88951000

CONTROL DATA CORPORATION

CDC® HARDWARE FLOATING-POINT UNIT

BT221-A

HARDWARE MAINTENANCE MANUAL

REVISION RECORD

REVISION	DESCRIPTION
01	Preliminary released.
(11/77)	
02	Manual revised to conform to corporate format standards. Released by ECO DS18852.
(3/78)	

Publication No.
88951000

REVISION LETTERS I, O, Q AND X ARE NOT USED

Printed in the United States of America

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, California 92037
or use Comment Sheet in the back of this manual.

MANUAL TO EQUIPMENT LEVEL CORRELATION SHEET

This manual reflects the equipment configurations listed below.

EXPLANATION: Locate the equipment type and series number, as shown on the equipment FCO log, in the list below. Immediately to the right of the series number is an FCO number. If that number and all of the numbers underneath it match all of the numbers on the equipment FCO log, then this manual accurately reflects the equipment.

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
BT221-A	01		

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	--								
Title Page	--								
ii	02								
iii/iv	02								
v/vi	02								
vii/viii	02								
1-1 thru 1-3	02								
2-1 thru 2-33	02								
3-1 thru 3-5	02								
4-1 thru 4-54	02								
5-1 thru 5-55	02								
6-1 thru 6-32	02								
A-1	02								
B-1 thru B-11	02								
C-1 thru C-5	02								
D-1 thru D-16	02								
E-1 thru E-30	02								
Comment Sheet	02								
Cover	--								

PREFACE

This manual describes the functional mechanical and operational characteristics of the CDC® BT221-A Hardware Floating-Point Unit (HFPU) used with the CYBER 18-17 (SYSTEM 17) Computer System.

It is assumed that the reader is familiar with CYBER 18-17 hardware and software.

For additional information, the following manuals may be obtained from Literature Distribution Services:

Title	Publication No.
1781-1 Hardware Floating-Point Unit Reference Manual	88951100
1784 Computer Reference Manual	89633400
1784 Computer Input/Output Specifications	89673100
CYBER 18-17 Installation Manual	88996000

CONTENTS

1	GENERAL DESCRIPTION	1-1
1.1	Physical Description	1-2
1.1.1	Components	1-2
1.1.2	Slot Assignments	1-2
1.2	Functional Description	1-3
2	OPERATION AND PROGRAMMING	2-1
2.1	Equipment Definition	2-1
2.2	Characteristics	2-13
2.2.1	Command Description	2-19
2.2.1.1	Operand Addressing	2-21
2.2.1.2	Operand/FPAC Format	2-25
2.2.1.3	Rounding	2-26
2.2.1.4	Fix Float Number Conversions	2-27
2.2.1.5	HFPU Initialition Sequences	2-27
2.2.1.6	HFPU Stop/Restart Sequence	2-28
2.2.1.7	Function/Status Register Definitions	2-29
2.2.1.8	Hardware Execution Times	2-30
3	INSTALLATION	3-1
3.1	Logic Card Installation	3-1
3.1.1	Inspection	3-1
3.1.2	Installation of Jumpers	3-1
3.1.2.1	DSA Board	33-2
3.1.2.2	A/Q Board	3-3
3.1.2.3	SPALU Board	3-4
3.1.3	Board Installation	3-4
3.2	Mother-Board Installation and Removal	3-5
3.2.1	Preparation	3-5
3.2.1.1	The Backplane	3-5
3.2.1.2	The Mother Boards	3-5
3.2.2	Installation	3-5
3.2.3	Removal	3-5
4	THEORY OF OPERATION	4-1
4.1	Hardware Organization	4-1
4.1.1	Device Structure	4-1
4.1.2	The Micro-Processor Concept	4-1
4.1.3	The Programmable Elements	4-9
4.1.4	The Micro-Instruction Set	4-14
4.2	Description of Algorithms	4-34
4.2.1	Introduction to Flowcharts and Listings	4-34
4.2.2	The Algorithms	4-34

4.2.2.1	OP-Code Fetch/Cold Start	4-34
4.2.2.2	The SPEC Group	4-34
4.2.2.3	Single Micro-Instruction Group	4-37
4.2.2.4	Floating Point Group	4-38
4.2.2.5	Index Register Group	4-50
4.2.2.6	The A/Q STOP Command	4-51
4.2.2.7	Restart	4-53

APPENDIXES

A	Glossary	A-1
B	Micro-Code Listings and Flow Charts	B-1
C	ROM Truth Tables and A/Q Decoding ROMS	C-1
D	Floating-Point A Code and Flow Charts	D-1
E	Master Control Micro-code and Flow Charts	E-1

FIGURES

2.1	HFPU Q-Register Function Format	2-3
2.2	FSR Bit Assignment	2-4
2.3	Addressing Examples	2-22
2.4	Execution Time Examples	2-33
4.1	HFPU Data Paths	4-2
4.2	Floating Point Micro-Processor Block Diagram	4-6
4.3	Master Micro-Processor Block Diagram	4-7
4.4	Arithmetic Shifting	4-12
4.5	Master Micro-Processor Instruction Format	4-15
4.6	Floating-Point Micro-Processor Instruction Format	4-16
4.7	Flow Chart Conventions	4-35

TABLES

1.1	HFPU Board Summary	1-2
2.1	Function/Status Register Bit Assignment	2-5
2.2	Command-Code Definition	2-14
2.3	Execution Times (worst case operands)(440ns Tac) (600ns cycle)	2-31
3.1	DSA Scanner Position Select Jumpers	3-2
3.2	A/Q Equipment Address, Protect Mode, and Single-Precision Device Jumpers	3-3
3.3	Hexadecimal Code for Equipment Select	3-4
4.1	Master Micro-Processor Instruction Format	4-17
4.2	Floating Point Micro-Processor Instruction Format	4-25

This manual describes the functional, mechanical, and operational characteristics of the System 17 Hardware Floating Point Unit, herein after referred as the HFPU. This device is designed to provide improvement in the execution time of programs running under MSOS FORTRAN IV. It provides a fully compatible replacement for the software Floating-Point Interpreter packages, FLOT (single-precision) and DFLOT (double-precision). The HFPU interprets and executes the same calling sequences as those used by the software. Thus the software package can be replaced by a small driver for the HFPU with no change in user written programs.

All Floating Point arithmetic in MSOS FORTRAN is done through an interpretive package of subroutines, This package, FLOT or DFLOT, was designed to minimize the amount of memory required for user written programs. In order to do this, a calling sequence structure was established. The calling sequence consists of a command word which may contain up to 4 function commands followed by address words which point to the locations in memory of the operands required for the function. This technique, since it is basically an expansion of the instruction set of the System 17, lends itself very nicely to the construction of a special purpose processor which executes the floating-point calling sequence.

The HFPU is such a device. It consists of a fast, floating-point arithmetic processor coupled to an efficient command interpreter that is interfaced to the A/Q and DSA channels of the System 17 CPU. The HFPU responds to a group of A/Q commands which are used for initialization and diagnostic purposes. Once initialized, the HFPU utlizes the DSA channel to fetch the calling sequence from memory and to retrieve and store operands as required. Upon completion of the execution of the calling sequence, the HFPU returns a pointer to the System 17 CPU via the A/Q channel which indicates the next location in memory following the calling sequence. This is done so that System 17 program can continue execution at the next executable instruction following the calling sequence.

The FLOT calling sequence command set has been expanded for the HFPU to include program-control type commands (Jump and conditional Branch). This opens up the possibility of system software optimization by having the HFPU run in parallel with the System 17 CPU.

As with the reentrant and non-reentrant versions of FLOT and DFLOT, the HFPU has been provided with a reetrancy capability in the form of STOP and RESTART commands. By using these commands, the HFPU can be interrupted and then reintialized without any loss of information.

1.1 Physical Description

1.1.1 Components. The unit consists of seven logic cards and three backplane interconnect assemblies. Each interconnect assembly consists of two printed circuit cards (mother-boards) which are coupled via a short cable. Table 1.1 summarizes these cards by name and PWA part number. All power is described from the +5 V supply of the expansion chassis.

1.1.2 Slot Assignments. The logic cards may be installed in two different positions in the expansion chassis. The cards must be in slots in the order detailed in Table 1.1 . The major constraint is that the DSA card must be installed in one of the Prewired DSA slots (slot 22 or 14).

The Mother Boards are pushed onto the backplane on the side opposite from the slots occupied by the logic cards.

TABLE 1.1. HFPU BOARD SUMMARY

Name	PWA No.	Standard Slot No.	Alternate Slot No.	Function
ADDR	88953800	23	15	Address Preparation
DSA	88953700	22	14	DSA Interface & Master Control
A/Q	88953400	21	13	A/Q Interface & Master Control
DPALU(SP)	88953100	18	10	Master Control
DPALU(DP)	88954100	18	10	Double Precision Extension & Master Control
SPALU	88952800	17	9	Single Precision Mantissa Arithmetic
FPHMP	88952500	16	8	Floating Point Micro-Processor
EXP&TIM	88952200	15	7	Exponent & Floating-Point Timing

NAME	Mother Board PWA No.	Location
P1	88954400	P1 Mother Board
P2 TOP	88954500	P2 pins 1 to 15 Mother Board
P2 BTM	88954600	P2 pins 16 to 31 Mother Board

1.2 FUNCTIONAL DESCRIPTION

Functionally, the HFPU is provided with a look-ahead feature which allows it to fetch the operand required for a succeeding operation while a preceding floating-point operation is in progress. Thus, although the worse case double-precision FDIV time is approximately 16 micro-seconds, the effective time may be 13 micro-seconds or even lower depending on number of overlapped operations. This feature implies, for instance, that a typical FORTRAN program utilizing single-subscripted variables with execute floating-point operations in nearly the same time as a program utilizing unsubscripted variables.

2.1 Equipment Definition.

The System 17 HFPU is an addressable I/O type of equipment connected to the A/Q and DSA I/O channels of the CPU. It is activated and monitored via the A/Q I/O channel and performs floating-point calculations with data parameters obtained via the DSA I/O channel.

The HFPU uses an operating format that is identical to the FLOT subroutine format and executes all of the FLOT call-operations plus the additional call-operations which are defined in paragraph 2.2.

Two modes of floating-point arithmetic capability are available to the HFPU user. These modes are:

a) Single-Precision Arithmetic (32-bit operand)
b) Double-Precision Arithmetic (48-bit operand)

In addition to the two floating-point operation modes, the HFPU has four types of operand-addressing modes. These modes are:

a) Absolute (16-bit)
b) Relative (16 bits with bit 15 - sign)
c) Indexed (16-bit)
d) Indexed un-multiplied (16-bit)

These operand addressing modes allow the user to access all permissible memory locations within a 65K-word memory.

After the HFPU is activated by the appropriate A/Q channel command, it obtains all Command-Code instructions and data operands directly from the System 17 memory via DSA access. It executes these Command-Code instructions and returns the results of the operations to memory as directed. When the HFPU is in Block or Hog Mode, it utilizes the "priority" signal line to enhance the DSA speed for its access to memory.

The HFPU also incorporates an A/Q and DSA protect feature. The A/Q portion of the protect feature consists of a jumper plug on the A/Q Interface board. Presence of a jumper plug is defined as "Protected Mode." Absence of a jumper plug is defined as "Unprotected Mode."

When the HFPU is set to "Protect Mode", it will set FSR bit 4, accept only protected A/Q Write commands and will cause an "External Reject" to the CPU for any unprotected A/Q Write Commands it receives. When the HFPU is set to "Unprotected Mode", it will accept all legal A/Q I/O commands. Unprotected STOP Commands and unprotected RE-START Commands are defined as illegal.

The DSA protect mode feature is activated by setting bit-4 in the HFPU Function Status Register (FSR). This bit is set by four methods which are:

(1) A protected A/Q Write Command to Q-Station 0 (A to FSR) with A-Register Bit-4 set.
(2) A protected A/Q Write Command to Q-Station 3 or 4 (Cold Start SP or DP).

(3) A protected A/Q Write Command to Q-Station A (STOP).
(4) Presence of the A/Q Protect jumper.
NOTE: The above three A/Q Write Commands must be protected to set FSR bit 4 regardless of the A/Q Protect jumper position.

When the DSA Protect Mode is active, it will allow the HFPU to Write data words or store Register contents into protected memory locations without incurring program protect errors. FSR bit 4 stored in memory during the STOP Command will reflect the DSA protect state of the HFPU prior to execution of the STOP Command.

When the DSA Protect Mode is active (FSR bit 4 set) all unprotected A/Q Write Commands will be rejected.

The HFPU contains six functional registers that are accessible through the A/Q I/O channel. These registers are addressed by using the Q-register bits as defined in figure 2.1. The six registers and their use are defined as follows:

a) FSR = Function/Status Register

This is the main control register for the HFPU and will accept A/Q I/O commands at any time. If active, the HFPU accepts an A/Q Write Command to the FSR only if A-bit 00 (PCLR) is set. Any other A-bits will be ignored. The functions of the FSR bits are summarized in figure 2.2.

b) CCR = Command-Code Register

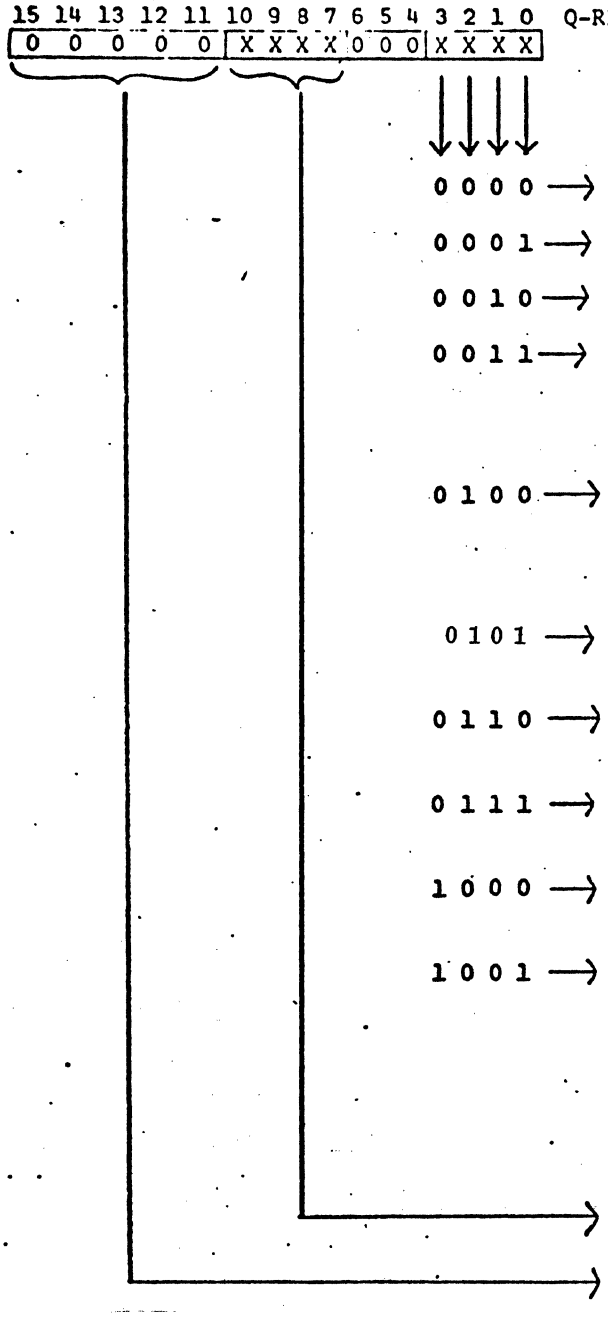
This register is normally loaded via the DSA channel and contains the command code instruction word. It can be read on the A/Q channel (see 2.2.1.6 for format) at any time but can only be loaded by an A/Q channel write when the unit is not active.

c) IR = Index Register

This register contains a 16-bit digital number that is used during operand address formation for floating-point calculations. It is normally loaded via the DSA channel by an INDX command. It can be read at any time on an A/Q Read Command but can only be loaded by an A/Q Write command when the unit is not active. The value loaded or written via the A/Q Read and Write commands is always the raw, un-multiplied 16-bit number.

d) PCR = Program Counter Register

This register contains a 15-bit digital number used as the base address during operand address formation. It is normally loaded via the A/Q channel by a Cold Start Command and incremented during floating-point operations. It is also loaded via the A/Q channel by an A-Reg to PCR Command or via the DSA channel on a Restart Command.



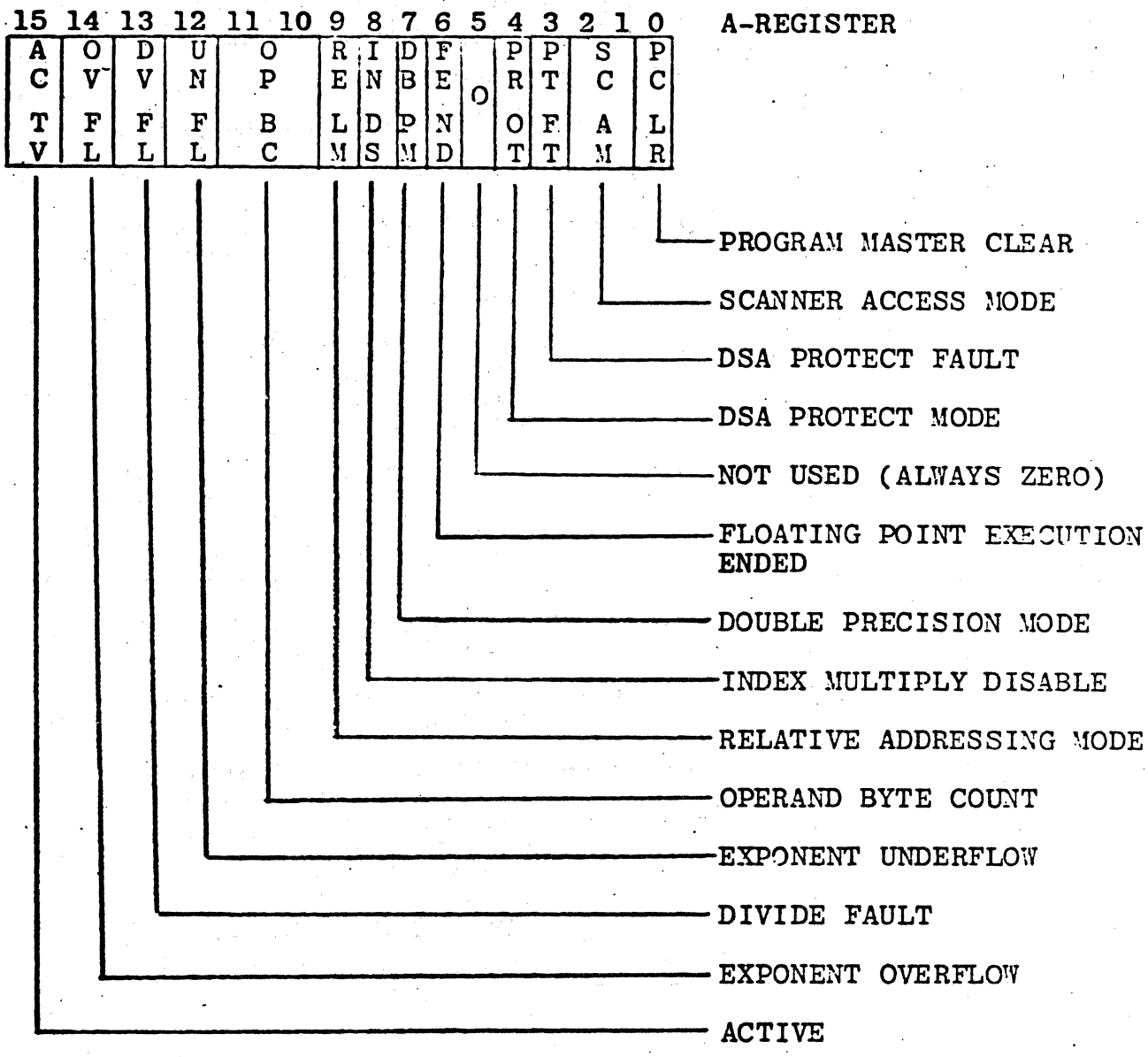
	Transfer Function On A/Q WRITE	Transfer Function On A/Q READ
0 0 0 0 →	(A-REG) → FSR+	(FSR) → A-REG
0 0 0 1 →	(A-REG) → CCR*	(CCR) → A-REG
0 0 1 0 →	(A-REG) → IR *	(IR) → A-REG
0 0 1 1 →	(A-REG) → PCR * (Cold Start Address) (Single Precision)	(PCR) → A-REG * (Address Status) (If not Active)
0 1 0 0 →	(A-REG) → PCR * (Cold Start Address) (Double Precision)	(PCR) → A-REG * (Address Status) (If not Active)
0 1 0 1 →	(A-REG) → SSAR * (Restart Address)	(PCR) → A-REG (Address Status)
0 1 1 0 →	(A-REG) → FPAC * (FPAC BITS 00 - 15)	(FPAC) → A-REG * (FPAC BITS 00 - 15)
0 1 1 1 →	(A-REG) → FPAC * (FPAC BITS 16 - 31)	(FPAC) → A-REG * (FPAC BITS 16 - 31)
1 0 0 0 →	(A-REG) → FPAC * (FPAC BITS 32 - 47)	(FPAC) → A-REG * (FPAC BITS 32 - 47)
1 0 0 1 →	(A-REG) → SSAR (Stop Order, HFPU will stop and use contents of A-register as first address for saving registers)	(SSAR) → A-REG (SSAR Status)

*The HFPU returns an "EXTERNAL REJECT" to the CPU if an attempt is made to address these registers while the HFPU is in an active state (in process of calculation or bit 15 FSR set).

+The HFPU returns an "EXTERNAL REJECT" while it is in an active state if A-register Bit Ø is not set. If A-Reg Bit Ø is set the HFPU returns a "REPLY".

The HFPU will return an External Reject to the CPU on any other A/Q Read or Write Command if the HFPU cannot respond within 4 microseconds. This condition can occur if the Read or Write Command is issued at the time the HFPU is raising its DSA Need signal for a series of address/operand retrievals in Priority mode and the DSA is already active (HFPU must for scanner).

Figure 2.1 HFPU Q-Register Function Format



NOTE 1: Refer to table 2.1 for detailed explanation of bit assignments.

NOTE 2: Console Master Clear referred to in table 2.1 clears all HFPU timing, resets the HFPU to an idle state, and clears all registers with the exception of the PCR and the FPAC. Console Master Clear enters the HFPU via a pin on the A/Q Channel bus.

Figure 2.2. FSR Bit Assignment

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
15	ACTV	Bit is set by A/Q Channel Write Command to FSR with A-bit 15 set (HFPU must be inactive) or by HFPU when it is in an active state. When this bit is set, it will cause the HFPU to reject all A/Q channel Write Commands except A Reg to FSR and Protected Stop (A Reg to SSAR). Bit is cleared or reset by: a) Inactive HFPU status. b) Program Master Clear. c) Console Master Clear. Inactive status does not necessarily indicate that the HFPU has completed the FLOT subroutine as the STOP Command will clear FSR bit 15 after storing all appropriate Registers. FSR bit 15 stored at SSAR during the STOP Command will reflect the condition of the HFPU prior to the STOP Command. WARNING: Setting this bit via an A/Q Write Command to FSR will place the HFPU in a state such that it will return an External Reject to all A/Q commands except a Program Master Clear (A/Q Write to FSR with A-bit 00 set).
14	OVFL	EXPONENT OVERFLOW. Bit is set by: a) HFPU arithmetic operation in which the exponent of result was too large to be represented by the eight binary bits. When this bit is set as a result of an arithmetic operation, the HFPU will force-set the FPAC to the largest floating-point number expressible with the correct F.P. sign. b) A to FSR Command (HFPU inactive) from CPU and A-bit 14 = 1 . This action sets only this bit and does not affect the contents of the FPAC. Bit is reset by: a) A to FSR Command (HFPU inactive) from CPU and A-bit 14 = 0 . b) Program Master Clear. c) Console Master Clear.

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT (Contd)

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
13	DVFL	DIVIDE FAULT. Bit is set by: a) HFPU when an attempt is made to divide by a zero or by an un-normalized operand. When bit is set as a result of an arithmetic operation, the HFPU will force-set the FPAC to the largest floating-point number expressible with the sign of the Dividend. b) A to FSR Command (HFPU inactive) from CPU and A-bit 13 = 1 . This action sets only this bit and does not affect the contents of the FPAC. Bit is reset by: a) A to FSR (HFPU inactive) Command from CPU and A-bit 13 = 0 . b) Program Master Clear. c) Console Master Clear.
12	UNFL	EXPONENT UNDERFLOW. Bit is set by: a) HFPU arithmetic operation in which the exponent of the result was too small to be represented by the eight binary bits. When this bit is set as a result of an arithmetic operation, the HFPU will force-set the FPAC to zero. b) A to FSR Command (HFPU inactive) from CPU and A-bit 12 = 1 . This action sets only the bit and does not affect the contents of the FPAC. Bit is reset by: a) A to FSR (HFPU inactive) Command from CPU and A-bit 12 = 0 . b) Program Master Clear. c) Console Master Clear.

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT (Contd)

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
11 and 10	OPBC	OPERAND BYTE COUNT. Indicates which of of the four bytes in the CCR is about to be executed. It has the following bit format: Bit 11 / Bit 10 0 / 0 Operand byte one. 0 / 1 Operand byte two. 1 / 0 Operand byte three. 1 / 1 Operand byte four. These bits can be set to any initial state by an A to FSR (HFPU inactive) Command from the CPU and A-bits 11 and 10. Bits are reset by: a) A to FSR (HFPU inactive) Command from CPU and A-bits 11 and 10 set to zero. b) Cold Start Command. c) Program Master Clear. d) Console Master Clear. NOTE: A/Q Write Command to Q-Station 1 (A Reg to CCR) does not affect the state of FSR bits 11 and 10.
9	RELM	RELATIVE ADDRESSING MODE. Bit is set or reset by: a) The HFPU execution of a CHMD instruction. Refer to paragraph 2.2.1 for detailed explanation. b) A to FSR (HFPU inactive) Command from CPU and the state of A-bit 09 Bit Cleared By: a) Cold Start Command. b) Program Master Clear. c) Console Master Clear.

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT (Contd)

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
8	INDS	INDEX MULTIPLY DISABLE. This bit is used to inhibit the logic that multiplies the Index Register Contents by 2 or 3 during effective address formation. Bit is set by: a) A to FSR (HFPU inactive) Command from CPU and A-bit 08 set to 1 . Bit is reset by: a) A to FSR (HFPU inactive) Command from CPU and A-bit 08 set to 0 . b) Program Master Clear. c) Console Master Clear. NOTE: A/Q Write Command to Q Station 2 (A-Reg to IR) does not affect the state of FSR bit 08.
7	DBPM	DOUBLE PRECISION MODE Bit is set by an A to FSR Command (HFPU inactive) and A-bit 07 set or by a Cold Start Command in Double Precision (Q station 4). When bit is set, all floating-point calculations are performed in double-precision mode (48 bits). When bit is reset, all floating-point calculations are performed in single-precision mode (32 bits). Bit is reset by: a) Program Master Clear. b) Console Master Clear. c) Cold Start Command in Single Precision (Q station 3).
6	FEND	FLOATING POINT EXECUTION ENDED. Bit is set by: a) The HFPU execution of a FEND instruction. b) An A to FSR (HFPU inactive) Command from the CPU and A-bit 06 set to a 1 .

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT (Contd)

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
		Bit is reset by: a) An A to FSR (HFPU inactive) Command from the CPU and A-bit 06 set to a 0 . b) Cold Start Command. c) Program Master Clear. d) Console Master Clear.
5	UNUSED	Bit is always reset.
4	PROT	PROTECT MODE When bit is set it places the HFPU in a protected device mode. This mode allows the HFPU to write into protected memory locations via the DSA channel. Bit is set by a protected A-Reg to FSR Command from the CPU and A-bit 04 set to 1 , by a protected A/Q Cold Start command, by a protected A/Q Stop Command, or by the presence of the A/Q protect jumper. Bit is reset by: a) An unprotected A Reg to FSR Command. b) An unprotected A/Q Cold Start Command. c) Program Master Clear. d) Console Master Clear. FSR Bit 4 stored at SSAR during the STOP Command will reflect the DSA protect mode of the HFPU prior to the STOP Command.
3	PTFT	PROTECT FAULT When bit is set, it indicates that the HFPU was not in protect mode and made a write data access to a protected memory location. Bit is also set or reset by an A to FSR (HFPU inactive) Command from the CPU and the state of A-bit 03. Bit is also reset by: a) Cold Start Command. b) Program Master Clear. c) Console Master Clear.

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT (Contd)

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
2 and 1	SCAM	SCANNER ACCESS MODE State of these bits selects or indicates one of three modes of HFPU DSA Channel accesses. These modes are: Bit 2 / Bit 1 / Access Mode 0 / 0 / BLOCK. The HFPU will stop the scanner for up to five successive memory cycles during a Command code word fetch. The HFPU will not release the scanner before determining if the first command yte of that word requires memory. If the first command requires memory the HFPU will hold the scanner and access memory to fetch the address-pointer word and one, two, or three operands. If the first command byte does not require memory or is a Branch Accumulator command and the FPAC is active the HFPU will release the scanner. In either case, the second, third, and fourth command bytes that require memory must wait for the scanner to return to the HFPU. These bytes can hold the scanner for up to four memory cycles. Block mode will activate (first access) or maintain (second through fifth access) the DSA PRIORITY signal for all memory accesses subject to restrictions found elsewhere in this specification. 0 / 1 / HOG. Once the HFPU is started the scanner will be held until the HFPU executes a FEND instruction. DSA PRIORITY signal will be active from start to finish. 1 / 0 / WORD. Scanner will be released after every DSA data word access. DSA PRIORITY signal will not be active.

TABLE 2.1. FUNCTION/STATUS REGISTER BIT ASSIGNMENT (Contd)

BIT POSITION	BIT MNEMONIC	BIT DEFINITION
		These bits are set by: a) An A to FSR (HFPU inactive) Command from the CPU with A-bit 02 set to a 1 and/or A-bit 01 set to a 1 . These bits are reset by: a) An A to FSR (HFPU inactive) Command from the CPU with A-bit 02 set to a "0 and/or A-bit 01 set to a 0 . b) Program Master Clear. c) Console Master Clear.
0	PCLR	PROGRAM MASTER CLEAR When HFPU receives A-bit 00 set and an A to FSR Command, it will clear all timing, reset the unit to an idle state and clear all registers with the exception of the PCR and the FPAC. The HFPU will ignore any other A bits that are set. Bit is not used on an A/Q Read Command. The PCLR function is identical in all respects to a Console Master Clear.

The HFPU will Externally Reject any attempt to Read/Write the PCR while the HFPU is active and Q-Station 3 or 4 is used. The HFPU will permit the PCR to be read at any time with an A/Q Read Command to Q-Station 5.

e) FPAC = Floating Point Accumulator

This register is the main arithmetic register in the HFPU. It is 32-bits wide for single precision and 48-bits wide for double precision. (See paragraph 2.2.1.1, FPAC format). The FPAC can be accessed via A/Q channel Writes or Reads to Q-Station 6, 7, 8 or via the DSA by any of several command codes. The HFPU will externally reject any attempt to Write/Read the FPAC via the A/Q channel if the HFPU is active.

f) SSAR = Stop Save Address Register

This register contains a 16-bit digital number used as an absolute address for the starting location in memory of where to save the HFPU registers when a Stop order is issued. It is addressable only by the A/Q channel. The HFPU will accept an SSAR write Command at any time if the SSAR Command is protected. The HFPU will return an external reject to the CPU if the SSAR write Command is not protected regardless of the A/Q protect jumper setting.

In addition to the accessible registers, the HFPU contains several internal registers, the most important of which is the Look-Ahead Buffer (LABF) which, combined with some parallelism in the logic, is used to speed sequential operations. The LABF consists of three 16-bit registers which are used to hold the operand for the next floating-point calculation. This extra register allows operands to be fetched from memory while a preceeding floating-point operation on the FPAC is still in process. Additionally, the logic parallelism alluded to above allows certain HFPU operations to execute to completion while an FPAC operation is in process.

The effects of this look-ahead feature are discussed further in section 2.2.1.8.

2.2 Characteristics

The HFPU recognizes 16 unique command-codes in its CCR. Command-code 0 is recongnized as a special two-byte command-code; that is, the next byte is the command to be executed. This increases the number of available command-codes to 31, of which 25 are used in the HFPU. These command codes are listed in table 2.2. After the HFPU is activated, it responds to a FLOT calling sequence.

A basic FLOT calling sequence consists of an instruction word consisting of four commands, followed by the operand addresses (Address Pointers). The left most 4-bit byte is the first operation; the operand addresses, if they are required, follow in the same order as the operation bytes, one word per byte. As many bytes may exist as desired, but the terminating byte must be a 4, the operation FEND.

Example:

		15 12	11 8	7 4	3 0 ← CPU Bits
CPU WORD LOCATIONS	P	OP1	OP2	OP3	Op4
	P+1		A1		
	P+2		A2		
	P+3		A4		
	P+4	OP5	OP6	4(Fend)	- - -
	P+5		A5		
	P+6		A6		

The OP's are the operation codes; the A's are their operand addresses. Not all operations require memory access; in the example, OP3 does not have a corresponding A3.

TABLE 2.2. COMMAND-CODE DEFINITION

CODE MNEMONIC	4-BIT CODE	DESCRIPTION
FLOF	1	FLOAT TO FIXED The contents of the FPAC are converted to fixed point and the results stored at the effective operand address. FPAC Bits 16-31 will contain the fixed-point number. If positive overflow occurs, FPAC 16-31 will contain 7FFF. If negative overflow occurs, FPAC 16-31 will contain 8000. The raw, unmultiplied Index value will be used in effective address formation for FLOF.
FIXF	2	FIXED TO FLOAT The contents of the effective operand address are converted to floating point and the result placed in the FPAC. The raw, unmultiplied, index value will be used in effective address formation for FIXF.
STRI	3	STORE INDEX Stores the contents of the Index Register at the effective operand address. Does not alter the contents of the Index Register. Indexed address information is inhibited during the execution of this instruction.
FEND	4	END of calling sequence. This operation terminates the calling sequence and causes the HFPU to return to an idle state. Execution of this code sets bit 6 and clears bit 15 in the FSR. No operand address is needed for this code.
CHMD	5	CHANGE MODE All operand addresses following this operation code in the calling sequence are made relative if the preceding addresses were absolute and absolute if preceding addresses were relative. Does not affect the Index Register value. Sets bit 9 of the FSR when relative mode address is in effect. No operand address in needed for this code.

TABLE 2.2. COMMAND-CODE DEFINITION (Contd)

CODE MNEMONIC	4-BIT Code	DEFINITION
NIDX	6	NO INDEX Clears the Index Register which disables the indexing of operand addresses. No operand address is needed for this code.
FCOM	7	FLOATING COMPLEMENT Complements the contents of the FPAC. NO operand address is needed for this code.
FSUB	8	FLOATING SUBTRACT The contents found at the effective operand address is subtracted from the contents of the FPAC and the results are then placed in the FPAC.
FMPY	9	FLOATING MULTIPLY The contents found at the effective operand address is multiplied by the contents of the FPAC and the results are placed in the FPAC.
FDIV	A_{16}	FLOATING DIVIDE The contents of the FPAC is divided by the contents found at the effective operand address and the results are placed in the FPAC.
FLDD	B_{16}	FLOATING LOAD The contents found at the effective operand address are loaded into the FPAC. This must be a normalized floating-point number.
ADDI	C_{16}	ADD TO INDEX Adds the contents of the effective operand address to the contents of the Index Register and places the result in the Index Register. Indexed address formation is inhibited during the execution of the instruction.
FLST	D_{16}	FLOATING STORE The contents of the FPAC are stored at the effective operand address. The contents of the FPAC are not altered by this operation.
FADD	E_{16}	FLOATING ADD The contents found at the effective operand addresses are added to the contents of the FPAC and the results are placed in the FPAC.

TABLE 2.2. COMMAND-CODE DEFINITION (Contd)

CODE MNEMONIC	4-BIT CODE	DEFINITION
INDX	F_{16}	The contents found at the effective operand address are loaded into the Index Register. The operand addresses of all subsequent FLOF, FLDD, FLST, FADD, FSUB, FMPY, FDIV and FIXF operations will be affected in the following manner: a) If FSR bit 8 is clear, the contents of the Index Register will be multiplied by 2 when the unit is in single precision mode and the effective operand address is being formed. The contents of the Index register will not be changed. b) If FSR bit 8 is clear, the contents of the Index Register will be multiplied by 3 when the unit is in double precision mode and the effective operand address is being formed. The contents of the Index Register will not be changed. c) If FSR bit 8 is set, the raw Index Register contents will be added to the base address when the effective address is being formed. d) For the functions FLOF and FIXF, the raw Index value will always be used.
SPEC	0	SPECIAL COMMAND CODE This code causes the HFPU to recognize the next byte as a code within the following Branch (jump) command-code subset. If the next byte is a "0", a FEND will be executed.
*CACS	1	CONTINUE ANOTHER CALLING SEQUENCE Starts a new floating-point instruction sequence by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the Command-Code Register (CCR). The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction.

*These command-codes are executed only if the preceding byte is a SPEC code.

TABLE 2.2. COMMAND-CODE DEFINITION (Contd)

CODE MNEMONIC	4-BIT CODE	DEFINITION
*BRAM	2	BRANCH ACCUMULATOR MINUS If the condition is satisfied (FPAC Negative), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the Program Count Register will be incremented by (+1) before the next command code is executed.
*BRAZ	3	BRANCH ACCUMULATOR ZERO If the condition is statisfied (FPAC Zero), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.
BRAN	4	BRANCH ACCUMULATOR NON-ZERO If the condition is satisfied (FPAC non-zero), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.
*BRAP	5	BRANCH ACCUMULATOR POSITIVE If the condition is satisfied (FPAC POSITIVE including POSITIVE ZERO), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.

*These command-codes are executed only if the preceding byte is a SPEC. NOTE: Codes A-F, when preceded by a SPEC code, will be executed as FEND.

TABLE 2.2. COMMAND-CODE DEFINITION (Contd)

CODE MNEMONIC	4-BIT CODE	DESCRIPTION
*BRIM	6	BRANCH INDEX REGISTER MINUS If the condition is satisfied (IR NEGATIVE), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.
*BRIZ	7	BRANCH INDEX REGISTER ZERO If the condition is satisfied (IR ZERO), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.
*BRIN	8	BRANCH INDEX REGISTER NON-ZERO If the condition is satisfied (IR NON-ZERO) the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instriction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.
BRIP	9	BRANCH INDEX REGISTER POSITIVE If the condition is satisfied (IR POSITIVE), the HFPU continues execution by loading the effective operand address into the PCR and then loading the contents of the effective operand address into the CCR. The new code execution will start at OP byte one. Indexed address formation is inhibited during the execution of this instruction. If the condition is not satisfied, the PCR will be incremented by (+1) before the next command is executed.

*These command-codes are executed only if the preceding byte is a SPEC. NOTE: Codes A-F, when preceded by a SPEC code, will be executed as FEND.

2.2.1 Command Description

Code Mnemonic	4-bit Code	Brief Description	Indexed Addressing
SPEC	0	"Special" (2-byte) command Code	N/A
FLOF	-1	FLOAT to FIXED conversion	X1
FIXF	2	FIXED to Floating Conversion	X1
STRI	3	STORE Index value	NO
FEND	4	END of calling sequence	N/A
CHMD	5	Change Relative Address Mode	N/A
NIDX	6	No Index	N/A
FCOM	7	Floating Complement	N/A
FSUB	8	Floating Subtract	X1,2,3
FMPY	9	Floating Multiply	X1,2,3
FDIV	A	Floating Divide	X1,2,3
FLDD	B	Floating Load	X1,2,3
ADDI	C	Add to Index	NO
FLST	D	Floating Store	X1,2,3
FADD	E	Floating Add	X1,2,3
INDX	F	Load Index value	NO
*FEND	Ø	End of Calling Sequence	N/A
*CACS	1	Continue Another Calling Sequence	NO
*BRAM	2	Branch if Accumulator Minus	NO
*BRAZ	3	Branch if Accumulator Zero	NO
*BRAN	4	Branch if Accumulator Non-zero	NO
*BRAP	5	Branch if Accumulator Positive	NO
*BRIM	6	Branch if Index Minus	NO
*BRIZ	7	Branch if Index Zero	NO
*BRIN	8	Branch if Index Non-zero	NO
*BRIP	9	Branch if Index Positive	NO
*FEND	A-F	End of Calling Sequences	N/A

*These command codes are executed only if the preceding byte is a SPEC code.

The Operation codes listed above which do not require an address have N/A in the indexed addressing column. All other operation codes require the presence of an address word.

For the special command code operations, the effective address itself is the argument for the function (the effective address is loaded into the PCR). For all other functions, including INDX, ADDI and STRI, the effective address points to a memory location (or locations) which contains or will contain the argument.

The address for all functions can be either absolute or relative as determined by the state of the Relative Mode bit (bit 9) in the FSR. If bit 9 is clear, addresses are absolute. If bit 9 is set, addresses are Relative to the location in which the address-pointer word resides (to the PCR). If relative, the PCR will be added to the Address Pointer word in the process of forming the effective address.

For the functions which specify "X1" or "X1,2,3" in the indexed addressing column, the index value will also be added to the address-pointer word in forming the effective address. The index value may be multiplied by 1,2 or 3 before the addition depending on the state of the double-precision bit in the FSR (bit 7) for the functions with "X1,2,3". For the functions with "X1" in the indexed addressing column, the index times one is always used.

2.2.1.1 Operand Addressing. All operand addresses used within the HFPU will conform to one of the following methods:

a) Absolute (16-bits)
b) Relative (16-bits with Bit 15 = Sign)
c) Indexed (16-bits)
   Value in Index register will be multiplied by 2 for single-precision operations and by 3 for double precision operation if FSR bit 8 is clear.
d) Relative Indexed. (2 x Index or 3 x Index; 1 x Index if FSR bit 8 is set)

Figure 2.3 depicts the address methods.
All address arithmetic is 16-bit, ones-complement arithmetic. It is identical with the 16-bit arithmetic of the System 17 CPU.

OPERATION NOTES:

If FSR bit 9 is set, relative-addressing mode is in effect. If FSR bit 9 is clear, absolute addressing is in effect. Absolute addressing means that the pointer word is in an absolute address; conversely, relative-addressing means that the pointer word is a 16-bit signed displacement from the current PCR.

If FSR bit 8 is clear, the contents of the index register will be multiplied by 2 or by 3 and added to the argument address (pointer word) to obtain the final address. If FSR bit 8 is set, the contents of the index register will be added to the argument address to obtain the final address.

Abbreviations EA = Effective address
(PCR) = Program Counter Register contents
(IR) = Index Register Contents
PA = Pointer Address

1. ABSOLUTE (IR) = 0

LOCATION		CONTENTS		
0100_{16}	=	$B444_{16}$	=	Command-Code (FLDD, FEND. . . .)
0101_{16}	=	0200_{16}	=	Pointer Address (ABS) Effective Address = PA = 200_{16}
0200_{16}	=	$XXXX_{16}$	=	Operand
$C201_{16}$	=	$XXXX_{16}$	=	Operand
0202_{16}	=	$XXXX_{16}$	=	Operand (D.P. Only)

2. RELATIVE (IR) = 0

0100_{16}	=	$B444_{16}$	=	Command-Code (FLDD, FEND. . . .)
0101_{16}	=	0200_{16}	=	Pointer Address (Rel) EA = PA + (PCR) = 200 + 101 = 301_{16}
0301_{16}	=	$XXXX_{16}$	=	Operand
0302_{16}	=	$XXXX_{16}$	=	Operand
0302_{16}	=	$XXXX_{16}$	=	Operand (D.P. Only)

3. INDEXED where (IR) = 100, S.P. mode and FSR Bit 8 clear

0100_{16}	=	$B444_{16}$	=	Command-Code (FLDD, FEND . . . .)
0101_{16}	=	0200_{16}	=	Pointer Address EA = PA +2*(IR) = 200 + 200 = 400_{16}
0400_{16}	=	$XXXX_{16}$	=	Operand
0401_{16}	=	$XXXX_{16}$	=	Operand

Figure 2.3. Addressing Examples (Sheet 1 of 3)

7. Indexed where (IR) = 100 and Command Code is FLOF or FIXF FSR bit 8 set or clear.

100_{16}	=	1444_{16}	= Command Code (FLOF, FEND . . . .)
101_{16}	=	200_{16}	= Pointer address EA = PA + (IR) = 200 + 100 = 300
300_{16}	=	$XXXX_{16}$	= FLOF Result will be stored here.

8. Indexed where (IR) = 100, and FSR bit 8 is set (compare with #3 and #4 above)

100_{16}	=	$B444_{16}$	= (FLDD, FEND . . .)
101_{16}	=	0200_{16}	= Pointer address EA = PA + (IR) = $200 + 100 = 300_{16}$
300_{16}	=	$XXXX_{16}$	= Operand
		$XXXX_{16}$	= Operand

9. Relative Indexed where (IR) = 100 and FSR bit 8 is set (compare with #5 and #6 above)

100_{16}	=	$B444_{16}$	= (FLDD, FEND . . .)
101_{16}	=	0200_{16}	= Pointer Address EA = PA + (PCR) + (IR) = $200+101+100 = 401_{16}$
401_{16}	=	$XXXX_{16}$	= Operand
402_{16}	=	$XXXX_{16}$	= Operand
403_{16}	=	$XXXX_{16}$	= Operand

10. Special Command Code, Relative mode

100_{16}	=	0100_{16}	= (SPEC, CACS . . .)
101_{16}		0200_{16}	= Pointer Address EA = PA + (PCR) = 200 + 101 = 301
301_{16}	=	$XXXX_{16}$	= Next command Code word. Beginning of next calling sequence.

Figure 2.3. Addressing Examples (Sheet 2 of 3)

12. Index command (ABS)

100_{16}	=	$F400_{16}$	=	(INDX, FEND . . .)
101_{16}	=	0200_{16}	=	Pointer Address EA = PA
200_{16}	=	$XXXX_{16}$	=	Operand to be loaded into the IR

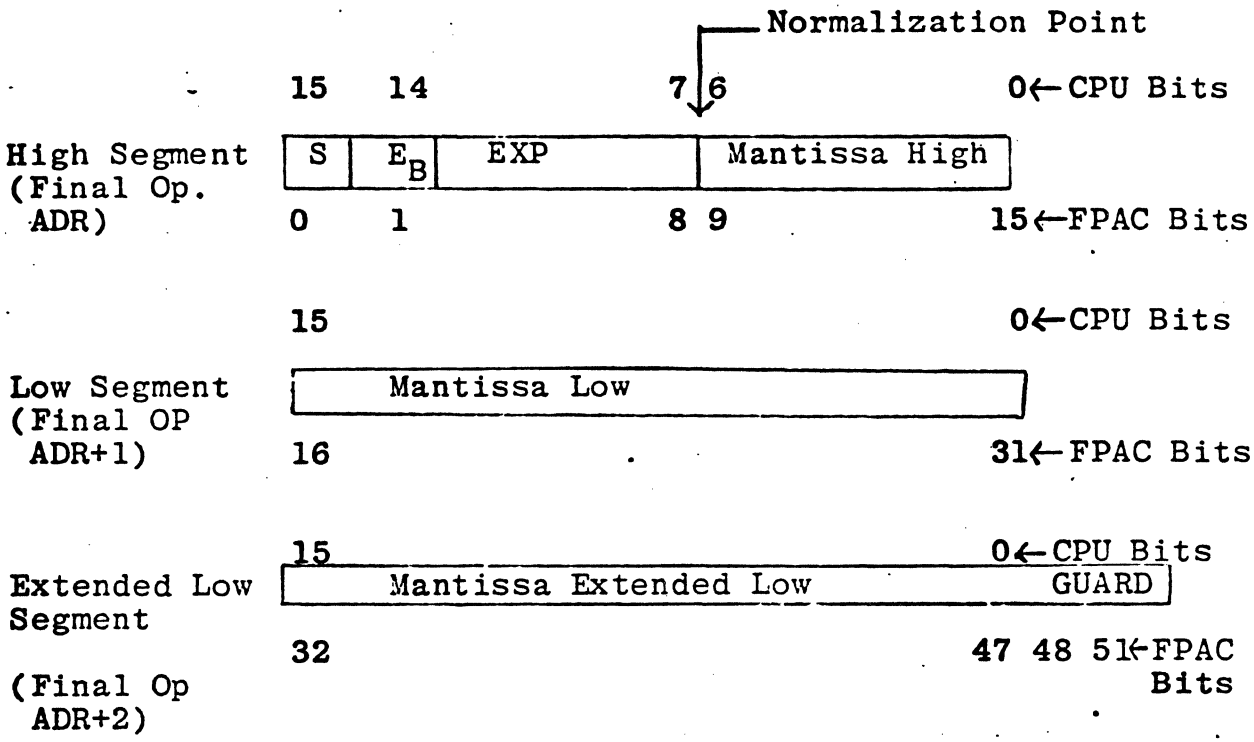
13. Index command (REL)

100_{16}	=	$F400_{16}$	=	(INDX, FEND . . .)
101_{16}	=	FFFD	=	Pointer Address EA = PA + (PCR) = FFFD + 101 = $00FF_{16}$
$00FF_{16}$	=	$XXXX_{16}$	=	Operand to be loaded into the IR.

NOTE: This last example demonstrates the effect of the memory wrap-around in a "backwards" relative pointer address. It is simply a case of an end-around carry resulting from the use of one's complement arithmetic.

Figure 2.3. Addressing Examples (Sheet 3 of 3)

2.2.1.2 Operand/FPAC Format. Floating-point numbers used in the arithmetic operations have the following format.



Where:

S = Sign bit of the entire floating-point number. When the Sign bit = 0 , the floating-point number is positive. When the Sign bit = 1 , the floating-point number is negative.

E_B = Exponent Sign Bit which is biased by an exclusive OR with 80_{16}.

EXP = Seven binary bits which represent the magnitude of the exponent. ($-127 \leq EXP \leq 127$).

Mantissa = Normalized magnitude of the floating-point number which is a fractional coefficient. A normalized positive coefficient has the form (.1XXX...X_{Low}) where S = 0 . A normalized negative coefficient has the form (.0XXXX...X_{Low}) where S = "1".

NOTES: 1) A single-precision number has the expressable number range:

$$-2^{127}(1-2^{-23}) \leq X \leq 2^{127}(1-2^{-23})$$

2) A double-precision number has the expressable number range:

$$-2^{127}(1-2^{-39}) \leq X \leq 2^{127}(1-2^{-39})$$

3) When the floating-point number is negative, the entire FPAC including the Exponent is in ONE'S complement form.

4) A floating-point zero is represented as all bits set to 0 . It is the only legal unnormalized number.

5) The floating-point number should always be normalized for any floating-point arithmetic operation including FLST and FLDD.
The use of unnormalized numbers as inputs to any floating-point operation except (FIXF) will generally result in incorrect answers. Teh result of FADD, FSUB, FMPY, FDIV and FIXF will always be a normalized number or zero.

6) The extended low segment of the operand is used for double-precision mode.

7) If the exponent of the result of a FADD, FSUB, FMPY or FDIV is larger than 127, exponent overflow has occurred and the answer is set to the largest value having the same sign as the actual result (7FFF,FFFF, FFFF or 8000, 0000, 0000 in D.P.; 7FFF, FFFF or 8000, 0000 in S.P.). If the exponent of the result is less than -127, exponent underflow has occurred and the result is set to floating-point zero.

8) If the divisor for an FDIV is unnormalized or equal to zero, a divide fault has occurred and the result is set to the largest value having the same sign as the dividend.

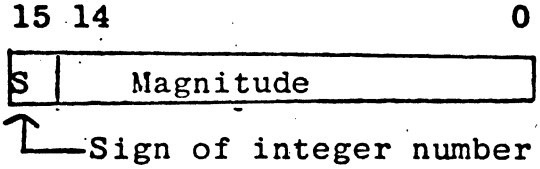
2.2.1.3 Rounding. Internally, the FPAC has four extra bits as shown in the diagram of the preceding section. These extra bits on the least significant end (FPAC bits 48 to 51) are referred to as a guard digit and are used to increase the accuracy of the calculations by providing an arithmetic residue which is used to round the final result.

The rounding algorithm used is of the non-convergent, away-from-zero type. That is, if the number is positive and the residue is greater than or equal to one-half the value of the least significant bit (lsb), then one lsb is added to the result. If the number is negative and the residue is less than one-half the lsb, then one lsb is subtracted from the result (one's complement arithmetic assumed).

After rounding, the bits of the guard digit are set equal to the sign bit i.e., equal to zero in one's complement arithmetic.

Note that in single precision, bits 32 to 51 of the FPAC act as the guard digit.

2.2.1.4 Fix Float Number Conversions. The integer (fixed) number format is:



Where: S=0 = positive number

S=1 = negative number with the magnitude in ONE's complement form.

The Float-to-Fixed operation is performed by executing command code 1 which converts the floating-point number in the FPAC register to an integer and transfers the integer to the effective operand address. FPAC 31-16 will also contain the result.

The Fixed-to-Float operation is performed by executing command-code 2 which loads an integer number into the HFPU, begins a conversion process, and upon completion, places the floating-point number into the FPAC. This number may be retrieved in one of two ways.

1) A status of the HFPU FPAC register by successive A/Q Read Commands to Q-stations 7,8 and 9.

2) Executing a FLST instruction to a specified memory location.

2.2.1.5 HFPU Initialition Sequences. There are three methods used to initialize the HFPU. These methods are:

1) Cold Start - Single precision (S.P.)

2) Cold Start - Double precision (D.P.)

3) Protected Re-Start - Single or Double precision.

A Cold Start (S.P.) Command is used when first entering the FLOT subroutine, and a Cold Start (D.P.) Command is used when first entering the DFLOT subroutine. Each type of Cold Start uses a unique Q-Station Address. A Re-Start Command is used when re-entering either the FLOT or DFLOT subroutine after the HFPU has been interrupted by a stop order command for service of a higher priority routine. Refer to figure 2-1 as an aid for the following description of events:

A Cold Start Sequence is initiated by the following sequence of events:

a) The FSR is loaded from the CPU A-register by an A/Q Write Command to Q-station 0 if a special set-up such as a change in scanner access mode is desired. The format used for the FSR is depicted in figure 2.2 and the FSR bit definition is listed in table 2.1. If no special set-up is required, the starting point for a Cold Start.

b) The PCR is loaded from the CPU A-register by an A/Q Write Command to Q-Station 3 or 4. If the A/Q Write Command is to Q-Station 3, the unit will start in single-precision mode and will clear bit 7 in the FSR. If the A/Q Write Command is to Q-Station 4, the unit will start in double-precision mode and will set bit 7 in the FSR. Either Cold Start Command will clear the Index Register and clear FSR bits 3, 6, 9, 10, and 11. The address transferred to the PCR is the address of the first command-code instruction word. When the HFPU accepts the starting address word, it goes into an active state (Bit 15 of the FSR is set) and loads the CCR via the DSA channel. The unit will remain in an active state until it either executes a FEND instruction, receives the Stop order command described in 5.2.1.6, or receives an A/Q Write Command to Q-Station 0 with A-Bit 00 = 1 (PCLR).

2.2.1.6 HFPU Stop/Restart Sequence. A Protected Stop order may be issued at any time while the HFPU is in an active or inactive state. The HFPU will reject an unprotected Stop Command regardless of the setting of the HFPU A/Q Protect Bit jumper plug. A Stop Order is accomplished by the following sequence of events.

a) An A/Q Write Command to Q-Station 9 where the CPU A-register is transferred to the SSAR as the Stop and Save address.

b) As soon as the HFPU completes its present arithmetic operation, it will use the contents of the SSAR as the ABSOLUTE address in CPU memory of where to start storing the contents of the following registers.

SSAR = (FSR)

SSAR+1 = (CCR)*

SSAR+2 = (IR)

SSAR+3 = (PCR)

SSAR+4 = (FPAC, $BITS_{00}$ - 15)

SSAR+5 = (FPAC, $BITS_{16}$ - 31)

SSAR+6 = (FPAC, $BITS_{32}$ - 47)

*The CCR format will reflect the current status of the Command Code Word, that is, bits 15 - 12 will contain the next command code to be executed. Example:

1) CCR read from CPU

OP1	OP2	OP3	OP4

2) CCR stored on STOP command

OP2	OP3	OP4	OP1

c) When the HFPU has completed the storing of the last register, it will go inactive and clear bit 15 of the FSR.

NOTE: A Stop Order issued while the HFPU is inactive will cause the HFPU to go active (Bit 15 of FSR set) for the time required to store the six registers. The HFPU will return to the inactive

state (Bit 15 of FSR clear) upon completion. The stored FSR will reflect the state of the HFPU when the stop order was issued (Bit 15 clear).

After a Stop Order is issued, the HFPU may be restarted from the point of interruption by a protected RE-start command. The HFPU will reject an unprotected Re-start command regardless of the setting of the HFPU A/Q Protect Bit jumper plug. A Re-start command is an A/Q Write command to Q-station 5 where the contents of the CPU A-register is transferred to the SSAR and the following events take place:

a) The HFPU goes to an active state and bit 15 of the FSR is set.

b) The HFPU uses the SSAR contents as an absolute starting address of where to start the retrieval of the registers saved on the receipt of the Stop order in the following manner.

SSAR	Restore FSR
SSAR+1	Restore CCR
SSAR+2	Restore IR
SSAR+3	Restore PCR
SSAR+4	Restore FPAC (Bits 00-15)
SSAR+5	Restore FPAC (bits 16-31)
SSAR+6	Restore FPAC (Bits 32-47)

c) When the HFPU registers are restored, the unit will pick-up where it left off and continue to execute command-codes if the active bit in the restored FSR (Bit 15) is set. If this bit is not set, the HFPU will go to a not active or idle state.

2.2.1.7 Function/Status Register Definitions

The function/status register definitions are shown in figure 2.2 and detailed in table 2.1.

2.2.1.8 Hardware Execution Times. Table 2.3 lists the worst case execution times for the functions performed by the HFPU. This table also displays the improvement in execution times that can be expected in "typical" usage due to the presence of the hardware look-ahead feature. This feature allows parallelism to take place within the HFPU. This parallelism can occur because of the ability of the HFPU to perform non-FPAC operations (Fetch of Command-Code words, Index Register operations, Fetch of operands to Look-Ahead-Buffer etc.) while an operation involving the FPAC is in process (FADD, FSUB, FMPY, FDIV, FLDD, FIXF, or FCOM).

Three columns in table 2.3 illustrate the effects of this overlap. The column labeled "Overlappable Component" shows the portion of the FPAC functions that can operate in parallel with other non-FPAC functions. The next column, labeled "Irreducible Component", shows the portion of the execution time that cannot execute in parallel with any other functions. For the FPAC functions, this is the time required to transfer the Look-Ahead-Buffer contents into the Floating Point Arithmetic unit and to start the FPAC portion of the function. For the functions which require the contents of the FPAC (FIXF, FLST, BRAM BRAZ, BRAN, BRAP, FEND), this is typically the total execution time for that function, since it must wait for the FPAC portion of the preceding function to complete before it can begin. The Irreducible portion of the FLST function consists only of the time required to store the FPAC since it can overlap the fetch of the address with the preceding FPAC function. The next column, labeled "Overlapping Component", shows the portion of any function that can operate in parallel with the FPAC protion of the preceding function. For the non-FPAC functions (Command-Code Fetch, SPEC, STRI, CHMD, NIDX, ADDI, INDX, CACS, BRIM, BRIZ, BRIN, and BRIP) this is the total execution time for that function. For the FPAC functions this is the time required to fetch the argument address and to transfer the argument from memory to the Look-Ahead-Buffer.

The next three columns of the table show the amount of DSA channel activity that will occur during any given function. The latency columns show the amount of added time that will be incurred due to delays in obtaining DSA channel access.

In most cases these latencies are incurred during the overlapping component of the function and thus will not add appreciably to the overall execution time of a given calling sequence.

The final two columns show the typical effective execution time that can be achieved if full advantage is taken of the overlap. These times are generally the sum of the overlappable component plus the irreducible component. The two exceptions are FLDD and FIXF where the apparent time is shown equal to the total time. These two functions ignore the previous contents of the FPAC and thus it is unlikely that they would be overlapped with a preceding FPAC function.

Figure 2.4 shows several example execution time computations. The execution time for a given function equals the irreducible component plus the overlappable component plus that portion of the overlapping component that is not overlapped.

TABLE 2.3. EXECUTION TIMES (worst case operands) (440ns Tac) (600ns cycle)

FUNCTION	Total Time (Hog Mode)	900ns Add	Overlappable Component	Irreducible Component	Overlapping Component	DSA CYCLES	Latencies Word Mode	Latencies Block Mode	Apparent Time with "typical" overlap (600ns)	Apparent Time with "typical" overlap (900ns)
Command-Code fetch	1.25usec	.30usec	0usec	0usec	1.25usec	1	1	1	0usec	0
SPEC	.20	0	0	0	.20	0	0	0	0	0
FLOF	4.84	.30	0	4.84	0	2	1	1	4.84	5.14
FIXF	6.77	.60	4.47	.20	2.11	2	2	1*	6.77	7.37
STRI	2.11	.60	0	0	2.11	2	2	1*	0	0
FEND	.20	0	0	.20	0	0	0	0	.20	.20
CEID	.20	0	0	0	.20	0	0	0	0	0
NIDX	.20	0	0	0	.20	0	0	0	0	0
FCOM	.71	0	.51	.20	0	0	0	0	.71	.71
FSUB(SP)	8.76	.90	5.46	.59	2.71	3	3	1*	6.05	6.05
FSUB(DP)	11.12	1.20	7.22	.59	3.31	4	4	1*	7.81	7.81
FMPY(SP)	11.62	.90	8.32	.59	2.71	3	3	1*	8.91	8.91
FMPY(DP)	15.74	1.20	11.84	.59	3.31	4	4	1*	12.43	12.43
FDIV(SP)	12.06	.90	8.76	.59	2.71	3	3	1*	9.35	9.35
FDIV(DP)	16.18	1.20	12.28	.59	3.31	4	4	1*	12.87	12.87
FLDD(SP)	4.03	.90	.73	.59	2.71	3	3	1*	4.03	4.93
FLDD(DP)	4.63	1.20	.73	.59	3.31	4	4	1*	4.63	5.83
ADDI	2.11	.60	0	0	2.11	2	2	1*	0	0
FLST(SP)	2.71	.90	0	1.65	1.06	3	3	1*	1.65	2.25
FLST(DP)	3.31	1.20	0	2.25	1.06	4	4	2*	2.25	3.15
FADD(SP)	8.76	.90	5.46	.59	2.71	3	3	2*	6.05	6.05
FADD(DP)	11.12	1.20	7.22	.59	3.31	4	4	1*	7.81	7.81
INDX	2.11	.60	0	0	2.11	2	2	1*	0	0
CACS	1.06	.30	0	0	1.06	1	1	1*	0	0
BRAM	1.45	.30	0	1.45	0	1	1	1	1.45	1.45
BRAZ	1.45	.30	0	1.45	0	1	1	1	1.45	1.45
BRAN	1.45	.30	0	1.45	0	1	1	1	1.45	1.45
BRAP	1.45	.30	0	1.45	0	1	1	1	1.45	1.45
BRIM	1.45	.30	0	0	1.45	1	1	1*	0	0
BRIZ	1.45	.30	0	0	1.45	1	1	1*	0	0
BRIN	1.45	.30	0	0	1.45	1	1	1*	0	0
BRIP	1.45	.30	0	0	1.45	1	1	1*	0	0
BRAx(false)	.39	0	0	.39	0	0	0	0	.39	.39
BRIx(false)	.39	0	0	0	.39	0	0	0	0	0
STOP	5.24	2.10	0	5.24	0	7	7	1	5.24	7.34
RESTART	6.35	2.10	0	6.35	0	7	7	1	6.35	8.45

*one fewer latency required if first command in a newly fetched Command-Code word.

Latency = .74 to 1.32usec (600ns) (no Refresh)
= .85 to 1.68usec (900ns) (no Refresh)

Latency figures include typical scanner delay (300ns) plus observed Tac degradation due to DSA TTL expander (220ns).

Single Precision assumed (600ns) BLOCK mode

a) FORTRAN expression A=B+C*D

Calling Sequence

```
B9ED      (FLDD, FMPY, FADD, FLST)
 D        Address of D
<C>       Address of C
<B>       Address of B
<A>       Address of A
4000      (FEND, - - - - )
```

Function	Time	Latencies	Comments
Fetch Command Code	1.25	1	Total time no overlap
FLDD	4.63	0	Total time no overlap
FMPY	10.89	1	Total less the overlappable of FLDD
FADD	6.05	0*	Total less the overlapping of FADD
FLST	2.25	1	Irreducible Component
Fetch C.C.	1.25	1	No overlap
FEND	.20	0	No overlap
	26.52usec	1	latencies

*Latency overlaps preceding function

b) FORTRAN Expression A(I) = B(J)+C(K)*D(L)

Calling Sequence

```
FBF9      (INDX, FLDD, INDX, FMPY)
<L>       address of L
<D>       address of array D
<K>       address of K
<C>       address of array C
FEFD      (INDX, FADD, INDX, FLST)
<J>       address of J
<B>       address of array B
<A>       address of array A
4000      (FEND . . . . )
```

Figure 2.4. Execution Time Examples (Sheet 1 of 2)

Function	Time	Latencies	Comments
Fetch CC	1.25	1	No overlap
INDX	2.11	0	No overlap
FLDD	4.63	1	No overlap
INDX	1,38	1	Total-overlappable of FLDD
FMPY	11.62	1	FLDD overlappable used up
Fetch CC	0	0*	overlapped, 2.57 used+ 5.75 left
INDX	0	0	overlapped, 2.11 used 3.64 left
FADD	6.44	0*	partially overlapped 4.03 used+ -0.39 left FMPY overlappable used up .39 added to FADD
INDX	0	0+	overlapped 3.43 used+ 2.03 left
FLST	2.60	0+	partially overlapped 2.38 used+ -.35 left FADD overlappable used up
Fetch CC	1.25	1	.35 added to FLST
FEND	.20	0	
	31.48 usec	5 latencies	

*Latency overlapped

+ used time includes the overlapping component of the function plus the latency (1.32usec). It is the amount of the preceding functions overlappable component used up by the current function.

Figure 2.4. Execution Time Examples (Sheet 2 of 2)

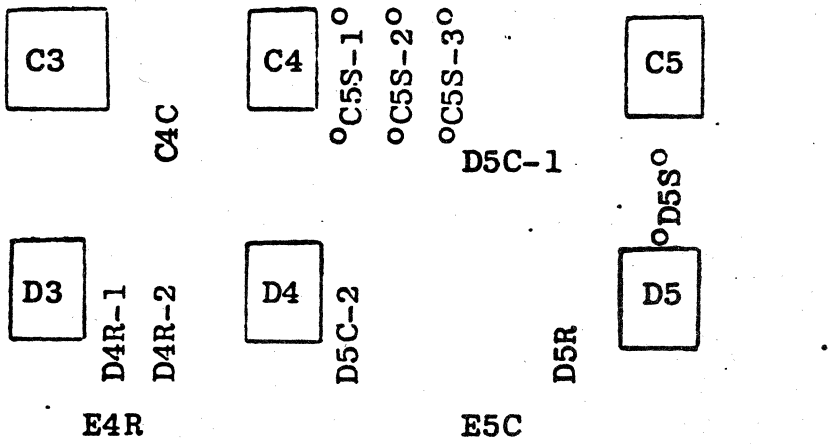
3.1 LOGIC CARD INSTALLATION

3.1.1 Inspection. Examine the cards closely for evidence of damage in shipping, broken or missing components, gouges in board coating, etc. Record all discrepancies.

3.1.2 Installation of Jumpers. A rectangular coordinate system is used for locating components on the logic cards. Facing the card from the component side with the backplane connector at the bottom, the integrated circuits appear to be laid out in four horizontal rows with 16 chips in each row. These rows are labeled A, B, C and D going from top to bottom. The columns of integrated circuits are labeled from 1 to 16 going from left to right. Labels on the rows and columns appear at the left and top edges of the board, respectively. Thus the chip at the upper left-hand corner is labeled A1 and the chip at the lower right-hand corner is labeled D16.

Passive components are given unique locating labels with respect to this grid. Components which lie to the left and/or above an integrated circuit grid position are given a designator that consists of that grid position, a letter (R = resistor, C = capacitor, S = strap or jumper) and a consecutive number (if there is more than one component of the same type within a given grid position). The consecutive numbers are assigned in the order: top, left to right; side, left to right. For the purposes of labeling passive components near the bottom edge of the board, the E row of chips is assumed to exist.

Example:



All components are identified relative to this grid in the schematics and parts lists so that direct references to the physical boards can be made without the need to refer to a topology or illustrated parts list.

3.1.2.1 DSA Board. There are five jumper (strap) locations on the DSA board. A single jumper in one of these locations is used to determine the position of HFPU in the DSA scanner chain. The jumper locations and their functions are given in table 3.1

TABLE 3.1. DSA Scanner Position Select Jumpers

Scanner Position	Jumper Location on DSA Board
Middle	C11S-2
First	C12S-2
Last	C11S-1
Only	C12S-1
Out	C11S-3

For correct operation of the DSA scanner, one jumper should be installed in one of the locations specified above in order to select the desired DSA Scanner position for the HFPU.

3.1.2.2 A/Q Board. Jumpers are provided on the A/Q board to select the HFPU equipment address and to place the HFPU in the Protected Mode . An Additional jumper has been provided for use with HFPU units which do not have the double-precision option. This jumper forces the HFPU to respond to all commands as if they were single-precision commands. Table 3.2 summarizes the jumpers on the A/Q Board.

TABLE 3.2. A/Q EQUIPMENT ADDRESS, PROTECT MODE, AND SINGLE-PRECISION DEVICE JUMPERS

Mnemonic	Function Location	Function Description
Q10	E14S-1	MSB of equipment address select. Install jumper for a "1" in the Address.
Q9	E14S-2	Next MSB of equipment address.
Q8	E14S-3	Next MSB of equipment address.
Q7	E13S	LSB of equipment address.
PTCT	B12S	Protected Mode jumper. Install for Protected Mode Remove for Unprotected Mode
SPDEV	B13S	Single-Precision Device. Install if single precision; i.e., if double-precision option is not present.

TABLE 3.3. HEXADECIMAL CODE FOR EQUIPMENT SELECT

Jumper Location		E14S-1	E14S-2	E14S-3	E13S	
Hexadecimal Code (Q10-Q	0	0	0	0	0	
	1	0	0	0	1	Note:
	2	0	0	1	0	a 1 in the
	3	0	0	1	1	binary code
	4	0	1	0	0	indicates the
	5	0	1	0	1	presence of a
	6	0	1	1	0	jumper.
	7	0	1	1	1	
	8	1	0	0	0	
	9	1	0	0	1	
	A	1	0	1	0	
	B	1	0	1	1	
	C	1	1	0	0	
	D	1	1	0	1	
	E	1	1	1	0	
	F	1	1	1	1	

3.1.2.3 SPALU Board. One jumper is provided on this board to accommodate the double-precision option. Its function is to insure the correct propogation of carry through the mantissa arithmetic logic when the double-precision option is not selected. A second jumper is provided for end-around shifting when the double-precision option is not installed.

Jumper Location	Function
D9S, E14S	These jumpers must be installed if the double-precision option is not present. If the double-precision option is installed, remove these jumpers.

3.1.3 Board Installation. The boards should be inserted in the standard or alternate slots as indicated in table 1.1. The power in the CPU and the expansion chassis should be off.

Examine the expansion chassis backplane for possible bent pins and straighten them. Insert and remove each card in sequence checking the backplane for bent pins afterwards.

Carefully straighten any resulting bent pins and insert all the cards.

3.2 Mother-Board Installation and Removal.

3.2.1 Preparation.

3.2.1.1 The Backplane. Visually inspect the area of the backplane opposite to the slots used for the HFPU logic cards for bent pins. A pin misalignment of approximately the width of the backplane pin itself (25 mils) can be tolerated by the vertical receptacles on the mother board.

3.2.1.2 The Mother Boards. Viewing each mother board from the side and top edge, sight down the rows of receptacles looking for ones that may have been bent out of alignment. A receptacle misalignment of approximately one-half the width of the opening at the top of the receptacle (25 mils) can be tolerated. The receptacles can be straightened using a needle-nosed pliers.

The examination and straightening (as required) should be carried out for all rows as viewed from both the side and the top edge of each mother-board card.

3.2.2 Installation. Begin with the boards that cover the high numbered pins on the P2 (bottom) row of connectors. Orient each board with the lettering up and the receptacles pointing towards the backplane (away from you). Carefully align two corner receptacles with the backplane pins on the slot chosen for one of the outside logic cards (ADDR, slot 23, or EXP & TIM, slot 15 in the standard configuration). Start the receptacles onto the backplane pins along the chosen column to a depth of about 1/32 inch. Gently push against and oscillate the board until it drops down onto all of the pins.

Once the board has mated with all the pins (it will be parallel to the backplane and the pins will have entered approximately 1/16 inch into the receptacles), it needs to be pushed down onto the pins to make electrical contact. The fibre-glass epoxy board will flex slightly so that it is not necessary to overcome the insertion force of all the receptacles at once. Holding the board in place with one hand, force one corner down about 1/32". Work around the board forcing each corner down a little further until the pins can just be seen through the holes in the bottom of the receptacles. Proper mating can be checked at this point by examining each receptacle to see the backplane pin within it.

After installing the bottom boards proceed to the next pair of boards (P2 low numbered pins) and then to the P1 Boards.

3.2.3 Removal. Attach the removal tool to the vertical edges of the mother board to be removed. Alternately lift the right side and then the left side of the board and slowly "walk" the mother-board off of the backplane pins. The P1 boards may require some manual assistance in order to get the top and bottom rows of pins started moving. CAUTION: Use one hand on the tool and the other hand to restrict movement, so that the last step does not result in an abrupt, large movement, since this will sometimes cause bent plus if one end (or side) releases before the other.

4.1 HARDWARE ORGANIZATION.

4.1.1 Device Structure. The Hardware Floating-Point Unit is structured into two semi-independent sections. The first, the interface and Master Control, handles the communication with the System 17 CPU and the interpretation of the various op-codes and interface commands. Additionally, it issues commands to the second section within the HFPU, the Hardware Floating-Point section. This Floating-Point section performs all of the arithmetic operations on the FPAC. The Master Control section is contained primarily on three boards, the ADDR, CSA, and A/Q boards. A small portion of the master control section is contained on the DPALU board. The second section of the unit, the hardware floating-point device is contained on four boards, the DPALU, SPALU, FPHMP, and EXP and TIMING. Each of these two independent sections, the Master Control and Floating Point, is controlled by its own independent Micro-Processor. The structure of the micro-processors is described more fully in section 4.1.2.

Figure 4.1 shows in more detail the internal structure of the elements that make up the HFPU and the data paths that interconnect them. The "backbone" of the device is a single, 16-bit, bidirectional bus (DATA 0 to 15). This bus is interfaced via a transceiver on the DSA and A/Q boards to the respective I/O busses of the System 17 CPU. All data transfers within the HFPU take place in 16-bit words on this bus. The structure of each of the boards that makes up the HFPU is described more fully in section 4.1.3.

