901060B
TECHNICAL MANUAL

SIGMA 7
CENTRAL PROCESSOR
MODEL 8401
ASSEMBLY NO. 117307

XEROX

PUBLICATION DATA QUICK

PDQ NO. 72-056

PUBLICATION NO. 901060B-1

DATE November 1972

Page 1 of 1

TO: ALL HOLDERS OF XEROX Sigma 7 Central Processor, Model 8401

SUBJECT: TEMPORARY CHANGES TO TECHNICAL MANUAL

The following changes to Technical Manual 901060B are necessary to reflect the latest technical information. The changes are released in this manner for purposes of expediency. The next scheduled revision to the manual will incorporate these changes formally.

PURPOSE: To make corrections in the Environmental Requirements (temperature and humidity) and Primary Power Requirements in this technical manual.

INSTRUCTIONS: Make the following pen and ink changes:

1. Page 1-7, Table 1-3. General Specifications: Under the Specification column for Temperature (electronics), Operating, change to read... 60°F to 90°F; under Characteristics column, immediately below Temperature (electronics), Operating, add the category Relative Humidity and in the Specification column for the Relative Humidity add 25% to 80%.

2. Remove and destroy old title page, insert new title page.

Insert this PDQ instruction sheet between the cover and title page of the manual. Do not remove until this change has been incorporated in a released or re-issue of the technical manual.

TECHNICAL MANUAL

SIGMA 7
CENTRAL PROCESSOR

MODEL 8401
ASSEMBLY NO. 117307

Specifications, equipment descriptions, and procedures contained herein are subject to change without notice.

November 1972

This revision is a supplement to
901060B, dated March 1970

Prepared by
Field Engineering Publications

XEROX

701 So. Aviation Blvd., El Segundo, Calif. 90245, 213 679-4511

XDS

Xerox Data Systems

PUBLICATION DATA QUICK

PDQ NO. 71-016

PUBLICATION NO. XDS 901060

DATE 10/14/71

Page 1 of 2

TO: ALL HOLDERS OF XDS Sigma 7 Computer

SUBJECT: TEMPORARY CHANGES TO TECHNICAL MANUAL

The following changes to Technical Manual XDS 901060B are necessary to reflect the latest technical information. The changes are released in this manner for purposes of expediency. The next scheduled revision to the manual will incorporate these changes formally.

PURPOSE: The purpose of this PDQ is to correct the phase sequence charts per Technical Action Report No. 20890.

INSTRUCTIONS: Make the following changes to the manual with pen and ink.

1. Page 3-661, table 3-143: Add timing signal "T6L" to the Phase column of the table under PCP2.
2. Page 3-663, table 3-143: Add timing signal "T6L" to the Phase column in two places in the table: under PCP7 and PCP1.
3. Page 3-665, table 3-144: Add timing signal "T6L" to the Phase column of the table under PCP2.
4. Page 3-666, table 3-144: Add timing signal "T6L" to the Phase column of the table under PCP2.
5. Page 3-668, table 3-145: In the "Function Performed" column for phase PCP3, change Q15-Q31 ↛ PO-P31 to read Q15-Q31 ↛ P15-P31.

APPROVED: ____________________
MANAGER, PUBLICATIONS DEVELOPMENT

INSTRUCTIONS (Cont.)

6. Page 3-669, table 3-145:

   a. In the "Signals Involved" column for phase PCP5, change:

      LB15-LB31 = P15-P31 NCMXQ NCMXC

      to read

      LB15-LB31 = P15-P31 NLMXQ NLMXC

   b. Add timing signal "T6L" to the Phase column of the table under PCP2.

7. Page 3-670 and page 3-671, table 3-146: Add timing signal "T6L" (one place on each page) to the Phase column of the table under PCP2.

8. Page 3-674, table 3-147, and page 3-676, table 3-148: Add timing signal "T6L" (one place on each page) to the Phase column of the table under PCP2.

9. Page 3-679, table 3-149: Change the first entry in the "Function Performed" column from "Repeated until switch released" to "Repeated until P28 goes high".

10. Page 3-680, table 3-149: Add a horizontal line through the middle of the table to separate the PCP6 phases from the PCP7 phases.

11. Page 3-681, table 3-149: Add timing signal "T6L" to the Phase column of the table under PCP2.

PUBLICATION DATA QUICK

PDQ NO. 70-053

PUBLICATION NO. XDS 901060B

DATE 8 June 1970

Page 1 of 1

TO: ALL HOLDERS OF XDS Sigma 7 Central Processor Model 8401 Assembly 117307

SUBJECT: TEMPORARY CHANGES TO TECHNICAL MANUAL

The following changes to Technical Manual XDS 901060B are necessary to reflect the latest technical information. The changes are released in this manner for purposes of expediency. The next scheduled revision to the manual will incorporate these changes formally.

PURPOSE: The purpose of this PDQ is to update the text and illustrations that apply to the power monitor change affected by Field Modification Kit 8001, 8201, 8401 (drawing 159871).

INSTRUCTIONS: Make the following changes to the manual with pen and ink:

1. Page 3-128, paragraph 3-98. Change AT13 line driver (or module) to AT68 line driver (or module). Total of seven changes.
2. Page 3-129, figure 3-117. Change LINE DRIVER AT13 to LINE DRIVER AT68
3. Page 3-129, paragraph 3-103. Change AT13 modules to AT68 modules
4. Page 3-129, paragraph 3-105. Change AT13 line drivers to AT68 line drivers
5. Page 3-130, Figure 3-118:
   a. Change AT13 (below J3) to AT68.
   b. Add pin 44 to lower part of J3.
   c. Add a connecting line between pin 44 of J3 and pin 8 of P1. Label the connection "+ 60 V filtered".
6. Pages 3-140 and 3-142, paragraph 3-107. Change AT13 to AT68 (three places)

APPROVED: J Alan Crisman

MANAGER, PUBLICATIONS DEVELOPMENT

901060B

Price: $28.25

TECHNICAL MANUAL

SIGMA 7
CENTRAL PROCESSOR

MODEL 8401
ASSEMBLY NO. 117307

Specifications, equipment descriptions, and procedures contained herein are subject to change without notice.

March 1970

This publication supersedes 901060A-1,
dated January 1967, revised April 1967

Prepared by
Field Engineering Publications

XEROX

701 So. Aviation Blvd., El Segundo, Calif. 90245, 213 679-4511

LIST OF EFFECTIVE PAGES

Total number of pages is 826, as follows:

Page No.	Issue	Page No.	Issue
Title	Original		
A	Original		
i thru xx	Original		
1-1 thru 1-8	Original		
2-1 thru 2-14	Original		
3-1 thru 3-736	Original		
4-1 thru 4-10	Original		
A-1 thru A-36	Original		

FOREWORD

This B reissue supersedes XDS publication No. 901060A (preliminary) and contains the following information and changes:

1. Incorporation of the latest revision of all applicable engineering drawings

2. Incorporation, in section IV, of the information previously released in the form of Publications Data Quick (PDQ) 70-001. PDQ 70-001 relates to the replacement of miniature incandescent lamps that are mounted on the processor control panel

3. Deletion of sections V through VIII

4. Addition of an appendix which includes a parts list

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

Section Title Page

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont.)

LIST OF ILLUSTRATIONS (Cont.)

LIST OF ILLUSTRATIONS (Cont.)

LIST OF ILLUSTRATIONS (Cont.)

LIST OF ILLUSTRATIONS (Cont.)

LIST OF TABLES

LIST OF TABLES (Cont.)

LIST OF TABLES (Cont.)

LIST OF TABLES (Cont.)

LIST OF RELATED PUBLICATIONS

The following publications contain information not included in this manual but necessary for a complete understanding of the Sigma 7 Central Processing Unit when used with related XDS equipment.

Publication Title	Publication No.
XDS T-Series, Integrated Circuit Logic Modules, Description and Specifications	645103
Sigma 7 CPU Diagnostic System (Sense), Diagnostic Program Manual	900824
Sigma 5 and 7 CPU Diagnostic Program (Verify), Diagnostic Program Manual	900870
Sigma 7 CPU Diagnostic-Auto, Diagnostic Program Manual	900872
Sigma 7 CPU Diagnostic System (Pattern), Diagnostic Program Manual	900891
Sigma 7 CPU Diagnostic (Suffix), Diagnostic Program Manual	900893
Sigma 5 and 7 CPU Diagnostic-Float, Diagnostic Program Manual	900898
Sigma 7 CPU Diagnostic System (Decimal), Diagnostic Program Manual	900908
Sigma 7 MAP Diagnostic Program, Diagnostic Program Manual	900920
Sigma 7 Computer Reference Manual	900950
Sigma 5 and 7 Systems Test Monitor, Diagnostic Program Manual	901076
Sigma 5 and 7 Interrupt Test, Diagnostic Program Manual	901134
Sigma 5 and 7 Power Fail-Safe Test, Diagnostic Program Manual	901135
Sigma 5 and 7 Real-Time Clock Test, Diagnostic Program Manual	901136
Sigma 5 and 7 CPU Diagnostic Program (Memory Protect), Diagnostic Program Manual	901516
Sigma 5 and 7 Core Memory, Technical Manual	901586
Decimal Arithmetic Unit Model 8419, Technical Manual	901587

SECTION I
GENERAL DESCRIPTION

1-1 INTRODUCTION

This manual contains information necessary to install, operate, and maintain the Sigma 7 Computer System manufactured by Xerox Data Systems, El Segundo, California. Section I provides a general physical and functional description. Subsequent sections contain information on operation and programming, complete principles of operation, maintenance instructions, and tabular parts lists of printed circuit modules.

The List of Related Publications in the front matter of this manual references technical manuals and programming manuals describing equipment and programs associated with the Sigma 7 computer system.

The Sigma 7 computer is a high-speed, general-purpose digital computer for use in business, scientific, process control, hybrid, and systems applications. The computer functions efficiently in real-time, time-sharing, and multiusage computing environments.

Figure 1-1 shows a maximum Sigma 7 computer system configuration without peripheral input/output equipment.

1-2 PHYSICAL DESCRIPTION

1-3 BASIC COMPUTER SYSTEM

The basic computer system contains a central processing unit (CPU) contained in two cabinets, an expandable memory contained in one to four cabinets, and an input/output processor (IOP). The CPU, IOP, and memory are comprised of printed circuit modules inserted into chassis each of which may contain up to 32 modules. Pins at the rear of the modules are plugged into sockets mounted on a rear wiring board that contains all wire connections. Module sizes are identical except for the core diode modules in the memory units. These modules occupy the vertical space normally filled by two standard modules and requires a double-sized chassis.

Table 1-1 lists the main units in a basic Sigma 7 computer system.

1-4 OPTIONAL FEATURES

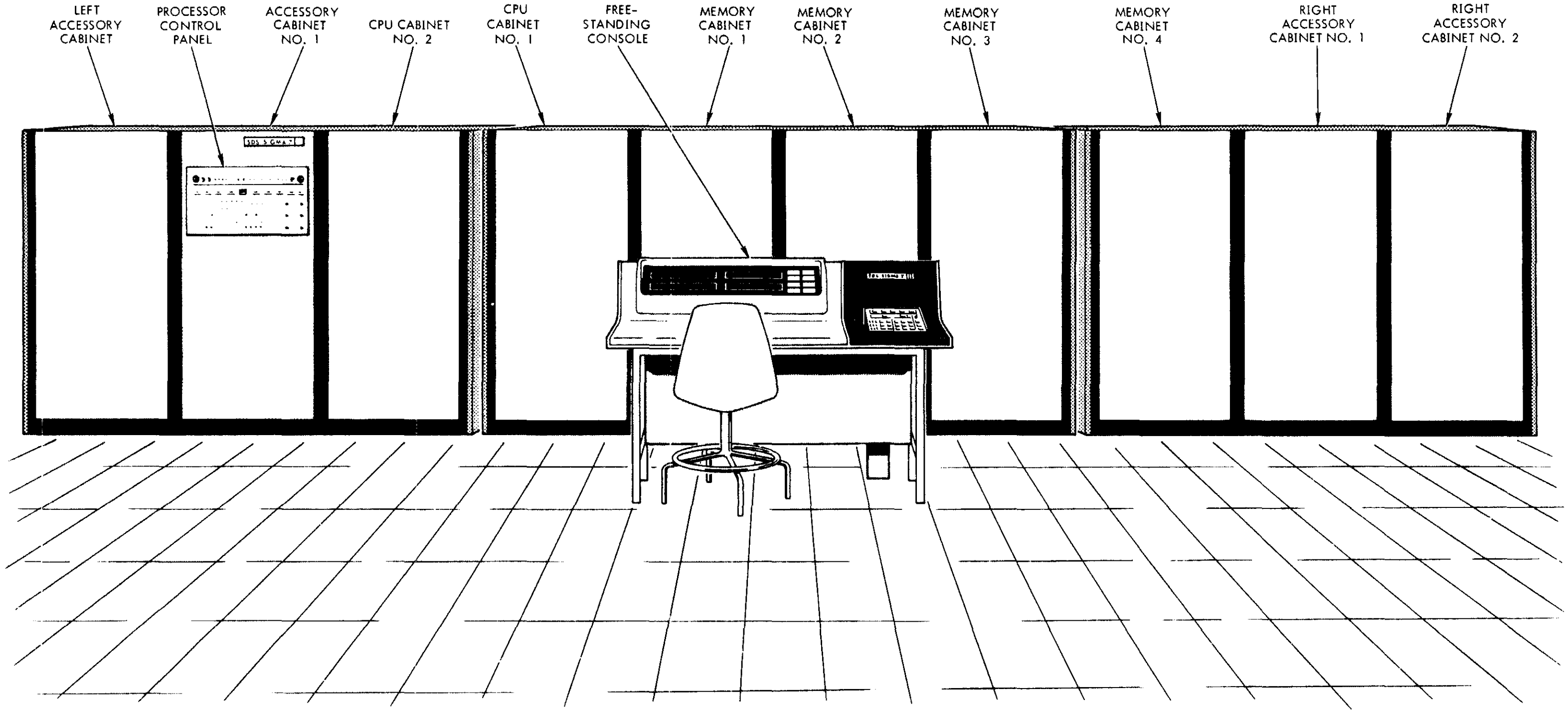
Optional features that may be added to the computer are listed in table 1-2. Many of these features are additional modules that are plugged into the CPU; others are added to accessory cabinets or memory cabinets as needed. The following optional equipment is added by plugging additional modules into the chassis in CPU cabinets 1 and 2: power fail-safe, memory protection, memory map, floating point, two additional real-time clocks (two real-time clocks are part of the basic computer), three additional high-speed register banks used as private memory registers, and six internal interrupt levels. The main part of the optional decimal arithmetic feature is contained in three separate chassis located in CPU cabinet No. 2, and some of the decimal modules are in CPU cabinet No. 1. Three private memory register banks, in addition to the one bank contained in the basic omputer, may be included in the CPU logic modules in CPU cabinet No. 1; the remaining additional register banks are obtained by adding separate chassis to the memory cabinets. Each external interrupt chassis provides control and mounting space for up to eight interrupt modules and contains two interrupt levels per module.

1-5 CABINET TYPES

Two types of cabinets are used in Sigma 7 computers. The memory cabinets may contain two hinged frames and a fixed

Table 1-1. Main Units

Model No.	Nameplate Nomenclature or Assembly Drawing Title	Common Name	Assembly Drawing No.	Location (See figure 1-1)
8401	XDS Sigma 7	Central Processing Unit (CPU)	117307	CPU cabinets Nos. 1 and 2
8451	Basic 4K x 33 Bit	4K Memory	132546	Memory cabinet
8471	Input/Output Processor (IOP)	Multiplexing Input/Output Processor (MIOP)	117610	CPU cabinet No. 2



901060B.101

Figure 1-1. Sigma 7 Computer (Typical Configuration)

Table 1-2. Optional Features

Model No.	Nameplate Nomenclature or Assembly Drawing Title	Common Name	Assembly Drawing No.	Location (See figure 1-1)
8411	Real-Time Clock	Two Additional Real-Time Clocks	117616	CPU cabinet No. 2
8413	Power Fail-Safe	Power Fail-Safe	117612	CPU cabinet No. 2
8414	Memory Protection Feature	Memory Protection	117617	CPU cabinet No. 2
8415	Memory Map Feature	Memory Map	117615	CPU cabinet No. 2
8416	High-Speed Register Bank or	Private Memory Extension	117621	CPU cabinet No. 1
	High-Speed Register Bank and	Private Memory Extension	117621	Memory cabinet No. 1 or 2
	REU and	Register Extension Unit	130071	Memory cabinet No. 1 or 2
	REU Interface	REU Interface	132208	CPU cabinet No. 1
8418	Floating Point Feature	Floating Point Feature	117611	CPU cabinet No. 1
8419	Decimal Arithmetic Feature	Decimal Unit	117613	CPU cabinet No. 1 and CPU cabinet No. 2 or accessory cabinet No. 1
8421	Priority Interrupt (Nameplate - ICC)	External Interrupt Chassis	117330	CPU cabinet No. 2
8422	Interrupt 2 Level	Internal Interrupt, Two Levels	132206	CPU cabinet No. 2
8422	Interrupt 2 Level	External Interrupt, Two Levels	132206	External interrupt chassis, CPU cabinet No. 2
8452	Memory Expansion Kit 4K to 8K	Memory Expansion to 8K	117638	Memory cabinet
	Memory Expansion Kit 8K to 12K	Memory Expansion to 12K	117639	Memory cabinet
	Memory Expansion Kit 12K to 16K	Memory Expansion to 16K	117640	Memory cabinet
8456	Port Expansion 2 x 3	Two- to Three-Port Expansion	128125	Memory cabinet
8457	Memory Port Expander "F"	Three- to Six-Port Expansion	130625 (one memory)	Memory cabinet
	Memory Port Expander "S"		130626	
	Additional MIOP		(two memories)	
8472	IOP/DC Expansion	Additional Eight Subchannels	117618	IOP

(Continued)

Table 1-2. Optional Features (Cont.)

Model No.	Nameplate Nomenclature or Assembly Drawing Title	Common Name	Assembly Drawing No.	Location (See figure 1-1)
8482	Secondary Selector IOP	Additional Selector Channel	133997	CPU cabinet No. 2, accessory cabinet No. 1 or No. 2
8485	Selector Input/Output Processor	Selector Input/Output Processor	117620	Memory or accessory cabinet
8491	Controller, CFE-3	CFE-3	139237	Accessory cabinet
8492	Multi-Add Unit, CFE-3	CFE-3	139228	Accessory cabinet
8495	Free-Standing Console	Free-Standing Console	127418	Separate console

frame; the CPU, accessory, and I/O cabinets may contain two hinged frames. Power supplies are mounted on the side of the frames. In the CPU, I/O, and accessory cabinets, power supplies are also mounted in the rear of the cabinet. Examples of the two typesof cabinets areshown in figure 1-2.

1-6 COMPUTER CONFIGURATION

A maximum computer system is comprised of the cabinets shown in figure 1-1. The two CPU cabinets, a memory cabinet, and an I/O cabinet are always included. Accessory cabinets, additional memory, and I/O cabinets are included as needed.

1-7 FUNCTIONAL DESCRIPTION

1-8 BASIC COMPUTER SYSTEM

For purposes of description, a minimum computer system is defined as a CPU, a 4K memory, an IOP, a device controller, and a device, as shown in figure 1-3. The computer is comprised of a CPU, a 4K memory, and an IOP.

Although the CPU physically consists of rows of modules, certain basic functional elements can be identified. These are arithmetic and control logic and associated registers, two real-time clocks, a watchdog timer, seven internal interrupt levels, a clock generator, a 1.024-MHz clock oscillator, and a 16-register bank of private memory. The functions of the CPU are to address core memory, to fetch and store information, to perform arithmetic and logical operations, to sequence and control instruction execution, and to control the exchange of information between core memory and other elements of the system.

The memory contains magnetic core storage; addressing, port priority, and control logic, a timing signal generator, and drive, predrive, inhibit, and sensing circuits. The memory stores 32-bit words and generates and checks parity for each word. Partial words may be stored in the form of 8-bit bytes and 16-bit halfwords. All memory is directly addressable by both the CPU and the IOP.

The IOP contains input and output data storage registers and buffers, fast-access memory registers for command manipulation, a timing signal generator, and control logic.

The function of the multiplexing IOP is to control and sequence simultaneously input and output operations for eight (expandable to 32) peripheral devices, allowing the CPU to concentrate on program execution. The active devices timeshare the hardware in the IOP. For each device connected to the IOP, a storage unit (subchannel) is included in the IOP. Any input/output events that require CPU intervention are brought to the attention of the CPU by means of the interrupt system. Device controller and other devices are described in other technical manuals.

1-9 OPTIONAL FEATURES

1-10 Two Additional Real-Time Clocks

This feature adds interrupt capability for two real-time clocks in addition to the two already in the CPU. When this feature is installed, the CPU has four independent real-time clocks, and each is controlled separately by programming. The clocks can be used as either elapsed-time counters or as real-pulse accumulators. The real-time clock frequency is selectable by means of a switch module in CPU cabinet No. 2. The optional frequencies of operation are 8 kHz, 2 kHz, 500 Hz, 60 Hz, and an external frequency supplied by the user.

1-11 Power Fail-Safe Feature

The power fail-safe feature detects an imminent failure of primary power and, with the help of programming, brings the system to an orderly halt while power is still at a sufficient level to permit reliable operation. After a shutdown, this feature automatically senses that power has returned to

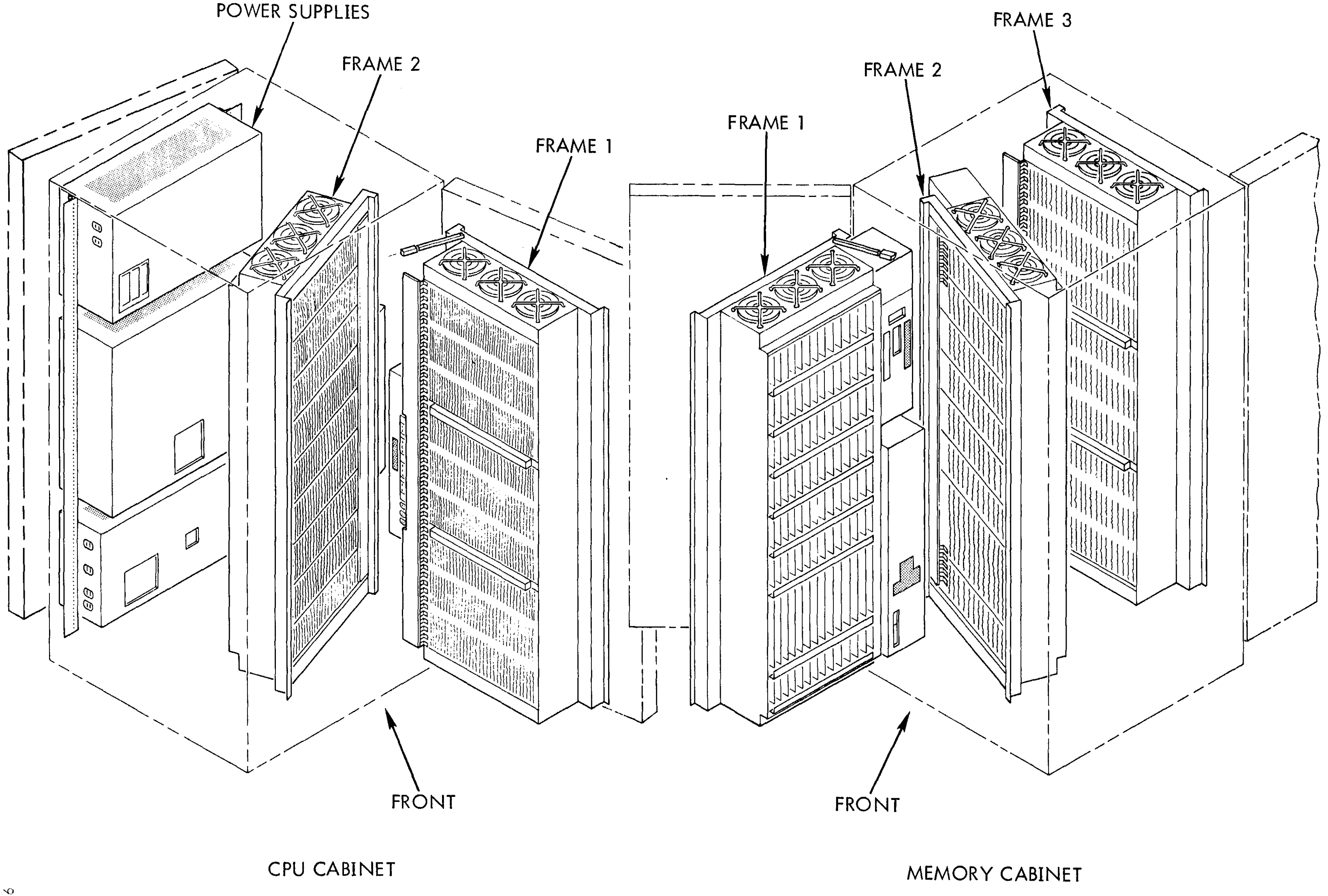


Figure 1-2. Cabinet Types

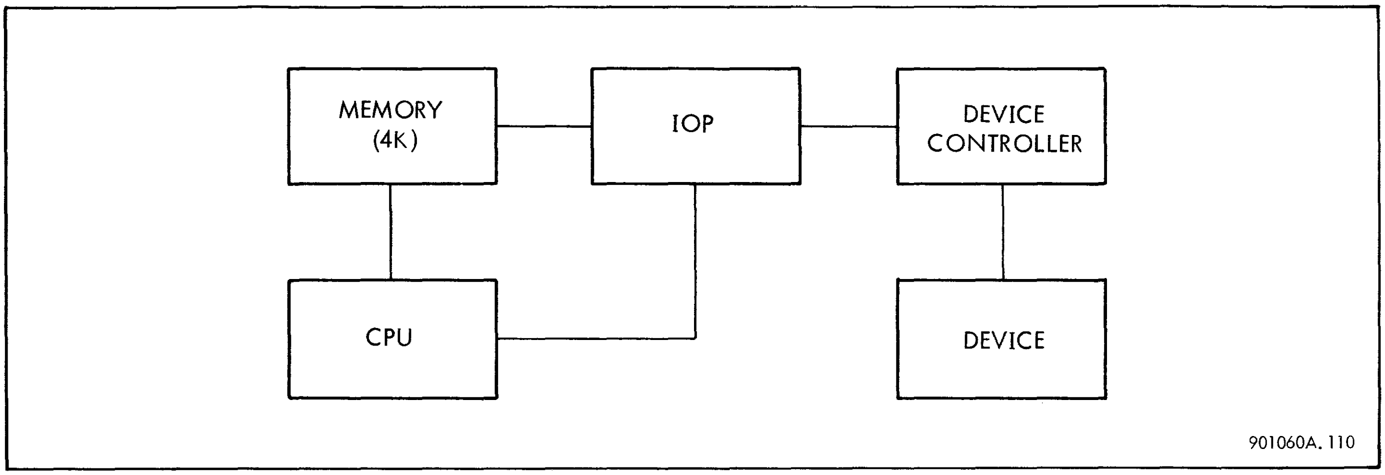


Figure 1-3. Sigma 7 Minimum System

a normal level and causes the machine to resume computation under program control at the point of prior interruption. The contents of all volatile registers are saved in nonvolatile magnetic core memory before shutdown occurs. The register contents are restored as part of the start-up routine.

1-12 Memory Protection

The memory protection feature allows concurrent running of both real-time (foreground) and background programs. A foreground program is protected against destruction by an unchecked background program. The memory protection feature allows protected areas of memory to be written in only under specified conditions.

1-13 Memory Map

The memory map provides the means for automatic relocation of programs and controls the problems of memory fragmentation. Regardless of the addresses used by the operating program, the memory map permits the program to reside anywhere in memory, including in noncontiguous areas. Access to these areas for reading or writing can be program-controlled.

1-14 Private Memory Register Extension

The private memory register extension provides additional private memory registers in blocks of 16 registers each. Up to 31 private memory register blocks may be added, making a total of 32 banks in the computer.

1-15 Floating Point Feature

The floating point feature enables floating point arithmetic to be performed, using an 8-bit exponent and a 24- or 56-bit fraction. Normalized or unnormalized modes of addition and subtraction may be selected by the program.

1-16 Decimal Arithmetic Feature

The decimal arithmetic feature performs arithmetic and logical operations on binary-coded decimal data, adding decimal add, subtract, multiply, divide, compare, shift, load, store, pack, and unpack logical operations and a comprehensive edit-byte instruction to the CPU functions.

1-17 External Interrupts

The maximum external interrupt system provides 224 interrupt levels in addition to those already existing internally in the CPU. Each level can be individually armed or enabled under program control. External interrupts are added to the computer with a maximum of 16 per chassis. Priorities are established at the time of installation.

1-18 Memory Expansion

Memory size can be expanded in increments of 4096 words to a maximum of 131,072 words.

1-19 Memory Port Expansion

Each memory bank may have from two to six entry ports, each of which may be connected to a memory bus containing data, address lines, and control signal lines. Each memory bus provides access to memory for one CPU, IOP, or other unit. The basic computer system includes two ports for each memory bank; an optional third port may be added for three-way access, and, if the third port is included, an optional port expander of four ports may be used to provide a total of six entry ports. Since each CPU or IOP has its own bus to any memory bank, a computer system with more than one memory bank can have more than one memory access occurring simultaneously.

1-20 Additional Eight Subchannels (IOP)

To increase the number of devices connected to one IOP, additional subchannels may be added in increments of eight up to a maximum of 32 subchannels for 32 devices.

1-21 Selector Input/Output Processor

The selector IOP provides control, sequencing, and data transmission for up to 32 high-speed peripheral devices operating one at a time. These devices may have data rates that would exceed the bandwidth of the multiplexing IOP or would use up such a large percentage of the bandwidth that it would be impractical to run any other device concurrently.

An optional selector channel identical to the first selector IOP may be added when a second high-speed data path is required. This optional additional channel may share the same memory with the first selector channel.

1-22 System Supervisory Console

The system supervisory free-standing console is a separate operator's control and display unit, featuring hexadecimal entry and display. Total system status is constantly presented for monitoring of system performance. A display of the rate of instruction execution is included.

1-23 Six Internal Interrupt Levels

Seven internal interrupt levels are included in the standard computer. The following internal interrupts are optional: two power interrupts (power fail-safe), two counter interrupts, and two counter zero interrupts.

1-24 SPECIFICATIONS

The general specifications for the Sigma 7 computer are given in table 1-3.

The input power specifications for the power supplies used in the computer are given in table 1-4. Power supply PT14 receives 60-Hz power from the main power source and supplies 60 Vdc to the PT15 power supply. The 120V, 2000-Hz output of the PT15 power supply is used as an input to the PT16, PT17, and PT18 power supplies. Since the PT14 and PT15 power supplies are always in series, the input and output power specifications are given as if the two were one power supply. The power output in watts from the PT16, PT17, and PT18 power supplies is determined by the power requirements of the computer. Power requirements for peripheral devices are given in the technical manuals for those devices.

The fuses and circuit breakers for the computer are contained in the power supplies and in the processor control panel and are listed in table 1-5.

The indicator lamps are located on the processor control panel and are listed in table 1-6 with reference designators and ratings.

Table 1-3. General Specifications

Characteristic	Specification
Temperature (electronics)	
Nonoperating	-40°C to 60°C (-40°F to 140°F)
Operating	5°C to 50°C (41°F to 122°F)
Altitude	
Nonoperating	20,000 ft max
Operating	10,000 ft max
Memory cycle	
Without interleaving	850 ns
With interleaving	635 ns effective time
Logic signal levels	ONE,+4V; ZERO, 0V
Word length	32 bits plus parity bit
Data format	8-bit byte, 16-bit halfword, fixed and floating point word, fixed and floating point doubleword
Coding	Binary

Table 1-4. Power Supply Input Power Specifications

Power Supply	Power Input
PT14, PT15	1.66 times VA output (2000 Hz)
PT17 (2400 Hz input)	150 +1.36 times dc output in watts (table 1-5)

Table 1-5. Fuses

Name	Type	Reference Designator	Rating	Location	Circuit Protected
Circuit breaker	PAM 666	CB1	24 Vdc	Power Supply PT14	Main power
Fuse	3AG	F1	125V, 8A	Power Supply PT16	120V, 2000 Hz input
Fuse	3AG	F2	125V, 8A	Power Supply PT16	120V 2000 Hz input
Fuse	3AG	F3	250V, 10A	Power Supply PT16	120V 2000 Hz input
Fuse	3AB	F1	250V, 15A	Power Supply PT17	Input circuit
Fuse	3AG	F1	250V, 4.0A	Power Supply PT18	120 Vac input
Fuse	3AG	F2	32V, 7.5A	Power Supply PT18	+25V
Fuse	3AG	F3	250V, 1.5A	Power Supply PT18	-25V
Fuse	3AG	F4	32V, 10.0A	Power Supply PT18	+8V
Fuse	3AG	F5	250V, 1.5A	Power Supply PT18	+50V

Table 1-6. Lamps

Type	Reference Designator	Rating	Quantity
Miniature incandescent	DS1 - DS21, DS29 - DS98	5V, 53 mA	91
Miniature incandescent	DS22 - DS26, DS28, DS100	6.0V, 0.20A	14

SECTION II

OPERATION AND PROGRAMMING

2-1 INTRODUCTION

This section contains operating instructions and programming description. Sigma 7 operating instructions describe the purpose and function of the processor control panel (PCP), its control switches and displays. The paragraphs describing programming include instruction and data formats and computer modes. Descriptions of individual instructions can be found in the Sigma 7 Computer Reference Manual, publication No. 900950, and in the operation code descriptions in section III of this manual.

2-2 OPERATION

The following paragraphs describe the operating procedures required during computer maintenance. All operations are carried out from the PCP.

2-3 CONTROLS AND INDICATORS

The PCP is divided into two parts: a maintenance section on the upper half of the panel and an operator's or programmer's section on the lower half of the panel. The various control switches, indicators, and displays on the PCP are shown in figure 2-1. Table 2-1 lists the switches and indicators found on the programmer's section of the PCP with their reference designator and a brief functional description. A similar list of the switches and indicators on the the maintenance section of the PCP is given in table 2-2.

2-4 OPERATING PROCEDURES

The following paragraphs describe step-by-step manual procedures for the various operations from the processor control panel.

2-5 Applying Power

Both ac and dc power are applied to the CPU and to all units connected to it when the POWER switch is pressed. The POWER switch is lighted when ac power is applied to the system.

2-6 Displaying Contents of Memory Location

2-7 SELECTED ADDRESS. To display the contents of any core memory location or private memory register, perform the following steps.

a. Set the COMPUTE switch to IDLE. The PHASE display indicates that the CPU is in phase PCP2.

b. Place the address of the memory location (or of the register in the current private memory register bank) into the SELECT ADDRESS switches.

c. Place the DISPLAY switch in the momentary SELECT ADDR position.

d. Observe the binary contents of the selected address in the DISPLAY indicators.

e. Release the DISPLAY switch. The DISPLAY indicators still contain the contents of the selected location. The current instruction must be returned to the DISPLAY indicators by setting the DISPLAY switch to the momentary INSTR ADDR position, if the program is to be resumed by setting the COMPUTE switch to RUN or STEP.

The contents of the POINTER field must be changed to point to the desired register bank, to observe the contents of a private memory register in a register bank other than the one currently displayed by the POINTER field of PSW2. This operation is described in paragraph 2-12.

2-8 INSTRUCTION ADDRESS. To display the contents of the memory location indicated by the INSTRUCTION ADDRESS indicators, perform the following steps:

a. Set the COMPUTE switch to IDLE. The PHASE indicators show that the CPU is in phase PCP2.

b. Place the DISPLAY switch in the momentary INSTR ADDR position.

c. Observe the DISPLAY indicators, which now display the contents of the instruction address location.

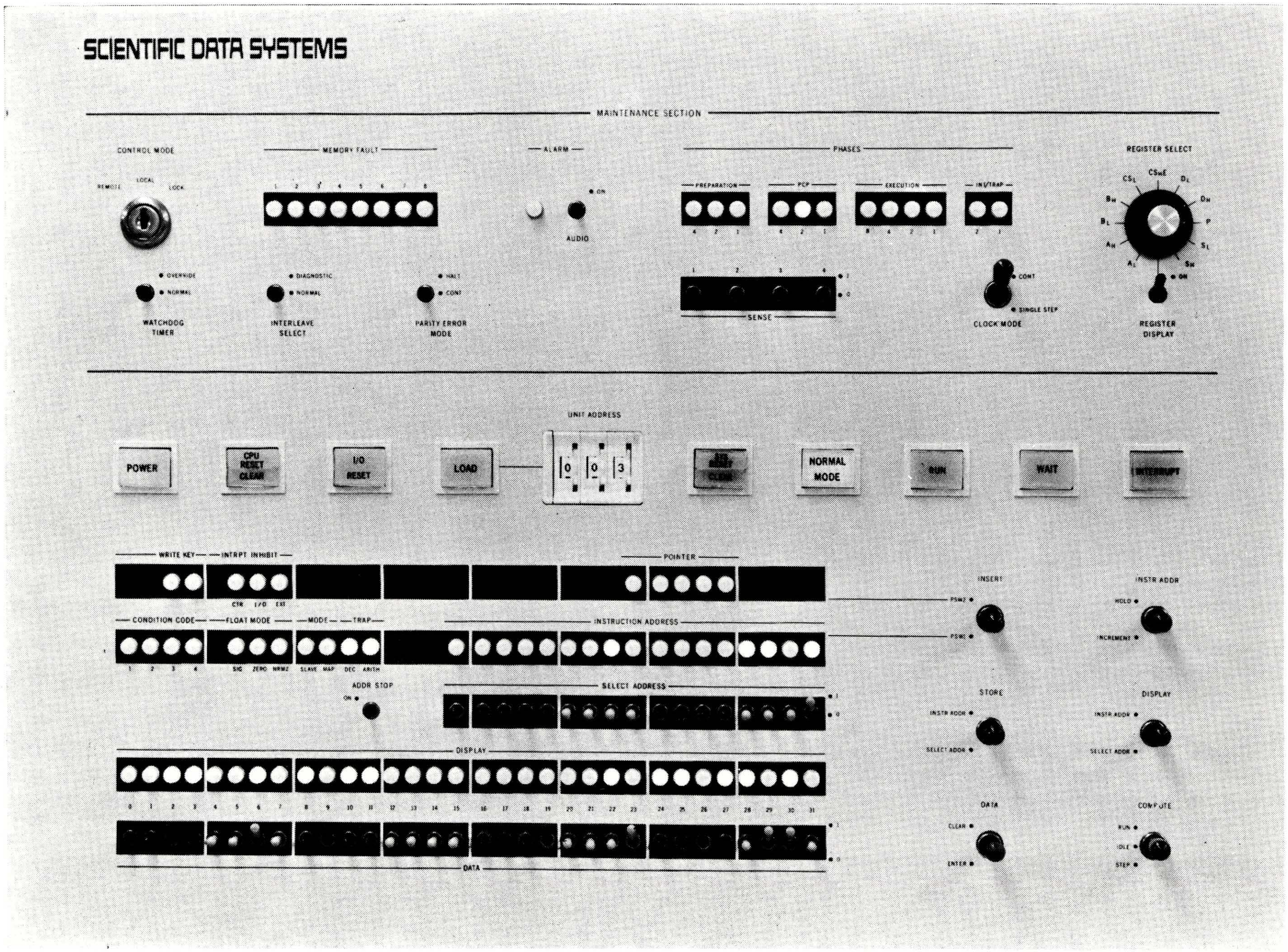
d. If successive memory locations are to be displayed, perform steps a, b, and c and momentarily set the INSTR ADDR switch to the INCREMENT position. The display indicates the contents of the INSTRUCTION ADDRESS indicator, which have been incremented by one. Continue setting the INSTR ADDR switch to INCREMENT to successive memory locations to be displayed.

The memory map is in effect if the memory map option is included and if the MAP MODE indicator is lighted. Memory protection, if included, is inhibited in these PCP operations.

2-9 Changing Contents of a Memory Location

2-10 SELECTED ADDRESS. To store data into selected memory locations, in core memory or in private memory, perform the following steps.

a. Set the COMPUTE switch to IDLE. The PHASE display indicates that the CPU is in phase PCP2.



901060B. 233

Figure 2-1. Processor Control Panel

Table 2-1. Controls and Indicators, PCP Programmer's Section

Control or Indicator	Function
POWER Pushbutton	Push-on, push-off switch supplies ac power to CPU and to all units under its control. When power is first applied, indicator lights and a signal is generated in the power monitor to initialize system. All reset functions normally performed by CPU RESET/CLEAR and SYSTEM RESET/CLEAR switches are performed by POWER switch. When power is applied to system (indicator DS28 lit), pressing POWER switch removes power from system
CPU RESET/CLEAR Pushbutton	Pressing switch with COMPUTE switch at IDLE, establishes the following initial conditions within CPU: a. All interrupts except power on and power off are disarmed and disabled b. MEMORY FAULT and ALARM indicators are turned off c. WRITE KEY, INTRPT INHIBIT, POINTER, CONDITION CODE, FLOAT MODE, MODE and TRAP indicators are turned off d. INSTRUCTION ADDRESS indicators are set to X'25' e. DISPLAY indicators are set to X'02000000', which is Load Conditions and Floating Control Immediate instruction with R-field of zero, to produce No Operation instruction Setting CPU to initial conditions by pressing CPU RESET/CLEAR switch does not affect any current input/output operation that may be in progress
I/O RESET	With COMPUTE switch at IDLE, resets all peripheral devices under control of CPU to "ready" condition. Resets all status, interrupt, and control indicators in I/O system. Does not affect current operation of CPU
SYSTEM RESET/CLEAR Pushbutton	With COMPUTE switch at IDLE, pressing SYSTEM RESET/CLEAR switch performs all operations described for the CPU RESET/CLEAR switch and establishes initial conditions in standard I/O system. All peripheral devices under control of the CPU are halted, and all status and control indicators in I/O system are reset. MEMORY FAULT indicators are turned off, and control signals in the memory are reset to their initial states. SYSTEM RESET/CLEAR switch and CPU RESET/CLEAR switch are interlocked so that pressing both switches simultaneously clears the core memory to zeros. Switches do not affect private memory registers
LOAD Pushbutton	Pressing LOAD switch loads a bootstrap program into locations X'20' through X'29' in core memory so that an input operation may be performed using a peripheral input device selected by UNIT ADDRESS switches
UNIT ADDRESS Thumbwheel Switches	Three UNIT ADDRESS switches select the peripheral unit to be used in the loading process. Unit addresses are hexadecimally notated. The first thumbwheel switch is for IOP addressing (0-7). The second and third thumbwheels are for device controller and device addressing (X'00' - X'FF)
NORMAL MODE Indicator	Indicator lights when all of the following conditions are satisfied: a. WATCHDOG TIMER switch is set to NORMAL b. INTERLEAVE SELECT switch is set to NORMAL

(Continued)

Table 2-1. Controls and Indicators, PCP Programmer's Section (Cont.)

Control or Indicator	Function
	c. PARITY ERROR MODE switch is set to CONT (continue) d. CLOCK MODE switch is set to CONT (continuous) e. All CPU, memory, and DC logic power margins are normal
RUN Indicator	Indicator lights when COMPUTE switch is set to RUN and no halt condition exists
WAIT Indicator	Indicator lights when any of the following conditions exist: a. CPU is executing Wait instruction b. Program is stopped because ADDRESS STOP switch is on and SELECT ADDRESS location is addressed c. CPU has attempted to execute instruction in interrupt location other than Exchange Program Status Doubleword or Modify and Test instruction
INTERRUPT Pushbutton	Switch is used by operator to activate control panel interrupt. If PCP interrupt level is armed, a single pulse is transmitted to interrupt level, and it is advanced to waiting state. INTERRUPT switch lights when this interrupt level is in waiting state and remains lit until interrupt level advances to active state
WRITE KEY Indicators	Two indicators, part of program status word 2 (PSW2), are used to control write access in areas of memory when memory protection option is used
INTRPT INHIBIT Indicators, CTR, I/O, EXT	Three indicators, part of program status word 2 (PSW2), are used to designate which groups of interrupts are allowed or are inhibited: the counter (CTR), the input/output (I/O), or the external group (EXT). When an indicator is on, the associated interrupt group is inhibited
POINTER Indicators	Five indicators, part of PSW2, are used to represent current status of register pointer in CPU
CONDITION CODE Indicators	Four CONDITION CODE indicators, part of program status word 1 (PSW1), are used to indicate nature of results of instruction after instruction execution. They are lighted if corresponding condition code flip-flop is set
FLOAT MODE SIG ZERO NRMZ Indicators	Three indicators, part of PSW1, represent current control modes for floating point operations: significance, zero, and normalize. If SIG is on, the computer traps to location X'44' when more than two hexadecimal places of postnormalization shifting are required, or if the result is zero. If ZERO is on, underflow causes the computer to trap to location X'44'. If NRMZ is on, postnormalization of the results of additions or subtractions is inhibited
MODE SLAVE MAP Indicators	Two indicators, part of PSW1, represent current mode of operation of CPU: whether it is in slave or master mode and whether map option is in effect when available
TRAP DEC ARITH Indicators	Indicator DEC, when lighted, indicates that trap is in effect with certain decimal operations if decimal option is present. Indicator ARITH, when lighted, indicates that trap is in effect with certain fixed-point arithmetic operations. These two indicators are part of PSW1

(Continued)

Table 2-1. Controls and Indicators, PCP Programmer's Section (Cont.)

Control or Indicator	Function
INSTRUCTION ADDRESS Indicators	Seventeen indicators, part of PSW1, represent current contents of Q-register in CPU. Address displayed in this field is address of next instruction
DISPLAY Indicators	Thirty-two indicators are used to display contents of memory word when used with INSTR ADDR, STORE and DISPLAY (INSTR ADDR and SELECT ADDR) and DATA switches. DISPLAY indicators normally represent current contents of internal CPU sum bus. When moving COMPUTE switch from IDLE to RUN or STEP, the first instruction executed is the one displayed in the DISPLAY indicators
DATA Switches	Thirty-two DATA switches are used to change contents of program status doubleword when used with INSERT switch and to alter value of DISPLAY indicators when used with DATA CLEAR/ENTER switch. Each DATA switch is inactive in center position and is latching in both upper (1) and lower (0) positions. In center position, DATA switch represents no change. In upper or lower position, each switch represents 1 or 0, respectively
INSERT Switch	Used manually to make changes in program status doubleword. Switch is inactive in center position and is momentary in upper (PSW2) and lower (PSW1) positions. When switch is moved to either PSW1 or PSW2, corresponding portion of program status doubleword is altered according to current state of DATA switches
STORE Switch	Used to change contents of either general register or memory location. Switch is inactive in center position and is momentary in INSTR ADDR and SELECT ADDR positions. When switch is moved to INSTR ADDR, current value of DISPLAY indicators is stored in location shown by INSTRUCTION ADDRESS indicators. When switch is moved to SELECT ADDR, current value of DISPLAY indicators is stored in location shown by SELECT ADDRESS switches
DATA Switch	Single DATA switch is used to change state of DISPLAY indicators. Switch is not active in center position and is momentary in CLEAR and ENTER positions. When switch is moved to CLEAR, all DISPLAY indicators are turned off. When switch is moved to ENTER, DISPLAY indicators are altered according to state of 32 DATA switches
INSTR ADDR Switch	INSTR ADDR (instruction address) switch is inactive in center position. Upper position (HOLD) is latching, and lower position (INCREMENT) is momentary. When switch is placed in HOLD, normal process of incrementing instruction address portion of program status doubleword with each instruction is inhibited. If COMPUTE switch is placed in RUN while INSTR ADDR switch is at HOLD, instruction in location displayed by INSTRUCTION ADDRESS indicators is executed repeatedly and INSTRUCTION ADDRESS indicators remain unchanged. However, if the instruction contains a branch or is a Load or Exchange Doubleword instruction, the specified instruction is executed. If COMPUTE switch is moved to STEP while INSTR ADDR switch is at HOLD, the instruction is executed every time the COMPUTE switch is moved to STEP, and INSTRUCTION ADDRESS indicators remain unchanged. The following operations are performed each time INSTR ADDR switch is moved from center position to INCREMENT: a. Current value of INSTRUCTION ADDRESS indicators is counted up by one b. Contents of virtual address displayed by INSTRUCTION ADDRESS indicators (subject to current memory map, if MAP mode indicator is lighted) are shown in DISPLAY indicators
DISPLAY Switch	Displays contents of either general register or memory location. Switch is inactive in center position and is momentary in both INSTR ADDR and SELECT

(Continued)

Table 2-1. Controls and Indicators, PCP Programmer's Section (Cont.)

Control or Indicator	Function
	ADDR positions. When switch is moved to INSTR ADDR or SELECT ADDR, contents of location shown by indicators or switches, respectively, appear in DISPLAY indicators
COMPUTE Switch	Controls execution of instructions. Center position (IDLE) and upper position (RUN) are both latching. Lower position (STEP) is momentary. When COMPUTE switch is at IDLE, all other control panel switches are operative and interrupts are disabled. When COMPUTE switch is moved from IDLE to RUN, RUN indicator lights, and CPU begins to execute instructions as follows: a. Current setting of DISPLAY indicators is taken as next instruction to be executed regardless of contents of location shown by current value of INSTRUCTION ADDRESS indicators b. Value in INSTRUCTION ADDRESS indicators is counted up by one c. Instruction execution continues with instruction in location shown by new value of INSTRUCTION ADDRESS indicators, unless a branch instruction is in effect When COMPUTE switch is in RUN, the only operative switches are POWER, INTERRUPT, ADDR STOP, INSTR ADDR (in HOLD position), and switches in maintenance section of control panel. Each time COMPUTE switch is moved from IDLE to STEP, the following operations occur: a. Current setting of DISPLAY indicators is taken as an instruction, and instruction is executed b. Current value of INSTRUCTION ADDRESS indicators is counted up by one. If stepped instruction was a branch instruction, and if a branch should occur, INSTRUCTION ADDRESS indicators are set to value of effective address of branch instruction c. Instruction in location shown by new value of INSTRUCTION ADDRESS indicators is displayed in DISPLAY indicators If instruction is being stepped (executed by moving COMPUTE switch from IDLE to STEP), all interrupt levels are inhibited during instruction execution. However, trap can occur, and the XPSD instruction in appropriate trap location is executed as if COMPUTE switch were in RUN. Thus, if trap occurs during stepped instruction, program status doubleword display (PSW1 and PSW2) automatically reflects the effects of the XPSD instruction in the trap location, and DISPLAY indicators then contain the first instruction of the trap routine
ADDR STOP Switch	ADDR STOP (address stop) switch causes CPU to halt whenever a core memory access is made to the location set in the value of INSTRUCTION ADDRESS indicators and the value set in SELECT ADDRESS switches. When halt occurs, WAIT indicator lights, and instruction in the location displayed by INSTRUCTION ADDRESS indicators appears in DISPLAY indicators. Instruction displayed is one that would be executed next if halt had not occurred. Address-stop halt is reset when COMPUTE switch is moved from RUN to IDLE. If COMPUTE switch is then moved back to RUN (or to STEP), instruction shown in DISPLAY indicators is next instruction executed. CPU continues until the addresses match again
SELECT ADDRESS Switch	Used with ADDR STOP switch to select virtual address where program is halted. Also used to select virtual address of location altered when used with STORE switch or to select virtual address of word displayed when used with DISPLAY switch

Table 2-2. Controls and Indicators, PCP Maintenance Section

Control or Indicator	Function
CONTROL MODE Switch	CONTROL MODE switch is a three-position key lock. When switch is in LOCAL, all controls and indicators on PCP are operative, and most controls on free-standing console are inoperative. However, indicators on the free-standing console (if attached) continue to display the same information as indicators in programmer's portion of PCP. POWER and INTERRUPT switches on free-standing console are operative when CONTROL MODE switch is in LOCAL When switch is in REMOTE, all controls on free-standing console are operative, and all indicators on PCP maintenance section are operative. The WATCHDOG TIMER, INTERLEAVE SELECT, PARITY ERROR MODE, and AUDIO switches on the maintenance section are operative. All indicators on programmer's portion of PCP continue to display the same information as those on free-standing console. However, most of the switches on programmer's section are inoperative, The POWER switch is operative When CONTROL MODE switch is in LOCK, all controls on free-standing console (except POWER, INTERRUPT and SENSE) are inoperative. Most controls on programmer's section of PCP are inoperative, although all indicators on both free-standing console and PCP continue to indicate various computer conditions. With CONTROL MODE switch in LOCK, the following switches on the PCP remain operative: POWER, INTERRUPT, AUDIO and all SENSE switches. The following switches are interlocked to the following states regardless of their actual settings when CONTROL MODE switch is in LOCK: a. COMPUTE switch to RUN b. WATCHDOG TIMER switch to NORMAL c. INTERLEAVE SELECT switch to NORMAL d. PARITY ERROR MODE switch to CONT e. CLOCK MODE switch to CONT
MEMORY FAULT Indicators	Since the system is limited to no more than eight memory banks, each MEMORY FAULT indicator corresponds to a specific memory bank. Whenever a memory parity error occurs in a memory bank, the appropriate indicator lights and remains lighted until the indicator is reset. MEMORY FAULT indicators can be turned off by pressing CPU RESET/CLEAR or SYS RESET/CLEAR switches or by Read Direct instruction coded to read MEMORY FAULT indicators
ALARM Switch and Indicator	Indicator is used to attract operator's attention to some urgent operating condition and is turned on and off under program control by executing properly coded Write Direct instruction. When ALARM indicator is lighted and AUDIO switch is ON, a 1-kHz signal is sent to PCP speaker. Speaker is disconnected when AUDIO switch is not set to ON. AUDIO switch does not affect state of ALARM indicator. ALARM indicator is turned off whenever CPU RESET/CLEAR or SYS RESET/CLEAR switches are pressed
PREPARATION PHASES Indicators	Three indicators display in binary notation one of four CPU phases during preparation portion of instruction cycle
PCP PHASES Indicators	Three PCP indicators display CPU phases during processor control panel operations. These phases, ranging from PCP1 to PCP7, are displayed in binary notation

(Continued)

Table 2-2. Controls and Indicators, PCP Maintenance Section (Cont.)

Control or Indicator	Function
EXECUTION PHASES Indicators	Four EXECUTION indicators display CPU phases during execution portion of instruction cycle. These phases, ranging from PH1 to PH15, are displayed in binary notation
INT/TRAP PHASES Indicators	Two indicators display four interrupt/trap phases during interrupt or trap operation
WATCHDOG TIMER Switch	When WATCHDOG TIMER switch is in OVERRIDE, watchdog timer is inoperative. When switch is in NORMAL, watchdog timer is operative. CPU can be interrupted at end of each instruction and at certain points during execution of some instructions. An interval of not more than 40 μs may occur between any two interruptible points. Watchdog timer is reset at each interruptible point and counts at a rate of 1-MHz between interruptible points. If count in watchdog timer reaches 40 μs,CPU traps to location X'46'
INTERLEAVE SELECT Switch	When switch is in NORMAL, interleaving between memory banks can be in effect according to the memory internal switches. When switch is at DIAGNOSTIC, memory addresses may not be interleaved between memory banks
PARITY ERROR MODE Switch	Controls action of CPU if a memory error occurs when the CPU is accessing. If switch is at CONT (continue) when parity error occurs, appropriate MEMORY FAULT indicator lights, and an interrupt signal is transmitted to memory parity interrupt level. If switch is at HALT when a parity error occurs, appropriate MEMORY FAULT indicator lights and CPU halts operation. Memory bank in which error has occurred is not available until its MEMORY FAULT indicator is reset or the PARITY ERROR MODE switch is set to CONT. In order to proceed, COMPUTE switch must be set to IDLE, CPU RESET or SYS RESET switches must be pressed, and COMPUTE switch then set to STEP or to RUN
SENSE Switches	Switches are used, under program control, to set the condition code portion of program status doubleword. When Read Direct or Write Direct instruction is executed in internal control mode, condition code is set according to state of four SENSE switches. SENSE switches are always operative and are normally used in this manner during diagnostic or other test routines
CLOCK MODE Switch	Controls internal CPU clock. When switch is at CONT (continue), clock operates at normal speed. When switch is in inactive (center) position, CPU clocks are not generated. Under these circumstances a single clock is generated each time CLOCK MODE switch is moved to SINGLE STEP position. As clock is stepped manually, PHASE indicators reflect CPU phase
REGISTER DISPLAY Switch	Normally, REGISTER DISPLAY switch is not at ON. When switch is at ON, contents of register selected by REGISTER SELECT switch are displayed in DISPLAY indicators if the CLOCK MODE switch is in the center position
REGISTER SELECT Switch	Used to display contents of selected internal registers. When CLOCK MODE switch is in center position, register selected by REGISTER SELECT switch may be shown in DISPLAY indicators by moving REGISTER DISPLAY switch to ON position. When REGISTER DISPLAY switch is returned to inactive position, DISPLAY indicators return to state previous to display of selected register

b. Place the address of the memory location into which the data is to be stored in the SELECT ADDRESS switches.

c. If the entire contents of the memory word are to be changed, set the single DATA ENTER/CLEAR switch to CLEAR. This resets the DISPLAY indicators.

d. If the DISPLAY indicators have been cleared, set the DATA switches to one wherever a one is to be stored in the memory word. If only a part of the memory word is to be changed, set the DISPLAY switch to SELECT ADDR. The selected memory word appears in the DISPLAY indicators. Set the DATA switches to the center position wherever a bit is to remain unchanged. Where a one is to be changed to a zero, set the corresponding DATA switch to 0; where a zero is to be changed to a one, set the corresponding DATA switch to 1.

e. Set the single DATA switch to ENTER and then release. The DISPLAY indicators contain the information to be stored.

f. Set the STORE switch momentarily to the SELECT ADDR position and release. The data in the DISPLAY indicators are stored in the selected memory location. The current instruction must be returned to the DISPLAY indicators by momentarily setting the DISPLAY switch to INSTR ADDR before program resumption by setting the COMPUTE switch to RUN or STEP.

2-11 INSTRUCTION ADDRESS. To store new data in the contents of the current instruction address, perform the following steps:

a. Set the COMPUTE switch to IDLE. The PHASE display indicates that the CPU is in phase PCP2.

b. Set the single DATA switch to CLEAR. This resets the display indicators.

c. Place the binary information to be stored in the instruction address into the DATA switches.

d. Set the single DATA switch to ENTER and then release. The DISPLAY indicators then contain the same information as the DATA switches.

e. Set the STORE switch momentarily to the INSTR ADDR position and release. The data in the DISPLAY indicators are stored in the memory location indicated by the INSTRUCTION ADDRESS indicators. If the program is resumed by setting the COMPUTE switch to RUN or STEP, the new instruction set in the DISPLAY indicators is executed as the current instruction, followed by the instructions in successive memory locations.

If the memory map option is included and the MAP MODE indicator is lighted, the memory map is in effect as the data is stored. Memory protection, if included, is inhibited in this PCP operation. To store data into a private memory register bank other than the one currently displayed by the POINTER field of PSW2, the contents of the POINTER field must be changed to point to the desired register bank. This operation is described in the next paragraph.

2-12 Changing the Current Program Status Doubleword

Changing any data in the current program status doubleword (PSD) requires that PSW1 and PSW2 be treated separately and that the following steps be performed:

a. Set the COMPUTE switch to IDLE.

b. Enter the desired information into the 32 DATA switches in the bit positions of PSW1 or PSW2 being changed. In bit positions where no changes are made, the corresponding DATA switches must be in the center (no change) position. Be sure to set DATA switches to 0 where zeros are desired.

c. Set the INSERT switch to PSW1 or to PSW2, depending on the portion of the PSD that is being changed.

d. Release the INSERT switch. The new information is entered in the appropriate word of the program status doubleword.

2-13 Manual Branching

Perform the following steps to cause the CPU to branch to any instruction in memory, regardless of the instruction that is currently being executed.

a. Set the COMPUTE switch to IDLE.

b. Enter the address of the instruction to which the CPU is to branch in the 17 least significant DATA switches which correspond to the INSTRUCTION ADDRESS field of PSW1. Set the DATA switches to 0 where zeros are desired.

c. Set the INSERT switch to PSW1 and release. The INSTRUCTION ADDRESS indicators display the address set in the DATA switches. The DISPLAY indicators display the instruction ready for execution when the CPU went into the idle state.

d. Set the DISPLAY switch to INST ADDR and release. The contents of the location in INSTRUCTION ADDRESS indicators appear in the DISPLAY indicators. The INSTRUCTION ADDRESS indicators continue to display the address set in the DATA switches. The instruction read from the location set in the data switches is the next instruction executed by the CPU.

e. If the COMPUTE switch is set to RUN, the CPU executes the instruction in the address in the INSTRUCTION ADDRESS indicators, then continues from the point in memory, executing instructions in successive memory locations.

2-14 Stepping Through a Program

It is often necessary when debugging programs or when maintaining the equipment to sequence slowly through the program one instruction at a time and to observe the results of each instruction after it has been executed. This is done by performing the following steps:

a. Set the COMPUTE switch to IDLE, and use the manual branching procedure to branch to the part of the program from which stepping is required. See paragraph 2-13.

b. Set the COMPUTE switch to STEP. The instruction in the branch location is executed. When the switch is released, the following instruction is displayed in the DISPLAY and the INSTRUCTION ADDRESS indicators.

c. The results of the executed instruction can be seen by displaying the contents of the memory location or private memory register affected by the instruction. See paragraph 2-7.

d. Repeat steps b and c to continue the program sequence step by step.

2-15 Single Clocking an Instruction

During maintenance operations, it is often necessary to sequence through individual instructions from one clock period to the next, observing the results in the CPU internal registers after each clock pulse. This technique can often be used to pinpoint a malfunctioning logic gate or an active circuit element without observing the action on an oscilloscope. However, this operation is not usually of value until after the malfunctioning instruction has been located. To single-clock instructions, perform the following steps:

a. Branch to the malfunctioning instruction, using the manual branching procedure in paragraph 2-13 or store an identical instruction in memory (paragraph 2-10) and branch to this location.

b. Set the CLOCK MODE switch to the center position. (This inhibits all clock pulses.) Set the COMPUTE switch to RUN or STEP.

c. Set the CLOCK MODE switch to SINGLE STEP. This causes the CPU to go to phase PCP3. PCP4 and PCP5 enter with further single clocking, then phase PRE1 of the instruction.

d. Observe the contents of the affected internal registers by setting the REGISTER SELECT switch to the proper register position and by setting the REGISTER DISPLAY switch to ON.

e. After all affected internal registers have been observed and if no malfunction is seen, repeat steps c and d.

In most single-clock operations, the INSTR ADDR switch can be placed in the HOLD position if the single-clock operation is to be repeated more than once.

2-16 Single Instruction Repetition

Single clocking a malfunctioning instruction (paragraph 2-15) may pinpoint the area of the malfunction, but the observer may be unable to determine what is causing the faulty condition. In some cases, an error may consistently occur while the CLOCK MODE switch is in the CONT (continuous) position, but may never occur when the switch is in the SINGLE STEP position. This could be caused by a slow gate or by a slow active circuit element. In such a case, the operator should repeatedly run the single malfunctioning instruction using the oscilloscope to observe all signals that could be the cause of the error condition.

To run a single instruction repeatedly, perform the following steps:

a. Branch to the malfunctioning instruction. (See paragraph 2-13.)

b. Set the INSTR ADDR switch to HOLD. This prevents the instruction address field of PSW1 from changing after each instruction execution.

c. Set the COMPUTE switch to RUN and observe all pertinent signals on the oscilloscope as the instruction is repeatedly executed.

Certain instructions (for example, multiply and divide instructions which change an operand each time the instruction is executed) cannot be repeated in this manner without destroying data meaningful to the observer. For this type of instruction, it may be necessary to enter a small four- or five-word instruction program loop to establish initial conditions each time the instruction is observed.

2-17 Loading Program

After the input device has been loaded with the program tape or cards and has been properly prepared to read, the following steps should be completed to load the program into memory:

a. Set the COMPUTE switch to IDLE.

b. Press the SYS RESET/CLEAR switch.

c. Set the UNIT ADDRESS switches to the address of the desired input peripheral device.

d. Press the LOAD switch.

e. Set the COMPUTE switch to RUN. The CPU reads the program from the input device and stores it in memory.

2-18 PROGRAMMING

2-19 BASIC INSTRUCTIONS

Table 2-3 lists all of the basic operation codes, options included. Operation codes and corresponding mnemonics are given in table 2-4. This table also indicates optional, privileged, or nonexistent instructions. For detailed operation of each instruction see the Sigma 7 Reference Manual, XDS publication No. 900950, or refer to the operation code descriptions in section III of this manual.

2-20 COMPUTER MODES

The Sigma 7 computer operates in either the master mode or the slave mode. The mode of operation is determined by the slave mode bit in program status doubleword 1. When this bit is set, the computer is in the slave mode. When the bit is reset, the program is in the master mode. The mode control bit can be changed when the computer is in the master mode and when a RESET signal is generated. The bit can also be changed when a trap or interrupt takes the program to an interrupt or trap location that contains a Load or Exchange Program Status Doubleword instruction. Therefore, a slave program cannot directly change the computer from slave to master mode.

In the master mode, any legal Sigma 7 instruction may be executed. In the slave mode, privileged instructions are prohibited. The following are privileged instructions:

a. Load Program Status Doubleword

b. Exchange Program Status Doubleword

c. Load Register Pointer

d. Move to Memory Control

e. Wait

f. Read Direct

g. Write Direct

h. Start Input/Output

i. Halt Input/Output

j. Test Input/Output

k. Test Device

l. Acknowledge Input/Output Interrupt

An executive program, which is stored in memory and operates in the master mode, controls and supports the operation of other programs in the master or slave mode. Control returns to the executive program if any attempt is made by a program to execute a privileged instruction while the computer is in the slave mode. However, the slave program can gain direct access to certain executive program operations without the use of the executive program by means of Call instructions. The operations available through the Call instructions are established by the executive program.

Table 2-3. Basic Instructions

Mnemonic	Code	Instruction Name
LOAD-STORE		
LI	22	Load Immediate
LB	72	Load Byte
LH	52	Load Halfword
LW	32	Load Word
LD	12	Load Doubleword
LCH	5A	Load Complement Halfword
LAH	5B	Load Absolute Halfword
LCW	3A	Load Complement Word
LAW	3B	Load Absolute Word
LCD	1A	Load Complement Doubleword
LAD	1B	Load Absolute Doubleword
LS	4A	Load Selective
LM	2A	Load Multiple
LCFI	02	Load Conditions and Floating Control Immediate
LCF	70	Load Conditions and Floating Control
XW	46	Exchange Word
STB	75	Store Byte
STH	55	Store Halfword
STW	35	Store Word
STD	15	Store Doubleword

(Continued)

Table 2-3. Basic Instructions (Cont.)

Mnemonic	Code	Instruction Name
LOAD-STORE (Cont.)		
STS	47	Store Selective
STM	2B	Store Multiple
STCF	74	Store Conditions and Floating Control
ANALYZE-INTERPRET		
ANLZ	44	Analyze
INT	6B	Interpret
LOGICAL		
OR	49	OR Word
EOR	48	Exclusive OR Word
AND	4B	AND Word
CONVERSION		
CVA	29	Convert by Addition
CVS	28	Convert by Subtraction
FLOATING POINT ARITHMETIC		
FAS	3D	Floating Add Short
FAL	1D	Floating Add Long
FSS	3C	Floating Subtract Short
FSL	1C	Floating Subtract Long
FMS	3F	Floating Multiply Short
FML	1F	Floating Multiply Long
FDS	3E	Floating Divide Short
FDL	1E	Floating Divide Long
DECIMAL		
DL	7E	Decimal Load
DST	7F	Decimal Store
DA	79	Decimal Add
DS	78	Decimal Subtract
DM	7B	Decimal Multiply
DD	7A	Decimal Divide
DC	7D	Decimal Compare
DSA	7C	Decimal Shift Arithmetic
PACK	76	Pack Decimal Digits
UNPK	77	Unpack Decimal Digits
FIXED-POINT ARITHMETIC		
AI	20	Add Immediate
AH	50	Add Halfword
AW	30	Add Word
AD	10	Add Doubleword
SH	58	Subtract Halfword
SW	38	Subtract Word
SD	18	Subtract Doubleword
MI	23	Multiply Immediate

(Continued)

Table 2-3. Basic Instructions (Cont.)

Mnemonic	Code	Instruction Name
FIXED-POINT ARITHMETIC (Cont.)		
MH	57	Multiply Halfword
MW	37	Multiply Word
DH	56	Divide Halfword
DW	36	Divide Word
AWM	66	Add Word to Memory
MTB	73	Modify and Test Byte
MTH	53	Modify and Test Halfword
MTW	33	Modify and Test Word
COMPARISON		
CI	21	Compare Immediate
CB	71	Compare Byte
CH	51	Compare Halfword
CW	31	Compare Word
CD	11	Compare Doubleword
CS	45	Compare Selective
CLR	39	Compare with Limits in Register
CLM	19	Compare with Limits in Memory
SHIFT		
S	25	Shift
SF	24	Shift Floating
BYTE STRING		
MBS	61	Move Byte String
CBS	60	Compare Byte String
TBS	41	Translate Byte String
TTBS	40	Translate and Test Byte String
EBS	63	Edit Byte String
PUSH-DOWN		
PSW	09	Push Word
PLW	60	Pull Word
PSM	0B	Push Multiple
PLM	0A	Pull Multiple
MSP	13	Modify Stack Pointer
EXECUTE-BRANCH		
EXU	67	Execute
BCS	69	Branch on Conditions Set
BCR	68	Branch on Conditions Reset
BIR	65	Branch on Incrementing Register
BDR	64	Branch on Decrementing Register
BAL	6A	Branch and Link
CALL		
CAL1	04	Call 1
CAL2	05	Call 2

(Continued)

Table 2-3. Basic Instructions (Cont.)

Mnemonic	Code	Instruction Name
CALL (Cont.)		
CAL3	06	Call 3
CAL4	07	Call 4
CONTROL		
LSPD	0E	Load Program Status Doubleword
XPSD	0F	Exchange Program Status Doubleword
LRP	2F	Load Register Pointer
MMC	6F	Move to Memory Control
WAIT	2E	Wait
RD	6C	Read Direct
WD	6D	Write Direct
INPUT/OUTPUT		
SIO	4C	Start Input/Output
HIO	4F	Halt Input/Output
TIO	4D	Test Input/Output
TDV	4E	Test Device
AIO	6E	Acknowledge Input/Output

Table 2-4. Basic Instruction Codes, Cross-Reference

LEAST SIGNIFICANT CHARACTER	MOST SIGNIFICANT CHARACTER							
	0	1	2	3	4	5	6	7
0	*	AD	AI	AW	TTBS	AH	CBS	LCF
1	*	CD	CI	CW	TBS	CH	MBS	CB
2	LCFI	LD	LI	LW	*	LH	*	LB
3	*	MSP	MI	MTW	*	MTH	EBS†	MTB
4	CAL1	*	SF	*	ANLZ	*	BDR	STCF
5	CAL2	STD	S	STW	CS	STH	BIR	STB
6	CAL3	*	*	DW	XW	DH	AWM	PACK†
7	CAL4	*	*	MW	STS	MH	EXU	UNPK†
8	PLW	SD	CVS	SW	EOR	SH	BCR	DS†
9	PSW	CLM	CVA	CLR	OR	*	BCS	DA†
A	PLM	LCD	LM	LCW	LS	LCH	BAL	DA†
B	PSM	LAD	STM	LAW	AND	LAH	INT	DD†
C	**	FSL†	**	FSS†	SIO**	*	RD**	DM†
D	**	FAL†	**	FAS†	TIO**	*	WD**	DSA†
E	LPSD**	FDL†	WAIT**	FDS†	TDV**	*	AIO**	DC†
F	XPSD**	FML†	LRP**	FMS†	HIO**	*	MMC**	DST†

*Nonexistent instruction
†Optional instruction
**Privileged instruction

SECTION III
PRINCIPLES OF OPERATION

3-1 INTRODUCTION

This section describes the principles of operation of the Sigma 7 computer system on a general information, or block diagram, level and on a detailed information, or circuit diagram, level. The general principles describe the overall operation of the computer equipment, beginning with the main functions and proceeding logically through the subfunctions. The detailed principles describe the operation of each physical or functional unit of the equipment in terms of logic elements.

3-2 GENERAL PRINCIPLES

The computer system is organized around one or more central processing units, magnetic core memories, and input/output processors, device controllers, and devices such as card readers and punches, magnetic tape units, and printers. Two types of input/output processors are available: a multiplexing input/output processor, which allows up to 32 peripheral devices to operate simultaneously, and a selector input/output processor, which allows one device at a time to operate at a high-speed transfer rate. The major system elements are shown in figure 3-1. Each of the illustrated system elements performs asynchronously with respect to the other elements.

3-3 CORE MEMORY

The core memory consists of from one to eight memory banks, each containing from 4K to 16K of memory. Thus, a maximum memory system may contain 128K of memory storage. Memory banks are connected in parallel to the central processing unit as shown in figure 3-2.

Each memory bank in the system contains three ports, designated A, B, and C. A port is a section of memory logic that controls entry priorities during memory access. Each port may be connected to a memory bus. A controlling central processing unit must be connected to port C to provide auxiliary signals not present on the other ports. Port A and Port B may be connected to an input/output processor or to another CPU. Port A has the highest priority and port C, the lowest priority.

Either port A or port B may be replaced by a three- to six-port expansion unit, which expands one port to four ports. The three available ports on one memory bank are thereby increased to six. The four ports on the port expansion unit, designated ports 1, 2, 3, and 4, are assigned priorities in numerical order, port 1 having the highest priority and port 4 the lowest.

The memory is organized in banks as shown in figure 3-3.

A memory bank contains from one to four memory stacks plugged into one memory frame; hence, a memory bank may contain 4K, 8K, 12K, or 16K words.

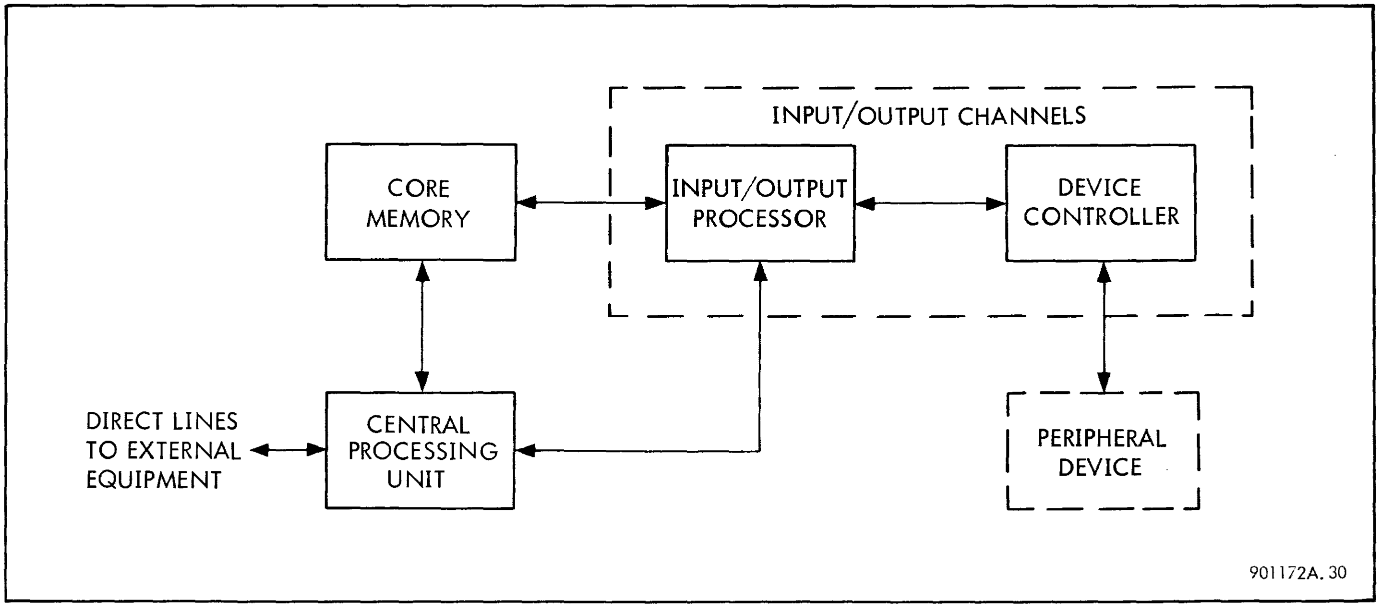


Figure 3-1. Sigma 7 System Elements

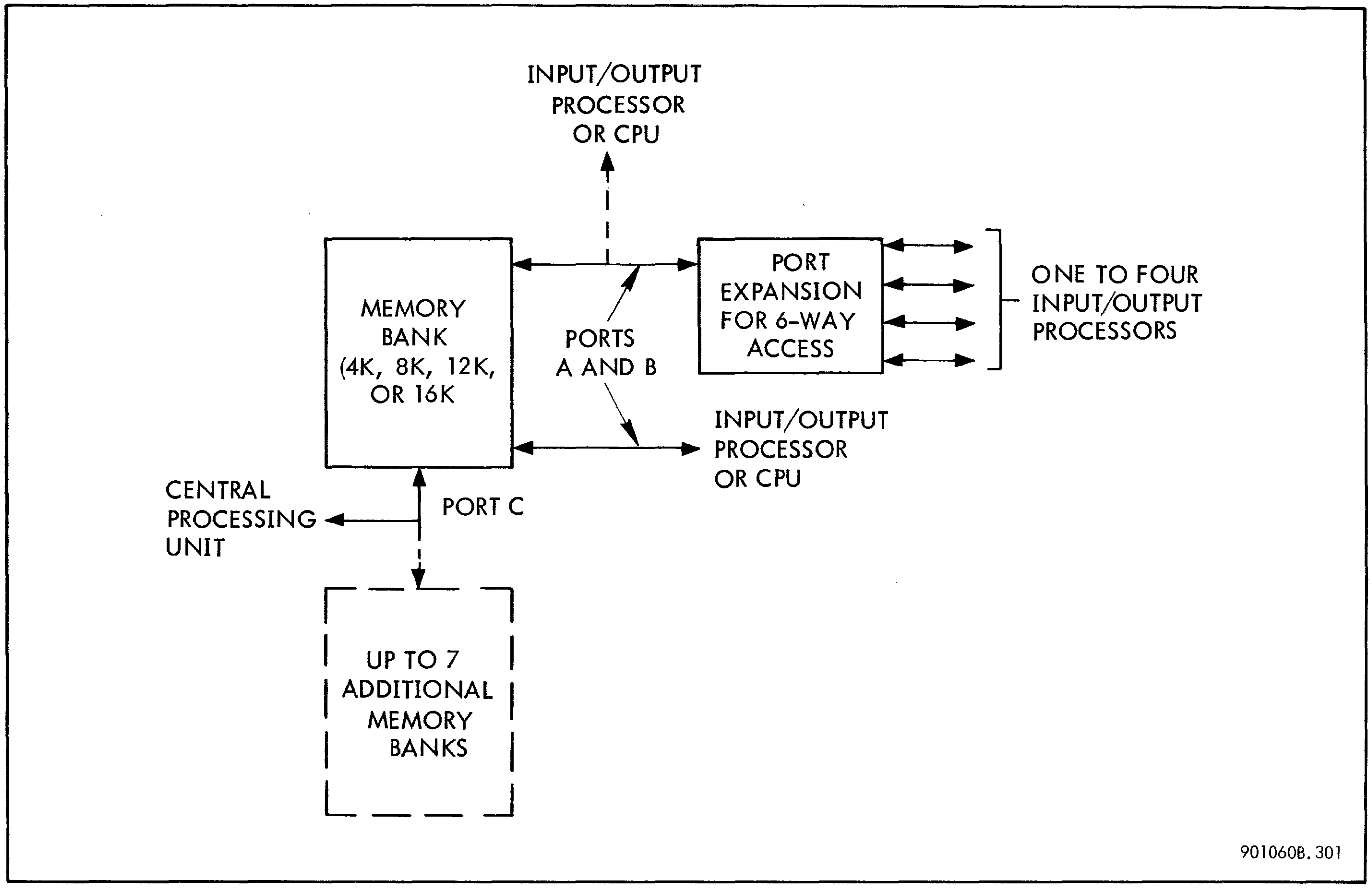


Figure 3-2. Memory Connection

3-4 Three-Wire Core Selection

The Sigma 7 combines the three-dimensional coincident-current selection method with the two-dimensional linear selection method and is commonly known as the 2-1/2D system. The system has a coincident-current read cycle and a linear-select write cycle. Three wires are threaded through each core: an X wire (word wire), a Y wire, and a sense wire. No inhibit winding is necessary.

On each bit plane there are 128 X wires. Each X wire also threads all other bit planes on a core module. A bit plane contains 16 Y wires, which are separate for each bit plane. Each Y wire doubles back through a second row of cores to provide 32 Y wires in all. Typical X and Y wiring for two cores in each of two bit planes on a memory module is shown in figure 3-4. A sense wire threads through all of the cores on one bit plane.

When writing ones, half current is passed through one word wire, and half current is passed through the proper Y wire to select one out of 4096 cores. The addition of the two half-currents at the intersection forces the core to the logical one state. To write zeros, the Y current is inhibited on the bit planes where zero bits are desired. This is similar to the linear-select method because the digit current is added to, rather than subtracted from, the word current.

When reading data, half current is passed through the appropriate X wire, and half current is passed through the same Y wire on all bit planes. All cores in the selected word location are forced to zero, and the sense wires detect current from the bit planes that contained ones in the selected location.

As shown in figure 3-4, each Y wire on a bit plane passes through two rows of cores; hence, there are two core intersections for each combination of X and Y wires. For a given direction of current, the currents add in the core at one intersection and cancel in the other. Therefore, core selection is determined by current direction as well as by wire location.

3-5 Data Paths

Interchange of data between the CPU and memory takes place on the bidirectional 32-bit memory bus. Data entering memory from the CPU is gated onto the memory bus from the sum bus in the CPU and is placed in the memory data (M) register. Outputs from the M-register flip-flops control current flow in the Y wires, causing ones or zeros to

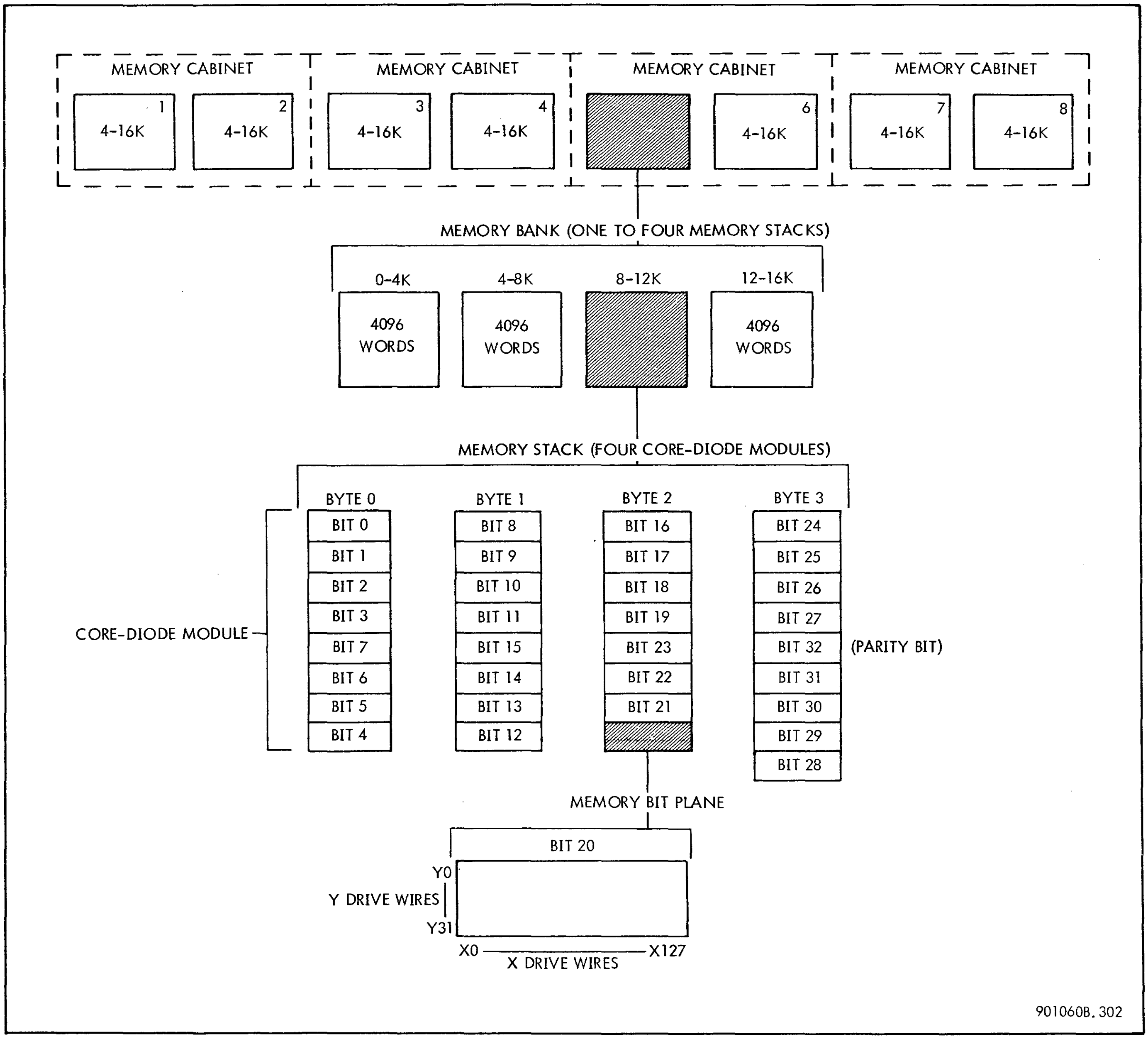


Figure 3-3. Memory Organization

appear in the corresponding core locations. During read operation, outputs from the sense amplifiers cause ones or zeros to be placed in the M-register flip-flops. This information is transmitted on the memory bus to the C-register in the CPU. A similar memory bus connects the memory with the IOP, and data is transferred in both directions between the M-register in memory and a data register in the IOP, also designated the M-register.

3-6 Memory Timing

Two delay lines are used in a memory bank to control the timing; one controls the read cycle and the other controls the write cycle. The delay lines have taps at each 20-ns interval. Both a read and a write cycle are required for each memory access. When a read cycle is used to read data from memory, it must be followed by a write cycle to restore the data to the same location. When a write cycle is used to enter data, it must be preceded by a read cycle to clear the location to zero.

Three types of memory access are used: read-restore, write full, and write partial. The latter is used when data is being entered only into 1, 2, or 3 bytes of the memory word. The timing is slightly different for each type of access.

The read cycle is the same in all three memory accesses. As a result of a memory request, a pulse is started down the read-delay line. Outputs from the delay line taps are used

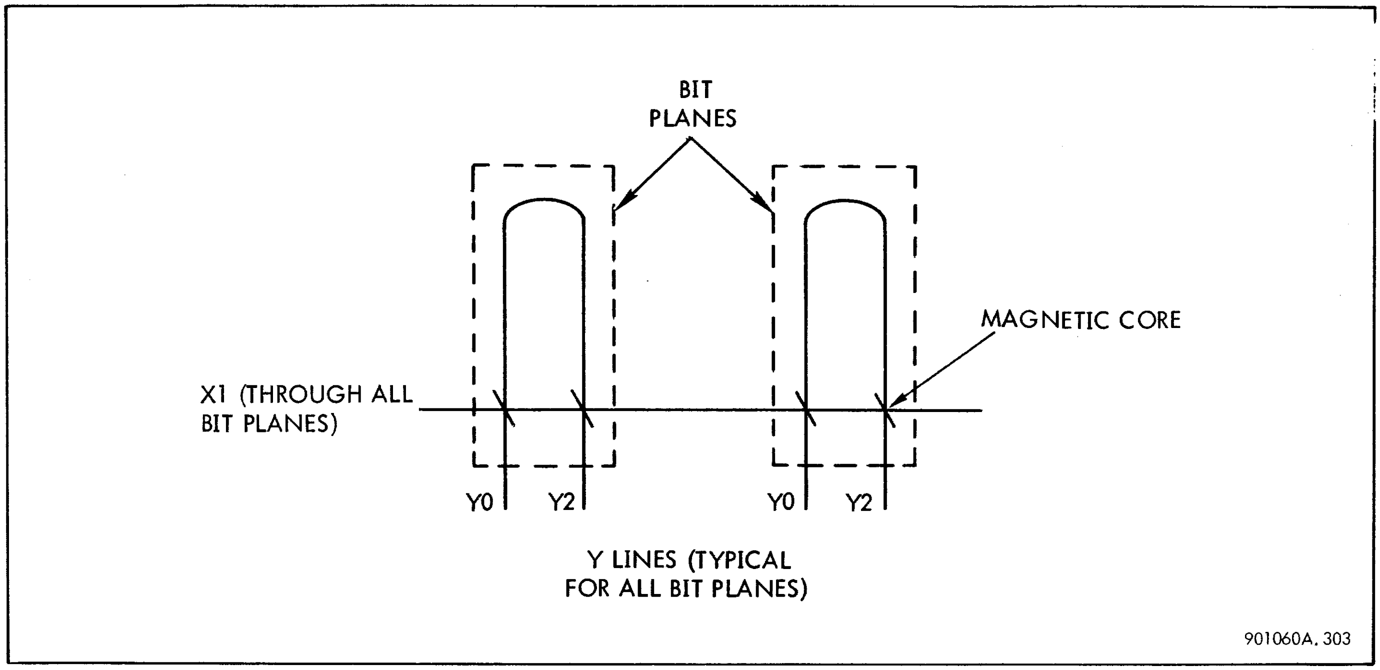

BIT PLANES
MAGNETIC CORE
X1 (THROUGH ALL BIT PLANES)
Y0
Y2
Y0
Y2
Y LINES (TYPICAL FOR ALL BIT PLANES)
901060A. 303

Figure 3-4. X and Y Core Winding, Typical

to control the timing signals to energize the X and Y drive lines, to set latches enabling data to enter the M-register, and to control the operations strobing data into the M-register. In the write-full operation, the read data is not gated into the M-register and is lost. In the write-partial operation, the read data is placed in the M-register, and parity is checked, but the data is not gated out of memory. Parity is also checked in the read-restore operation.

The write cycle is the same for read-restore and write-full operations. An output from the read-delay lines starts a pulse down the write-delay line. Outputs from the write-delay lines are used to send current through the X and Y drive lines in the opposite direction from that in read operation and to inhibit current in the Y lines on the bit planes where zeros are to be stored. Odd parity is generated in write-full operation by setting the parity flip-flop if the M-register contains an even number of ones.

Write partial timing is the same as for read-restore and write-full except that energizing the drive lines is delayed long enough to set byte indicators that steer the information into the desired byte locations.

3-7 Interleaving

Memory speed can be increased by interleaving or overlapping the second cycle of one access with the first cycle of the following access as shown for a read-restore cycle in figure 3-5. This can be done only if successive words are stored in different memory banks, because both the read and write cycles of one access must be completed before another access is started when addressing twice in one memory bank. As an example of interleaving, consider two 4K memory banks and a program that stores data in a series of sequential memory locations. The first word is stored in one 4K bank, the second word is stored in the same location in the second 4K bank, the third word is stored in the first 4K bank, and so forth. In larger memories with various bank sizes, the process is more complicated, but two successive words are never stored in the same memory bank.

3-8 CENTRAL PROCESSING UNIT

The basic central processing unit (CPU) consists of eight elements: private memory, arithmetic, control and address unit, processor control panel, internal interrupts, real-time clock, watchdog timer, clock-generation circuits, and power supplies (see figure 3-6).

3-9 Private Memory

The private memory is a bank of 16 integrated circuit flip-flop registers containing 32 bits each. These registers which are loaded and unloaded through the arithmetic and control unit of the central processing unit are not available to the input/output units. The private memory registers may be used as accumulators, temporary storage, or to contain information such as data addresses, counts, points, and so forth. Any or all of private memory registers 1 through 7 may be used as index registers.

3-10 Arithmetic and Control Section

The registers in the arithmetic and control unit with data flow paths are shown in block diagram form in figure 3-7. The paths that are used at any one time are determined by the instruction being executed.

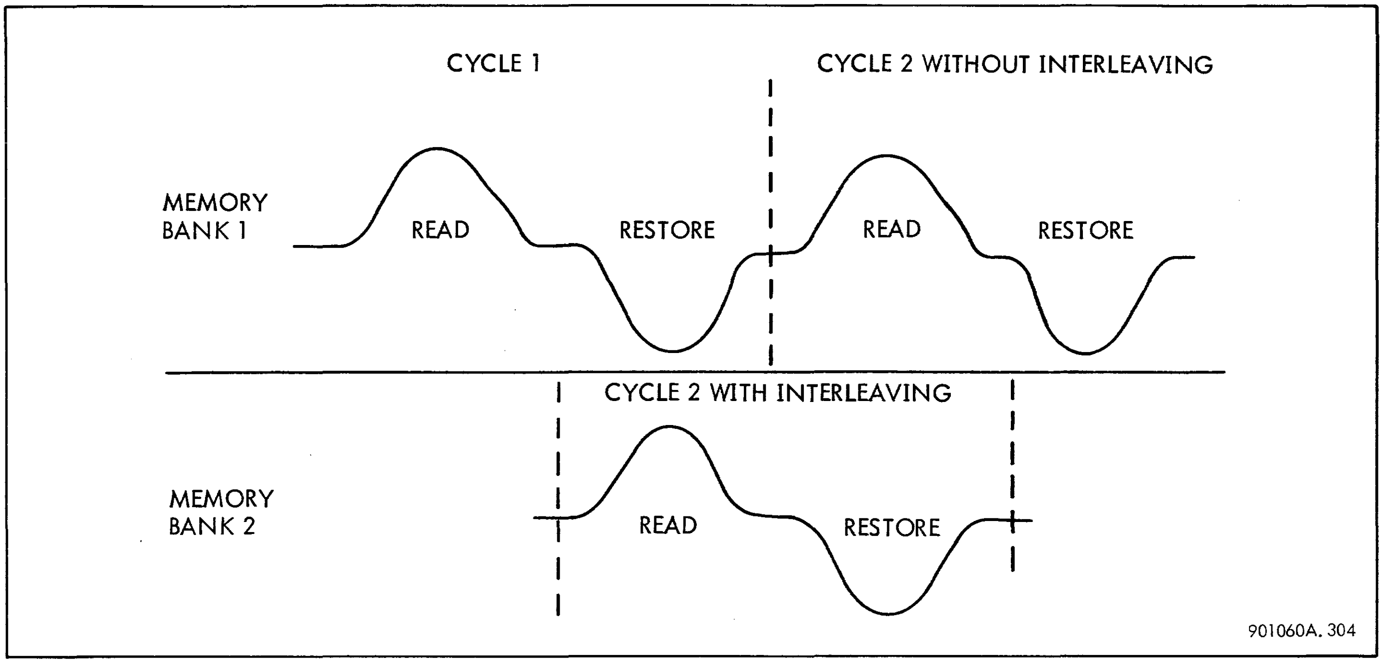


Figure 3-5. Interleaving Timing

The functions of the individual registers are as follows:

a. C-Register. Receives data and instructions from private and core memory. The instruction operation code is placed in the O-register for decoding, and the private memory address from the R-field of the instruction is placed in the R-register. All C-register information used in the adder or placed on the sum bus which is a 32-bit data path with multiple sources goes through the D-register. Data may be modified in the adder and returned to the C-register via this sum bus. Addresses from the address field in the C-register are placed on the core memory address lines if the address is greater than 15, or on the private memory address lines if the address is 15 or less.

b. O-Register. Receives operation code from the operation field in the C-register.

c. R-Register. Receives private memory address from R-field in C-register and places the address on private memory address lines. In some instructions, the R-register contains the number used to modify the memory contents.

d. D-Register. Receives all C-register information except addresses. During manual entry of data, the D-register receives information from the control panel DATA switches. The D-register receives condition codes from CC flip-flops and places data on the sum bus for memory storage and control panel display. It places the index register address on private memory address lines and supplies register pointer, RP, with a code indicating the private memory currently in use.

e. A-Register. Serves as one of the adder inputs. It receives direct inputs from private memory and receives inputs from the sum bus so that A-register information may be modified in the adder and returned to the A-register. The A-register places information on the sum bus for memory storage and for control panel display.

f. B-Register. Used for temporary storage and as an intermediate register in certain operations.

g. CS-Register. Stores the carry in some adder operations. This register is also used for complementing and for other arithmetic functions.

h. RP-Register. Receives the most significant five bits of the private memory address from the D-register. These bits indicate the private memory block being used. The RP-register places these bits on the private memory address lines.

i. Q-Register. Holds the address of the next instruction and is displayed in the INSTRUCTION ADDRESS indicators on the control panel. The Q-register receives the address from the P-register, and sends the address to the P-register to be increased by one for the next instruction access. The Q-register also supplies the address to the private memory and to the core memory address lines.

j. P-Register. Holds the current instruction address, obtained from the Q-register, the sum bus, or the control panel INSTRUCTION ADDRESS switches. It puts information on the sum bus for storage or display, and supplies the address to core memory address lines. The P-register adds one to the address received from the Q-register.

