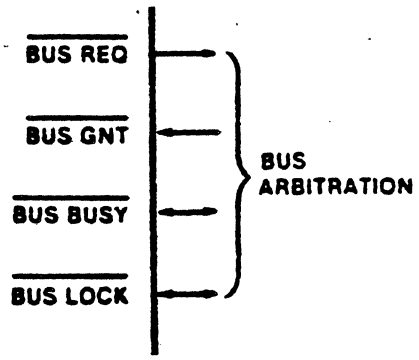
Information Bus

Information Bus — An active-high bidirectional time-multiplexed address/data bus that is 3-state during bus cycles not assigned to this CPU; IB_0 is the most significant bit.



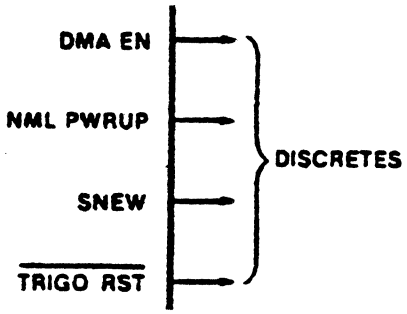
Bus Control

Read or Write	An output signal that indicates direction of data flow with respect to the current bus master: a high indicates a read or input operation and a low indicates a write or output operation. The signal is 3-state during bus cycles not assigned to this CPU.
Data or Instruction	An output signal that indicates whether the current bus cycle access is for Data (high) or Instruction (low); 3-state during bus cycles not assigned to this CPU. This line can be used as an additional memory address bit for systems that require separate data and program memory.
Data or Instruction Output	Output signal that indicates whether the current bus cycle is memory (high) or I/O (low). This signal is 3-state during bus cycles not assigned to this CPU.
Address Strobe	An active-high output that can be used to externally latch the memory or I/O address at the high-to-low transition of the strobe. The signal is 3-state during bus cycles not assigned to this CPU.
Address Ready	An active-high input that can be used to extend the address phase of a bus cycle. When RDYA is not active, wait states are inserted by the F9450 timing unit to accommodate slower memory or I/O devices.
Data Strobe	An active-low output that can be used to strobe data in memory and XIO cycles. This signal is 3-state during bus cycles not assigned to this CPU.
Data Ready	An active-high input that extends the data phase of a bus cycle. When RDYD is not active, wait states are inserted by the F9450 timing unit to accommodate slower memory or I/O devices.



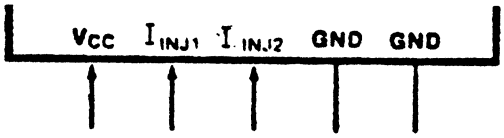
Bus Arbitration

Bus Request	An active-low output that indicates the CPU requires the bus; becomes inactive when the CPU has acquired the bus and started the bus cycle.
Bus Grant	An active-low input from an external arbiter that indicates the CPU currently has the highest priority bus request. If the bus is not locked, the CPU may begin a bus cycle, commencing with the next CPU clock.
Bus Busy	An active-low bidirectional signal used to establish the beginning and end of a bus cycle. The trailing edge (low-to-high transition) is used for sampling bits into the Fault Register. It is 3-state in bus cycles not assigned to this CPU; however, the CPU monitors the BUS BUSY line for latching non-CPU bus-cycle faults into the Fault Register.
Bus-Lock	An active-low, bidirectional signal used to lock the bus for successive bus cycles. During non-locked bus cycles, the BUS LOCK signal mimics the BUS BUSY signal. It is 3-state during bus cycles not assigned to this CPU. The following instructions will lock the bus: INCM, DECM, SB, RB, TSB, SRM, STUB, STLB.



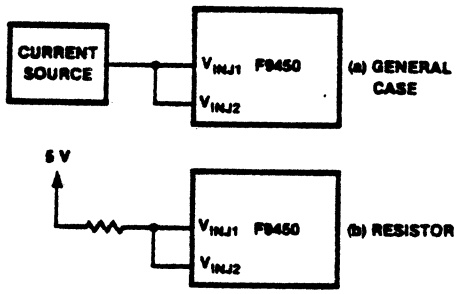
Discrete Control

Direct Memory Access Enable	An active-high output that indicates the DMA is enabled. It is disabled when the CPU is initialized (reset) and is enabled under program control.
Normal Power Up	An active-high output that is set when the CPU has successfully completed the built-in test in the initialization sequence.
Start New	An active-high output that indicates a new instruction will start executing in the next cycle; useful for instruction tracing function.
Trigger Go Reset	An active-low discrete output. This signal can be pulsed low under program control [I/O address 400B (Hex)] and is automatically pulsed once during processor initialization.



Power

Power Supply	+5 V, 200 mA typical power supply.
Ground	0 V reference. **These pins should be tied together as close to the chip as possible.**
Injector Current	Current source to provide bias for the injection logic. **These pins should be tied together as close to the chip as possible.**



I_{INJ} Supply.

INSTRUCTION SET & PROGRAMMING

SAMPLE PROGRAM

- Towers of Hanoi
- Input/Output Subroutines

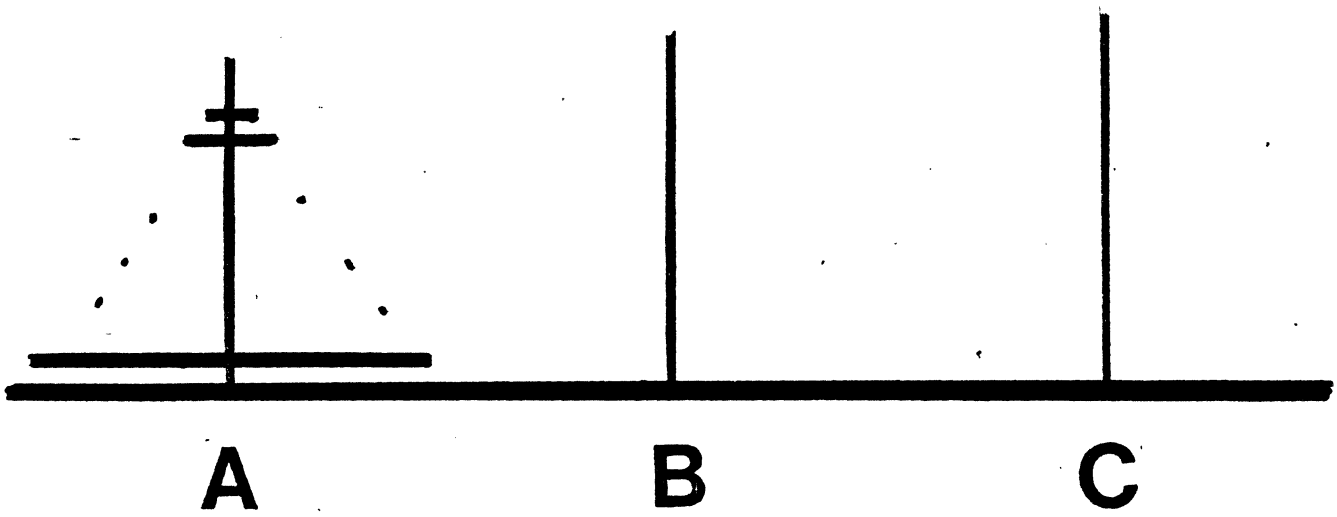
INSTRUCTION SET ARCHITECTURE

- Data Types
- Addressing Modes
- Instruction Set Overview

ANOTHER PROGRAM

- Matrix Multiplication
- ASCII <---> Binary Conversions

TOWERS OF HANOI

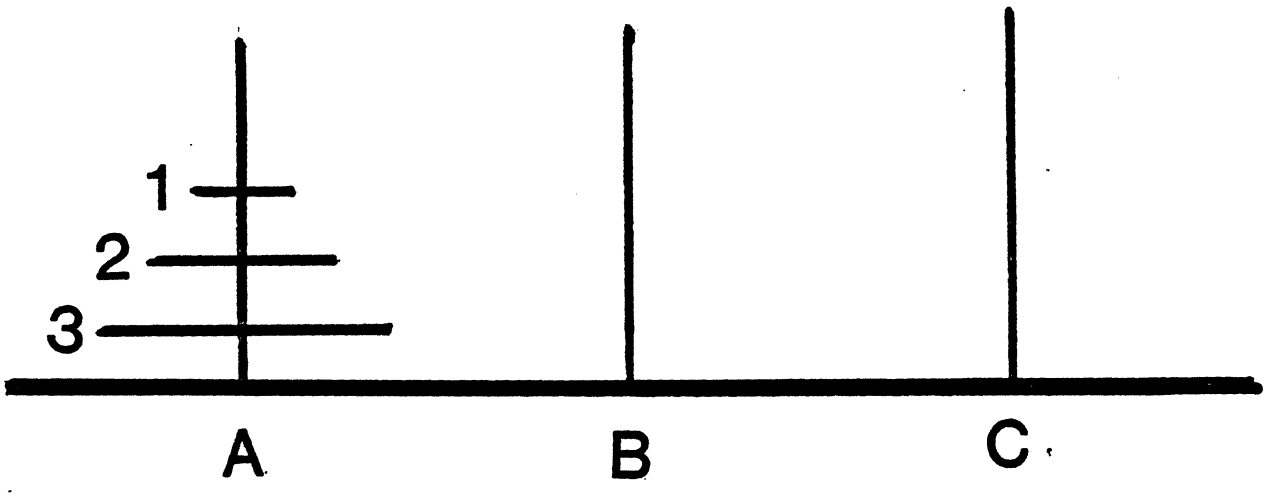


GIVEN: 3 towers and N disks
of increasing size

GOAL: move all N disks from A to C
by moving one disk at a time
and never placing a larger disk
on top of a smaller disk

BK:5/84

SOLUTION FOR 3 DISKS



disk	from	to	encoding
1	A	C	A 1 C
2	A	B	A 2 B
1	C	B	C 1 B
3	A	C	A 3 C
1	B	A	B 1 A
2	B	C	B 2 C
1	A	C	A 1 C

BK:5/84

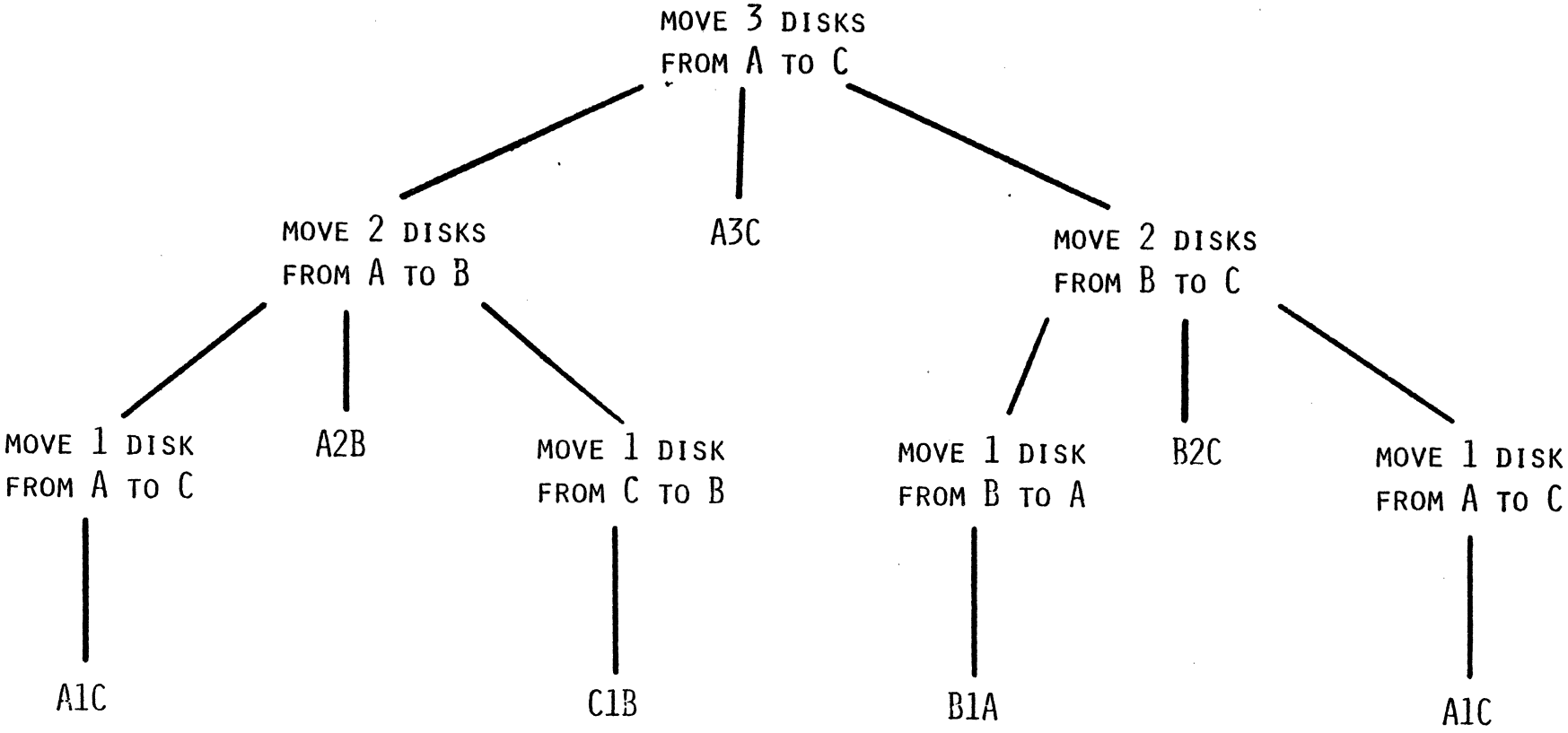
RECURSIVE SOLUTION FOR N DISKS

if N=1 then

move the disk from the starting tower to the destination tower

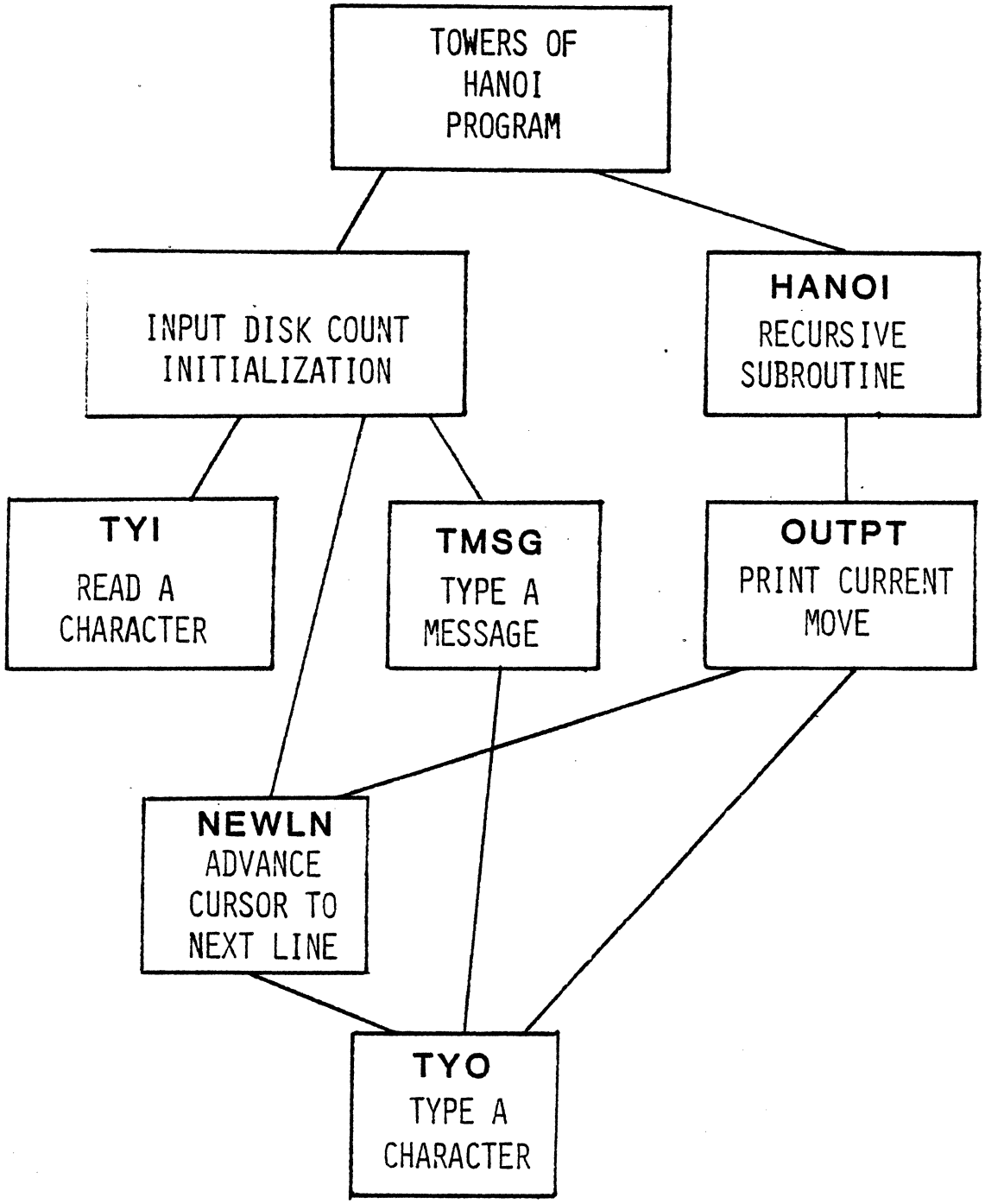
if N > 1 then

- move N-1 disks from the starting tower to the intermediate tower
- move the Nth disk from the starting tower to the destination tower
- move N-1 disks from the intermediate tower to the destination tower



RECURSIVE SOLUTION FOR 3 DISKS

PROGRAM STRUCTURE CHART



BK:5/84

TOWERS OF HANOI DEMO FOR THE F9450

```
; MAIN PROGRAM FOR THE TOWERS OF HANOI DEMONSTRATION.
; THE USER IS ASKED TO SPECIFY THE NUMBER OF DISKS, 0..9;
; AN ENTRY OF 0 TERMINATES THE PROGRAM. THE SOLUTION IS
; PRINTED AS A SEQUENCE OF MOVES, SUCH AS A3B (THIS MEANS
; TO MOVE DISK 3 FROM A TO B). THE COMPLETE SOLUTION MOVES
; THE SPECIFIED NUMBER OF DISKS FROM TOWER A TO TOWER C.
; THE PROGRAM IS REPEATED UNTIL 0 DISKS ARE REQUESTED.
;
; REGISTERS USED:
;       R0 - NUMBER OF DISKS TO BE MOVED
;       R1 - SOURCE TOWER (A, B, OR C) (REFERRED TO AS X TOWER)
;       R2 - INTERMEDIATE TOWER (A,B,OR C) (REFERRED TO AS Y TOWER)
;       R3 - DESTINATION TOWER (A,B,OR C) (REFERRED TO AS Z TOWER)
;       R13 - POINTER TO MESSAGE TO BE PRINTED
;       R15 - SUBROUTINE LINKAGE REGISTER
; SUBROUTINES  CALLED:
;       HANOI - SOLVES THE TOWERS OF HANOI PUZZLE
;       TMSG - TYPE A MESSAGE
;       NEWLN - TYPE A BLANK LINE
;       TYI - READ A CHARACTER
;
        JC      7,START         ; CONS50 STARTS AT LOCATION 0
        .LOC    1000            ; LEAVE ROOM FOR A STACK
START:  LIM     R15,START       ; INITIALIZE THE STACK POINTER
        LIM     R13,SZMSG
        SJS     R15,TMSG        ; HOW MANY DISKS?
        SJS     R15,TYI         ; READ IN NUMBER OF DISKS
        ANDM    R0,000F         ; CONVERT TO DIGIT
        SJS     R15,NEWLN       ; GO TO NEXT LINE
        CIM     R0,0            ; N = 0?
        BEZ     EXIT            ; IF SO, EXIT
        LIM     R1,"A           ; INITIALIZE X TO A
        LIM     R2,"B           ; INITIALIZE Y TO B
        LIM     R3,"C           ; INITIALIZE Z TO C
        SJS     R15,HANOI       ; SOLVE THE PUZZLE
        BR      START           ; REPEAT PROGRAM
EXIT:   BPT                     ; HALT PROGRAM
SZMSG:  .TXT    "How MANY DISKS? (0..9) <0>"
```

HANOI - solves towers of hanoi puzzle

```
; THIS SUBROUTINE SOLVES THE TOWERS OF HANOI PUZZLE FOR N DISKS
; USING THE FOLLOWING RECURSIVE STRATEGY:
;   IF N = 1
;       MOVE THE DISK FROM THE STARTING TOWER TO THE DESTINATION TOWER
;   IF N >1
;       - MOVE N-1 DISKS FROM THE STARTING TOWER TO THE
;         INTERMEDIATE TOWER
;       - MOVE THE NTH DISK FROM THAT STARTING TOWER TO THE
;         DESTINATION TOWER
;       - MOVE N-1 DISKS FROM THE INTERMEDIATE TOWER TO THE
;         DESTINATION TOWER
;
; REGISTERS USED:
;       R0 - NUMBER OF DISKS TO BE MOVED
;       R1 - STARTING TOWER (A, B, OR C), REFERRED TO AS X
;       R2 - INTERMEDIATE TOWER (A, B, OR C), REFERRED TO AS Y
;       R3 - DESTINATION TOWER (A, B, OR C), REFERRED TO AS Z
; SUBROUTINES CALLED:
;       HANOI - SINCE IT IS RECURSIVE
;       OUTPT - PRINT THE CURRENT MOVE

HANOI:  CIM     R0,1            ; N = 1 ?
        BEZ     BASIS           ; IF SO, OUTPUT BASIS MOVE
        PSHM    R0,R3           ; SAVE N,X,Y,Z ON STACK
        SISP    R0,1            ; SET NUMBER OF DISKS TO N-1
        XWR     R2,R3           ; EXCHANGE Y AND Z
        SJS     R15,HANOI       ; MOVE N-1 DISKS FROM X TO Y
        POPM    R0,R3           ; RESTORE N,X,Y,Z
        SJS     R15,OUTPT       ; PRINT X N Z
        PSHM    R0,R3           ; SAVE N,X,Y,Z ON STACK
        SISP    R0,1            ; SET NUMBER OF DISKS TO N-1
        XWR     R1,R2           ; EXCHANGE X AND Y
        SJS     R15,HANOI       ; MOVE N-1 DISKS FROM Y TO Z
        POPM    R0,R3           ; RESTORE N,X,Y,Z
        URS     R15             ; RETURN
BASIS:  SJS     R15,OUTPT       ; PRINT X 1 Z
        URS     R15             ; RETURN
```