4.1.2 The Micro-Processor concept. As was mentioned above the HFPU contains two micro-processors. The first of these the Master Micro-Processor, is shown as three blocks labeled Master Control A, Master Control B, and Micro-Processor Address, on the DSA, A/Q, and DPALU boards in figure 6.1. The second micro-processor, the Floating-Point Micro-Processor, is shown as the block labeled FPH-CONTROL on the FPHMP board in figure 6.1. The function of these micro-processors is to control the sequence in which data transfers take place within the HFPU. The heart of a micro-processor is its control store, in this case READ ONLY MEMORY (ROM). The outputs of the ROM are applied via instruction register to the data path controllers within the device and also to the clocks that are used to enter data into the device registers. For each step of an algorithm the bits in the ROM are programmed to generate the desired data transfer that is required by the algorithm. Sequence control is achieved by utilizing a group of bits in the ROM to specify the next ROM address that is to be accessed. This allows the micro-processor to execute essentially random sequences of micro instructions, which allows it to perform the sequences required by the algorithms. It also gives the micro-processor a great deal of flexibility in that the changing of an algorithm will require only the change in a few locations in the READ ONLY MEMORY. Additional power is given to the micro-processor sequencing by providing

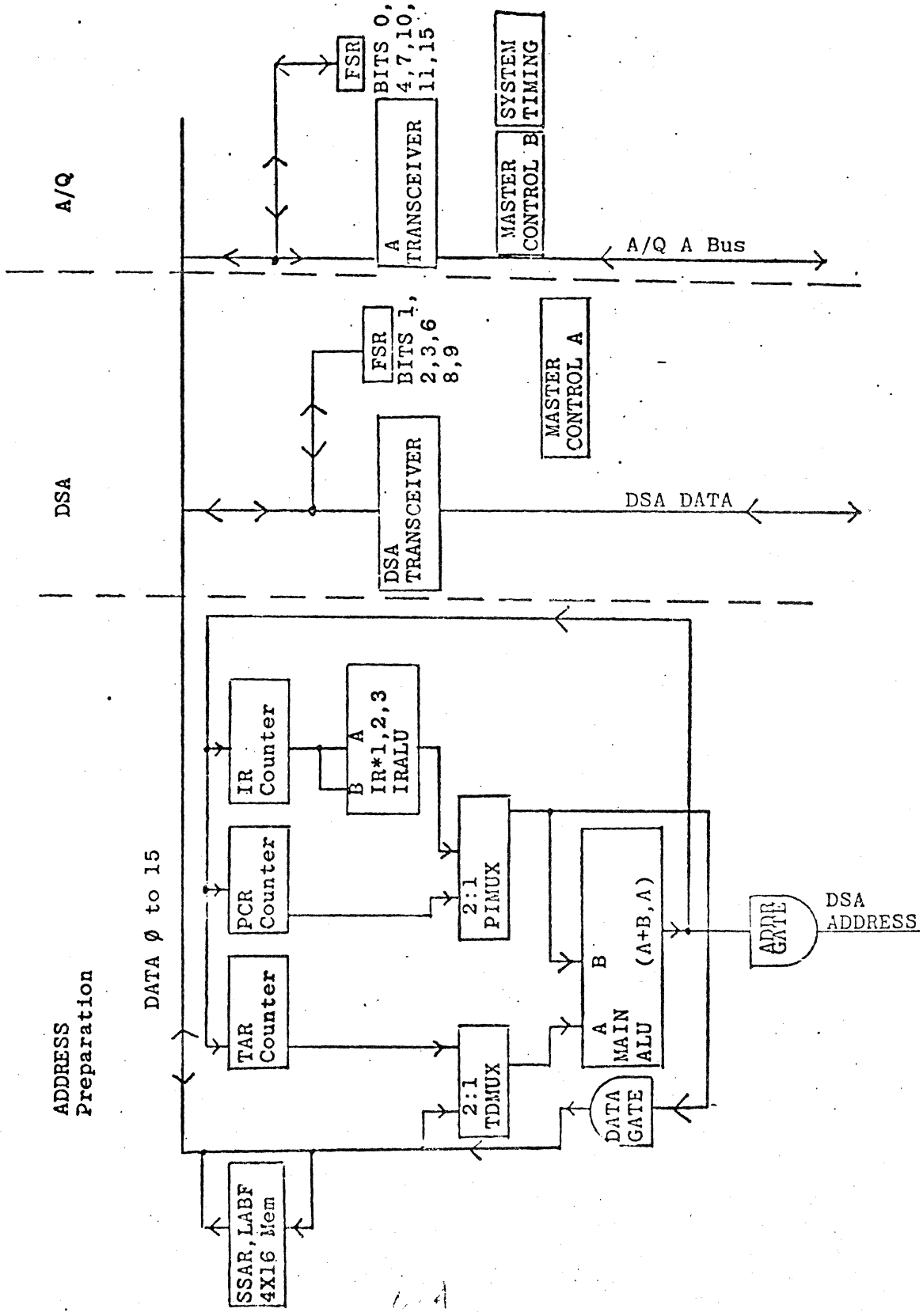


Figure 4.1. HFPU Data Paths

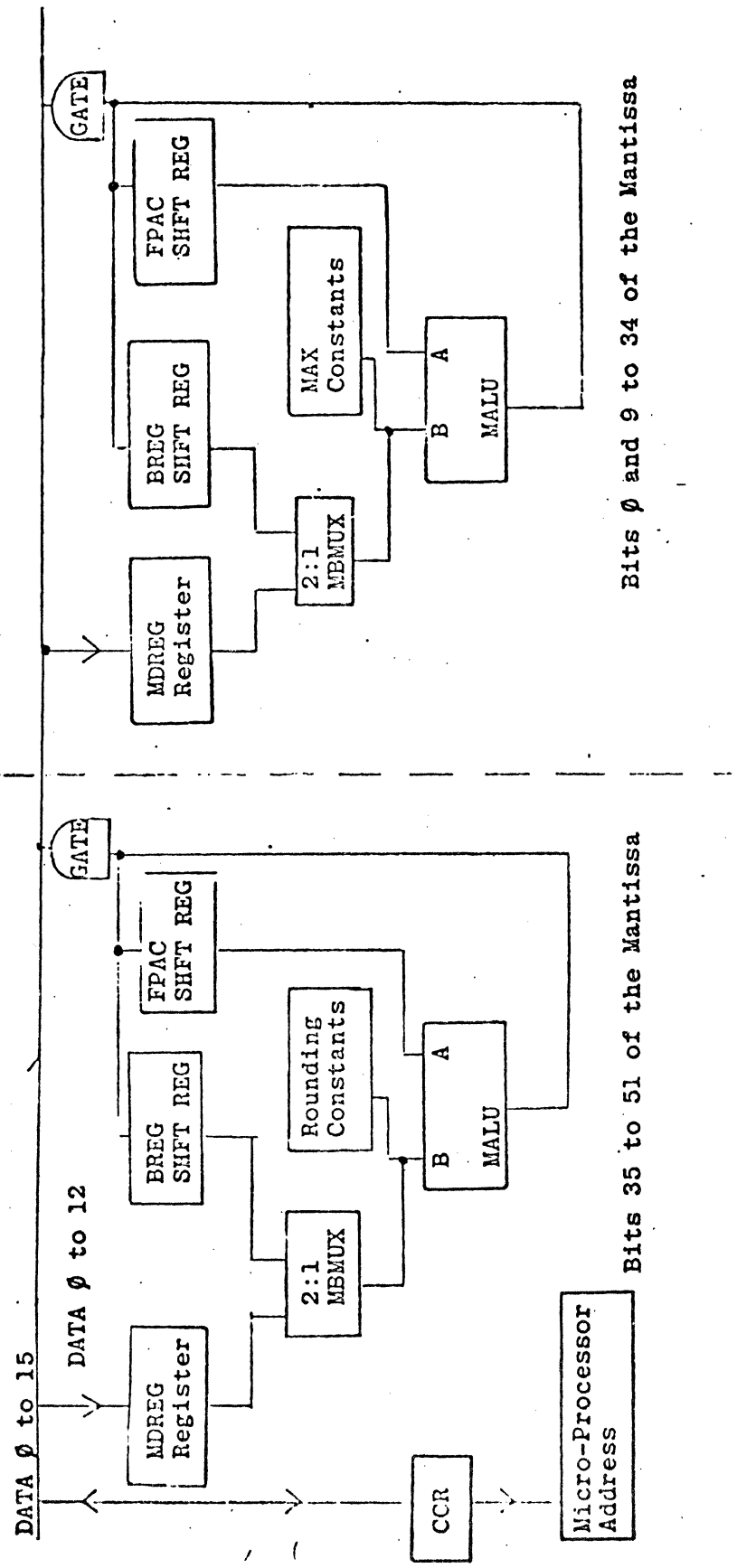


 Figure 4.1. (continued) HFPU Data Paths

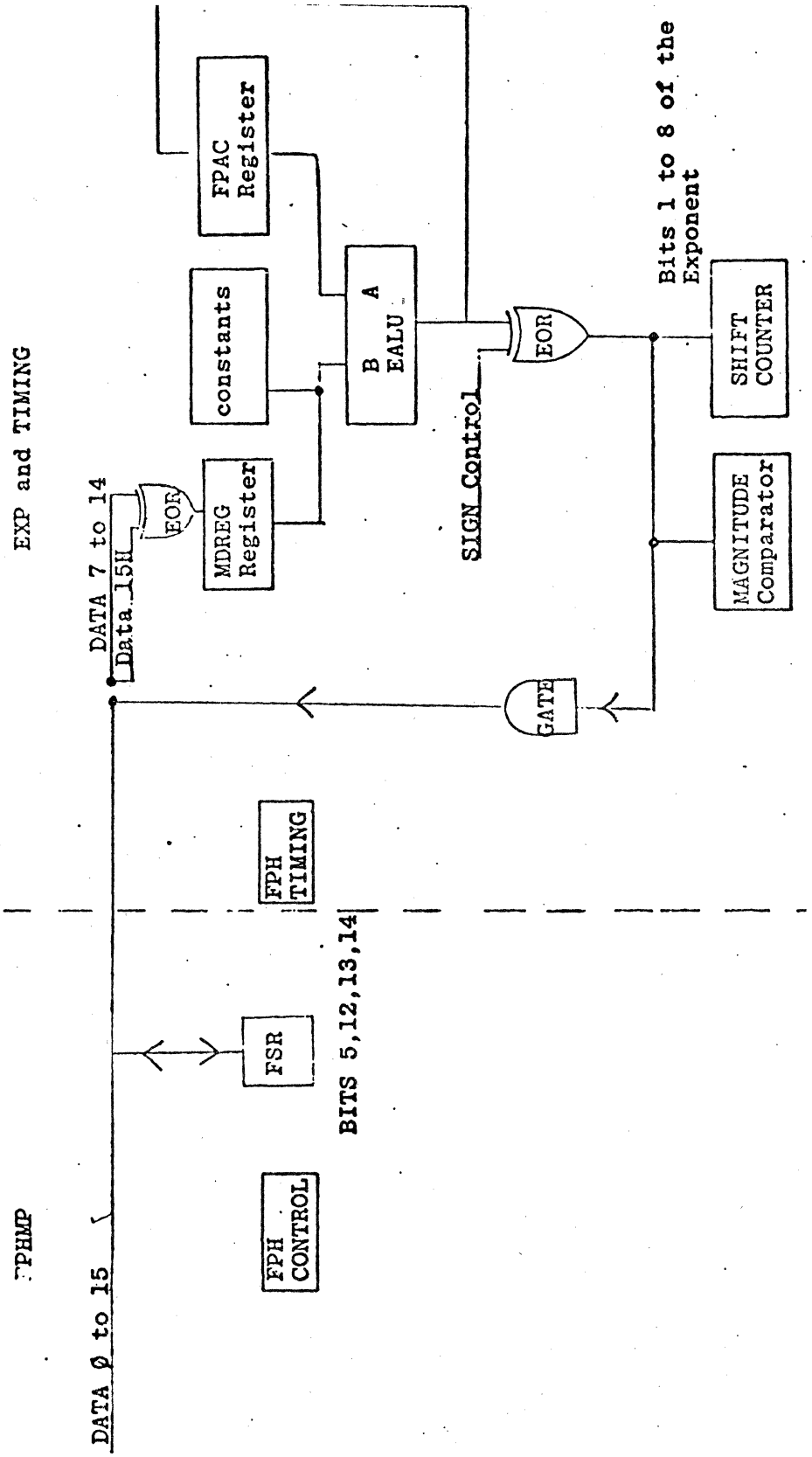


Figure 4.1. (continued) HFPU Data Paths

it with the ability to modify next instruction address based on external conditions. This allows the micro-processor to execute algorithms containing conditional steps.

Figures 4.2 and 4.3 are block diagrams of the two micro-processors in the HFPU. Refer to section 4.1.4 for detailed description of the micro-instruction formats for each of the micro-processors.

The Floating-Point Micro-Processor, shown on figure 4.2, utilizes a Read Only Memory consisting of 32 words of forty bits. The outputs of the Read Only Memory are applied to the inputs of the instruction register. Data is entered into the instruction register on the trailing edge of a clock signal INSCLK. For the Floating-Point Micro-Processor this clock signal has a period of 220 nanoseconds, thus this micro-processor is capable of executing one micro-instruction every 220 nanoseconds. The instruction register helps to speed the operation of the Micro-processor by holding the current micro-instruction while the next instruction is being fetched from the Read Only Memory. The Floating-Point Micro-Processor is started in a two step process by the master processor. When the Floating-Point Micro-Processor is stopped, the Next Instruction Address out of its instruction register is disabled. The Master Micro-Processor then can force the address of the first micro-instruction onto the Next Instruction Address Bus. This allows the first micro-instruction to come out of the Read Only Memory. The Master then forces an INSCLK which loads this instruction into the instruction register and starts the timing of the Floating-Point Micro-Processor running to gen erate its own clock signals to advance it from instruction to instruction. As was mentioned above the outputs of the micro-processor (outputs of the instructure register) fall into two classes. The first class consists of essentially unbuffered outputs which are used to control the gating in the data paths. In the Floating-Point Micro-Processor the main function of these signals is to control the data multiplexers and the function performed by the ALU. The second important class of instruction register outputs consists of clock signals to the various registers within the floating-point arithmetic section. As the diagram shows, these clocks are conditioned by INSCLK so that they occur in coincidence with the entry of new micro-instructions into the instruction register. The phasing of these clocks is arranged so that the entry of the data occurs on the same edge as entry of the new instruction into the instruction register. Thus in effect, each INSCLK enters a new micro-instruction to the instruction register and and completes the execution (by entering data to destination registers) of the preceding micro-instruction.

The Floating-Point Micro-Processor has one additional class of instruction register outputs which are used to control he operation of its hard-wired algorithms. Certain of the operations performed by the Floating-Point Micro-Processor are too fast to be controlled directly by the micro-processor with its cycle time of 220 nanoseconds. These operations, mantissa multiply, divide, shift and normalize, are controlled by the floating-point hardware timing which resides on the EXP and

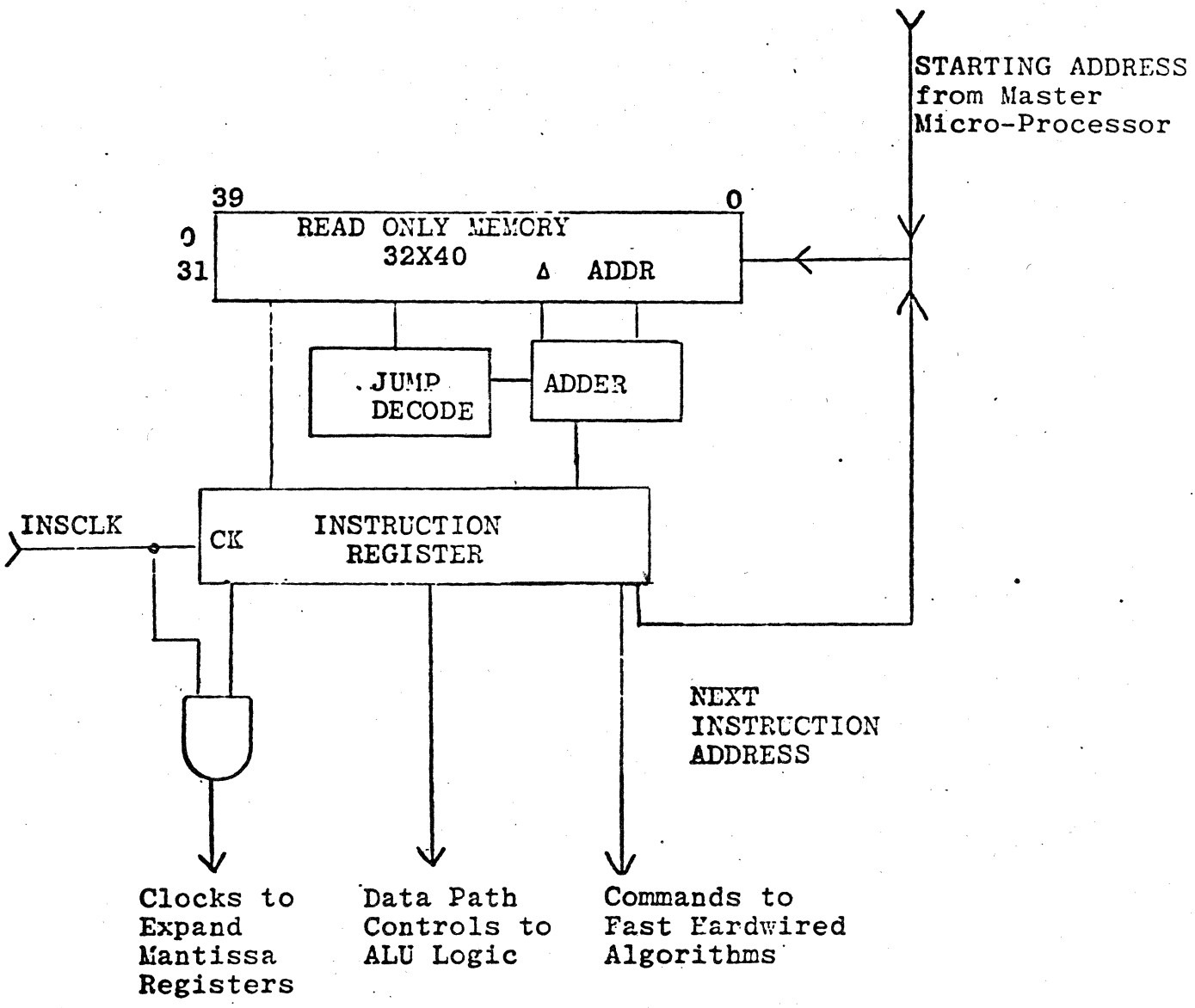


Figure 4.2. Floating Point Micro-processor Block Diagram

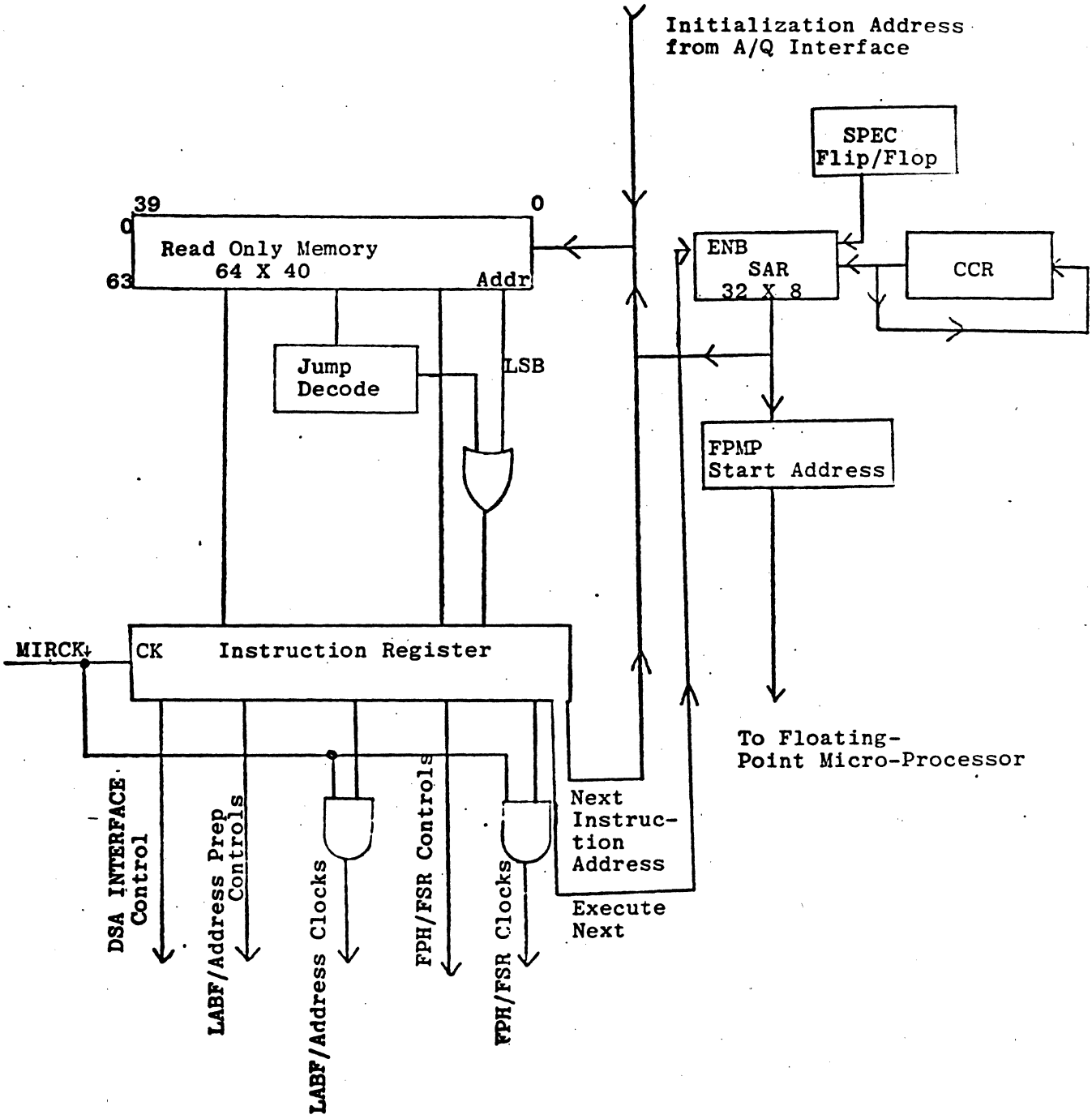


Figure 4.3. Master Micro-processor Block Diagram

TIMING board. When the micro-processor detects a command to one of the hard-wired algorthms, it stops its INSCLK and allows the hardware timing to execute the algorithm to completion. When the hardware timing is finished it restarts the micro-processor INSCLK so that micro-program execution may preceed.

Finally the instruction register contains a HALT bit which is used to stop micro-processor action when the end of the algorithm is reached. When the Floating Point Micro-processor stops its timing, it informs the Master Micro-Processor that it is available to perform a new floating-point function and disables its Next Instruction Address so that the Master Micro-Processor can start it executing another algorithm.

Figure 4.3 is a block diagram of the Master Micro-Processor. This micro-processor is similar in structure to the Floating-Point Micro-Processor. Its ROM consists of 64 words of 40 bits each. Its instruction register clock is called MIRCLK and has a period of approximately 200 nanoseconds. The outputs of the Master Micro-Processor instruction register can also be broken into basically two classes of signals, those which control data paths, and those which clock data into destination registers. The Master Micro-Processor instruction register provides control and clock signals to the DSA interface, the Look Ahead Buffer and Address Preparation ALU, the FSR and CCR and to the Floating-Point input register and output gating. The Next Instruction Address logic of this micro-processor is some what more complicated than that of the Floating-Point Micro-Processor. The next instruction address can come from one of three sources. There is an external starting address source which comes from the A/Q interface and allows the System 17 CPU to start the Master Micro-Processor executing on one of four functions (COLD START, STOP, RESTART and A/Q LOAD FPAC). Secondly there is the normal internal source of next instruction addresses which comes from the ROM. Thirdly, there is a source of next instruction addresses which allows the Master Micro-Processor to interpret the Op-Codes contained in the CURRENT COMMAND REGISTER (CCR). The output of the CCR is applied to the address input of a small ROM. This ROM is referred to as the STARTING ADDRESS ROM (SAR). When the micro-program is ready to begin execution of a Command-Code in the CCR, it turns on the Execute Next bit in its instruction register. This bit disables the next instruction address output of the instruction register and enables the output of the SAR onto the Next Instruction Address Bus. This causes the Master Micro-Processor to begin execution of the micro-instruction sequence corresponding to the new Command-Code. When the SAR is enabled, five bits of the Next Instruction Address Bus are recorded in the FPMP Starting Address Buffer so that they may be used by the Master Micro-Processor to start the Floating Point Micro-Processor running. Thus the starting addresses for both micro-processors for each Command-Code are interlocked, and the micro-programmer must write the micro-code carefully to insure that the two micro-processors will be correctly started. The Master Micro-Processor uses the

Floating Point Micro-Processor starting address to start the Floating Point Micro-Processor running at the appropriate point in Master Micro-Proccessors sequence. As with the Floating Point Micro-Processor, there are several circumstances in which the Master Micro-Processor will stop its MIRCLK in order to wait for completion of some external event. When a DSA memory cycle is requested by the DSA interface control outputs of the Master Micro-Processor instruction register, the Master Micro-Processor timing will stop and wait for the receipt of the DSA RESUME signal. RESUME forces MIRCLK which restarts the micro-processor timing. The Master Micro-Processor will also stop its timing when it is ready to start a new Floating-Point Processor operation and the Floating-Point Micro-Processor is still in the process of executing a preceding command. As with the Floating-Point Micro-Processor the Master Micro-Processor also has a HALT bit. This bit is used to stop Master Micro-Processor execution upon decode of FEND Command-Code and also upon completion of a STOP A/Q command execution.

4.1.3 The programmable elements. Fundamental to the understanding of the operation of a micro-processor is a detailed knowledge of the elements that it controls. This section gives an overview of these elements within the HFPU on a board-by-board basis in order to give the background necessary for the understanding of the detailed description of the micro-instruction set which is follows in section 4.1.4.

a. Address Preparation. This board contains the basic arithmetic for all of the address operations performed by the HFPU. It contains the externally accessable registers, the PCR and the IR. In addition, it also contains a TEMPORARY ADDRESS REGISTER (TAR) which is used for holding the address of memory arguments. The Master Micro-Processor has the ability to load and increment TAR and PCR and to load and clear the IR. The ARITHMETIC LOGIC UNIT (ALU) labeled IR*1,2,3 in figure 4.1 is used to perform multiplication of the index times 1, 2 or 3. The output of the IR is applied directly to the A input of the IRALU and is rotated left one position (multiplied by 2) before being applied to the B input to the IRALU. To multiply the IR by 1, the Master Micro-Processor sets this ALU to gate the A input through to its output. To multiply the IR by 2 the Master Micro-Processor sets this ALU to select the B input to its output. To multiply the IR by 3, the Master Micro-Processor sets this ALU to add the A and B inputs together and apply the result to its outputs. The outputs of the PCR and IRALU are applied to a 2:1 multiplexer called the PIMUX. This multiplexer performs two functions. It is used to select the register to be read, whether PCR or IR, in an A/Q READ operation. Secondly, it selects the register that is to be added to argument address through the main ALU of the address arithmetic section. A second 2:1 multiplexer, the TDMUX, is used to select the source of the input to the A side of the main ALU. To load an absolute address into one of the three registers of the address logic, the TDMUX is set to select the DATA 0 to 15 input and the main ALU is set to gate its A input to its output. To load a relative address into TAR, the TDMUX is set to select the DATA 0 to 15 input, the PIMUX is set to select its PCR input and to apply that to the B input of the main ALU, and the main ALU is set to add its A and B inputs together apply that to its output.

To utilize an absolute or relative address that has been loaded into TAR, the TDMUX is set to select its TAR input, the PIMUX is set to select its IRALU input, the main ALU is set to add its A and B inputs and apply that to its outputs, and the output of the main ALU is driven to the DSA address bus via the ADDR GATE. If the address required is not to be indexed, the main ALU will be set to select its A input. To advance the address through sequential locations, the TAR counter is incremented by the Master Micro-Processor.

The Address Preparation Board also contains the STOP and SAVE ADDRESS REGISTER (SSAR) and the LOOK AHEAD BUFFER (LABF). These registers are contained in a single 4-word by 16-bit memory. The SSAR occupies location 0 in this memory and the portions of the LABF corresponding to FPAC bits 0 to 15, 16 to 31, and 32 to 47 reside in words 1,2 and 3 respectively. The Master Micro-Processor has the ability to read, and write the locations within this memory.

b. DSA BOARD. The elements under the control of the Master Micro-Processor on the DSA board are bits 1,2, 3, 6, 8, and 9 of the FSR and the DSA interface. The micro-processor can load the FSR from DATA 0 to 15 and read the FSR onto the DATA 0 to 15. Additionally, it has the ability to set FSR bit 6, the FEND bit. The micro-processor controls the DSA interface by requesting memory cycles as required and controlling the direction of transfer, whether read or write. Additionally, it has the ability to request consecutive memory cycles and to control the release of the DSA scanner in BLOCK MODE. The DSA interface itself controls the operation of the DSA transceiver which passes data between the DSA data bus and the HFPU internal DATA 0 to 15 lines.

c. A/Q BOARD. On this board the only element under the direct control of the Master Micro-Processor is the FSR (bits 0 4, 7, 10, 11 and 15). The micro-processor has the ability to set bits 4,7 and 15 (DBPM, PROTECT, ACTIVE) and to clear bit 15, the ACTIVE BIT. Additionally, the micro-processor controls the incrementing and clearing of bits 10 and 11, the Operand Byte Count (OPBC).

The A transceiver is under the control of the A/Q interface which also resides on this board. The A/Q interface essentially controls the Master Micro-Processor by supplying it with starting micro-program addresses when A/Q commands which require Master Micro-Processor action are received.

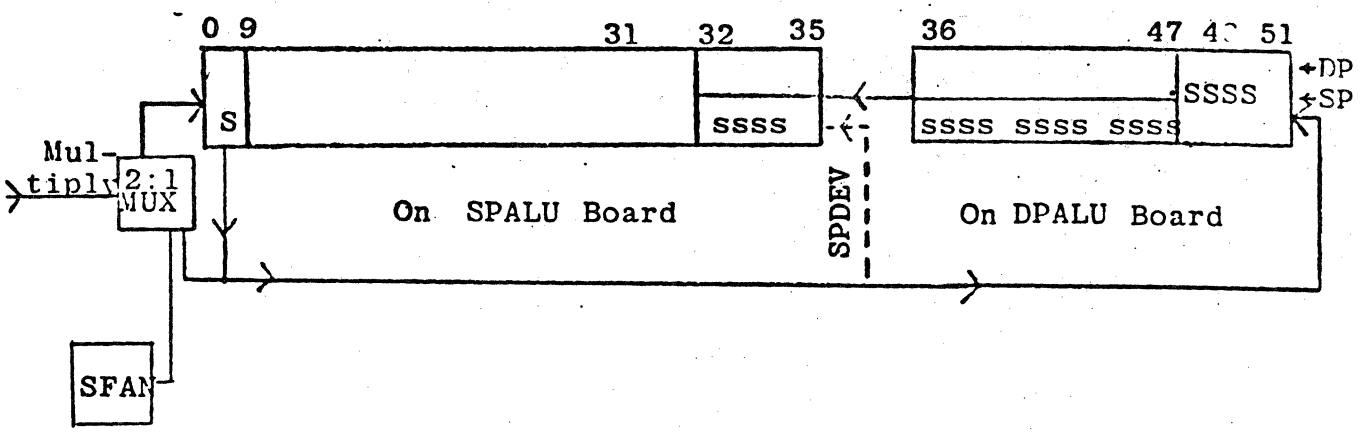
d. DPALU BOARD. The major function of this board is to provide the double precision extension to the mantissa arithmetic for floating-point operations. It does contain the CURRENT COMMAND REGISTER which is under the control of Master Micro-Processor. The Master Micro-Processor has the ability to load this register from DATA 0 to 15, to shift it left by 4 places as each command code is executed, and to read the contents back onto DATA 0 to 15 for transmission back to the System 17 memory in a STOP command.

Note that the structure of the mantissa arithmetic section contained on the DPALU board and the SPALU board are essentially identical.

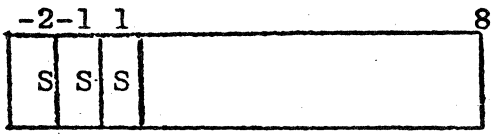
The input to the mantissa arithmetic section is called the MULTIPLICAND/DIVISOR REGISTER (MDREG). This register can be loaded in three sections, corresponding to FPAC bits 0 to 15, 16 to 31, and 32 to 47, by the master microprocessor. The Floating Point Micro-Processor controls the 2:1 multiplexer (MDMUX) to select either the MDREG or the BREG to the B input to the MANTISSA ARITHMETIC LOGIC UNIT (MALU). The FPAC is applied directly to the A input of the MALU. The Floating-Point Micro-Processor has the ability to direct the MALU to perform 8 different functions, A (ARITHMETIC), A-1, A(logical), A complement, A+B, A-B, B, and B complement. The bulk of these functions are self-explanatory with two exceptions. The A (logical) function simply passes the A input of the MALU to its outputs. The A(ARITHMETIC) function checks the A input to the MALU for negative Ø before passing it to the outputs. If the input is negative Ø it will be converted into positive Ø. The FPAC and the BREG are universal shift registers. The Floating-Point Micro-Processor has the ability to load these registers, shift them left, or shift them right. To perform an FLDD function for example, the Master Micro-Processor would load the argument fetched from memory into the MDREG. The Floating Point Micro-Processor would then set the MDMUX to select the MDREG to the B input of the MALU, it would set the MALU to the "B" mode and would load the output of the MALU into the FPAC. To eliminate negative Ø, the Floating-Point Micro-Processor then sets the MALU to the A (arithmetic) mode and again loads the outputs of the MALU into the FPAC. When the Floating-Point Micro-Processor is stopped, the MALU is left in the A (arithmetic) mode: Thus the output of the FPAC is being applied to the GATE which is used by the Master Micro-Processor to read the FPAC, onto the DATA 0 to 15 lines.

e. SPALU BOARD. This board consists almost entirely of mantissa arithmetic logic that is essentially identical in structure to that described above with respect to the DPALU board. The SPALU board contains bits 0 and bits 9 to 34 of the mantissa. Thus it contains the entire single-precision mantissa plus 4 bits of the double-precision extension. When used in single precision, these 4 bits behave as a guard digit. The DPALU board contains the low 12 bits of the double-precision extension of the mantissa plus 4 extra bits of guard digit. Figure 4.4 shows schematically the arrangement of these bits within the mantissa logic. Note that in single precision bits 32 to 51 are loaded with sign bits thus effectively filling them with true 0's (1's complement arithmetic). In double precision only bits 48 to 51 of the mantissa are set equal to the sign. Figure 4.4 also illustrates shift conventions that apply within the mantissa arithmetic section. Note that with one exceptions all right shifting of both of the FPAC and the BREG is arithmetic i.e., the sign bit is shifted from bit 0 to 9 and also into bit 0 on a right shift. The one exception is that during the mantissa multiply portion of FMPY, the output of the special sign holding latch (SFAN) is shifted into Bit Ø of the FPAC. The FPAC is also shifted left arithmetically. It is rotated left with the sign bit going into the least significant bit. In an HFPU that is not equipped with the double-precision option, the sign bit of the FPAC is routed by a jumper into bit position 35 instead of bit position 51. The FPAC is shifted right during the exponent alignment portion of FADD and FSUB

FPAC Mantissa



FPAC Exponent



On EXP & Timing Board

BREG MANTISSA

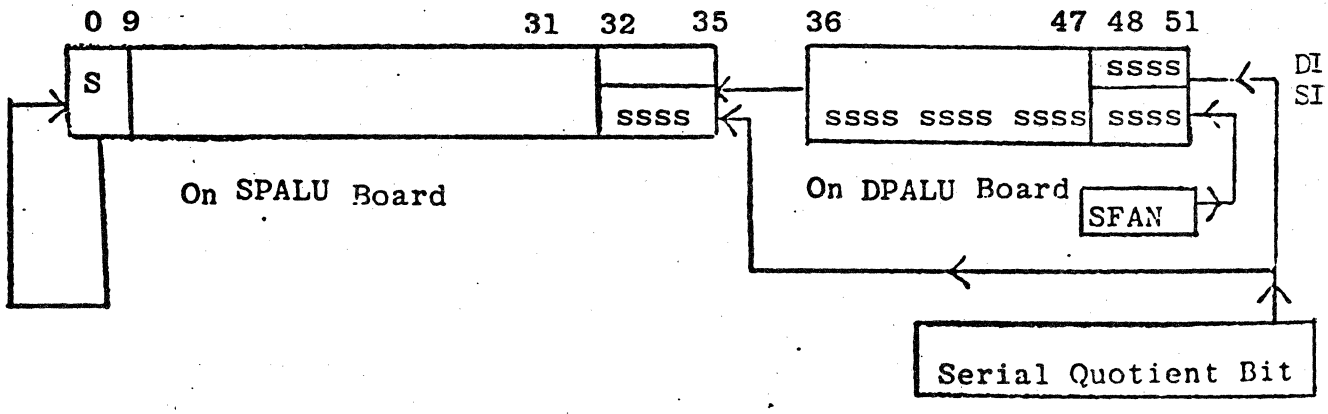


Figure 4.4. Arithmetic Shifting

and during the mantissa multiply portion of FMPY. It is shifted left during normalization and during the mantissa divide portion of FDIV. The FPAC holds the product in multiply and the dividend in divide. The BREG is shifted right during the exponent alignment portions of FADD and FSUB and during the mantissa multiply portion of FMPY where it holds the multiplier. It is shifted left only during the mantissa division portion of FDIV where it is used to assemble the quotient of the result. In double precision, the quotient bits are shifted into the BREG at bit position 51. In single precision they are input at bit position 35 and the output of a special sign holding latch called SFAN is input at bit 51 to insure that a true single-precision result is generated.

One additional controllable feature of the SPALU is the block in figure 4.1 labeled MAX constants. The Floating Point Micro-Processor has the ability to drive several numerical constants to the B input of the MALU. These constants are used in the rounding algorithm and also to force the mantissa of the result in the case of exponent overflow and FLOF overflow.

f. FPHMP BOARD. This contains the Floating Point Micro-Processor which is labeled FPH CONTROL in figure 4.1. The only programmable element on this board is the FSR (bits 5, 12, 13 and 14). The Floating Point Micro-Processor has the ability set bits 12, 13 and 14 (UNFL, DVFL, OVFL) if one of these conditions occurred in the course of a floating-point alculation. The Master Micro-Processor has the ability to read and load the FSR from the DATA 0 to 15 lines.

g. EXP and TIMING. This board contains the basic timing for the Floating Point Micro-Processor and its hard-wired functions. The programmable elements on this board constitute the exponent arithmetic of the HFPU. As in the mantissa ALU sections the MDREG is the input register to the exponent ALU. This register is loaded by the Master Micro-Processor. The Exclusive OR (EOR) gates on the input to the MDREG are used to remove the effects of the mantissa sign on the exponent of the floating-point number. If DATA bit 15 is true, high, then DATA bits 7 to 14 will be inverted before being load into MDREG. If DATA 15 is false, low, then DATA bits 7 to 14 will be applied uninverted to the inputs of MDREG. The output of the MDREG is under the control of the Floating Point Micro-Processor so that either the contents of the MDREG may be applied to the B input of the EALU or if the register is disabled, a 0 can be applied to the B input of the EALU. The FPAC exponent register is applied directly to the A input of the EALU. The EALU can perform a total of 4 functions, A (arithmetic), A-B, A+B, B. The output of the EALU is applied to a second EOR gate which is used to perform two functions. When the Floating Point Micro-Processor is stopped and the master Micro-Processor wishes to read the contents of the FPAC exponent, the sign of the mantissa is applied to this EOR function so that the exponent can be complemented accordingly. For internal exponent operations which require the magnitude of the difference between two exponents, the Floating Point Micro-Processor can use the sign bit out of the EALU to control this EOR function, thus applying the magnitude of the EALU output to the input of the Shift Counter. The Shift Counter is used during

the exponent alignment portions of FADD and FSUB. The Magnitude Comparator is used in the same function and also in FLOF to inhibit shifting when the number of positions to be shifted as represented by the contents of the Shift Counter is larger than the length of the mantissa registers. The box labeled constants in figure 4.1 is used to supply a source of the maximum positive and maximum negative exponent for overflow and underflow and also to supply several exponent values required during FLOF and FIXF.

4.1.4 The Micro-Instruction Set. This section describes in detail the functions performed by the two micro-processors in the HFPU. The instruction format for the Master Micro-Processor appears in Figure 4.5 and the format for the Floating Point Micro-Processor appears in Figure 4.6. These figures display in a schematic form the functions performed by each bit of the READ ONLY MEMORIES of the two micro-processors. Tables 4.1 and 4.2 define in greater detail the mnemonics used in figures 4.5 and 4.6, respectively.

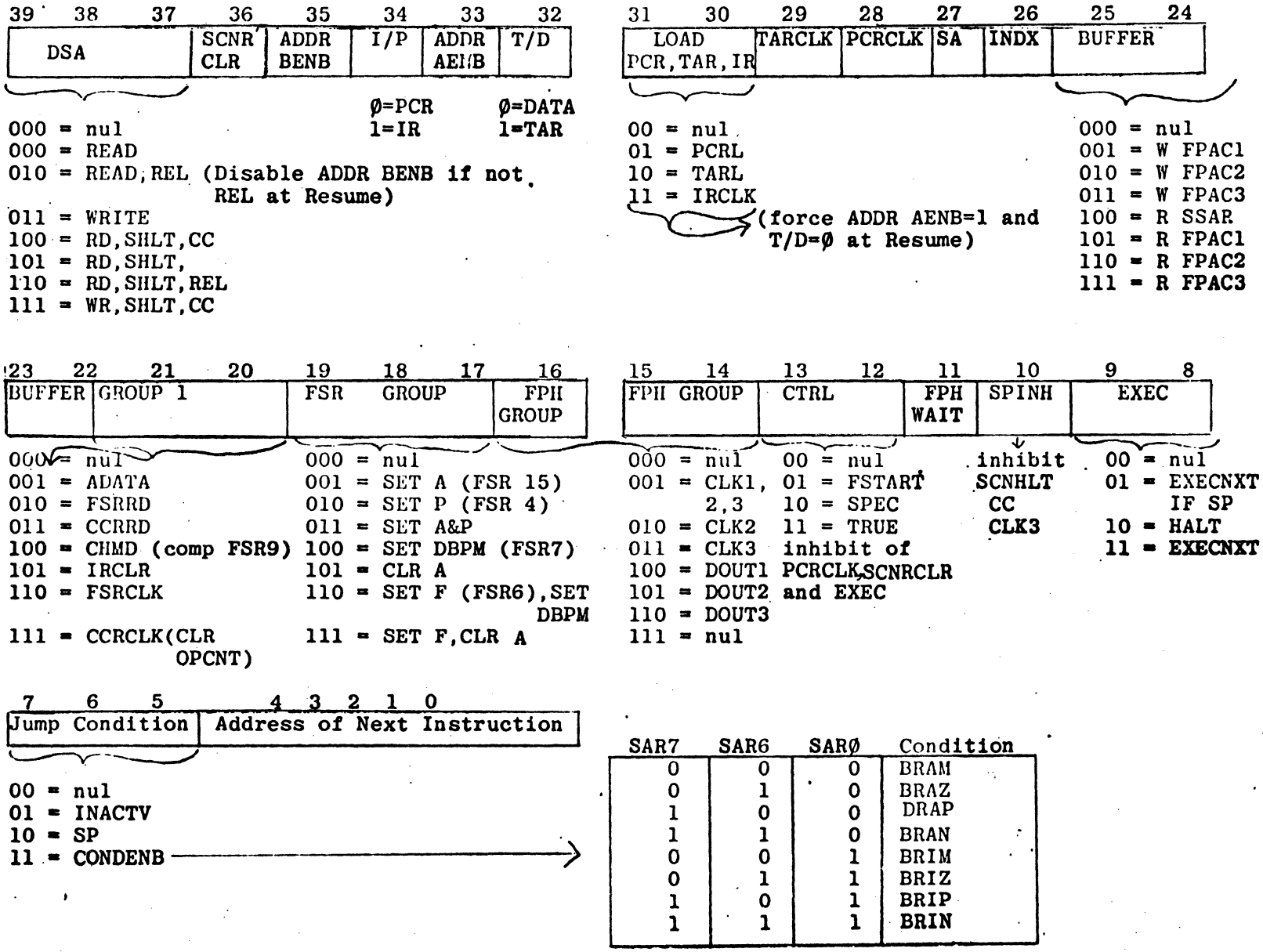


Figure 4.5. Master Micro Processor Instruction Format

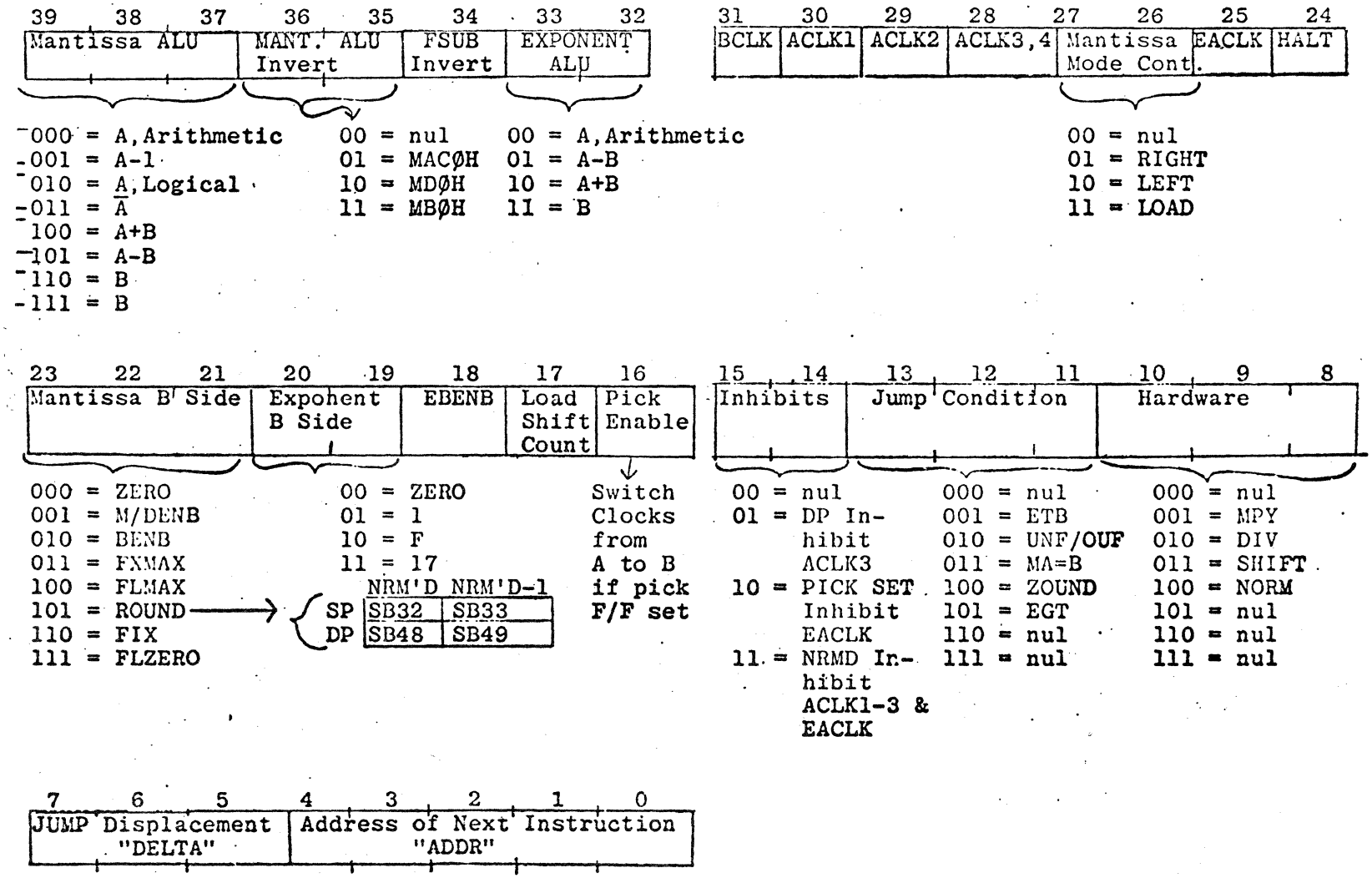


Figure 4.6. Floating-Point Micro Processor Instruction Format

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
39,38,37		DSA	These bits, if not equal to 0, are used to command the DSA interface to perform DSA memory access cycles. Seven different types of cycles can be performed.
	001	READ	Commands DSA Interface to perform a single read-from-memory cycle.
	010	READ,REL	Commands the DSA Interface to perform a read-from-memory cycle and controls the address preparation board to perform the addition of PCR to the incoming address before it is loaded into TAR. To function correctly bits 35,34 (described below) must be a 0 and a 1 respectively, thus selecting the PIMUX to the PCR and enabling its output to the B side of the main ALU on the address preparation board. This micro-instruction code actually functions by disabling bit 35 (ADDR BEND) while the DSA RESUME signal is true if the RELATIVE MODE bit mode in the FSR is not set. Thus, if Relative Mode is not set, the DSA data will pass through the address ALU into TAR. If the RELATIVE Mode bit is set, the DSA data will be added to the contents of the PCR before being loaded into TAR.
	011	WRITE	This code directs the DSA Interface to perform a single memory write cycle to System 17 memory.

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
39.38,37 (Contd)	100	RD,SHLT,CC	This code directs the DSA Interface to perform consecutive memory read cycles. In this mode the DSA Interface will generate a second DSA REQUEST signal upon the receipt of the DSA RESUME signal, thus causing the interface to steal consecutive memory cycles. The CC mnemonic indicates the request for consecutive cycles. The HOST mnemonic indicates that the scanner will remain halted for the duration of of the consecutive cycles.
	101	READ,SHLT	This code requests a DSA read from memory cycle and directs the DSA interface to keep the scanner halted following the cycle.
	110	RD,SHLT, REL	This code requests a DSA read from memory cycle and allows the relative addressing calculations to take place as was described above for code 010 (READ,REL). The scanner remains halted following the memory cycle.
	111	WR,SHLT CC	This code requests consecutive DSA Write memory cycles. The scanner remains halted during the memory cycles.
36		SCNRCLR	A 1 in this bit position directs DSA Interface to release the scanner.
35		ADDR BENB	A 1 in this bit position enables the PIMUX output to the B input of the main ALU. Note that this bit can be disabled during DSA Resume if the code in bits 37, 38, and 39 is 010 (READ, REL) or 110 (RD, SHLT, REL).
34		I/P	This bit drives the select control on the PIMUX. A 0 in this bit causes the PCR to be selected. A 1 in this bit causes the IR to be selected.
33		ADDR AENB	A 1 in this bit position causes the output of the TDMUX to be applied to the A input of the main ALU on the address preparation board.
32		T/D	This drives the select control on the TDMUX. A 0 in this bit selects the DATA 0 to 15 input and a 1 in this bit selects the TAR input.

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	DESCRIPTION	
31,30		LOAD	The two bits in this field are used to select 1 of the three address registers on the address board for loading. If one of the registers is selected, then bit 33 (ADDR AENB) will be forced to a 1 and bit 32 (T/D) will be forced to a 0 during the DSA Resume signal. This combination has the effect of enabling DATA 0 to 15 into the A side of the main ALU thus allowing the information on the DATA bus to pass through the ALU to the selected destination register.
	01	PCRL	This code enables the PCR load control.
	01	TARL	Tar Load enable.
	11	IRCLK	Load the IR. Causes a clock signal to the IR.
29		TARCLK	A 1 in this bit causes a clock signal to be sent to TAR during MIRCLK. If bits 31 and 30 are not equal to 10 then this clock will cause TAR to be incremented. If bits 31 and 30 are equal to 10 then this clock will cause TAR to be loaded from the output of the main ALU on the address board.
28		PCRCLK	A 1 in this bit position causes a clock to be sent to the PCR. If bits 31 and 30 are not equal to 01 then PCR will be incremented. If bits 31 and 30 are equal to 01 then the PCR will be loaded from the output of the main ALU.
27		SA	Select A . A 1 in this bit causes the main ALU on the address board to select its A input for presentation to its output. A 0 in this bit directs the main ALU to add its A and B inputs together for presentation to its output. This bit allows data on the A side of the ALU to pass through to the inputs to the registers or to the DSA Address bus without regard to the data that may be present on the B input to the ALU.

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITION	VALUE	MNEMONIC	DESCRIPTION
26		INDX	A 1 in this bit enables the multiplication of the IR by 2 in single precision or by 3 in double precision. This is performed by setting the IRALU to gate its B input to its output in single precision and by setting it to add its A and B inputs together in double precision. If the INDX bit is equal to 0 then the IRALU is set to select the A input, thus passing the IR through without multiplication.
25,24,23		BUFFER	Codes on these three bits are used to read and write the locations within the 4-word by 16-bit memory (the Look Ahead Buffer and the SSAR) on the address board.
	001	WFPAC1	Write the contents of DATA 0 to 15 into word number 1 of the memory, the portion of the LABF that corresponds to FPAC bits 0 to 15.
	010	WFPAC2	Write into word 2 of the memory, the portion of the LABF that correspond to FPAC bits 16 to 31.
	011	WFPAC3	Write into memory word 3, the portion of the LABF that correspond to FPAC bits 32 to 47.
	100	RSSAR	Read word 0 of the memory, the SSAR, onto DATA 0 to 15.
	101	RFPAC1	Read word 1 of memory, LABF bits 0 to 15.
	110	RFPAC2	Read word 2 of the memory, LABF bits 15 to 31.
	111	RFPAC3	Read word 3 of the memory, LABF bits 32 to 47.
22,21,20		GROUP 1	
	001	ADATA	Enables the output of the PIMUX on the address board onto DATA 0 to 15. This code is used for storing IR and PCR during a STOP operation.
	010	FSRRD	FSR READ . Read the contents of FSR onto DATA 0 to 15.

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
22, 21, 20 (Contd)	011	CCRRD	Read the contents of the CCR onto DATA 0 to 15.
	100	CHMD	This code is used in the execution of the CHMD command code. It complements FSR bit 9, the Relative Mode bit.
	101	IRCLR	This code is used in the execution of the NIDX command code. It clears the IR.
	110	FSRCLK	Load the FSR from DATA 0 to 15.
	111	CCRCLK	Load the CCR from DATA 0 to 15 and clear the operand byte count, OPBC, bits 10 and 11 of the FSR.
19,18,17		FSR GROUP	These codes are used to set and clear selected bits in the FSR.
	001	SET A	Set the Active bit, FSR bit 15.
	010	SET P	Set FSR bit 4, the Protect Mode bit.
	011	SET A&P	Set the Active and the Protect bits in the FSR.
	100	SET DBPM	Set the double-precision mode bit in the FSR, bit 7.
	101	CLR A	Clear the active bit in the FSR.
	110	SET F,SET DBPM	Set the FEND bit and the DBPM bit in the FSR.
	111	SET F,CLR A	Set the FEND bit and clear Active bit in the FSR.
16,15,14		FPH GROUP	The codes in this group are used to load the input register to the Floating Point Hardware portion of the HFPU and to read the output of the FPAC.
	011	CLK 1,2,3	Load the contents of DATA 0 to 15 into all three sections of the floating-point input register, the MDREG, simultaneously. This code is used to load the high word out of the Look Ahead Buffer into the high and middle word of the MDREG and to load the sign of this word, the sign of the floating-point number, into the low word of the MDREG (sign-extension) if in single-precision mode.

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITION	VALUE	MNEMONIC	DESCRIPTION
16,15,14 (Contd)	011	CLK 3	Load the low word of the MDREG.
	100	DOUT1	Read the contents of the high word of the FPAC (bits 0 to 15) onto DATA 0 to 15.
	101	DOUT2	Read the middle word of the FPAC.
	110	DOUT3	Read the low word of the FPAC.
	111	NUL	Undefined.
13,12		CTRL	The codes in this field are used for special micro-processor control functions.
	01	FSTART	This code is used to start the Floating Point Micro-Processor running. The starting address for the Floating Point Micro-Processor was saved in the FPMP Start Address Register on the DPALU board at the time when the Master Micro-Processor began execution of the current Command-Code.
	10	SPEC	This bit is used in the execution of the SPEC Command-Code. It sets the SPEC Flip/Flop that appears in figure 4.3. The output of this flip/flop drives the most significant bit of the input address to the SAR. This causes the starting address for the next Command-Code to come from locations 16 to 31 within the SAR. The SPEC Flip/Flop is cleared automatically by the execution of the next Command-Code.
	11	TRUE	If the jump condition was specified by bits 6 and 7 is true, the action of the following micro-proccesor output bits will be inhibited; bit 36, SCNRCLR; bit 28, PCRCLK; bits 9 and 8, EXEC. This bit is used in the execution of the branch Command-Codes to allow the increment of PCR, the release of the scanner and the execution of the next Command-Code if the branch condition is false. It is false used to allow the micro-processor to jump to the code that executes a CACS if the jump condition is true.

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
11		FPH WAIT	If this bit is set the Master Micro-Processor will stop the execution of micro-instructions to wait for the Floating Point Micro-Processor to complete its execution. This bit is used whenever the Master Micro-Processor needs to start the Floating Point Micro-Processor running or when it needs the result of a Floating Point Micro-Processor operation.
10		SPINH	If this bit is set and the FHPU is in single-precision mode then code 011 in bits 16,15 and 14 (CLK3), and the scanner halt and consecutive cycle portions of bits 39,38 and 37, will be inhibited. This bit allows the same micro instruction, the one that fetches the second word of the argument or the one that loads the third word of the argument into the MDREG, to be used in either single or double precision.
9,8		EXEC	The codes in this field are used for executing the next Command-Code and for stopping micro-processor action.
	01	EXEC NXT IF SP	Execute Next Command-Code if the HFPU is in Single Precision mode. The Execute Next function of the Master Micro-Processor needs some discussion. When the Execute Next function comes true, the Master Micro-Processor inhibits the next instruction address output of its instruction register and enables the output of the Starting Address ROM (SAR). The SAR is a ROM that contains 32 words of 8-bits each. The least significant four bits of the input address to this ROM are the actual Command-Code that is to be executed. The most significant bit of the input address comes from the SPEC Flip/Flop. The SAR translates the Current Command-Code into a starting ROM address for the Master Micro-Processor. The three least-significant bits out of the SAR are concatenated with the two most-significant bits out of the SAR and loaded into the Floating Point Micro-Processor Starting Address Register to form a start-

BIT POSITION	VALUE	MNEMONIC	DESCRIPTION
9,8 (Contd)	01	EXEC NXT IF SP	ing address for the Floating Point Micro-Processor which can be used at a later time by the Master Micro-Processor. This technique allows the starting address for the next micro-instruction sequence be applied to the ROM while the last instruction of the current sequence is completing execution. Thus no micro-processor overhead is incurred in the process of changing from one micro-instruction sequence to the next.
	10	HALT	Master micro-processor halt. Upon completion of the execution of the current micro-instruction, the master micro-processor clock is stopped. This code is used to stop the micro-processor after the detection of a FEND Command-Code and at the completion of the STOP sequence.
	11	EXEC NEXT	Unconditional Execute Next function. See EXEC NXT IF SP above.
7,6		JUMP CONDITION	The codes in this group specify the type of condition that is to be tested for a micro-processor skip. If the condition is found to be true, the least-significant bit of the next instruction address will be forced to 1, thus causing a skip if the next instruction address is even. If the jump condition is false the next instruction address will not be modified.
	01	INACTV	In the execution of a RESTART A/Q Command this jump condition is used to test the state of the FSR that was fetched from memory. It is used to cause the micro-processor to execute the next sequential micro-instruction which is a HALT instruction instead of proceeding to execute the next Command-Code sequence.
	10	SP	This command code causes a skip if the HFPU is in single-precision mode.
	11	COND ENB	This jump condition is used in the execution of the BRANCH Command-Codes. The actual condition to be tested is determined by bits 7,6 and 0 of the micro-processor starting address of the current Command-Code sequence as saved in the FPMP Starting Address Register. The table on figure 4.5 illustrates the relationship between these bits and the condition being tested.

TABLE 4.1. MASTER MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
5,4,3,2, 1,0		ADDR	These last six bits of the Master Micro-Processor instruction contain the address -f the next instruction to be executed. As was described above this address can be modified in two ways. If the jump condition is true, then the least-significant bit of this address, bit 0, will be forced true. If the execute next Command Code field is true, this address will be ignored and will be replaced by the output of the Starting Address ROM.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
39,38,37		Mantissa ALU	The codes in the field are used to control the mode of operation of the Arithmetic Logic Units on the DPALU and the SPALU boards.
	000	A,Arithmetic	The ALU passes its A input to its outputs in the arithmetic mode. Negative 0 will be converted to positive 0.
	001	A-1	The A input minus the least-significant bit of the guard digit is transferred to the output in arithmetic mode.
	010	A,Logical	The A input is transferred to the output in logical mode. Negative 0 is left as negative 0.
	011	$\bar{A}$	The complement of the A input is transferred to the output in logical mode.
	100	A+B	The arithmetic sum of the A input and B input is transferred to the output.
	101	A-B	The arithmetic difference of A and B is transferred to the output.
	110	B	The B input is transferred to the output in logical mode.
	111	$\bar{B}$	The complement of the B input is transferred to the output.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
36,35		Mant. ALU Invert	The codes in this field are used to perform various types of conditional negation of the operand passing through the mantissa ALU. Note that in figure 4.6 the codes in the Mantissa ALU field are grouped into pairs with small brackets on the left hand margin. The codes in this the Mantissa ALU Invert field are used to modify the Mantissa ALU field depending upon some external condition. They have the ability to switch the Mantissa ALU field between the pairs of codes within the brackets in figure 4.6. Thus, for example, if the Mantissa ALU field specifies code 100, A+B, and the condition specified by the Mantissa ALU Invert field is true then the actual micro-processor output will correspond to code 101, A-B. If the code specified in the Mantissa ALU field were a 101, then the Mantissa ALU Invert field could switch it to a 100 code, A+B. An examination of the codes the Mantissa ALU field will show that this interchange of codes is caused simply by inverting micro-processor bit 37.
	01	MACOH	Invert micro-processor bit 37 if mantissa accumulator bit 0 is true i.e., if the contents of FPAC is a negative number.
	10	MDOH	If the Hardware field (Bits 10,9,8) is an MPY, then micro-processor bit 37 will be inverted if the sign of the FPAC and that of the MDREG are different. If the Hardware field is DIV, then micro-processor bit 37 will be inverted if the sign bit of the MDREG is set.
	11	MBOH	Invert micro-processor bit 37 if the sign bit of the BREG is set.
34		FSUB Invert	Invert micro-processor bit 37 if the current micro-code sequence is that for Command-Code FSUB.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
33,32		Exponent ALU	The codes in this field determine the function performed by the Arithmetic Logic Units in the exponent arithmetic section.
	00	A,Arithmetic	The EALU passes its A input to its output in arithmetic mode. Negative 0 is converted to positive 0.
	01	A-B	The B input is subtracted from the A input and passed to the output of the EALU.
	10	A+B	The sum of the A and B inputs is passed to the output of the EALU.
	11	B	The B input is passed to output of the EALU.
31		BCLK	If this bit is a 1 a clock signal will be sent to the BREG on the INSCLK that enters the next micro-instruction.
30		ACLK1	Send a clock to FPAC bits 0 and 9 to 15.
29		ACLK2	Send a clock to FPAC bits 16 to 31.
28		ACLK3,4	Send a clock to FPAC bits 32 to 47 and bits 48 to 51.
27,26		Mantissa Mode Control	The codes in this field control the type of operation performed by the shift registers which constitute the BREG and FPAC.
	01	RIGHT	Select the shift registers to the shift-right mode.
	10	LEFT	Select the shift registers to the shift-left mode.
	11	LOAD	Select the shift registers to the Load mode. In this mode, if a clock specified in bits 28 to 31, the register will be loaded from the output of the ALU.
25		EACLK	Send a clock to Bits -2,-1 and bits 1 to 8 of the FPAC, the exponent.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
24		HALT	This is the stop control for the Floating Point Micro-Processor. If this bit is set the micro-processor will stop action on completion of the current micro-instruction and drop its busy signal to the Master Micro-Processor.
23,22,21		Mantissa B Side	The codes in this field determine the source of data to be applied to the B input to the Mantissa ALU and in some cases also to the B input of the Exponent ALU.
	000	ZERO	All zero's are applied to the B input of the Mantissa ALU (bits 0 and 9 to 51).
	001	M/DENB	The MDMUX is enabled and the MDREG input is selected.
	010	BENB	The MDMUX is enabled and the BREG input is selected.
	011	FXMAX	The maximum negative integer (8000_{16} is applied to bits 16 to 31. This constant is used for forcing the maximum integer result in the FLOF function if the floating-point number was to large to represent as a 16-bit integer.
	100	FLMAX	A constant of 127_{10}, FF_{16} is applied to the B side of the Exponent ALU and the maximum negative mantissa value, bit 0 = 1 and bits 9 to 51 = 0, is applied to the B side of Mantissa ALU.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
23,22,21 (Contd)	101	ROUND	This code is used to effect the rounding of floating-point results. The truth table in figure 6.5 shows the input bit to the Mantissa ALU that will be driven in single precision and double precison depending on whether the mantissa is normalized (NRM'D) or one position short of being normalized (NRM'D-1). This bit is effectively the most-significant bit of the true guard digit. If the number to be rounded is positive, the selected bit will be added to it. If the number to be rounded is negative, the selected bit will be subtracted from it.
	110	FIX	This code is used to control the comparison value input to the Magnitude Comparitor shown in figure 4.1 on the EXP and TIMING board. This special comparison value of 22_{10} is used to avoid excessive shifting of numbers which are smaller in magnitude than 1.0 and to result in their being correctly converted to integer 0.
	111	FLZERO	This code is used to force a true floating-point zero result. It applies an exponent value of -127_{10} to the B side of the EALU and a value of 0 to the B side of the EALU.
20,19		Exponent B Side	The codes in this field are used to apply selected constants to the B side of the EALU.
	00	ZERO	Zero's are applied to the B input to the EALU.
	01	1	A value of 1 corresponding to a 1 in bit position 8 of the FPAC is applied to the B side of the EALU. This constant is used to increment the contents of the FPAC exponent.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
20,19 (Contd)	01	1	A value of 1 corresponding to a 1 in bit position 8 of the FPAC is applied to the B side of the EALU. This constant is used to increment the contents of the FPAC exponent.
	10	F	A value of 15_{10} is applied to the B side of the EALU. This constant is used as a comparison value to check to see if a floating-point number is too large to be converted to a integer in the FLOF function.
	11	17	A value of 23_{10} is applied to the B side of the EALU. This value is used to generate the shift count in the FLOF function.
18		EBENB	This bit, if a 1, enables the output of the MDREG to the B side of the EALU.
17		Load Shift Count	This bit is used to load the magnitude of the output of the EALU into the shift counter. This bit causes the Sign Control for the EOR function on the outputs of the EALU to be driven from the sign (bit -2) output of the EALU so that the EALU outputs will be inverted if negative. Additionally, this micro-processor function loads the sign of the EALU output into the PICK flip/flop. The PICK F/F will be set if the EALU sign is positive and it will clear if the EALU sign is negative.
16		PICK Enable	If the PICK F/F is set, bits 30 to 28 (ACLK1,ACLK2, ACLK3,4) will be inhibited and bit 31 (BCLK) will be forced.
15,14		Inhibits	The codes in this field are used to inhibit other micro-processor instruction fields in the presence of certain conditions.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
15,14 (Contd)	01	DPInhibit ACLK3	The ACLK3 portion of micro-processor bit 28 will be inhibited if the HFPU is in Double Precision mode. This feature is used during the truncation of the guard digi portion of the results.
	10	PICK SET Inhibit EACLK	Micro-processor bit 25, EACLK will be inhibited if the PICK F/F is set. This bit is used in the selection of the larger exponent during floating add.
	11	NRMDInhibit ACLK1-3 & EACLK	Inhibits micro-processor bits 30, 29, 28 and 25 if the argument in the FPAC is normalized.
13,12,11		Jump Condition	The codes in this field are used to test for certain conditions which will dynamically modify the micro-processor sequence.
	000	nul	No jump. Do not modify the address of the next instruction that appears in bits 0 to 4.
	001	ETB	Exponent Too Big . This condition tests the output of the Magnitude Comparator on the EXP TIMING board. If the value being loaded into the Shift Counter is larger than the comparison value, then the jump will take place, i.e., the jump displacement in bits 5 to 7 of the micro instruction will be added to the ADDR field, bits 0 to 4.
	010	UNF/OVF	If the Exponent overflow condition is true, the next instruction address is ADDR plus DFLTA. If Exponent Underflow is true, the next instruction address is ADDR plus DELTA + 1. If neither condition is true, the next instruction address is ADDR.
	011	MA=B	Jump, address of next instruction is ADDR + Delta, if the output of the MALU is equal to 0.

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
13,12,11 (Contd)	100	ZOUND	Jump if the divisor is 0 or un-normalized.
	101	EGT	Exponent Greater Than 0 . Jump if the FPAC Exponent is greater than 0.
	110	nul	unused.
	111	nul	unused.
10,9,8		Hardware	The codes in this field are used to call up the high-speed, hard-wired algorithms on the EXP and TIMING board. The next instruction in sequence will not execute until the hard wired algorithm has completed its function.
	001	MPY	Multiply. Multiplicand in MDREG. Multiplier in DREG. MALU set to ADD/SUB MDREG to/from FPAC.
	010	DIV	Divide. Dividend in FPAC. Divisor in MDREG. MALU set to SUB/ADD MDREG from/to FPAC. Quotient goes to BREG.
	011	SHIFT	If the PICK F/F is set, the BREG will be shifted right a number of places equal to the count in the Shift Counter. If the PICK F/F is clear, the FPAC mantissa register will be shifted right.
	100	NORM	Normalize. If the FPAC is normalized, or is 1 bit-position short of being normalized, this instruction does nothing. If the FPAC is more than one position un-normalized, then it will be shifted left and the FPAC exponent decremented until it is one bit-position short of being normalized. The Exponent ALU field and the Exponent B Side must be set correctly to result in the exponent decrement, since the normalize hardware merely generates an appropriate number of FPAC exponent clocks (EACLK).

TABLE 4.2. FLOATING POINT MICRO-PROCESSOR INSTRUCTION FORMAT (Contd)

BIT POSITIONS	VALUE	MNEMONIC	DESCRIPTION
7, 6, 5		DELTA	Jump Displacement. DELTA will be added to ADDR if the Jump Condition is true. The addition is done modulo 16, thus the most significant bit of ADDR will not change in a jump.
4,3,2,1,0		ADDR	Address of next instruction.

4.2 DESCRIPTION OF ALGORITHMS

4.2.1 Introduction to Flowcharts and Listings. The algorithms used in the HFPU to perform its various functions are described in 4.2.2 . The descriptions of that section are all keyed to the micro-code listings and flow charts that reside in appendix B. Figure 4.7 summarizes the terminology and diagramatic conventions used in the flow charts of appendix A. The only unusual characteristic of these flow charts are the brackets that appear to the left. These brackets encompass groups of flow charts operations that correspond to the manipulations performed by individual micro-instructions in the micro-code listing. A step number (e.g., STEP#1) and the micro-code sequence number (e.g.,LOC 10) is indicated to the left of the bracket in the flow chart. Algebraic operations are indicated in an ALGOL-like manner. Figure 4.7 also provides a short glossary which defines the mnemonics used in the flow charts to reference the various elements of the HFPU.

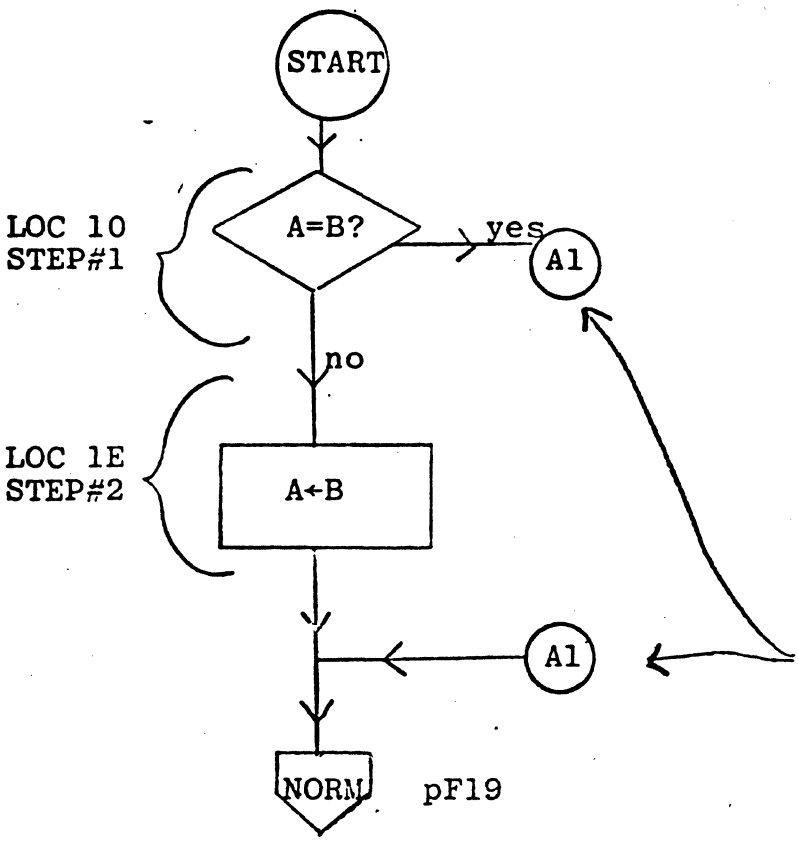
4.2.2 The Algorithms.

4.2.2.1 OP-CODE FETCH/COLD START. Master Control flow charts page M17.

Step 1. Micro-code location 3. The same micro-code is used for both the fetch of a new Command-Code word and for the start-up on an A/Q Cold Start command. This micro-instruction sets the Active bit in the FSR and initiates a DSA request for a memory-read cycle with the PCR as the memory address. The DSA Data-out is loaded into the CCR and the PCR is incremented on the trailing edge of RESUME. The Operand Byte Count (OPBC) is set to 0 to point to the first Command-Code in the CCR. The micro-instruction sequence will return to this point when the OPBC reaches 4.

Step 2. Micro-Code location E. This instruction contains an unconditional Execute Next which causes the microprocessor to branch to the first instruction of the sequence corresponding to the first OP-Code in the CCR. Page M26 of the flow charts illustrates the decisions that will be made in the course of the Execute Next operation. If the OPBC is equal to 4, then the micro-code will branch back to location 3 to fetch the next OP WORD. If the SPEC F/F is set, the microprocessor will branch to the first instruction of the next OP-CODE. If neither of these conditions is true, the logic then checks to see if an A/Q STOP command has been issued. If so, it branches to the first micro-instruction of the STOP sequence. If not, it branches to the first instruction corresponding to the nest OP-CODE.

4.2.2.2 The SPEC GROUP. M. C. Flow chart pages M15 and M16. Execution of the SPEC Command-Code is accomplished entirely by single micro-instruction at location 32 which sets the SPEC F/F. Once this flip/flop has been set, the next Command-Code sequence will come from the upper 16 locations in the SAR.



A ← B	Replace A with B
A ← $\overline{B}$	Replace A with B complemented
|B|	Absolute value
↑	Low to High transition of a logic signal
EACC	Exponent portion of FPAC
MACC	Mantissa portion of FPAC
FPAC	Floating Point Accumulator
MACnn	Bit nn of MACC
MARG	Memory Argument mantissa
EARG	Exponent of Memory Argument
BREG	Mantissa B Register
PICK	Single Bit register used to save sign of exponent difference
MBnn	Bit nn of B register
SAN/SFAN	Sign of Answer
EOR	Exclusive or
DEC	Decrement

Figure 4.7. Flow Chart Conventions (Sheet 1 of 2)

ADDR	Address driven to DSA address Bus
TAR	Temporary Address Register
PCR	Programs Counter Register
IR	Index Register
DBPM-	Double Precision Mode Bit in FSR
BUFØ to 15	First word of Look-Ahead-Buffer
BUF 16 to 31	Second word of Look-Ahead-Buffer
BUF 32 to 47	Third Word of Look-Ahead-Buffer
MDØ to 51	Multiplicand/Divisor Register. Input register to Floating Point ALU. Used to hold Memory Argument (MARG).
CCR	Current Command Register
DPBC	Operand Byte Count.
FEND	FEND Bit in FSR
ACTIVE	Active Bit in FSR
PROT	Protect Bit in FSR
SSAR	Stop and save address register
AND	Logial and Function
IADD	Micro-Processor Instruction Address applied to ROM

Figure 4.7. Flow Chart Conventions (Sheet 2 of 2)

4.2.2.2.1 CACS. The execution of a CACS consists entirely of the micro-instruction at location 35. This instruction initiates a DSA READ cycle with the PCR as the memory address. If the HFPU is not in Relative Mode, then the DSA data is loaded into the PCR. If the HFPU is in Relative Mode, then the DSA data is added to the old contents of the PCR and the result placed into the PCR.

4.2.2.2.2 BRAZ,BRAN,BRAP,BRAM. These Command-Codes require two or three micro-instructions depending on the state of the condition being tested. Bits 7,6 and 0 of the Starting Address as given in the Flow chart Index of appendix A determine the condition to be tested. Refer to Fig. 4.5 for a description of the relationship between these bits and the tested condition.

Step 1. Location 36. This instruction performs an FP WAIT to allow the Floating Point Micro-Processor (FPMP) to complete its operation so that the data in the FPAC will be valid.

Step 2. Location 37. This instruction tests the specified condition. If it is true, then the next instruction to be executed is the CACS instruction at location 35. If the condition is false, the program counter is incremented once to advance it past the Address Word and execution proceeds with the next sequencial Command-Code.

4.2.2.2.3 BRIZ,BRIN,BRIP,BRIM.

Step 1. Location 33. This instruction is simply a no-op to allow the data in the Index Register to settle so that the next instruction can properly test it.

Step 2. Location 34. If the condition is true, the next instruction to be executed is the CACS instruction at location 35. If the condition is false, the PCR will be incremented and the next sequencial Command-Code will be executed.

4.2.2.3 Single Micro-Instruction Group. The Command-Codes in this group require only a single micro-instruction cycle for their execution.

4.2.2.3.1 FEND. M.C. Flow charts page M18. The FEND operation actually involves two micro instructions. The second micro-instruction is used to place the HFPU in a state so that it will be receptive to A/Q commands.

Step 1. Location 1D. This instruction waits for the FPMP to complete its current operation and then proceeds to set the FEND bit and clear the ACTIVE bit in the FSR.

Step 2. Location E. This instruction contains simply the Execute Next field which is used to disable the Next Instruction Address output of the Master Micro-Processor (MMP) instruction register. If an A/Q STOP command has not been received, the MMP Clock will be stopped and the HFPU will await further A/Q commands. If a Stop Request is pending, the micro-processor will proceed to execute the first instruction of the STOP sequence at location 2.

4.2.2.3.2 CHMD. Location M11. This single micro-instruction complements the state of the Relative Mode bit in the FSR and proceeds to first instruction of the next Command-Code.

4.2.2.3.3 NIDX. M.C. Flowchart page M11. This micro-instruction clears the Index Register and proceeds to the next Command-Code sequence.