The control flip-flops in the arithmetic and control unir are shown in figure 3-8. These flip-flops, together with the

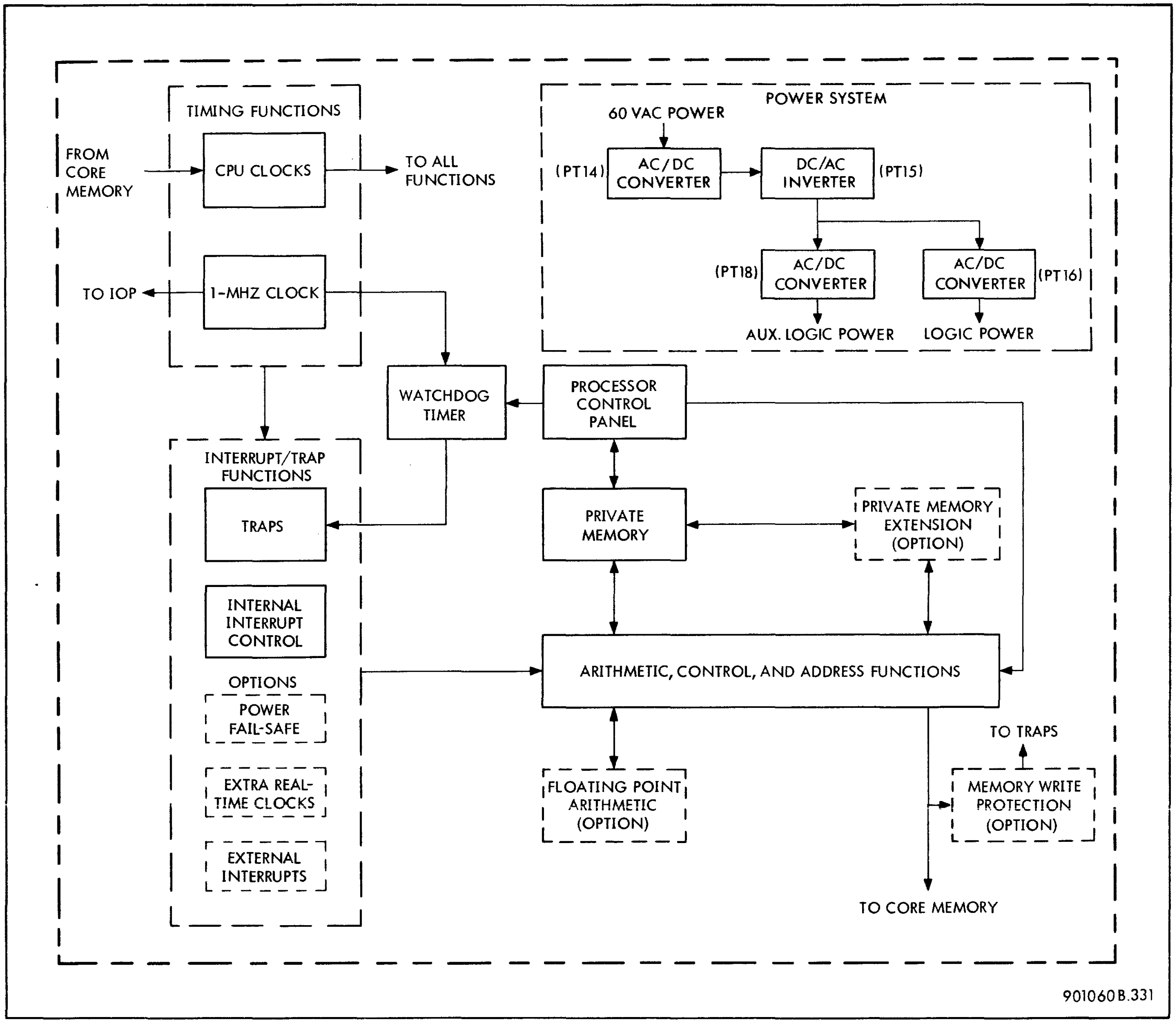


Figure 3-6. Central Processing Unit, Functional Block Diagram

Q-register and RP-register, are part of the program status doubleword and are displayed as such on the control panel. The outputs of the flip-flops are used throughout the computer logic to control modes and operations.

The control flip-flops, their functions and setting conditions, are as follows:

a. Flip-Flops CC1 Through CC4. Condition code flip-flops which indicate the nature of the results of an instruction. These flip-flops are set by various conditions at the end of an instruction, or may also be set by the control panel SENSE switches or the INSERT switch. When the INSERT switch is used, the information enters the CC flip-flops by way of S0 through S4 of the sum bus. The trap condition code flip-flops and signals from the decimal unit may also set CC1 through CC4.

b. Flip-Flops FS, FZ, and FN. Flip-flops used for control purposes in floating point instructions.

c. Flip-Flop MASTERF. When set, puts the computer in the slave mode. This flip-flop is set by bit 8 of the sum bus when a load or exchange program status doubleword instruction is being executed.

d. Flip-Flop MAPF. When set, puts the memory map in effect. It is set by bit 9 of the sum bus when a load or exchange program status doubleword instruction is being executed.

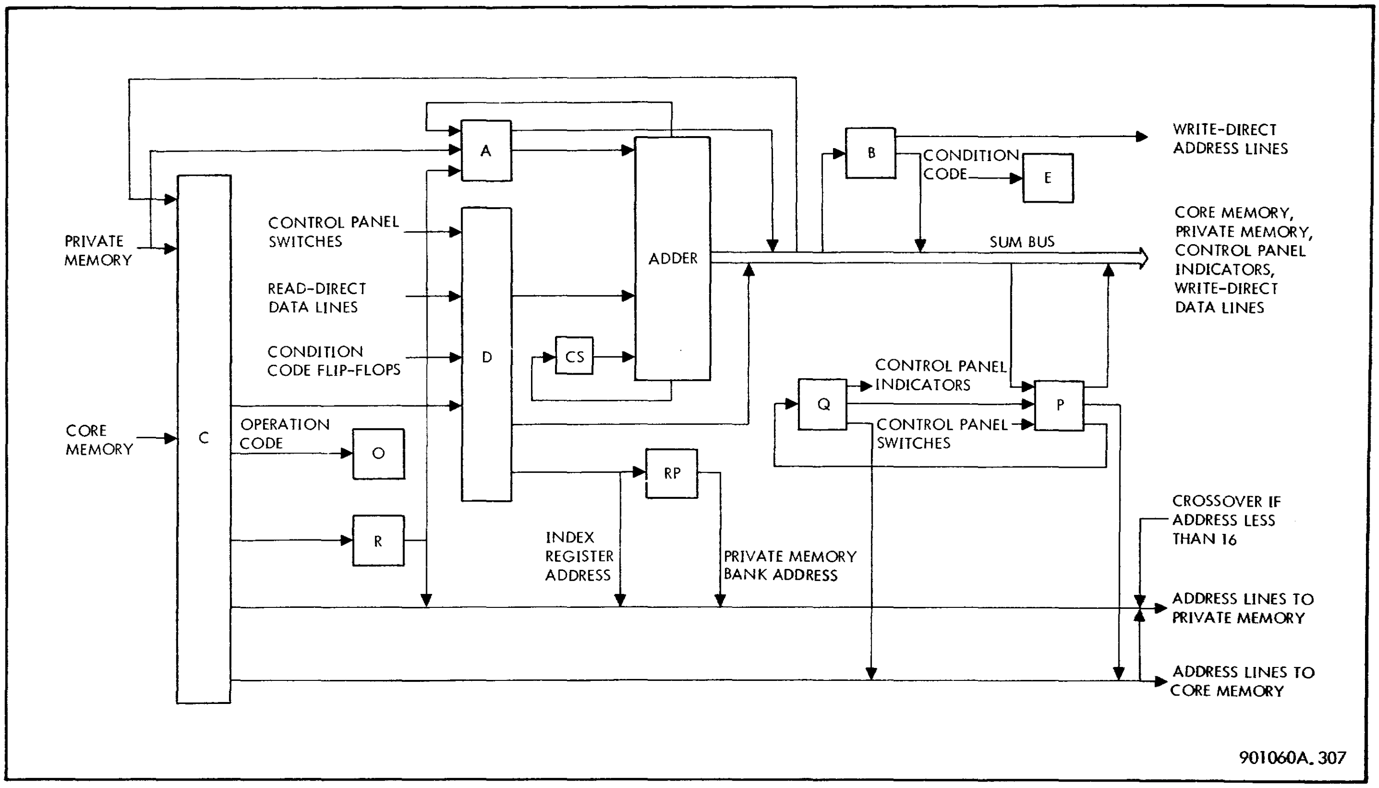

PRIVATE MEMORY
CORE MEMORY
C
CONTROL PANEL SWITCHES
READ-DIRECT DATA LINES
CONDITION CODE FLIP-FLOPS
OPERATION CODE
O
R
A
D
CS
ADDER
RP
INDEX REGISTER ADDRESS
PRIVATE MEMORY BANK ADDRESS
B
CONDITION CODE
E
SUM BUS
Q
CONTROL PANEL INDICATORS
CONTROL PANEL SWITCHES
P
WRITE-DIRECT ADDRESS LINES
CORE MEMORY, PRIVATE MEMORY, CONTROL PANEL INDICATORS, WRITE-DIRECT DATA LINES
CROSSOVER IF ADDRESS LESS THAN 16
ADDRESS LINES TO PRIVATE MEMORY
ADDRESS LINES TO CORE MEMORY
901060A. 307

Figure 3-7. Arithmetic and Control Unit Registers

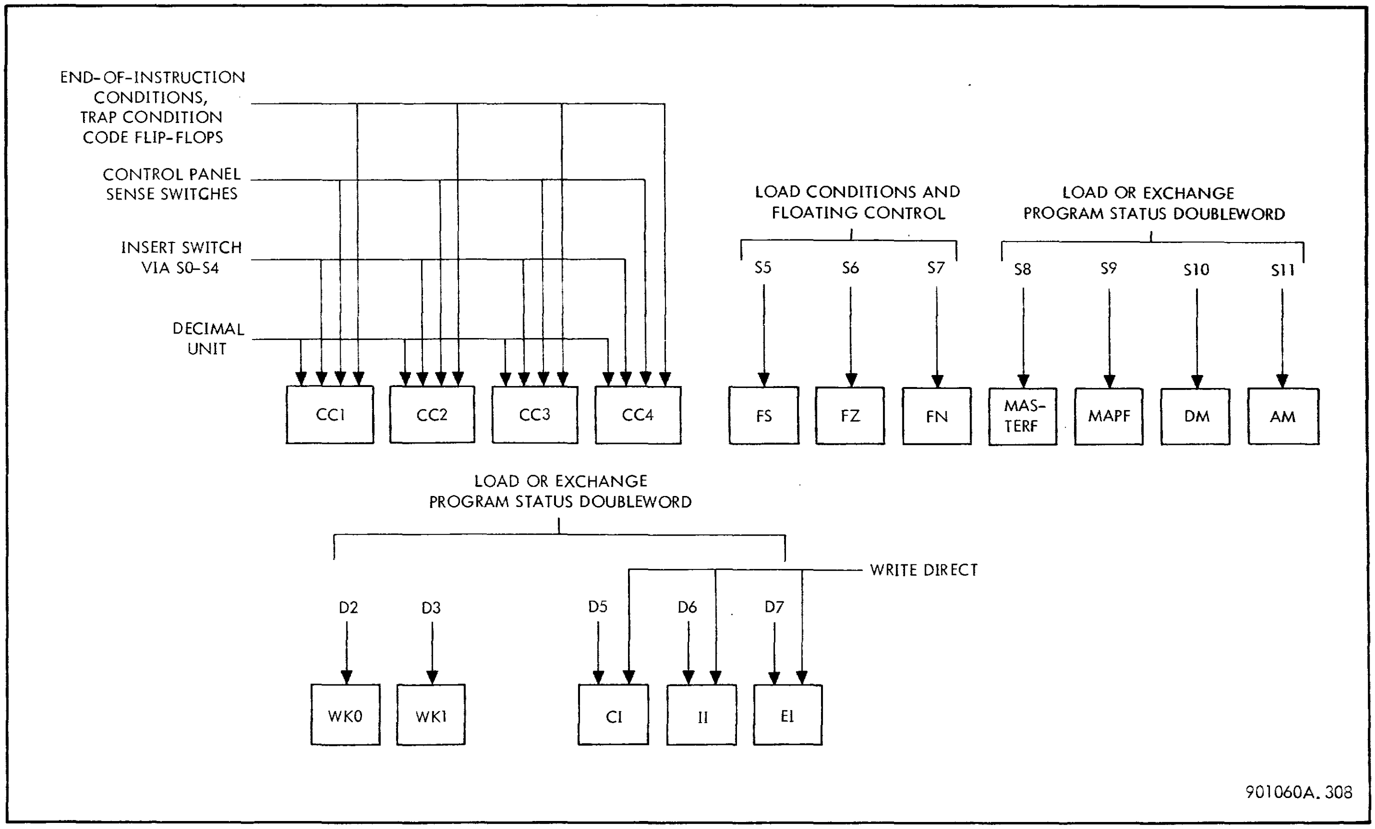

END-OF-INSTRUCTION CONDITIONS, TRAP CONDITION CODE FLIP-FLOPS
CONTROL PANEL SENSE SWITCHES
INSERT SWITCH VIA S0-S4
DECIMAL UNIT
CC1
CC2
CC3
CC4
LOAD CONDITIONS AND FLOATING CONTROL
S5
S6
S7
FS
FZ
FN
LOAD OR EXCHANGE PROGRAM STATUS DOUBLEWORD
S8
S9
S10
S11
MAS-TERF
MAPF
DM
AM
LOAD OR EXCHANGE PROGRAM STATUS DOUBLEWORD
WRITE DIRECT
D2
D3
D5
D6
D7
WK0
WK1
CI
II
EI
901060A. 308

Figure 3-8. Control Flip-Flops and Setting Conditions

e. Flip-Flop DM. When set, puts the decimal arithmetic trap in effect; it is set by bit 10 of the sum bus when a load or exchange program status doubleword is being executed.

f. Flip-Flop AM. When set, puts the arithmetic overflow trap in effect. Flip-flop AM is set by bit 11 of the sum bus when a load or exchange program status doubleword is being executed.

g. Flip-Flops WK0 and WK1. Write key bits used with the optional memory protection unit. These flip-flops are loaded from bits 2 and 3 of the D-register during load and exchange program status doubleword instructions.

h. Flip-Flops C1, II, and EI. Counter interrupt, input/output interrupt, and external interrupt group inhibit flip-flops. These are set by bits 5, 6, and 7, respectively, of the D-register during load and exchange program status doubleword instructions and are also set during a write direct instruction according to the configuration of the instruction.

3-11 Processor Control Panel

The processor control panel (PCP, figure 2-1) indicates the states of selected registers in the central processing unit and provides switch-controlled signals for manual computer operation. The upper section is specifically for maintenance purposes; the lower section is for the computer operator.

3-12 Internal Interrupt Circuits

The 16 internal interrupts in the central processing unit are classified as override, counter, and input/output as shown in figure 3-9. The override interrupts always have the highest priority. The counter zero interrupts are activated when the corresponding counter in core memory reaches a count of zero. These interrupts may be inhibited as a group. The input/output interrupt levels receive signals from the input/output system or from the control panel. The interrupts, in order of their priority, take the program to the memory location which may contain a subroutine for processing the interrupt. Two levels of interrupt are contained on one interrupt module.

3-13 Clock Generator

The CPU clock generator produces an irregular clock signal. The time intervals of this signal are determined by the number of logic levels through which the computer control signals must pass before another clock is needed. Three types of clocks are generated: a CL clock for trailing-edge triggering of ac flip-flops in the computer system, a general register CK clock to gate information into the general registers, and a dc hold DCH clock to latch data in the C-register.

A simplified block diagram of the clock generator is presented in figure 3-10. A clock pulse starts a pulse down delay line 4. This pulse is transmitted into delay line 1, and then into delay line 2 or 3, depending on the length of the clock interval required. Signals are tapped off at various points on the three delay lines. To obtain time intervals longer than from the combined delay lines, the pulse is recirculated through delay line 2 from one to three times. Since a clock pulse cannot enter delay line 1 without a previous clock pulse, delay line 4 is used to produce

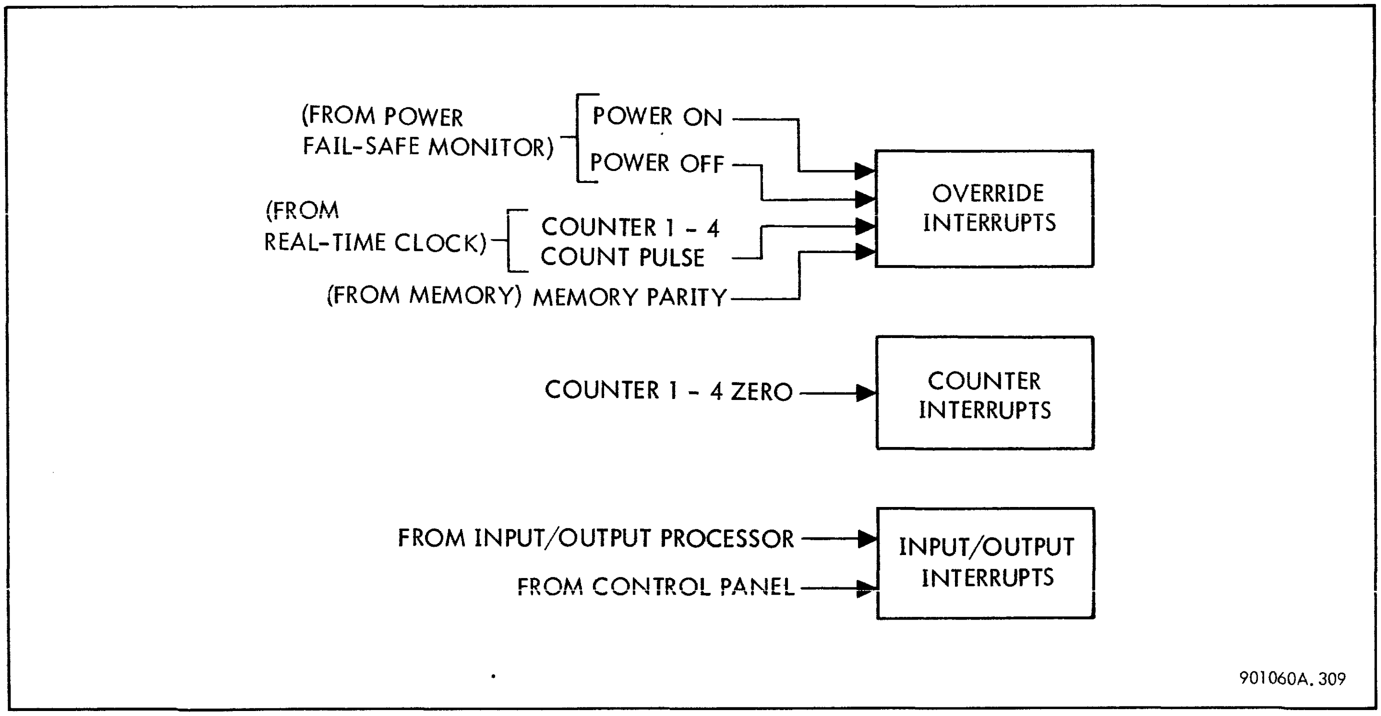


Figure 3-9. Internal Interrupt System, Block Diagram

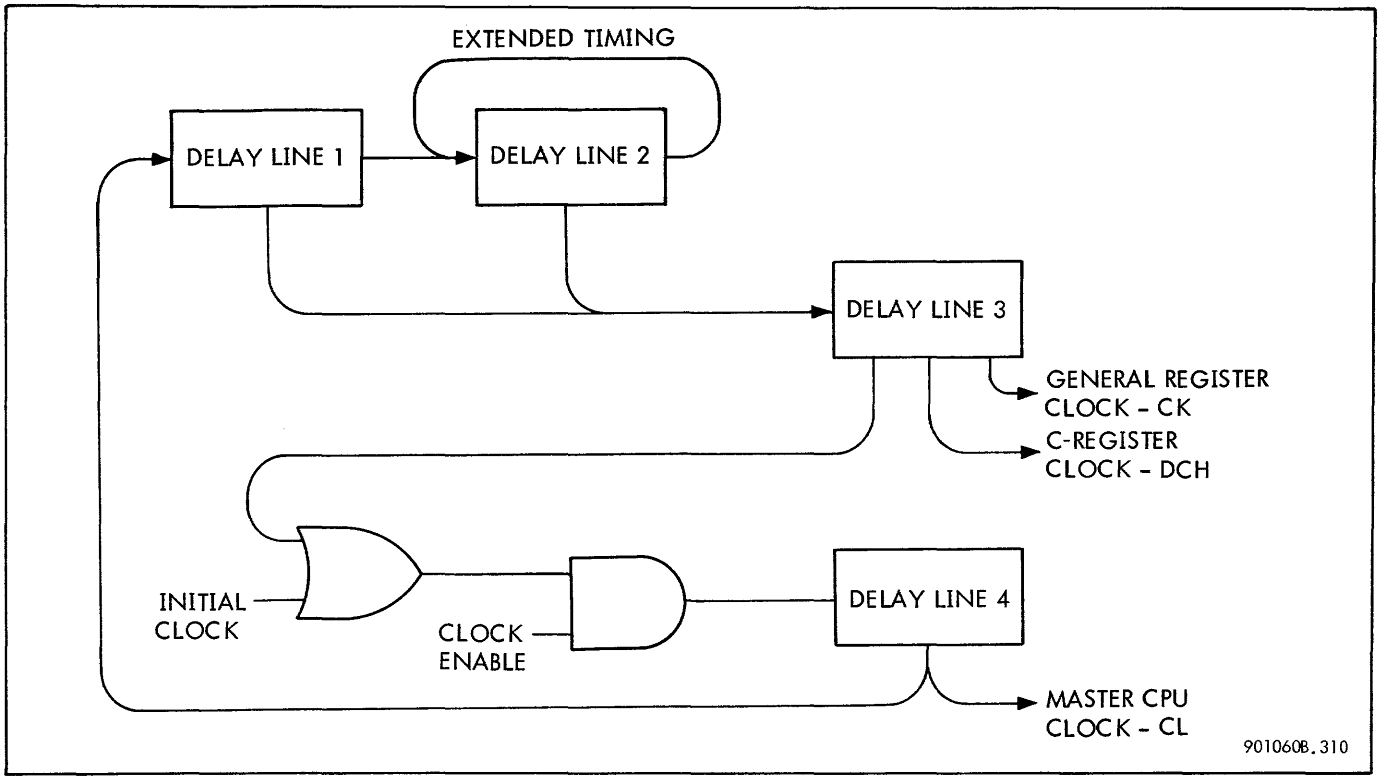


Figure 3-10. Clock Generator, Simplified Block Diagram

a clock pulse when the control panel CPU RESET pushbutton is pressed at the start of the operation. Delay line 4 also produces a clock pulse after the clock logic has been disabled for a period of time, and a clock enable signal is received.

A 2-MHz clock oscillator is included in the CPU in addition to the delay line clock circuits. The actual frequency is 2.048-MHz, and is crystal-controlled. This frequency is then down-counted by a flip-flop counter to make the following frequencies available:

2.048 MHz	-	used in the interrupt system
1.024 MHz	-	used to provide clock for single-clock mode. Also available for use by all units in computer system
512 kHz	-	unused
256 kHz	-	unused
128 kHz	-	unused
64 kHz	-	unused
32 kHz	-	unused
16 kHz	-	unused
8 kHz	-	input to time base selector
4 kHz	-	unused
2 kHz	-	input to time base selector
1 kHz	-	input to time base selector. Also used to drive PCP speaker
500 Hz	-	input to time base selector

3-14 Real-Time Clock

The real-time clock basically consists of an oscillator and a time base selector as shown in figure 3-11. Two of the outputs from the time base selector are connected to the internal interrupt circuits as counter 3 and counter 4 count pulses. (See paragraph 3-12.) The desired frequencies may be selected by means of switches on the time base selector module. Two additional outputs from the time base selector are part of optional equipment and are used as counter 1 and counter 2 count pulses in the internal interrupt circuits.

3-15 Watchdog Timer

The watchdog timer ensures that the program periodically reaches interruptible points in instruction execution. The timer is a six-bit counter triggered by a 1.024-MHz clock signal. The counter starts at the end of every instruction or at interruptible points in long instructions, and sends a trap signal to the trap circuits if a count of 40 is reached.

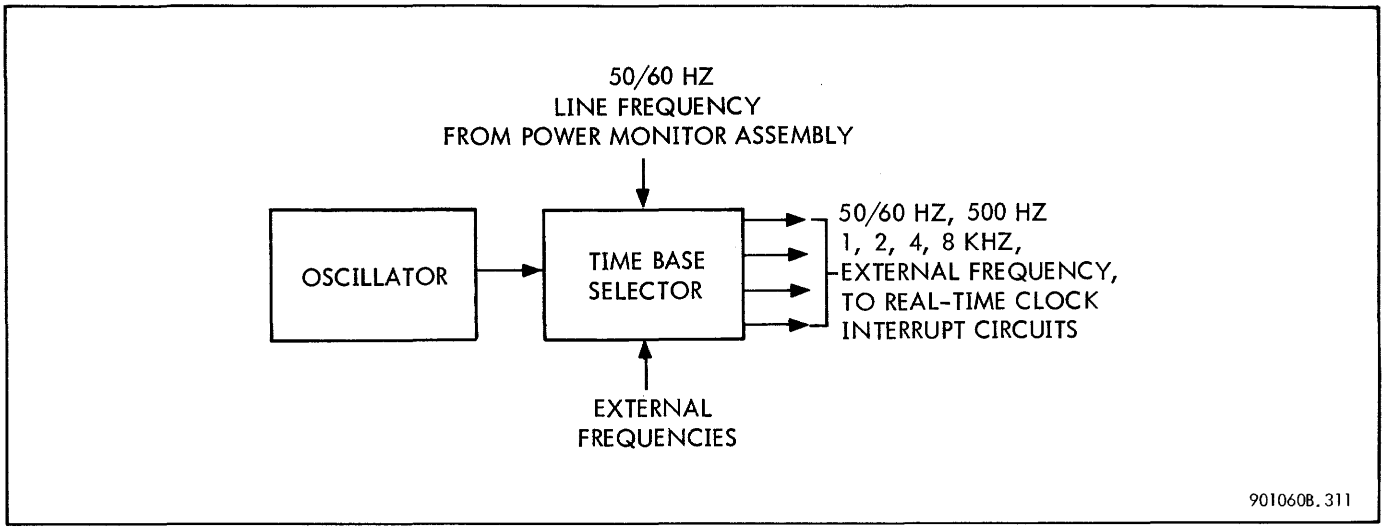


Figure 3-11. Real-Time Clock, Simplified Block Diagram

This indicates an elapsed time of approximately 40 μs before another interruptible point or end of instruction is detected. Programs trapped by the watchdog timer usually cannot be continued.

3-16 Power Supplies

Power for the basic CPU is furnished by two PT14 ac-dc converters, two PT15 dc-ac inverters, four PT16 logic power supplies, and one PT18 auxiliary power supply. An output from one of the PT15 converters supplies 120V, 2000 Hz power for the core memory power supply (PT17).

3-17 OPTIONAL FEATURES

3-18 Decimal Arithmetic Unit

This unit contains the logic circuits to perform arithmetic and logic operations on four-bit binary-coded decimal data. Decimal instructions that may be executed are Add, Subtract, Multiply, Divide, Compare, Shift, Load, Store, Pack, Unpack, and comprehensive Edit. One eight-bit byte in memory contains two decimal digits. Decimal fields are variable in length, the longest being 31 digits plus sign.

3-19 Floating Point Arithmetic Feature

This feature extends the A-, B-, C-, D-, and CS-registers and the sum bus from 32 to 57 or 58 bits so that floating point arithmetic may be performed.

3-20 External Interrupts

Fourteen groups of external interrupts are available for recognizing and processing selected special conditions that exist in external equipment. All external interrupt signals are wired into the interrupt chassis according to the requirements of the individual system. The groups of external interrupts can be connected in any priority sequence, determined by the manner in which it is connected to the rest of the interrupt system. Each group of external interrupts contains a maximum of 16 levels. The priority assignment of interrupts within these 16 levels is fixed, the level assigned the lowest memory address having the highest priority.

The individual interrupt modules, each containing the logic for two interrupt levels, are identical to the interrupt modules used in the internal interrupt system described in paragraph 3-12. The external interrupts operate in the same manner as the internal interrupts except for the source of the interrupt signal.

3-21 Power Fail-Safe Feature

The power fail-safe feature enables the computer to retain programs and processed or partially processed data when the primary power source fails. When a power failure occurs, the power fail-safe circuits notify the computer by an interrupt. Sufficient energy is stored in the Sigma 7 power supply system to maintain dc power for the duration of a short power failure subroutine. Such a subroutine usually stores in core memory all pertinent volatile data held in flip-flops in the CPU. On reapplication of primary power, a second interrupt causes the computer to enter a power-on recovery subroutine that restores the computer to the state existing prior to the lapse of power.

The primary power source is monitored by a power monitor assembly located in the power distribution panel. When the power drops below a preset threshold, a power-off signal is generated and a power-on signal is generated when the line voltage goes above the preset level. Both of these signals, connected to the internal interrupt system, are part of the override group described in paragraph 3-12. The power-on and power-off signals have the highest priority in the override group and in the entire interrupt system.

3-22 Two Additional Real-Time Clocks

The optional two additional real-time clocks are identical to the two real-time clocks included with the standard computer system. Two additional interrupt modules must be added to the internal interrupt chassis to receive the counter 1 and counter 2 pulses described in paragraphs 3-12 and 3-14. The four counter interrupts are part of the override group in the internal interrupt system.

3-23 Additional Private Memory Register Banks

Up to 31 additional private memory register banks may be added to the CPU. Each bank is identical to the private memory bank included with the standard computer system and described in paragraph 3-9. One of a possible 32 private memory register banks is selected by a code in the program status doubleword and this code is placed in the RP-register.

3-24 Memory Map

When the computer is in the memory map mode (MM-bit in the program status doubleword true), all memory references used by the program (direct, indirect, or indexed) are referenced as program addresses. The memory map divides the program addresses into a series of 256 512-word blocks called memory map pages. The nine low-order bits of the program address select a word within a 512-word page of addresses. The eight high-order bits select one of 256 pages. The addresses of these pages appear in the CPU registers, but do not necessarily represent the same physical locations in core memory.

Associated with each page of addresses is an eight-bit map register. The 256 map registers are designated page 0 through 255 as shown in figure 3-12. These registers are integrated-circuit, fast-access memory registers identical to those used for the CPU private memory.

The program places in each map page register an eight-bit code representing the eight high-order bits of the address of an actual core memory location. When memory is addressed in the map mode and when indexing has been completed, the nine low-order bits of the address field in the C-, P-, or Q-register are presented to the nine low-order memory address lines. The eight high-order bits of the program address go into the map logic to select the outputs of the map register associated with the addressed page. The map register outputs replace the high-order bits of the program address and are presented to the high-order address lines to select an actual 512-word block of memory locations.

Using the memory map registers, the programmer may locate 512-word blocks of information in any desired positions in memory, either in a selected set of sequential locations or distributed throughout memory. Since the map is used during reading as well as during writing, the program address is altered by the map registers to read the information out of the block of memory locations into which it was written.

The program used to load the map register contents into memory must organize the codes in a byte string as shown in figure 3-13. This series of codes in memory, the memory map control image, may reside in any set of sequential memory locations. A control image for the entire set of 256 map registers may be contained in 64 words in memory.

The map control image is transferred from memory to the map registers one word at a time by a Move to Memory Control instruction. Each memory word is transferred into four successive map registers.

Not all memory locations may be addressed at all times in the map mode. Program addresses in the range 0 through F are not used to address core memory. The four low-order bits of the program address used as a private memory register address are the source or destination of the instruction operand.

In the master mode, the CPU has access to all memory locations. In the slave mode, however, memory access is controlled by a set of 256 program control registers that are part of the memory map option. A two-bit program control register is assigned to each page of program addresses as shown in figure 3-12. These registers are referenced as access control code registers in the Sigma 7 Reference Manual. The program control codes are interpreted by the CPU as shown in table 3-1.

The program control bits are stored in integrated-circuit fast-access memory registers like those used for the memory map codes. The program must first store in memory a series of two-bit codes, 16 codes to a memory word as shown in figure 3-14. This string of codes in memory is known as the program control image. The image is transferred to the program control registers by a Move to Memory Control instruction which reads a word at a time from memory and places the codes in a succession of locations in the fast-access memory registers.

In the slave mode, when a program address is received by the CPU, the logic first checks the program control registers to see the type of access that may be made into the page of program addresses.

Note

> The memory map controls access into program addresses, not into actual core memory locations.

3-25 Memory Protection Feature

Access to core memory locations may be program-controlled by using a set of write lock registers identical to those used for the control codes. A two-bit write lock code is assigned

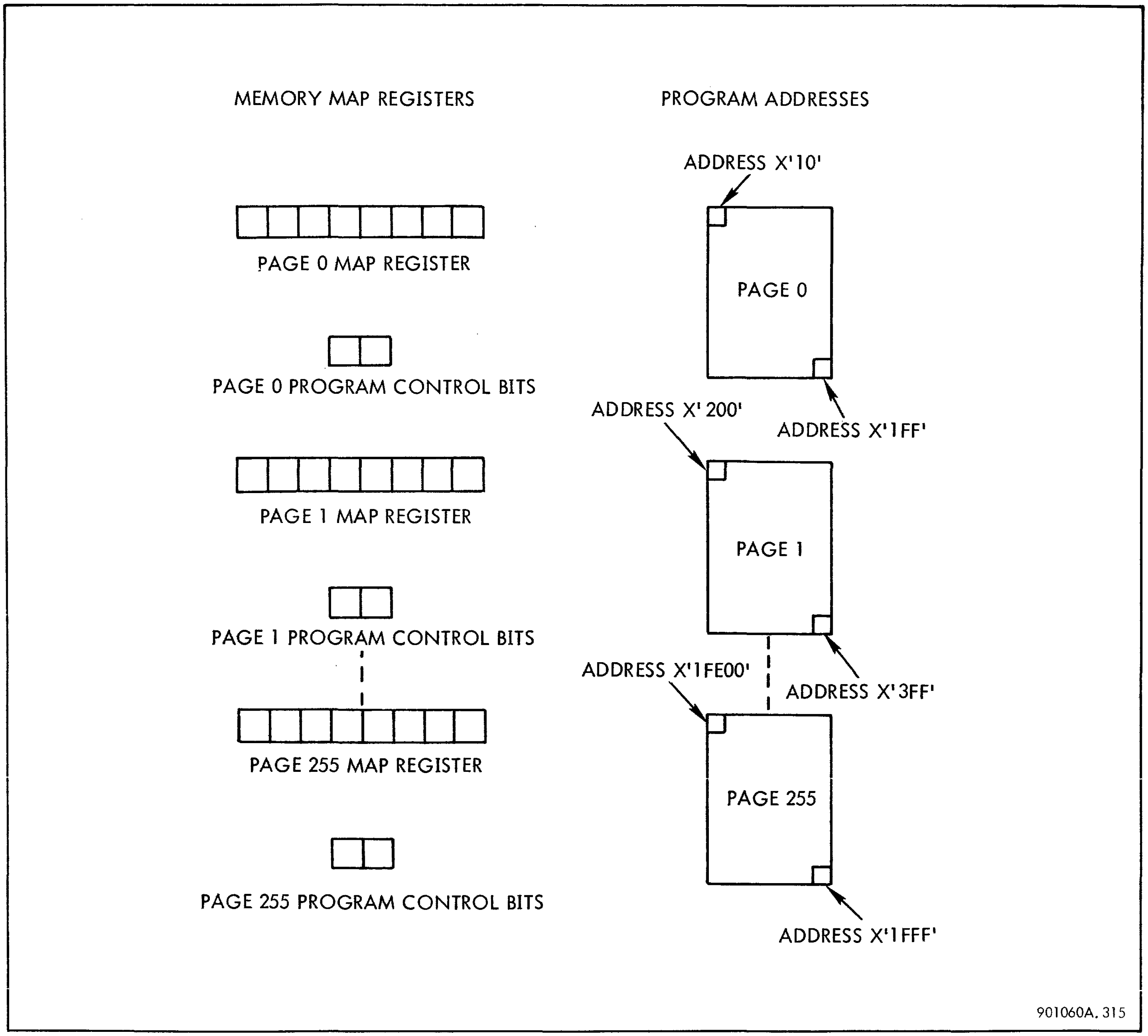


Figure 3-12. Memory Map Organization

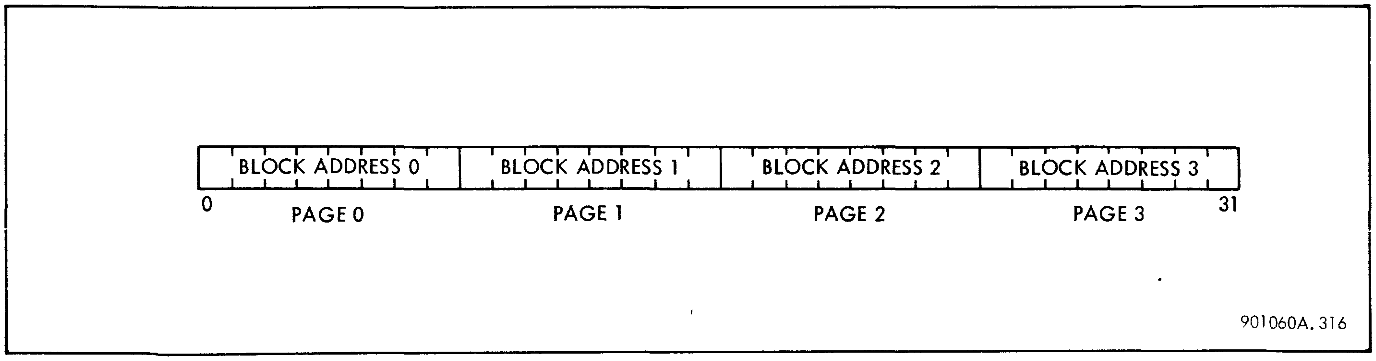


Figure 3-13. Organization of Block Addresses in Memory Map Word

Table 3-1. Program Control Code Functions

PCB0	PCB1	Function
0	0	Permits program to write into, read from, or take instructions from this page of program addresses
0	1	Prevents writing into this page of program addresses, but permits reading or taking instructions from this page
1	0	Prevents writing into or taking instructions from this page of program addresses, but permits reading from the page
1	1	Prevents any access to this page of program addresses by the slave program

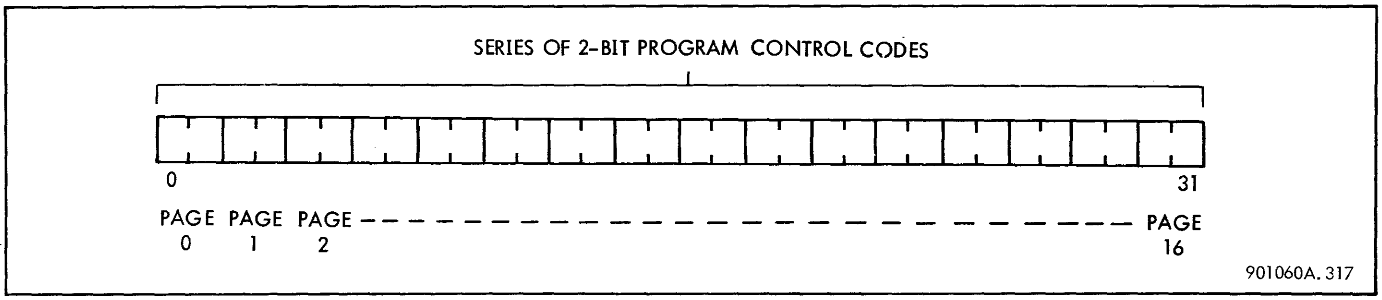


Figure 3-14. Organization of Program Control Codes in Memory Word

to each 512-word block of core memory addresses. The write lock codes with 16 codes to a memory word are first written into memory as a lock control image. The lock control image is transferred to the write lock registers by a Move to Memory Control instruction. When memory access is required, the write lock codes are compared with two write key bits in the program status doubleword to determine if the memory bank addressed may be used for reading or writing.

Note

The memory protection feature protects actual core memory addresses, not program addresses.

3-26 BASIC TIMING CHARACTERISTICS

Timing signals for instruction execution are taken from the clocks generated from delay line 1 or 3 and from two sets of timing sources: phase control flip-flops and clock-gating flip-flops. The phase control flip-flops determine thephase of the instruction being performed; that is, PRE1 through PRE4 are for preparation phases and PH1 through PH15, for execution phases. Only one phase flip-flop is set at a time.

Every instruction must use preparation phases PRE1 and PRE2 to clear registers and to prepare for operand access. Phases PRE3 and PRE4 are used for operations that require special preparation sequences such as indirect addressing.

When preparation is complete, flip-flops PH1 through PH15 are set in numerical order unless a branch signal is given during one phase to skip one or more phases at the next transition or to return to an earlier phase. Generally, only a few of the available phases are used, and the instruction may end at any point in the sequence.

The transition between phases is marked by a CL clock signal. Although the clock signal always comes from the same source, the time interval between clock pulses varies according to the time required to generate the signals needed before the next clock pulse. The time inverval is defined by one of the clock gating flip-flops, T1L, T4L, T8L, or T10L, or by a clock-gating signal, T6L, which is true whenever none of the clock-gating flip-flops is set. The time intervals are given in table 3-2. Gating the clock signals from the delay lines in the clock generator is shown in figure 3-10. A timing diagram of a typical sequence of CL clock pulses is given in figure 3-15.

3-27 CENTRAL PROCESSOR OPERATIONS

Computer operations are controlled by instructions, which are first loaded into memory and then read out of memory by the arithmetic and control units.

Reference addresses in the instruction word can be interpreted in three ways: the reference address refers to an actual location in memory, the location specified in the reference address contains the address of the instruction to be executed (indirect addressing), or the contents of the index register must be added to the reference address before memory access (indexing). Indirect addressing and indexing may be performed on the same instruction; that is, memory is first indirectly addressed and then the contents of the index register are added to the address read from memory.

Table 3-2. CL Clock Pulse Intervals

Gating Signal	Time Interval (ns)
T1L	150
T4L	280
T6L	320
T8L	380
T10L	440

An immediate instruction contains in the reference address field a number to be operated on, rather than an address. If bit position 0 of an immediate instruction word contains a one, indicating indirect addressing, the operation is illegal, and the program branches to a trap memory location which may contain a subroutine to carry out the necessary remedial steps.

In all computer operations, words are read from memory a full 32-bit word at a time; however, writing may be done on a byte basis, using four signal lines to indicate which of four bytes are to be written.

3-28 PROCESSOR CONTROL PANEL OPERATIONS

When control switches are operated on the processor control panel, a phase sequence is entered similar to the phases used in central processor operations. The CPU contains a set of phase flip-flops designated PCP1 through PCP7 for use of the processor control panel. The flip-flops are set and reset by ac clock signals from the CPU delay lines, usually using the T6L clock-gating signal.

Operating the RESET switch places the control panel logic in PCP2. When operating other switches, the other phases follow in numerical order, and the computer functions are performed according to the setting of the switches. PCP1 follows PCP7, except during a LOAD operation, and the operation waits in PCP2 until the COMPUTE switch is placed in RUN or STEP. The PCP then goes through the sequence to PCP5, and the next clock sets the first preparation phase flip-flop to start normal computer operation.

3-29 INTERRUPT OPERATION

This system permits the interruption of the normal sequence of events in the CPU by a signal generated either internal or external to the CPU. Each interrupt has an assigned priority and can be individually armed and enabled by the program. An active interrupt causes the program to branch to a memory address assigned to the interrupt, and the instruction in that address is executed. The interrupt location may contain a single instruction or may take the program to a subroutine. Normal operation can usually be resumed from the point of interruption.

Interrupt operations are controlled by phase flip-flops INTRAP1 and INTRAP2. The flip-flops are clocked by CPU ac clock signals; however, actual control of the phases is in the interrupt circuits which are clocked by a 2.048 MHz clock from the CPU. The synchronization of the clocks is accomplished by disabling the CPU clock until a clock is received from the interrupt chassis. During the interrupt phases, the next instruction address is stored, and the interrupt address associated with the interrupt in progress is received by the CPU for memory access.

Flip-flops in the interrupt circuits, which indicate whether the interrupt is armed, disarmed, enabled, waiting, or active, determine whether or not an interrupt is admitted. Priority is established by cabling between these circuits,

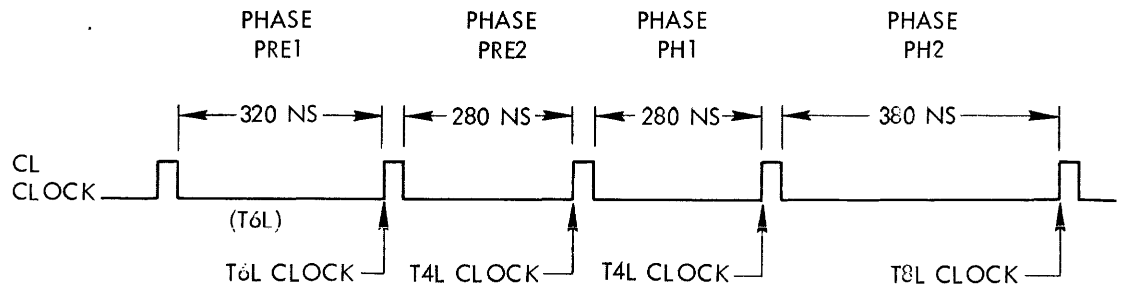


Figure 3-15. CL Clock Timing, Typical

because only the highest priority interrupt may advance from the waiting to the active state.

3-30 TRAP OPERATION

The trap system is used basically for error detection. A trap indication results from a condition such as a nonexistent instruction, addressing a nonexistent memory location, watchdog timer runout, or an instruction calling for decimal or floating point operation when the option is not included in the system. The detection of a trap condition causes the immediate execution of a trap instruction in a unique location in memory. Trap operation is also used to simulate instructions not included in the system logic; that is, a Call instruction causes the program to trap to a specific location from which a branch is made to a subroutine to carry out the desired operation.

Trap operations are controlled by the two phase flip-flops used in interrupt operation: INTRAP1 and INTRAP2. A trap operation has priority over any interrupt except when a power on or power off interrupt has been detected and the INTRAP phases are entered in the first cycle following the end phase of the instruction in process. Trap phases are clocked by T6L and T10L clocks and are not affected by the 2.048 MHz clock in the interrupt circuits. During the trap phases, the next instruction is stored and the trap address associated with the trap signal received is presented to memory for access. The program then branches to the memory address stored in the trap location.

3-31 INPUT/OUTPUT CHANNEL

An input/output channel is composed of a multiplexing input/output processor or a selector input/output processor and of one or more device controllers, each connected to one or more peripheral input/output devices such as card readers and punches, magnetic tape units, and printers. Operation of the device controllers and devices is covered in other technical manuals. Therefore, it is not explained in detail in this section.

The input/output processor controls all data transfers between the computer memory and the peripheral devices. After receiving an instruction from the CPU to start operation, the IOP executes input/output commands in much the same manner as the CPU executes instructions. Input/output is performed simultaneously with central processing operations, since the IOP operates independently.

3-32 Multiplexing Input/Output Processor

The multiplexing input/output processor (MIOP) contains an address register, a data register, fast access memory registers, adding circuits, and delay lines for generating timing pulses. Interface signal exchange is via cable receivers and cable drivers, and a logical one is represented by +2 volts and a logical zero is represented by 0 volts.

Information exchange between the IOP and the CPU takes place on three function lines, three IOP address lines, and two condition code lines. The function lines carry a three-bit code which indicates one of five input/output instructions being executed. The IOP address is a three-bit code designating one of eight possible IOP's being addressed. The IOP addresses, in the range 0 through 7, are set by manual switches in the individual IOP units. Information exchange between the IOP and other units is shown in figure 3-16. However, timing, control, and interrupt signals are not shown on the diagram.

Each IOP is connected to one memory port and uses a single memory bus. Memory addresses are placed on the 17 address lines for memory access. Commands, responses, and data are carried on 32 memory data lines as shown in figure 3-16.

The eight data lines between the IOP and the device controller carry data in both directions and also carry the eight order bits from the input/output commands. Interrupts are generated in the device controller and are acknowledged by the IOP.

A general flow chart of the IOP functions during an input/output operation is shown in figure 3-17. The functions in the instruction execution block are carried out in cooperation with the CPU. Except for the Halt instruction, the functions under command execution proceed without help from the CPU. Commands are first stored in memory by the CPU.

After a Start Input/Output instruction is indicated on the function lines, and an IOP recognizes its address on the IOP address lines, all communication between the IOP and the CPU takes place by way of locations X'20' and X'21' in memory. The IOP reads the first command address and the device controller (or device and device controller) address from location X'20'. Information on the device status is then placed in locations X'20' and X'21' by the IOP for use by the computer. During command execution, the IOP acts independently and transfers a series of bytes between the device and successive locations in core memory until the number of bytes sepcified in the command doubleword is transferred. One word at a time is transferred between the IOP and memory on the 32 data lines. Transfer between the IOP and the device controller is a byte at a time on the eight data lines.

The IOP contains a fast-access memory register for each device controller. Each command goes into the register associated with the device controller specified in the input/output instruction. In the command word there is an eight-bit order field directing the peripheral unit to read, write, read backward, and so forth. These order bits are transferred directly to the device on the eight data lines without entering the memory register.

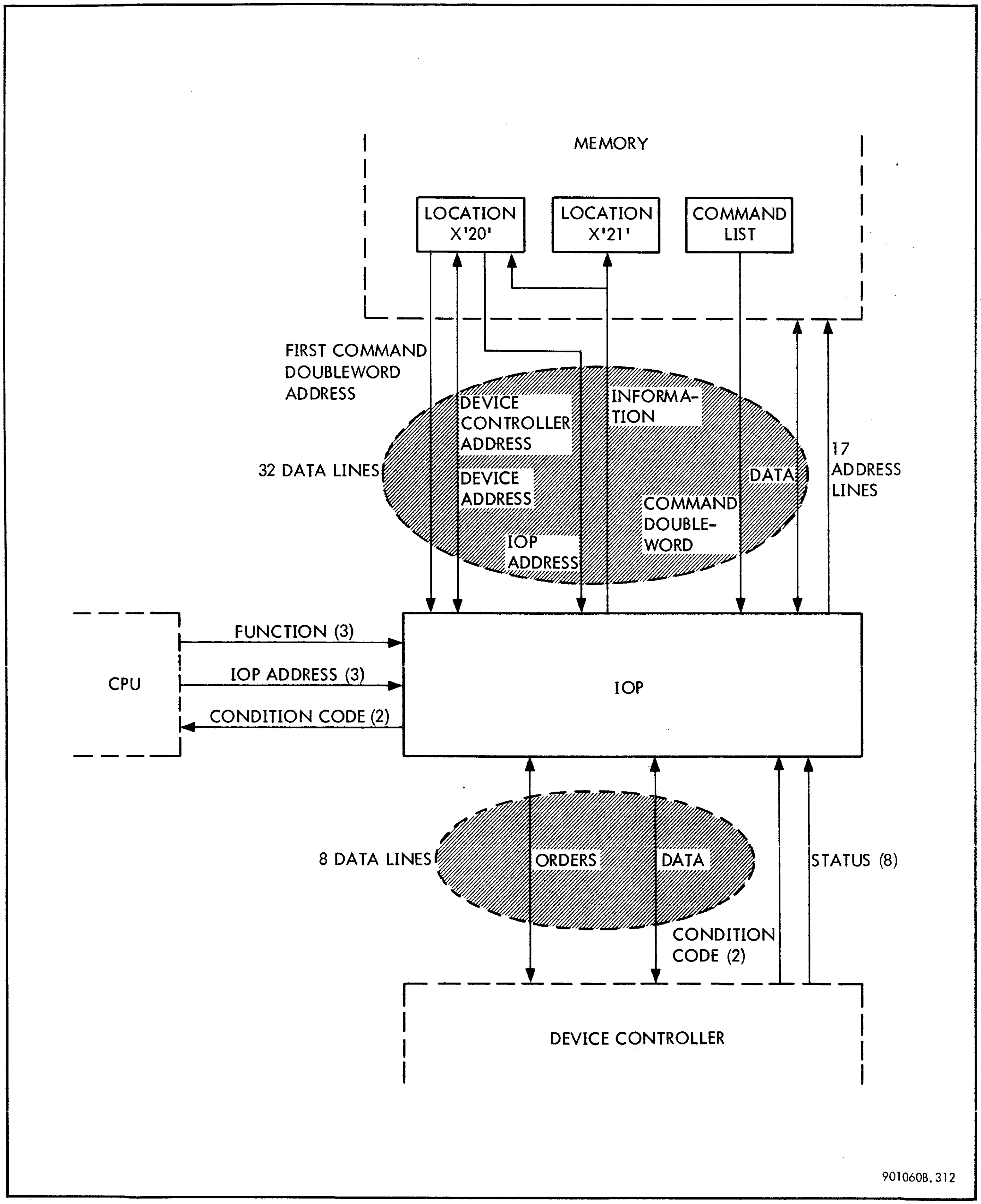


Figure 3-16. IOP Information Exchange

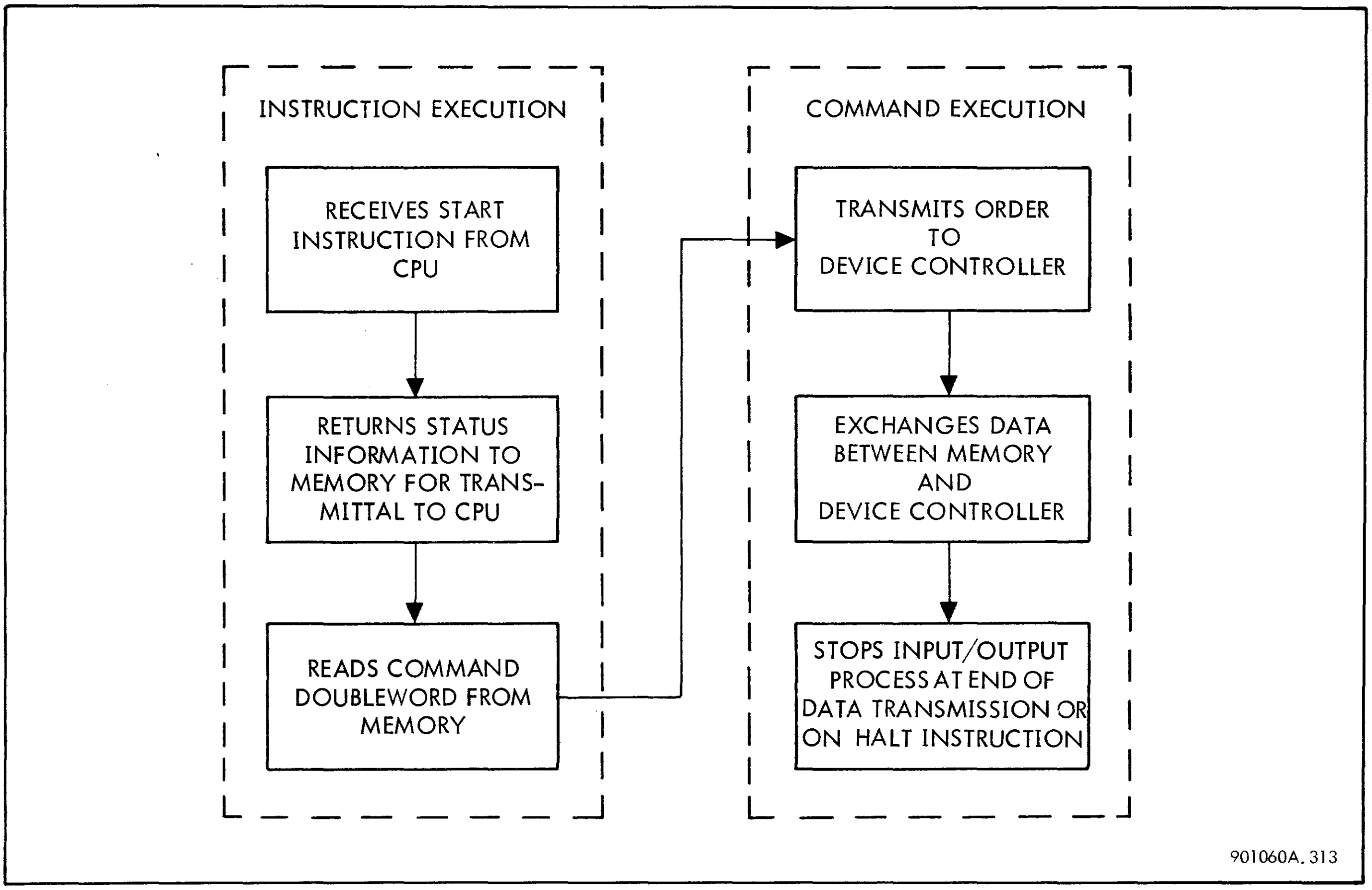

INSTRUCTION EXECUTION
RECEIVES START INSTRUCTION FROM CPU
RETURNS STATUS INFORMATION TO MEMORY FOR TRANS-MITTAL TO CPU
READS COMMAND DOUBLEWORD FROM MEMORY
COMMAND EXECUTION
TRANSMITS ORDER TO DEVICE CONTROLLER
EXCHANGES DATA BETWEEN MEMORY AND DEVICE CONTROLLER
STOPS INPUT/OUTPUT PROCESS AT END OF DATA TRANSMISSION OR ON HALT INSTRUCTION
901060A. 313

Figure 3-17. IOP Functional Flow Chart for Input/Output

After input/output has started, the computer communicates with the IOP only to halt the operation, to test the status of data transmission or of the device, or to acknowledge an interrupt. Halt, Test, and Interrupt Acknowledge instructions are available for these communications. During execution of these instructions, the instruction code is placed on the function lines to the IOP. After an IOP address or an interrupt has been recognized, the remainder of the communication between the CPU and the IOP takes place via memory locations X'20' and X'21' as in the Start instruction. The type of information carried on the condition code, data, and status lines varies according to the instruction being executed.

Two types of chaining are used to permit the IOP to take more than one command from memory with only one instruction from the CPU.

a. Command Chaining. Stipulated by setting a command chain flag bit in the command doubleword in memory. When the command has been executed, the IOP, instead of terminating the service, steps up a command counter by one and reads the next command from the next memory location. This process continues until the IOP receives a command with a zero in the command chain flag bit. The command chain function permits the IOP to read a chain of commands from core memory and to execute these commands until the chain is completed without communicating with the CPU.

b. Data Chaining. Setting a data chain flag bit in the command doubleword permits scatter-reading and gather-writing. Scatter-reading is placing information from one physical record in a device into one or more noncontiguous memory locations. Gather-writing is taking information from one or more noncontiguous memory locations and writing it into one physical record in a device, such as a magnetic tape unit or a card punch. When a data chain flag bit is detected, the IOP reads a command from the next memory location as in command chaining; however, the order bits in the command are not passed on to the device controller. Thus, the operation required in the previous order is continued without starting a new record. Data chaining stops when a zero is detected in the data chain flag bit in a command word.

3-33 IOP Priority

Two types of priority are established in the system with regard to the IOP: interrupt priority and memory access priority.

Each IOP is recognized in the order of its assigned priority, if more than one IOP sends an interrupt request to the CPU while an interrupt is being processed. The priority is determined by the cable connection as shown in figure 3-18. IOP 0 has the highest priority and IOP 4 has the lowest priority. All of the IOP's share a single interrupt request line to the CPU.

If more than one IOP requests memory access from the same memory bank at the same time, the priority is determined by the memory port to which the IOP is connected. Since memory port A has a higher priority than port B, and the four port expander outputs have priority corresponding to their numerical order, the IOP units have memory access priority in the following order: IOP 4, IOP 3, IOP 2, IOP 1, and IOP 0.

3-34 DETAILED PRINCIPLES

3-35 INTRODUCTION

This subsection describes the operation of each functional unit of equipment, primarily in terms of logic elements (flip-flops, gates, registers) rather than in terms of circuit components (capacitors, resistors, transistors). Also included are functional, logic, timing, and simplified schematic diagrams.

3-36 Circuit Description

Diode-transistor logic is used for all Sigma 7 T-Series module logical functions. Integrated circuit flip-flops, inverters, and buffer amplifiers are combined with discrete electronic components such as diodes, transistors, and resistors to form logic element modules, flip-flop and register modules, and special circuit modules. The logic element modules contain AND, OR, NAND, and NOR gates. The outputs are logically amplified or amplified and inverted. The flip-flop modules are used for storage, counting, and shifting. The special circuits, such as cable drivers and receivers, one-shots, and lamp drivers, are used primarily for communication with other equipment or display devices.

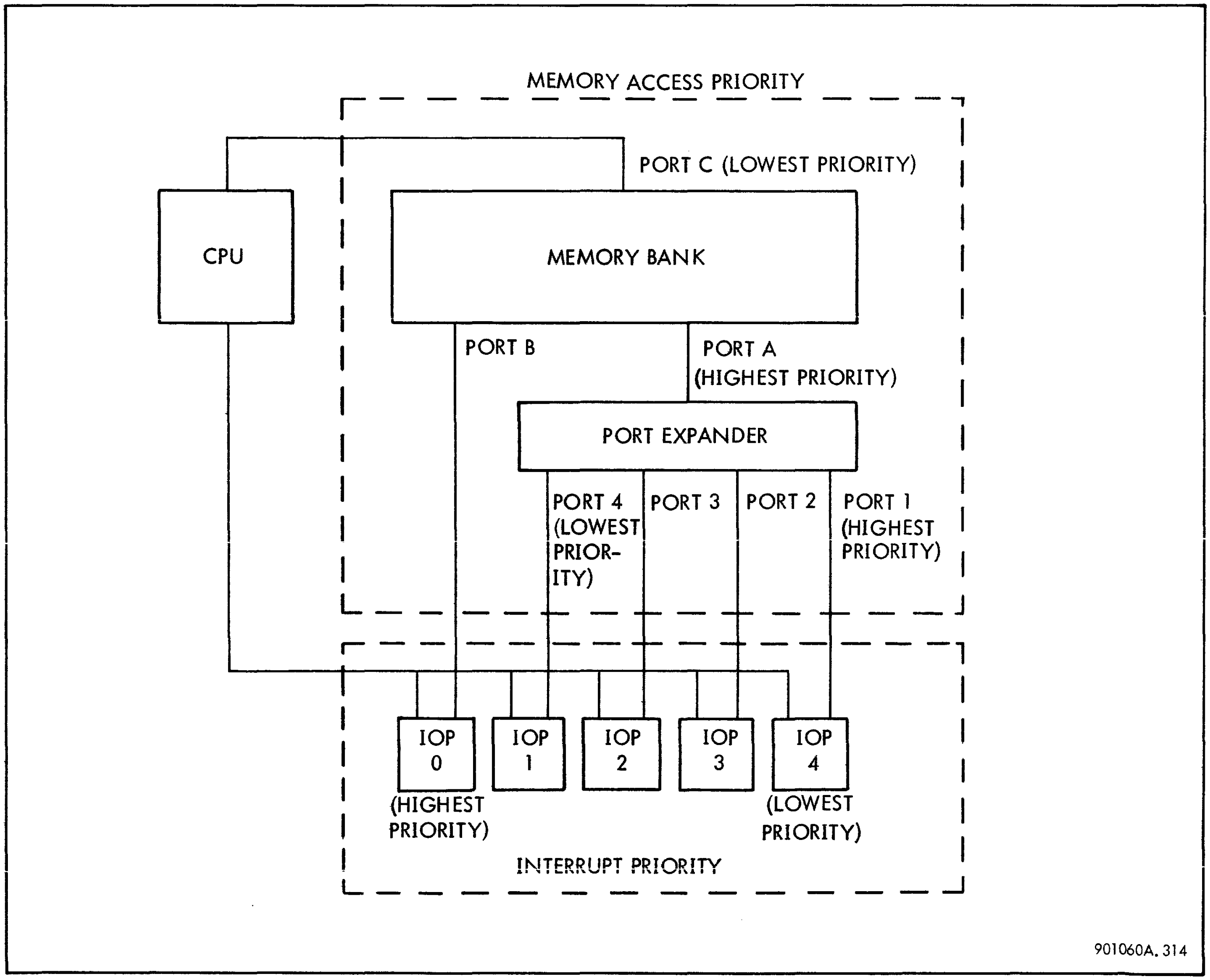


Figure 3-18. IOP Priority Assignments

The clock pulse that is applied to the T-Series flip-flops is a standard 4-volt logic signal that is shaped and synchronized to provide proper system operation.

Standard supply voltages for the logic circuitry are +4, +8, and -8 volts. Voltage tolerances must be within ±10 percent. The logic levels are +3.6 volts for a minimum logical 1 and +0.05 volts for a maximum logical 0 (nominal 0 volts for a zero and nominal +4 volts for a one). Typical noise margins are 1.1 volts for both one and zero rejections. The typical propagation delay time is 40 ns for flip-flops and 18 ns for buffered AND/OR or NAND/NOR gate circuits. Propagation time is defined as the time between a 50 percent change of input level and a 50 percent change in output level.

3-37 TYPICAL T-SERIES CIRCUIT. The AT21 cable driver schematic diagram in figure 3-19 shows a typical combination of an integrated circuit with diodes and resistors on a logic module. The module contains 14 cable driver circuits, only one of which is shown in the figure.

A logic diagram of the AT21 cable driver is shown in figure 3-20. Diodes CR1 and CR2 make up one AND gate; diodes CR3 and CR4 make up a second AND gate; diodes CR5 and CR6 form an OR gate whose output is amplified by the integrated circuit buffer (XDS 306) A4 through A7. Emitter follower Q1 transmits an uninverted signal to the cable connected to P2.

As shown in figures 3-19 and 3-20, circuits 2 through 7 contain only one AND gate. In the remaining circuits, the outputs of two AND gates are joined in an OR gate.

The integrated circuit chip (XDS 306) consists of four buffer circuits, as shown in the schematic diagram of A4 through A7 (figure 3-19). The circuits are designated -1 through -4. As shown in figure 3-19, all of the buffer circuits are used on the AT21 module except A6-1 and A6-2.

3-38 INTEGRATED CIRCUITS. Six integrated circuits are used as the active logic elements in the Sigma 7 modules. The logic output of each circuit may be considered like a semiconductor switch at ground. A conducting output (transistor on) propagates a zero; an open output (transistor off) propagates a one. The +4, +8, and -8 supply voltages are standard for the integrated logic circuits.

The following integrated circuits are used in Sigma 7:

a. Memory Element (XDS 304). Contains eight storage bits in one integrated circuit element. A schematic diagram of the memory element is shown in figure 3-21. Each bit in the memory element consists of a flip-flop with the collectors of the two transistors, Q1 and Q2, connected to the source voltage. The control line is connected to transistor Q3 which supplies power to the decoding and writing circuits. When the control line is false, these circuits are deprived of power, and the data output line is the collector of a cutoff transistor, Q4. The three address lines are decoded so that only one of the eight flip-flops in the memory element is subjected to a write line input. Also, only that bit controls the data output line. The register-write input must be true in order for a data input signal to set the selected flip-flop. A timing diagram of memory element operation is shown in figure 3-22.

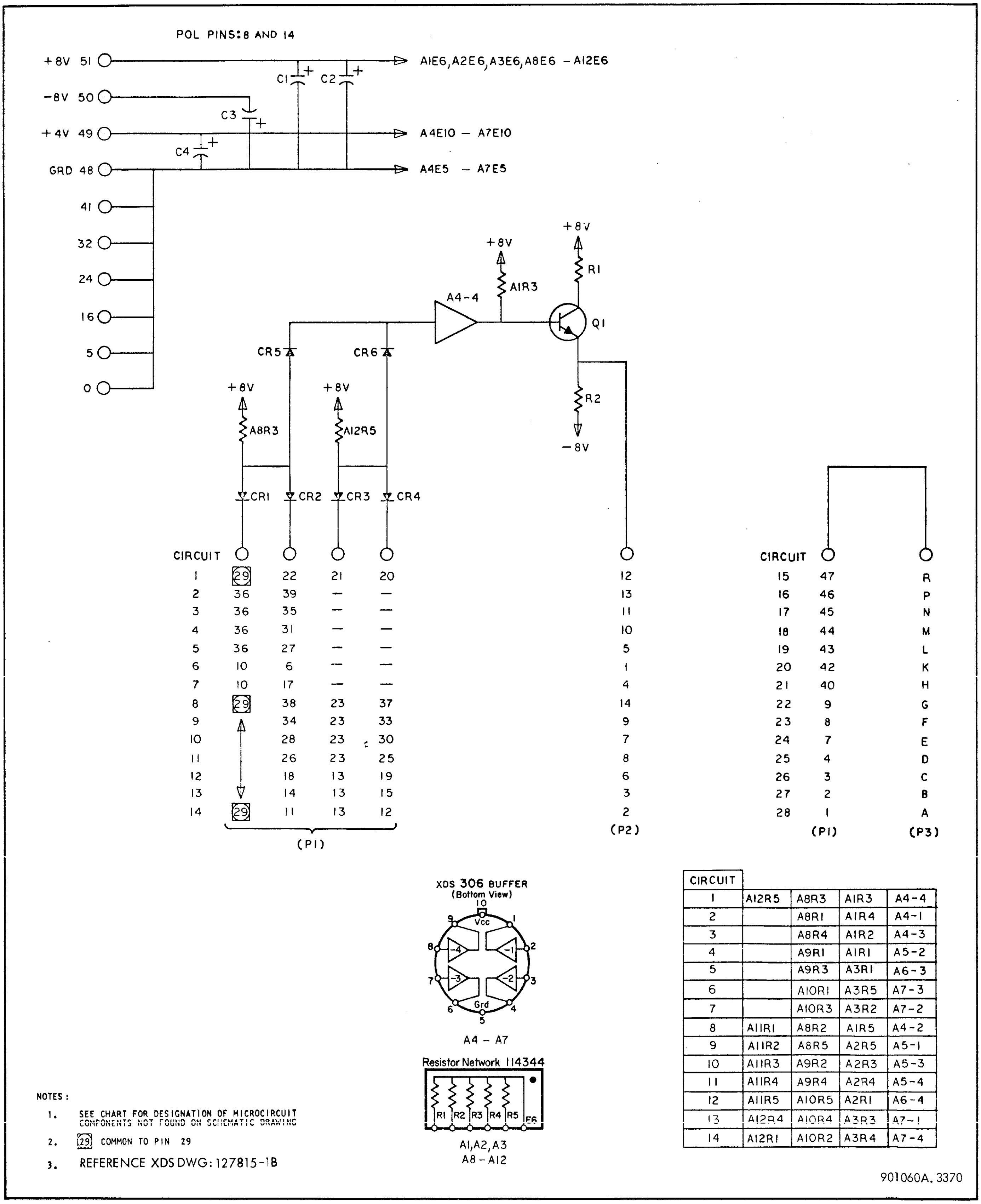
b. Inverter (XDS 305). Contains four inverter circuits in one package. The inverters drive and are driven by diode-resistor AND/OR gates. The nominal threshold is +2.0 volts. Each inverter circuit presents the logical inversion of the input signal at the output. The typical gate structure for the inverter is shown in figure 3-23, and a schematic diagram of one inverter is shown in figure 3-24.

c. Buffer (XDS 306). The XDS 306 buffer contains four buffer circuits in one package. The buffers drive and are driven by diode-resistor AND/OR gates. The nominal threshold is 2.0 volts. The typical gate structure for the buffer is shown in figure 3-25. Each buffer circuit presents the logical equivalent of the input signal at the output. A schematic diagram of one buffer circuit is shown in figure 3-26.

d. Flip-Flop (XDS 307). Contains one flip-flop circuit in one package. The flip-flop drives and is driven by diode-resistor AND-OR gates. The nominal threshold is 2.0 volts. The typical gate structure is shown in figure 3-27. A schematic diagram of the flip-flop is shown in figure 3-28, and a logic diagram is shown in figure 3-29. The flip-flop is set or reset on the trailing edge of the clock pulse. The flip-flop sets, if both set and reset inputs are true. The dc set and dc reset inputs act independently of the others. When the dc set input becomes one, the set output becomes one and remains one after the dc set input becomes zero. When the dc reset input becomes one, the reset output becomes one and remains one after the dc reset input becomes zero. When the dc set and dc reset inputs are one simultaneously, the state of the flip-flop is indeterminate.

e. Discriminator (XDS 308). Contains two discriminator circuits in one integrated circuit package. A schematic diagram of the discriminator is shown in figure 3-30. If the input signal is greater than the reference voltage, the discriminator generates a logical 1 at the output. If the input signal is less than the reference voltage, the discriminator generates a logical 0 at the output. The discriminator is driven by an emitter follower circuit of the type shown in figure 3-31, and drives diode-resistor AND/OR gates of the type shown in figure 3-32. The voltage discrimination level for each circuit is set externally. Input signals must range from -2.0 to 4.4 volts with a voltage discrimination level between -1.5 and +1.5 volts.

f. Flip-Flop (XDS 311). Contains two flip-flop circuits in one package. A schematic diagram of the flip-flop is shown in figure 3-33. The flip-flop changes state at the trailing edge of the clock pulse. If both the set and reset inputs are true, the flip-flop sets. The dc set and dc reset inputs act independently of the others.



CIRCUIT					
1	29	22	21	20	12
2	36	39	—	—	13
3	36	35	—	—	11
4	36	31	—	—	10
5	36	27	—	—	5
6	10	6	—	—	1
7	10	17	—	—	4
8	29	38	23	37	14
9	↑	34	23	33	9
10		28	23	30	7
11		26	23	25	8
12		18	13	19	6
13	↓	14	13	15	3
14	29	11	13	12	2
	(PI)				(P2)

CIRCUIT		
15	47	R
16	46	P
17	45	N
18	44	M
19	43	L
20	42	K
21	40	H
22	9	G
23	8	F
24	7	E
25	4	D
26	3	C
27	2	B
28	1	A
	(PI)	(P3)

CIRCUIT				
1	AI2R5	A8R3	AIR3	A4-4
2		A8RI	AIR4	A4-I
3		A8R4	AIR2	A4-3
4		A9RI	AIRI	A5-2
5		A9R3	A3RI	A6-3
6		AIORI	A3R5	A7-3
7		AIOR3	A3R2	A7-2
8	AIIRI	A8R2	AIR5	A4-2
9	AIIR2	A8R5	A2R5	A5-I
10	AIIR3	A9R2	A2R3	A5-3
11	AIIR4	A9R4	A2R4	A5-4
12	AIIR5	AIOR5	A2RI	A6-4
13	AI2R4	AIOR4	A3R3	A7-I
14	AI2RI	AIOR2	A3R4	A7-4

NOTES:

1. SEE CHART FOR DESIGNATION OF MICROCIRCUIT COMPONENTS NOT FOUND ON SCHEMATIC DRAWING
2. 29 COMMON TO PIN 29
3. REFERENCE XDS DWG: 127815-1B

901060A. 3370

Figure 3-19. AT21 Cable Driver, Schematic Diagram

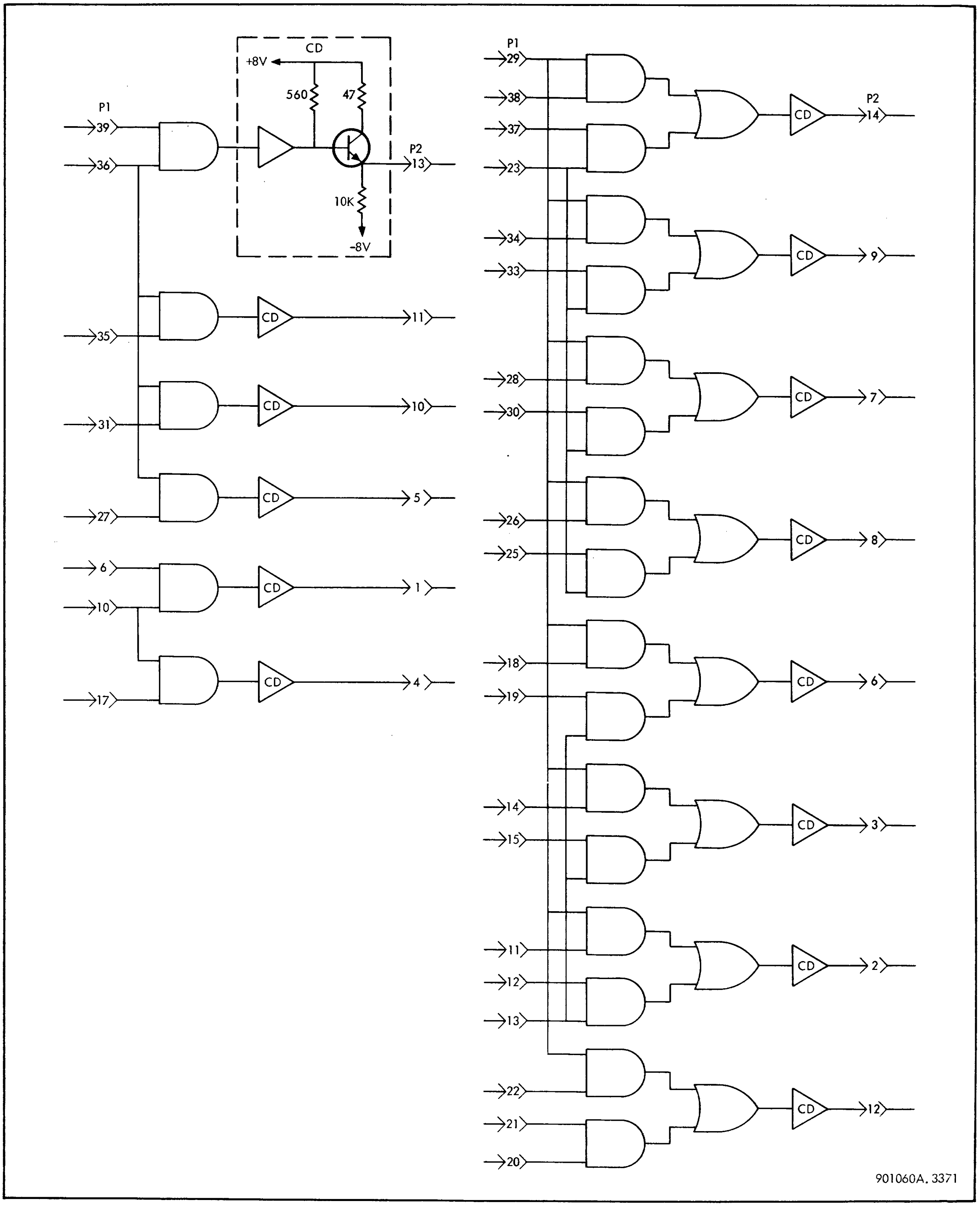


Figure 3-20. AT21 Cable Driver, Logic Diagram

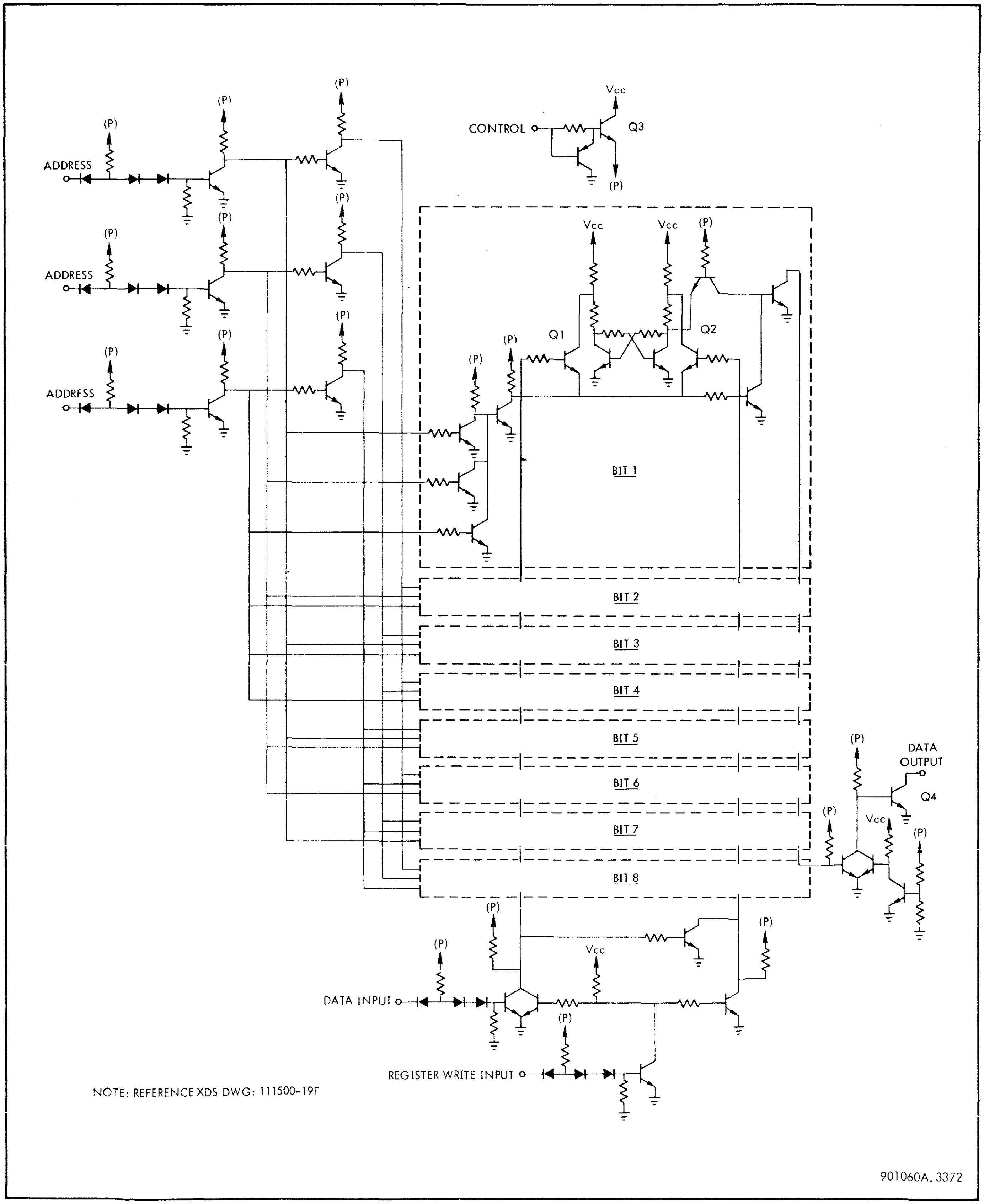


Figure 3-21. Memory Element (XDS 304), Schematic Diagram

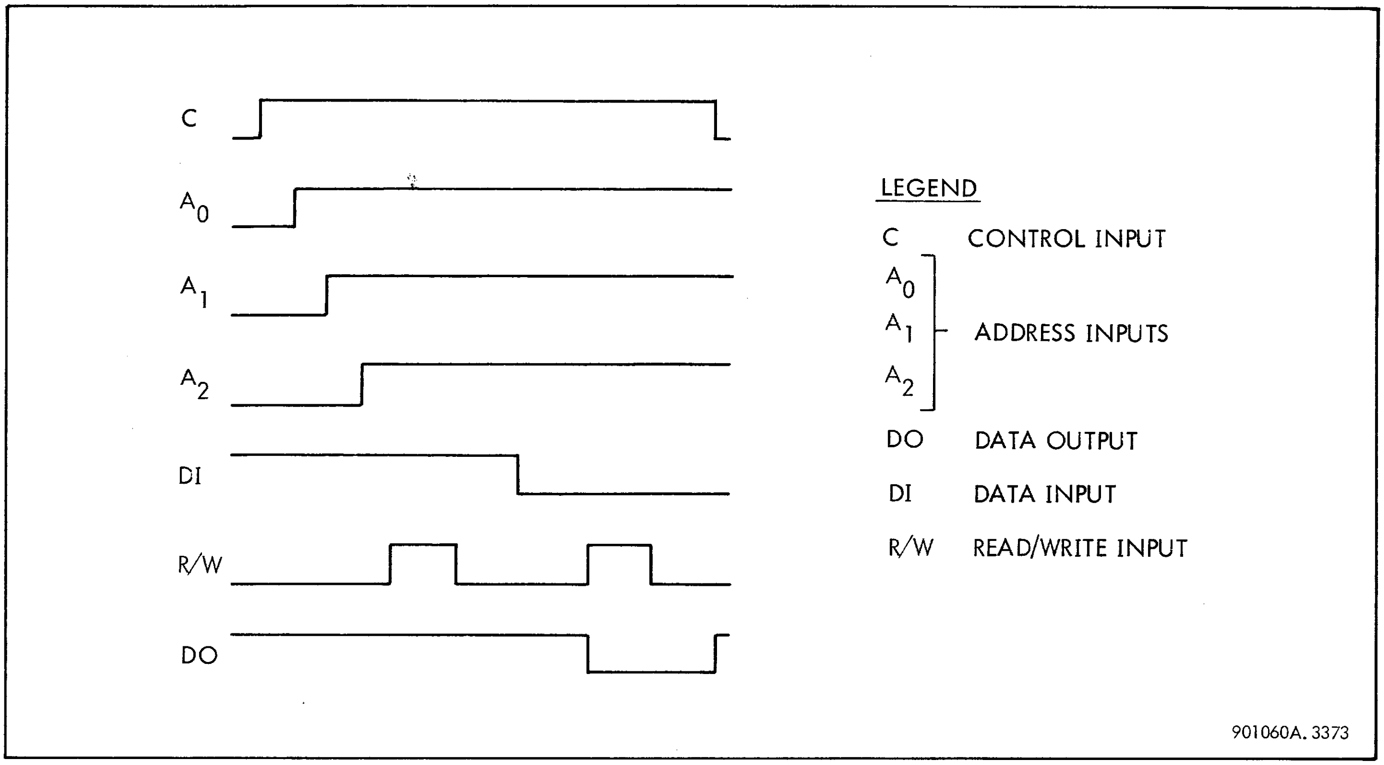


Figure 3-22. Memory Element Operation, Timing Diagram

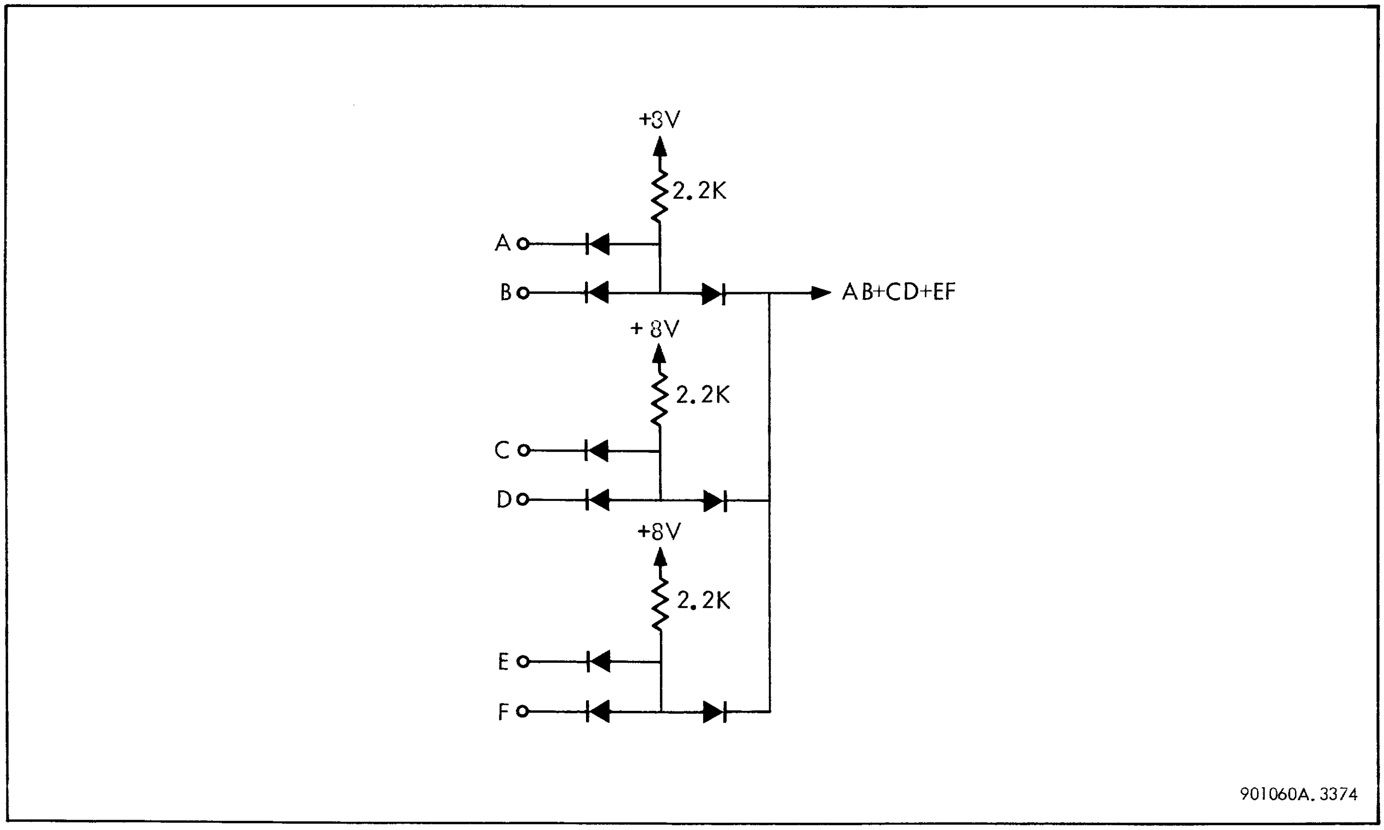


Figure 3-23. Gate Structure for Inverter (XDS 305), Typical

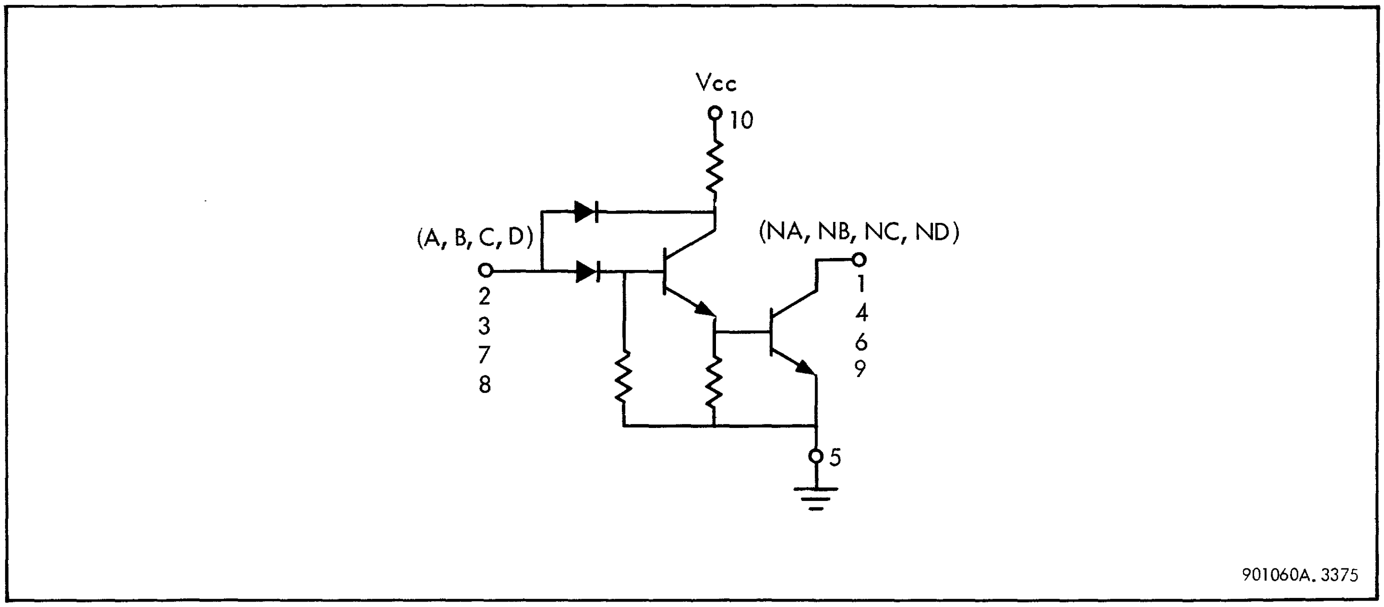


Figure 3-24. Inverter (XDS 305), Schematic Diagram

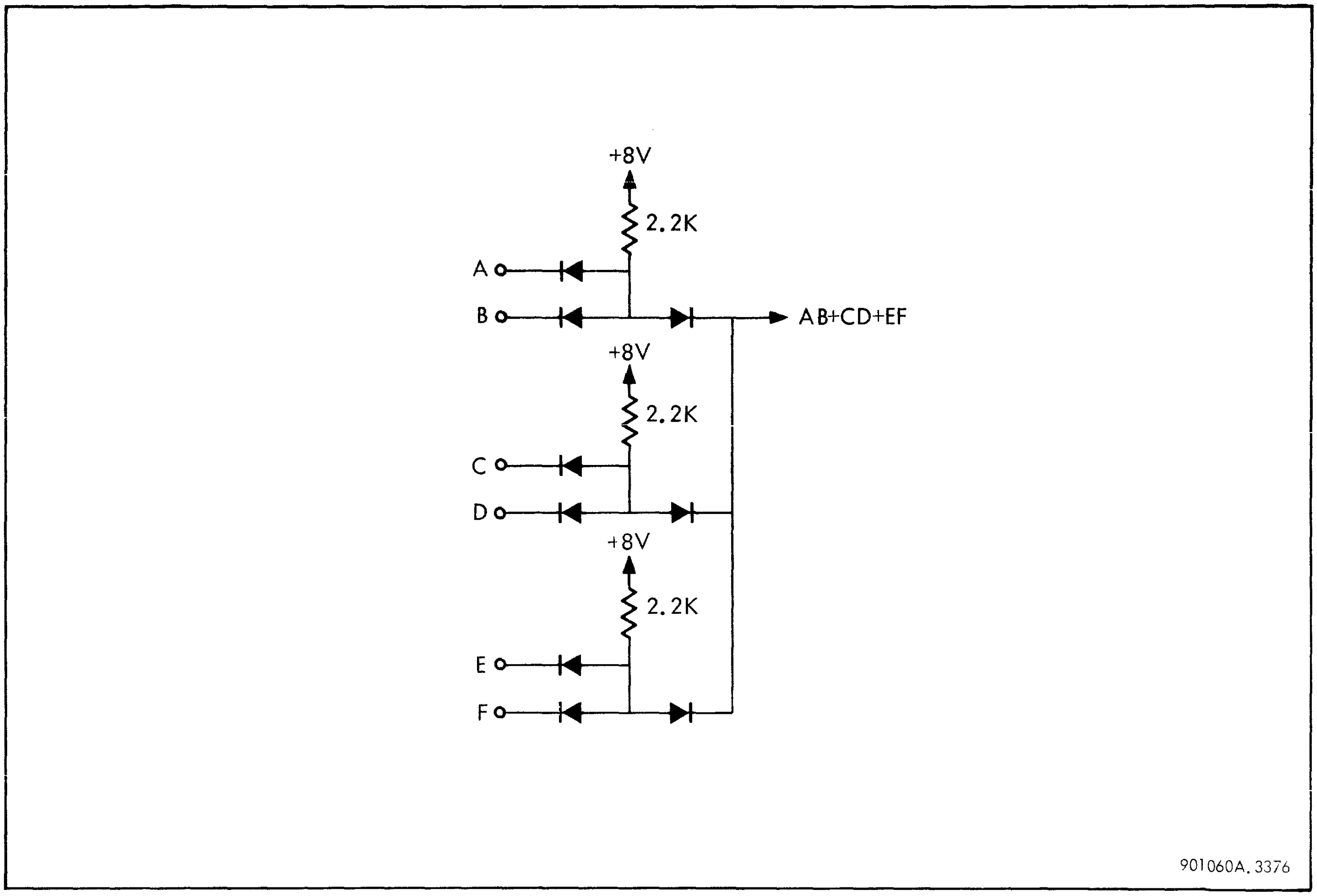


Figure 3-25. Gate Structure for Buffer (XDS 306), Typical

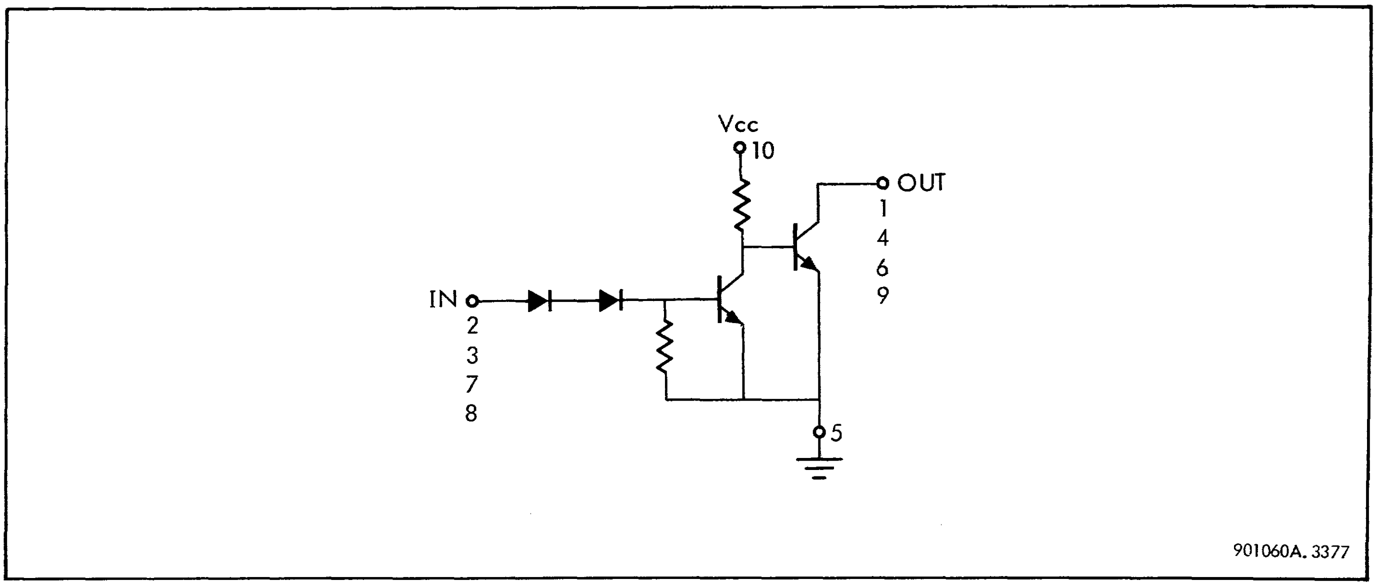


Figure 3-26. Buffer (XDS 306), Schematic Diagram

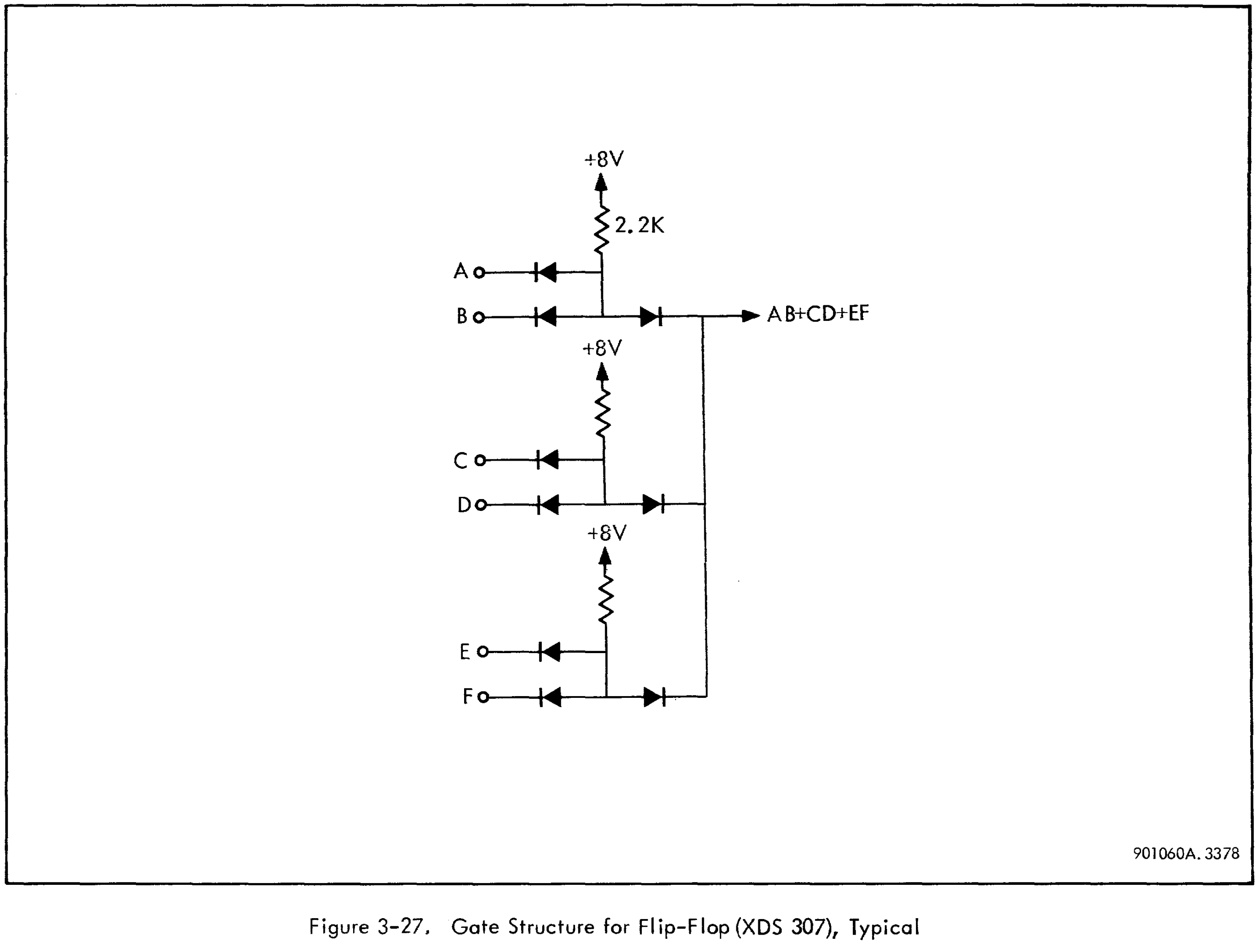
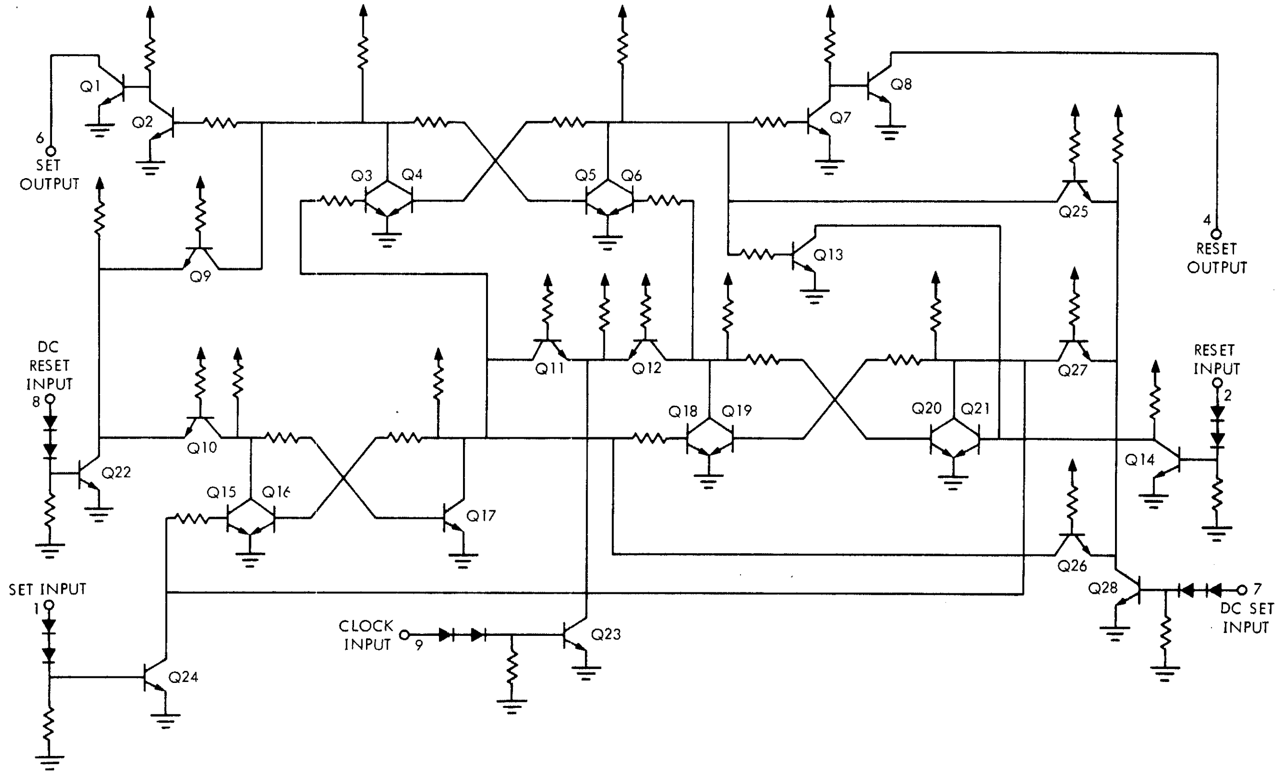