BK:5/84

OUTPT – outputs movement of a disk

```
; THIS SUBROUTINE PRINTS X N Z BASED ON THE CONTENTS OF
; R1, R0, AND R3
;
; REGISTERS USED:
;       R0 - N, NUMBER OF DISKS TO BE MOVED
;       R1 - X, STARTING TOWER (A, B, OR C)
;       R3 - Z, DESTINATION TOWER (A, B, OR C)
;       R11 - CHARACTER TO BE PRINTED
;       R15 - SUBROUTINE LINKAGE REGISTER
; SUBROUTINES CALLED:
;       TYO - TYPE A CHARACTER
;       NEWLN - GO TO NEXT LINE

OUTPT:  LR      R11,R1          ; PREPARE TO PRINT X
        SJS     R15,TYO         ; PRINT VALUE OF X
        LR      R11,R0          ; LOAD VALUE OF N
        AIM     R11,"0          ; CONVERT N TO ASCII
        SJS     R15,TYO         ; PRINT N
        LR      R11,R3          ; PREPARE TO PRINT Z
        SJS     R15,TYO         ; PRINT VALUE OF Z
        SJS     R15,NEWLN       ; SKIP TO NEXT LINE
        URS     R15             ; RETURN
```

NEWLN - advance cursor to next line

```
; TYPE THE ASCII CHARACTERS RETURN & LINE FEED
;
; CALLING SEQUENCE:
;       SJS     R15,NEWLN
;
; REGISTERS USED:
;       R11 - WORKING REGISTER TO STORE CHARACTERS
;       R15 - LINKAGE REGISTER
;
; SUBROUTINES USED:
;       TYO - TYPE A CHARACTER TO THE TERMINAL
;
NEWLN:  LIM     R11,000D
        SJS     R15,TYO         ; PRINT RETURN
        LIM     R11,000A
        SJS     R15,TYO         ; PRINT LINE FEED
        URS     R15             ; RETURN
```

TMSG - type a message

```
;  TYPE A STRING TO THE INTERCOM PORT
;
;  CALLING SEQUENCE:
;       LIM     R13,ADDRESS
;       SJS     R15,TMSG
;
;  REGISTERS USED:
;       R11 - WORKING REGISTER TO STORE CHARACTERS
;       R13 - ADDRESS OF THE MESSAGE TO BE TYPED.  MESSAGE
;             IS TERMINATED WITH A NULL.
;       R15 - SUBROUTINE LINKAGE REGISTER
;
TMSG:   XORR    R11,R11         ; CLEAR R11
        LUB     R11,+0,R13      ; LOAD UPPER BYTE OF WORD
        BEZ     TMSGX           ; EXIT IF NULL
        SJS     R15,TYO         ; OTHERWISE, PRINT CHAR.
        LLB     R11,+0,R13      ; LOAD LOWER BYTE OF WORD
        BEZ     TMSGX           ; EXIT IF NULL
        SJS     R15,TYO         ; OTHERWISE, PRINT CHAR.
        AISP    R13,1           ; INCREMENT MESSAGE POINTER
        BR      TMSG            ; LOOP

TMSGX:  URS     R15             ; RETURN FROM SUBROUTINE
```

TYO - type a character

```
, TYPE A CHARACTER TO THE MAIN TERMINAL
,
, CALLING SEQUENCE:
;       L       R11,CHAR        ; PUT CHARACTER INTO R11
,       SJS     R15,TYO
,
, REGISTERS USED:
,       R11 - HOLD CHARACTER FOR PRINTING  (ANY REG. MAY BE USED)
,       R15 - SUBROUTINE LINKAGE REGISTER
,
; NOTE: THE SPECIFICS OF THIS ROUTINE ARE DEPENDENT ON THE
,       VERSION OF THE INTERFACE BOARD USED

        .DUSR   STATPORT = 0FD
        .DUSR   OUTPORT = 0FE

   TYO: ANDM    R11,007F        ; MASK OUT TRASH
        ST      R11,STATPORT    ; CLEAR EXISTING ACKNOWLEDGE
        AIM     R11,8000        ; SET MSB TO 1
        ST      R11,OUTPORT     ;WRITE CHARACTER
  TYO1: L       R11,OUTPORT     ; CHECK STATUS
        BLT     TYO1            ; WAIT UNTIL READ BY FS-1
        L       R11,OUTPORT     ; CHECK STATUS
        BLT     TYO1            ; WAIT UNTIL READ BY FS-1
        ST      R11,STATPORT    ; ACKNOWLEDGE
        URS     R15             ; RETURN
```

TYI – read a character

```
; READ A CHARACTER FROM THE INTERCOM PORT
;
; CALLING SEQUENCE:
;       SJS     R15,TYI
;       ST      R0,CHAR         ; CHARACTER WILL BE IN R0
;
; REGISTERS USED:
;       R0 - RECEIVES CHARACTER FROM INTERCOM PORT (ANY REG. MAY BE USED)
;       R15 - SUBROUTINE LINKAGE
;
; NOTE: THE SPECIFICS OF THIS ROUTINE ARE DEPENDENT ON THE
;       VERSION OF THE INTERFACE BOARD USED
        .DUSR   INPORT = OFF

   TYI: L       R0,INPORT       ; READ INTERCOM PORT
        BLE     TYI             ; FLUSH NULLS (OTHERWISE USE BLT)
        L       R0,INPORT       ; READ INTERCOM PORT
        BLE     TYI             ; WAIT UNTIL CHAR. IS AVAILABLE
        ST      R0,INPORT       ; ACKNOWLEDGE
        ANDM    R0,007F         ; MASK OFF TRASH
        URS     R15             ; RETURN
```

What is MIL-STD 1750A ?

STANDARD SIXTEEN BIT COMPUTER INSTRUCTION SET ARCHITECTURE (ISA) FOR ALL THREE MILITARY SERVICES

SPECIFIED BY MIL-STD 1750A (USAF), UPDATED BY NOTICE 1, 21 MAY 1982

THE ISA SPECIFIES:

- DATA FORMATS
- INSTRUCTION FORMATS
- ADDRESSING MODES
- REGISTER MODEL
- SUPPORT FEATURES (E.G. STACK, POWER-UP, TIMERS, ETC.)
- MEMORY (INCLUDING MEMORY MANAGEMENT AND BLOCK PROTECT)
- INTERRUPT CONTROL
- INPUT/OUTPUT OPERATIONS
- DETAILED INSTRUCTION SET REQUIREMENTS

THE STANDARD IS AVAILABLE FREE. USE THE ORDER FORM ON THE NEXT PAGE.

BK:5/84

DEPARTMENT OF THE NAVY
NAVAL PUBLICATIONS AND FORMS CENTER
5801 Tabor Ave.
Philadelphia, Pa. 19120
OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

POSTAGE AND FEES PAID
DEPARTMENT OF THE NAVY
DOD - 316

U.S.MAIL

Please self address the above label

SPECIFICATIONS AND STANDARDS REQUISITION

Send the number of copies of documents listed below which are in the DoD Index of Specifications and Standards. Limit 5 line items per request. Requests submitted on this form will speed service. Reorder forms will be enclosed with each shipment. *FORWARD REQUEST TO:* Naval Publications and Forms Center, Code 3015, 5801 Tabor Avenue, Philadelphia, PA 19120.

1. QUANTITY	2. STANDARDIZATION DOCUMENT SYMBOL	3. TITLE *(From DoD Index of Specifications and Standards)*	4. NONAVAILABILITY CODE * *(NPFC use only)*
	MIL-STD-1750A (USAF)	MILITARY STANDARD SIXTEEN BIT COMPUTER INSTRUCTION SET	
		ARCHITECTURE (with notice 1)	

*NONAVAILABILITY CODE EXPLANATION

B – Item temporarily not in stock. Resubmit request in 30 days.
E – Item not identifiable as an active document listed in the DODISS. Direct questions concerning this action to NPFC, Attn: Code 1052 or call (215) 697-3321.
I – For Official Use Only. Submit request via cognizant DoD Inspection Office or Contract Administrator for certification of "need to know."
L – Industry standardization documents will not be furnished by NPFC to commercial concerns. Copies may be purchased from the appropriate Industry Association.

5. SIGNATURE OF REQUESTOR	6. DATE PREPARED *(YY, MM, DD)*	7. CLOSING DATE *(YY, MM, DD) (IFB, RFQ, or RFP)*

DD FORM 1425
82 MAY

Previous editions of this form are obsolete

S/N 0102-LF-001-4252

F9450 DATA TYPES

BITS – can be set, reset, and tested

BYTES – can be loaded, stored, & swapped

SINGLE PRECISION FIXED POINT (one word)

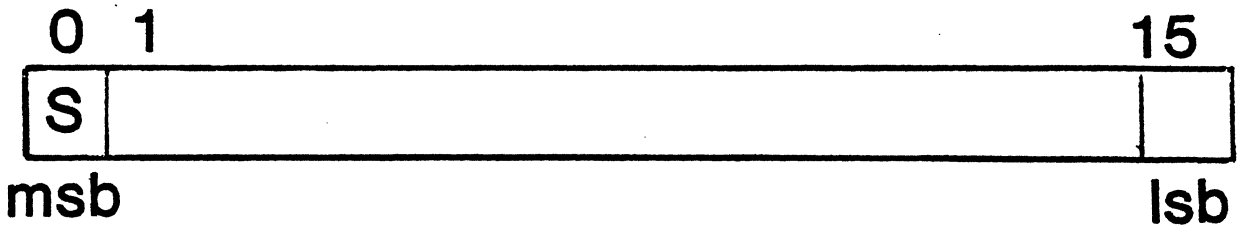
DOUBLE PRECISION FIXED POINT
(two words)

SINGLE PRECISION FLOATING POINT
(two words)

EXTENDED PRECISION FLOATING POINT
(three words)

BK:5/84

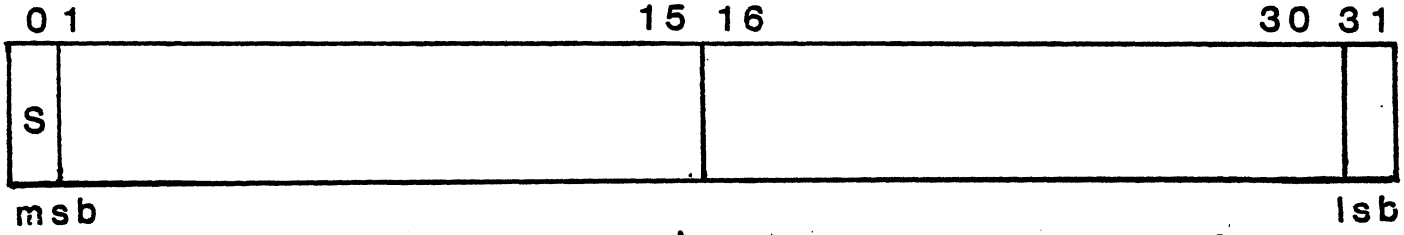
Single Precision Fixed Point Numbers



Integer	16 bit Hex Word
32767	7FFF
32766	7FFE
⋮	⋮
1	0001
0	0000
-1	FFFF
-2	FFFE
⋮	⋮
-32767	8001
-32768	8000

BK:5/84

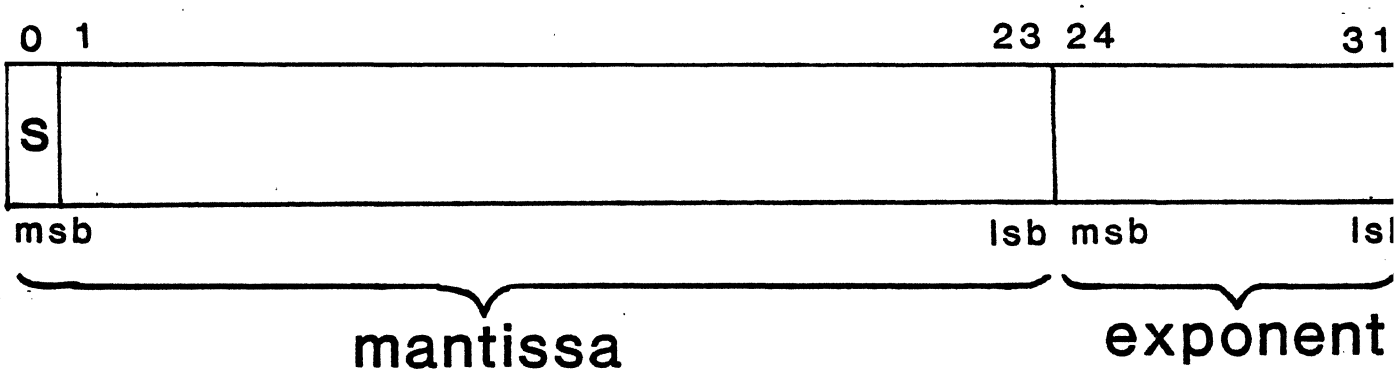
Double Precision Fixed Point Numbers



Integer	32 bit Hex word
2,147,483,647	7FFF FFFF
2,147,483,646	7FFF FFFE
⋮	⋮
1	0000 0001
0	0000 0000
-1	FFFF FFFF
-2	FFFF FFFE
⋮	⋮
-2,147,483,647	8000 0001
-2,147,483,648	8000 0000

BK:5/84

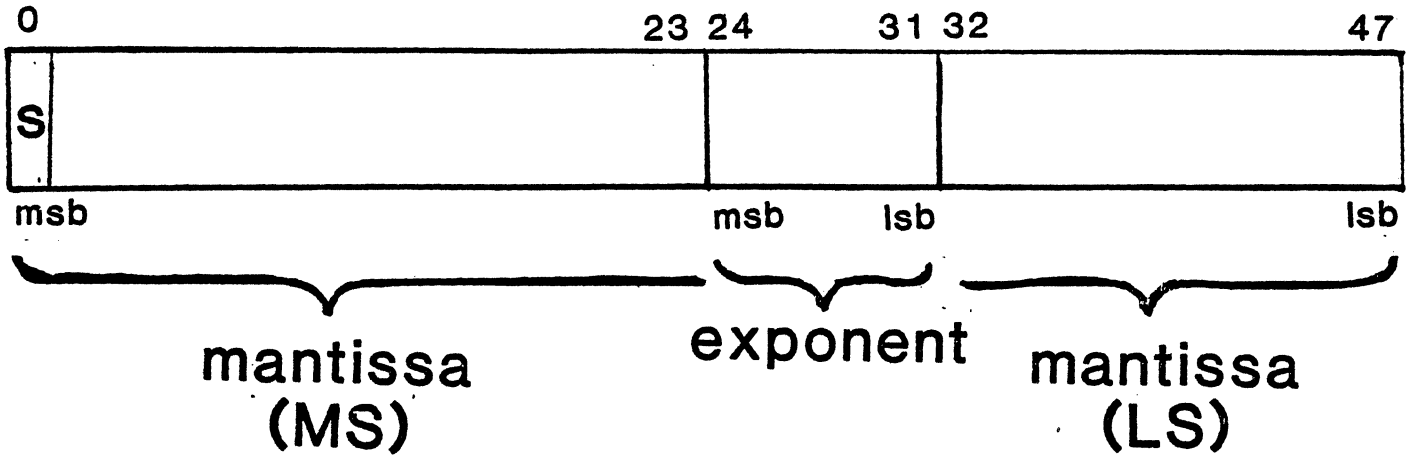
Single Precision Floating Point Numbers



Decimal Number	Hexidecimal Notation	
	Mantissa	Exponent
0.9999998×2^{127}	7FFF FF	7F
0.5×2^{127}	4000 00	7F
0.625×2^{4}	5000 00	04
0.5×2^{-1}	4000 00	FF
0.5×2^{-128}	4000 00	80
0.0×2^{0}	0000 00	00
-1.0×2^{0}	8000 00	00
$-0.5000001 \times 2^{-128}$	BFFF FF	80
-0.7500001×2^{4}	9FFF FF	04

BK:5/84

Extended Precision Floating Point Numbers



Decimal Number	Hexidecimal Notation		
	Mantissa (MS)	Exp	Mantissa (LS)
0.5×2^{127}	400000	7F	0000
0.5×2^{-1}	400000	FF	0000
-1.0×2^{127}	800000	7F	0000
-1.0×2^{-1}	800000	FF	0000
-1.0×2^{-128}	800000	80	0000
-0.75×2^{-1}	A00000	FF	0000

BK:5/84

ADDRESSING MODES

Register Direct

Memory Direct *

Memory Indirect *

Immediate Long *

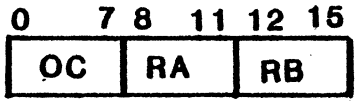
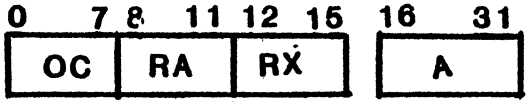
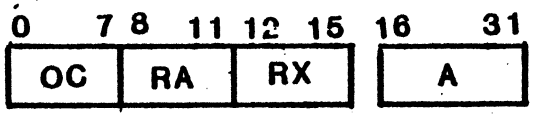
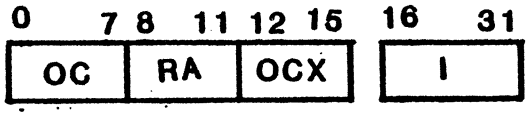
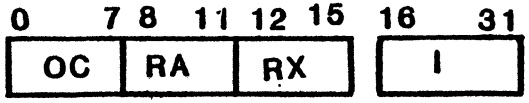
Immediate Short

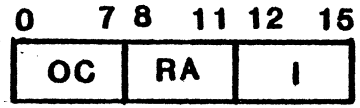
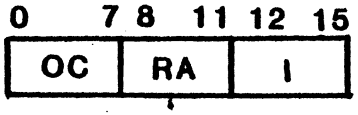
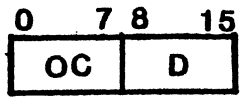
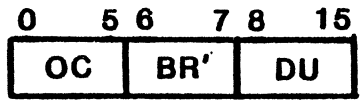
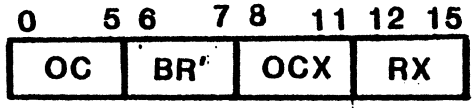
IC Relative

Base Relative *

*can be indexed by a register

BK:5/84

Mode	Format	Der. Oper.	Der. Addr.
Register Direct			
"R"		(RB)	RB
Memory Direct			
"D" Non-indexed (RX is 0)		[A]	A
"DX" Indexed (RX is not 0)		[A+(RX)]	A+(RX)
Memory Indirect			
"I" Non-indexed (RX is 0)		[[A]]	[A]
"IX" Indexed (RX is not 0)		[[A+(RX)]]	[A+(RX)]
Immediate Long			
NOT INDEXIBLE			
"IM"		I	
INDEXIBLE			
"IM" non-indexed (RX is 0)		I	
"IMX" indexed (RX is not 0)		I+(RX)	

Mode	Format	Der. Oper.	Der. Addr.
Immediate Short			
POSITIVE "ISP"		(I + 1)	
NEGATIVE "ISN"		-(I+1)	
IC Relative			
"ICR"	 -128≤D≤127		D + (IC - 1)
Base Relative			
NOT INDEXIBLE "B"	 BR=BR'+12 0≤DU≤255	[DU+(BR)]	DU+(BR)
INDEXIBLE			
"B" Non-indexed (RX is 0)		[(BR)]	(BR)
"BX" Indexed (RX is not 0)		[(BR)+(RX)]	(BR)+(RX)

OVERVIEW OF THE F9450

INSTRUCTION SET

BK:5/84

INTEGER ARITHMETIC/LOGIC

MNEMONIC	ADDRESSING MODE	FUNCTION
A	R,B,BX,ISP,D,DX,IM	Single Precision Add
DA	R,D,DX	Double Precision Add
INCM	D,DX	Increment Memory by Positive Integer
ABS	R	Single Precision Absolute Value
DABS	R	Double Precision Absolute Value
S	R,B,BX,ISP,D,DX,IM	Single Precision Subtract
DS	R,D,DX	Double Precision Subtract
DECM	D,DX	Decrement Memory by Positive Integer
NEG	R	Single Precision Negate
DNEG	R	Double Precision Negate
MS	R,ISP,ISN,D,DX,IM	Single Prec. Multiply - 16 bit product
M	R,B,BX,D,DX,IM	Single Prec. Multiply - 32 bit product
DM	R,D,DX	Double Precision Multiply
DV	R,ISP,ISN,D,DX,IM	Single Prec. Divide - 16 bit dividend
D	R,B,BX,D,DX,IM	Single Prec. Divide - 32 bit dividend
DD	R,D,DX	Double Precision Divide
C	R,B,BX,ISP,ISN,D,DX,IM	Single Precision Compare
CBL	D,DX	Compare Between Limits
DC	R,D,DX	Double Precision Compare
OR	R,B,BX,D,DX,IM	Inclusive OR
AND	R,B,BX,D,DX,IM	AND
XOR	R,D,DX,IM	Exclusive OR
NAND	R,D,DX,IM	Not AND

BK:5/84

FLOATING POINT

MNEMONIC	ADDRESSING MODE	FUNCTION
FA	R,B,BX,D,DX	Floating point Add
EFA	R,D,DX	Extended precision Floating point Add
FABS	R	Floating point ABSolute value
FS	R,B,BX,D,DX	Floating point Subtract
EFS	R,D,DX	Extended prec. Floating point Subtract
FNEG	R	Floating point Negate
FM	R,B,BX,D,DX	Floating point Multiply
EFM	R,D,DX	Extended prec. Floating point Multiply
FD	R,B,BX,D,DX	Floating point Divide
EFD	R,D,DX	Extended prec. Floating point Divide
FC	R,B,BX,D,DX	Floating point Compare
EFC	R,D,DX	Extended prec. Floating point Compare
FIX	R	Convert Floating point to 16-bit Integer
FLT	R	Convert 16-bit Integer to Floating point
EFIX	R	Conv. Ext. Prec. Fl. Pt. to 32-bit Integ
EFLT	R	Conv. 32-bit Integer to Ext. Prec. Fl. P