4.2.2.4 Floating Point Group. These commands all require action on the part of the FPMP. Four bits of the MMP Starting Address are used to provide a starting address for the FPMP. The bits are, in order from most significant to least significant bit, bits 1,Ø,7,6. A fifth bit of the MMP starting address, bit 2, is used to indicate that the FPMP is to perform an FSUB function instead an FADD. Note that the MMP activity involved in the five functions, FLDD, FADD, FSUB, FMPY, and FDIV is the same. Thus, the MMP action for these five functions is described only once in the following section on FLDD.

4.2.2.4.1 FLDD, M. C. Flowcharts page M3. FPH Flow charts page F9.

Step 1. Location 20 if FADD, FMPY, FDIV or FLDD. Location 24 if FSUB. Initiate a DSA memory read request with the PCR as the memory address. If the Relative Mode bit is false, the DSA data will be loaded into TAR. If the Relative Mode bit is true, the PCR will be added to the DSA data and the result loaded into TAR. On the trailing edge of RESUME, the PCR will be incremented.

Step 2. Location 21. Initiate consecutive DSA memory read requests. If FSR bit 8 is set, select the IRALU to multiply the IR by 1. If FSR bit 8 is clear, and the HFPU is in single-precision select the IRALU multiply IR by 2; if in double-precision, multiply the IR by 3. Add the outputs of the IRALU to the data in TAR and apply the result to the DSA address bus. Load the DSA data into the LABF word 1 which corresponds to FPAC bits 0 to 15. The DSA Interface automatically increments TAR on the leading edge of RESUME, so that the address is advanced in time for the next cycle, which will be stolen consecutively.

Step 3. Location 25. Request a DSA memory-read cycle with the address generation as in Step 2. If the unit is in single-precision, disable the consecutive cycle request and allow the release of the scanner. Load the DSA data into the LABF word 2. If the unit is in single-precision mode skip to Step 5.

Step 4. Location 26. Request a DSA memory-read cycle with the address generation as in Step 2. Load the DSA data into LABF word 3.

Step 5. Location 27. Wait for the FPMP to complete its current operation before proceeding. Transfer word 1 of the LABF into the high word of the MDREG (bits 0 to 15).

If in single-precision, load the sign bit into bits 32 to 51 of the MDREG. If in double-precision, load the sign bit into bits 48 to 51 only. The guard digits are thus set to true Ø in ones complement arithmetic.

Step 6. Location 28. Transfer word 2 of the LABF to the middle word of the MDREG.

Step 7. Location 29. If the unit is in double-precision mode, transfer word 3 of the LABF to the low word of the MDREG. If in single precision, CLK3 is inhibited thus leaving the guard digits unaffected. The MIRCLK that terminates this micro-instruction sends a start signal to the FPMP so that it can begin its portion of the FLDD function. The MMP now proceeds to the first micro-instruction of the next Command-Code.

This completes the MMP action during a floating load. The following steps refer to the action taken by the FPMP. Refer to page F2 of the FPH Flow charts. Note that the FPH Flow charts in many places are actually drawn as two parallel flow charts, one for the exponent and the other for the mantissa arithmetic.

Step 1. Location 3. Remove the exponent bias and the effects of the mantissa sign by complementing bit 1 and then complementing bits 1 to 8 if the sign bit is set. The complementing referred to here occurs on the input to the MDREG, so that the exponent value in the MDREG is a valid ones-complement number. The FPMP selects its ALU'S to transfer the MDREG into the FPAC in this micro-instruction.

Step 2. Location 18. Sign-extend the mantissa. This micro-instruction is the first example of the conditional ALU control in the FPMP. The MALU is set to pass a zero on its B input through to its output. If bit 0 of the FPAC is set, however, the MALU function will be inverted to a B Complement thus producing all 1's on its output. Thus the output of the MALU is equal to the sign of the FPAC. This result is clocked into section 4, bits 48 to 51, of the FPAC if the unit is in double-precision. If the unit is in single-precision, this result will be clocked in bits 32 to 51 of the FPAC.

Step 3. Location 15. This step is used to clear negative 0's in both the exponent and the mantissa. Both the EALU and MALU are set to the A Arithmetic mode and the result loaded into the FPAC. The FPMP halts upon the completion of this instruction.

4.2.2.4.2 FADD/FSUB. FPH Flowcharts page F6. All of the action described here tskes place in the FPMP. The MMP action required for these functions was described above in the section on FLDD.

Step 1. Location Ø of the floating-point micro-code. The first micro-instruction of FADD transfers the mantissa of the memory argument from the MDREG to the BREG. In the exponent arithmetic, the exponent of the memory argument is subtracted from the exponent of the FPAC. If this difference is positive, the PICK F/F will be set and the Shift Counter will be loaded with the difference. If this difference is

negative, the PICK F/F will be cleared and the Shift Counter will be loaded with the complement of this difference. The Shift Counter is thus loaded with the magnitude of the difference of the exponents and the PICK F/F has the sign of the difference stored in it.

Step 2. Location 1C. The mantissa portions of the FPAC and BREG are shifted right one place arithmetically to open up an overflow bit in bit position 9. This allows the sum of the mantissa magnitudes to overflow without interferring with the sign bit. The exponent of the result will be incremented at step 4 below in order to mantain the correct value for the floating-point number. Note that shifting the mantissa right one place divides the floating-point number by 2 and that adding 1 to the exponent multiplies the floating-point number by 2 thus leaving its value unchanged. In the exponent arithmetic, if the PICK F/F is clear, the exponent of the memory argument is transferred to the exponent of the FPAC. If the PICK F/F is set, the clock to the EACC is inhibited thus leaving the exponent unchanged. This has the effect of selecting the larger of the two exponents for the exponent of the result. This micro instruction at location 1C also performs a conditional test on the magnitude of the exponent difference. If the magnitude of the exponent difference is too big for the size of the register (greater than or equal to 26 in single precision, 42 in double precision), then the next micro-instruction will be the one at location 1F. If the shift count is smaller than the register size, then the next micro-instruction will beat location 1D.

Step 3. Location 1F or 1D. If the shift count was to big, then the instruction at location 1F will be executed. This micro-instruction clears the mantissa of the floating-point number having the smaller exponent. If the shift count is smaller than the register size, then the instruction at location 1D is executed. This micro-instruction calls on the hard-wired shift logic to shift the mantissa of the argument having the smaller exponent right a number of places equal to the exponent difference. The micro-instruction sets the mantissa shift register mode controls into the right shift mode and the hard-wired shift logic shifts the mantissa of the smaller number right until the shift counter reaches zero. The shift counter was loaded with the magnitude of the exponent difference at step 1 above. This micro-instruction accomplishes the "exponent-alignment" portion of the floating addition. In effect, the smaller floating-point number has its mantissa shifted right arithmetically and its exponent incremented (thus maintaining its value unchanged) until its exponent is equal to the exponent of the larger number. Once exponent equality has been achieved, then the two mantissa's can be added together.

Step 4. Location 1E. This micro-instruction performs the mantissa addition. If the operation being performed is FADD, then the two mantissas are added together using one's-complement arithmetic. If the operation is FSUB, then the complement of the BREG is added to the mantissa of FPAC using one's complement arithmetic. The exponent value in the EACC is incremented by 1 to compensate for the right

shift of the mantissas that occurred in step 2 above. At this point the floating-point addition is completed and all that remains is the normalization of the result. What follows is the description of the common normalize logic that is used by all of the functions that require post-normalization.

NORMALIZE,FPH Flowchart page F12 and F13.

Step 1. Location 11. The first micro-instruction of normalize simply sets up the test for a zero mantissa by setting the MALU to the A,Arithmetic mode. Since this ALU mode converts negative Ø to positive Ø, this instruction allows a test for true zero mantissa.

Step 2. Location 12. If the mantissa portion of the FPAC is equal to zero, then this instruction becomes a jump to the micro-instruction at location 17 where the microprocessor forces a true Ø result (mantissa = Ø and exponent equals -127_{10}) and halts. If the mantissa is not equal to Ø, then the hard-wired Normalize function will proceed to normalize the floating-point number. The hard-wired Normalized function proceeds in a rather peculiar fashion due to the presence of the rounding which occurs in step 3 below. The problem is that if the mantissa is effectively all 1's and a round is performed, then a carry can ripple through into the most significant bit causing a mantissa overflow. A thorough consideration of the floating-point functions (Add, Subtract, Multiply, Divide and FIXF) shows that if the raw, pre-normalization, mantissa is already normalized, then the rounding of this result will not cause a mantissa overflow. This is due to the fact that a normalized raw result will always contain at least one zero in the middle bit positions in the mantissa thus preventing a carry from rippling through to the most significant bit. If, on the other hand, the raw result is unnormalized and the normalize hardware is allowed to normalize it completely, then the rounding may result in mantissa overflow. To prevent this from happening, the normalize hardware shifts the mantissa left and decrements the exponent until the mantissa is one bit-position short of being normalized if normalization was required. If the raw mantissa is normalized, the normalize hardware does not not shift it.

Step 3. Location 13. This step performs the rounding by adding/subtracting the most significant of the true guard digit to/from the mantissa if the mantissa is positive/negative. If the mantissa resulting from the normalization is normalized, then the most significant bit of the guard digit is one bit off the end of the mantissa (bit 32 in single precision and bit 48 in double precision). If the mantissa is one bit-position short of being normalized, then the most-significant bit of the true guard digit lies two bit positions off the end of the mantissa (bit 33 in single-precision and bit 49 in double-precision).

Step 4. Location 14. This step completes the normalization of the result. If the mantissa resulting from steps 2 and 3 is normalized, no action is taken. If the mantissa is unnormalized (at most it will be unnormalized by one bit position), then the mantissa is shifted left once and

the exponent is decremented to produce a normalized result.

Step 5. Location 18. This step truncates the guard digit to remove spurious information that may be residing in those bits following the rounding. Using one's complement arithmetic, all the bits of the guard digit (bits 32 to 51 in single precision and bit 48 to 51 in double precision), are set equal to the sign of the number true 0 condition.

Step 6. Location 15. This step checks for the possibility of exponent overflow or underflow resulting from the calculation. Within the HFPU, two extra overflow bits, labeled EACC (-2) and (EACC (-1), are carried to allow for the correct detection of exponent overflow or underflow. When a floating-point number is loaded, the overflow bits are set equal to the sign bit of the exponent (EACC1). This sign-extension of the EACC converts it into a ten-bit one's complement number. At the end of the a calculation if EACC1, (-1), and (-2) are all equal, then the exponent is within range i.e., it can expressed in 8 bits, and the result is valid. In this case, the FPMP halts at location 15 and drops its busy signal to the MMP. If EACC1 and EACC (-1) are not the same, then the exponent of the result cannot be represented in 8 bits and an error has occurred. In this case EACC (-2) indicates the true sign of the exponent of the result. If it is alone, then the exponent of the result is negative and exponent underflow has occurred. Micro-program control will then be transferred to location 17. If EACC (-2) is false, then the exponent of the result is positive and exponent overflow has occurred. Micro-program control will then be transferred to location 16.

Step 7. Location 17 if underflow, location 16 if overflow. In the case of underflow the mantissa is set to Ø and the exponent of result is set to the maximum negative value of -127_{10}. In the case of overflow, the exponent is set to the maximum positive value of $+127_{10}$ and the mantissa is set to the maximum signed value. Note that in the case of overflow, micro-program control returns to step 5 (location 18) where the bits of the guard digit will be truncated so that the result is a valid single or double precision floating-point number. In the case of underflow, the micro-processor simply halts at location 17.

4.2.2.4.3 FMPY. FPH FLOWCHART Page F8. Only the FPMP portion of FMPY is described here. The MMP portion was described above in section 4.2.2.4.1, FLDD.

Step 1. Location 1. The FPAC mantissa is transferred to the BREG so that it may be used as the multiplier in the mantissa multiplication. The exponent of the memory argument is added to the EACC and the result is placed in the EACC.

Step 2. Location D. In this step the mantissa of the FPAC is set to positive 0 if the sign of the result as indicated by the Exclusive OR of the FPAC sign (MAC0) and the memory argument sign (MD0) is positive. The MACC is set to negative Ø if the sign of the result is negative 0. This step is necessary so that the arithmetic right shifting of the MACC which occurrs during step 3 below will proceed correctly.

Step 3. Location C. In this step the hard-wired multiply logic performs the mantissa multiplication portion of FMPY. The mulplication is performed using a one's-complement version of the usual binary multiplication algorithm in which the multiplicand is added to the partial product for each true bit in the multiplier and the partial product is shifted right for every bit in the multiplier. In the one's-complement version of this algorithm, a true bit in the multiplier is a bit which has the opposite sense from the sign of the multiplier i.e., if the multiplier is positive, a true bit is a 1, if the multiplier is negative, a true bit is a Ø. Additionally, if the multiplier is negative, the multiplicand is subtracted from the partial product instead of being added to it as in the binary algorithm. If the multiplier is positive, the one's complement algorithm proceeds exactly as in the binary algorithm.

The hard-wired function begins by loading a counter register with the number of steps to be performed (27 in single precision and 43 in double precision). This step count is such as to provide for a correct fractional multiplication with the binary point of the result lying to the left of FPAC bit 9. This corresponds to the mantissa result being either normalized or one bit position short of being normalized. The step count is then immediately decremented preparatory to the test for algorithm completion occurs at the end of the loop in the flowchart. If the "true" least-significant bit condition exists, then the mantissa of the Memory Argument in the MDREG is added to or subtracted from the mantissa of the FPAC depending on the sign of the multiplier in the BREG. If the "true" bit condition is not satisfied, then the mantissa of the FPAC is left unmodified. The next step in the algorithm is to shift both the mantissa of the FPAC and the mantissa of the multiplier (in the BREG) one position to the right. The BREG is shifted right arithmetically thus preserving its sign in bit position Ø so that the tests for a "true" bit and the add/subtract decision will proceed correctly in succeeding steps. As was illustrated in figure 4.4 of section 4.1.3, the content of the FSAN F/F is shifted in to the sign position of the MACC. The SFAN F/F was loaded with the expected sign of the multiply result at the time that the MMP started the FPMP. This sign is given by the Exclusive OR of the sign of the memory argument and the sign of the FPAC. The SFAN F/F is provided as

the right serial input to the MACC so that the MACC will shift right in a true arithmetic fashion even if some of the intermediate steps in the multiply algorithm result in a temporary overflow of the mantissa into the mantissa sign bit. The final step in the algorithm is a check of the status of the step count. If it has reached Ø, the proper number of steps have been completed and microprocessor control is transferred to the normalize routine at location 11 in the micro code, otherwise the hard-wired algorithm proceeds to check the next bit in the multiplier for the "true" condition. The Action of the normalized routine was described in the preceding section of FADD/FSUB.

4.2.2.4.4 FDIV. FPH Flowcharts page F9. MMP action for the FDIV function was described above in the section on FLDD. What follows then is a description of the FPMP action involved in FDIV.

Step 1. Location 2. When the MMP starts the FPMP, the potential sign of the result (equal to the Exclusive OR of the sign of the memory argument and the sign of the FPAC) is loaded into the SFAN F/F. SFAN is set if the sign of the result will be negative. The first step of FDIV shifts the mantissa of the FPAC right arithmetically one position and increments its exponent thus maintaining its value unchanged. This right shift of the dividend is performed in order to open up an overflow bit so that the resulting quotient in the BREG will not overflow that register. Mathematically, the division of two normalized mantissas can result in a quotient that lies between .5 and 1.999. Shifting the MACC right one position effectively divides the quotient by two so that it lies in the range .25 to .999. Thus the result of the divide will either be normalized or one bit-position short of being normalized and thus will be compatible with the hard-wired normalize algorithm. Step 1 also checks the divisior in the MDREG to verify that it is a proper normalized floating-point number. If it is not normalized, then a divide error would occur and program control is transferred to location 16 where the maximum signed result will be forced. The microprocessor action involved in forcing the maximum result has been described above in the normalize section of FADD/FSUB. If the divisor is a proper normalized floating-point number then the micro-program proceeds to step 2.

Step 2. Location 10. In this step, the exponent of the divisor is subtracted from the exponent of the dividend to generate the exponent of the result. The mantissa of the dividend, in the FPAC, is converted to absolute value. This is performed by setting the MALU to the A, logical mode and allowing inversion of the MALU to the A, complement mode if the sign of the FPAC (MACØ-H) is set. The bits in the mantissa of the FPAC are thus complemented if it is negative. The MACC is converted to an absolute value in order to simplify the decisions that are made during the mantissa division which is described in the following step.

Step 3. Location 1A. This step utilizes the hard-wired divide logic to perform the division of the two mantissas. The algorithm used is a one's-complement version of the standard binary division algorithm. As in the hard-wired multiply logic the sequence begins by initializing the step counter so that the result will end up either normalized or one bit-position short of being normalized. Note that the value loaded in the count (28 in single precision or 44 in double precision) is exactly equal to the number of bits in the mantissa including the guard digit and the sign bit.

Figure 4.4 of section 4.1.3 illustrates the bit position at which the quotient bits are shifted into the BREG as the divide result is generated. The hard-wired logic assumes that the magnitude of the dividend is in the FPAC and that the one's-complement divisor is in the MDREG. The micro-code sets the MALU to the A-B mode with an ALU inversion (A-B goes to A+B) if the sign of the divisor (MDØH) is negative. Thus the MALU is continuously subtracting the magnitude of the divisor from the magnitude of the dividend. The algorithm proceeds by first decrementing the count preparatory to the final test of the count at the end of the algorithm loop. The logic now tests the sign of the output of the MALU (the magnitude of the dividend minus the magnitude of the divisor). If this sign is positive then it is time to enter a true bit into the quotient and to replace the dividend with the difference between the dividend and the magnitude of the divisor. A "True" bit entered into the quotient consists of a 1 bit if the sign of the answer is positive and a Ø if the sign answer is negative. If the sign of the difference between the magnitude of the dividend and the magnitude of the divisor was negative, then no change is made in the dividend and a false bit must be entered to the quotient. A false bit consists of a Ø if the sign of the answer is postive and 1 if the sign of the answer is negative. Thus the quotient developed is a one's-complement number. The next step in the algorithm consists in multipling the dividend by 2 by shifting it left one position. The final step in the hard-wired algorithm is to test the count to see if has reached Ø. If it has not, the algorithm loops around and performs another step. If it has reached Ø, then the micro-program proceeds to location 11 where the normalization takes place. The normalization was described above in section 4.2.2.4.2, FADD/FSUB.

4.2.2.4.5 FLST. Master control flowcharts page M9. FPH flowcharts page F3. There is no FPMP micro-processor action involved in FLST. The flowchart on pg. F3 merely shows the effect of the passive logic in the floating-point portion of the HFPU during FLST. If the sign of the floating-point number in the FPAC is set then all

8 bits of the exponent are complemented so that they will correspond to the external floating-point format. If the sign of the FPAC is positive, then the bits of the exponent are read un-complemented. Following the above step, the most-significant bit of the exponent is unconditionally complemented so as to put the exponent into the proper biased form. The following steps describe the action taking place in the MMP during FLST.

Step 1. Location 2E. The first step of FLST is to fetch the address of the argument and perform the Relative Address Mode calculation as was described above in step 1 of FLDD, section 4.2.2.4.1 This step is performed in parallel with any preceding FPMP action that may be in progress.

Step 2. Location 2F. The MMP now interrupts its micro-instruction flow to wait for the FPMP to complete its action. When the FPMP drops its busy signal to the MMP, the MMP proceeds to generate a memory-write cycle request to the DSA interface. The multiplied Index value is added to the address in TAR as was described above in step 2 of FLDD. The MMP enables bits 0 to 15 of the FPAC onto the DATA 0 to 15 lines and the DSA interface drives them at the appropriate time onto the DSA data bus. The address in TAR is incremented by the DSA interface on the leading edge of the RESUME signal. Note that this cycle is performed in consecutive-cycle mode with the scanner halted if the HFPU s in BLOCK mode. Because of the wait at the beginning of this step for the FPMP, the first step did not hold the scanner at the completion of its cycle.

Step 3. Location 30. This micro-instruction generates the second write-cycle request to the DSA interface with the address information as in step 2 above. The MMP enables bits 16 to 31 of the FPAC onto the DATA 0 to 15 lines so that the DSA interface can drive them to the DSA data bus. If the HFPU is in single-precision mode then this micro-instruction constitutes the final step of FLST. The SPINH bit disables the SHLT and CC functions in the DSA field and the "EXEC-NXT if SP" code causes the MMP to perform its Execute Next function which takes the Micro-program sequence to the next Command-Code. If the HFPU is in double-precision mode, then the cycle of step 3 proceeds exactly as the cycle of step 2 with the scanner remaining halted and the Consecutive-Cycle mode in effect. In this mode, TAR is incremented on the leading edge of RESUME so that the address is ready for the next cycle.

Step 4. Location 31. This step is performed only if the HFPU is in double-precision mode. This step consists of a third DSA memory-write cycle with the address formation as in steps 2 and 3 above and with bits 32 to 47 of the FPAC being enabled to the DSA data bus. Upon completion of this step, the micro-instuction sequence proceeds to the beginning of the next Command-Code.

4.2.2.4.6 FIXF. MMP flowcharts pg. 7, FPH flowcharts pg. F4. For the FIXF function, the master control fetches the integer at the effective operand address, and the FPMP performs the integer to floating-point conversion by supplying the appropriate exponent and normalizing the result. The discussion below begins with the action on the part of the Master Micro-Processor.

Step 1. Location 1A of the MMP micro-code. The first step of FIXF involves the fetch of the address with the relative address computation being performed as in step 1 of FLDD, section 4.2.2.4.1

Step 2. Location 1B. In this micro-instruction, the MMP requests a single DSA memory-read cycle with unmultiplied indexed addressing allowed. This step is exactly the same as step 3 of FLDD with the exception that the IR will not be multiplied by 2 or 3. This characteristic allows the HFPU to access sequencial elements in dimensioned integer variables exactly as it does for dmensioned real and double-precision variables. The integer fetched from memory is placed into the middle word of the Look-Ahead Buffer.

Step 3. Location 1C. The MMP first interrupts its micro-instruction sequence to wait for the FPMP to complete any preceding function. The MMP then transfers the integer to the middle word of the input register of the floating-point arithmetic, the MDREG, and it also places the sign in the sign bit of that register by loading the integer into the entire high word of the MDREG. The FPMP will ignore the surrious bits loaded into the other bit positions and will concern itself only with the bit positions 0 and 16 to 31. The MMP starts the FPMP at the beginning of its FIXF function. This completes the MMP action during the FIXF function. It now proceeds to the beginning of the sequence for the next Command-Code. What follows then is a description of the FPMP action involved in FIXF.

Step 1. Location 8 of the Floating-Point micro-code, pg. F4 of the FPH flowcharts. In this step the FPMP utilizes the exponent constant 17_{16} to present the exponent of the FPAC to 23_{10}. This value is chosen so that when the integer is shifted from its present position in bits 16 to 31 to its final position somewhere in bits 9 to 23, the resulting floating-point number will have the correct exponent value. In the mantissa, the FPMP transfers the input argument from the MDREG to the mantissa of the FPAC. This results in the integer lying in bits 16 to 31 and the sign bit in bit position 0.

Step 2. Location A. This step sets all of the bits of the mantissa of the FPAC equal to the sign bit except for bits 16 to 31 which are left unmodified. This is done by selecting the MALU to the B mode with a $\emptyset$ supplied to the B input. An ALU inversion is allowed

if the sign of the FPAC (MACØ) is true. In that case, the MALU function will be changed to a B, complement, thus producing all 1's on the MALU output if the FPAC is negative. The output of the MALU, 0's if positive, 1's if negative, is entered into sections 1,3, and 4 of the MACC. Micro-program control is now transferred to the normalize section of the micro-code at location 11. This is exactly the same normalize code that was described above in section 4.2.2.4.2 on FADD/FSUB.

4.2.2.4.7 FLOF. Master Control flowcharts pg. M6, FPH flowcharts pg. F5. The FLOF function is unique in that the action of the MMP and that of the FPMP are more interlocked than they are in the other functions. The MMP starts the FPMP running on the conversion of the contents of the FPAC from floating-point to integer. It then proceeds to the address generation and then waits for the FPMP to complete its operation before storing the result.

MMP Step 1. Location 22 of the Master Control micro-code. The MMP first waits for the FPMP to complete any proceeding function that may be progress and then starts it executing on the FLOF function.

FPMP Step 1. Location 9 of the FPMP micro-code. In the first step of FLOF, the EALU is set to compare the EACC against an exponent value of 15_{10} (F_{16}). If the exponent value in the FPAC is greater than 15_{10}, then the floating-point number in the FPAC is too large to be converted to a 16 bit, one's-complement integer value. If the exponent is greater than 15, then program control transfers to location F. If the exponent is less than or equal to 15, then program control transfers to location D.

Step 2. Location D or F. The micro-insruction at location F is used to force the maximum signed-integer result if the floating-point number is too large to be converted to an integer. This is done by driving the maximum negative integer value, 8000_{16}, to the B input of the MALU, setting the MALU to the B, complement mode, and allowing an ALU inversion if the FPAC sign is set. Thus, if the FPAC is positive, the maximum positive integer, $7FFF_{16}$, (the one's-complement of 8000_{16}) will be loaded intc the middle portion of the FPAC. If the FPAC is negative, the MALU function will be changed to B and the middle word of the FPAC will be loaded with the maximum negative integer. The micro-instruction at location F is the final step of FLOF if the floating-point number in the FPAC was to large to be converted to an integer. If the floating-point number was within range, then the micro instruction at location D is executed. This micro-instruction loads the Shift Count register with the value 23_{10}-EACC. This value is the number of positions that the mantissa of the FPAC must the shifted to the right in order to place the most-significant 15 bits into FPAC bits 16 to

31. This micro-instruction also tests the magnitude of the value loaded into the Shift Count register to see if any bits of significance will remain in FPAC 16 to 31. If the value is less than 23_{10}, then the integer result will be greater than Ø and micro-programmed control transfers to location E. If the value is greater than or equal to 23_{10} than all the bits significance of the mantissa of the FPAC would be shifted past bit position 31 and the result will be 0. In order to avoid unnecessarily long shifts and to prevent the possibility of a negative Ø result, micro-program control transfers in this case to the force-zero micro-instruction at location 17 which was previously described in the description of normalize in section 4.2.2.4.2, FADD/FSUB.

FPMP Step 3. Location E. In this step, the hard-wired shift logic shifts the mantissa of the FPAC right a umber of places equal to the value in the Shift Count register.

FPMP Step 4. Location 15. The FPMP comes to a halt at this location thus completing FPMP action in FLOF.

MMP step 2. Location 38 of the Master Micro-Code. In this step, the MMP fetches the address and performs the relative address calculation that was described above in step 1 of FLDD. This MMP step occurs in parallel with the FPMP operations described above.

MMP step 3. Location 23. The MMP first interrupts its micro-program sequence to wait for the FPMP to complete its FLOF operation. The MMP then proceeds to generate a DSA memory-write request with the index addressing as in step 2 of FIXF. The raw, unmultiplied index value is used so that the integer result of the FLOF function can be stored into a dimensioned integer array. Bits 16 to 31 of the FPAC are enabled onto the DATA 0 to 15 lines so that the DSA interface can drive them to the DSA data bus. This completes the MMP action in FLOF and micro-program control now transfers to the next Command-Code.

4.2.2.4.8 FCOM.MMP flowcharts pg. M8, FPH flowcharts pg. F4. At location 19 in the Master Micro-Code, the MMP waits for the FPMP to complete its previous function and then starts the FPMP executing on the FCOM function. The MMP then proceeds to execute the next Command-Code. The following is a description of the FPMP action in FCOM.

Step 1. Location 7 of the FPMP micro-code. In this step, the FPMP complements the mantissa of the FPAC with the MALU in the logical A Complement mode.

Step 2. Location 15. This micro-instruction passes the FPAC through the EALU and the MALU with the ALU's in the A, arithmetic code. This has the effect of converting negative in the mantissa and exponent to positive Ø. The FPMP then halts.

4.2.2.4.9 A/Q Load FPAC Commands. MMP flowcharts pg. M8, FPMP flowcharts pg. F2. Both micro-processors must be activated in order to complete the execution of the three A/Q commands which are used to load the three sections of the FPAC. The decoding logic in the A/Q interface loads the 16 bits of the A bus data into the appropriate section of the MDREG and starts the MMP executing the micro-instruction at location 1 in the MMP micro-code. Bits 7 and 6 of the MMP starting address are used to inform the FPMP as to which of the three A/Q load commands it is to execute. The MMP simply starts the FPMP running and proceeds to location E where it halts. The FPMP then proceeds in a manner essentially identical of that of the FLDD function. Its first micro-instruction lies at location 4,5 or 6 respectively for the functions load FPAC bits 0 to 15, 16 to 31, or 32 to 47. The FPMP transfers the data in the MDREG into the appropriate bits of the FPAC. If it is performing the load FPAC bits 0 to 15 function, then it also loads the sign bits into the guard digits. The micro-program proceeds to location 18 where it performs a second sign-extension of the sign into the guard digits. This step is performed in order to prevent a possible exponent-error detection in the succeeding step. The program then proceeds to location 15, where it clears negative Ø's in the exponent and mantissa and halts.

4.2.2.5 Index Register Group. The three commands in this group are used to load, modify and store the contents of the IR. These three Command-Codes all execute without requiring any action on the part of the FPMP. Additionally, they all begin with the fetch of the address and the relative address calculation that was described above in step 1 of FLDD. In what follows then, the second step of each of these three functions will be desribed.

4.2.2.5.1 INDX. Page M12 of the MC Flowcharts. Locations 2C and 2D in the MMP Micro-Code. The second step of INDX involves a DSA memory-read request with the unmodified contents of TAR bring used as the memory address. The effective address for the INDX function is not indexed. The specification of the IRCLK field in the micro-instruction causes the DSA data to be passed through the main ALU on the ADDR board by forcing the TDMUX to select its DATA 0 to 15 input during the RESUME signal. The micro-instruction sets the ALU to select its A input and directed the TDMUX to select its TAR input during the the REQUEST portion of the memory cycle, so that the contents of TAR were passed through the ALU where they were driven to the DSA address bus by the DSA interface. During

RESUME the DSA data will pass through the ALU irrespective of what data may present on the B input to the ALU. The output of the ALU is loaded into IR on the trailing edge of RESUME. The MMP then proceeds to execute the first instruction of the next Command-Code.

4.2.2.5.2 ADDI. MC Flowcharts pg. M13. Micro-code locations 2A and 2B. The second step of ADDI is essentially identical to the second step of INDX with the exception being that the PIMUX is set to select its IR input and the main ALU on the ADDR board is set to add its A and B inputs together. During RESUME the PIMUX is enabled and the TDMUX is set to select its DATA 0 to 15 input so that the DSA data is added to the contents of the IR and the result loaded into the IR. As in INDX, the main ALU presents the contents of TAR to its outputs during the REQUEST portion of the DSA cycle so that it can be used as the memory address. The MMP then proceeds to execute the next Command-Code.

4.2.2.5.3 STRI. MC Flowcharts pg. M14. Micro-code locations 17 and 18. The second step of STRI consists of a DSA memory-write request with TAR being used as the address. The main ALU on the ADDR board is set to select its A input with the TDMUX being set to select the TAR input. Thus the output of the ALU which is used as the DSA address corresponds to the contents of TAR. The PIMUX is enabled and set to select its IR input and the GROUP1 field is set to ADATA. This drives the output of the PIMUX, which corresponds to the IR, onto the DATA 0 to 15 lines so that the DSA interface can drive it to the DSA data bus during the memory cycle. The MMP then proceeds to execute the next Command-Code.

4.2.2.6 The A/Q STOP Command. MC flowchart pg. M19 FPH flowcharts pg. F3 (same as in FLST). If the HFPU is inactive and the A/Q STOP Command is received, the A/Q interface forces the MMP to begin running at location 2. If the HFPU is active when the command is received, the A/Q interface sets the STOP REQUEST F/F. When the MMP tries to execute the next Command-Code with the Operand Byte Count not equal to 4 and the SPEC F/F clear, then the presence of a Stop Request will cause it to jump to location 2 and execute the STOP sequence which is described below.

Step 1. Location 2 of the MMP Micro-Code. Since the STOP command requires that the Protect Mode bit in the FSR be set, the actual FSR value must be saved before the sequence of memory-write cycles begins. Thus the MMP first waits for the FPMP to complete its action so that any FSR changes (DVFL,OVFL or UNFL) will have been recorded. The The MMP then reads the FSR onto the DATA lines and stores it in the first word of the LABF.

Step 2. Location F. Having saved the FSR, the MMP sets the Active and Protect bits and transfers word zero of the LABF, the SSAR, into TAR. The SSAR was loaded directly by the decoding logic in the decoding A/Q interface at the time that the STOP command was received.

Step 3. Location 10. This DSA memory-write cycle stores the value of the FSR that was saved in LABF at the address contained in TAR (SSAR). The cycle is performed in Consecutive Cycle mode so that TAR is incremented on the leading edge on RESUME.

Step 4. Location 11. This cycle stores the CCR at the address contained in TAR (SSAR + 1). TAR is incremented on the edge RESUME.

Step 5. Location 12. This cycle stores the IR at the address contained in TAR (SSAR + 2). The TDMUX is set to select its TAR input and the main ALU is set to select its A input so that the output of the ALU corresponds to TAR. The PIMUX is set to select its IR input so that the value in the IR can be driven to the DATA 0 to 15 lines and from there to the DSA data lines by the DSA interface. TAR is incremented on the leading edge of RESUME.

Step 6. Location 13. This cycle stores the PCR at the address contained TAR (SSAR + 3) in a manner essentially identical to the store of the IR in step 5. The only difference that the PIMUX is set to select its PCR input. TAR is again incremented on the leading edge of RESUME.

Step 7. Location 14. This cycle stores the high word of the FPAC at the address contained TAR (SSAR + 4). TAR is incremented to the leading edge of RESUME.

Step 8. Location 15. This cycle stores the middle word of the FPAC at the address contained in TAR (SSAR + 5). TAR is incremented from the leading edge of RESUME.

Step 9. Location 16. This cycle stores the low word of the FPAC at the address contained in TAR (SSAR + 6). This is the final memory cycle of the STOP sequence thus the SHLT and CC fields are cleared so that the scanner will be released at the end of this cycle.

Step 10. Location D. This instruction clears the Active bit in the FSR and starts the micro-processor HALT sequence.

Step 11. Location E. This micro-instruction is entered as start of the HALT sequence. The only field that is set in it is the EXEC NXT field. This places the micro-processor in the proper state to accept new A/Q Commands. If a Stop Request is not pending the micro-processor clock will stop at this point. If a Stop Request is pending, than the clock will not stop and the

micro-processor will now proceed to execute the first micro-instruction of the STOP sequence at location 2. Note that the Stop Request that initiated the current STOP sequence was cleared automatically by the entry into this sequence. The ability of the MMP to execute sequential Stop Commands allows it to be operated in a multi-level interrupt environment.

4.2.2.7 RESTART. MC flowcharts pg. M22, FPH flow chart pg F2. The restart sequence is performed primarily by the MMP. The FPMP is utilized towards the end of the sequence to performe an FLDD sequence to transfer the FPAC contents fetched from memory out of the MDREG into the FPAC.

Step 1. Location Ø. This instruction sets the Active bit in the FSR and transfers word Ø of the LABF (the SSAR) into TAR. Note that in both the STOP and the RESTART sequences, the value in the SSAR is left unchanged by the sequence. This is the value that would be read by an A/Q read SSAR command.

Step 2. Location 4. This Micro-instruction sets the double-precision mode bit in the FSR so that the FPAC value that will be fetched from memory can be loaded correctly into the FPAC reguardless of the precision that it was originally expressed in. This instruction requests a DSA memory recycle with TAR as the address. The DSA data is loaded into word 1 of the LABF. The first word fetched is the value that will ultimately be loaded into the FSR. This value is saved in the LABF until the end of the sequence to prevent any conflicts that might arise from its being loaded into FSR at this point. This cycle is performed in Consecutive-Cycle mode so that TAR is incremented on the leading edge of RESUME.

Step 3. Location 5. This instruction fetches the data at the location pointed to by TAR (SSAR + 1) into the CCR and causes TAR to be incremented on the leading edge of RESUME.

Step 4. Location 6. This instruction reads the data at the location pointed to by TAR (SSAR + 2) and causes that data to be loaded into the IR. The load of the IR proceeds in a manner identical to that described in section 4.2.2.5.1, INDX. The setting of the IRCLK bit in the micro-instruction causes the main ALU and the TBMUX to switch from TAR to DATA 0 to 15 during the RESUME signal, thus allowing the data to be presented to the input of the IR. This, unfortunately, inhibits TAR from being presented to the DSA address bus during the later part of RESUME. Thus consecutive cycles can not be used for this and the following cycle. TAR is incremented explicitly by the TARCLK bit in the micro-instruction.

Step 5. Location 7. This cycle fetches the data at the address contained in TAR (SSAR + 3) and loads it into PCR. It functions in a manner identical to that of step 4 above with the exception that the PCR is loaded instead of the IR. TAR is incremented explicitly by the micro-instruction.

Step 6. Location 8. This cycle fetches the data at the address contained in TAR (SSAR +4) and loads it into the high word of the MDREG. Additionally, the most significant bit is loaded into the guard digit bits of the MDREG to allow for sign-extension. This cycle is performed in Consecutive-Cycle mode so that TAR is incremented automatically by the DSA interface on the leading edge of RESUME.

Step 7. Location 9. This instruction fetches the data at TAR (SSAR + 5) into the middle word of the MDREG. TAR is incremented on the leading edge of RESUME.

Step 8. Location A. This cycle fetches the data at the location pointed to by TAR (SSAR + 6) and loads it into the low word of the MDREG. This is the final memory cycle of RESTART, thus the "SHLT,CC" field is cleared so that the scanner will be released at the end of this cycle. In the same micro-instruction, the MMP intiates the FPMP FLDD function. The FPMP then proceeds to transfer the MDREG into the FPAC in an FLDD function as was described above in section 4.2.2.4.1.

Step 9. Location B. The MMP now fetches the old FSR value from word 1 of the LABF and loads it all except for the Active bit into the FSR.

Step 10. Location C. This micro-instruction tests the state of DATA line 0, the old Active bit, to see if it is true. If it is true, i.e., if Active, then the MMP proceeds to execute the next sequential Command-Code. If the old FSR was not active, then the MMP proceeds to location D where it clears the Active bit in the FSR and halts. The old FSR Active bit was not loaded into the FSR at step 9 so as to prevent a possible discontinuity in the Active state of the FHPU, since the HFPU does not really go inactive until it completes the execution of the HALT instruction at location D.

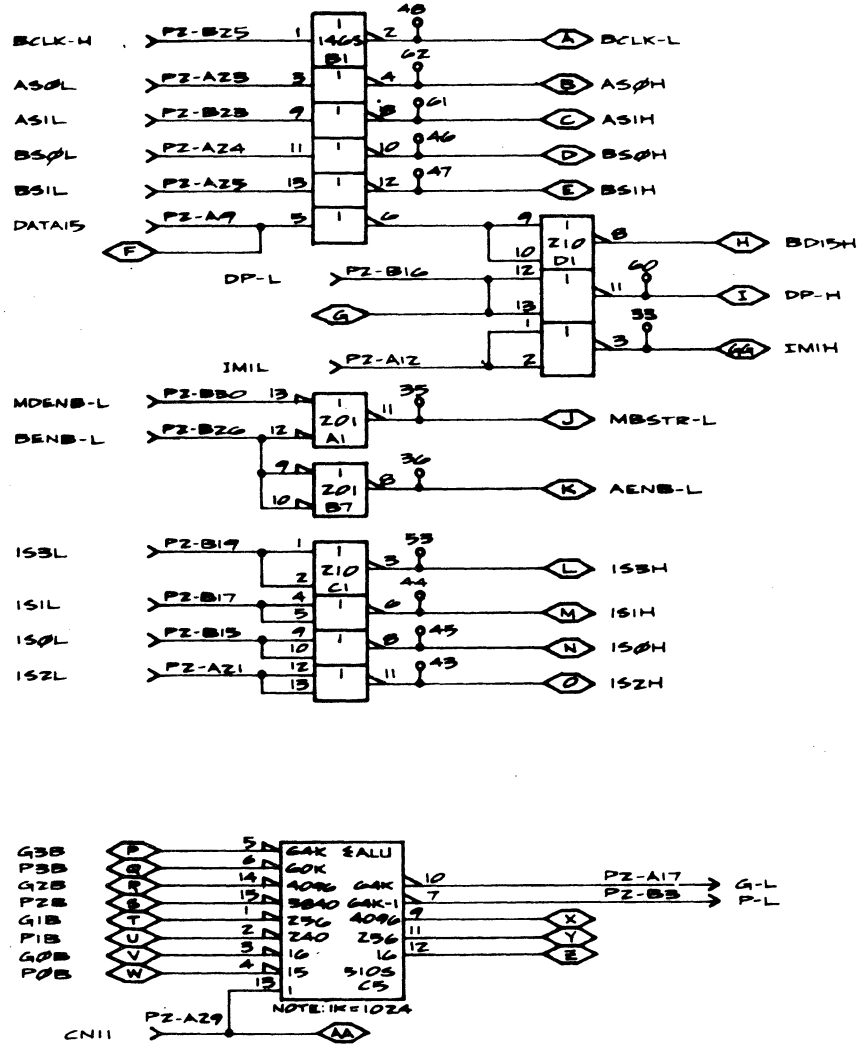
This section of the manual contains the logic diagrams for the hardware floating-point unit.

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8
A	X			X	X	X	X
B	X			X	X	X	X
C	X			X	X	X	X
D	X			X	X	X	X
E	X			X	X	X	X
F	X	X					
G	X		X				
H	X			X	X	X	X
I	X			X	X	X	
J	X			X	X	X	X
K	X			X	X	X	X
L	X			X	X	X	X
M	X			X	X	X	X
N	X			X	X	X	X
O	X			X	X	X	X
P	X			X			
Q	X			X			
R	X				X		
S	X				X		
T	X					X	
U	X					X	
V	X						X
W	X						X
X	X			X			
Y	X				X		
Z	X					X	
AA	X						X
BB				X	X	X	
CC				X	X	X	
DD				X	X		
EE				X	X		
FF				X	X	X	
GG	X			X	X	X	X
HH		X		X			
II		X			X		

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8
JJ		X				X	
KK		X	X				
LL		X	X				
MM		X		X			
NN		X			X		
OO		X				X	
PP		X	X				
QQ		X	X				
RR		X		X			
SS		X			X		
TT		X				X	
UU		X	X				
VV		X		X			
WW		X			X		
XX		X				X	
YY			X				X
ZZ							
AB			X		X		
AC			X		X		
AD			X		X	X	X
AE					X	X	X
AF					X	X	
AG					X	X	
AH					X	X	
AI						X	X
AJ						X	X
AK					X	X	
AL					X	X	

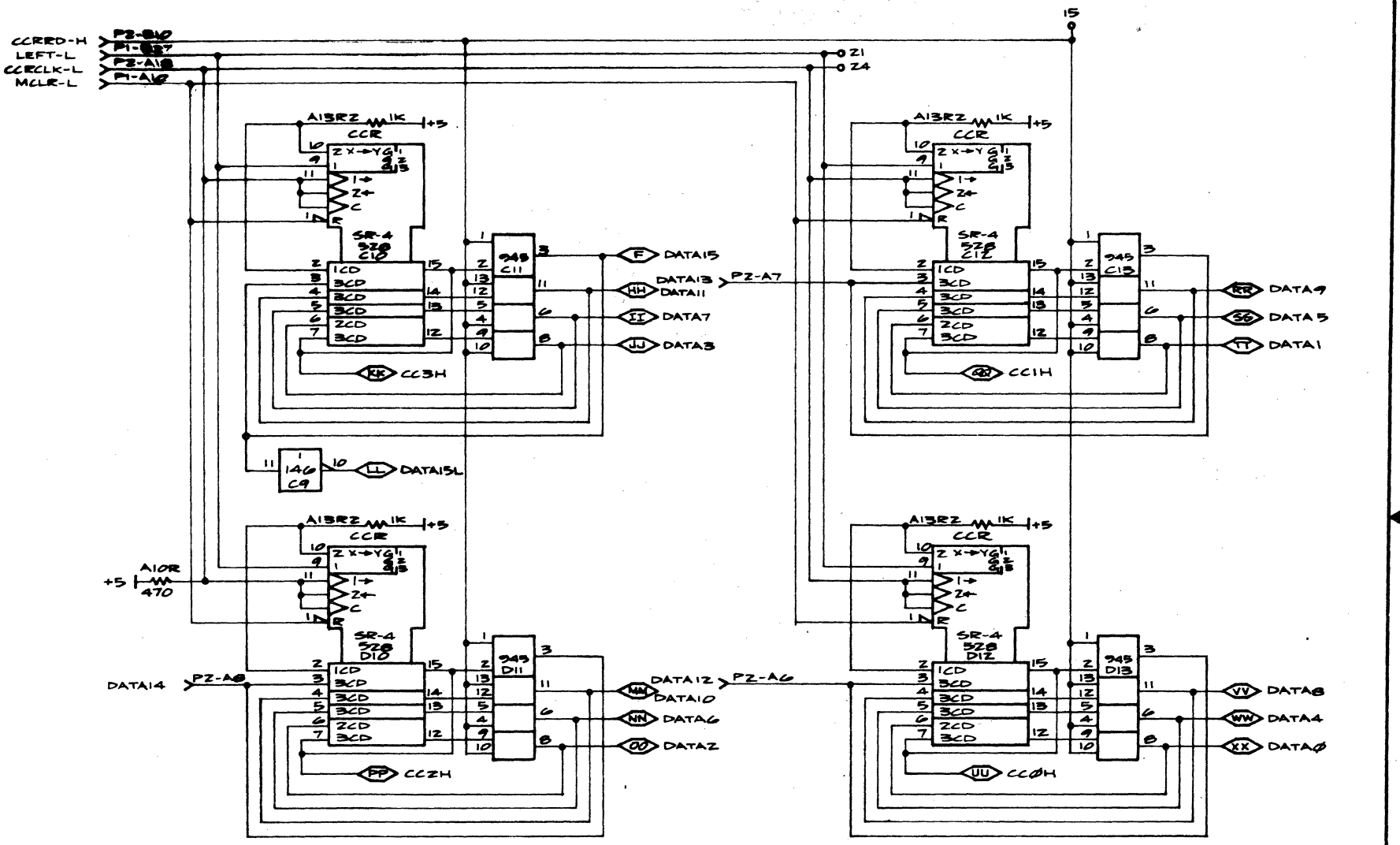
LOGIC TYPE

CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	Bd139 CDC DPALU	01 5/19/75	C	09132	88953200	
	SHEET					SHEET 1 OF 8



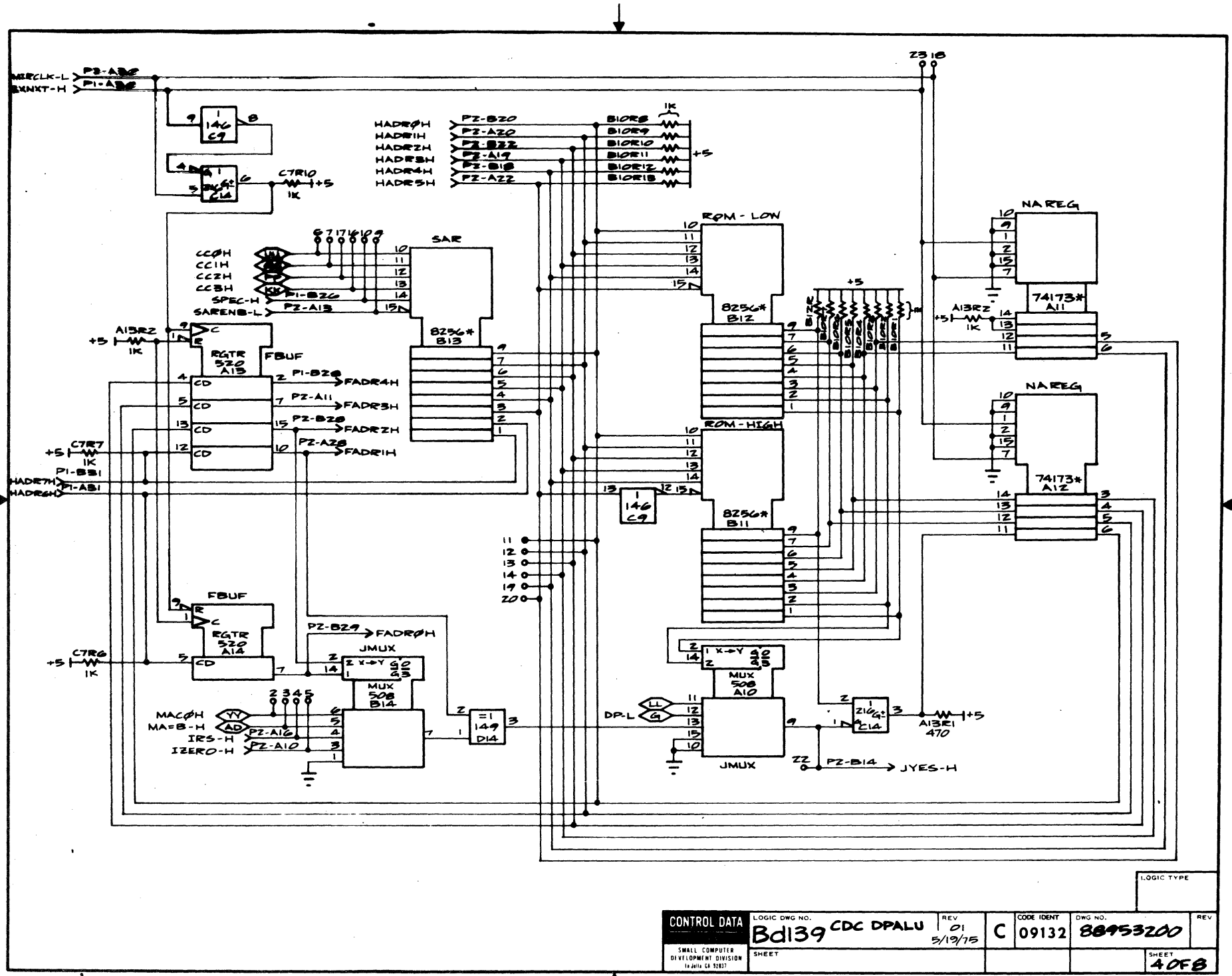
CONTROL DATA	LOGIC DWG NO. Bd139 CDC DPALU	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 80955200	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	SHEET					SHEET 2 OF 8

LOGIC TYPE



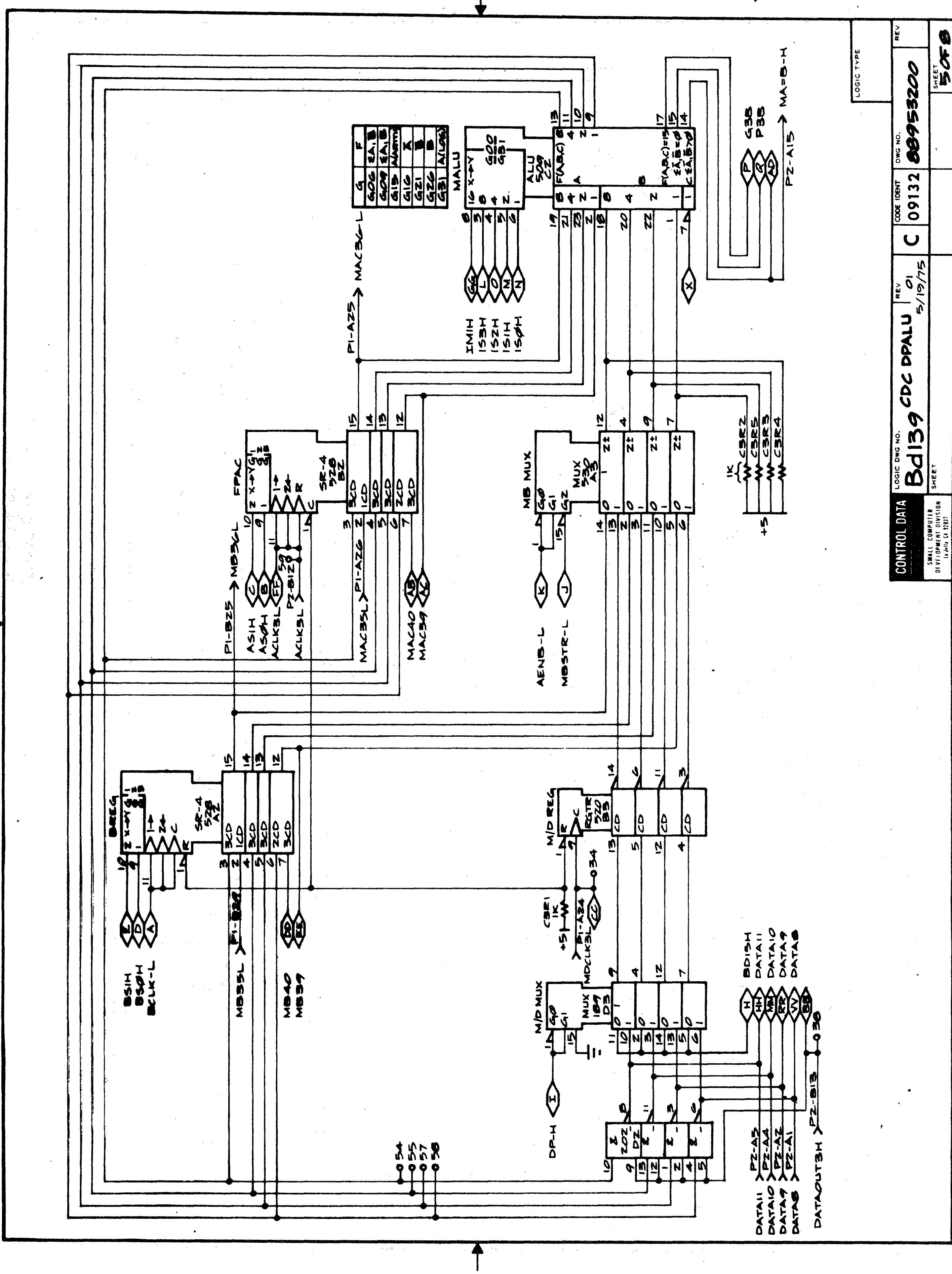
CONTROL DATA	LOGIC DWG NO. Bd139 CDC DPALU	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88953200	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	SHEET				SHEET 3 OF 8	

LOGIC TYPE

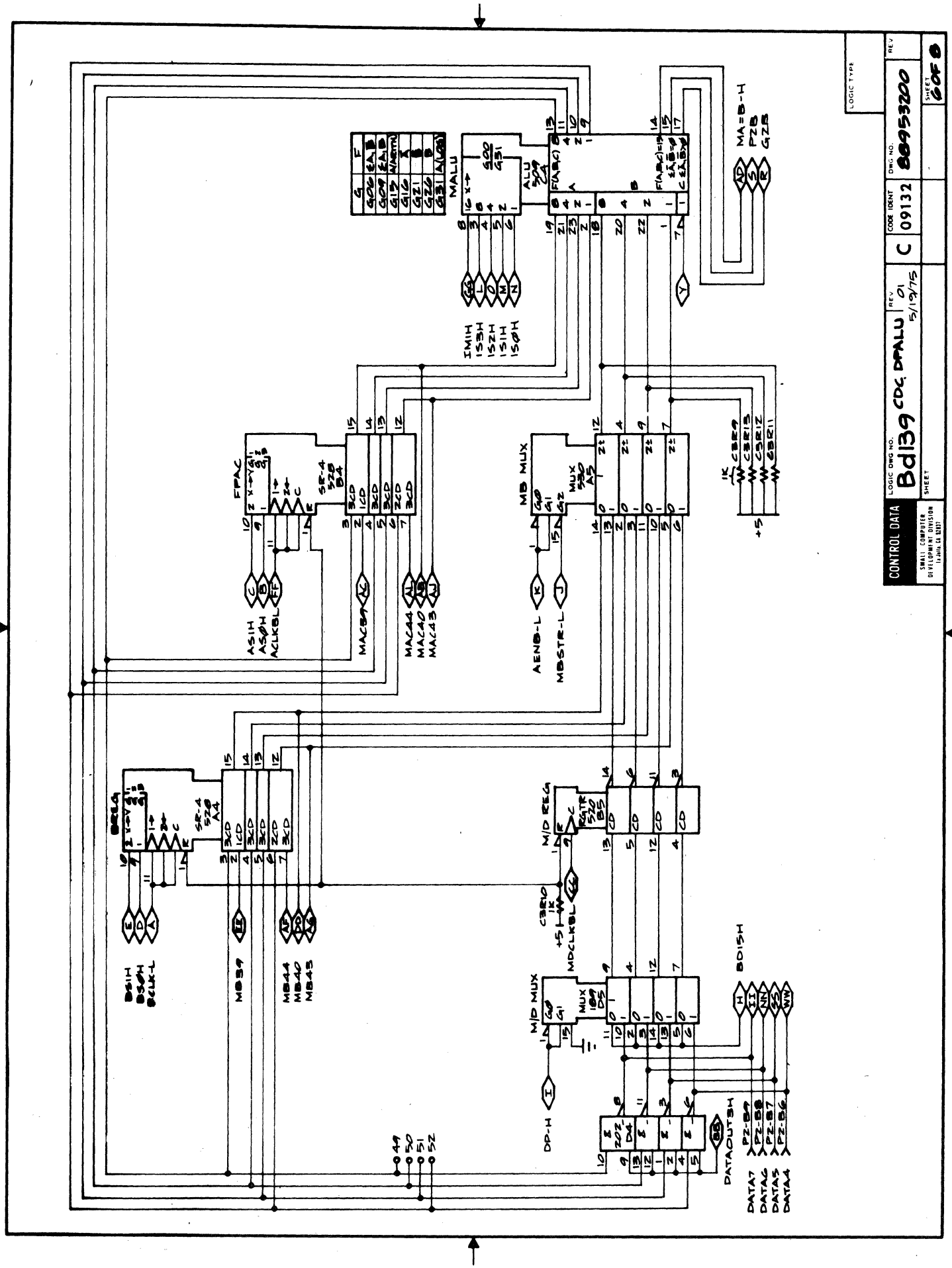


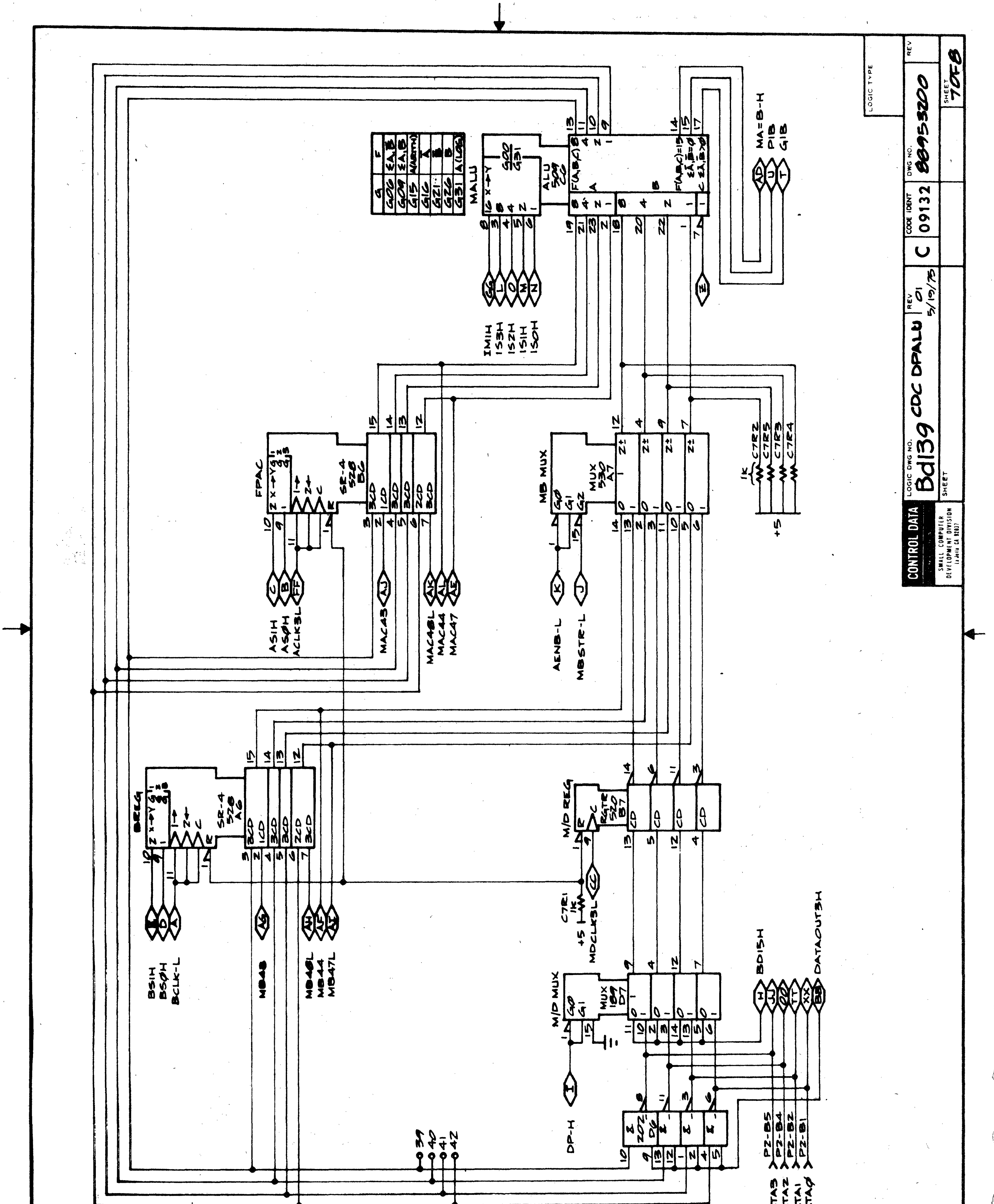
CONTROL DATA	LOGIC DWG NO. Bd139 CDC DPALU	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88953200	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	SHEET				SHEET 40F8	

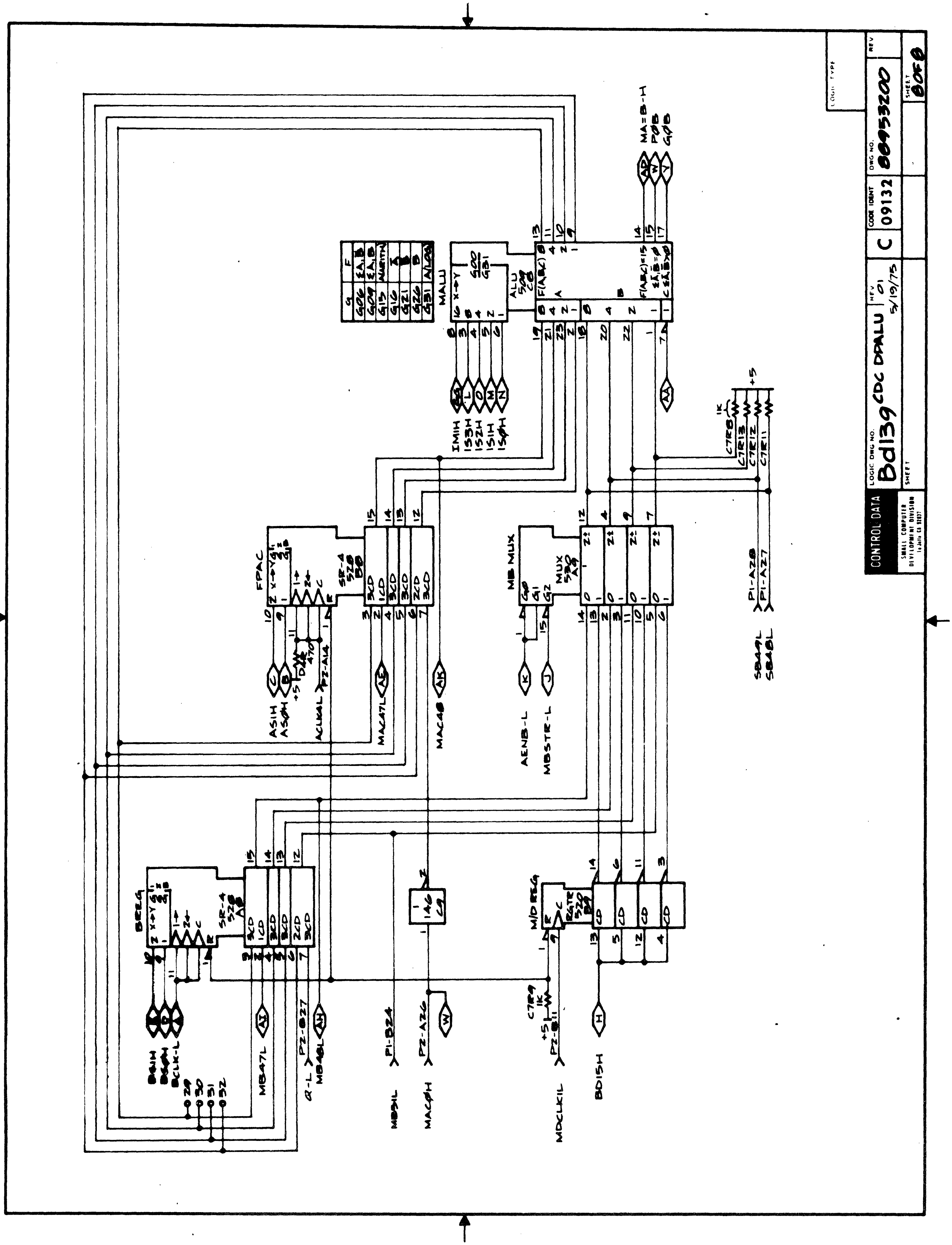
LOGIC TYPE



MALU
FPAC
MB MUX
BREG
M/D REG
M/D MUX
CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
La Jolla CA 92037
LOGIC DWG NO.
Bd139
CDC DPALU
REV
01
5/19/75
C
CODE IDENT
09132
DWG NO.
88953200
SHEET
5 OF 8
LOGIC TYPE
REV


CONTROL DATA
SMALL COMPUTER DIVISION LOS ANGELES
LOGIC DWG NO. Bd139 CDC DFALU
REV. 01 5/19/75
CODE IDENT 09132
DWG NO. 88953200
REV
SHEET 6 OF 8
LOGIC TYPE
MALU
ALU 524
FABG
MA=B-H
PZB
GZB
IMIH
ISSH
ISZH
ISIH
ISØH
FPAC
SE-4 528 B4
MB MUX
MUX 530 A5
AENB-L
MBSTR-L
M/D REG
REGTR 520 B5
M/D MUX
MUX 189 D5
MDCLKBL
BDISH
DP-H
DATAOUTSH
PZ-B9
PZ-B8
PZ-B7
PZ-B6
DATA7
DATA6
DATA5
DATA4
ASIH
ASØH
ACLKBL
MA39
MA44
MA40
MA43
BSIH
BSØH
BCLK-L
MB39
MB44
MB40
MB43
+5


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
Bd139 CDC DPALU
C
09132
88953200
7 OF 8
LOGIC TYPE


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
Bd139 CDC DPALU
C
09132
89953200
5/19/75
MALU
FPAC
BREG
MB MUX
MDREG
ALU
MUX
SR-4
MA=B-H
P0B
G0B
IMIH
I53H
I52H
I51H
I50H
ASIH
AS0H
ACLKAL
MAC47L
MAC48
AENB-L
MBSTR-L
SBA9L
SBA8L
P1-AZ8
P1-AZ7
C7R8
C7R13
C7R12
C7R11
B0IH
B00H
BCLK-L
MB47L
Q-L
MBA8L
P2-B27
P1-B24
P2-AZ6
MB31L
MAC0H
C7R9
P2-B11
MDCLKIL
BDISH

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7
A	X		X			
B	X	X				
C	X		X			
D	X	X				
E	X		X			
F	X	X	X	X	X	X
G	X		X			
H	X		X			
I	X	X				
J	X		X	X		
K	X		X	X		
L	X		X			
M	X	X	X	X		X
N		X	X			
O		X		X		
P		X			X	
Q		X	X	X		
R	X	X	X			
S		X			X	
T		X			X	
U		X				X
V		X			X	X
W		X			X	X
X			X			X
Y			X			X
Z			X			X
AA			X	X		
BB			X		X	X
CC			X		X	

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7
DD			X		X	
EE			X	X		
FF			X	X		
GG			X		X	
HH			X		X	
II			X	X		
JJ			X			X
KK			X			X
LL			X			X
MM			X			X
NN			X	X		
OO			X	X		
PP			X	X		X
QQ		X	X			
RR				X		X
SS				X	X	
TT				X		X
UU				X		X
VV					X	X
WW						
XX					X	X
YY					X	X
ZZ					X	X
AB					X	X
AC					X	X
AD					X	X
AE						
AF	X				X	
AG		X				X

NOTES: UNLESS OTHERWISE SPECIFIED

DR DETACHED LIST

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

LOGIC TITLE

LOGIC DWG. NO. | REV. 01 5/19/75

PWA NO. | PWB NO.

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037

FIRST USED ON		
DR		
CHK	R.G.	5/21/75
ENGR		
MFG		
APPD		

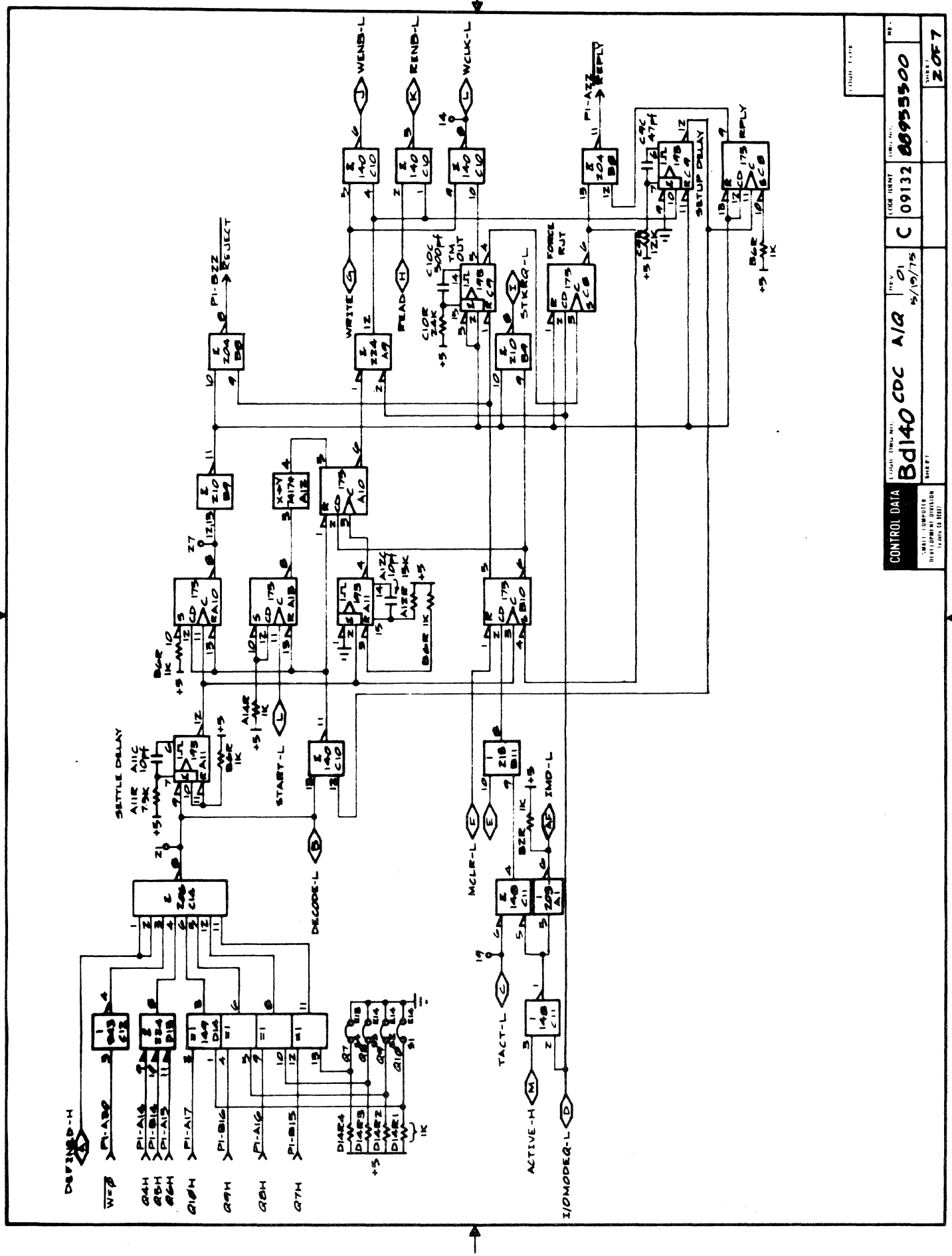
TITLE: Bd140 CDC A/Q

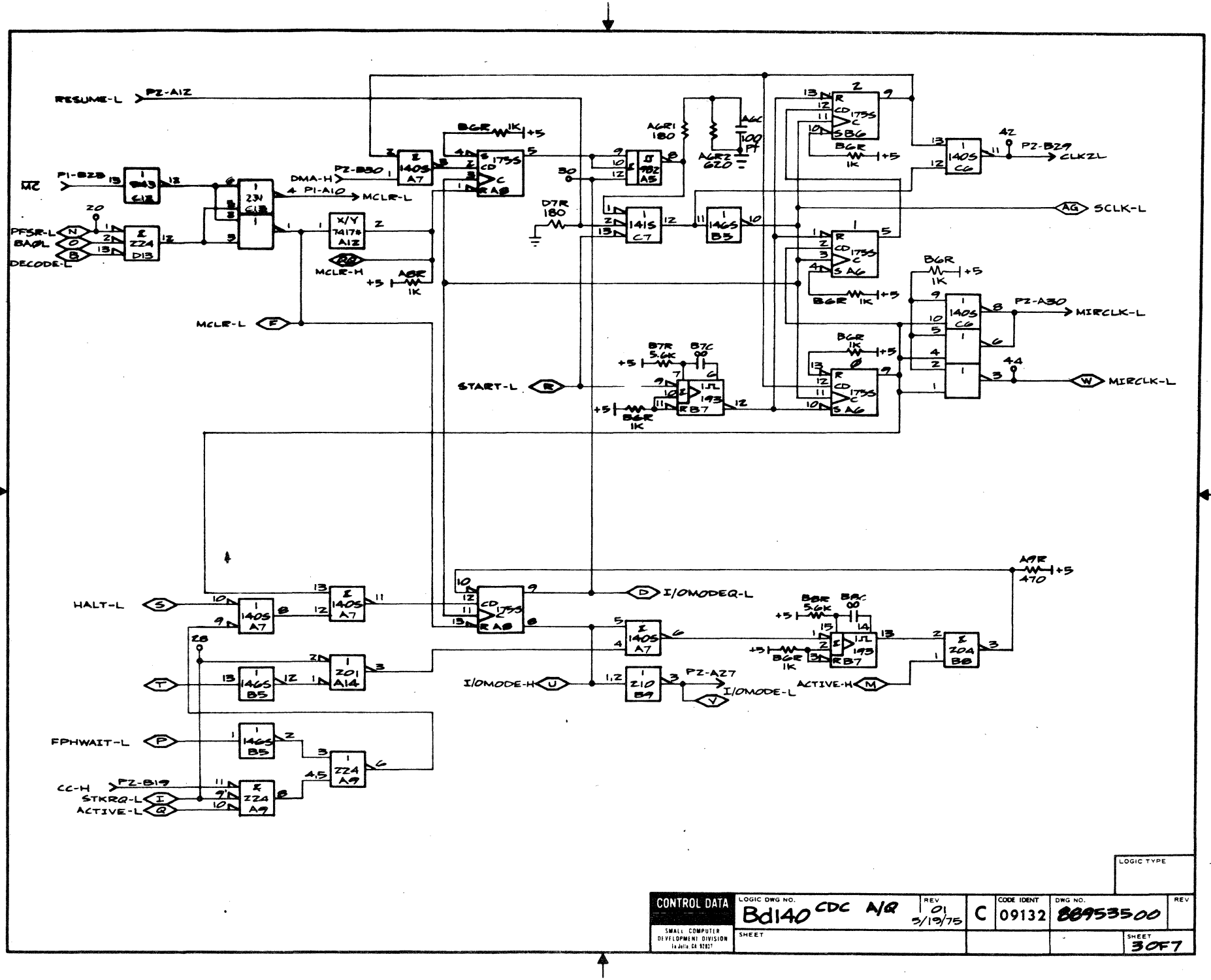
CARD POS.