Figure 3-27. Gate Structure for Flip-Flop (XDS 307), Typical



NOTES: 1. TRANSISTORS Q9, 10, 11, 12, 25, 26, 27 SHALL HAVE DUAL COLLECTOR CONNECTIONS TO REDUCE Vce (SAT)

2. REFERENCE XDS DWG: 111503-12C

Figure 3-28. Flip-Flop (XDS 307), Schematic Diagram

901060A.3379

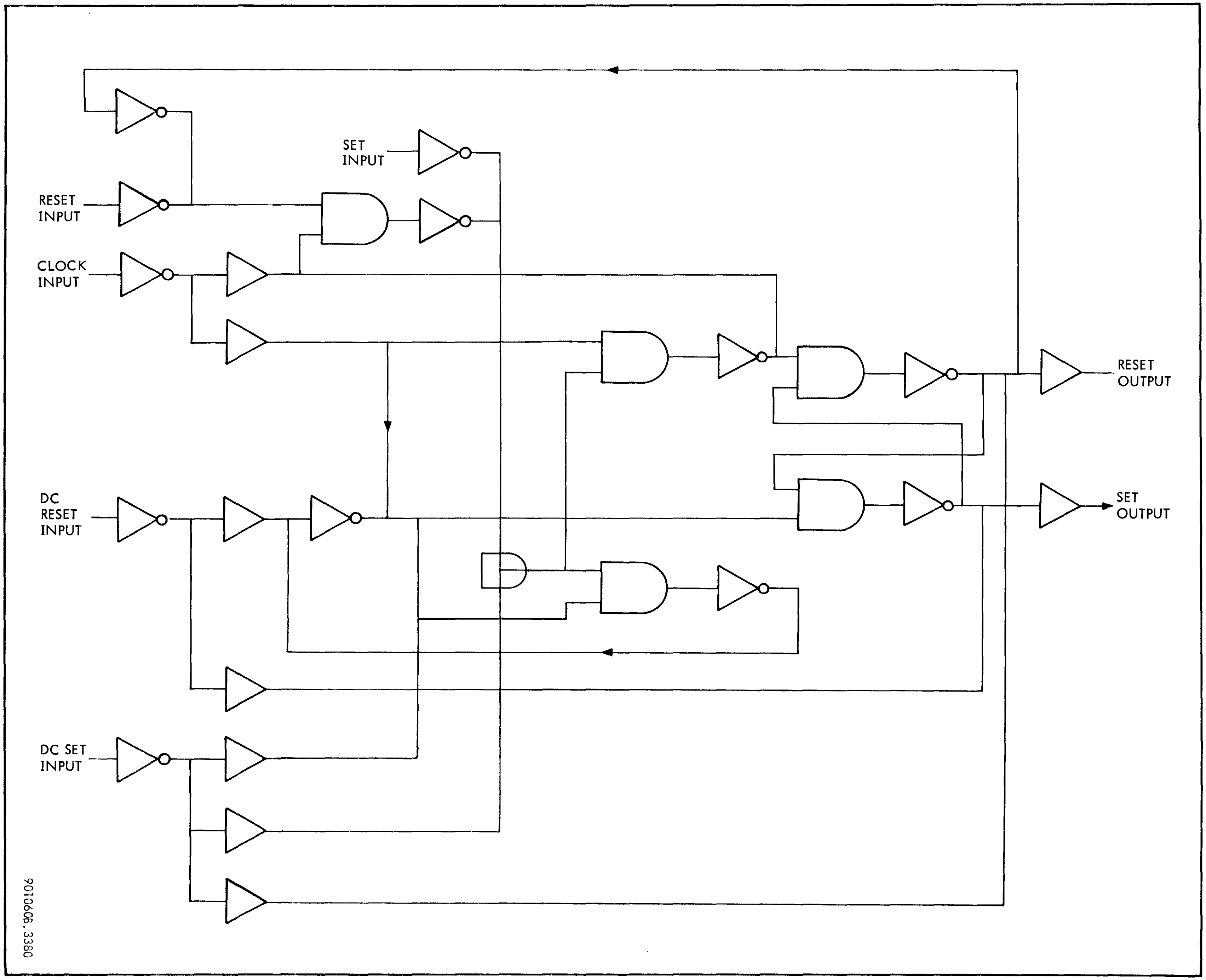
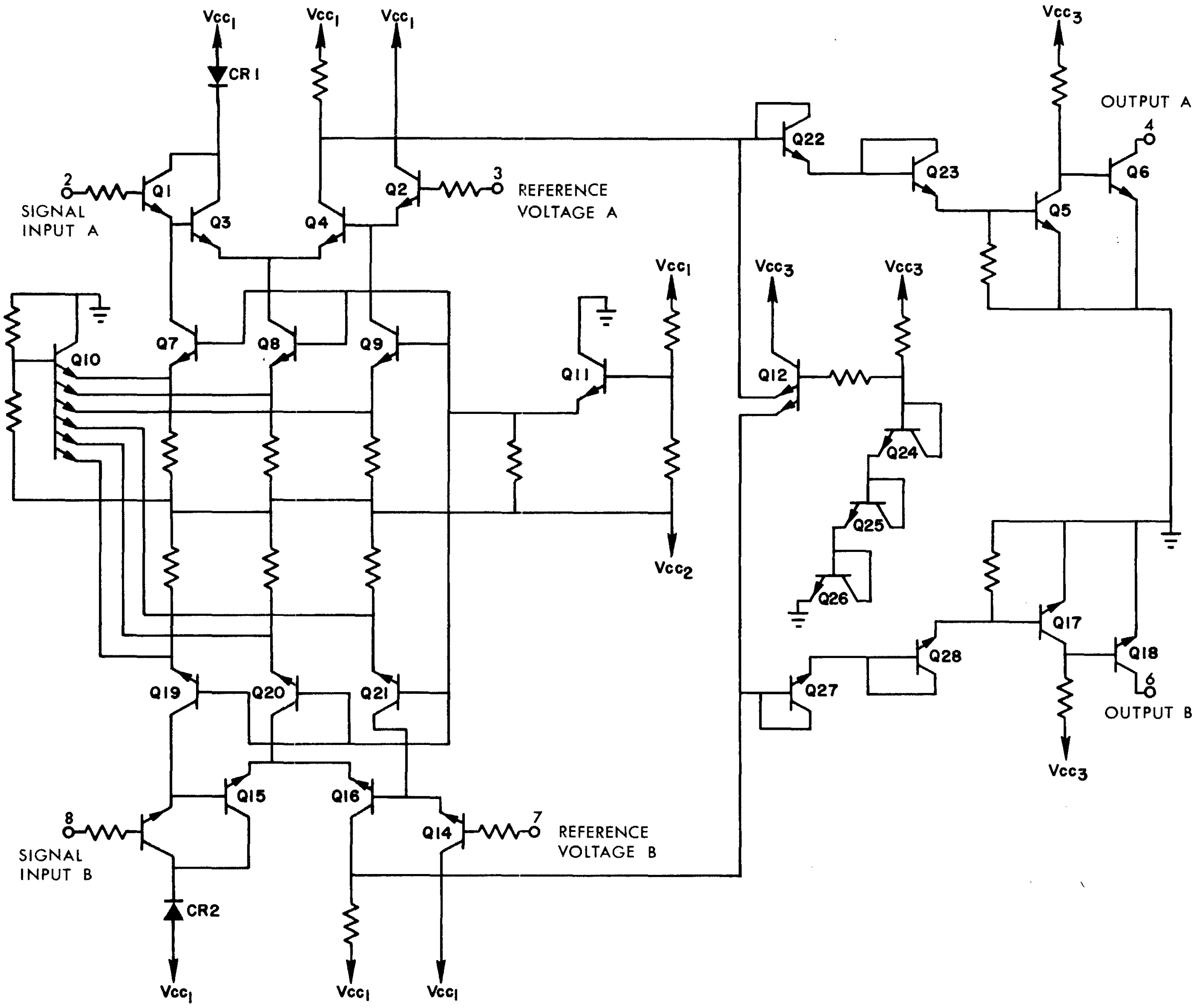


Figure 3-29. Flip-Flop ' 307), Logic Diagram



901060A.3381

Figure 3-30. Discriminator 308), Schematic Diagram

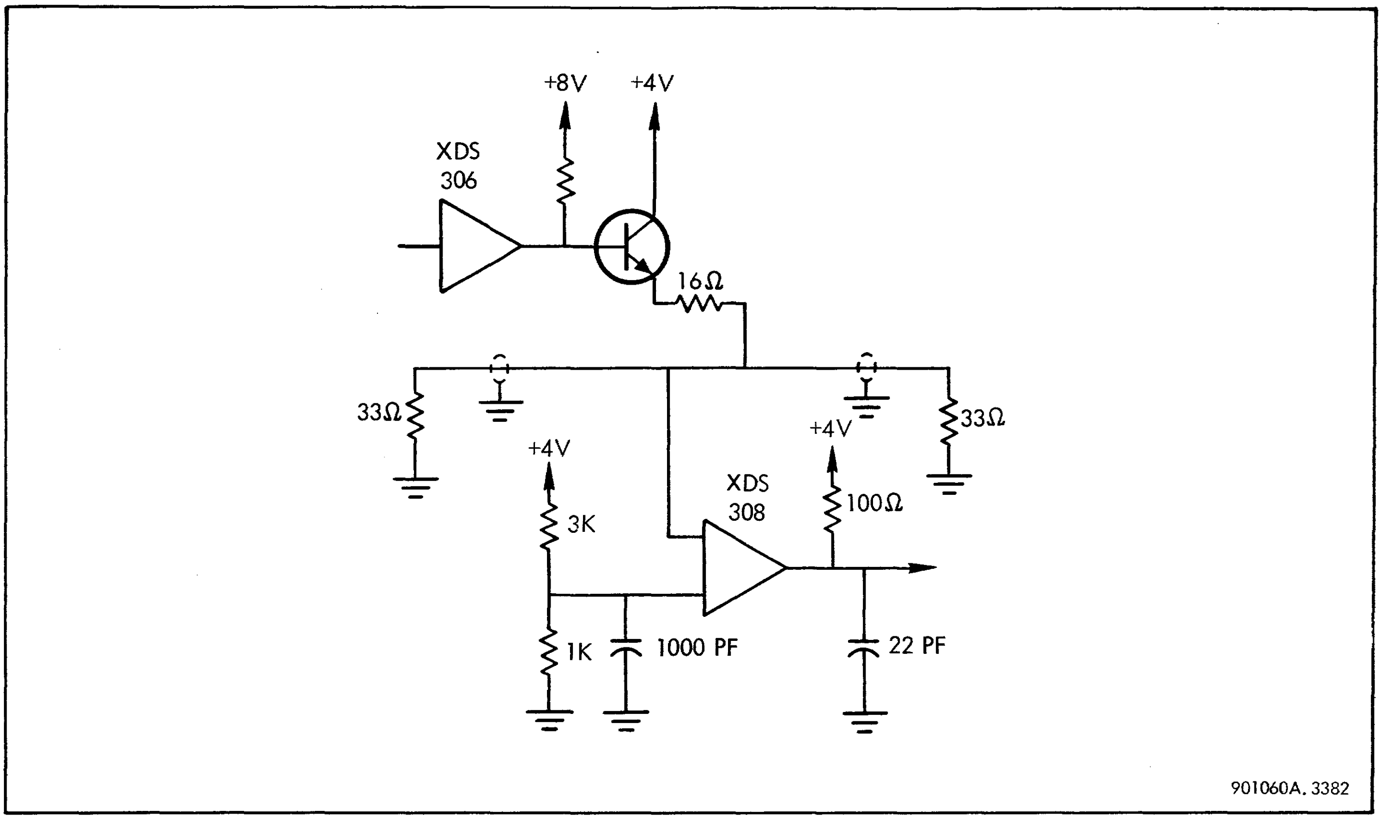


Figure 3-31. Use of Discriminator (XDS 308), Typical

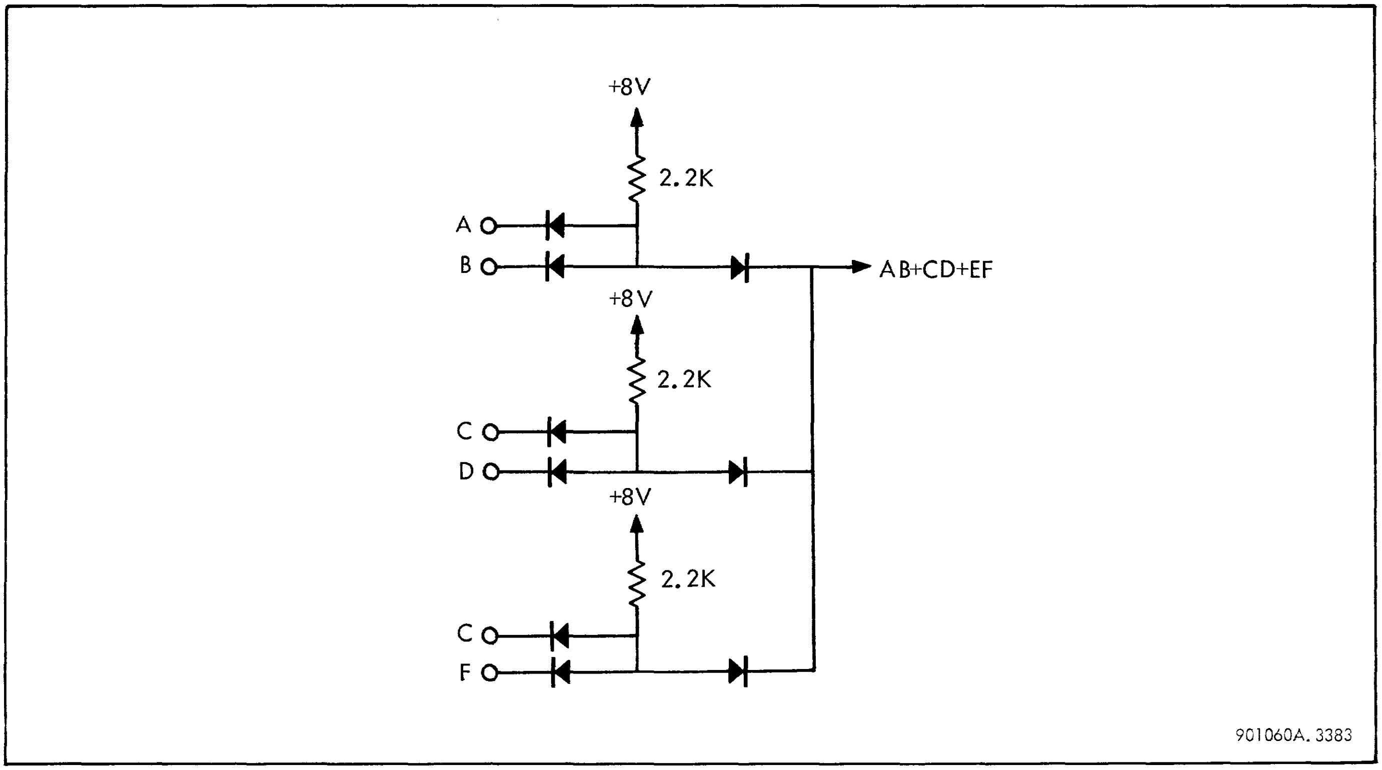
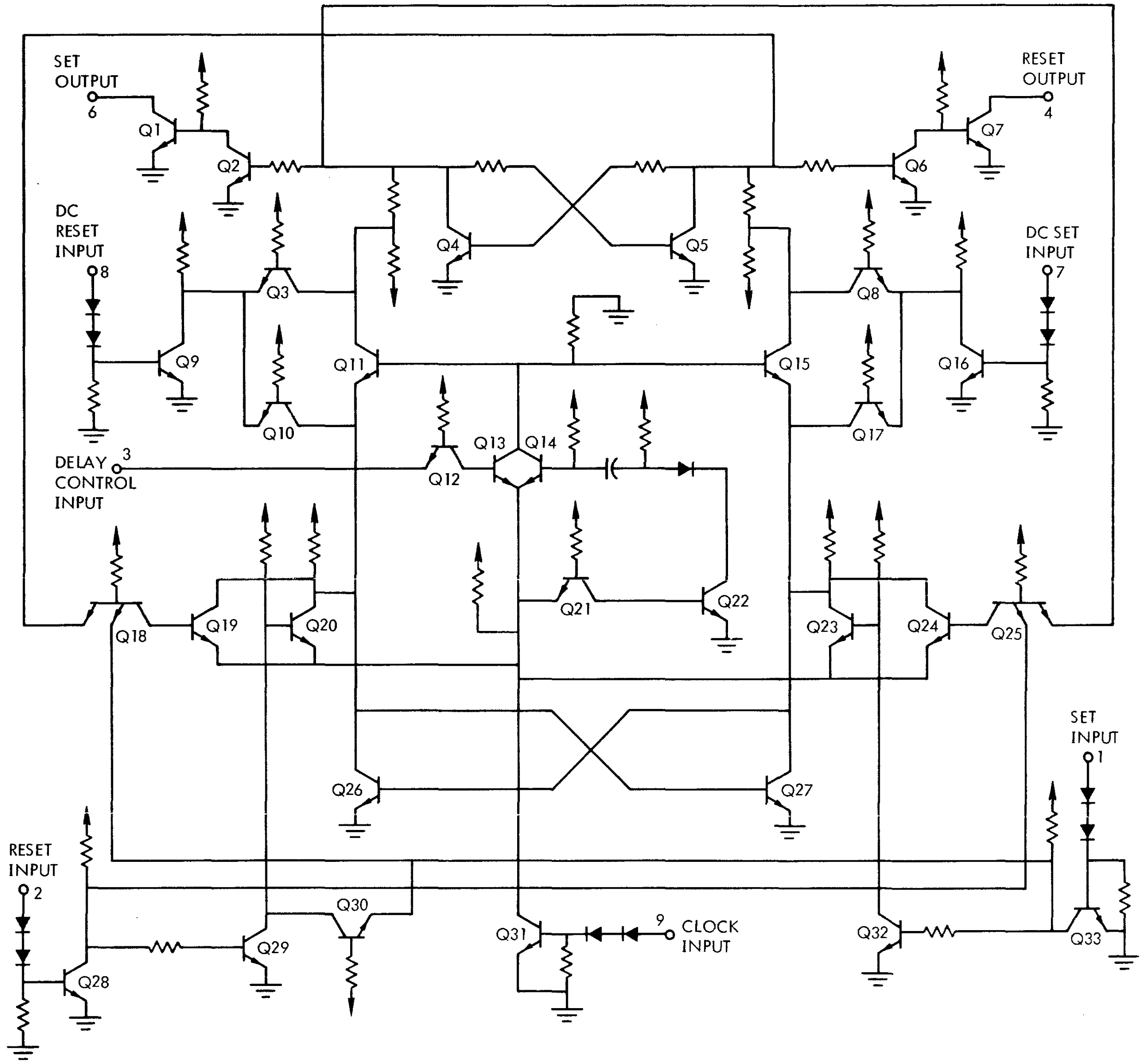


Figure 3-32. Discriminator (XDS 308) Gate Structure, Typical



901060A. 3384

Figure 3-33. Flip-Flop (XDS 311), Schematic Diagram

When the dc set input becomes one, the set output becomes one and remains one after the dc set input becomes zero. When the dc reset input becomes one, the reset output becomes one and remains one after the dc reset input becomes zero. When the dc set and dc reset inputs are one simultaneously, the state of the flip-flop is indeterminate. The logical function is independent of the delay control input, which provides an output signal delay of 80 to 200 ns after the fall of the input clock pulse. This flip-flop is functional only if the delay control input is grounded. If the input is left open, the flip-flop (XDS 311) functions exactly like the flip-flop (XDS 307). The flip-flop drives and is driven by diode-resistor AND/OR gates of the type shown in figure 3-34.

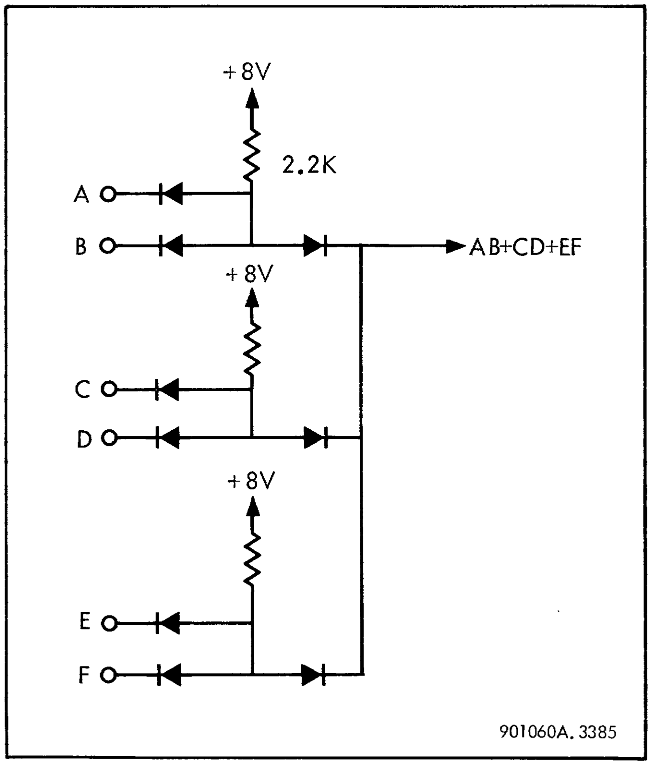


Figure 3-34. Flip-Flop (XDS 311) Gate Structure, Typical

3-39 CPU INFORMATION TRANSFER AND STORAGE

3-40 Control and Data Registers

Information transfer and storage in the CPU is performed by 12 registers, designated A, B, C, CC, CS, D, E, O, P, Q, R, and RP. Temporary storage is provided by a set of private memory registers, and all data except address information leaves the CPU via the sum bus. An adder is used in conjunction with the A-, D-, and CS-registers for calculation purposes. Figure 3-35 shows the data transfer between the registers, the sum bus, and the adder.

This description includes a diagram of each register that shows inputs to the register and the enabling signal for each input. The byte inputs are not shown separately except for the C-register diagram. For example, the transfer of the sum bus to the B-register is shown as S0-S31 into B0-B31. The enabling signal is given as BXS-0 through BXS-3. The actual enabling signals are: BXS-0 for byte 0, BXS-1 for byte 1, BXS-2 for byte 2, and BXS-3 for byte 3.

The generation of the enabling signals is shown in the set of logic diagrams that follow each register diagram. In most cases, the origin of an enabling signal is an instruction family and an instruction phase. The interpretation of the instruction family mnemonics and of other signals found in the logic diagrams is contained in the glossary of terms in this manual. Register data transfers are given detailed explanation in discussions of the individual instructions.

Enabling terms for data transfer must be connected to the reset sides of the flip-flops so that when the input is a zero, a one previously in the flip-flop may be cleared. When the set and reset sides of a flip-flop in a CPU register are made true at the same time, the set side overrides the reset side, and the flip-flop sets.

3-41. A-REGISTER. The A-register is used for the following purposes:

a. Providing an accumulator for arithmetic calculations

b. Providing an intermediate register during storage of private memory information in core memory

c. Accepting R-register information when the R-field in an instruction is used as an operand during Modify and Test Word, Modify and Test Halfword, and Modify and Test Byte instructions

d. Shifting words right and left during shift instructions

e. Shifting private memory words right for byte and halfword instructions

f. Shifting private memory words left for indexing during doubleword operation

g. Receiving E-register information during exponent operation

h. Storing count from the E-register during byte string operations

i. Receiving count information from the E-register when storing the control image for the memory map and memory protection

j. Forming words for storage in core memory location X'20' for communication with the IOP

The A-register consists of 32 register flip-flops when the floating point option is not included. The floating point option adds 25 additional flip-flops. The flip-flops have both set and reset inputs.

Inputs to the A-register, with the enabling signals for each input, are shown in figure 3-36. The inputs to the A-register floating point extension, with enabling signals, are shown in figure 3-37.

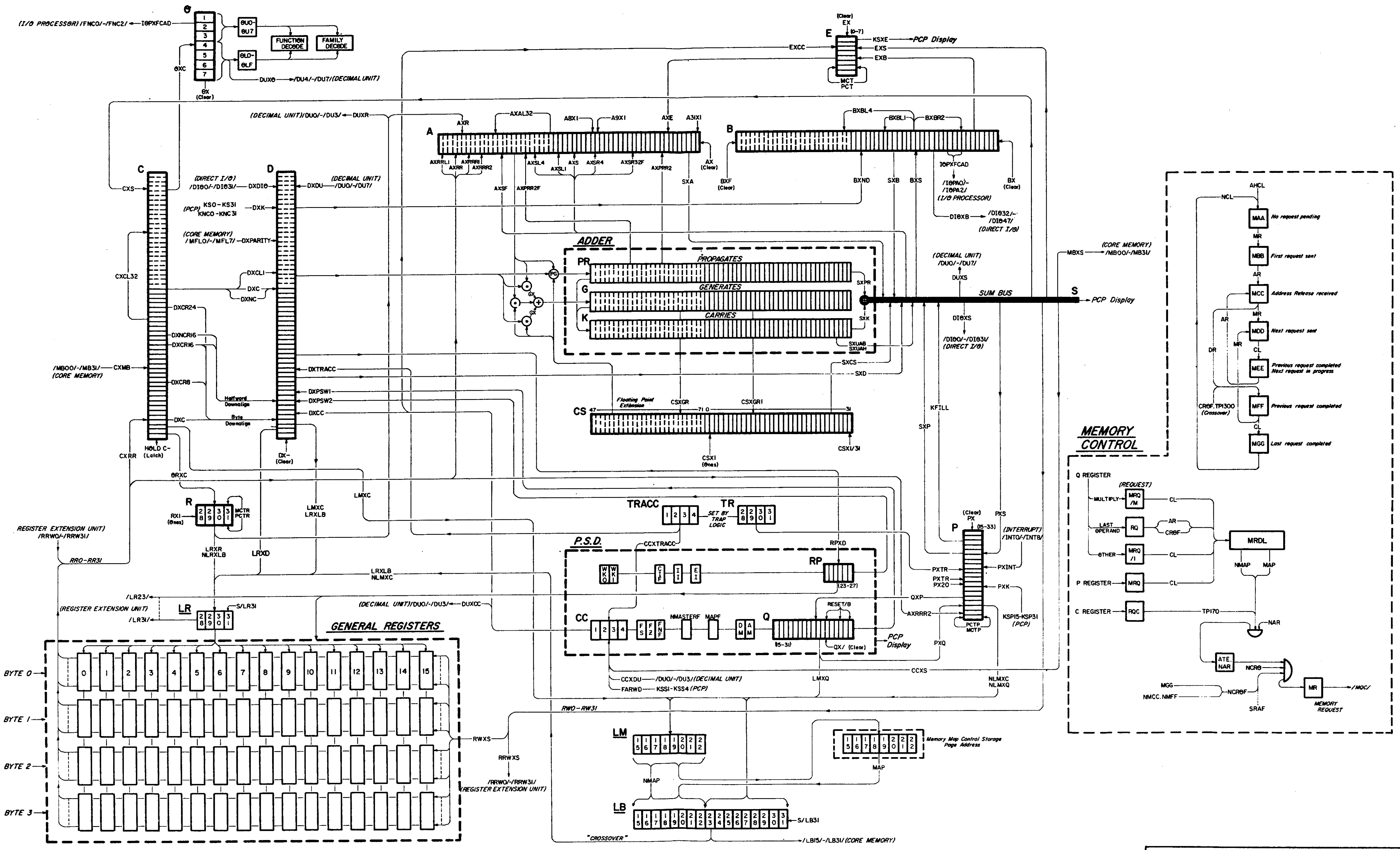


Figure 3-35. Arithmetic and Control Unit Data Transfer

901060B.3300

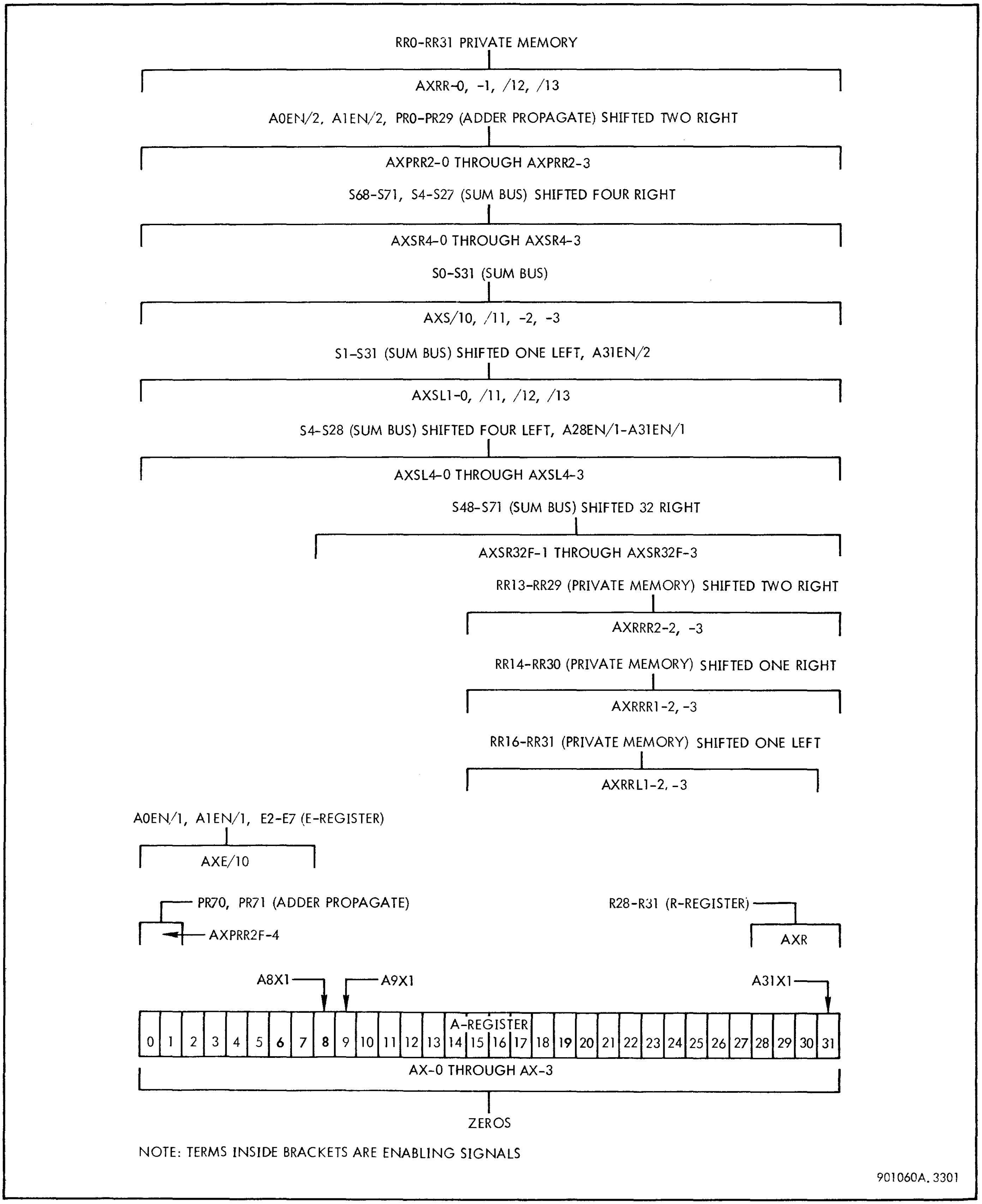

RR0-RR31 PRIVATE MEMORY
AXRR-0, -1, /12, /13
A0EN/2, A1EN/2, PR0-PR29 (ADDER PROPAGATE) SHIFTED TWO RIGHT
AXPRR2-0 THROUGH AXPRR2-3
S68-S71, S4-S27 (SUM BUS) SHIFTED FOUR RIGHT
AXSR4-0 THROUGH AXSR4-3
S0-S31 (SUM BUS)
AXS/10, /11, -2, -3
S1-S31 (SUM BUS) SHIFTED ONE LEFT, A31EN/2
AXSL1-0, /11, /12, /13
S4-S28 (SUM BUS) SHIFTED FOUR LEFT, A28EN/1-A31EN/1
AXSL4-0 THROUGH AXSL4-3
S48-S71 (SUM BUS) SHIFTED 32 RIGHT
AXSR32F-1 THROUGH AXSR32F-3
RR13-RR29 (PRIVATE MEMORY) SHIFTED TWO RIGHT
AXRRR2-2, -3
RR14-RR30 (PRIVATE MEMORY) SHIFTED ONE RIGHT
AXRRR1-2, -3
RR16-RR31 (PRIVATE MEMORY) SHIFTED ONE LEFT
AXRRL1-2, -3
A0EN/1, A1EN/1, E2-E7 (E-REGISTER)
AXE/10
PR70, PR71 (ADDER PROPAGATE)
AXPRR2F-4
R28-R31 (R-REGISTER)
AXR
A8X1
A9X1
A31X1
A-REGISTER
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
AX-0 THROUGH AX-3
ZEROS
NOTE: TERMS INSIDE BRACKETS ARE ENABLING SIGNALS
901060A. 3301

Figure 3-36. A-Register Inputs and Enabling Signals

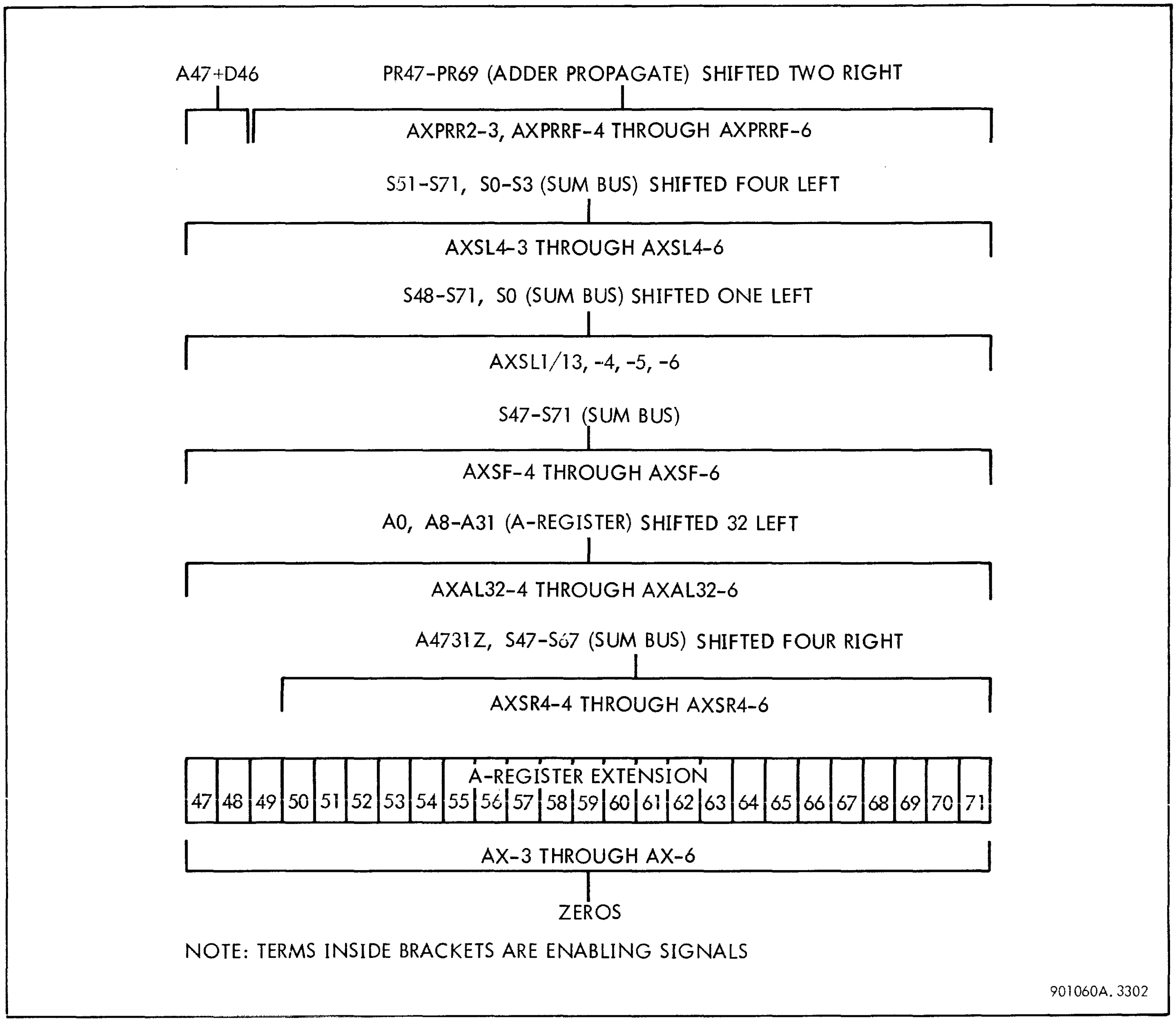

A47+D46
PR47-PR69 (ADDER PROPAGATE) SHIFTED TWO RIGHT
AXPRR2-3, AXPRRF-4 THROUGH AXPRRF-6
S51-S71, S0-S3 (SUM BUS) SHIFTED FOUR LEFT
AXSL4-3 THROUGH AXSL4-6
S48-S71, S0 (SUM BUS) SHIFTED ONE LEFT
AXSL1/13, -4, -5, -6
S47-S71 (SUM BUS)
AXSF-4 THROUGH AXSF-6
A0, A8-A31 (A-REGISTER) SHIFTED 32 LEFT
AXAL32-4 THROUGH AXAL32-6
A4731Z, S47-S67 (SUM BUS) SHIFTED FOUR RIGHT
AXSR4-4 THROUGH AXSR4-6
A-REGISTER EXTENSION
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
AX-3 THROUGH AX-6
ZEROS
NOTE: TERMS INSIDE BRACKETS ARE ENABLING SIGNALS
901060A. 3302

Figure 3-37. A-Register Extension Inputs and Enabling Signals

3-42 B-REGISTER. The B-register is used as follows:

a. To operate in conjunction with the D-register and the sum bus for arithmetic calculations

b. To indicate shift left and right for shift instructions

c. To save byte address for the P-register during byte string instructions

d. To contain the upper half of doubleword during doubleword instructions

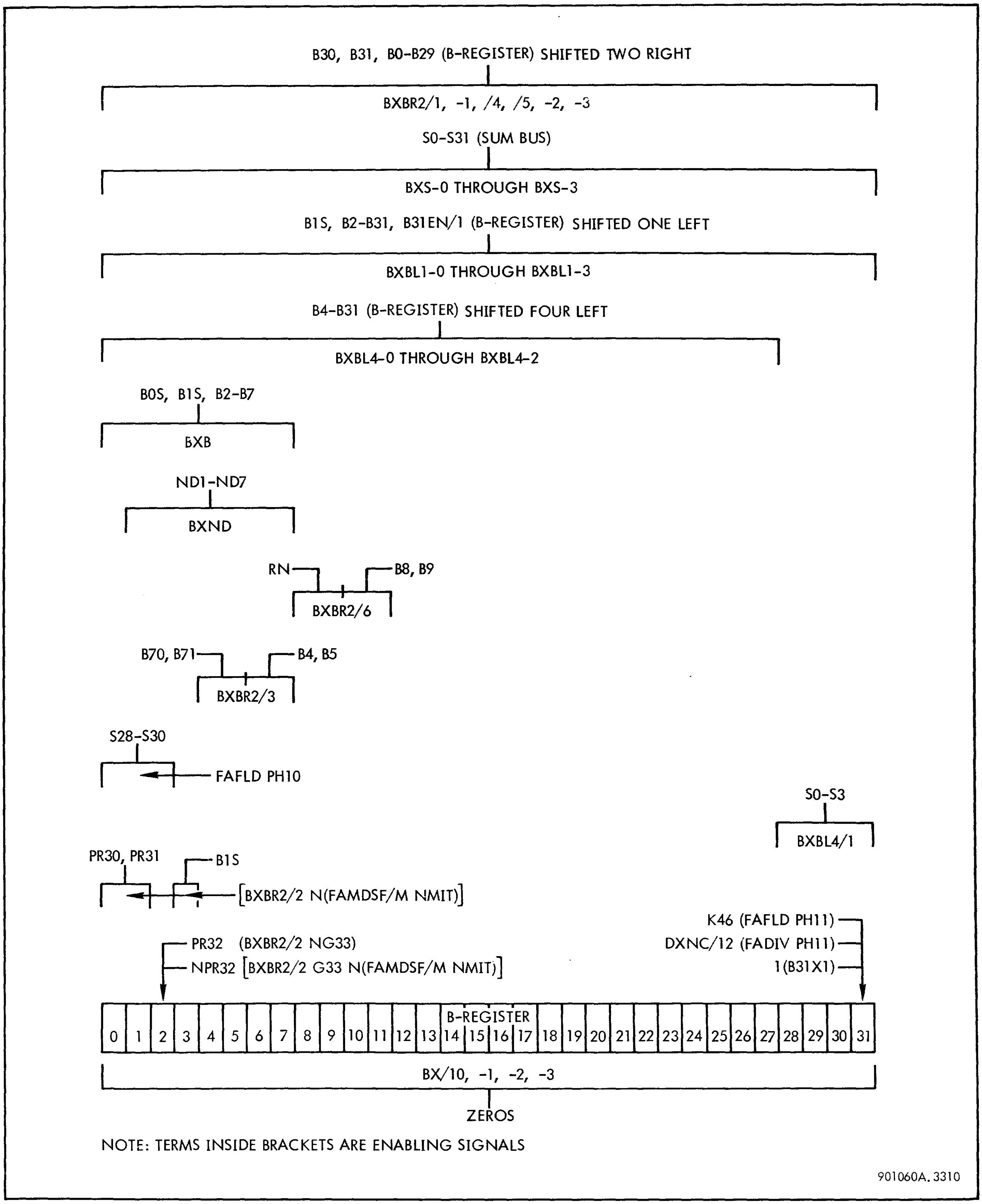
e. To store address during interrupt operation

f. To store exponents

The standard B-register consists of 32 register flip-flops. With the floating point option, 25 additional flip-flops are added. The flip-flops have both set and reset inputs.

Inputs to the B-register, with the enabling signals for each input, are shown in figure 3-38. The inputs to the B-register floating point extension are shown in figure 3-39.

3-43 C-REGISTER. The C-register serves as an instruction register and is used in calculations in conjunction with the A-, D-, and CS-registers. All core memory information enters the CPU via the C-register, and part of the private memory information also uses this path. Inputs for calculation purposes are received from the sum bus. Opcodes and private memory addresses are sent from the C-register to the O- and



NOTE: TERMS INSIDE BRACKETS ARE ENABLING SIGNALS

901060A.3310

Figure 3-38. B-Register Inputs and Enabling Signals

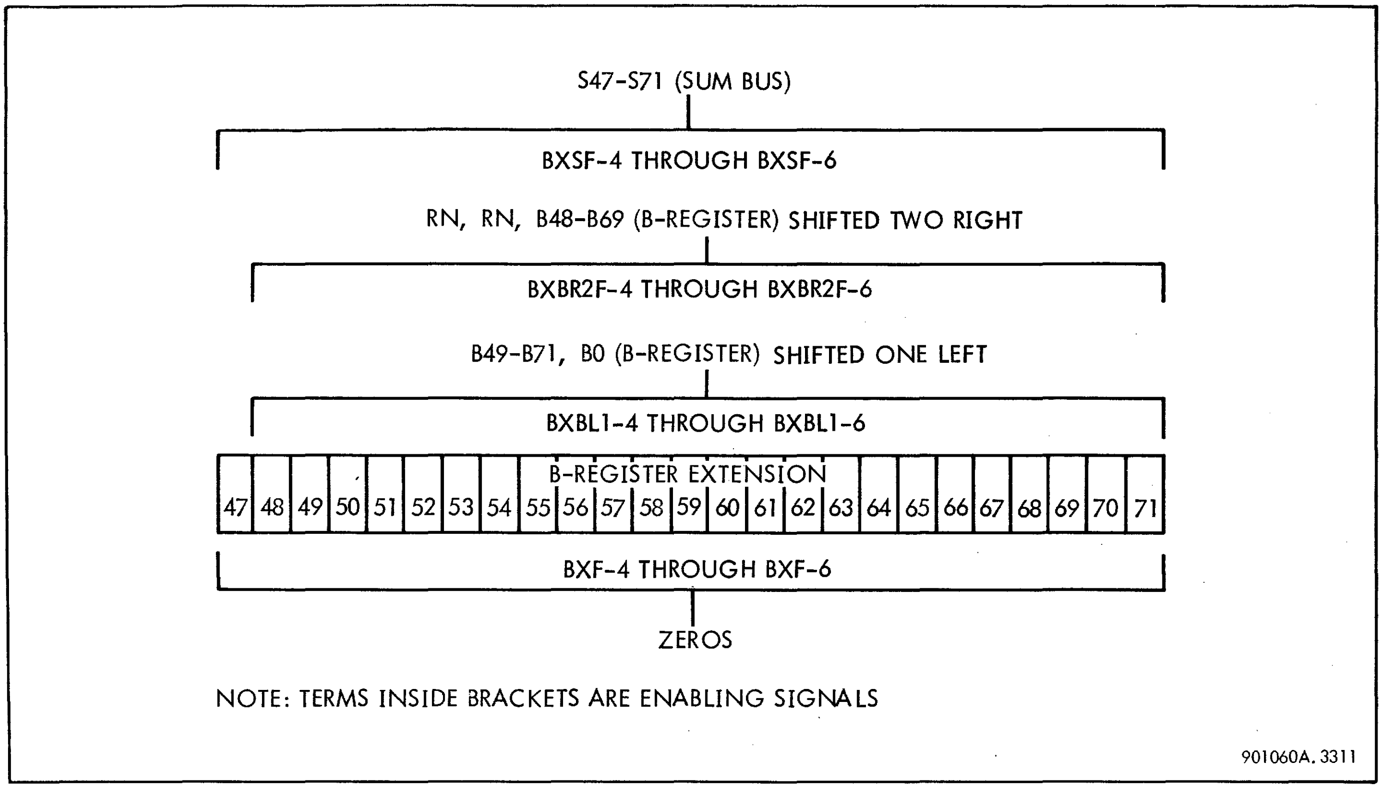


Figure 3-39. B-Register Extension Inputs and Enabling Signals

R-registers, respectively. Core memory addresses go directly from the C-register to the memory address lines. All other outputs from the C-register are taken to the D-register. In doubleword operations, the C-register holds the 32 high-order bits of the effective doubleword.

The C-register is unique among the CPU registers in that its storage circuits are made of buffered latches instead of flip-flops. In the logic equations, these buffered latches are referenced as buffer flip-flops, designated by symbol FB.

The operation of a buffered latch, using bit one of the C-register as an example, is illustrated in figure 3-40. Other bits are the same, except that with bytes 1, 2, and 3 the dash number after the HOLDC term and the number after the slash in the enabling terms is 1, 2, or 3 instead of 0.

When the C-register is loaded from private memory, the sum bus, or the memory bus, one of the three lower inputs to the OR gate (see figure 3-40) goes true, and buffer output C1 is driven true. The C1 output is fed back to the input of an AND gate containing holding term HOLDC-0. As long as HOLDC-0 is true, C1 contains a logical 1, even after the qualifying signal has dropped. A zero is placed in C1 when a false output from one of the inverters causes HOLDC-0 to go false.

A timing diagram for loading C1 from the sum bus is shown in figure 3-41. The generation of enabling signals CXS/0 through CXS/6 is shown in figure 3-42. A low N(S/CXS) signal is generated at the output of an AND gate whenever a combination of instruction logic and phasing indicates that a transfer from the sum bus to the C-register is to be performed. When ac clock signal occurs, CXSE goes true and is latched by feedback because CXS is false. After a 45-ns delay, CXSE/D rises, causing CXS to go true. Signal CXS is latched by feedback to the AND gate which contains DCH, true in its inactive state. Forty-five nanoseconds later, time delay output CXS/D causes CXS/0 to rise, and a one is gated into C1 by S1 and CXS/0.

When a given time interval has elapsed after the occurrence of the ac clock, a pulse traveling down the clock generator delay line causes DCH to go false for 40 ns. The CXS latch is then released. Dropping CXS allows HOLDC-0 to go true, and the bit in C1 is latched until a subsequent signal drives HOLDC-0 false. Because of circuit delays between HOLDC-0 and CXS/0, CXS/0 remains true for a short time after HOLDC-0 rises. The overlap ensures that the latch will hold while the signals are changing.

Figure 4-43 shows all possible inputs to the C-register and the enabling signal for each input. Signal CX/1 clears the C-register; CXRR transfers data from private memory; CXMB transfers data from the memory bus; and CXS transfers data from the sum bus. Signal CXCL32 moves data into bits 47 through 71 during floating point operation. Figure 3-44 is a logic diagram showing the generation of the enabling signals.

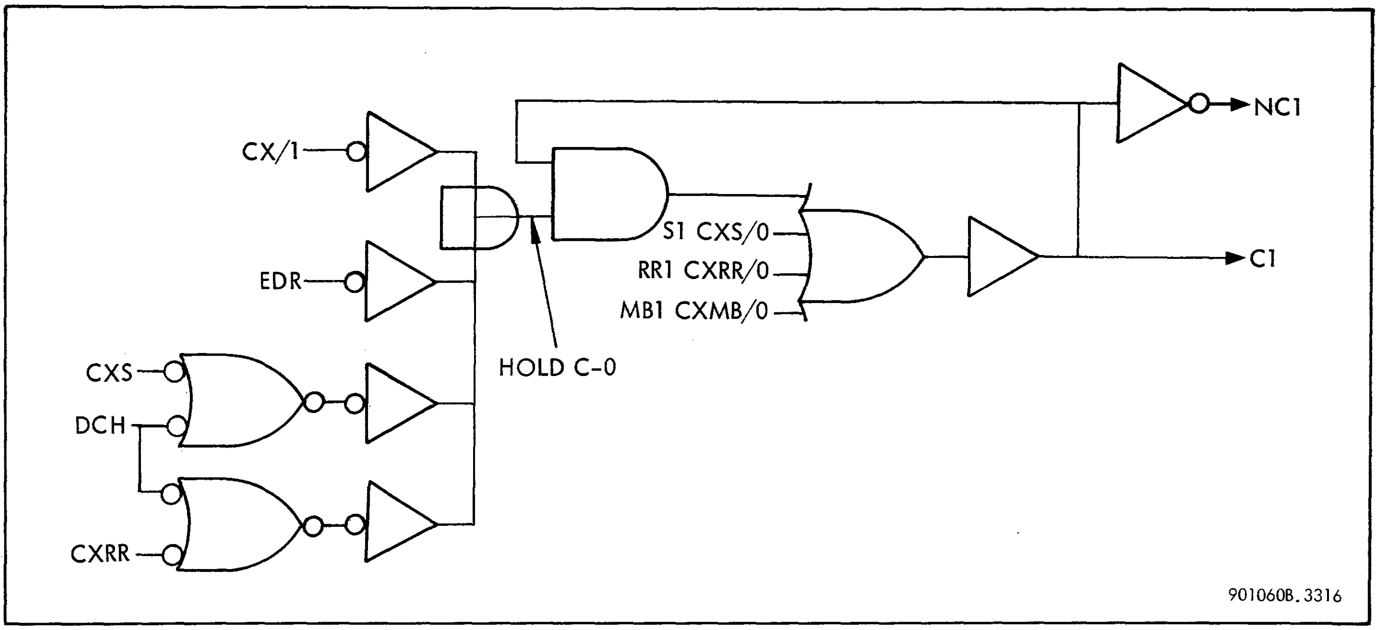


Figure 3-40. C-Register Bit 1, Simplified Logic Diagram

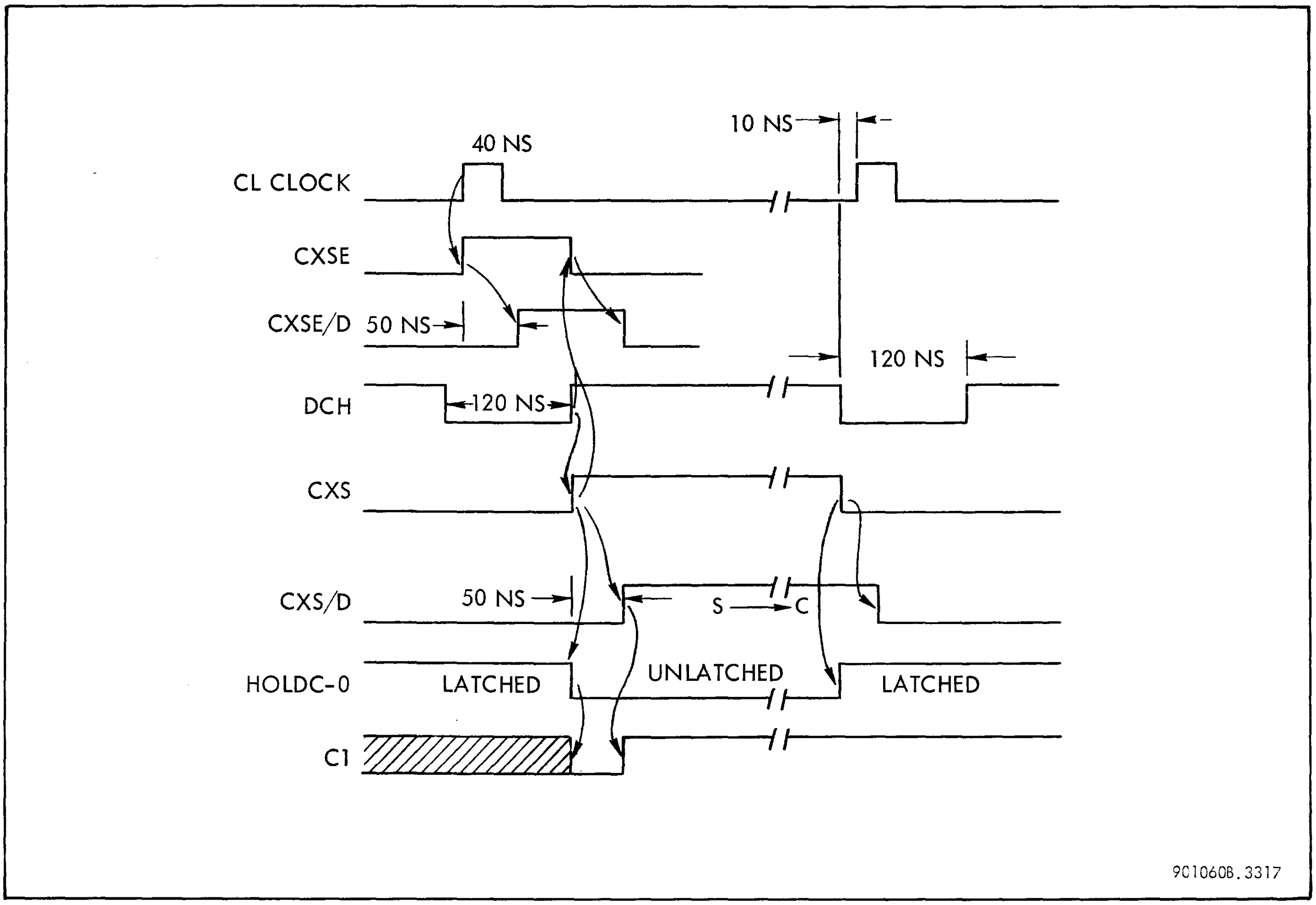


Figure 3-41. Loading C1 from Sum Bus, Timing Diagram

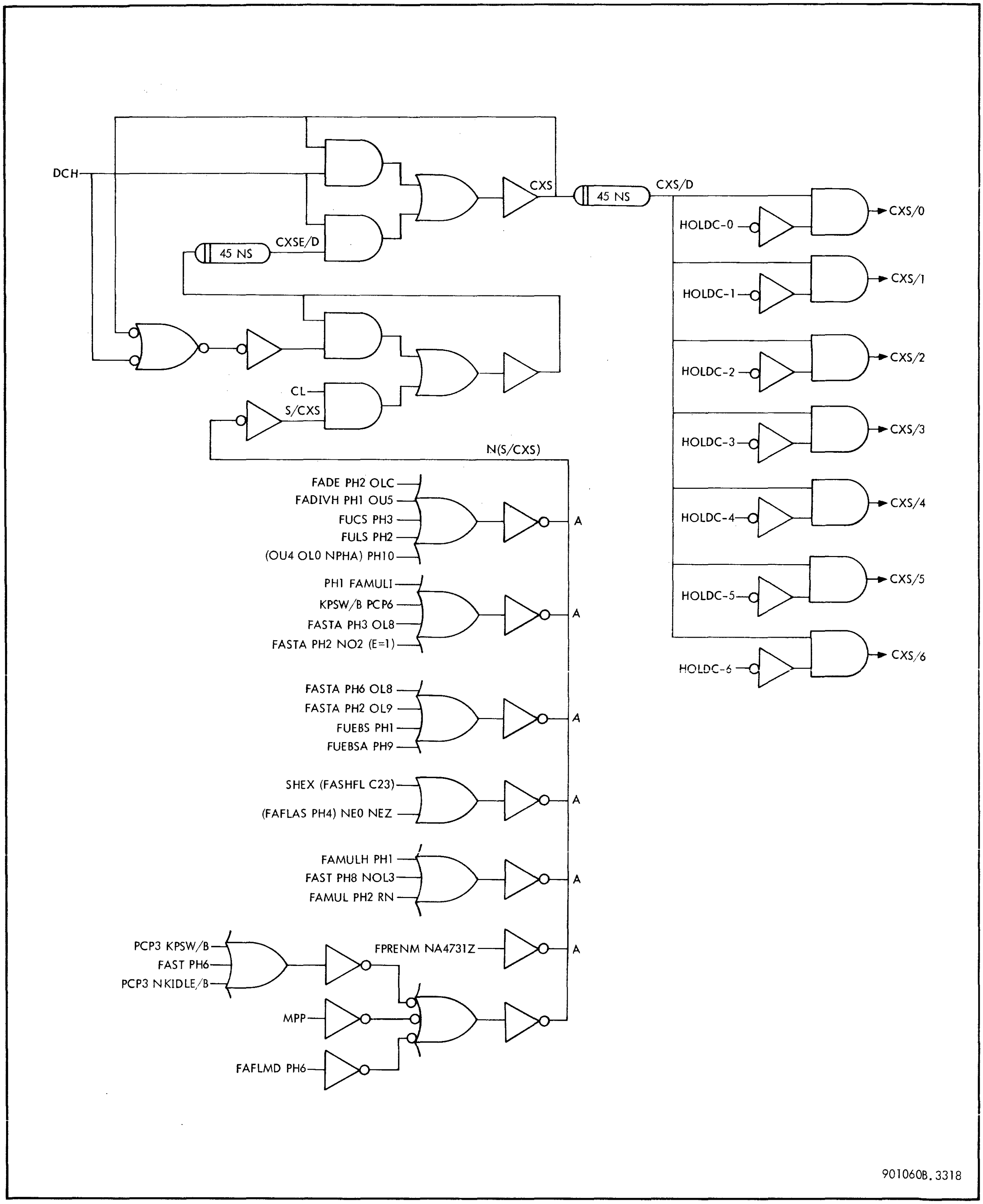
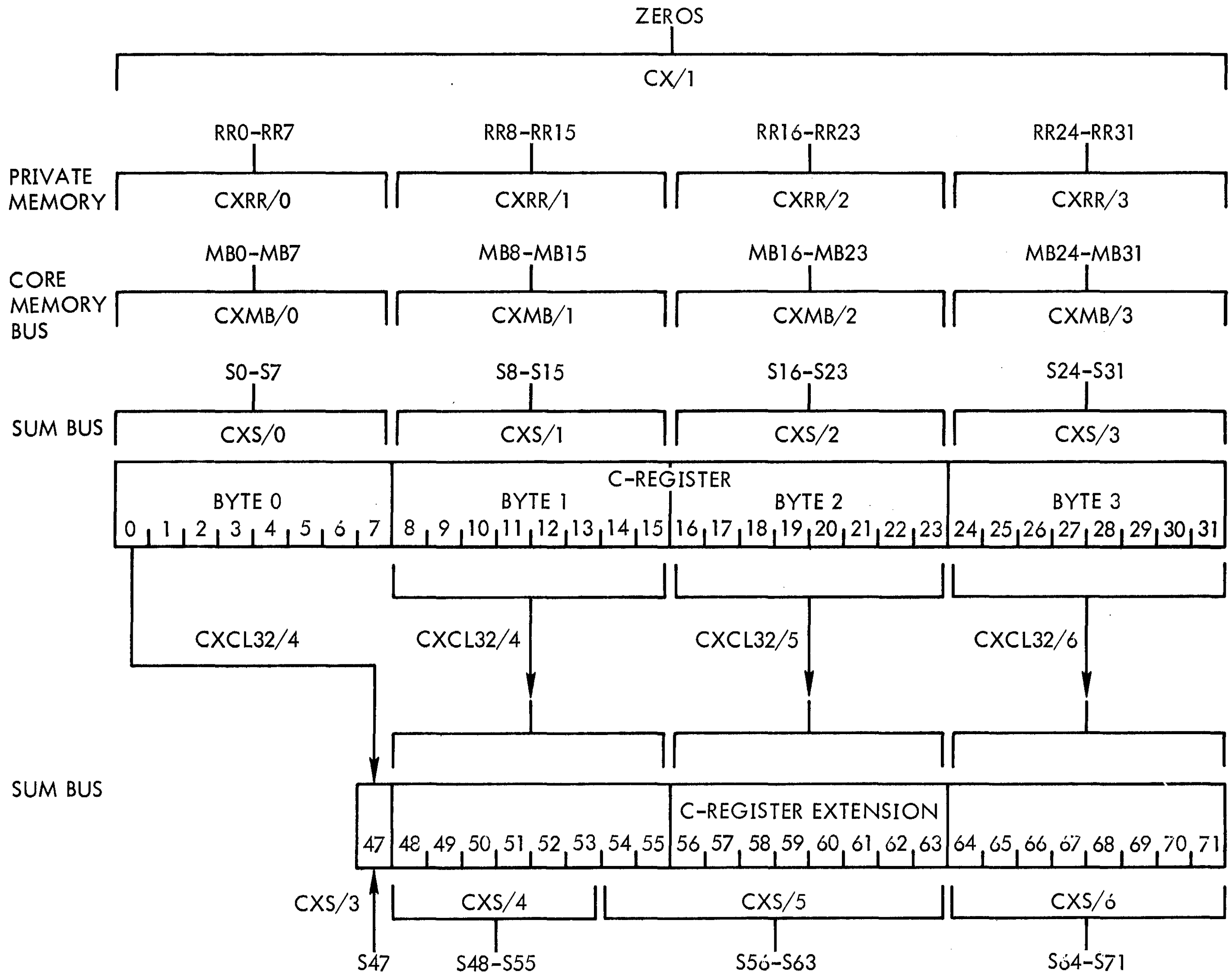


Figure 3-42. CXS Enabling Signal Generation



NOTE: TERMS INSIDE BRACKETS AND BESIDE LONG
ARROWS ARE ENABLING SIGNALS

901060A.3319

Figure 3-43. C-Register Inputs and Enabling Signals

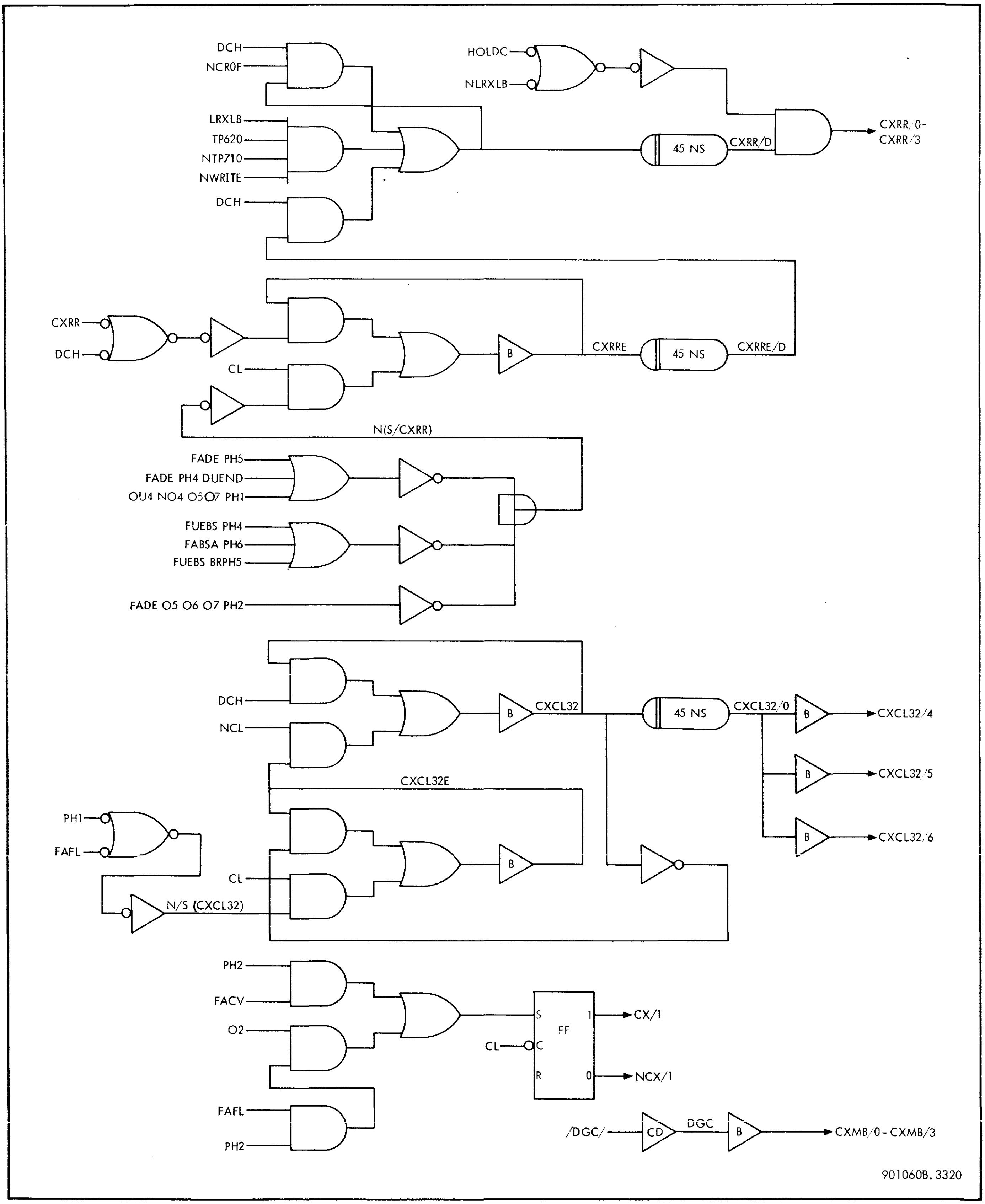


Figure 3-44. C-Register Enabling Signal Generation

Bits 46 through 71 of the C-register are included only when the CPU contains the floating point option.

3-44 CONDITION CODE REGISTER. The condition code register contains a four-bit code that indicates the nature of the results of an instruction. The significance of the condition code bits depends on the instruction just executed. The condition code register is also used as a control register during execution of Load Multiple, Store Multiple, Edit Byte String, Push Multiple, and Pull Multiple instructions. The outputs of the condition code register are fed to the D-register for storage in memory as part of the program status doubleword. The CC-register is part of the program status doubleword. The condition code register consists of four flip-flops. Common inputs to all four flip-flops, with the enabling signal for each input, are shown in figure 3-45.

3-45 CS-REGISTER. The carry save (CS) register is used for the following purposes:

a. To save the carry bits in multiplication

b. To help form the two's complement of the data in the A- or D-register

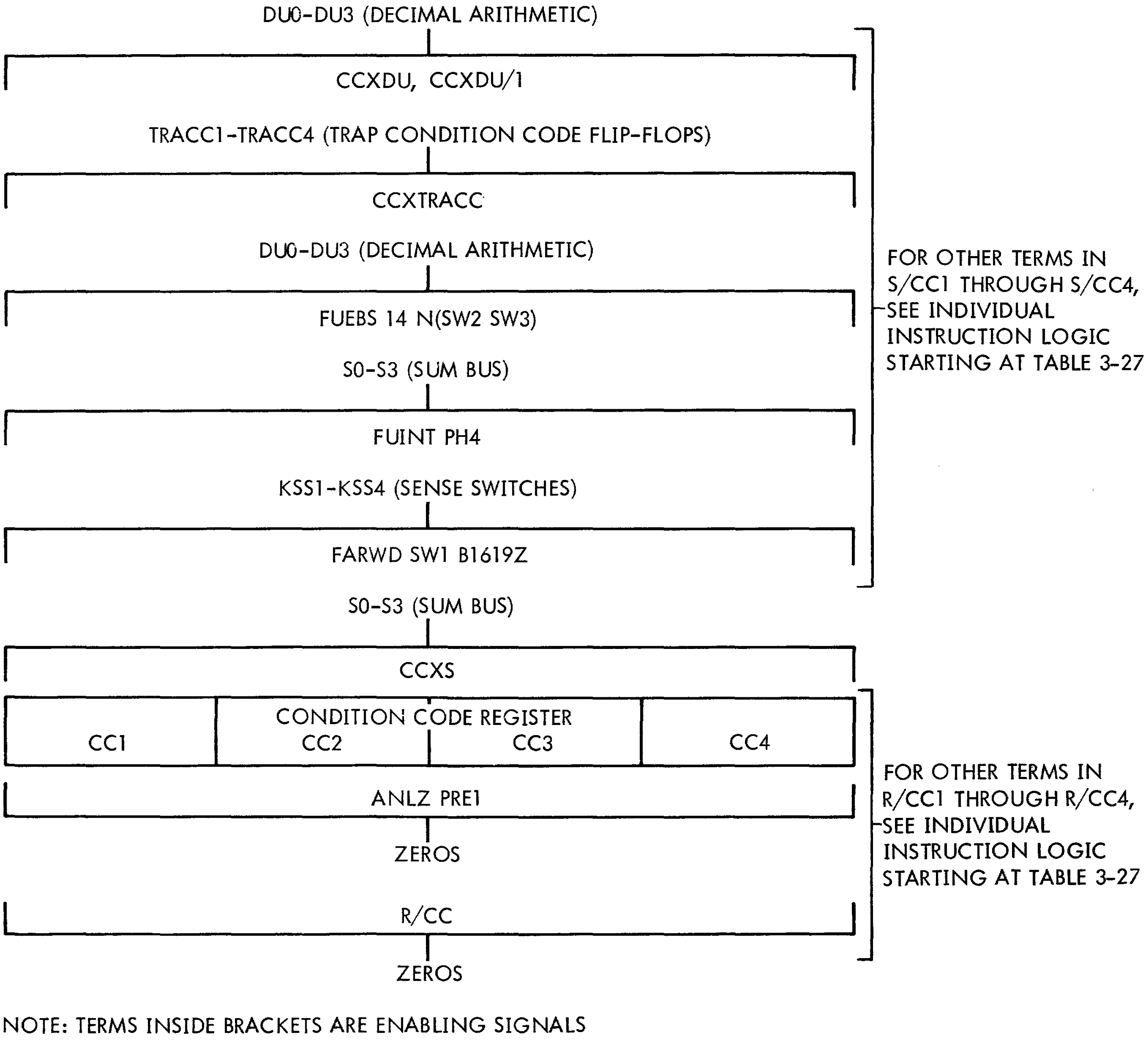

DU0-DU3 (DECIMAL ARITHMETIC)
CCXDU, CCXDU/1
TRACC1-TRACC4 (TRAP CONDITION CODE FLIP-FLOPS)
CCXTRACC
DU0-DU3 (DECIMAL ARITHMETIC)
FUEBS 14 N(SW2 SW3)
S0-S3 (SUM BUS)
FUINT PH4
KSS1-KSS4 (SENSE SWITCHES)
FARWD SW1 B1619Z
S0-S3 (SUM BUS)
CCXS
CONDITION CODE REGISTER
CC1
CC2
CC3
CC4
ANLZ PRE1
ZEROS
R/CC
ZEROS
FOR OTHER TERMS IN S/CC1 THROUGH S/CC4, SEE INDIVIDUAL INSTRUCTION LOGIC STARTING AT TABLE 3-27
FOR OTHER TERMS IN R/CC1 THROUGH R/CC4, SEE INDIVIDUAL INSTRUCTION LOGIC STARTING AT TABLE 3-27
NOTE: TERMS INSIDE BRACKETS ARE ENABLING SIGNALS
901060A. 3321

Figure 3-45. Condition Code Register Inputs and Enabling Signals

c. To extend the sign of the R-field in the instruction when this field is used as a number

Only one bit (CS31) in the CS-register has a reset term. The reset sides of all the other flip-flops are connected to an ac-clock signal. These flip-flops reset at the first clock pulse after the setting term for the flip-flop has dropped.

The standard CS-register consists of 34 flip-flops, CS0 through CS33. For the floating point option, 25 more flip-flops, CS47 through CS71, are added. The inputs to the register are generate terms from the adder, designated G, and ones. Outputs are taken to the adder and to the sum bus. A block diagram of the CS-register with inputs is shown in figure 3-46.

3-46 D-REGISTER. The D-register is used for the following purposes:

a. To enter information from the data keys on the processor control panel

b. To receive data from the decimal arithmetic option

c. To receive direct input from external equipment by way of the read/write direct data lines

d. To form the program status doubleword for storage in memory

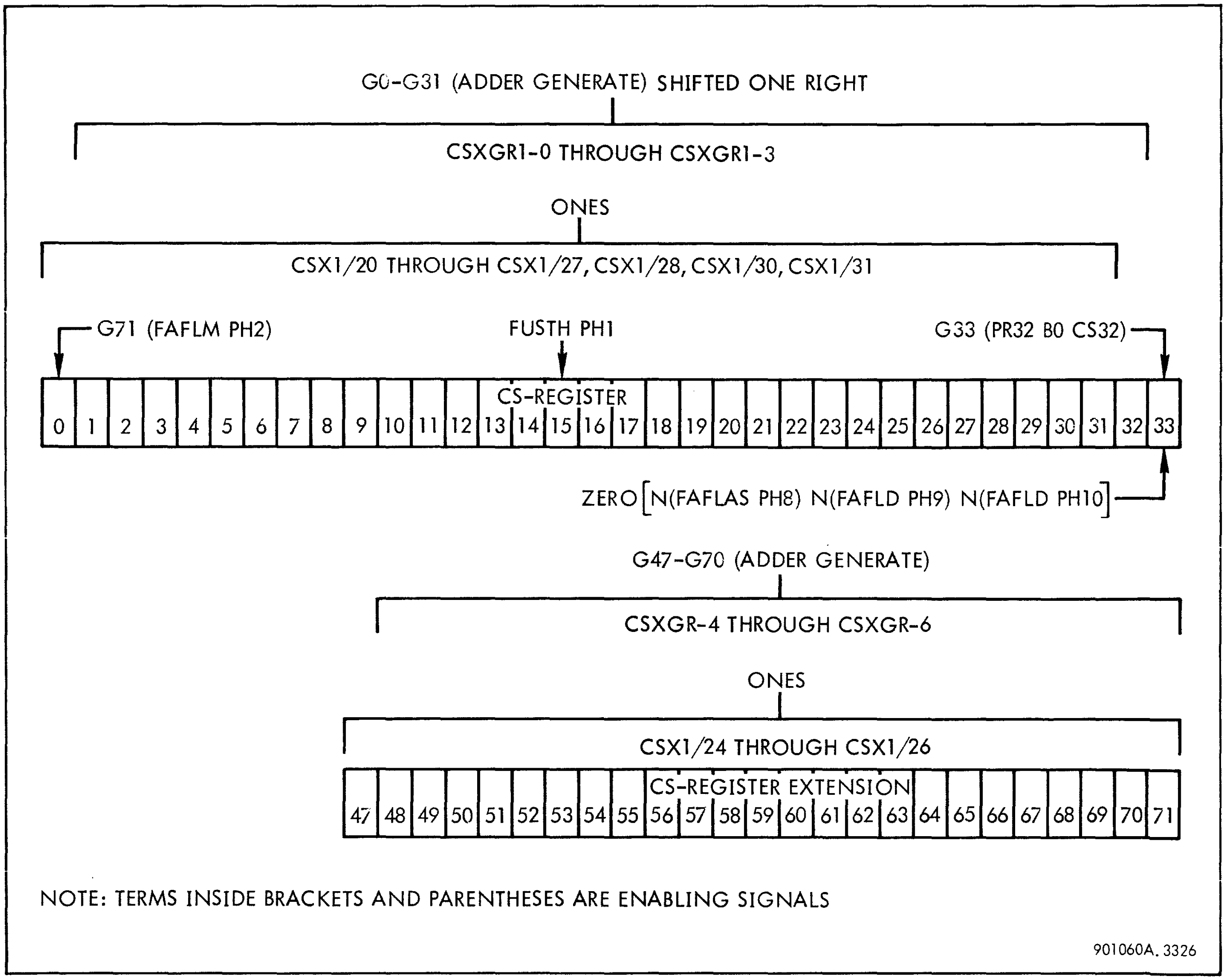
e. To place the index register address and private memory page address on the private memory address lines

f. To display the contents of the memory fault flip-flops in the MEMORY FAULT indicators on the processor control panel

g. To align bytes and halfwords downward from the C-register for storage in memory

h. To perform arithmetic calculations

i. To provide intermediate storage during transfer from core memory into private memory



NOTE: TERMS INSIDE BRACKETS AND PARENTHESES ARE ENABLING SIGNALS

Figure 3-46. CS-Register Inputs and Enabling Signals

j. To store full words in core memory

k. To load into memory the complement of the word in the C-register

l. To extend the sign bit

m. To hold the 32 low-order bits of the effective doubleword in doubleword operation

The D-register consists of 33 register flip-flops, D0 through D31 and D71, when the floating point option is not included; 25 register flip-flops, D46-D70, are added for floating point operation. Flip-flop D71 is used in multiplication in the standard computer. The inputs to the D-register and to the D-register extension, with the enabling signals for each input, are shown in figures 3-47 and 3-48. The four-byte enabling signals are given under one bracket for a specified transfer instead of showing individual byte transfer. For example, C is gated into D with DXC/10 through DXC/13. Byte 0 is gated with DXC/10, byte 1 with DXC/11, byte 2 with DXC/12, and byte 3 with DXC/13. Other byte-enabling signals are similarly gated.

3-47 E-REGISTER. The E-register is used for the following purposes:

a. To hold the exponent in floating point operation

b. To check for a nonzero product in multiplication

c. To act as a counter in byte operation and in floating point shift instructions, and to place the control image in memory during memory map and memory protection operations

d. To store the condition code in stack instructions

The E-register consists of eight counter flip-flops. Data enters the register via the B-register, the condition code register, and the sum bus. Outputs from the E-register are applied to the A-register. Internal E-register count logic adds or subtracts one from the contents of the register or subtracts four from the contents.

Figure 3-49 shows inputs to the E-register, with enabling signals. Count logic is not shown in this figure, but the complete count logic is given in table 3-3.

3-48 O-REGISTER. The O-register is an operation code register. Decoding to determine the instruction to be executed is done from this register. The operation code is taken from bits 1 through 7 of the C-register which contain the operation field of the instruction.

The O-register is made up of seven flip-flops. The inputs to the O-register and the enabling signals are shown in figure 3-50.

3-49 P-REGISTER. The P-register is used as a storage register for core memory addresses and as a counter. During address storage, the count logic increases or decreases the contents of the P-register in increments of one in order to address the next memory location in sequence or to address the previous memory location. The P-register is part of the program status doubleword; however, the INSTRUCTION ADDRESS indicators in the program status doubleword on the processor control panel are lighted by the outputs of the Q-register that contain the same information.

The P-register consists of 19 counter flip-flops. Addresses are placed in the register by way of the sum bus, the Q-register, the interrupt address flip-flops, the trap address flip-flops, and the SELECT ADDRESS switches on the processor control panel. Address information goes out of the P-register to the core memory by direct input, via the LB lines, to the Q-register and to the processor control panel INSTRUCTION ADDRESS indicators via the sum bus.

Information input to the P-register and the enabling signals are shown in figure 3-51. The count logic for the P-register is not shown in this figure, but is discussed separately.

The counting function of the P-register is used for the following purposes:

a. To add or subtract increments to or from the entire P-register contents for address sequencing

b. To count shifts in shift instructions

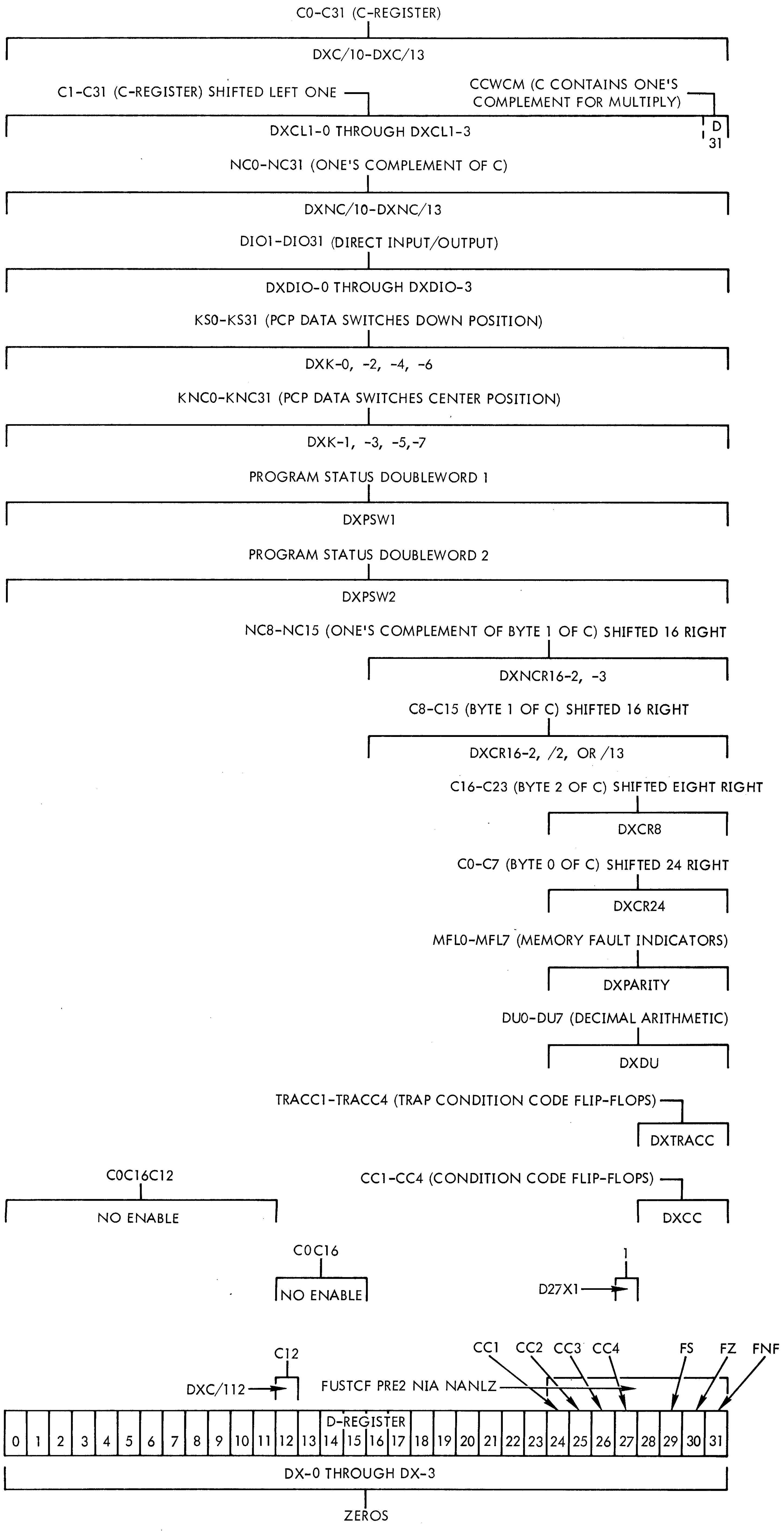
c. To count iterations in multiply and divide instructions

d. To count words when storing the memory map or memory protection control image in memory

The P-register count-control logic operates on five separate sections of the register as shown in figure 3-52. The enabling signals are the plus-count P-register signals, PCTP1 through PCTP5, for adding increments to all five sections; and the minus-count P-register signals, MCTP1, MCTP2, and MCTP5 for subtracting increments from the corresponding sections.