BIT & SHIFT OPERATIONS

MNEMONIC	ADDRESSING MODE	FUNCTION
SB	R,D,DX,I,IX	Set Bit
RB	R,D,DX,I,IX	Reset Bit
TB	R,D,DX,I,IX	Test Bit
TSB	D,DX	Test and Set Bit
SVBR	R	Set Variable Bit in Register
RVBR	R	Reset Variable Bit in Register
TVBR	R	Test Variable Bit in Register
SLL	R	Shift Left Logical
SRL	R	Shift Right Logical
SRA	R	Shift Right Arithmetic
SLC	R	Shift Left Cyclic
DSLL	R	Double Shift Left Logical
DSRL	R	Double Shift Right Logical
DSRA	R	Double Shift Right Arithmetic
DSLC	R	Double Shift Left Cyclic
SLR	R	Shift Logical, count in Register
SAR	R	Shift Arithmetic, count in Register
SCR	R	Shift Cyclic, count in Register
DSLR	R	Double Shift Logical, count in Register
DSAR	R	Double Shift Arithmetic, count in Register
DSCR	R	Double Shift Cyclic, count in Register

LOAD/STORE/EXCHANGE

MNEMONIC	ADDRESSING MODE	FUNCTION
L	R,B,BX,ISP,ISN,D,DX, IM,IMX,I,IX	SINGLE PRECISION LOAD
DL	R,B,BX,D,DX,I,IX	DOUBLE PRECISION LOAD
EFL	D,DX	EXTENDED PREC. FLOATING POINT LOAD
LUB	D,DX,I,IX	LOAD FROM UPPER BYTE
LLB	D,DX,I,IX	LOAD FROM LOWER BYTE
ST	B,BX,D,DX,I,IX	SINGLE PRECISION STORE
STC	D,DX,I,IX	STORE A NON-NEGATIVE CONSTANT
DST	B,BX,D,DX,I,IX	DOUBLE PRECISION STORE
SRM	D,DX	STORE REGISTER THROUGH MASK
EFST	D,DX	EXTENDED PRECISION FLOATING POINT STORE
STUB	D,DX,I,IX	STORE IN UPPER BYTE
STLB	D,DX,I,IX	STORE INTO LOWER BYTE
XBR	S*	EXCHANGE BYTES IN REGISTER
XWR	R	EXCHANGE WORDS IN REGISTERS
PSHM	S*	PUSH MULTIPLE REGISTERS ONTO THE STACK
POPM	S*	POP MULTIPLE REGISTERS OFF THE STACK
LM	D,DX	LOAD MULTIPLE REGISTERS
STM	D,DX	STORE MULTIPLE REGISTERS
MOV	S*	MULTIPLE WORD MOVE, MEMORY TO MEMORY

* S MEANS SPECIAL FORMAT

PROGRAM CONTROL

MNEMONIC	ADDRESSING MODE	FUNCTION
JC	D,DX,I,IX	Jump on Condition
JS	D,DX	Jump to Subroutine
SOJ	D,DX	Subtract One and Jump
BR	ICR	Branch Unconditionally
BEZ	ICR	Branch if Equal to (Zero)
BLT	ICR	Branch if Less Than (zero)
BLE	ICR	Branch if Less than or Equal to (zero)
BGT	ICR	Branch if Greater Than (zero)
BNZ	ICR	Branch if Not equal to (Zero)
BGE	ICR	Branch if Greater than or Equal to (zero)
BEX	S*	Branch to EXecutive
LST**	D,DX,I,IX	Load STatus
SJS	D,DX	Stack IC and Jump to Subroutine
URS	S*	Unstack IC and Return from Subroutine
NOP	S*	No OPeration
BPT	S*	Breakpoint
BIF	S*	Built In Function (escape code)

* S means Special Format

** Privileged Instruction

BK:5/84

INPUT/OUTPUT INSTRUCTIONS

TIMER CONTROL , INTERRUPT/DMA/FAULT CONTROL

MNEMONIC	ADDRESSING MODE	FUNCTION
XIO	IM, IMX	EXECUTE INPUT/OUTPUT
VIO	D,DX	VECTORED INPUT/OUTPUT
TAS		TIMER A START
TAH		TIMER A HALT
OTA		OUTPUT TIMER A
ITA		INPUT TIMER A
TBS		TIMER B START
TBH		TIMER B HALT
OTB		OUTPUT TIMER B
ITB		INPUT TIMER B
SMK		SET INTERRUPT MASK
CLIR		CLEAR INTERRUPT MASK
ENBL		ENABLE INTERRUPTS
DSBL		DISABLE INTERRUPTS
RPI		RESET PENDING INTERRUPT
SPI		SET PENDING INTERRUPT REGISTER
RMK		READ INTERRUPT MASK
RPIR		READ PENDING INTERRUPT REGISTER
RCFR		READ AND CLEAR FAULT REGISTER
DMAE		DMA ENABLE
DMAD		DMA DISABLE

NOTE: INPUT/OUTPUT INSTRUCTIONS ARE PRIVILEGED

BK:5/84

INPUT/OUTPUT (cont.)

MMU CONTROL, BPU CONTROL & MISCELLANEOUS

MNEMONIC	ADDRESSING MODE	FUNCTION
WIPR		Write Instruction Page Register
WOPR		Write Operand Page Register
RIPR		Read Instruction Page Register
ROPR		Read Operand Page Register
LMP		Load Memory Protect RAM
RMP		Read Memory Protect RAM
MPEN		Memory Protect Enable
WSW		Write Status Word
RSW		Read Status Word
RSN		Reset Normal Power Up discrete
GO		Reset Trigger GO indicator
PO		Programmed Output
PI		Programmed Input
OD		Output Discretes
RCS		Read Console Status
CLC		Clear Console
ESUR		Enable Start Up ROM
DSUR		Disable Start Up ROM
RIC1		Read I/O Interrupt code, Level 1
RIC2		Read I/O Interrupt code, Level 2
RDOR		Read Discrete Output Register
RDI		Read Discrete Input
TPIO		Test Programmed Output
RMFS		Read Memory Fault Status

INSTRUCTION EXECUTION TIMES (20MHz)

Basic Instructions	Register	Direct	Direct Index	Indirect	Immediate	Immediate Short	Base Relative	Base Relative Index
Single Precision								
Load/Store	0.2	0.6	0.6	0.8	0.55	0.35	0.55	0.55
Add/Sub	0.2	0.6	0.6		0.55	0.35	0.55	0.55
Multiply	1.85	2.25	2.25		2.2		2.2	2.2
Divide	4.85	5.1	5.1		5.05		5.05	5.05
Compare	0.4	0.8	0.8		0.75	0.55	0.75	0.75
Set/Reset Bit	0.35	0.8	0.8	1.0				
Double Precision								
Load	0.5	1.1	1.1	1.3			1.05	1.05
Store		0.8	0.8	1.0			0.75	0.75
Add/Sub	0.8	1.1	1.1					
Multiply	6.15	6.45	6.45					
Divide	11.95	12.25	12.25					
Compare	1.05	1.35	1.35					
Floating Point								
Add/Sub*	3.1	3.4	3.4				3.35	3.35
Multiply*	6.0	6.3	6.3				6.25	6.25
Divide*	11.7	12.0	12.0				11.95	11.95
Compare	2.6	2.9	2.9				2.85	2.85
Extended Floating-Point								
Load		1.3	1.3					
Store		1.0	1.0					
Add/Sub*	3.55	3.9	3.9					
Multiply*	12.55	12.9	12.9					
Divide*	24.0	24.35	24.35					
Compare*	2.6	3.25	3.25					
Branch	**Taken**	**Not Taken**						
	0.75	0.2						

SAMPLE PROGRAM -

MATRIX MULTIPLICATION

LEFT MATRIX RIGHT MATRIX RESULT MATRIX

A B C

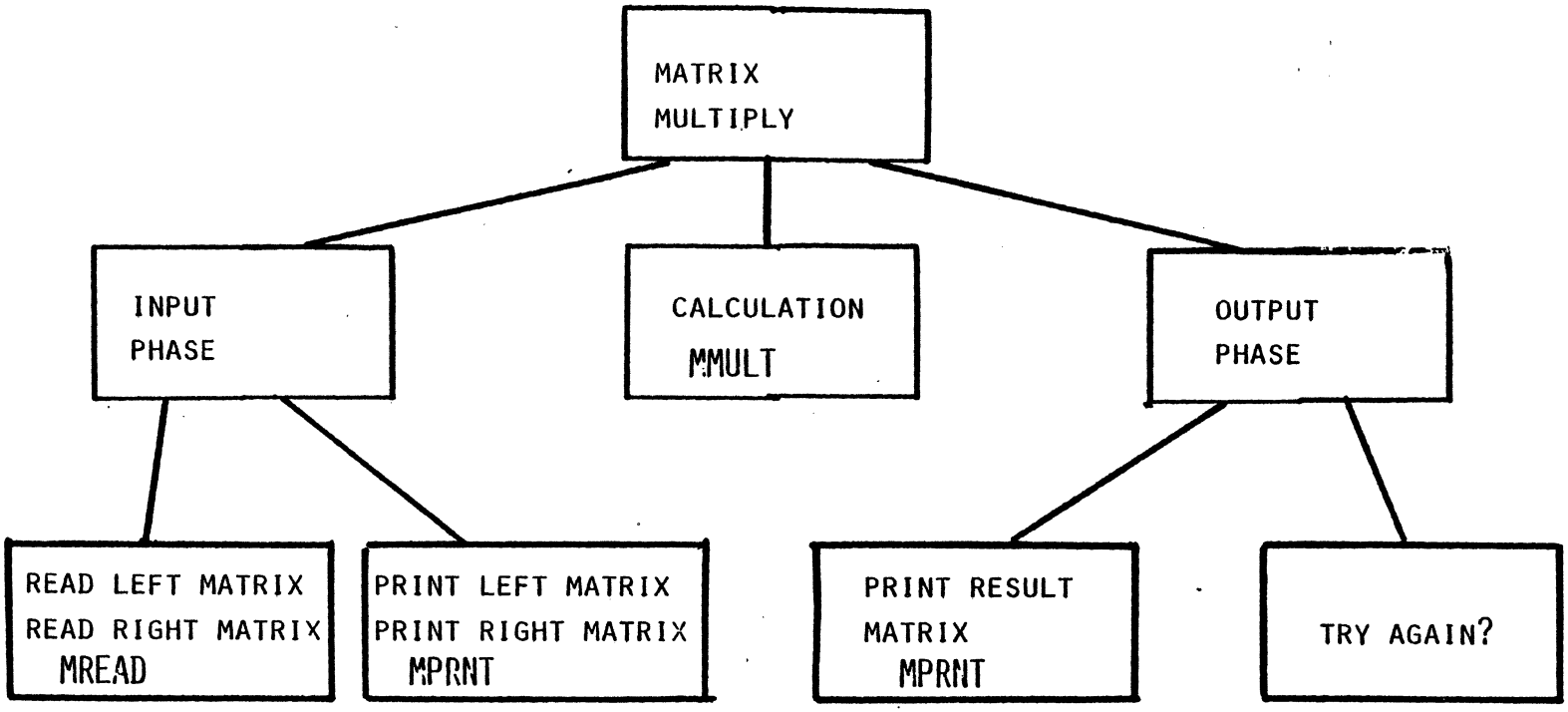
$$\begin{pmatrix} -7 & 11 \\ 10 & -4 \\ 8 & 6 \end{pmatrix} \times \begin{pmatrix} 15 & 6 & -4 \\ -2 & 7 & -3 \end{pmatrix} = \begin{pmatrix} -127 & 35 & -5 \\ 158 & 32 & -28 \\ 108 & 90 & -50 \end{pmatrix}$$

3 x 2 2 x 3 3 x 3

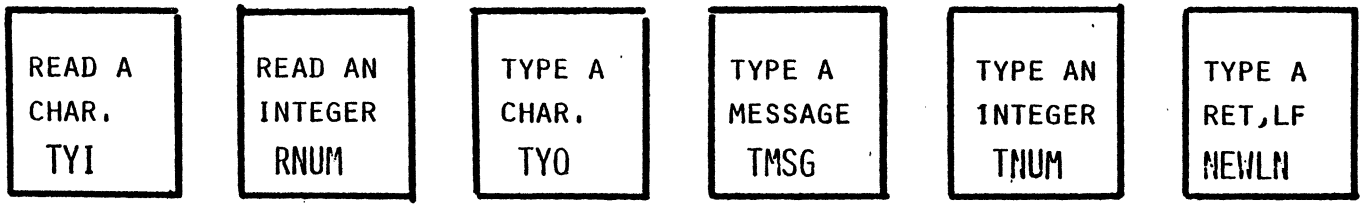
A Pascal Procedure To Multiply Matrices

```
CONST MAXDIM = 10;

TYPE MATRIX = RECORD
                ROWSIZE, COLSIZE: 0..MAXDIM;
                DATA:  ARRAY [1..MAXDIM, 1..MAXDIM] OF INTEGER;
              END;

PROCEDURE MULTMATRIX (LEFT, RIGHT: MATRIX; VAR RESULT: MATRIX);
VAR ROW, COL, TERM: 1..MAXDIM;
    SUM: INTEGER;
BEGIN (*MULTMATRIX*)
  IF LEFT.COLSIZE    RIGHT.ROWSIZE THEN BEGIN
    WRITELN ('MULTIPLICATION NOT POSSIBLE FOR THESE DEMENSIONS');
    RESULT.ROWSIZE:= 0, RESULT.COLSIZE:= 0; END
  ELSE BEGIN (*DO THE MULTIPLICATION*)
    RESULT.ROWSIZE:= LEFT.ROWSIZE; RESULT.COLSIZE:= RIGHT.COLSIZE;
    FOR ROW:= 1 TO RESULT.ROWSIZE DO
      FOR COL:= 1 TO RESULT.COLSIZE DO BEGIN
        SUM:= 0;
        FOR TERM:= 1 LEFT.COLSIZE DO
          SUM:= SUM + LEFT.DATA [ROW, TERM] *RIGHT.DATA [TERM,COL];
        RESULT.DATA [ROW,COL]:= SUM;
      END; (*FOR COL*)
  END; (*ELSE*)
END; (*MULTMATRIX*)
```

BK:5/84



COMMON SUBROUTINES



BK:5/84

MATRIX MULTIPLICATION DEMO

FOR THE F9450 INTERFACE BOARD

```
; MAIN PROGRAM FOR THE MATRIX MULTIPLY DEMONSTRATION.
; THE USER IS ASKED TO ENTER TWO MATRICES OF ANY SIZE
; UP TO 8 BY 8.  THIS PROGRAM PRINTS THE TWO MATRICES,
; MULTIPLIES THEM (IF POSSIBLE), AND PRINTS THE RESULT
; MATRIX.
;
; REGISTERS USED: (MAIN PROGRAM ONLY)
;       R0 - WORKING REGISTER, INCLUDING CHAR. INPUT
;       R11 - CHARACTER TO BE PRINTED
;       R13 - POINTER TO MESSAGE TO BE PRINTED
;       R15 - SUBROUTINE LINKAGE REGISTER
;
; SUBROUTINES CALLED:
;       TMSG - TYPE A MESSAGE
;       NEWLN - TYPE A BLANK LINE
;       TYO - TYPE A CHARACTER
;       TYI - INPUT A CHARACTER
;       MREAD - MATRIX READ
;       MPRNT - MATRIX PRINT
;       MM - MATRIX MULTIPLY
```

BK:5/84

```
        .LOC    0
        JC      7,MULT          ; INTERFACE BOARD STARTS AT 0
        .LOC    1000            ; ROOM FOR A STACK
MULT:   LIM     R15,MULT        ; INITIALIZE SP SO STACK
                                ;   PRECEDES THE CODE
        LIM     R13,MSG1
        SJS     R15,TMSG        ; THIS PROGRAM MULTIPLIES MATRICES
        LIM     R13,MSG2
        SJS     R15,TMSG        ; ENTER SIZE AND TERMS
AGAIN:  LIM     R13,LMSG        ;
        SJS     R15,TMSG        ; ENTER LEFT MATRIX FIRST
        LIM     R0,ASIZ
        PSHM    R0,R0           ; PUSH ADDR. OF A ONTO STACK
        SJS     R15,MREAD       ; READ MATRIX A

        LIM     R13,RMSG
        SJS     R15,TMSG        ; ENTER RIGHT MATRIX NEXT
        LIM     R0,BSIZ
        PSHM    R0,R0           ; PUSH ADDR. OF B ONTO STACK
        SJS     R15,MREAD       ; READ MATRIX B

        LIM     R0,ASIZ
        PSHM    R0,R0           ; PUSH ADDR. OF A ONTO STACK
        SJS     R15,MPRNT       ; PRINT MATRIX A
        SJS     R15,NEWLN       ; PRINT BLANK LINE
        LIM     R11,"X          ;
        SJS     R15,TYO         ; PRINT MULTIPLICATION SIGN
        SJS     R15,NEWLN       ; PRINT BLANK LINE
        LIM     R0,BSIZ
        PSHM    R0,R0           ; PUSH ADDR. OF B ONTO STACK
        SJS     R15,MPRNT       ; PRINT MATRIX B
        SJS     R15,NEWLN       ; PRINT BLANK LINE
        LIM     R11,"=
        SJS     R15,TYO         ; PRINT EQUAL SIGN
        SJS     R15,NEWLN       ; PRINT BLANK LINE
```

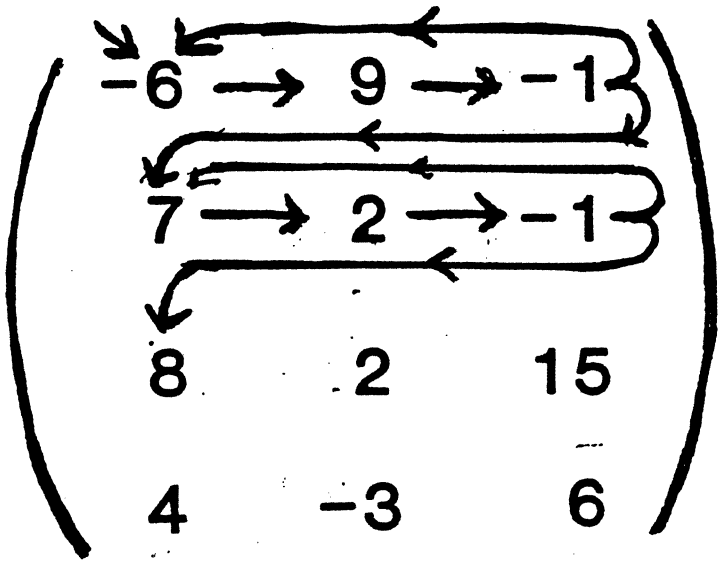
```
        L       R0,ASIZ+1       ; # COL OF LEFT MATRIX
        C       R0,BSIZ         ; COMPARE WITH # ROWS RIGHT MATRIX
        BEZ     MULT1           ; IF EQUAL, PROCEED WITH MULT.
        LIM     R13,MSG7        ;
        SJS     R15,TMSG        ; OTHERWISE, PRINT ERROR MESSAGE
        BR      TRY             ; BRANCH TO TRY AGAIN

MULT1:  SJS     R15,MM          ; MULTIPLY THE MATRICES
        LIM     R0,CSIZ
        PSHM    R0,R0           ; PUSH ADDR. OF C ONTO STACK
        SJS     R15,MPRNT       ; PRINT THE RESULT MATRIX

TRY:    LIM     R13,MSG6
        SJS     R15,TMSG        ; TRY AGAIN?
        SJS     R15,TYI         ; READ RESPONSE
        ANDM    R0,005F         ; CONVERT TO UPPER CASE
        CIM     R0,"Y
        BEZ     AGAIN           ; IF YES, REPEAT PROGRAM
        BPT                     ; OTHERWISE, HALT

MSG1:   .TXT    "<0D><0A>THIS PROGRAM MULTIPLIES TWO MATRICES
                WITH INTEGER ENTRIES.<0D><0A><0>"
MSG2:   .TXT    "YOU WILL BE ABLE TO SPECIFY THE SIZE AND ENTRIES
                FOR EACH MATRIX.<0D><0A><0>"
LMSG:   .TXT    "<0D><0A>FIRST YOU WILL ENTER THESIZE FOR THE
                LEFT MATRIX. <0D><0A><0>"
RMSG:   .TXT    "<0D><0A>AND NOW PLEASE SPECIFY THE SIZE OF THE
                RIGHT MATRIX.<0D><0A><0>"
MSG6:   .TXT    "<0D><0A>WOULD YOU LIKE TO TRY AGAIN? (Y/N) <0>"
MSG7:   .TXT    "<0D><0A>MULTIPLICATION IMPOSSIBLE DUE TO MATRIX
                DIMENSIONS.<0D><0A><0>"
ASIZ:   .BLK    2               ; # ROWS AND # COLS, LEFT MATRIX
AMAT:   .BLK    64              ; UP TO 64 TERMS
BSIZ:   .BLK    2               ; # ROWS AND # COLS, RIGHT MATRIX
BMAT:   .BLK    64              ; UP TO 64 TERMS
CSIZ:   .BLK    2               ; # ROWS & # COLS FOR RESULT
CMAT:   .BLK    64              ; UP TO 64 TERMS
```