LOGIC TYPE

C | CODE IDENT NO. 09132 | DRAWING NUMBER 88953500

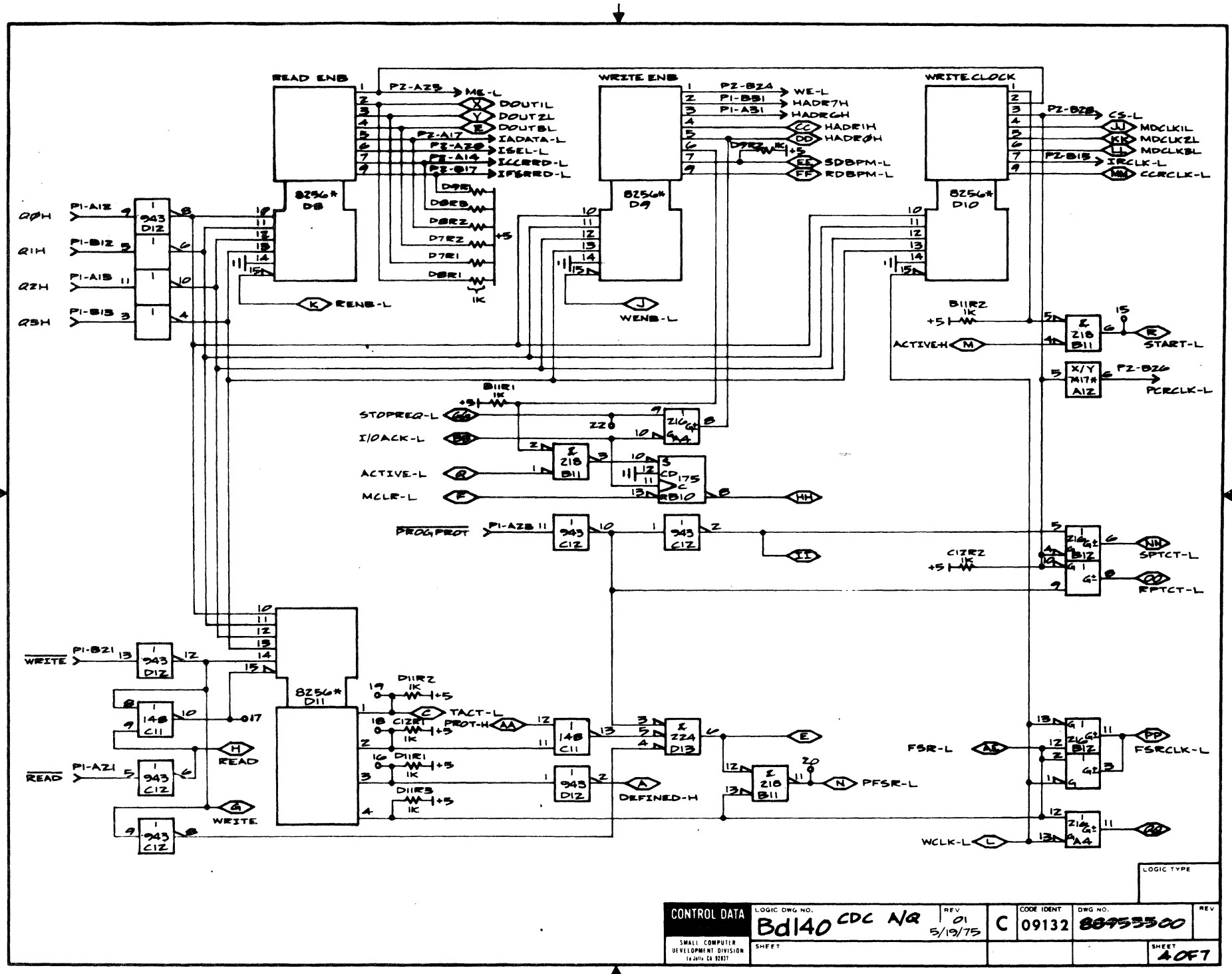
SHEET 1 OF 7


CONTROL DATA
Bd140 CDC A/Q
REV. 01 5/10/75
C
CODE IDENT 09132
PART NO. 88933500
SHEET 2 OF 7
WENB-L
RENB-L
WCLK-L
PI-AZZ REPLY
PI-BZZ REJECT
WRITE
READ
STKEQ-L
FORCE RJT
TM OUT
SETUP DELAY
REPLY
SETTLE DELAY
START-L
DECODE-L
MCLR-L
IMD-L
TACT-L
ACTIVE-H
I/OMODEQ-L
PI-A16
PI-A15
PI-A14
PI-A13
PI-A17
PI-B16
PI-A16
PI-B15



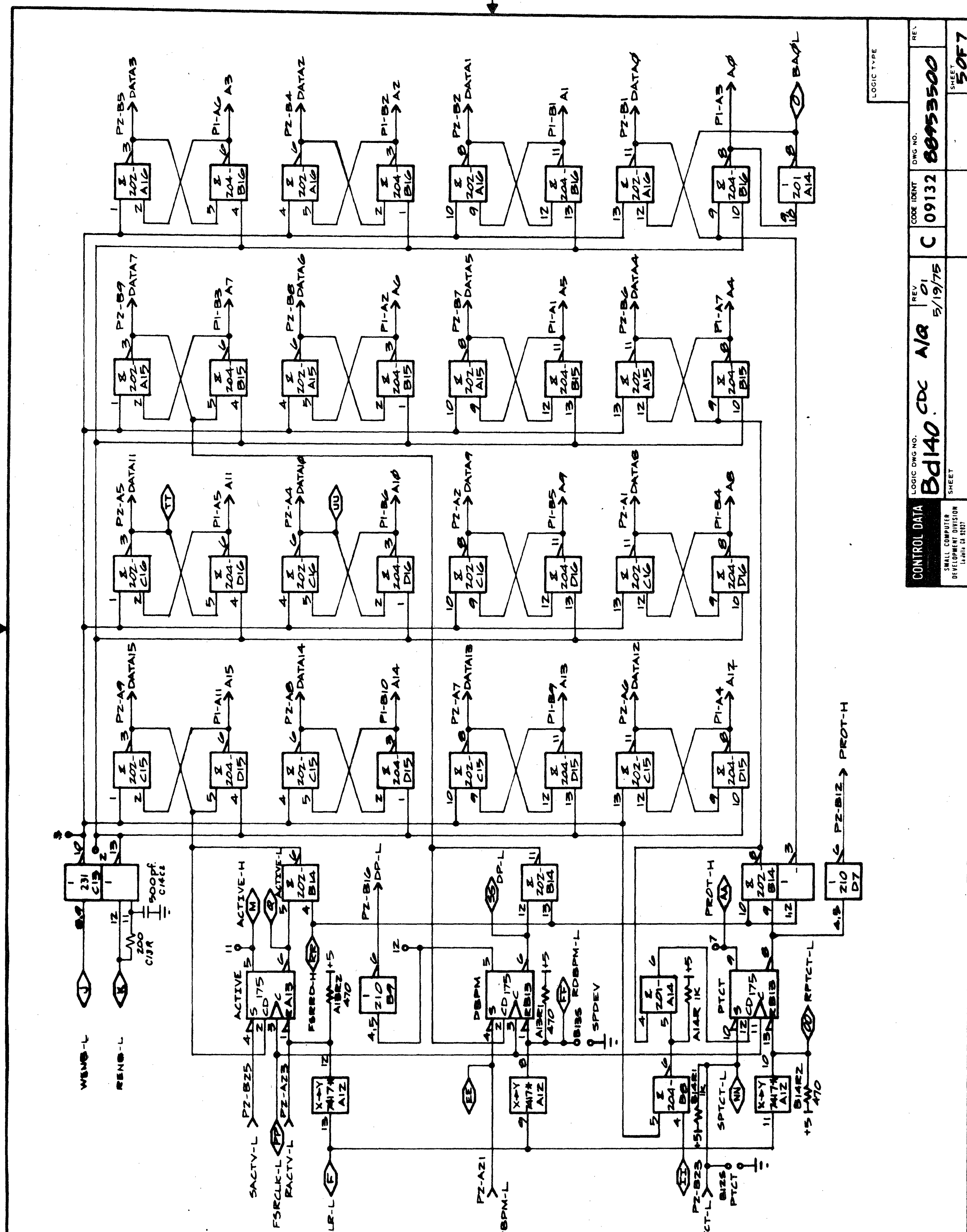
CONTROL DATA	LOGIC DWG NO. Bd140 CDC A/Q	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88953500	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	SHEET				SHEET 3OF7	

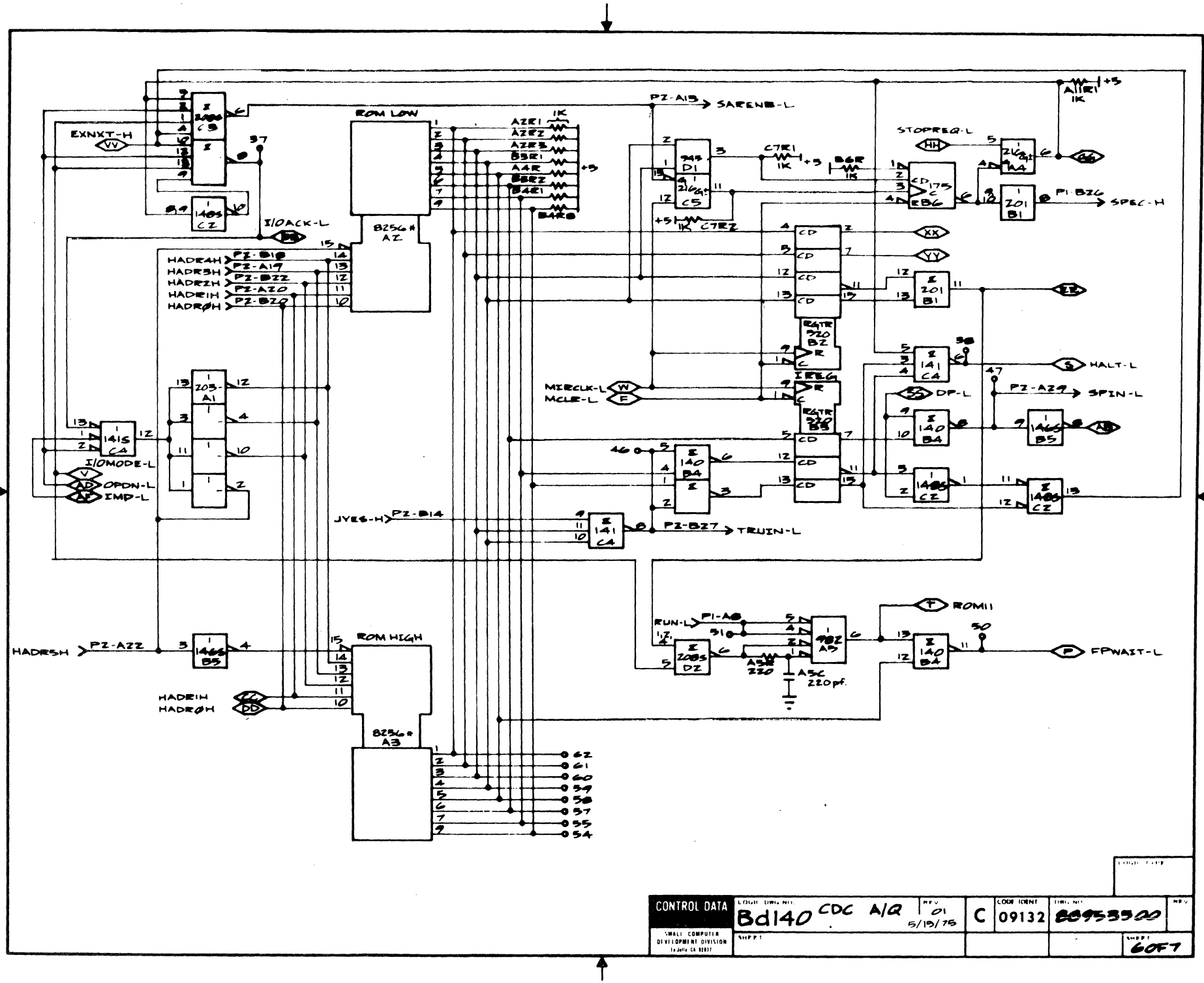
LOGIC TYPE



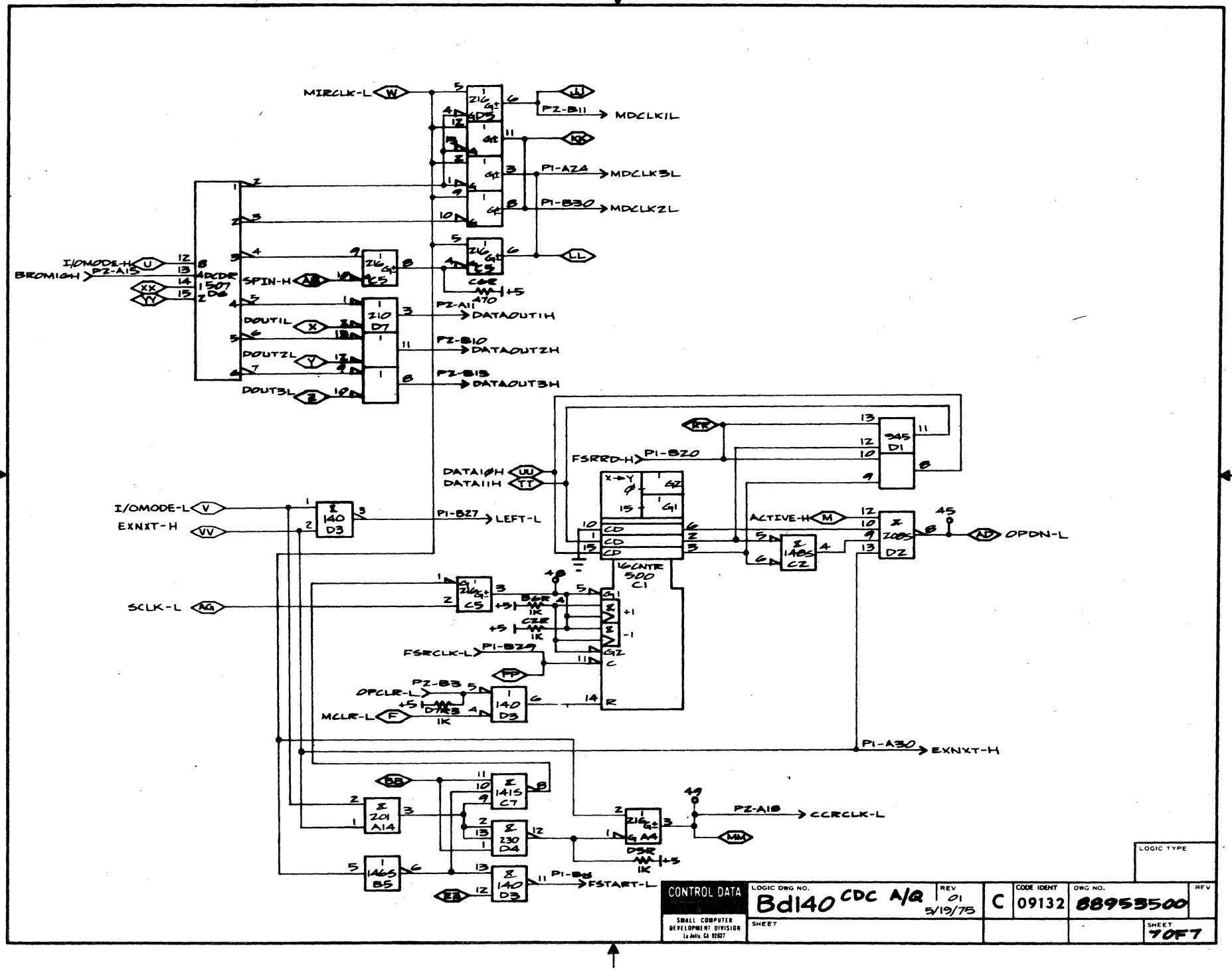
CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	Bd140 CDC N/Q	01 5/19/75	C	09132	88955500	
	SHEET					SHEET 4 OF 7

LOGIC TYPE


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO.
Bd140 CDC A/Q
REV 01 5/19/75
C
CODE IDENT 09132
DWG NO. 88953500
SHEET 5 OF 7
LOGIC TYPE



CONTROL DATA	LOGIC DWG NO. Bd140 CDC A/Q	REV 01 5/15/75	C	CODE IDENT 09132	DWG NO. 60953500	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	SHEET				SHEET 6OF7	


MIRCLK-L
PZ-B11
MDCLK1L
PI-A24
MDCLK3L
PI-B30
MDCLK2L
I/OMODE-H
PZ-A15
BROMIGH
SPIN-H
CCRE
470
+5
PZ-A11
DATAOUT1H
PZ-B10
DATAOUT2H
PZ-B15
DATAOUT3H
DOUT1L
DOUT2L
DOUT3L
DATAIØH
DATAIIH
FSRRD-H
PI-B20
I/OMODE-L
EXNXT-H
PI-B27
LEFT-L
ACTIVE-H
OPDN-L
16CNTR
500
C1
SCLK-L
BCRE
1K
CZRE
FSRCLK-L
PI-B29
OPCLR-L
PZ-B3
MCLR-L
PI-A30
EXNXT-H
PZ-A18
CCRCLK-L
DSR
FSTART-L
CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
La Jolla, CA 92037
LOGIC DWG NO.
Bd140 CDC A/Q
REV 01
5/19/75
C
CODE IDENT 09132
DWG NO. 88953500
REV
LOGIC TYPE
SHEET
SHEET 7OF7

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8
A	X		X				
B	X		X				
C	X		X				
D		X	X				
E		X					X
F		X	X		X	X	X
G		X					X
H		X	X				
I		X	X				
J		X					X
K		X			X		
L			X		X		
M			X		X		
N			X			X	
O			X		X		
P			X		X		
Q			X		X		
R		X	X				
S		X	X				
T				X	X		
U				X	X		
V				X	X		
W				X	X		
X				X	X		
Y				X	X		
Z				X	X		
AA				X			
BB				X		X	
CC				X		X	
DD				X		X	
EE				X		X	
FF				X		X	

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8
GG				X		X	X
HH				X		X	
II				X			
JJ				X			X
KK				X			X
LL				X			X
MM				X			
NN				X			X
OO				X			X
PP				X			
QQ				X			X
RR					X	X	
SS					X		X
TT					X		X
UU					X		X
VV					X	X	
WW					X	X	
XX					X	X	
YY		X			X		
ZZ							X
AB						X	X
AC							
AD							X
AE					X	X	
AF						X	
AG		X					X
AH		X					X
AI		X					X
AJ					X		X
AK			X			X	
AL			X		X	X	

NOTES UNLESS OTHERWISE SPECIFIED

DR DETACHED LIST

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

CODE TITLE

CODE IDENT NO

REV 01 5/19/75

PWA NO PWB NO

CONTROL DATA SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037

DR		
CHK		
ENGR	LPG	5/21/75
MFG		
APPD		

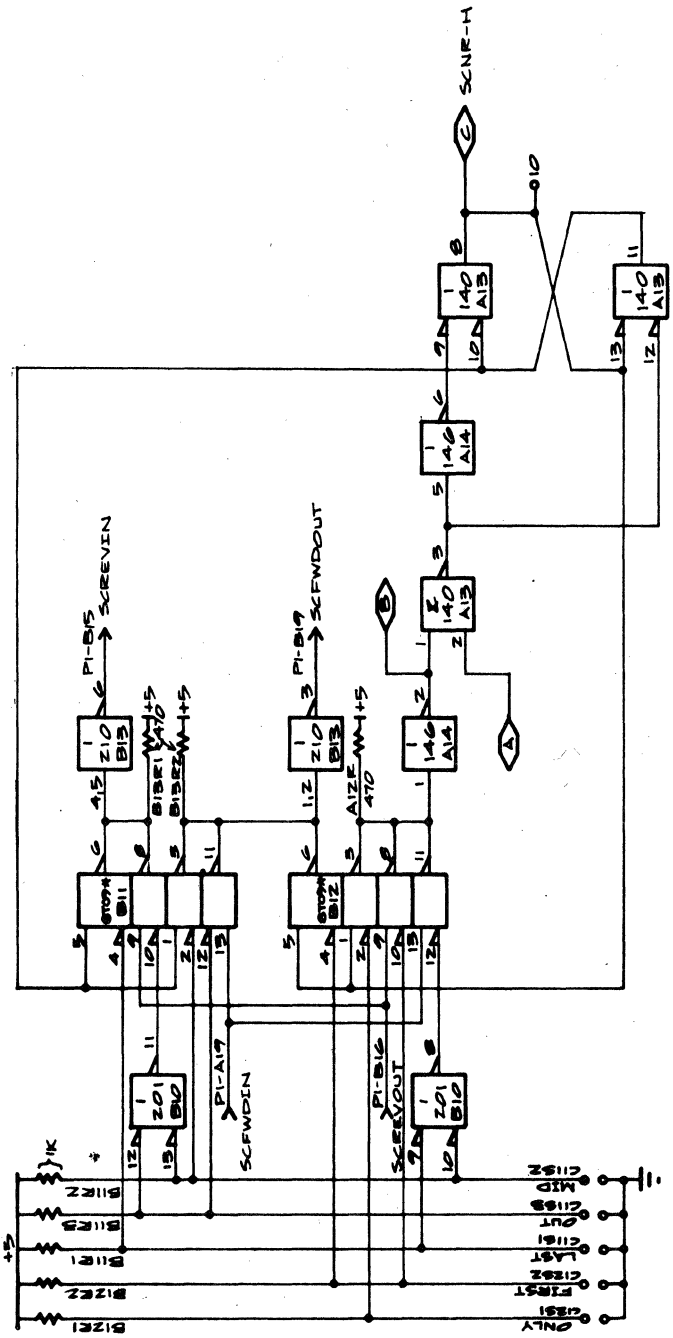
TITLE Bd141 CDC DSA

CARD PTN

CODE TYPE

C 09132 88953800

SHEET 1 OF 8



CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION, La Jolla, CA 92037

TITLE: CDC DSA Bd141

CARD POS | LOGIC TYPE

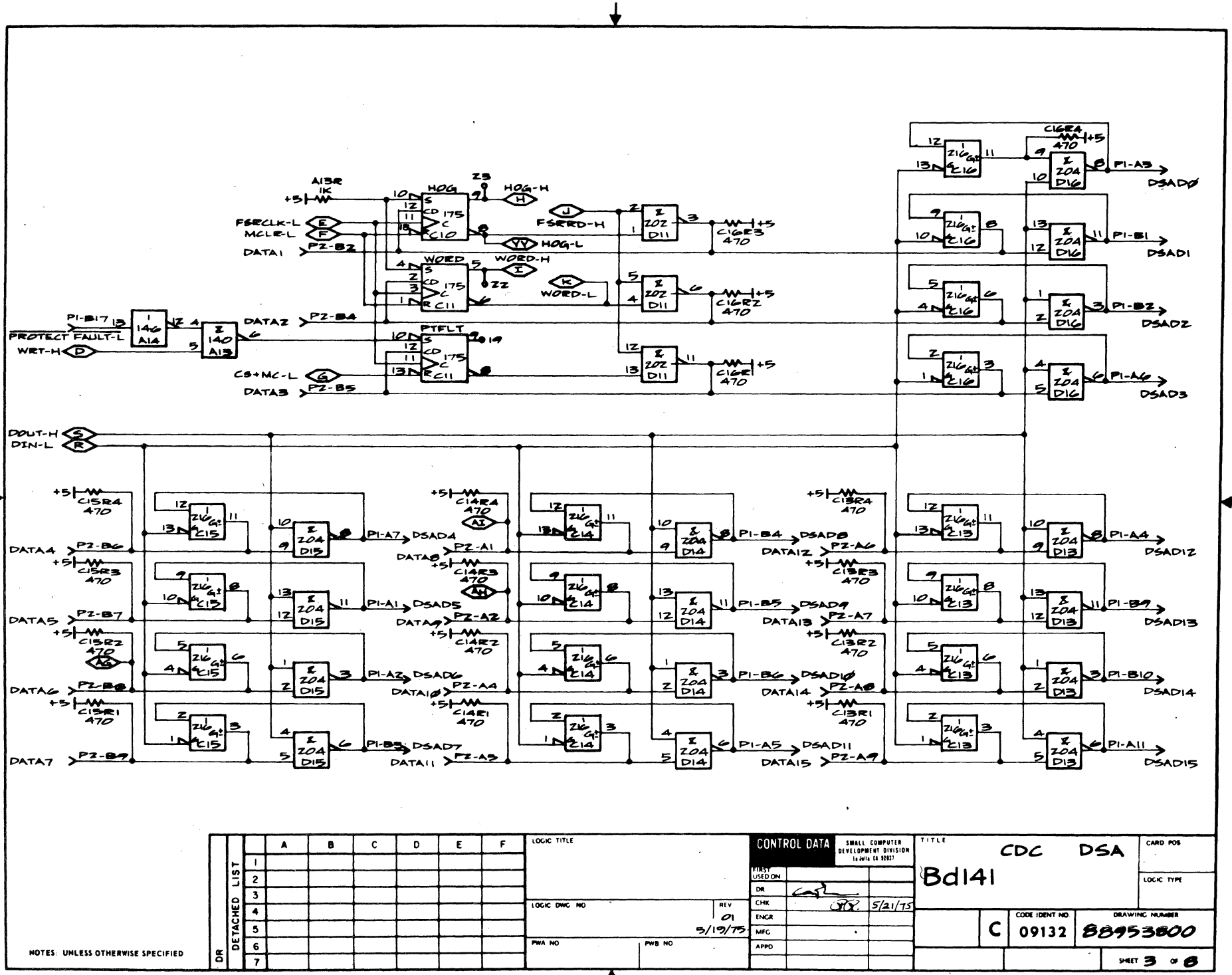
CODE IDENT NO. 09132 | DRAWING NUMBER 88953800 | SHEET 2 OF 8

DR 5/21/75 | CHK | ENGR | MFG | APPD

LOGIC TITLE | LOGIC DWG. NO | REV. 01 5/19/75 | PWB NO | PWA NO

DETACHED LIST

NOTES: UNLESS OTHERWISE SPECIFIED



NOTES: UNLESS OTHERWISE SPECIFIED

DR	DETACHED LIST	A	B	C	D	E	F
	1						
	2						
	3						
	4						
	5						
	6						
	7						

LOGIC TITLE

LOGIC DWG NO | REV 01 | 5/19/75

PWA NO | PWB NO

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION, La Jolla CA 92037

FIRST USED ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

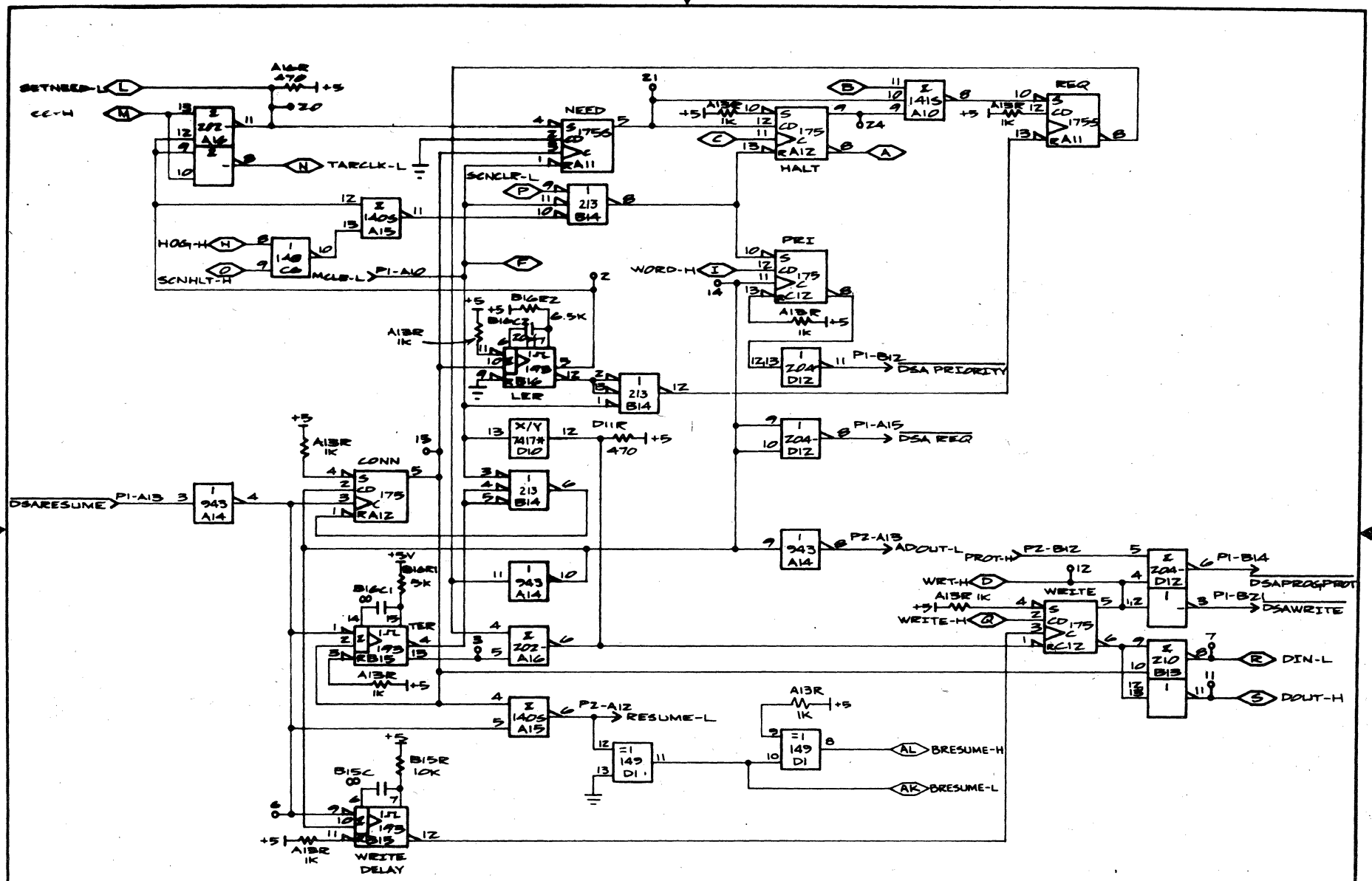
TITLE: CDC DSA Bd141

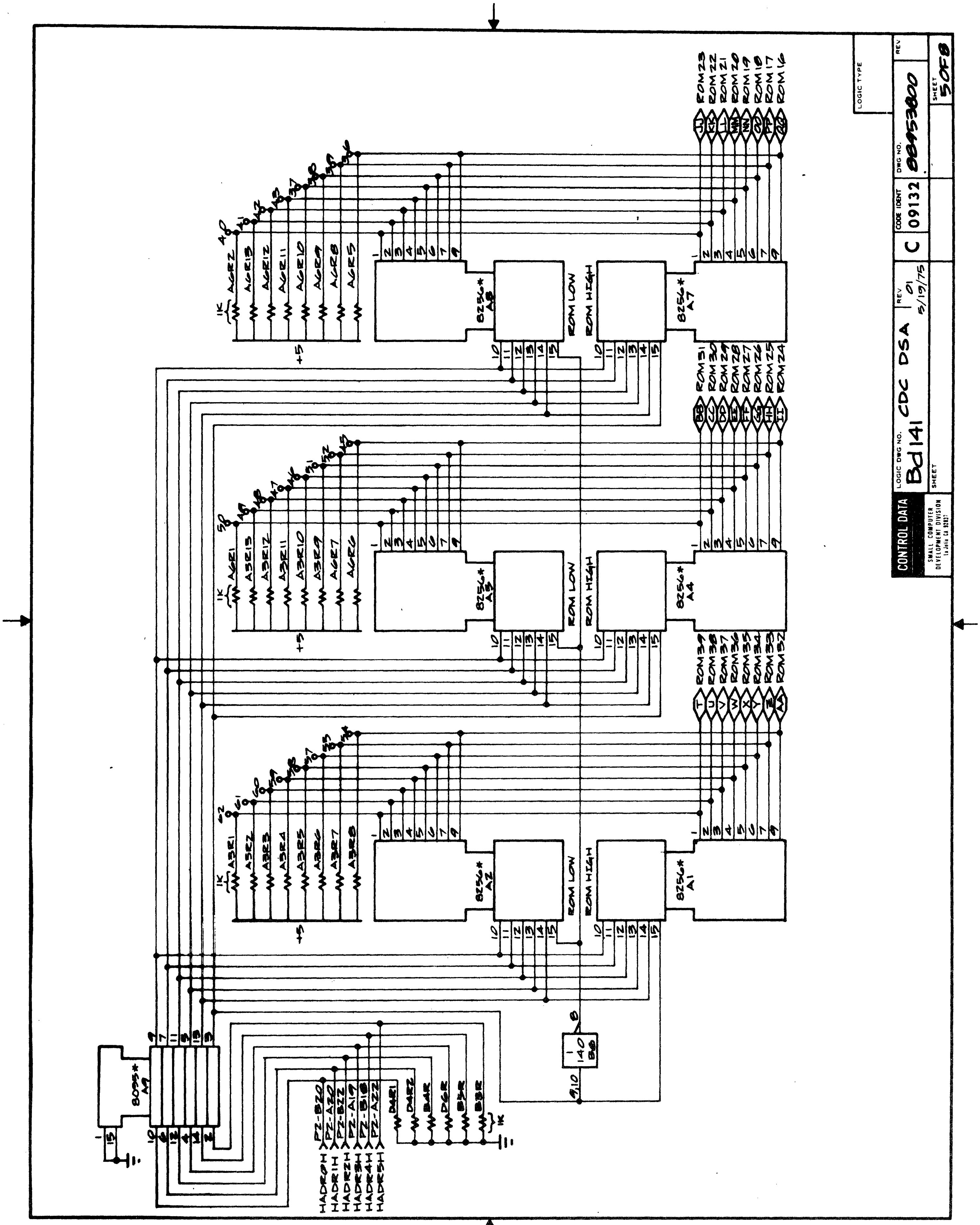
CARD POS

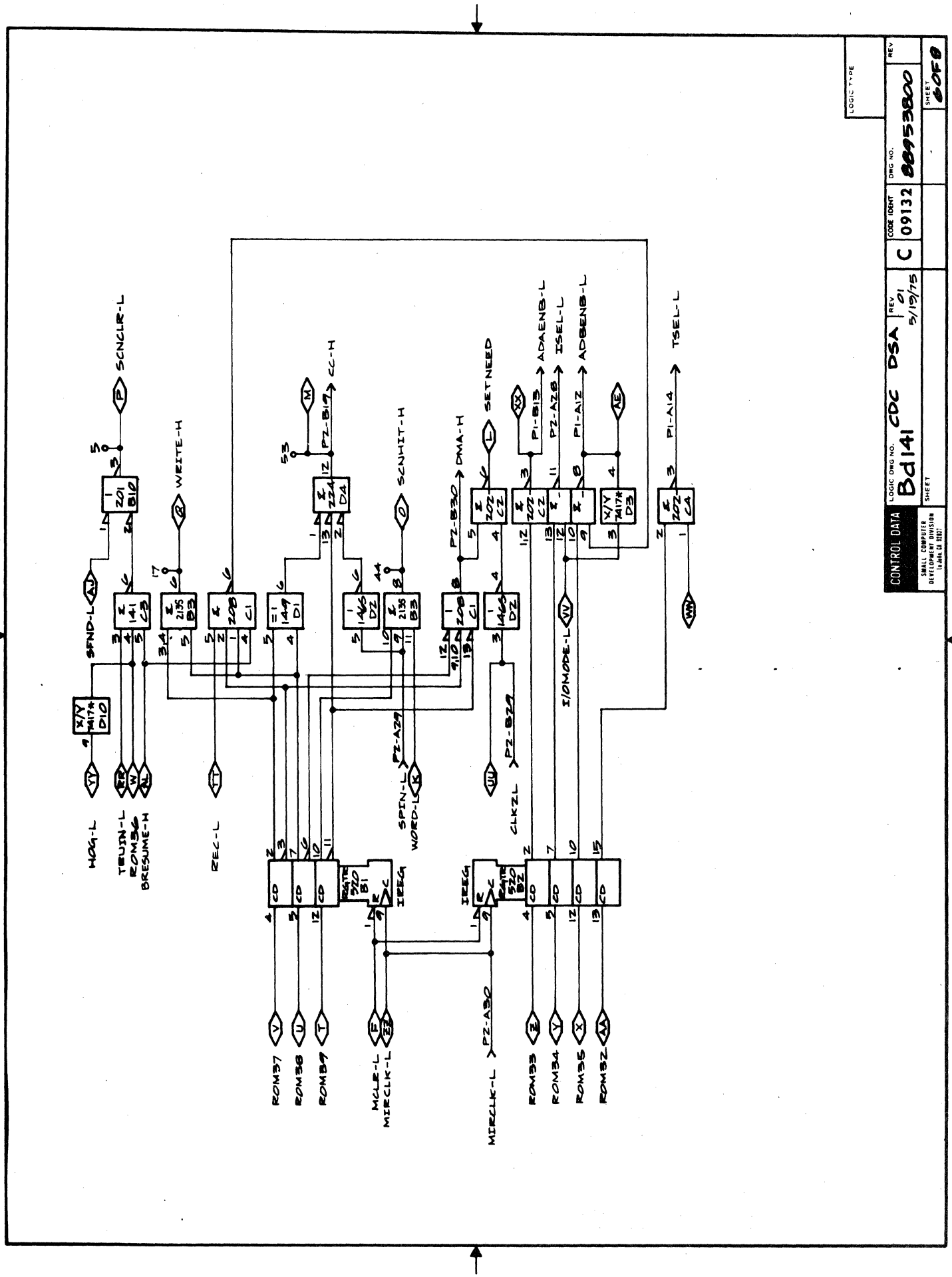
LOGIC TYPE

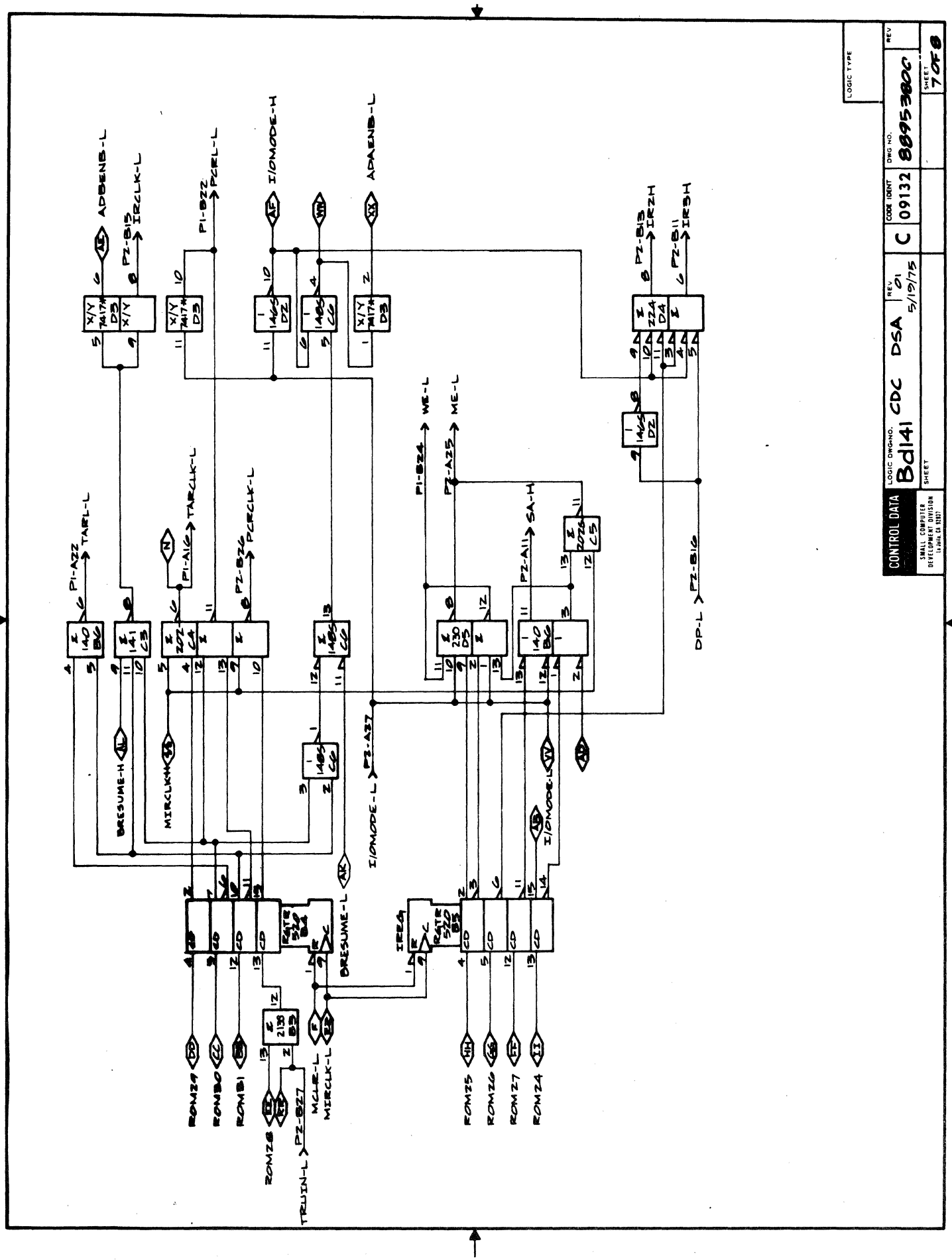
C | CODE IDENT NO. 09132 | DRAWING NUMBER 88953800

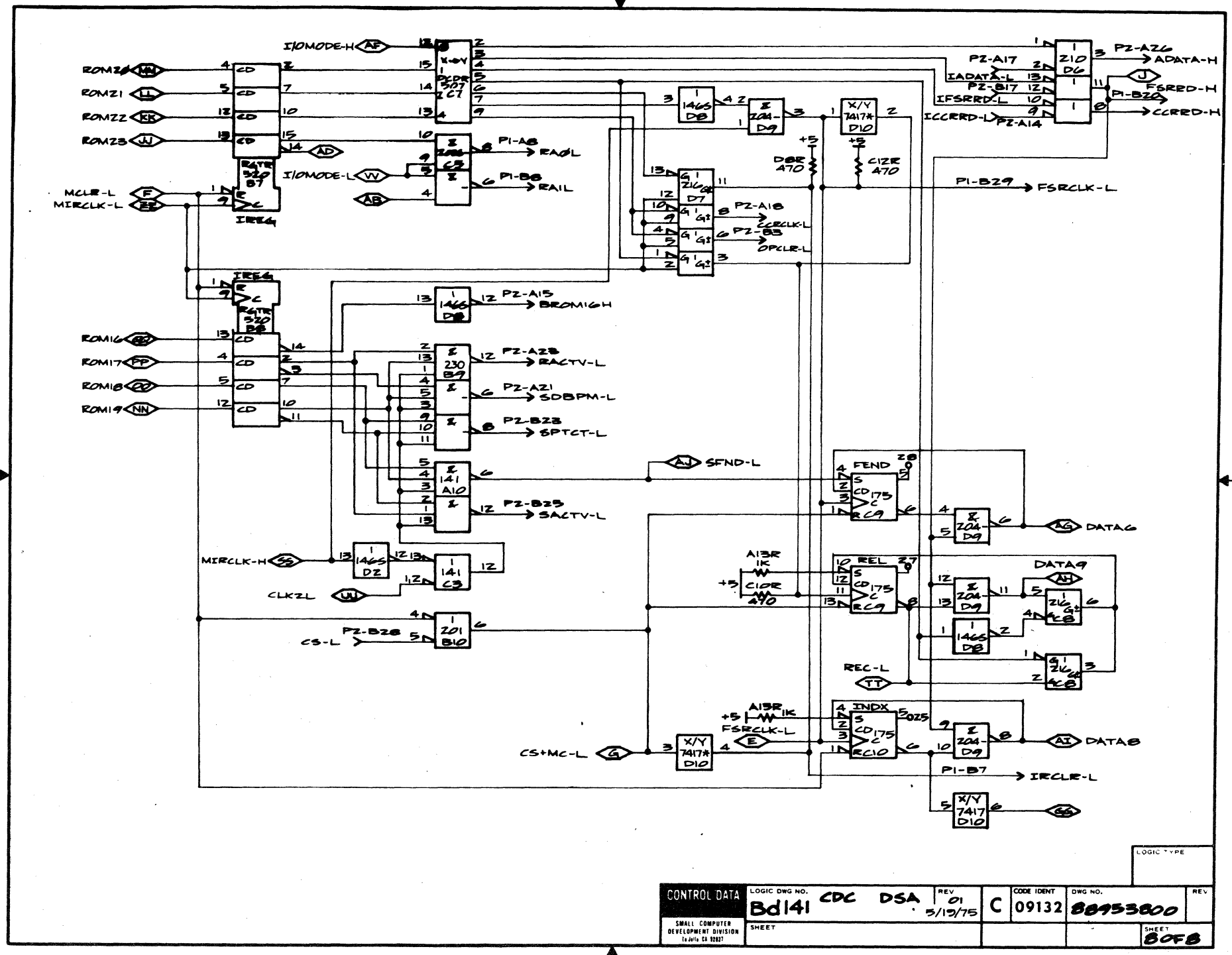
SHEET 3 OF 8


SETNEED-L
CE-H
TARCLK-L
NEED
SCNCLR-L
HOG-H
SCNHLT-H
MCLR-L
P1-A10
HALT
REQ
PRI
WORD-H
P1-B12
DSA PRIORITY
LER
P1-A15
DSA REQ
CONN
DSARESUME
P1-A13
P2-A13
ADOUT-L
PROT-H
P2-B12
P1-B14
DSAPROGPROT
WRT-H
WRITE
WRITE-H
P1-B21
DSAWRITE
TER
P2-A12
RESUME-L
DIN-L
DOUT-H
BRESUME-H
BRESUME-L
WRITE DELAY
NOTES: UNLESS OTHERWISE SPECIFIED
DR
DETACHED LIST
LOGIC TITLE
LOGIC DWG NO
REV 01 5/19/75
PWA NO
PWB NO
CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037
FIRST USED ON
DR
CHK
ENGR
MFG
APPD
5/21/75
TITLE
CDC DSA Bd141
CARD POS
LOGIC TYPE
CODE IDENT NO. 09132
DRAWING NUMBER 88953800
SHEET 4 OF 8


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
CDC DSA
Bd141
C
09132
88953800
01
5/19/75
5 OF 8
8256* A1
8256* A2
8256* A3
8256* A4
8256* A7
8256* A8
8095* A9
140 B8
ROM LOW
ROM HIGH
HADR0H
HADR1H
HADR2H
HADR3H
HADR4H
HADR5H
+5
1K


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd1141 CDC DSA
REV 01 5/19/75
C
CODE IDENT 09132
DWG NO. 88953800
REV
SHEET 8 OF 8
LOGIC TYPE
SHEET
SCNCLR-L
WRITE-H
SFND-L
SCNHIT-H
CC-H
DMA-H
SETNEED
ADAENB-L
ISEL-L
ADBENB-L
TSEL-L
P1-A14
P1-A12
P1-A10
P1-B13
P1-A12
P1-A10
PZ-B30
PZ-B17
PZ-A29
PZ-B29
HOG-L
TBUIN-L
ROM36
BRESUME-H
REC-L
SPIN-L
WORD-L
CLKZL
I/OMODE-L
IREQ
MCLE-L
MIRCLK-L
PZ-A30
ROM37
ROM36
ROM35
ROM34
ROM33
ROM32


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION LA JOLLA, CA 92037
LOGIC DWGNO: Bd141 CDC DSA
REV 01 5/19/75
C
CODE IDENT 09132
DWG NO: 8895380C
REV
SHEET 7 OF 8
LOGIC TYPE
ADBENB-L
PZ-B15 IRZCLK-L
PI-BZZ PZREL-L
I/OMODE-H
ADAENB-L
PZ-B13 IRZH
PZ-B11 IRSH
PI-AZZ TAREL-L
PI-A16 TARCLK-L
PZ-B26 PCRCLK-L
PI-B24 WE-L
PZ-A25 ME-L
PZ-A11 SA-H
DP-L PZ-B16
BRESUME-H
MIRCLKH
I/OMODE-L PZ-A37
BRESUME-L
IREG4
ROMZ4 ROMZ5 ROMZ6 ROMZ7
ROMB0 ROMB1
ROMZ8
TRUIN-L PZ-B27
MCLR-L
MIRCLK-L



CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	Bd141 CDC DSA	01 5/19/75	C	09132	88953800	
	SHEET					SHEET 8 OF 8

LOGIC TYPE

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6
A	X	X			
B	X	X			
C	X	X			
D	X	X			
E	X		X		
F	X		X		
G	X		X		
H	X		X		
I	X			X	
J	X			X	
K	X			X	
L	X			X	
M	X				X
N	X				X
O	X				X
P	X				X
Q		X	X	X	X
R		X	X		
S		X			X
T		X	X	X	X
U		X	X	X	X
V		X	X	X	X
W		X	X	X	X
X		X	X	X	X
Y		X	X		
Z		X			X
AA		X	X	X	X
BB		X	X	X	X
CC		X	X	X	X
DD		X	X	X	X
EE		X	X	X	X
FF		X	X		
GG		X	X		
HH		X	X		
II		X	X		
JJ		X		X	
KK		X			X
LL		X	X	X	X

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6
MM		X	X	X	X
NN		X	X	X	X
OO		X	X		
PP		X	X		
QQ		X	X		
RR		X	X	X	X
SS		X	X	X	X
TT		X	X	X	X
UU		X	X	X	X
VV		X	X		
WW		X	X		
XX		X	X		
YY		X	X		
ZZ		X	X	X	X
AB		X	X	X	X
AC			X	X	
AD			X	X	
AE			X	X	
AF			X	X	
AG			X	X	
AH			X		X
AI			X		X
AJ			X	X	
AK			X	X	
AL			X	X	
AM			X		X
AN			X		X
AO			X	X	
AP				X	X
AQ				X	X
AR				X	X
AS				X	X
AT				X	X
AU			X		X
AV			X		X

NOTES: UNLESS OTHERWISE SPECIFIED

DR

DETACHED LIST

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

LOGIC TITLE

LOGIC DWG. NO. | REV 01 5/19/75

PWA NO. | PWB NO.

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037

FIRST USED ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

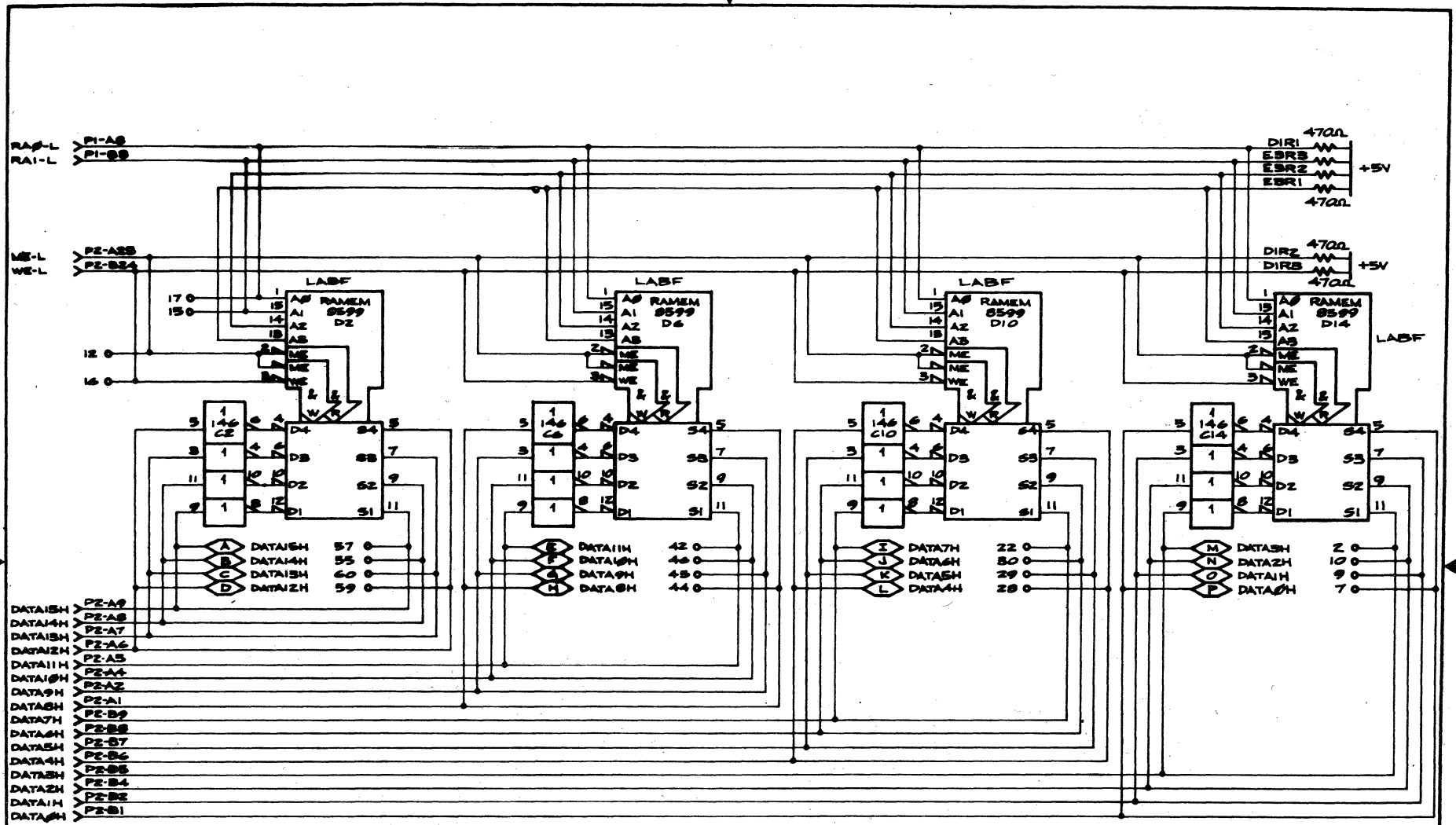
TITLE: Bd142 CDC ADDALU ADDRESS ALU

CARD POS. 23

LOGIC TYPE

C | CODE IDENT NO. 09132 | DRAWING NUMBER 88954100

SHEET 1 OF 6



NOTES: UNLESS OTHERWISE SPECIFIED

DR DETACHED LIST

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

LOGIC TITLE

LOGIC DWG NO | REV. 01 5/13/75

PWA NO | PWB NO.

CONTROL DATA SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037

FIRST USED ON		
DR		5/15/74
CHK		5/21/75
ENGR		
MFG		
APPD		

TITLE: Bd142 CDC ADDALU ADDRESS ALU

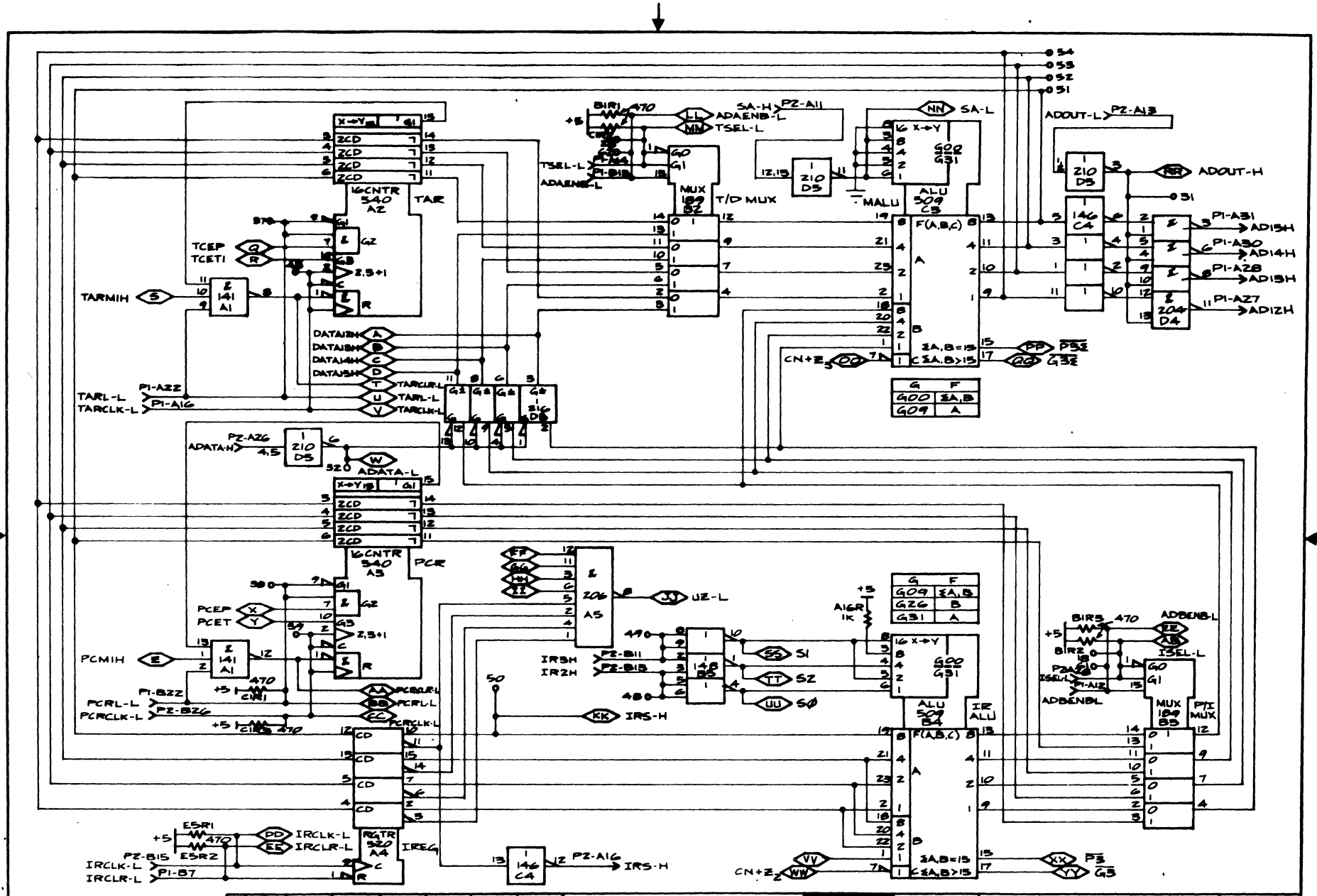
CARD POS 23

LOGIC TYPE

CODE IDENT NO. C 09132

DRAWING NUMBER 88954100

SHEET 2 OF 6



	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

DR — DETACHED LIST

NOTES UNLESS OTHERWISE SPECIFIED

LOGIC TITLE

LOGIC DWG NO — REV 01 — 5/19/75

PWA NO — PWB NO

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037

FIRST LAYOUT ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

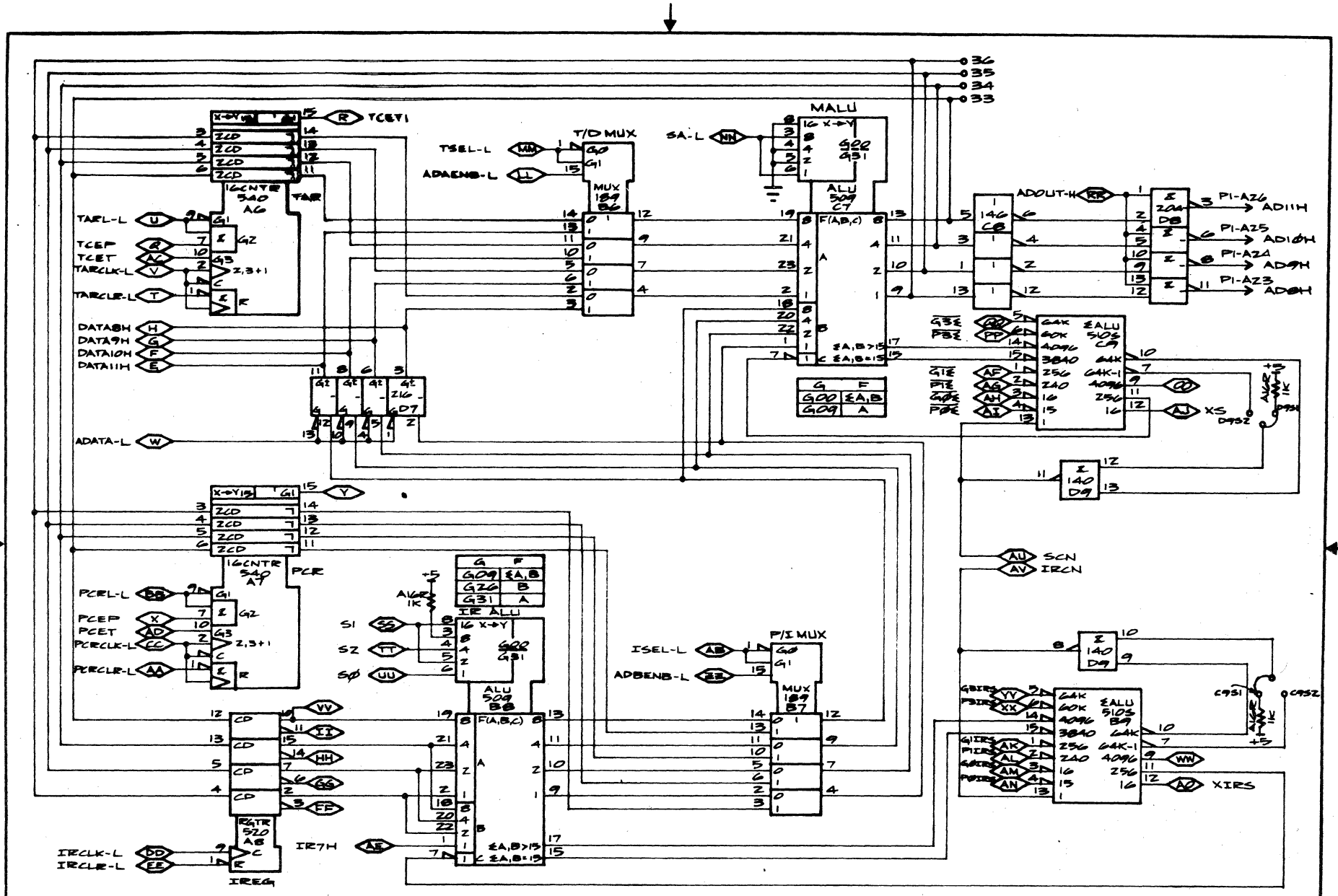
TITLE: Bd142 CDC ADDALU ADDRESS ALU

CARD POS 23 — LOGIC TYPE

CODE IDENT NO: C 09132

DRAWING NUMBER: 88954100

SHEET 5 OF 6



NOTES: UNLESS OTHERWISE SPECIFIED

DR	DETACHED LIST	A	B	C	D	E	F
	1						
	2						
	3						
	4						
	5						
	6						
	7						

LOGIC TITLE

LOGIC DWG NO | REV. 01 5/19/75

PWA NO | PWB NO.

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION, La Jolla CA 92037

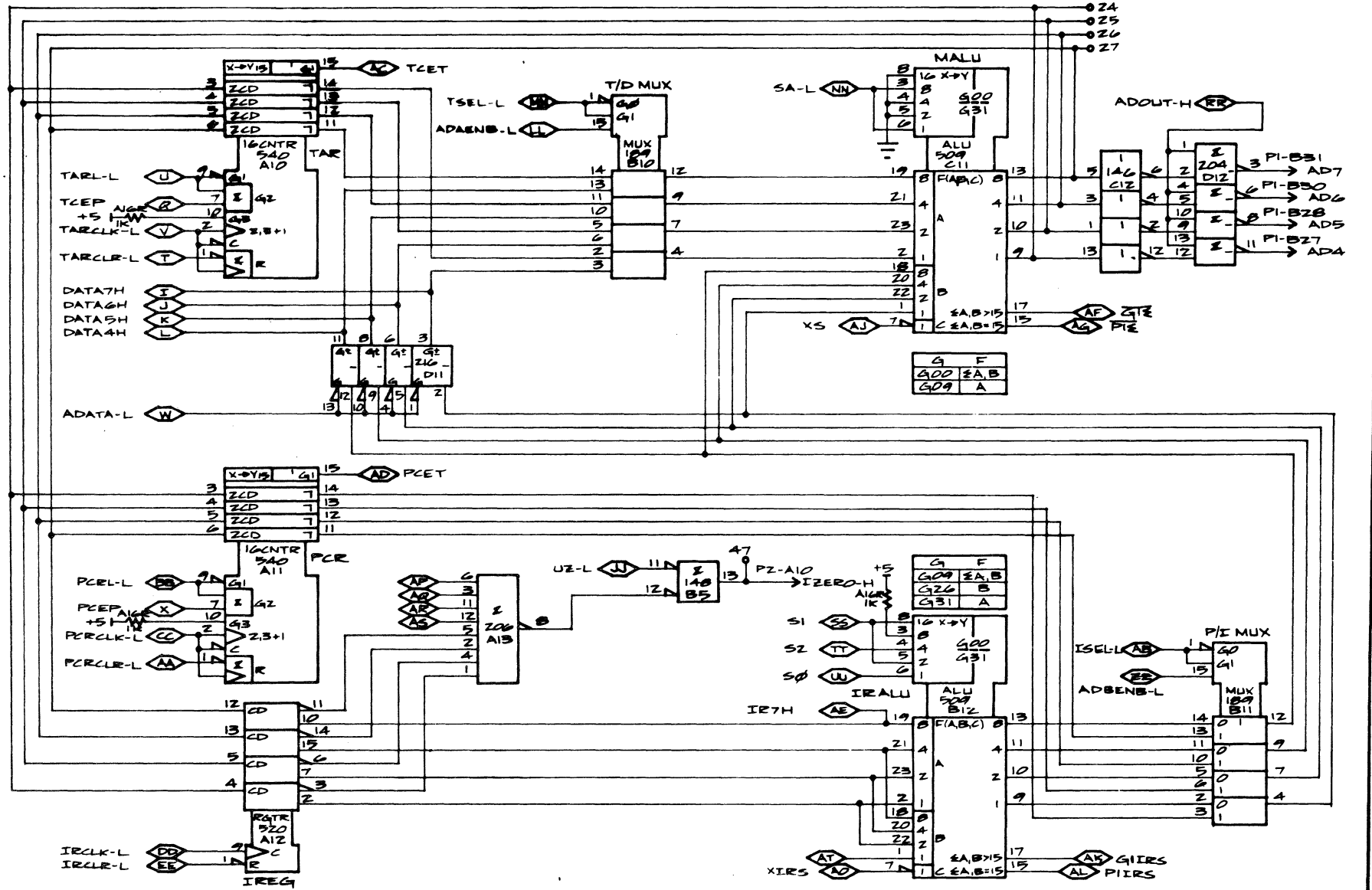
FIRST USED ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

TITLE: Bd142 CDC ADDALU ADDRESS ALU

CARD POS 23 | LOGIC TYPE

C | CODE IDENT NO. 09132 | DRAWING NUMBER 88954100

SHEET 4 OF 6



G	F
G00	ΣA,B
G09	A

G	F
G09	ΣA,B
G26	B
G31	A

DR	DETACHED LIST	A	B	C	D	E	F
		1					
		2					
		3					
		4					
		5					
		6					
		7					

NOTES. UNLESS OTHERWISE SPECIFIED

LOGIC TITLE

LOGIC DWG NO | REV 01 | 5/19/75

PWA NO | PWB NO

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION (La Jolla CA 92037)

FIRST USED ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

TITLE: Bd142 CDC ADDALU ADDRESS ALU

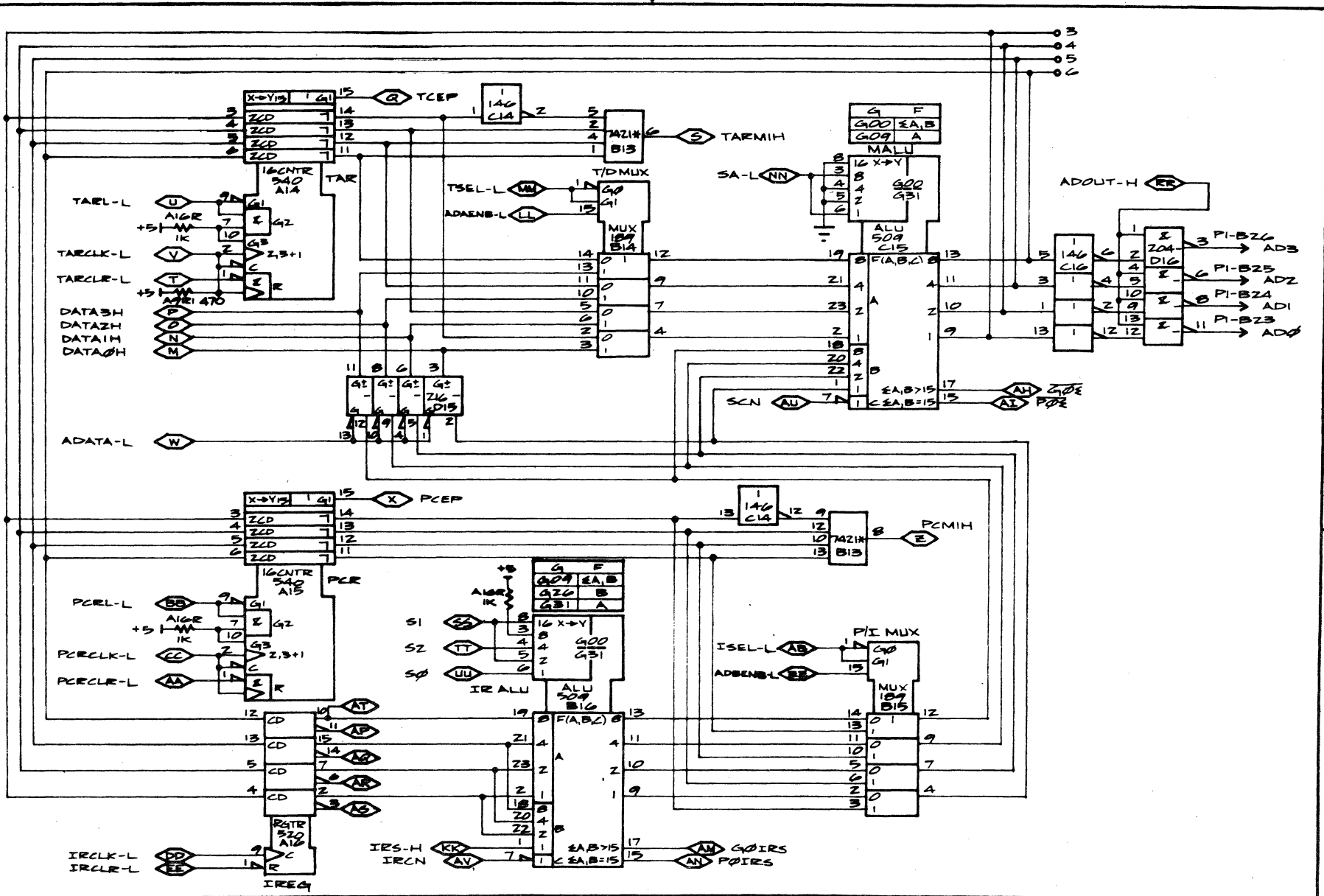
CARD POS 23

LOGIC TYPE

CODE IDENT NO: C 09132

DRAWING NUMBER: 88954100

SHEET 5 OF 6



NOTES: UNLESS OTHERWISE SPECIFIED

DR	DETACHED LIST	A	B	C	D	E	F
	1						
	2						
	3						
	4						
	5						
	6						
	7						

LOGIC TITLE

LOGIC DWG NO | REV 01 5/19/75

PWA NO | PWB NO

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION, La Jolla, CA 92037

FIRST USED ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

TITLE: Bd142 CDC ADDALU ADDRESS ALU

CARD POS 23 — LOGIC TYPE

C | CODE IDENT NO. 09132 | DRAWING NUMBER 88954100

SHEET 6 OF 6

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6
A	X	X	X	X	X
B					
C	X		X	X	
D					
E	X	X	X	X	X
F	X	X	X	X	X
G	X	X	X	X	X
H	X	X	X	X	X
I		X	X		
J		X		X	
K		X		X	
L		X		X	
M		X		X	
N		X		X	
O		X			X
P	X	X	X	X	X
Q	X	X	X	X	X
R			X	X	
S			X	X	
T				X	X
U				X	X
V				X	X
W			X	X	
X			X	X	X
Y				X	X
Z	X				X