The outputs of the P-register flip-flops and the plus-count signals PCTP1 through PCTP5 are used to add one to the contents of the P-register for address sequencing. Signal PCTP1 is generated for this purpose from the ENDE signal during the end phase of each instruction or from the following processor control panel switch inputs to the NPCTP1 logic:

KFILL	(LOAD switch)
KINCREMENT	(INCREMENT position of INSTR ADDR switch)
CLEARMEM	(CPU RESET/CLEAR switch and SYSTEM RESET/CLEAR switch, pressed simultaneously)


C0-C31 (C-REGISTER)
DXC/10-DXC/13
C1-C31 (C-REGISTER) SHIFTED LEFT ONE
CCWCM (C CONTAINS ONE'S COMPLEMENT FOR MULTIPLY)
DXCL1-0 THROUGH DXCL1-3
D 31
NC0-NC31 (ONE'S COMPLEMENT OF C)
DXNC/10-DXNC/13
DIO1-DIO31 (DIRECT INPUT/OUTPUT)
DXDIO-0 THROUGH DXDIO-3
KS0-KS31 (PCP DATA SWITCHES DOWN POSITION)
DXK-0, -2, -4, -6
KNC0-KNC31 (PCP DATA SWITCHES CENTER POSITION)
DXK-1, -3, -5,-7
PROGRAM STATUS DOUBLEWORD 1
DXPSW1
PROGRAM STATUS DOUBLEWORD 2
DXPSW2
NC8-NC15 (ONE'S COMPLEMENT OF BYTE 1 OF C) SHIFTED 16 RIGHT
DXNCR16-2, -3
C8-C15 (BYTE 1 OF C) SHIFTED 16 RIGHT
DXCR16-2, /2, OR /13
C16-C23 (BYTE 2 OF C) SHIFTED EIGHT RIGHT
DXCR8
C0-C7 (BYTE 0 OF C) SHIFTED 24 RIGHT
DXCR24
MFL0-MFL7 (MEMORY FAULT INDICATORS)
DXPARITY
DU0-DU7 (DECIMAL ARITHMETIC)
DXDU
TRACC1-TRACC4 (TRAP CONDITION CODE FLIP-FLOPS)
DXTRACC
C0C16C12
NO ENABLE
CC1-CC4 (CONDITION CODE FLIP-FLOPS)
DXCC
C0C16
NO ENABLE
1
D27X1
C12
DXC/112
CC1 CC2 CC3 CC4
FS FZ FNF
FUSTCF PRE2 NIA NANLZ
D-REGISTER
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
DX-0 THROUGH DX-3
ZEROS

NOTE: TERMS INSIDE BRACKETS ARE ENABLING SIGNALS

Figure 3-47. D-Register Inputs and Enabling Signals

901060A. 3328

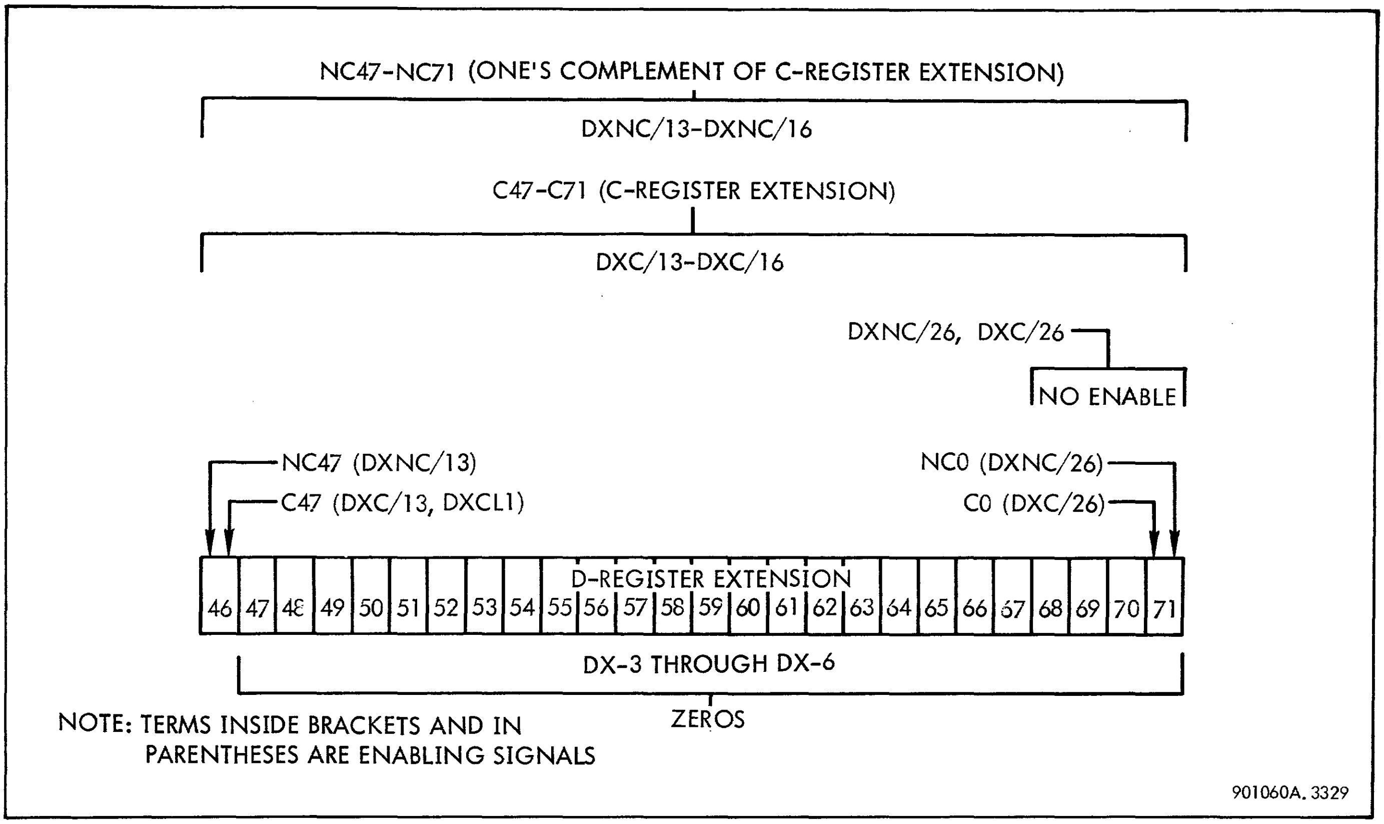


Figure 3-48. D-Register Extension Inputs and Enabling Signals

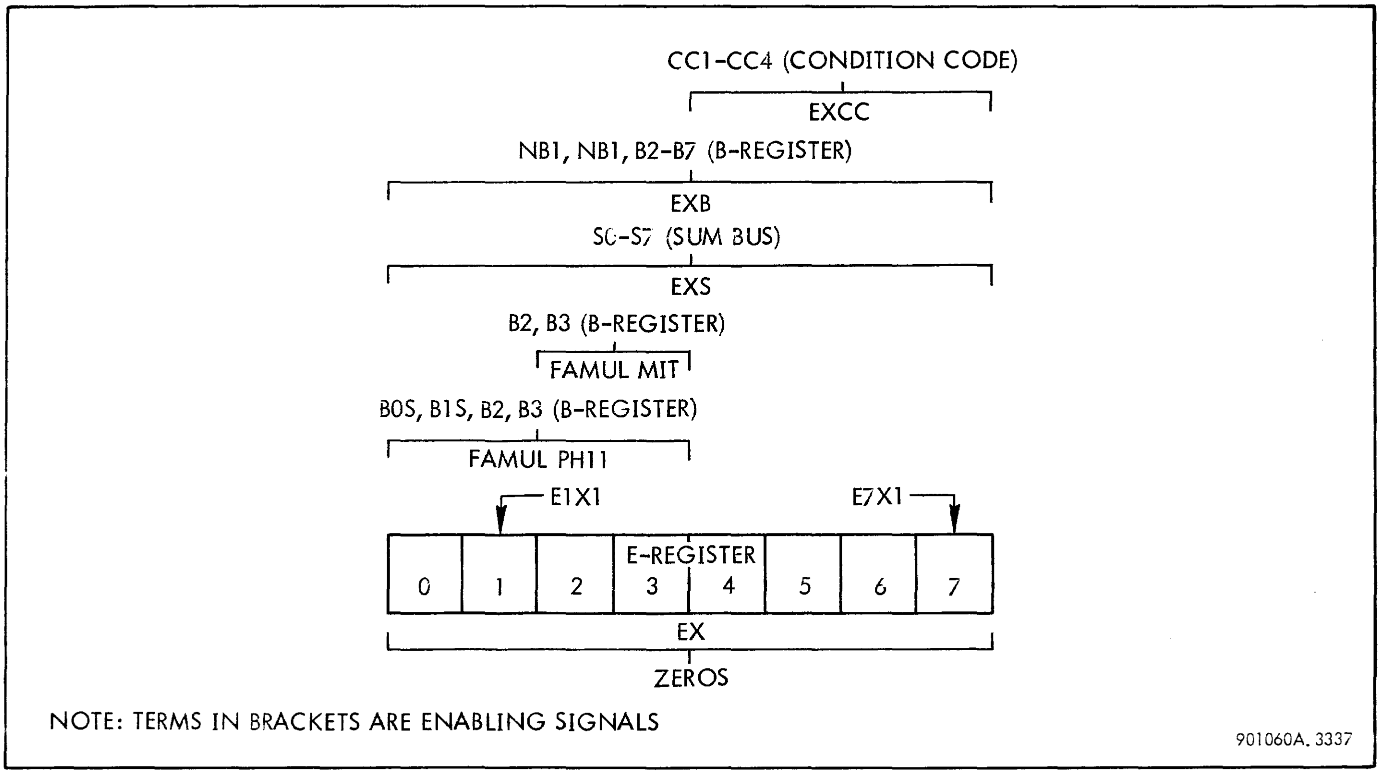


Figure 3-49. E-Register Inputs and Enabling Signals

Table 3-3. E-Register Count Logic

Count Enabling Signal	Flip-Flop	Set Logic	Reset Logic	Count
PCTE1	E4	NE4 E5 E6 E7	E5 E6 E7	Counts up by ones. ES4 = 1 during PCTE1
	E5	NE5 E6 E7	E6 E7	
	E6	NE6 E7 ES4	E7 ES4	
	E7	NE7 ES4	ES4	
MCTE1	E4	NE4 NE5 (NE6 NE7 + NES4)	NE5 (NE6 NE7 + NES4)	If ES4 = 1, counts down by ones. If ES4 = 0, counts down by fours
	E5	NE5 (NE6 NE7 + NES4)	NE6 NE7 + NES4	
	E6	NE6 NE7 ES4	NE7 ES4	
	E7	NE7 ES4	ES4	
PCTE2	E0	NE0 E1 E2 E3	E1 E2 E3	Counts up by ones
	E1	NE1 E2 E3	E2 E3	
	E2	NE2 E3	E3	
	E3	NE3	PCTE2	
MCTE2	E0	NE0 NE1 NE2 NE3	NE1 NE2 NE3	Counts down by ones
	E1	NE1 NE2 NE3	NE2 NE3	
	E2	NE2 NE3	NE3	
	E3	NE3	MCTE2	

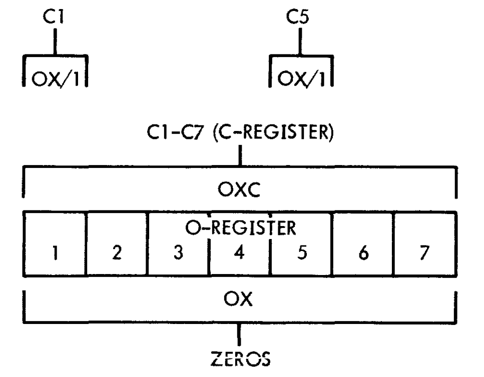
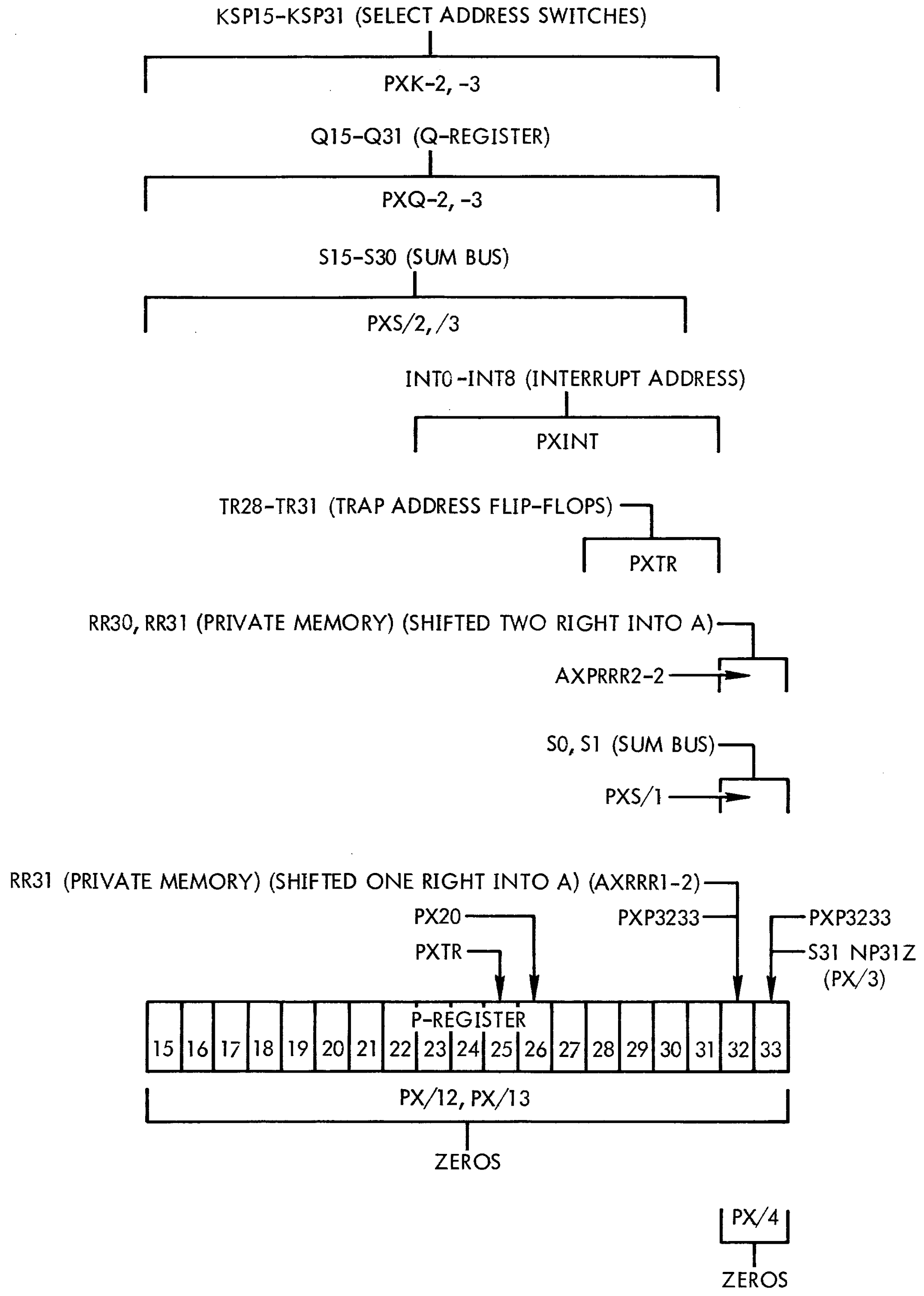


Figure 3-50. O-Register Inputs and Enabling Signals


KSP15-KSP31 (SELECT ADDRESS SWITCHES)
PXK-2, -3
Q15-Q31 (Q-REGISTER)
PXQ-2, -3
S15-S30 (SUM BUS)
PXS/2, /3
INT0-INT8 (INTERRUPT ADDRESS)
PXINT
TR28-TR31 (TRAP ADDRESS FLIP-FLOPS)
PXTR
RR30, RR31 (PRIVATE MEMORY) (SHIFTED TWO RIGHT INTO A)
AXPRRR2-2
S0, S1 (SUM BUS)
PXS/1
RR31 (PRIVATE MEMORY) (SHIFTED ONE RIGHT INTO A) (AXRRR1-2)
PX20
PXTR
PXP3233
PXP3233
S31 NP31Z
(PX/3)
P-REGISTER
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
PX/12, PX/13
ZEROS
PX/4
ZEROS

NOTE: TERMS INSIDE BRACKETS AND IN PARENTHESES ARE ENABLING SIGNALS

901060A. 3340

Figure 3-51. P-Register Inputs and Enabling Signals

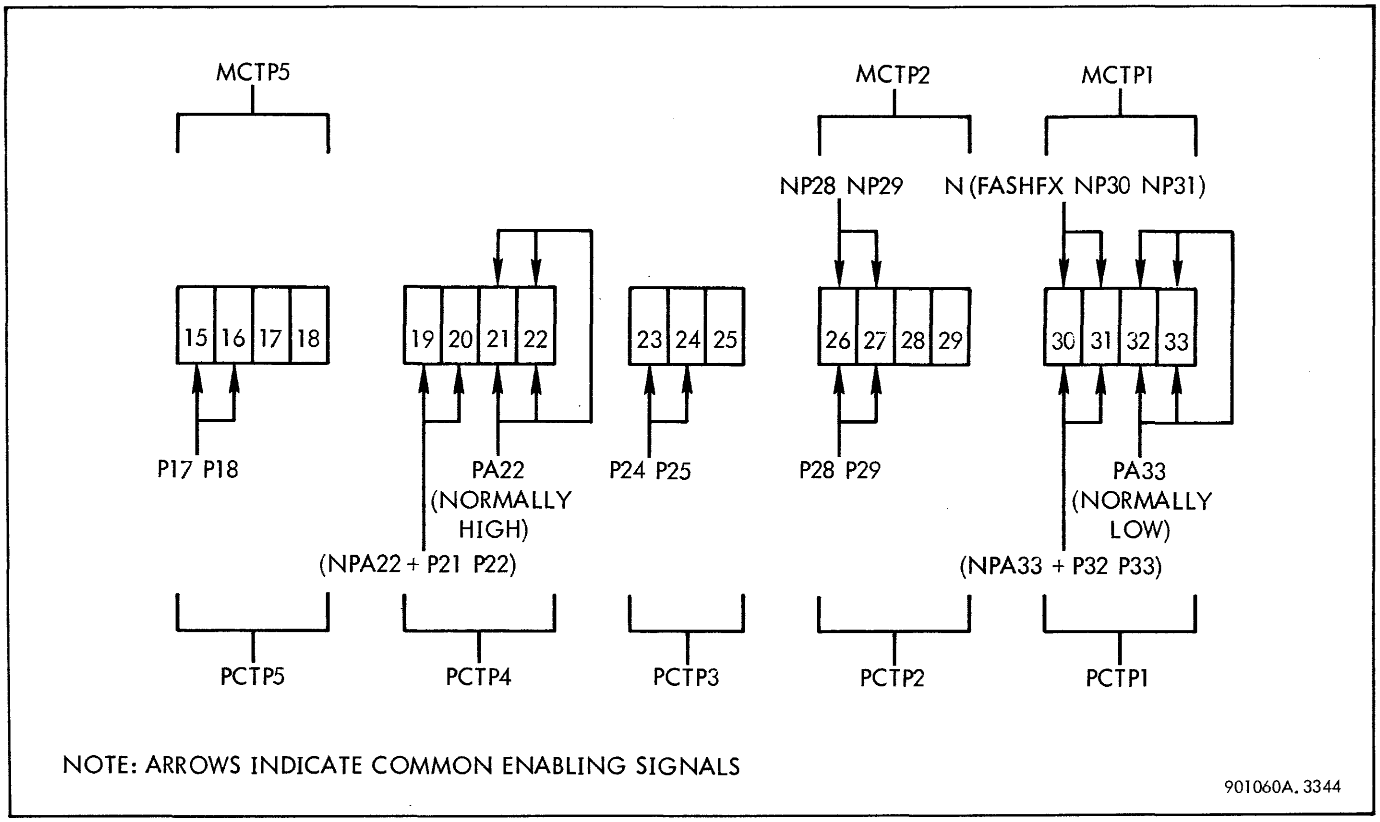


Figure 3-52. P-Register Count Gating

The count begins in a two-bit counter, P30 and P31. When these flip-flops contain ones, PCTP2/1 is true, and P26 through P29 act as a four-bit counter until all of these flip-flops contain ones. This causes P2629W to go true. Signal PCTP3 is then generated, and P23 through P25 count up by ones. Signal P2329W is now true, and PCTP4 counts up the section consisting of P19 through P21. At the end of this count, P1929W is true, and P15 through P18 are allowed to count to all ones. When the P-register is full of ones, the count is 131,072 which includes all locations in a maximum 128K memory.

The set and reset logic for each bit in the P-register during count operations is shown in table 3-4. Signal PA33 is true during byte string, decimal, translate, and move-to-memory control instructions. Signal FASHFX is true during shift instructions.

3-50 Q-REGISTER. The Q-register stores the next instruction address in order to release the P-register for other functions. In some cases, the Q-register outputs are applied directly to the core memory address lines for memory access. The Q-register information is transferred to the P-register when the P-register is ready to supply the address. When the computer is being set to the initial conditions by the CPU RESET switch, a hexadecimal 25 is placed directly in the Q-register.

Lamp drivers are connected to the Q-register flip-flops for display in the INSTRUCTION ADDRESS indicators in the program status doubleword on the processor control panel. The information in the Q-register, the same as that in the P-register, is part of the program status doubleword. A block diagram of the Q-register and the enabling signals is shown in figure 3-53.

3-51 R-REGISTER. The R-register stores the address of a private memory register. The register addresses, ranging from 0 to 15, address the 16 registers in one register bank. The address is received from the C-register, bits 8 through 11, which contains the R-field of the instruction word. Outputs from the R-register go to the private memory address lines.

The R-register acts as a counter when it is necessary to address sequential private memory registers. Signal PCTR (plus count the R-register) adds one to the contents of the R-register, and signal MCTR (minus count the R-register) subtracts one from the contents of the R-register. Table 3-5 contains set and reset logic for each R-register flip-flop when counting up and down.

Figure 3-54 is a diagram of the R-register inputs and enabling signals. The count logic for the R-register is not shown.

Table 3-4. P-Register Count Logic

Count Enabling Signal	Flip-Flop	Set Logic	Reset Logic	Count
PCTP1	P30	NP30 P31 (P32 P33 + NPA33)	P31 (P32 P33 + NPA33)	NPA true, P30 and P31 count up by ones; P32 and P33 remain unchanged. PA33 true. Four bits count up by ones
	P31	NP31 (P32 P33 + NPA33)	(P32 P33 + NPA33)	
	P32	NP32 P33 PA33	P33 PA33	
	P33	NP33 PA33	PA33	
MCTP1	P30	NP31 NP30 N(FASHFX NP30 NP31)	NP31 N(FASHFX NP30 NP31)	FASHFX true. P30 and P31 count down by ones to zero, then remain unchanged. P32 and P33 cannot set or reset. PA33 true. P30 and P31 count down by ones; P32 and P33 count down by ones
	P31	NP31 N(FASHFX NP30 NP31)	N(FASHFX NP30 NP31)	
	P32	PA33 NP33 NP32	PA33 NP33	
	P33	PA33 NP33	PA33	
PCTP2	P26	NP26 P27 P28 P29	P27 P28 P29	Counts up by ones
	P27	NP27 P28 P29	P28 P29	
	P28	NP28 P29	P29	
	P29	NP29	PCTP2	
MCTP2	P26	NP26 NP27 NP28 NP29	NP27 NP28 NP29	Counts down by ones
	P27	NP27 NP28 NP29	NP28 NP29	
	P28	NP28 NP29	NP29	
	P29	NP29	MCTP2	
PCTP3	P23	NP23 P24 P25	P24 P25	Counts up by ones
	P24	NP24 P25	P25	
	P25	NP25	PCTP3	
PCTP4	P19	NP19 P20 (P21 P22 + NPA22)	P20 (P21 P22 + NPA22)	NPA22 true. P21 and P22 remain unchanged; P19 and P20 count up by ones. PA22 true. Four bits count up by ones
	P20	NP20 (P21 P22 + NPA22)	(P21 P22 + NPA22)	
	P21	NP21 P22 PA22	P22 PA22	
	P22	NP22 PA22	PA22	
PCTP5	P15	NP15 P16 P17 P18	P16 P17 P18	Counts up by ones
	P16	NP16 P17 P18	P17 P18	
	P17	NP17 P18	P18	
	P18	NP18	PCTP5	
MCTP5	P15	NP16 NP15	NP16	Counts 0000 down to 1111
	P16	NP16	MCTP5	
	P17	NP17 NP18	NP18	
	P18	NP18	MCTP5	

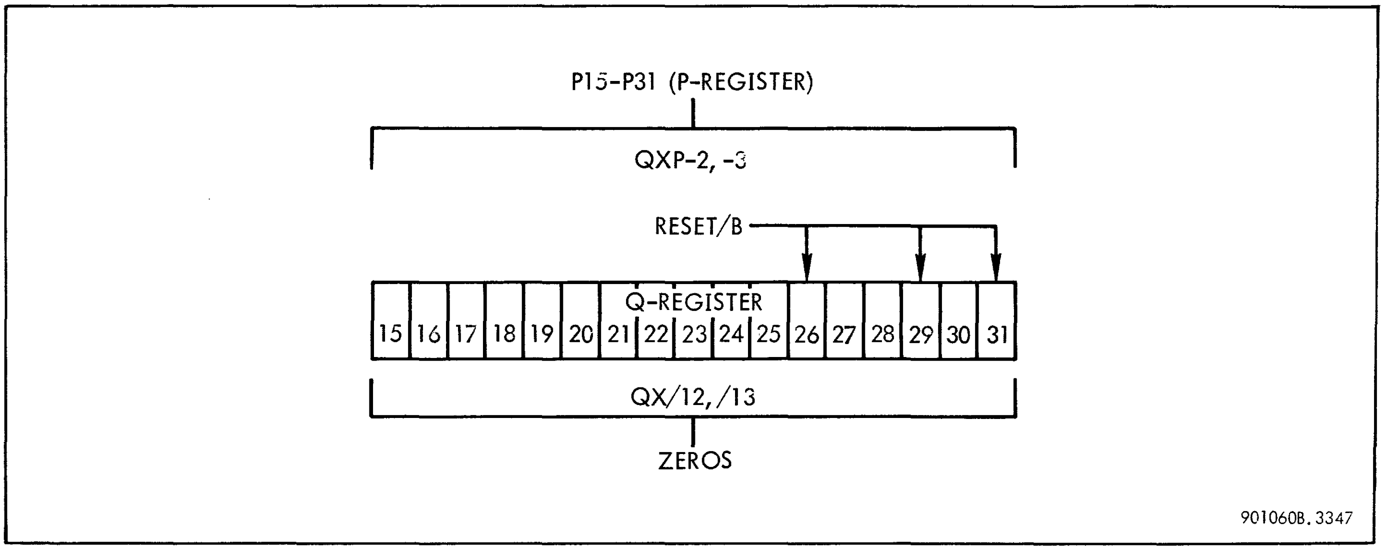


Figure 3-53. Q-Register Inputs and Enabling Signals

Table 3-5. R-Register Count Logic

Count Enabling Signal	Flip-Flop	Set Logic	Reset Logic
PCTR	R28	NR28 R29 R30 R31	R29 R30 R31
	R29	NR29 R30 R31	R30 R31
	R30	NR30 R31	R31
	R31	NR31	PCTR
MCTR	R28	NR28 NR29 NR30 NR31	NR29 NR30 NR31
	R29	NR29 NR30 NR31	NR30 NR31
	R30	NR30 NR31	NR31
	R31	NR31	MCTR

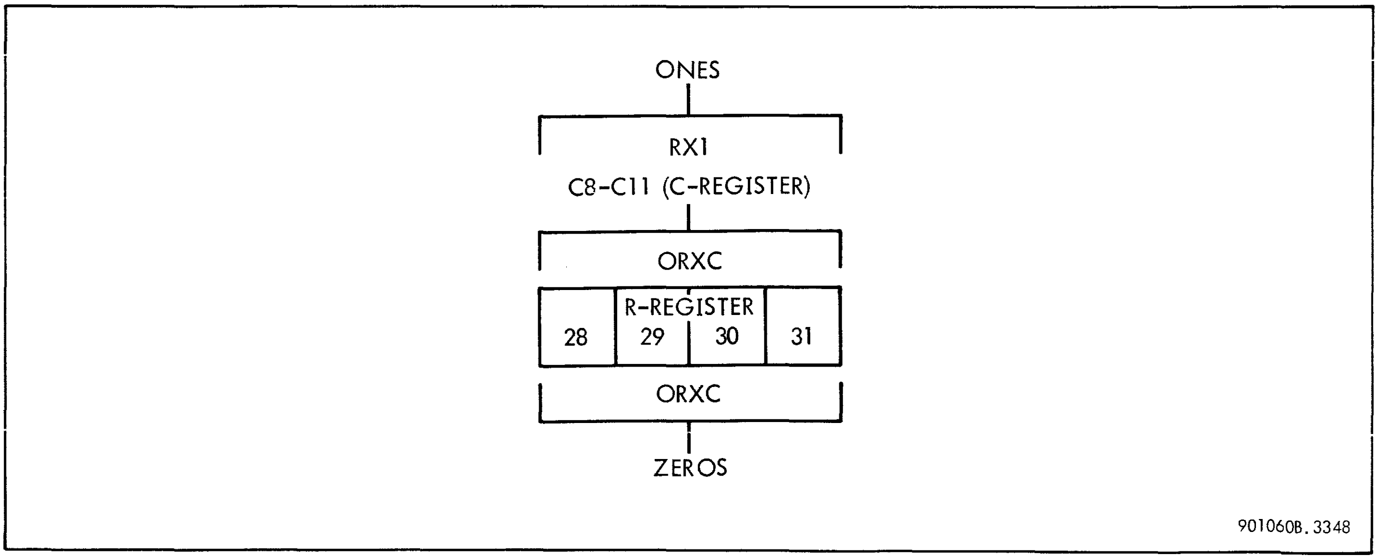


Figure 3-54. R-Register Inputs and Enabling Signals

3-52 RP-REGISTER. The register pointer (RP) register is part of the program status doubleword and is used to store the address of a private memory extension register bank. This register is used when one or more additional private memory banks are included in the system to augment the 16 basic private memory registers in the standard computer. When the D-register contains PSW2 of the program status doubleword during PSD instructions, the register pointer is loaded from the D-register. During the Load Register Pointer instruction, the register pointer address is taken from memory and is stored in the RP-register through the D-register.

The RP-register consists of five flip-flops with inputs from the D-register and outputs to the private memory address lines. Figure 3-55 is a block diagram of the register and the enabling signal.

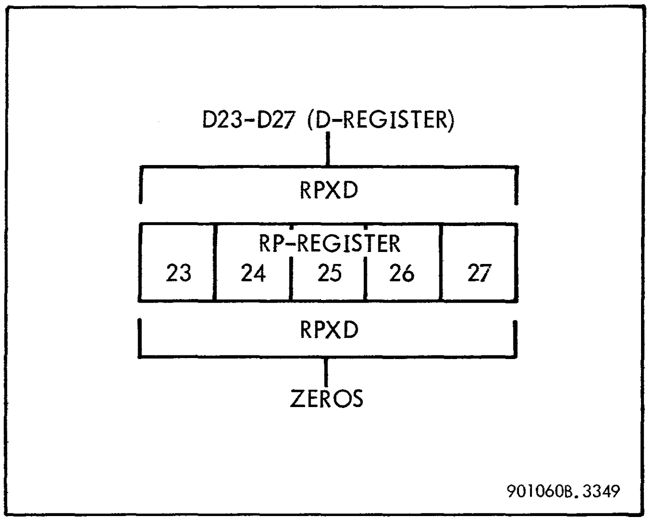


Figure 3-55. RP-Register Inputs and Enabling Signals

3-53 PRIVATE MEMORY REGISTERS. The private memory registers are located on a set of FT25 fast-access memory modules. The registers are installed in banks of 16, numbered register 0 through F in hexadecimal notation. Each register contains a 32-bit word.

A maximum of 32 private memory banks may be installed in the computer. Each bank is assigned a page number, 0 through 31, and is addressed by the RP-register with codes from 00000 to 11111. Page 0 is included in the standard computer; pages 1 through 31 are optional. Pages 0 through 3 are installed in frame 2 of computer unit 1. Additional pages are installed in the register extension chassis in frame 2 of unit 2.

Each private memory bank consists of four FT25 fast-access memory modules. The distribution of the words among the four modules is shown in block diagram form in figure 3-56. Each module contains one byte of any given word.

One FT25 module conains 16 eight-bit integrated circuit memory elements (XDS 304). A simplified diagram of a single memory element is shown in figure 3-57. The control, address, and V_{CC} inputs are applied to all flip-flops in the element, although it is not shown in the diagram.

Each bit of the element is individually addressed by the address lines on pins 2, 3, and 4. The three-bit code selects one of eight flip-flops. The control line also contains address information. When the control line is false, the states of all bits in the memory element remain unchanged, regardless of the state of the read/write line. When the control line is true, bits of the element may change state if the read/write line is true; the output of each bit may be read when that bit is addressed regardless of the state of the read/write line.

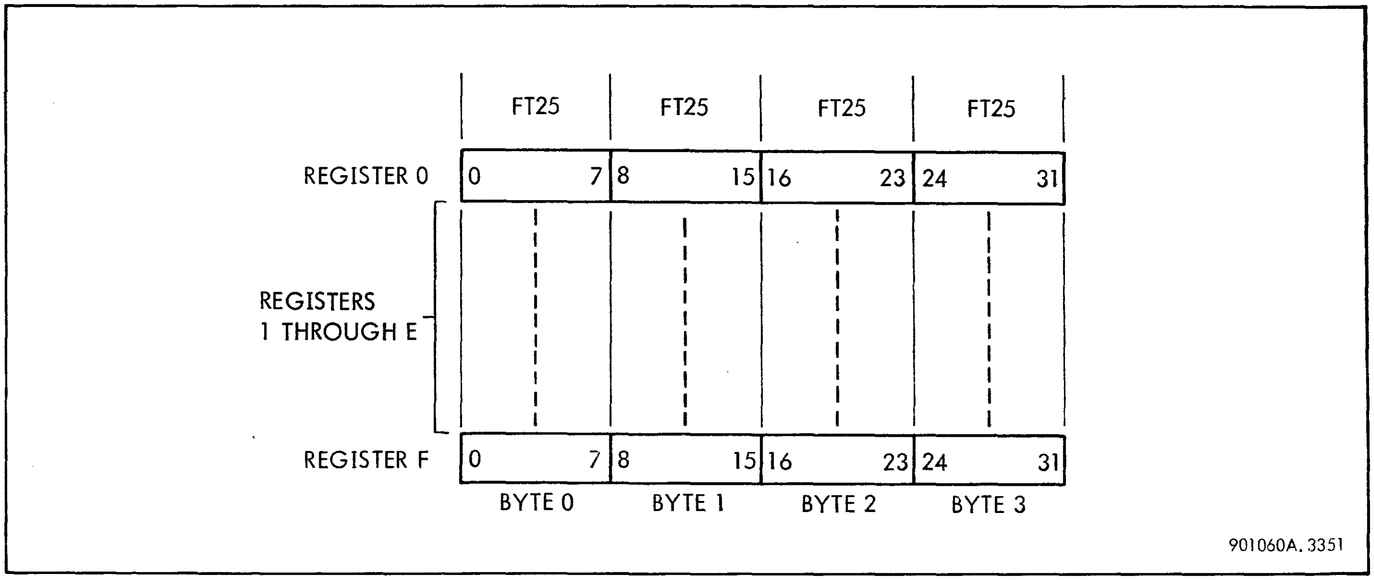


Figure 3-56. Word Distribution in Private Memory Bank

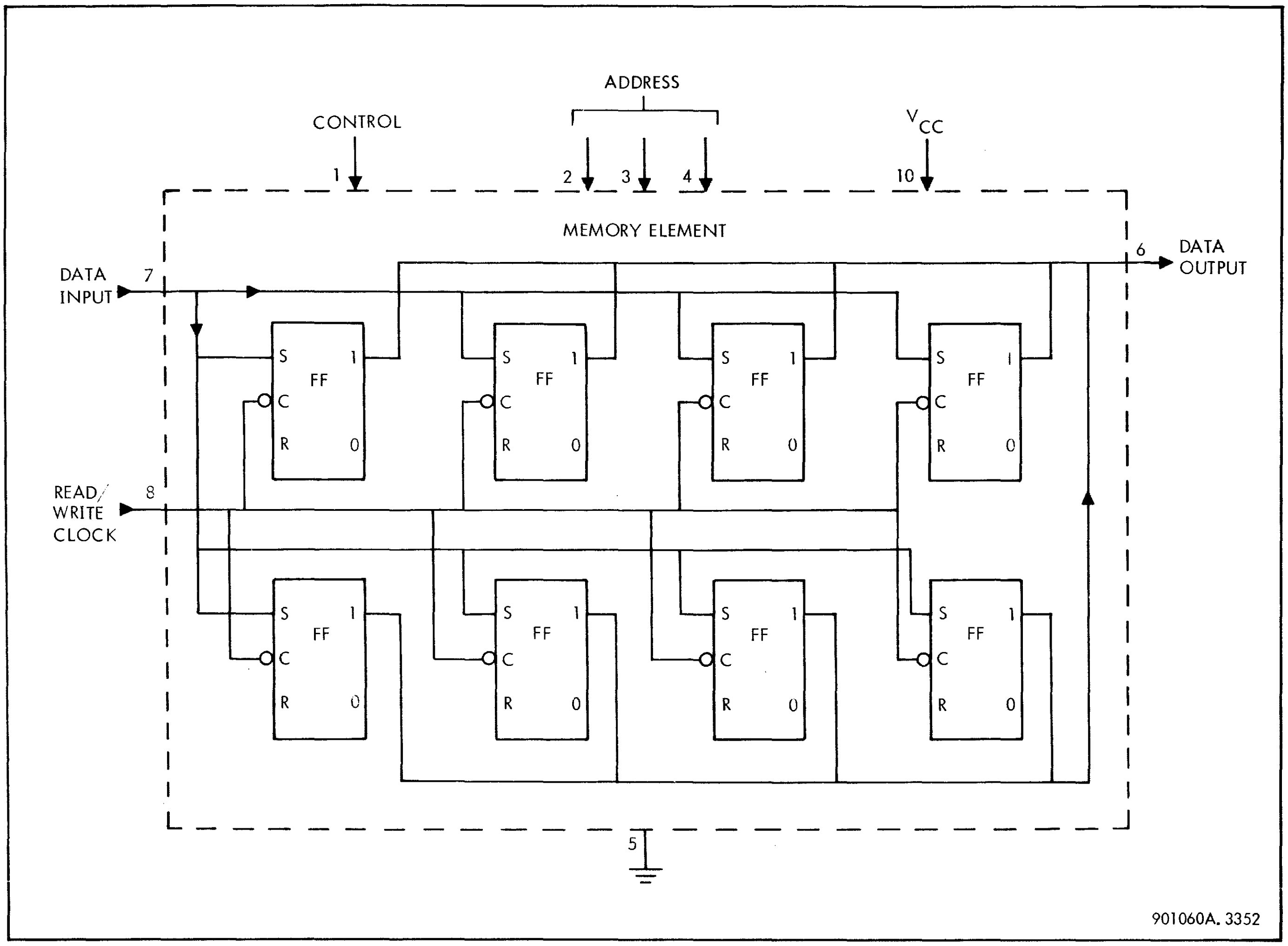


Figure 3-57. Memory Element (XDS 304), Simplified Block Diagram

The arrangement of bits in the memory elements on one FT25 fast-access memory module is shown in figure 3-58. The module shown contains byte 0 of the standard private memory bank, designated page 0. Each of the 16 memory elements contains eight corresponding bits in eight registers. The data, address, and write clock signals are interpreted as follows:

W/RP0B0/0

WRITE PAGE 0 BYTE 0 BIT 0

L/RP0B0/X

ADDRESS PAGE 0 BYTE 0 /1 through /5 (bit selection)

K/RP0B0

CLOCK PAGE 0 BYTE 0

Input and output data signals for the four modules in a memory bank are shown in figure 3-59. The address lines in the four modules are identical.

Individual bits in each memory element on the FT25 modules are selected by address lines LR28 through LR31. As indicated in figure 3-58, a memory element contains a corresponding bit for each of eight registers. Dividing the memory elements into two sets of eight (figure 3-58), the three least significant bits (LR29, LR30, and LR31) select one of eight registers in each set (see figure 3-60). Address line NLR28 gates information into the memory elements containing registers 0 through 7, and LR28 gates information into the elements containing registers 8 through F. The gating signal is connected to the control line input of every memory element. The total effect of the LR28 through LR31 address lines is to select one of 16 registers for input or output of data. Gating signals RP23 through RP27 which designate the private memory page are taken from the RP-register. Since each memory bank is equivalent to one page, the four FT25 modules in one bank receive the same code from the RP-register and are enabled at one time.

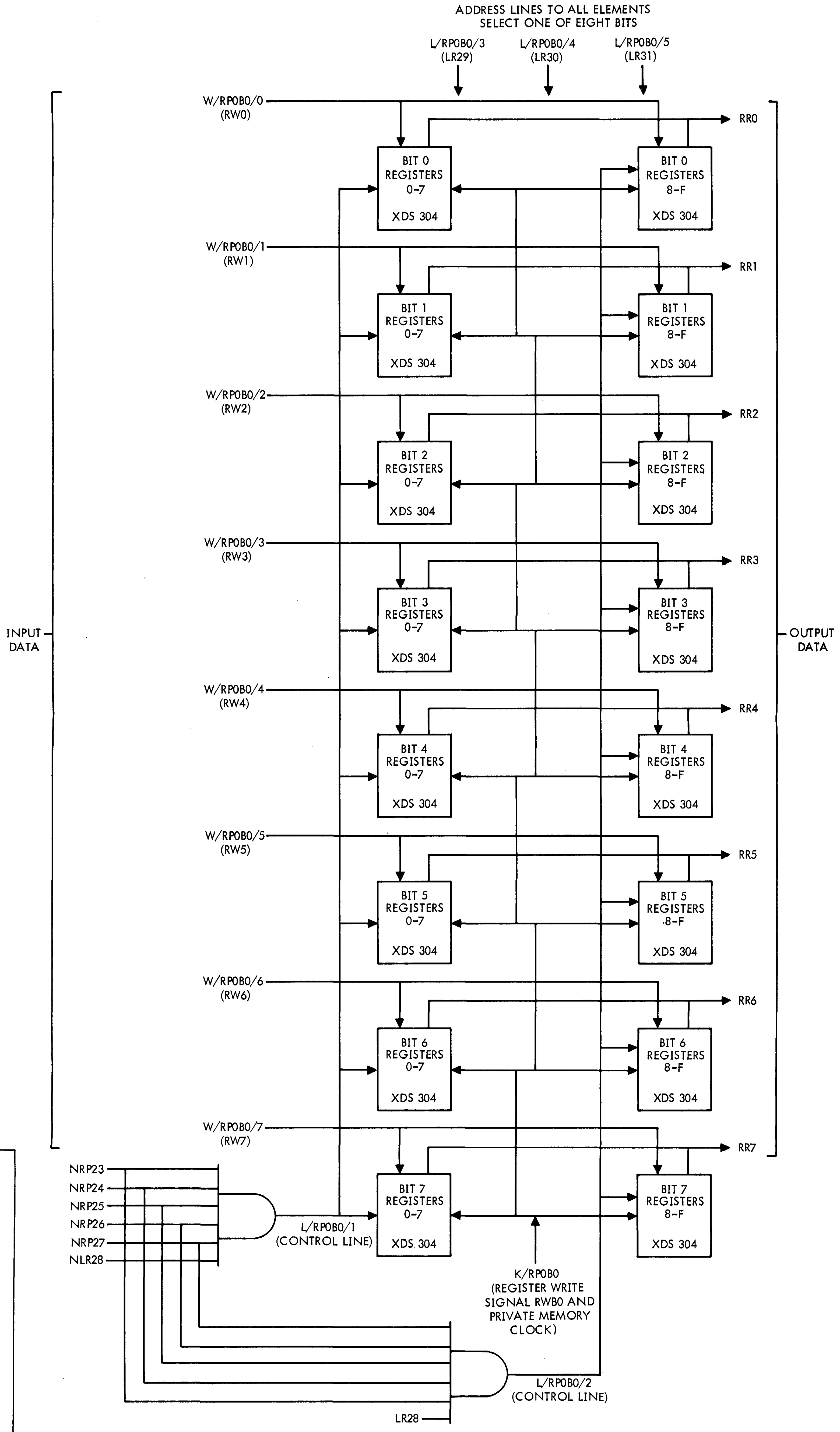

ADDRESS LINES TO ALL ELEMENTS SELECT ONE OF EIGHT BITS
L/RP0B0/3 (LR29)
L/RP0B0/4 (LR30)
L/RP0B0/5 (LR31)
W/RP0B0/0 (RW0)
W/RP0B0/1 (RW1)
W/RP0B0/2 (RW2)
W/RP0B0/3 (RW3)
W/RP0B0/4 (RW4)
W/RP0B0/5 (RW5)
W/RP0B0/6 (RW6)
W/RP0B0/7 (RW7)
INPUT DATA
BIT 0 REGISTERS 0-7 XDS 304
BIT 0 REGISTERS 8-F XDS 304
BIT 1 REGISTERS 0-7 XDS 304
BIT 1 REGISTERS 8-F XDS 304
BIT 2 REGISTERS 0-7 XDS 304
BIT 2 REGISTERS 8-F XDS 304
BIT 3 REGISTERS 0-7 XDS 304
BIT 3 REGISTERS 8-F XDS 304
BIT 4 REGISTERS 0-7 XDS 304
BIT 4 REGISTERS 8-F XDS 304
BIT 5 REGISTERS 0-7 XDS 304
BIT 5 REGISTERS .8-F XDS 304
BIT 6 REGISTERS 0-7 XDS 304
BIT 6 REGISTERS 8-F XDS 304
BIT 7 REGISTERS 0-7 XDS. 304
BIT 7 REGISTERS 8-F XDS 304
RR0
RR1
RR2
RR3
RR4
RR5
RR6
RR7
OUTPUT DATA
NRP23
NRP24
NRP25
NRP26
NRP27
NLR28
L/RP0B0/1 (CONTROL LINE)
K/RP0B0 (REGISTER WRITE SIGNAL RWB0 AND PRIVATE MEMORY CLOCK)
L/RP0B0/2 (CONTROL LINE)
LR28

Figure 3-58. FT25 Module, Page 0, Byte 0, Simplified Diagram

901060A.3353

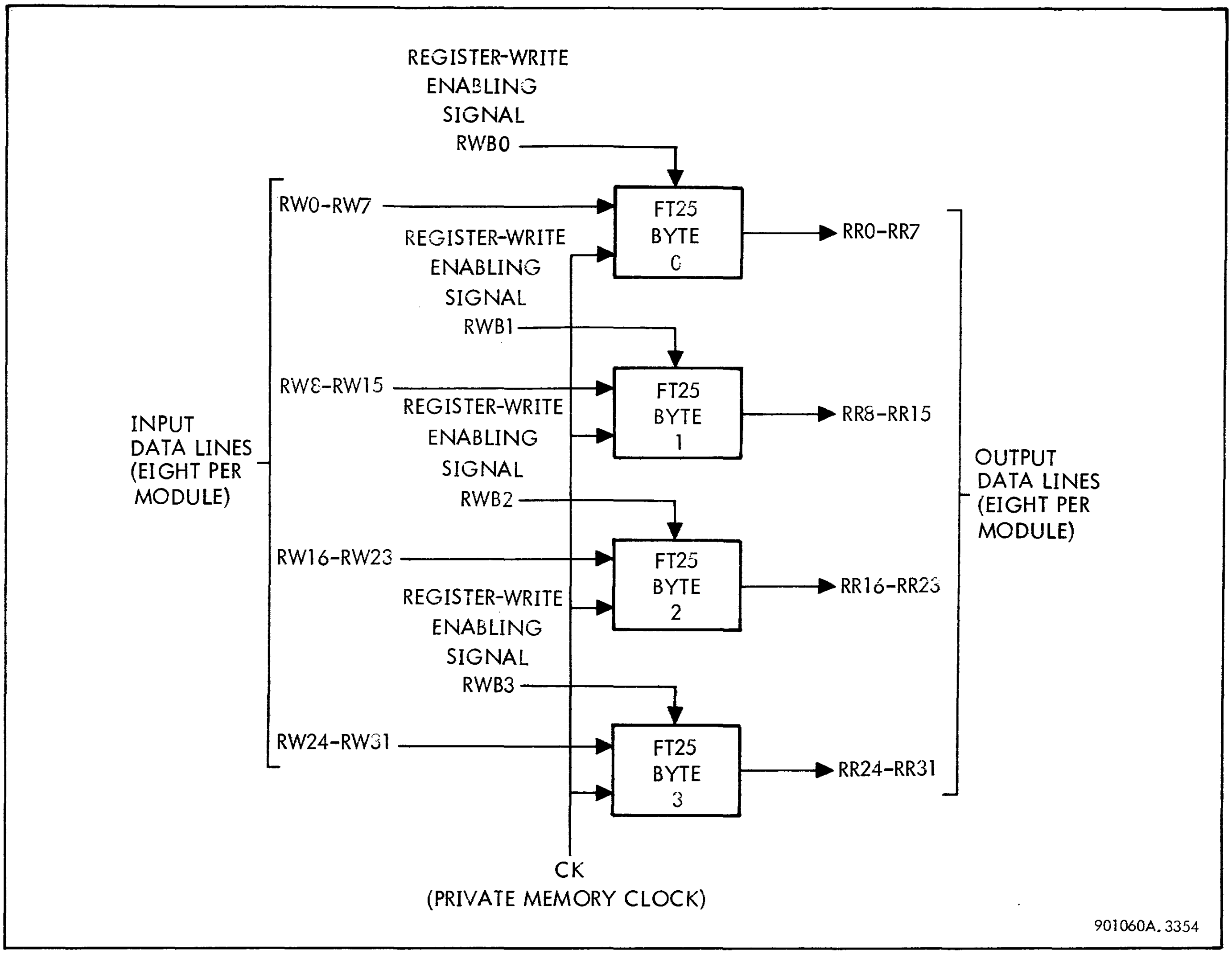
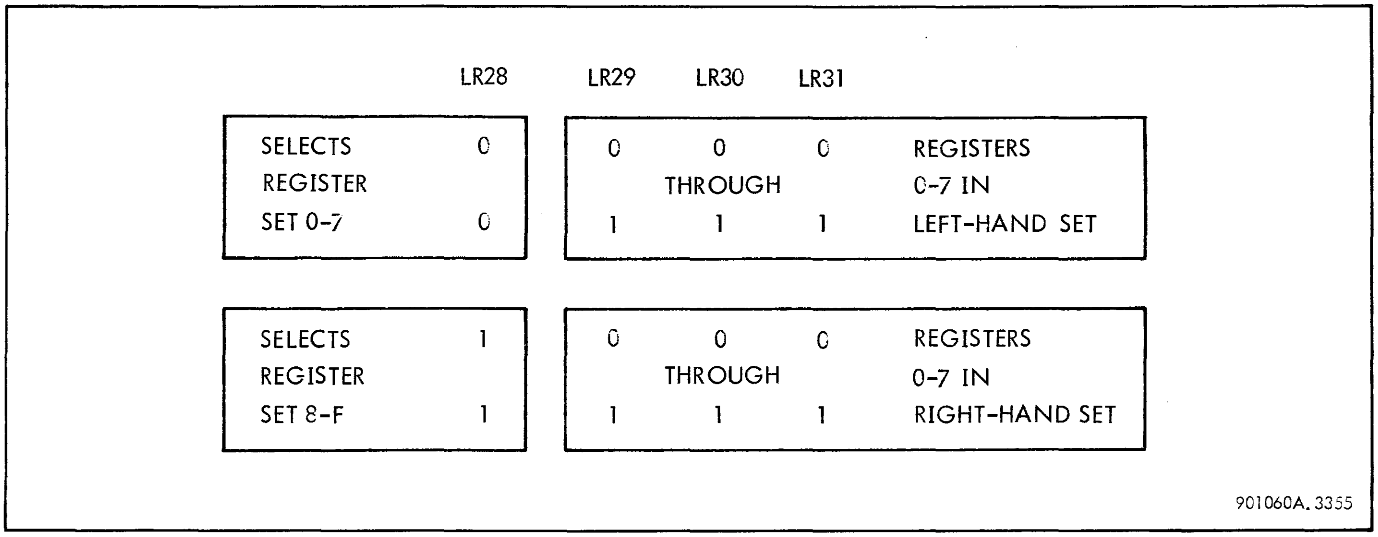

REGISTER-WRITE ENABLING SIGNAL RWB0
RW0-RW7
FT25 BYTE 0
RR0-RR7
REGISTER-WRITE ENABLING SIGNAL RWB1
RW8-RW15
FT25 BYTE 1
RR8-RR15
INPUT DATA LINES (EIGHT PER MODULE)
REGISTER-WRITE ENABLING SIGNAL RWB2
OUTPUT DATA LINES (EIGHT PER MODULE)
RW16-RW23
FT25 BYTE 2
RR16-RR23
REGISTER-WRITE ENABLING SIGNAL RWB3
RW24-RW31
FT25 BYTE 3
RR24-RR31
CK (PRIVATE MEMORY CLOCK)
901060A. 3354

Figure 3-59. Private Memory Data Organization



	LR28	LR29	LR30	LR31	
SELECTS	0	0	0	0	REGISTERS
REGISTER			THROUGH		0-7 IN
SET 0-7	0	1	1	1	LEFT-HAND SET
SELECTS	1	0	0	0	REGISTERS
REGISTER			THROUGH		0-7 IN
SET 8-F	1	1	1	1	RIGHT-HAND SET

901060A. 3355

Figure 3-60. Bit Addressing on FT25 Module

The RP lines for the module in figure 3-58 contain NRP23 through NPR27 which select page 00000.

Address signals LR28 through LR31 are generated, in general, from either the R-register or the D-register. The R-register contains the number of the private memory register to be addressed and is used when no crossover occurs. The equation is as follows:

LR28-LR31 = R28-R31 (LRXR NLRXLB) + . . .

Crossover occurs when a core memory location with an address of less than 16 is addressed. During crossover, LR28 through LR30 are taken from the D-register, when transferring from the C-register to the core memory address lines with the equation

LR28-LR30 = (D28-D30 LMXC + . . .) LRXLB

or are taken from the LB lines, when transferring from the Q-register or the P-register with the equation

LR28-LR31 = (LB28-LB31 NLMXC + . . .) LRXLB

In cases of doubleword operation or multiple-word operation, LR31 is generated from other sources either to select an odd numbered register or to implement the function R+1, as explained in the discussion on individual instructions. During the indexing operation, the private memory address is taken from bits 12 through 14 (the index field) of the D-register.

LR29-LR30 = D12-D14 LRXD + . . .

The equation for the register-write byte signals is as follows:

RWB0-RWB3 = [RW + MBXS
+ N(NRWB0) - N(NRWB3)] NRWDIS

Where RW is a register-write enabling signal, MBXS gates the sum bus onto the core memory bus; NRWB0-NRWB3 are byte 0 through 3 gating terms generated from instruction and phase logic; and RWDIS is a private memory write-disabling signal. Signal MBXS is used when crossover occurs because of a core memory address less than 16. The CK clock signal, gated with the write byte signals, is the private memory clock which comes true 20 ns before the ac clock signal.

Data signals are gated from the sum bus into the private memory as follows:

RW0-RW31 = S0-S31 RWXS

3-54 REGISTER EXTENSION CHASSIS (REU). A register extension chassis contains up to 16 FT25 fast-access memory modules and adds from one to four banks of additional private memory to the central processor. Since one bank of private memory requires four FT25 modules, these modules in the register extension chassis must be added in multiples of four. Up to seven register extension chassis may be added to the computer, making a maximum of 32 private memory banks, including the four banks in the CPU.

Additional modules in the REU provide cable drivers and receivers, terminators, chassis-selection switches and switch comparators, and logic circuits for selection and conversion of addresses and data signals. Figure 3-61 is a simplified logic diagram of the REU.

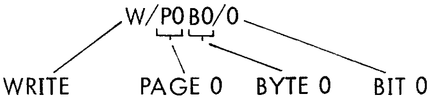
Address, data, and control signals are transmitted from the CPU on cables and are applied to cable receivers in the register extension unit. These bidirectional data cables also have cable driver inputs from the REU. Clock signals are taken from the ac clock circuits, CL1-CL12, in the CPU. The nomenclature, functions, and decoding of the interface signals between the CPU and the REU are given in table 3-6.

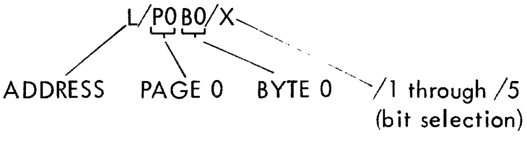
Each register extension chassis is assigned an address from 001 through 111 by manually setting switches S3-2, S3-1, and S2-2 on an LT26 switch comparator module in the desired configuration, with S2-2 as the least significant bit. The outputs of these switches are designated SWI0 for S3-2, SWI1 for S3-1, and SWI2 for S2-2. A MATCH signal is generated in the selected REU by comparing the switch signals with the chassis-selection bits in the address as follows:

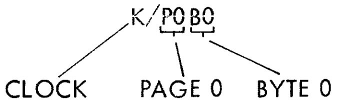
MATCH = N(SWI0 NREU0 + NSWI0 REU0
+ SWI1 NREU1 + NSWI1 REU1
+ SWI2 NREU2 + NSWI2 REU2)

This MATCH signal is applied to an AND gate containing another input, NREUZ, to indicate that page 0 is not being addressed, and the AND gate output, register extension unit select (REUSEL), is connected to all of the FT25 modules in the selected REU.

Address lines LR26 and LR27, designated PAG0 and PAG1 in the REU, are decoded to select one of four blocks or pages in the selected REU. Figure 3-62 is a simplified logic diagram of page 0 of the selected REU, a typical connection. The data, address, and clock signals are interpreted as follows:







Within each register extension unit, the pages are numbered individually 0 through 3.

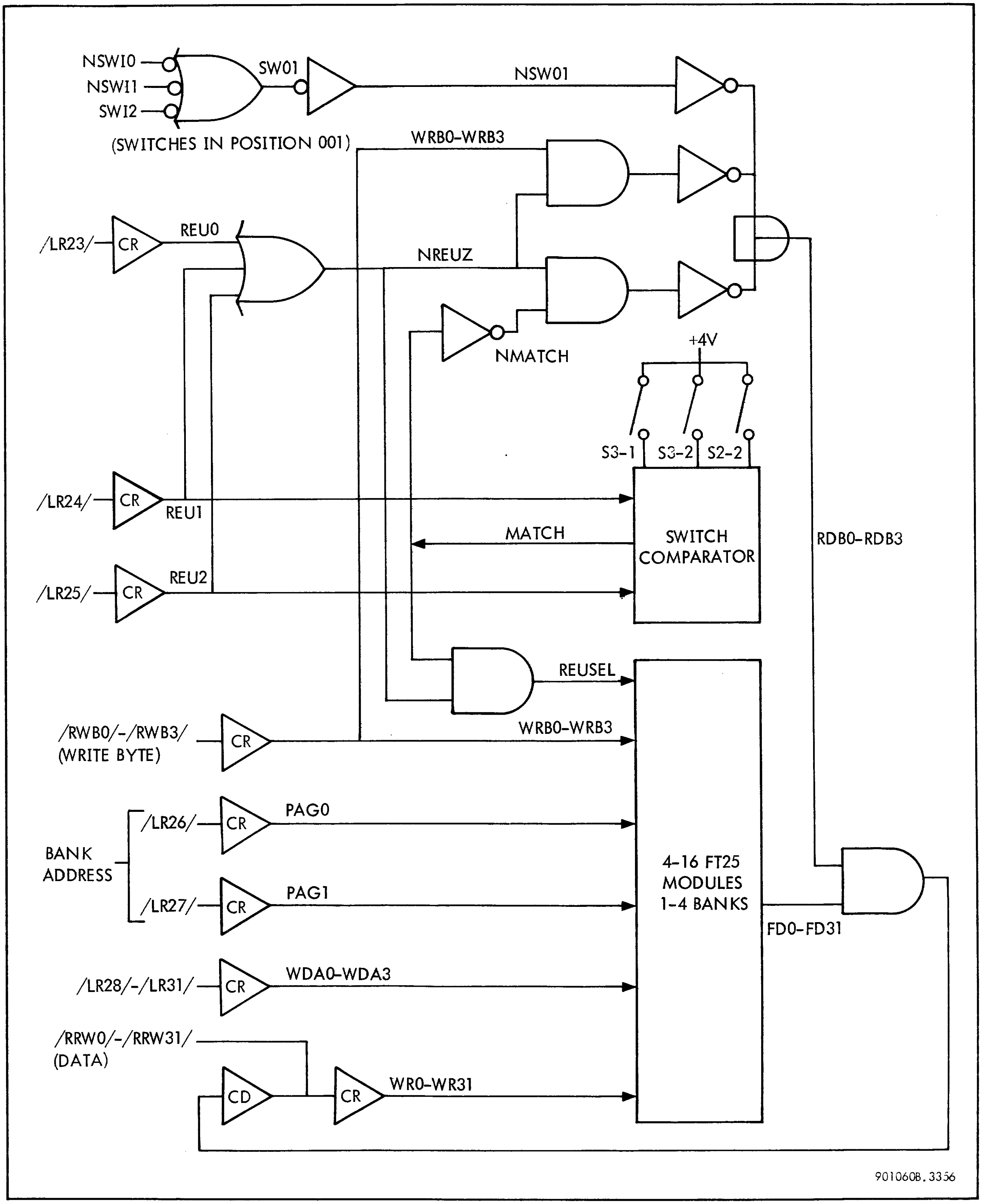


Figure 3-61. Register Extension Chassis, Simplified Block Diagram

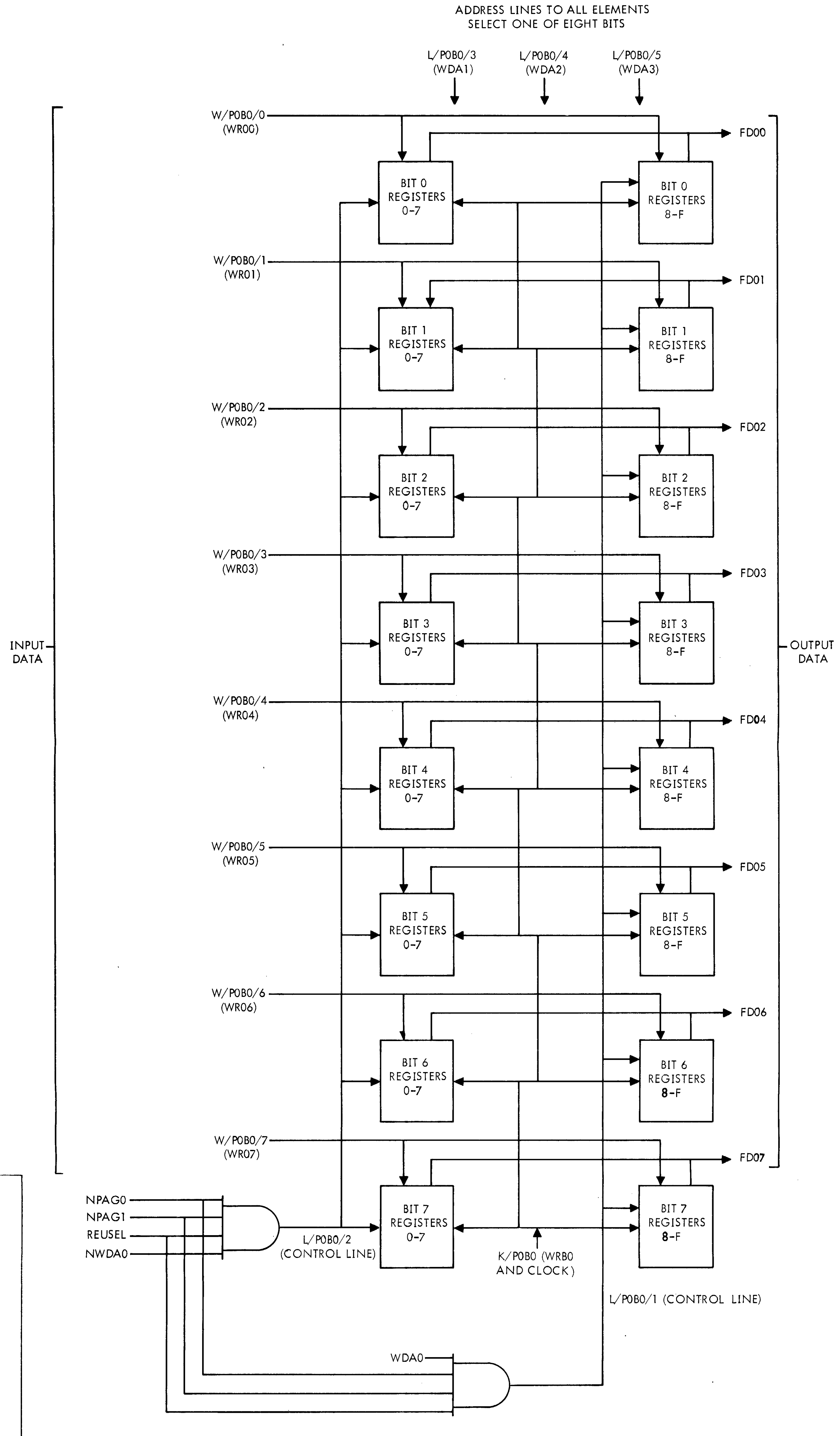


Figure 3-62. Register Extension FT25 Module, Page 0, Byte 0, Simplified Diagram

901060A. 3357

Table 3-6. REU Interface Signals

Input Cable	Function	Cable Receiver Output	Address Decoding
/LR23/ /LR24/ /LR25/	Address Address Address	REU0 REU1 REU2	Seven chassis (in addition to one 4-module set in CPU)
/LR26/ /LR27/	Address Address	PAG0 PAG1	Four 16-register backs in a chassis
/LR28/	Address	WDA0 or WDB0	Two sets of 8 memory elements on an FT25
/LR29/ /LR30/ /LR31/	Address Address Address	WDA1 or WDB1 WDA2 or WDB2 WDA3 or WDB3	Eight flip-flops in a memory element
/RRW0/- /RRW31/	Data	Cable receiver output: WR0-WR31 Cable driver input: FD0-FD31 RDB0-RDB3	
/RWB01/- /RWB3/	Write byte	WRB0-WRB3	

Read byte signals RDB0 through RDB3 are generated when the switch settings match the chassis-selecting address lines and when the write byte signals are low as shown in figure 3-61. The SW01 term is added to save power by turning off circuits in the unselected REU's.

Data is gated from the sum bus into the private memories in the CPU and in the REU as follows:

RW0-RW31 = S0-S31 RWXS/1 - RWXS/3

(CPU private memory)

/RRW0/-/RRW31/ = S0-S31 RRWXS/1 - RRWXS/3

(cables to REU)

The arrangement of bits in the memory elements on one FT25 fast-access memory module is shown in figure 3-62.

3-55 SUM BUS. The sum bus is a set of 32 wires (57 wires when extended) for floating point operation which carries all output information, except address signals, from the arithmetic and control section of the CPU. The sum bus is also used for the transfer of certain types of data between registers in the CPU.

Information is gated onto the sum bus from the A-, the B-, the CS- and D-register, and from the adder. Since the sum bus bits are not stored in flip-flops, the terms are true only while the gating terms are true. Figure 3-63 shows the sum bus inputs and gating signals.

Outputs from the sum bus are sent to processor control panel indicators, decimal arithmetic chassis, core memory, private memory, and to external equipment via the direct input/output data lines. Gating signals for these outputs are shown in table 3-7.

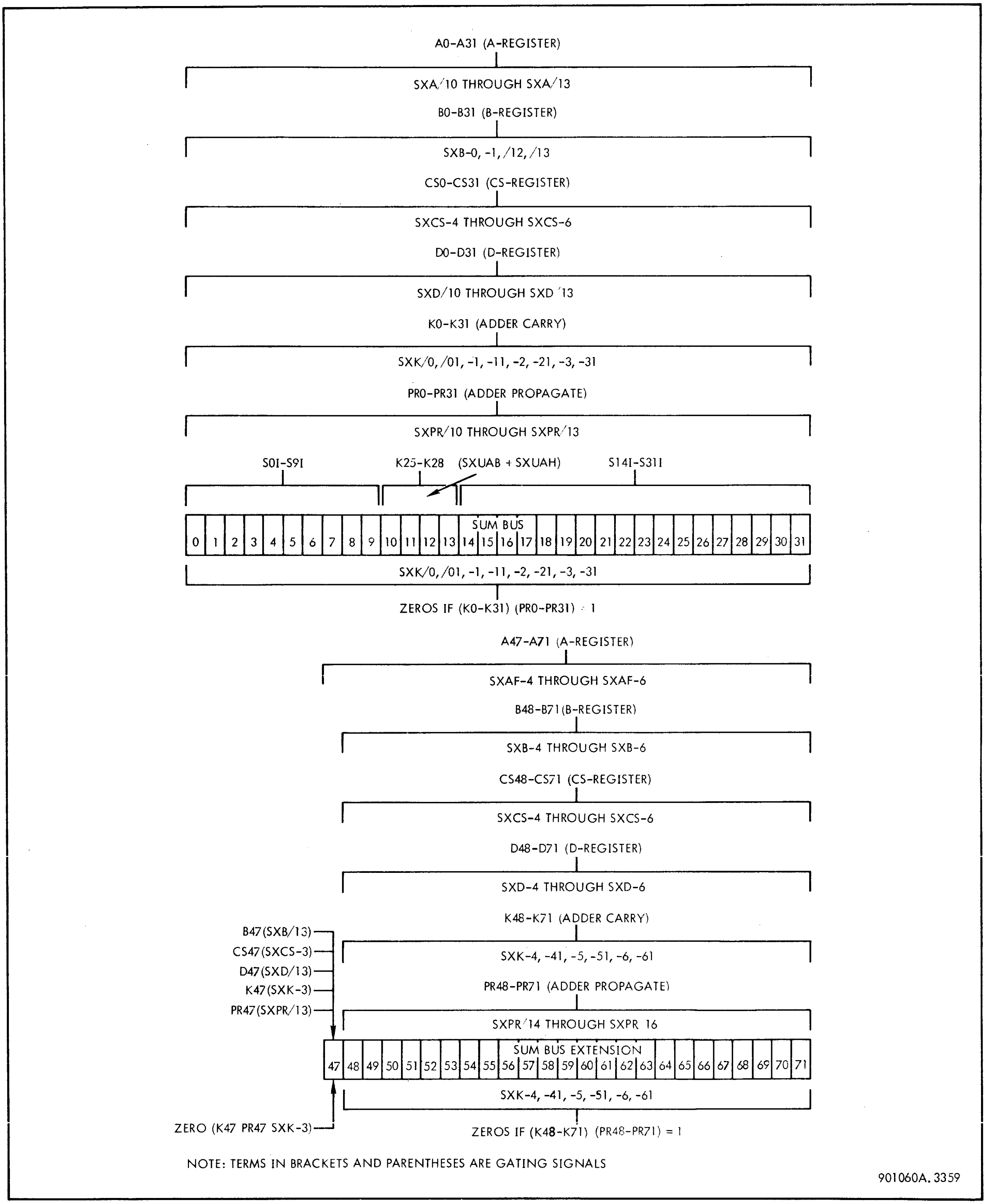
3-56 Adder

The adder performs the basic arithmetic and logical functions of the computer and is one of the outputs to the sum bus. The sum bus circuitry also gates other signals onto the sum bus. Figure 3-64 illustrates the adder and sum bus gating. For discussion purposes, both the adder circuitry and the sum bus gating circuitry are called sum bus gating. The sum bus gating performs four functions, as shown in the figure:

a. Adder Operation. Logical and arithmetic operations performed on the contents of the A-, D-, and CS-register contents

b. Register Gating. Direct gating of the A-, B-, CS-, D-, and P-registers to the sum bus

c. Display Lamp Gating. Gating of the sum bus outputs to the PCP DISPLAY lamps



NOTE: TERMS IN BRACKETS AND PARENTHESES ARE GATING SIGNALS

Figure 3-63. Sum Bus Inputs and Gating Signals

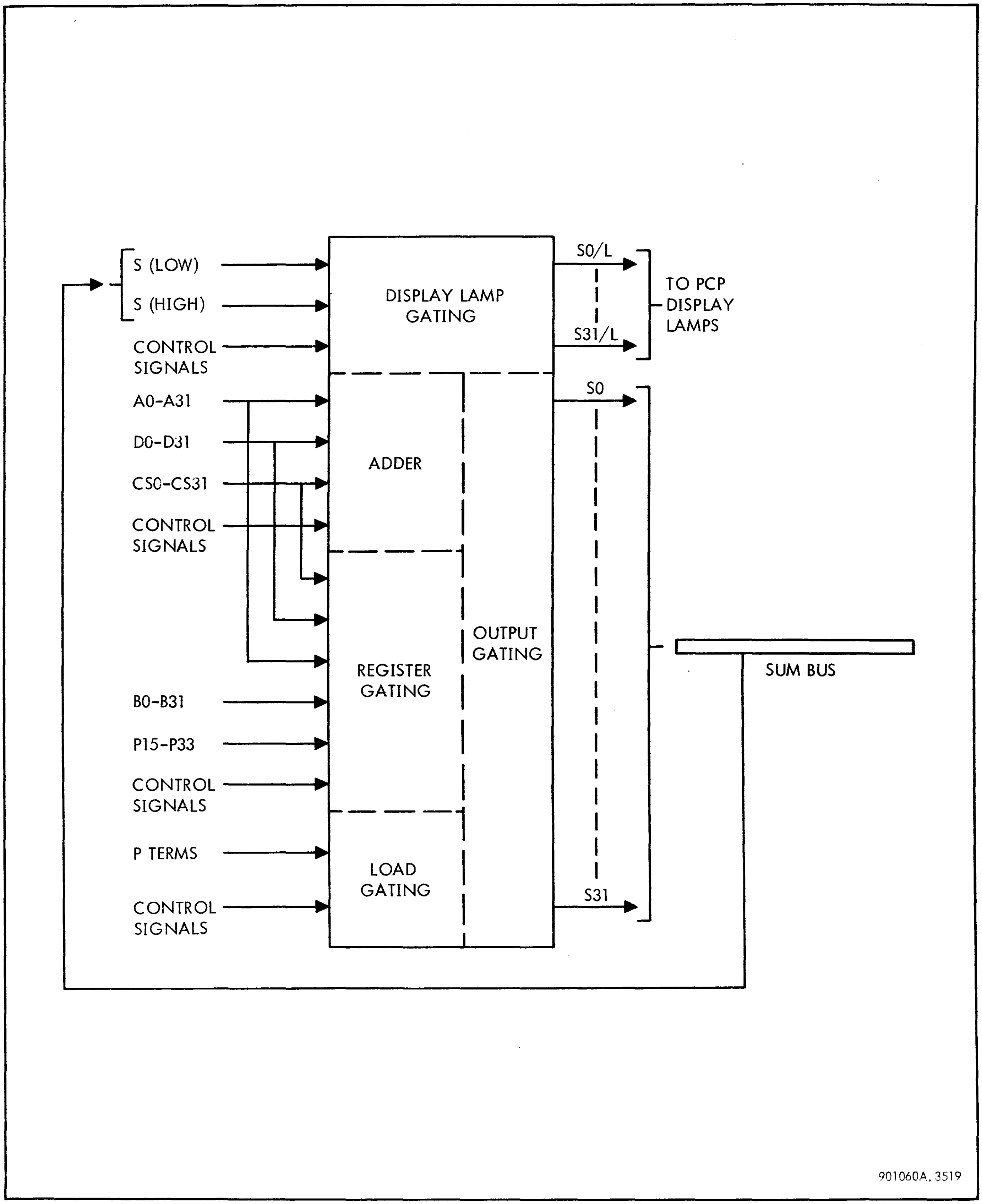


Figure 3-64. Adder and Sum Bus Gating, Block Diagram

d. Load Gating. Gating of manually entered data (PCP LOAD switch entry) to the sum bus

Table 3-8 lists the functions that the sum bus gating performs, the controlling signals for each function, and the prior conditions for the performance of the function. The control signals listed in the table are generated by the operation code control logic circuits and are described in the operation code descriptions in this section.

In a computer without the floating point option, there are 32 individual adder and sum bus gating combinations, one for each of the 32 sum bus outputs, S0 through S31, similar to the circuit in figure 3-65. A computer with the floating point option utilizes 25 additional adder and sum bus circuits to perform the floating point operation. In the following paragraphs, discussion of the four sum bus gating functions is with reference to figure 3-65.

3-57 ADDER OPERATION. Parallel addition is used in the adder, that is, all bits in one register are added at the same time to the corresponding bits in another register. Figure 3-66 shows the logical operation of each of the 32 individual adder stages in the sum bus gating. Adder stage N sums the Nth bit of the A-, CS-, and D-registers for all arithmetic and logical functions listed in table 3-8. The arithmetic and logical functions are divided into five groups: straight addition, exclusive OR addition, AND and OR functions, complementing, and upward alignment.

The highest order adder stage is adder 0, the lowest order is adder 31. Adder N designates any adder (0 through 31); adder N+1 is the adder one order below adder N.

Each adder stage is a full adder which generates a sum signal (SN), a propagate signal (PRN), and a generate signal (GN). Signals PRN and GN are developed in place of a carry. They control certain logical operations and are used in developing carry signals for higher order adders. One carry signal, KN, decoded from the previous (lower order) GN and PRN signals, may be received by each adder stage; a carry is added to the Nth-bit sum of the A-, CS-, and D-registers. To accelerate adder operation, propagate, generate, and carry signals from lower order adders are gated together to produce carry signals to individual adders. Thus, a carry to an adder may be immediately detected rather than waiting until it is propagated over a long chain. Carry gating is discussed later in this section.

3-58 Generate Signal. The generate signal is used to develop carry signals for higher order adders and is routed to the higher order carry gating. Generate signal GN from adder N is generally true whenever two of the three bits in bit positions AN, CSN, and DN are ones, regardless of the carry from the lower orders. In figure 3-66, for example, signal GN is true whenever signals AN and DN are true (the CS-register has been previously cleared). For every adder function:

GN = AN DN + AN CSN NDN GX
+ NAN CSN DN GX

where:

GN = generate signal from adder N

AN, DN, CSN = Nth bit of applicable register

GX = enabling signal for portion of generate gating

Table 3-7. Sum Bus Output Gating

Signal Name	Logic Element	Destination	Equations
RW	Buffer	Private memory	RW0-RW31 = (S0-S31) RWXS /RRW0/-/RRW31/ = (S0-S31) RRWXS
SX/L (X = 0-31, 56-71)	Lamp driver	Processor control panel indicators	S16/L-S31/L = (S16-S21) NKSHI + (S56-S71) KSHI where KSHI specifies high-order bits 56-71
/MB/	Cable driver	Core memory	/MB0/-/MB31/ = (S0-S31) MBXS
/DU/	Cable driver	Decimal arithmetic chassis	/DU0/-/DU7/ = DU0/1-DU7/1 + . . . DU0/1-DU0/4 = (S24-S31) DUXS + . . .
/DIO/	Cable driver	External equipment via direct output	/DIO0/-/DIO31/ = (S0-S31) DIOXS

Table 3-8. Sum Bus Gating Functions

FUNCTION Arithmetic and Logical Functions	CONTROL SIGNALS											PRIOR CONDITIONS
	SXA	SXB	SXCS	SXD	GX	PRX	SXK	SXPR	SXP	SXUAB	SXUAH	
(A plus D) → S	0	0	0	0	1	1	1	1	0	0	0	0's → CS
(A plus CS) → S	0	0	0	0	1	1	1	1	0	0	0	0's → D
(D plus CS) → S	0	0	0	0	1	1	1	1	0	0	0	0's → A
(A ⊕ D) → S	0	0	0	0	1	1	0	1	0	0	0	0's → CS
(A ⊕ CS) → S	0	0	0	0	1	1	0	1	0	0	0	0's → D
(D ⊕ CS) → S	0	0	0	0	1	1	0	1	0	0	0	0's → A
(A or D) → S	1	0	0	1	1	0	0	0	0	0	0	0's → CS
(A and D) → S	0	0	0	0	1	0	0	1	0	0	0	1's → CS
(-A) → S	0	0	0	0	0	1	1	1	0	0	0	1's → CS 0's → D NGX ⇒ 1 → K31
(-D) → S	0	0	0	0	0	1	1	1	0	0	0	1's → CS 0's → A NGX ⇒ 1 → K31
A upward aligned (byte): A24-A31 → S16-S23 A24-A31 → S8-S15 A24-A31 → S0-S7 A24-A31 → S24-S31	1 (A24 to A31 only)	0	0	0	1	0	0	0	0	1	0	0's → D 0's → CS ↓
A upward aligned (halfword): A16-A31 → S0-S15 A16-A31 → S16-S31	1 (A16 to A31 only)	0	0	0	1	0	0	0	0	0	1	↓
D upward aligned (byte): D24-D31 → S16-S23 D24-D31 → S8-S15	0	0	0	1 (D24 to D31 only)	1	0	0	0	0	1	0	0's → A 1's → CS ↓

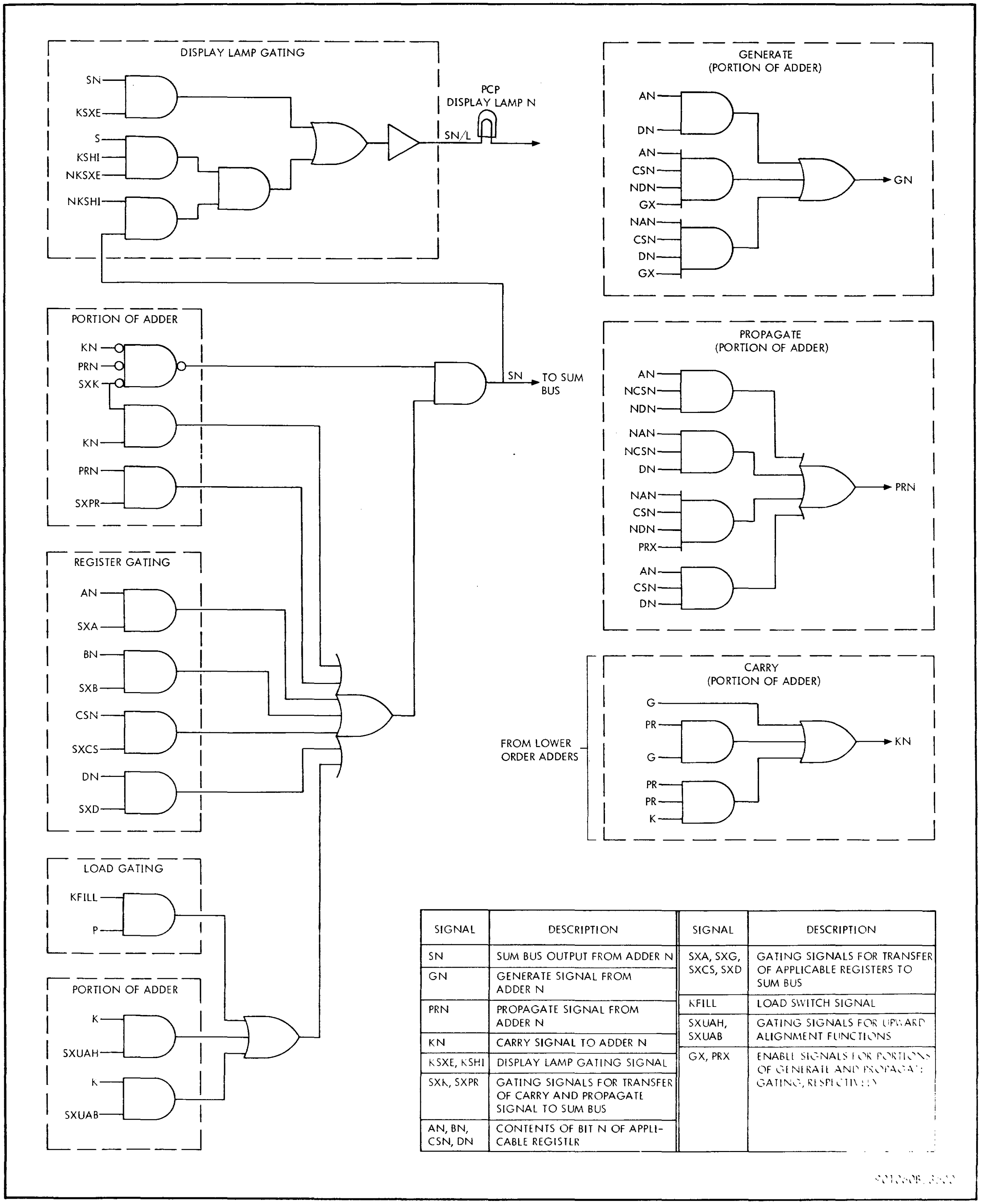
(Continued)

Table 3-8. Sum Bus Gating Functions (Cont.)

FUNCTIONS Arithmetic and Logical Functions	CONTROL SIGNALS											PRIOR CONDITIONS
	SXA	SXB	SXCS	SXD	GX	PRX	SXK	SXPR	SXP	SXUAB	SXUAH	
D24-D31 → S0-S7 D24-D31 → S24-S31	0	0	0	1	1	0	0	0	0	1	0	0's → A 1's → CS
D upward aligned (halfword): D16-D31 → S0-S15 D16-D31 → S16-S31	0	0	0	1 (D16 to D31 only)	1	0	0	0	0	0	1	
Register Gating Functions												
A → S	1	0	0	0	0	0	0	0	0	0	0	
B → S	0	1	0	0	0	0	0	0	0	0	0	
CS → S	0	0	1	0	0	0	0	0	0	0	0	
D → S	0	0	0	1	0	0	0	0	0	0	0	
P → S	0	0	0	0	0	0	0	0	1	0	0	

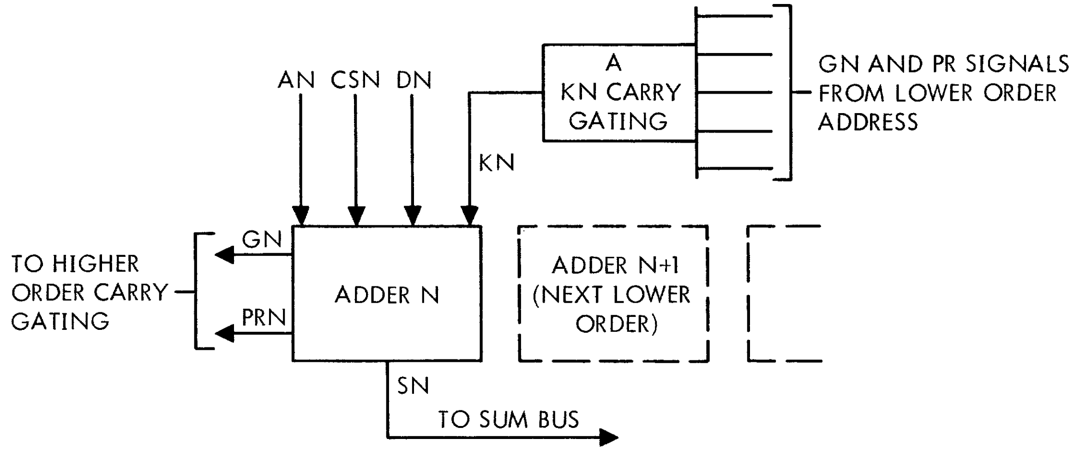
Notes

SXA, SXB, SXCS, SXD, SXP	Gating signal for transfer of applicable register to sum bus
GX, PRX	Enabling signal for portion of generate and propagate gating, respectively
SXK, SXPR	Gating signal for transfer of carry and propagate signals, respectively, to sum bus
SXUAB, SXUAH	Gating signal for upward alignment of byte and upward alignment of halfword to sum bus
⊕	Exclusive OR
-	Negation (two's complement)
A → B	A-register transferred to B-register
⇒	Implies



SIGNAL	DESCRIPTION
SN	SUM BUS OUTPUT FROM ADDER N
GN	GENERATE SIGNAL FROM ADDER N
PRN	PROPAGATE SIGNAL FROM ADDER N
KN	CARRY SIGNAL TO ADDER N
KSXE, KSHI	DISPLAY LAMP GATING SIGNAL
SXK, SXPR	GATING SIGNALS FOR TRANSFER OF CARRY AND PROPAGATE SIGNAL TO SUM BUS
AN, BN, CSN, DN	CONTENTS OF BIT N OF APPLICABLE REGISTER
SXA, SXG, SXCS, SXD	GATING SIGNALS FOR TRANSFER OF APPLICABLE REGISTERS TO SUM BUS
KFILL	LOAD SWITCH SIGNAL
SXUAH, SXUAB	GATING SIGNALS FOR UPWARD ALIGNMENT FUNCTIONS
GX, PRX	ENABLE SIGNALS FOR PORTIONS OF GENERATE AND PROPAGATE GATING, RESPECTIVELY

Figure 3-65. Typical Adder and Sum Bus Gating, Logic Diagram



ADD AN + DN

AN	DN	GN	PRN	KN	SN
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	0	0	0
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	1

NO CARRY SIGNAL (NKN) FROM LOWER ORDERS

CARRY SIGNAL (KN) FROM LOWER ORDERS

901060A. 3501

Figure 3-66. Adder Logical Operation

3-59 Propagate Signal. The propagate signal PRN is used in developing the sum bus output for adder N, SN; it is also used to develop carry signals for higher order adders and is routed to higher order carry gating. Propagate signal PRN from adder N is generally true whenever one or three bits in flip-flops AN, CSN, and DN are ones, regardless of the carry from the lower orders. In figure 3-66, signal PRN is true whenever either signal AN or DN is true (exclusive OR operation; CS-register has been cleared). For every adder function:

$$\begin{aligned} PRN &= AN\ NCSN\ NDN + NAN\ NCSN\ DN \\ &\quad + NAN\ CSN\ NDN\ PRX + AN\ CSN\ DN \end{aligned}$$

where:

PRN = propagate signal from N-adder

PRX = enabling signal for portion propagate gating

3-60 Carry Signal. Carry signal KN to adder N is added to signals AN, CSN, and DN. Signal KN is true when there is a generate signal from the next lower order adder, G(N+1); a generate signal from adder N+1 always results in a carry to adder N whenever lower order additions are performed. For example:

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 1 \\ \hline 1 & 1 & 0 \end{array} \quad \text{Carry}$$

GN G(N+1)

There is also a carry to adder N whenever adder N+1 produces a propagate signal and there is a carry signal to adder N+1, as in the following example:

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 0 & 1 \\ \hline 1 & 0 & 0 \end{array} \quad \text{Carry}$$

GN PR(N+1)

The partial equation for a carry to adder N is:

$$KN = G(N+1) + PR(N+1)\ \ K(N+1) + \ldots$$

In some additions, a chain of carries is produced by an addition in one of the least significant orders. For example:

$$\begin{array}{cccccccccc l}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \text{Carry} \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & \\ \hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
\uparrow & & & & & & & & & & \\
\text{MSD} & & & & & & & & & &
\end{array}$$

The chain of carries is started at the extreme right of the addition. The final carry takes a relatively long time to reach the most significant digit (MSD); if a carry for the MSD could be detected before it is transmitted by the previous order (that is, before going through the chain), addition would be faster. This is accomplished by gating in the adder. Carry gating for any adder N is composed of AND-gated inputs of preceding (lower order) propagate, generate, and carry terms. The carry for adder N is enabled whenever there are digits in the lower orders that produce the proper PR, G, and K terms which are digits that result in a carry to adder N. The carry to adder N is then immediately detected and is sent to the adder. The carry gating equations that describe these configurations follow.

K NUMBER

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4

Equation No.

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4

Equation No.

Equations for carry (KN) for bit N:

1. $KN = G(N+1) + PR(N+1)\ K\ (N+1)$

2. $$\begin{aligned} KN &= G(N+1) + PR(N+1)\ G\ (N+2) \\ &\quad + PR(N+1)\ PR(N+2)\ K(N+2) \end{aligned}$$

3. $$\begin{aligned} KN &= G(N+1) + PR(N+1)\ G(N+2) \\ &\quad + PR(N+1)\ PR(N+2)\ G(N+3) \\ &\quad + PR(N+1)\ PR(N+2)\ PR(N+3)\ K(N+3) \end{aligned}$$

4. $$\begin{aligned} K3 &= K19 = [G\ (N+1) - (N+4)] \\ &\quad + PR\ [(N+1) - (N+4)]\ G\ [(N+5) - (N-8)] \\ &\quad + PR\ [(N+1) - (N+4)]\ PR\ [(N+5) - (N-8)] \\ &\quad G\,[(N+9) - (N+12)] + PR\ [(N+1) - (N+4)] \\ &\quad PR[\ (N+5) - (N+8)]\,PR\ [(N+9) - (N+12)] \\ &\quad K(N+12) \end{aligned}$$

$$\begin{aligned} K7 &= K23 = G\ [(N+1) - (N+4)] \\ &\quad + PR\ [(N+1) - (N+4)]\ G\ [(N+5) - (N-8)] \\ &\quad + PR\ [(N+1) - (N-4)]\ PR\ [(N+5) - (N-8)] \\ &\quad K(N+8) \end{aligned}$$

$$\begin{aligned} K11 &= K27 = G\ [(N+1) - (N+4)] \\ &\quad + PR\ [(N+1) - (N+4)]\ K(N+4) \end{aligned}$$

$$\begin{aligned} K15 &= G1619 + PR1619\ G2023 \\ &\quad + PR1619\ PR2023\ G2427 \\ &\quad + PR1619\ PR2023\ PR2427\ G2831 \\ &\quad + PR1631\ K31 + I(PH6\ FAST) \end{aligned}$$

$$\begin{aligned} K31 &= (K00H + NFASHFL)\ (BO\ CS32 \\ &\quad + PR32\ G33 + NGX) \end{aligned}$$

Note

A logic term in the form AB1115 means that logic signals AB11 through AB15 are true simultaneously. That is, AB1115 = AB11 AB12 AB13 AB14 AB15

There are four sets of conditions that result in a carry to adder 20, for example:

$$\begin{array}{lcccc l}
\text{Bit} & 20 & 21 & 22 & 23 & \text{Carry} \\
 & 0 & 1 & X & X & \\
 & 0 & 1 & X & X & \\ \hline
 & 1 & 0 & X & X &
\end{array}
\qquad
\begin{array}{lcccc}
\text{Bit} & 20 & 21 & 22 & 23 \\
 & 0 & 0 & 1 & X \\
 & 0 & 1 & 1 & X \\ \hline
 & 1 & 0 & 0 & X
\end{array}$$

$$\begin{array}{cccc l}
0 & 0 & 1 & 1 & \text{Carry} \\
0 & 1 & 0 & 1 & \\ \hline
1 & 0 & 0 & 0 &
\end{array}
\qquad
\begin{array}{cccc}
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\ \hline
1 & 0 & 0 & 0
\end{array}$$

The remainder of the right-hand side of the partial equation

$$KN = G(N+1) + PR(N+1)\quad K(N+1) + \ldots$$

is composed of propagate and generate terms from the preceding adders. (See preceding paragraphs for an explanation of the equation.)

3-61 Sum Bus Output. Sum bus output SN for any adder N in the sum bus gating is always false if there are, simultaneously, a carry to the adder and a propagate signal from the adder.

```
    0 1   Carry
    1 1
    ---
  1 0 0

  PRN G(N+1)
```

Either of these alone, however, produces a true output for SN. The equation for the adder N portion of the sum bus output is:

$$SN = N(KN\ PRN\ SXK)\ (KN\ SXK + PRN\ SXPR + \ldots)$$

where:

SXK = enabling term for carry gating
SXPR = enabling term for propagate gating
SN = sum bus output for bit N

Figure 3-66 includes a truth table for an individual adder that illustrates a sample addition.

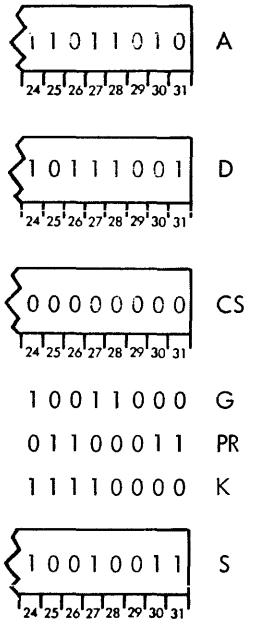
3-62 <u>Addition</u>. The adder adds the contents of two of the three registers together in a straight binary addition. The remaining register is cleared before the addition takes place. The sum output of each adder stage goes directly to the sum bus.

Figures 3-67 through 3-69 illustrate the three additive functions and list the simplified logic equations for each function. Only the last byte of each register is shown. Applicable control signals are defined in table 3-8.

3-63 <u>Exclusive OR Operation</u>. The adder adds the contents of two of the three registers together in an exclusive OR operation, that is, there is a true output for SN when there is a one in the Nth bit position of one, but not both registers. Since the propagate signal is developed according to the same rules as an exclusive OR addition, it alone is used as a signal to the sum bus gating; a propagate signal produces a true sum bus output. Carry and generate signals do not enter into the gating logic. The third register is cleared before the operation takes place.

Figures 3-70 through 3-72 illustrate the three exclusive OR operations and list the simplified logic equations of each. Only the last byte of each register is shown. Applicable control signals are defined in table 3-8.

3-64 <u>AND and OR Operations</u>. The AND operation is performed by filling the CS-register with ones and by adding the contents of the A- and D-registers. The propagate signals, developed when there is a one in bit position AN and a one in bit position DN, are the only enabling signals to the sum bus gating used in this operation. A true propagate signal produces a true sum bus output.



$$GN = AN\ DN$$
$$PR = AN\ NDN + NAN\ DN$$
$$KN = G(N+1) + PR(N+1)\ K(N+1) + \cdots$$
$$SN = N(KN\ PRN)\ (KN + PRN + \cdots)$$

901060A. 3503

Figure 3-67. (A Plus D) Gated to S

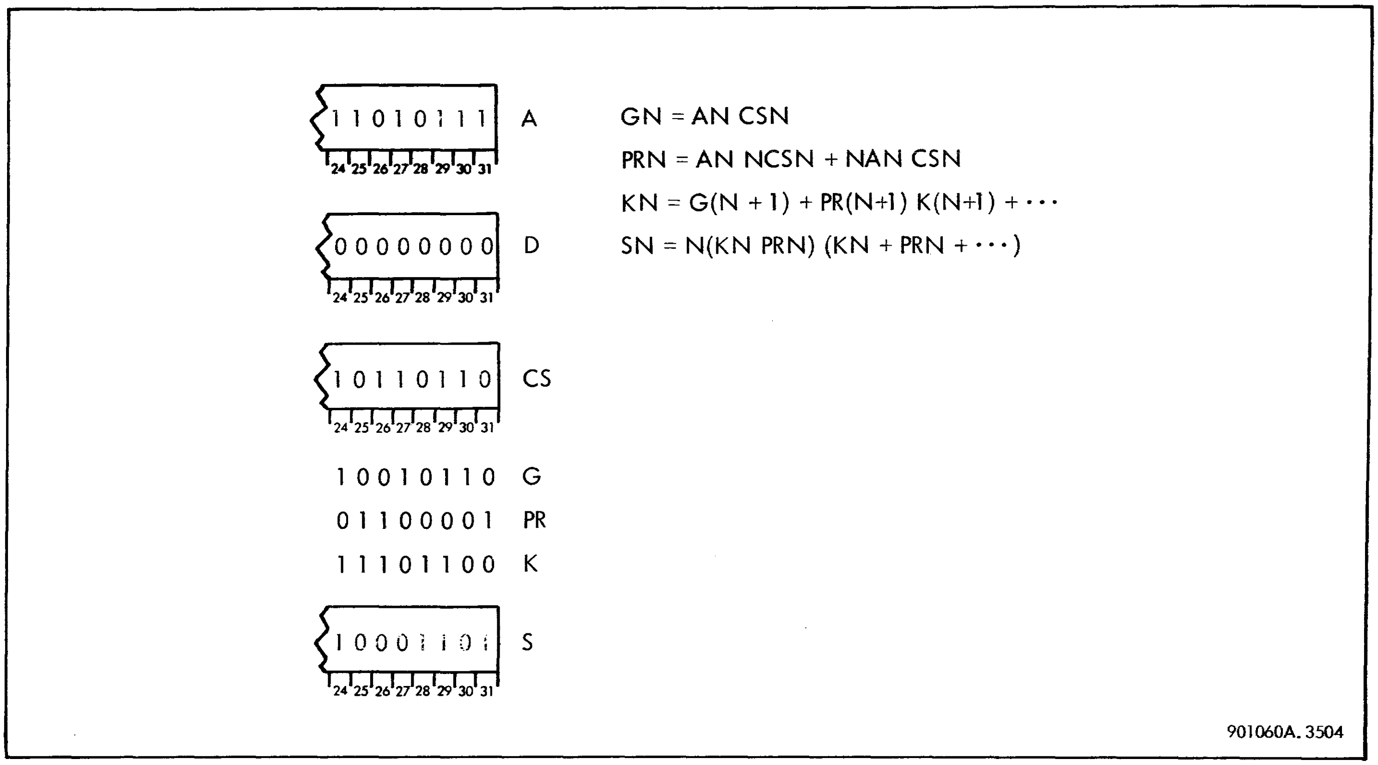


Figure 3-68. (A Plus CS) Gated to S

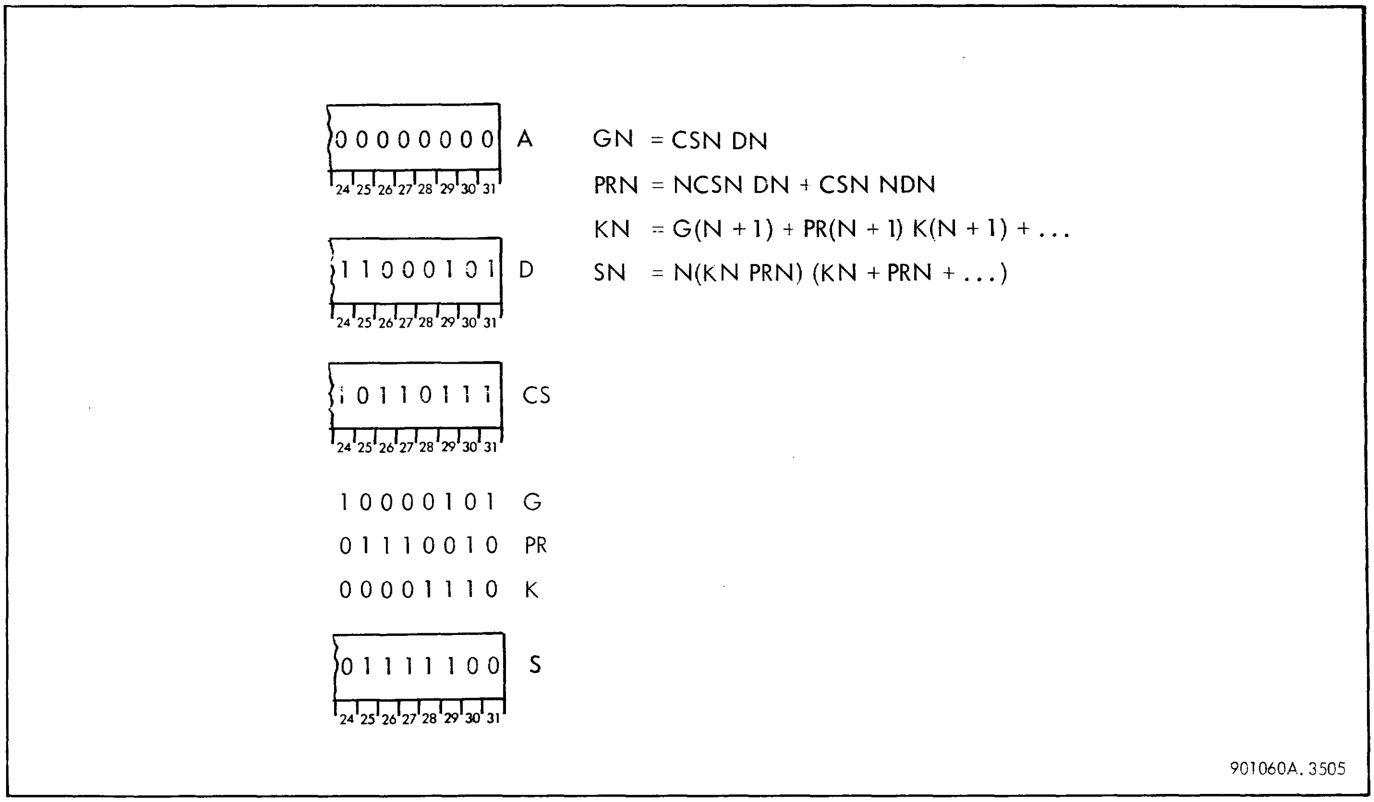
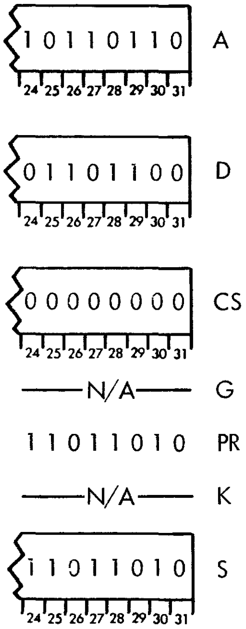


Figure 3-69. (D Plus CS) Gated to S



GN = N/A

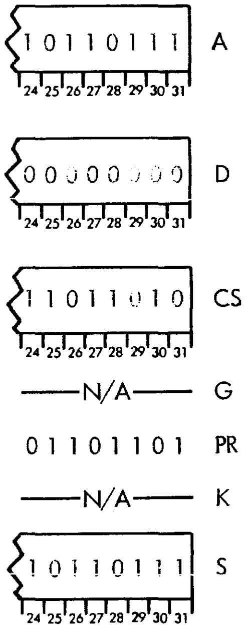
PRN = AN NDN + NAN DN

KN = N/A

SN = PRN

901060A. 3506

Figure 3-70. (A ⊕ D) Gated to S



GN = N/A

PR = AN NCSN + NAN CSN

KN = N/A

SN = PRN

901060A. 3507

Figure 3-71. (A ⊕ CS) Gated to S

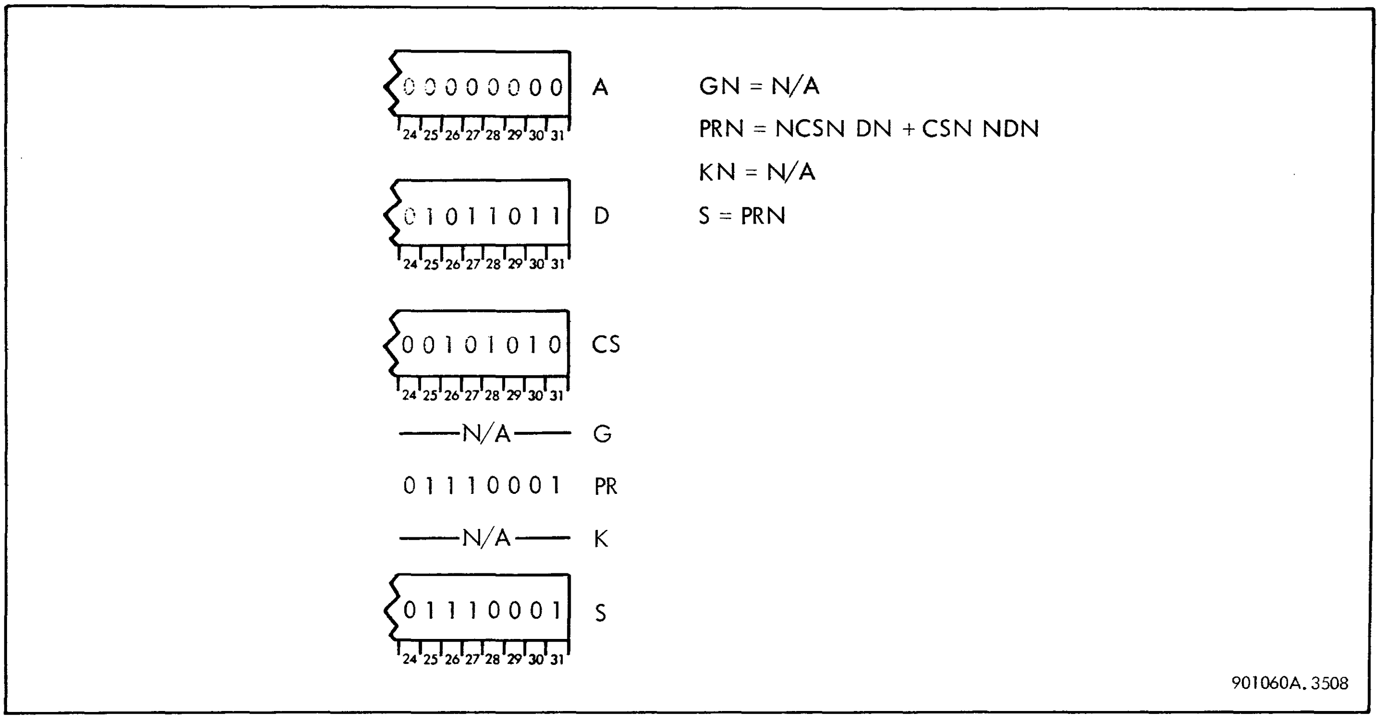


Figure 3-72. (D $\oplus$ CS) Gated to S

The OR operation is not strictly an adder function since the two registers involved are simply gated to the sum bus at the same time. The adder acts as a two-input OR gate. The CS-register is cleared before the operation takes place.