MREAD - matrix read

```
; THIS SUBROUTINE READS A MATRIX WHOSE ADDRESS HAS BEEN PASSED
; ON THE STACK.  THE USER IS ASKED TO ENTER THE NUMBER OF ROWS
; AND THE NUMBER OF COLUMNS, BOTH OF WHICH ARE STORED AS THE
; FIRST TWO ENTRIES.  THE USER THEN ENTERS THE TERMS GOING
; ACROSS THE ROWS. ALL TERMS MUST BE SINGLE PRECISION INTEGERS.
;
; CALLING SEQUENCE:
;       LIM     R0,MTADR
;       PSHM    R0,R0           ; PUSH_MATRIX ADDR. ONTO STACK
;       SJS     R15,MREAD
;
; REGISTERS USED:
;       R0,R1,R2 - WORKING REGISTERS
;       R11 - CHARACTER TO BE PRINTED
;       R13 - POINTER TO MESSAGE TO BE PRINTED
;       R15 - SUBROUTINE LINKAGE REGISTER
;
; SUBROUTINES CALLED:
;       TMSG - TYPE A MESSAGE
;       NEWLN - TYPE A BLANK LINE
;       TYO - TYPE A CHARACTER
;       TYI - READ A CHARACTER
;       RNUM - READ A NUMBER (SIGNED INTEGER)
;       GETNM - INTERNAL SUBROUTINE TO FETCH MATRIX DIMENSIONS
;
; THE FOLLOWING TEXT MESSAGES ARE PRINTED BY THIS ROUTINE

ROWMSG: .TXT    "How many rows? (1..8) <0>"
COLMSG: .TXT    "How many columns? (1..8) <0>"
MSG3:   .TXT    "<0D><0A>Now enter all terms (signed integers)
                        going across the rows of the matrix.<0>"
MSG4:   .TXT    "<0D><0A>You will enter one term per line at the
                        ' ' prompt.  If you make an error,<0>"
MSG5:   .TXT    "<0D><0A>a '?' will appear indicating the term
                        must be re-entered.<0D><0A><0>"
```

```
MREAD:  POPM    R0,R1           ; GET ADDRESS OF MATRIX
        PSHM    R0,R0           ; RESTORE RETURN ADDRESS
        ST      R1,MPTR         ; STORE AS MATRIX POINTER
        LIM     R13,ROWMSG      ; HOW MANY ROWS?
        SJS     R15,GETNM       ; GET THE NUMBER
        LR      R2,R0           ; MOVE TO R2
        LIM     R13,COLMSG      ; HOW MANY COLUMNS?
        SJS     R15,GETNM       ; GET THE NUMBER
        MSR     R2,R0           ; R2 CONTAINS TOTAL # OF TERMS
        LIM     R13,MSG3        ;
        SJS     R15,TMSG        ; ENTER TERMS ACROSS ROWS
        LIM     R13,MSG4
        SJS     R15,TMSG        ; ONE TERM PER PROMPT LINE
        LIM     R13,MSG5
        SJS     R15,TMSG        ; A ? INDICATES AN ERROR

IREAD:  SJS     R15,NEWLN
        LIM     R11,">          ; PREPARE TO READ EACH INTEGER
        SJS     R15,TYO         ; PRINT PROMPT CHAR.
        SJS     R15,RNUM        ; READ INTEGER
        POPM    R0,R0           ; POP NUMBER OFF STACK
        STI     R0,MPTR         ; AND STORE IT
        INCM    1,MPTR          ; ADVANCE MATRIX POINTER
        SOJ     R2,IREAD        ; IF NOT DONE, READ NEXT VALUE
        SJS     R15,NEWLN       ; OTHERWISE, PRINT BLANK LINE
        URS     R15             ; AND RETURN

GETNM:  SJS     R15,TMSG        ; PRINT MESSAGE
        SJS     R15,TYI         ; FETCH SINGLE CHAR. RESPONSE
        ANDM    R0,000F         ; CONVERT TO DIGIT
        STI     R0,MPTR         ; STORE IN MATRIX
        INCM    1,MPTR          ; ADVANCE MATRIX POINTER
        SJS     R15,NEWLN       ; GO TO NEXT LINE
        URS     R15             ; RETURN

MPTR:   0                       ; POINTER TO MATRIX ENTRY
```

REPEATED SCANNING OF A ROW



SIZ: 4

MAT:

3

-6

9

-1

7

2

-1

8

2

15

4

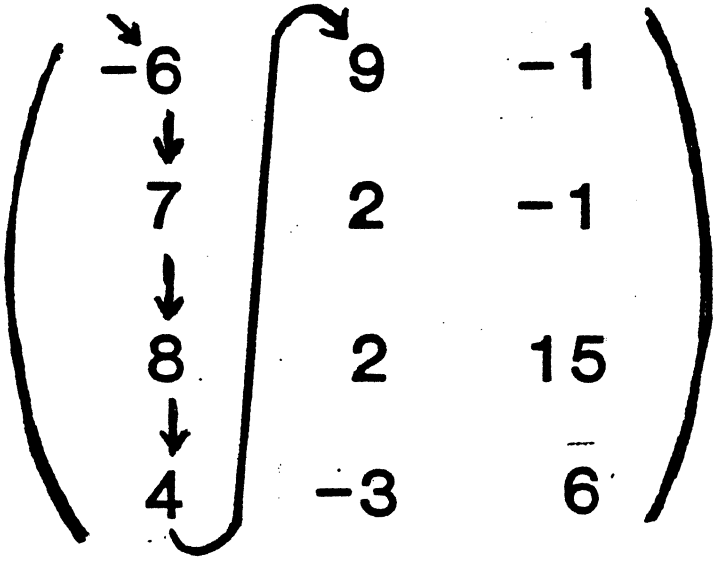
-3

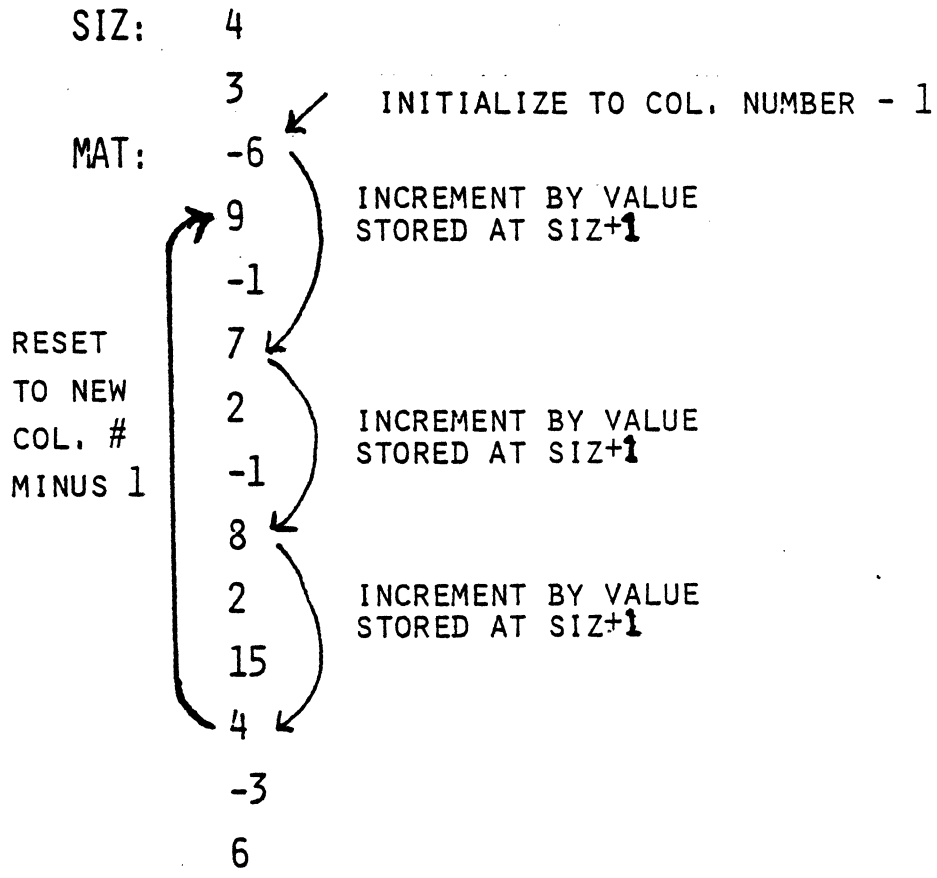
6

RELOAD START OF ROW POINTER

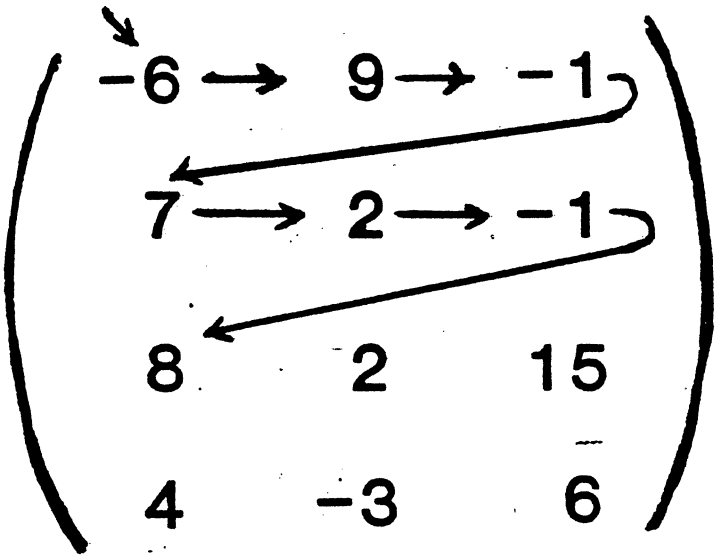
RELOAD START OF ROW POINTER

SCANNING OF SUCCESSIVE COLUMNS





SCANNING AN ENTIRE MATRIX



SIZ: 4
3

MAT: -6
9
-1
7
2
-1
8
2
15
4
-3
6

MM - matrix multiplication

```
; THIS SUBROUTINE MULTIPLIES A MATRIX ON THE LEFT, CALLED A,
; BY A MATRIX ON THE RIGHT, CALLED B.  THE RESULTANT MATRIX
; IS CALLED C.  IT IS ASSUMED THE CALLING PROGRAM HAS VERIFIED
; THAT THE NUMBER OF COLUMNS OF A MATCHES THE NUMBER OF ROWS
; OF B.  THE FIRST TWO ENTRIES FOR EACH MATRIX ARE THE NUMBER
; OF ROWS FOLLOWED BY THE NUMBER OF COLUMNS.  MATRIX ENTRIES,
; WHICH MUST BE SINGLE PRECISION INTEGERS, ARE ENTERED ACROSS
; ROWS.  THE LABELS ASIZ, BSIZ, AND CSIZ ARE ASSUMED.
;
; CALLING SEQUENCE:
;       SJS     R15,MM
;
; REGISTERS USED:
;       R0 - ROW COUNT
;       R1 - COLUMN COUNT
;       R2 - TERM COUNT
;       R3 - SUM OF PARTIAL PRODUCTS
;       R5 - OPERAND FROM MATRIX A
;       R7 - OPERAND FROM MATRIX B
;       R12 - INDEX TO MATRIX A
;       R13 - INDEX TO MATRIX B
;       R14 - INDEX TO MATRIX C
;
; SUBROUTINES USED:  NONE
```

BK:5/84

```
MM:     L       R0,ASIZ         ; RESULT MATRIX HAS THE SAME
        ST      R0,CSIZ         ; NUMBER OF ROWS AS A (LEFT)
        L       R1,BSIZ+1       ; AND THE SAME NUMBER OF
        ST      R1,CSIZ+1       ; COLUMNS AS B (RIGHT)
        XORR    R12,R12         ; CLEAR INDEX REG. FOR A
        XORR    R14,R14         ; CLEAR INDEX REG. FOR C
RWSCN:  ST      R12,RWNDX       ; SAVE TO ROW INDEX FOR A
        L       R1,CSIZ+1       ; LOAD THE COLUMN COUNT
CLSCN:  L       R13,BSIZ+1      ; CALCULATE THE COLUMN NUMBER
        SR      R13,R1          ; WITH ZERO FOR THE FIRST COL.
        L       R12,RWNDX       ; INITIALIZE ROW INDEX FOR A
        L       R2,ASIZ+1       ; INITIALIZE THE TERM COUNTER
        XORR    R3,R3           ; CLEAR THE SUM REGISTER
TMSCN:  L       R5,AMAT,R12     ; LOAD OPERAND FROM A
        AISP    R12,1           ; INCREMENT INDEX FOR A
        L       R7,BMAT,R13     ; LOAD OPERAND FROM B
        MSR     R5,R7           ; MULTIPLY THE OPERANDS
        AR      R3,R5           ; ADD PARTIAL PROD. TO SUME
        L       R7,BSIZ+1       ; FETCH NUMBER OF COLS IN B
        AR      R13,R7          ; INCREMENT B INDEX BY THIS AMOUNT
        SOJ     R2,TMSCN        ; IF NOT DONE, GET ANOTHER TERM
        ST      R3,CMAT,R14     ; OTHERWISE, STORE SUM IN C
        AISP    R14,1           ; INCREMENT INDEX FOR C
        SOJ     R1,CLSCN        ; IF NOT DONE, GO TO NEXT COLUMN
        SOJ     R0,RWSCN        ; IF NOT DONE, GO TO NEXT ROW
        URS     R15             ; OTHERWISE FINISHED, SO RETURN

RWNDX:  0                       ; START OF ROW INDEX FOR A
```

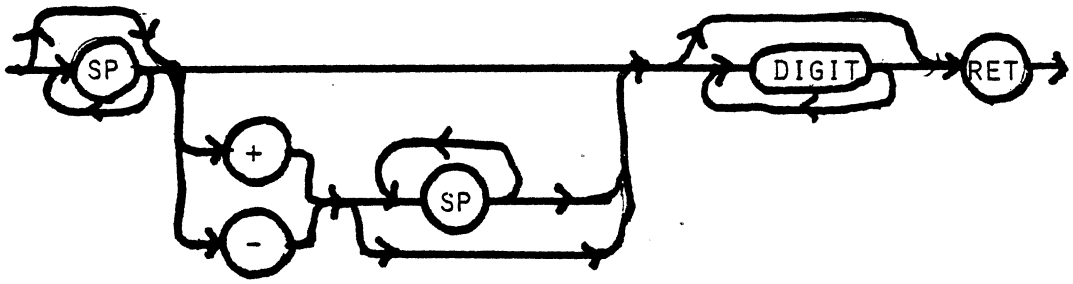
TNUM - type a number (signed integer)

```
; TYPE A NUMBER (SIGNED INTEGER) TO THE INTERCOM PORT
;
; CALLING SEQUENCE:
;       L       R11,NUMBR               ; PUT INTEGER IN R11
;       SJS     R15,TNUM                ; R15 IS LINKAGE REG.
; REGISTERS USED:
;       R10 - TEMPORARY WORKING REGISTER
;       R11 - CONTAINS NUMBER TO BE TYPED
;
TNUM:   LIM     R10,-1          ; MARK STACK WITH NEG. DIGIT
        PSHM    R10,R10         ;
        LR      R10,R11         ; MOVE NUMBER TO WORKING REG.
        BGE     TNUM1           ; SKIP IF POSITIVE INTEGER
        SB      0,NFLG          ; OTHERWISE, SET NEG. FLAG
        NEG     R10,R10         ; CONVERT TO POSITIVE NUMBER
TNUM1:  DVIM    R10,RADIX       ; DIVIDE BY RADIX
        BEZ     TNUM2           ; IF QUOTIENT=0, GO TO PRINTOUT
        PSHM    R11,R11         ; PUT REMAINDER ON THE STACK
        BR      TNUM1           ; DO NEXT DIGIT
TNUM2:  TB      0,NFLG          ; NEG. FLAG SET?
        BEZ     TNUM3           ; BRANCH IF NOT SET
        PSHM    R11,R11         ; OTHERWISE, SAVE LEADING DIGIT
        LIM     R11,"-          ; PRINT A MINUS SIGN
        SJS     R15,TYO
        POPM    R11,R11         ; RESTORE LEADING DIGIT
        RB      0,NFLG          ; RESET NEG. FLAG
TNUM3:  TBR     0,R11           ; END OF NUMBER?
        BLT     TNUMX           ; IF SO, EXIT
        CIM     R11,10.
        BGE     TNUM5           ; HANDLE HEX DIGITS A...F
        AIM     R11,"0          ; FORM ASCII CHAR. 0...9
TNUM4:  SJS     R15,TYO         ; PRINT THE DIGIT
        POPM    R11,R11         ; POP NEXT DIGIT OFF STACK
        BR      TNUM3           ; LOOP UNTIL FINISHED
TNUM5:  AIM     R11,"A-10.      ; FORM ASCII CHAR. A...F
        BR      TNUM4           ; BRANCH TO PRINT
TNUMX:  USR     R15             ; RETURN

NFLG:   0
```

BK:5/84

RNUM - read a number (signed integer)

```
; READ A NUMBER (SIGNED INTEGER) FROM THE INTERCOM PORT
;
; THE FORM OF THE INTEGER IS
;
;
;
;
;
;
;
;
;
;
;
;
;
; CALLING SEQUENCE:
;       SJS     R15,RNUM        ; READ INTEGER
;       POPM    R0,R0           ; POP RESULT INTO REG.
;
; REGISTERS USED:
;       R0 - WORKING REGISTER FOR CHARACTER BEING PROCESSED
;       R1 - HOLDS CURRENT INTEGER VALUE
;       R11 - USED TO PRINT A ? IN CASE OF AN ERROR (CALLS TYO)
;       R15 - SUBROUTINE LINKAGE REGISTER
;
;
; NOTE: AN OVERFLOW ERROR WOULD CAUSE AN INTERRUPT TO AN
;       OVERFLOW ROUTINE.  THERE IS NO PROVISION FOR CHARACTER
;       CORRECTION VIA BACKSPACE.  IF AN ILLEGAL CHARACTER IS
;       ENTERED, A ? IS PRINTED AND ENTRY MUST BE RESTARTED.
;
```



```
RNUM:   XORR    R1,R1           ; CLEAR TOTAL
        ST      R1,NFLAG        ; CLEAR NEGATIVE FLAG
RNUM1:  SJS     R15, TYI        ; FETCH CHARACTER
        CIM     R0,32.          ; SPACE?
        BEZ     RNUM1           ; IF SO, GET NEXT CHAR
        CIM     R0,"-           ; MINUS SIGN?
        BNZ     RNUM2           ; IF NOT, KEEP CHECKING
        SB      0,NFLAG         ; OTHERWISE, SET NEG. FLAG
        BR      RNUM3           ; LOOK FOR SPACES
RNUM2:  CIM     R0,"+           ; PLUS SIGN?
        BNZ     RNUM4           ; IF NOT, CHECK FOR DIGIT
RNUM3:  SJS     R15,TYI         ; SEARCH FOR NONBLANK CHAR.
        CIM     R0,32.          ; SPACE?
        BEZ     RNUM3           ; IF SO, GET NEXT CHAR.
RNUM4:  CIM     R0,13.          ; RETURN?
        BEZ     RNUM5           ; IF SO, EXIT
        CIM     R0,"0
        BLT     ERROR           ; ERROR IF LESS THAN "0
        CIM     R0,"9
        BGT     ERROR           ; ERROR IF GR. THAN "9
        SIM     R0,"0           ; CONVERT TO INTEGER
        MISP    R1,RADIX        ; MULT. PREV. TOTAL BY RADIX
        AR      R1,R0           ; ADD CURRENT DIGIT VALUE
        SJS     R15,TYI         ; FETCH NEXT CHARACTER
        BR      RNUM4           ; CHECK FOR RET. OR DIGIT
RNUM5:  TB      0,NFLAG         ; NEG. FLAG SET?
        BNZ     RNUM6           ; SKIP IF NOT SET
        NEG     R1,R1           ; OTHERWISE, NEGATE RESULT
RNUM6:  POPM    R0,R0           ; POP RETURN ADDRESS
        PSHM    R0,R1           ; PUSH RESULT ON STACK
        URS     R15             ; RETURN

ERROR:  SJS     R15,NEWLN       ; GO TO NEXT LINE
        LIM     R11,"?
        SJS     R15,TYO         ; PRINT ? PROMPT
        BR      RNUM            ; START OVER
```

MPRNT - matrix print subroutine

```
; THIS SUBROUTINE PRINTS THE MATRIX WHOSE ADDRESS IS PASSED ON
; THE STACK.  IT IS ASSUMED THAT THE FIRST TWO ENTRIES ARE THE
; NUMBER OF ROWS FOLLOWED BY THE NUMBER OF COLUMNS.  MATRIX
; ENTRIES, WHICH MUST BE SINGLE PRECISION INTEGERS, ARE ENTERED
; ACROSS ROWS.
;
; CALLING SEQUENCE:
;       LIM     R0,MTADR        ; PUT MATRIX ADDRESS IN R0
;       PSHM    R0,R0           ; AND PUSH ONTO STACK
;       SJS     R15,MPRNT
; REGISTERS USED:
;       R0,R1 - WORKING REGISTERS
;       R11 - USED FOR PRINT SUBROUTINES
;       R15 - LINKAGE REGISTER
;
MPRNT:  POPM    R0,R1           ; GET MATRIX ADDRESS
        PSHM    R0,R0           ; RESTORE RETURN ADDRESS
        ST      R1,ADPTR        ; STORE MATRIX ADDRESS POINTER
        LI      R0,ADPTR        ; LOAD NUMBER OF ROWS
        INCM    1,ADPTR         ; ADVANCE ADDRESS POINTER
        LI      R1,ADPTR        ; LOAD NUMBER OF COLUMNS
        ST      R1,COLCT        ; SAVE AS COLUMN COUNT
RWPRT:  L       R1,COLCT        ; RESTORE COLUMN COUNT
        SJS     R15,NEWLN       ; ADVANCE TO NEW LINE
CLPRT:  INCM    1,ADPTR         ; ADVANCE ADDRESS POINTER
        LI      R11,ADPTR       ; FETCH NEXT ENTRY
        SJS     R15,TNUM        ; AND PRINT IT
        LIM     R11,0009        ; LOAD ASCII FOR A TAB
        SJS     R15,TYO         ; AND PRINT IT
        SOJ     R1,CLPRT        ; LOOP TO PRINT NEXT COL. ENTRY
        SOJ     R0,RWPRT        ; LOOP TO PRINT NEXT ROW
        SJS     R15,NEWLN       ; FINISH WITH A NEW LINE
        URS     R15             ; RETURN

ADPTR:  0                       ; MATRIX ADDRESS POINTER
COLCT:  0                       ; NUMBER OF COLUMNS
```