NOTES: UNLESS OTHERWISE SPECIFIED

DR DETACHED LIST

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

LOGIC TITLE

LOGIC DWG NO | REV 01 5/19/75

PWA NO | PWB NO

CONTROL DATA SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037

FIRST USED ON		
DR	Carl	
CHK	[illegible]	5/21/75
ENGR		
MFG		
APPD		

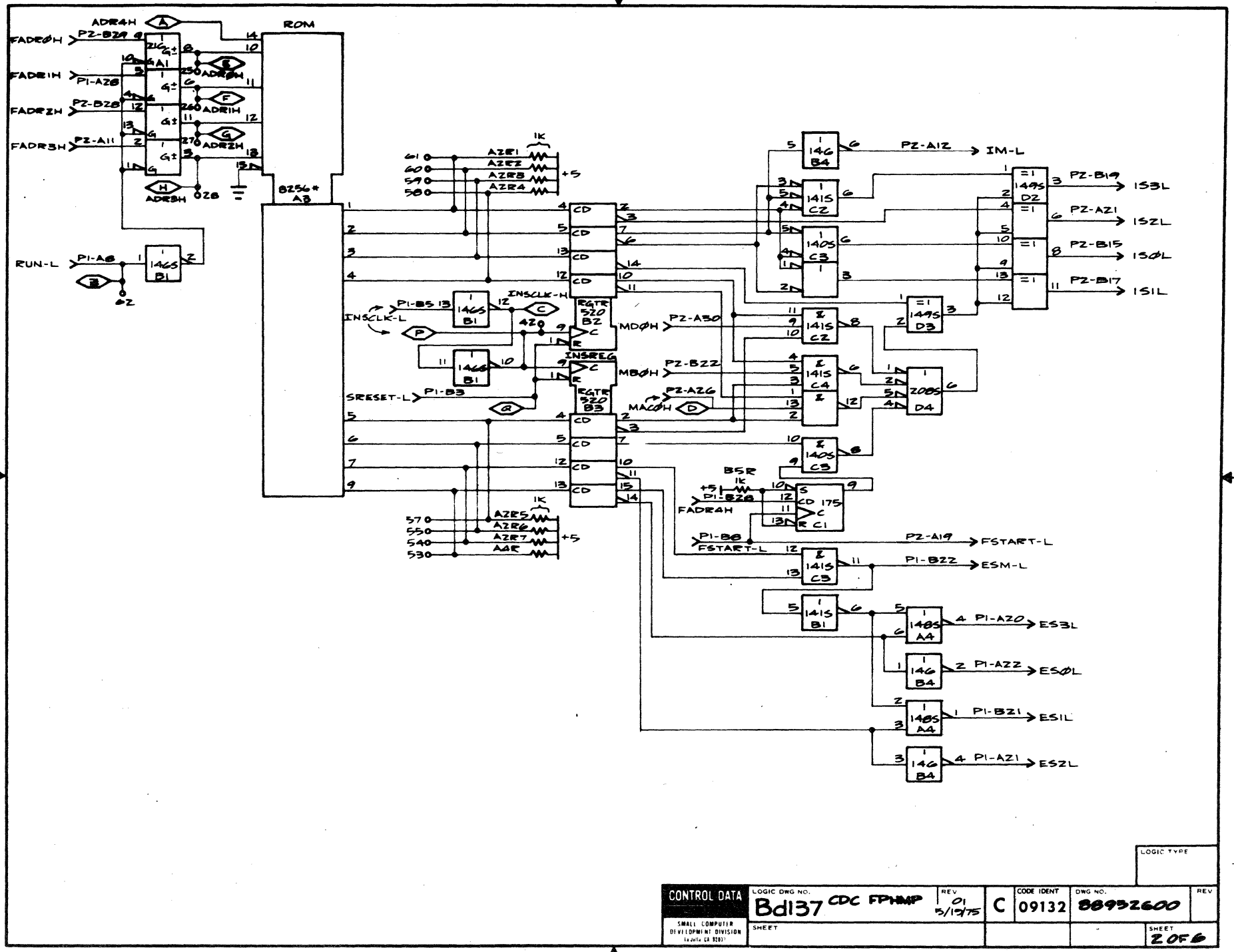
TITLE: Bd137 CDC FPHMP

CARD POS

LOGIC TYPE

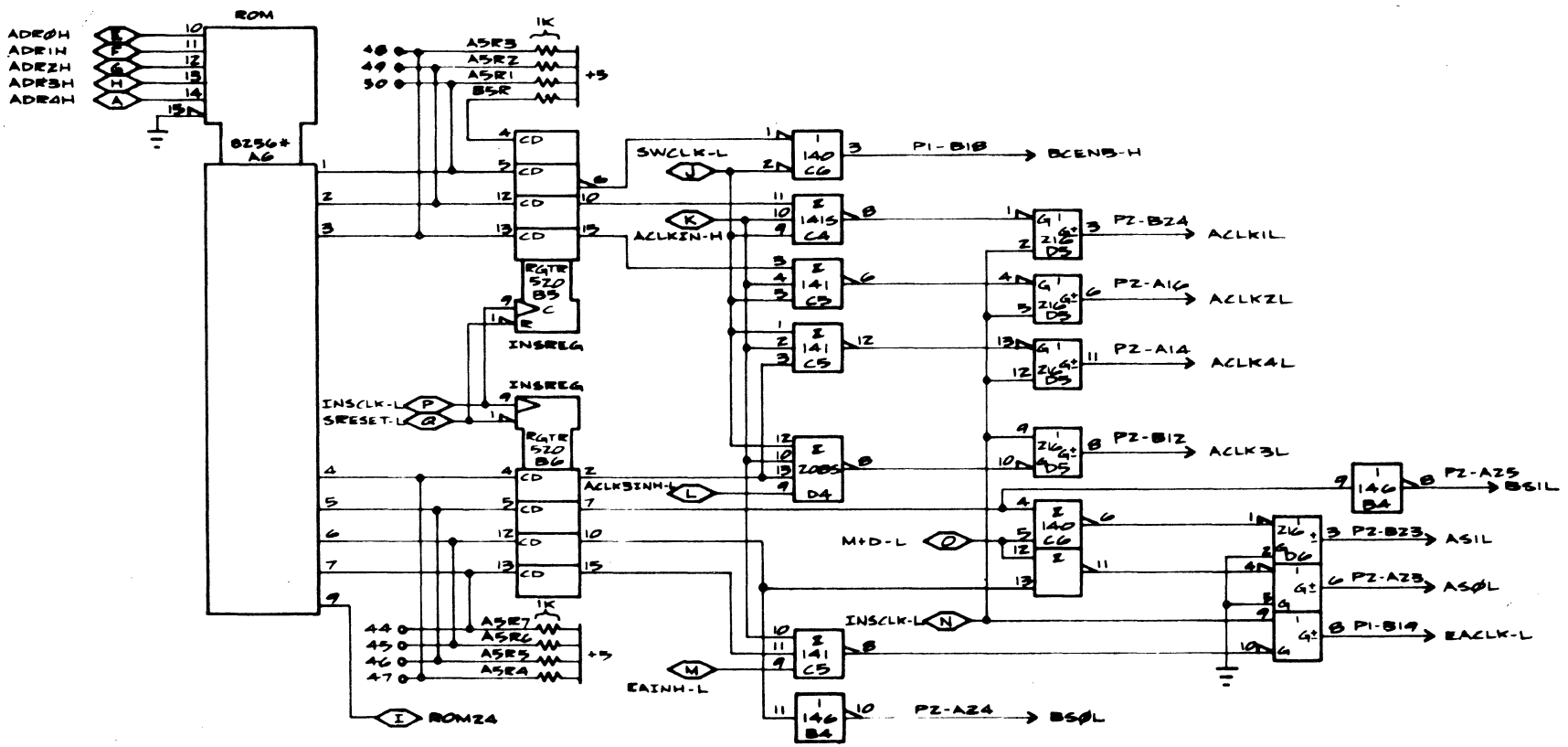
C | CODE IDENT NO 09132 | DRAWING NUMBER 88952600

SHEET 1 OF 6

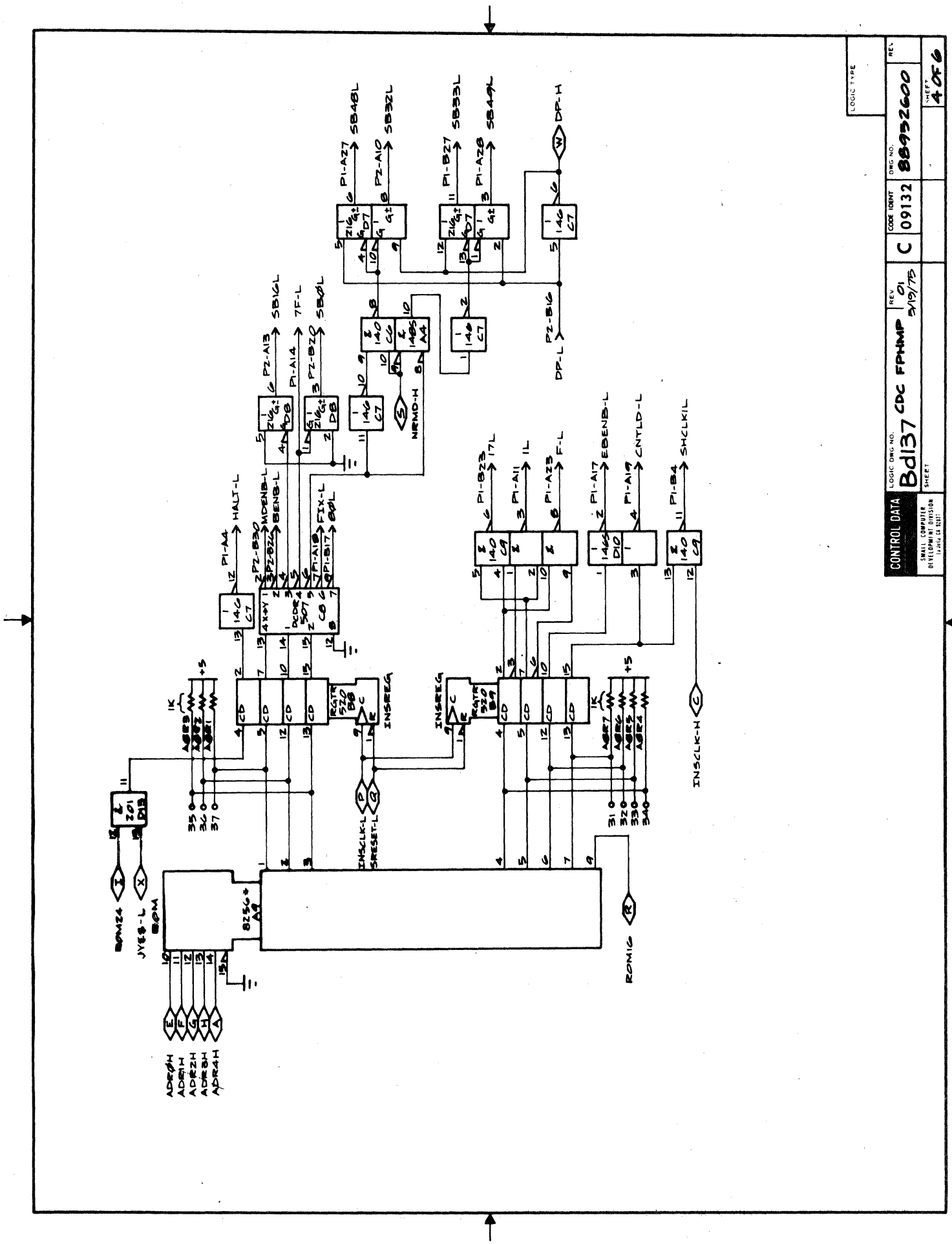


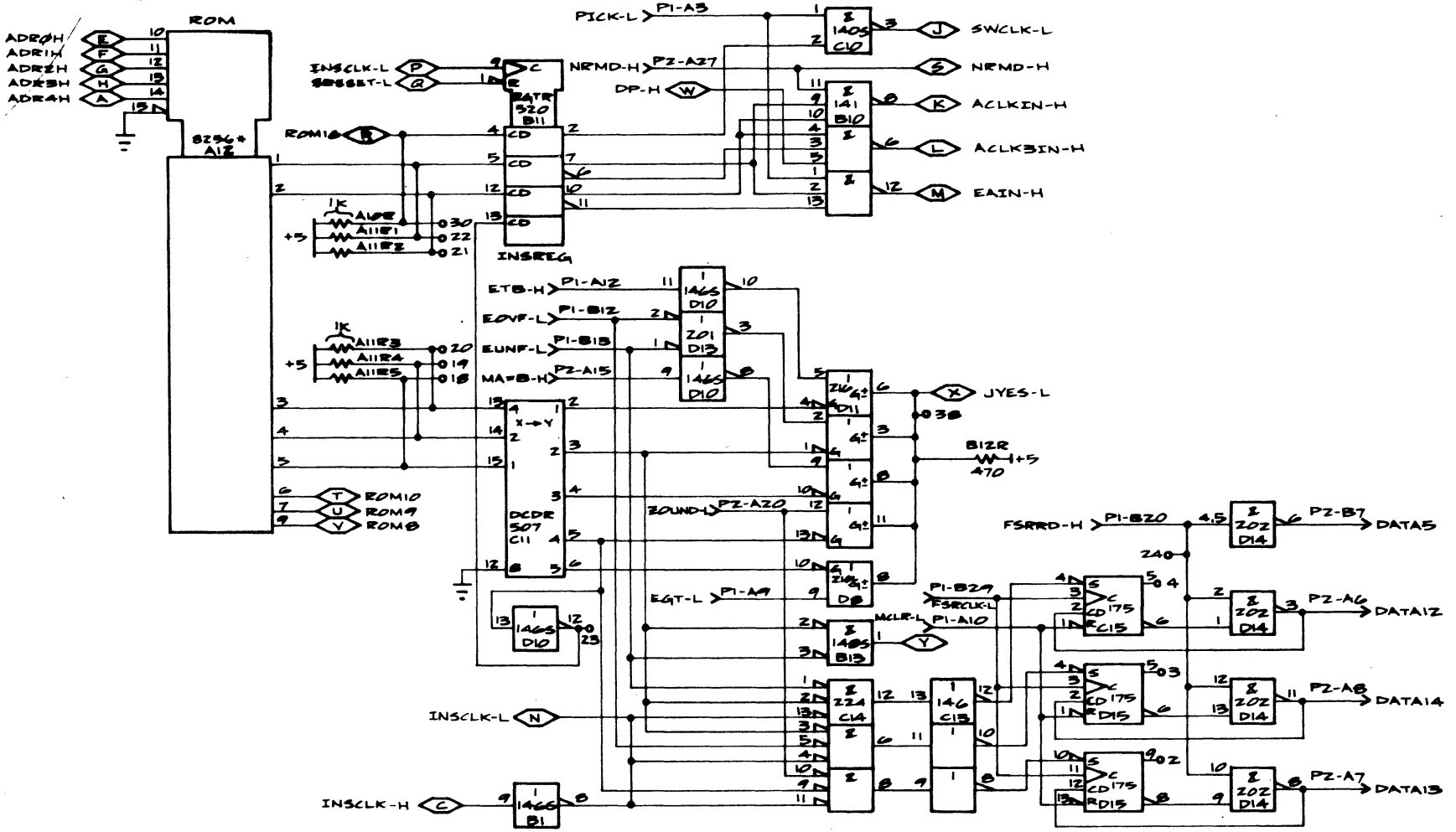
CONTROL DATA	LOGIC DWG NO. Bd137 CDC FPHMP	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88952600	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	SHEET				SHEET 2 OF 6	

LOGIC TYPE



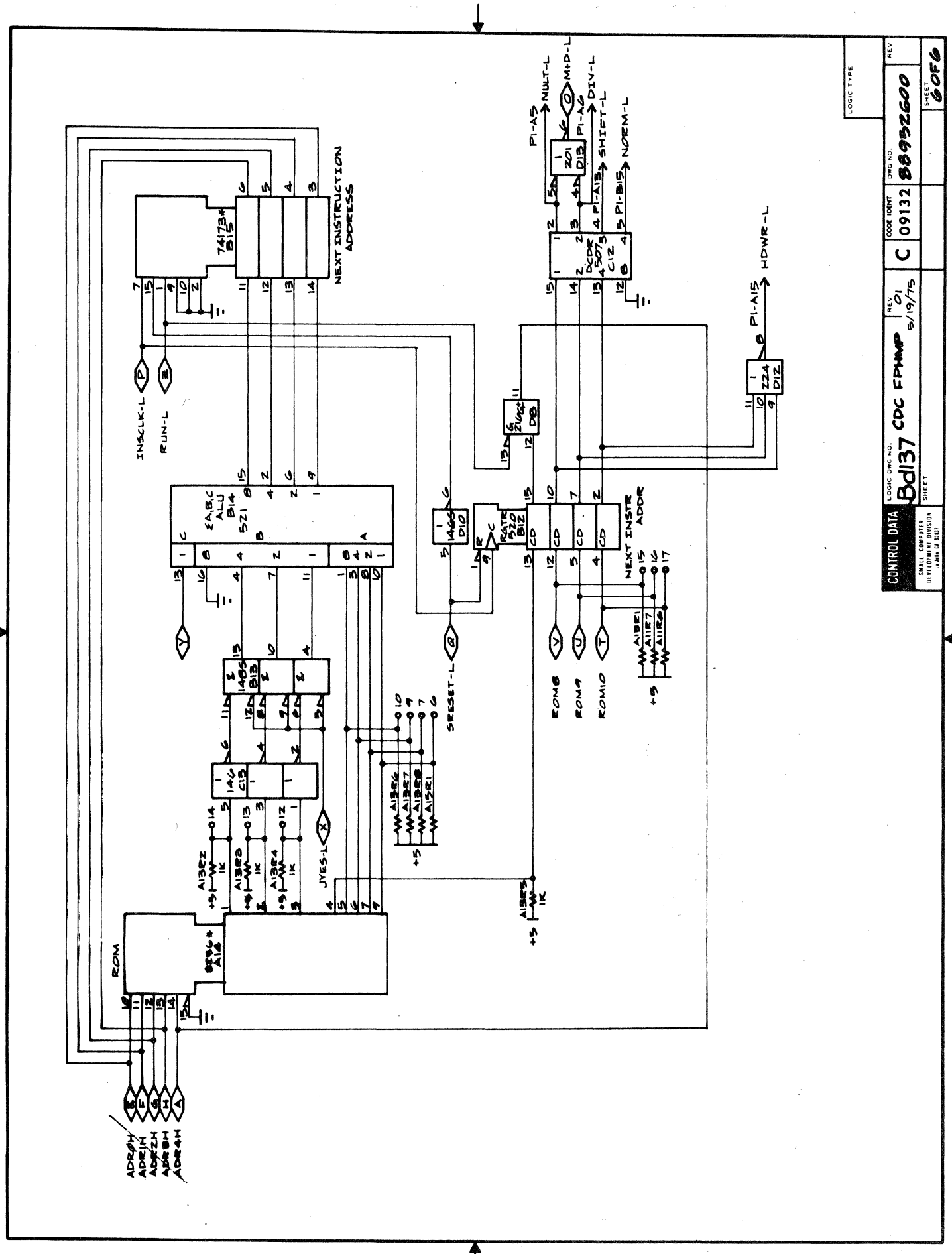
CONTROL DATA	TITLE DWG NO.	REV.		CODE IDENT	DWG NO.	
SMALL COMPUTER DEVELOPMENT DIVISION	Bd137 CDC FPHMP	01 5/19/75	C	09132	88952600	
	SHEET				SHEET 3 OF 6	


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd137 CDC FPHMP
REV 01 3/19/75
C
CODE IDENT 09132
DWG NO. 88952600
REV
SHEET 4 OF 6
LOGIC TYPE
SHEET
PI-A27 SB48L
PZ-A10 SB32L
PI-B27 SB33L
PI-A28 SB49L
DP-H
PZ-B16
DP-L
PZ-A13 SB16L
PI-A14 7F-L
PZ-B20 SB20L
NEMD-H
PI-A4 HALT-L
PZ-B30 MOENB-L
PZ-B22 BENB-L
PI-A18 FIX-L
PI-B17 80L
PI-B23 17L
PI-A11 1L
PI-A23 F-L
PI-A17 EBENB-L
PI-A19 CNTLD-L
PI-B4 SHCLKL
INSCLK-H
INSREEG
INSCLK-L
SRESET-L
BOMZ4
JYES-L
BOM
RONIG
ADR0H
ADR1H
ADR2H
ADR3H
ADR4H



CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	Bd137 CDC FPHMP	01 5/19/75	C	09132	88952600	
	SHEET				SHEET 5 OF 6	

LOGIC TYPE


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd137 CDC FPHMP
REV. 01 5/19/75
C
CODE IDENT 09132
DWG NO. 88952600
REV
SHEET 6 OF 6
LOGIC TYPE
NEXT INSTRUCTION ADDRESS
INSCLK-L
RUN-L
MULT-L
DIV-L
SHIFT-L
NORM-L
PI-A5
PI-A6
PI-A13
PI-B15
PI-A15
HDWE-L
SRESET-L
ROM
JYES-L
NEXT INSTR ADDR
ADR4H
ADR3H
ADR2H
ADR1H
ADR0H

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8	SHT 9	SHT 10
A	X			X					
B	X			X					
C	X			X					
D	X				X				
E	X				X				
F	X				X				
G	X				X				
H	X	X	X						
I	X		X						
J	X		X						
K	X		X						
L	X		X						
M	X		X						
N	X		X						
O	X		X						
P		X	X			X			
Q		X	X						
R		X	X						
S		X	X						
T		X	X						
U		X	X						
V		X	X						
W		X	X						
X		X		X					X
Y		X	X						
Z		X	X						
AA		X	X						
BB		X	X						
CC		X	X						
DD		X	X						
EE			X	X					
FF			X	X					
GG			X	X					
HH			X	X					
II			X		X				
JJ			X		X				
KK			X		X				
LL			X		X				
MM				X	X				
NN				X				X	

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8	SHT 9	SHT 10
OO				X	X	X			
PP				X	X				
QQ				X	X				
RR				X	X				
SS				X	X				
TT				X	X				
UU				X	X				
VV				X		X			X
WW				X		X			
XX				X		X			
YY				X		X			
ZZ	X			X					
AB				X	X				
AC				X	X				X
AD				X	X				
AE						X	X		
AF						X		X	
AG						X		X	X
AH						X		X	X
AI						X	X	X	
AJ						X			X
AK						X		X	
AL						X	X		
AM						X			X
AN						X			X
AO						X	X		
AP							X	X	
AQ							X	X	
AR							X		X
AS							X		X
AT						X	X	X	
AU							X		X
AV								X	X
AW								X	X
AX						X		X	
AY						X		X	
AZ						X			X
BC								X	X

NOTES UNLESS OTHERWISE SPECIFIED

DR DETACHED LIST

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						

LOGIC TITLE

LOGIC DWG NO | REV 01 1/31/5

PWA NO | PWB NO

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037

FIRST USED ON		
DR		
CHK		5/21/75
ENGR		
MFG		
APPD		

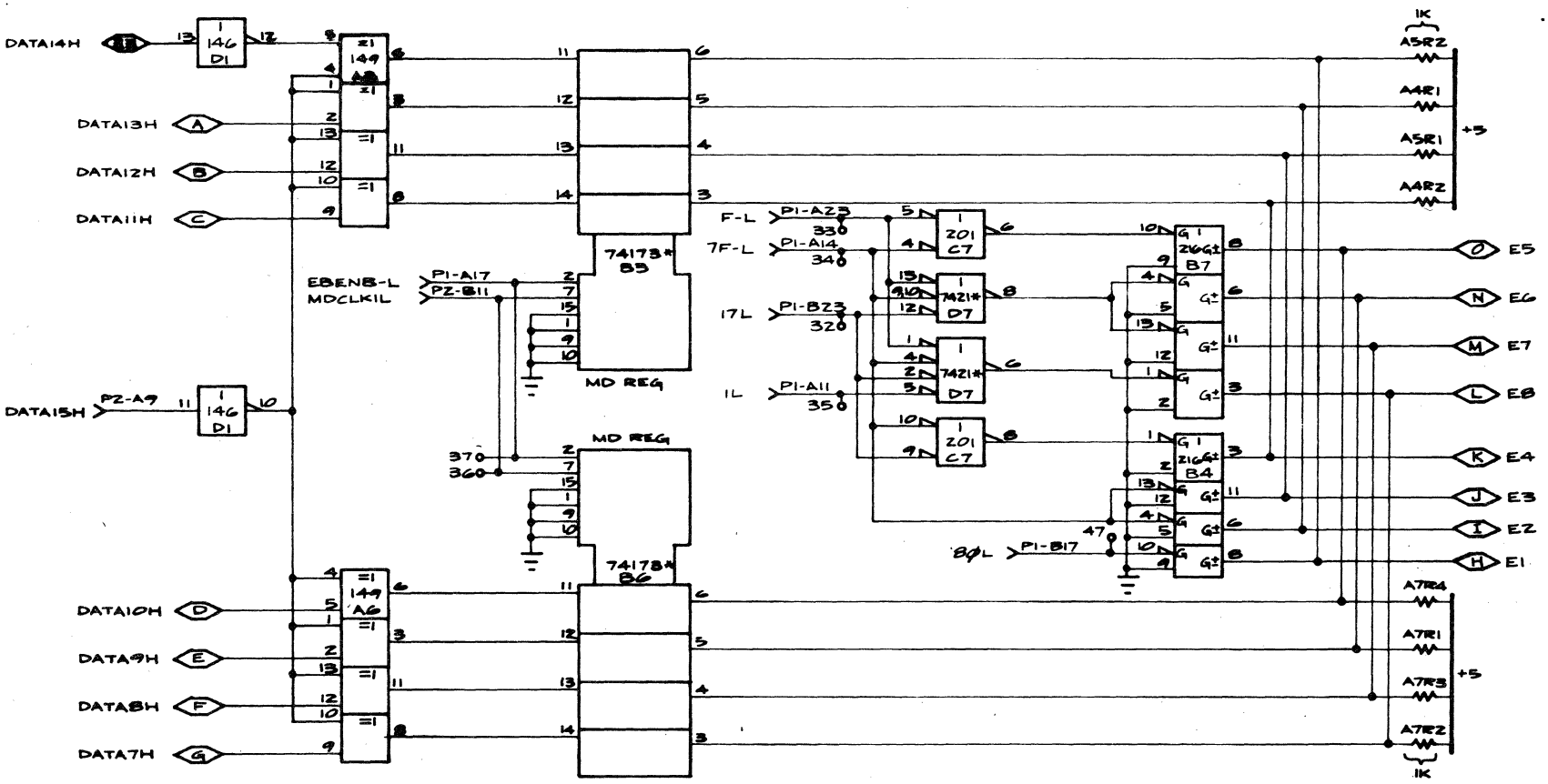
TITLE: Bd136 CDC EXP/TIM

CARD POS

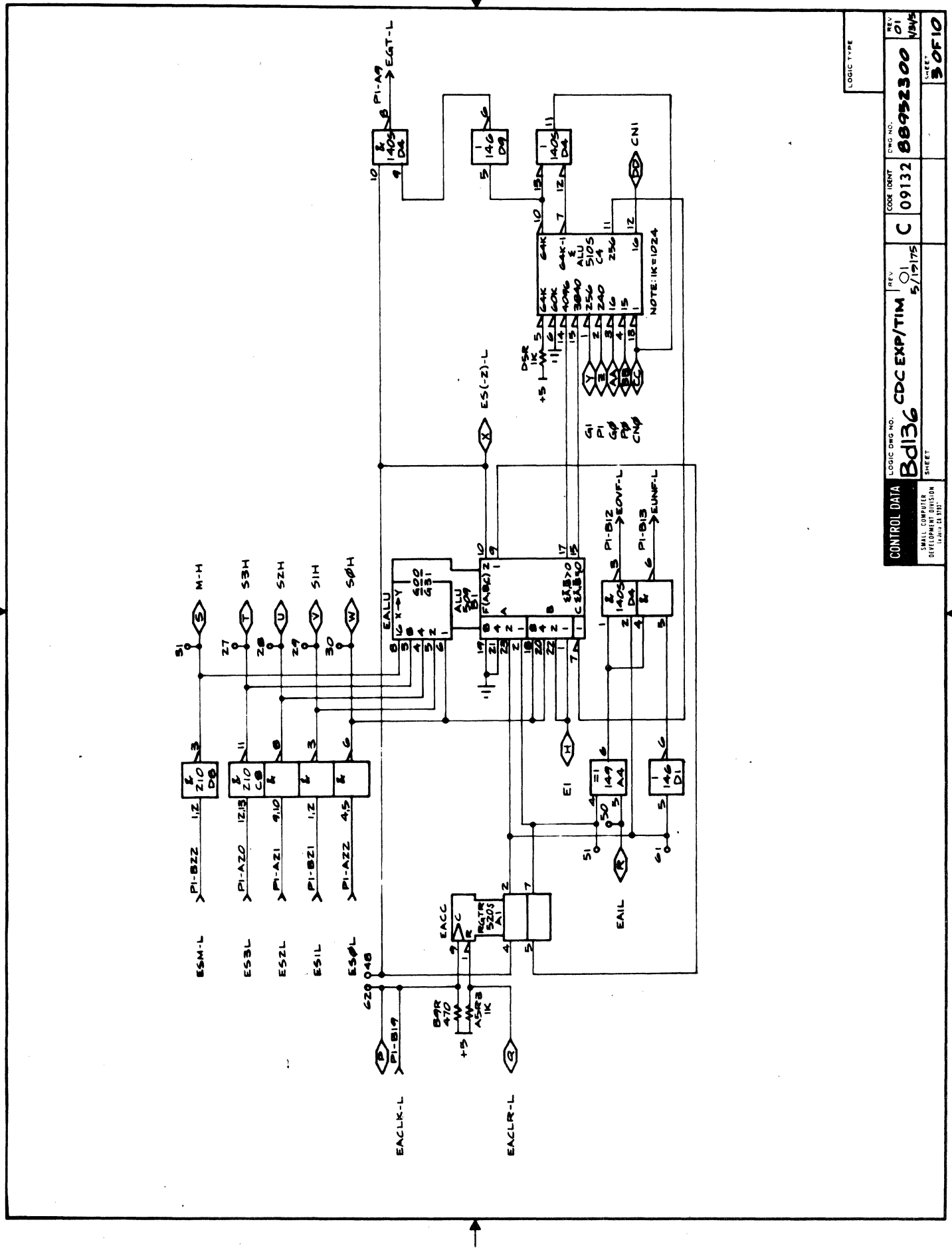
LOGIC TYPE

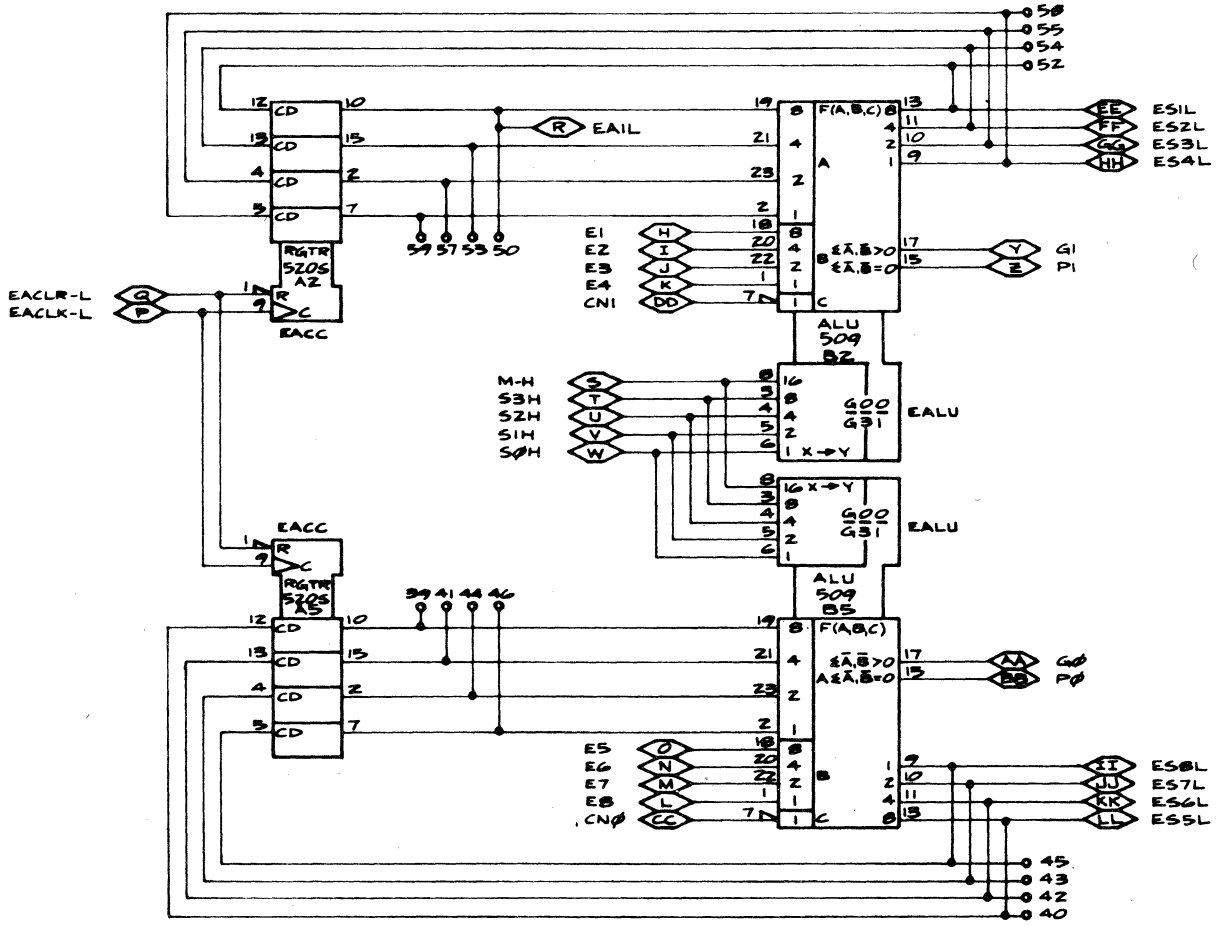
C | CODE IDENT NO 09132 | DRAWING NUMBER 88952300

SHEET 1 OF 10



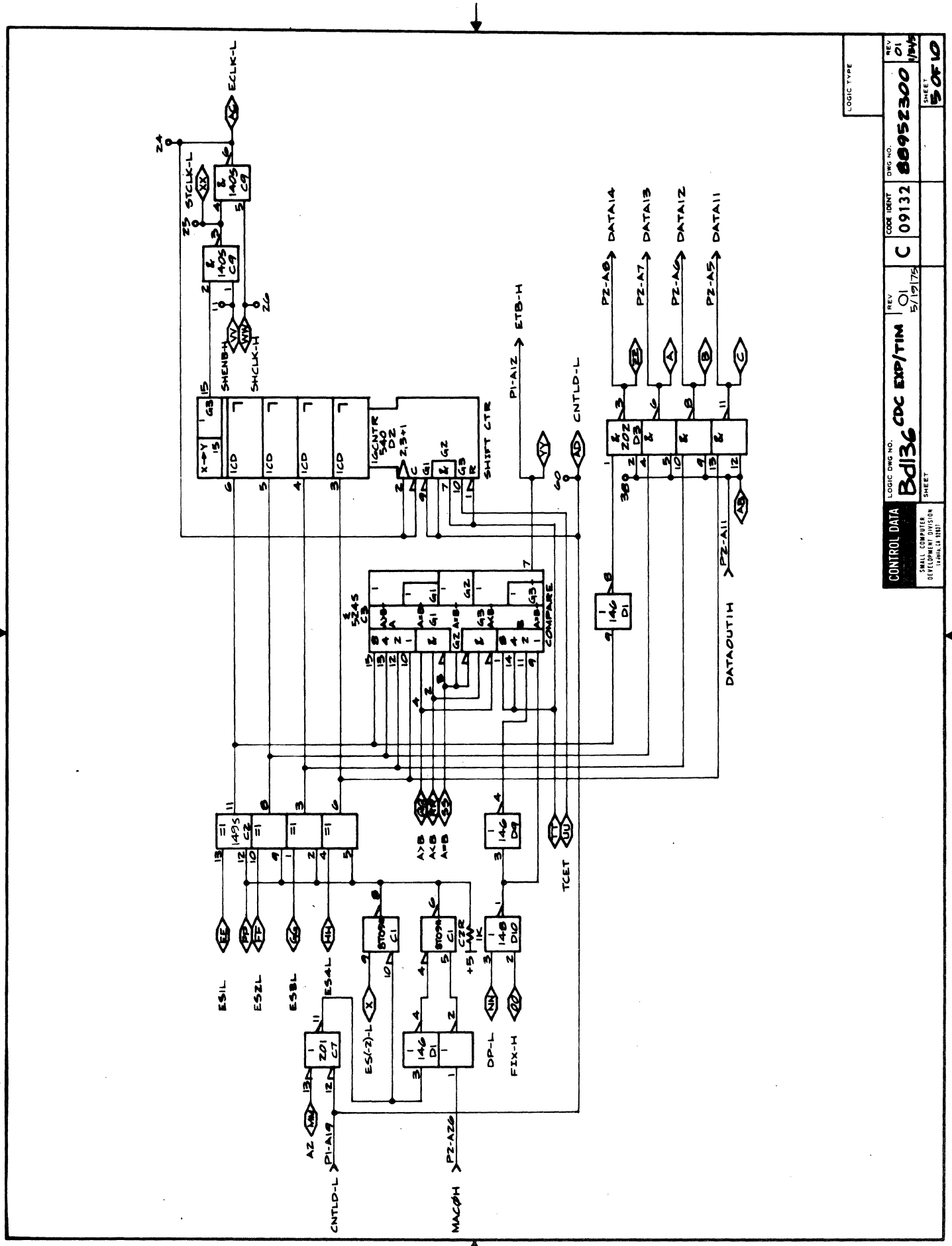
CONTROL DATA SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	LOGIC DWG NO. Bd136 CDC EXP/TIM	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88952300	REV 01 1/31/5	LOGIC TYPE
	SHEET					SHEET 2 OF 10	

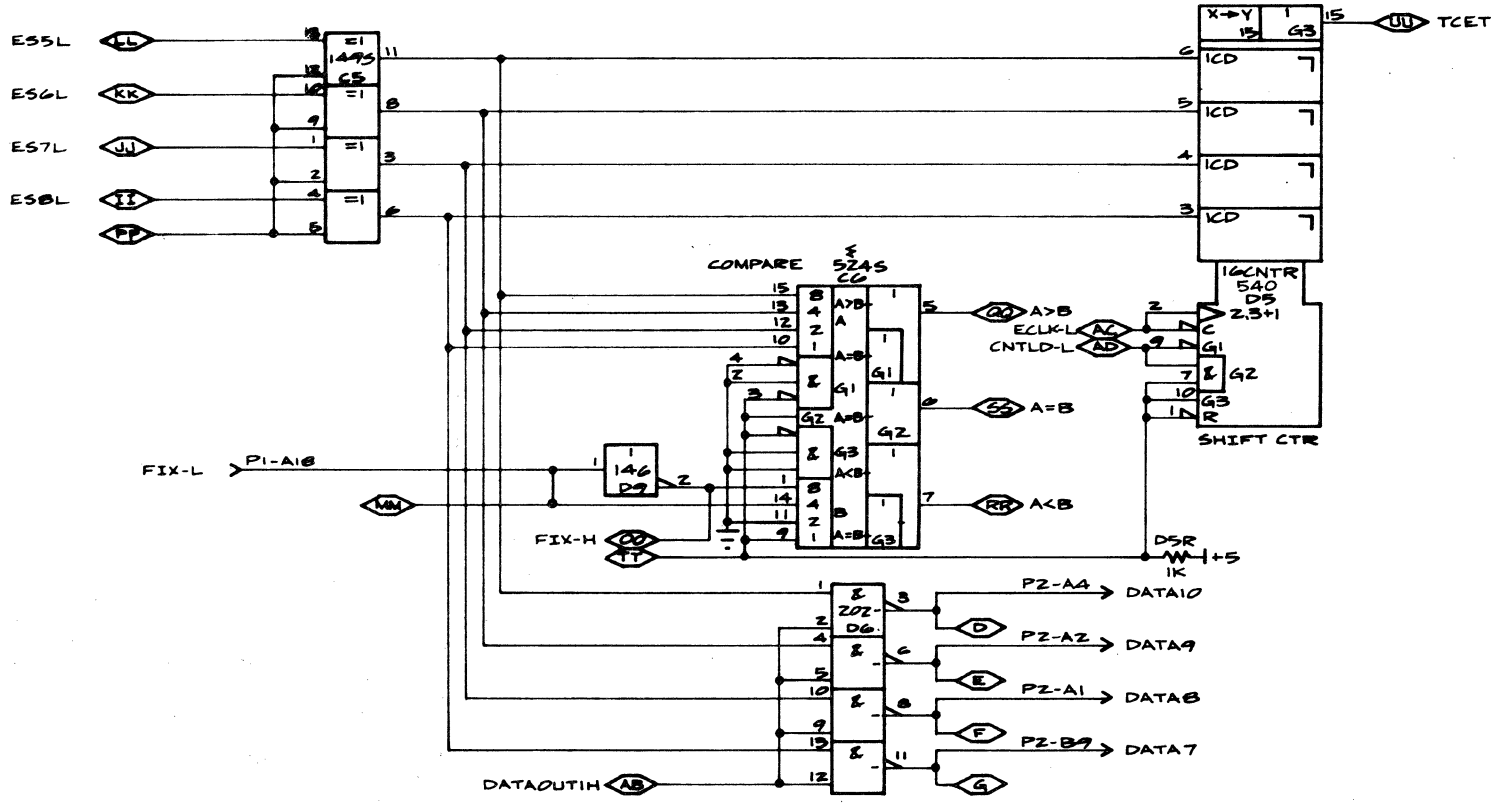

CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd136
SHEET
CDC EXP/TIM
REV 01 5/19/75
C
CODE IDENT 09132
DWG NO. 88932300
REV 01
SHEET 3 OF 10
LOGIC TYPE
NOTE: 1K=1024
EGT-L
ES(-Z)-L
EALU
ALU 5209 B1
400 431
ALU 5105 C4 256
PSR 1K
EACC
REGTR 5203 A1
BSR 470
ASR8 1K
EAIL
EACLK-L
EACLR-L
ESM-L
ES3L
ES2L
ES1L
ES0L
EI
CNI
PI-A9
PI-B22
PI-A20
PI-A21
PI-B21
PI-A22
PI-B19
PI-B12
PI-B13
EOVF-L
EUNF-L
M-H
S3H
S2H
S1H
S0H



CONTROL DATA	LOGIC DWG NO. Bd136 CDC EXP/TIM	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88952300	REV 01
SMALL COMPUTER DEVELOPMENT DIVISION (La Jolla CA 92037)	SHEET					SHEET 4 OF 10

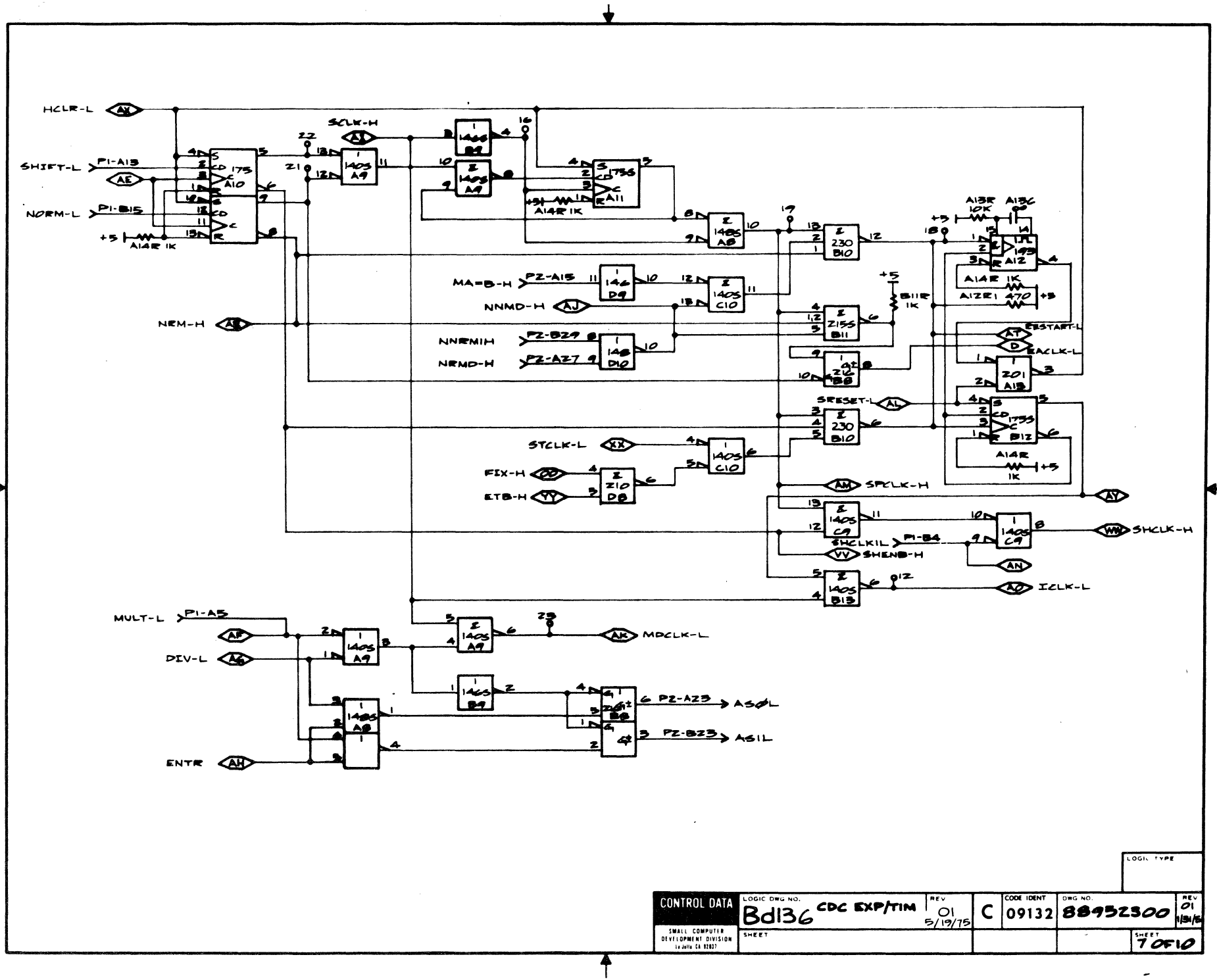
LOGIC TYPE


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd136 CDC EXP/TIM
REV 01 5/19/75
CODE IDENT 09132
DWG NO. 88952300
REV 01
SHEET 5 OF 10
LOGIC TYPE
ECLK-L
STCLK-L
SHENB-H
SHCLK-H
SHIFT CTR
P1-A12 → ETB-H
CNTLD-L
PZ-A8 DATA14
PZ-A7 DATA13
PZ-A6 DATA12
PZ-A5 DATA11
PZ-A11
DATAOUTIH
COMPARE
A>B
A<B
A=B
TCET
ESIL
ESZL
ESBL
ESAL
ES(-2)-L
DP-L
FIX-H
AZ
P1-A19 → CNTLD-L
PZ-A26 → MACØH



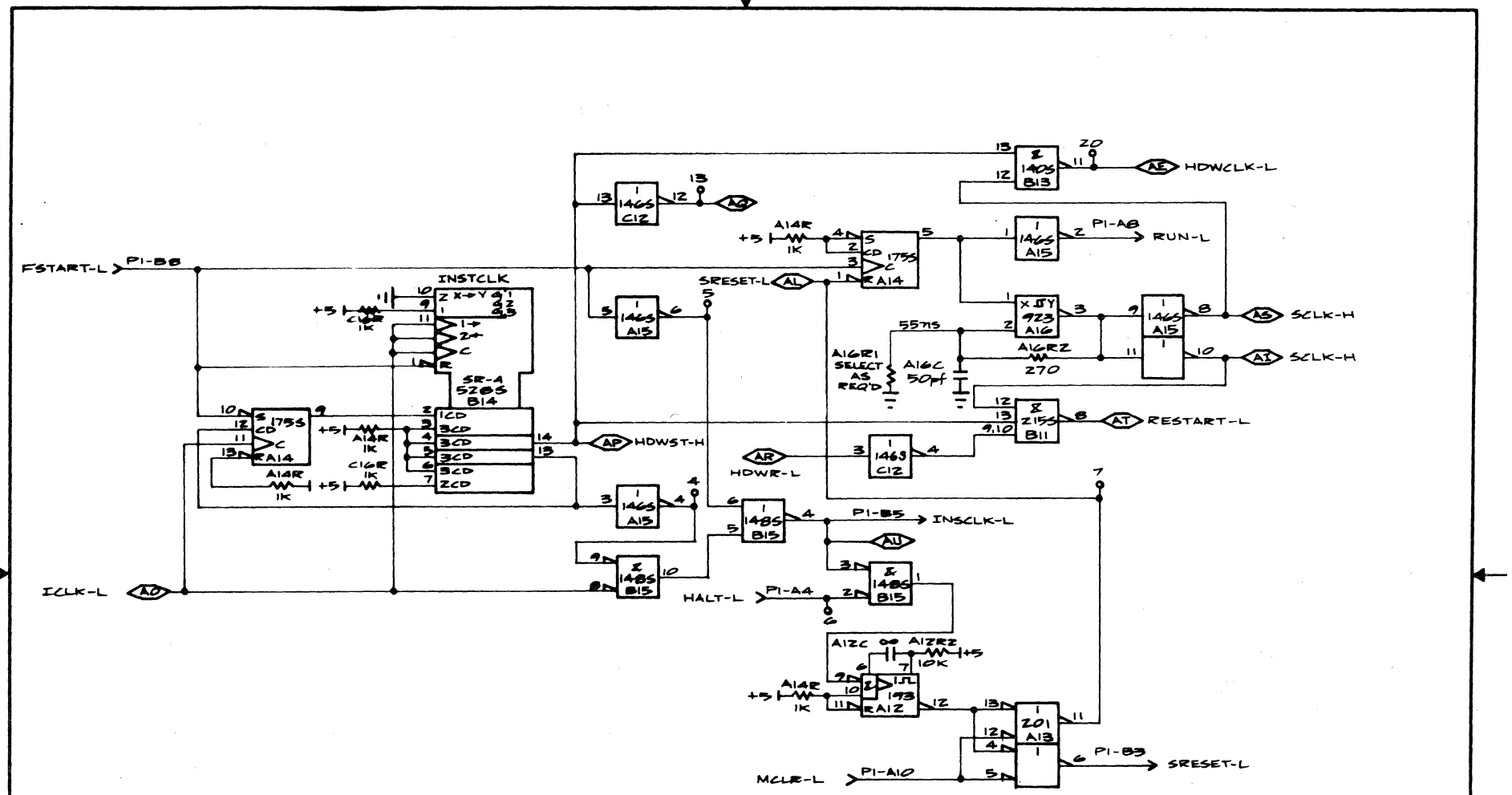
CONTROL DATA	LOGIC DWG NO. Bd136 CDC EXP/TIM	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88952300	REV 01 1/31/75
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	SHEET					SHEET 6 OF 10

LOGIC TYPE

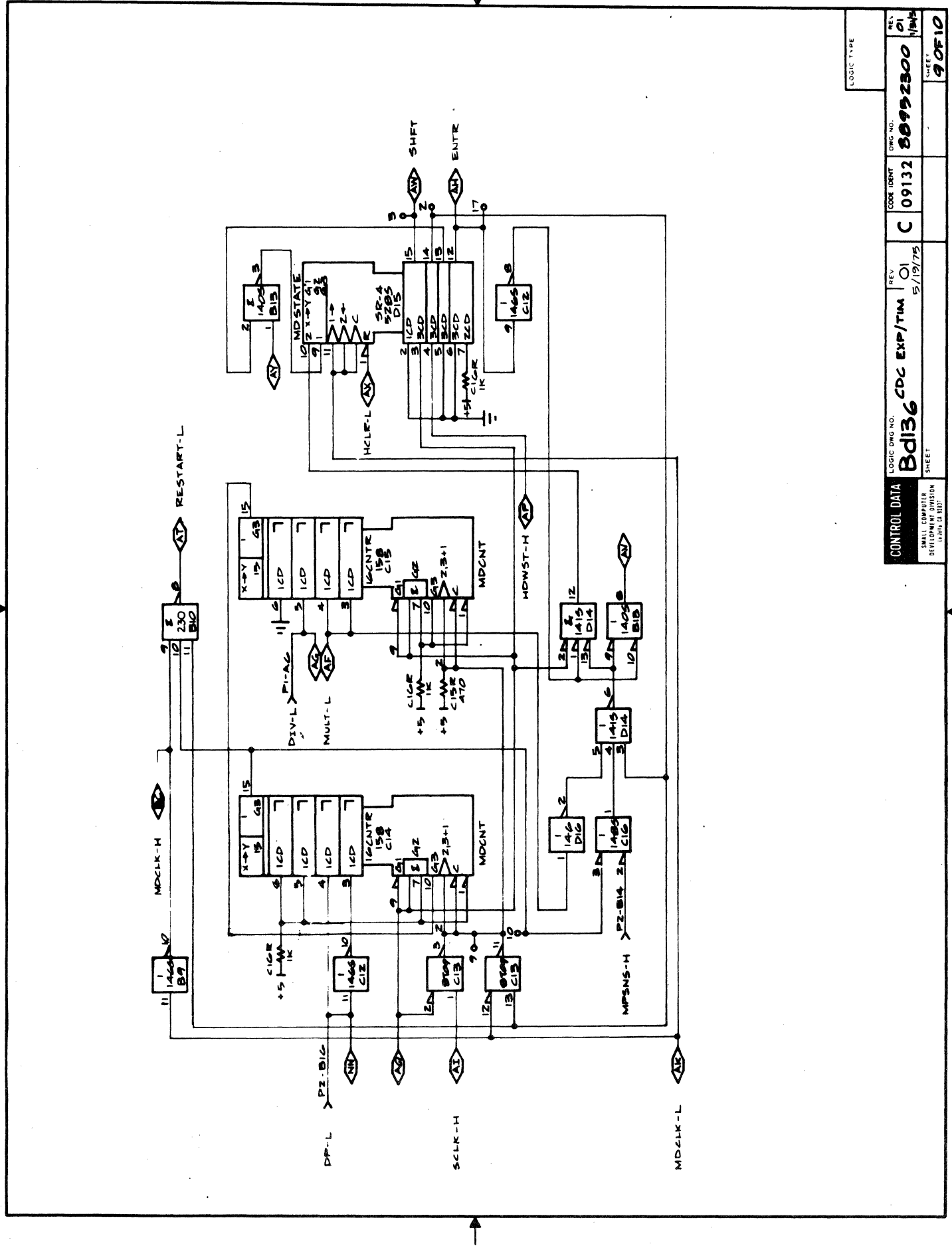


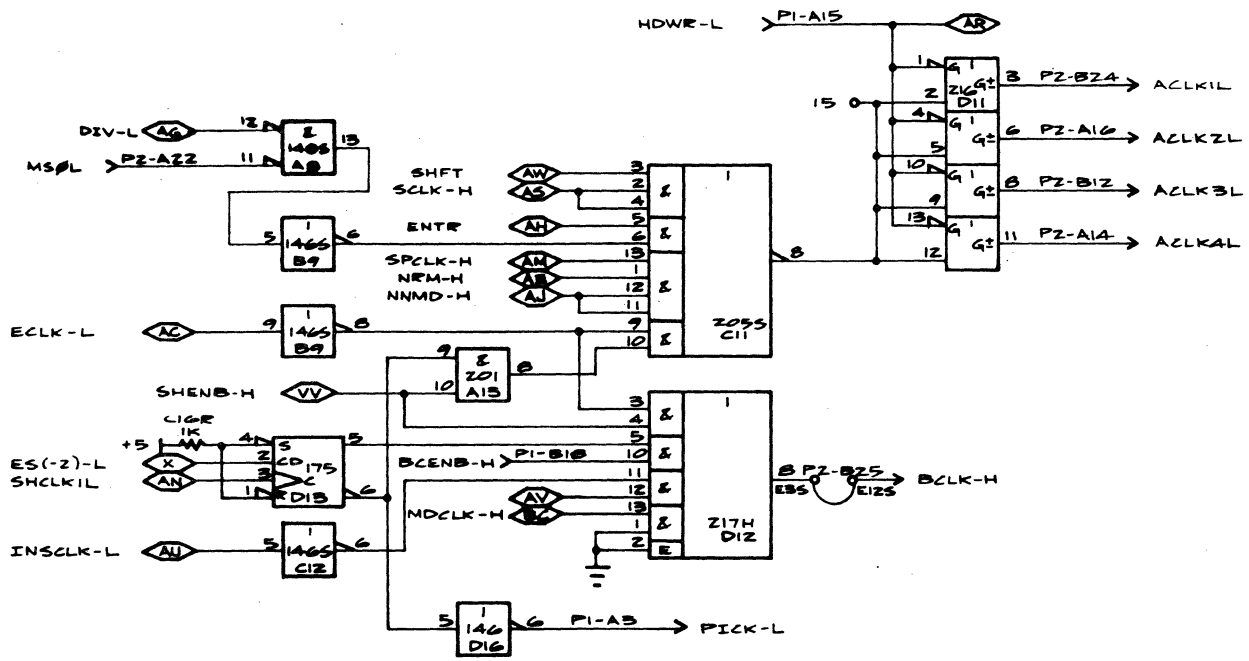
CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	Bd136 CDC EXP/TIM	01 5/19/75	C	09132	88952300	01
	SHEET					SHEET 7 OF 10

LOGIC TYPE



CONTROL DATA	LOGIC DWG NO. Bd136 CDC EXP/TIM	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88952300	REV 01 1/31/75	LOGIC TYPE
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	SHEET					SHEET 8 OF 10	


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd136 CDC EXP/TIM
REV. 01 5/19/75
SHEET
CODE IDENT C 09132
DWG NO. 88952300
REV. 01
SHEET 9 OF 10
LOGIC TYPE
SHFT
ENTR
RESTART-L
MDCLK-H
MDSTATE
SR-4
MDCNT
16CNTR
HDWST-H
DIV-L
MULT-L
HCLF-L
MPSNS-H
DP-L
SCLK-H
MDCLK-L



DR	DETACHED LIST	A	B	C	D	E	F
	1						
	2						
	3						
	4						
	5						
	6						
	7						

NOTES: UNLESS OTHERWISE SPECIFIED

LOGIC TITLE: CDC EXP/TIM
Bd136

LOGIC DWG NO | REV 01 5/19/75

PWA NO | PWB NO

CONTROL DATA — SMALL COMPUTER DEVELOPMENT DIVISION, La Jolla CA 92037

FIRST USED ON		
DR		
CHK	R.B.	5/21/75
ENGR		
MFG		
APPD		

TITLE

CARD POS

LOGIC TYPE

C | CODE IDENT NO. 09132 | DRAWING NUMBER 88952300

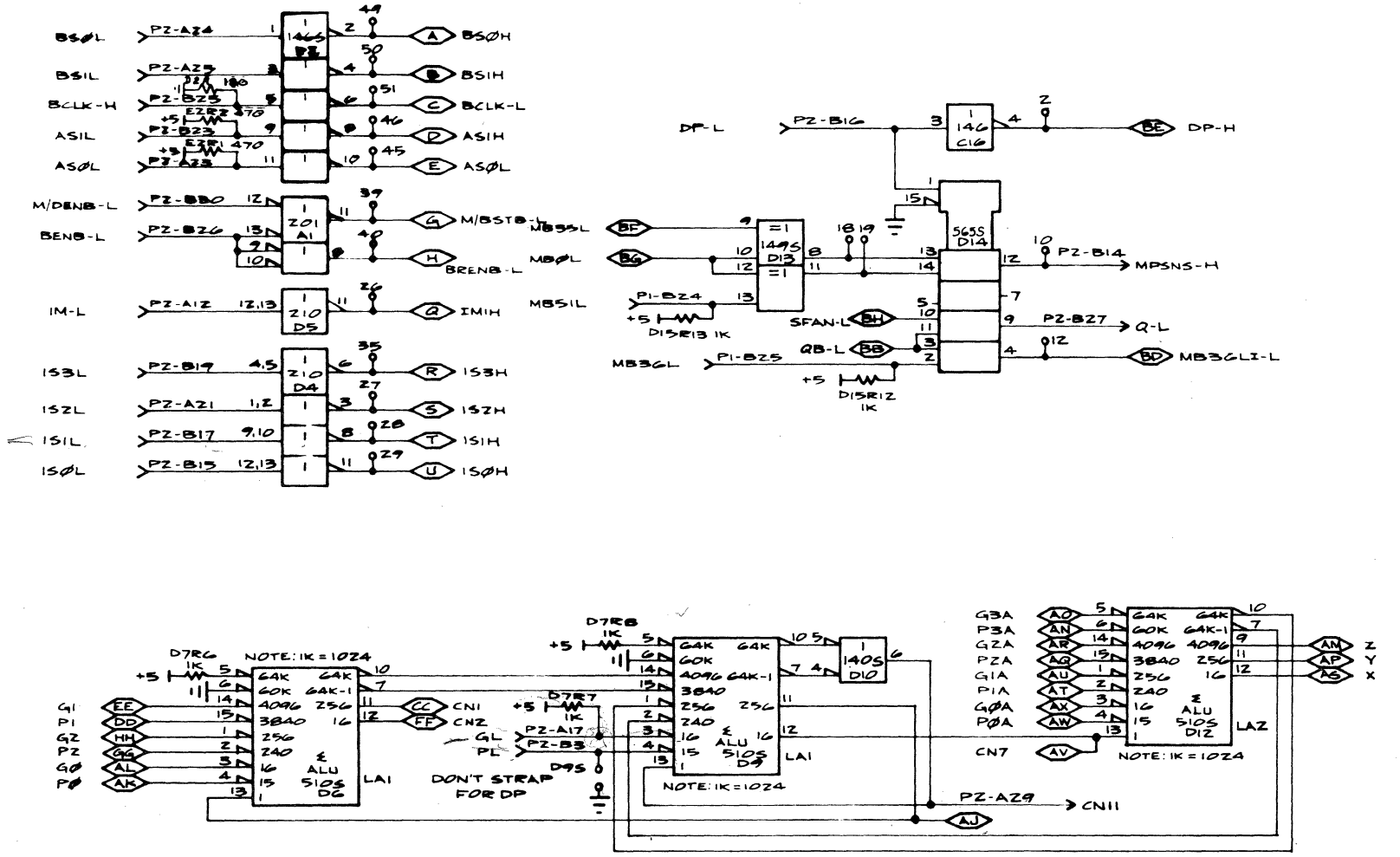
SHEET 10 OF 10

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8	SHT 9
A	X	X	X	X	X	X	X	X
B	X	X	X	X	X	X	X	X
C	X	X	X	X	X	X	X	X
D	X	X	X	X	X	X	X	X
E	X	X	X	X	X	X	X	X
F		X	X					
G	X	X	X	X	X	X	X	X
H	X	X	X	X	X	X	X	X
I		X	X					
J		X	X					
K		X	X					
L		X	X					
M		X	X					
N			X	X				
O			X	X				
P		X	X	X	X	X	X	X
Q	X	X	X	X	X	X	X	X
R	X	X	X	X	X	X	X	X
S	X	X	X	X	X	X	X	X
T	X	X	X	X	X	X	X	X
U	X	X	X	X	X	X	X	X
V		X	X					
W		X		X				X
X		X				X		
Y		X				X		
Z		X				X		
AA		X						
BB	X	X						
CC	X	X						
DD	X	X						
EE	X	X						
FF	X		X					
GG	X		X					
HH	X		X					
II				X	X	X	X	
JJ				X	X	X	X	
KK				X	X			
LL				X	X			
MM			X	X				
NN			X	X				
OO					X	X		
PP					X	X		

SIGNAL LETTER	SHT 2	SHT 3	SHT 4	SHT 5	SHT 6	SHT 7	SHT 8	SHT 9
QQ						X	X	
RR						X	X	
SS							X	X
TT							X	X
UU				X	X	X	X	
VV				X	X			
WW				X	X			
XX					X	X		
YY					X	X		
ZZ						X	X	
AB						X	X	
AC							X	X
AD							X	X
AE								
AF			X				X	
AG			X				X	
AH			X				X	
AI			X				X	
AJ	X			X				
AK	X			X				
AL	X			X				
AM	X				X			
AN	X				X			
AO	X				X			
AP	X					X		
AQ	X					X		
AR	X					X		
AS	X						X	
AT	X						X	
AU	X						X	
AV	X							X
AW	X							X
AX	X							X
AY				X				X
AZ				X				X
BC				X				X
BD	X							X
BE	X							X
BF	X							X
BG	X	X						
BH	X	X						

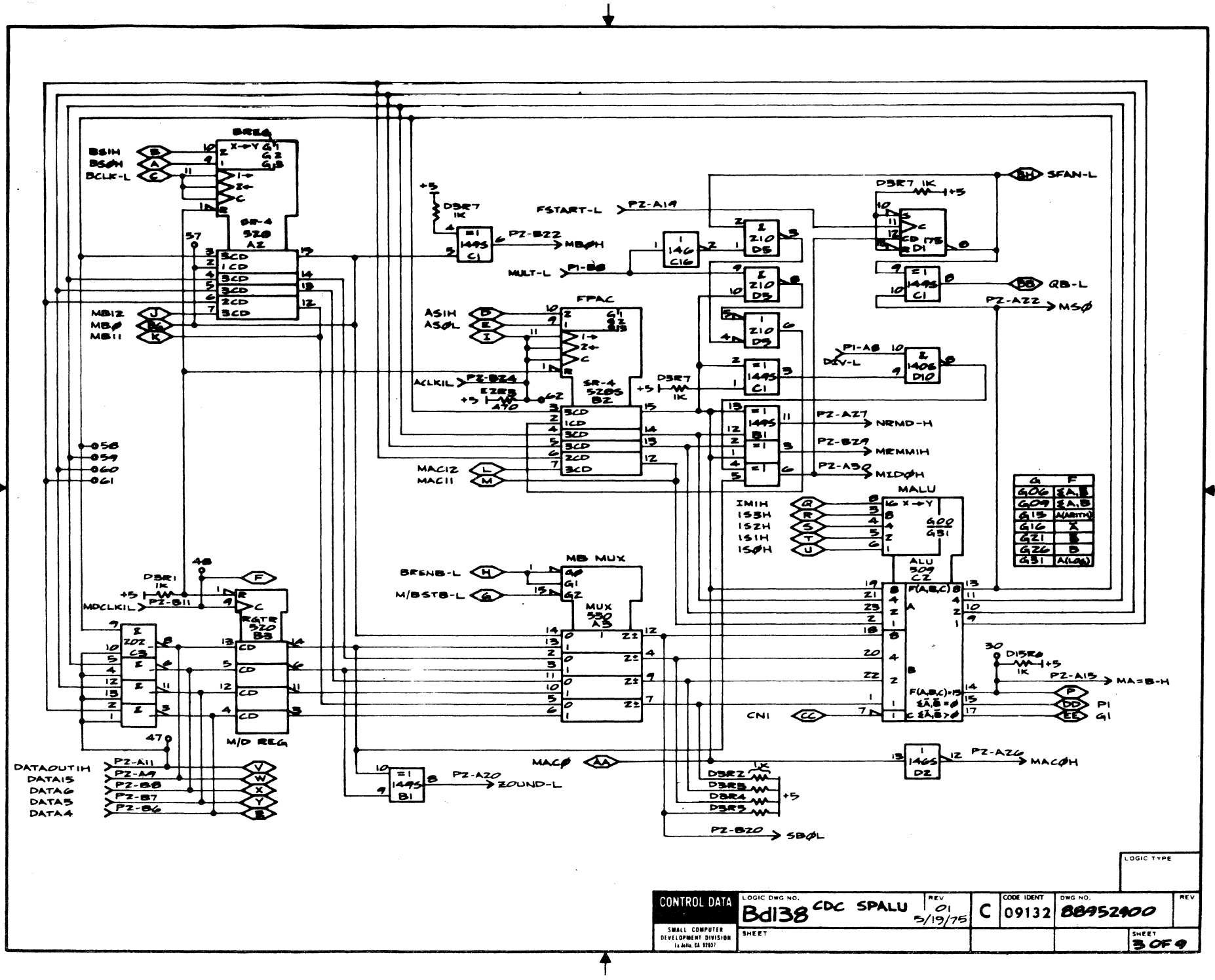
LOGIC TYPE

CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla CA 92037	Bd138 CDC SPALU	01 5/19/75	C	09132	88952900	
	SHEET					SHEET 10F9



CONTROL DATA	LOGIC DWG NO. Bd138 CDC SPALU	REV. 01 5/19/75	C	CODE IDENT 09132	DWG NO. 88952900	REV.
SMALL COMPUTER DEVELOPMENT DIVISION	SHEET					SHEET 2 OF 9

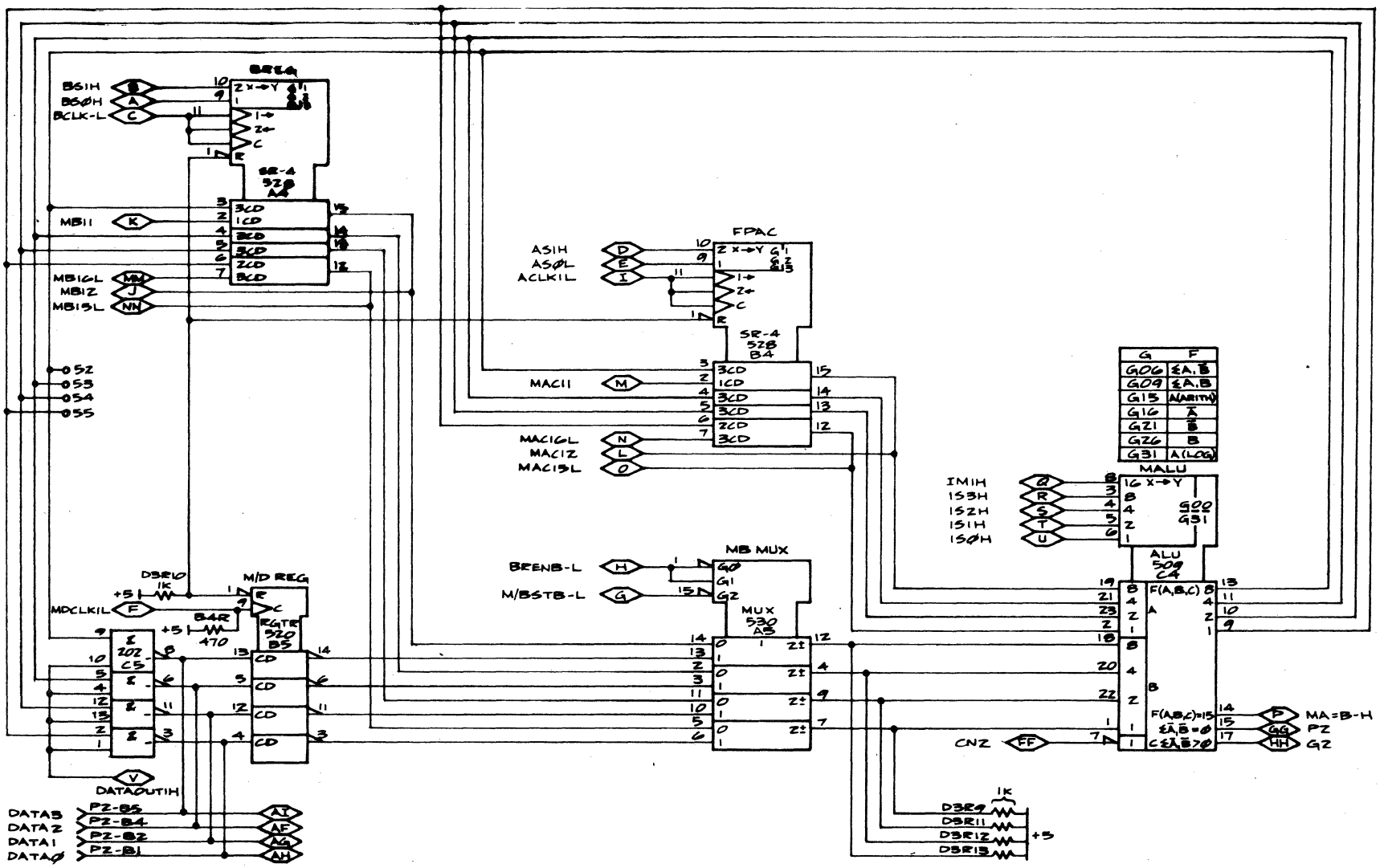
LOGIC TYPE



G	F
G06	Σ A,B
G09	Σ A,B
G15	A/ARITH
G16	X
G21	B
G26	D
G51	A(LM)

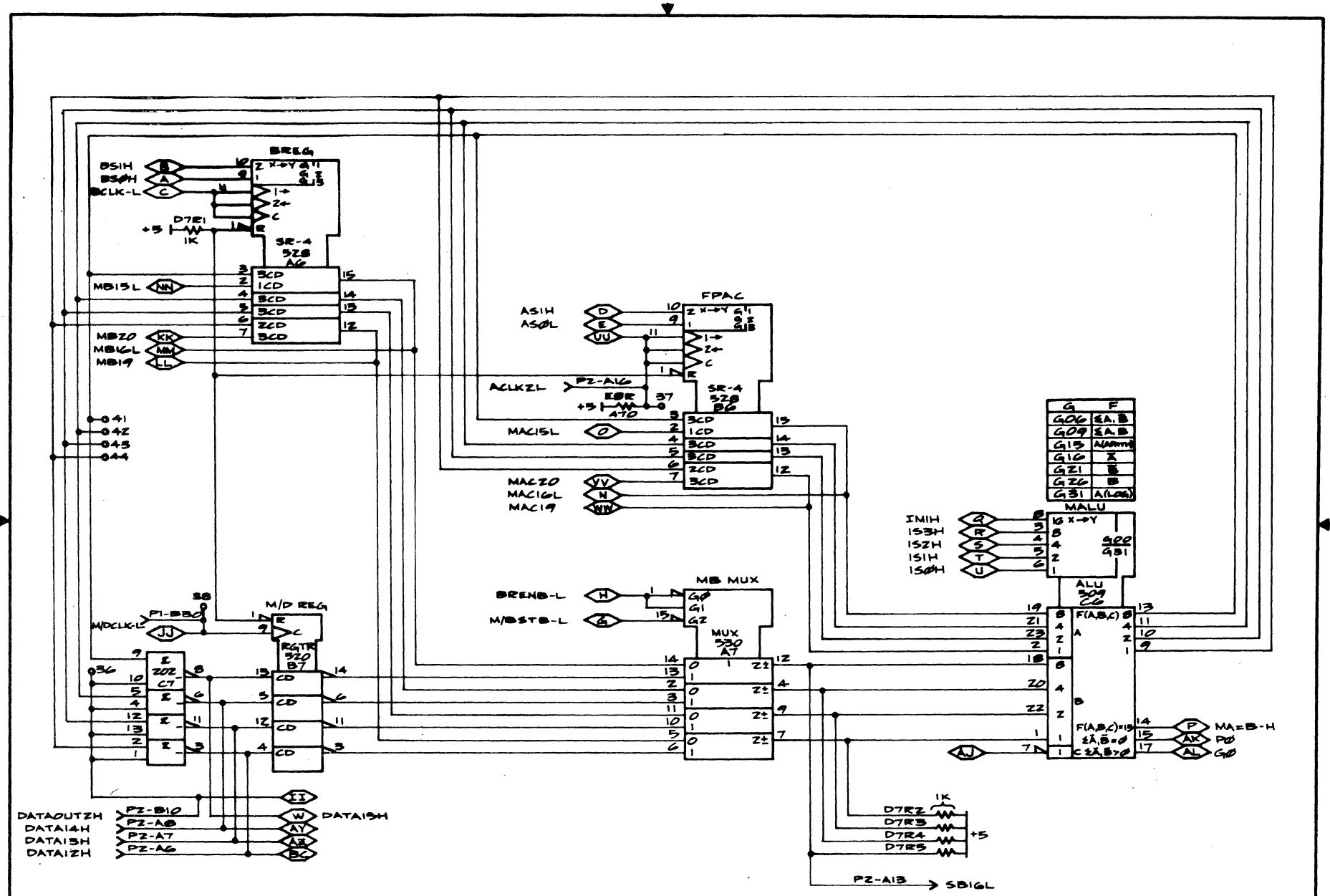
LOGIC TYPE

CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION (a Data Co 2207)	Bd138 CDC SPALU	01 5/19/75	C	09132	88952400	
	SHEET					SHEET 3 OF 9



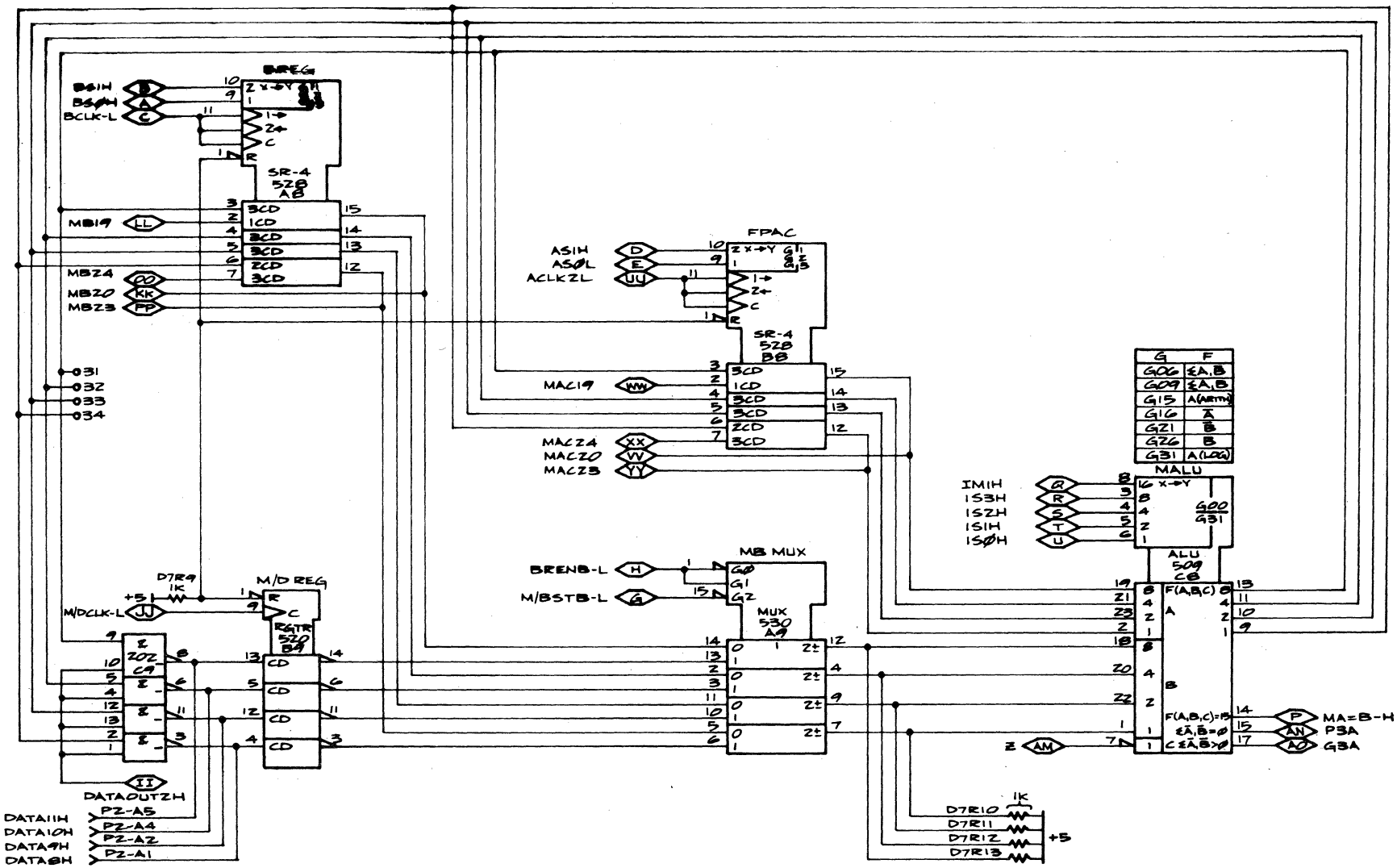
G	F
G06	Σ A.B
G09	Σ A.B
G15	A(ARITH)
G16	A
G21	B
G26	B
G31	A(LOG)

LOGIC TYPE				
CONTROL DATA SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	LOGIC DWG NO. Bd138 CDC SPALU	REV 01 5/19/75	C	CODE IDENT 09132
SHEET	DWG NO. 88952900	REV	SHEET 4 OF 9	



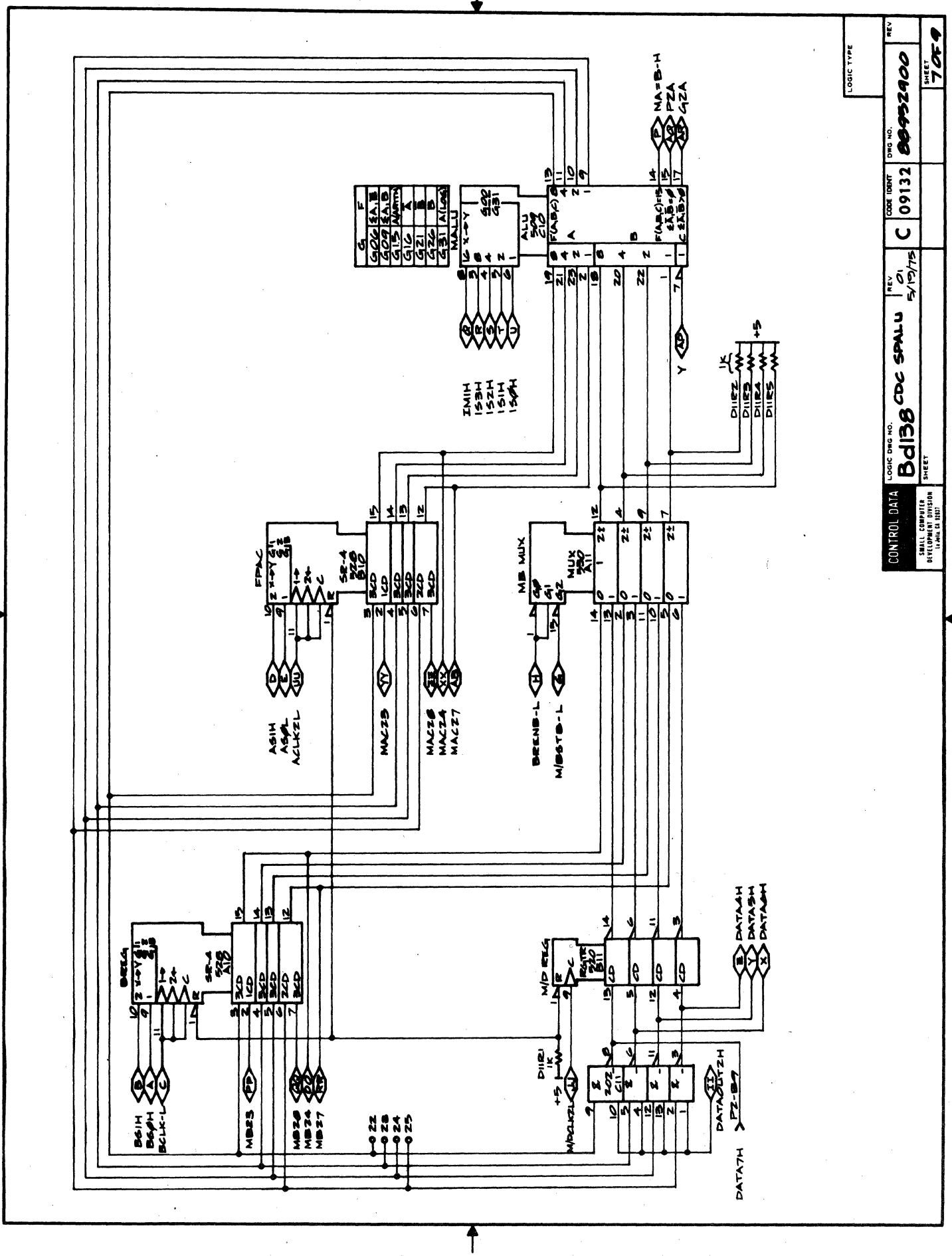
CONTROL DATA	LOGIC DWG NO. Bd138 CDC SPALU	REV 01 5/19/75	C	CODE IDENT 09132	DWG NO. 80952900	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	SHEET				SHEET 5 OF 9	

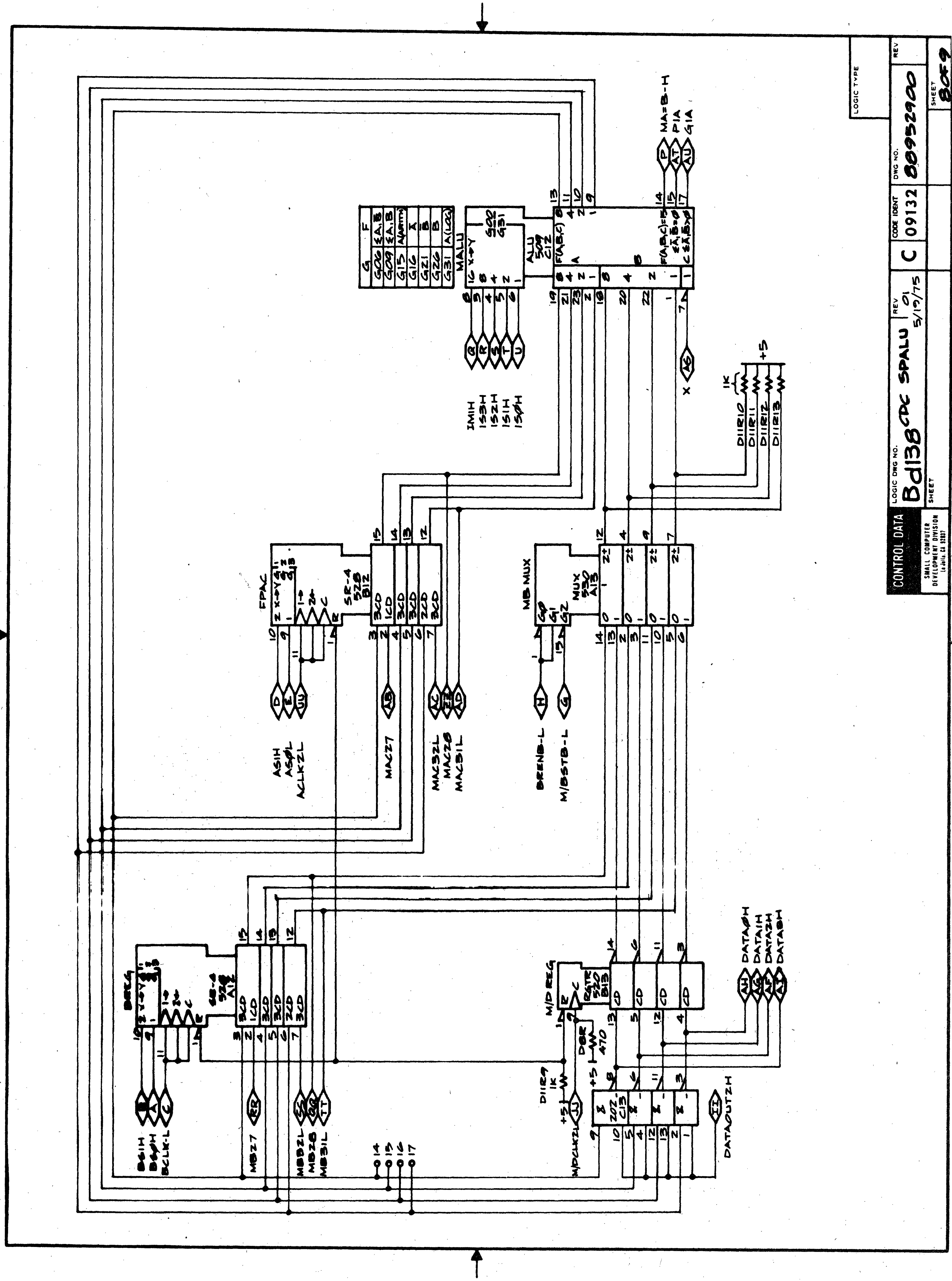
LOGIC TYPE

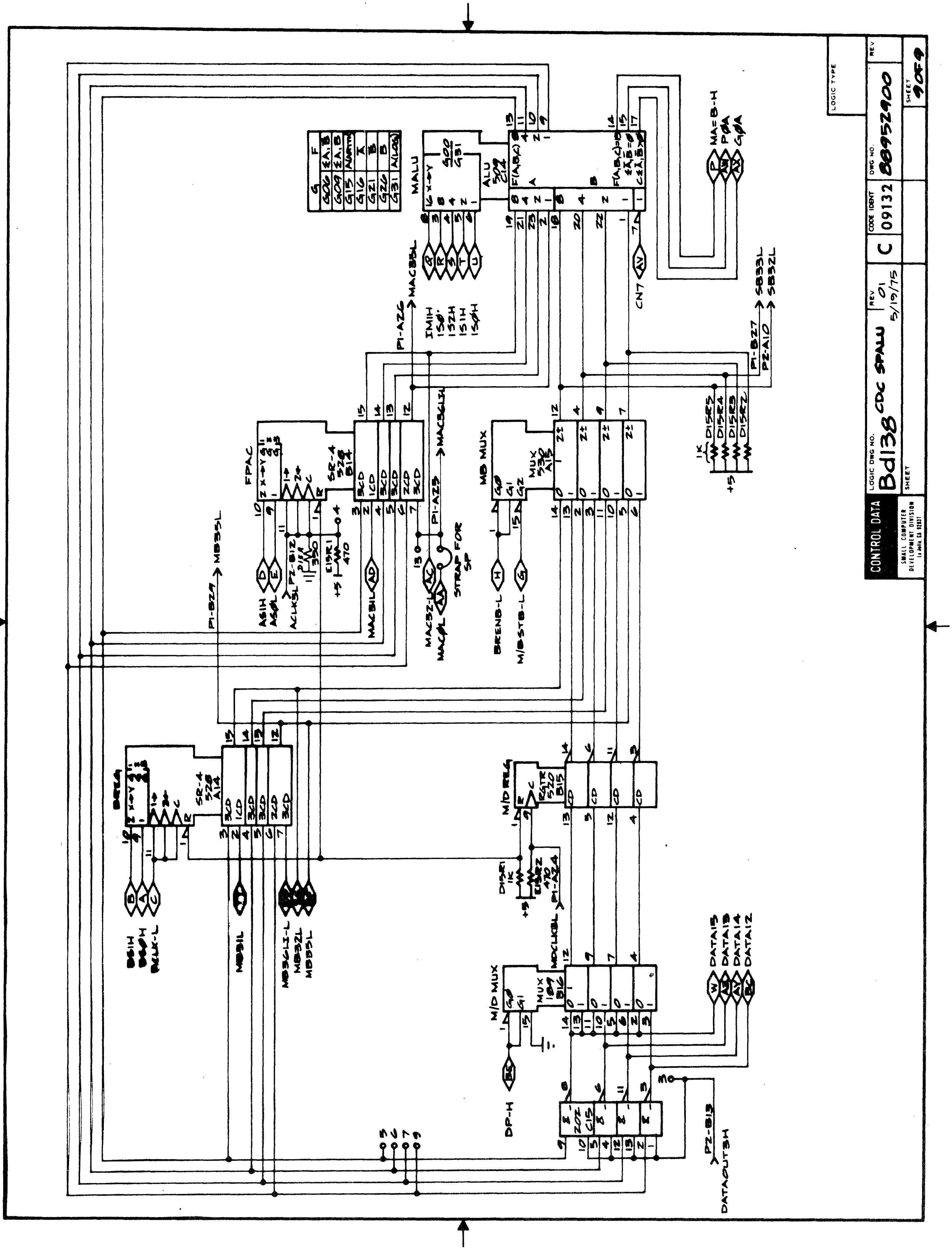


CONTROL DATA	LOGIC DWG NO.	REV		CODE IDENT	DWG NO.	REV
SMALL COMPUTER DEVELOPMENT DIVISION La Jolla, CA 92037	Bd138 CDC SPALU	01 5/19/75	C	09132	88952900	
	SHEET					SHEET 6 OF 9

LOGIC TYPE


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION (LAHABRA, CALIFORNIA)
LOGIC DWG NO. Bd138 CDC SPALU
SHEET
REV 01 5/15/75
C
CODE IDENT 09132
DWG NO. 88932900
REV
SHEET 7 OF 9
LOGIC TYPE
ALU
MALU
MB MUX
FPAC
M/D REG
+5
1K
DATA7H
DATA6H
DATA5H
DATA4H
MB23
MB26
MB24
MB27
ZZ
Z8
Z4
Z5
BCLK-L
M/BCLKZL
BEENB-L
M/BSTB-L
ASIH
ASGL
ACLKZL
IMIH
ISZH
ISIH
ISOH
NA=B-H
PZA
QZA
Y
DI12I
DI1EZ
DI1E3
DI1E4
DI1E5


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
La Jolla, CA 92037
Bd13B CDC SPALU
C 09132 88952900
5/19/75
8 OF 9
MALU
FPAC
MB MUX
BREG
M/D REG


CONTROL DATA
SMALL COMPUTER DEVELOPMENT DIVISION
LOGIC DWG NO. Bd138 CDC SPALU
REV 01 5/19/75
C 09132
DWG NO. 88952900
SHEET 9 OF 9
LOGIC TYPE

This section of the manual contains the wire list and signal glossary for the hardware floating-point unit.