Figures 3-73 and 3-74 illustrate the OR and AND operations, respectively, and list the simplified logic equations for each. Only the last byte of each register is shown. Applicable control signals are contained in table 3-8.

3-65. Complementing. This function is used to form the two's complement of the contents of either the A- or the D-register and to gate the result to the sum bus. The two's complement of any binary number is defined as $2^x - N$, where x is the number of digits in a register. The two's complement of a binary number can be calculated by changing all the ones in the number to zeros and all of the zeros to ones and by adding one to the result. The adder performs two's complementing in this manner. The CS-register is filled with ones, and the register containing the number to be complemented is added to the contents of the CS-register; this addition effectively changes all zeros to ones and ones to zeros in the number. Since the carry for bit 31 is set initially to one, one is added to the result at the same time that the two registers are added.

Figures 3-75 and 3-76 illustrate the two complementing operations and list the simplified logic equations for each function. Only the last byte of each register is shown. Applicable control signals are defined in table 3-8.

3-66 Upward Alignment. This function is used to:

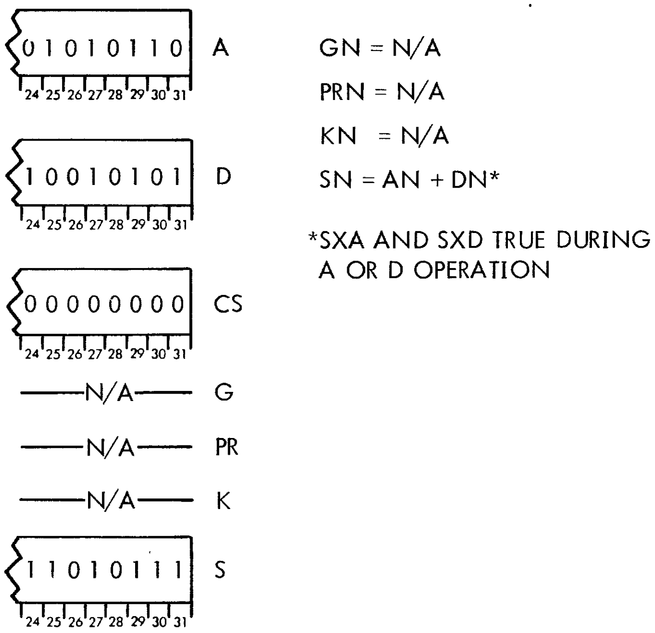
a. Transfer the last byte (bits 24 through 31) in the A- or D-register to all byte positions on the sum bus (bits S0 through S7, S8 through S15, S16 through S23, and S24 through S31). Figure 3-77 illustrates this operation.

b. Transfer the last halfword (bits 16 through 31) in the A- or D-registers to the two halfword positions on the sum bus (bits S0 through S15 and S16 through S31). Figure 3-77 shows this operation.

To perform the upward alignment, the CS-register is filled with ones, and the register not containing the byte or halfword to be transferred is cleared. The registers are then added and the choice of halfword or byte alignment is made by control signals SXUAH or SXUAB, respectively.

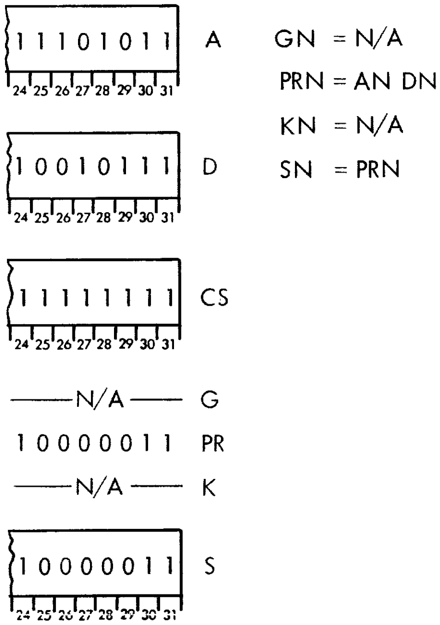
Figures 3-78 through 3-81 illustrate upward alignment for bytes and halfwords and list the simplified logic equations for each. Applicable control signals are shown in table 3-8.

3-67 REGISTER GATING. The register gating portion of the sum bus gating transfers the contents of different registers onto the sum bus with application of the proper control signal. The register gating functions are listed in table 3-8. Registers A, B, CS, and D are gated so that bits 0 through 31 go to sum bus outputs S0 through S31, respectively. Register P is gated so that bits 15 through 31 go to sum bus outputs S15 through S31, respectively; bit 32 goes to bit S0, and bit 33 goes to S1.



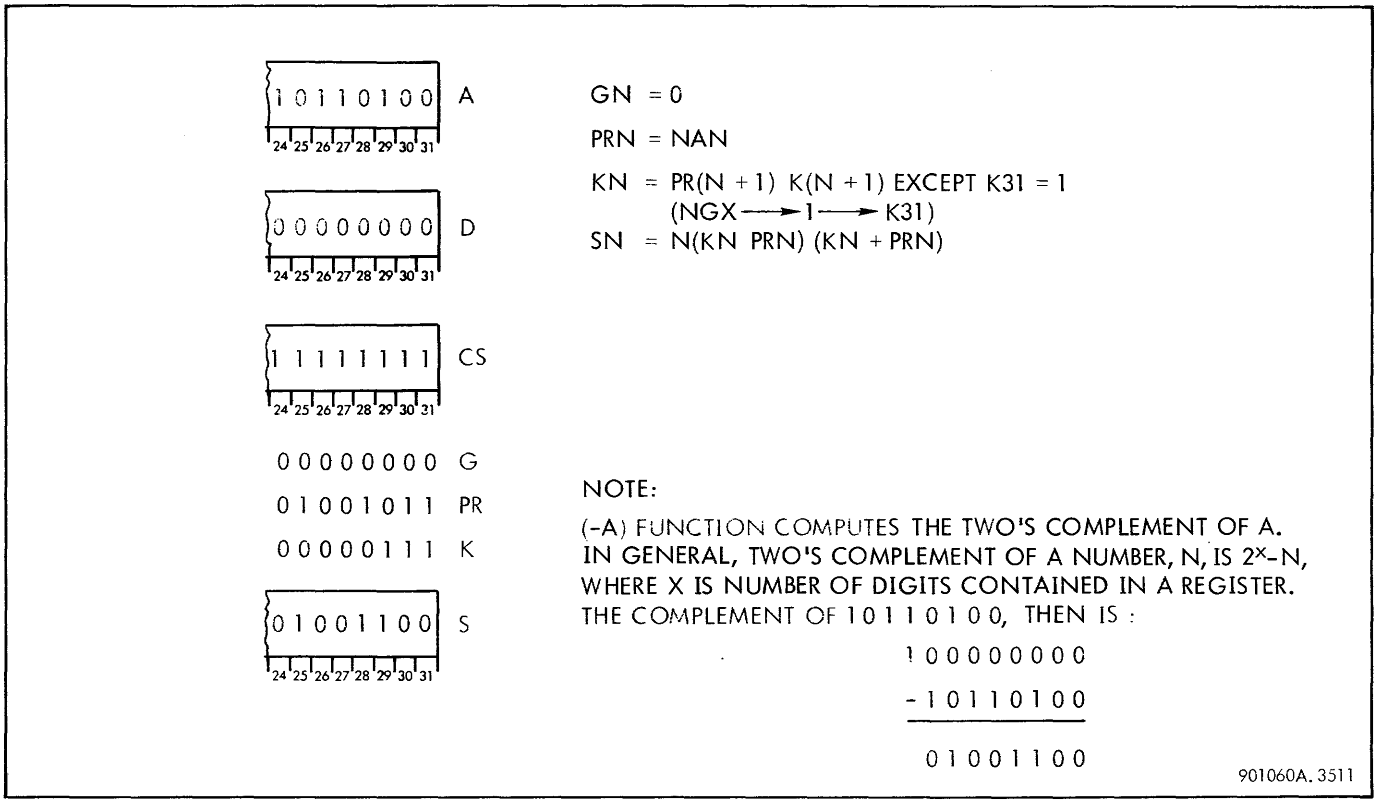
901060A. 3509

Figure 3-73. (A OR D) Gated to S



901060A. 3510

Figure 3-74. (A AND D) Gated to S



GN = 0

PRN = NAN

KN = PR(N + 1) K(N + 1) EXCEPT K31 = 1
(NGX ⟶ 1 ⟶ K31)

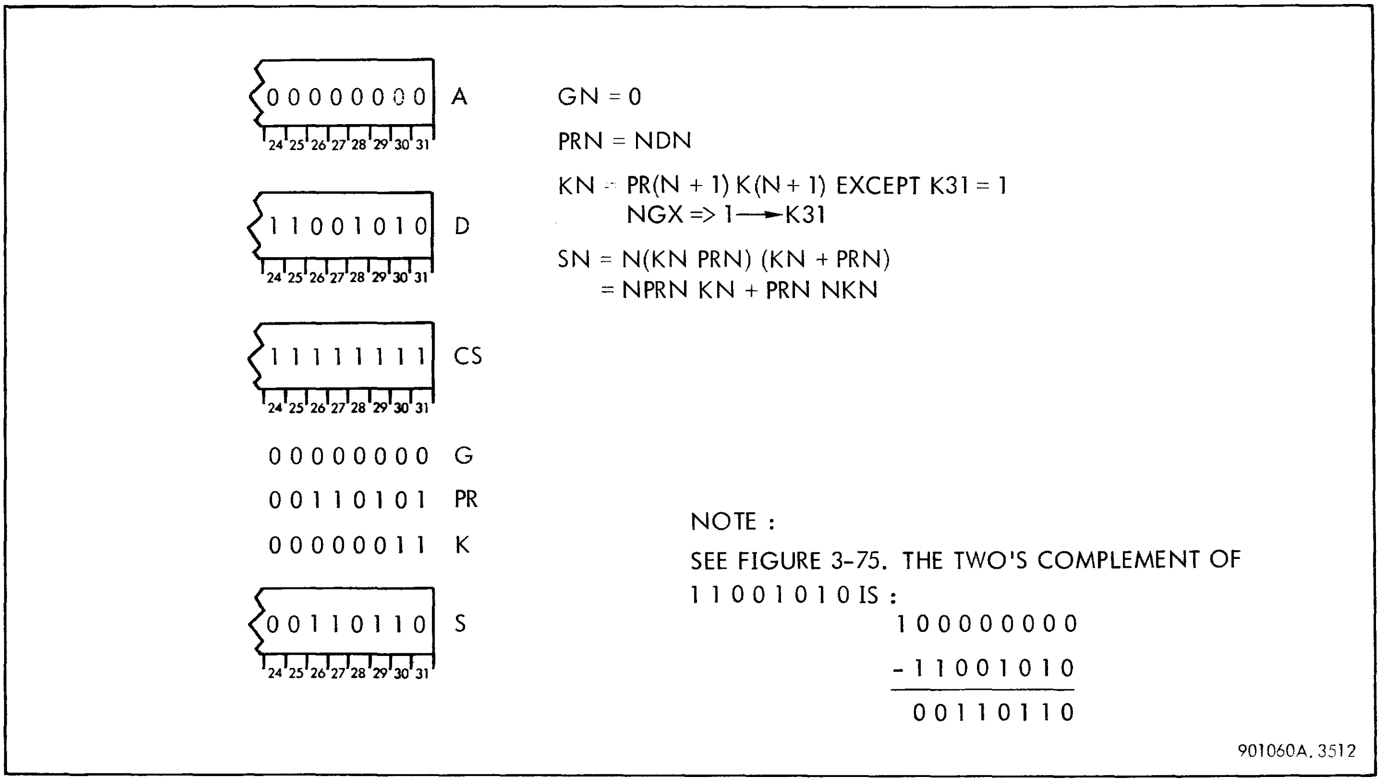
SN = N(KN PRN) (KN + PRN)

NOTE:

(-A) FUNCTION COMPUTES THE TWO'S COMPLEMENT OF A. IN GENERAL, TWO'S COMPLEMENT OF A NUMBER, N, IS 2^x-N, WHERE X IS NUMBER OF DIGITS CONTAINED IN A REGISTER. THE COMPLEMENT OF 1 0 1 1 0 1 0 0, THEN IS :

```
 1 0 0 0 0 0 0 0 0
 - 1 0 1 1 0 1 0 0
 -----------------
   0 1 0 0 1 1 0 0
```

Figure 3-75. Two's Complement of A Gated to S



GN = 0

PRN = NDN

KN = PR(N + 1) K(N + 1) EXCEPT K31 = 1
NGX => 1 ⟶ K31

SN = N(KN PRN) (KN + PRN)
= NPRN KN + PRN NKN

NOTE :

SEE FIGURE 3-75. THE TWO'S COMPLEMENT OF 1 1 0 0 1 0 1 0 IS :

```
 1 0 0 0 0 0 0 0 0
 - 1 1 0 0 1 0 1 0
 -----------------
   0 0 1 1 0 1 1 0
```

Figure 3-76. Two's Complement of D Gated to S

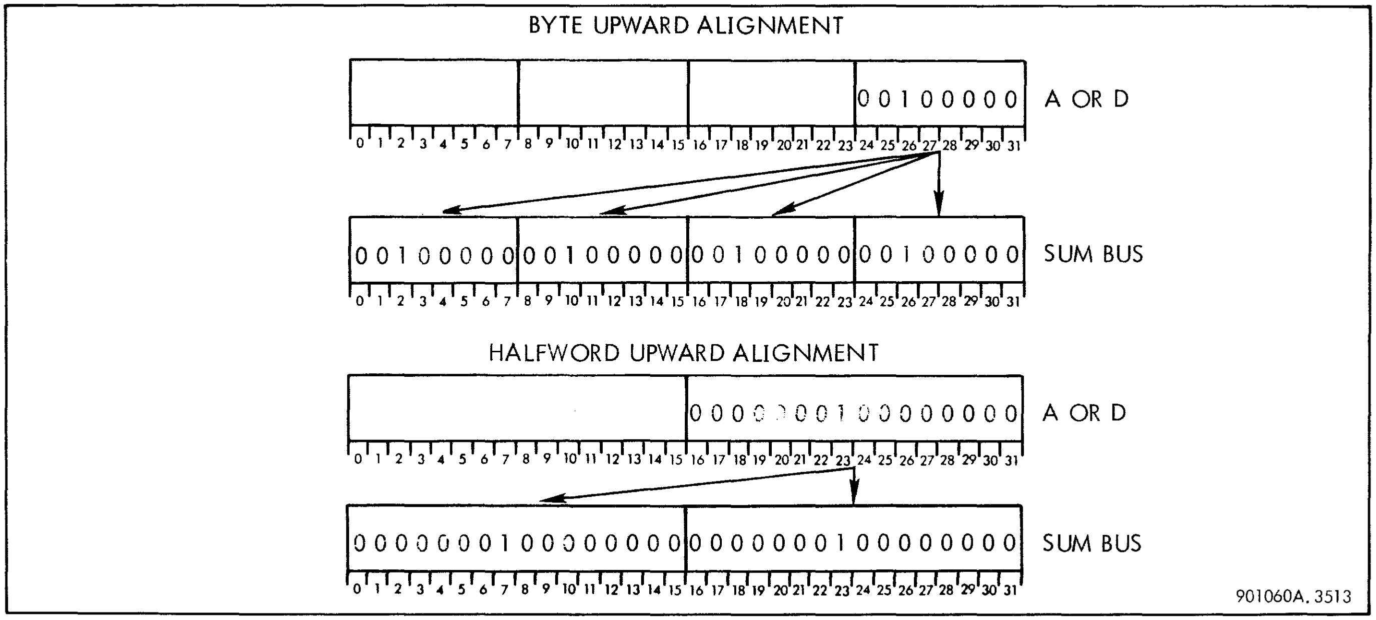
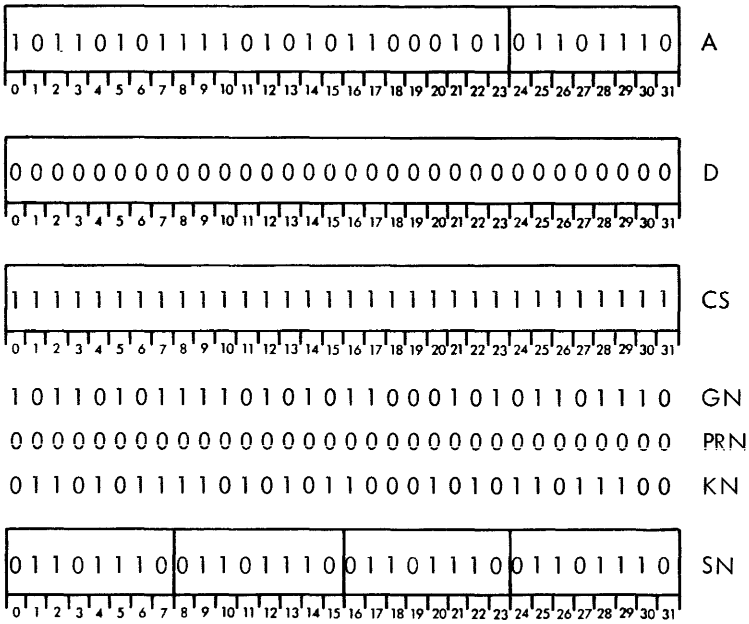


Figure 3-77. Upward Alignment, Functional Diagram



GN = AN

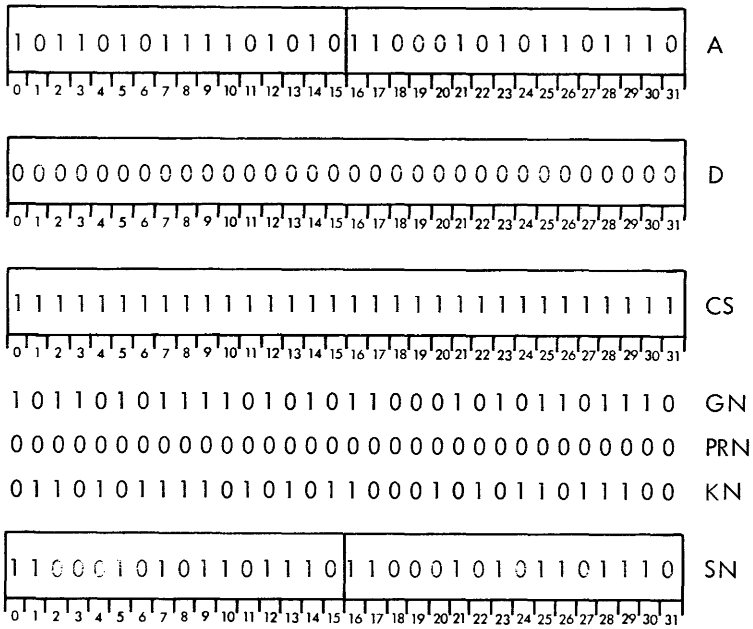
PRN = 0

KN = G(N + 1)

SN = KN' SXUAB
+ AN SXA] WHERE N' = N + 23 IF N = 0 TO 7
= N + 15 IF N = 8 TO 15
= N + 7 IF N = 16 TO 23

901060A. 3514

Figure 3-78. A-Register Upward Alignment (Byte)



```
1 0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 | 1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 0    A
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 0 0 0    D
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1    CS
1 0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 0    GN
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    PRN
0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0    KN
1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 0 | 1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 0    SN
```

GN = AN

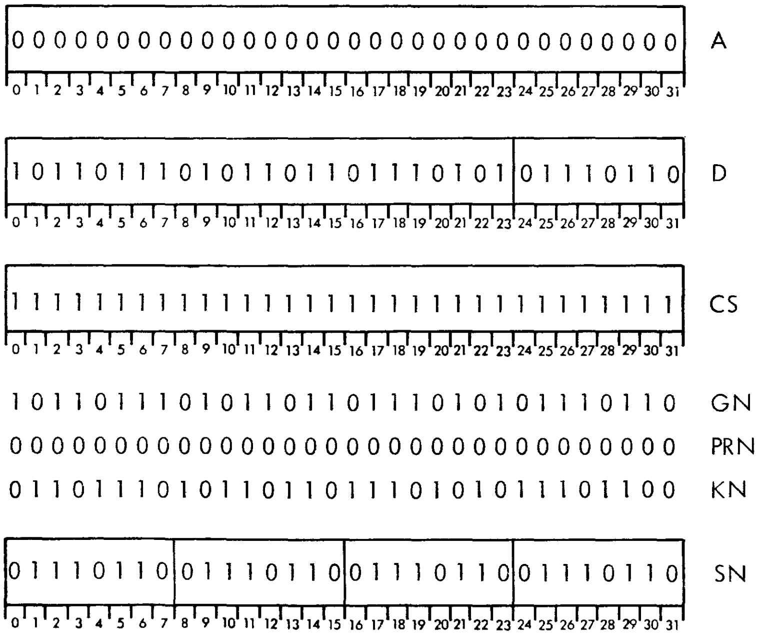
PRN = 0

KN = G(N + 1)

SN = KN' SXUAH
+ AN SXA] WHERE N' = N + 15 IF N = 0 TO 15

901060A. 3515

Figure 3-79. A-Register Upward Alignment (Halfword)



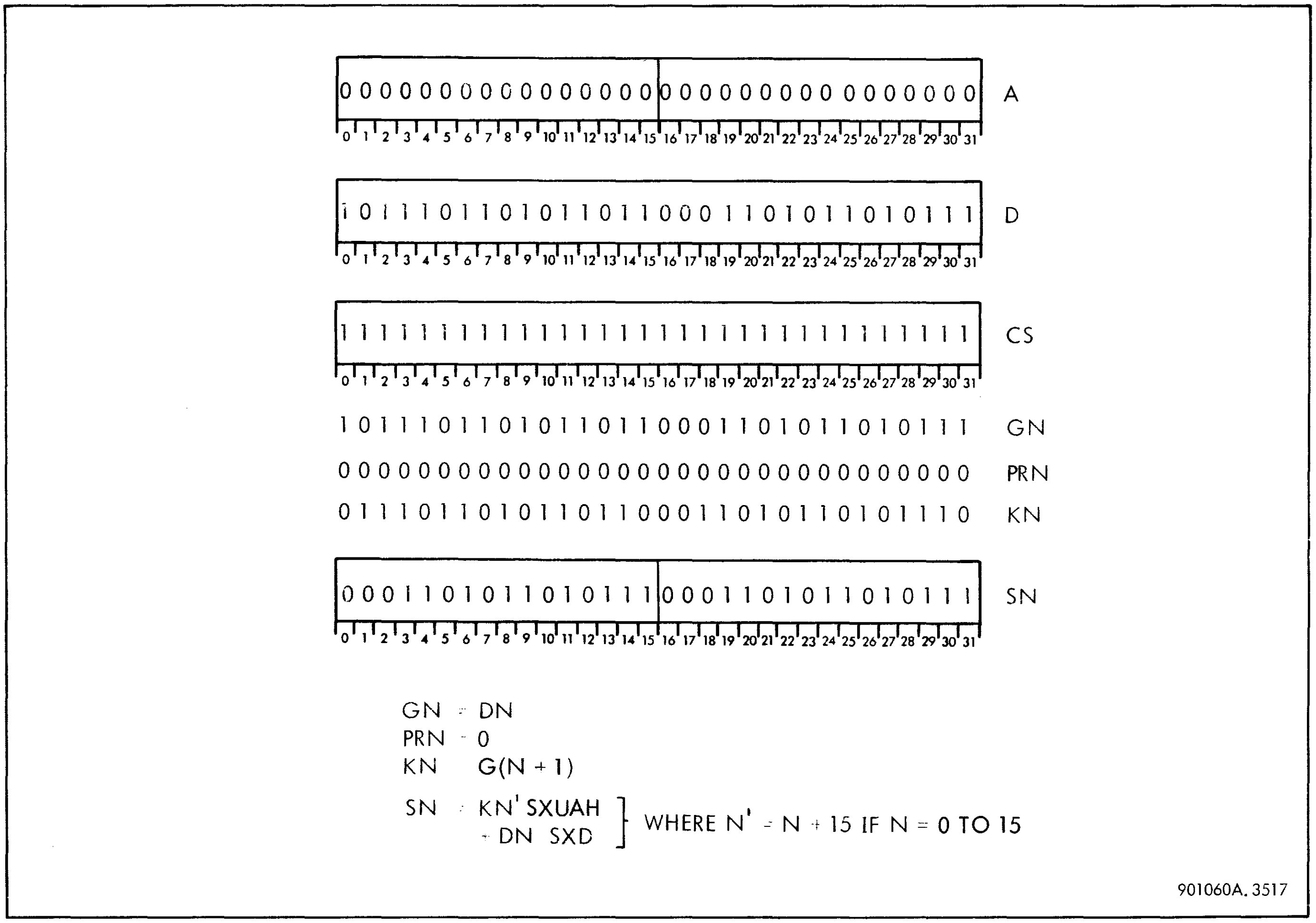
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    A
1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 | 0 1 1 1 0 1 1 0    D
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1    CS
1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0    GN
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    PRN
0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 0    KN
0 1 1 1 0 1 1 0 | 0 1 1 1 0 1 1 0 | 0 1 1 1 0 1 1 0 | 0 1 1 1 0 1 1 0    SN
```

GN = DN
PRN = 0
KN = G(N + 1)

SN = KN' SXUAB
+ DN SXA] WHERE N' = N + 23 IF N = 0 TO 7
N + 15 IF N = 8 TO 15
N + 7 IF N = 16 TO 23

901060A. 3516

Figure 3-80. D-Register Upward Alignment (Byte)



```
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   A
1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1     D
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     CS
1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1     GN
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     PRN
0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0     KN
0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 | 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1   SN
```

GN = DN

PRN = 0

KN = G(N + 1)

$$\left.\begin{array}{l} SN = KN'\,SXUAH \\ \quad + DN\ SXD \end{array}\right\} \text{WHERE } N' = N + 15 \text{ IF } N = 0 \text{ TO } 15$$

901060A. 3517

Figure 3-81. D-Register Upward Alignment (Halfword)

Sum bus outputs S2 through S14 have no output for P to S. The equation for the register gating portion of the sum bus output is:

$$SN = AN\ SXA + BN\ SXB + CSN\ SXCS$$
$$+ DN\ SXD + PN'\ SXP + \ldots$$

Where:

$N = 0 - 31$

$N' = 15 - 31$ when $N = 15 - 31$

$= 32$ when $N = 0$

$= 33$ when $N = 1$

3-68 DISPLAY LAMP GATING. Display lamp gating is not strictly a part of sum bus gating but is related to it. The display gating enables the sum bus outputs to be displayed on the PCP display lamps, if the REGISTER DISPLAY switch is ON, the CLOCK MODE switch is not set to CONT, and the REGISTER SELECT switch is set to the appropriate position. The REGISTER SELECT switch generates control signals which select the register to be gated to the sum bus for later gating to the DISPLAY lamps. Figure 3-82 illustrates the display lamp gating, and table 3-9 lists the possible displays and the logic equations for the gating.

3-69 LOAD GATING. The load gating portion of the sum bus gating enables decoded data from the UNIT ADDRESS switches that has been entered by the LOAD switch to be gated to the sum bus. The entire loading process and switch functions are described in the discussion of the PCP in this section (paragraphs 3-303 through 3-340).

3-70 FIXED-POINT OVERFLOW. Fixed-point overflow occurs when the sum of an addition is greater than $2^{32}-1$; that is, when the sum is greater than the largest number that can be held in a 32-bit register. The buffered gate PROBEOVER (figure 3-83) determines those instructions in which an arithmetic overflow can occur. The equation for the buffered gate PROBEOVER is:

$$PROBEOVER = PH5(FAS3 + FAS12 + FAS19 + FAS26)$$

The instructions during which PROBEOVER is true are:

Load Complement Doubleword

Load Complement Word

Load Absolute Doubleword

Load Absolute Word

Add Immediate

Add Word

Add Halfword

Add Doubleword

Subtract Word

Subtract Doubleword

Subtract Halfword

Addition of magnitudes with unlike signs can never cause an overflow. The addition of magnitudes with like signs causes an overflow when the sign bit of the sum is different from the sign of the magnitudes being added. The buffered gate OVER (figure 3-83) detects an overflow condition in the adder by the following logic:

$$\begin{aligned} \text{OVER} &= \text{NFAS10 (D0 NK0 NPR0)} \\ &\quad + \text{NFAS10 (NA0 ND0 K0 NPR0)} \end{aligned}$$

Although PROBEOVER is true during Load Absolute Word and Load Absolute Halfword instructions, OVER cannot come true during these two instructions because the term NFAS10 inhibits the OVER gate.

The gate term (NA0 ND0 K0 NPR0) in the above equation applies to the addition of two positive numbers in the A- and the D-registers. If K0 is true, the sign of the sum of the two positive numbers is negative, causing OVER to be true. The gate term (D0 NK0 NPR0) in the above equation applies to the addition of two negative numbers in the A- and the D-registers.

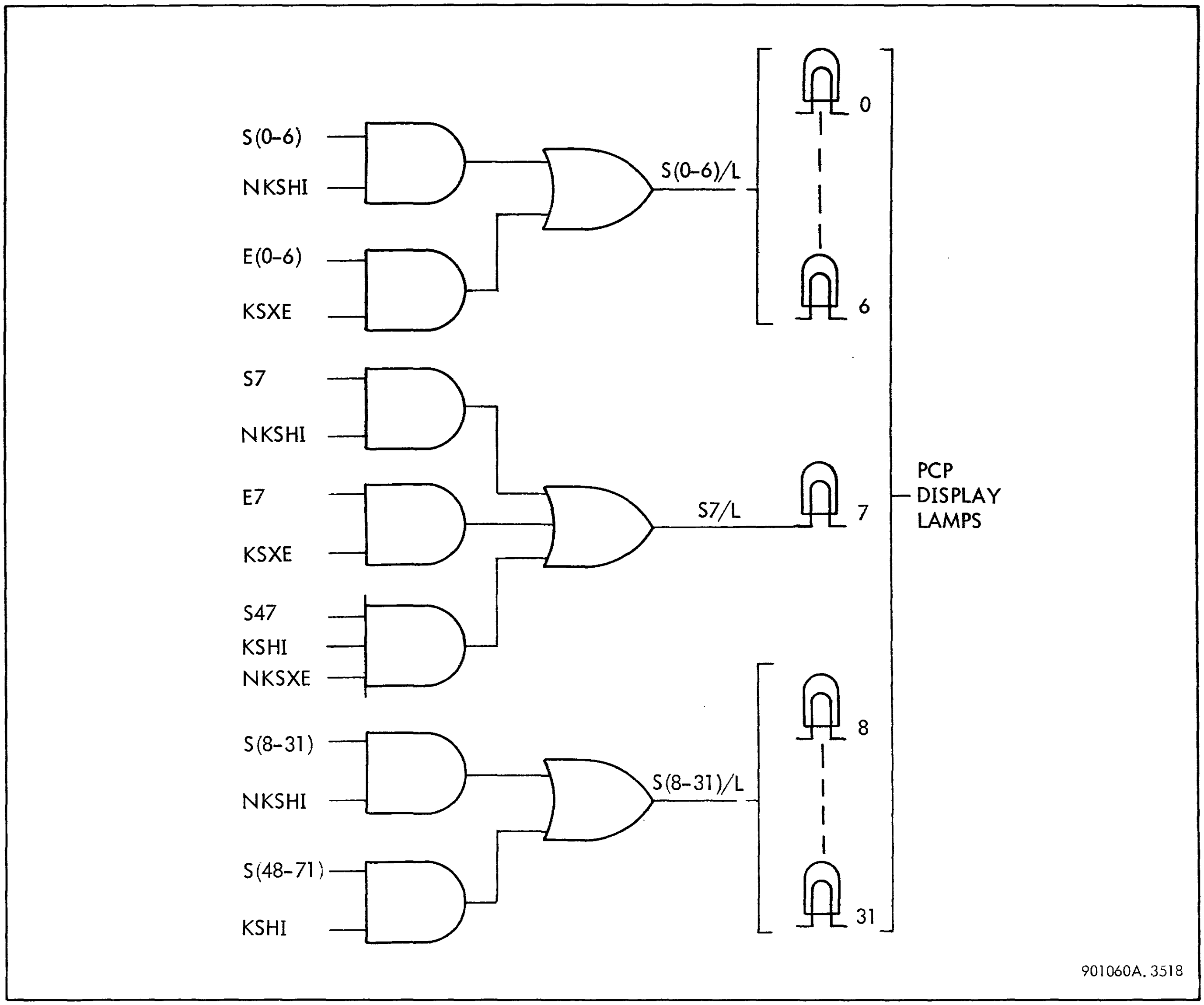


Figure 3-82. Display Lamp Gating, Logic Diagram

Table 3-9. Display Lamp Gating, Logic

REGISTER SELECT POSITION	CONTROL SIGNALS							REGISTER DISPLAYED
	KSHI	KSXE	SXA	SXB	SXCS	SXD	SXP	
A_L	0	0	1	0	0	0	0	A0-A31 on lamps 0-31
A_H	1	0	1	0	0	0	0	A47-A71 on lamps 7-31
B_L	0	0	0	1	0	0	0	B0-B31 on lamps 0-31
B_H	1	0	0	1	0	0	0	B47-B71 on lamps 7-31
CS_L	0	0	0	0	1	0	0	CS0-CS31 on lamps 0-31
CS_HE	1	1	0	0	1	0	0	CS47 not observed CS48-CS71 on lamps 8-71 E0-E7 on lamps 0-7
D_L	0	0	0	0	0	1	0	D0-D31 on lamps 0-31
D_H	1	0	0	0	0	1	0	D47-D71 on lamps 7-31
P	0	0	0	0	0	0	1	P15-P31 on lamps 15-31 P32 on lamp 0, P33 on lamp 1
S_L	0	0	0	0	0	0	0	S0-S31 on lamps 0-31
S_H	1	0	0	0	0	0	0	S47-S71 on lamps 7-31

Notes

KSHI = REGISTER DISPLAY switch ON. CLOCK MODE switch not in COUNT. REGISTER SELECT switch in A_H, B_H, CS_HE, D_H, or S_H

KSXE = KSHI. REGISTER SELECT switch in CSL

SXA, SXB, SXCS, SXD, SXP = Gating signals for transfer of applicable register to sum bus

Logic Equations

$S(0\text{-}6)_L$ = S(0-6) NKSHI + E(0-6) KSXE

$S7/_L$ = S7 NKSHI + E7 KSXE + S47 KSHI NKSXE

$S(8\text{-}31)/_L$ = S(8-31) NKSHI + S(47-71)

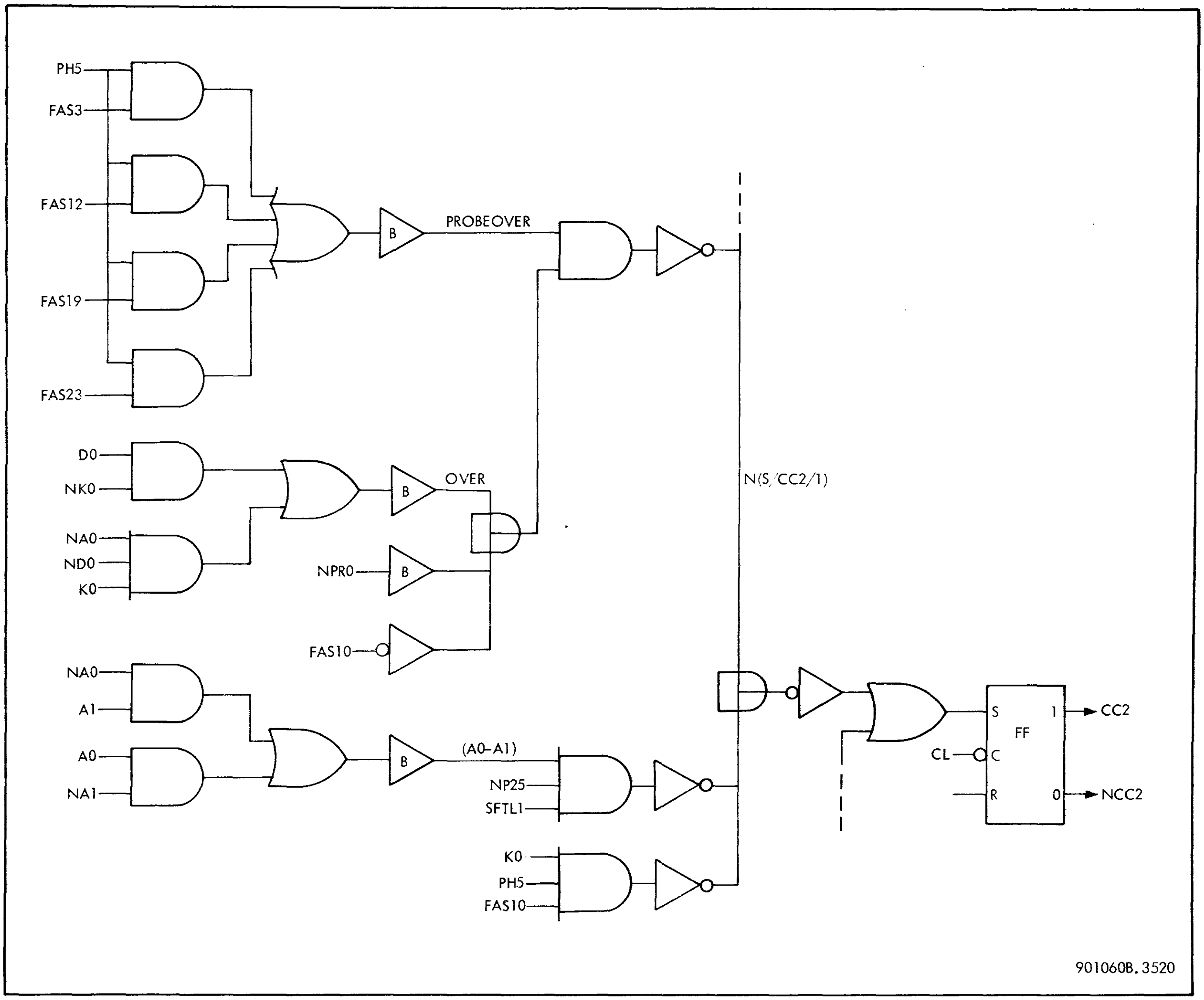


Figure 3-83. Overflow into Condition Code Bit CC2, Simplified Logic Diagram

If D0 is true and NPR0 is true, then either CS0 or A0, but not both, must contain a one. If K0 is not true, the sign of the sum of the two negative numbers being added is positive, causing OVER to be true.

In all instructions in which a fixed-point overflow can occur, condition code bit CC2 is set to a one if overflow occurs.

S/CC2 = OVER PROBEOVER + . . .

Condition code bit CC2 is set if an overflow should occur during a Load Absolute Word instruction (FAS10). This happens only if the effective word contained a one in bit position 0 and zeros in bit positions 1 through 31 (negative zero). Carry K0 can never be true (overflow can never occur) in a Load Absolute Halfword instruction.

S/CC2 = FAS10 PH5 K0 + . . .

An overflow occurs during a fixed-point left shift instruction when bit 0 and bit 1 of the A-register are not of the same binary magnitude.

S/CC2 = SFTL1 NP25 (NA0 A1 + A0 NA1)

Overflow can occur as the result of certain conditions during a divide instruction. Since overflow by divide is not strictly an adder function, it is treated separately in the Divide instructions (paragraph 3-201).

3-71 FIXED-POINT END CARRY. The buffered gate K00 checks for an end carry condition during all additions onto the sum bus. An end carry is meaningful only in those operations in which bit 0 of both the addend and the augend are considered to be magnitude bits rather than sign bits. End carry generation is independent and exclusive of overflow generation. For example, if the overflow has significance in a specific instruction, the end carry, if generated, has no significance. Conversely, if the end carry has significance in a specific instruction, the overflow, if generated, is ignored.

In many instructions, both overflow and end carry are recorded in CC2 and CC1, respectively, after the instruction has been completed.

In both Add Doubleword and Subtract Doubleword instructions, the end carry out of the low order sum is transferred to flip-flop K00H. See figure 3-84. Flip-flop K00H forces a one into CS31 for the addition of the high order sum, which, in effect, adds the end carry from the low order sum to the high order sum. In an Add Word or a Subtract Word instruction, an end carry is recorded in condition code bit CC1. This end carry has meaning only if the program is dealing with multiple precision fixed-point numbers.

The following five examples describe conditions in which combinations of overflow and end carry can occur.

Example 1. No overflow, no end carry

Sign Bit		
0	0 1 1 0 0 . . . 0	Augend
0	1 0 0 1 0 . . . 0	Addend
0	1 1 1 1 0 . . . 0	Sum

Example 2. Overflow, no end carry

Sign Bit		
0	1 0 0 0 0 . . . 0	Augend
0	1 0 0 1 0 . . . 0	Addend
1	0 0 0 1 0 . . . 0	Sum

Example 3. No overflow, end carry

Sign Bit		
1	0 1 0 0 0 . . . 0	Augend
1	1 1 1 0 0 . . . 0	Addend
1	0 0 1 0 0 . . . 0	Sum

Example 4. No overflow, end carry

Sign Bit		
0	1 0 1 0 0 . . . 0	Augend
1	1 0 0 0 0 . . . 0	Addend
0	0 0 1 0 0 . . . 0	Sum

Example 5. Overflow, end carry

Sign Bit		
1	0 1 1 0 0 . . . 0	Augend
1	1 0 0 0 0 . . . 0	Addend
0	1 1 1 0 0 . . . 0	Sum

The buffered gate K00 detects an end carry by the following logic (see figure 3-84):

K00 = A0 D0 + A0 CS0 ND0 + NA0 CS0 D0 GX
+ PR0 G1 + PR0 PR1 G2 + PR0 PR1 PR2 G3
+ PR0003 G0407 + PR0003 PR0407 G0811
+ PR0003 PR0407 PR0811 G1215
+ PR0015 K15

After an end carry has been generated, it is stored in K00H as a one or zero depending on the state of S00X:

S/K00H = S00
R/K00H =
S00 = K00 S00X + NK00 NS00X
N[FASHFL (PH14/1 + PH1 + PH11)]
S00X = FAS11 [S00XN + (A0 $\oplus$ D0)]
+ S00XN (A0 D0 + NA0 ND0)
S00XN = FAS1 PH2 + FAS22 PH5

NK00H sets a one into CS31, before the addition of the high-order half of the doubleword instructions.

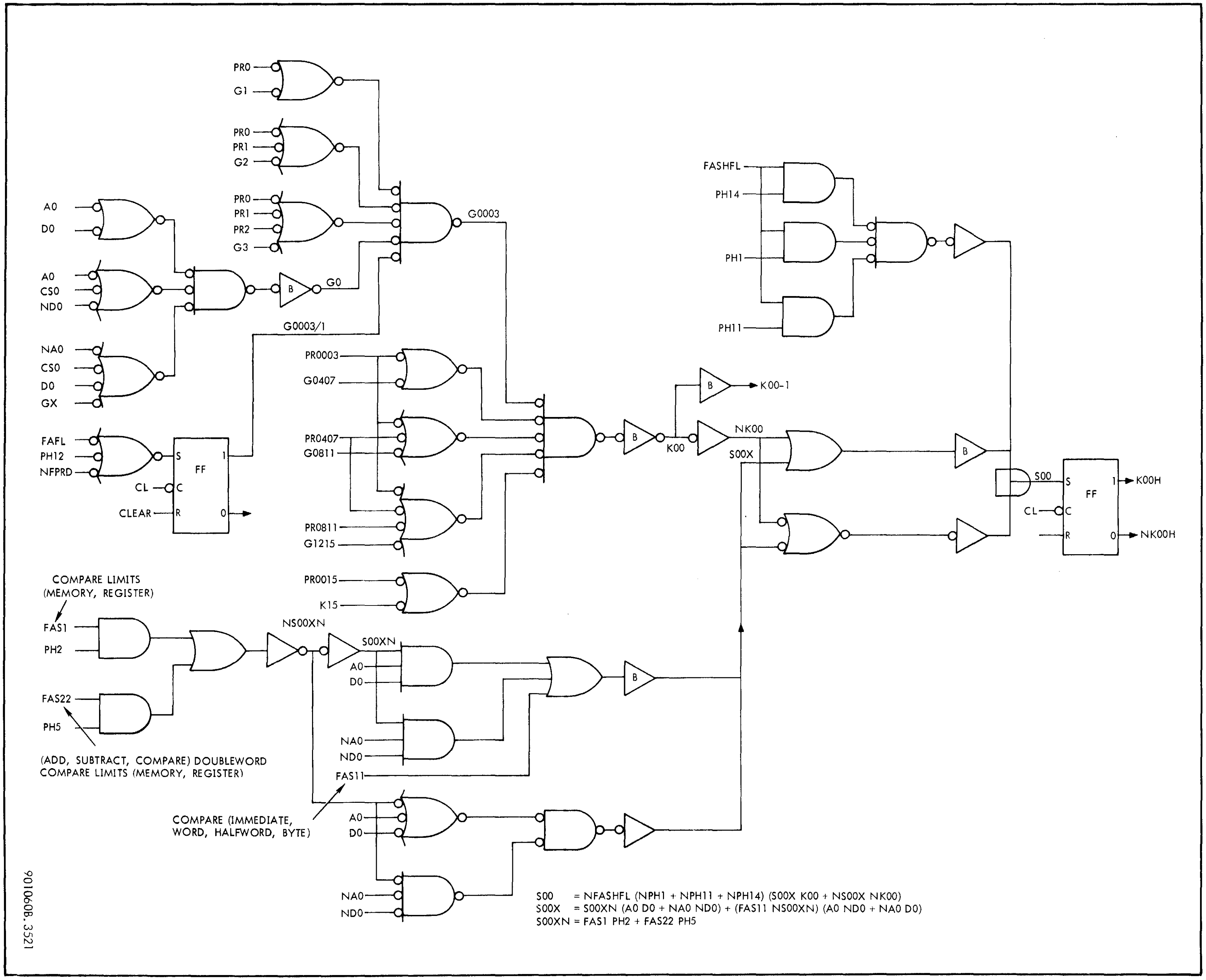
S/CS31 = (S/BWZ/1) NK00H NOLB
(S/BWZ/1) = (FAS16 + FAS22) PH3 NOL9

During Compare Doubleword, Compare with Limits in Memory, and Compare with Limits in Register instructions, K00H is used to generate an end carry from the low-order sum to the high-order sum. During all compare instructions K00H is also used to detect the equal, less-than and greater-than conditions of the words compared. Compare instructions are discussed in detail in paragraphs 3-209 to 3-216.

3-72 Basic Timing

3-73 CLOCK GENERATOR. The clock generator in the central processor consists of four delay lines with associated gates and amplifiers to tap off pulses at the desired time intervals. Three basic clock signals are produced:

a. A CL clock signal for trailing edge triggering of flip-flops throughout the CPU. This signal, when distributed, is referenced in the following discussion as CL clocks CL/1 through CL/12.



S00 = NFASHFL (NPH1 + NPH11 + NPH14) (S00X K00 + NS00X NK00)
S00X = S00XN (A0 D0 + NA0 ND0) + (FAS11 NS00XN) (A0 ND0 + NA0 D0)
S00XN = FAS1 PH2 + FAS22 PH5

Figure 3-84. K00H End Carry and Compare Flip-Flop, Logic Diagram

b. A private memory clock signal to gate information into the private memory registers. This signal, when distributed, is referenced as CK clocks CK/1 through CK/12.

c. A negative-going dc-hold signal to place data in the C-register buffer flip-flops, which are latching circuits and do not use an ac clock signal. This signal is referenced as the DCH clock signal.

All clock pulses are 40 ns wide.

Figure 3-85 is a simplified block diagram of the clock generator in which only the basic timing functions are shown. Gating, latching, pulse shaping, enabling, and other clock control functions are shown in detail later in this manual.

All clock signals except the first originate with a CL clock input to delay line 1. Taps are taken from this 300 ns delay line according to the time interval needed between one clock pulse and the next. These taps are controlled by flip-flops T1L, T4L, T8L, and T10L and signal T6L, and are selected according to the number of logical operations to be performed before another clock signal is needed.

A T1L clock request gives rise to an odd and an even CL clock signal every time a pulse travels down delay line 1 (except the first time when only an odd pulse is produced). Since the T1L delay line outputs are fed directly to the clock drivers, no DCH or clock signals are generated.

The T4L output from delay line 1 and the T6L, T8L, and T10L outputs, when not preceded by a T1L timing request, are fed into delay line 3, which times and shapes the three basic output clock signals. The positive-going CK clock signal is tapped off at the 20 ns point on delay line 3, and the CL clock signal is tapped off at 40 ns and is fed to delay line 4. A negative-going DCH signal appears at the 90 ns tap. The total delay times of these clock signals include a delay of approximately 40 ns at the inputs to the delay lines and in the clock-gating and amplifying circuits.

When a T6L, T8L, or T10L clock request follows a T1L clock signal, the time interval is longer than the normal T6L, T8L, and T10L clock intervals. A T1L post enable signal from a latch set by T1L disables the normal taps. When the even CL clock signal is generated from T1L, a pulse enters delay line 2. The T6L, T8L, and T10L taps are taken from delay line 2 and are gated into delay line 3. The CK, DCH, and CL clock signals are then tapped from delay line 3.

Timing intervals, longer than those from combining delay lines, can be obtained by recirculating the timing pulse through delay line 2 under the control of a two-bit trip counter which counts the number of times the pulse has recirculated.

Outputs, in addition to the three basic clock signals, are taken at timing points on delay lines 1 and 2 and are sent to various parts of the computer logic circuitry. Outputs from timing points on delay line 2 are gated by the trip counter to obtain specific extended time intervals.

All CL clock signals, except the T1L clock signals, are tapped from delay line 4. A clock enable signal is used to gate the pulses into delay line 4. Thus, the input to delay line 4 is disabled, when it is necessary to stop the clock pulse circulation temporarily because a memory access has not been completed. In this case, the DCH and CK clock pulses occur, and the CL clock pulse is held in a latch circuit at the input to delay line 4 until the clock enabling signal comes true.

Delay line 4 is used to generate the first clock signal at the start of computer operation, using a signal from the control panel as the first input to the delay line.

Logic diagrams of delay lines 1 through 4 are shown in figures 3-86 through 3-89. Figure 3-90 shows a timing diagram of the generation of the first two CL clock signals after the CPU RESET switch on the control panel is set. The timing is drawn on the assumption that the T6L clock request flip-flop is set, and the clock enable signal CE is true.

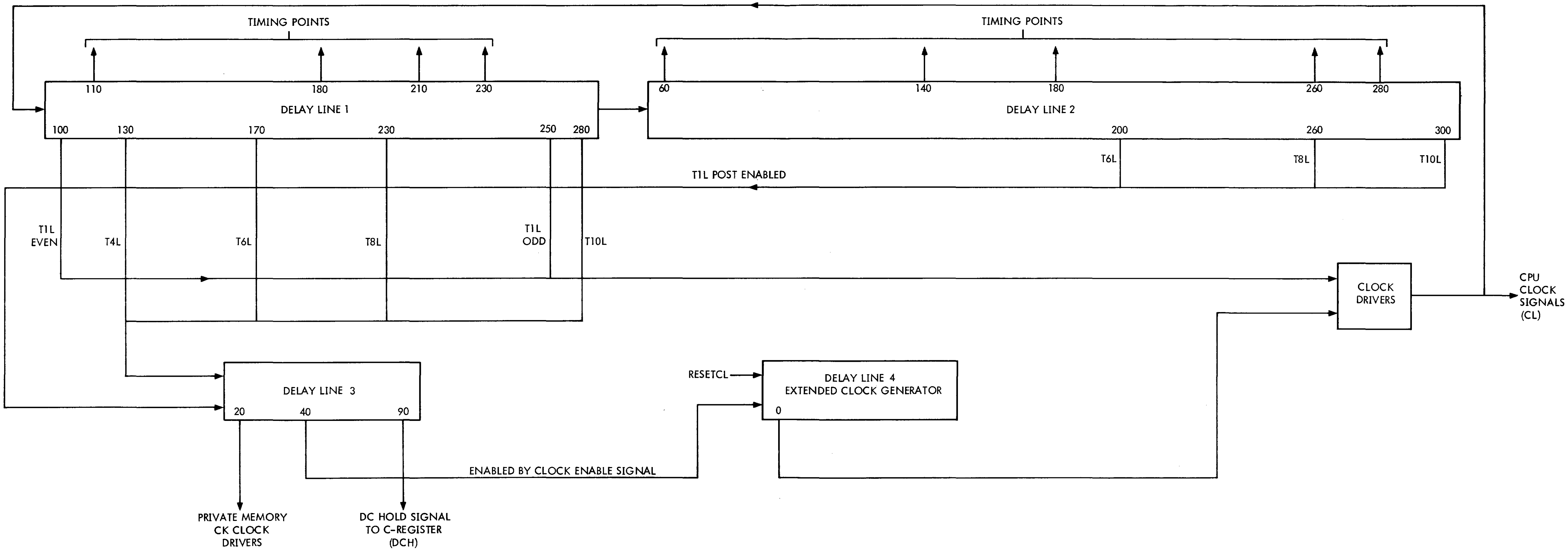
When the CPU RESET switch is set, reset clock signal RESETCL goes up and remains for an indeterminate period of time (figure 3-89). Signal CLEXTE goes true, and a 40 ns output, CLEXT, is generated from delay line 4. Clock driver outputs CL/1 through CL/12 are driven true as a result of CLEXT, and the first CL clock signal is distributed to the CPU. Signal RESETCL drives the T1L odd enable signal T1LOE and the T1L even enable signal T1LEE false and drives the DL1 enable signal DL1E true (see figures 3-91 and 3-92).

As a result of the delay line 1 input, delay line 1 pulse expansion signal DL1PE is brought up (see figure 3-90) and 40 ns later delay line 1 pulse cutoff signal DL1PC goes down. These two signals continue the input to delay line 1 and then terminate it after 40 ns which shapes the signal as it originally travels down the delay line. Delay line 1, delay 1, signal DL1/D1 is available at 30 ns, and the inverted delay 2 signal NDL1/D2 is true at 50 ns. Delay line 1 sensor enable signal DLSE1 follows DLSE1/S at 110 ns (figure 3-93), and this signal, with NT1L post enable signal NT1LPE and T6L, gate the output of a delay line sensor at 170 ns to provide the T6L clock generate signal CLT6LG.

The delay line 3 input, with its pulse expansion and pulse cutoff signals, is shown in figure 3-88. A low DL3PC signal releases the T1L post enable latch T1LPE if this latch is set, as shown in the following equation:

$$T1LPE = T1LPE\ DL3PC\ NRESETCL + T1L$$

At 20 ns, signal CKT2NL is taken from a delay line sensor to develop the private memory clock signal outputs CK/1 through CK/12. Forty nanoseconds down delay line 3, a



NOTE: DELAY LINE TAPS ARE IN NANOSECONDS

Figure 3-85. Clock Generator, Simplified Diagram

901060B.3400

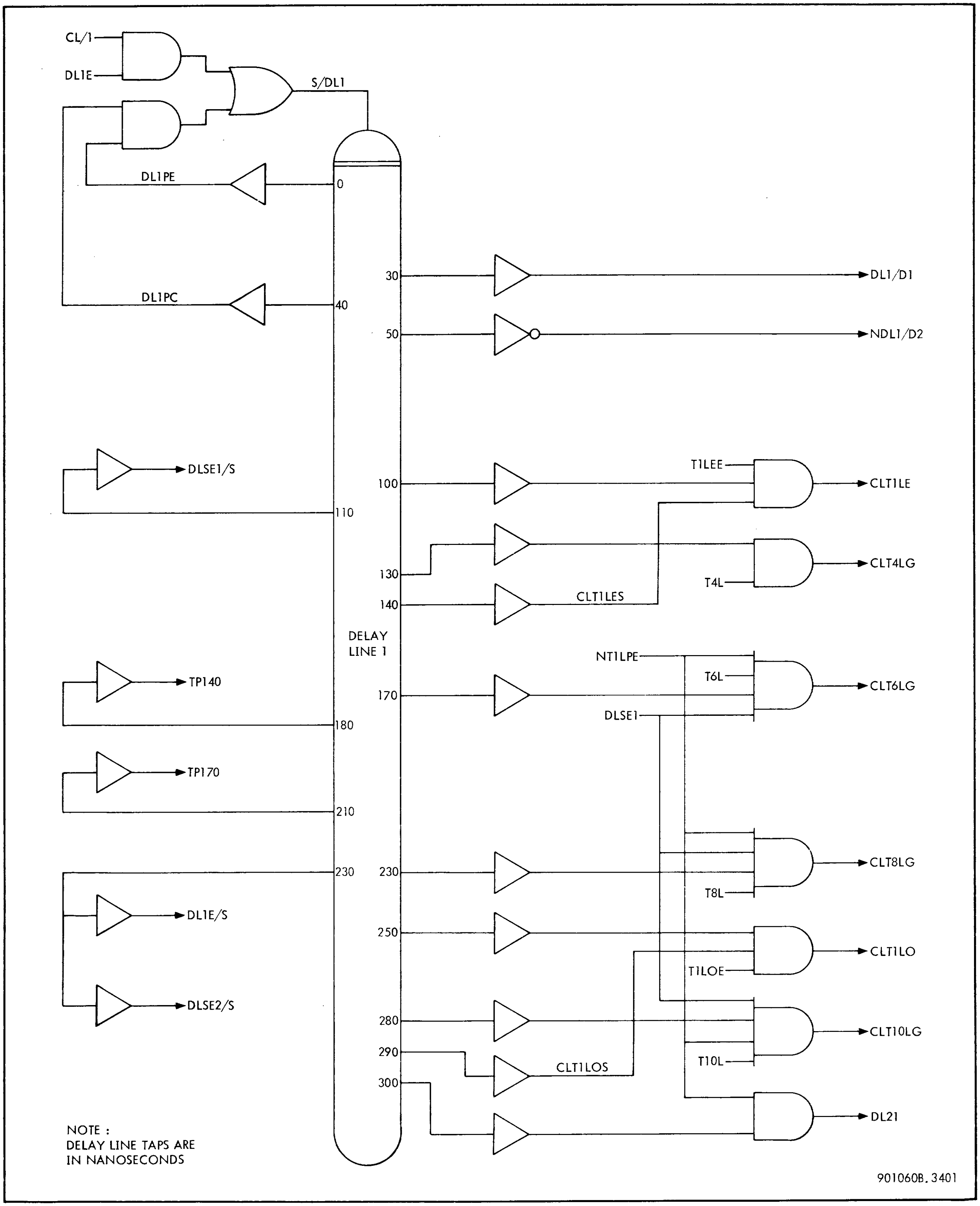


Figure 3-86. Clock Generator Delay Line 1, Logic Diagram

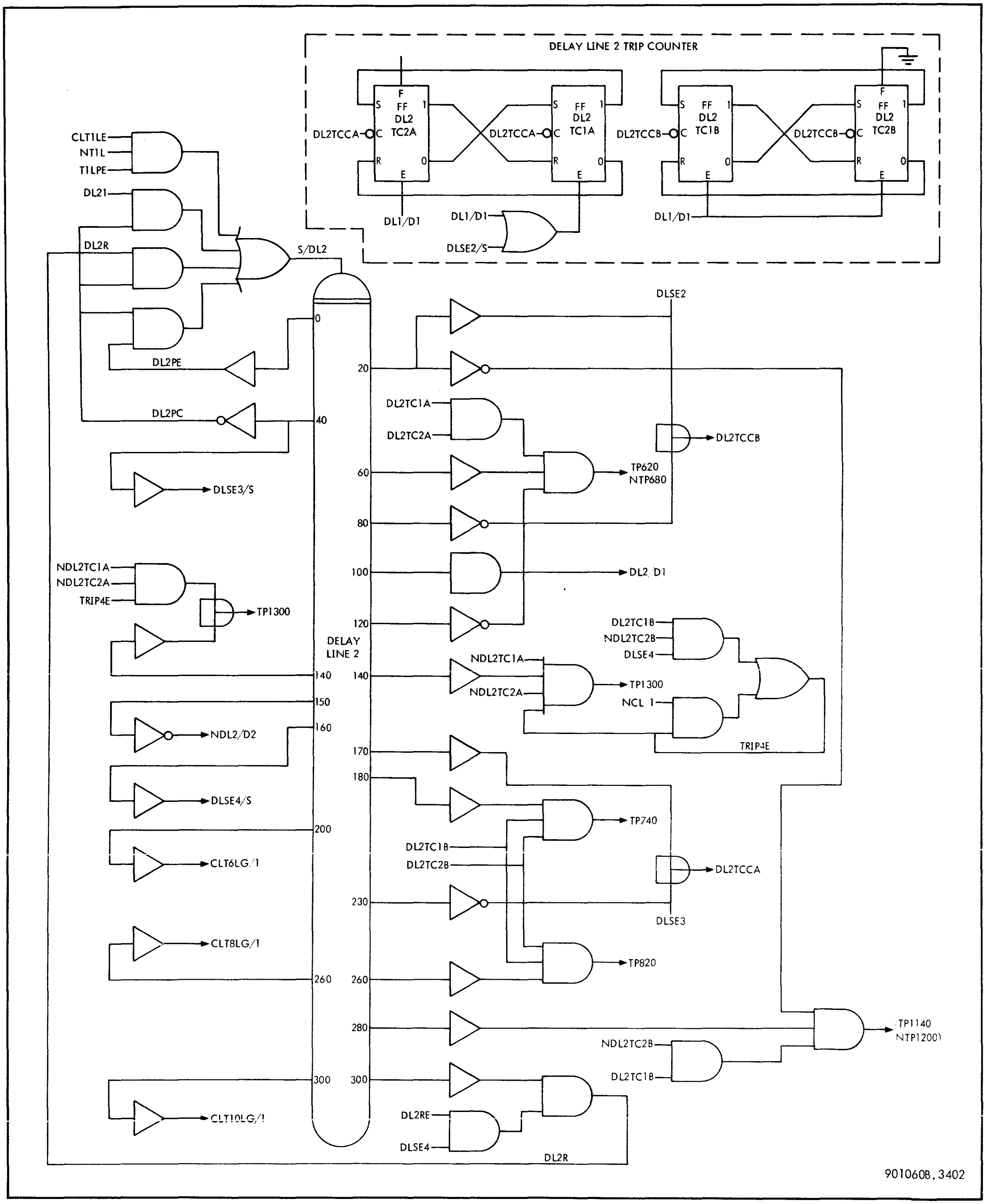


Figure 3-87. Clock Generator Delay Line 2, Logic Diagram

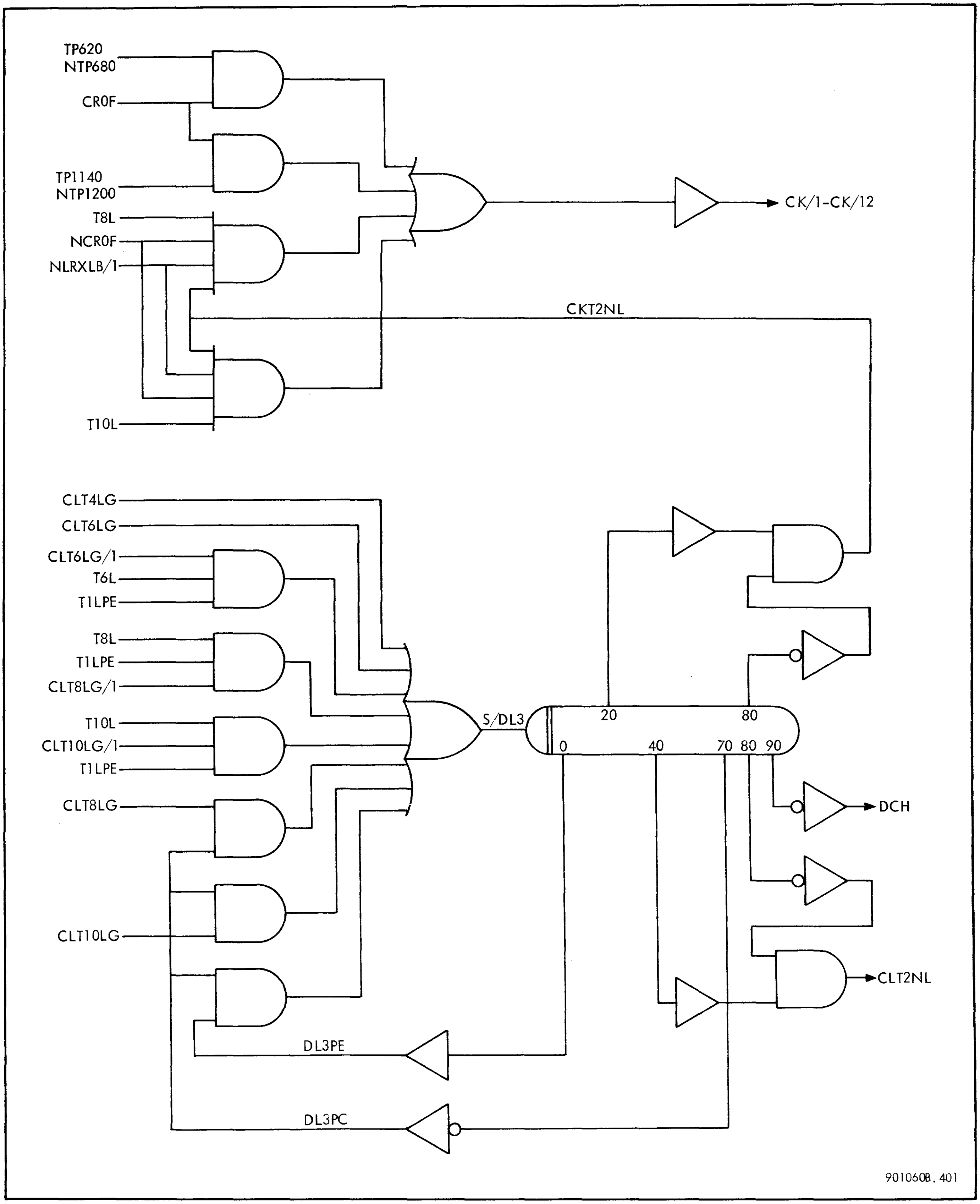


Figure 3-88. Clock Generator Delay Line 3, Logic Diagram

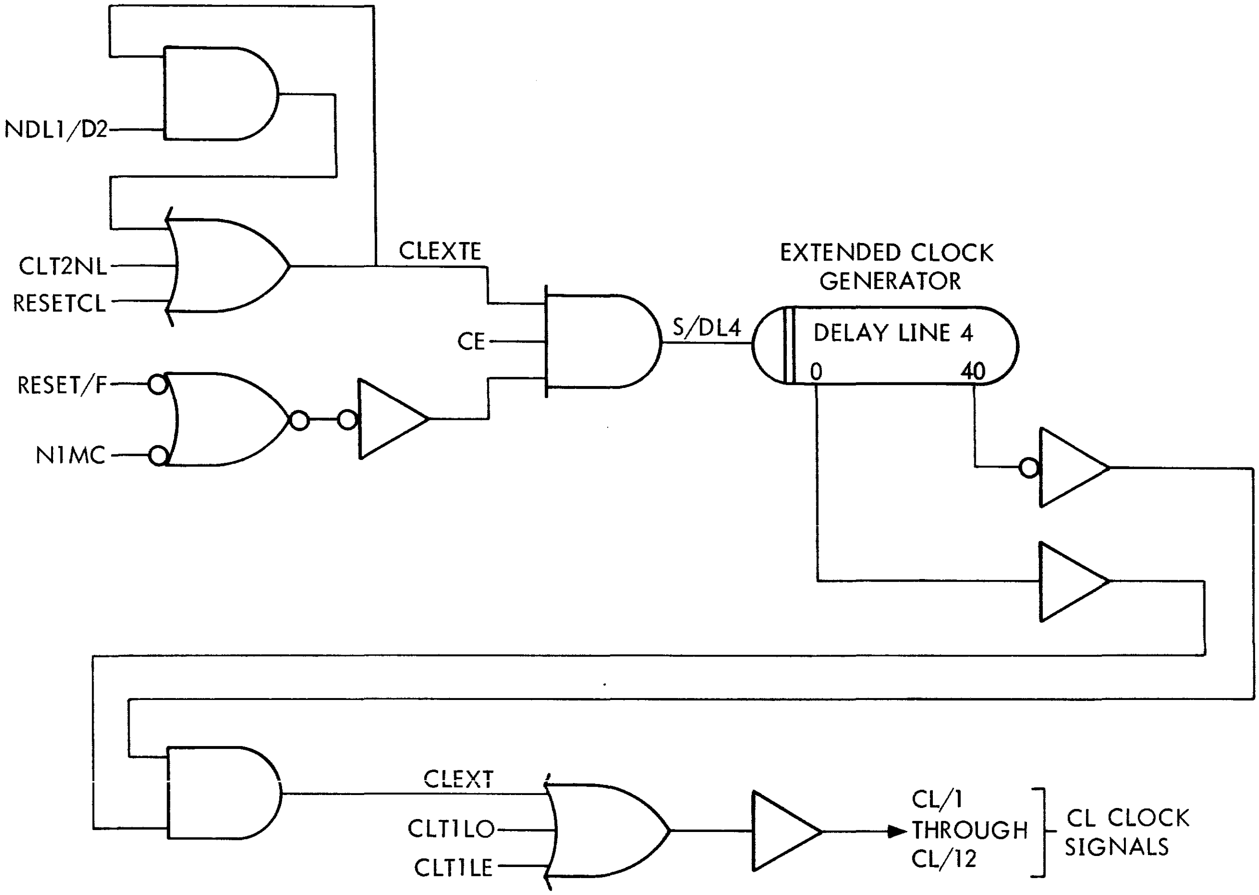


Figure 3-89. Clock Generator Delay Line 4, Logic Diagram

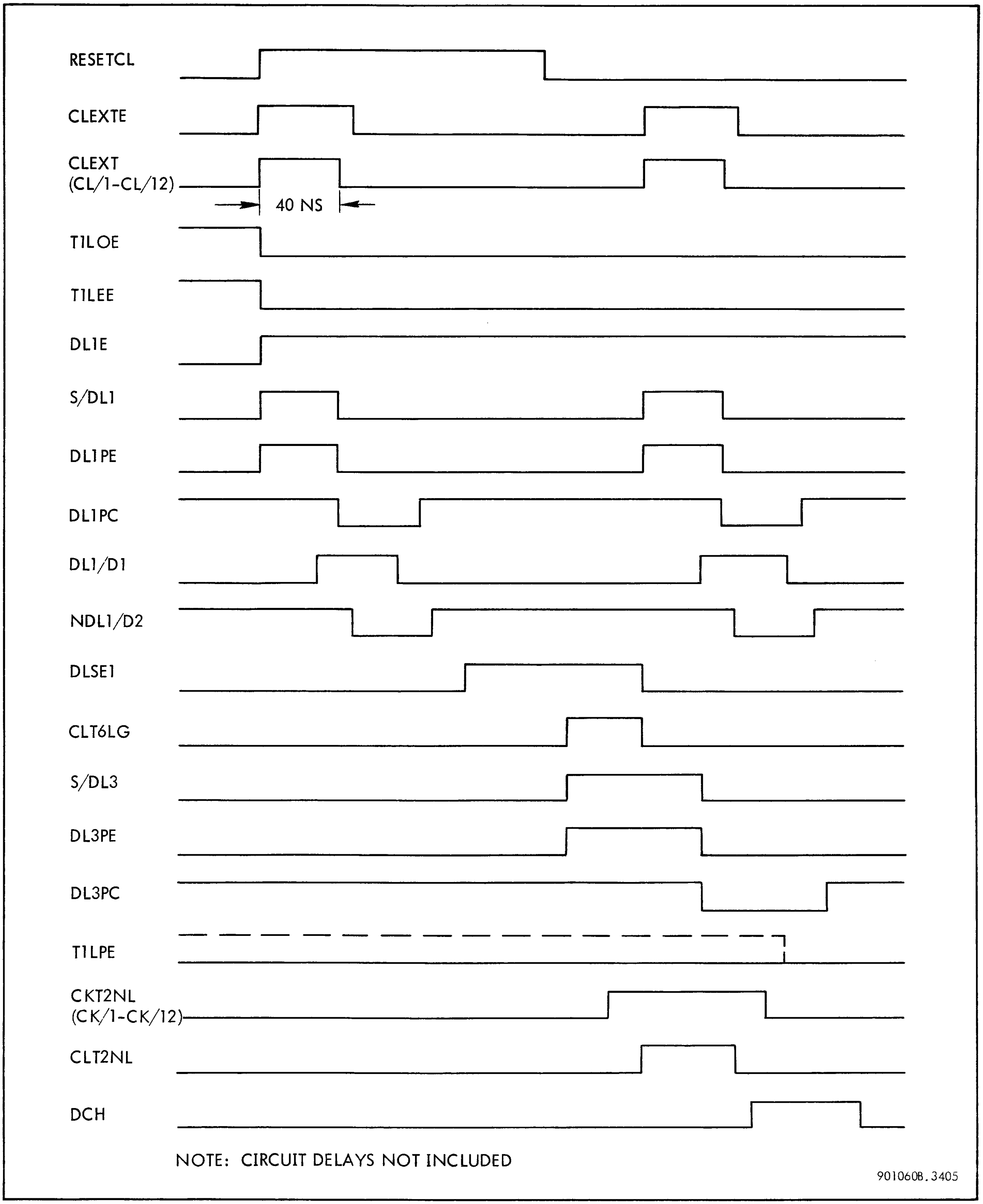


Figure 3-90. Original CL Clock Generation (TL1), Timing Diagram

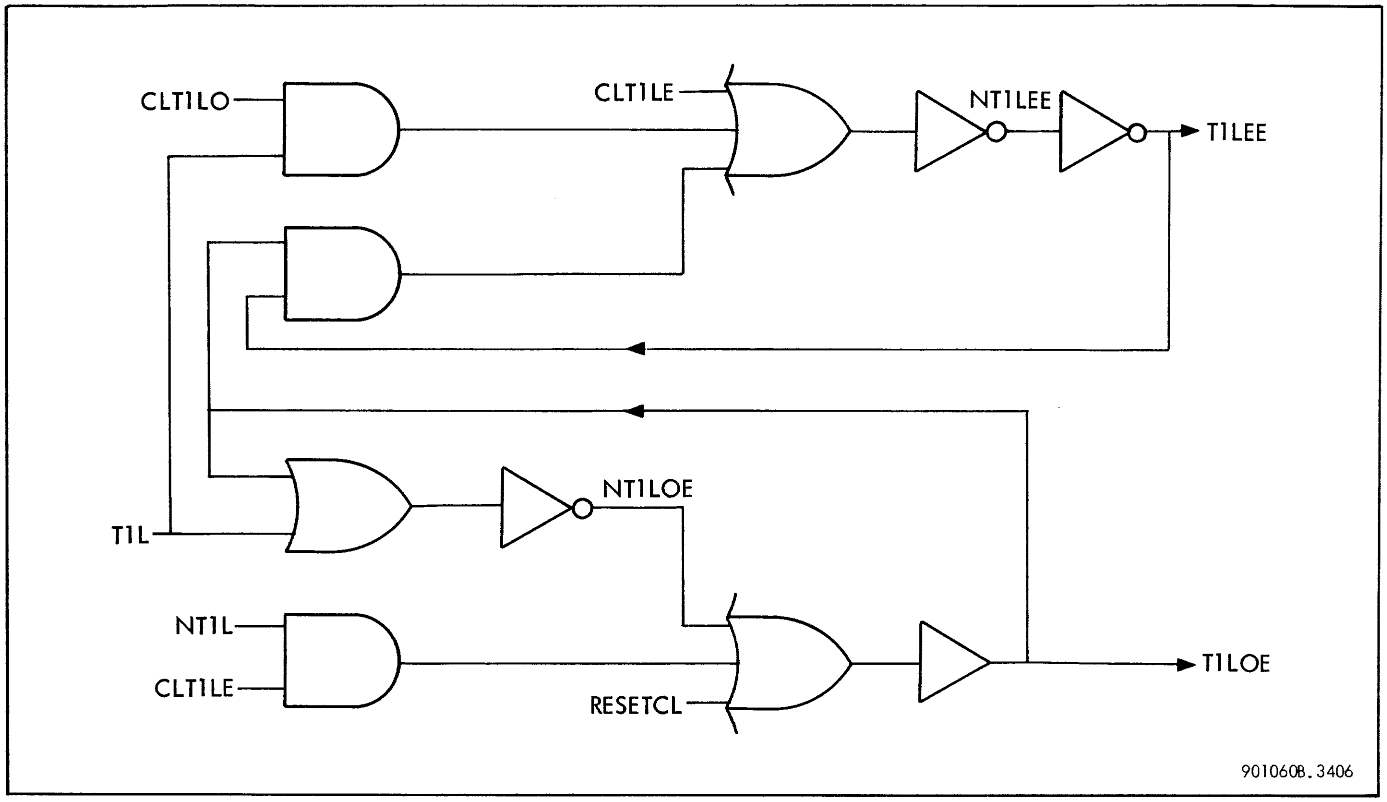


Figure 3-91. T1L Clock Logic

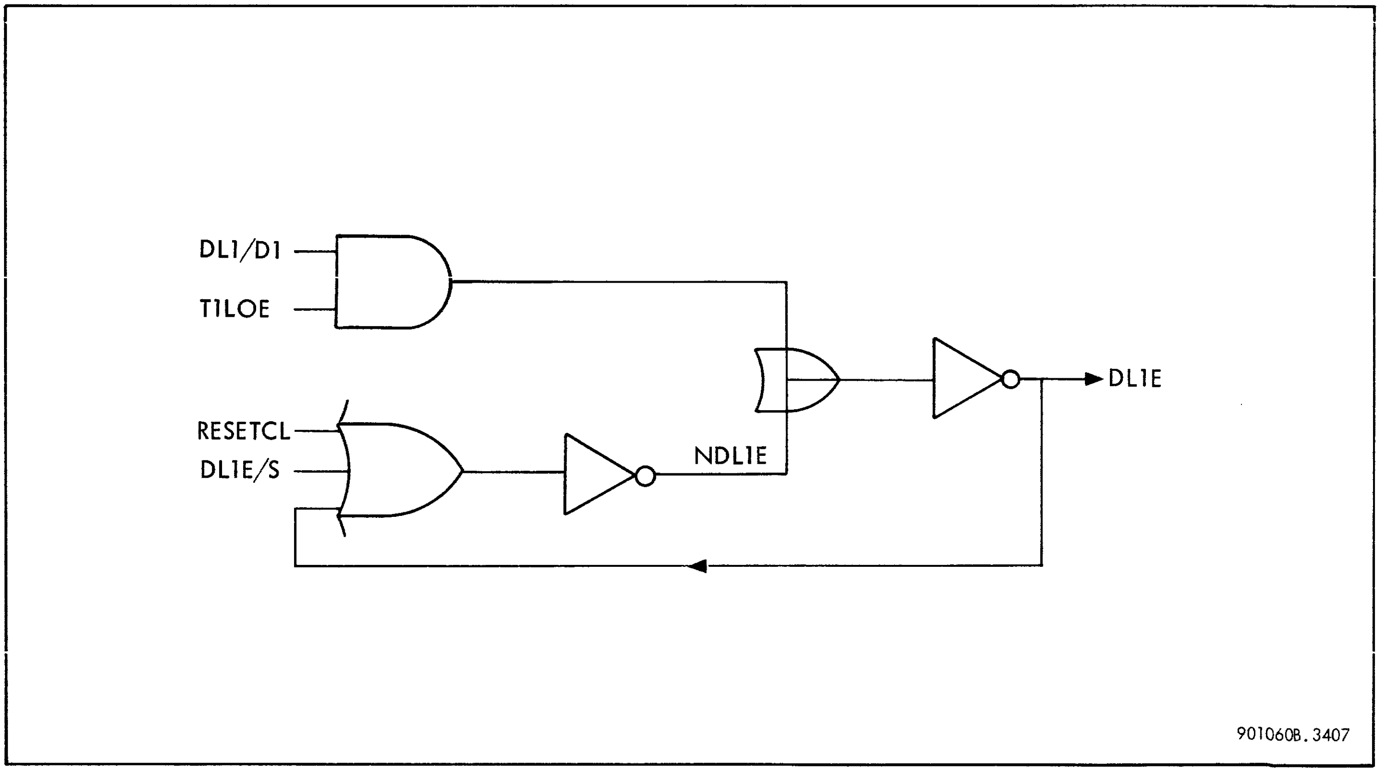


Figure 3-92. Delay Line 1 Lockout Circuit

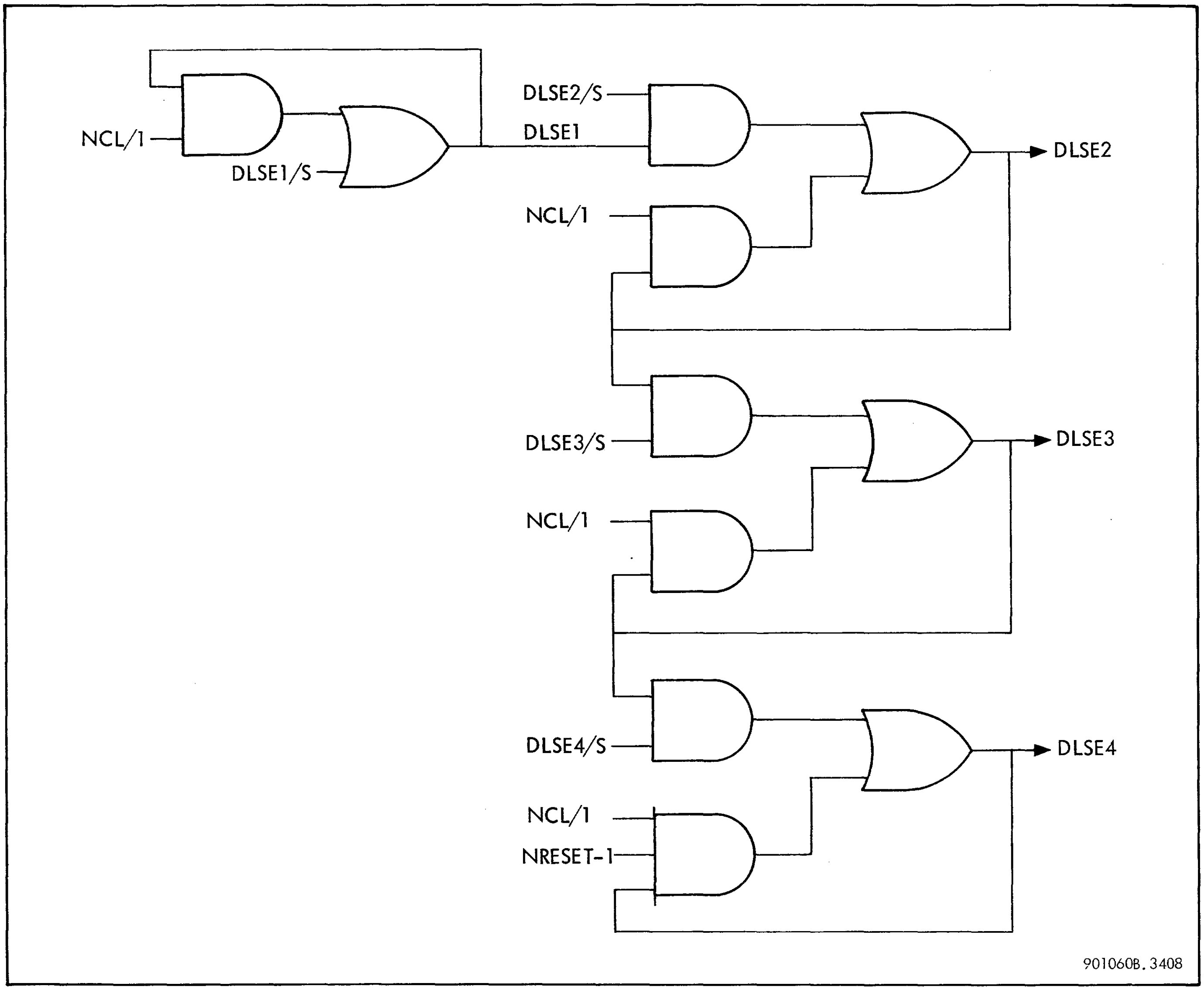


Figure 3-93. Delay Line Sensor Enabling Signal Generation

CLT2NL signal is sensed and applied to the input circuitry to delay line 4. A CLEXT signal at the output of delay line 4 produces the second set of CL clock signals. Signal DCH which appears 90 ns down delay line 3 is fed to the C-register latching circuits. A timing diagram of the CL, DCH, and CK clock signals as actually seen on an oscilloscope is shown in figure 3-94.

3-74 Clock-Enabling Function. The timing diagram in figure 3-95 shows how the CL clock signal is temporarily disabled when clock enabling signal CE goes false and then goes true again. The assumption is made that the T8L timing flip-flop has been turned on by a prior clock pulse and that timing flip-flop T4L is turned on by the T8L clock pulse. In this example, signal CE is dropped during the interval between the clock signals. The diagram shows that the CK and DCH clock signals occur as usual, but that the CL clock signals are delayed until CE rises again.

Signal CLEXTE is latched by its own feedback and by NDL1/D2 which remains true as long as no clock pulse goes down delay line 1. The input to delay line 4 is held false by the negative signal at the CE input to the S/DL4 AND gate. As soon as CE goes true, the AND gate is enabled, a pulse starts down delay line 4, CLEXT goes true, and a CL clock signal is produced.

3-75 T1L Clock Generation. The timing diagram in figure 3-96 shows the generation of two T1L clock signals every time that a pulse travels down delay line 1. In the diagram, the previous clock signal is generated from a T6L clock request. Delay line 1 enable signal DL1E has been latched on during the previous clock cycle (see figure 3-92); therefore, S/DL1 goes true and a pulse starts down delay line 1. On the falling edge of the T6L clock signal, flip-flop T6L is reset, and flip-flop T1L is set. The T1L signal drives the T1L odd enable signal T1LOE true. Since T1LEE is

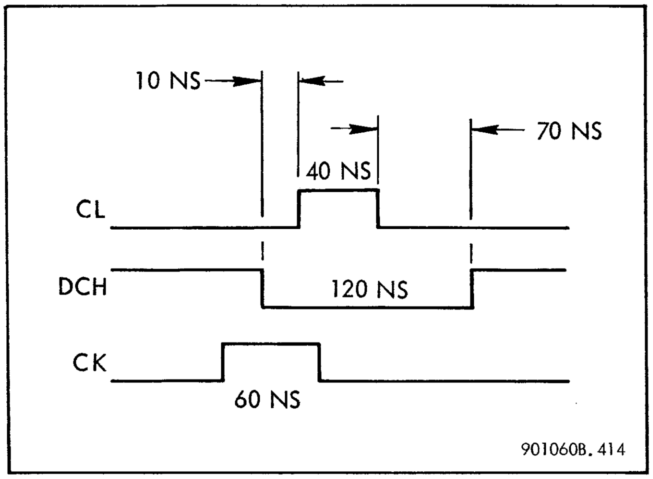


Figure 3-94. CL, DCH, and CK Clock Signals Actual Timing

false, an even T1L clock signal is not gated from the delay line during this first trip.

About 300 ns after the previous clock pulse (including circuit delays), a clock signal is gated from the 250 ns tap by T1LOE and DLSE1. The DLSE1 signal has been latched on since the pulse passed the 110 ns tap, driving DLSE1/S true (see figure 3-93). The odd clock signal CLT1LO drives the T1L even enable signal T1LEE true, which enables a gate that passes an even clock signal on the next trip of a pulse down the delay line.

The CLT1LO signal goes to the CL clock drivers through the OR gate shown in figure 3-89. The clock pulse is applied to the input of delay line 1 through another gating circuit. The total delay time of the circuits external to the delay line is about 50 ns. An even clock pulse is tapped at 100 ns, giving a total time interval of about 150 ns between the odd clock signal and the following even

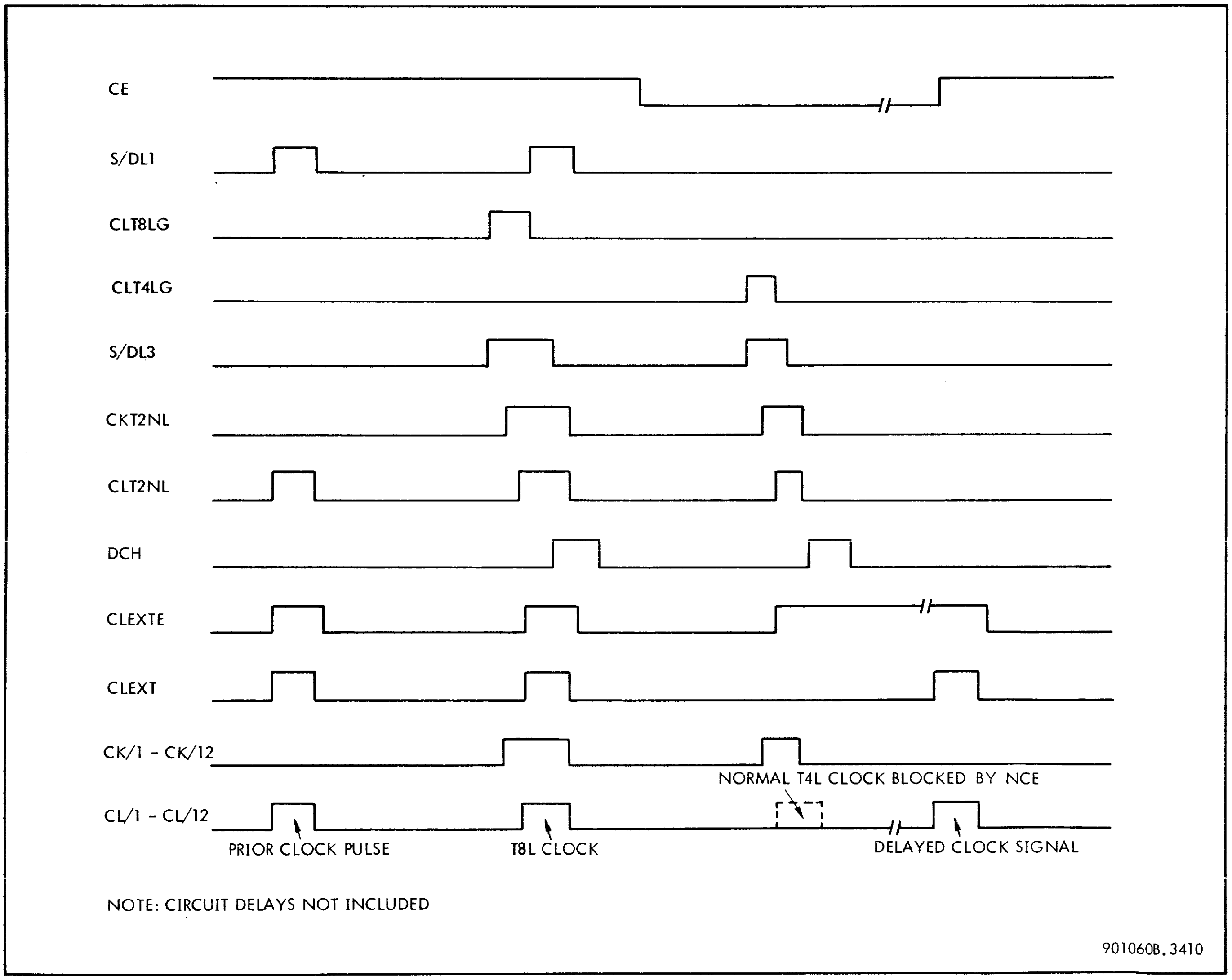


Figure 3-95. Clock Enabling Function, Timing Diagram

clock signal. This even clock pulse is locked out of the delay line 1 input because DL1E has been driven false by DL1/D1 and T1LOE, as shown in figure 3-92.

There are normally at least 16 clock pulses after T1L is set. Figure 3-96 is an abbreviated example that shows T1L reset at the fall of the first even clock pulse. Two more T1L clock signals occur after the fall of T1L which is explained in the following paragraph.

Down the delay line and 150 ns from the even tap, an odd clock pulse is sensed. Signal DL1E has risen by this time, and the pulse is allowed to reenter delay line 1. When the second even pulse is gated, CLT1LE and NT1L drive T1LOE false (see figure 3-91). Further odd pulses are now disabled. When CLT1LE drops, T1LEE can no longer remain true, and the even pulses are terminated. The last T1L clock signal is always an even one.

A T1L post enable signal, T1LPE, is set by T1L to allow the next clock pulse following the last T1L clock to be tapped from delay line 2 (see figure 3-87).

3-76 T1L Post Enabling Function. A T1L clock request is always followed by a T6L, T8L, or T10L clock request. These clock signals, when preceded by a T1L request, are taken from delay line 2 instead of from delay line 1 because the last T1L clock signal always comes from an earlier portion of delay line 1. If the next T6L clock signal were taken from the normal point on delay line 1, it would not be a true T6L time interval. Therefore, the signal must be propagated from delay line 1 into delay line 2 for an additional period of time to get a normal T6L delay. A T1L post enable signal, T1LPE, is driven true by T1L and is latched until the DL3 pulse cutoff signal, DL3PC, from delay line 3 at the end of the following clock cycle goes false (see figure 3-97).

The timing diagram in figure 3-97 shows a T6L clock signal generated from delay line 2 with T1LPE true and from delay line 1 with T1LPE false.

The last even T1L pulse is gated into delay line 2 as shown in figure 3-87:

S/DL2 = CLT1LE NT1L T1LPE + . . .

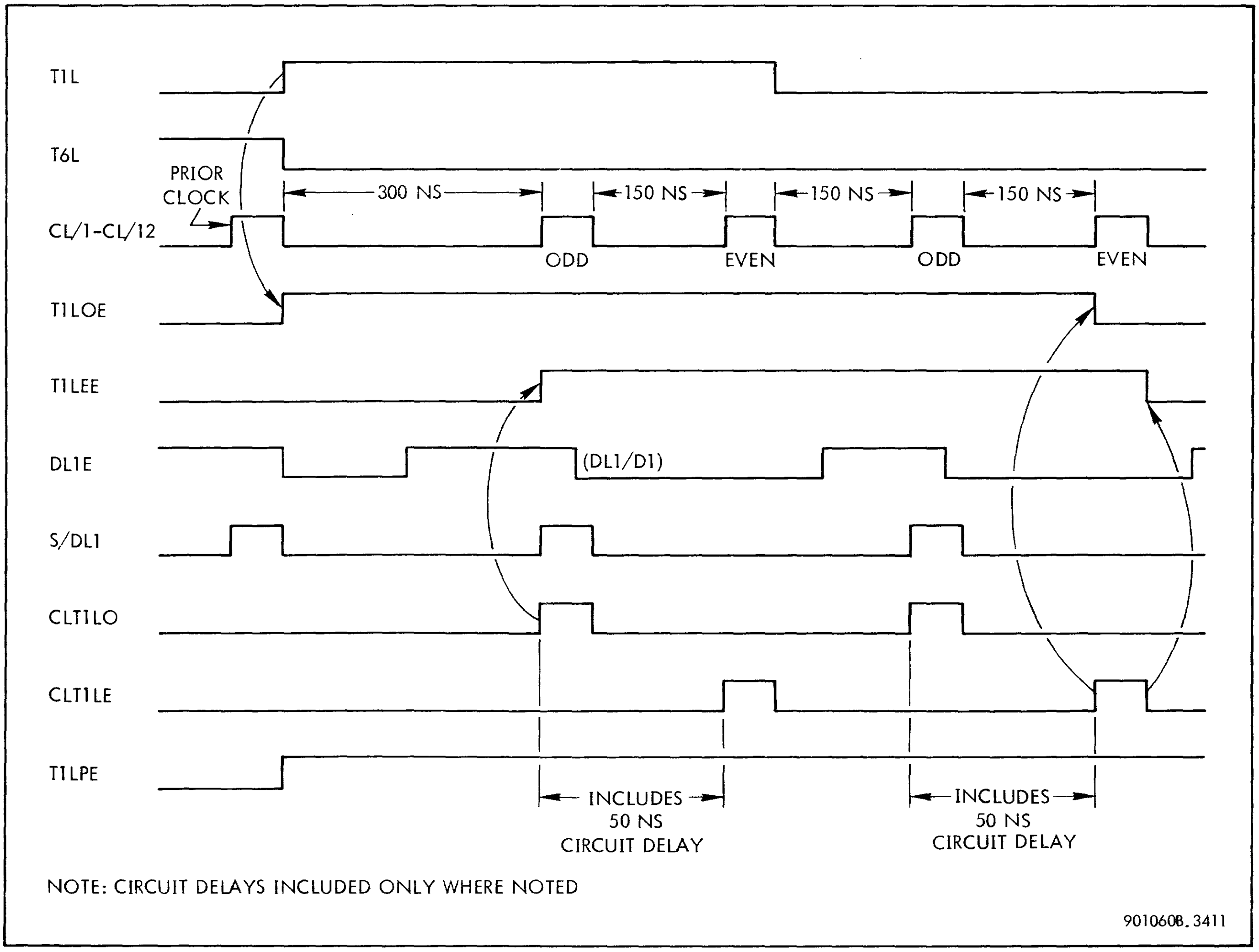


Figure 3-96. T1L Clock Generation, Timing Diagram

At the 200 ns tap, CLT6LG/1 is sensed and is gated into delay line 3 as shown in figure 3-88:

S/DL3 = CLT6LG/1 T6L T1LPE + . . .

The CL, CK, and DCH clock signals are then tapped from delay line 3 as in normal clock generation.

Signal T1LPE drops at the end of the first T6L cycle (figure 3-97). The second T6L pulse is gated from delay line 1 by NT1LPE. The T6L tap, CLT6LG, is applied to the delay line 3 input gates, and the three basic clock signals are obtained as usual.

3-77 Trip Counter. Time signals are required from delay line 2 for length intervals beyond the last clock. For example, if clock enable signal CE goes false, it is necessary to have some means of determining whether a signal such as an address from memory has been received within a certain interval of time. If the signal is not received, appropriate action must be taken to generate a clock, to note that an error has occurred, and to continue from this point with the error condition. For this reason, it may be necessary to get timing pulses at intervals greater than those provided by delay lines 1 or 2. This is done by repeatedly circulating a pulse through delay line 2. A trip counter is used

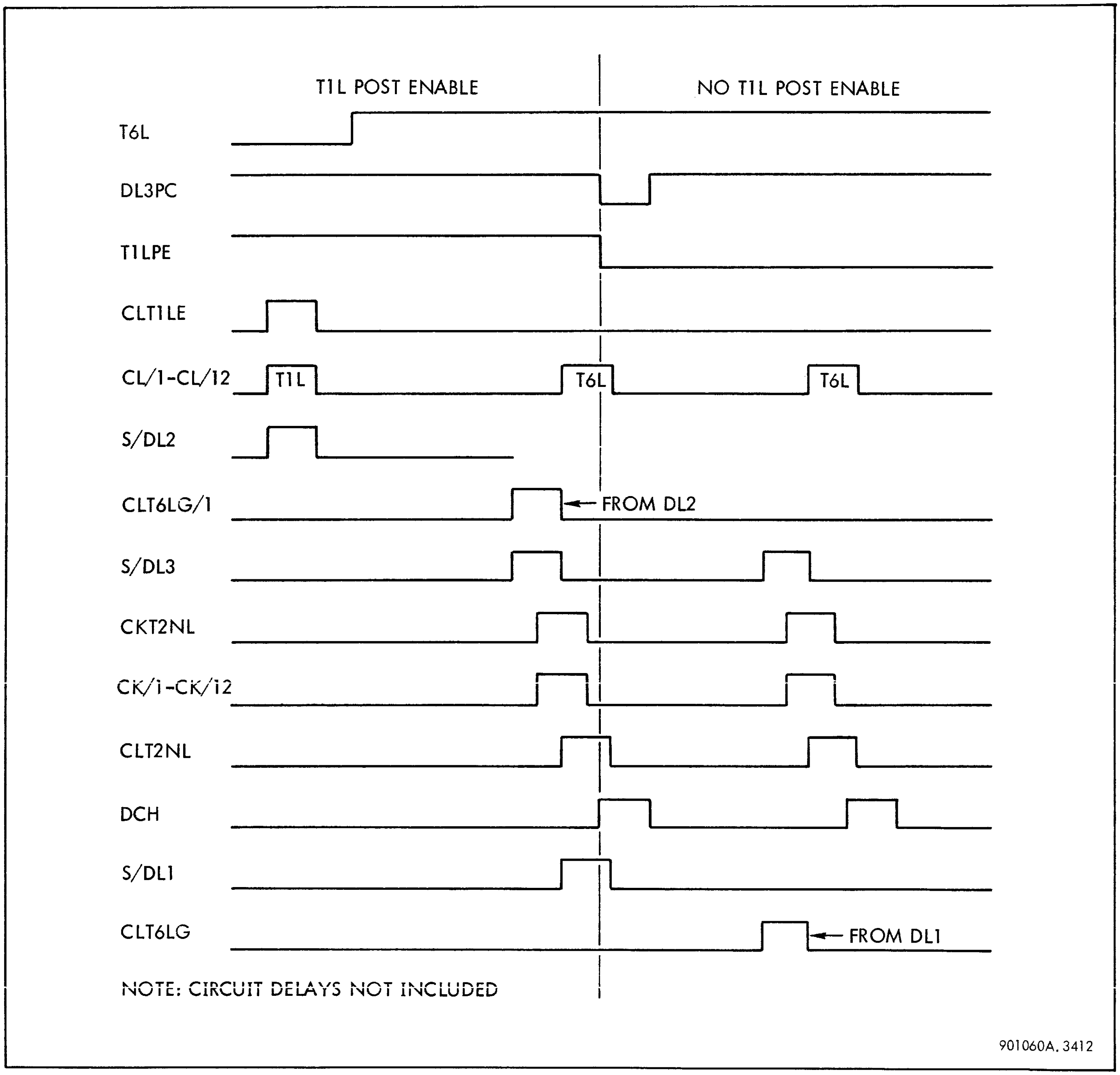


Figure 3-97. T1L Post Enabling Timing Diagram

to count the number of trips that the pulse makes through delay line 2.

The trip counter, shown in figure 3-87, consists of four flip-flops, DL2TC1A, DL2TC1B, DL2TC2A, and DL2TC2B. Each of the two flip-flop combinations in the trip counter is placed in the 0-0 state when signal DL1/D1 is tapped from delay line 1. In the absence of a clock signal, the count progresses as the clock pulse circulates through the delay line as shown in table 3-10.

Table 3-10. Trip Counter Progression

DL2TC1A DL2TC1B	DL2TC2A DL2TC2B	State
0	0	0
0	1	1
1	1	3
1	0	2
0	0	0

The use of the trip counter is shown in figure 3-98. The top waveforms show the input to delay line 1, followed by delay line sensor enable 1 (DLSE1) at the 110 ns tap and delay line sensor enable 2 (DLSE2) at 230 ns. At 300 ns, the pulse enters delay line 2, and delay line sensor enable 3 (DLSE3) rises at the 40 ns tap on delay line 2. The logic of these sensor enabling signals is shown in figure 3-93.

The delay line 2 recirculation enabling signal, DL2RE, rises and is latched when the delay line 2 input occurs (see figure 3-99). Each time the pulse reaches a tap labelled DL2TCCA or DL2TCCB (delay line 2 trip counter clocks), the two halves of the trip counter are advanced to their following states. The timing point signals shown in figure 3-98 are gated by the trip counter outputs. The CL clock signal, which is disabled by a false CE signal during the delay line 2 recirculation, causes the sensor enabling signals to go false.

3-78 Timing Request Flip-Flops. The T1L, T4L, T8L, and T10L clock request flip-flops are set by instruction and phase logic so that the time interval between one clock signal and the next is adequate to allow the necessary operations to be performed. Clock request signal T6L, which is not a flip-flop, is true when no other timing request is set:

T6L = NT1L NT4L NT8L NT10L

The time intervals obtained by the clock request are shown in table 3-11.

Table 3-11. Clock Time Intervals

Clock Request	Time Interval (ns)
T1L	150
T4L	280
T6L	320
T8L	380
T10L	440

The time required when private memory is addressed is dependent, in part, on the location of the private memory register bank being addressed. Because of the physical distance the signals must travel, more time is required to address private memory extension register banks in frame 2 of CPU cabinet No. 2 than to address one of the first four pages of private memory registers in CPU cabinet No. 1. Signals T4RL and T6RL are used when the private memory extension registers might be addressed. Signal T4RL requests a T4L clock signal if the CPU private memory is addressed and a T10L clock signal if the register extension is addressed. Signal T6RL requests a T6L clock signal for the CPU registers and a T10L clock signal for the register extension.

In either case, the T10L request flip-flop is set if either RP23, RP24, or RP25 in the register pointer register contains a one, which indicates a register designation higher than four. If RP23 through RP25 contains zeros, only RP26 and RP27 select the private memory register bank one of banks 0 through 3 in the CPU. The T4RL and T6RL signals are gated with the outputs of flip-flops RP23 through RP25 to select a T10L clock signal. A T4L clock is selected when these flip-flops contain zeros.

3-79 CLOCK ENABLING LOGIC CIRCUITS. As explained in paragraph 3-74, a CLEXT signal for clock generation is obtained from delay line 4 only if clock enable signal CE is true. When it is necessary to delay the clock, CE is held false until a CL clock signal is desired.

A logic diagram of the clock enabling signal is shown in figure 3-100. To illustrate the various functions of the clock-enabling gates, figure 3-101 shows the equation for CE divided into sections. The function of each section is described separately.

If all other gates are true, gate 1 allows a CL clock to be generated at the proper time in a memory access. Any one of the terms listed in figure 3-101 produces a clock. Beginning at the top, AHCL (address here clock) is a timing signal that goes true when the memory acknowledges an address. Signal AR comes from an address release signal from memory. Signal NARQ is the reverse of signal ARQ which indicates that after a memory request, an address release signal must be received from memory before another clock signal is generated. Signal NRIP means that no memory request is in progress, and signal MAA indicates that no memory request is pending. Signals NMAA and NMR indicate that a memory request is pending, but that a memory request signal has not been sent to the memory.

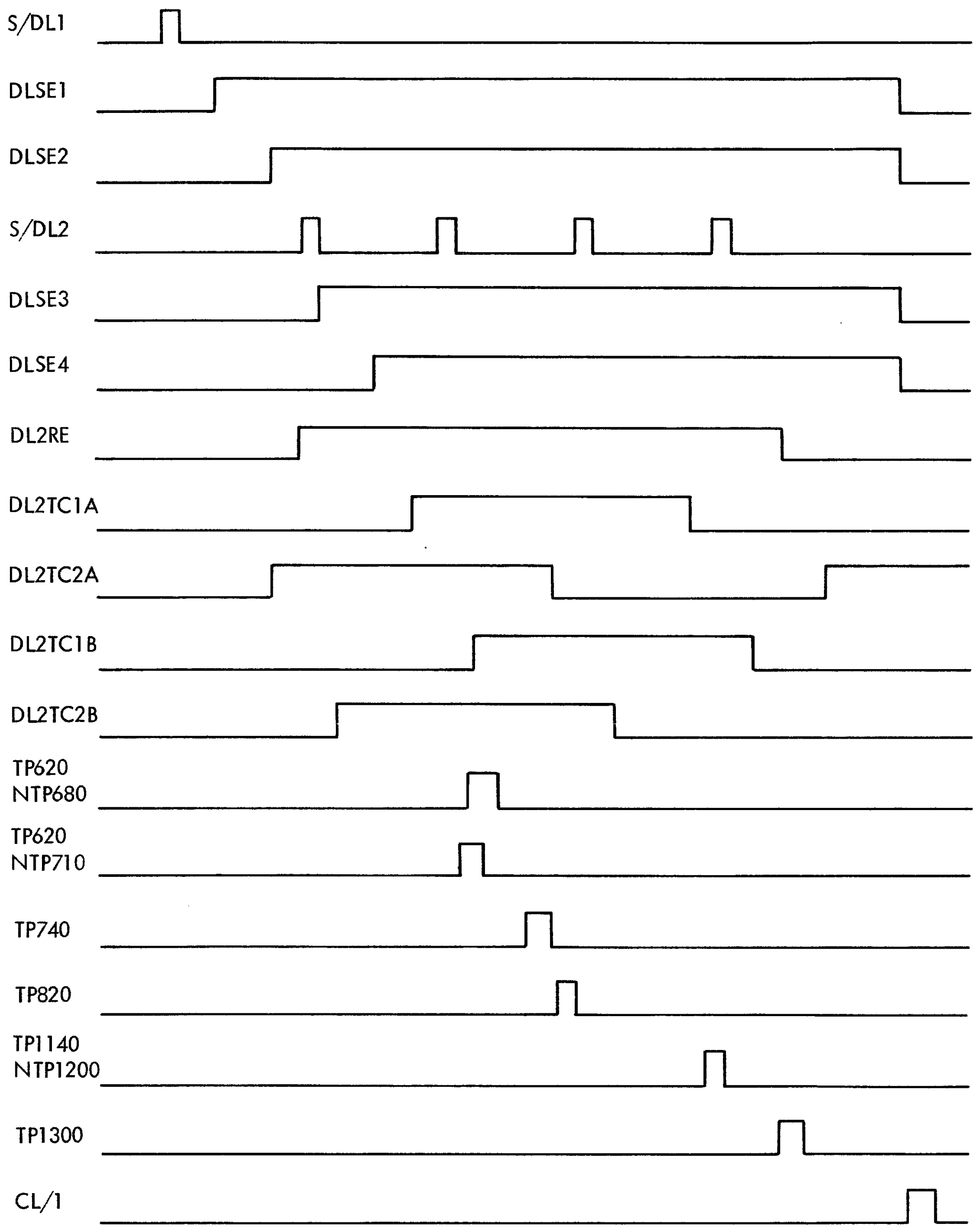


Figure 3-98. Trip Counter Functions, Timing Diagram

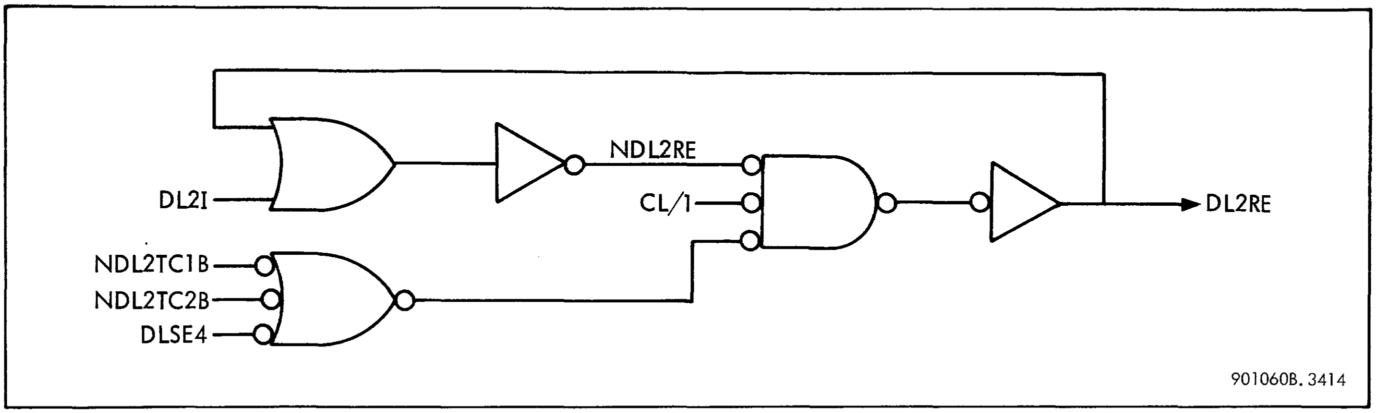


Figure 3-99. Delay Line 2 Recirculation Enabling Signal Generation

Gate 2 of the clock enabling signal coordinates the CL clock with the data request and data release signals between memory and the CPU. Signal NDRQ is the reverse of signal DRQ which indicates that if a memory request is out, the clock must be delayed until a data release signal has been received from memory. Signals MAA and NRIP indicate that no memory request is in progress and no memory request is pending. Signal MFF means that one memory request has been sent to memory, and an early data release signal has come back. Signal DRL is the output of a flip-flop set when data release signal DR is received from memory and is latched if a second memory request is not pending.

S/DRL = DR + (DRL NMGG NMEE)

If all the other gates are true, any one of the signals on gate 2 drives CE high.

Gate 3 provides a means of disabling signal CE if crossover is taking place, if the watchdog timer has been activated, or if a memory parity error signal has been received when the PARITY ERROR MODE switch is in the HALT position.

Gate 4 is used to synchronize the CL clock signal with the 2.048 MHz clock signal used in the interrupt circuits. This gate is also used to synchronize the CL clock signal with the 1.024 MHz clock signal when the CLOCK MODE switch is in the SINGLE STEP position. Gates 5 and 6 are used in the single clock mode.

3-80 Single Clock Generation. When the CLOCK MODE switch is moved from the CONT to the center position, KSC goes true and signal CE is disabled by NKSC and SC2 at gate 5 (figure 3-101). Setting the switch to the SINGLE STEP position drives KC true, and KSC remains true. On the trailing edge of the following 1.024 MHz clock signal, flip-flop SC1 is set. Flip-flop SC2 sets on the next 1.024 MHz clock signal, and the output of gate 5 goes true, generating a CL clock signal. Assuming that the AND gate inputs to SCEN are all false, the clock signal drives SCD true, and this signal is latched by feedback as long as the switch is kept in the SINGLE STEP position. While the latch is set, CE is disabled by gate 6, and no CL clock signals may be generated. Releasing the switch resets SC1 and SC2, and CE is disabled by gate 5 until the switch is set in the SINGLE STEP position again. Releasing the switch also drops the latch holding SCD true. A timing diagram of this operation is shown in figure 3-102.

Gate 4 and the SCEN signal on gate 6 are used to synchronize the CL clock signal with the 2.048 MHz clock signal used in the interrupt system. Signal SCEN is false when the following conditions are true:

a. Phase 6 of Load Program Status Doubleword instruction with R30 true

b. Interrupt operation with INTRAP1 set and INTRAP2 reset. This is the first phase of interrupt operation

c. Phase 3 of Modify and Test Word, Halfword, or Byte instruction with INTRAPF set

When the CLOCK MODE switch is in the SINGLE STEP position and signal SCEN is false, a CL clock signal is generated through gate 5, but SCD may not go true. The SCD latch may no longer disable signal CE; however, the CL clock signal produced by the SINGLE STEP position of the CLOCK MODE switch moves the instruction to the following phase, and signal CE is disabled by a low signal on the NCEINT line. In the case of conditions a and c above, a GARF signal is generated at the next 1MC clock signal:

S/GARF = AIB

C/GARF = 1MC

S/AIB = LEVACT NAIB NGARF

LEVACT = (FAPSD PH7 NO7 R30)
+ (FAS7 PH4 INTRAPF)

Note

The term 1MC in the interrupt system is actually 2.048 MHz.

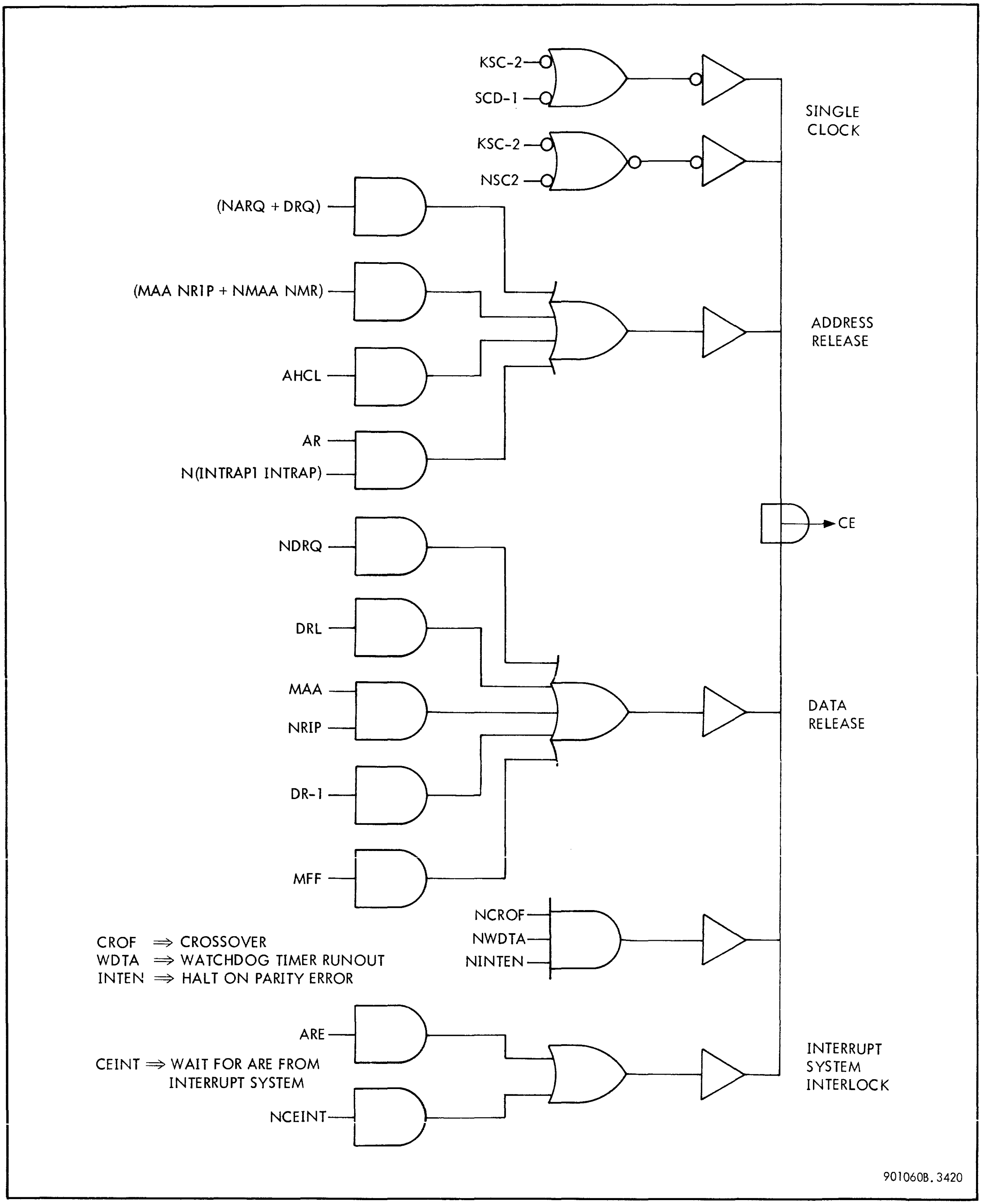


Figure 3-100. Clock Enabling Logic

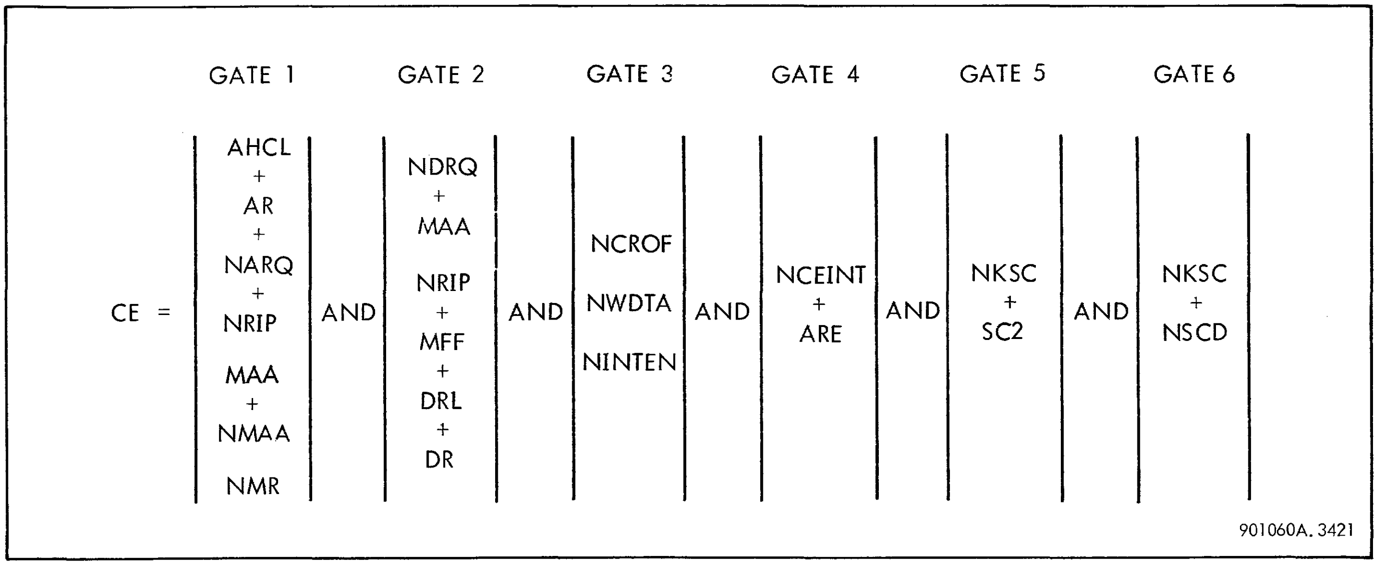


Figure 3-101. Clock Enabling Gates

When GARF is true, NGARF is false, and one input to the ARE gate is enabled. At the leading edge of the 1MC clock signal, ARE goes true, and signal CE is enabled, which produces a CL clock signal in synchronization with the 1MC clock signal.

During interrupt operation, NCEINT is driven low by INTRAP2, INTRAP1, and NTRAP. Signal NAIEB immediately goes false.

S/AIEB = AIEA NAIEB

S/AIEA = NAIEA SAIEA

SAIEA = INTRAP1 INTRAP2 NEWDM

This enables CE when the 1MC signal goes true, as explained in the previous paragraph.

3-81 Memory Map

The function of the memory map as an addressing feature and the use of the memory map registers for memory addressing and access control are described in the general principles. This section describes the type of registers, the logic used to load the registers, and the gating logic for the register outputs during memory access.

3-82 MAP REGISTER ORGANIZATION. The registers that store memory map information are made up of FT25 fast-access memory modules and are identical to the private memory registers. The page address code registers may be represented as shown in figure 3-103. Each FT25 module contains 16 eight-bit registers; 16 modules are used to hold the required 256 registers.

The actual organization of the codes in the integrated circuit memory elements (XDS 304) on an FT25 module is shown in figure 3-104, using the first FT25 module as an example. A description of the flip-flops in a memory element and their logical characteristics are given in the circuit description. Each memory element on an FT25 module contains a corresponding bit in each of eight page address code registers. The elements on the left-hand side of figure 3-104 contain the codes for pages 0-7 and those on the right-hand side for pages 8-F. The other 15 FT25 modules are arranged in a similar manner. The last module contains codes for pages 239 through 247 on the left side and for pages 248 through 256 on the right side.

3-83 LOADING THE MEMORY MAP. A Move to Memory Control instruction with a one in bit 12 transfers the page address codes in core memory, known as the memory map control image, from memory to the memory map registers. The data is first placed in the C-register, then is transferred a byte at a time to D24-D31. From the D-register, each byte is gated onto the memory map data lines, MAPW15 through MAPW22, with the equation:

MAPW15-MAPW22 = D24-D31 MAPWXD

The eight write data lines on each module are connected to the MAPW lines as follows:

W/MP0/15-W/MP0/22 = MAPW15-MAPW22

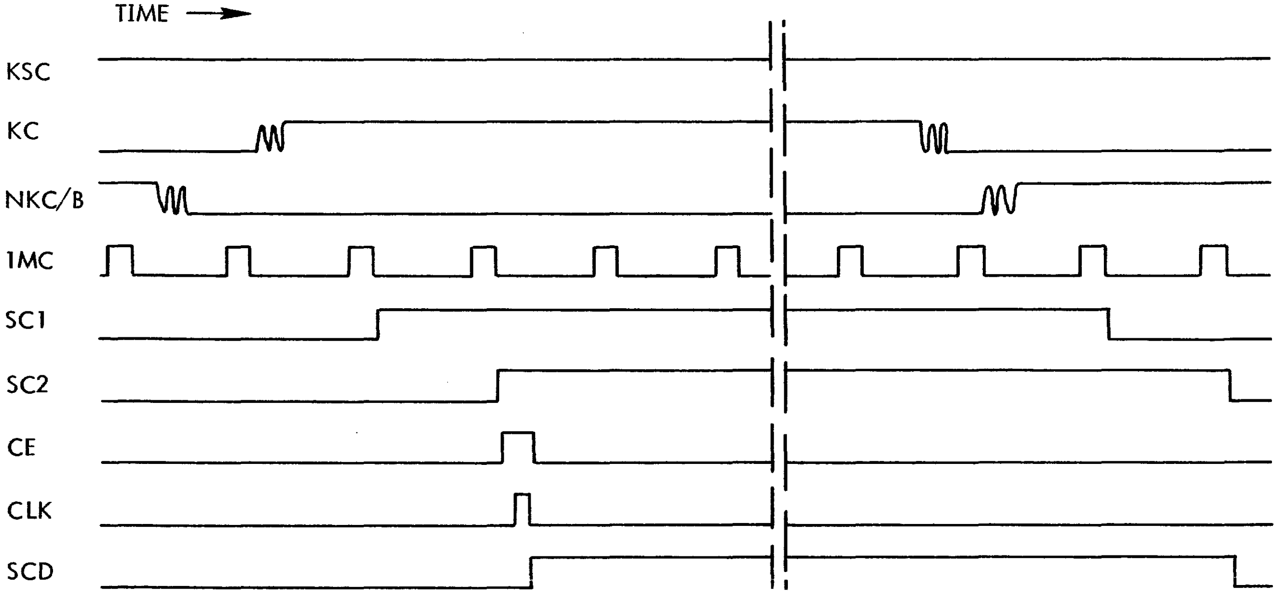
for the first FT25 module, MP0. Other modules are designated MP1 through MP15.

The individual bits in the memory element are selected by address signals generated from the LM lines with the following equations for the first FT25 module:

L/MP0/3 = LM20

L/MP0/4 = LM21

L/MP0/5 = LM22



S/SC1 = KSC KC
R/SC1 = NKC/B SCD
S/SC2 = SC1
R/SC2 = NSC1

SCD = SCD SC2
CLK SC2 SCEN

SCEN = N(FAPSD PH6 NO7 R30)

CE = KSC SCD
KASC NSC2

KSC, CLOCK MODE SW. NOT IN CONTINUOUS POSITION
KC, CLOCK MODE SW. IN SINGLE STEP POSITION
SCD, BUFFER LATCH TO DISABLE CE AFTER SINGLE STEP CLOCK HAS OCCURRED
SC1, FLIP-FLOP
SC2, FLIP-FLOP
CE, CLOCK ENABLE
CLK, BUFFERED CLOCK

901060B.3422

Figure 3-102. Single Clock Mode, Timing Diagram

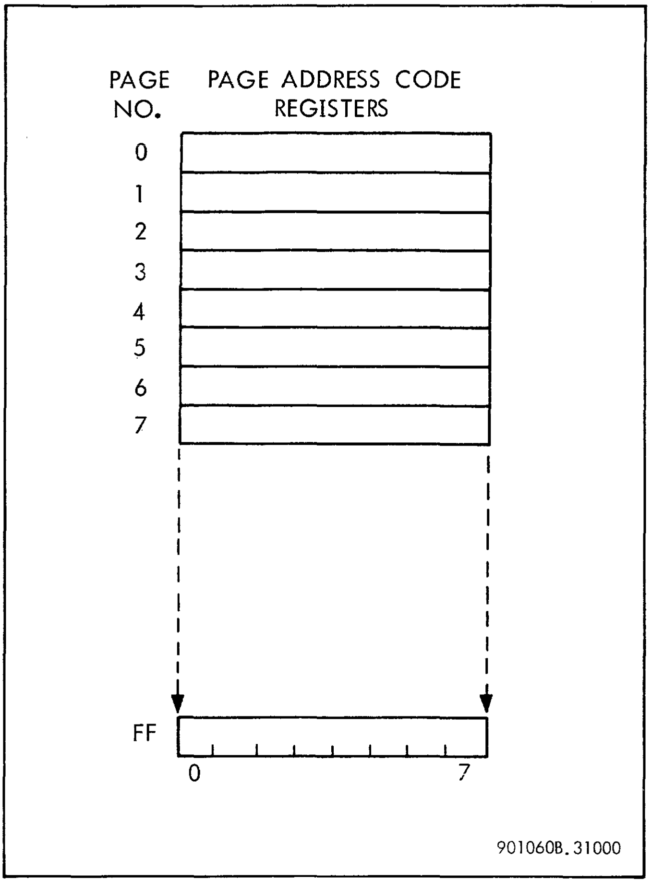


Figure 3-103. Memory Map Page Address Code Registers

The modules are selected by decoding address lines LM15 through LM18, which are inputs to the AND gates in figure 3-104. The two halves of the module are selected by address line LM19, as shown in figure 3-105. The write clock signal is generated from a map write signal and a private memory clock signal with the equation for the first FT25 module:

K/MP0 = MAPW CK

Clock signals for the other modules are K/MP1 through K/MP15.

3-84 PROGRAM CONTROL CODE ORGANIZATION. Four FT25 fast-access memory modules are used to store the 256 two-bit program control codes (called access control codes or access protection codes in the reference manual) associated with the page address codes. The functions of the program control codes are described under General Principles. The registers are represented in figure 3-106.

The actual organization of the codes in the integrated circuit memory elements (XDS 304) on an FT25 module is shown in figure 3-107, and uses the first FT25 module as an example. Memory elements on the left half of the diagram contain the codes for pages 0 through 1F; those on the right half contain the codes for pages 20 through 3F, for a total of 128 bits, or 64 codes, on one module. Because of the limited space, only the first, second, and last bits in each memory element are shown.

3-85 LOADING THE PROGRAM CONTROL CODES. The codes are entered a byte at a time from the D-register (figure 3-108). When one word has been transferred, the eight elements on the left side are half full. The second word fills the left side, and the third word is loaded into the top half of the elements on the right side. After the fourth word, the second FT25 module is selected and the loading continues as described.

A Move to Memory Control instruction with a one in bit 13 transfers the program control codes in core memory, known as the program control image (also referred to in the reference manual as the access protection control image) from memory to the program control registers. The data is first placed in the C-register, then is transferred a byte at a time to D24 through D31. From the D-register, each byte is inverted and is gated onto the program control data lines with the equation for the first FT25 module, PB0:

MAPW15-MAPW22 = D24-D31 MAPWXD

W/PB0/15-W/PB0/22 = NMAPW15-NMAPW22

Other modules are designated PB1 through PB3.

The individual bits in the memory elements are selected by address signals generated from the LM lines with the equations for the first FT25 module:

L/PB0/3 = LM18

L/PB0/4 = LM19

L/PB0/5 = LM20

The modules are selected by decoding address lines LM15 and LM16 which are inputs to the AND gates in figure 3-108. The two halves of the module are selected by address line LM17, as shown in figure 3-107. The write clock signal is generated from a program control bit write signal and a private memory clock signal with the equation for the first FT25 module:

K/PB0 = PCBW CK

Clock signals for the other modules are K/PB1 through K/PB3.

3-86 USING THE MEMORY MAP. Paragraphs 3-87 and 3-88 present a detailed description of the control registers used for the memory map.

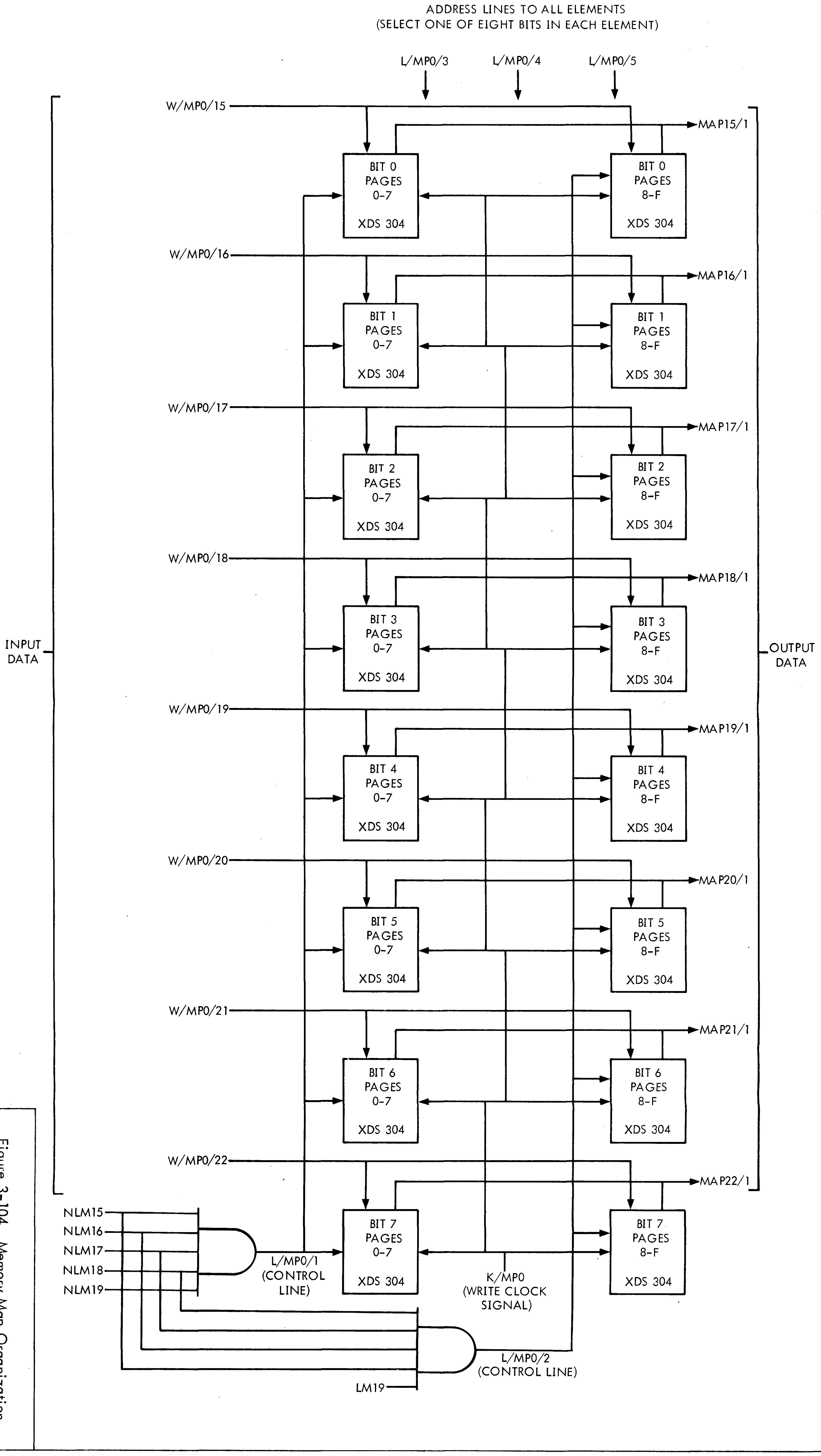


Figure 3-104. Memory Map Organization

901060B. 31301

901060

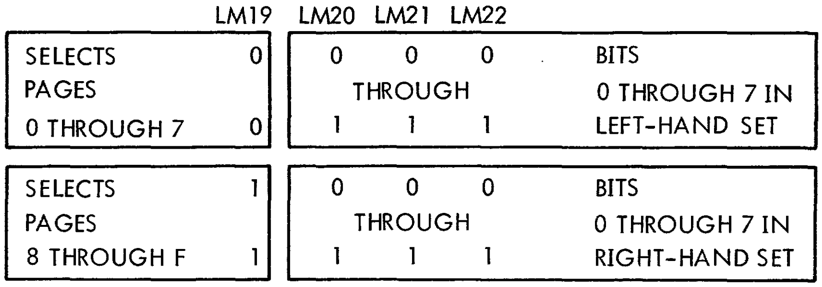

LM19
LM20
LM21
LM22
SELECTS PAGES 0 THROUGH 7
0
0
0 0 0
THROUGH
1 1 1
BITS 0 THROUGH 7 IN LEFT-HAND SET
SELECTS PAGES 8 THROUGH F
1
1
0 0 0
THROUGH
1 1 1
BITS 0 THROUGH 7 IN RIGHT-HAND SET
901060A.31302

Figure 3-105. Memory Map Bit Addressing

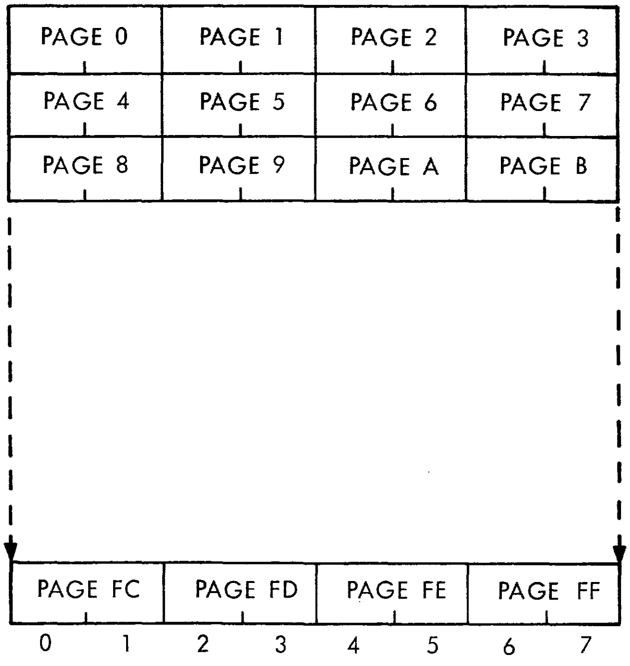
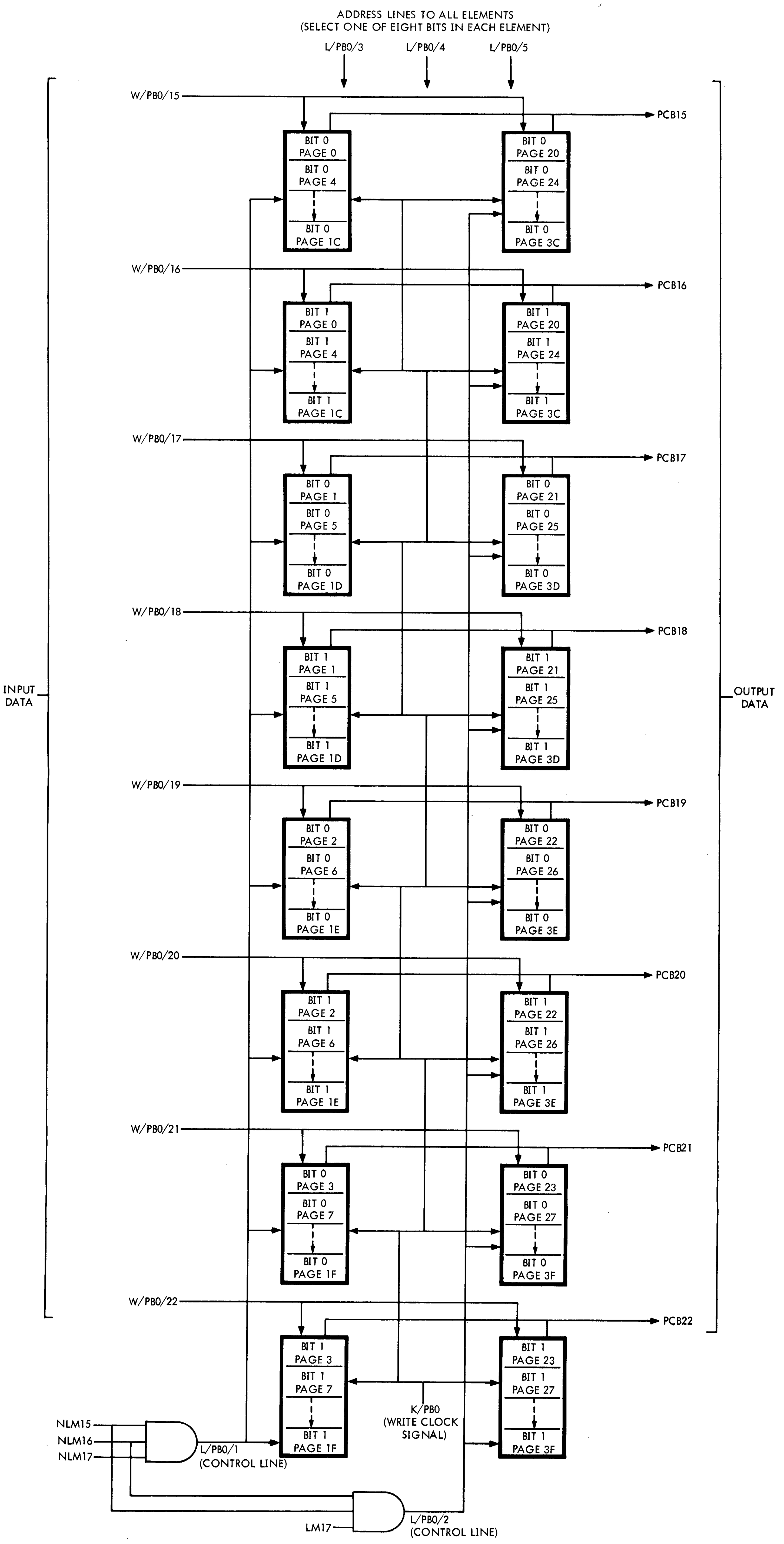


Figure 3-106. Memory Map Program Control Registers



901060

Figure 3-107. Program Control Register Organization

901060A.31304

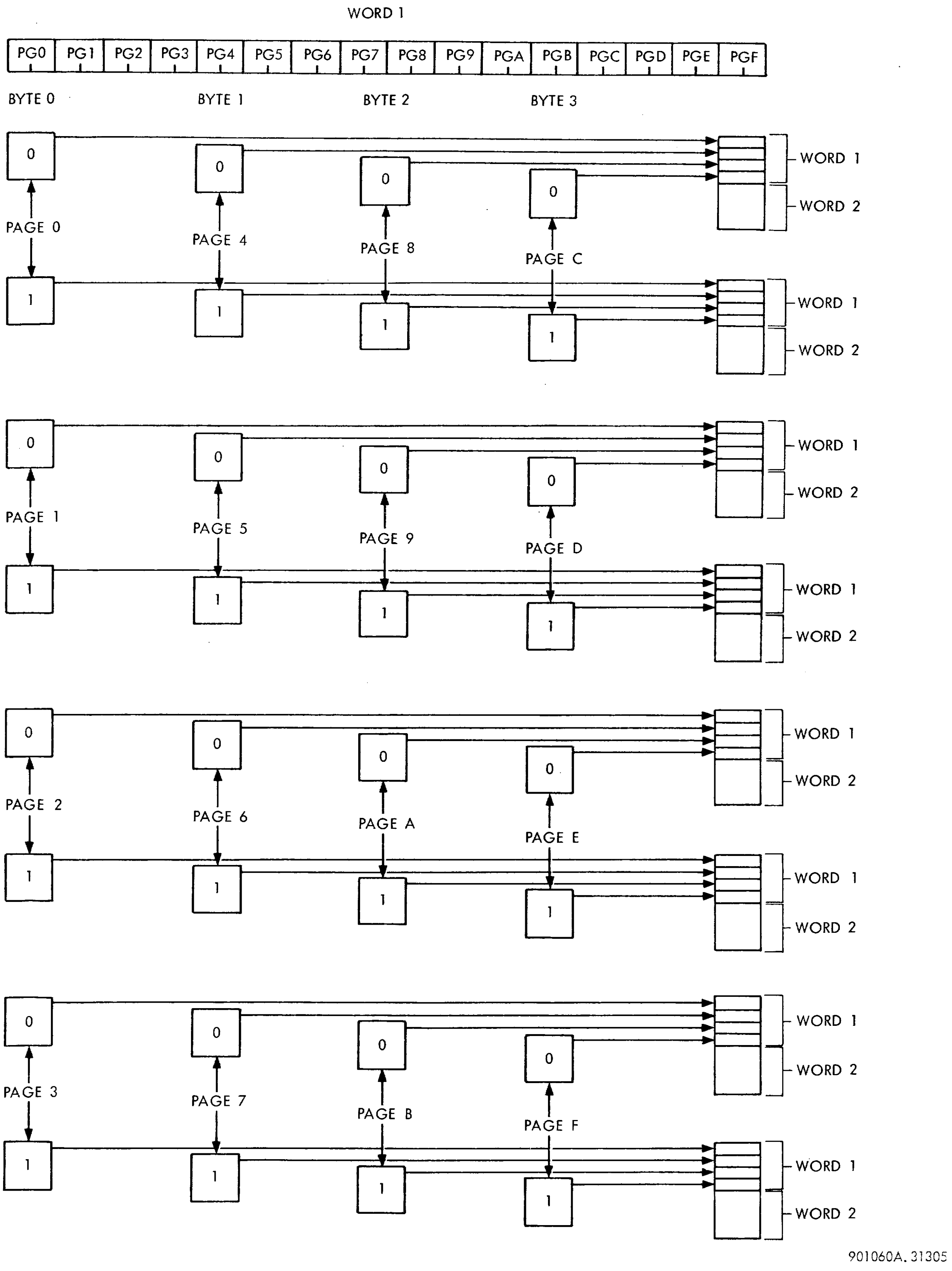


Figure 3-108. Loading One Word of Program Control Codes

3-87 Memory Map Control Registers. The virtual address of an instruction or operand is divided between the LM and the LB lines. The LB lines, LB23 through LB31, go directly to the memory address cable drivers. The LM lines, LM15 through LM22, which are select groups of 512 memory locations known as pages, are connected to the memory map registers. A page of virtual addresses may be located anywhere in memory, depending on the code stored in the map register for that page.

Figure 3-109 shows a typical virtual address on the LM and the LB lines. The most significant eight bits of the address, which specify the page, contain a two, and map register 2 is selected by the address lines (figure 3-105). The outputs of map register 2, which in this case contain a four, are gated onto the most significant eight bits of the LB address lines to memory. Thus, page 2 on the input lines to the memory map is converted to page 4 for actual addressing.

When reading the map register outputs, the modules, module halves, and bits in the memory elements are selected by the LM address lines in the same manner as when storing data. In this case, the register write clock signal need not be true. As long as the control line is true, the outputs of the memory elements connected to that control line may be sensed. Since one control line selects the elements on one-half a module, those eight bits are read at one time on the lines designated MAP15/1 through MAP22/1 or MAP15/2 through MAP22/2 (figure 3-104). The signals ending in /1 are taken from the first eight modules, and the signals ending in /2 are taken from the second eight modules (figure 3-110). The outputs of unselected memory elements remain high; therefore, several outputs may be connected and applied to the same AND gate. If the selected flip-flop contains a one, the outputs of all of the memory elements are true, and the LB line at the AND gate output senses a one. If the selected flip-flop contains a zero, all of the eight lines connected to that flip-flop output are brought low, one input to the AND gate becomes false, and a zero is sensed at the AND gate output. Signal MAP is true when bit 9 in program status doubleword 1 is true, which indicates that the memory map is operative.

Figure 3-110 shows how eight bits at a time are read in parallel from corresponding flip-flops in eight selected memory elements containing one page address code. Only one register on one module is selected at any one time.

3-88 Program Control Registers. The program control registers are used to prevent the slave program from performing certain operations on the associated page of virtual addresses.

To read the program control register outputs, address lines LM15 through LM20 select eight memory elements on a module of map registers and the upper or lower half of each memory element (see figure 3-104). These address lines also select a module of program control registers, a control line for half the module, and one bit in each of the eight memory elements (see figure 3-107). A one on the control line allows outputs PCB15 through PCB22 for the half module to be sensed. Address lines LM21 and LM22 select the page code within the map memory elements. Those lines are also decoded to select one of four two-bit codes in the half module of program control codes, as shown in figure 3-111. The outputs of all the decoding circuits NPCB0 and NPCB1 contain the complement of the code stored with the selected virtual page address. The complement is used because the codes were inverted when they were stored in the program control registers.

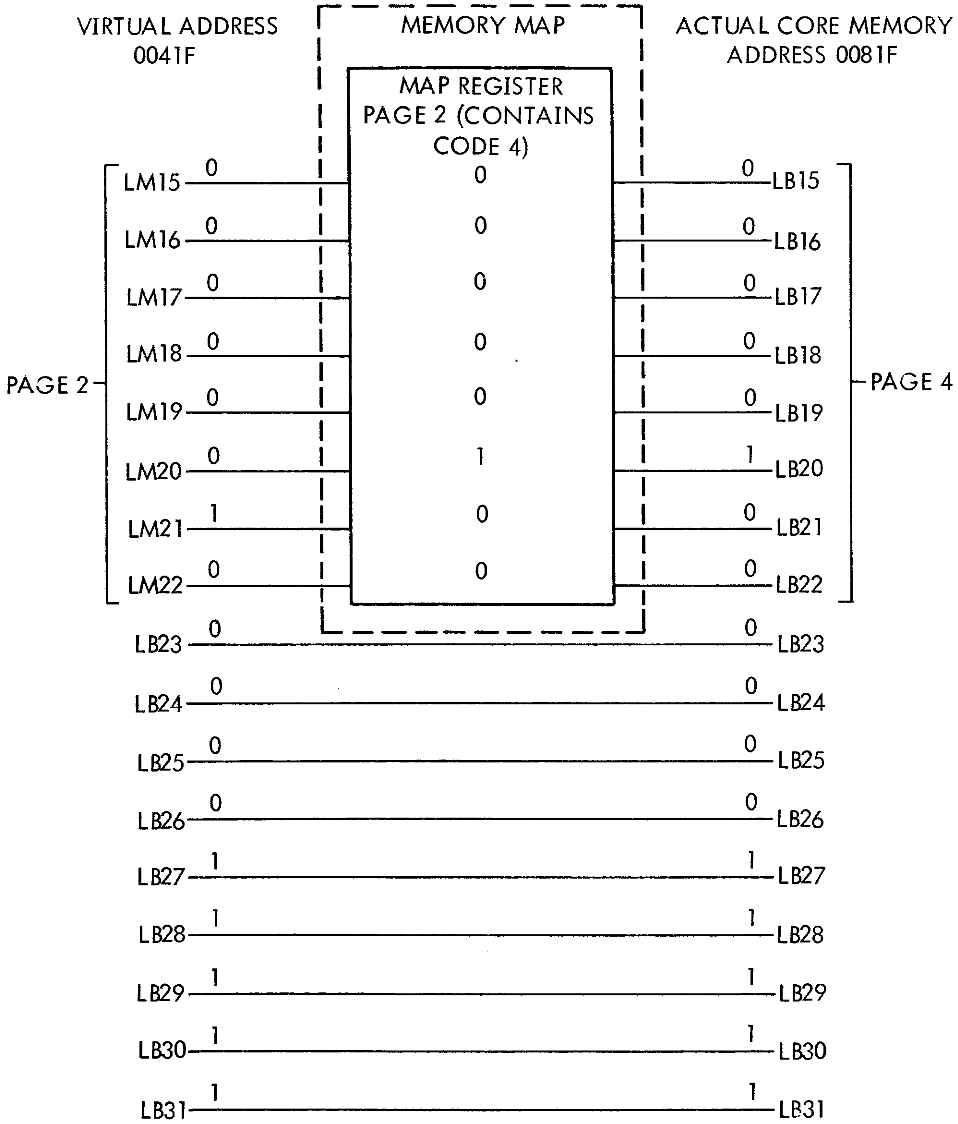
Signals NPCB0 and NPCB1 are interpreted during write operation as shown in table 3-12. A trap condition is produced if the slave program attempts to perform a forbidden operation. Signal PROTECTD (protect data) or PROTECTI (protect instruction) is generated from the program control bits, and the trap flip-flop is set with the equations:

```
S/TRAP      = NRESET S/TRACC4/1
S/TRACC4/1  = (PROTECTD NPROTECTDIS NTRAP
              + PROTECTI ENDE NFUEXU
              NTRAP) NRESET
  PROTECTD  = PROTD DRQ
  PROTD     = NPCP5 NPCP6 FAILD
              + NPCB1 N(FAI0 EXU) NPCB0
              PCB NPCP5 NPCP6 NLMXQ
              MR
              + . . .
  FAILD     = MR [WRITE (NPCB0 PCB + NPCB1
              N(FAI0 EXU) PCB NLMXQ + . . .)]
              + . . .
  PROTECTI  = PROTI DRQ
  PROTI     = (MR NPCB0 PCB NPCP5 NPCP6
              + . . .)
  PCB       = NMASTER MAP
```

If both NPCB0 and NPCB1 are false, no trap condition is produced; reading, writing, and instruction access may be performed.

If NPCB0 is false and NPCB1 is true, only writing is prohibited. A one in NPCB1 and a WRITE signal generate FAILD, which causes PROTECTD to go true, setting the trap flip-flop at the next clock signal. The NLMXQ term on the FAILD AND gate is for mechanization convenience. The NPCP5 and NPCP6 terms disable the PROTD gate during processor control panel phases when the memory is being cleared. The N(FAI0 EXU) term is included because other logic aborts an instruction when the slave program attempts an input/output operation.

If NPCB0 is true and NPCP1 is false, reading operands is permitted while reading instructions and writing are forbidden. Signal FAILD is generated from NPCP0 and WRITE which causes a trap condition through the PROTECTD logic when attempting to write into memory.



901060A. 31306

Figure 3-109. Memory Map Address Conversion

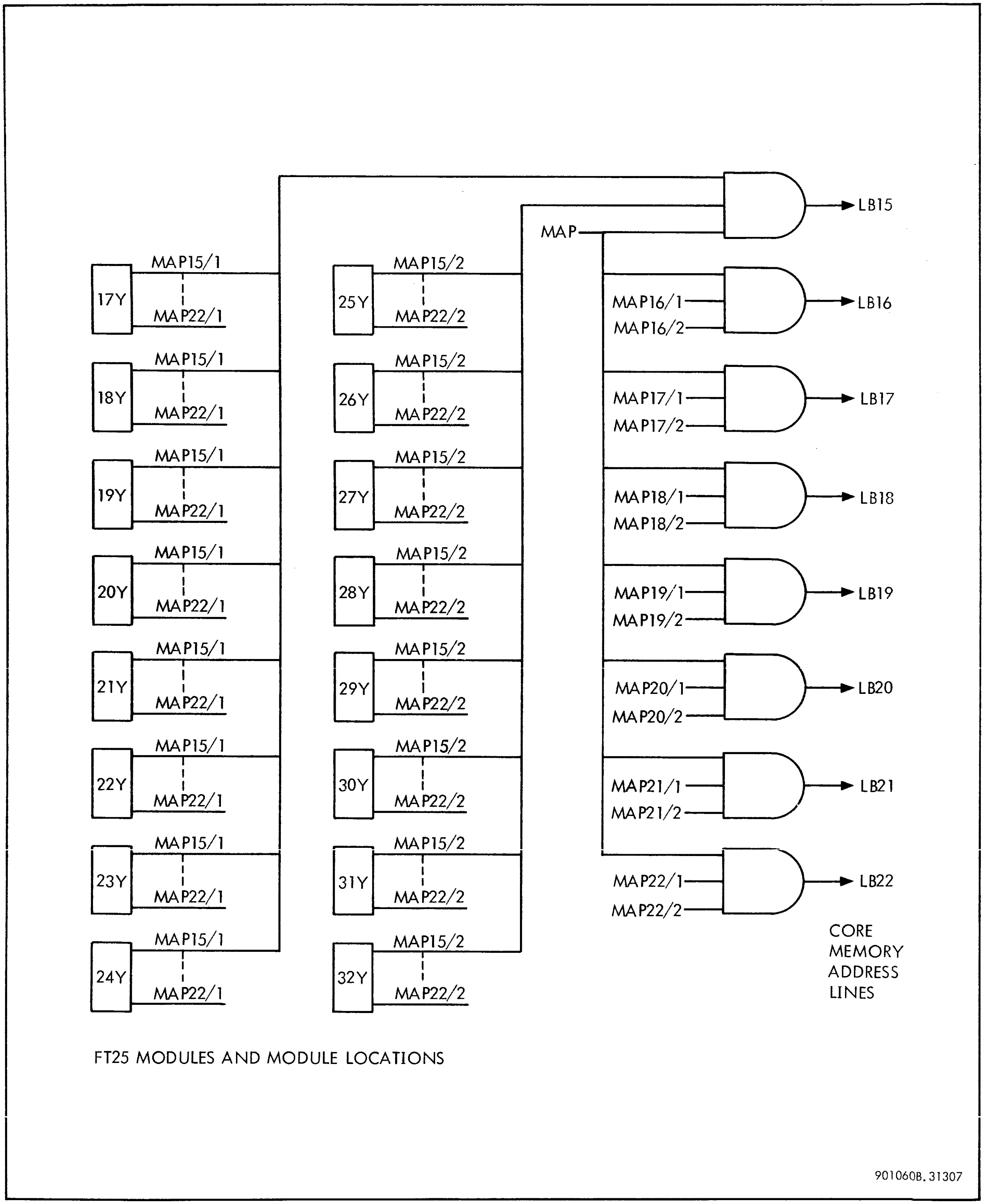


Figure 3-110. Map Register Output Gating

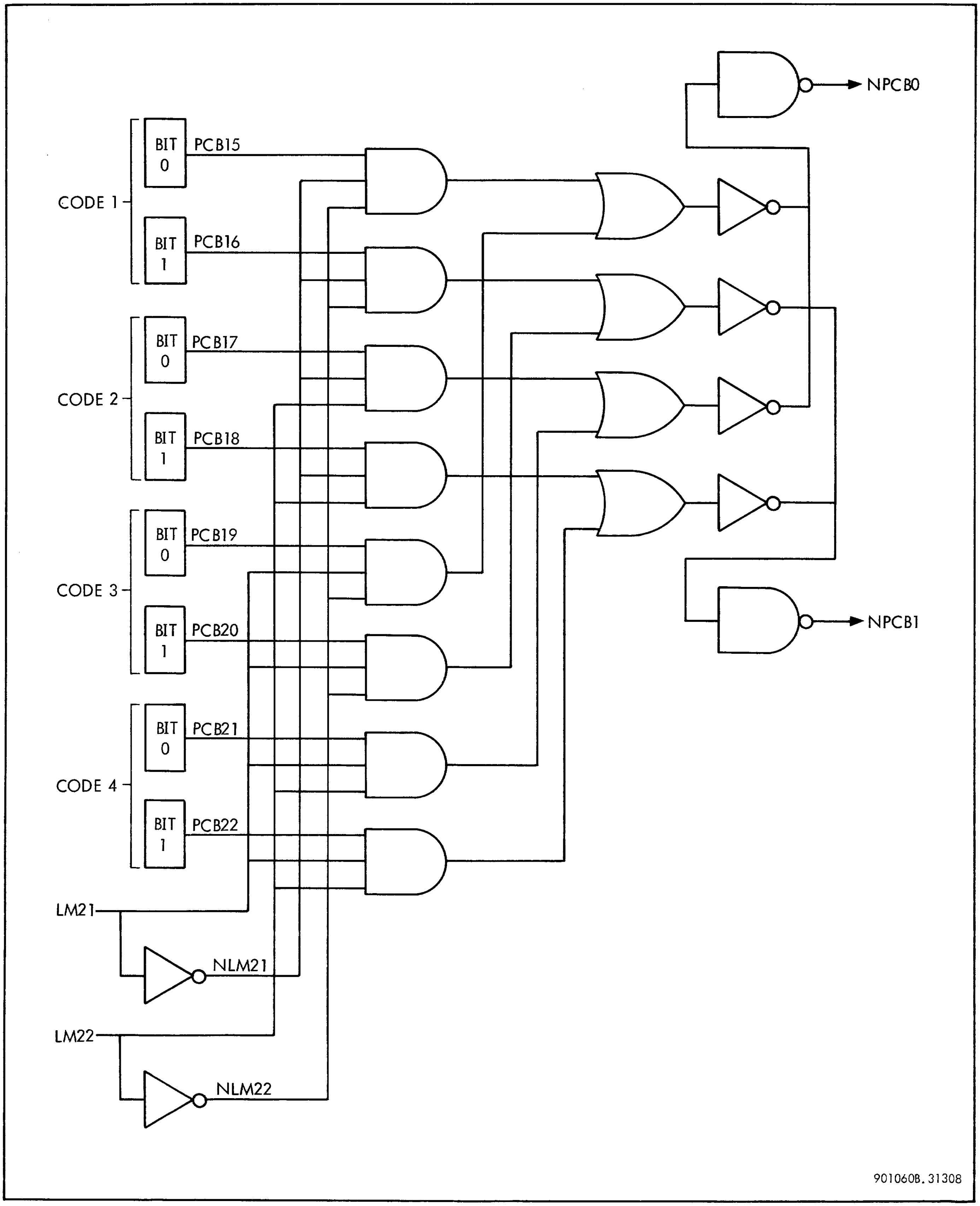


Figure 3-111. Program Control Register Output Decoding

Signal PROTECTI is generated from NPCB0, PCB, and MR, so that a trap occurs when the next instruction is read from memory. Reading data occurs before ENDE is true, and is therefore allowed.

If both NPCB0 and NPCB1 are true, all memory access is prohibited. Writing is inhibited by FAILD, as described above. Reading is disqualified by the gate containing NPCB0 and NPCB1, which produces PROTECTD. If LMXQ is true, indicating that instruction access is being performed, PROTECTI is generated instead of PROTECTD. The difference is one of timing, since PROTECTD sets the trap flip-flop at the next clock, and PROTECTI waits until ENDE is true.

Table 3-12. Program Control Code Functions

NPCB0	NPCB1	Read	Write	Instruction Address
0	0	Yes	Yes	Yes
0	1	Yes	No	Yes
1	0	Yes	No	No
1	1	No	No	No

3-89 Memory Protection

The memory protection feature, which operates independently of the memory map, prevents alteration of specified areas of actual addresses in memory. The program is subjected to memory protection in both master and slave modes. If the CPU is in the slave mode using the memory map, the program control codes are examined when the virtual addresses are converted into actual addresses. The memory protection feature is then used to determine if the program is allowed to alter the contents of the core memory location corresponding to the final actual address.

The memory protection registers contain a two-bit write lock code for each 512-word page of core memory addresses. Two flip-flops designated the write key are set by bits 3 and 4 of program status doubleword 2. The write locks and write key are interpreted as shown in table 3-13. If an instruction attempts to write into a protected memory page, the trap flip-flop is set, and a trap sequence is entered.

Table 3-13. Memory Protection Functions

Write Lock	Write Key	Protection
0 0	x x	Write access permitted independent of key value
x x	0 0	Write access permitted independent of lock value
0 1 through 1 1	0 1 through 1 1	Write access permitted only if lock value matches key value

3-90 MEMORY PROTECTION REGISTER ORGANIZATION.

The 256 memory protection codes are located on four FT25 fast-access memory modules. The registers may be represented as shown in figure 3-106.

The actual organization of the write locks in the integrated circuit memory elements (XDS 304) on an FT25 module is shown in figure 3-112, using the first FT25 module as an example. The write lock bits are placed in the memory elements in the same manner as the codes in the memory map program control registers. The memory elements on the left half of the diagram contain the write locks for pages 0 through 1F; those on the right half contain the write locks for pages 20 through 3F, for a total of 128 bits or 64 write locks on one module. Only the first, second, and last bits in each memory element are shown in the figure.

3-91 LOADING THE WRITE LOCKS.

A Move to Memory Control instruction with a one in bit 14 transfers the write locks in core memory, referenced in the reference manual as the memory lock control image from core memory to the memory protection registers. The data is first placed in the C-register, then transferred a byte at a time to D24 through D31. From the D-register, each byte is inverted and gated onto the lock data lines with the equations:

MAPW15 - MAPW22 = D24 - D31 MAPWXD

W/LK0/15 - W/LK0/22 = NMAPW15 - NMAPW22

where MAPWXD is true during a Move to Memory Control instruction. The second equation applies only to the first FT25 module. Other modules are designated LK1 through LK3 instead of LK0.

The individual bits in the memory elements are selected by address signals generated from the LB lines with the equations for the first FT25 module:

L/LK0/3 = LB18

L/LK0/4 = LB19

L/LK0/5 = LB20

The modules are selected by the decoding address lines LB15 and LB16 which are inputs to the AND gates in figure 3-112. The two halves of the module are selected by address line LB17, as shown in figure 3-112. Use of the LB lines instead of the LM lines ensures that any mapping function takes place before the memory protection feature takes effect.

The write clock signal is generated from write lock signal LOCKW and a private memory clock signal with the equation for the first FT25 module:

K/LK0 = LOCKW CK

Clock signals for the other three modules are K/LK1 through K/LK3.

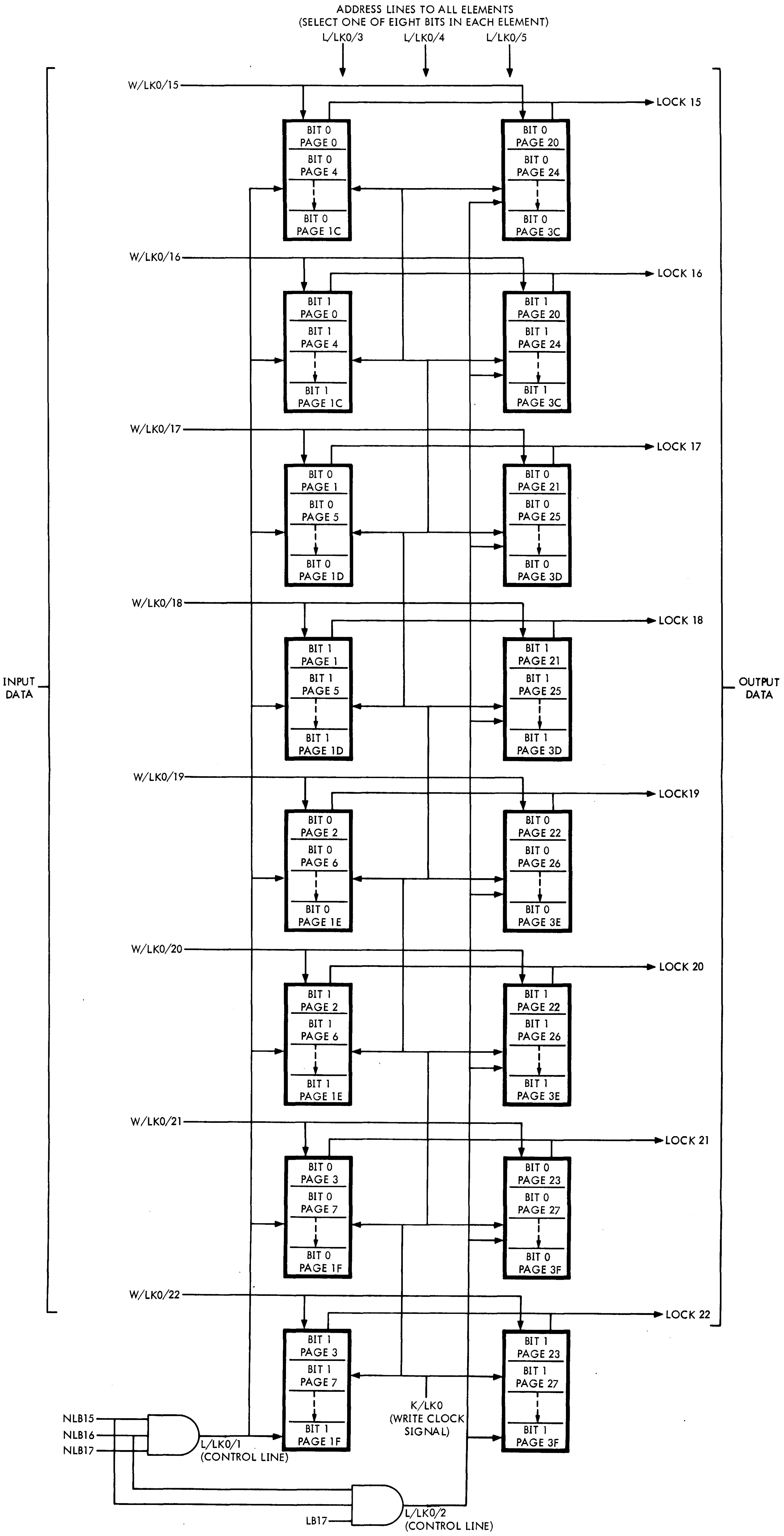


Figure 3-112. Memory Protection Register Organization

901060A. 31309

3-92 USING THE WRITE LOCKS. To read the memory protection register outputs, address lines LB15 through LB20, select a module of write lock registers, a control line for half the module, and one bit in each of the eight memory elements (see figure 3-112). A one on the control line allows sensing of outputs LOCK15 through LOCK22. Address lines LB21 and LB22 are decoded to select one of four two-bit codes in the half module of write locks, as shown in figure 3-113. The outputs of the decoding circuit, NLOCK0 and NLOCK1, contain the complement of the write lock stored with the selected actual page address. The complement is used because the codes were inverted when they were stored in the memory protection registers.

A mismatch of a write lock and the write key generates a FAILD signal, as shown in table 3-14. The logic equation for the FAILD signal is as follows:

$$\begin{aligned}\text{FAILD} = {} & \text{LOCK0 NLOCK1 WK0 CMPKEYS} \\ & + \text{NLOCK0 LOCK1 WK1} \\ & \text{CMPKEYS} \\ & + \text{WRITE (NLOCK0 NWK0} \\ & \text{CMPKEYS} + \text{NLOCK1 NWK1} \\ & \text{CMPKEYS} + \ldots)\end{aligned}$$

$$\text{CMPKEYS} = \text{WRITE NINTRAPF (WK0} + \text{WK1)}$$

From the FAILD signal, a PROTECTD signal is produced, the trap flip-flop is set, and a trap sequence is entered.

Table 3-14. FAILD Signal Generation in Memory Protection

Write Lock	Write Key
0 1	1 x
1 0	x 1
1 x	0 x
x 1	x 0

3-93 Watchdog Timer

The watchdog timer is a six-bit flip-flop binary counter, triggered by the 1-MHz clock signal. The counter is set at interruptible points in the program and counts to 40, which allows 40 μs before runout. If the timer runs out before another interruptible point is reached, a trap sequence is entered.

A logic diagram of the watchdog timer counter is shown in figure 3-114. The control circuits and trap-setting logic are shown in figure 3-115. The counter is started by loading it with ones at the following times:

a. When the counter contains 1010XX

b. At the first clock pulse after the interrupt enable flip-flop IEN has been set

c. At the first clock pulse after the END signal goes true during the last execution phase of an instruction

d. At phase 4 of a Move to Memory Control instruction

e. While the WATCHDOG TIMER switch is set at the OVERRIDE position

f. At phase 6 of byte operation instructions

g. While the CLOCK MODE switch is not in the CONT position

h. While the COMPUTE switch is not in the RUN position

i. At processor control panel phases 2 through 6

j. At phase 6 of Decimal Add and Decimal Compare instructions

Any one of these conditions drives WDTRAC true, and flip-flops WCT1 through WCT6 are set.

When the watchdog timer has counted to 40 without being restarted, flip-flop WDTA is set by WCT1, NWCT2, WCT3, and NWCT4. A one at the set output of WDTA causes WDTRAC to be set which restarts the counter and causes WDTLATCH to go true. Feedback from WDTLATCH holds this signal true until STRAP and TRAP are true.

When WDTLATCH goes true, STRAP is latched true, and the trap flip-flop is set. As soon as STRAP and TRAP are true, WDTLATCH drops, and is followed by STRAP. A trap sequence is then entered to take the program to location X'46'.

Any one of the conditions stated in the above-mentioned steps restarts the counter without causing a trap. Signal WDTR goes true, setting flip-flop WDTRAC, but WDTLATCH stays low because WDTA has not been set. In the case of KWDTR, KSC, and NRRUN/B, the set input to flip-flop WDTRAC is held true as long as the switch is in the position indicated, and the counter is not allowed to step.

3-94 Real-Time Clocks

The real-time clock signals are obtained from an ST29 Time Base Selector, which, in turn, receives its clock pulses from a CT16 Medium Frequency Oscillator. A simplified diagram of these circuits is shown in figure 3-116. Outputs from the time base selector are applied to the interrupt circuits on the LT16 Priority Interrupt modules in the CPU.

A 2-MHz sine wave from the oscillator, shown in figure 3-116, goes through a frequency divider consisting of seven flip-flops, each of which divides the frequency by two. The 16-kHz signal at the output of the oscillator module is connected to the time base selector, where the frequency is again divided by a series of five flip-flops.

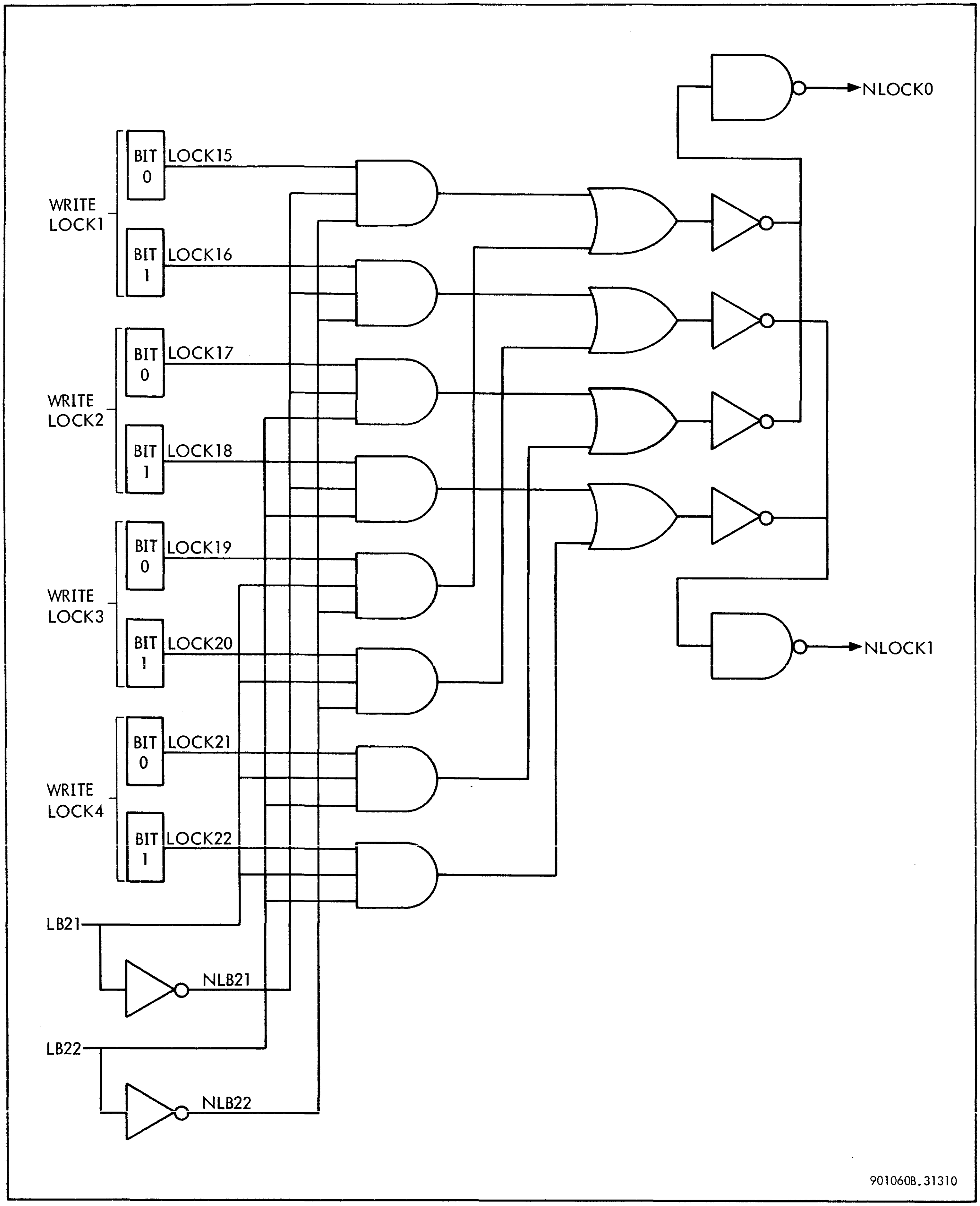


Figure 3-113. Memory Protection Register Output Decoding

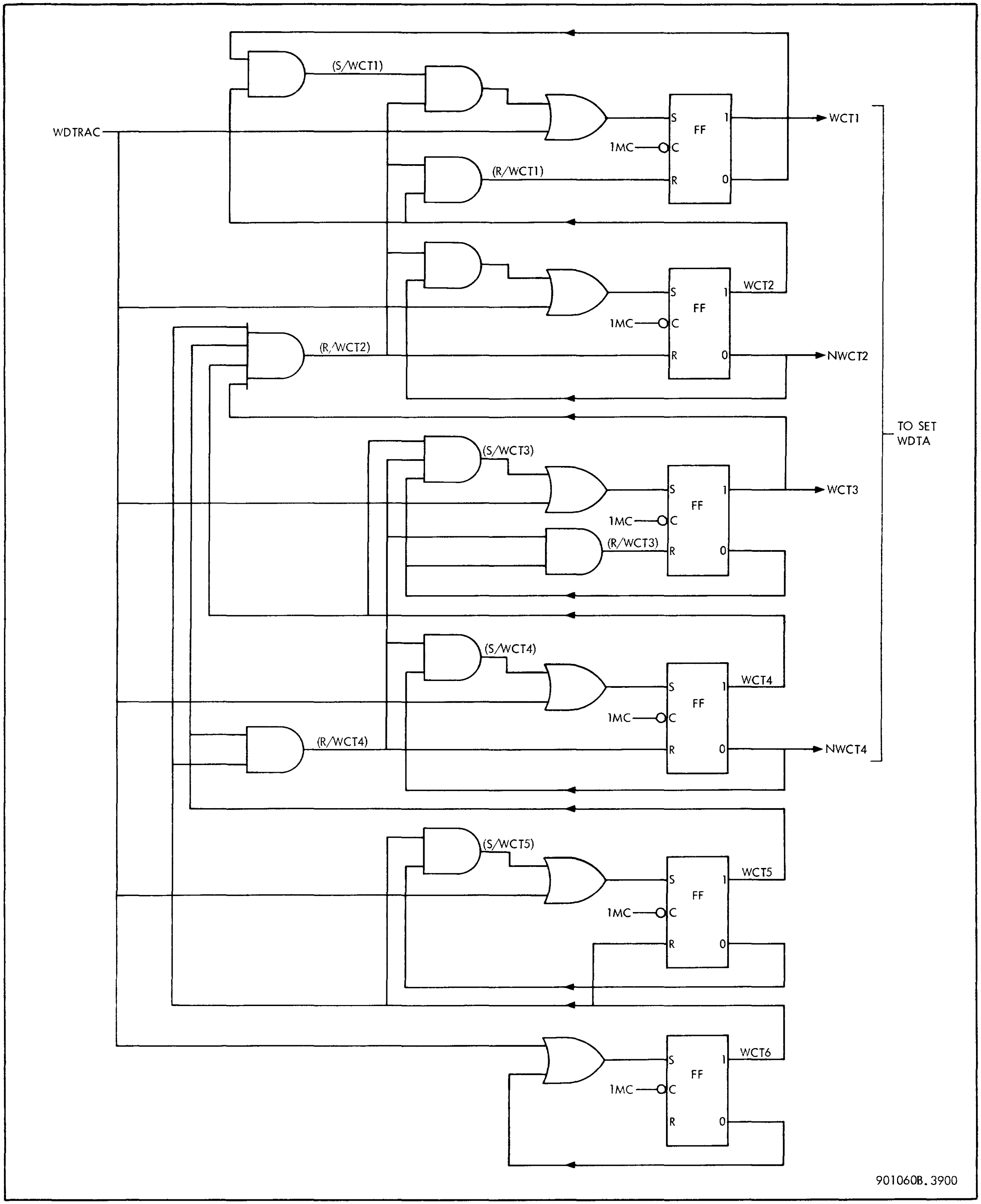


Figure 3-114. Watchdog Timer Counter, Logic Diagram

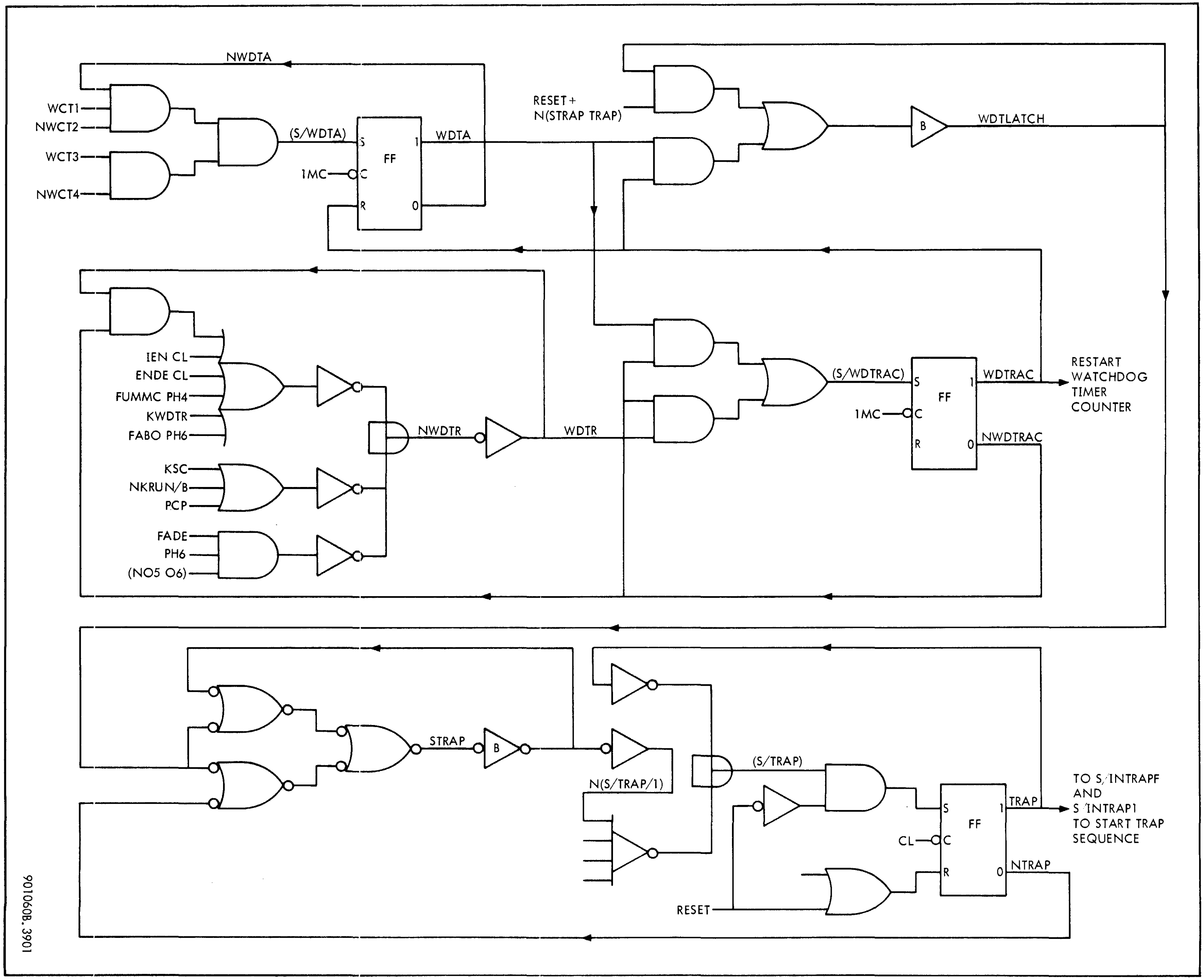


Figure 3-115. Watchdog Timer Control Circuits, Logic Diagram

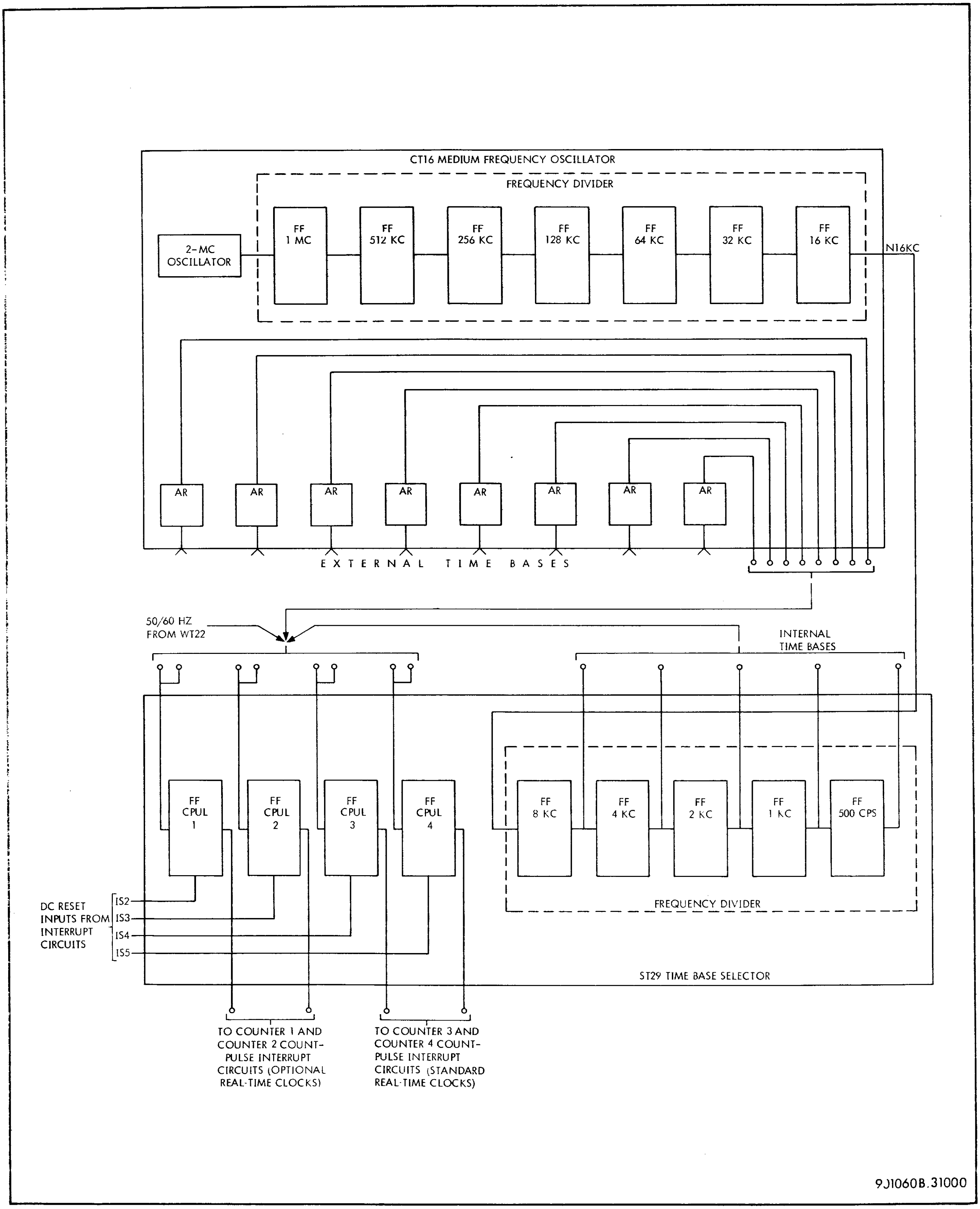


Figure 3-116. Real-Time Clock, Simplified Diagram

The 8-kHz, 4-kHz, 2-kHz, 1-kHz, and 500-kHz square wave outputs of these flip-flops are available as internal time bases, to be used as inputs to the real-time clock flip-flops CPUL1 through CPUL4. The clock frequencies may be selected by jumpers from the frequency divider outputs to the clock pulse flip-flop inputs.

Time bases from sources outside the CPU may be used by connecting external frequencies to the amplifiers shown on the oscillator module in figure 3-116. The amplifier outputs may be connected by jumpers to the desired clock pulse flip-flops on the time base selector. The line frequency of 50 or 60 Hz may also be used by connecting the clock pulse flip-flop inputs to a 50/60-Hz source elsewhere in the computer.

The outputs of flip-flops CPUL3 and CPUL4 are taken to the priority interrupt modules used to process the standard counter 3 count-pulse and counter 4 count-pulse interrupts. The outputs of flip-flops CPUL1 and CPUL2 are used only if the optional counter 1 count-pulse and counter 2 count-pulse interrupt levels are included in the CPU. The events that occur after a count-pulse signal enters the interrupt circuits are described in the section on interrupts (beginning with paragraph 3-108).

3-95 Power Fail-Safe

3-96 GENERAL. The Sigma 7 optional power fail-safe feature monitors the primary power sources and provides automatic shutdown in cases of partial or complete power failure where power drops below given limits. If power returns to an acceptable level, normal operation automatically resumes. During power fail-safe shutdown, information in certain volatile flip-flop registers is stored in core memory to prevent critical program data loss. When power is restored, the information in core memory is returned to the volatile flip-flop registers so that the program can resume at or near the interrupted point. Core memory serves as the storage device during power fail-safe operation because the cores are nonvolatile and retain information without the presence of power.

The power fail-safe feature is composed of two major components: the power fail-safe interrupts, which initiate the save and recovery programs, and the power monitor assembly, which monitors the primary power source.

3-97 POWER FAIL-SAFE INTERRUPTS. When a power failure occurs, the power fail-safe feature notifies the CPU by means of a power-off interrupt. Sufficient energy is stored in the Sigma 7 power supply system to maintain dc power for the duration of a short power failure subroutine. When primary power resumes, a power-on interrupt causes the CPU to enter a recovery subroutine that restores the CPU to the state existing prior to the lapse of power.

The interrupt memory locations are X'50' for the power-on interrupt and X'51' for the power-off interrupt. The power-on interrupt is the highest priority interrupt in the system; power-off interrupt has second highest priority. Both of these interrupt levels are always enabled; neither can be disarmed, disabled, inhibited, or triggered under program control.

3-98 POWER MONITOR ASSEMBLY. Figure 3-117 is a simplified block diagram of the power monitor assembly, which is a standard equipment item in Sigma 7, consisting of three standard modules: the WT21 regulator and independent power supply, the WT22 line detector, and the AT13 line driver.

The WT21 regulator is used to supply regulated dc voltages to the WT22 line detector and the AT13 line driver.

The WT22 line detector performs the function of detecting a power failure and indirectly providing the necessary signals to the CPU for a start-up or shutdown sequence.

The AT13 line driver is basically a cable driver which is used to drive the output signals of the WT22 module.

Although the primary power sources are optional, depending on user requirements, the primary power source shown in the simplified block diagram is single phase 120 Vac.

The application of primary power to the Sigma 7 system power supplies provides the power fail-safe feature with the voltage necessary to power the WT21, WT22, and AT13 modules. These standard dc voltages are provided by the internal power supply in the WT21 regulator. The power fail-safe feature receives 120 Vac and 60 Vdc power when primary power is applied. The 120 Vac power is transmitted to the WT21 regulator, which in turn is converted to regulated +4 Vdc, +8 Vdc, and -8 Vdc. The regulated dc voltages are transmitted to the WT22 and AT13 module.

The 60-Vdc power output from the PT14 power supply is monitored directly to the WT22 line detector, which senses this input and the other dc voltages to determine if they are within acceptable limits.

The requirements for a start-up or shutdown sequence are governed by the WT22 line detector, which contains the basic sensing circuits within the power fail-safe feature. Detection of one or more out-of-tolerance voltages by the WT22 line detector generates the necessary logic signals to the AT13 line driver for a fail-safe shutdown. A subsequent return of voltages that are within tolerance generates the necessary logic signals to the AT13 line driver for a fail-safe start-up.

3-99 Real-Time Clock Line Frequency. The real-time clock circuit on the WT22 module generates a stable clock frequency synchronized to the line frequency. This real-time clock is not an integral part of the power fail-safe feature and is located on the WT22 module primarily for purposes of convenience.

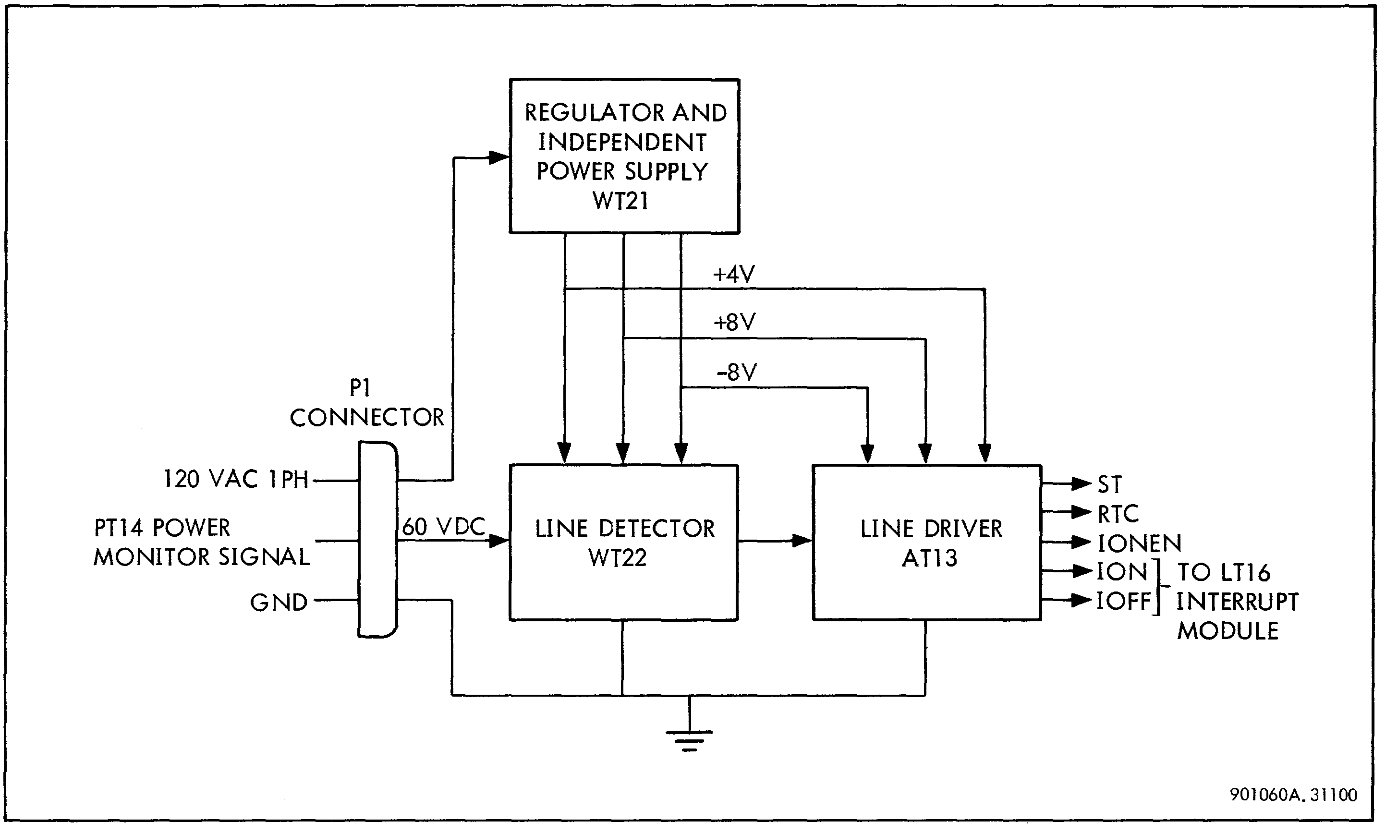


Figure 3-117. Power Fail-Safe Feature, Simplified Block Diagram

3-100 Input Requirements. The input power source varies according to user requirements. For information on the various power sources, refer to the section on Power Distribution (paragraph 3-341).

3-101 Three-Phase Input Detection. When three-phase operation is used, one phase is used to supply power to the power monitor. The presence of three phases is detected by sensing the presence of a three-phase rectified but unfiltered 60-Vdc signal supplied by the PT14 power supply.

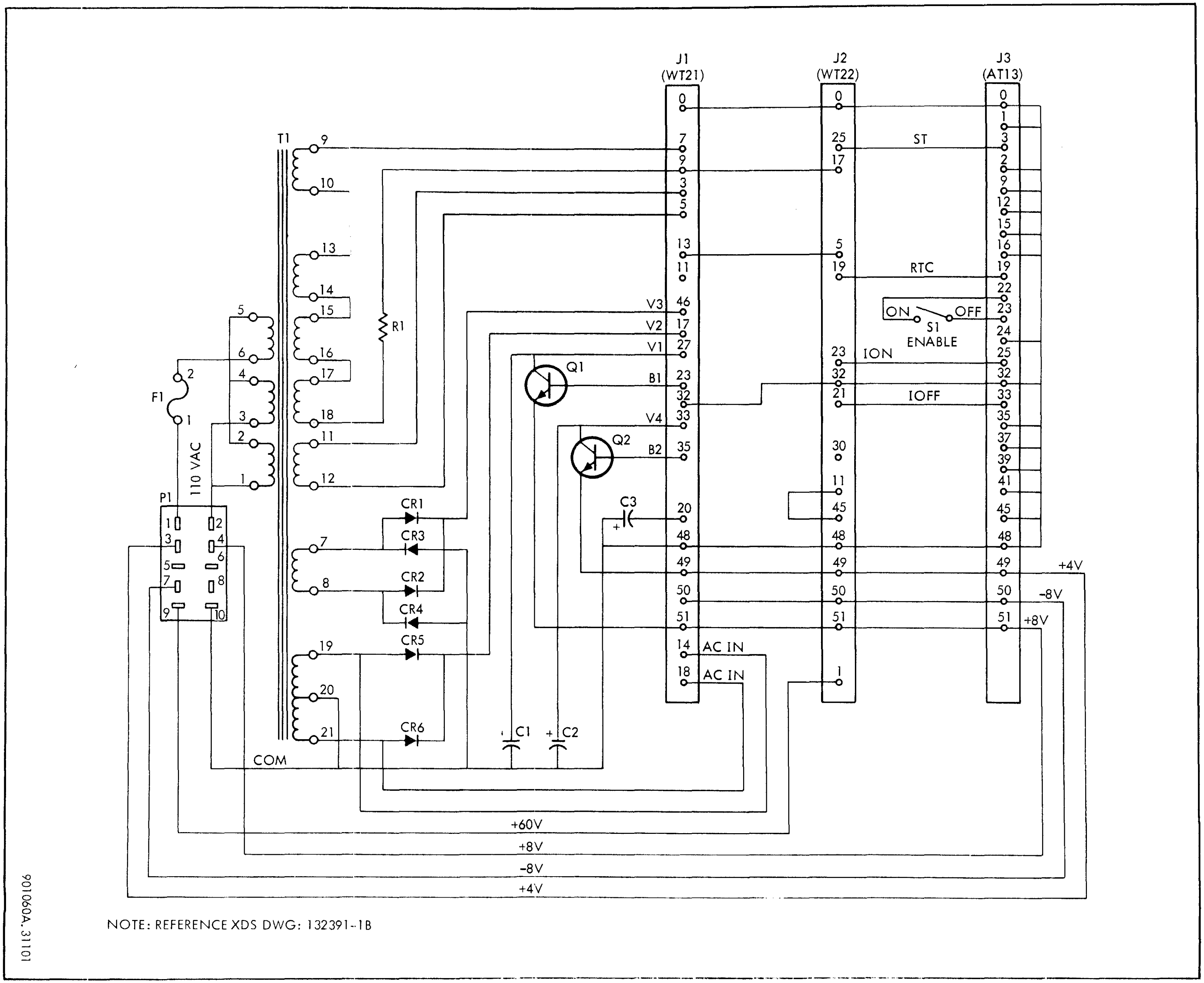
3-102 Single-Phase Detection. Single-phase detection is provided for by a simple rewiring in the power monitor. When rewired, this standard 110-Vac line is the only external input to the power monitor assembly.

3-103 Internal Power Supply. The power monitor has its own internal power supply capable of delivering power to the WT21, WT22, and AT13 modules. This supply comes into operation when external power is applied. When power is shut off, this internal supply outlasts the dc supplies in the computer, thereby keeping logic signals in their appropriate states as power decays and the power-off subroutine is executed.

3-104 Parallel Operation. The power monitor is capable of paralleling its output with the output of other power monitors. This is necessary, since several power monitors may be used to monitor individual lines and power supplies in a system. Therefore, if more than one power monitor is used in a given installation, the equivalent logic outputs of the power monitors are OR-gated.

3-105 Output Signals. There are five output signals from the AT13 line drivers: ST, the master reset signal; ION, which initiates the start-up sequence; IOFF, which initiates the shutdown sequence; IONEN, the ION enable signal, which performs an AND function for the output of the power monitor assembly; and RTC, the real-time clock signal, which is a clock synchronized to the line frequency, but which is not used directly in the power fail-safe feature.

3-106 Circuit Description. Figure 3-118 is a functional schematic of the power monitor assembly. Input power to the internal power supply on the WT21 regulator is shown to be single phase 110/120 Vac from pins 1 and 2 of the P1 connector. This input power is transmitted to transformer T1, which is in the internal power supply of the WT21 regulator package. Diodes CR1 through CR4 comprise a full-wave bridge rectifier and provide the dc inputs to the +8 Vdc and the +4 Vdc regulator drivers. Diodes CR5 and CR6 act as a full wave rectifier providing ac input to the -8 Vdc regulator circuit.



NOTE: REFERENCE XDS DWG: 132391-1B

901060A.31101

Figure 3-118. Power Monitor, Functional Schematic Diagram

a. WT21 Regulator. The WT21 (see figure 3-119 for the schematic) is the voltage regulator-driver used to supply regulated dc to the WT22 power monitor. The WT21 contains a +8 Vdc regulator-driver, +4 Vdc regulator-driver, and -8 Vdc regulator. The +8-volt and +4-volt drivers are used with external pass transistors. The -8-volt regulator contains the pass transistor located on the module. The -8-volt regulator contains a rectifier circuit to allow operation from ac inputs at pins 14 and 18. The +8 and +4-volt regulator-drivers require dc input voltages. Two additional bridge rectifying circuits are located on the WT21 to provide 24 Vdc and 50 Vdc.

Filter capacitors for input filtering to the series regulators are not located on the module; however, provisions are made for external connections. Surge resistors are located on the WT21 to prevent damage to the external rectifiers that supply current to the +8-Vdc and +4-Vdc regulators.

R1 through R4 are the surge protection resistors. CR1 through CR4 and CR5 through CR8 rectify the ac input voltages, which are then used to operate the power monitor with 24 Vdc and 50 Vdc.

Transistors Q2 and Q1 are the sense and drive transistors used in the +8-Vdc regulator. The input of the regulator is pin 17 (V2). Pin 27 (V1) is brought out to connect to an external filter capacitor. Voltage adjustment is accomplished by controlling the current in Q2. This current is determined by the emitter voltage (reference voltage) and the sampled base voltage as adjusted by R8. The Q1 emitter output drives the external pass transistor.

Transistors Q3 and Q4 are the drive and sense transistors for the +4-volt regulator. Collector voltages to Q3 are derived from the +8-volt supply. Drive voltage to Q3 also comes from the +8 volts, therefore providing preregulation for the +4-volt regulator. Q3 drives a power transistor external to the WT21. The base of Q4 is connected to the +4-volt output. Current is controlled by changing the reference voltage, R12, at the emitter of Q4.

The -8-volt regulator, including pass transistor Q5, is located on the WT21. CR9 and CR10 provide a negative supply voltage when ac is applied to pins 18 and 14. Pin 20 is the common and external capacitor connection. R17 provides voltage control of the output.

b. WT22 Line Detector. Figure 3-120 is a block diagram of the WT22; figure 3-121 is the WT22 schematic; and figure 3-122 shows the WT22 waveforms. Basic timing and input power for the WT22 are as follows:

Delay time D	adjustable from 5 to 20 ms
ION time A	300 ms ±10%
Power failure detection time	< 3 ms
Input power	+8 Vdc @ 40 mA +4 Vdc @ 30 mA +60 Vdc @ 50 mA for 3Ø detection +22 Vdc @ 10 mA for 1Ø detection

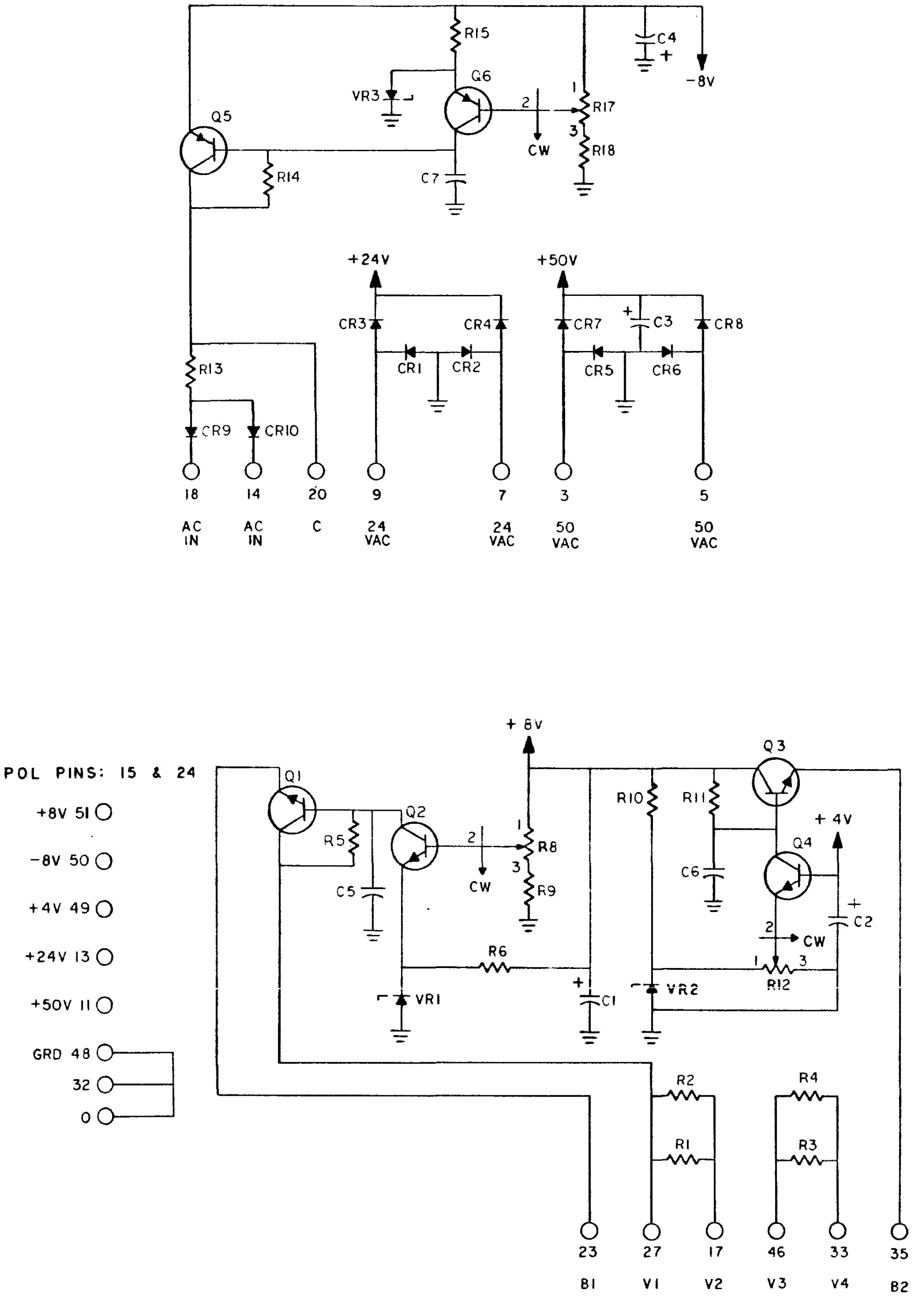
The WT22 is the line detector module that provides all output signals for the power monitor. This is the basic unit within the power fail-safe feature. The principal function of the WT22 is to detect a power failure and provide the necessary reset and interrupt signals for the CPU. These signals provide start-up and shutdown sequences when power comes on and goes off.

When power is first turned on (figure 3-121) the 307 flip-flop is dc set by V1 (+8 Vdc), charging C8, causing ST to go high. VD, the ION threshold voltage, is also applied and charges C4 through R13. The voltage across C4 is determined by the magnitude of VD through R13, as well as the time constant R13-C4. This time constant determines the time after power has been turned on that the threshold sensing circuit triggers the ION pulse. As shown in figure 3-122, the occurrence of ION is time A, or the time necessary for all dc power supplies in the computer to stabilize. The ION pulse resets the 307 flip-flop, and ST falls to zero. If ST is zero, the reset of the 307 flip-flop is high and prevents C4 from charging by holding the NOR gate on.

When the line detection circuit indicates power failing, it generates the IOFF signal. At the same time, the IOFF signal is delayed by the period D which is the maximum time that dc power supplies remain within regulation after a power failure (figure 3-122). IOFF is applied to the clock input of the 307 flip-flop, and since the set is held high, the flip-flop sets when IOFF returns to zero at the conclusion of period D. IOFF pulses continue to be generated as long as power is below the acceptable threshold level.

As shown in figure 3-122, IOFF pulses and NST prevent C4 from charging in case of a short power interrupt. As long as IOFF produces a pulse, C4 discharges, preventing an ION pulse until C4 charges up again. During this time ST is held high by IOFF, setting the 307 flip-flop continuously.

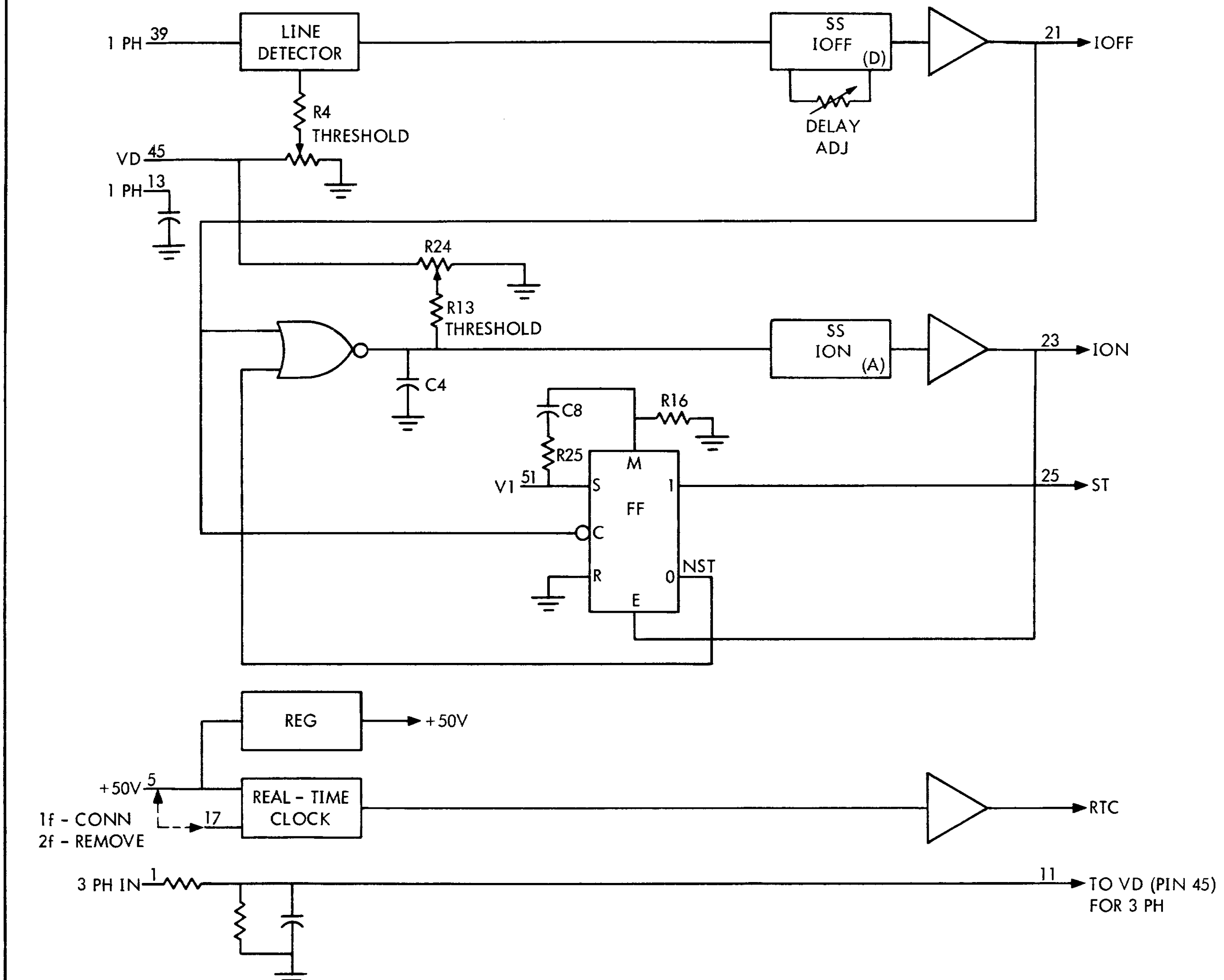
ST goes false when ION goes true, and there are no IOFF pulses present. This means that any time an IOFF pulse occurs, the entire start-up sequence takes place. When power returns, the line detection circuit generates IOFF pulses whenever the line voltage drops below threshold. Threshold oscillation is prevented by a preset hysteresis. IOFF and ION, as determined by their respective threshold settings, may be set 10 volts apart; for example, IOFF is present at 80-volts line and ION occurs at 90-volts line. ST goes high when the line drops below 80 volts and remains high until the line raises above 90 volts.



NOTE: REFERENCE XDS DWG: 132374-1B

901060A.31102

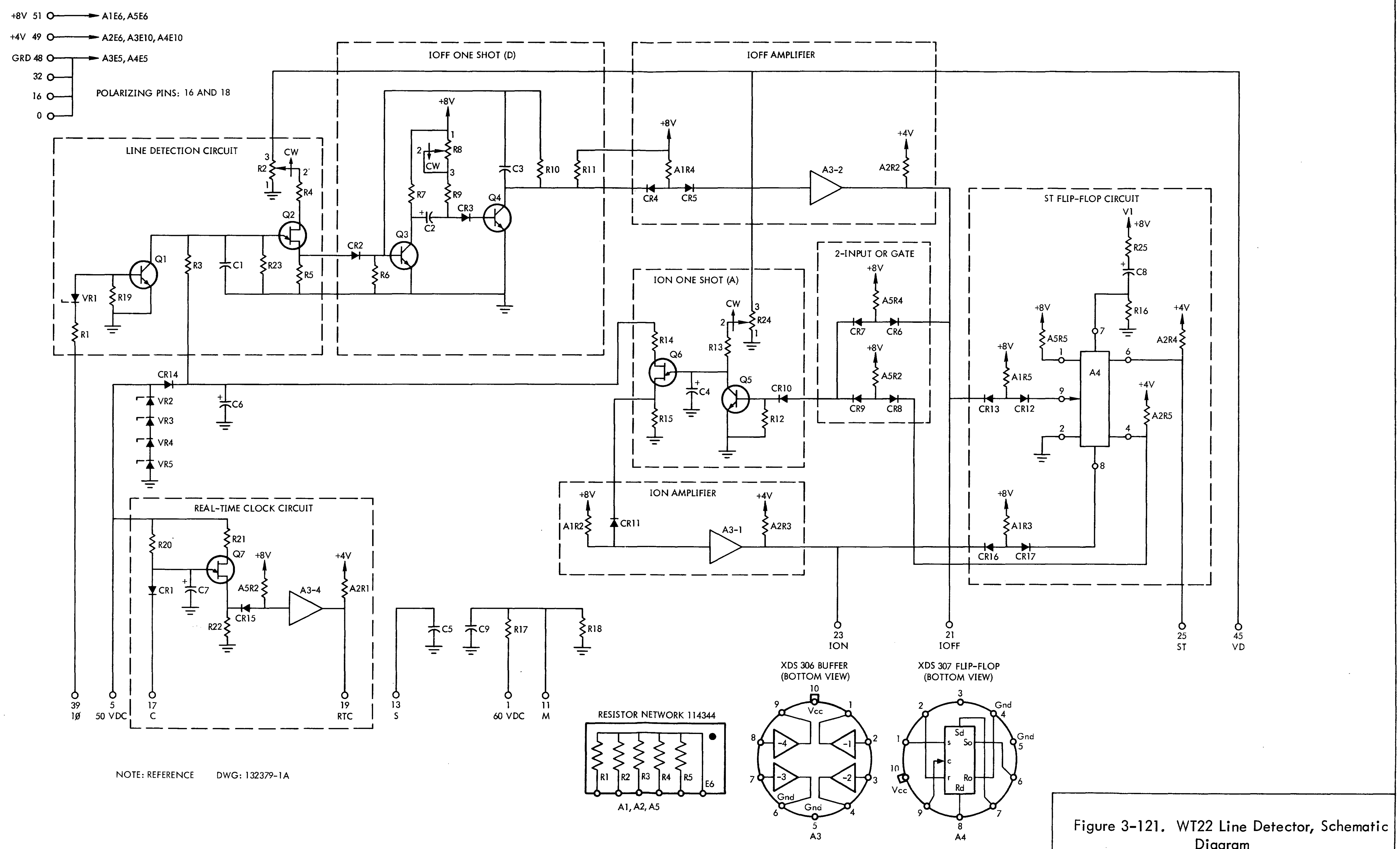
Figure 3-119. WT21 Regulator, Schematic Diagram



CONNECTIONS :
1 PH = 22V UNREG TO 13 AND 45; CONN 5 TO 39 ; 1 AND 11 NO CONN
3 PH = INPUT TO 1; CONN 11 TO 45; 13 AND 39 NO CONN

901060A. 31103

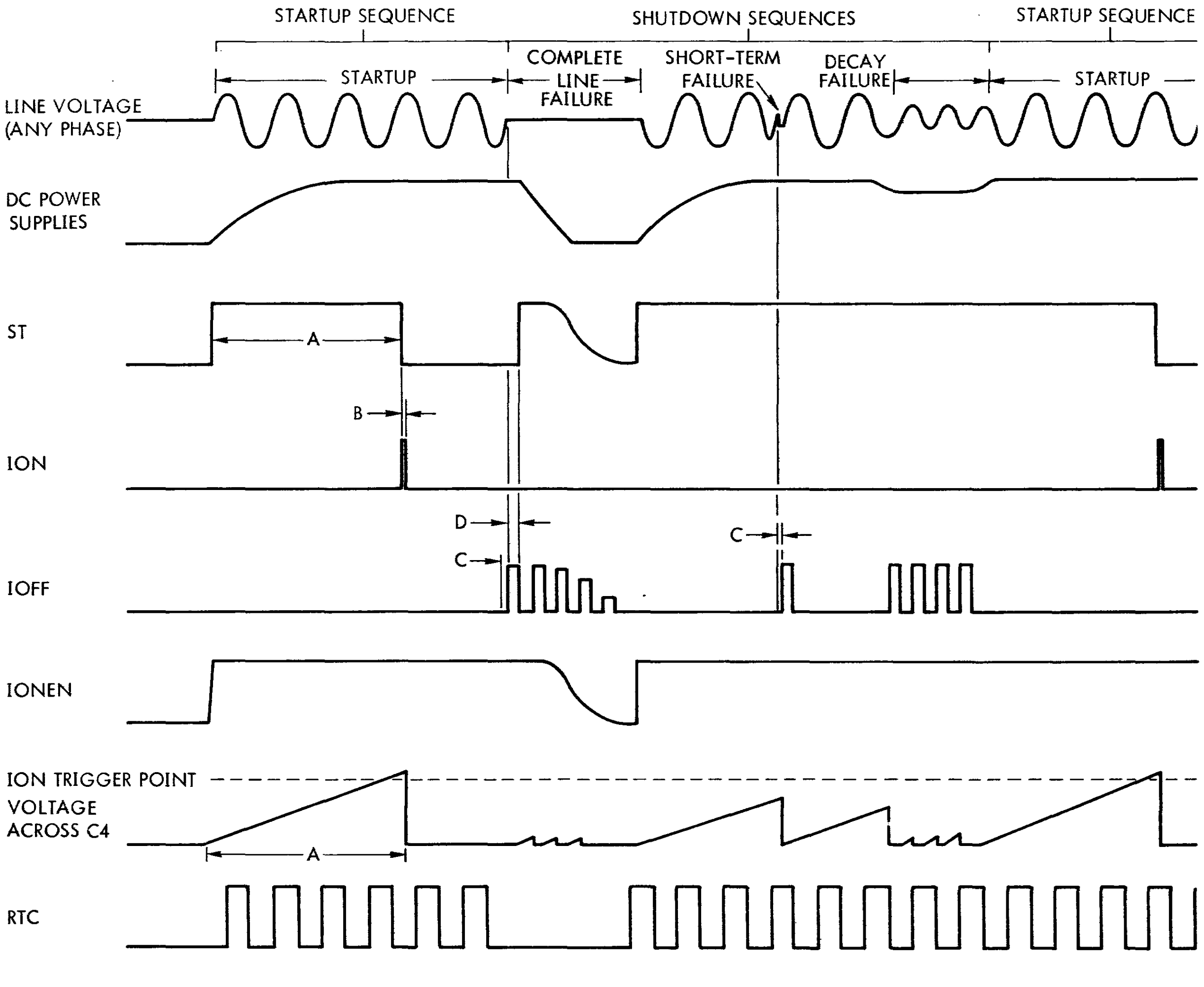
Figure 3-120. WT22 Line Detector, Block Diagram



NOTE: REFERENCE DWG: 132379-1A

Figure 3-121. WT22 Line Detector, Schematic Diagram

901060A. 31104



LEGEND

A. INITIAL TIME FOR DC SUPPLIES TO STABILIZE IS APPROXIMATELY 300 MS
B. 5-100 μS AFTER ST FALLS
C. RESPONSE OF IOFF AFTER COMPLETE LINE FAILURE <1.0 MS
D. TIME BETWEEN IOFF AND ST OR MAXIMUM TIME POWER SUPPLIES REMAIN WITHIN SPECIFICATIONS AFTER A POWER FAILURE IS 5 TO 20 MS

901060A.31105

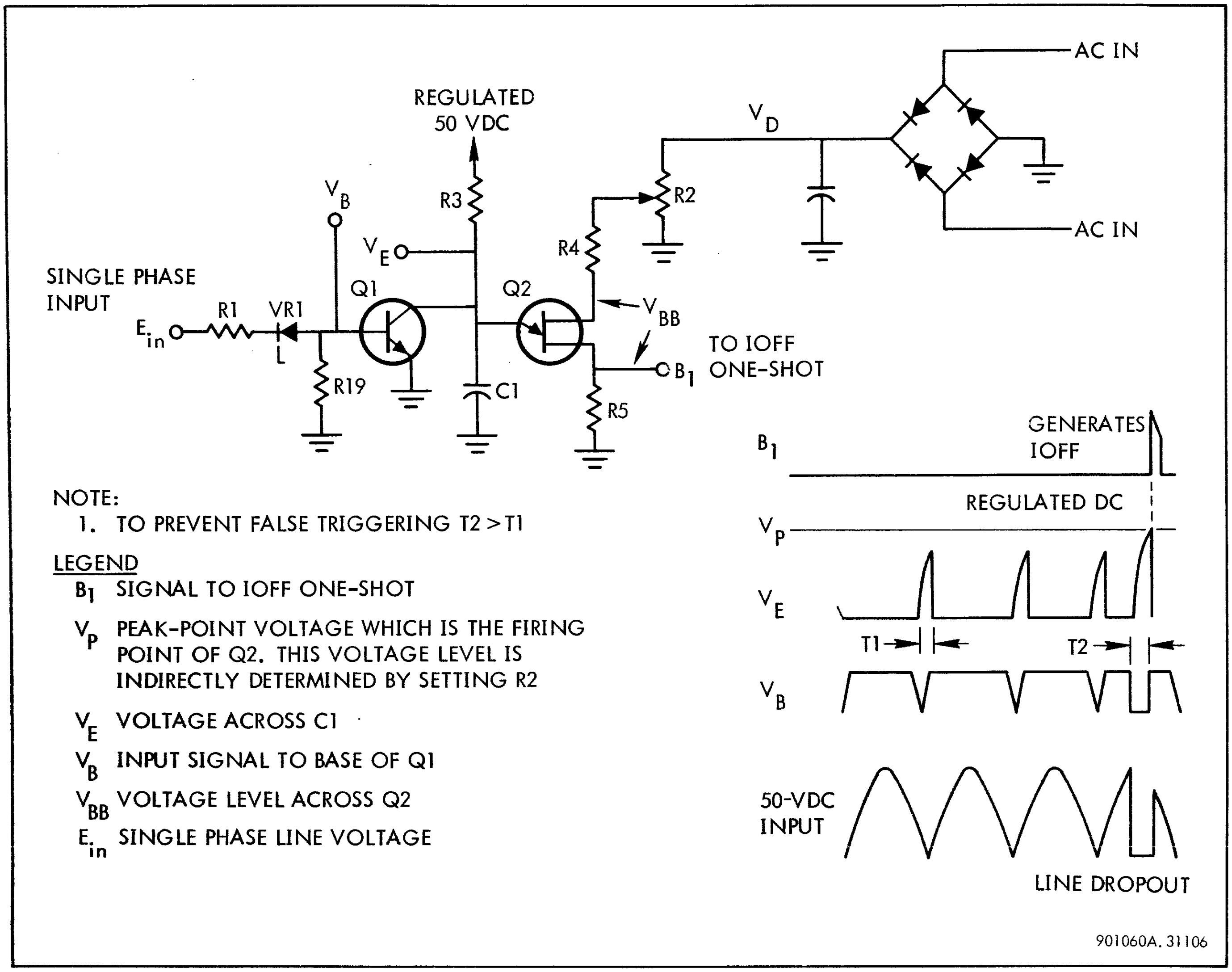
Figure 3-122. Power Fail-Safe Waveforms

In addition to the ST, ION and IOFF signals, the WT22 generates a real-time clock pulse (RTC) synchronized to the line frequency. By selecting the 1F or 2F term on the module, this pulse is at the line frequency or twice the line frequency.

The WT22 is supplied with power by the internal power supply in the power monitor. A schematic of this power supply and its relationship to all the modules in the power monitor assembly is shown in figure 3-118.

The line detection circuit is that part of the WT22 module which detects an ac line failure. The detection scheme is slightly different for single-phase and three-phase operation; however, both phases are detected by the WT22.

Single phase operation is shown in figure 3-123. C1 cannot charge to level V_P if an ac signal is present at E_{in}, the single-phase input. E_{in} is generated by an unfiltered dc signal that is clamped to provide a steep rise at the zero crossing. This rise time determines the minimum response time of the IOFF pulse. C1 must charge to V_p, and V_p is determined by setting potentiometer R2. The time constant of R3C1 is longer than T1 as the voltage charges to V_p. In addition, the base voltage, V_D, is derived from an unregulated source so it decreases with the line voltage, causing V_p to be at a lower point ($V_P = n\ V_{BB}$ where n is 0.7 and V_P is the firing point of the unijunction transistor Q2). If power drops out completely, Q2 fires at less than one-quarter of a cycle, as set by R2.



NOTE:

1. TO PREVENT FALSE TRIGGERING T2 > T1

LEGEND

B_1 SIGNAL TO IOFF ONE-SHOT

V_P PEAK-POINT VOLTAGE WHICH IS THE FIRING POINT OF Q2. THIS VOLTAGE LEVEL IS INDIRECTLY DETERMINED BY SETTING R2

V_E VOLTAGE ACROSS C1

V_B INPUT SIGNAL TO BASE OF Q1

V_{BB} VOLTAGE LEVEL ACROSS Q2

E_{in} SINGLE PHASE LINE VOLTAGE

Figure 3-123. Single-Phase Detection

If power goes down slowly below threshold, Q2 fires at a worst-case condition of one-half cycle caused by the decrease in V_{BB}; however, this is only if power drops slowly below threshold and is not a worst-case condition. This happens because the power supplies take longer to come out of regulation.

For three-phase operation, the threshold is set below the crest of the multiphase signal, as shown in figure 3-124. This unfiltered signal supplies the VBB source voltage in three-phase operation. If any phase falls below threshold the unijunction transistor triggers. Since the voltage is now sampled at six times the line frequency, response time is faster than in single-phase operation. The additional transistor across C1 is not used and is therefore disconnected in three-phase operation by selecting the proper input connections to the WT22 module.

For single-phase operation, pin 39 is connected to pin 5, and 22 Vdc is applied to pins 13 and 45. For three-phase operation, 60-Vdc unfiltered three-phase is applied to pin 1, and pin 11 is connected to pin 45.

The ION one-shot is used to generate the ION pulse. This occurs when there are no IOFF pulses, and the 307 flip-flop is in the reset state with ST low if the line is above the ION threshold. The one-shot shown in figure 3-125 consists of a unijunction transistor threshold circuit, an inverter, and a two-input OR gate. If pulses are applied to D, the emitter voltages never reach V_p on Q6; likewise, if R is positive Q5 conducts and C4 never charges. This provides the inhibit function of the one-shot. C4 only charges if power is on and no IOFF pulses are present. This is the start-up routine. When Q6 fires, it produces a pulse across R15 that is used to generate ION.

The real-time clock puts out pulses at the line frequency or twice the line frequency, as shown in figure 3-126. Q7 derives its interbase voltage V_{BB} from a clamped,

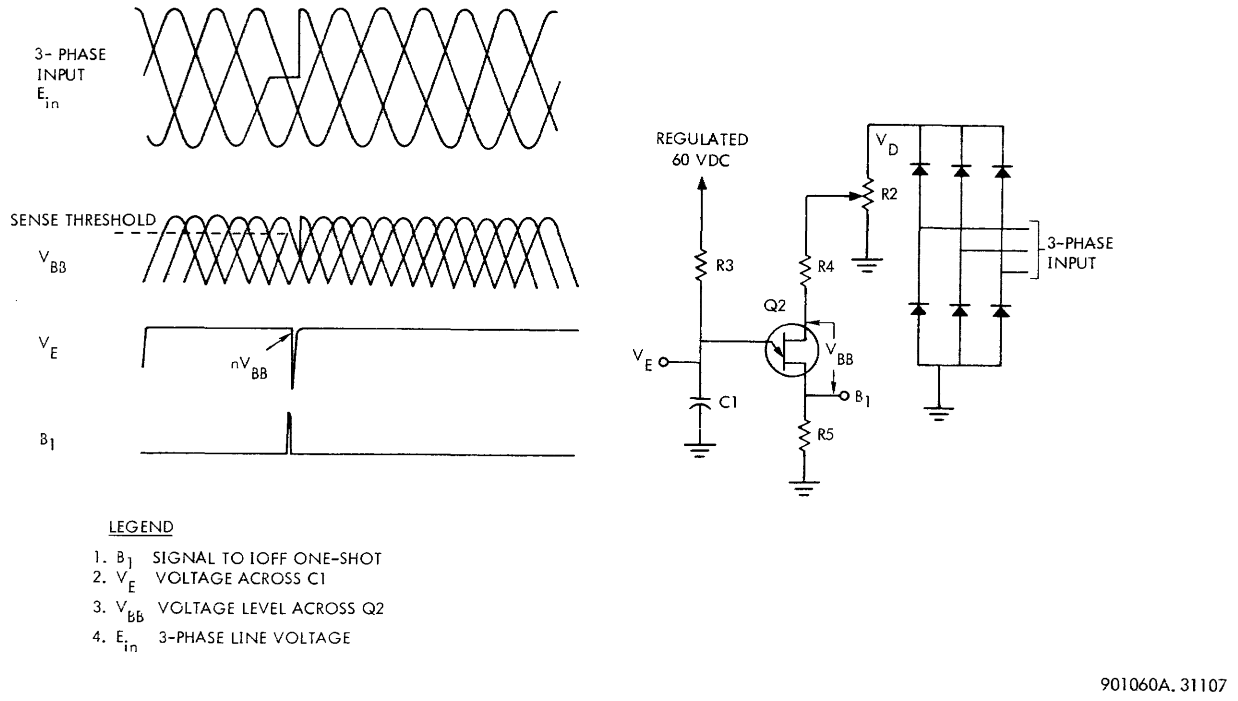


Figure 3-124. Three-Phase Detection

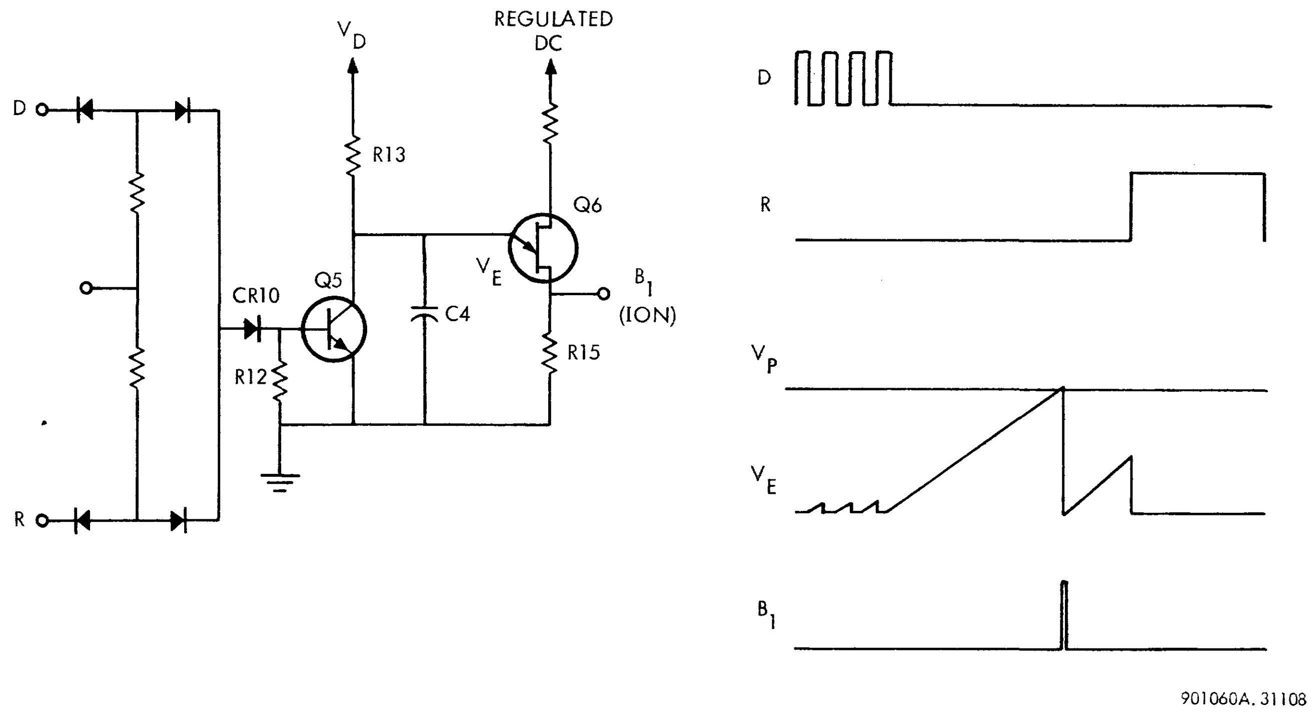
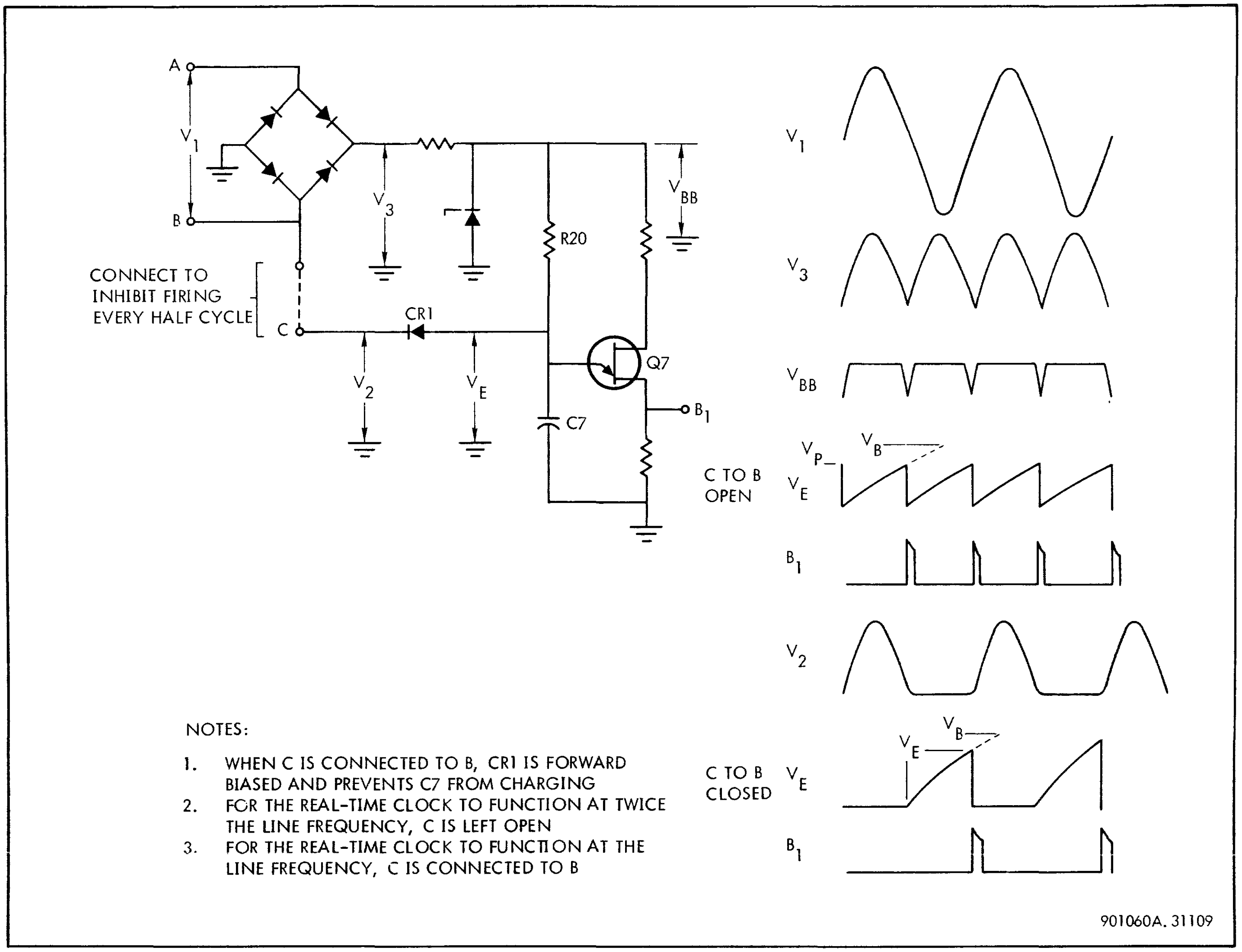


Figure 3-125. ION One-Shot Operation



NOTES:

1. WHEN C IS CONNECTED TO B, CR1 IS FORWARD BIASED AND PREVENTS C7 FROM CHARGING
2. FOR THE REAL-TIME CLOCK TO FUNCTION AT TWICE THE LINE FREQUENCY, C IS LEFT OPEN
3. FOR THE REAL-TIME CLOCK TO FUNCTION AT THE LINE FREQUENCY, C IS CONNECTED TO B

Figure 3-126. Real-Time Clock Operation

high-voltage, unfiltered, dc source. R20C7 is set to be longer than one cycle. As C7 is charged, V_{BB} is suddenly reduced, and Q7 fires when $V_p = n\ V_{BB}$ and produces a pulse at B1. If single frequency pulses are required, Q7 must not fire every half cycle. C7 is prevented from charging by diverting the current through R20 through CR1 every other cycle.

3-107 Power Monitor Logic. There are five power monitor logic signals put out by the AT13 logic module. The signals and their cable pins are as follows:

Signal	Cable Pin
ST	04
RTC	07
IONEN	08
ION	09
IOFF	10

The ION and IOFF signals are input to the LT16 interrupt module as shown in figure 3-127.

a. Start-up Routine. The following steps outline the logic signals that make up the power monitor assembly start-up routine.

1. ST is the master reset signal that is true when power on/off transitions are occurring. When power is applied, this signal comes true as soon as possible (determined by the internal power supply). ST remains high initially as the power supplies in the computer stabilize. This time is determined by ION occurring.

2. ION occurs only if the line voltage is above a preset level, the ION threshold. ION is generated approximately 300 ms after power is turned on. When ION occurs, ST falls to 0. ION should outlast ST by more than 2 μs but less than 100 μs.

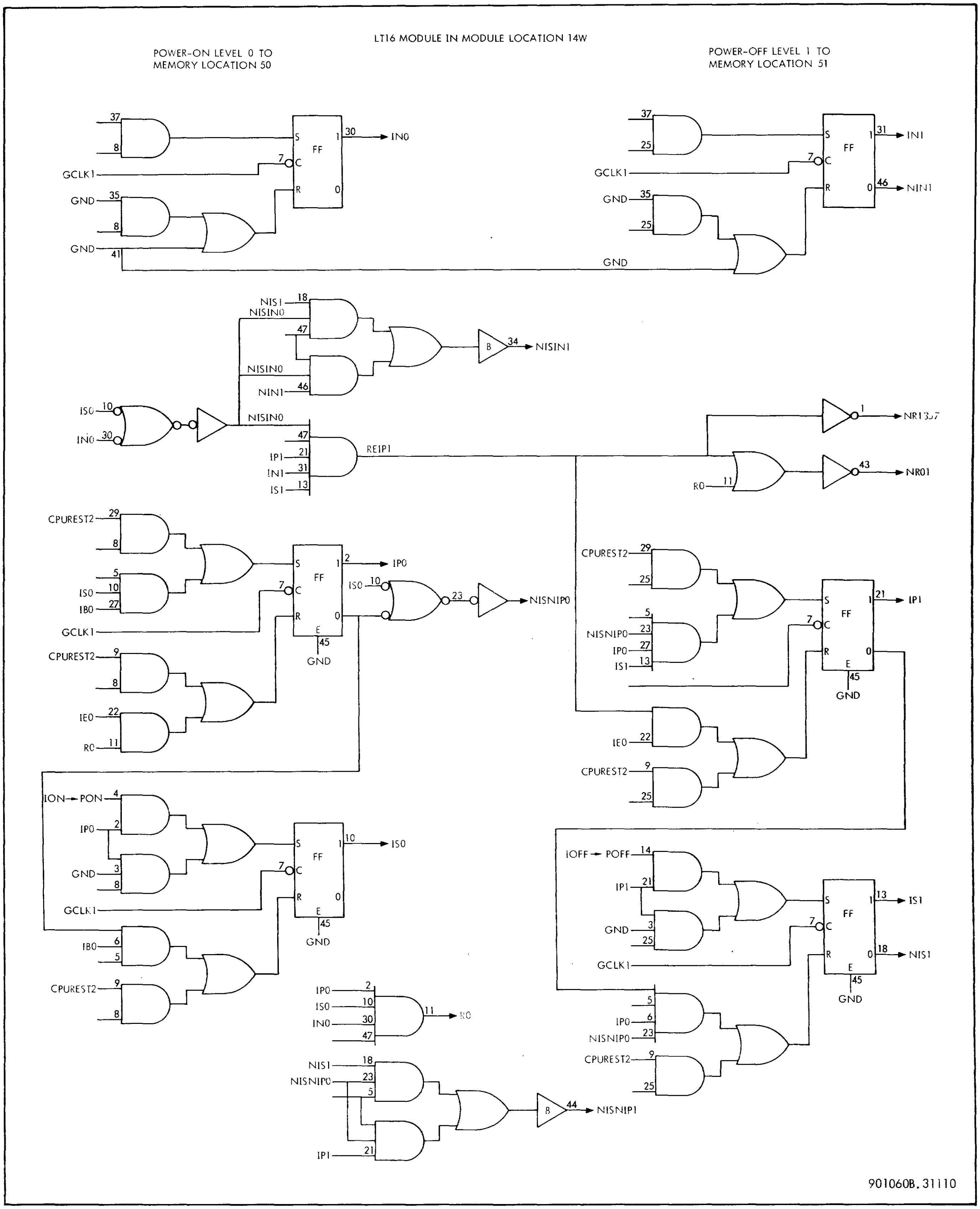


Figure 3-127. Power Fail-Safe, Logic Diagram

3. IONEN is a true signal as long as a power monitor is operating. This signal is necessary only when using more than one power monitor per system. It is available on an AT13 cable driver-receiver where the receivers and drivers are externally connected. This signal from a driver of one power monitor connects to the receiver of another, cascading the signals. IONEN becomes an AND function which is only true if all power monitors are operative.

As shown in figure 3-117, the IONEN switch S1, is left open if only one power monitor is used in a system. If more than one is used, the IONEN switches are closed on all power monitors except the last switch in the cable scheme which is the IONEN switch closest to the cable terminator. It is always left open. With S1 open, the IONEN signal remains high as long as primary power is applied to the power monitor. IONEN is AND-gated with ION to produce the PON signal for the LT16 interrupt module so that the power-on interrupt subroutine can be initiated.

b. Real-Time Clock (RTC). A clock pulse that is jitter-free and synchronized to the line frequency is one of the outputs. This output is arranged so that the interconnecting cables do not parallel one RTC signal with other RTC signals. One of the isolated receiver-drivers on the AT13 is used for this purpose. This precaution is necessary since these RTC signals may be on different phases of the line.

c. Shutdown Routine. The following steps outline the logic signals that make up the power monitor assembly shutdown routine.

1. IOFF is a signal that sets an interrupt channel indicating to the CPU that the line voltage is below a preset threshold. This interrupt initiates a shutdown subroutine that stores all volatile data into core storage before the master reset signal ST causes a cessation of memory operations. The IOFF pulse should be greater than 2 μs, but less than 20 ms. The delay between a power failure and the IOFF pulse going true should be minimized (less than 2 ms for single phase, less than 1 ms for three phase).

2. ST goes true, after a delay time, when power fails. This delay time is determined by the amount of time it takes for the external dc supplies in the computer to fall below their specified tolerances. This delay time or the time between IOFF occurring and ST going true should be adjusted to the maximum possible duration to allow sufficient time to store data before shutting down the memory input. The delay is adjustable between 5 to 20 ms.

3-108 Interrupts

The interrupt system provides a maximum of 237 interrupt levels, of which 13 originate inside the CPU, and 224 are external, originiating in equipment outside the CPU. The 13 internal interrupt levels include seven standard features: two count-pulse interrupts, a memory parity interrupt, two counter-equals-zero interrupts, an input/output interrupt, and a control panel interrupt. Six optional features are also available: the power-on interrupt, the power-off interrupt, two additional count-pulse interrupts, and two additional counter-equals-zero interrupts. The 224 external interrupts are divided into 14 groups of 16 interrupt levels each. Chassis wiring in the CPU divides the internal interrupts into the override group, the counter-equals-zero group, and the input/output group. The override group has priority over other interrupts. As described in the Sigma 7 Computer Reference Manual in the Interrupt System section, the priority sequence of all other groups is optional. Also see the reference manual for a detailed description of each group of interrupts.

3-109 INTERRUPT CONTROL. Interrupt operations are controlled by logic and by programming. Each of the 237 interrupt levels is assigned a unique memory location to which the CPU branches when the interrupt level is acknowledged. The contents of the memory location are transferred to the CPU. The interrupt location must contain one of the following instructions: Modify and Test Byte (MTB), Modify and Test Halfword (MTH), Modify and Test Word (MTW), or Exchange Program Status Doubleword (XPSD). The MTB, MTH, and MTW instructions are single instruction interrupts. The XPSD instruction transfers control of the CPU to a service routine stored in memory. The service routine must end with a Load Program Status Doubleword instruction (LPSD).

3-110 INTERRUPT LEVELS. Each of the 237 interrupt levels includes an interrupt circuit consisting of three flip-flops. The state of the interrupt circuit indicates the status of the interrupt level. An armed circuit is transferred to the waiting state when an event or condition associated with the circuit is detected. For example, the event or condition may be a power failure, a programmed count sequence, or a control panel operation. If a circuit in the waiting state is enabled, it causes an interrupt operation to begin when that interrupt level has priority. Any number of interrupt circuits may be in the waiting and enabled state, but only one may be in the active state at any one time. Priority is established by a combination of signals generated by interrupt circuits and by system cabling.

3-111 INTERRUPT SEQUENCE. Any interrupt except the power-on and power-off interrupts may be individually armed, disarmed, enabled, or disabled by a Write Direct instruction. An armed interrupt level accepts and remembers an input interrupt signal, and remains in the waiting state until enabled. A disarmed interrupt cannot enter the waiting state; that is, it cannot accept an input interrupt signal and is effectively removed from the system. A disabled interrupt may go into the waiting state when the event connected with that interrupt occurs, but interrupt operation cannot begin until the interrupt is enabled.

All interrupts except the override interrupts may be inhibited by groups by means of three flip-flops whose outputs are used in the program status doubleword. If counter inhibit flip-flop CIF which represents bit 37 in the program status doubleword is set, the counter-equals-zero interrupts are inhibited.

If input/output inhibit flip-flop II, representing bit 38, is set, the input/output and control panel interrupts are inhibited. If external interrupt inhibit flip-flop EI, representing bit 39, is set, all of the external interrupts are inhibited. An inhibited interrupt may accept an input and go into the waiting state, but that input may not begin operation until the flip-flop representing its respective program status doubleword bit is reset.

Any interrupt except the power-on and power-off interrupts may be triggered by means of a Write Direct instruction which is described later in this section. When a Write Direct instruction triggers an armed interrupt, the specified interrupt level goes into the waiting state as if its associated event or condition has been detected. The triggering function allows the use of diagnostic programs to test the interrupt system.

3-112 INTERRUPT CIRCUITS. The state of each interrupt level is controlled by an interrupt circuit that generates signals to control the priority of each level. Each interrupt consists of three flip-flops: ISn, IPn, INn, (n = 0, 1, 2, . . . 15). These flip-flops are clocked at a 2.048 MHz rate by signals GCLK1 and GCLK2, which are generated from signal 1MC-2, the output of an amplifier whose input is 2 MHz signal 2MC. Other interrupt circuit flip-flops are also clocked at 2.048 MHz, by signal 1MC-1, which is developed from signal 2MC. The five significant states of an interrupt circuit and the condition established for the level are summarized in table 3-15.

3-113 Power Fail-Safe Interrupts. Interrupt levels 0 and 1 are the power-on and power-off interrupts which are controlled by optional equipment. These circuits have the highest priority level and are always enabled because the input to INn flip-flops are hard wired.

S/IN0 = . . .

R/IN0 = GND

S/IN1 = . . .

R/IN1 = GND

The circuits are placed in the armed state by a reset signal and are normally in the armed and enabled state.

S/IP0 = CPUREST2 + . . .

R/IS0 = CPUREST2 RESET + . . .

S/IP1 = CPUREST2 RESET + . . .

R/IS1 = CPUREST2 RESET + . . .

When power fails, signal IOFF is true, and interrupt level 1 is placed in the waiting and enabled state.

S/IS1 = IP1 POFF

An interrupt sequence takes place, during which the interrupt level is placed in the active state.

Table 3-15. Significant States of Interrupt Circuit

FLIP-FLOPS			STATE
ISn*	IPn*	INn†	
0	0	X	Disarmed. Circuit removed from interrupt system. Interrupt signal neither accepted nor remembered. Change of state only by program intervention
0	1	X	Armed. Can accept and remember interrupt signal. Advances to waiting state when interrupt signal is recognized
1	1	0	Waiting and disabled. Cannot advance to active state. Requires program intervention to be enabled
1	1	1	Waiting and enabled. Can advance to active state if interrupt circuit has highest priority
1	0	1	Active or waiting. The highest priority circuit in this state becomes active when accepted as an interrupt by the CPU. Other circuits in this state wait for acceptance in priority sequence

* n = 0, 1, 2, . . . 15

† Power fail-safe interrupt circuits (0 and 1) are always enabled, so that state of flip-flops is XX1 at all times (INn set)

R/IP1 = REIP1 IE0 + . . .

REIP1 = IN1 IP1 IS1 NISIN0 NISNIP0

NISIN0 = NIS0 + NIN0

NISNIP0 = N(IS0 NIP0) = NIS0 + IP0

IE0 = AIEB ENOVRD

ENOVRD = NENFNL OVRQ

OVRQ = R01 + . . .

R01 = REIP1 + . . .

Signals NISIN0 and NISNIP0 are priority signals that prevent a change of state if the higher priority 0 level is waiting and enabled, or is active. Signal ENOVRD initiates the interrupt sequence. Signal AIEB is a timing signal generated during an interrupt sequence.

After the XPSD instruction in the interrupt location has been executed, the associated service routine is followed, and the interrupt circuit returns to the armed state.

R/IS1 = NISNIP0 NIP1 IB0 + . . .

IB0 = AIB CHA0

S/IP1 = IS1 NISNIP0 IB0

Signal CHA0 is a timing signal generated during the interrupt sequence. Signal AIB is generated from signal LEVACT which is an interrupt exit signal in the service routine:

S/AIB = LEVACT N(. . .)

When power is applied, signal PON is true, and interrupt circuit 0 goes through a similar sequence of operations.

S/IP0 = IS0 IB0 + CPUREST2

R/IP0 = R0 IE0 + . . .

R0 = IN0 IP0 IS0

S/IS0 = IP0 PON

R/IS0 = NIP0 IB0 + CPUREST2

Because interrupt level 0 has the highest priority of all interrupts, no priority signals such as NISNIP0 or NISIN0 are required. Signal ENOVRD is enabled by signal R0, as explained above.

3-114 Count Pulse Interrupts. Interrupt levels 2 through 5 are count pulse interrupts, two of which are standard and two of which are optional. The interrupt circuit feature that distinguishes its function is the input that enables the change of state from armed to waiting (01X to 11X). For the count pulse interrupts, the signals are CPUL1 through CPUL4.

S/IS2 = IP2 CPUL1 + . . .

S/IS3 = IP3 CPUL2 + . . .

S/IS4 = IP4 CPUL3 + . . .

S/IS5 = IP5 CPUL4 + . . .

These signals are generated by real-time counter flip-flops, which are direct reset after the transfer from the armed state (01X) to the waiting state (11X).

E/CPUL1 = IS2

E/CPUL2 = IS3

E/CPUL3 = IS4

E/CPUL4 = IS5

The inputs to interrupt circuit 2 are typical of interrupt circuits 2 through 15.

S/IN2 = DATA16 AEENLE

R/IN2 = DATA16 ADBDB1 + REN

S/IP2 = IS2 NISNIP1 ARMOVD + DATA16
AEADB1

NISNIP1 = NIS1 NISNIP0 + IP1 NISNIP0

ARMOVD = IB0 LEVARM
FAPSD PH7 N07 R30 R31 + . . .

R/IP2 = R2 IE0 + DATA16 DARM1

R2 = IS2 IP2 IN2 NISIN1

NISIN1 = NIS1 NISIN0 NISNIP0 + NIN1
NISIN0 NISNIP0

S/IS2 = IP2 CPUL1 + DATA16 IP2 TRIG1

R/IS2 = NIP2 NISNIP1 IB0 + DATA DARM1

Signals associated with DATA16 are controlled by a Write Direct (WD) instruction; therefore, the circuit can be enabled (placed in state XX1) only during a WD instruction. The circuit is also initially placed in the armed state (010 or 011) by a WD instruction.

Signals NISNIP1, NISNIP0, NISIN0, and NISIN1 are priority signals that prevent changes of state when higher priority circuits are active, or are waiting and enabled. Such signals are generated at all levels which makes it possible for only one interrupt circuit to transfer to the active state (101) at any time. More than one interrupt circuit can be in the active state if a higher priority interrupt circuit goes active during a subroutine for a lower priority interrupt circuit. Several interrupt circuits may be in the waiting and enabled state (111) at one time; however, only the circuit having the highest priority can be transferred to the active state (111 to 101).

Signals IE0, IB0, and ARMOVD are generated during the interrupt sequence for the override interrupts. Corresponding signals for the counter-equals-zero interrupts are IE1, IB1, and ARMCNTR; and IE2, IB2, and ARMIO are corresponding signals for the input/output interrupts.

The normal sequence of operations for an interrupt circuit begins when the circuit is armed (placed in state 01X). When the interrupt input signal is true, the circuit is placed in the waiting state (11X).

S/IS2 = IP2 CPUL1 + . . .

When the circuit is enabled and waiting and has highest priority, it initiates an interrupt sequence and is transferred to the active state (101).

R/IP2 = R2 IE0 + . . .

The instruction stored in the associated memory location is executed while the interrupt circuit is in the active state. After all operations associated with the interrupt have been completed, the interrupt circuit leaves the active state (101 to 011 or 001). The circuit cannot be disabled by an interrupt sequence.

S/IP2 = IS2 NISNIP1 ARMOVD + . . .

R/IS2 = NIP2 NISNIP1 IB0

ARMOVD = IB0 LEVARM

Signal IB0 is always true at the end of the interrupt sequence and causes a transfer from state 101 to state 0X1. If signal LEVARM is true at the end of the active state, the transfer is from state 101 to 011; if LEVARM is false, the transfer is from state 101 to 001.

3-115 Memory Parity Interrupt. Interrupt circuit 6 is transferred from the armed state to the waiting state (01X to 11X) if PEI is true which indicates a core memory parity error has occurred while the CPU was accessing core memory.

S/IS6 = IP6 PEI + . . .

3-116 Counter-Equals-Zero Interrupts. Interrupt circuits 8, 9, 10, and 11 are controlled by interrupt circuits 2, 3, 4, and 5, respectively.

S/IS8 = IP8 SR8 + . . .

SR8 = IJ8 OF

S/IJ8 = R2 IE0

R2 = IP2 IS2 IN2 NISIN1

IE0 = AIEB ENOVRD

S/AIEB = AIEA NAIEB

S/AIEA = NAIEA SAIEA

SAIEA = INTRAP1 INTRAP2 NEWDM NAIEB

S/OF = IB0 CNTZREQ

CNTZREQ = FAS7 PH4 INTRAPF A0031Z

S/IS9 = IP9 SR9 + . . .

SR9 = IJ9 OF

S/IJ9 = SIJ9 IE0

S/IJ9 = NR2 R23

R23 = REIP3 + . . .

REIP3 = IP3 IN3 IS3 NISIN1 NISIN2 NISNIP2

S/IS10 = IP10 SR10 + . . .

SR10 = IJ10 OF

S/IJ10 = R4 IE0

R4 = IP4 IS4 IN4 NISIN3

S/IS11 = IP11 SR11 + . . .

SR11 = IJ11 OF

S/IJ11 = SIJ11 IE0

SIJ11 = NR4 R45

R45 = REIP5 + . . .

REIP5 = IP5 IN5 IS5 NISIN3 NISIN4 NISNIP4

When one of the count pulse interrupt circuits is in the active state (101), and the count has been reduced to zero (A0031Z), one of the counter-equals-zero interrupt circuits is placed in the waiting state (01X to 11X).

3-117 Input/Output Interrupt. Interrupt circuit 12 is placed in the waiting state (11X) by an IOP interrupt request signal.

S/IS12 = IP12 IR + . . .

3-118 Control Panel Interrupt. Interrupt circuit 13 is placed in the waiting state (11X) by control panel switch signal KINTRP interlocked with a flip-flop.

S/IS13 = IP13 SR13 + . . .

SR13 = KINTRP NCNLK

S/CNLK = IS13

R/CNLK = NKINTRP/B + . . .

3-119 PRIORITY SIGNALS. Signals generated by interrupt circuits are interconnected in order to control priority of interrupt levels. Interrupt levels 0 through 7 have the highest priority. Counter-equals-zero interrupts, input/output interrupts, and external interrupts in groups of 16 may be connected in any priority sequence at the option of the user.

The priority signals permit only one interrupt circuit in the waiting and enabled state (111) to be transferred to the active or waiting state (101) on a particular interrupt clock. However, more than one interrupt circuit may be in the active or waiting state at a given time; for example, if a high-priority interrupt circuit is transferred to the active or waiting state while a low-priority interrupt circuit is active, the high-priority circuit overrides the low-priority circuit. While the high-priority circuit is active, the low-priority circuit which was previously active is dormant until the high-priority circuit has completed its operation. The low-priority circuit, having remained in the active or waiting state (101), then resumes operation.

Priority signals are associated with each interrupt circuit. Typical signals for even-numbered circuits are:

R10 = IN10 IP10 IS10 NISIN9

NISIN9 = NIS9 NISIN8 NISNIP8 + . . .

NISNIP8 = NIS8 + IP8

Thus, R10 can be true only if interrupt circuit 10 is waiting and enabled and if no high priority interrupt in that group is waiting or active. Signal NISIN10 is true only if circuit 10 is not active, and not waiting and enabled: NISNIP10 is true only if circuit 10 is not active.

Typical signals for odd-numbered circuits are:

REIP11 = IN11 IP11 IS11 NISIN10 NISNIP10 NISIN9

NISIN10 = NIS10 + NIN10

NISNIP10 = NIS10 + IP10

Thus, REIP11 can be true only if interrupt circuit 11 is waiting and enabled and if no higher priority interrupt in that group is waiting or active. Signal NISIN10 can be true only if interrupt circuit 10 is not active, and not waiting and enabled and if no higher priority interrupt in that group is waiting or active.

Signals NISIN9 and NISNIP9 are controlled by all interrupt circuits from 0 through 9. Similar signals are generated at all levels of interrupts.

3-120 Priority Chain Signals. An interrupt sequence is initiated after signal INT9 is true, enabling flip-flop INT to be set. Signal INT9 is controlled by inputs from all interrupt circuits, including external interrupts.

/INT09/ = TNIREQ

TNIREQ = ENOVRD OVRQ
+ ENCNTR CNRQ
+ ENIO IORQ + LINREQ NEWDM

Signal ENOVRD is true if any of the first eight interrupt circuits is waiting and enabled.

ENOVRD = OVRQ NENFNL

OVRQ = R01 + R23 + R45 + R67

R01 = IP0 IS0 IN0 (typical)

More than one interrupt circuit may be waiting and enabled; however, only one can generate a true Rx signal or REIPy signal.

Signal ENCNTR is true if any of interrupt circuits 8 through 11 is waiting and enabled (R89 + R1011), provided that the group is not inhibited by the program status doubleword (NCIF), and that no higher priority group is waiting and enabled.

ENCNTR = NCIF NHRQBZ1 NENFNL CNRQ

HRQBZ1 = /RQY1/

CNRQ = R89 + R1011

Signal RQY1 goes outside the CPU to provide for the option of external interrupts with higher priority than the counter-equals-zero interrupt group or the input/output interrupt group.

Signal ENIO is true if any of the interrupt circuits 12 through 15 is waiting and enabled (R1213 + R1415), provided that the group is not inhibited by the program status doubleword (NII), and that no higher priority group is waiting and enabled.

ENIO = NII NENFNL IORQ

IORQ = R1213 + R1415

Signal LINREQ is generated in external equipment when an external interrupt is waiting and enabled, and starts an interrupt sequence if no Write Direct instruction in the interrupt mode (0001) is active. The NEWDM term is required because /DATm/ lines are shared between the trigger arm, enable data on output during a Write Direct instruction, and the memory address data on input during interrupt operations.

INT09 = TNIREQ

TNIREQ = LINREQ NEWDM + . . .

LINREQ = /DATA25/ NEI

EWDM = B1619ONE WD

3-121 Group Control. Priority signals generated by the interrupt circuits also control signals which cause changes of state in the interrupt circuits during an interrupt sequence. These signals permit only one group of interrupts to be controlled at any time. The family of IEx signals causes a change of state from waiting and enabled (111) to active (101).

IE0 = AIEB ENOVRD NEWDM

IE1 = AIEB ENCNTR ENEXSTR

IE2 = AIEB ENIO ENEXSTR

The family of IBx signals remove an interrupt circuit from the active state.

IB0 = AIB CHA0

IB1 = AIB CHA1

CHA0 = CHA0 RCCHA0 NCPURESET
+ IE0 1MC

RCCHA0 = RCCHA NISNIP7

CHA1 = CHA1 RCCHA1 NCPURESET + IE1

RCCHA1 = RCCHA NISNIP11

CHA2 = CHA2 RCCHA2 NCPURESET + IE2

RCCHA2 = RCCHA NISNIP15

RCCHA = GARF + . . .

S/GARF = AIB

The family of ARMx signals causes a change of state from active to armed (011).

ARMOVD = IB0 LEVARM

ARMCTR = IB1 LEVARM

ARMIO = IB2 LEVARM

Signals AIB and LEVARM are generated during an interrupt sequence. If the LEVARM signal is false, the interrupt circuit is left in the disarmed state (001) after transfer from the active state.

3-122 Memory Address Control. Signals INT00 through INT08 retain a code which addresses the memory location associated with an interrupt level. This code, which is transferred to the P-register during the interrupt sequence, is established by priority signals generated in the interrupt circuits.

Signals INT0, INT1, and INT3 are false for any internal interrupt level; signals INT2 and INT4 are true for any internal interrupt level.

INT2 = TNI02 = ENOVRD OVRQ + ENCNTR CNRQ + ENIO IORQ + . . .

INT4 = TNI04 = ENOVRD OVRQ + ENCNTR CNRQ + ENIO IORQ + . . .

Signals INT00 through INT08 hold the code 0 0101 XXXX for any internal interrupt level. Signals INT05 through INT08 hold a code that is dependent upon the internal interrupt level enabled.

INT05 = TNI05 = ENCNTR CNRQ + ENIO IORQ + . . .

INT06 = TNI06 = OVLN6 ENOVRD + ENIO IORQ + . . .

OVLN6 = R45 + R67

INT07 = TNI07 = OVLN7 ENOVRD + CNLN7 ENCNTR + IOLN7 EN10 + . . .

OVLN7 = R23 + R67

CNLN7 = R1011

IOLN7 = R1415

INT08 = OVLN8 ENOVRD + CNLN8 ENCNTR + IOLN8 ENIO + . . .

OVLN8 = R1357

CNLN8 = R911

IOLN8 = R1315

Although more than one circuit may be waiting and enabled for a time, only one of the Rx signals associated with the internal interrupt circuits can be true at any time. The last four bits of the code held by signals INT00 through INT08 are any of 0000 through 1111, depending upon the interrupt circuit controlling the interrupt. Therefore, the address code for an internal interrupt is any value between 0 0101 0000 and 0 0101 1111 (hexadecimal 050 through 05F).

The logic diagram in figure 3-128 shows how a specific interrupt level generates address data to a specific memory location. The counter 1 zero interrupt is used for this example.

When flip-flop outputs IN8, IP8, and IS8 are true, R8 is gated on. When R8 is true, the term NR89 goes false. NR89 which is one of the input terms to the CNRQ buffer-inverter causes CNRQ output to go true. CNRQ generates counter group service request signal which is fed into the TNI buffer gates. These buffers are the interrupt service routine address lines to the CPU, and because both CNRQ and the enable counter group signal ENCNTR are true, the input to the TNI buffers in this example has a binary equivalent of 0101 1000. This is equivalent to X'58', the memory location to be addressed.

The TNI buffer gating is input to the /INT00/ through /INT09/ cable drivers that drive the data into the CPU. Cable receivers INT0 to INT9 receive the data. The terms INT0 through INT8 are gated with the term PXINT and are clocked into P-register flip-flops P23 through P31.

Signal INT9 generates interrupt request INT. When the request is received and acknowledged by the CPU, signal IE1 is generated, flip-flop IP8 is reset, and the interrupt sequence in the CPU is started. A detailed sequence chart is given later in this section.

3-123 SERVICE ROUTINE SEQUENCE. Figure 3-129 is a timing diagram for an interrupt operation which transfers control to a stored service routine.

When an interrupt circuit is waiting and enabled, INT is set by the interrupt service request signal, INT9.

S/INT = INT9

The three interrupt flip-flops are then set during a program instruction.

S/INTRAPF = (S/INTRAPF) NRESET

(S/INTRAPF) = INT (S/INTRAPF/1) (IEN + ENDE) + . . .

(S/INTRAPF1) = NINTRAPF NINTEN KRUN/B INT

Flip-flop IEN is set during phase 1, 2, 8, 9 or 10 of an instruction, depending on the type of instruction.

S/INTRAP1 = (S/INTRAPF) NRESET

S/INTRAP2 = INTRAP1 NRESET

Signal CEINT is generated and INTRAP2 is reset.

CEINT = INTRAP1 INTRAP2 NTRAP + . . .

R/INTRAP2 = INTRAP2

The CPU clock is inhibited until signal ARE is true, and signal PXINT is true enabling transfer of address data to the P-register.

CLEN = NCEINT + CEINT ARE + . . .

ARE = 1MC AIEB + . . .

S/AIEB = AIEA NAIEB

S/AIEA = SAIEA NAIEA

SAIEA = INTRAP1 INTRAP2 NEWDM NAIEB

PXINT = INTRAP1 NINTRAP2 NTRAP

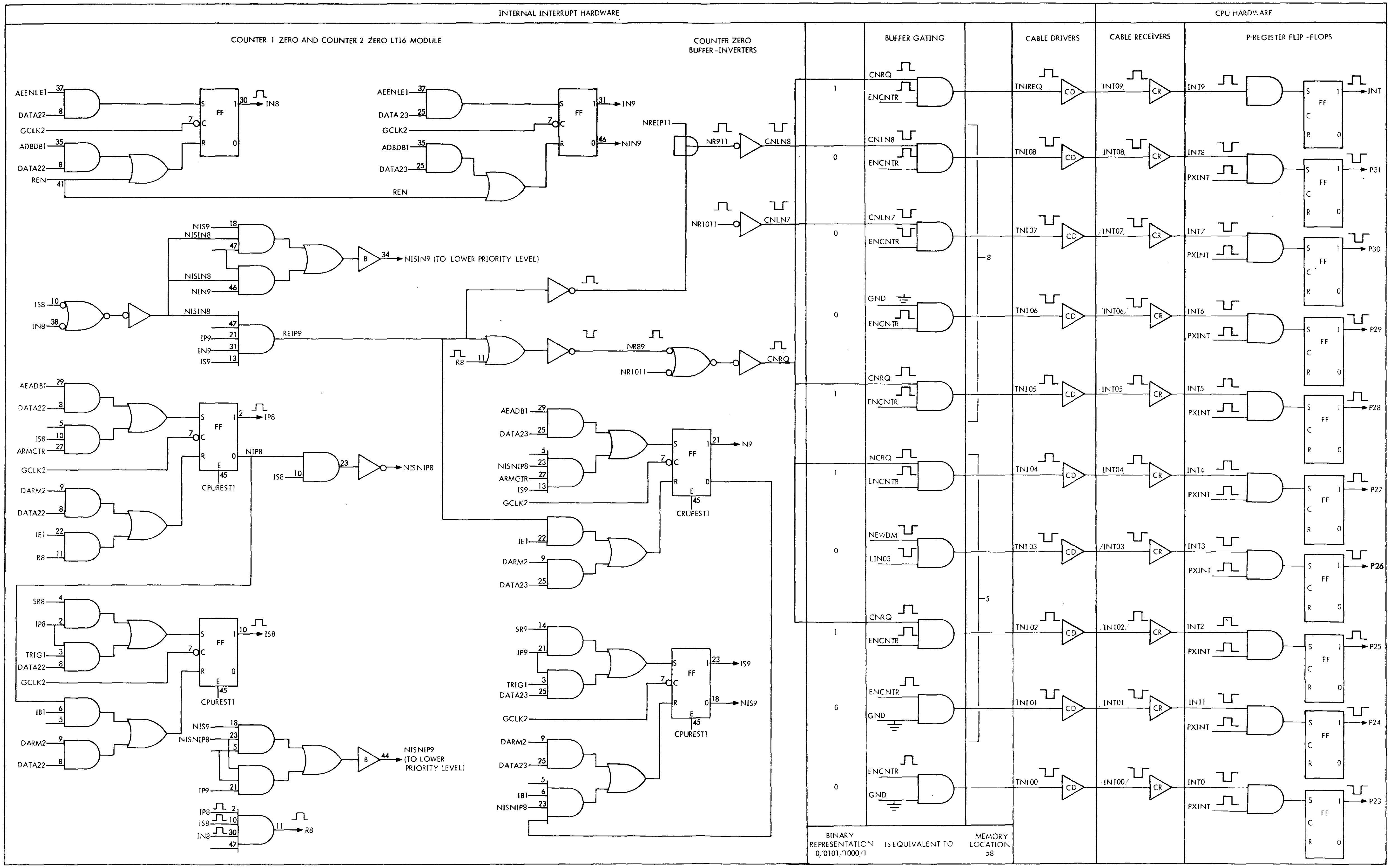
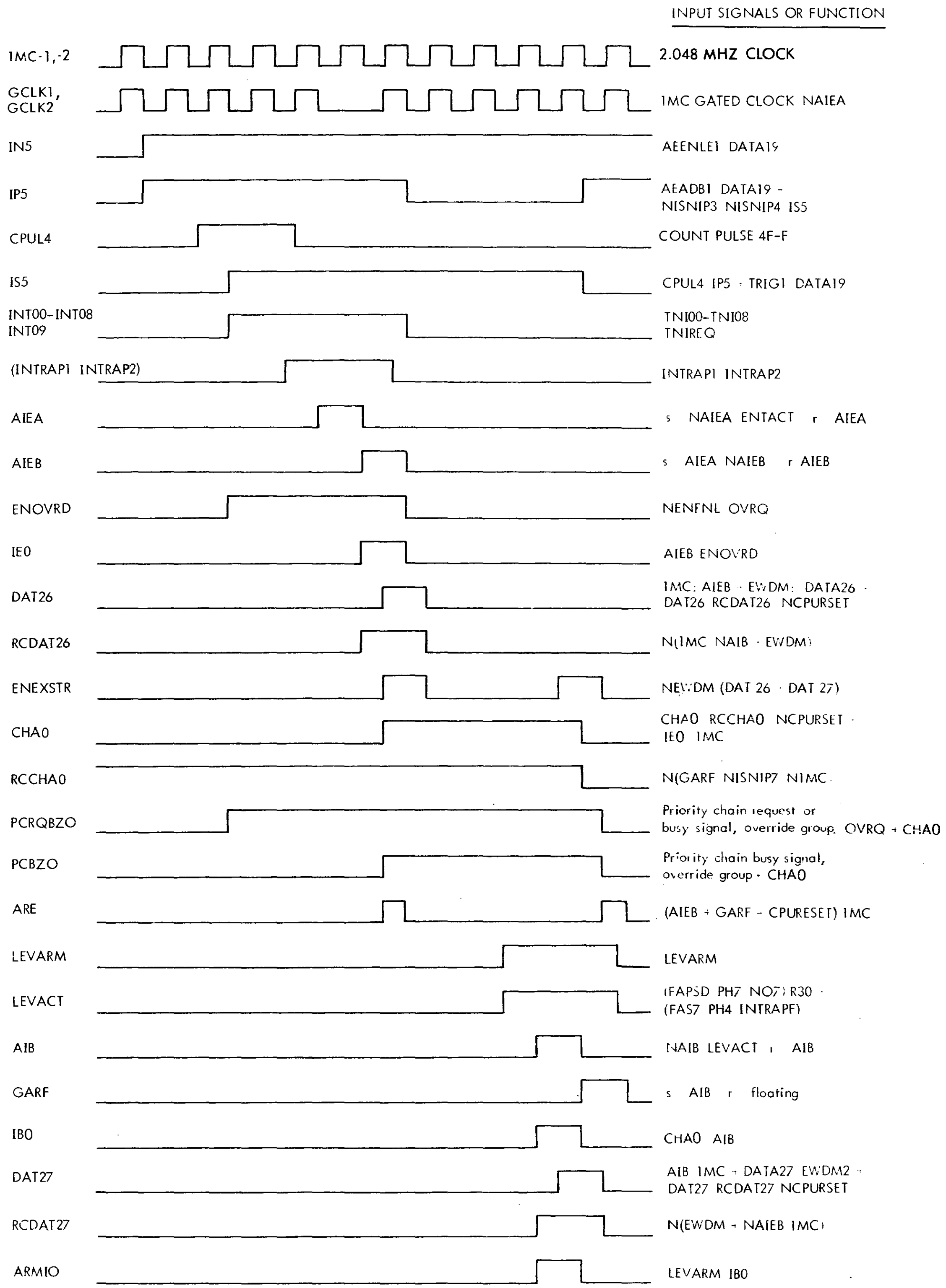


Figure 3-128. Interrupt Memory Location Addressing

901060B.3824



NOTE: THE SIGNALS SHOWN IN THIS DIAGRAM ILLUSTRATE THE TIMING FOR INTERNAL INTERRUPTS BEFORE CPU PROCESSING

901060A.3806A

Figure 3-129. Interrupt Timing Diagram (Counter 4 Count Pulse)

Signal CLEN is a clock enable signal.

Signal ARE is controlled by the 2-MHz clock through signal 1MC-1 and prevents changes of state in the interrupt circuit by inhibiting gated clock signal GCLK.

GCLK1,2 = IMC NAIEA

While signal AIEB is true, a true IE0, IE1, or IE2 signal causes the interrupt circuit to transfer from waiting and enabled (111) to active (101).

R/IPn = Rn IEx + . . . (typical)

IE0 = AIEB ENOVRD NEWDM

IE1 = AIEB ENCNTR ENEXSTR

IE2 = AIEB ENIO ENEXSTR

Signal DATA26 is generated to enable the change of state in the external circuit, if the interrupt logic is servicing an external interrupt.

DAT26 = AIEB 1MC + . . .

Signal DAT26 becomes signal DATA26 in the external interrupt circuits. This signal is used to enable the change of state of the external interrupts from waiting and enabled to active, as explained in paragraph 3-125, External Interrupts.

After signal ARE is true, a CPU clock is generated which resets INTRAP1, sets INTRAP2, and drops CEINT.

R/INTRAP1 = INTRAP2 + . . .

S/INTRAP2 = INTRAP1 NRESET

At this time, the XPSD instruction in the assigned memory location is extracted from memory and is executed. The service routine addressed by the XPSD instruction may be interrupted. A service routine that is not interrupted is terminated by an LPSD instruction which generates CEINT during phase 7 and causes signal LEVACT to be true.

CEINT = FAPSD PH7 N07 R30 + . . .

LEVACT = FAPSD PH7 N07 R30 + . . .

The CPU clock is inhibited until signal ARE is true.

CE = NCEINT + CEINT ARE + . . . N (. . .)

ARE = 1MC AIEB (. . .)

When the interrupt service routine ends, the interrupt circuit that initiated the operation is transferred from the active state (101) to either the disarmed state (001) or the armed state (011). Bit position 10 of the LPSD instruction must contain a one which causes R30 to be set, and enables LEVACT to be true for any change of state.

R/ISn = NIPn NISNIPy IB0 + . . . (typical)

IB0 = AIB CHA0

If bit position 11 of the LPSD instruction contains a one, R31 is set, LEVARM is true, and the change of state is from active to armed (101 to 011).

S/IPn = ISn NISNIPy ARMOVD + . . . (typical)

ARMOVD = IB0 LEVARM

LEVARM = FAPSD PH7 O7 R30 R31

If bit position 11 of the LPSD instruction contains a zero, R31 is not set, and the change of state is from active to disarmed (101 to 001).

If the interrupt logic is servicing an external interrupt, signals DAT27 and DAT28 are generated to enable changes of state in the external circuit.

DAT27 = AIB 1MC + . . .

DAT28 = LEVARM NEWDM + . . .

These signals become signals DATA27 and DATA28 in the external interrupt circuits. Signal DATA27 enables removal of an external interrupt from the waiting or active state, and signal DATA28 is used to place the interrupt in the armed state. This logic is explained in paragraph 3-125.

3-124 MODIFY AND TEST SEQUENCE. The sequence of operations for an interrupt that transfers control to a Modify and Test instruction is similar to the sequence that transfers control to an XPSD instruction and the associated service routine. When the interrupt circuit is waiting and enabled and has priority, INT is set, and the interrupt operations follow the end phase of the instruction or the interruptible point in an instruction as determined by flip-flop IEN. After INTRAP, INTRAP1, and INTRAP2 are set, the CPU clock is inhibited, the interrupt circuit is placed in the active state, and the contents of the memory location are extracted and executed, as described for the service routine sequence (paragraph 3-123).

During a modify and test sequence of a counter interrupt, count-equals-zero A0031Z is sampled, and a count-equals-zero interrupt circuit may be placed in the waiting state (11X) if the effective location of the Modify and Test instruction contains all zeros when A0031Z indicates that the modified word in the effective location equals zero.

S/IS8 = IP8 SR8 + . . . (typical)

SR8 = IJ8 OF

S/IJ8 = R2 IE0

R2 = IP2 IS2 IN2 NISIN1 (count pulse interrupt)

S/OF = IB0 CNTZREQ

CNTZREQ = FAS7 PH4 INTRAPF A0031Z

The interrupt circuit is always transferred from the active state (101) to the armed state (011) because LEVARM is always true.

R/ISn = NIPn NISNIPy IB0 + . . . (typical)
IB0 = AIB CHA0
AIB = LEVACTN (. . .)
LEVACT = FAS7 PH4 INTRAPF + . . .
S/IPn = ISn NISNIPy ARMOVD + . . .
ARMOVD = IB0 LEVARM
LEVARM = FAS7 PH4 INTRAPF + . . .

After the Modify and Test instruction has been extracted from memory, it inhibits and enables the CPU clock by controlling CEINT

CEINT = FAS7 PH4 INTRAPF + . . .

and terminates the sequence by resetting INTRAPF.

R/INTRAP = FAS7 PH4 INTRAPF + . . .

The modify and test sequence is controlled by the Modify and Test Word instruction. The address of the next instruction in sequence is stored during PREP phases; therefore, a modify and test sequence is a single-instruction interrupt.

3-125 EXTERNAL INTERRUPTS. External interrupts control an interrupt circuit containing three flip-flops. The priority of an external interrupt group depends upon the cable connection with the CPU. The priority of the individual interrupts within a group is established in the same way as in the internal interrupt circuits.

When an external interrupt is waiting and enabled and has priority, it generates a true INT9 signal, and causes INT to be set, like an internal interrupt. The true INT9 signal is generated by a DAT25 signal when no WD instruction in the interrupt mode is active where /DAT25/ is the interrupt signal from the external interrupt chassis.

S/INT = INT9 = /INT09/
INT9 = LINREQ NEWDM + . . .
/INT09/ = TNIREQ
TNIREQ = LINREQ NEWDM
LINREQ = LINREQ NEWDM
LINREQ = /DAT25/
EWDM = NB16, NB17, NB18, B19 DIOWD

The interrupt address is transmitted from the external to the internal interrupt circuits over lines /DAT16/ through /DAT24/.

INT0 = /INT00/ = TNI00 = LIN00 NEWDM (internal)
⋮ ⋮
INT8 = /INT08/ = TNI08 = LIN08 NEWDM
LIN00 = /DAT16/ (external)
⋮ ⋮
LIN08 = /DAT24/

The change of state of an external interrupt circuit from waiting and enabled (111) to active (101) is controlled by DAT26, originating in the internal interrupt circuits, when ENEXSTR is an interrupt enable signal generated in the internal circuits.

R/IPn = REIPn EIE (external)
EIE = DATA26 ENEXSTR LINREQ
DATA26 = /DAT26/ (from internal circuits)

Signal DAT27 from the internal interrupt circuits controls removing an external interrupt from the active state.

R/ISn = EIB (. . .) (external)
EIB = DATA27 ENEXSTR (external)
DATA27 = /DAT27/ (from internal circuits)

Signal DAT28 from the internal interrupt circuits controls placing the interrupt in the armed state on removal from the active state.

S/IPn = ISn LEVARM (. . .) (external)
LEVARM = ENEXSTR DATA28 EIB (external)
DATA28 = /DAT28/ (from internal circuits)

All external interrupts are inhibited if flip-flop EI, part of the program status doubleword, is set.

/ERQY/ = PCRQBZE (external interrupt request)
PCRQBZE = DATA29 LINREQ NEWDM ECHA + . . .
DATA29 = /DAT29/ (from internal circuits)
/DAT29/ = NEWDM NEI + . . . (internal)

3-126 WRITE DIRECT INTERRUPT OPERATION. A Write Direct (WD) instruction can control the interrupt operation in two ways. When operating in the internal mode, it can control the states of flip-flops CIF, EI, and II, which can inhibit the priority chain signals. A WD enables signals DAT16 through DAT31, which select individual interrupt circuits, when operating in the interrupt mode. These inputs have the following general form, y = x + 14 for x = 2, 3. . . 15.

S/INx = DATAy AEENLE1 + . . .
R/INx = DATAy ADBDB1 + REN + . . .

S/IPx	=	DATAy AEADB1 + . . .
R/IPx	=	DATAy DARM1 + . . .
S/ISx	=	DATAy TRIG1 IPx + . . .
R/ISx	=	DATAy DARM1 + . . .
DATAy	=	/DIOy/ = Sy DIOXS
DIOXS	=	OLD OU6 SW3 NRZ (Write Direct)

These signals change the state of interrupt circuits when a WD instruction in the interrupt control mode, bits 16 through 19, presents a code, bits 21 through 23, addressed to any group, bits 28 through 31. The details of this operation are explained in the following paragraphs.

When a WD instruction in the interrupt control mode is executed, signal EWDM is true.

EWDM	=	B1619ONE WD
WD	=	/DIO50/ = DIO50X1 = OLD OU6 SW3

During phase 3 of the WD instruction, /DIO48/ is true and CNA is set.

S/CNA	=	NCNA NCNB SCNA
SCNA	=	FS EWDM
FS	=	/DIO48/
/DIO48/	=	DIO48X1 = FARWD SW4

Flip-flop CNB is set on the next 1-MHz clock following the setting of flip-flop CNA, and flip-flop CNA is reset by signal CNB:

S/CNB	=	CNA NCNB
R/CNA	=	CNB + . . .

A function strobe acknowledge signal is generated in the interrupt circuits and is sent to the CPU on line /DIO49/.

FSA	=	NCNA CNB
/DIO49/	=	FSA

These signals are generated in the internal interrupt circuits for use during both internal and external interrupt operation. Figure 3-130 is a timing diagram of these signals.

Changes of state in the interrupt circuits may take place only when signal CNA is true. In the internal interrupt circuits, this is ensured by enabling the group 0 signal with CNA:

GRP0	=	CNA NADDR12 NADDR13 NADDR14 NADDR15

where ADDR12 through ADDR15 represent the group code in B28 through B31. The external interrupt operation is controlled by using signal CNA to generate function line enable signal ENFNL.

ENFNL	=	CNA 1MC + . . .
FNL00 - FNL02	=	(ADDR05 - ADDR07) ENFNL

The function lines to the external interrupt circuits can be active only when signal CNA is true.

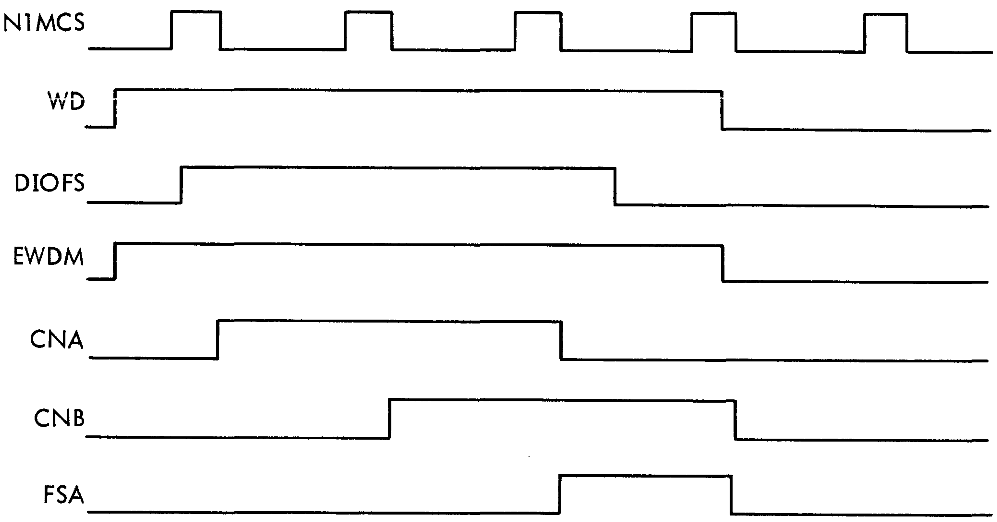


Figure 3-130. Write Direct Sequence, Timing Diagram

The next phase of the Write Direct instruction is entered, when the CPU receives signal FSA from the interrupt circuits.

The internal interrupts are selected when the Write Direct instruction selects group 0 by zeros in bits 28 through 31 of the instruction word. Signals DAT16 through DAT29, generated from the contents of the private memory register by way of the DIO lines, select interrupt circuits 2 through 15. The first two internal interrupt circuits are the power fail-safe interrupts and are not controlled by a Write Direct instruction.

An external interrupt group has been chosen if any interrupt group other than group 0 is selected, and signals DAT16 through DAT31 determine which of the 16 interrupts in the group are to be controlled.