BK:5/84

SUPPORT DEVICES & APPLICATIONS

MEMORY INTERFACE

F9451 MEMORY MANAGEMENT UNIT

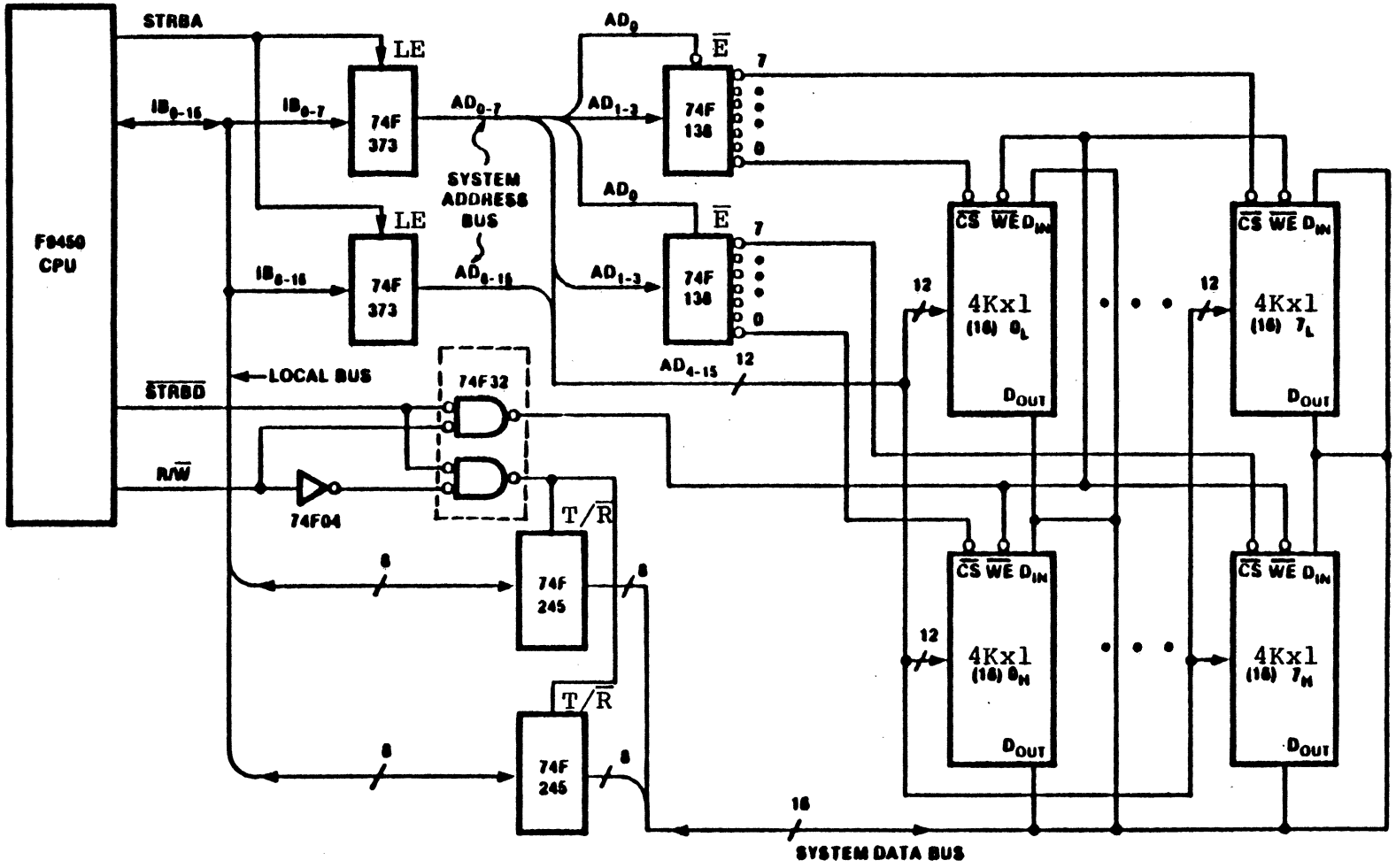
- Features & Map Structure
- Block Diagram
- CPU/MMU Configuration

F9452 BLOCK PROTECT UNIT

- Features & Look-Up Table
- Block Diagram
- CPU/BPU Configuration
- CPU/MMU/BPU Configuration

MULTIPROCESSOR/BUS ARBITERS

F9450-Based System
with High-Speed 64K Memory

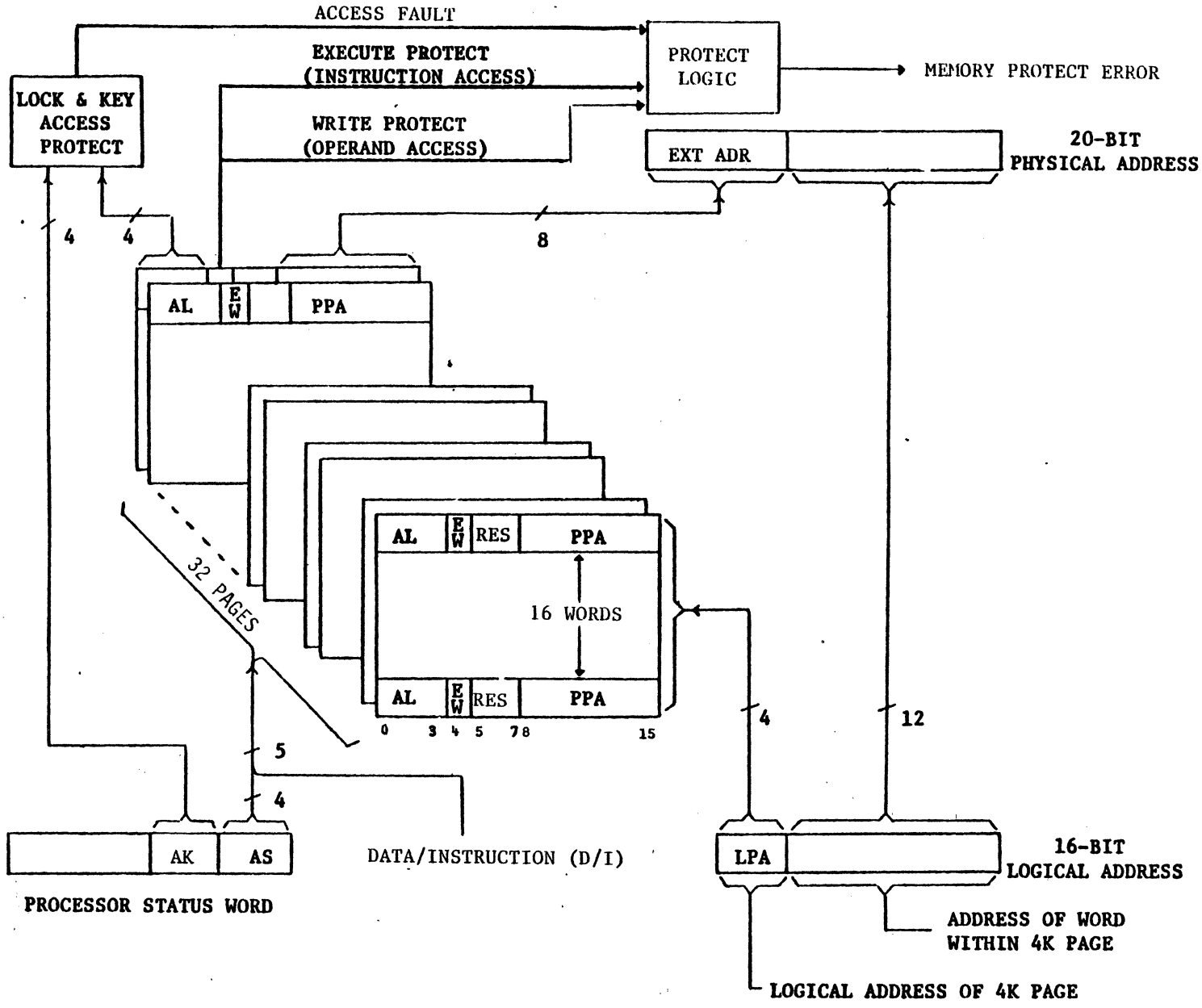


JH:1/85

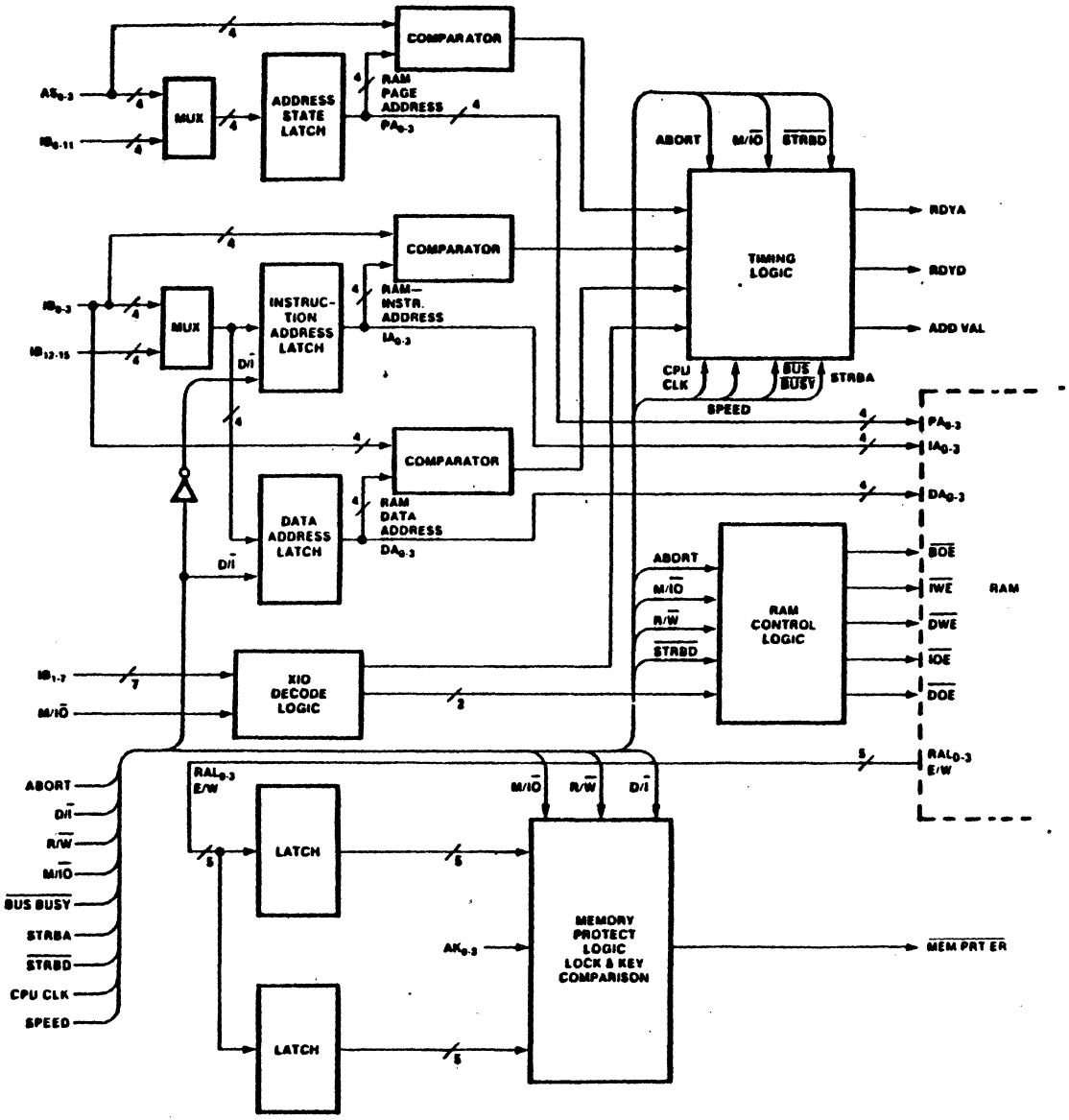
F9451 Memory Management Unit (MMU)

- MEMORY PROTECTION IN 4K WORD PAGES
  - ACCESS KEY/ACCESS LOCK MATCH
  - WRITE PROTECT IN OPERAND SPACE
  - EXECUTE PROTECT IN INSTRUCTION SPACE
- LOGICAL TO PHYSICAL ADDRESS TRANSLATION FOR INSTRUCTION AND OPERAND SPACES
- 2 M WORD SEPARATE DATA & INSTRUCTION ADDRESSING SPACE
  - EXPANDABLE TO 16 M WORD FOR NON-MIL-STD 1750A APPLICATIONS
- INTERNAL CACHE MECHANISM ENHANCES PERFORMANCE
- MAP RAM (FOUR F93479s) AND BUS TRANSCEIVER (TWO 54/74F245s) INTERFACE

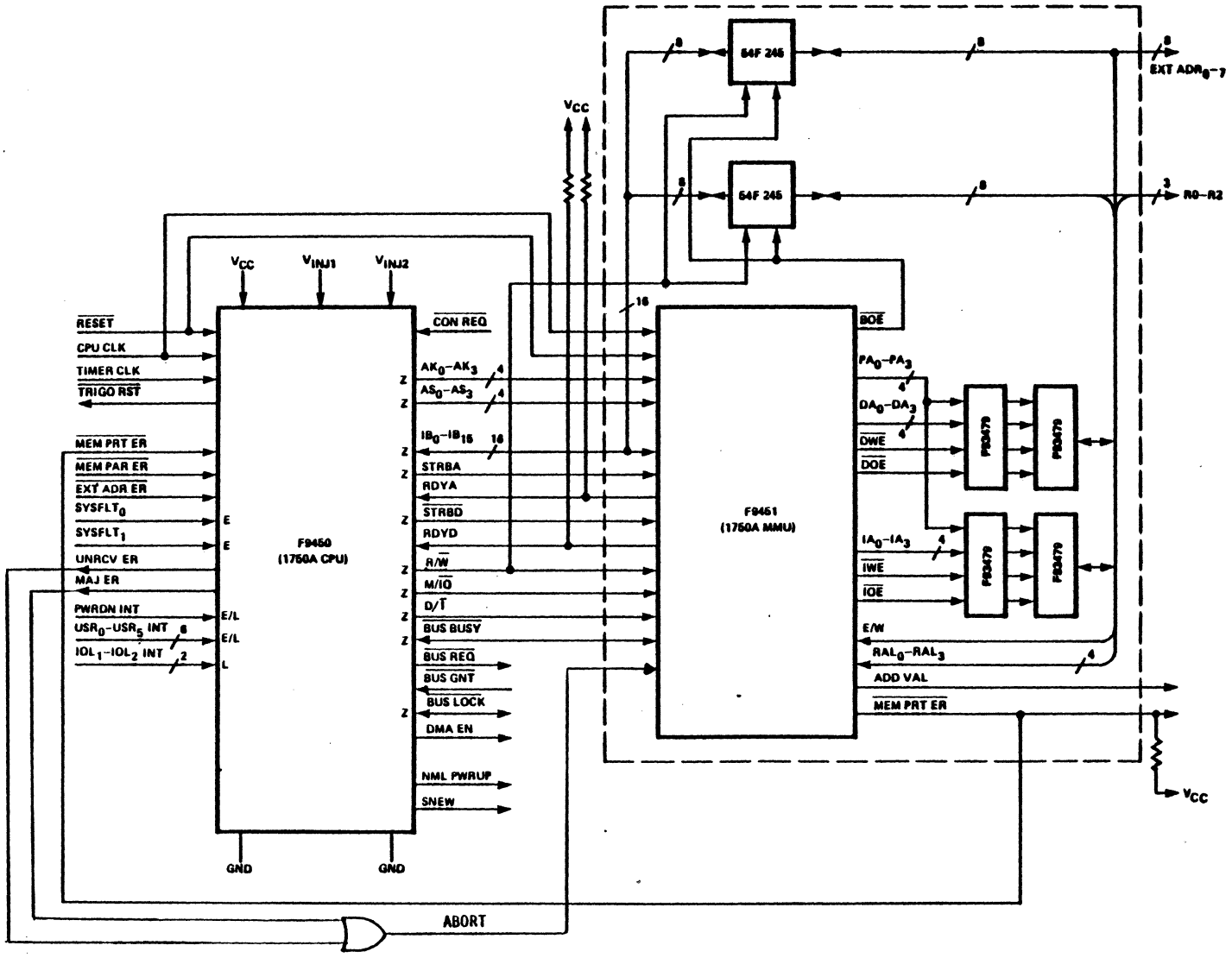
F9451 MAP STRUCTURE



F9451 MMU Block Diagram



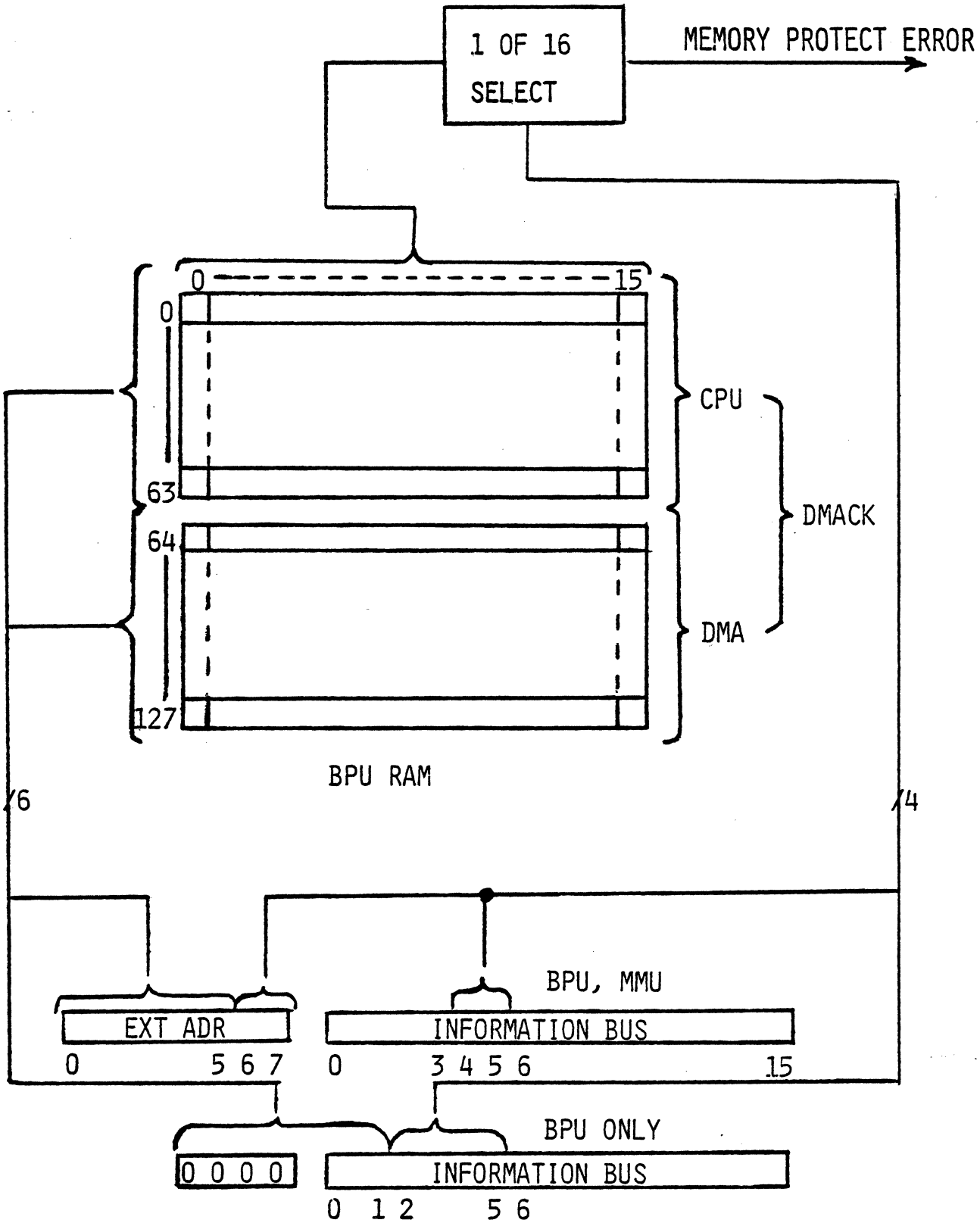
CPU/MMU Configuration



F9452 Block Protect Unit (BPU)

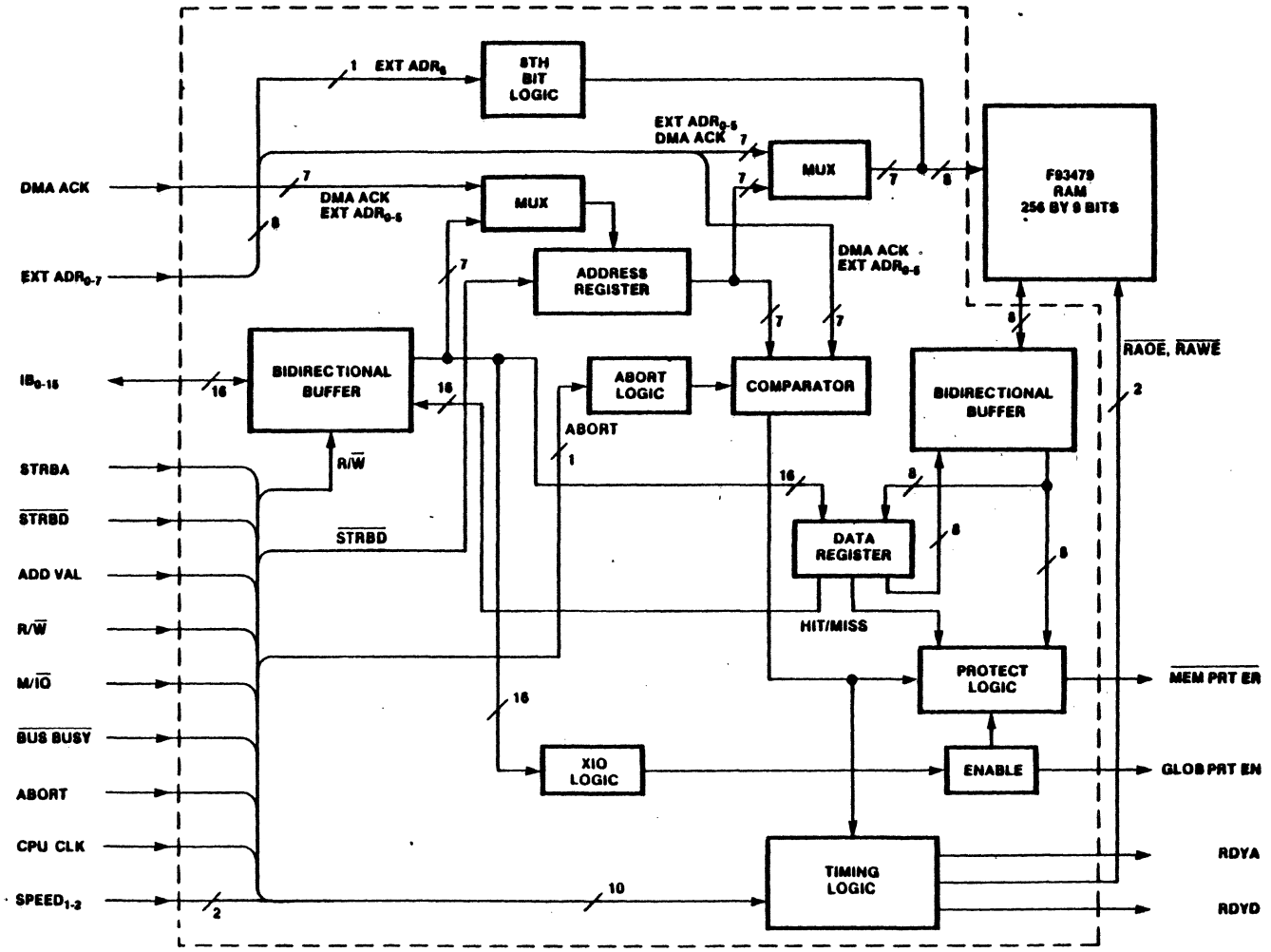
- WRITE PROTECTION OF PHYSICAL MEMORY SPACE FOR CPU AND DMA DATA SPACES
- PROTECTION IN 1K WORD BLOCKS
- INTERNAL CACHE SCHEME ENHANCES PERFORMANCE
- PROTECTION LOOK-UP TABLE RAM INTERFACE
- CPU ONLY OR CPU/MMU SYSTEM COMPATIBILITY