NOTE

Signals with names ending in an H are high true, that is, a high equals a 1 and a low equals a Ø.

Signals with names ending in an L are low true, that is, a low equals a 1 and a high equals a Ø.

Signal Name	Board Name	Pin	Description
17L			Drives Exponent Constant of $23_{10}(17_{16})$
	EXP	P1B23	to A side of Exponent ALU. (used in
	FPHMP	P1B23	FLOF)
1L			Exponent Constant of 1. (used to
	EXP	P1A11	increment the exponent)
	FPHMP	P1A11	
1M1L			Mode Control to Mantissa ALU 74181's
	FPHMP	P2A12	
	SPALU	P2A12	
	DPALU	P2A12	
1SØL			Function Selects for Mantissa 74181's
	FPHMP	P2B15	
	SPALU	P2B15	
	DPALU	P2B15	
1S1L			Function Selects for Mantissa 74181's
	FPHMP	P2B17	
	SPALU	P2B17	
	DPALU	P2B17	
1S2L			Function Selects for Mantissa 74181's
	FPHMP	P2A21	
	SPALU	P2A21	
	DPALU	P2A21	
1S3L			Function Selects for Mantissa 74181's
	FPHMP	P2B19	
	SPALU	P2B19	
	DPALU	P2B19	
7FL			Exponent Constant of $127_{10}(7F_{16})$
	EXP	P1A14	(used for forcing maximum in case of
	FPHMP	P1A14	overflow)
8ØL			Exponent Constant of -127_{10} (80_{16})
	EXP	P1B17	(used to force zero result in case of
	FPHMP	P1B17	zero mantissa or underflow)
AØ			A/Q Data Bus
	AQ	P1AØ3	
A1			A/Q Data Bus
	AQ	P1BØ1	

A1Ø			A/Q Data Bus
	AQ	P1BØ6	
A11			A/Q Data Bus
	AQ	P1AØ5	
A12			A/Q Data Bus
	AQ	P1AØ4	
A13			A/Q Data Bus
	AQ	P1BØ9	
A14			A/Q Data Bus
	AQ	P1B1Ø	
A15			A/Q Data Bus
	AQ	P1A11	
A2			A/Q Data Bus
	AQ	P1BØ2	
A3			A/Q Data Bus
	AQ	P1BØ6	
A4			A/Q Data Bus
	AQ	P1AØ7	
A5			A/Q Data Bus
	AQ	P1AØ1	
A6			A/Q Data Bus
	AQ	P1AØ2	
A7			A/Q Data Bus
	AQ	P1BØ3	
A8			A/Q Data Bus
	AQ	P1BØ4	
A9			A/Q Data Bus
	AQ	P1BØ5	
ACLK1L			Mantissa FPAC Clock Bits Ø and 9 to 23
	EXP	P2B24	
	FPHMP	P2B24	
	SPALU	P2B24	
ACLK2L			FPAC Clock Bits 24 to 31
	EXP	P2A16	
	FPHMP	P2A16	
	SPALU	P2A16	

ACLK3L			FPAC Clock Bits 32 to 47
	EXP	P2B12	
	FPHMP	P2B12	
	SPALU	P2B12	
	DPALU	P2B12	
ACLK4L			FPAC Clock Bits 48 to 51
	EXP	P2A14	
	FPHMP	P2A14	
	DPALU	P2A14	
AD0			DSA Address Bus
	DSA	P1B23	
	ADDALU	P1B23	
AD1			DSA Address Bus
	DSA	P1B24	
	ADDALU	P1B24	
AD10			DSA Address Bus
	DSA	P1A25	
	ADDALU	P1A25	
AD11			DSA Address Bus
	DSA	P1A26	
	ADDALU	P1A26	
AD12			DSA Address Bus
	DSA	P1A27	
	ADDALU	P1A27	
AD13			DSA Address Bus
	DSA	P1A28	
	ADDALU	P1A28	
AD14			DSA Address Bus
	DSA	P1A30	
	ADDALU	P1A30	
AD15			DSA Address Bus
	DSA	P1A31	
	ADDALU	P1A31	
AD2			DSA Address Bus
	DSA	P1B25	
	ADDALU	P1B25	
AD3			DSA Address Bus
	DSA	P1B26	
	ADDALU	P1B26	
AD4			DSA Address Bus
	DSA	P1B27	
	ADDALU	P1B27	

AD5			DSA Address Bus
	DSA	P1B28	
	ADDALU	P1B28	
AD6			DSA Address Bus
	DSA	P1B3Ø	
	ADDALU	P1B3Ø	
AD7			DSA Address Bus
	DSA	P1B31	
	ADDALU	P1B31	
AD8			DSA Address Bus
	DSA	P1A23	
	ADDALU	P1A23	
AD9			DSA Address Bus
	DSA	P1A24	
	ADDALU	P1A24	
ADAENBL			Enables ADDR Bd MUX to A side of
	DSA	P1B13	ADDR ALU
	ADDALU	P1B13	
ADATAH			Enables output of PCR/IR MUX on
	DSA	P2A26	ADDR Bd to HFPU DATA Bus
	ADDALU	P2A26	
ADBENBL			Enables ADDR Bd MUX to B side of
	DSA	P1A12	ADDR ALU
	ADDALU	P1A12	
ADOUTL			Enables output of ADDR ALU to DSA
	DSA	P2A13	Address Bus
	ADDALU	P2A13	
ASØL			FPAC mantissa shift register mode control
	EXP	P2A23	
	FPHMP	P2A23	
	SPALU	P2A23	
	DPALU	P2A23	
AS1L			FPAC mantissa shift register mode control
	EXP	P2B23	
	FPHMP	P2B23	
	SPALU	P2B23	
	DPALU	P2B23	
BCENBH			B Register Clock Enable from FPH MP
	EXP	P1B18	
	FPHMP	P1B18	

BCLKH			Clock to B Register on SP and DP ALU
	EXP	P2B25	
	SPALU	P2B25	
	DPALU	P2B25	
BENBL			Enables B Register to B side of 74181's
	FPHMP	P2B26	on SP ALU and DP ALU
	SPALU	P2B26	
	DPALU	P2B26	
BROM16H			Buffered ROM bit 16 (Master control
	AQ	P2A15	micro-processor)
	DSA	P2A15	
BSØL			Mode Control for B Register
	FPHMP	P2A24	
	SPALU	P2A24	
	DPALU	P2A24	
BS1L			Mode Control for B Register
	FPHMP	P2A25	
	SPALU	P2A25	
	DPALU	P2A25	
CCH			Master Control Consecutive Cycle Request
	AQ	P2B19	to DSA
	DSA	P2B19	
CCRCLKL			Current Command Register clock
	DPALU	P2A18	
	AQ	P2A18	
	DSA	P2A18	
CCRRDH			Current Command Register Read
	DPALU	P2B1Ø	
	DSA	P2B1Ø	
CKSANL			Clock Sign of Answer
	FPHMP	P2A19	
	SPALU	P2A19	
CLK2L			2nd Clock of Master Control micro-processor
	AQ	P2B29	Instruction Cycle
	DSA	P2B29	
CN11			End-around carry bit for mantissa ALU
	SPALU	P2A29	
	DPALU	P2A29	
CNTLDL			Load Shift-Counter
	EXP	P1A19	
	FPHMP	P1A19	

CSL			COLD START command from A/Q to Master Control.
	AQ	P2B28	
	DSA	P2B28	
D0			DSA Data Bus
	DSA	P1AØ3	
D1			DSA Data Bus
	DSA	P1BØ1	
D1Ø			DSA Data Bus
	DSA	P1BØ6	
D11			DSA Data Bus
	DSA	P1AØ5	
D12			DSA Data Bus
	DSA	P1AØ4	
D13			DSA Data Bus
	DSA	P1BØ9	
D14			DSA Data Bus
	DSA	P1B1Ø	
D15			DSA Data Bus
	DSA	P1A11	
D2			DSA Data Bus
	DSA	P1BØ2	
D3			DSA Data Bus
	DSA	P1AØ6	
D4			DSA Data Bus
	DSA	P1AØ7	
D5			DSA Data Bus
	DSA	P1AØ1	
D6			DSA Data Bus
	DSA	P1AØ2	
D7			DSA Data Bus
	DSA	P1BØ3	
D8			DSA Data Bus
	DSA	P1BØ4	
D9			DSA Data Bus
	DSA	P1BØ5	

DATAØH			HFPU Internal Data Bus
	SPALU	P2BØ1	
	DPALU	P2BØ1	
	AQ	P2BØ1	
	DSA	P2BØ1	
	ADDALU	P2BØ1	
DATA1H			HFPU Internal Data Bus
	SPALU	P2BØ2	
	DPALU	P2BØ2	
	AQ	P2BØ2	
	DSA	P2BØ2	
	ADDALU	P2BØ2	
DATA1ØH			HFPU Internal Data Bus
	EXP	P2AØ4	
	SPALU	P2AØ4	
	DPALU	P2AØ4	
	AQ	P2AØ4	
	DSA	P2AØ4	
	ADDALU	P2AØ4	
DATA11H			HFPU Internal Data Bus
	EXP	P2AØ5	
	SPALU	P2AØ5	
	DPALU	P2AØ5	
	AQ	P2AØ5	
	DSA	P2AØ5	
	ADDALU	P2AØ5	
DATA12H			HFPU Internal Data Bus
	EXP	P2AØ6	
	FPHMP	P2AØ6	
	SPALU	P2AØ6	
	DPALU	P2AØ6	
	AQ	P2AØ6	
	DSA	P2AØ6	
	ADDALU	P2AØ6	
DATA13H	EXP	P2AØ7	HFPU Internal Data Bus
	FPHMP	P2AØ7	
	SPALU	P2AØ7	
	DPALU	P2AØ7	
	AQ	P2AØ7	
	ADDALU	P2AØ7	
DATA14H			HFPU Internal Data Bus
	EXP	P2AØ8	
	FPHMP	P2AØ8	
	SPALU	P2AØ8	
	DPALU	P2AØ8	
	AQ	P2AØ8	
	DSA	P2AØ8	
	ADDALU	P2AØ8	

DATA15H			HFPU Internal Data Bus
	EXP	P2AØ9	
	SPALU	P2AØ9	
	DPALU	P2AØ9	
	AQ	P2AØ9	
	DSA	P2AØ9	
	ADDALU	P2AØ9	
DATA2H			HFPU Internal Data Bus
	SPALU	P2BØ4	
	DPALU	P2BØ4	
	AQ	P2BØ4	
	DSA	P2BØ4	
	ADDALU	P2BØ4	
DATA3H			HFPU Internal Data Bus
	SPALU	P2BØ5	
	DPALU	P2BØ5	
	AQ	P2BØ5	
	DSA	P2BØ5	
	ADDALU	P2BØ5	
DATA4H			HFPU Internal Data Bus
	SPALU	P2BØ6	
	DPALU	P2BØ6	
	AQ	P2BØ6	
	DSA	P2BØ6	
	ADDALU	P2BØ6	
DATA5H			HFPU Internal Data Bus
	FPHMP	P2BØ7	
	SPALU	P2BØ7	
	DPALU	P2BØ7	
	AQ	P2BØ7	
	DSA	P2BØ7	
	ADDALU	P2BØ7	
DATA6H			HFPU Internal Data Bus
	SPALU	P2BØ8	
	DPALU	P2BØ8	
	AQ	P2BØ8	
	DSA	P2BØ8	
	ADDALU	P2BØ8	
			HFPU Internal Data Bus
DATA7H	EXP	P2BØ9	
	SPALU	P2BØ9	
	DPALU	P2BØ9	
	AQ	P2BØ9	
	DSA	P2BØ9	
	ADDALU	P2BØ9	

DATA8H			HFPU Internal Data Bus
	EXP	P2AØ1	
	SPALU	P2AØ1	
	DPALU	P2AØ1	
	AQ	P2AØ1	
	DSA	P2AØ1	
	ADDALU	P2AØ1	
DATA9H			HFPU Internal Data Bus
	EXP	P2AØ2	
	SPALU	P2AØ2	
	DPALU	P2AØ2	
	AQ	P2AØ2	
	DSA	P2AØ2	
	ADDALU	P2AØ2	
DATAOUT1H			Read FPAC Ø-15 to HFPU DATA
	EXP	P2A11	
	SPALU	P2A11	
	AQ	P2A11	
DATAOUT2H			Read FPAC 16-31
	SPALU	P2B1Ø	
	AQ	P2B1Ø	
DATAOUT3H			Read FPAC 32-47
	SPALU	P2B13	
	DPALU	P2B13	
	AQ	P2B13	
DIVL			FPH MP Hardware Divide Command Line
	EXP	P1AØ6	
	FPHMP	P1AØ6	
	SPALU	P1AØ8	
DMAH			Master Control DSA instruction execution
	AQ	P2B3Ø	
	DSA	P2B3Ø	
DPL			Double Precision bit from FSR
	EXP	P2B16	
	FPHMP	P2B16	
	SPALU	P2B16	
	DPALU	P2B16	
	AQ	P2B16	
	DSA	P2B16	
DSA PRIORITY			DSA Bus Signals
	DSA	P1B12	
DSA PROG PROT			DSA Bus Signals
	DSA	P1B14	

DSA REQ			DSA Bus Signals
	DSA	P1A15	
DSA RESUME			DSA Bus Signals
	DSA	P1A13	
DSA WRITE			DSA Bus Signals
	DSA	P1B21	
EACLKL			Exponent Accumulator Clock
	EXP	P1B19	
	FPHMP	P1B19	
EBENBL			Enables output of Exponent B Reg to B side of 74181
	EXP	P1A17	
	FPHMP	P1A17	
EGTL			Exponent ALU output greater than zero
	EXP	P1AØ9	
	FPHMP	P1AØ9	
EOVFL			Exponent Overflow
	EXP	P1B12	
	FPHMP	P1B12	
ESØL			Exponent ALU function Select lines
	EXP	P1A22	
	FPHMP	P1A22	
ES1L			Exponent ALU function Select lines
	EXP	P1B21	
	FPHMP	P1B21	
ES2L			Exponent ALU function Select lines
	EXP	P1A21	
	FPHMP	P1A21	
ES3L			Exponent ALU function Select lines
	EXP	P1A2Ø	
	FPHMP	P1A2Ø	
ESML			Exponent ALU function Select lines
	EXP	P1B22	
	FPHMP	P1B22	
ETBH			Exponent difference Too Big
	EXP	P1A12	
	FPHMP	P1A12	
EUNFL			Exponent UNder Flow
	EXP	P1B13	
	FPHMP	P1B13	

EXNXTH			Execute Next op Code. Enable from Master micro-processor
	DPALU	P1A3Ø	
	AQ	P1A3Ø	
FADRØH			Starting Address Lines for FPH MP
	FPHMP	P2B29	
	DPALU	P2B29	
FADR1H			Starting Address Lines for FPH MP
	FPHMP	P2A28	
	DPALU	P2A28	
FADR2H			Starting Address Lines for FPH MP
	FPHMP	P2B28	
	DPALU	P2B28	
FADR3H			Starting Address Lines for FPH MP
	FPHMP	P2A11	
	DPALU	P2A11	
FADR4H			Starting Address Lines for FPH MP
	FPHMP	P1B28	
	DPALU	P1B28	
FIXL			Control to ETB Comparator used during FIX
	EXP	P1A18	
	FPHMP	P1A18	
FL			Exponent Constant of $15_{10}(F_{16})$
	EXP	P1A23	
	FPHMP	P1A23	
FSRCLKL			Function Status Register Clock
	FPHMP	P1B29	
	AQ	P1B29	
	DSA	P1B29	
FSRRDH			FSR Read
	FPHMP	P1B2Ø	
	AQ	P1B2Ø	
	DSA	P1B2Ø	
FSTARTL			Start Command to FPHMP
	EXP	P1BØ8	
	FPHMP	P1BØ8	
	AQ	P1BØ8	
GL			Carry Generate out of DPALU
	SPALU	P2A17	
	DPALU	P2A17	

GND			Ground
	EXP	P1A29	
	EXP	P1BØ1	
	EXP	P1B11	
	EXP	P2AØ3	
	EXP	P2B21	
	EXP	P2B31	
	FPHMP	P1A29	
	FPHMP	P1BØ1	
	FPHMP	P1B11	
	FPHMP	P2AØ3	
	FPHMP	P2B21	
	FPHMP	P2B31	
	SPALU	P1A29	
	SPALU	P1B11	
	SPALU	P2AØ3	
	SPALU	P2B21	
	SPALU	P2B31	
	DPALU	P1A29	
	DPALU	P1B11	
	DPALU	P2AØ3	
	DPALU	P2B21	
	DPALU	P2B31	
	AQ	P1A29	
	AQ	P1B11	
	AQ	P2AØ3	
	AQ	P2B21	
	AQ	P2B31	
	DSA	P1A29	
	DSA	P1B11	
	DSA	P2AØ3	
	DSA	P2B21	
	DSA	P2B31	
	ADDALU	P1A29	
	ADDALU	P1BØ1	
	ADDALU	P1B11	
	ADDALU	P2AØ3	
	ADDALU	P2B21	
	ADDALU	P2B31	
HADRØH			Master Control micro-processor Instruction
	DPALU	P2B2Ø	Address lines
	AQ	P2B2Ø	
	DSA	P2B2Ø	
HADR1H			Master Control micro-processor Instruction
	DPALU	P2A2Ø	Address lines
	AQ	P2A2Ø	
	DSA	P2A2Ø	
HADR2H			Master Control
	DPALU	P2B22	Micro-Processor
	AQ	P2B22	instruction Address
	DSA	P2B22	lines.

HADR3H			Master Control micro-processor Instruction
	DPALU	P2A19	Address lines
	AQ	P2A19	
	DSA	P2A19	
HADR4H			Master Control micro-processor Instruction
	DPALU	P2B18	Address lines
	AQ	P2B18	
	DSA	P2B18	
HADR5H			Master Control micro-processor Instruction
	DPALU	P2A22	Address lines
	AQ	P2A22	
	DSA	P2A22	
HADR6H			Extra Master Control Starting address
	DPALU	P1A31	ROM outputs used to generate the Two Least
	AQ	P1A31	Significant Bits of the FPH MP Starting address
HADR7H			Extra Master Control Starting address
	DPALU	P1B31	ROM outputs used to generate the Two Least
	AQ	P1B31	Significant Bits of the FPH MP Starting address
HALTL			FPH MP HALT instruction execution
	EXP	P1AØ4	
	FPHMP	P1AØ4	
HDWRL			FPH MP Hardware micro-instruction (SHIFT,
	EXP	P1A15	NORM, MULT, DIV)
	FPHMP	P1A15	
I/O MODEL			Input/Output interrupt of Master Control
	AQ	P2A27	
	DSA	P2A27	
IADATAL			A/Q interface drive of ADATAH (Enable
	AQ	P2A17	PCR/IR to HFPU DATA Bus).
	DSA	P2A17	
ICCRRDL			A/Q interface Drive of CCRRDH
	AQ	P2A14	
	DSA	P2A14	
IFSRRDL			A/Q interface Drive of FSRRDH
	AQ	P2B17	
	DSA	P2B17	
INSCLKL			FPH MP Instruction Clock
	EXP	P1BØ5	
	FPHMP	P1BØ5	
IR2H			Index Register Times 2
	DSA	P2B13	
	ADDALU	P2B13	
IR3H			Index Register Times 3
	DSA	P2B11	
	ADDALU	P2B11	

IRCLKL			Index Register Clock
	AQ	P2B15	
	DSA	P2B15	
	ADDALU	P2B15	
IRCLRL			Index Register Clear
	DSA	P2BØ7	
	ADDALU	P2BØ7	
IRSH			Index Register Sign
	DPALU	P2A16	
	ADDALU	P2A16	
ISELL			Select Control to PCR/IR MUX
	AQ	P2A28	
	DSA	P2A28	
	ADDALU	P2A28	
IZEROH			Index Register = Zero
	DPALU	P2AlØ	
	ADDALU	P2AlØ	
J-YESH			Jump Yes . Master Control jump condition true.
	DPALU	P2B14	
	AQ	P2B14	
LEFTL			Shift Left mode Control to CCR
	DPALU	P1B27	
	AQ	P1B27	
MA=BH			Mantissa A=B. Indicates that outputs of Mantissa ALU are all high
	EXP	P2A15	
	FPHMP	P2A15	
	SPALU	P2A15	
	DPALU	P2A15	
MACØH			Mantissa Accumulator Bit Ø.
	EXP	P2A26	
	FPHMP	P2A26	
	SPALU	P2A26	
	DPALU	P2A26	
MAC35L			Mantissa Accumulator Bit 35.
	SPALU	P1A26	
	DPALU	P1A26	
MAC36L			Mantissa Accumulator Bit 36.
	SPALU	P1A25	
	DPALU	P1A25	

MBØH			Mantissa B Register Bit Ø
	FPHMP	P2B22	
	SPALU	P2B22	
MB35L			Mantissa B Register Bit 35
	SPALU	P1B29	
	DPALU	P1B29	
MB36L			Mantissa B Register Bit 36
	SPALU	P1B25	
	DPALU	P1B25	
MB51L			Mantissa B Register Bit 51
	SPALU	P1B24	
	DPALU	P1B24	
MC			A/Q Bus Master Clear
	AQ	P1B23	
MCLRL			HFPU Master Clear. Inclusive OR of MC and PCLR
	EXP	P1A1Ø	
	FPHMP	P1A1Ø	
	DPALU	P1A1Ø	
	AQ	P1A1Ø	
	DSA	P1A1Ø	
MDØH			Multiplicand/Divisor Register Bit Ø
	FPHMP	P2A3Ø	
	SPALU	P2A3Ø	
MDCLK1L			M/D Register Clock, Bits Ø to 15
	EXP	P2B11	
	SPALU	P2B11	
	DPALU	P2B11	
	AQ	P2B11	
MDCLK2L			M/D Register Clock, Bits 16 to 31
	SPALU	P1B3Ø	
	AQ	P1B3Ø	
MDCLK3L			M/D Register Clock, Bits 32 to 47
	SPALU	P1A24	
	DPALU	P1A24	
	AQ	P1A24	
MDENBL			Enables M/D Reg to Mantissa ALU B side
	FPHMP	P2B3Ø	
	SPALU	P2B3Ø	
	DPALU	P2B3Ø	
MEL			Look-Ahead Buffer Register Memory Enable
	AQ	P2A25	
	DSA	P2A25	
	ADDALU	P2A25	

MIRCLKL			Master Control Instruction Register Clock
	DPALU	P2A3Ø	
	AQ	P2A3Ø	
	DSA	P2A3Ø	
MPSNSH			Multiply Sense line
	EXP	P2B14	
	SPALU	P2B14	
MSØL			Mantissa Summer Bit Ø
	EXP	P2A22	
	SPALU	P2A22	
MULTL			FPH MP Hardware Multiply Command line
	EXP	P1AØ5	
	FPHMP	P1AØ5	
	SPALU	P1BØ8	
NORML			FPH MP Normalize Command
	EXP	P1B15	
	FPHMP	P1B15	
NRMDH			Normalized indicates that FPAC Bits Ø and 9 differ
	EXP	P2A27	
	FPHMP	P2A27	
	SPALU	P2A27	
NRMM1H			Normalized minus one. FPAC Bits Ø and 10 differ.
	EXP	P2B29	
	SPALU	P2B29	
OPCLRL			Operand Byte Counter Clear
	AQ	P2BØ3	
	DSA	P2BØ3	
PCRCLKL			Program Counter Register Clock
	AQ	P2B26	
	DSA	P2B26	
	ADDALU	P2B26	
PCRLL			Program Counter Register Load enable
	DSA	P1B22	
	ADDALU	P1B22	
PICKL			Latch for storing sign of the exponent
	EXP	P1AØ3	difference for use in ADD/SUB to Pick the
	FPHMP	P1AØ3	larger exponent
PL			Propagated Carry from DPALU to SP ALU
	SPALU	P2BØ3	
	DPALU	P2BØ3	
PROG PROT			A/Q Bus Protected Command Line
	AQ	P1A23	

PROTECT FAULT L			DSA Bus Protect Fault
	DSA	P1B17	
PROTH			FSR Protect Mode Status Bit
	AQ	P2B12	
	DSA	P2B12	
Q0H			A/Q Bus Address lines
	AQ	P1A12	
Q10H			A/Q Bus Address lines
	AQ	P1A17	
Q1H			A/Q Bus Address lines
	AQ	P1B12	
Q2H			A/Q Bus Address lines
	AQ	P1A13	
Q3H			A/Q Bus Address lines
	AQ	P1B13	
Q4H			A/Q Bus Address lines
	AQ	P1A14	
Q5H			A/Q Bus Address lines
	AQ	P1B14	
Q6H			A/Q Bus Address lines
	AQ	P1A15	
Q7H			A/Q Bus Address lines
	AQ	P1B15	
Q8H			A/Q Bus Address lines
	AQ	P1A16	
Q9H			A/Q Bus Address lines
	AQ	P1B16	
QL			Serial Quotient Bit
	SPALU	P2B27	
	DPALU	P2B27	
RA0L			Look-Ahead Buffer Address Lines (RAM Address)
	DSA	P1A08	
	ADDALU	P1A08	
RA1L			Look-Ahead Buffer Address Lines (RAM Address)
	DSA	P1B08	
	ADDALU	P1B08	
RACTVL			Reset the Active Bit of FSR
	AQ	P2A23	
	DSA	P2A23	

READ			A/Q READ
	AQ	P1A21	
REJECT			A/Q Reject
	AQ	P1B22	
REPLY			A/Q Reply
	AQ	P1A22	
RESUMEL			DSA Resume
	AQ	P2A12	
	DSA	P2A12	
RUNL			FPH MP Active line
	EXP	P1AØ8	
	FPHMP	P1AØ8	
	AQ	P1AØ8	
SACTVL			Set the Active Bit in FSR
	AQ	P2B25	
	DSA	P2B25	
SAH			Select A mode for Address ALU
	DSA	P2A11	
	ADDALU	P2A11	
SARENBL			Starting Address Register Enable. (Reads Location Ø of Look-ahead Buffer)
	DPALU	P2A13	
	AQ	P2A13	
SBØL			Summer B input Bit Ø (forces max mantissa)
	FPHMP	P2B2Ø	
	SPALU	P2B2Ø	
SB16L			Summer B input Bit 16 (FLOF overflow Result)
	FPHMP	P2A13	
	SPALU	P2A13	
SB32L			Summer B input Bit 32 (single precision Normalized Round)
	FPHMP	P2A1Ø	
	SPALU	P2A1Ø	
SB33L			Summer B input Bit 33 (single precision Un-normalized Round)
	FPHMP	P1B27	
	SPALU	P1B27	
SB48L			Summer B input Bit 48 (Double precision Nrm'd Round)
	FPHMP	P1A27	
	DPALU	P1A27	
SB49L			Summer B input Bit 49 (Double precision un-normalized Round)
	FPHMP	P1A28	
	DPALU	P1A28	

SCFWDIN			DSA Scanner lines
	DSA	P1A19	
SCFWDOUT			DSA Scanner lines
	DSA	P1B19	
SCREV IN			DSA Scanner lines
	DSA	P1B15	
SCREV OUT			DSA Scanner lines
	dSA	P1B16	
SDBPML			Set the Double Precision Bit in FSR
	AQ	P2A21	
	DSA	P2A21	
SHCLK1L			Shift Counter Clock from FPH MP
	EXP	P1BØ4	
	FPHMP	P1BØ4	
SHIFTL			FPH MP Hardware Shift Command
	EXP	P1A13	
	FPHMP	P1A13	
SPECH			Master Control execution of SPEC OpCode.
	DPALU	P1B26	Forces msb of OpCode Decode -ROM input.
	AQ	P1B26	
SPINL			Single Precision Inhibit to Master Control
	AQ	P2A29	Instruction
	DSA	P2A29	
SPTCTL			Set the Protect Bit in FSR
	AQ	P2B23	
	DSA	P2B23	
SRESETL			FPH MP System Reset
	EXP	P1BØ3	
	FPHMP	P1BØ3	
TARCLKL			Temporary Address Register Clock
	DSA	P1A16	
	ADDALU	P1A16	
TARLL			TAR Load enable
	DSA	P1A22	
	ADDALU	P1A22	
TRUINL			Jump condition true Inhibit to Master Control
	AQ	P2B27	Instruction
	DSA	P2B27	

TSELL			Select Control to DATA/TAR MUX on ADDR Bd.
	DSA	P1A14	
	ADDALU	P1A14	
VCC			+5V
	EXP	P2A31	
	FPHMP	P2A31	
	SPALU	P2A31	
	DPALU	P2A31	
	AQ	P2A31	
	DSA	P2A31	
	ADDALU	P2A31	
W=Ø			A/Q W=0
	AQ	P1A2Ø	
WEL			Write enable to Look-Ahead Buffer
	AQ	P2B24	
	DSA	P2B24	
	ADDALU	P2B24	
WRITE			A/Q Write
	AQ	P1B21	
ZOUNDL			Zero or unnormalized Divisor
	FPHMP	P2A2Ø	
	SPALU	P2A2Ø	

Address ALU Test Point Signal Glossary

Name	T.P.	Description
AΣØ	3	Address ALU Summer outputs (lsb)
AΣ1H	4	"
AΣ2H	5	"
AΣ3H	6	"
AΣ4H	24	"
AΣ5H	25	"
AΣ6H	26	"
AΣ7H	27	"
AΣ8H	36	"
AΣ9H	35	"
AΣ10H	34	"
AΣ11H	33	"
AΣ12H	54	"
AΣ13H	53	"
AΣ14H	52	"
AΣ15H	51	" (msb)
ADADAENB-L	62	MUX enable to Summer A side
ADATA-L	32	Enable PCR/IR MUX to HFPU DATA Bus
ADBENB-L	61	MUX enable to Summer B side
ADOUT-H	31	Enable Summer to DSA Address
DATAØH	7	HFPU internal Data Bus high true (lsb)
DATA1H	9	"
DATA2H	10	"
DATA3H	2	"
DATA4H	28	"
DATA5H	29	"
DATA6H	30	"
DATA7H	22	"
DATA8H	44	"
DATA9H	45	"
DATA10H	46	"
DATA11H	42	"
DATA12H	59	"
DATA13H	60	"
DATA14H	55	"
DATA15H	57	" (msb)
IR2-H	48	INDEX Register times 2
IR3-H	49	" " " 3
IRS-H	50	" " Sign
ISEL-L	18	Select IR input to PCR/IR MUX
IZERO-H	47	IR = zero
ME-L	12	Memory enable to Look-Ahead Buffer
PCRCLK-L	39	Program Counter Register Clock
PCRL-L	38	Program Counter Register Load Enable
RAØ-L	17	Look-ahead and SSAR Buffer address lines
RA1-L	15	"
TARCLK-L	43	Temporary Address Register Clock
TARL-L	37	TAR Load Enable
TSEL-L	23	Select TAR input to DATA/TAR MUX
WE-L	16	Write Enable to Look-Ahead Buffer

DSA Test Points and Signal Glossary

Name	T.P.	Description
CC-H	53	Consecutive Cycle Request
CONN-H	15	Connected to DSA (memory cycle in progress)
DIN-L	7	Enable DATA to DSA D Bus
DOUT-H	11	Enable DSA D Bus to HFPU DATA Bus
FEND-H	28	FEND Bit in FSR
HALT-H	24	Scanner Halt Flip/Flop
HOG-H	23	HOG Bit in FSR
INDX-H	25	FSR INDX mode Bit
LER-H	2	Leading edge of Resume
NEED-H	21	Need flip/flop in DSA Request logic
PTFLT-H	19	Protect Fault Bit in FSR
REL-H	27	Relative Addressing mode Bit in FSR
RES-H	6	Buffered DSA RESUME
REQ-H	14	DSA Request flip/flop
ROM16-H	36	Master Control micro-processor ROM outputs
ROM17-H	39	"
ROM18-H	38	"
ROM19-H	37	"
ROM20-H	43	"
ROM21-H	42	"
ROM22-H	41	"
ROM23-H	40	"
ROM24-H	45	"
ROM25-H	52	"
ROM26-H	51	"
ROM27-H	46	"
ROM28-H	47	"
ROM29-H	48	"
ROM30-H	49	"
ROM31-H	50	"
ROM32-H	54	"
ROM33-H	55	"
ROM34-H	57	"
ROM35-H	58	"
ROM36-H	59	"
ROM37-H	60	"
ROM38-H	61	"
ROM39-H	62	"
SCNCLR-L	5	Scanner Clear from Master Control
SCNHLT-H	44	Scanner Halt for Consecutive Cycles
SCNR-H	10	Scanner flip/flop
SET NEED-L	20	DSA Cycle initiate
TER-H	3	Trailing edge of Resume
WORD-H	22	WORD mode in FSR
WRITE-H	17	Write cycle Control
WRT-H	12	DSA write flip/flop

Name	T.P.	Description
PIN 1 = GRND		
ACTIVE-H	11	Active Bit in FSR
CLK2-L	42	Second Clock of Master Control Instr Cycle
CCRCLK-L	49	Current Command Register Clock
DADATA-H	2	Drive HFPU DATA to A/Q A Bus
DECODE-L	21	Decode of a Valid Q address
DEFINED-L	16	Indicates that the Q station Code is defined
DP-H	12	Double Precision mode Bit in FSR
FPWAIT-L	50	Wait for Floating Point execution Completion
HALT-L	38	Master control microprocessor Halt
I/O ACK-L	37	I/O Acknowledge, Master Clock Stopped
I/O MODE-L	30	I/O command being executed
MIRCLK-L	44	Master Control instruction Register clock
OPCNT-L	48	Clock to OP Byte Counter
OPDN-L	45	Op word Done, DP Byte Counter = 4
PCD-H	18	Protected Command Required
PFSR-L	20	Protected write FSR command
PROT-H	7	Protect Bit in FSR
R+W-L	17	A/Q Read or Write
RADATA-H	3	Read A/Q A Bus to HFPU DATA Bus
ROM8H	54	Master control micro-processor ROM outputs
ROM9H	55	"
ROM10H	57	"
ROM11H	58	"
ROM12H	59	"
ROM13H	60	"
ROM14H	61	"
ROM15H	62	"
RUN-L	51	FPH MP Active
SPIN-L	47	Single Precision Inhibit
START-L	15	Start command to Master Control
STKRQ-L	28	I/O Request for stop of Master Clock
STOPREQ-L	22	A/Q STOP command Pending
TACT-L	19	Test Active before Reply to A/Q Write
TRUIN-L	46	Jump Condition true inhibit
US-L	27	Q Register address defined
WCLK-L	14	Write Clock, generated to strobe data to destination on A/Q write

DP ALU Test Points and Signal Glossary

Signal	Test Point	Description
1M1H	33	Mantissa ALU 74181 function Select Lines
1SØH	45	"
1S1H	44	"
1S2H	43	"
1S3H	53	"
ACLK3-L	59	FPAC Clock 3 (Bits 32 to 47)
AENB-L	36	Select B Register input to MD/B MUX
ASØ-L	62	ACCumulator shift Register mode Controls
AS1-L	61	"
BCLK-L	48	B Register Clock
BSØ-H	46	B Register mode Controls
BS1-H	47	"
CCØH	6	Current Command Bits (actually next opByte)
CC1H	7	"
CC2H	17	"
CC3H	16	"
CCRCLK-L	24	Current Command Register Clock
CCRRD-H	15	" " " Read
DATAOUT3H	38	Enable FPAC Bits 32 to 47 onto HFPU DATA
DP-H	60	Double Precision
EXNXT-H	23	Execute Next
HADRØH	11	Master Control micro-processor ROM Address
HADR1H	12	"
HADR2H	13	"
HADR3H	14	"
HADR4H	19	"
HADR5H	20	"
IRS-H	4	Index Register Sign
IZERO-H	5	Index = Zero
JYES-H	22	Jump Condition satisfied
LEFT-L	21	Enable left shift of CCR
MA=B-H	3	Mantissa A = B
MACØH	2	Mantissa Accumulator Bit Ø
MBSTR-L	35	Enable output of MD/B Mux into B side of 74181's
MDCLK3L	34	Multiplicand/Divisor Register Clock
MIRCLK-L	18	Master Control Instruction Register Clock
MS36L	54	Mantissa Summer Bits
MS37L	55	"
MS38L	57	"
MS39L	58	"
MS40L	49	"
MS41L	50	"
MS42L	51	"
MS43L	52	"
MS44L	39	"
MS45L	40	"
MS46L	41	"

DP ALU Test Points and Signal Glossary

Signal	Test Point	Description
MS47L	42	Mantissa Summer Bits
MS48L	29	"
MS49L	30	"
MS50L	31	"
MS51L	32	"
SARENB-L	9	Starting Address ROM enable
SPEC-H	10	Master Control execution of SPEC op Byte

SP ALU Test Points and Signal Glossary

Signal	T.P.	Description
1M1H	26	Mantissa ALU 74181 Function Select Lines
1SØH	29	"
1S1H	28	"
1S2H	27	"
1S3H	35	"
ACLK1L	62	FPAC Clock 1 (Bits Ø and 9 to 15)
ACLK2L	37	FPAC Clock 2 (Bits 16 to 31)
ACLK3L	4	FPAC Clock 3 (Bits 32 to 47)
ASØL	45	Accumulator Shift Register mode Controls
AS1L	46	"
BBENB-L	40	Select B Register to B side of 74181
BCLK-L	51	B Register Clock
BSØH	49	B Register Mode Controls
BS1H	50	"
DATAOUT1H	47	Enable FPAC Ø to 15 to HFPU DATA Bus
DATAOUT2H	36	" " 16 to 31 " "
DATAOUT3H	3	" " 32 to 47 " "
DP-H	2	Double Precision
DPSNS-H	19	Double Precision Multiply Sense Bit
M/BSTB	39	Enable MD/B MUX output to B side of 74181's
MA = B-H	30	Mantissa A = B
MAC36LI-L	13	Mantissa Accumulator Left Serial Input to Bit 36
MBØL	57	Mantissa B Register Bit Ø
MB36LI-L	12	" " " Left Serial Input to Bit 36
MDCLK1L	48	Multiplicand/Divisor Reg Clock 1 (Bits Ø to 15)
MDCLK2L	38	" " " " 2 (Bits 16 to 31)
MPSNS-H	10	Multiplier Sense Bit
MSØL	58	Mantissa Summer output Bits
MS9L	59	"
MS10L	60	"
MS11L	61	"
MS12L	52	"
MS13L	53	"
MS14L	54	"
MS15L	55	"
MS16L	41	"
MS17L	42	"
MS18L	43	"
MS19L	44	"
MS20L	31	"
MS21L	32	"
MS22L	33	"
MS23L	34	"
MS24L	22	"
MS25L	23	"
MS26L	24	"
MS27L	25	"
MS28L	14	"
MS29L	15	"
MS30L	16	"

SPALU Test Points and Signal Glossary continued

Signal	T.P.	Description
MS31L	17	Mantissa Summer Output Bits
MS32L	5	"
MS33L	6	"
MS34L	7	"
MS35L	9	"
SPSNS-H	18	Single Precision Multiply Sense Bit

FPH MP Test Points and Signal Glossary

Signal	T.P.	Description
ADRØH	25	Floating Point Hardware Micro-Processor ROM Address Bits
ADR1H	26	"
ADR2H	27	"
ADR3H	28	"
ADR4H	29	"
BROM13	23	Buffered ROM Bit 13
DF	2	Divide Fault
FSRCLK-L	5	Function Status Register Clock
FSRRD-H	24	FSR Read
INSCLK-L	42	FPH MP Instruction Register Clock
JYES-L	38	Jump Condition satisfied
OVF	3	Exponent Overflow
ROMØH	6	FPH MP ROM outputs
ROM1H	7	"
ROM2H	9	"
ROM3H	10	"
ROM4H	11	"
ROM5H	12	"
ROM6H	13	"
ROM7H	14	"
ROM8H	15	"
ROM9H	16	"
ROM10H	17	"
ROM11H	18	"
ROM12H	19	"
ROM13H	20	"
ROM14H	21	"
ROM15H	22	"
ROM16H	30	"
ROM17H	31	"
ROM18H	32	"
ROM19H	33	"
ROM20H	34	"
ROM21H	35	"
ROM22H	36	"
ROM23H	37	"
ROM24H	43	"
ROM25H	44	"
ROM26H	45	"
ROM27H	46	"
ROM28H	47	"
ROM29H	48	"
ROM30H	49	"
ROM31H	50	"
ROM32H	53	"
ROM33H	54	"
ROM34H	55	"
ROM35H	57	"

FPH MP Test Points and Signal Glossary continued

Signal	T.P.	Description
ROM36H	58	FPH MP ROM outputs
ROM37H	59	"
ROM38H	60	"
ROM39H	61	"
RUN-L	62	FPH MP Active
UNF	4	Exponent Underflow

EXP and Timing Test Points and Signal Glossary

Signal	T.P.	Description
1-L	35	Exponent Constant of 1
17-L	32	Exponent Constant of $23_{10}(17_{16})$
80-L	47	Exponent Constant of $-127_{10}(80_{16})$
ACLK-L	15	Accumulator Clock
C4L	4	Fourth time state of FPH MP instruction Cycle
CNTLD-L	60	Load Shift Counter Register
DATAOUT1-H	38	Read FPAC Ø to 15 to HFPU DATA
EA(-2)-L	61	Exponent Accumulator Bits
EA(-1)-L	51	"
EA1L	50	"
EA2L	53	"
EA3L	57	"
EA4L	59	"
EA5L	39	"
EA6L	41	"
EA7L	44	"
EA8L	46	"
EACLK-L	62	Exponent Accumulator Clock
EBENB-L	37	Exponent B Register Output enable
ECLK-L	24	Clock derived from SHCLK-L during Shift
ENTR	17	Enter state of Multiply/Divide timing
ES(-2)-L	48	Exponent Summer output Bits
ES(-1)-L	49	"
ES1-L	52	"
ES2-L	54	"
ES3-L	55	"
ES4-L	58	"
ES5-L	40	"
ES6-L	42	"
ES7-L	43	"
ES8-L	45	"
F-L	33	Exponent Constant of $15_{10}(17_{16})$
F7-L	34	" $127_{10}(F7_{16})$
FSTRT-H	5	FPH MP Start Command from Master Control
HALT-L	6	Halt Bit from FPH MP
HDWCLK-L	20	Hardware Clock (during multiply/divide)
HDWST-L	13	Hardware Start (used to Start Shift, Norm, Mult or Div)
ICLK-L	12	FPH MP Instruction Register Clock
M-H	31	Exponent ALU 74181 Mode Control
MDCLK-L	23	Multiply/Divide hardware Clock
MDCLK1-L	36	Multiplicand/Divisor Clock 1 (Bits Ø to 15)
NRM-L	21	Normalize flip/flop output
RESTART-L	18	Hardware Command Line used to Restart FPH MP after SHFT, NORM, MULT or DIV
SØH	30	Exponent ALU 74181 Function Selects
S1H	29	"
S2H	28	"
S3H	27	"
SCLK-L	16	FPH MP System Clock

EXP and Timing Test Points and Signal Glossary continued

Signal	T.P.	Description
SFT-L	22	Shift Clock of Multiply/Divide timing
SHCLK-L	25	Shift Clock
SHENB-H	11	Shift Enable
SHFT	3	Shift flip/flop
SPCLK-H	19	Special clock (SCLK÷2)
SRESET-L	7	FPH MP System Reset
STCLK-L	25	Stop Shift Clock (Count = Ø)
STCNT-L	9	Step Count (used in Mult/Div)
TST	2	Test State of Mult/Divide timing
ZED-H	10	Multiply/Divide step count = Ø

Calling Sequence	String of command words and operand addresses residing in SYSTEM 17 memory that is used to direct HFPU activity.
Command-Code / OP-Byte / OP-Code	Terms used to reference individual 4-bit commands within the command words of a calling sequence.
FPAC	Floating-point accumulator register within the FHPU.
FPMP	Floating-point micro-processor. Portion of the HFPU that performs arithmetic operations on the FPAC.
MMP	Master Micro-Processor. Portion of the HFPU that interprets command codes and communicates with the A/Q, DSA interface.

The following pages contain the micro-code listings for the Floating-Point and Master micro-processors. The index below identifies the page on which the flow charts for each function will be found. The flow charts are divided into separate sections covering the operation of the two micro-processors. Figure 4.7 illustrates the conventions and mnemonics used in these flow-charts. The mnemonics used in the listings were defined in section 4.1.4.

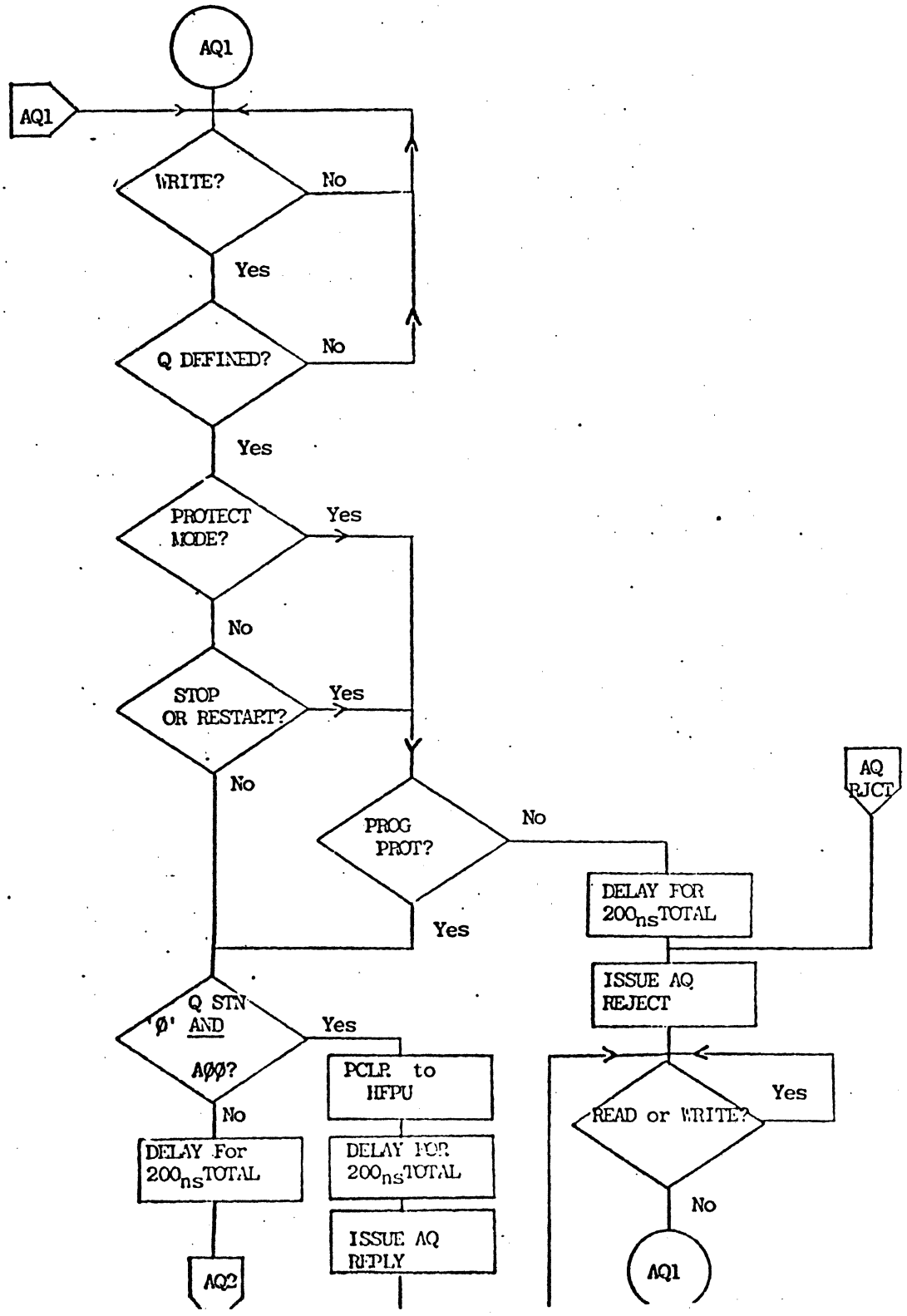
The flow charts are keyed to the micro-code listings and to the algorithm steps as defined in section 4.2. The nomenclature "LOCnn" to the left of the flow chart indicates that that step occurs at location nn in the micro-code listing.

In addition to the flow charts of the micro-code, this appendix also contains flow charts of the major logic timing loops within the HFPU.

HFPU FLOW CHART INDEX

FUNCTION or LOGIC ELEMENT	OP CODE	MASTER CONTROL PAGE	MASTER CONTROL STARTING ADDRESS	FLOATING POINT MICRO-PROCESSOR PAGE	FLOATING POINT MICRO-PROCESSOR STARTING ADDRESS
LDWD1	N/A	M8	Ø1	F2	4
LDWD2	N/A	M8	41	F2	5
LDWD3	N/A	M8	81	F2	6
STOP	N/A	M19	C2	F3	(FLST)
RESTART	N/A	M22	CØ	F2	3 (FLDD)
COLD START	N/A	M17	C3	N/A	
FPMP Timing	N/A			F14	
A/Q Timing	N/A	AQ-1			
DSA Timing	N/A	DSA-1			
Master Timing	N/A	M27			
OP Code Fetch	N/A	M17			
Execute Next	N/A	M26			
SPEC	0	M15	32	N/A	
FLOF	1	M6	62	F5	9
FIXF	2	M7	1A	F4	8
STRI	3	M14	17	N/A	
FEND	4	M18	1D	N/A	
CHMD	5	M11	1E	N/A	
NIDX	6	M11	1F	N/A	
FCOM	7	M8	D9	F4	7
FSUB	8	M3	24	F6	20
FMPY	9	M3	60	F8	1
FDIV	A	M3	A0	F9	2
FLDD	B	M3	E0	F2	3
ADDI	C	M13	2A	N/A	
FLST	D	M9	2E	F3	(FLST)
FADD	E	M3	20	F6	0
INDX	F	M12	2C	N/A	
FEND	10	M18	1D	N/A	
CACS	11	M15	35	N/A	
BRAM	12	M16	36	N/A	
BRAZ	13	M16	76	N/A	
BRAN	14	M16	F6	N/A	
BRAP	15	M16	B6	N/A	
BRIM	16	M16	33	N/A	
BRIZ	17	M16	73	N/A	
BRIN	18	M16	F3	N/A	
BRIP	19	M16	B3	N/A	
FEND	1A	M18	1D	N/A	
⋮	⋮	⋮	⋮	N/A	
FEND	1F	M18	1D	N/A	
FPMP Micro-Code				F1	
Master Micro-Code				M1	

A/Q Flow Chart - Output to HFPU



ROM LOC	Chip Location →	A/Q - D11								A/Q - D8								A/Q - D10								A/Q - D9							
	Mnemonic →	TACT	PTCT	DEFN	WFSR	UNUSED				MEM	DOUT1	DOUT2	DOUT3	ADATA	ISELR	CCRD	FSRD		STR	ME	CS	CLK1	CLK2	CLK3	IR	CCR	WE	HADR7	HADR6	HADR1	HADR0	STDPQ	SBPM	RDBPM
		1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
	A/Q Command																	A/Q Command																
0		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	RD SSAR	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	STOP	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1	1
7	RD WD3	0	0	0	1	0	0	0	0	1	1	1	0	1	1	1	1	WR WD3	0	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1
8	RD WD2	0	0	0	1	0	0	0	0	1	1	0	1	1	1	1	1	WR WD2	0	1	1	1	0	1	1	1	1	0	1	0	0	1	1	1
9	RD WD1	0	0	0	1	0	0	0	0	1	0	1	1	1	1	1	1	WR WD1	0	1	1	0	1	0	1	1	1	0	0	0	0	1	1	1
A	RD PCR	1	0	0	1	0	0	0	0	1	1	1	1	0	1	1	1	RESTART	0	0	1	1	1	1	1	1	0	1	1	0	0	1	1	1
B	RD PCR	0	0	0	1	0	0	0	0	1	1	1	1	0	1	1	1	COLD START(DP)	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
C	RD PCR	0	0	0	1	0	0	0	0	1	1	1	1	0	1	1	1	COLD START(SP)	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0
D	RD IR	1	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	WR IR	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
E	RD CCR	1	0	0	1	0	0	0	0	1	1	1	1	1	1	0	1	WR CCR	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
F	RD FSR	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	0	WR FSR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	STOP	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	WR WD3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	WR WD2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	WR WD1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1A	RESTART	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1B	COLD START(DP)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1C	COLD START(SP)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1D	WR IR	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1E	WR CCR	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1F	WR FSR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STARTING ADDRESS ROM (SAR)

DPALU BOARD LOCATION B13

ROM Location	COMMAND Code	HADR 7	HADR 6	HADR 5	HADR 4	HADR 3	HADR 2	HADR 1	HADR 0 ← Mnemonic
		1	2	3	4	5	6	7	8 ← ROM OUTPUT BIT
0	SPEC	0	0	1	1	0	0	1	0
1	FLOF	0	1	1	0	0	0	1	0
2	FIXF	0	0	0	1	1	0	1	0
3	STRI	0	0	0	1	0	1	1	1
4	FEND	0	0	0	1	1	1	0	1
5	CHMD	0	0	0	1	1	1	1	0
6	NIDX	0	0	0	1	1	1	1	1
7	FCOM	1	1	0	1	1	0	0	1
8	FSUB	0	0	1	0	0	1	0	0
9	FMPY	0	1	1	0	0	0	0	0
A	FDIV	1	0	1	0	0	0	0	0
B	FLDD	1	1	1	0	0	0	0	0
C	ADDI	0	0	1	0	1	0	1	0
D	FLST	0	0	1	0	1	1	1	0
E	FADD	0	0	1	0	0	0	0	0
F	INDX	0	0	1	0	1	1	0	0
10	FEND	0	0	0	1	1	1	0	1
11	CACS	0	0	1	1	0	1	0	1
12	BRAM	0	0	1	1	0	1	1	0
13	BRAZ	0	1	1	1	0	1	1	0
14	BRAN	1	1	1	1	0	1	1	0
15	BRAP	1	0	1	1	0	1	1	0
16	BRIM	0	0	1	1	0	0	1	1
17	BRIZ	0	1	1	1	0	0	1	1
18	BRIN	1	1	1	1	0	0	1	1
19	BRIP	1	0	1	1	0	0	1	1
1A	FEND	0	0	0	1	1	1	0	1
1B	FEND	0	0	0	1	1	1	0	1
1C	FEND	0	0	0	1	1	1	0	1
1D	FEND	0	0	0	1	1	1	0	1
1E	FEND	0	0	0	1	1	1	0	1
1F	FEND	0	0	0	1	1	1	0	1

Micro-Code ROM Truth Tables

MASTER CONTROL PAGE 1

Location	Label	DSA - A1								DSA - A4								DSA - A7								A/Q - A2								DPALU - B12								CHIP LOCATION
		39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	←Bit Position
0	RSTRT	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
1	LDWD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	1	0	
2	STOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	
3	FTCHNXT	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	
4	R2	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
5	R3	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
6	R4	1	0	1	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
7	R5	1	0	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
8	R6	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	
9	R7	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	
A	R8	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	1	1	
B	R9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
C	R10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	
D	R11	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	
E	F2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
F	S2	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
10	S3	1	1	1	0	0	0	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	
11	S4	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
12	S5	1	1	1	0	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
13	S6	1	1	1	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
14	S7	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	
15	S8	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	
16	S9	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	
17	STRI	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
18	STI2	0	1	1	0	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
19	FCOM	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	
1A	FIXF	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	
1B	FXF2	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	
1C	FXF3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	
1D	FEND	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	1	0	
1E	CHMD	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
1F	NIDX	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	

Micro-Code ROM Truth Tables

MASTER CONTROL PAGE 2

Location	Label	DSA - A2								DSA - A5								DSA - A8								A/Q - A3								DPALU - B11								CHIP LOCA
		39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit Posit
20	AMDF	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	
21	A2	1	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	
22	FLOF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	
23	FLF3	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	
24	FSUB	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	
25	A3	1	0	0	0	1	1	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1	1	0	
26	A4	0	0	1	0	1	1	1	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	
27	A5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0	
28	A6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	
29	A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	
2A	ADDI	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	
2B	ADI2	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
2C	INDX	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	
2D	INDX2	0	0	1	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
2E	FLST	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	
2F	FST2	1	1	1	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	
30	FST3	1	1	1	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	1	
31	FST4	0	1	1	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
32	SPEC	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	
33	BRI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	
34	BRI2	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	0	1	0	0	0	
35	CACS	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
36	BRA	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	1	
37	BRA2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	0	1	0	0	0	
38	FLF2	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	

Micro-Code ROM Truth Tables

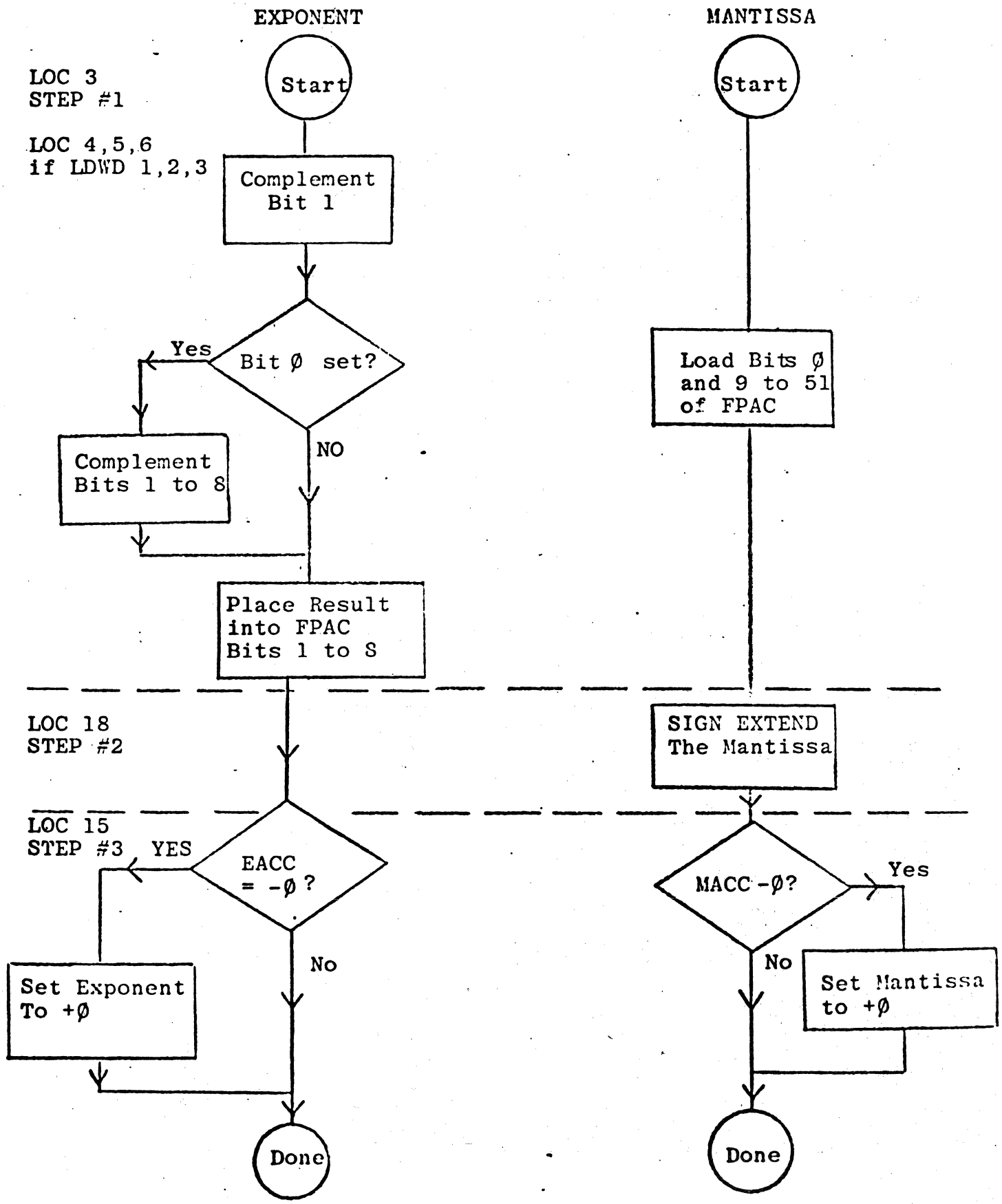
FPH MP

		FPMP - A3								FPMP - A6								FPMP - A9								FPMP - A12								FPMP - A14								CHIP LOCAT
LOCATION	LABEL	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	BIT POSITI
0	ADD/SUB	1	1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	
1	MPY	0	0	0	0	0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	
2	DIV	0	0	0	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	
3	FLDD	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
4	LDW1	1	1	0	0	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
5	LDW2	1	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
6	LDW3	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
7	FCOM	0	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	
8	FIXF	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	
9	FLOF	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	
A	FIXF2	1	1	0	0	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	
B	MPY2	1	1	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
C	MPY3	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	
D	FLOF2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	1	1	1	0	
E	FLOF3	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	
F	FOVA	1	1	1	0	1	0	0	0	0	1	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
10	DIV2	0	1	0	0	1	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	
11	NORM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
12	N2	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	0	0	1	1	
13	N3	1	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
14	N4	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
15	NHLT	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	0	1	
16	MAX	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
17	ZERO	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	
18	SEXT	1	1	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	
19		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1A	DIV3	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1	
1B	DIV4	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	
1C	ADD2	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
1D	ADD3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1	0	
1E	ADD4	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
1F	ETB	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	

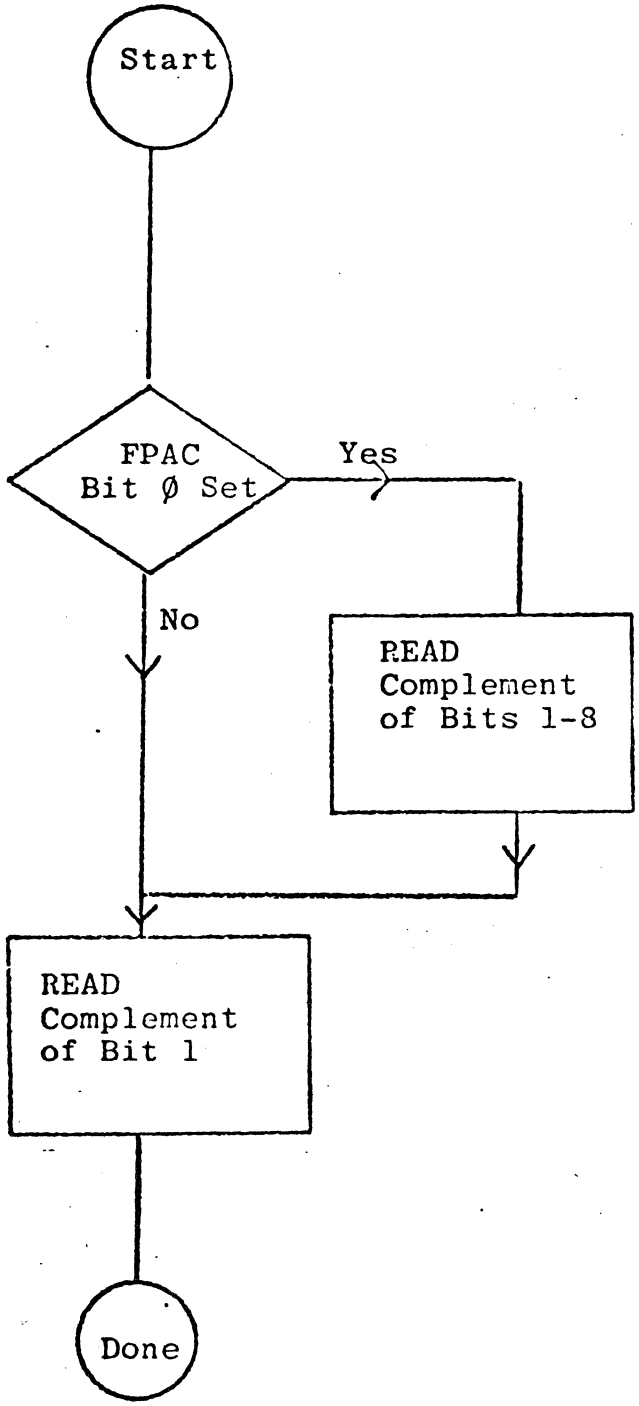
CDC SYSTEM 17 FLOATING POINT A CODE

	MALU	ALUIV	EALU	CLKS	S1,SØ	M-B	E-B	Load/Pick	Inhibits	Cond	HDWR	Disp	NxtInst
ØAdd/Sub	B	-	A-B	B	LOAD	M/DENB	EBENB	LDCNT	-	-	-	-	1C
1 MPY	Aarith	-	A+B	B,E	LOAD	-	EBENB	-	-	-	-	-	B
2 DIV	-	-	A+B	E,A1,2,3,4	RIGHT	-	1	-	-	ZOUND	-	6(OVF-20)	10
3 FLDD	B	-	B	E,A1,2,3,4	LOAD	M/DENB	EBENB	-	-	-	-	-	18
4 W1	B	-	B	E,A1,3,4	LOAD	M/DENB	EBENB	-	DPinhA3	-	-	-	18
5 W2	B	-	-	A2	LOAD	M/DENB	-	-	-	-	-	-	18
6 W3	$\overline{B}$	-	-	A3,4	LOAD	M/DENB	-	-	-	-	-	-	18
7 COM	$\overline{A}$	-	-	A1,2,3,4	LOAD	-	-	-	-	-	-	-	15
8 FIXF	B	-	B	E,A1,2,3,4	LOAD	M/DENB	17	-	-	-	-	-	A
9 FLOF	-	-	A-B	-	-	-	F	-	-	-	-	-	D
A FIXF2	B	MACØH	-	A1,3,4	LOAD	-	-	-	-	-	-	-	11(NORM)
B MPYCNT	B	MDØH	-	A1,2,3,4	LOAD	-	-	-	-	-	-	-	C
C	A+B	MBØ-H	-	-	RIGHT	M/DENB	-	-	-	-	MULT	-	11(NORM)
DFLCNT	-	-	A-B	-	-	FIX	17	LDCNT	-	EGT	-	1(F-E)	E
EFLETB	-	-	A-B	-	RIGHT	FIX	17	-	-	ETB	SHIFT	2(ZERO-15)	15 (NHLT)
F FOVA	B	MACØ-H	-	A1,2,3,4	LOAD	FXMAX	-	-	-	-	HALT	-	F
10DIVCNT	ALog	MACØ-H	A-B	E,A1,2,3,4	LOAD	-	EBENB	-	-	-	-	-	1A
11 NORM	Aarith	-	-	-	-	-	-	-	-	-	-	-	12
12 N_2	Aarith	-	A-B	-	LEFT	-	1	-	-	MA=B	NORM	4(ZERO-13)	13
13 N_3	A+B	MACØH	-	A1,2,3,4	LOAD	ROUND	-	-	-	-	-	-	14
14 N_1	-	-	A-B	E,A1,2,3,4	LEFT	-	1	-	NRMD	-	-	-	18
15 NHLT	AArith	-	Aarith	E,A1,2,3,4	LOAD	-	-	-	-	OUF/UNF	HALT	1	15
16 MAX	$\overline{B}$	MACØH	B	E,A1,2,3,4	LOAD	FLMAX	-	-	-	-	-	-	18(SEXT)
17 Zero	B	-	B	E,A1,2,3,4	LOAD	FLZERO	-	-	-	-	HALT	-	17
18 SEXT	B	MACØH	-	A3,4	LOAD	-	-	-	DPinA3	-	-	-	15(NHLT)
19	-	-	-	-	-	-	-	-	-	-	-	-	-
1A DIV	A-B	MDØH	-	-	LEFT	M/DENB	-	-	-	-	DIV	-	1B
1B	B	-	-	A1,2,3,4	LOAD	BENB	-	-	-	-	-	-	11 (NORM)
1CADCNT	-	-	B	E,B,A1234	RIGHT	-	EBENB	-	Pick SET	ETB	-	2	1D
1D SHFT	-	-	-	-	RIGHT	-	-	-	-	-	SHIFT	-	1E
1E	A+B	FSUB	A+B	E,A1,2,3,4	LOAD	BENB	1	-	-	-	-	-	11(NORM)
1F zero	B	-	-	A1,2,3,4	LOAD	-	-	Pick ENB	-	-	-	-	1E