BPU LOOK-UP TABLE STRUCTURE

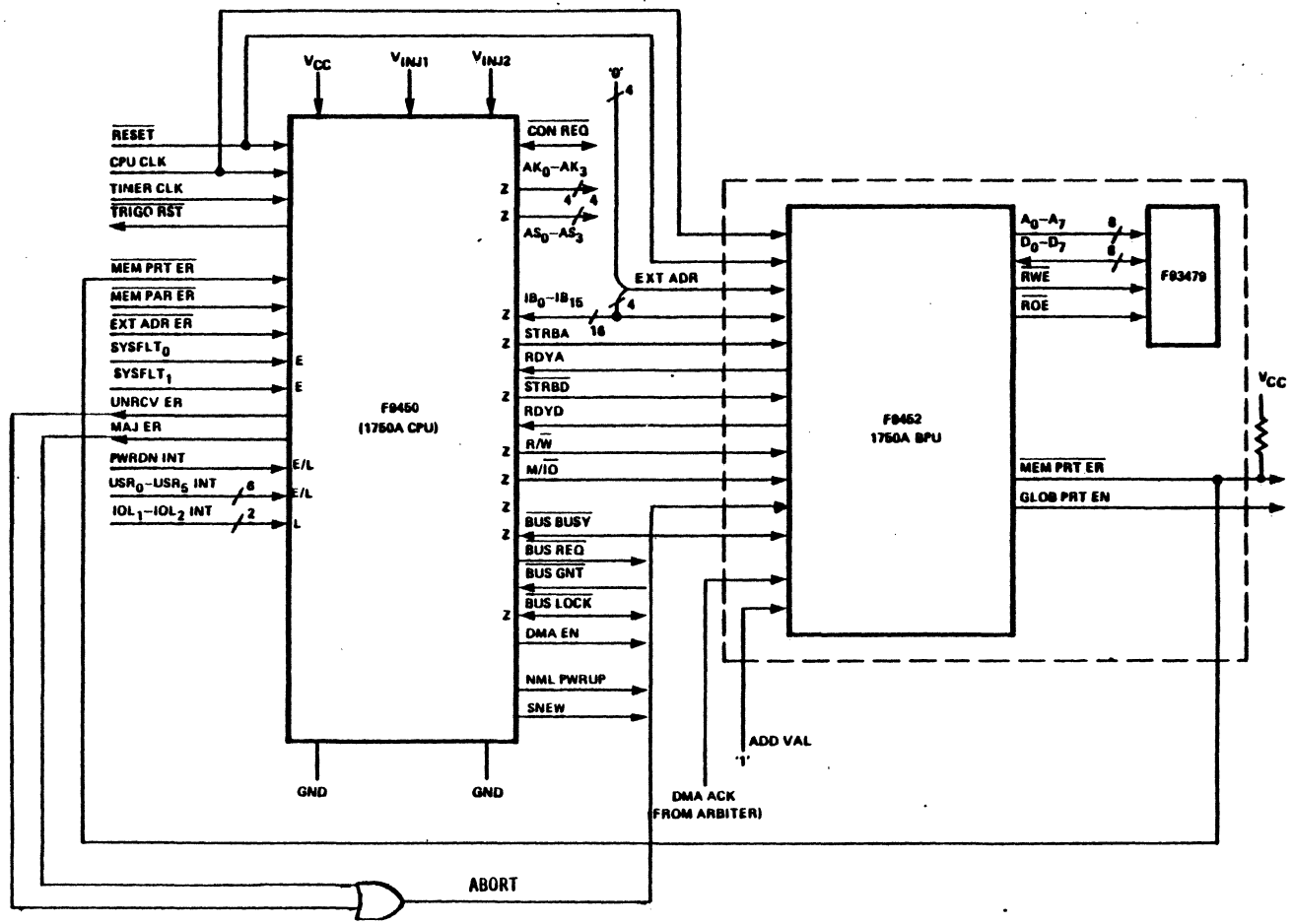


RM:5/84

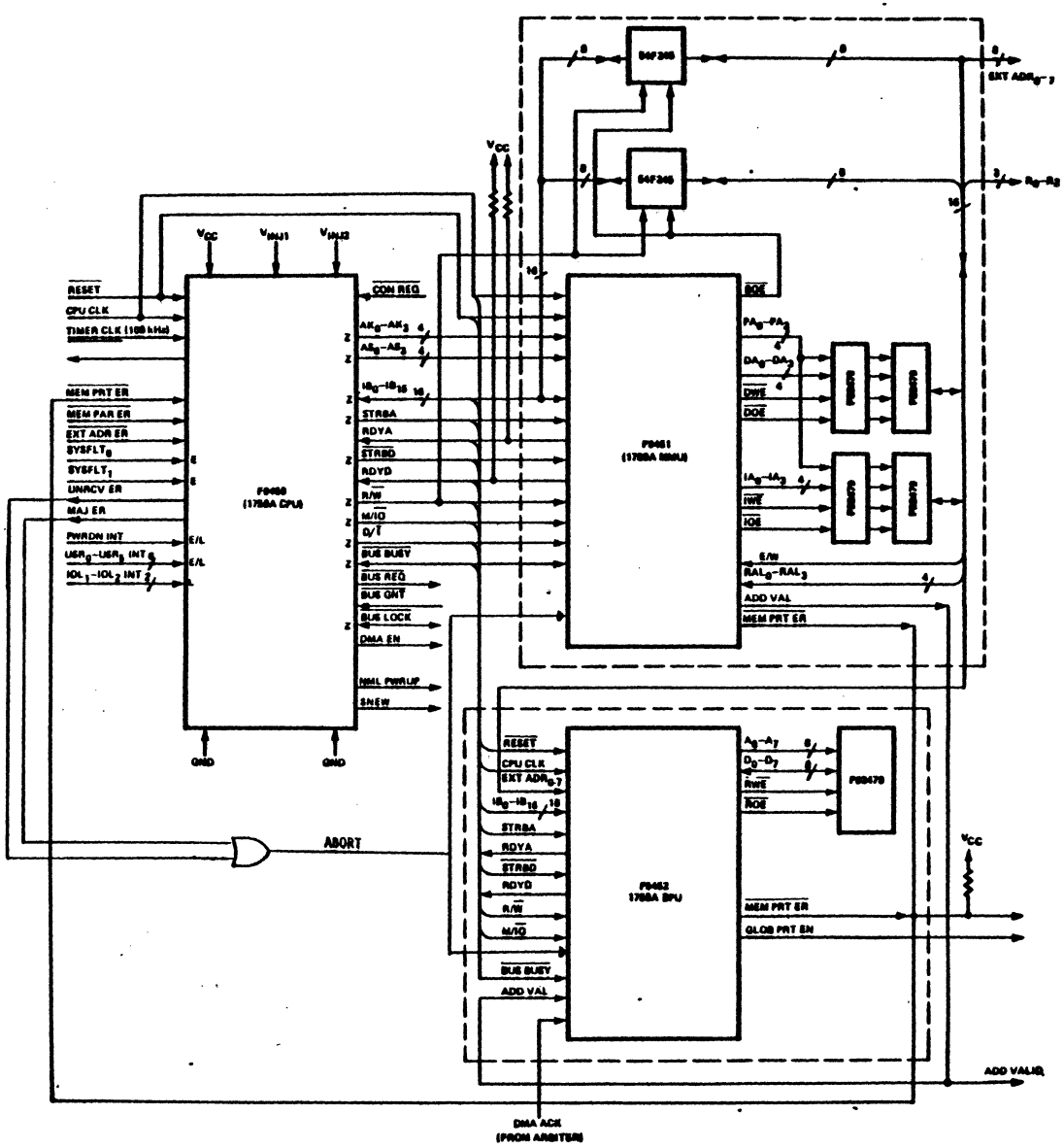
F9452 BPU Block Diagram



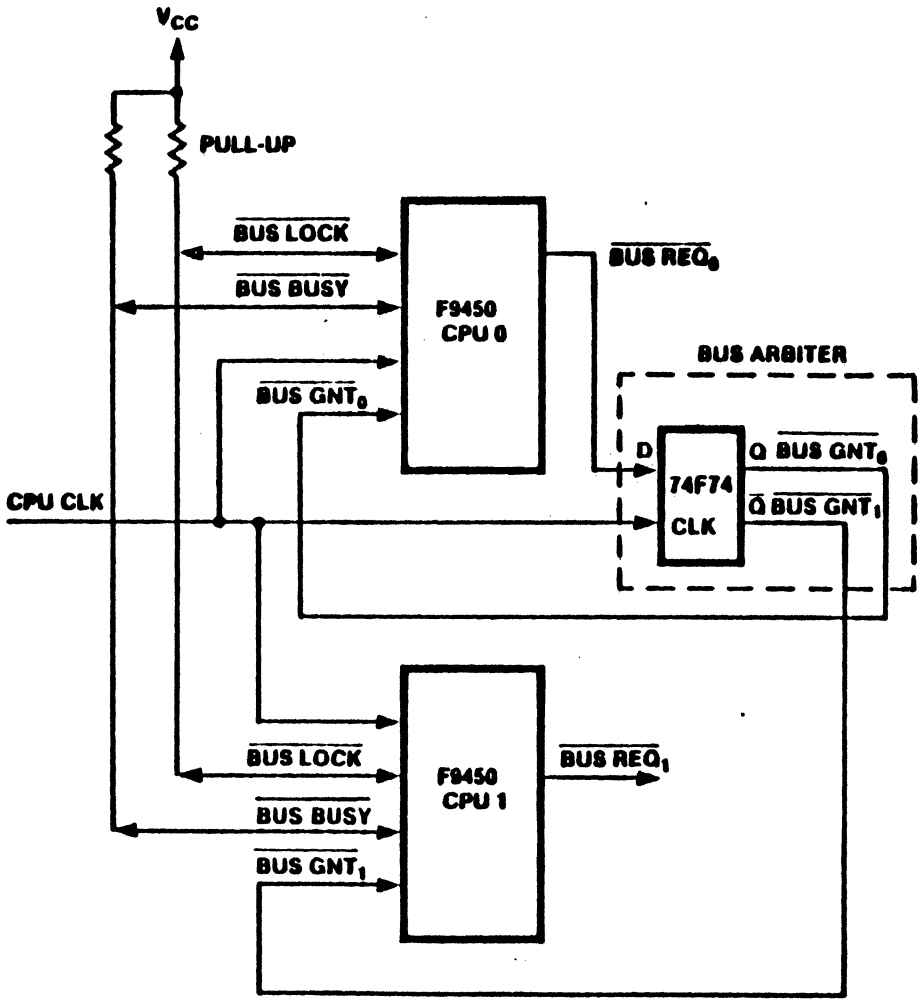
CPU/BPU Configuration



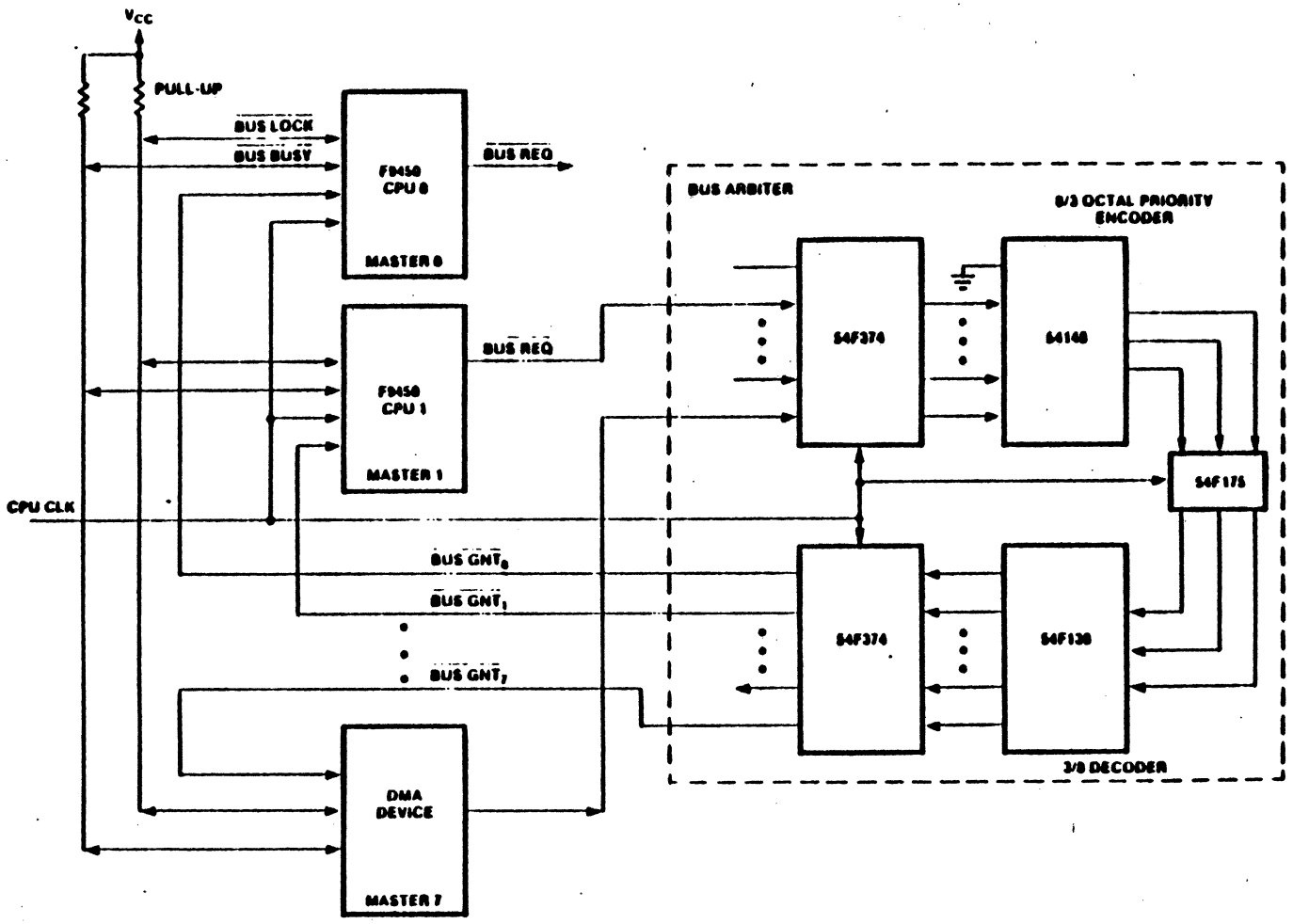
CPU/MMU/BPU Configuration



Dual F9450 CPU with External Arbiter



F9450 External Arbitration for up to 8 Bus Masters



DEVELOPMENT ENVIRONMENT

FAIRCHILD SUPPORT

- VAX/VMS* Based Tools
- SINGLE BOARD COMPUTER 50

THIRD PARTY SOFTWARE SUPPORT

- VAX* Based Tools
- Real Time Executives

THIRD PARTY HARDWARE SUPPORT

- Logic Analyzers
- Emulators

* VAX and VAX/VMS are trademarks of Digital Equipment Corporation.

F9450/VAX CROSS SOFTWARE

MACRO50

RELOAD50

LIBEDIT

SVTOLDM

VPEPLINK

LDMTOSV

MACRO50 / RELOAD50

Symbolic Instructions, Addresses & Data

Arithmetic/Logic Expression Evaluation

Conditional Assembly

Complete Macro Capability

Cross-Referencing

External Symbols

Absolute or Relocatable Object Code

SINGLE BOARD COMPUTER 50 (SBC50)

HIGH PERFORMANCE EVALUATION AND PROTOTYPING OF F9450 BASED SYSTEMS

F9450 CPU

16K x 16 BIT HIGH SPEED DUAL PORT RAM EXPANDABLE TO 64K ON BOARD

16K x 16 BIT EPROM EXPANDABLE TO 32K ON BOARD

OPTIONAL F9451 MMU AND F9452 BPU

DUAL ASYNCHRONOUS RS232C COMPATIBLE COMMUNICATION PORTS (19.2K BAUD MAX)

iSBX* COMPATIBLE I/O BUS

IEEE 796 COMPATIBLE SYSTEM BUS

NATIVE BUS FOR COMMUNICATION WITH CUSTOM MEMORY AND F9450 I/O DEVICES

SOFTWARE MONITOR WITH VAX/VMS** INTERFACE FOR DEBUGGING AND DATA UPLOAD/DOWNLOAD

* iSBX IS A TRADEMARK OF INTEL CORP.

** VAX/VMS ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION.

THIRD PARTY

SOFTWARE SUPPORT

VAX SUPPORT

1750A Assemblers

Jovial Compilers

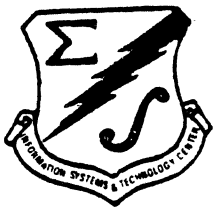
Linkers/Loaders

Simulators

Interactive Debuggers

BK:5/84

LANGUAGE CONTROL FACILITY



Ada®-JOVIAL Newsletter

Vol. 7, No. 1 **January 1985**

The Air Force requires the JOVIAL (J73) computer language to be used on all avionics embedded computer systems until the DoD-wide language, Ada, becomes available. To support this requirement the Air Force goal is to standardize and control JOVIAL. This Newsletter supports this goal by disseminating information about JOVIAL, standardization and language control activities, training, compilers and tools, development efforts, JOVIAL applications, and user services.

The JOVIAL Language Control Facility is operated by the Language Control Branch, **ASD/ADOL, Computer Operations Division, Information Systems & Technology Center,** Wright-Patterson AFB OH 45433-6503, (513) 255-4472/4473 (AV 785-4472). Support is provided by **SofTech, Inc.** (513) 429-3241.

POINTS OF CONTACT

Language Control Agent
Lt Col Joseph Dangerfield
ASD-AFALC/AXT (513) 255-5941
Wright-Patterson AFB OH 45433-6503 AV 785-5941

ARPANET: LCA @ WPAFB-JALCF

Language Control Facility
LCF Manager and Ada Task Leader
Ms Pat Knoop

JOVIAL Task Leader
Mrs Georgeanne Chitwood

ASD/ADOL (513) 255-4472/4473
Wright-Patterson AFB OH 45433-6503 AV 785-4472

ARPANET: LCF @ WPAFB-JALCF

Embedded Computer Standardization Program Office
Lt Col Kent Murphy
ASD-AFALC/AXTS (513) 255-5945
Wright-Patterson AFB OH 45433-6503 AV 785-5945

ARPANET: ECSPO @ WPAFB-JALCF

J73 AND 1750/1750A SOFTWARE

Many Government agency support software tools are available for release to AdaJUG and 1750 Users Group members. In most cases, users may obtain this software from the specified Government contact point by submitting a Terms and Conditions form and a blank magnetic tape. The tape containing the software will be returned to the user with accompanying documentation. Where the LCF has been informed of a distribution charge, it is so noted.

The following table lists each tool, its host and target machines, and its projected delivery date. Also shown are contact reference numbers which identify specific contact points for each tool. These reference numbers correspond to the numbers for the contact points listed on page 7.

TOOL	SPONSOR	DEVELOPER	HOST	TARGET	VERSION	DELIVERY[1,2]	CONTACT
I. 1750/1750A Tools							
Macro Preprocessor, Macro Library, Assembler, Linker, and Simulator	ECSPO	McDonnell Douglas	PDP-11/60, VAX-11/780, IBM 370, DEC-10	1750A	2	Completed	6
Assembler (MPP, SML, MSA)	ECSPO	McDonnell Douglas	VAX-11/780	1750A	3	Completed	6
Assembler (MPP, SML, MSA)	ECSPO	McDonnell Douglas	VAX-11/780	1750A	4	Completed	6
Macro Preprocessor, Assembler, Linker, and Simulator	ASD	CDC (Rehosting MDAC Tool Set)	CDC Cyber 170, 6000 Series	1750A		Completed	17
Assembler (ALAP)	ECSPO	TRW	DEC-10, IBM 370	1750A	5	Completed	6
Assembler (ALAP)	ECSPO	TRW	IBM 370	1750A	6	NOV 84	6
Assembler (ALAP)	ECSPO	TRW	DEC-10	1750A	6	DEC 84	6
Simulator (SIMS0A)	ECSPO	TRW	DEC-10, IBM 370	1750A	3	Completed	6
Simulator (SIMS0A)	ECSPO	TRW	VAX-11/780	1750A	4	Completed	6
Simulator (SIMS0A)	ECSPO	TRW	VAX-11/780	1750A	5	Completed	6
Simulator (SIMS0A)	ECSPO	TRW	IBM 370	1750A	5	NOV 84	6
Linker (EM)[3] (ALINKS)	ECSPO	TRW	DEC-10, IBM 370	1750A	3	Completed	6
Linker (ALINKS)	ECSPO	TRW	VAX-11/780	1750A	4	Completed	6
Linker (ALINKS)	ECSPO	TRW	IBM 370	1750A	5	Completed	6
Reformatter (RFMTR)	ECSPO	TRW	VAX-11/780	1750A	1	Completed	6
Architectural Test Program	AFWAL	TRW	(Loaded in 1750A Computers)		Change Notice 1	Completed	4
Dual Meta Assembler Linker	F-16 SPO	PSS	DEC-10, IBM 370, VAX-11/780, Others	1750A		Completed	1
Functional Simulator	F-16 SPO	PSS	IBM 370, VAX-11/780	1750A		Completed	1
1750A Instruction Set Architecture Simulation	Intermetrics[4]	Intermetrics	Apple, DEC-10, IBM 370, VAX-11/780, Others	1750A	Change Notice 1	Completed	18
1750A Link Editor & Macro Assembler	Intermetrics[4]	Intermetrics	Apple, DEC-10, IBM 370, VAX-11/780, Others	1750A		Completed	18
Macro Assembler, Linker	F-16 SPO	ACT	IBM 370, VAX-11/780	1750A		NOV 84	13
Customized Extended Memory Linker plus Macro Assembler	PSS	PSS	VAX-11/780, IBM 370, DEC-10/20, Others	1750A		Completed	1
JOVIAL Interactive Debugger (RAID)	PSS	PSS	VAX-11/780, Tektronix 8561	1750A		JUL 84	1
Realtime Operating System (ESCAPE)	PSS	PSS	1750A	1750A		AUG 84	1
FORTRAN 77 Compiler	ACT	ACT	VAX/VMS	1750A		SEP 84	13
Versatile Realtime Executive (VRTX/1750)	Hunter & Ready	Hunter & Ready	VAX, IBM, Tektronix, others	1750A		Completed	24
II. Tools for Other Embedded Computer Targets							
Dual Meta Assembler/Linker	F-16 SPO	PSS	DEC-10, IBM 370, VAX-11/780, Others	Z-8002		Completed	1
Functional Simulator	F-16 SPO	PSS	IBM 370, VAX-11/780	Z-8002		Completed	1
Macro Assembler, Linker	F-16 SPO	ACT	IBM 370, VAX-11/780	Z-8002		NOV 84	13

J73 AND 1750/1750A SOFTWARE (Continued)

TOOL	SPONSOR	DEVELOPER	HOST	TARGET	VERSION	DELIVERY[1,3]	CONTACT
III. J73 Support Tools							
J73 Interactive Debugger	ECSPO	TRW	DEC-10	DEC-10	3	DEC 84	6
Integrated Support Software System [4]	AFWAL	General Dynamics (GD)	VAX-11/780	VAX-11/780 & AVSAIL		JAN 85	14
J73AVS	RADC, ECSPO	GRC	IBM 370, VAX-11/780	N/A		Completed	8
Code Auditor	RADC	PSS	IBM 370	N/A	011882	Completed	8
PSL	ASD/ADOL	SofTech	IBM 370	N/A	B (2.0)	Completed	8
JCVS (1589A, B, C)	ASD/ADOL	SofTech	N/A	N/A	7	SEP 84	23
JOVIAL (J73) Workbooks	ASD/ADOL	SofTech	N/A	N/A	1589A, 1589B	Completed	15
Communication Software Development System	RADC	CSC	DEC-20 (Initial)	DEC-20		Completed	19
J73/Ada Microprocessor Study RADC-TR-82-61, AD A116352	RADC	SEA	N/A	N/A		Completed	2
J73 Ada Translator Study, AD A120472	RADC	PSS	N/A	N/A		Completed	2
JOVIAL (J73) Video Tapes	RADC	SofTech	N/A	N/A	1589A	Completed	15
JOVIAL (J73) Benchmarks	ASD/ADOL	SofTech	IBM 370	N/A	1589B	Completed	8
1750A Hardware Benchmarks	PSS	PSS	N/A	N/A	1750A	Completed	1
1750A Interactive Debugger	PSS	PSS	VAX-11/780	N/A	1750A	JUL 84	1
Integrated Tools System	AFSCF	System Dev. Corp.	IBM 370	IBM 370	1	Completed	21
Programming Standards & Conventions	AFSCF	System Dev. Corp.	N/A	N/A	N/A	N/A	22

1589A	1589B	SPONSOR	DEVELOPER	HOST	TARGET	VERSION	DELIVERY	CONTACT
IV. J73 Compilers								
	X	ECSPO	SEA	DEC-10	DEC-10, MIL-1750A	3	Validated	6
	X	ECSPO	SEA	DEC-10	MAGIC 362F,[6] AN/AYK-15[6], MIL-1750A	3	Validated	6
	X	ECSPO	SEA	IBM 370, 303X 43XX	MIL-1750A, IBM 370	3	Validated	6
	X	ECSPO	PSS	VAX-11/780	VAX-11/780, MIL-1750A	3	Validated	6
X		RADC	SofTech	IBM 370	IBM 370	9.5	Validated	8
	X	Boeing [5]	SofTech	IBM 370	IBM 370		Completed	16
	X	Boeing [5]	SofTech	IBM 370	MIL-1750A		Terminated	16
X		AD/DIS	GD/SofTech	IBM 370/3033	DIS/Z-8002	9.1	Completed	8
X		MIRASM [7]	GD/SofTech	IBM 370	DIS/Z-8002	76A6740-010	Completed	3
X		Army Pershing	Martin M./SofTech	IBM 370	Bendix 930		OCT 81	9
X		BMO/Peacekeeper	SofTech	IBM 370	MECA, IBM 370		TBD	5
	X	BMO/Peacekeeper	SEA TLD	VAX-11/780	VAX-11/780, PDP-11/70, ROLM 16668, WSC (Weapon System Controller)		TBD	20

J73 AND 1750/1750A SOFTWARE (Continued)

1589A	1589B	SPONSOR	DEVELOPER	HOST	TARGET	VERSION	DELIVERY	CONTACT
IV. J73 Compilers (continued)								
	X	TI	SEA	IBM 4341	TI-990, TI-9900	3.002	Validated	12
	X	INTEL[4]	PSS	IBM 370, 3033, 4341	INTEL 8086 (iAPX 86 family)	1	Completed	1
	X	INTEL[4]	PSS	DEC-10	INTEL 8086 (iAPX 86 family)	1	Completed	1
	X	SM-ALC [5]	Singer/PSS	IBM 4341	CP-2EX		Validated	11
	X	SM-ALC [5]	Singer/PSS	IBM 4341	MIL-1750A		Validated	11
	X	F-16 SPO	ACT	IBM 370, 303X, 43XX	Z-8002, MIL-1750A		Validated	13
	X	F-16 SPO	PSS	IBM 370, 303X, 43XX	Z-8002, MIL-1750A		NOV 84	10
	X	Martin-Marietta	SEA	DEC-10	Z-8002		Validated	12
	X	Boeing [5]	PSS	IBM 370	MIL-1750A		Completed	1
	X	Boeing	PSS	VAX-11/780	MIL-1750A		Completed	1
	X	F-16 SPO	ACT	VAX-11/780	Z-8002, MIL-1750A		TBD	13
	X	GTE/SEA	SEA	DEC-20, VAX-11/780	Z-8002		Completed	12
	X	AFWAL/GTE	SEA	DEC-20	Z-8001		Completed	12
	X	GTE/SEA	SEA	DEC-10/20	VAX-11/780		Completed	12
	X	SEA	SEA	VAX-11/780	VAX-11/780		Completed	12
	X	SEA	SEA	VAX-11/780	MIL-1750A		Completed	12
	X	SEA	SEA	DEC-20	DEC-10		Completed	12
	X	PSS	PSS	IBM 370, 303X, 43XX	IBM, Improved, Self-Compiling		Completed	1
	X	SEA	SEA	VAX-11/780	Z-8002		Completed	12
	X	Litton	PSS	VAX-11/780	MIL-1750A & ZILOG		Completed	1
	X	PSS	PSS	IBM 370	MIL-1750A, New, High Performance		Completed	1
	X	Honeywell	PSS	VAX-11/780	MIL-1750A, New, High Performance		Completed	1
	X	SEA	SEA	VAX-11/780	Z-8001		Completed	12
	X	SEA	SEA	VAX-11/780	VAX-11/780 Global Optimizer		JUN 84	12
	X	SEA	SEA	VAX-11/780	MIL-1750A Global Optimizer		JUN 84	12
	X	SEA	SEA	VAX-UNIX	VAX-11/780, MIL-1750A, Z-8001, PDP		OCT 84	12
	X	PSS	PSS	VAX-11/780	MIL-1750A, New, High Performance		OCT 83	1
	X	PSS	PSS	IBM, VAX-11/780	Global Optimizer (All Targets)		DEC 84	1
	X	GD/Litton/PSS	PSS	IBM, VAX-11/780	MIL-1750A Extended Memory Tool Set		NOV 83	1

* Unmaintained targets.

NOTES:
1. The delivery dates are estimates of when the product will be released by the developer to the sponsor. In most cases, availability dates of these products through the GSA Federal Software Exchange (FSE) have not yet been established.
2. EM = Extended Memory
3. TBD means 'To Be Determined'
4. This software is not available to the public for free.
5. These are Government sponsored and will be made available at no cost at a later date.
6. This software is not currently available.
7. Distribution charge.

JOVIAL (J73) AND 1750 POINTS OF CONTACT

1. PSS J73 Compilers and Support Tools:
Joel Fleiss
Proprietary Software Systems (PSS)
429 Santa Monica Blvd.
Suite 430
Santa Monica CA 90401
(213) 394-5233

2. National Technical Information Service:
5285 Port Royal Road
Springfield VA 22161
(703) 487-4650

3. MRASM DIS Compiler:
Dee Graumann
SofTech, Inc.
16885 West Bernardo Dr
San Diego CA 92127
(619) 451-3350

4. 1750/1750A Test Tools:
Ron Vokits
ASD-AFALC/AXT
WPAFB OH 45433-6503
(513) 255-5941
AV 785-5941

5. BMO/Peacekeeper J73 Compiler:
Capt Harold J. Kunnen
BMO/ENMG
Norton AFB CA 92409
(714) 382-6021
AV 876-6021

6. Air Force J73 Compilers and J73/1750/1750A Tools:
Carmen Marano
ASD-AFALC/AXTS
WPAFB OH 45433-6503
(513) 255-5945
AV 785-5945

7. Air Force Focal Point for Fairchild:
Microprocessor Chip Set
Jeffery L. Pesler
ASD/EN (CRFP)
WPAFB OH 45433-6503
(513) 255-3656
AV 785-3656
MILNET:
PESLERJ @ WPAFB-JALCF

8. Software in LCF Repository:
Georgeanne Chitwood
ASD/ADOL
WPAFB OH 45433-6503
(513) 255-4472
AV 785-4472

9. Army Pershing Compiler:
Douglas A. Wise
US Missile Command
DRSMI-OA
Redstone Arsenal AL 35898
(205) 876-3797
AV 746-3797

10. F-16 SPO Compilers:
Tim Sparling
ASD/YPEA
WPAFB OH 45433-6503
(513) 255-4776
AV 785-4776

11. F-111 Support Software:
Tom Broers
SM-ALC/MMECA
McClellan AFB CA 95652
(916) 643-6042
AV 633-6042

12. SEA Tools:
Gil Weisbord
Software Engineering Associates
23864 Hawthorne Blvd.
Suite 200
Torrance CA 90505
(213) 373-8901

13. ACT Tools:
Leslie B. Hersh
Advanced Computer Techniques Corp. (ACT)
16 East 32nd St.
New York NY 10016
(212) 696-3600

14. Integrated Support Software System:
Lt Mark Stephenson
AFWAL/AAAF
WPAFB OH 45433-6543
(513) 255-6548
AV 785-6548

15. Video Tapes and Workbooks:
Diane Katterheinrich
ASD/ADOL
WPAFB OH 45433-6503
(513) 255-4472
AV 785-4472

16. Boeing/SofTech Compiler:
John Walsh
SofTech, Inc.
460 Totten Pond Road
Waltham MA 02154
(617) 890-6900

17. CDC 1750A Tools:
R. L. Didrikson
Control Data Corp.
PO Box 609
Minneapolis MN 55440
(612) 853-5091

18. Intermetrics 1750A Tools:
John A. Dunford
Intermetrics, Inc.
5162 Springfield Pike
Dayton OH 45431
(513) 258-1271

19. CSDS Tool:
Federal Software Exchange
GSA 5203 Leesburg Pike
Suite 1100
Falls Church VA 22041
(703) 756-6150

20. BMO/Peacekeeper J73 Compiler:
Maj John Leighty /
Lt Ron Rozzo
BMO/ENBC
Norton AFB CA 92409
(714) 382-6651
AV 876-6651

21. J73 Test Tools:
Leo B. Collins
System Development Corp.
2500 Colorado Ave.
Santa Monica CA 90406
(213) 453-5126

22. J73 Programming Standards & Conventions:
1st Lt Joe Jarzombek
AFSCF/DVI
PO Box 3430
Sunnyvale AFS CA 94088
(408) 744-6439
AV 799-6439

23. JCVS, JOVIAL (J73) Validations:
Lt Col Joseph Dangerfield
ASD-AFALC/AXT
WPAFB OH 45433-6503
(513) 255-5941
AV 785-5941

24. Hunter & Ready VRTX/1750:
David Marquardt
Hunter & Ready, Inc.
445 Sherman Ave
PO Box 60803
Palo Alto CA 94306-0803
(415) 326-1950

Ada® COMPILERS AND TOOLS

I. Validated Ada Compilers

NAME	SPONSOR	HOST/TARGET MACHINE	HOST/TARGET OPERATING SYSTEM	VERSION	ACVC VERSION	EXPIRATION DATE	CONTACT
ALS	U.S. Army (CECOM), SofTech	VAX-11/780	VMS 3.5	1.25	1.4	11/84*	10
ALSYCOMP_0001	ALSYS	VAX-11/785, 780, 750, 730, MICRO VAX	VMS 4.0	1.0	1.5	12/08/85	5
Callan	TeleSoft, Callan	Callan UNISTAR 300	System V	2.0A2	1.3	05/21/85	2
Dansk	Dansk Datamatik Center	VAX-11/750	VMS 3.5	1.1	1.4	10/29/85	3
DEC VAX Ada	Digital Equip. Corp.	VAX-11/785, 780, 750, 730, MICRO VAX	VMS 4.0	T0.6-2	1.4	09/17/85	6
GMD/German MOD/Siemens	German Ministry of Defense, GMD	Siemens 7.571	BS2000, 7.1	840404	1.4	11/14/85	4
Intellimac/ TeleSoft Ada	TeleSoft, Intellimac	IN/7000M	ROS	2.0A1	1.3	05/84	8
Labtek (Wicat)	TeleSoft, Labtek	Motorola MC 68000	ROS	2.0A1	1.3	05/21/85	2
NYU Ada/ED	U.S. Army (CECOM), NYU	VAX-11/780	VMS 3.5	1.4	1.4	08/06/85	1
ROLM Ada Compiler	ROLM, Data General	DG LMV 4000, 6000, 8000, 10000	AOS/VS-A DA	5.234	1.4	05/21/85	2
Sun Microsystems	TeleSoft, Sun	Sun 120	UNIX	2.0A2	1.3	05/21/85	2
SYSTEAM/German MOD/VAX	German Ministry of Defense, SYSTEAM	VAX-11/750	VMS 3.0	1.0	1.4	11/12/85	4
TeleSoft VAX/UNIX	TeleSoft	VAX-11/780	UNIX 4.2	2.0A6	1.4	11/84*	2
TeleSoft VAX/VMS	TeleSoft	VAX-11/780	VMS 3.4	2.0A6	1.4	11/84*	2
TeleSoft/O-Bus	TeleSoft	Motorola MC 68000	ROS	2.0A1	1.3	05/21/85	2
VERDIX VADA 010-0101	VERDIX Corp.	VAX-11/750	UNIX 4.2	V03.04	1.4	12/84*	7

II. Ada Tools

TeleSoft Ada Development Kit Plus	TeleSoft	Motorola 68000, VAX-11/780, IBM 370	ROS, UNIX VMS, UNIX VMS, CMS				2

*Date shown is testing date. Otherwise, date shown is certificate expiration date.

Ada® POINTS OF CONTACT

1. Validated Ada Compilers:
   Audrey Hook
   Computer and Software Engineering Division
   Institute for Defense Analysis
   1801 N. Beauregard
   Alexandria VA 22311
   (703) 845-2516

2. TeleSoft Ada Compilers and Tools:
   Tom Dent
   TeleSoft, Inc.
   10639 Roselle Street
   San Diego CA 92121
   (619) 457-2700

3. Dansk Datamatik Compilers:
   Mr Jan Pedersen
   Dansk Datamatik Center
   Lundtoftevej 1C
   DK-2800
   Lyngby, Denmark
   (45) 2-872622

4. Siemens Compiler:
   Dr Guido Persch
   Gesellschaft für Mathematik und Datenverarbeitung (GMD)
   Haid- und Neustrasse 10-14
   D-7500 Karlsruhe 1
   Karlsruhe, West Germany
   (49) 721 690633

5. German Ministry of Defense VAX Compiler:
   Dr G. Winterstein
   SYSTEAM KG
   Am Entenfang 10
   D-7500 Karlsruhe 21
   Karlsruhe, West Germany

6. ALSYS Compilers:
   Mr J-C Heliard
   ALSYS
   29 Avenue de Versailles
   La Chataigneaie
   78170 La Celle Saint-Cloud, France
   (33) 3-9181244

7. VERDIX Compiler:
   Mr Donn Milton
   VERDIX Corporation
   7655 Old Springhouse Road
   McLean VA 22102
   (703) 448-1980

8. Intellimac Compiler:
   Mr David Dikel
   Intellimac
   6001 Montrose Road 6th Floor
   Rockville MD 20852
   (301) 984-8000

9. NYU Ada/ED Compiler:
   Mr E. Schonberg
   New York University
   251 Mercer Street
   New York NY 10012
   (212) 460-7393

10. ALS Compiler:
    Mr Billy Mitchell
    SofTech, Inc.
    460 Totten Pond Road
    Waltham MA 02154
    (617) 890-6900

MSS: MIL-STD-1750A SIMULATOR

SYNTAX:

```
$RUN  [ACCT]  MSS  <RET>
DO YOU WANT STATISTICS? Y <RET>          ; SET UP SIMULATOR
MAXIMUM ADDRESS FOR M? FFFF <RET>
HOW MANY PAGE FRAMES FOR M? 3 <RET>
MAXIMUM ADDRESS FOR ROM? 0 <RET>

*L LDM FILE.LDM <RET>                    ; LOAD .LDM FILE
LOADED FROM 00000 TO 0023A
*E M 100 TO 23A   M  <RET>               ; EXAMINE MEMORY IN MNEMONICS
        :          :
        :          :                     ; OTHER SIMULATOR COMMANDS
        :          :
<CTRL Y>                                 ; EXIT THE SIMULATOR
$ <ESC> E                                ; EXIT VAXLINK
```

FEATURES:

- THE MODEL INCLUDES REGISTERS, MEMORIES, STATISTICAL TABLES, REAL TIME CLOCK
- THE INTERPRETER ALLOWS SINGLE STEP EXECUTION OF MIL-STD-1750A MACHINE CODE
- THE EXECUTIVE CONTROLS THE MODEL AND THE INTERPRETER
- UP TO 10 TRAPS CAN BE INSERTED
- UP TO 10 SNAPS CAN BE USED TO DISPLAY SELECTED VALUES IN THE MODEL
- UP TO 10 MONITORS CAN GUARD A PARTICULAR WORD (EITHER REGISTER OR MEMORY ENTRY)
- EXTENSIVE ON-LINE HELP IS AVAILABLE

BK:5/84

SIMULATOR COMMANDS

E	EXAMINE	IS	INSERT SNAP
L	LOAD	DS	DELETE SNAP
S	STEP	SNP	SNAP TABLE
;	STEP & SNAP	XIT	EXIT
I	INITIALIZE	QS	OPEN SYMBOL FILE
IT	INSERT TRAP		EXAMINE (LIKE E)
DT	DELETE TRAP	V	VERIFY
TRP	TRAP TABLE	RS	RUN/SLEEP
PA	PAUSE	OD	END DO
HELP	GIVE HELP	ST	STATUS
=	ASSIGN	IM	INSERT MONITOR
W	WRITE	DM	DELETE MONITOR
R	RUN	MON	MONITOR TABLE
DO	READ FILE	*	DO NOTHING
G	GRAPH		

BK:5/84

SAMPLE SIMULATOR SESSION

```
*L LDM HANOI50.LDM                    ; LOAD THE PROGRAM
   LOADED FROM 01000 TO 01041
*IM M OFE                             ; INSERT MONITOR
*R 1000
IC: 1041    CL: ................
000FE: 0012                           ; LINE FEED
IC: 1041    CL: ................
000FE: 0015                           ; RETURN
  :     :
      0101                            ; A
      0061                            ; 1
      0103                            ; C
      0012                            ; LINE FEED
      0015                            ; RETURN
      0101                            ; A
      0062                            ; 2
      0102                            ; B
       :
       :
*XIT
EXECUTED     ..... INSTRUCTIONS
SIMULATED    ..... SECONDS
CONSUMED     ..... SECONDS
RAN          ..... SECONDS

FOR A RATIO OF     .........
```

BK:5/84

REAL TIME EXECUTIVES

VRTX

by Hunter & Ready

ESCAPE

by PSS (Proprietary Software Systems)

BK:5/84

Approaches To
Real Time Programming

CYCLIC EXECUTIVE:

- HARD TO MAINTAIN
- HARD TO INTERFACE TO HIGH LEVEL LANGUAGE (HLL)
- DOESN'T MATCH ADA MODEL

MULTITASKING EXECUTIVE:

- EASIER TO MAINTAIN
- EASY TO INTERFACE TO HLL
- CLOSER TO ADA MODEL

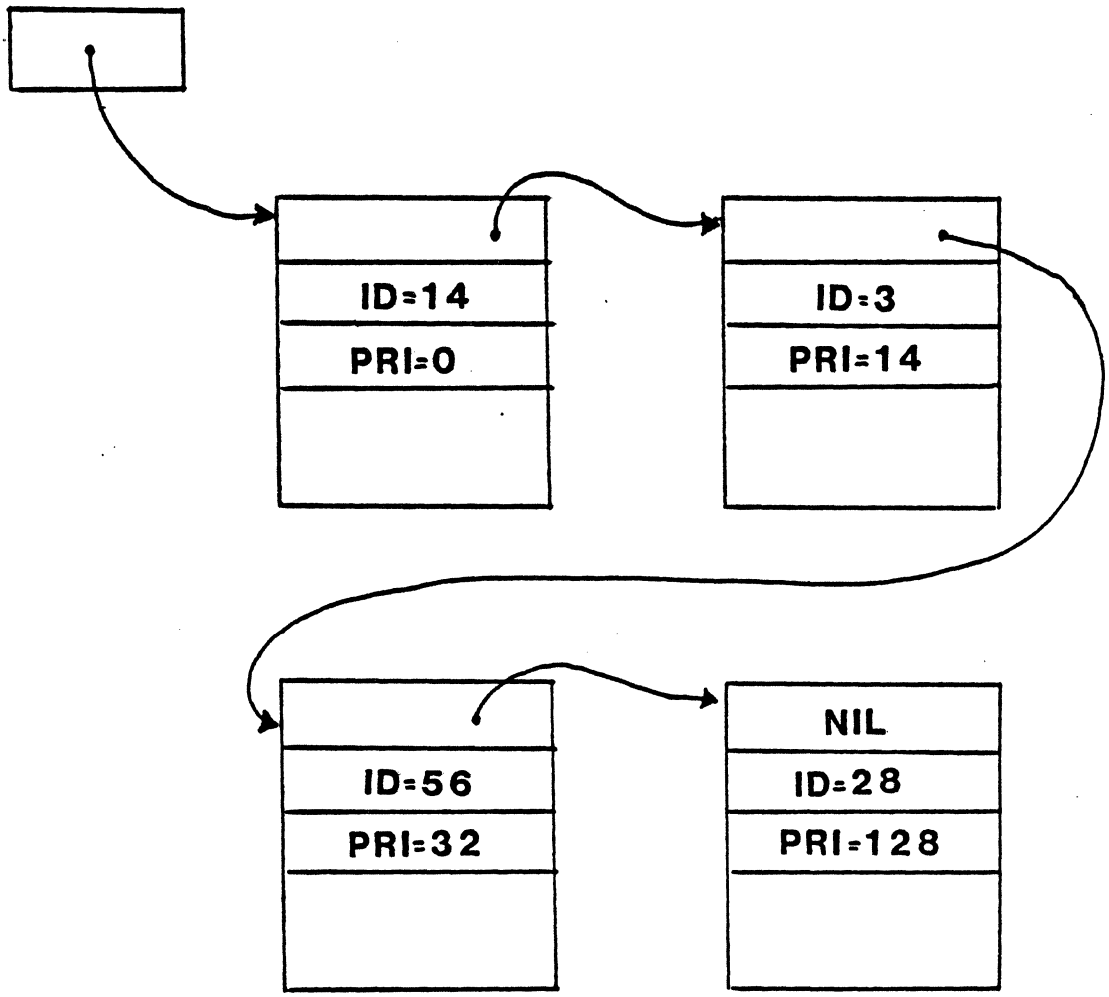
BK:5/84

Operating Systems

	REAL TIME	TIME SHARED
EXAMPLES	RTOS EMREX VRTX ESCAPE	VMS UNIX IMDOS RDOS
SPEED	250 US	2 SEC
PREDICTABILITY	DETERMINISTIC	TRY AGAIN LATER

BK:5/84

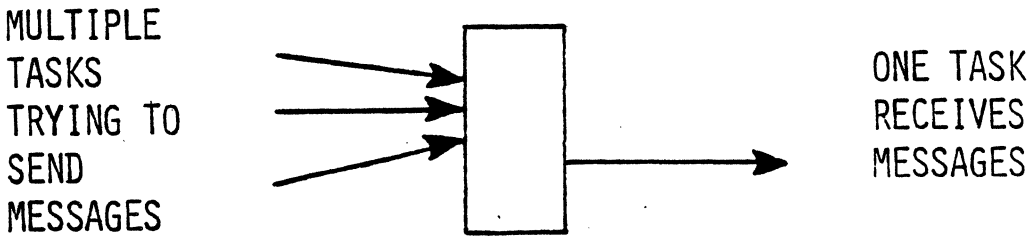
Multitasking Executives Use Priority Based Scheduling



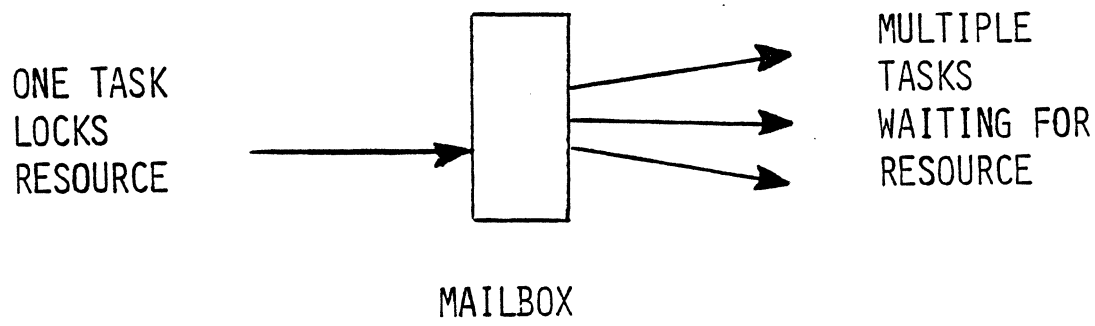
BK:5/84

COMMUNICATIONS & SYNCHRONIZATION

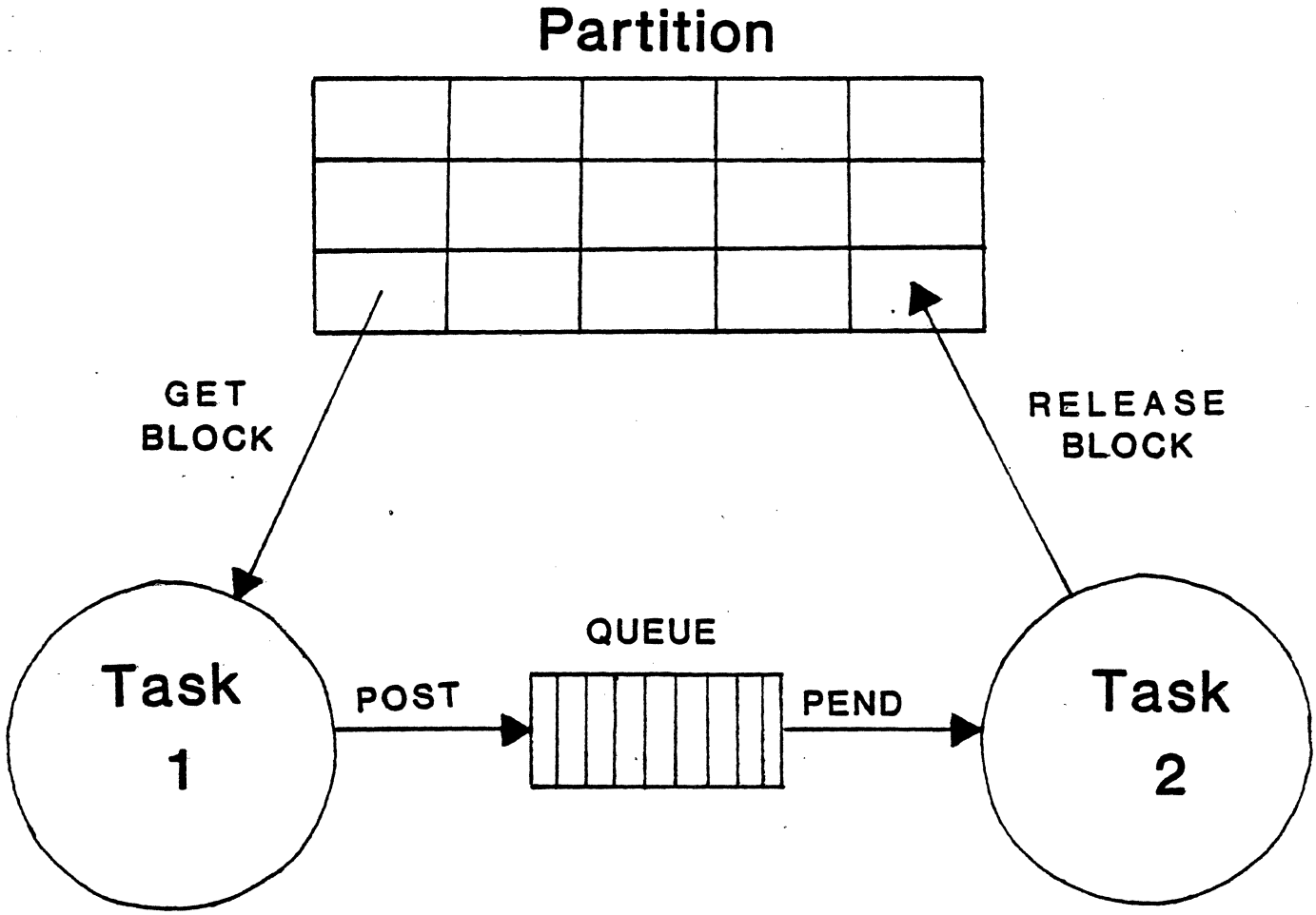
MESSAGE PASSING



RESOURCE LOCKS

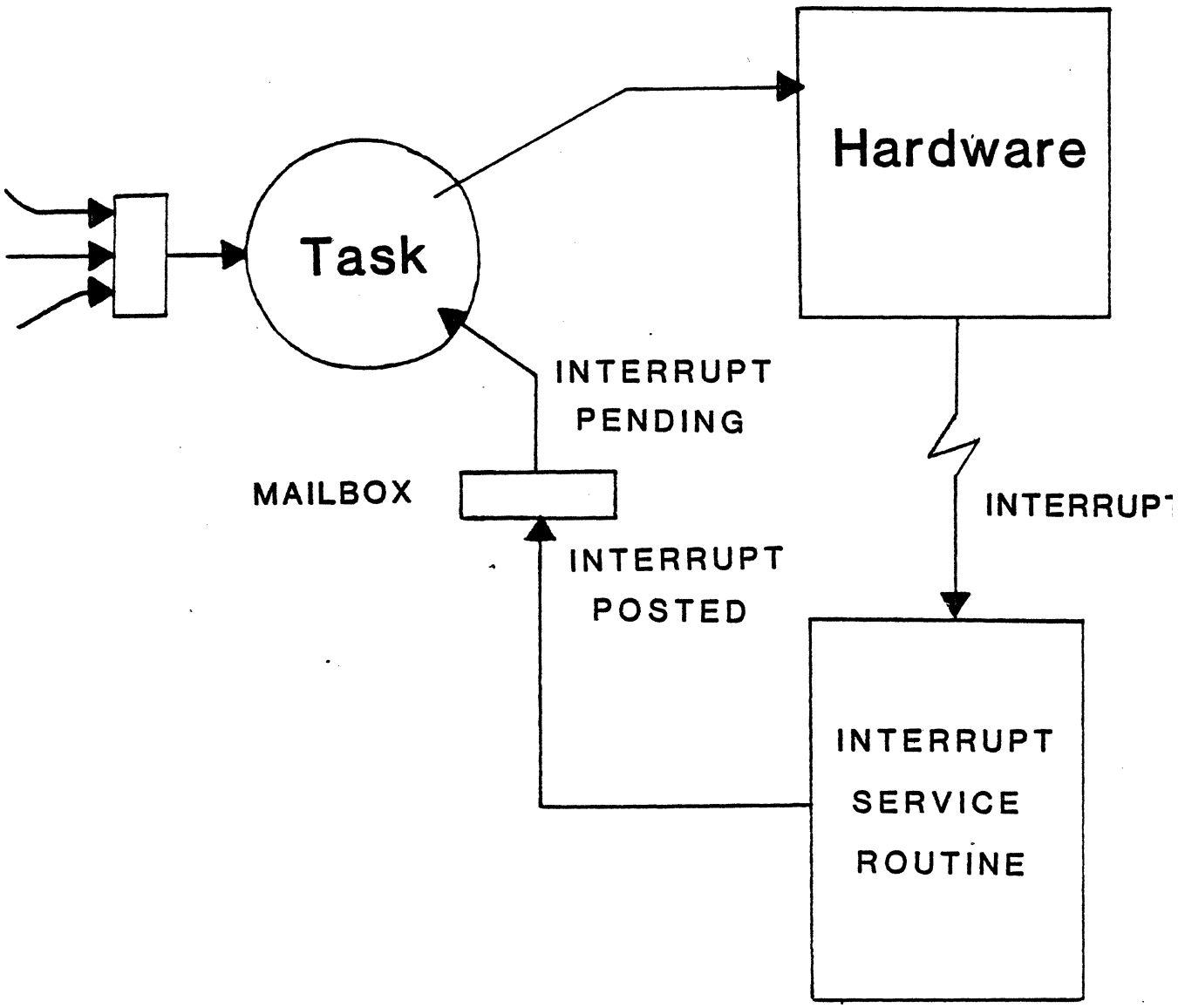


COMMUNICATIONS VIA A QUEUE



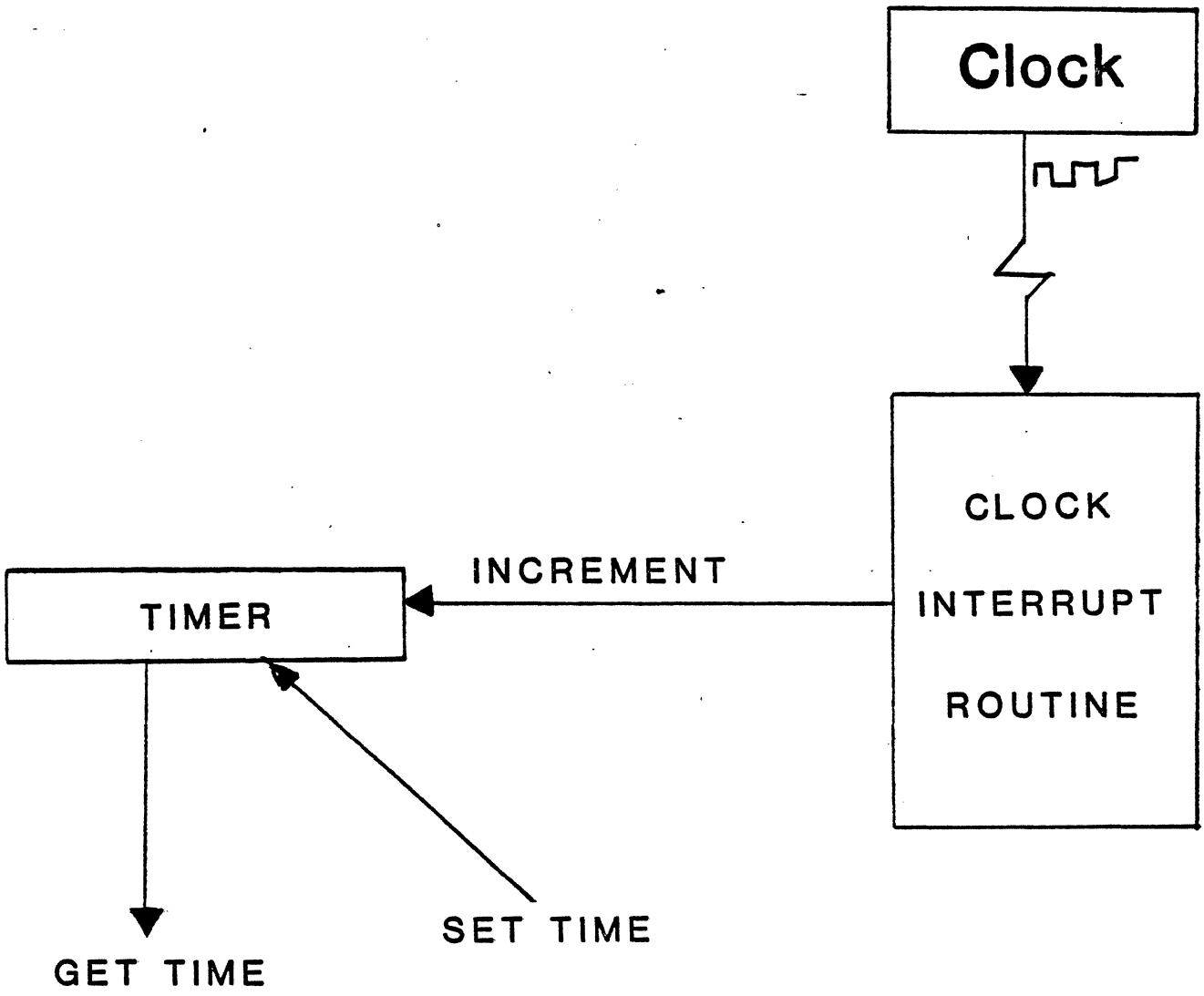
BK:5/84

INTERRUPT HANDLERS



BK:5/84

THE SYSTEM CLOCK



BK:5/84

VRTX SYSTEM CALLS

TASK MANAGEMENT

TCREATE

TDELETE

TSUSPEND

TRESUME

TPRIORITY

TINQUIRY

LOCK

UNLOCK

INTERTASK COMMUNICATION and SYNCHRONIZATION

POST

PEND

ACCEPT

QPOST

QPEND

QACCEPT

QCREATE

QINQUIRY

BK:5/84

VRTX SYSTEM CALLS (cont.)

MEMORY ALLOCATION

PCREATE

PEXTEND

GBLOCK

RBLOCK

SYSTEM CLOCK and INTERRUPTS

STIME

GTIME

TDELAY

TSLICE

UI-TIMER

UI-ENTER

UI-EXIT

ESCAPE

Priority-based Process Management, up to 255 Processes

System Configurator produces re-entrant, PROMable code

Memory Management and Block Protection

Communications through Mailboxes or Queues

Boolean Flags Provide Process Synchronization

Interrupt Vectoring Can Be Changed Dynamically

Clock Services

Support Framework for I/O Device Handlers

Exception Handling

Checkpoints with Restart Capability when using RAID

THIRD PARTY

HARDWARE SUPPORT

Features of F9450 Emulators

CHIP REPLACEMENT

REAL TIME OPERATION AT 20MHZ

EMULATION IN A VARIETY OF MODES

- FULL SOFTWARE SIMULATION, NO HARDWARE REQUIRED
- USE HARDWARE CLOCK AND I/O WITH EMULATOR MEMORY
- FULLY WORKING TARGET, EXTENSIVE TRACING

TRACE CAPABILITY

- INSTRUCTIONS
- REGISTERS
- SYMBOLIC CAPABILITY
- EVENT-TRIGGERED ANALYSIS

INTERFACE TO HOST COMPUTER

- SUPPLIED BY EMULATOR MANUFACTURER
- VAX INTERFACE

SUPPORT SOFTWARE

- MIL-STD 1750A ASSEMBLER & LINKER
- JOVIAL COMPILER
- ADA COMPILER IN THE FUTURE

Tektronix Hardware Support – Emulator

V1750A SYSTEM

- 8540 + TWO INTERFACE CARDS
- POD + PROBE
- 4105 COLOR TERMINAL

CHIP REPLACEMENT

- HYBRID PROBE FOR F9450
- POD WITH F9450 + DEBUG CODE
- FULL 20 MHz CLOCK

BUS EMULATION

- GRABBER CLIPS, DIN CONNECTOR, ETC.
- POD WITH DEBUG CODE
- FULL 5 MHz BUS EMULATION

INTERFACES TO:

- 8560
- VAX
- OTHER HOST VIA RS232

Tektronix Hardware Support - Logic Analyzers

DAS 9100

- UP TO 104 CHANNELS
- COLOR DISPLAY
- TAPE DRIVE
- 1750A MNEMONIC DISASSEMBLY
- F9450 PROBE

TEK 1240

- PORTABLE
- 1750A DISASSEMBLY USING PROM PACK
- F9450 PROBE

BK:5/84

HP64000 LOGIC DEVELOPMENT SYSTEM

10 CARD SLOTS AVAILABLE TO ADD SUBSYSTEMS

9450 USER DEFINABLE SUBSYSTEM (3 cards)

- 9450 SPECIFIC CONTROLLER CARD
- MEMORY CONTROLLER CARD
- MEMORY CARD WITH 128KB

INTERNAL ANALYZER (HP64302A, 1 card)

- PROVIDES SIMPLE TRACING CAPABILITY IN CONJUNCTION WITH EMULATOR SUBSYSTEM

LOGIC STATE/SOFTWARE ANALYZER (HP64620S, 2 or more cards)

- PROVIDES NONINTRUSIVE, COMPLEX TRACING CAPABILITIES
- CONTROLLER CARD
- STATE CHANNELS, 20 OR 40 PER CARD UP TO 120 MAXIMUM

SOFTWARE PERFORMANCE ANALYZER (HP64310A, 1 card)

- PROVIDES NONINTRUSIVE STATISICAL & TIMING MEASUREMENTS

LOGIC TIMING/HARDWARE (HP64600F, 2 or 3 cards)

- PROVIDES "GENERIC" HARDWARE ANALYSIS, CAN TRIGGER OTHER SUBSYSTEMS
- CONTROLLER CARD
- EIGHT CHANNEL CARD, 16 CHANNELS MAXIMUM (2 CARDS)

BK:5/84

HP 9450 USER-DEFINABLE EMULATOR

SPECIFICATIONS

- CHIP REPLACEMENT WITH F9450 IN POD
- IN-CIRCUIT OPERATION TO 20 MHz
- SHARED USE OF POWER-DOWN INTERRUPT FOR PROCESSOR BREAKING
- VECTOR JAMMING USED DURING BREAK
- APPROXIMATELY 55 ICs USED IN DESIGN
- AVAILABLE IN APPLICATION NOTE FORM

FEATURES

- RUN CONTROLS
- SINGLE STEPPING OF PROGRAM
- DISPLAY MEMORY, REGISTERS, AND I/O
- MODIFY MEMORY, REGISTERS, AND I/O
- EMULATION MEMORY SUPPORT UP TO 64K WORDS IN 256 BYTE BLOCKS
- COMPATIBLE WITH SOFTWARE PERFORMANCE ANALYZER
- SYMBOLIC DEBUG AND ANALYSIS WITH INTERNAL ANALYZER
- SIMULATED I/O FOR HP64000 SYSTEM RESOURCE USE
- MONITOR SUPPLIED WITH APPLICATION NOTE
- ADDITIONAL WAIT STATES MAY BE ADDED TO EMULATION MEMORY

BK:5/84

Fairchild Microcontroller Division, 450 National Avenue, Mountain View, California 94043 (415) 962-3888