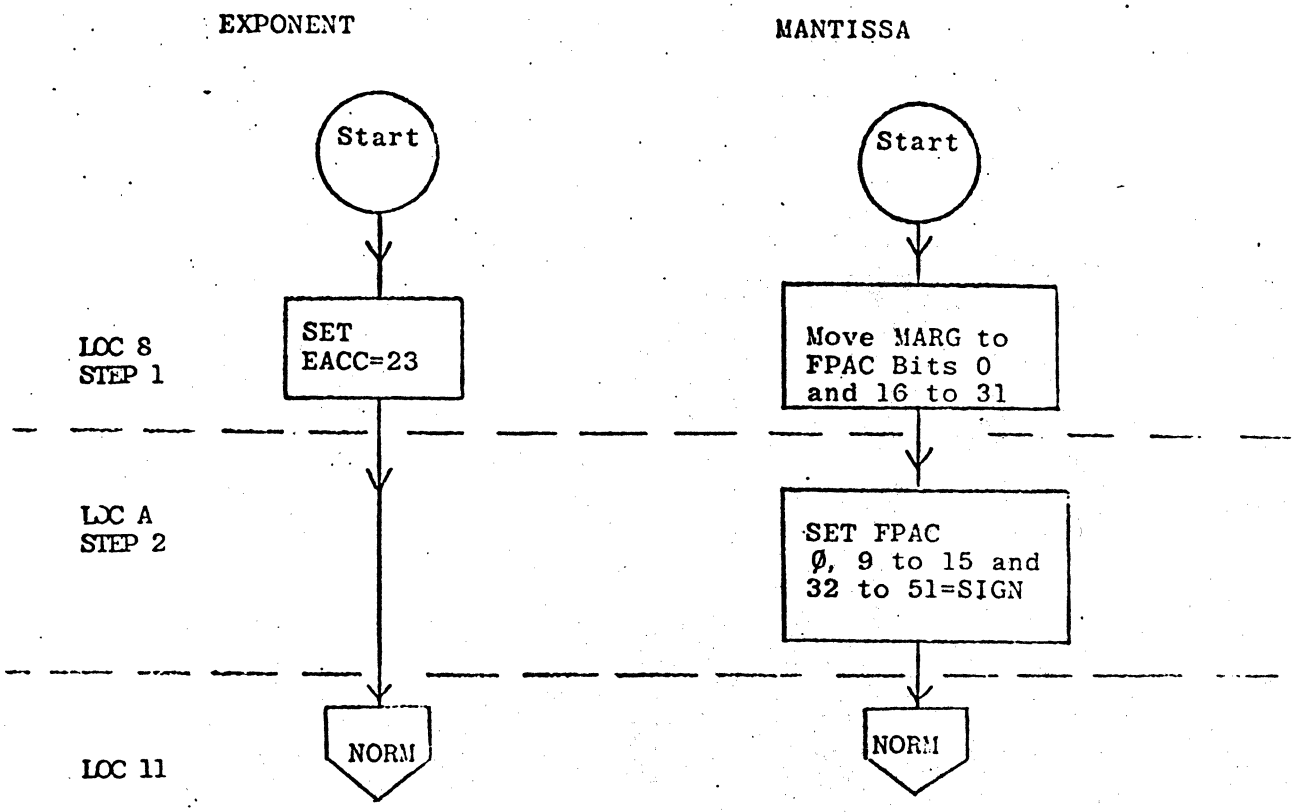
FPH FLDD and A/Q load FPAC flow chart



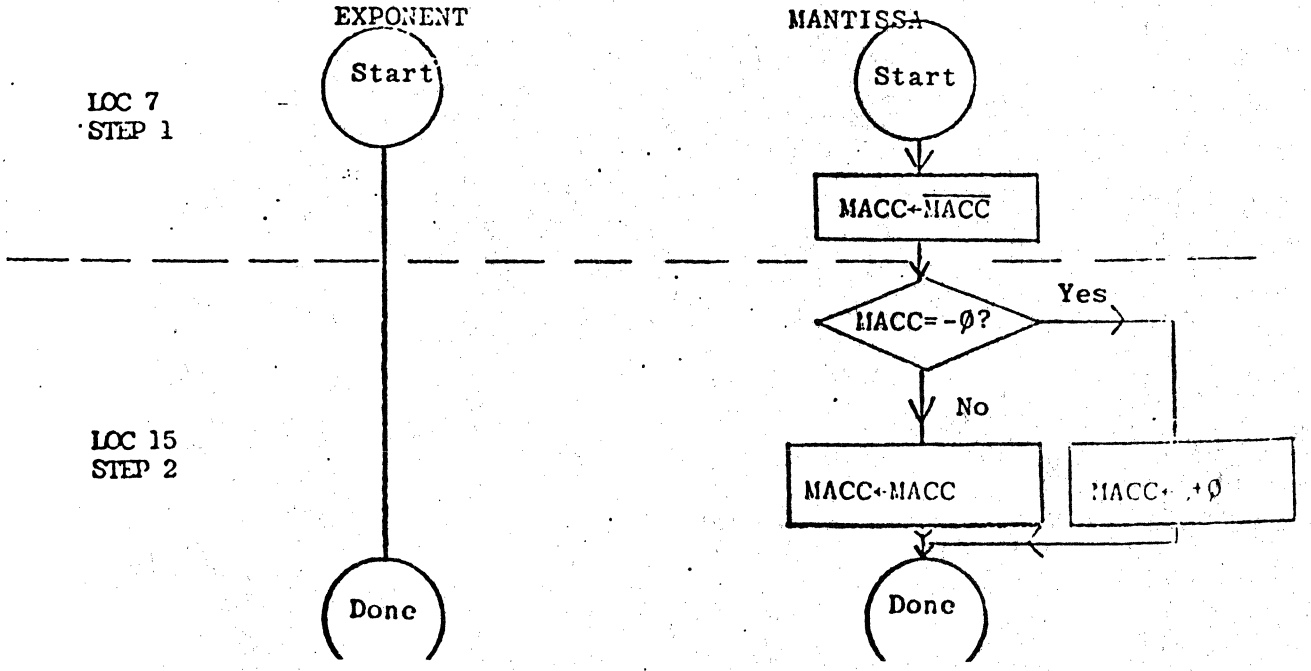
FPH FLST Logic flow chart for Read of FPAC Bits 1 to 8



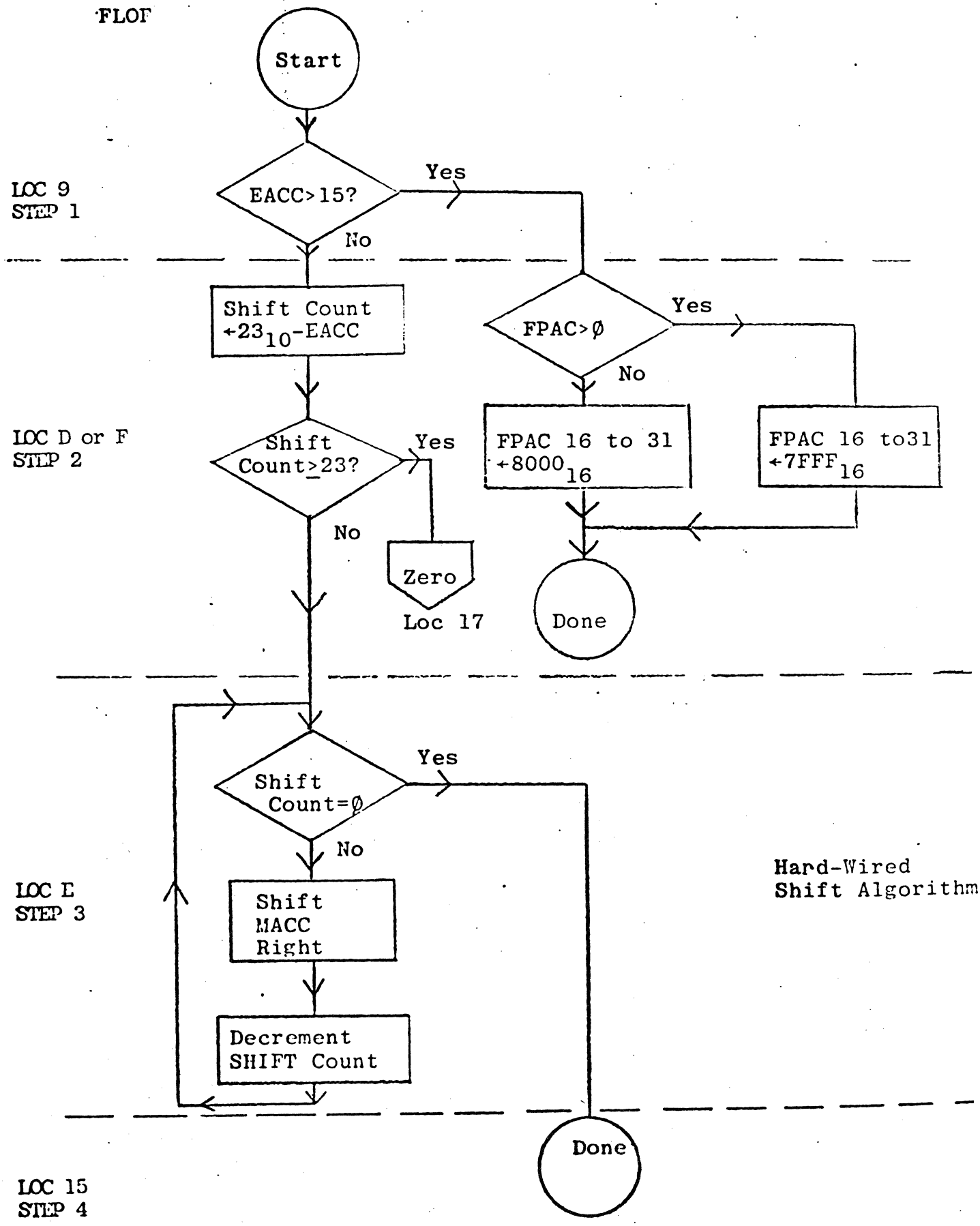
FPH FIXF Flow Chart

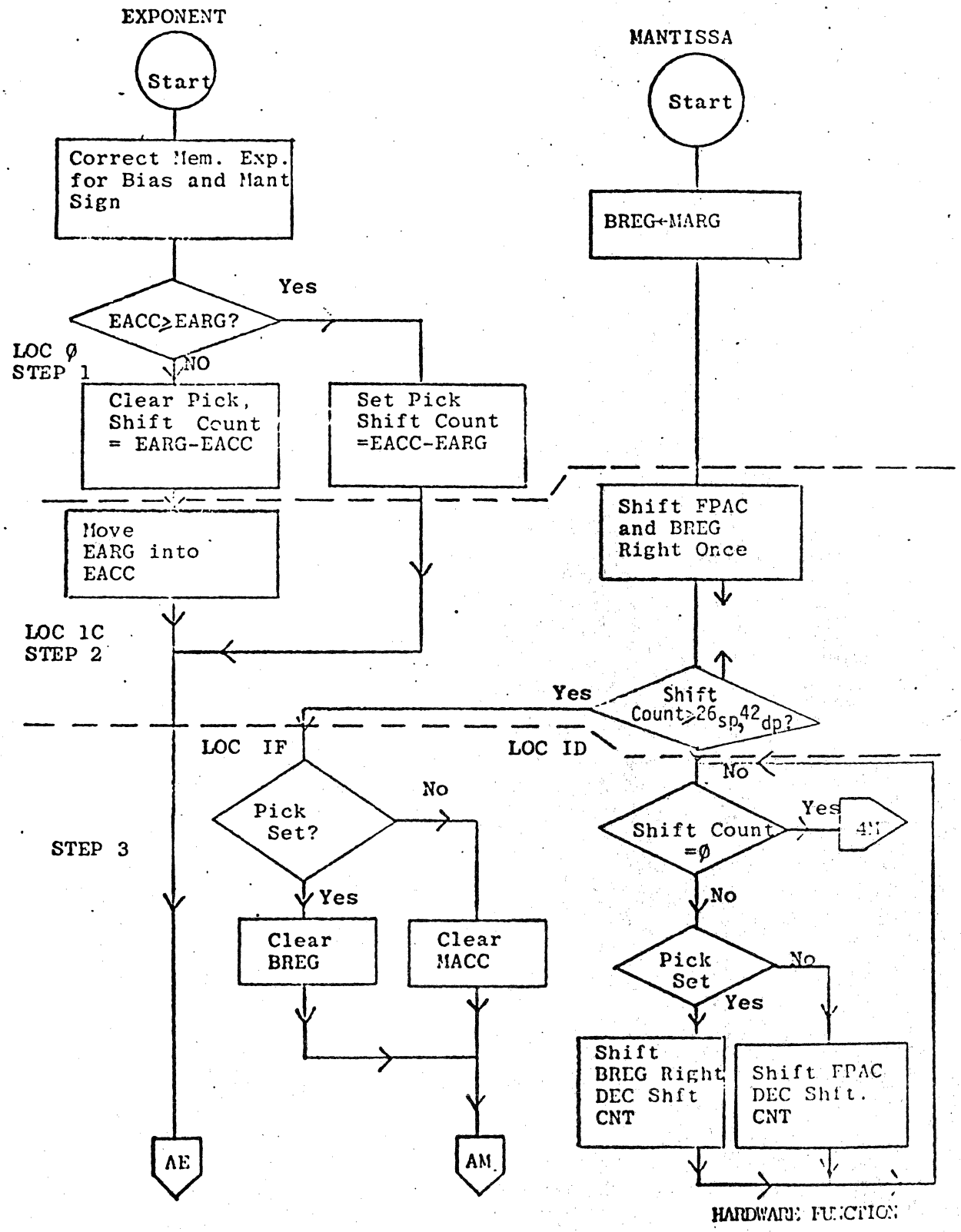


FCOM Flowchart

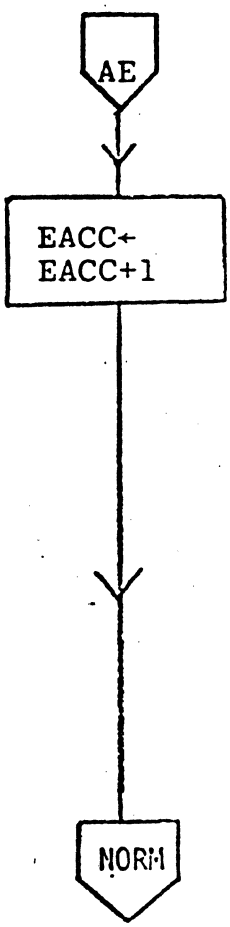


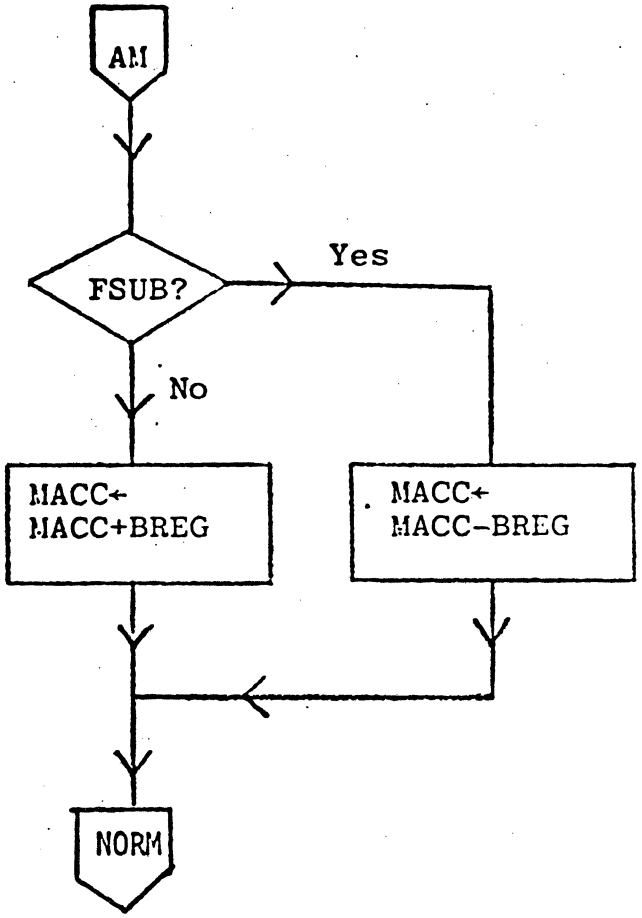
FPH Flow chart



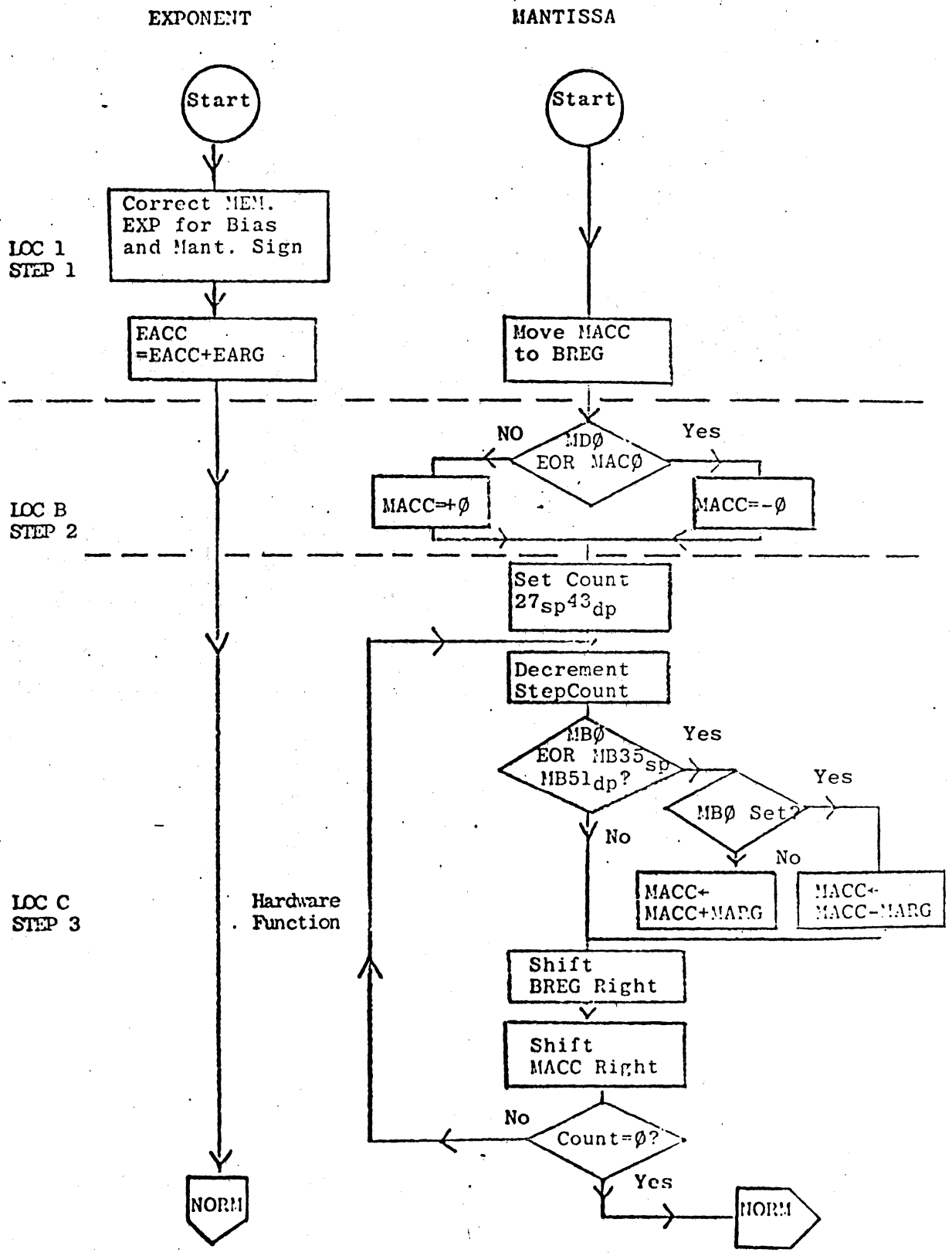

EXPONENT
Start
Correct Mem. Exp. for Bias and Mant Sign
EACC≥EARG?
Yes
NO
LOC Ø
STEP 1
Clear Pick, Shift Count = EARG-EACC
Set Pick Shift Count =EACC-EARG
Move EARG into EACC
LOC 1C
STEP 2
MANTISSA
Start
BREG←MARG
Shift FPAC and BREG Right Once
Shift Count≥26sp,42dp?
Yes
No
LOC IF
LOC ID
Pick Set?
No
Yes
Clear BREG
Clear MACC
STEP 3
Shift Count =Ø
Yes
No
Pick Set
No
Yes
Shift BREG Right DEC Shft CNT
Shift FPAC DEC Shft. CNT
AE
AM
HARDWARE FUNCTION

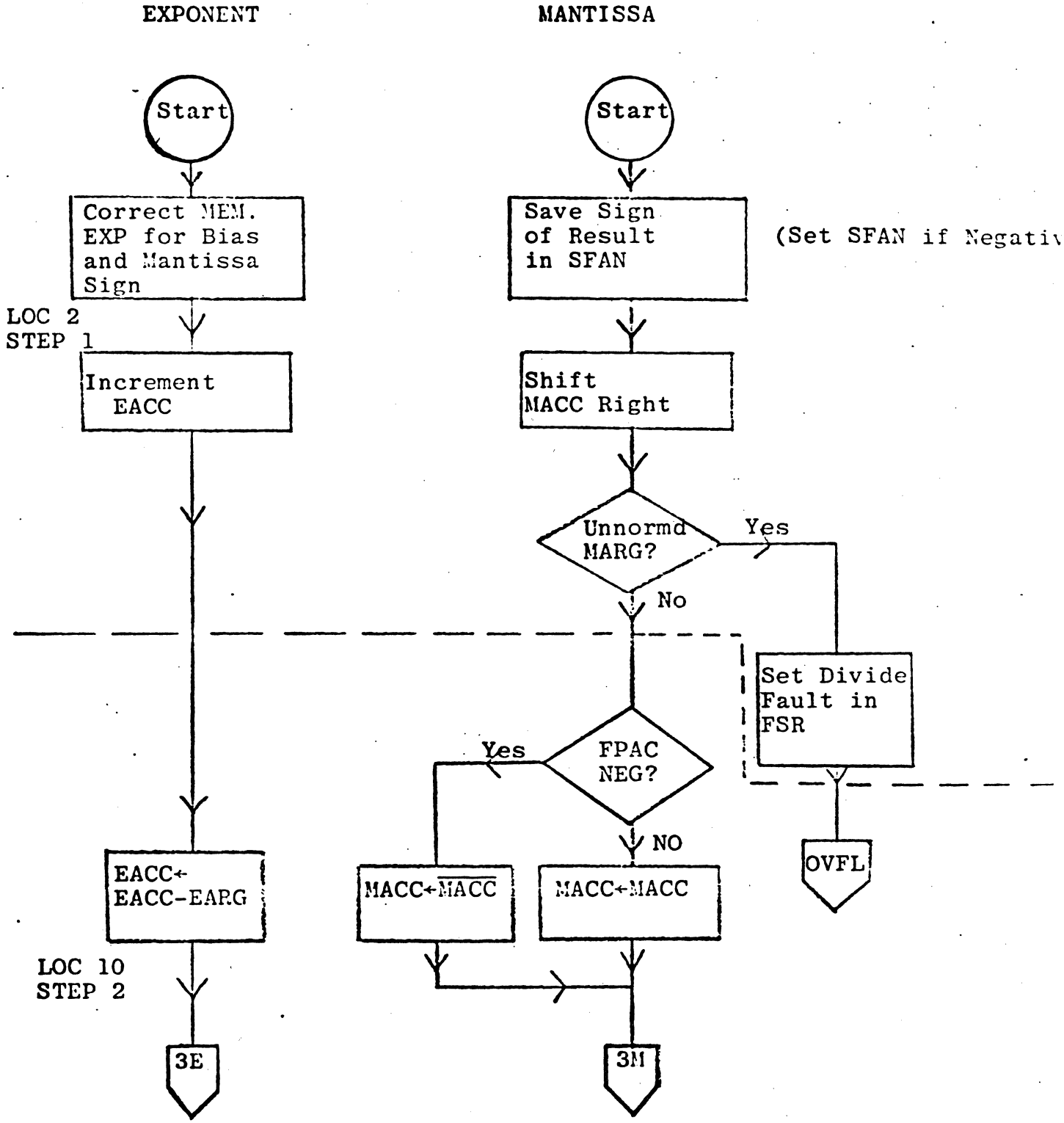
LOC 1E
STEP 4

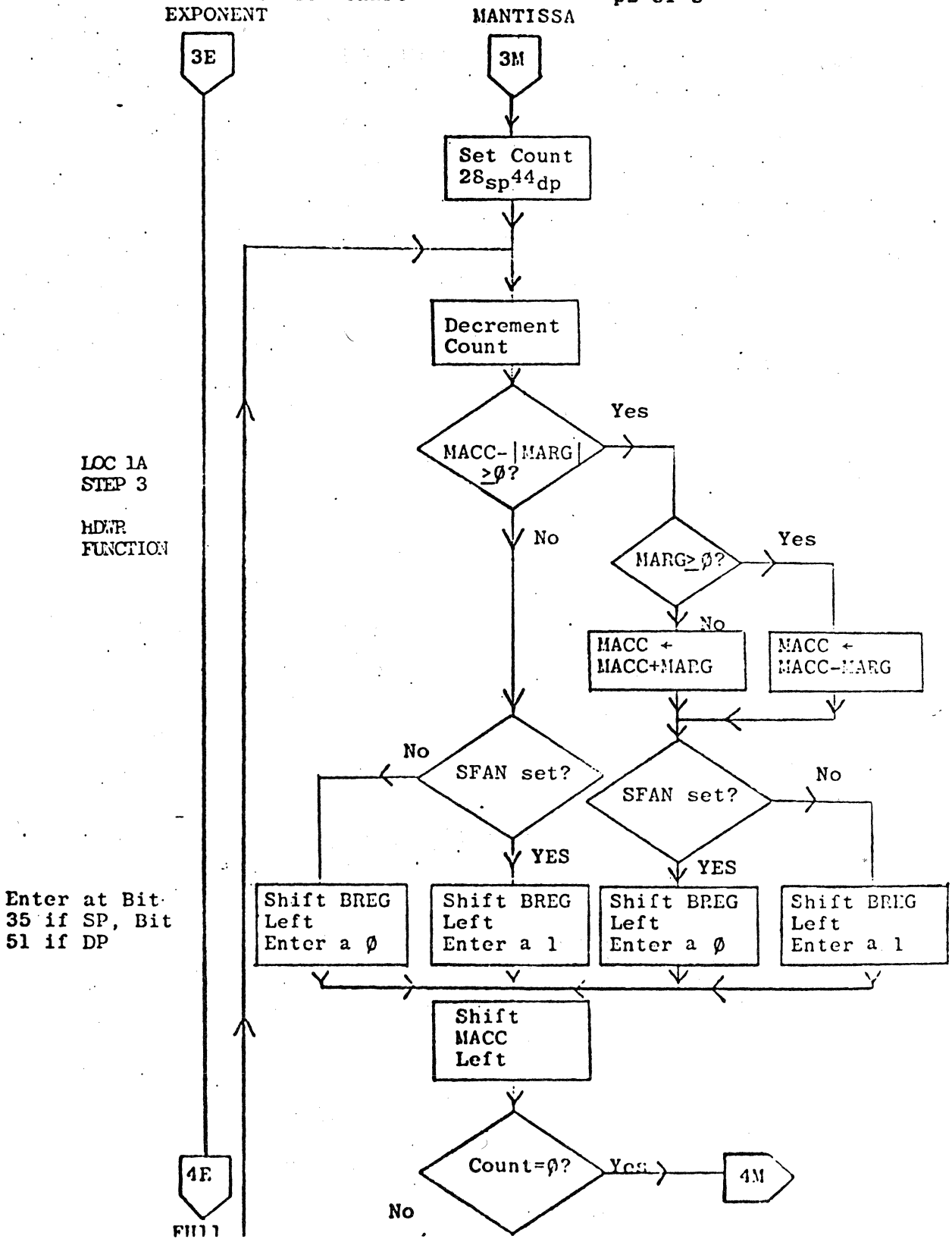

AE
EACC←
EACC+1
NORM

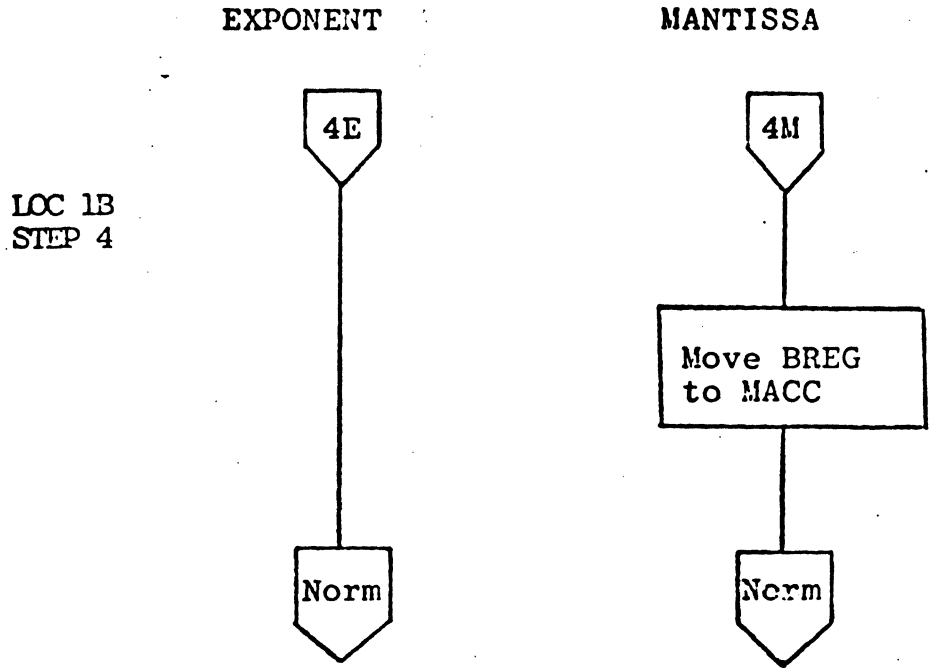

AM
FSUB?
Yes
No
MACC←
MACC+BREG
MACC←
MACC-BREG
NORM

FPH FMPY Flowcharts

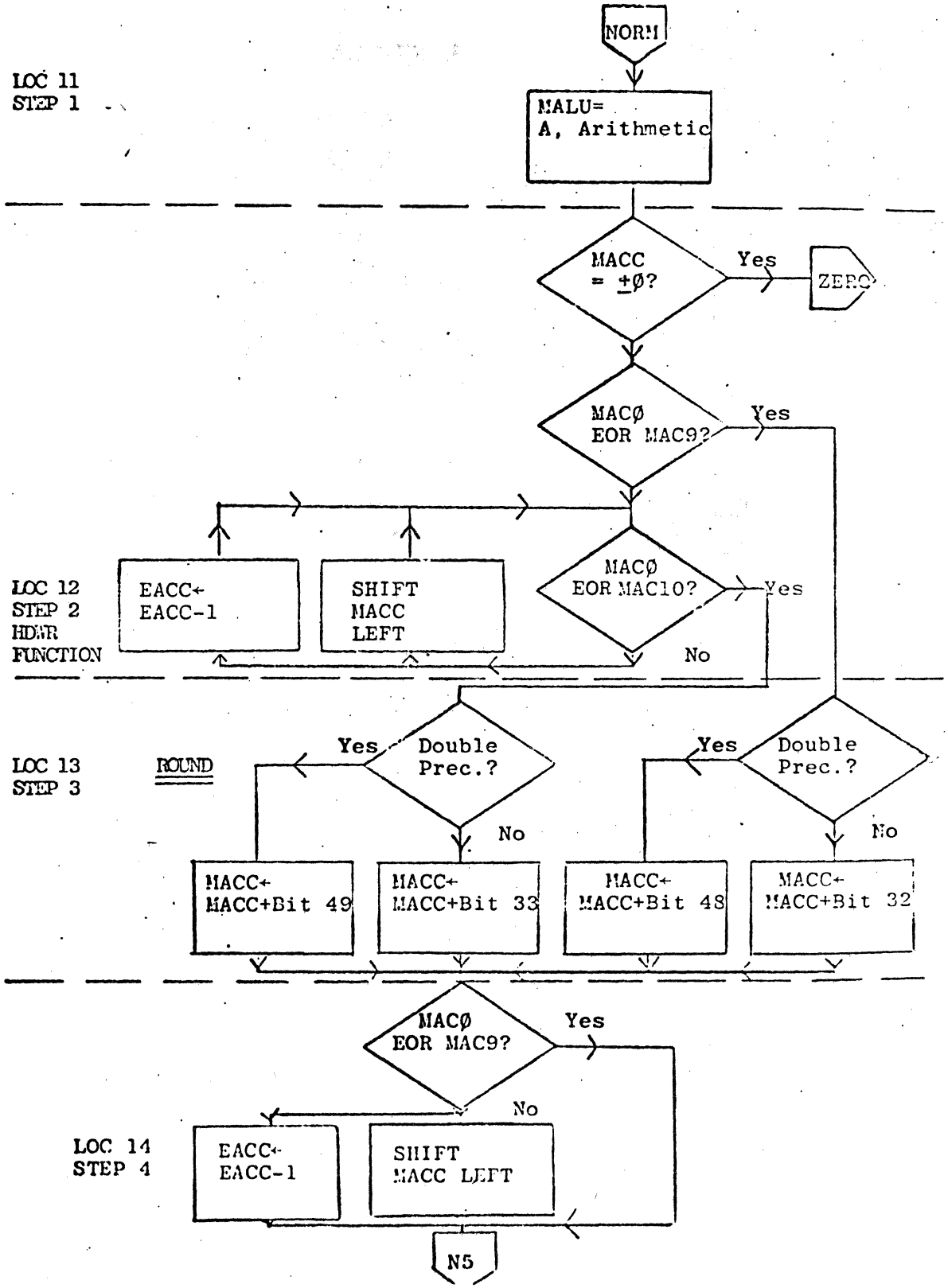



EXPONENT
MANTISSA
Start
Start
Correct MEM. EXP for Bias and Mantissa Sign
Save Sign of Result in SFAN
(Set SFAN if Negativ
LOC 2
STEP 1
Increment EACC
Shift MACC Right
Unnormd MARG?
Yes
No
Set Divide Fault in FSR
OVFL
FPAC NEG?
Yes
NO
EACC← EACC-EARG
MACC←MACC
MACC←MACC
LOC 10
STEP 2
3E
3M

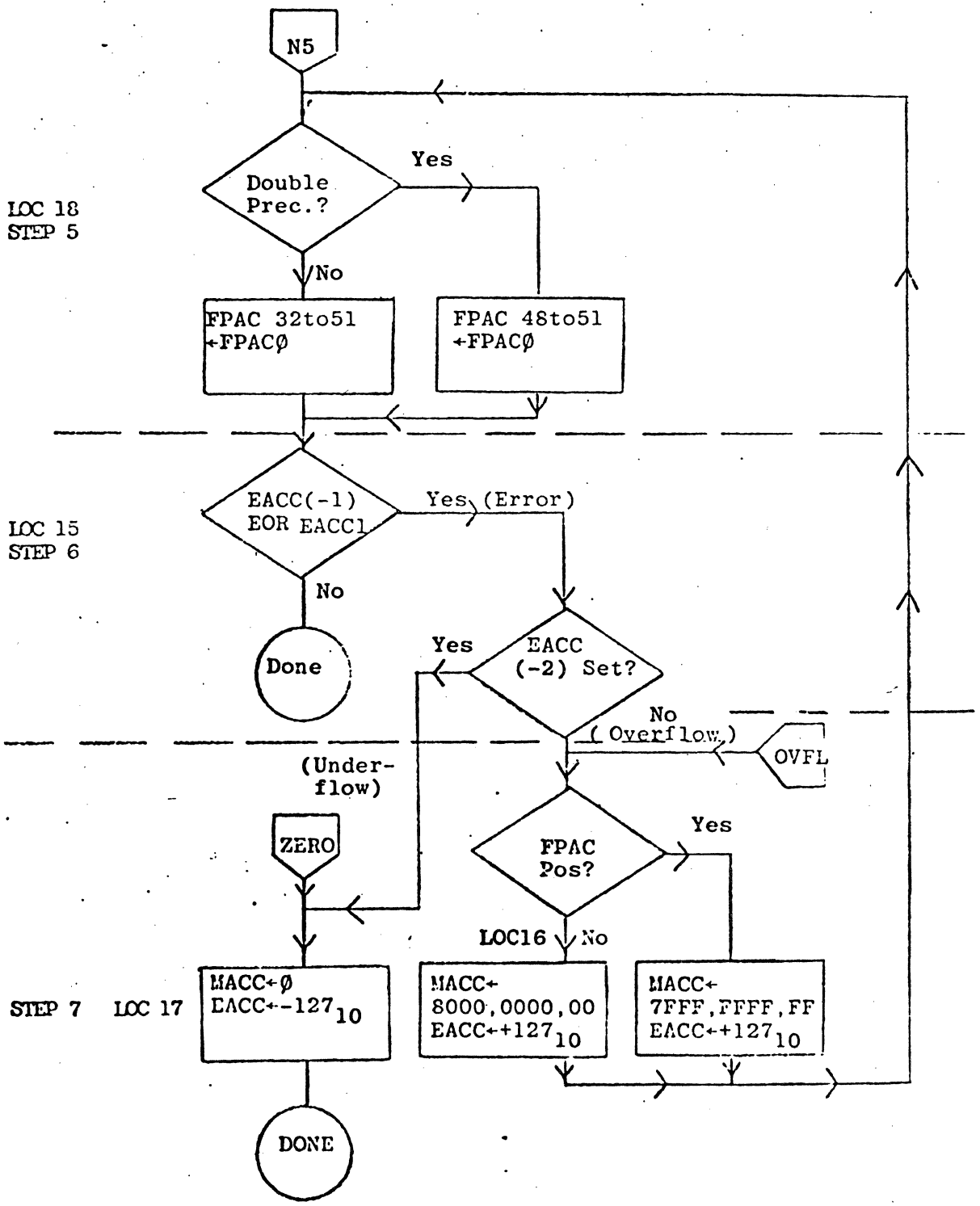

EXPONENT
MANTISSA
3E
3M
Set Count
28sp44dp
Decrement
Count
MACC-|MARG|
≥Ø?
Yes
No
MARG≥Ø?
Yes
No
MACC ←
MACC+MARG
MACC ←
MACC-MARG
SFAN set?
No
YES
SFAN set?
No
YES
Shift BREG
Left
Enter a Ø
Shift BREG
Left
Enter a 1
Shift BREG
Left
Enter a Ø
Shift BREG
Left
Enter a 1
Shift
MACC
Left
Count=Ø?
Yes
No
4M
4E
LOC 1A
STEP 3
HDWR
FUNCTION
Enter at Bit
35 if SP, Bit
51 if DP


EXPONENT
MANTISSA
LOC 1B
STEP 4
4E
4M
Move BREG
to MACC
Norm
Norm

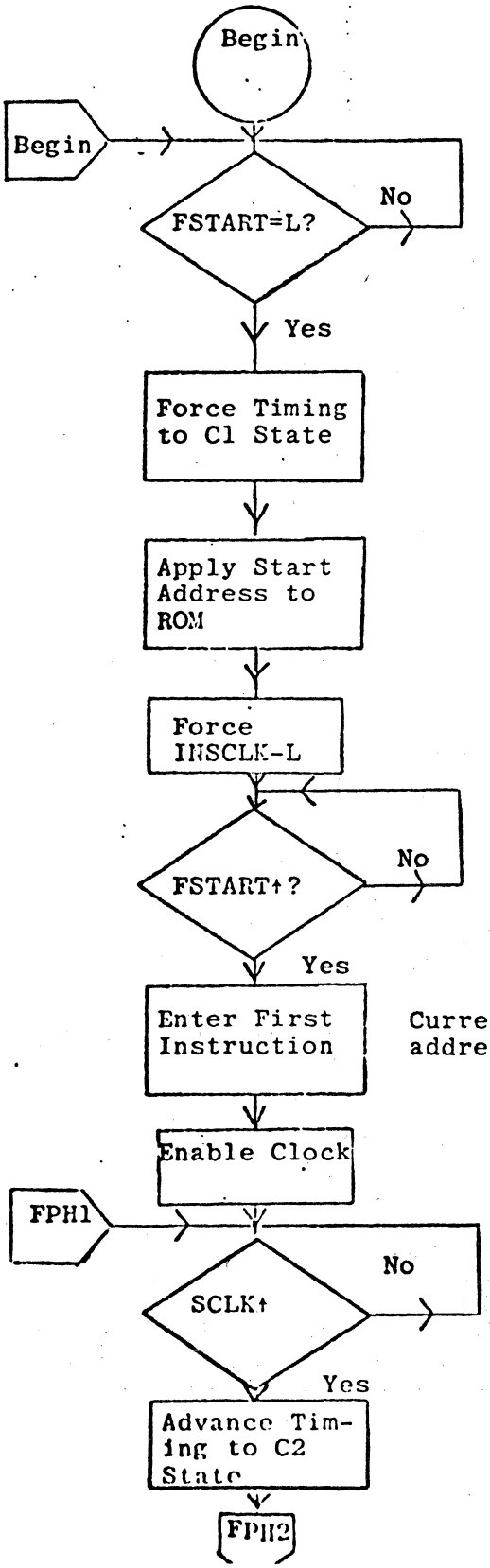
FPH Normalize Flow Chart



FPH · Normalize Flow Chart



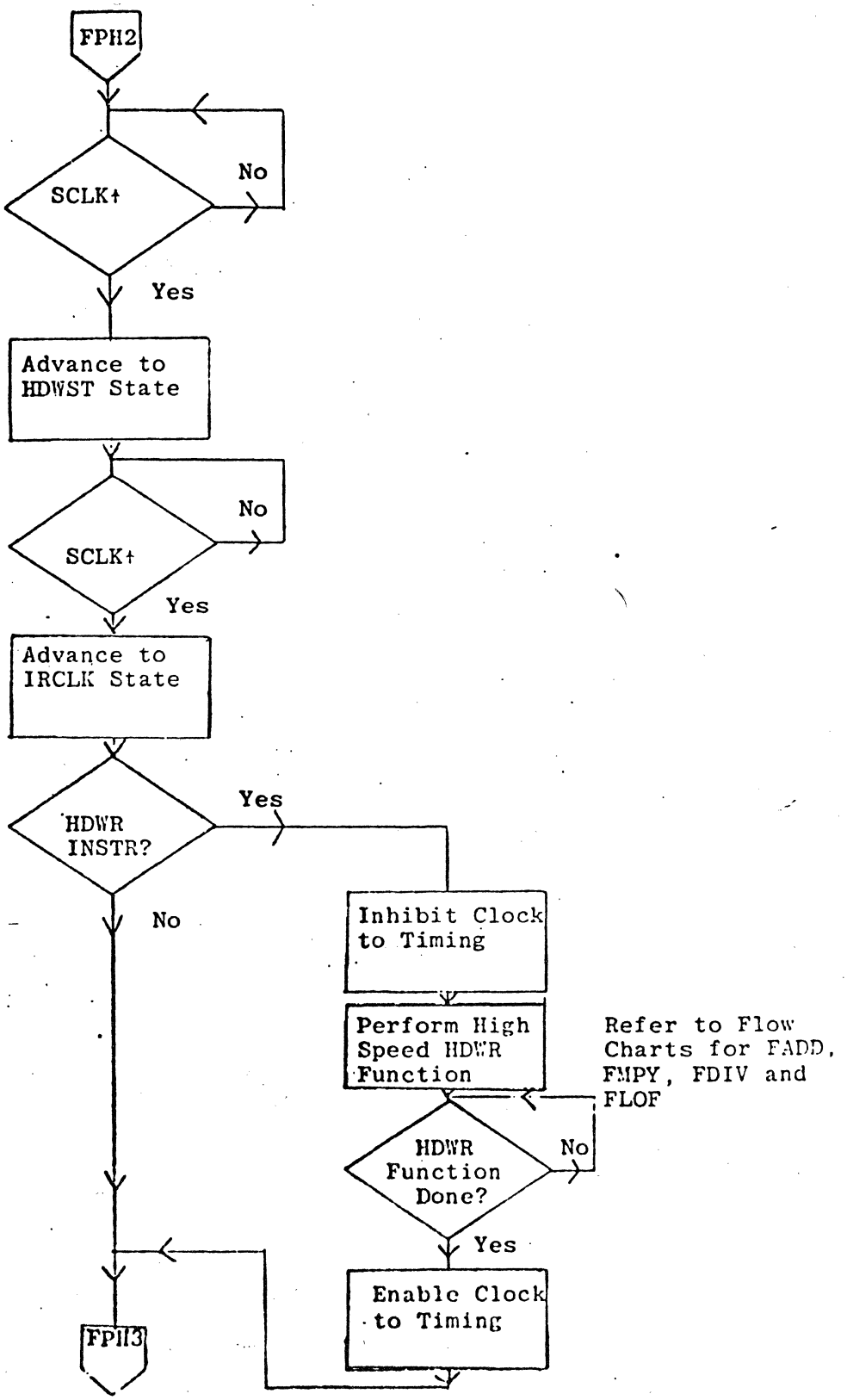
FPH Timing

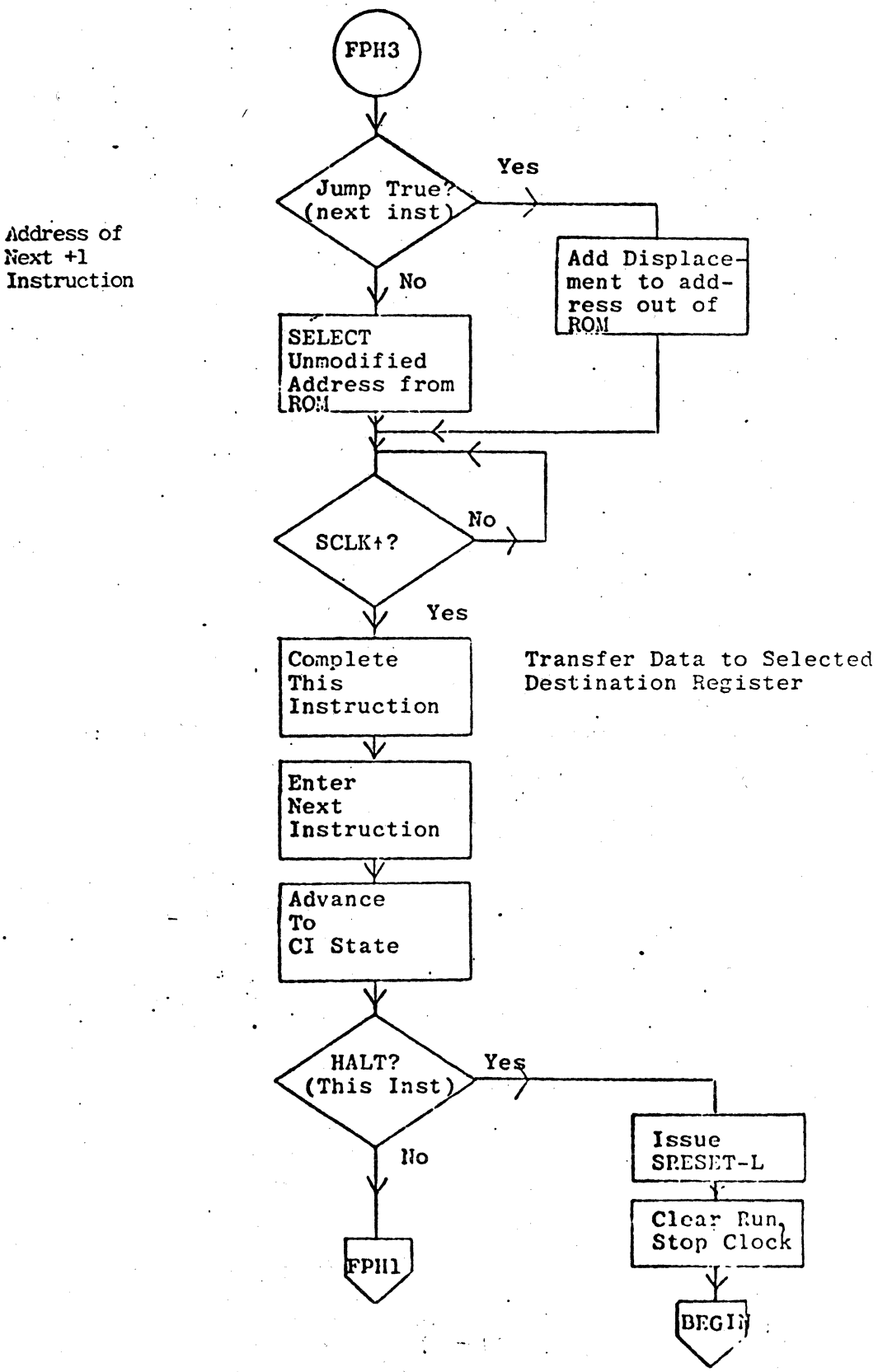


Function	Start Addr
FADD	0
FSUB	20_{16}
FMPY	1
FDIV	2
FLDD	3
LDWD1	4
LDWD2	5
LDWD3	6
FCOM	7
FIXF	8
FLOF	9

Current Instruction contains address of next instruction

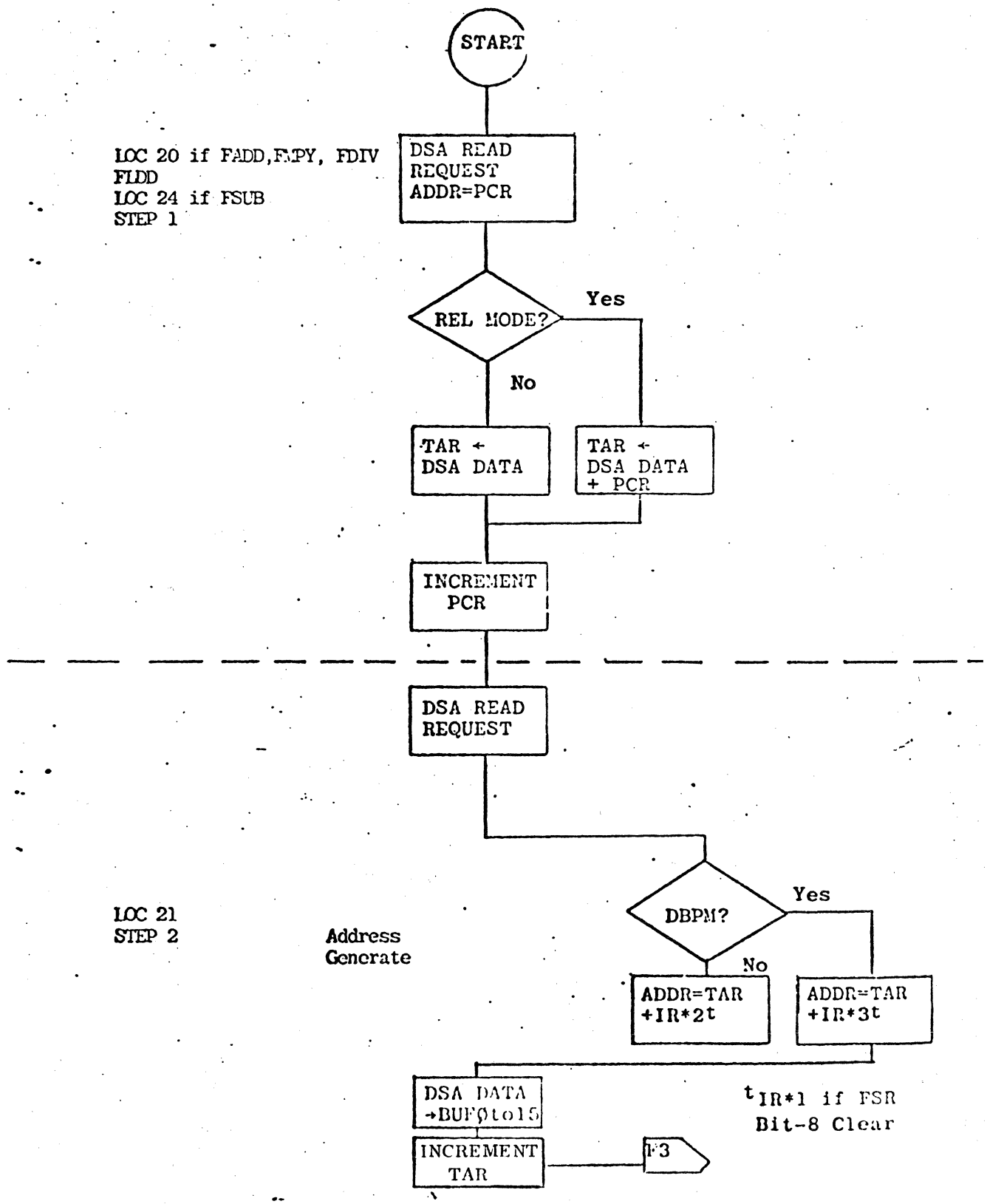
FPH Timing

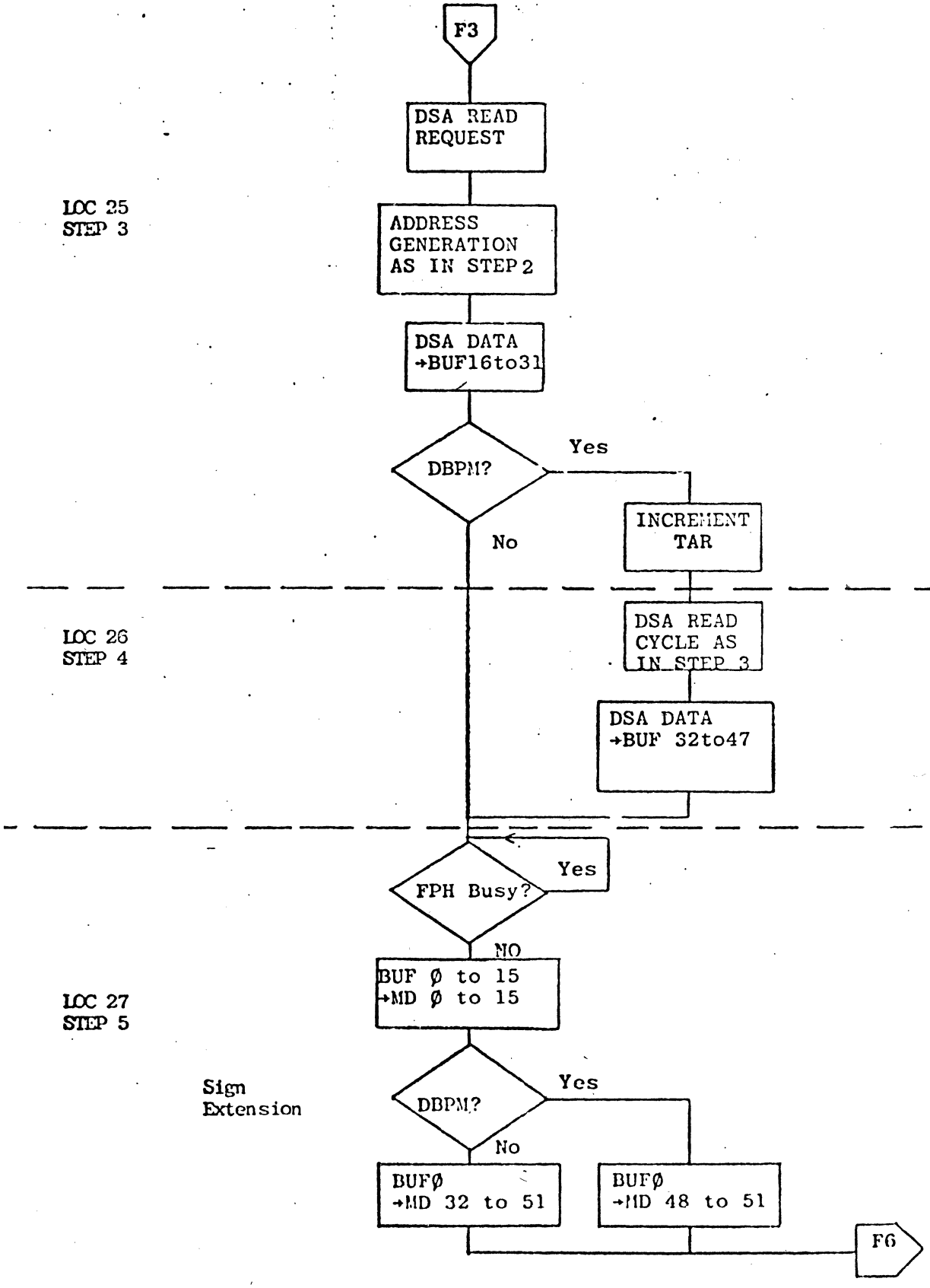

FPH2
SCLK↑
No
Yes
Advance to HDWST State
SCLK↑
No
Yes
Advance to IRCLK State
HDWR INSTR?
Yes
No
Inhibit Clock to Timing
Perform High Speed HDWR Function
Refer to Flow Charts for FADD, FMPY, FDIV and FLOF
HDWR Function Done?
No
Yes
Enable Clock to Timing
FPH3

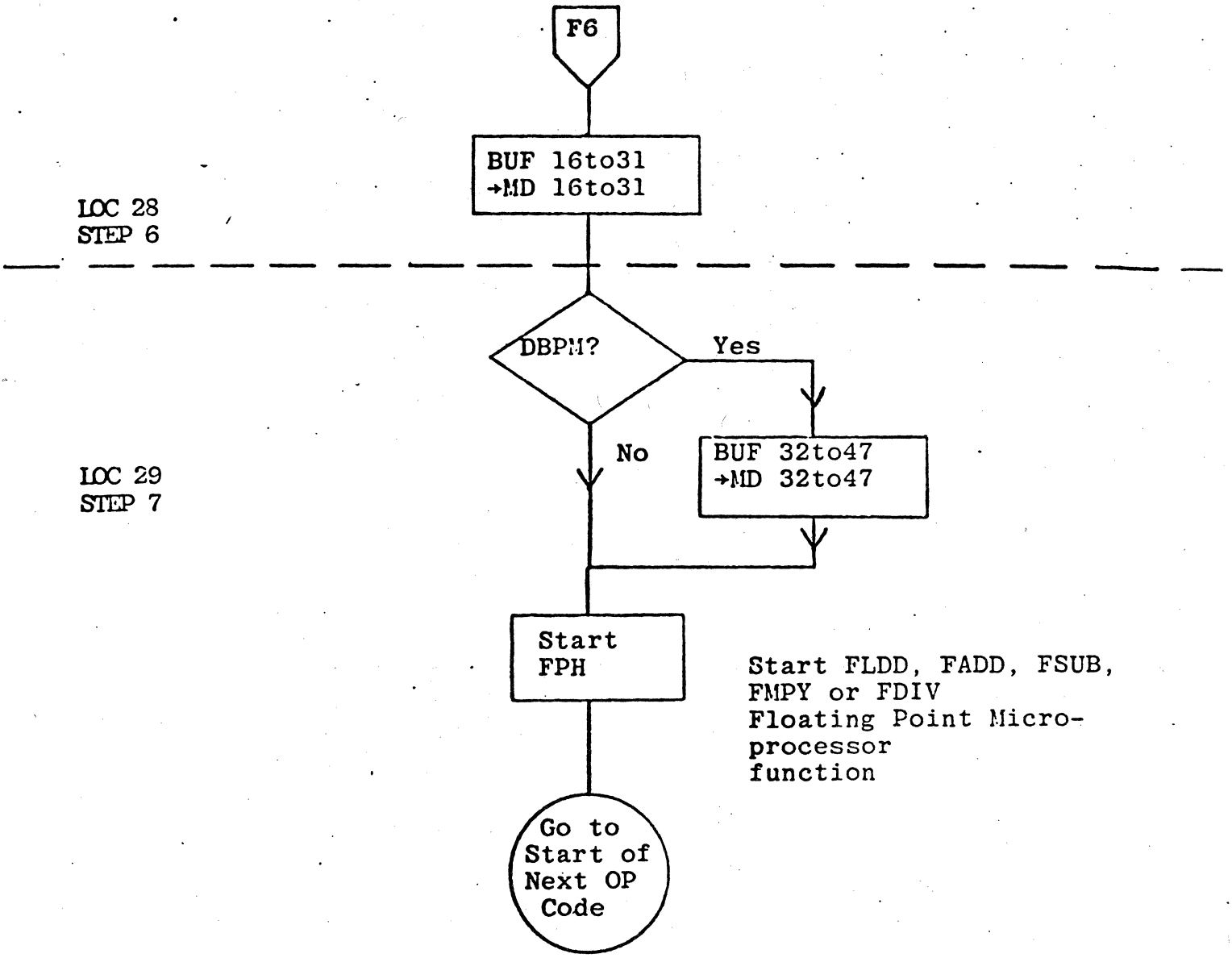

FPH3
Jump True? (next inst)
Yes
No
Address of Next +1 Instruction
Add Displacement to address out of ROM
SELECT Unmodified Address from ROM
SCLK↑?
No
Yes
Complete This Instruction
Transfer Data to Selected Destination Register
Enter Next Instruction
Advance To CI State
HALT? (This Inst)
Yes
No
Issue SRESET-L
Clear Run, Stop Clock
FPH1
BEGIN

LOC	LABEL	DSA	SCNR CLR	ADDR DENB	I/P	ADDR AENB	T/D	T LOAD	T CLK	P CLK	SA	INDX	BUFFER	GRP1	FSR	FPH	CTRL	WAIT	SP INH	EXEC	COND	NXT IN
0	RSTRT	-	-	-	-	1	0	TARL	1	-	1	-	RSSAR	-	SETA	-	-	-	-	-	-	4
1	LDWD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	FSTART	-	-	HALT	-	F
2	STOP	-	-	-	-	-	-	-	-	-	-	-	WFPAC1	FSRRD	-	-	-	1	-	-	-	F
3	FTCH NXT	RD,SCNHLT	-	1	0	-	-	-	-	1	-	-	-	CCR CLK	SETA	-	-	-	-	-	-	E
4	R2	RD,SHLT,CC	-	-	-	1	1	-	-	-	1	-	WFPAC1	-	SET DBPM	-	-	-	-	-	-	5
5	R3	RD,SHLT,CC	-	-	-	1	1	-	-	-	1	-	-	CCR CLK	-	-	-	-	-	-	-	6
6	R4	RD,SHLT	-	-	-	1	1	IRCLK	1	-	1	-	-	-	-	-	-	-	-	-	-	7
7	R5	RD,SHLT	-	-	-	1	1	PCRL	1	1	1	-	-	-	-	-	-	-	-	-	-	8
8	R6	RD,SHLT,CC	-	-	-	1	1	-	-	-	1	-	-	-	-	CLK 1,2,3	-	-	-	-	-	9
9	R7	RD,SHLT,CC	-	-	-	1	1	-	-	-	1	-	-	-	-	CLK2	-	-	-	-	-	A
A	R8	READ	-	-	-	1	1	-	-	-	1	-	-	-	-	CLK3	FSTART	-	-	-	-	B
B	R9	-	-	-	-	-	-	-	-	-	-	-	RFPAC1	FSR CLK	SET A	-	-	-	-	-	-	C
C	R10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TRUE INH	-	-	EXEC NXT	IN ACTV	C
D	R11	-	1	-	-	-	-	-	-	-	-	-	-	-	CLR A	-	-	-	-	HALT	-	E
E	F2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EXEC NXT	-	-
F	S2	-	-	-	-	1	0	TARL	1	-	1	-	RSSAR	-	SET A&P	-	-	-	-	-	-	10
10	S3	WR,SHLT,CC	-	-	-	1	1	-	-	-	1	-	RFPAC1	-	-	-	-	-	-	-	-	11
11	S4	WR,SHLT,CC	-	-	-	1	1	-	-	-	1	-	-	CCR RD	-	-	-	-	-	-	-	12
12	S5	WR,SHLT,CC	-	1	1	1	1	-	-	-	1	-	-	ADATA	-	-	-	-	-	-	-	13
13	S6	WR,SHLT,CC	-	1	0	1	1	-	-	-	1	-	-	ADATA	-	-	-	-	-	-	-	14
14	S7	WR,SHLT,CC	-	-	-	1	1	-	-	-	1	-	-	-	-	DOUT1	-	-	-	-	-	15
15	S8	WR,SHLT,CC	-	-	-	1	1	-	-	-	1	-	-	-	-	DOUT2	-	-	-	-	-	16
16	S9	WRITE	-	-	-	1	1	-	-	-	1	-	-	-	-	DOUT3	-	-	-	-	-	D
17	STRI	RD,SHLT,REL	-	1	0	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	18
18	STI2	WRITE	-	1	1	1	1	-	-	-	1	-	-	ADATA	-	-	-	-	-	EXEC NXT	-	-
19	FCOM	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	FSTART	1	-	EXEC NXT	-	-
1A	FIXF	RD,SHLT,REL	-	1	0	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	1B
1B	FXF2	READ	-	1	1	1	1	-	-	-	-	-	WRPAC2	-	-	-	-	-	-	-	-	1C
1C	FXF3	-	-	-	-	-	-	-	-	-	-	-	RFPAC2	-	-	CLK 1,2,3	FSTART	1	-	EXEC NXT	-	-
1D	FEND	-	1	-	-	-	-	-	-	-	-	-	-	-	SETF CLRA	-	-	1	-	HALT	-	E
1E	CHMD	-	1	-	-	-	-	-	-	-	-	-	-	CHMD	-	-	-	-	-	EXEC NXT	-	-
1F	NIDX	-	1	-	-	-	-	-	-	-	-	-	-	IRCLR	-	-	-	-	-	EXEC NXT	-	-
20	AMDF	RD,SHLT,REL	-	1	0	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	21
21	A2	RD,SHLT,CC	-	1	1	1	1	-	-	-	-	1	WFPAC1	-	-	-	-	-	-	-	-	38
22	FLOF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	FSTART	1	-	-	-	38
23	FLF3	WRITE	-	1	1	1	1	-	-	-	-	-	-	-	-	DOUT2	-	1	-	EXEC	-	-
24	FSUB	RD,SHLT,REL	-	1	0	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	21
25	A3	RD,SHLT,CC	-	1	1	1	1	-	-	-	-	1	WFPAC2	-	-	-	-	-	1	-	SP	26
26	A4	READ	-	1	1	1	1	-	-	-	-	1	WFPAC2	-	-	-	-	-	-	-	-	27
27	A5	-	-	-	-	-	-	-	-	-	-	-	RFPAC1	-	-	CLK 1,2,3	-	1	-	-	-	28
28	A6	-	-	-	-	-	-	-	-	-	-	-	RFPAC2	-	-	CLK2	-	-	-	-	-	29
29	A7	-	-	-	-	-	-	-	-	-	-	-	RFPAC3	-	-	CLK3	FSTART	-	1	EXEC NXT	-	-
2A	ADDI	RD,SHLT,REL	-	1	-	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	2B
2B	ADI2	READ	-	-	1	1	1	IRCLK	-	-	-	-	-	-	-	-	-	-	-	EXEC NXT	-	-
2C	INDX	RE,SHLT,REL	-	1	-	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	2D
2D	INX2	READ	-	-	-	1	1	IRCLK	-	-	1	-	-	-	-	-	-	-	-	EXEC NXT	-	-
2E	FLST	RD,REL	-	1	-	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	2F
2F	FST2	WR,SHLT,CC	-	1	1	1	1	-	-	-	-	1	-	-	-	DOUT1	-	1	-	-	-	30
30	FST3	WR,SHLT,CC	-	1	1	1	1	-	-	-	-	1	-	-	-	DOUT2	-	-	1	EXEC NXT if SP	-	31
31	FST4	WRITE	-	1	1	1	1	-	-	-	-	1	-	-	-	DOUT3	-	-	-	EXEC NXT	-	-
32	SPEC	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	SPEC	-	-	EXEC NXT	-	-
33	BRI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	34
34	BRI2	-	1	-	-	-	-	-	-	1	-	-	-	-	-	-	TRUE INH	-	-	EXEC NXT	COND ENB	34
35	CACS	RD,SHLT,REL	-	1	-	-	-	PCRL	-	1	-	-	-	-	-	-	-	-	-	-	-	3
36	BRA	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	37
37	BRA2	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	TRUE INH	-	-	EXEC NXT	COND ENB	34
38	FLF2	RD,REL	-	1	-	-	-	TARL	1	1	-	-	-	-	-	-	-	-	-	-	-	23

FLDD, FADD, FSUB, FMPY, FDIV

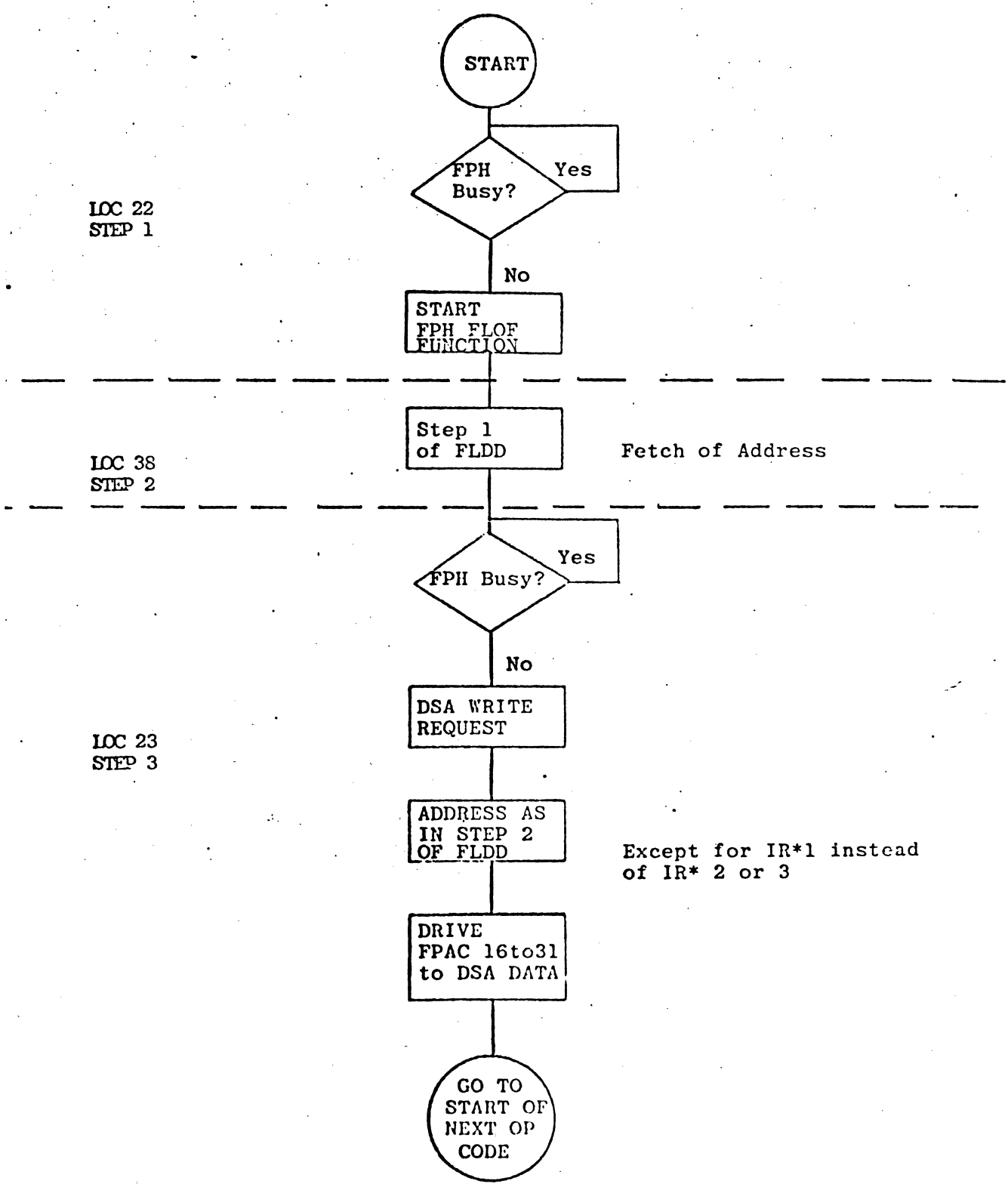



F3
DSA READ REQUEST
LOC 25 STEP 3
ADDRESS GENERATION AS IN STEP 2
DSA DATA →BUF16to31
DBPM?
Yes
No
INCREMENT TAR
LOC 26 STEP 4
DSA READ CYCLE AS IN STEP 3
DSA DATA →BUF 32to47
FPH Busy?
Yes
NO
LOC 27 STEP 5
BUF Ø to 15 →MD Ø to 15
Sign Extension
DBPM?
Yes
No
BUFØ →MD 32 to 51
BUFØ →MD 48 to 51
F6

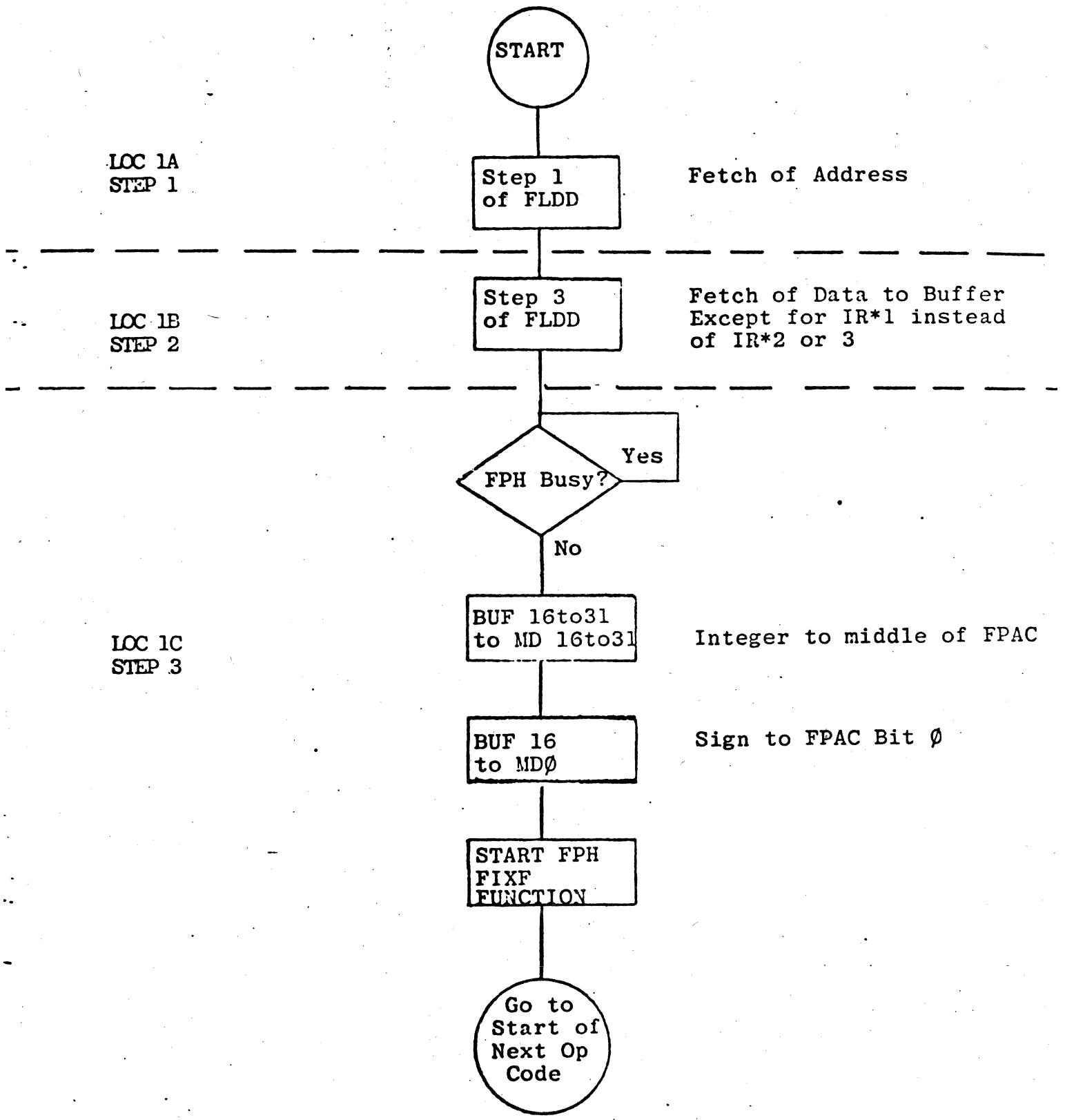

F6
BUF 16to31
→MD 16to31
LOC 28
STEP 6
DBPM?
Yes
No
BUF 32to47
→MD 32to47
LOC 29
STEP 7
Start
FPH
Start FLDD, FADD, FSUB,
FMPY or FDIV
Floating Point Micro-
processor
function
Go to
Start of
Next OP
Code

Master Control Flow Charts

FLOF

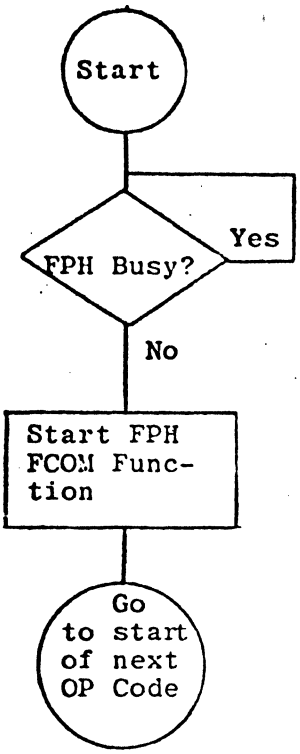


FIXF


START
LOC 1A
STEP 1
Step 1
of FLDD
Fetch of Address
LOC 1B
STEP 2
Step 3
of FLDD
Fetch of Data to Buffer
Except for IR*1 instead
of IR*2 or 3
FPH Busy?
Yes
No
LOC 1C
STEP 3
BUF 16to31
to MD 16to31
Integer to middle of FPAC
BUF 16
to MDØ
Sign to FPAC Bit Ø
START FPH
FIXF
FUNCTION
Go to
Start of
Next Op
Code

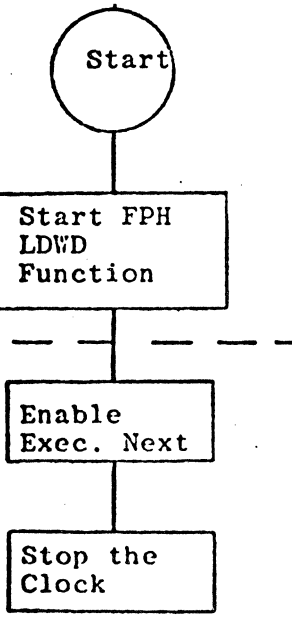
Master Control Flowcharts

FCOM



LOC 19

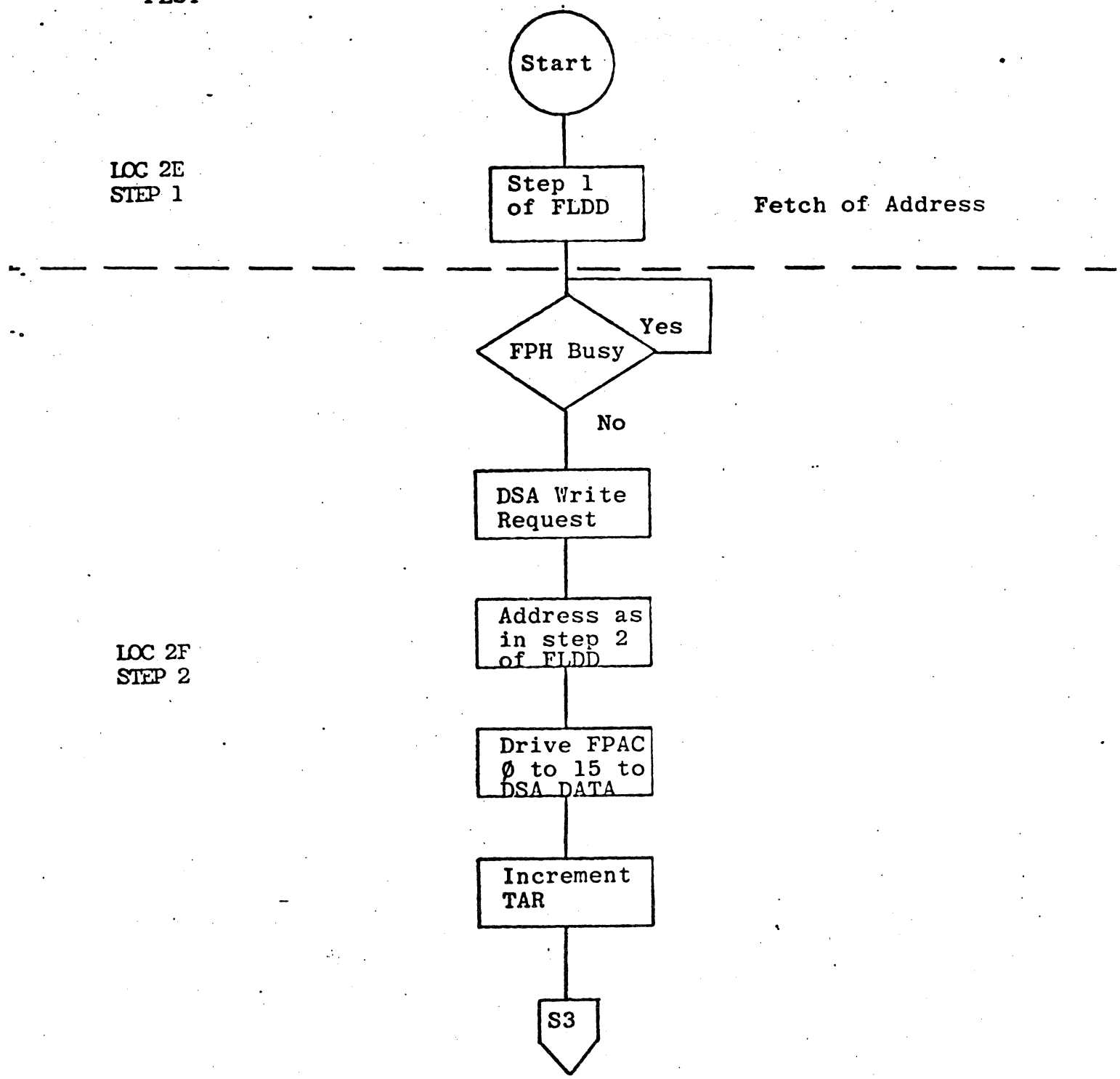
LDWD1, LDWD2, LDWD3



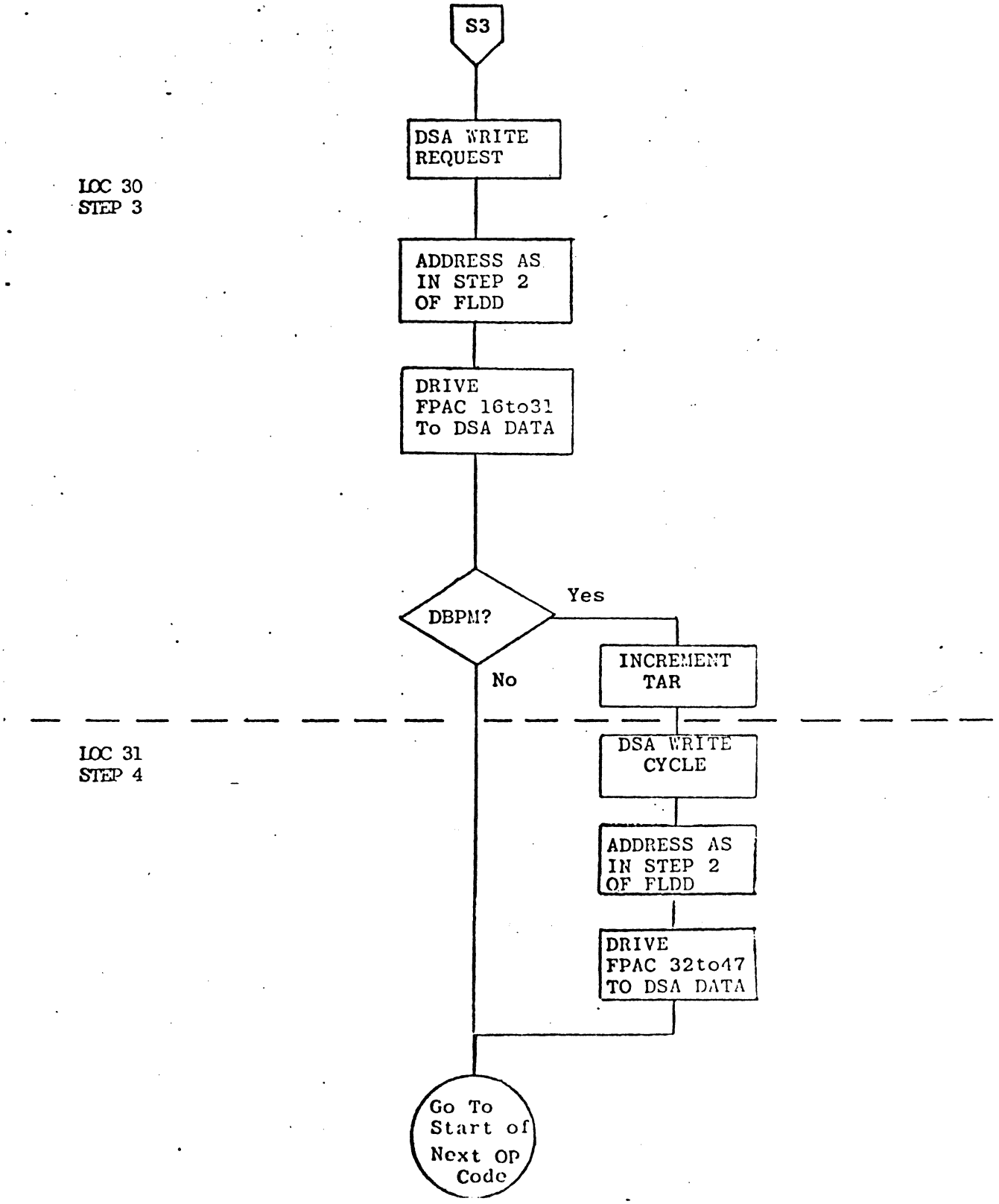
LOC 1
STEP 1

LOC E
STEP 2

FLST

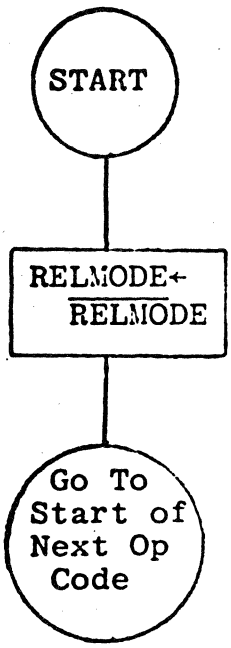

Start
LOC 2E
STEP 1
Step 1
of FLDD
Fetch of Address
FPH Busy
Yes
No
DSA Write
Request
Address as
in step 2
of FLDD
LOC 2F
STEP 2
Drive FPAC
Ø to 15 to
DSA DATA
Increment
TAR
S3

FLST Continued



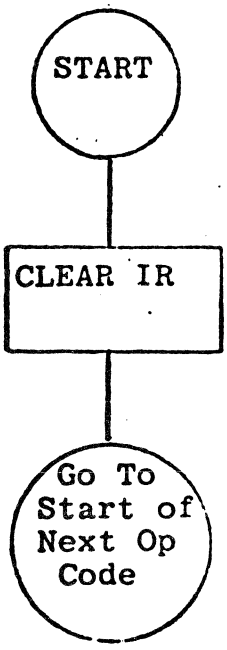
CHMD

LOC 1E

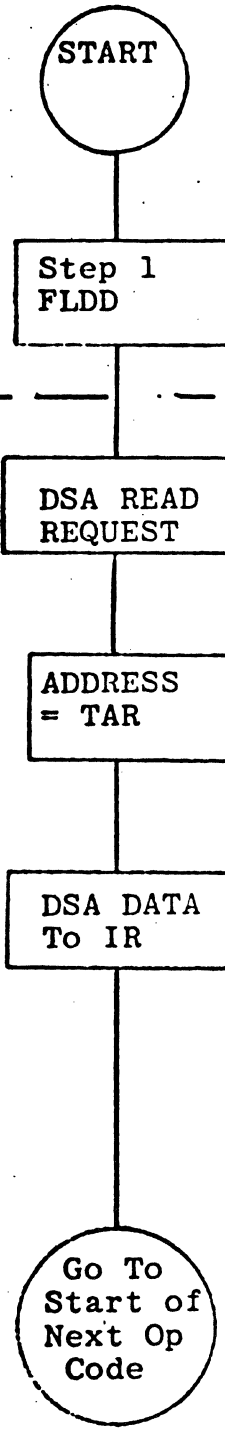


NIDX

LOC 1F



INDX

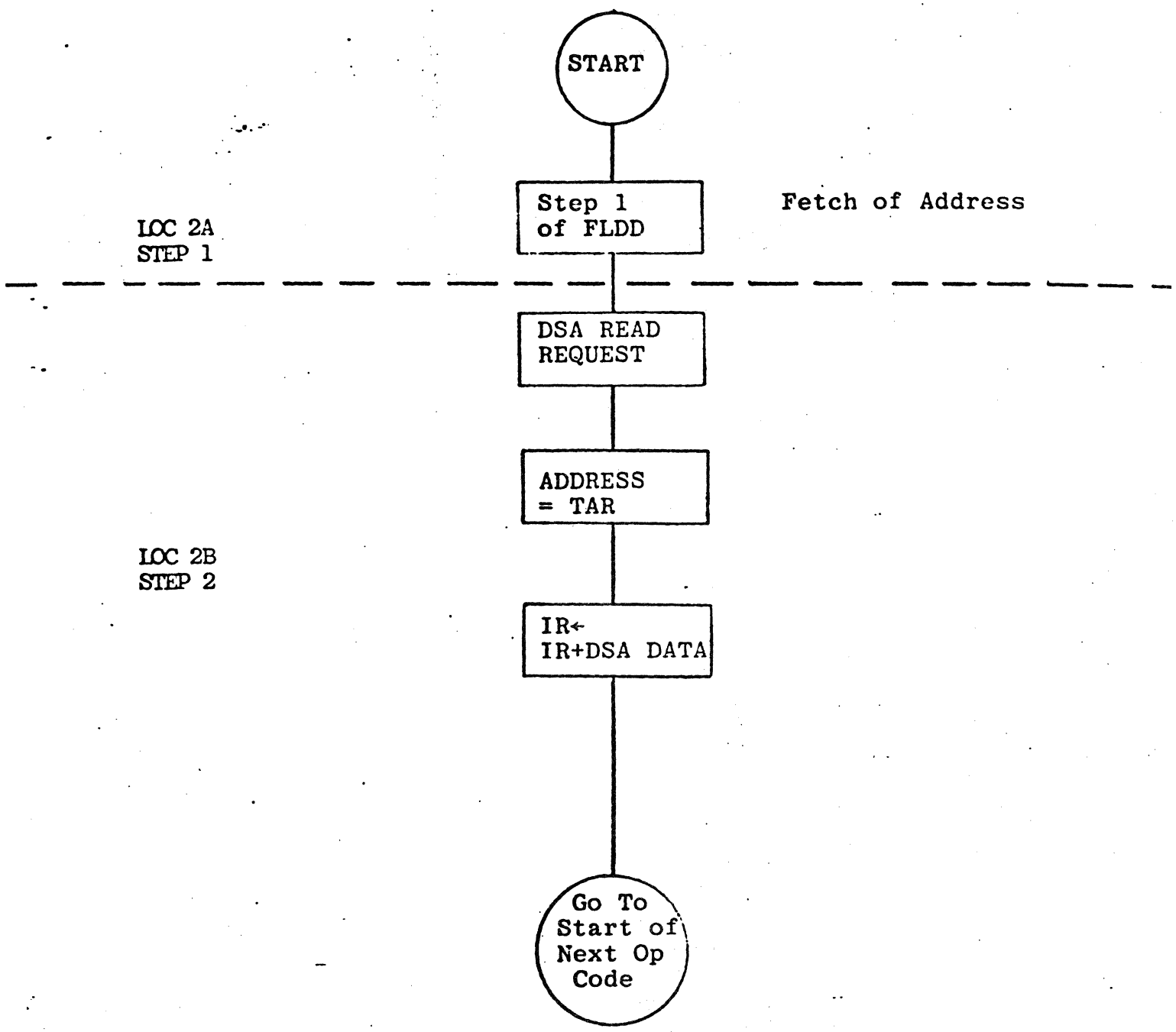


Fetch of Address

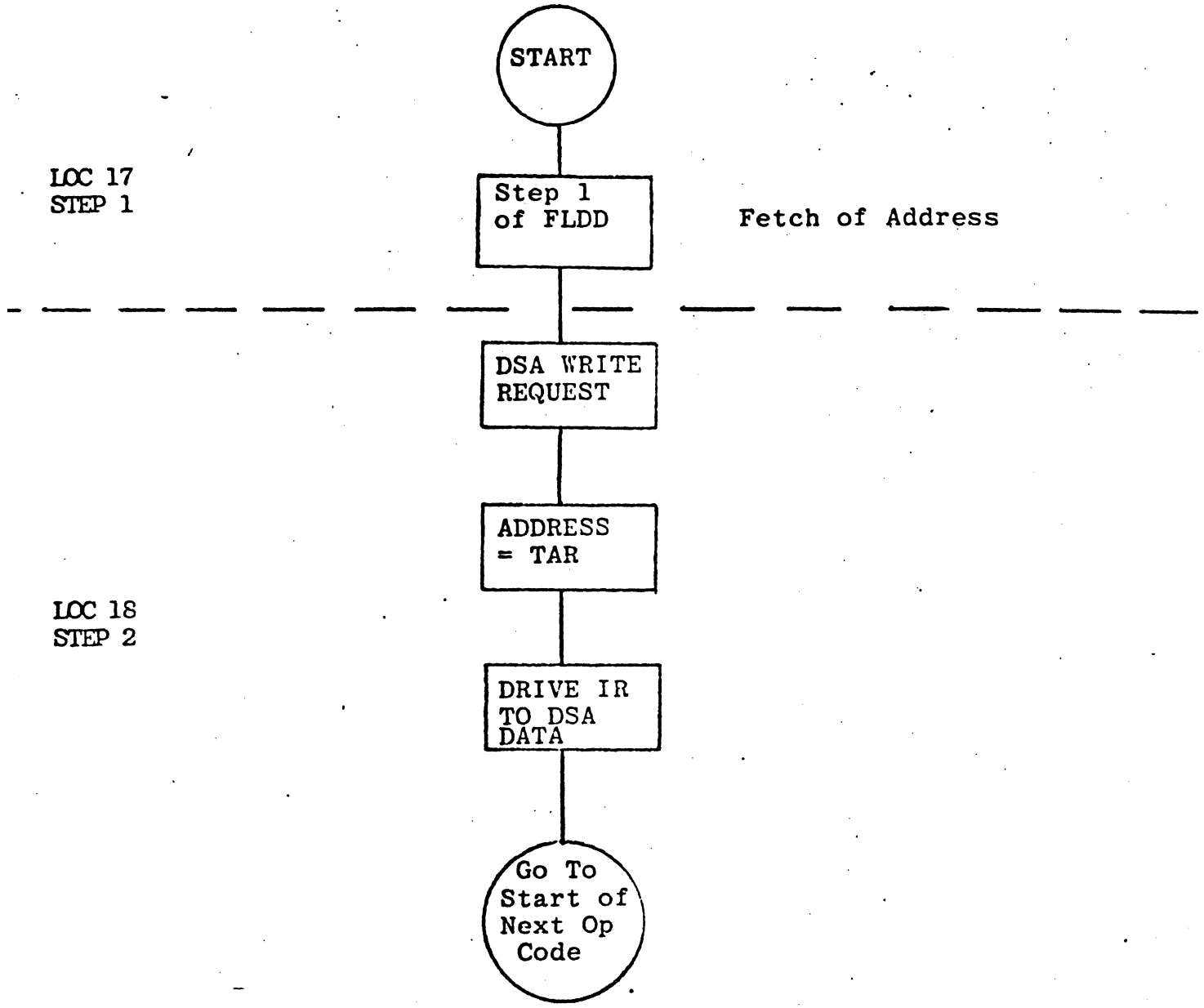
LOC 2C
STEP 1

LOC 2D
STEP 2

ADDI


START
Step 1
of FLDD
Fetch of Address
LOC 2A
STEP 1
DSA READ
REQUEST
ADDRESS
= TAR
LOC 2B
STEP 2
IR←
IR+DSA DATA
Go To
Start of
Next Op
Code

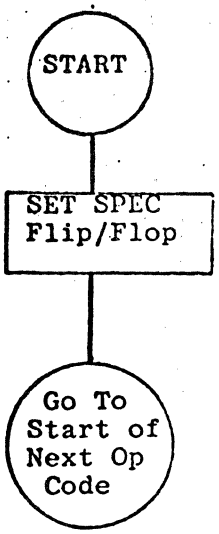
STRI


START
LOC 17
STEP 1
Step 1
of FLDD
Fetch of Address
DSA WRITE
REQUEST
ADDRESS
= TAR
LOC 18
STEP 2
DRIVE IR
TO DSA
DATA
Go To
Start of
Next Op
Code

M. C. Flow Charts

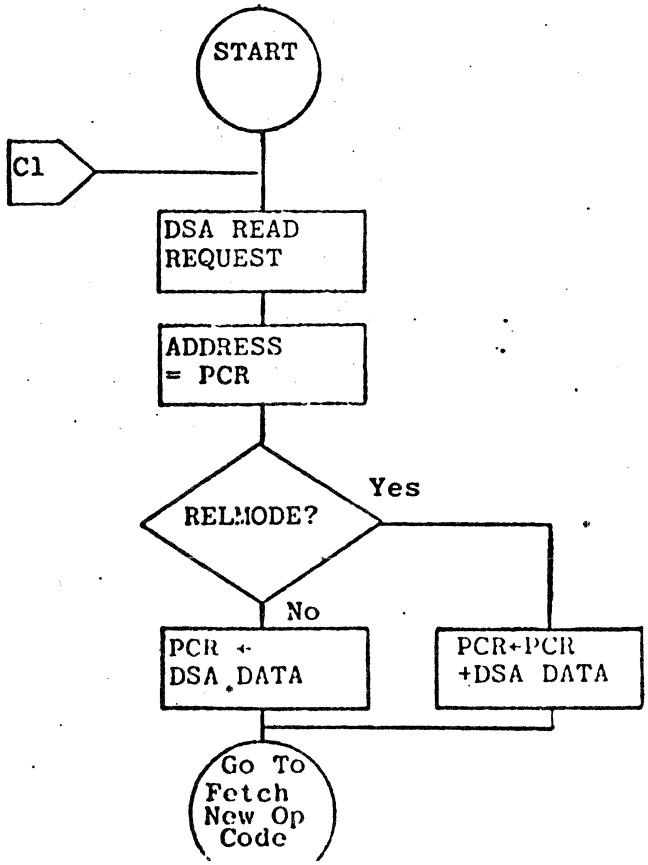
SPEC

LOC 32



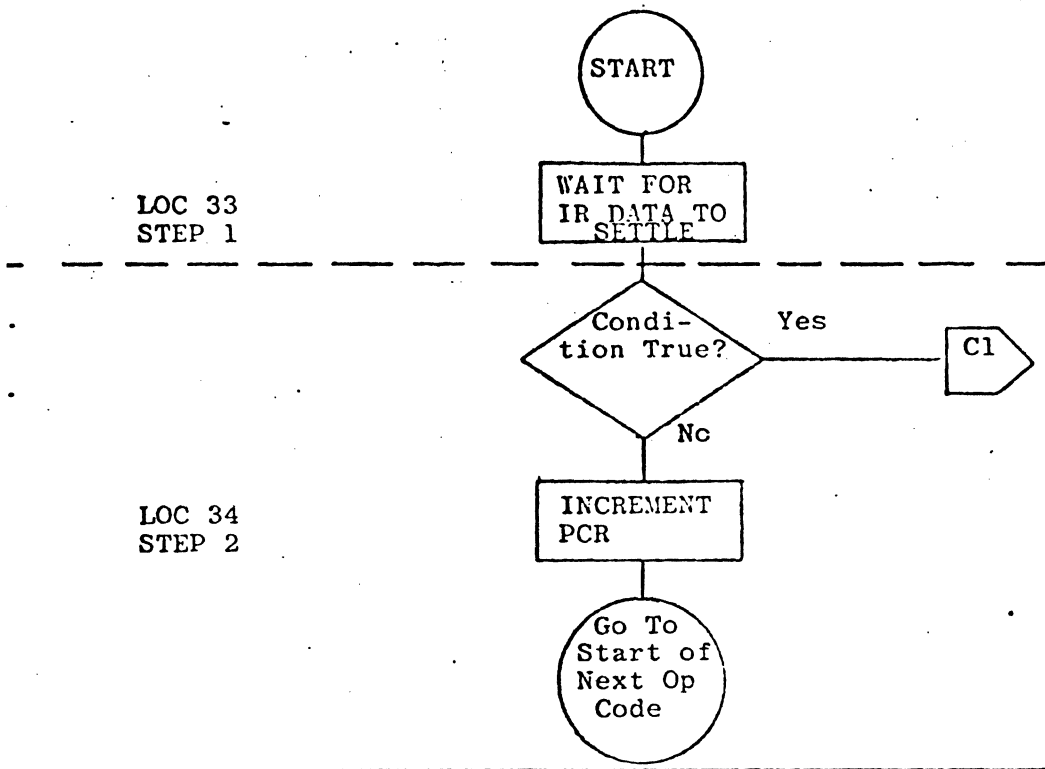
CACS

LOC 35

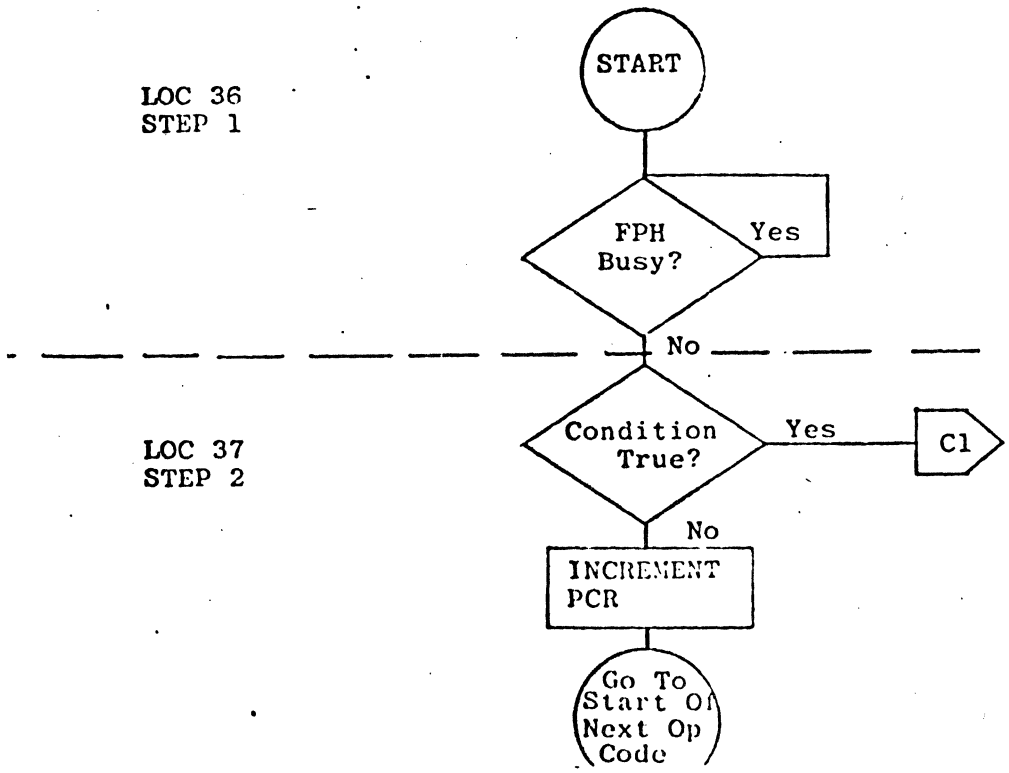


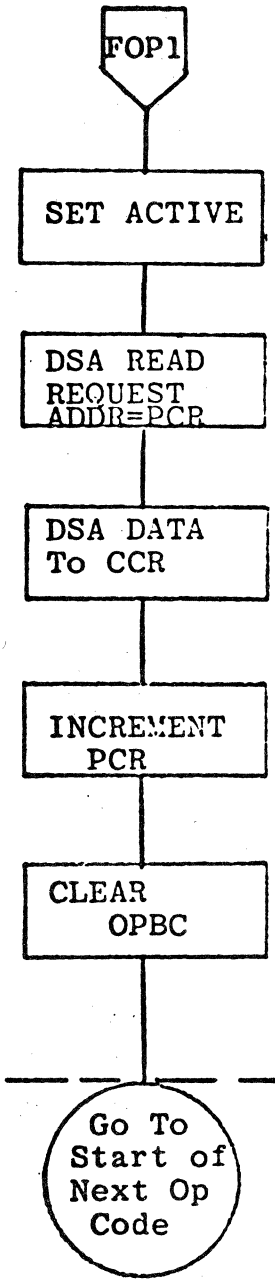
M. C. Flow Charts

BRIP, BRIZ, BRIM, BRIN



BRAP, BRAZ, BRAM, BRAN

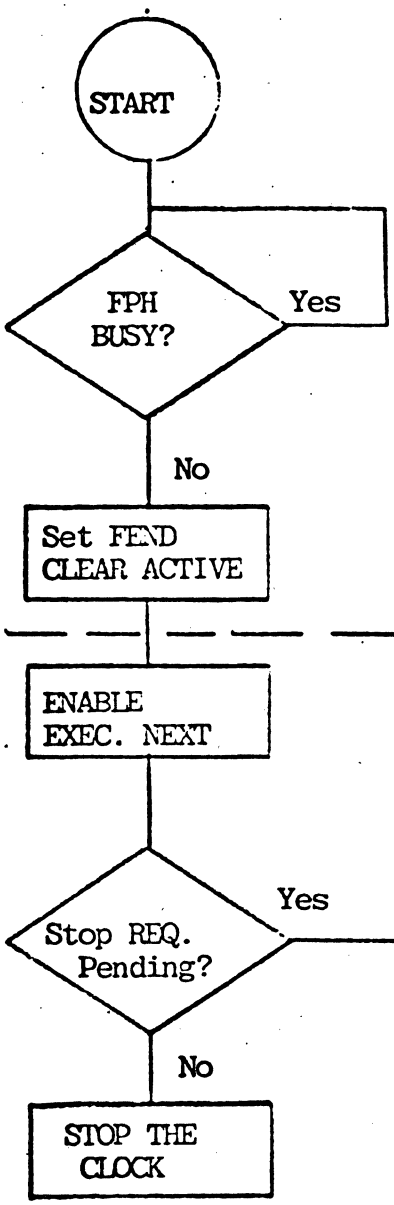




LOC 3
STEP 1

LOC E
STEP 2

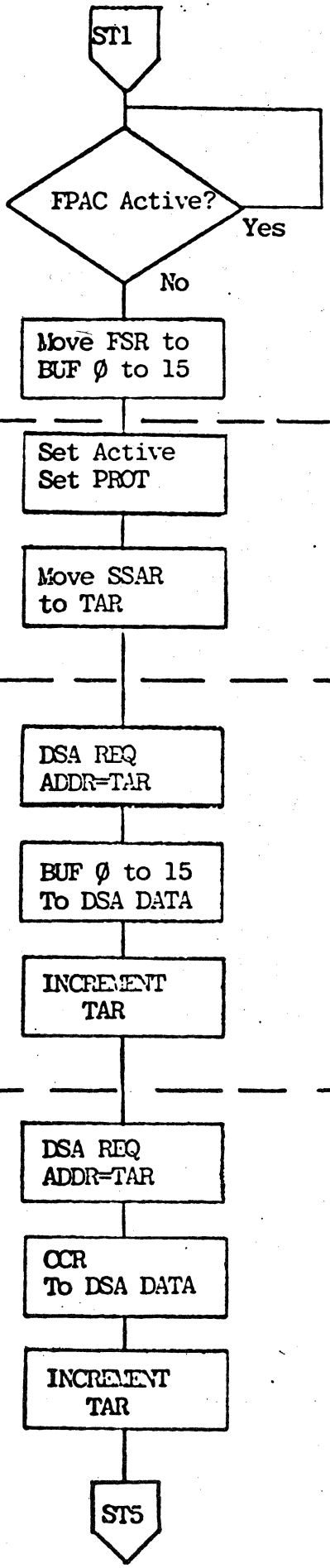
FEND

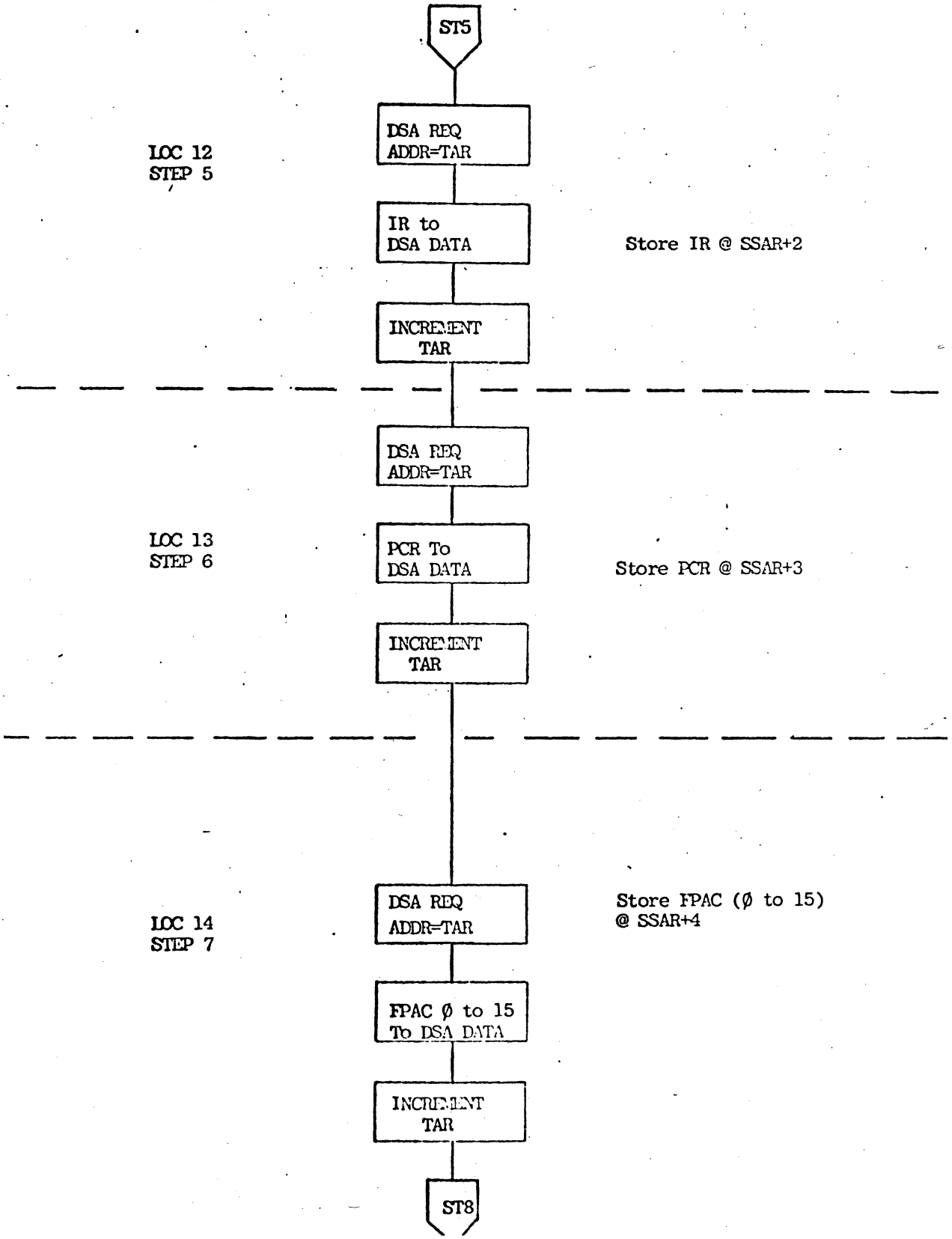


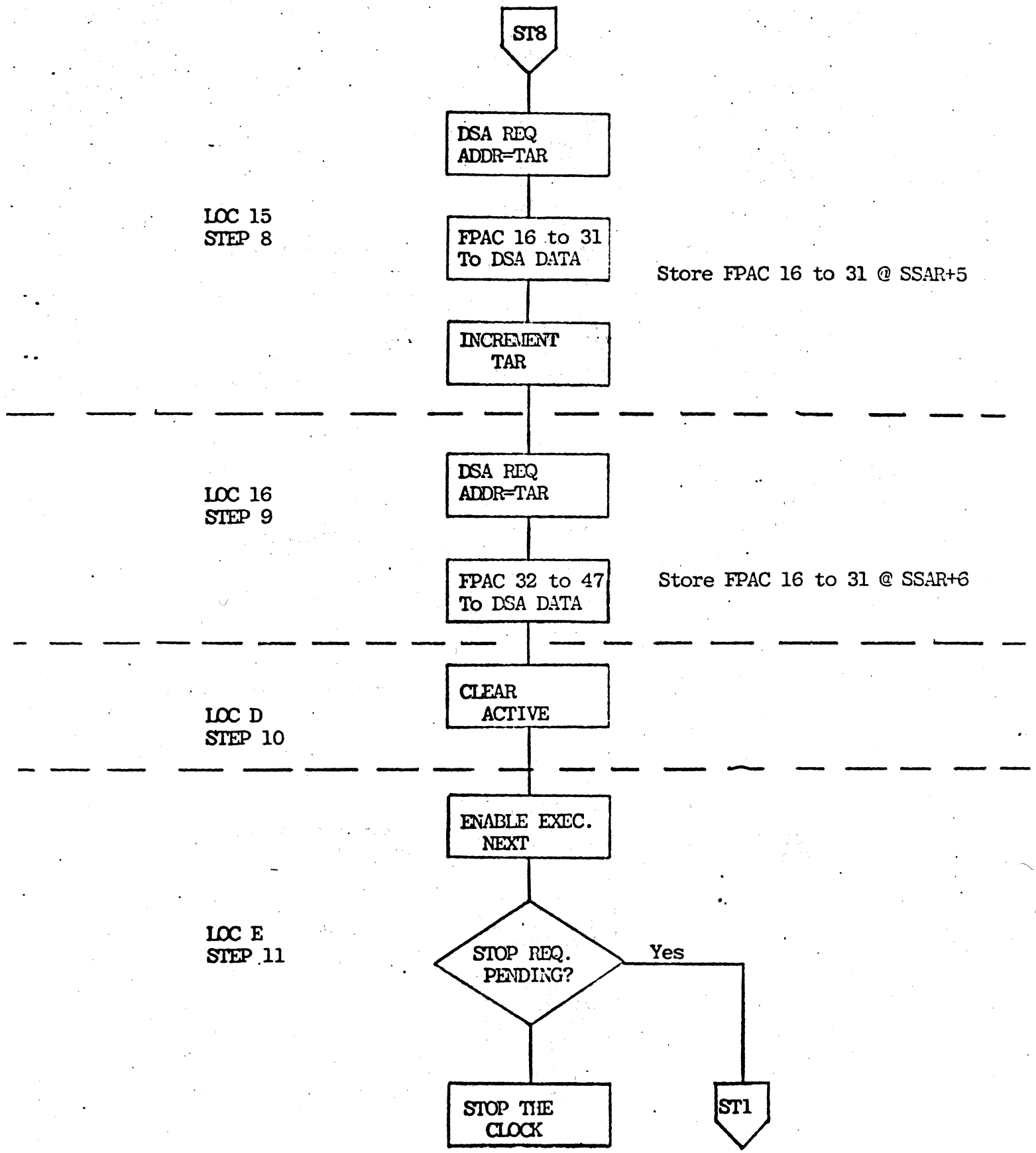
LOC ID
STEP 1

LOC E
STEP 2

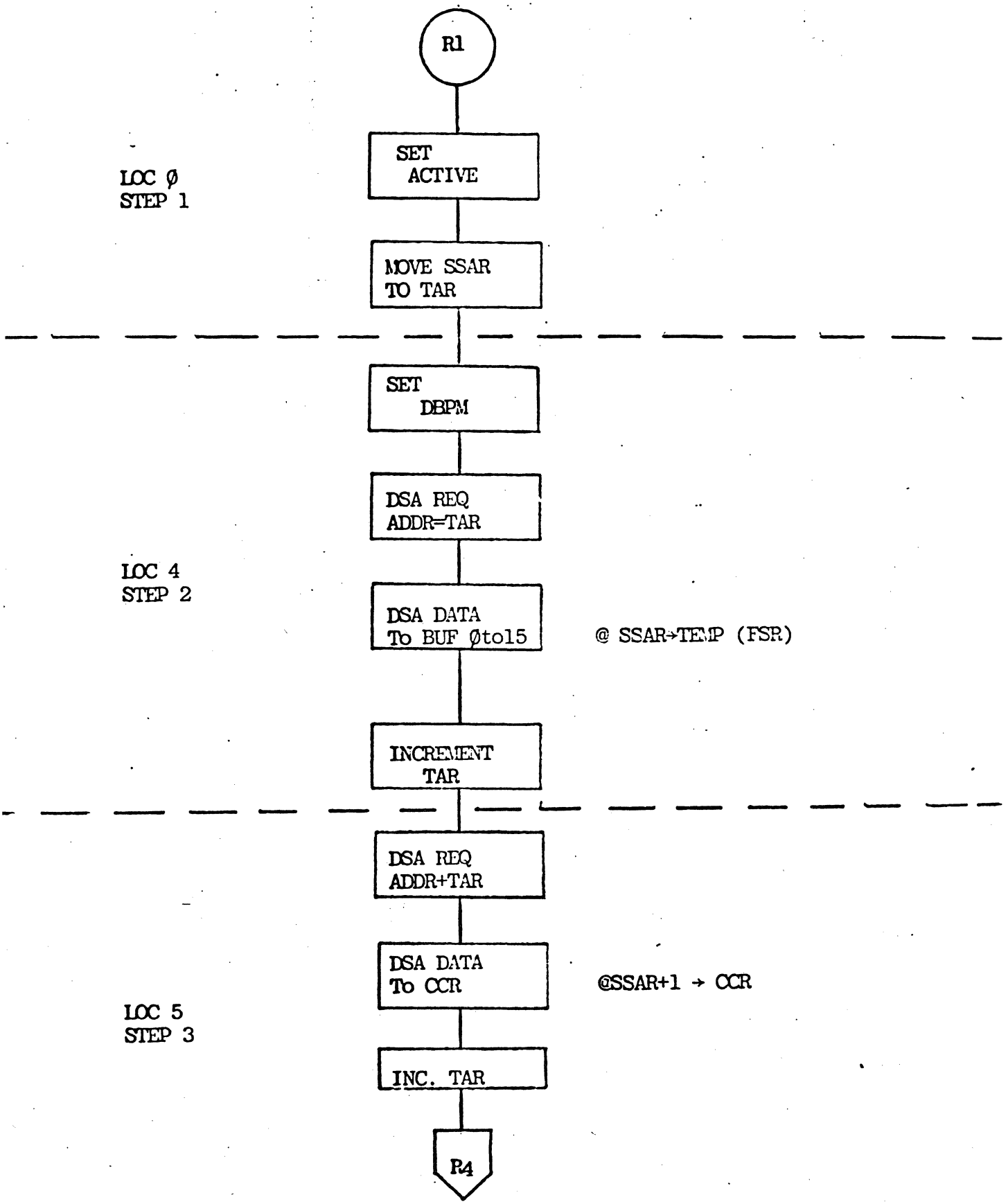
STOP

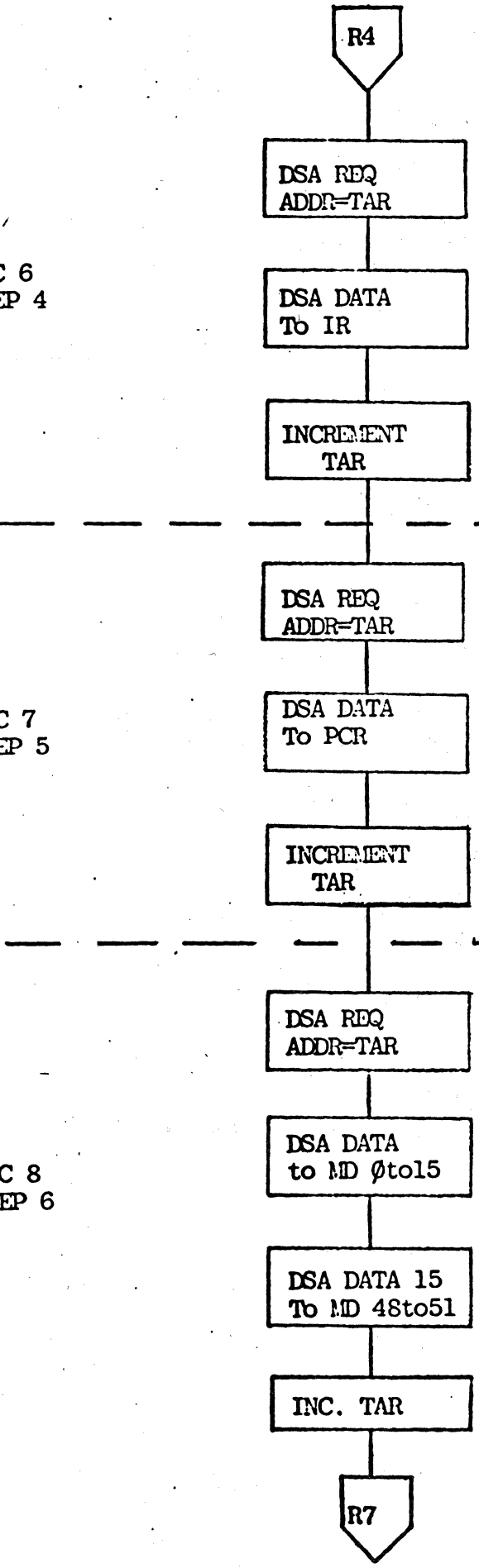

ST1
FPAC Active?
Yes
No
LOC 2
STEP 1
Move FSR to
BUF Ø to 15
Save FSR
Set Active
Set PROT
Get Address
LOC F
STEP 2
Move SSAR
to TAR
DSA REQ
ADDR=TAR
LOC 10
STEP 3
Store FSR @ SSAR
BUF Ø to 15
To DSA DATA
INCREMENT
TAR
DSA REQ
ADDR=TAR
LOC 11
STEP 4
CCR
To DSA DATA
Store CCR @ SSAR+1
INCREMENT
TAR
ST5

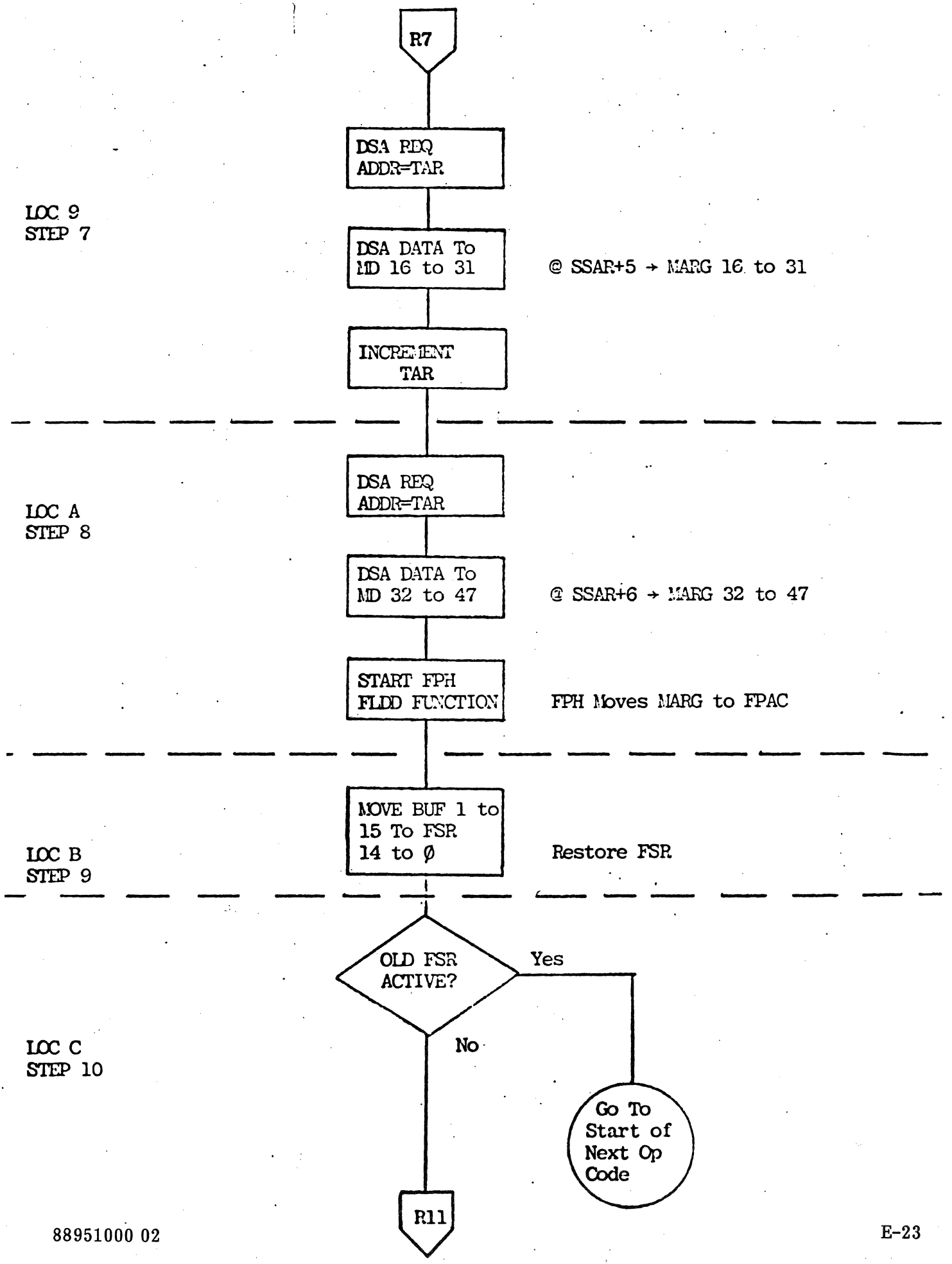

ST5
DSA REQ
ADDR=TAR
LOC 12
STEP 5
IR to
DSA DATA
Store IR @ SSAR+2
INCREMENT
TAR
DSA REQ
ADDR=TAR
LOC 13
STEP 6
PCR To
DSA DATA
Store PCR @ SSAR+3
INCREMENT
TAR
DSA REQ
ADDR=TAR
Store FPAC (Ø to 15)
@ SSAR+4
LOC 14
STEP 7
FPAC Ø to 15
To DSA DATA
INCREMENT
TAR
ST8


ST8
DSA REQ
ADDR=TAR
LOC 15
STEP 8
FPAC 16 to 31
To DSA DATA
Store FPAC 16 to 31 @ SSAR+5
INCREMENT
TAR
DSA REQ
ADDR=TAR
LOC 16
STEP 9
FPAC 32 to 47
To DSA DATA
Store FPAC 16 to 31 @ SSAR+6
CLEAR
ACTIVE
LOC D
STEP 10
ENABLE EXEC.
NEXT
LOC E
STEP 11
STOP REQ.
PENDING?
Yes
STOP THE
CLOCK
ST1

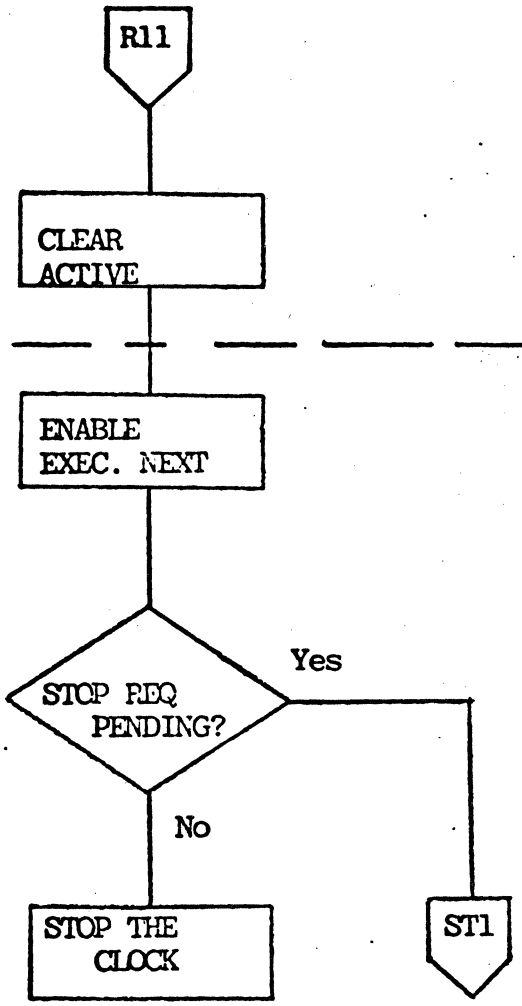
RESTART


R1
SET ACTIVE
LOC Ø
STEP 1
MOVE SSAR TO TAR
SET DBPM
DSA REQ ADDR=TAR
LOC 4
STEP 2
DSA DATA To BUF Øto15
@ SSAR→TEMP (FSR)
INCREMENT TAR
DSA REQ ADDR+TAR
DSA DATA To CCR
@SSAR+1 → CCR
LOC 5
STEP 3
INC. TAR
P4


R4
DSA REQ
ADDR=TAR
Not Consecutive Cycle Due To
Address ALU Interference
LOC 6
STEP 4
DSA DATA
To IR
@SSAR+2 → IR
INCREMENT
TAR
DSA REQ
ADDR=TAR
Not Consecutive Cycle Due To
Address ALU Interference
LOC 7
STEP 5
DSA DATA
To PCR
@ SSAR+3 → PCR
INCREMENT
TAR
DSA REQ
ADDR=TAR
LOC 8
STEP 6
DSA DATA
to MD Øto15
@ SSAR+4 → MARG Ø to 15
DSA DATA 15
To MD 48to51
SIGN EXTEND MARG
INC. TAR
R7


R7
DSA REQ
ADDR=TAR
LOC 9
STEP 7
DSA DATA To
MD 16 to 31
@ SSAR+5 → MARG 16 to 31
INCREMENT
TAR
DSA REQ
ADDR=TAR
LOC A
STEP 8
DSA DATA To
MD 32 to 47
@ SSAR+6 → MARG 32 to 47
START FPH
FLDD FUNCTION
FPH Moves MARG to FPAC
MOVE BUF 1 to
15 To FSR
14 to Ø
LOC B
STEP 9
Restore FSR
OLD FSR
ACTIVE?
Yes
No
LOC C
STEP 10
Go To
Start of
Next Op
Code
R11

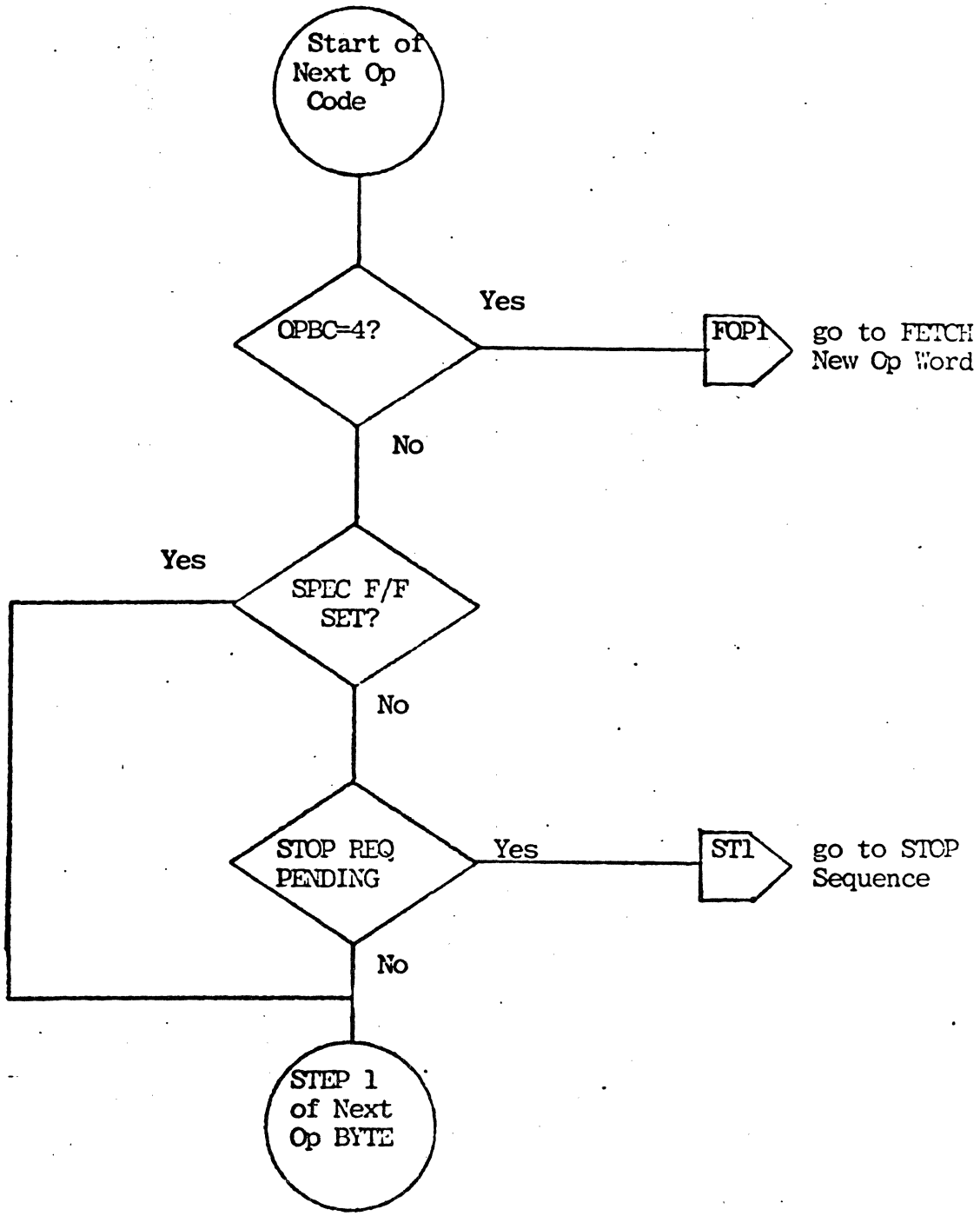
LOC D
STEP 11



LOC E
STEP 12

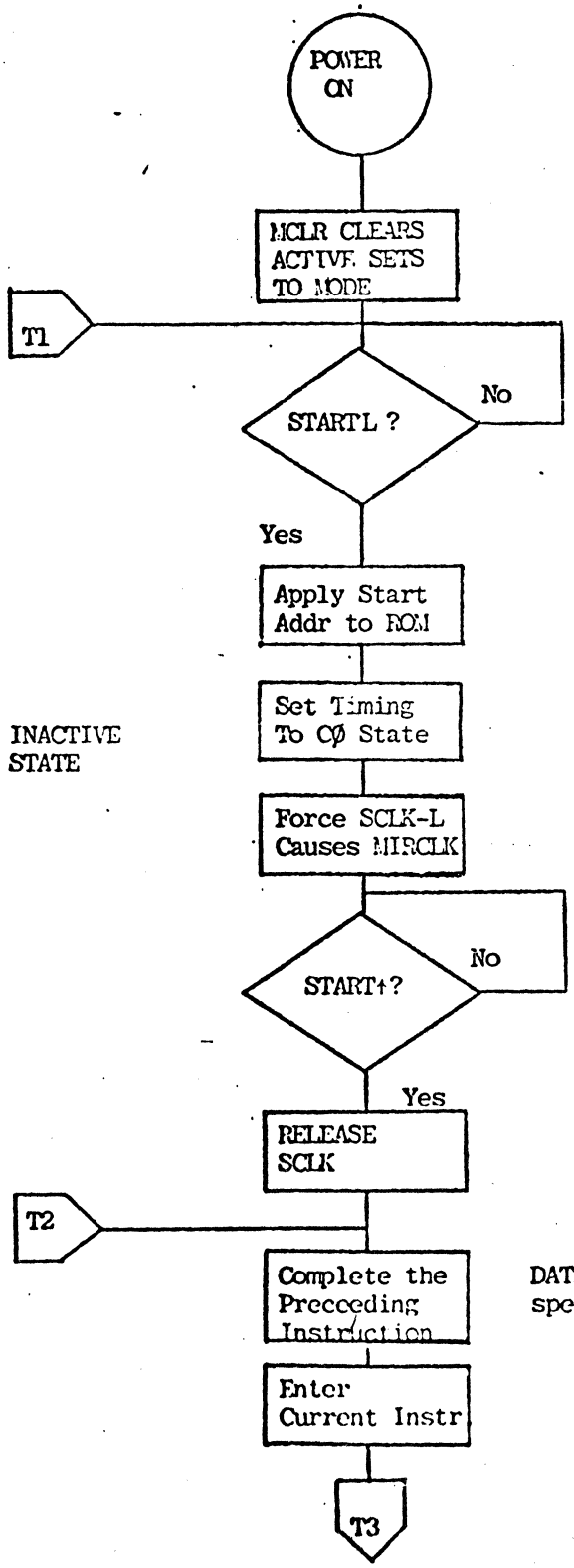
1. C. Flow Charts

Start of Next Op Code = EXEC NXT Bit in Micro-Code



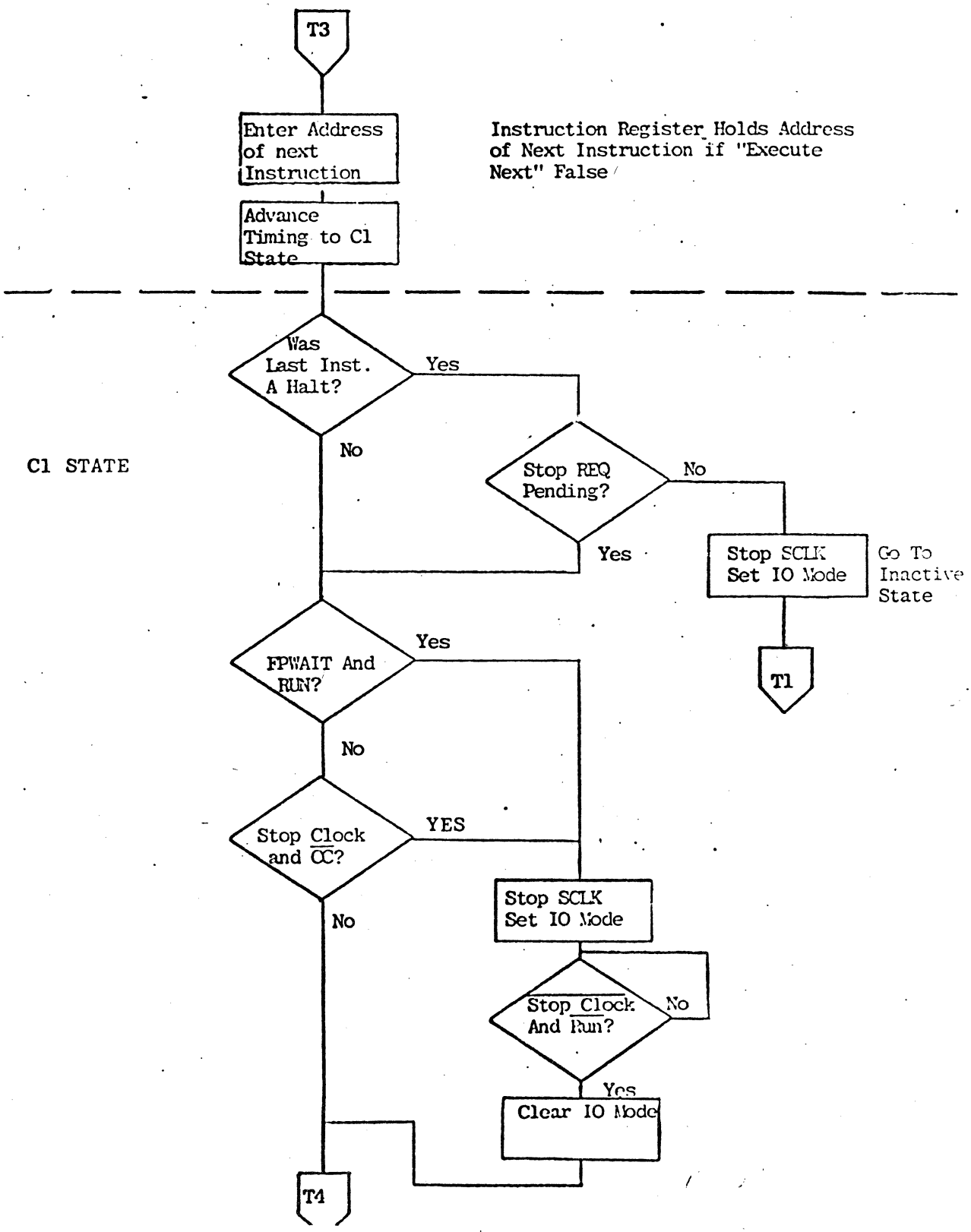
M. C. Flow Charts

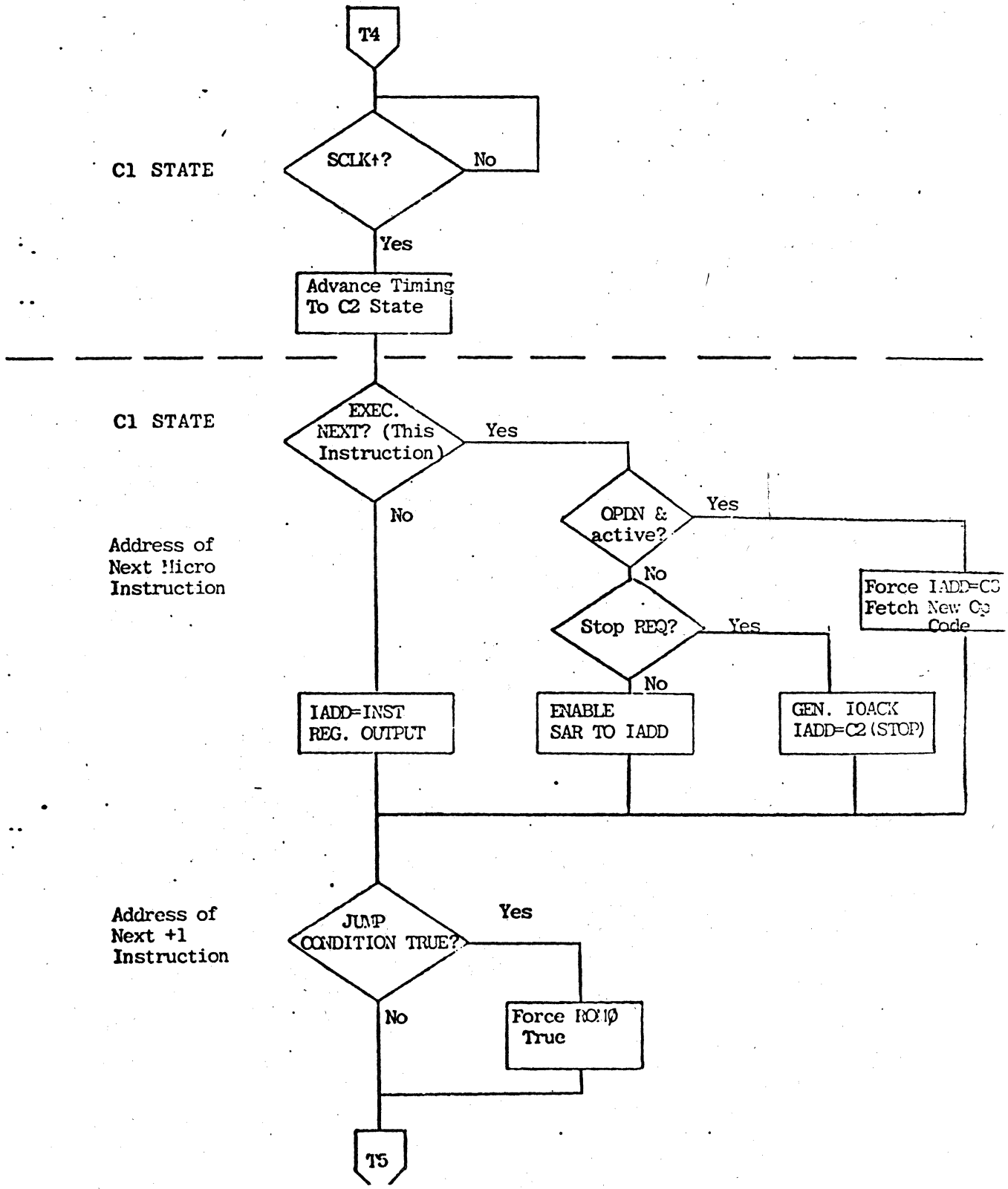
TIMING

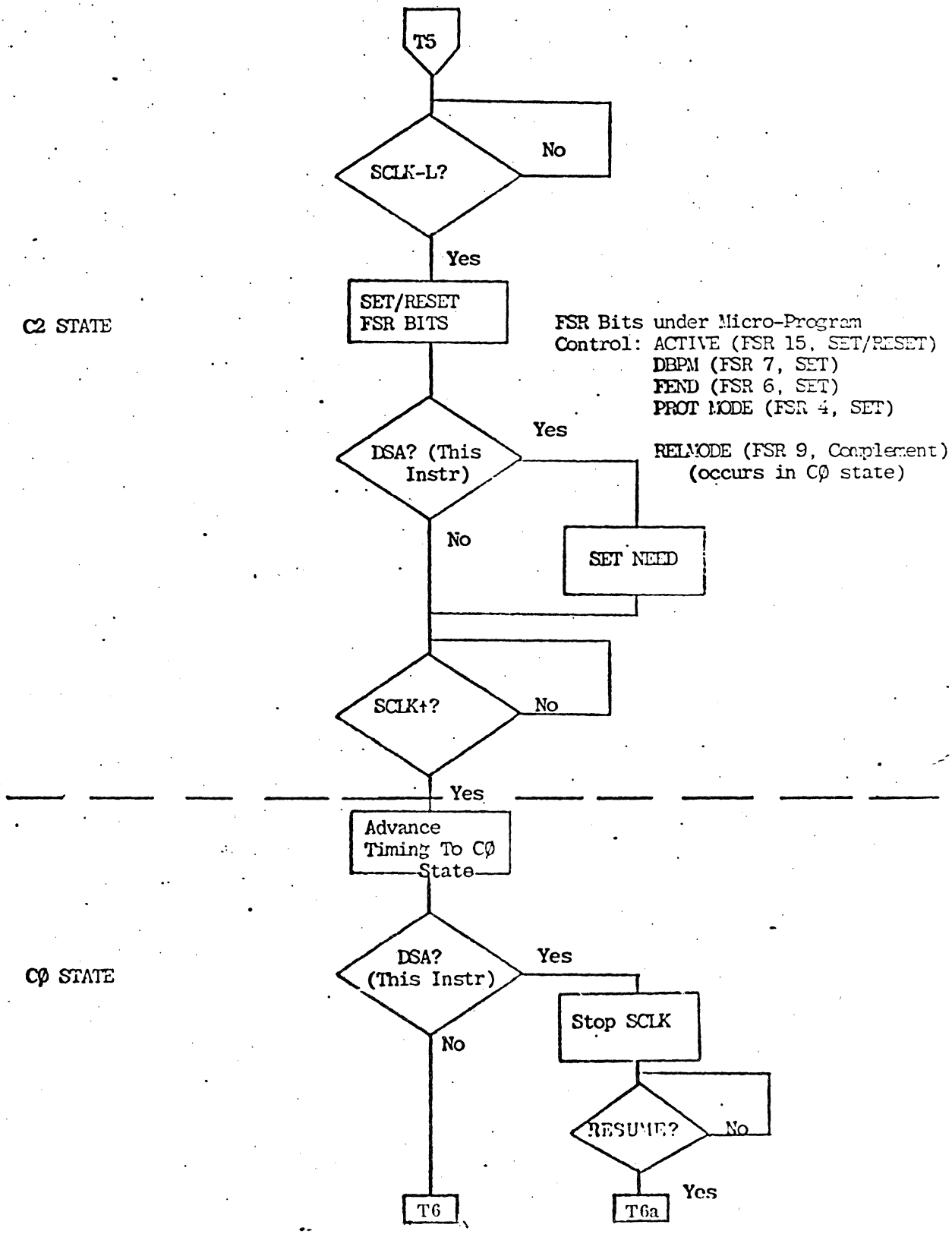


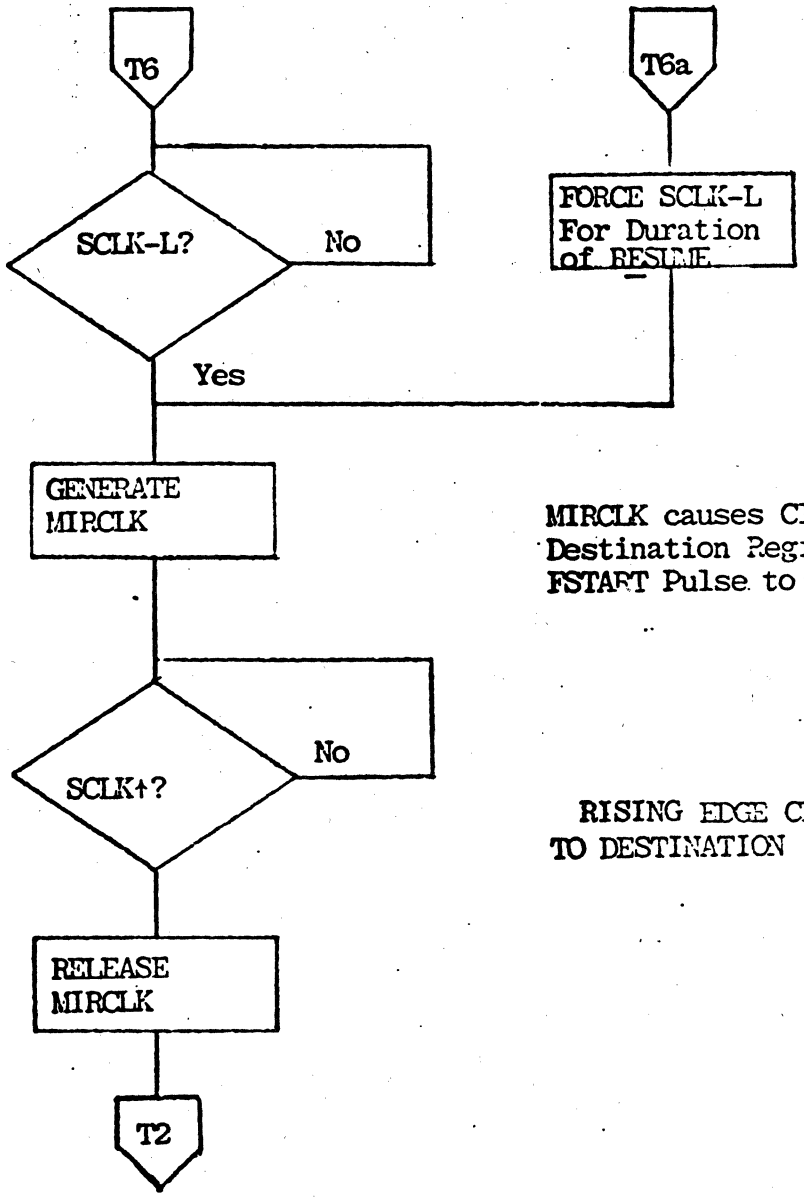
Start Address	Function
CØ	RESTART
01	LDMD1
41	LDMD2
81	LDMD3
C2	STOP
C3	COLD START

DATA is entered into Destination Register specified by Preceding Instruction


T3
Enter Address of next Instruction
Instruction Register Holds Address of Next Instruction if "Execute Next" False
Advance Timing to C1 State
C1 STATE
Was Last Inst. A Halt?
Yes
No
Stop REQ Pending?
No
Yes
Stop SCLK Set IO Mode
Go To Inactive State
T1
FPWAIT And RUN?
Yes
No
Stop Clock and CC?
YES
No
Stop SCLK Set IO Mode
Stop Clock And Run?
No
Yes
Clear IO Mode
T4


T4
C1 STATE
SCLK↑?
No
Yes
Advance Timing To C2 State
C1 STATE
EXEC. NEXT? (This Instruction)
Yes
No
OPDN & active?
Yes
No
Force IADD=C3 Fetch New Op Code
Address of Next Micro Instruction
Stop REQ?
Yes
No
IADD=INST REG. OUTPUT
ENABLE SAR TO IADD
GEN. IOACK IADD=C2 (STOP)
Address of Next +1 Instruction
JUMP CONDITION TRUE?
Yes
No
Force RO:1Ø True
T5


T5
SCLK-L?
No
Yes
C2 STATE
SET/RESET FSR BITS
FSR Bits under Micro-Program Control: ACTIVE (FSR 15, SET/RESET)
DBPM (FSR 7, SET)
FEND (FSR 6, SET)
PROT MODE (FSR 4, SET)
RELMODE (FSR 9, Complement) (occurs in CØ state)
DSA? (This Instr)
Yes
No
SET NEED
SCLK↑?
No
Yes
Advance Timing To CØ State
CØ STATE
DSA? (This Instr)
Yes
No
Stop SCLK
RESUME?
No
Yes
T6
T6a



CØ STATE

MIRCLK causes Clock to Destination Register or FSTART Pulse to FPH

RISING EDGE CLOCKS DATA TO DESTINATION

COMMENT SHEET

MANUAL TITLE CDC® Hardware Floating-Point Unit Hardware Reference Manual

PUBLICATION NO. 88951000 REVISION 02

FROM NAME:

BUSINESS ADDRESS:

COMMENTS: This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number to which your comment applies.

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 333

LA JOLLA, CA.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA 92037

CUT ALONG LINE

FOLD

STAPLE

STAPLE