The code stored in bits 21 through 23 of the instruction word causes changes of state in the interrupt circuits as shown in table 3-16. Signals generated from this code in the B-register are as follows:

ADDR05 - ADDR07 = /DIO37/ - /DIO39/
= B21 - B23

For the external interrupts, signals ADDR05 through ADDR07 become signals FNL00 through FNL02, which are used for function decoding. The control signals generated from the code are DARM1, AEADB1, AEENLE1, ADBDB1, REN, and TRIG1. A logic diagram of these signals is shown in the interrupt control mode part of the Write Direct instruction description. The signals are used to change the states of the IS, IP, and IN flip-flops as previously described in this discussion. The changes of state resulting from these signals are summarized in table 3-17.

3-127 CPU INTERRUPT SEQUENCE. An interrupt can occur during the ENDE or IEN phase of an instruction. Since most instructions have a short execution time, most interrupts occur before or after an instruction execution of ENDE time. However, during multiple operand instructions, iterative sequences have a longer execution period because the control and intermediate results of these instructions reside in registers and memory. Since it may be necessary to interrupt during multiple operand instructions, the interrupt enable flip-flop, IEN, is set between instruction iterations.

When IEN is set, an interrupt can be accepted on any clock. If an interrupt request is received and acknowledged, INT is set, and the term (S/INTRAPF) initiates the interrupt sequence. The CLEAR signal is generated during ENDE time or when (S/INTRAPF) is true. Flip-flop INTRAP1, the interrupt phase flip-flop, is set during ENDE or IEN in preparation for Pass 1 and Pass 2 of the INTRAP1 phasing.

If IEN is true and ENDE is false, BRQ is set. BRQ is a repeater flip-flop which causes the contents of Q to transfer to P. BRQ insures that the address stored by the XPSD instruction points at the right instruction in the iteration chain for processing after the interrupt is completed.

Table 3-16. Function of Codes for WD Interrupt Control Mode

CODE			OPERATION
B21	B22	B23	
0	0	0	Undefined
0	0	1	Disarm all levels selected by a one; all levels selected by a zero are not affected
0	1	0	Arm and enable all levels selected by a one; all levels selected by a zero are not affected
0	1	1	Arm and disable all levels selected by a one; all levels selected by a zero are not affected
1	0	0	Enable all levels selected by a one; all levels selected by a zero are not affected
1	0	1	Disable all levels selected by a one; all levels selected by a zero are not affected
1	1	0	Enable all levels selected by a one and disable all levels selected by a zero
1	1	1	Trigger all levels selected by a one. All such levels that are currently armed advance to the waiting state. Those levels currently disarmed are not altered, and all levels selected by a zero are not affected

Three interrupt phases are interposed between the interrupt acknowledgment at ENDE or IEN and preparation phase 1 of the following Program Status Doubleword instruction. These three phases are referenced in the following paragraphs as Pass 1, Pass 2 and Pass 3, where Pass 1 = INTRAP1 NINTRAP2, Pass 2 = INTRAP1 INTRAP2, and Pass 3 = NINTRAP1 INTRAP2.

a. Pass 1. During Pass 1, the address of the interrupted instruction is transferred from the P-register to the S-bus. From the S-bus, the interrupted instruction address is clocked into the A-register. The D-register is cleared of ones and is filled with zeros. The CS-register is filled with ones. Further action on the interrupted instruction address occurs during Pass 3.

The address in the P-register is transferred to Q at the Pass 1 clock, and, if BRQ is true, the contents of the Q-register are clocked into P. If BRQ is not true, the contents of the P-register are not disturbed at the Pass 1 clock. Thus, during Pass 2, the P-register contains the address of the next instruction if the interrupt was acknowledged at ENDE time, or contains the address of the current instruction if the interrupt was acknowledged at IEN time.

Table 3-17. Signals Enabled by Codes for WD Interrupt Control Mode, and Resulting Changes of State

CODE	TRUE CONTROL SIGNALS	CHANGE OF STATE (ISn, IPn, INn)	
		DATy = 1	DATy = 0
001	DARM	XXX→00X	No change
010	AEENLE, ADBDB, AEADB, DARM	XXX→011	No change
011	ADBDB, AEADB, DARM	XXX→010	No change
100	AEENLE, ADBDB	XXX→XX1	No change
101	ADBDB	XXX→XX0	No change
110	AEENLE, REN	XXX→XX1	XXX→XX0
111	TRIG (if IPn)	X1X→11X	No change
	(if NIPn)	X0X→X0X	

If the memory map feature is used in the system, the map disconnect flip-flop, MAPDIS, is set during INTRAP1 Pass 1 to inhibit mapping for the ensuing phases of the interrupt sequence. MAPDIS prevents any modification of the interrupt address if MAP is true.

At the end of Pass 1, DRQ and INTRAP2 are set. DRQ is a repeater flip-flop that indicates the next clock will not occur until the data release signal, DR, has been received by the CPU from memory, if a memory request, MRQ, has been generated.

The term that generates the T10L clock (S/T10L/1) is always true during Pass 1. Therefore, T10L measures the period for Pass 2.

b. Pass 2. The Pass 2 phase of the sequence sets the highest priority waiting interrupt to the active state, and the interrupt address is transferred to the P-register by the term PXINT. The contents of INT0-INT08 are clocked into P23-P31. Bits P15-P22 of the P-register are simultaneously filled with zeros.

The contents of P15-P31 are put on the LM-LB lines, and the memory request, MRQ, is generated during this pass. Signal CEINT is a clock enable interlock signal that causes the next clock to be synchronized with the 1 MHz interrupt clock. CEINT further insures that the address on the interrupt lines has settled before the next T6L clock is generated. Both INTRAP1 and INTRAP2 are reset and Pass 2 is clocked. During this pass, INTRAP2 is again set so that it goes true for the Pass 3 phase that follows.

c. Pass 3. During Pass 3, the A-register contains the next instruction address of the interrupted instruction. This was accomplished during Pass 1 by SXP and AXS. Since the next instruction address of the interrupted instruction is currently in the A-register, it is necessary to decrease the contents of A by a count of one. This is necessary because when the interrupted instruction address is returned to the P-register after the interrupt, it is increased by a count of one. Therefore, the contents of registers A, D, and CS are added for the decrement of one. The A-register contains the interrupted instruction address, the CS-register contains all ones and the D-register contains all zeros.

G0-G31 = A0-A31 CS0-CS31 ND0-ND31

Data from the memory bus lines, MB, is loaded into the C-register by the data release signal, DR, at any time during INTRAP2. At the next clock, the contents of the C-register are transferred to the O-, R-, and D-registers. Flip-flop SW1 is also set because INTRAPEND is generated.

At the next clock, flip-flop MAPDIS is reset if bit 10 of the Exchange Program Status Doubleword is set which indicates map mode. Flip-flop SW1 is set in this phase, which enables the setting of flip-flop PRE1 at the following clocks. The XPSD instruction is then executed.

The interrupt phase sequence chart is shown in table 3-18. See table 3-153 for an explanation of phase sequence chart symbology.

If HALT is true at this point in the sequence, PCP1 is set.

3-128 Traps

Trap is basically an automatic error detection system. If a call instruction, nonallowed operation, unimplemented instruction, fault condition, or watchdog timer runout is detected, a trap results. The CPU detection of a trap condition causes the immediate execution of the Trap instruction in a unique location in memory. Trap can also be used to simulate optional instructions that have not been physically implemented. The simulation is accomplished by the use of call instructions.

When a trap condition occurs, the CPU sets the trap state flip-flop, INTRAPF. Depending on the type of trap, the instruction currently being executed by the CPU may or may not be carried to completion. In any event, the instruction is terminated with a trap sequence. In this sequence, the instruction address portion of the Program Status Doubleword (PSD), which has already been incremented by one, is decremented by one, and the instruction in the location associated with the trap is executed.

Table 3-18. Interrupt Sequence

Phase	Function Performed	Signals Involved	Comments
ENDE or IEN	Set IEN	S/IEN = FUMMC PH9 + FADE PH8 SW1 + FAMDSF NFAMUL PH1 + FADE PH2 + FABOA PH10 + (FAST PH1) (O4 NO5 O6)	IEN is interrupt enable flip-flop. Interrupt can be accepted on any clock set. Otherwise, interrupt can be accepted only at ENDE. IEN can be set during phases noted in logic
	Interrupt request received	INT9	
	Set INT	S/INT = INT9	
	Set INTRAPF	(S/INTRAPF) = INT ENDE (S/INTRAPF/1) + INT IEN (S/INTRAPF/1) (S/INTRAPF/1) = INT + INTRAPF + INTEN + KRUN/B	(S/INTRAPF) initiates interrupt sequence
	Generate CLEAR signal	CLEAR = ENDE + (S/INTRAPF) + . . .	
	Set INTRAP1	S/INTRAP1 = (S/INTRAPF) NRESET	
	Set BRQ	S/BRQ = (S/INTRAPF) NENDE N(PRE1 NANLZ) + . . .	If interrupt is to point at current instruction rather than next instruction
PASS 1	Q ⇸ P	PXQ = BRQ	Current instruction address
	P ⇸ Q	QXP = NINTRAP2 NCLEAR MEM + . . .	Next instruction address
	P ⇸ S	SXP = INTRAP1 NSDIS + . . .	
IN-TRAP1	S ⇸ A	AXS = INTRAP1 + . . .	
NIN-TRAP2	Clear D-register	DX/1 = INTRAP1 + . . .	
	Fill CS-register with ones	CSX1 = INTRAP1 NSDIS + . . .	For decrementing operation
T6L	Set MAPDIS	S/MAPDIS = INTRAP1 + . . .	Prevents modifications of interrupt address by memory map
	Generate CLEAR signal	CLEAR = INTRAP1 + . . .	
	Set DRQ	S/DRQ = INTRAP1 + . . .	
	Enable T10L	(S/T10L/1) = INTRAP1 NINTRAP2 + . . .	

Mnemonic: Interrupt Sequence

(Continued)

Table 3-18. Interrupt Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
T6L (Cont.)	SCEN	SCEN	= N(INTRAP1 NINTRAP2 NTRAP)	SCEN goes false and inhibits single clock during following clock period
	Reset flip-flop HALT	R/HALT	= INTRAP1 + . . .	
	Set INTRAP2	S/INTRAP2	= INTRAP1 NRESET	
PASS 2	Interrupt address -/→ P-register	PXINT	= INTRAP1 INTRAP2 NTRAP	INT0-8 → P23-P31 0 → P15-P22 0 → P32-P33
IN-TRAP1	Disable signal LMXC	LMXC	= LMXC NAR N(INTRAP1 INTRAP2)	Disable C → LM and LB address lines
IN-TRAP2	Disable AHCL	AHCL	= AHCL N(INTRAP1 INTRAP)	Address here clock
		(INTRAP1 INTRAP)	= INTRAP1 INTRAP2	
	P → S -/→ A	SXP	= INTRAP1 NSDIS + . . .	Current instruction address
		AXS	= INTRAP1 + . . .	
	Generate memory request	MRQ	= INTRAP1 INTRAP2 + . . .	
T10L	Generate CEINT	CEINT	= INTRAP1 INTRAP2 NTRAP + . . .	Synchronizes next CPU clock with 1-MHz interrupt clock
PASS 3 IN-TRAP2 NIN-TRAP1	A-1 → S -/→ B	SXADD	= INTRAP2 NINTRAP1 NSDIS	Decrements next instruction address of interrupted instruction
		BXS	= INTRAP2 + . . .	
	P15-P31 → Q15-Q31	QXP	= INTRAP2 + . . .	Interrupt address
	MB0-MB31 → C0-C31	CXMB	= DGC	Read Program Status Doubleword instruction from interrupt address
	C0 -/→ IA C1-C7 -/→ O1-O7 C8-C11 -/→ R28-R31 C0-C31 -/→ D0-D31	OXC ORXC DXC	= ORXC + OXC = NINTRAP1 INTRAP2 = INTRAP2 + . . .	Instruction word -/→ O-, R-, and D-registers

Mnemonic: Interrupt Sequence

(Continued)

Table 3-18. Interrupt Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PASS 3 (Cont.)	Set flip-flop SW2	S/SW2 = INTRAPEND + . . . INTRAPEND = INTRAPF NINTRAP1 NPREP NEXU	
IN-TRAPF SW2	Reset flip-flop MAPDIS	R/MAPDIS = FAPSD NSW1 SW2 R30	R30⟹bit 10 of XPSD instruction = 1. Specifies map mode
	Set flip-flop SW1	S/SW1 = INTRAPEND SW2 + . . .	
IN-TRAPF SW1	Set flip-flop PRE1	S/PRE1 = INTRAPEND SW1 + . . .	Preparation phase of XPSD instruction

Mnemonic: Interrupt Sequence

An interrupt acknowledgement cannot occur until the execution of the instruction in the trap location is completed. The instruction in the trap location must be an Exchange Program Status Doubleword (XPSD) instruction; if the execution of any other instruction in a trap location is attempted as the result of a trap activation, the results of the instruction are unpredictable.

The XPSD instruction in a trap location is accessed without using the memory map, regardless of whether or not the memory map is in effect when the trap condition occurs. No memory protection violation or privileged instruction violation can occur as a result of either accessing or executing an XPSD instruction in a trap location.

3-129 NONALLOWED OPERATION. The occurrence of one of the nonallowed operations or an interrupt system fault always causes the computer to abort the instruction being executed at the time that the nonallowed operation is detected and to execute immediately the instruction in trap location X'40'.

3-130 NONEXISTENT INSTRUCTION. Any instruction that is neither standard nor optional is defined as nonexistent. This includes immediate addressing instructions that are indirectly addressed. If a nonexistent instruction is detected, the computer traps to location X'40' at the time the nonexistent instruction is detected. For this case, the operation of the XPSD instruction in trap location X'40' with respect to the condition code and instruction address portions of the PSD is as follows:

a. Store current PSD. Condition code stored existed at the end of instruction executed immediately before the nonexistent instruction.

b. Load new PSD. Current PSD is replaced by contents of the doubleword location following doubleword location in which current PSD was stored.

c. Modify new PSD.

1. Set CC1 to 1. CC2, CC3, and CC4 remain set at values loaded from memory.

2. If bit position 9 of XPSD contains a one, the instruction address loaded from memory is incremented by eight. If bit position 9 of XPSD contains a zero, instruction address remains at the value loaded from memory.

3-131 NONEXISTENT MEMORY ADDRESS. Any memory access to a nonexistent memory address causes a trap to location X'40' at the time of the request for memory service. A nonexistent memory address condition is detected by memory on the basis of the actual address presented. If the CPU is currently using the memory map, the virtual address has already been modified by the memory map to generate an actual, but nonexistent, address. The operation of the XPSD in trap location X'40' for this case is as follows:

a. Store current PSD.

b. Load new PSD.

c. Modify the new PSD.

1. Set CC2 to 1. CC1, CC3, and CC4 remain set at values loaded from memory.

2. If bit position 9 of XPSD contains a one, the instruction address loaded from memory is incremented by four. If bit position 9 of XPSD contains a zero, the instruction address remains at the value loaded from memory.

3-132 PRIVILEGED INSTRUCTION IN SLAVE MODE. An attempt to perform a privileged instruction while the CPU is in the slave mode causes a trap to location X'40' at the time of instruction decoding. In this case, the operation of the XPSD in trap location X'40' is as follows:

a. Store current PSD.

b. Load new PSD.

c. Modify the new PSD.

1. Set CC3 to 1. CC1, CC2, and CC4 remain at values loaded from memory.

2. If bit position 9 of XPSD contains a one, the instruction address loaded from memory is incremented by two. If bit position 9 of XPSD contains a zero, the instruction address remains at values loaded from memory.

The operation codes X'0C', X'0D', X'2C', X'2D' and their indirectly addressed forms X'8C', X'8D', X'AC', X'AD' are both nonexistent and privileged. If one of these operation codes is used while the CPU is in the slave state, both CC1 and CC3 are set to ones after the new PSD has been loaded and, if bit position 9 of XPSD contains a one, the instruction address loaded from memory is incremented by 10.

3-133 MEMORY PROTECTION VIOLATION. A memory protection violation can occur either because of a memory map access control bit violation by a slave program using the memory map or because of a memory write lock violation by either a slave or a master mode program. When either memory protection violation occurs, the CPU aborts execution of the current instruction without changing protected memory and traps to location X'40'. In this case, the operation of the XPSD in trap location X'40' is as follows:

a. Store current PSD.

b. Load new PSD.

c. Modify the new PSD.

1. Set CC4 to 1. CC1, CC2, and CC3 remain at values loaded from memory.

2. If bit position 9 of XPSD contains a one, the instruction address loaded from memory is incremented by one. If bit position 9 of XPSD contains a zero, the instruction address remains at the value loaded from memory.

An attempt to access a memory location that is both protected and nonexistent causes both CC2 and CC4 to be set to ones after the new PSD has been loaded and, if bit position 9 of XPSD contains a one, the instruction address loaded from memory is incremented by five.

3-134 UNIMPLEMENTED INSTRUCTIONS. There are two optional instruction groups for the system: the decimal option and the floating point option. The computer traps to location X'41', if an attempt is made to execute a directly or indirectly addressed instruction in either of these groups when the required option is not implemented. An indirectly addressed Edit Byte String (EBS) instruction is always treated as a nonexistent instruction rather than as an unimplemented instruction. The Move to Memory Control (MMC) instruction is always considered implemented even if the memory map or memory protection options are not implemented. The operation of the XPSD in trap location X'41' is as follows:

a. Store current PSD. The condition code stored is that which existed at the end of the instruction immediately before the unimplemented instruction.

b. Load new PSD. The condition code and instruction address portions of the PSD remain at the values loaded from memory.

3-135 PUSH-DOWN STACK LIMITS. Push-down stack overflow or underflow can occur during execution of any of the following instructions:

Push Word

Pull Word

Push Multiple

Pull Multiple

Modify Stack Pointer

During the execution of any stack-manipulating instruction, the stack is either pushed with words added to the stack or is pulled with words removed from the stack. In either case, the space and the word count fields of the stack pointer doubleword are tested before moving any words. If execution of the instruction would cause the space count to become either less than zero or greater than $2^{15}-1$, the instruction is aborted with memory and registers unchanged. If bit 32 (TS) of the stack pointer doubleword is zero, the CPU traps to location X'42'. If execution of the instruction would cause the word count to become either less than zero or greater than $2^{15}-1$, the instruction is aborted with memory and registers unchanged; then, if bit 48 (TW) of the stack pointer doubleword is zero, the CPU traps to location X'42'. If trapping does occur, the condition code remains at the value it had immediately before the instruction that caused the trap. When trapping is inhibited, either CC1 or CC3 is set to one, or both CC1 and CC3 are set to one, to indicate the reason for aborting the instruction. The stack pointer doubleword, memory, and registers are modified only if the instruction is successfully executed. The execution of the XPSD in trap location X'42' is as follows:

a. Store current PSD. The condition code stored is that which existed immediately before execution of the aborted push-down instruction.

b. Load new PSD. The condition code and instruction address portions of PSD remain at values loaded from memory.

3-136 FIXED-POINT OVERFLOW. Fixed-point overflow can occur for any of the following instructions:

Load Complement Word

Load Absolute Word

Load Complement Doubleword

Load Absolute Doubleword

Add Immediate

Add Halfword

Add Word

Add Doubleword

Subtract Halfword

Subtract Doubleword

Divide Halfword

Divide Word

Add Word to Memory

Modify and Test Halfword

Modify and Test Word

Except for the Divide Halfword (DH) and Divide Word (DW) instructions, the instruction execution is allowed to proceed to completion; CC2 is set to one. CC3 and CC4 represent the actual result (0, -, or +) after overflow.

If the fixed-point arithmetic trap mask, bit 11 of PSD, is a one, the CPU traps to location X'43' instead of executing the next instruction in sequence.

For DW and DH, the instruction execution is aborted without changing any registers, and CC2 is set to one. CC1, CC3, and CC4 remain unchanged from their values at the end of the instruction immediately before the DW or DH. If the fixed-point arithmetic trap mask is a one, the CPU traps to location X'43' instead of executing the next instruction in sequence as follows:

a. Store current PSD. If the instruction causing the trap was an instruction other than DW or DH, the stored condition code is interpreted as follows:

CC1	CC2	CC3	CC4	Meaning
-	1	0	0	Result after overflow is zero
-	1	0	1	Result after overflow is negative
-	1	1	0	Result after overflow is positive
0	-	-	-	No carry from bit position 0
1	-	-	-	Carry from bit position 0

CC1 remains unchanged for the LCW, LAW, LCD, and LAD instructions.

If the instruction causing the trap was DW or DH, the stored condition code is interpreted as follows:

CC1	CC2	CC3	CC4	Meaning
-	1	-	-	Overflow

b. Load new PSD. The condition code and instruction address portions of the PSD remain at the value loaded from memory.

3-137 FLOATING POINT ARITHMETIC FAULT CONDITION. Floating point fault detection is performed after the operation called by the instruction code is performed, but before any results are actually loaded into the general registers. Thus, the floating point operation that causes an arithmetic fault is not carried to completion in the sense that the original contents of the general registers remain unchanged. Instead, the computer traps to location X'44' with the current condition code indicating the reason for the trap. A characteristic overflow or an attempt to divide by zero always results in a trap condition. A significance check or a characteristic underflow results in a trap condition only if the floating point mode controls (FS, FZ, and FN) in the program status doubleword are set to the appropriate state.

If a floating point instruction causes a trap, the execution of XPSD in trap location X'44' is as follows:

a. Store current PSD. If division is attempted with a zero divider or if characteristic overflow occurs, stored condition code is interpreted as follows:

CC1	CC2	CC3	CC4	Meaning
0	1	0	0	Divide by zero
0	1	0	1	Characteristic overflow, negative result
0	1	1	0	Characteristic overflow, positive result

b. If none of the above conditions occurs, but characteristic underflow occurs with the floating zero (FZ) mode bit set to one, the stored condition is interpreted as follows:

CC1	CC2	CC3	CC4	Meaning
1	1	0	1	Characteristic underflow, negative result
1	1	1	0	Characteristic underflow, positive result

c. The stored condition code is interpreted in the following manner, if none of the above conditions occur, but an addition or subtraction results in either a zero result (with FS = 1 and FN = 0), or a postnormalization shift of more than two hexadecimal places (with FS = 1 and FN = 0).

CC1	CC2	CC3	CC4	Meaning
1	0	0	0	Zero result of addition or subtraction
1	0	0	1	More than two postnormalizing shifts, negative result
1	0	1	0	More than two postnormalizing shifts, positive result

d. Load the new PSD. The condition code and instruction address portions of the PSD remain at the values loaded from memory.

3-138 DECIMAL ARITHMETIC FAULT. When either of two decimal fault conditions occur, the normal sequencing of instruction execution is halted. CC1 and CC2 are set according to the reason for the fault condition; and CC3, CC4, memory, and the decimal accumulator remain unchanged by the instruction. If the decimal arithmetic trap mask, bit position 10 of the PSD, is a zero, the instruction execution sequence continues with the next instruction in the sequence of the time of fault detection; however, if the decimal arithmetic trap mask bit is a one, the computer traps to location X'45' at the time of fault detection. The execution of XPSD in trap location X'45' is as follows:

a. Store the current PSD. The stored condition code is interpreted as follows:

CC1	CC2	CC3	CC4	Meaning
0	1	-	-	All digits legal and overflow
1	0	-	-	Illegal digit detected

b. Load the new PSD. The condition code and instruction address portions of the PSD remain at the values loaded from memory.

3-139 WATCHDOG TIMER RUNOUT. The instruction watchdog timer ensures that the CPU must periodically reach interruptible points of operation in the execution of instructions. An interruptible point is a time during the execution of a program when an interrupt request, if present, is acknowledged. Interruptible points occur at the end of every instruction, ENDE time, and during the execution of some instructions, IEN time. The watchdog timer measures elapsed time from the last interruptible point. If the maximum allowable time has been reached before the next time that an interrupt could be recognized, the current instruction is aborted, and the watchdog timer runout trap is activated. Except for a nonexistent address used with Read Direct or Write Direct instructions, programs trapped because of watchdog timer runout cannot, in general, be continued. In this case, execution of the XPSD in trap location X'46' is as follows:

a. Store current PSD. Stored condition code is, in general, meaningless.

b. Load new PSD. Condition code and instruction address portions of the PSD remain at the values loaded from memory.

3-140 CALL INSTRUCTIONS. The four call instructions, CAL1, CAL2, CAL3, and CAL4, cause the computer to trap to location X'48' for CAL1, X'49' for CAL2, X'4A' for CAL3, or X'4B' for CAL4. Execution of the XPSD in the appropriate trap location is as follows:

a. Store current PSD. The stored condition code is that which existed at the end of the instruction immediately before the call instruction.

b. Load new PSD.

c. Modify new PSD.

1. The R-field of the call instruction is logically OR-gated with the condition code value loaded from memory, and the result is loaded into the condition code.

2. If bit 9 of XPSD conrains a one, the R-field of the call instruction is added to the instruction address loaded from memory.

3. If bit 9 of XPSD contains a zero, the instruction address remains at the value loaded from memory.

3-141 CPU TRAP SEQUENCE. A trap can occur during the phases noted in the S/TRAP or S/TRACC logic or during the ENDE phase of an instruction. The trap flip-flop remains true until phase 7 of the XPSD instruction sequence. This allows an XPSD to be executed in the slave mode although the XPSD is a privileged instruction.

If a trap condition is detected by the CPU, the term (S/INTRAPF) is one of the terms that initiates the sequence during ENDE. (S/TRAPF) sets INTRAP1, the trap phase flip-flop.

Flip-flop BRQ is set if the return address of the instruction following the trap is in the Q-register. This condition exists when both ENDE and PRE1 are false and (S/INTRAPF) is true. BRQ insures that the address stored by the XPSD instruction points at the correct next instruction address in the program sequence after the trap is processed. The CLEAR signal is generated during ENDE time or when (S/INTRAPF) is true.

Three trap phases are interposed between the trap acknowledgment at ENDE or IEN and preparation phase 1 of the following program status doubleword instruction. These three phases are referenced in the following paragraphs as Pass 1, Pass 2, and Pass 3, where Pass 1 = INTRAP1 NINTRAP2, Pass 2 = INTRAP1 INTRAP2, and Pass 3 = NINTRAP1 INTRAP2.

a. Pass 1. If the return address of the instruction aborted by the trap is in the Q-register during this period, BRQ is true. If BRQ is true, the contents of the Q-register are clocked into the P-register. During the Pass 1 phasing of the trap sequence, the aborted instruction address is transferred from the P-register to the S-bus. From the S-bus, the aborted instruction address is clocked into the A-register. The D-register is cleared of ones and is filled with zeros. The CS-register is filled with ones. MAPDIS is set during this pass to prevent modification of the trap address, and the contents of the P-register are transferred to Q.

The set logic for the DRQ flip-flop (S/DRQ) is true during this pass so that the DRQ flip-flop output S/DRQ is true at the next clock time. DRQ is a repeater flip-flop to indicate that the next clock does not occur until the CPU receives a data release signal from memory if MRQ is true. Since MRQ is true during Pass 2, DRQ is also true during Pass 2. DRQ is reset on the first clock after T10L time. INTRAP2 is also set at the end of Pass 1.

BRQ is reset at T10L time by the T10L clock. The term that generates the T10L clock (S/T10L/1) is always true during Pass 1. Therefore, T10L measures the period for Pass 2.

b. Pass 2. The Pass 2 phase of the sequence clocks the trap address into the P-register. PXINT must be false at this time so that PXTR can be true. The trap address in flip-flops TR28 through TR31 is clocked into the P-register flip-flops P28 through P31. At the same time, zeros are clocked into P15 through P24, P26, P27, P32, and P33. A one is clocked into P25 to establish the most significant digit (MSD) of the trap address. The MSD in all cases is X'4' since trap addresses run from X'40' through X'4B'.

The P-register flip-flop configuration is then as shown in figure 3-131.

The contents of P15-P31 are put on the LM-LB lines. The memory request, MRQ, is generated during this pass. INTRAP1 is reset during this pass and goes false at the next clock time.

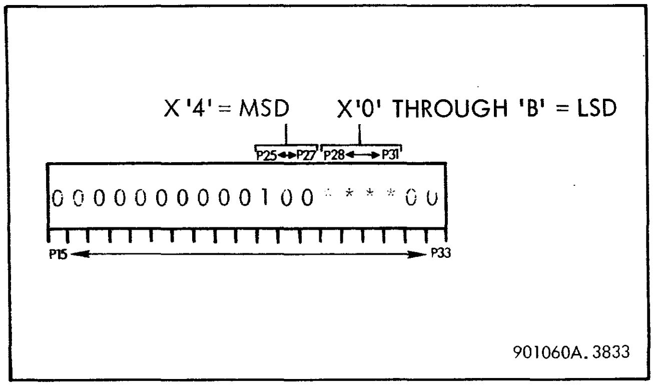


Figure 3-131. P-Register Configuration in Trap Sequence Pass 2

c. Pass 3. During Pass 3, the A-register contains the next instruction address of the aborted instruction. This is accomplished during Pass 1 by SXP and AXS. Since the next instruction address of the aborted instruction is currently in the A-register, it is necessary to decrease the contents of A by a count of one if the CPU operation is to continue after the trap sequence. This is necessary because when the aborted instruction address is returned to the P-register after the trap, it is increased by a count of one. Therefore, the contents of registers A, D, and CS are added together for the decrement of one. The A-register contains the aborted instruction address, the CS-register contains all ones, and the D-register contains all zeros.

The addition is performed in the adder. When the addition is completed, the sum is placed on the S-bus. From the S-bus, the data is transferred to the B-register for storage.

Data from the memory bus lines, MB, is loaded into the C-register by the data release signal DR at any time during INTRAP2. At the next clock, the contents of the C-register are transferred to registers O, R, and D, and PRE1 is set.

Table 3-19 details the logic sequence for the trap system.

3-142 TRAP ADDRESSING. Trap addressing is performed by the trap address flip-flops TR28-TR31 and by the logic term PXTR, as shown in figure 3-132. In all cases where a trap condition exists, P-register flip-flop P25 is set and represents the MSD of the trap address. The MSD is always X'4'. The LSD of the trap address is determined by the trap address flip-flops TR28 through TR31. In figure 3-132, the memory location of the trap is shown as X'41' which results from an unimplemented instruction. The term FANIMP (PRE1 NANLZ) initiates the trap in this example.

3-143 INSTRUCTION ANALYSIS

3-144 Data Formats

The 32-bit Sigma computer word has various data formats for the various instructions. The basic data formats are illustrated in figure 3-133. Signed negative numbers are represented in two's complement form. Decimal numbers are represented in sign-magnitude form. All fixed-point arithmetic operations are performed on integer operands.

3-145 Instruction Formats

The normal instruction format is illustrated in figure 3-134. A second instruction format, providing for immediate operands, is illustrated in the same figure.

3-146 Instruction Code

The seven bit operation code (opcode) field of the instruction word format provides for up to 128 instructions; however, not all of the available opcodes are used in the computer. Table 3-20 lists the available instructions, their hexadecimal opcodes for both direct and indirect addressing, and their names.

3-147 Opcode Families

Many opcodes have identical sequences that are performed during execution of the instruction. For logic purposes, these similar opcodes are grouped into families. Opcodes which are unique or which contain unique execution sequences are assigned function names. The opcodes and the families to which they are assigned are illustrated in figure 3-135.

3-148 Operation Code Decoding

Instructions are read into the C-register during the final phase of the previous instruction. As the CPU sequences into the first preparation state (PRE1) for the instruction in the C-register, the contents of the C-register are gated into the D-register. Bits C1 through C7 which represent the basic operation code of the instruction are also gated into the O-register and remain in the O-register until the instruction has been completed. Bit 0 of the operation code determines if the instruction is indirectly addressed, is gated into flip-flop IA and does not go into the O-register.

Bits O1, O2, and O3 are gated to form all possible combinations of the upper hexadecimal digit of the basic operation code. Since only three of the operation code bits are used, the upper hexadecimal digit can range only from 0 through 7. The lower four bits of the operation code in the O-register are gated to form all possible combinations of the lower hexadecimal digit of the operation code.

Table 3-19. Trap Sequence

Phase	Function Performed	Signals Involved	Comments
During phases noted in S/TR logic	Set TR31	S/TR31 = (S/TR20/1) N(S/TRACC/1)	To memory location X'45'
		+ TROVER N(S/TRACC/1)	To memory location X'43'
		+ FANIMP (PRE1 NANLZ)	To memory location X'41'
		+ (FADIV PH11 CC2 AM)	To memory location X'43'
		+ (FACAL PH1 07 NSTRAP)	To memory location X'49' or to '4B'
	Set TR30	S/TR30 = FACAL PH1 06	To memory location X'4A'
		+ TROVER N(S/TRACC4/1)	To memory location X'43'
		+ (S/TR30/1) N(S/TRACC4/1)	To memory location X'42'
		+ STRAP	To memory location X'46'
	Set TR29	S/TR29 = (S/TR29/1) N(S/TRACC4/1)	To memory location X'44' or '45'
		+ STRAP	To memory location X'46'
		+ FPRR FTRAP	To memory location X'44'
	Set TR28	S/TR28 = (FACAL PH1 NTRAP NSTRAP)	To memory location X'48'
During phases noted in S/TR-ACC logic	Set TRACC1	S/TRACC1 = R28 (FACAL PH1 NTRAP NSTRAP) + FAILL PRE1 NANLZ NTRAP	
	Set TRACC2	S/TRACC2 = R29 (FACAL PH1 NTRAP NSTRAP) + AHCL/1	
	Set TRACC3	TRACC3 = R30 (FACAL PH1 NTRAP NSTRAP) + FAPRIV PRE1 NMASTER NANLZ NTRAP	
	Set TRACC4	S/TRACC4 = R31 (FACAL PH1 NTRAP NSTRAP) + FABOA PH11 SW2 NTRAP + PROTECTD NPROTECTDIS NTRAP + PROTECTI ENDE NFUEXU NTRAP	
ENDE or phase noted in S/TR-AP logic	Set TRAP	S/TRAP = + FAILL (PRE1 NANLZ)	Nonexistent instruction. Trap to memory location X'40'. Set CC1
		+ AHCL/1 DRQ	Nonexistent instruction. Trap to memory location X'40'. Set CC2
		+ FAPRIV (PRE1 NANLZ) NMASTER	Privileged instruction in slave mode. Trap to memory location X'40'. Set CC3

Mnemonic: Trap Sequence

(Continued)

Table 3-19. Trap Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
ENDE or phase noted in S/TRAP logic (Cont.)	Set TRAP	S/TRAP = + (S/TRACC4/1)	Memory protection violation. Trap to memory location X'40'. Set CC4
		+ FANIMP (PRE1 NANLZ) S/TR30	Unimplemented instruction. Trap to memory location X'41'
		+ (S/TR30/1)	Push-down stack limit. Trap to memory location X'42'
		+ TROVER + FADIV PH1 CC2 AM	Fixed-point overflow. Trap to memory location X'43'
		+ FPRR FTRAP	Floating point fault. Trap to memory location X'44'
		+ (S/TR29/1) NPH8 S/TR29 S/TR31	Decimal unit fault. Trap to memory location X'45'
		+ STRAP	Watchdog timer runout. Trap to memory location X'46'
		+ FACAL PH1	Trap to memory location X'48', X'49', X'4A', or X'4B'
		R/TRAP = FAPSD PH7 + RESET	
	Set INTRAPF	S/INTRAPF = TRAP NINTRAPF + . . .	
	Set INTRAP1	S/INTRAP1 = (S/INTRAPF) NRESET	
	Set BRQ	S/BRQ = (S/INTRAPF) NENDE N(PRE1 NANLZ)	BRQ is set if trap is to point at current instruction rather than next instruction
		+ ENDE (S/TRAP)	
	Set flip-flop DRQ under conditions specified	S/DRQ = (NENDE + AHCL/1) (S/INTRAPF)	If nonexistent address or not end phase
	Generate CLEAR signal	CLEAR = ENDE + (S/INTRAPF)	
INTRAP1 INTRAP2	Q —/→ P if BRQ = 1	PXQ = BRQ	Current instruction address
	P ——→ S	SXP = INTRAP1 NSDIS	Next instruction address
	S —/→ A	AXS = INTRAP1	

Mnemonic: Trap Sequence

(Continued)

Table 3-19. Trap Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
Pass 1	Clear D-register	DX/1 = INTRAP1	
	Fill CS-register with ones	CSX1 = INTRAP1 NSDIS	For decrementing operation
	Set MAPDIS	S/MAPDIS = INTRAP1	Prevents modification of trap address by memory map
	Reset HALT flip-flop	R/HALT = INTRAP1	
T6L	Set DRQ	S/DRQ = INTRAP1	
	Reset BRQ	R/BRQ =	
	Enable T10L	(S/T10L/1) = INTRAP1 N(NCX/1 NBRQ)	
	Set flip-flop INTRAP2	S/INTRAP2 = INTRAP1 NRESET	
IN-TRAP1 IN-TRAP2	TR28-TR31 -/→ P28-P31	PXTR = INTRAP1 INTRAP2 TRAP	Trap address -/→ P-register
			0-/→ P15-P24
			1-/→ P25
			0-/→P26
			0-/→P32-P33
	P→S-/→A	AXS = INTRAP1 + . . .	Current instruction address
		SXP = INTRAP1 NSDIS + . . .	
Pass 2	P15-P31→LM15-LB31	LM15 = P15 NLMXC NLMXQ	Output of P-register flip-flops is put on LM-LB lines
		⋮ ⋮	
		LM22 = P22 NLMXC NLMXQ	
		LB23 = P23 NLMXC NLMXQ	
		⋮ ⋮	
		LB31 = P31 NLMXC NLMXQ	
	Disable signal LMXC	LMXC = LMXC NAR N(INTRAP1 INTRAP2)	
T10L	Generate memory request	MRQ = INTRAP1 (NCX/1 NBRQ)	
	Clear D-register	DX/1 = INTRAP1 + . . .	
	Disable AHCL	AHCL = AHCL N(INTRAP1 INTRAP)	
	Reset INTRAP1	(INTRAP1 INTRAP) = INTRAP1 INTRAP2	
	Set INTRAP2	R/INTRAP1 = NCX/1 NBRQ	
		S/INTRAP2 = INTRAP1 NCX/1 NBRQ	

Mnemonic: Trap Sequence

(Continued)

Table 3-19. Trap Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
		Note The trap flip-flop remains true until phase 7 of the XPSD sequence. This allows an XPSD to be executed in slave mode although XPSD is a privileged instruction.	
Pass 3 INTRAP2 NINTRAP1	A-1 → S ⇸ B	SXADD = INTRAP2 NINTRAP1 NSDIS	Decrements next instruction address of trapped instruction
		BXS = INTRAP2 + . . .	
	P15-P31 → Q15-Q31	QXP = INTRAP2 + . . .	Trap address
	MB0-MB31 → C0-C31	CXMB = DGC	Read program status doubleword instruction from trap address

	C0 ⇸ IA
	C1-C7 ⇸ O1-O7
	C8-C11 ⇸ R28-R31
	C0-C31 ⇸ D0-D31
		OXC = ORXC + OXC
		ORXC = NINTRAP1 INTRAP2
		DXC = INTRAP2 + . . .
			Instruction word ⇸ O-, R-, and D-registers

	Set flip-flop SW2	S/SW2 = INTRAPEND + . . . INTRAPEND = INTRAPF NINTRAP1 NPREP NEXU	
INTRAPF SW2	Reset flip-flop MAPDIS	R/MAPDIS = FAPSD NSW1 SW2 R30	R30 ⇒ bit 10 of XPSD instruction = 1. Specifies map mode
	Set flip-flop SW1	S/SW1 = INTRAPEND SW2 + . . .	
INTRAPF SW1	Set flip-flop PRE1	S/PRE1 = INTRAPEND SW1 + . . .	Preparation phase of XPSD instruction

Mnemonic: Trap Sequence

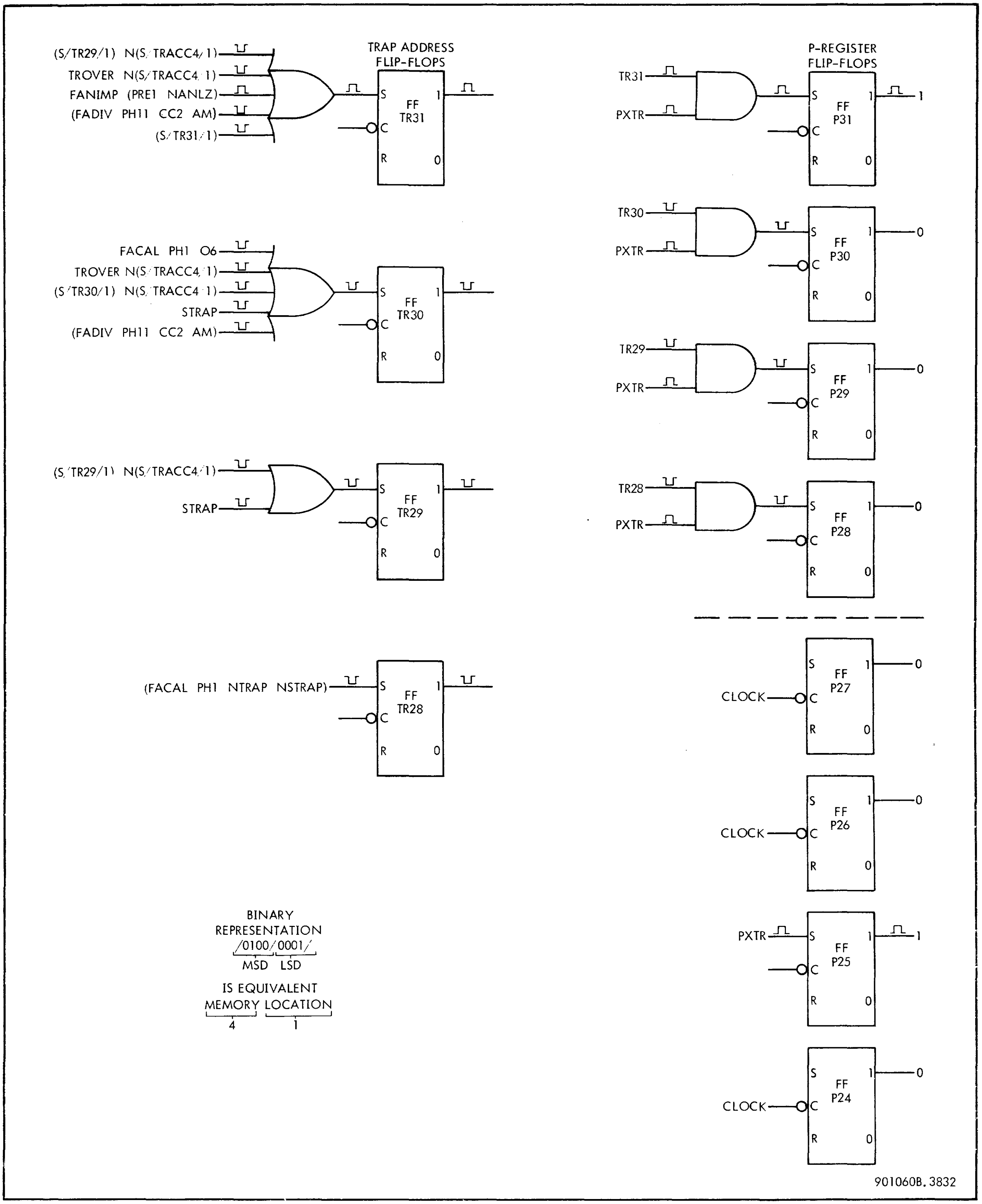
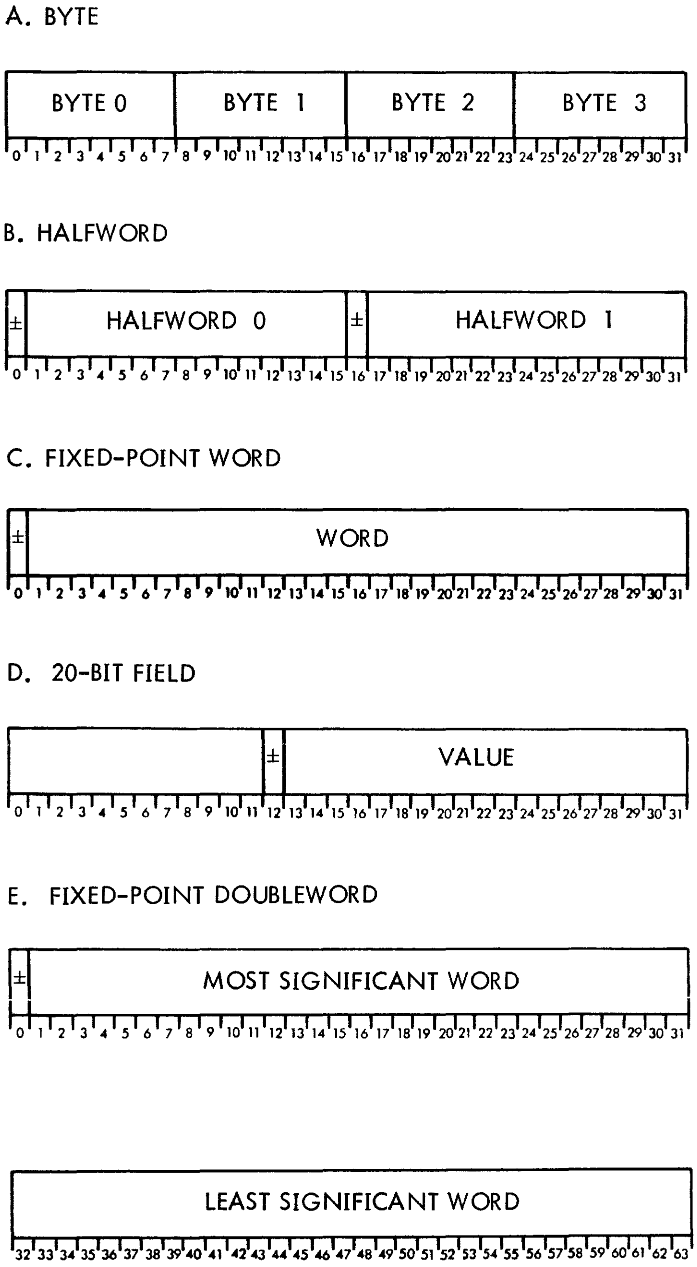
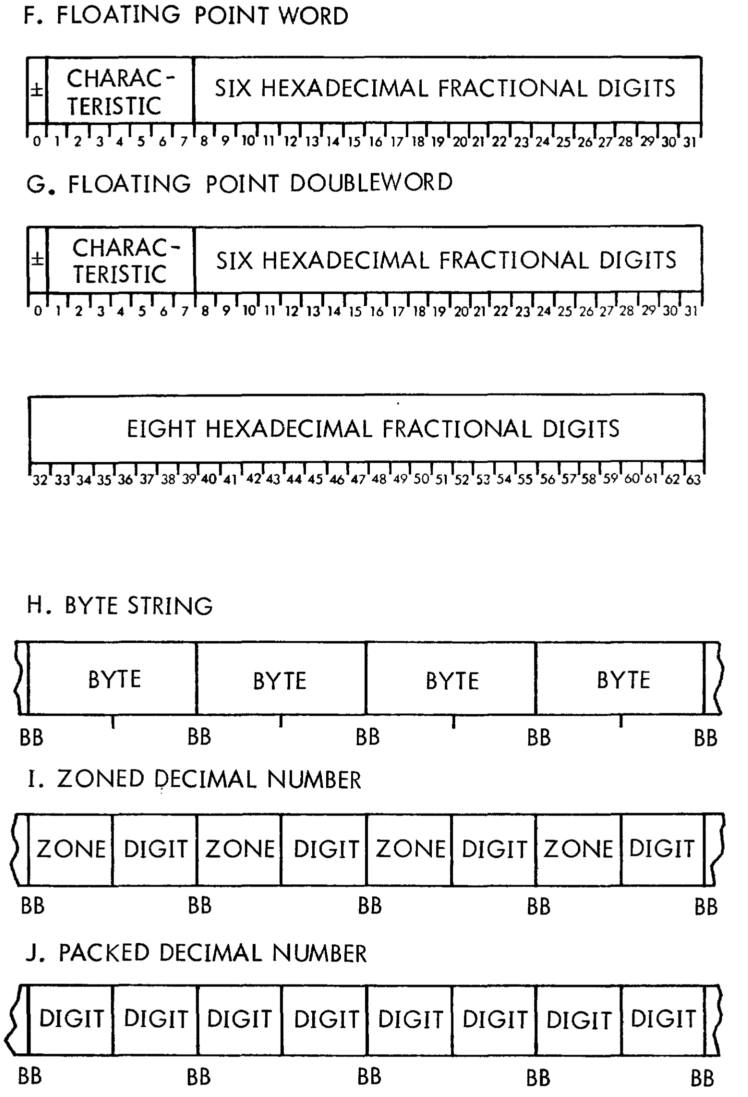


Figure 3-132. Trap Memory Location and Addressing



901060A.3600/1

Figure 3-133. Data Word Formats (Sheet 1 of 2)



NOTE: BB MUST BE A BYTE BOUNDARY. FIRST BIT AFTER BYTE BOUNDARY IS BIT 0, 8, 16, OR 24

901060A. 3600/2

Figure 3-133. Data Word Formats (Sheet 2 of 2)

IA	OPERATION	R OR L	X	ADDRESS
0	1 2 3 4 5 6 7	8 9 10 11	12 13 14	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

NOTES :

IA — INDIRECT ADDRESSING IS PERFORMED IF THIS BIT POSITION CONTAINS A ONE AND IS NOT PERFORMED IF THIS BIT POSITION CONTAINS A ZERO

OPERATION — SEVEN-BIT FIELD THAT DESIGNATES THE OPERATION TO BE PERFORMED

R — FOUR-BIT FIELD THAT DESIGNATES ANY OF THE 16 REGISTERS OF THE CURRENT REGISTER PAGE AS AN OPERAND SOURCE OR DESTINATION

L — IN DECIMAL OPERATIONS, THIS FIELD INDICATES THE LENGTH, IN BYTES, OF THE DECIMAL OPERAND. L = 1 DESIGNATES A 1-DIGIT SIGNED DECIMAL NUMBER; L = 2 DESIGNATES A 3-DIGIT SIGNED DECIMAL NUMBER. IN GENERAL, L = n DESIGNATES A 2n-1 DIGIT SIGNED DECIMAL NUMBER. L = 0 DESIGNATES 16 BYTES OR A 31-DIGIT SIGNED DECIMAL NUMBER

X — THREE-BIT FIELD THAT DESIGNATES ANY OF REGISTERS 1 THROUGH 7 OF THE CURRENT REGISTER PAGE AS AN INDEX REGISTER. X = 0 DESIGNATES NO INDEXING; THEREFORE, REGISTER 0 CANNOT BE USED AS AN INDEX REGISTER

ADDRESS — SEVENTEEN-BIT FIELD THAT CONTAINS THE INITIAL VIRTUAL ADDRESS OF THE INSTRUCTION OPERAND. THE ADDRESS FIELD ALLOWS ANY WORD, DOUBLEWORD, LEFT HALFWORD, OR LEFTMOST BYTE WITHIN A WORD IN MEMORY TO BE DIRECTLY ADDRESSED

A. NORMAL INSTRUCTION

0	OPERATION	R	±	VALUE OR DISPLACEMENT
0	1 2 3 4 5 6 7	8 9 10 11	12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

NOTES :

0 — INDIRECT ADDRESSING CANNOT BE USED WITH THIS TYPE OF INSTRUCTION. IF INDIRECT ADDRESSING IS USED, THE COMPUTER TRAPS TO LOCATION X'64' AT THE TIME OF OPERATION CODE DECODING

OPERATION — SEVEN-BIT FIELD THAT DESIGNATES THE OPERATION TO BE PERFORMED BETWEEN THE IMMEDIATE OPERAND AND THE DESIGNATED REGISTER

R — FOUR-BIT FIELD THAT DESIGNATES ANY OF THE 16 REGISTERS OF THE CURRENT REGISTER PAGE

VALUE — IMMEDIATE OPERAND 20 BITS IN LENGTH (19 MAGNITUDE BITS PLUS SIGN BIT WITH NEGATIVE NUMBERS IN TWO'S COMPLEMENT FORM). THE SIGN BIT (BIT POSITION 12) OF THE VALUE IS EXTENDED 12 BIT POSITIONS TO THE LEFT TO PROVIDE A 32-BIT OPERAND

DISPLACEMENT — A 20-BIT FIELD THAT CONTAINS A SIGNED 19-BIT QUANTITY USED AS DISPLACEMENT VALUE TO FORM AN EFFECTIVE BYTE ADDRESS IN BYTE STRING INSTRUCTIONS

B. IMMEDIATE OPERAND INSTRUCTION

901060A.3602

Figure 3-134. Instruction Word Formats

Table 3-20. Operation Code List

Mnemonic	Opcode*	Name
AD	10, 90	Add Doubleword
AH	50, D0	Add Halfword
AI	20	Add Immediate
AIO	6E, EE	Acknowledge I/O Interrupt (privileged)
AND	4B, CB	AND Word
ANLZ	44, C4	Analyze
AW	30, B0	Add Word
AWM	66, E6	Add Word to Memory
BAL	6A, EA	Branch and Link
BCR	68, E8	Branch on Conditions Reset
BCS	69, E9	Branch on Conditions Set
BDR	64, E4	Branch on Decrementing Register
BIR	65, E5	Branch on Incrementing Register
CAL1	04, 84	Call 1
CAL2	05, 85	Call 2
CAL3	06, 86	Call 3
CAL4	07, 87	Call 4
CB	71, F1	Compare Byte
CBS	60	Compare Byte String
CD	11, 91	Compare Doubleword
CH	51, D1	Compare Halfword
CI	21	Compare Immediate
CLM	19, 99	Compare with Limits in Memory
CLR	39, B9	Compare with Limits in Register
CS	45, C5	Compare Selective
CVA	29, A9	Convert by Addition
CVS	28, A8	Convert by Subtraction
CW	31, B1	Compare Word
DA	79, F9	Decimal Add (optional)
DC	7D, FD	Decimal Compare (optional)
DD	7A, FA	Decimal Divide (optional)
DH	56, D6	Divide Halfword
DL	7E, FE	Decimal Load (optional)
DM	7B, FB	Decimal Multiply (optional)
DS	78, F8	Decimal Subtract (optional)
DSA	7C, FC	Decimal Shift Arithmetic (optional)

*First opcode specifies direct addressing; second specifies indirect addressing

(Continued)

Table 3-20. Operation Code List (Cont.)

Mnemonic	Opcode*	Name
DST	7F, FF	Decimal Store (optional)
DW	36, B6	Divide Word
EBS	63	Edit Byte String (optional)
EOR	48, C8	Exclusive OR Word
EXU	67, E7	Execute
FAL	1D, 9D	Floating Add Long (optional)
FAS	3D, BD	Floating Add Short (optional)
FDL	1E, 9E	Floating Divide Long (optional)
FDS	3E, BE	Floating Divide Short (optional)
FML	1F, 9F	Floating Multiply Long (optional)
FMS	3F, BF	Floating Multiply Short (optional)
FSL	1C, 9C	Floating Subtract Long (optional)
FSS	3C, BC	Floating Subtract Short (optional)
HIO	4F, CF	Halt I/O (privileged)
INT	6B, EB	Interpret
LAD	1B, 9B	Load Absolute Doubleword
LAH	5B, DB	Load Absolute Halfword
LAW	3B, BB	Load Absolute Word
LB	72, F2	Load Byte
LCD	1A, 9A	Load Complement Doubleword
LCF	70, F0	Load Conditions and Floating Control
LCFI	02	Load Conditions and Floating Control Immediate
LCH	5A, DA	Load Complement Halfword
LCW	3A, BA	Load Complement Word
LD	12, 92	Load Doubleword
LH	52, D2	Load Halfword
LI	22	Load Immediate
LM	2A, AA	Load Multiple
LPSD	0E, 8E	Load Program Status Doubleword (privileged)
LRP	2F, AF	Load Register Pointer (privileged)
LS	4A, CA	Load Selective
LW	32, B2	Load Word
MBS	61	Move Byte String
MH	57, D7	Multiply Halfword
MI	23	Multiply Immediate
MMC	6F, EF	Move to Memory Control (privileged)

*First opcode specifies direct addressing; second specifies indirect addressing

(Continued)

Table 3-20. Operation Code List (Cont.)

Mnemonic	Opcode*	Name
MSP	13, 93	Modify Stack Pointer
MTB	73, F3	Modify and Test Byte
MTH	53, D3	Modify and Test Halfword
MTW	33, B3	Modify and Test Word
MW	37, B7	Multiply Word
OR	49, C9	OR Word
PACK	76, F6	Pack Decimal Digits (optional)
PLM	0A, 8A	Pull Multiple
PLW	08, 88	Pull Word
PSM	0B, 8B	Push Multiple
PSW	09, 89	Push Word
RD	6C, EC	Read Direct (privileged)
S	25, A5	Shift
SD	18, 98	Subtract Doubleword
SF	24, A4	Shift Floating
SH	58, D8	Subtract Halfword
SIO	4C, CC	Start I/O (privileged)
STB	75, F5	Store Byte
STCF	74, F4	Store Conditions and Floating Controls
STD	15, 95	Store Doubleword
STH	55, D5	Store Halfword
STM	2B, AB	Store Multiple
STS	47, C7	Store Selective
STW	35, B5	Store Word
SW	38, B8	Subtract Word
TBS	41	Translate Byte String
TDV	4E, CE	Test Device (privileged)
TIO	4D, CD	Test I/O (privileged)
TTBS	40	Translate and Test Byte String
UNPK	77, F7	Unpack Decimal Digits (optional)
WAIT	2E, AE	Wait (privileged)
WD	6D, ED	Write Direct (privileged)
XPSD	0F, 8F	Exchange Program Status Doubleword (privileged)
XW	46, C6	Exchange Word

*First opcode specifies direct addressing; second specifies indirect addressing

FAMILIES

MNEMONIC	FABC	FABO	FABO/1	FABO/2	FABOA	FABOA/1	FABOA/2	FABOX	FABR	FABRANCH	FABS	FABSA	FACAL	FACV	FADE	FADIO	FADIV	FADIVH	FADIVW	FADW	FAFL	FAFLM	FAFLMD	FAFLAS	FAFRR	FAFRR/1	FAILL	FAIM	FAIO	FAIO/1	FAMDSF/D	FAMDSF/M	FAMUL	FAMULH	FAMULI	FAMULNH
AD																				●																
AH																																				
AI																												●								
AIO																●													●							
AND																																				
ANLZ																																				
AW																																				
AWM																																				
BAL										●																										
BCR	●									●																										
BCS	●									●																										
BDR									●	●																										
BIR									●	●																										
CAL1													●																							
CAL2													●																							
CAL3													●																							
CAL4													●																							
CB																																				
CBS		●	●	●	●	●	●	●			●	●																●								
CD																				●																
CH																																				
CI																												●								
CLM																				●																
CLR																																				
CS																																				
CVA														●																						
CVS														●																						
CW																																				
DA															●																					
DC															●																					
DD															●																					
DH																	●	●													●					
DL															●																					
DM															●																					
DS															●																					
DSA															●																					
DST															●																					
DW																	●	●								●					●					
EBS		●		●	●		●	●																				●								
EOR																																				
EXU																																				
FAL																				●	●			●												
FAS																					●			●												
FDL																				●	●		●								●					
FDS																					●		●								●					
FML																				●	●	●	●			●						●				
FMS																					●	●	●									●				
FSL																				●	●			●												
FSS																					●			●												
HIO																●													●	●						
INT																																				
LAD																				●																
LAH																																				

MNEMONIC	FANIMP	FAPRIV	FAPSD	FARWD	FASH	FASHFL	FASHFX	FATR	FATRA	FATR5	FAS1	FAS2	FAS3	FAS6	FAS7	FAS8	FAS9	FAS10	FAS11	FAS12	FAS13	FAS14	FAS15	FAS16	FAS17	FAS18	FAS19	FAS21	FAS22	FAS23	FAS24	FAS26	FAST/1	FAST	FASTA	FAW
AD													●																●							
AH																				●																
AI																				●																
AIO		●																																		●
AND																	●																			●
ANLZ																																				●
AW																				●																●
AWM																●															●					●
BAL																																				●
BCR																																				●
BCS																																				●
BDR																																				●
BIR																																				●
CAL1																																				●
CAL2																																				●
CAL3																																				●
CAL4																																				●
CB																			●																	
CBS																																				
CD												●																	●							
CH																			●																	
CI																			●																	
CLM											●																		●							
CLR											●																		●							●
CS														●									●													●
CVA																																				●
CVS																																				●
CW																			●																	●
DA																																				
DC																																				
DD																																				
DH																																				
DL																																				
DM																																				
DS																																				
DSA																																				
DST																																				
DW																																				●
EBS																																				
EOR																	●																			●
EXU																																				●
FAL																																				
FAS																																				●
FDL																																				
FDS																																				●
FML																																				
FMS																																				●
FSL																																				
FSS																																				●
HIO		●																																		●
INT																																				●
LAD																								●			●									
LAH																		●										●								

MNEMONIC	FUANLZ	FUBAL	FUCS	FUCVS	FUEBS	FUEBSA	FUEBS8	FUEBS9	FUEBS14	FUEXU	FULAD	FULCD	FULD	FULS	FULRP	FUMMC	FUMTH	FUSTD	FUSTH	FUSTS	FUWAIT	FAFLD	FUAD	FUAWM	FUCB	FUCI	FUCLR	FUEOR	FAMDSF	FUMTB	FUMTW	FUOR	FUXW	FUINT
AD																							●											
AH																																		
AI																																		
AIO																																		
AND																																		
ANLZ	●																																	
AW																																		
AWM																								●										
BAL		●																																
BCR																																		
BCS																																		
BDR																																		
BIR																																		
CAL1																																		
CAL2																																		
CAL3																																		
CAL4																																		
CB																									●									
CBS																																		
CD																																		
CH																																		
CI																										●								
CLM																																		
CLR																											●							
CS			●																															
CVA																																		
CVS				●																														
CW																																		
DA																																		
DC																																		
DD																																		
DH																													●					
DL																																		
DM																																		
DS																																		
DSA																																		
DST																																		
DW																													●					
EBS					●	●	●	●	●																									
EOR																												●						
EXU										●																								
FAL																													●					
FAS																													●					
FDL																						●							●					
FDS																						●							●					
FML																													●					
FMS																													●					
FSL																													●					
FSS																													●					
HIO																																		
INT																																		●
LAD											●																							
LAH																																		

Figure 3-135. Instruction Menmonics and Families (Sheet 1 of 2)

901060A. 3604/1

FAMILIES

MNEMONIC	FABC	FABO	FABO/1	FABO/2	FABOA	FABOA/1	FABOA/2	FABOX	FABR	FABRANCH	FABS	FABSA	FACAL	FACV	FADE	FADIO	FADIV	FADIVH
LAW																		
LB																		
LCD																		
LCF																		
LCFI																		
LCH																		
LCW																		
LD																		
LH																		
LI																		
LM																		
LPSD																		
LRP																		
LS																		
LW																		
MBS		●	●	●	●	●	●	●			●	●						
MH																		
MI																		
MMC																		
MSP																		
MTB																		
MTH																		
MTW																		
MW																		
OR																		
PACK															●			
PLM																		
PLW																		
PSM																		
PSW																		
RD																●		
S																		
SD																		
SF																		
SH																		
SIO																●		
STB																		
STCF																		
STD																		
STH																		
STM																		
STS																		
STW																		
SW																		
TBS		●	●		●	●	●											
TDV																●		
TIO																●		
TTBS		●	●		●	●	●											
UNPK															●			
WAIT																		
WD																●		
XPSD																		
XW																		

MNEMONIC	FADIVW	FADW	FAFL	FAFLM	FAFLMD	FAFLAS	FAFRR	FAFRR/1	FAILL	FAIM	FAIO	FAIO/1	FAMDSF/D	FAMDSF/M	FAMUL	FAMULH	FAMULI	FAMULNH
LAW																		
LB																		
LCD		●																
LCF																		
LCFI										●								
LCH																		
LCW																		
LD		●																
LH																		
LI										●								
LM																		
LPSD																		
LRP																		
LS																		
LW																		
MBS										●								
MH														●	●	●		
MI							●			●				●	●		●	●
MMC																		
MSP		●																
MTB																		
MTH																		
MTW																		
MW								●						●	●			●
OR																		
PACK																		
PLM		●																
PLW		●																
PSM		●																
PSW		●																
RD																		
S							●											
SD		●																
SF																		
SH																		
SIO											●	●						
STB																		
STCF																		
STD		●					●											
STH																		
STM																		
STS								●										
STW																		
SW																		
TBS										●								
TDV											●	●						
TIO											●	●						
TTBS										●								
UNPK																		
WAIT																		
WD																		
XPSD																		
XW																		

MNEMONIC	FANIMP	FAPRIV	FAPSD	FARWD	FASH	FASHFL	FASHFX	FATR	FATRA	FATR5	FAS1	FAS2	FAS3	FAS6	FAS7	FAS8	FAS9	FAS10
LAW																		●
LB																		
LCD																		
LCF																		
LCFI																		
LCH																		
LCW																		
LD																		
LH																		
LI																		
LM																		
LPSD		●	●															
LRP		●																
LS														●				
LW																		
MBS																		
MH																		
MI																		
MMC		●																
MSP																		
MTB															●			
MTH															●			
MTW															●	●		
MW																		
OR																	●	
PACK																		
PLM																		
PLW																		
PSM																		
PSW																		
RD		●		●														
S					●		●											
SD												●	●					
SF					●	●												
SH																		
SIO		●																
STB																		
STCF																		
STD																		
STH																		
STM																		
STS														●				
STW																		
SW																		
TBS								●	●	●								
TDV		●																
TIO		●																
TTBS								●	●	●								
UNPK																		
WAIT		●																
WD		●		●														
XPSD		●	●															
XW																	●	

MNEMONIC	FAS11	FAS12	FAS13	FAS14	FAS15	FAS16	FAS17	FAS18	FAS19	FAS21	FAS22	FAS23	FAS24	FAS26	FAST/1	FAST	FASTA	FAW
LAW										●				●				●
LB										●		●						
LCD						●			●									
LCF							●											
LCFI							●											
LCH										●		●						
LCW										●		●		●				●
LD						●												
LH										●		●						
LI										●		●						
LM															●	●	●	●
LPSD																		
LRP																		●
LS					●													●
LW										●		●						
MBS																		
MH																		
MI																		
MMC																		●
MSP															●	●	●	
MTB			●										●					
MTH			●										●					●
MTW													●					
MW																		●
OR																		●
PACK																		
PLM															●	●	●	
PLW															●	●	●	
PSM															●	●	●	
PSW															●	●	●	
RD																		●
S																		●
SD											●							
SF																		●
SH		●																
SIO																		●
STB								●										
STCF								●										
STD				●														
STH								●										
STM															●	●	●	●
STS				●														●
STW								●										●
SW		●																●
TBS																		
TDV																		●
TIO																		●
TTBS																		
UNPK																		
WAIT																		●
WD																		●
XPSD																		
XW																		●

MNEMONIC	FUANLZ	FUBAL	FUCS	FUCVS	FUEBS	FUEBSA	FUEBS8	FUEBS9	FUEBS14	FUEXU	FULAD	FULCD	FULD	FULS	FULRP	FUMMC	FUMTH	FUSTD
LAW																		
LB																		
LCD												●						
LCF																		
LCFI																		
LCH																		
LCW																		
LD													●					
LH																		
LI																		
LM																		
LPSD																		
LRP															●			
LS														●				
LW																		
MBS																		
MH																		
MI																		
MMC																●		
MSP																		
MTB																		
MTH																	●	
MTW																		
MW																		
OR																		
PACK																		
PLM																		
PLW																		
PSM																		
PSW																		
RD																		
S																		
SD																		
SF																		
SH																		
SIO																		
STB																		
STCF																		
STD																		●
STH																		
STM																		
STS																		
STW																		
SW																		
TBS																		
TDV																		
TIO																		
TTBS																		
UNPK																		
WAIT																		
WD																		
XPSD																		
XW																		

MNEMONIC	FUSTH	FUSTS	FUWAIT	FAFLD	FUAD	FUAWM	FUCB	FUCI	FUCLR	FUFOR	FAMDSF	FUMTB	FUMTW	FUOR	FUXW	FUINT
LAW																
LB																
LCD																
LCF																
LCFI																
LCH																
LCW																
LD																
LH																
LI																
LM																
LPSD																
LRP																
LS																
LW																
MBS																
MH											●					
MI											●					
MMC																
MSP																
MTB												●				
MTH																
MTW													●			
MW											●					
OR														●		
PACK																
PLM																
PLW																
PSM																
PSW																
RD																
S											●					
SD																
SF											●					
SH																
SIO																
STB																
STCF																
STD																
STH	●															
STM																
STS		●														
STW																
SW																
TBS																
TDV																
TIO																
TTBS																
UNPK																
WAIT			●													
WD																
XPSD																
XW															●	

Figure 3-135.Instruction Menmonics and Families
(Sheet 2 of 2)

901060A.3604/2

OU0 = NO1 NO2 NO3

OU1 = NO1 NO2 O3

OU2 = NO1 O2 NO3

OU3 = NO1 O2 O3

OU4 = O1 NO2 NO3

OU5 = O1 NO2 O3

OU6 = O1 O2 NO3

OU7 = O1 O2 O3

OL0 = NO4 NO5 NO6 NO7

OL1 = NO4 NO5 NO6 NO7

OL2 = NO4 NO5 O6 NO7

OL3 = NO4 NO5 O6 O7

OL4 = NO4 O5 NO6 NO7

OL5 = NO4 O5 NO6 O7

OL6 = NO4 O5 O6 NO7

OL7 = NO4 O5 O6 O7

OL8 = O4 NO5 NO6 NO7

OL9 = O4 NO5 NO6 O7

OLA = O4 NO5 O6 NO7

OLB = O4 NO5 O6 O7

OLC = O4 O5 NO6 NO7

OLD = O4 O5 NO6 O7

OLE = O4 O5 O6 NO7

OLF = O4 O5 O6 O7

3-149 Function Code Logic

Certain combinations of the upper and lower hexadecimally decoded digits of the O-register are gated to form function codes. The decoded gates are designated as FUXX, where XX describes the function by its mnemonic. Only the following functions or instructions are logically represented.

FUAD = OU1 OL0

FUANLZ = OU4 OL4

FUAWM = OU6 OL6

FUBAL = OU6 OLA

FUCB = OU7 OL1

FUCI = OU2 OL1

FUCLR = OU3 OL9

FUCS = OU4 OL5

FUCVS = FACV NO7

FUEOR = OU4 OL8

FUEXU = OU6 OU7

FULAD = OU1 OLB

FULCD = OU1 OLA

FULD = OU1 OL2

FUINT = OU6 OLB

FULRP = OU2 OLF

FULS = OU4 OLA

FUMMC = OU6 OLF

FUMTB = OU7 OL3

FUMTH = OU5 OL3

FUMTW = OU3 OL3

FUSTD = OU1 OL5

FUSTH = OU5 OL5

FUOR = OU4 OL9

FUSTS = OU4 OL7

FUWAIT = OU2 OLE

FUXW = OU4 OL6

3-150 Family Code Logic

Families of instructions are made up of combinations of the function codes and O-register bits to form gates that describe instruction groups having common usage. These gates are designated as FAXX, where XX describes the family group. For example, FAIM represents the family of instructions that can be executed in the immediate mode. Table 3-21 describes most families of instructions. Most families that are qualified by terms other than the function codes or O-register bits are omitted; for example, FATR = FABOX NO2 NPHA.

Table 3-21. Family Code Logic

Families of Instructions		Hexadecimal Opcodes	Instruction
FABC	= OU6 O4 NO5 NO6	68/69	BCR, BCS
FABOX	= OU6 OL3	63	EBS
	+ O1 NO3 NO4 NO5 NO6	40/41, 60/61	TBS, CBS, MBS
FABR	= OU6 NO4 O5 NO6	64/65	BIR, BDR
FACAL	= OU0 NO4 O5	04→07	CAL1→CAL4
FACV	= OU2 O4 NO5 NO6	28/29	CVA, CVS
FADE	= OU7 OU4	78→7F	
	+ OU7 NO4 O5 O6	76/77	
FADIV	= OU5 OL6	56	DH
	+ OU3 OL6	36	DW
FADW	= OU1	10→1F	Doubleword functions
	+ OU0 O4	08→0F	PSW, PLM, PSM, LPSD, XPSD
FAFL	= NO1 O3 O4 O5	1C→1F 3C→3F	Floating point
FAFLD	= NO1 O3 OLE	1E/3E	Floating divide
FAFLM	= NO1 O3 OLF	1F/3F	Floating multiply
FAFRR	= OU2 OL3	23	MI
	+ OU2 OL5	25	S
	+ OU1 OL5	15	STD
FAFRR/1	= OU3 NO4 O5 O6	36/37	DW, MW
	+ OU4 OL7	47	SS
	+ OU1 OLF	1F	FML
FAILL	= OU5 O4 O5	5C→5F	Illegal operations causing traps
	+ NO1 O3 OL4	14/34	
	+ OU0 NO4 NO5 NO6	00/01	
	+ NO2 NO3 OL3	03/43	
	+ O1 NO3 OL2	42/62	
	+ OU5 O4 NO6 O7	59/5D	
	+ OU5 O5 NO6 NO7	54/5C	
	+ OU1 NO4 O5 O6	16/17	
	+ NO1 NO3 O4 O5 O6	0C/0D, 2C/2D	
	+ OU2 NO4 O5 O6	26/27	
	+ IA NO3 NO4 NO5		
FAIM	= NO3 NO4 NO5	00→03, 20→23	Immediate mode
FAIO	= OU4 O4 O5	4C→4F	I/O
	+ OU6 OLE	6E	AIO
FAMDSF	= FAMUL + FADIV + FASH + FAFL + SDIS		

(Continued)

Table 3-21. Family Code Logic (Cont.)

Families of Instructions	Hexadecimal Opcodes	Instruction
FAMDSF/D = FADIV + FAFLD		
FAMUL = OU5 OL7	57	MH
+ OU3 OL7	37	MW
+ OU2 OL3	23	SF
FANIMP = NO1 O3 O4 O5 NFPOPTION		Nonimplemented instructions
+ OU7 O4 NDEOPTION		
+ OU7 NO4 O5 O6 NDEOPTION		
+ FUEBS NIA NDEOPTION		
FAPRIV = NO3 O4 O5		Privileged instructions
FAPSD = OU0 O4 O5 O6	OE/OF	LPSD, XPSD
FARWD = OU6 O4 O5 O6	6C/6D	RD, WD
FAS1 = NO1 O3 OL9	19/39	CLM, CLR
FAS2 = OU1 OL8	18	SD
+ OU1 OL1	11	CD
FAS3 = OU1 NO5 NO6 NO7	10/18	AD, SD
FAS6 = OU4 NO4 O5 O7	45/47	CS, STS
+ OU4 OLA	0A	PLM
FAS7 = O2 O3 OL3	33/73	MTW, MTB
+ O1 O3 OL3	53/73	MTH, MTB
FAS8 = OU3 OL3	33	
+ OU6 OL6	66	
FAS9 = OU4 OL6	46	XW
+ OU4 O4 NO5 NO6	48/49	EOR, OR
+ OU4 O4 NO5 O7	49/4B	OR, AND
FAS10 = OU3 OLB	3B	LAW
OU5 OLB	5B	LAH
FAS11 = NO1 O2 OL1	21/31	CI, CW
+ O1 O3 OL1	51/71	CH, CB
FAS12 = NO1 O2 OL0	20/30	AI, AW
+ OU3 NO5 NO6 NO7	30/38	AW, SW
+ OU5 NO5 NO6 NO7	50/58	AH, SH
FAS13 = O1 O3 OL3	53/73	MTH, MTB
FAS14 = OU4 OL7	47	STS
+ OU1 OL5	15	STD
FAS15 = OU4 OL5	45	CS
+ OU4 OLA	4A	LS

(Continued)

Table 3-21. Family Code Logic (Cont.)

Families of Instructions			Hexadecimal Opcodes	Instruction
FAS16	=	OU1 O4 NO5 O6	1A/1B	LCD, LAD
		+ OU1 NO5 O6 NO7	12/1A	LD, LCD
FAS17	=	OU0 OL2	02	LCFI
		+ OU7 OL0	70	LCF
FAS18	=	OU7 NO4 O5 NO6	74/75	STCF, STB
		+ O2 O3 OL5	35/75	STW, STB
		+ O1 O3 OL5	55/75	STH, STB
FAS19	=	OU1 O4 NO5 O6	1A/1B	LCD, LAD
FAS21	=	FAS10		
		+ FAS23		
FAS22	=	OU1 NO5 NO6	10/11, 18/19	AD, CD, SD, CLM
		+ NO1 O3 OL9	19/39	CLM, CLR
FAS23	=	OU5 NO5 O6 NO7	52/5A	LH, LCW
		+ O1 O3 OL2	52/72	LH, LB
		+ NO1 O2 OL2	22/32	LI, LW
		+ OU3 NO5 O6 NO7	32/3A	LW, LCW
FAS24	=	OU6 OL6	66	AWM
		+ O2 O3 OL3	33/73	MTW, MTB
		+ O1 O3 OL3	53/73	MTH, MTB
FAS26	=	OU3 O4 NO5 NO6	3A/3B	LCW, LAW
FASH	=	OU2 NO4 O5 NO6	24/25	SF, S
FASHFL	=	FASH NO7	24	SF
FASHFX	=	FASH O7	25	S
FAST/1	=	OU0 O4 NO5	08→0B	PLW, PSW, PLM, PSM
		+ OU1 OL3	13	MSP
		+ OU2 OLA	2A	LM
		+ OU2 OLB	2B	STM
FAW	=	OU3	30→3F	
		+ NO3 NO4 O5	04→07, 24→27, 44→47, 64→67	
		+ O1 NO3 O4	48→4F, 68→6F	
		+ O2 NO3 O4	28→2F, 68→6F	

3-151 Logic Simplification

The logic equations used in this section are a simplified form of the complete logic equations and are not representative of the actual logic mechanization. For example, the logic to set flip-flop D16 for a Load Halfword instruction may be shown as the following:

S/D16 = C16 DXC + CO DXCR16 + . . .

R/D16 = DX

C/D16 = CL-32P19

DXC = DXC/5 P32

DXC/5 = OU5 (NO4 NO5) PH1 + . . .

DX = DXC/5

The actual logic equation to set flip-flop D16 as contained in the automated logic equations is as follows:

```
Line No.
1161320   S/D16  =  *+ .C16 .DXC/12
                    +.C17 .DXCL1-2 . NC16 .DXNC/12
1161330              + . DIO16 . DXDIO-2
                     +.KS16 .DXK-4
                     + .D16 .KNC16 .DXK-5
1161340              + .C0 .DXCR16-2
                     + .NC0 .DXNCR16-2
1161350   R/D16  =  DX-2
1161360   C/D16  =  CL-32P19
```

The particular instruction using this equation requires bit C0 or C16 to be clocked into flip-flop D16 as determined by flip-flop P32. The above logic equation, therefore, is simplified as follows·

```
          S/D16  =  C16 DXC/12 + C0 DXCR16-2 + . . .
          R/D16  =  DX-2
          C/D16  =  CL-32P19
```

The logic equation for term DXC/12 as contained in the automated logic equations is as follows:

```
Line No.
3908200   DXC/12  =  I .NDXC/7 .NDXCM .NDXC/D
```

Since signal DXC/12 is the output of an inverter, one of the input terms to the inverter must be false to make the output of the inverter true. The logic for the first term for signal DXC/12 is as follows:

```
Line No.
3907690   NDXC/7 = B .NDXC/2 .NDXC/6
                   .NDXC/4 .NDXC/3
```

If one of the terms in the AND gate for signal NDXC/7 is false, signal NDXC/7 is false, thereby making DXC/12 true. The logic for term NDXC/2 is as follows:

```
Line No.
3907190   NDXC/2 = I *.DXC/5 .P32
```

The instruction being discussed uses flip-flop P32 to indicate which half of the data word is used. The logic for signal NDXC/2 contains P32 true; therefore, only term DXC/5 is required to make signal NDXC/2 low. The logic for signal DXC/5 is as follows:

```
Line No.
3907401   DXC/5  =  B *+ .OU5 . (.NO4 .NO5) .PH1
3907402                +.OU5 . (.NO5 .O6 .O7) .PH1
```

```
Line No.
3907404          =     + .OU5 . (.NO4 .O5 .O6) .PH1
```

The particular instruction being discussed has an opcode of 52. Therefore, signal DXC/5 is true because expression OU5 (NO4 NO5) PH1 is true. With signal DXC/5 true and signal P32 true, signal NDXC/2 is low making signal NDXC/7 low. With signal NDXC/7 low, signal DXC/12 is high, thereby setting flip-flop D16 at the PH1 clock if bit C16 is a one. Eliminating the buffers and inverters contained in the actual logic mechanization, flip-flop D16 sets if bit C16 is a one, signal DXC/5 is true, and flip-flop P32 is set.

The reset term DX must also be true to reset flip-flop D16 if bit C16 is not a one. The logic for term DX-2, as contained in the automated logic equations, is as follows:

```
Line No.
3902200   DX-2  =  I .NDX
```

The logic for term NDX is as follows:

```
Line No.
3902610   NDX  =  B .NDXPARITY .NENDE .NDX/1
3902614           B .NDXPSW1 .NDXPSW2 .NDXCC
3902618           B . NDXDIO . NDX . CBP . NDXDU
3902622           B .NDXNCR16 .NDXNC/1
                  .NDXNC/2 .NDXNC
3902626           B .NDXCR16 .NDXCR16/1
                  .NDXCR24 .NDXCR8 .NDXC/9
3902630           B .NDXC/1 .NDXC/2 .NDXC/3
                  .NDXC/4 .NDXC/6
```

Term NDXC/2 is low. The output of the buffer is low, and signal NDX is low; therefore, signal DX-2 is high and flip-flop D16 is reset, if the set input is not true. The simplified logic equation for signal DX on the reset input to flip-flop D15 is indicated as:

```
          DX     =  DXC/5
```

The complete simplified logic equation for flip-flop D16 for a Load Halfword instruction with flip-flop P32 true is as follows:

```
          S/D16    =  C16 DXC + . . .
          R/D16    =  DX
          C/D16    =  CL-32P19
          DXC      =  DXC/5 P32
            DXC/5  =  OU5 (NO4 NO5) PH1 + . . .
          DX       =  DXC/5
```

3-152 INSTRUCTION TIMING

3-153 Phases

All operations required for an instruction are performed during two types of phase sequences: a preparation sequence consisting of from one to four phases, PRE1 through PRE4, and an execution phase sequence consisting of from 1 to 15 phases, PH1 through PH15. The timing of each phase is determined by the instruction being performed and by the family to which the instruction belongs. The timing for the next phase is set at the clock of the phase sequence being performed. Figure 3-136 shows the sequence of phases and the signals that enable the progression or branching from one phase to another. The processor control panel, interrupt, and trap phases are discussed in the paragraphs on these operations.

3-154 Memory Request Timing

The CPU-memory interface signals are subject to time delays because of the internal delays of active circuits (cable drivers and receivers), back board wiring, distributed capacitances, loading conditions, cable length, and other factors. Typically, single direction delays in the CPU-memory interface signals can range from 50 to 80 nanoseconds. Cable length and loading characteristics contribute the major portion of interface signal delay. The interface signals that control communication between the CPU and the memory are listed in table 3-22.

The interface signal timing characteristics described in this section are typical and do not necessarily describe the actual timing of any particular Sigma 7 system because of the number of variables that can exist from one system to another. Since the operation between the CPU and the memory is asynchronous, the logic must take into account all possible worst cases. As a result, the clock enable signals are dependent upon certain prior conditions resulting from the CPU-memory signal interchange. A discussion of propagation delay may be found in XDS T-Series, Integrated Circuit Logic Modules, Description and Specifications, publication No. 645103.

Figure 3-137 describes the timing characteristics of memory port C (the CPU is always attached to memory port C) for a read-restore, a full write, and a partial write operation.

The major interface signals and the basic logic control terms with a brief description of each are listed in table 3-23. Logic diagrams of signals AHCL and RESTM, ARQ, /MQC/, MAA through MGG, OPRQ, RQ and RQC are shown in figures 3-138 through 3-144, respectively. Instruction decoding for some of the preparation sequence control signals is shown in table 3-24.

In addition to the delays because of cable length and other factors previously described, other types of delays must be considered when addressing memory. These delays are the result of one or more of the following conditions:

a. The memory map is in effect

b. Crossover exists

c. An instruction is indexed

d. An instruction is indirectly addressed

Each of these conditions require some action that is not normally performed when the condition is nonexistent; therefore, time must be allowed to carry out the required action. In most cases, the extra delay time is accounted for by the memory request delay line (MRDL) in the CPU and its associated circuitry. See figure 3-140.

A delay occurs in the CPU when a memory read or write cycle has not been completed when the next clock should normally occur. This condition can occur when interleaving is either nonexistent or not in effect, or when two locations in the same memory bank are sequentially addressed. This delay is inserted by keeping the CPU in its current phase by withholding the clock enable signal (CE) until the memory cycle (including restoration of the data) has been completed.

Because of the possible delays described above, the time required to perform an instruction can vary. Four examples of memory access timing are shown in figures 3-145 through 3-148. These timing diagrams illustrate the functions of the several interface terms, and show the total operation time conditions that inject delays into the execution. The diagrams are presented to show the interaction of the major control terms that accomplish a successful memory access under the prescribed conditions.

If memory banks are not interleaved, or if two successive memory requests are made from the same memory bank, the second memory access is forced to wait until after the data restoration cycle of the first access has been accomplished. This noninterleaved condition requires an additional execution delay time of approximately 390 to 400 nanoseconds for each access to memory.

An indexed instruction also injects a delay into the execution time of the instruction. This delay is caused by the extra time required to fetch the index register, to add its contents to the virtual operand address, and to place the sum into the P-register where it is then placed onto the LB lines. (See figure 3-147.)

An indirectly addressed instruction requires two extra preparation phases to fetch the operand from the indirectly addressed word location.

When the operand address in the reference field of the instruction is X'0000F' or less, a crossover situation exists in which the private memory address lines (LR) receive their address from the core memory address lines (LB). See figure 3-146.

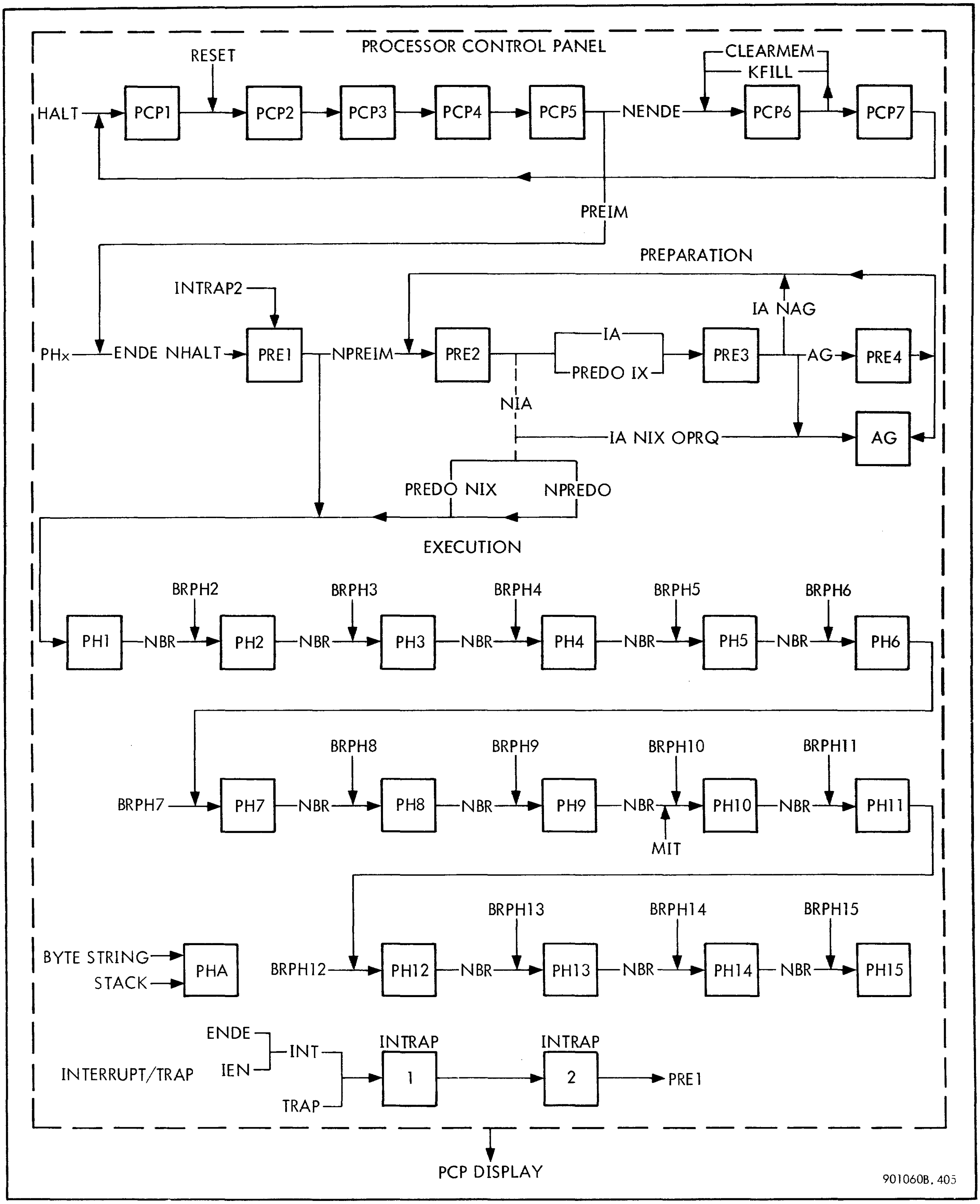


Figure 3-136. Phase Control

Table 3-22. CPU/Memory Interface Signals

Port C	M ⟶ CPU	CPU ⟶ M	Function
/MC00/ to /MC31/	X	X	32 data lines, two way
/LC15/ to /LC31/		X	17 address lines
/MW0C/ to /MW3C/		X	4 write byte lines
/MR/		X	Memory reset
/ORSP/		X	Override slow port
/ORIL/		X	Override interleave
/HOF/		X	Halt on fault
/ABOC/		X	Abort (change write to read)
/MQC/		X	Memory request
/AHC/	X		Address here signal
/ARC/	X		Address release signal
/DRC/	X		Data release signal
/PEC/	X		Parity error signal
/POKC/	X		Parity OK
/EDRC/	X		Early data release signal
/SRAC/	X		Second request allowed signal
/DGC/	X		Data gate signal
-	X		Parity OK signal
/MFL0/ to /MFL7/	X		Memory fault light signals
/MNN/	X		Margins not normal
/MFR/		X	Memory fault reset
/ST/	X		Start

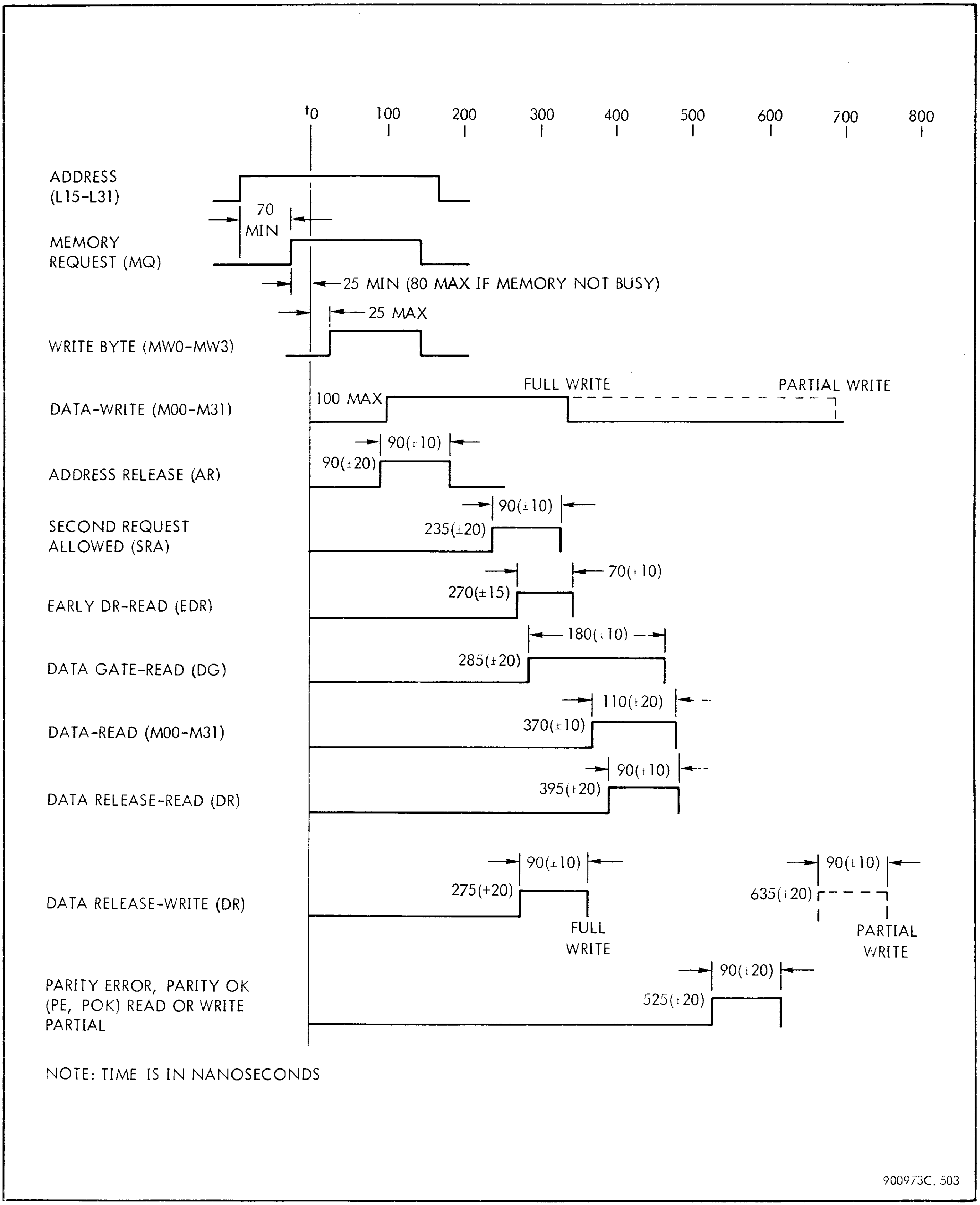


Figure 3-137. Memory Timing for Port C (Typical)

Table 3-23. Memory Request Timing and Control Signals

Signal	Definition
/ABOC/	An interface signal sent to memory when writing into memory is denied because of memory map or memory protection lockout. This signal is generated in the buffer amplifier PROTECTD in the CPU. For detailed logic, refer to the discussions of memory map and memory protection contained in this section
/AHC/	Address here - a signal generated in memory port C after the memory address has been verified and accepted. This signal becomes AH in the CPU after the cable receiver (figure 3-138)
AHCL	Address here clock - a signal generated in the CPU to enable the clock (figure 3-139)
/ARC/	Address release signal from memory port C informing the CPU that the address has been accepted, and that a memory cycle has started for the CPU. This signal becomes AR in the CPU after the cable receiver
ARQ	A flip-flop in the CPU which, if true, indicates that the next CL clock cannot occur until the address release signal (AR) has been received from memory if a memory request is pending (figure 3-139)
(ATE NAR)	A signal generated in the CPU designating that the address time has elapsed but that the address release signal has not arrived from memory (figure 3-140)
BRQ	A memory request flip-flop in the CPU that is set true whenever an interrupt or a trap occurs
/DGC/	Data gate signal from memory port C. Gates the MB (memory bus) lines into the CPU C-register
/DRC/	Data release signal from memory port C denoting that the data lines contain the requested memory data. Becomes signal DR in the CPU after the cable receiver
DRQ	A flip-flop in the CPU indicating, when true, that the next CL clock cannot occur until the data release signal (DR) has been received from memory if a memory request is pending
/EDRC/	Early data release signal from memory port C. Becomes signal EDR in the CPU after the cable receiver. This signal, when true, unlatches the C-register just before the new data from memory is placed in the C-register by signal DGC
/HOF/	A signal generated from the PCP when parity error mode switch is in HALT (KHOP). When this signal is true, the memory halts and remains busy at the end of the cycle in which a parity error has occurred
MAA through MGG	Buffer latches MAA, MBB, MCC, MDD, MEE, MFF, and MGG represent memory request states (see figure 3-141)
MAA	When signal MAA is true, no memory request is pending
MBB	When signal MBB is true, one memory request has been made, but the address release signal (AR) has not been received from memory
MCC	When signal MCC is true, one memory request has been made, and data release signal DR has not been received from memory

(Continued)

Table 3-23. Memory Request Timing and Control Signals (Cont.)

Signal	Definition
MDD	When signal MDD is true, a second memory request has been made before the DRQ clock for the first memory request
MEE	When signal MEE is true, the first memory request has been completed but a second request is pending
MFF	When signal MFF is true, one memory request is pending but DRQ CL has not occurred
MGG	When signal MGG is true, the last memory request has been completed
/MFL0/ through /MFL7/	Memory fault light signals from memory to PCP memory fault lights. When true, these signals indicate in which memory bank a parity error has occurred
/MNN/	Margins not normal signal from the memory logic power supply. True when the memory voltages are not within specified limits
MR	Memory request signal generated in the CPU. Becomes interface signal MQC to memory port C after cable driver (figure 3-140)
/MR/	A signal generated from the PCP when the CPU reset switch is pressed (KCPURESET)
MRD	When true, this signal injects a pulse into the memory request delay line MRDL (figure 3-140)
MRDL	Memory request delay line in the CPU (figure 3-140). Delays setting of memory request signal MR depending upon such conditions as crossover, map in effect, indirect addressing, and indexing
MRQ	A signal generated in the CPU requesting a memory cycle with the memory address specified by the P-register
MRQ/1	A signal similar to MRQ with the address specified by the Q-register. A memory request is initiated with either MRQ or MRQ/1 (MRQ + MRQ/1)
NPREFLL	A signal generated in the CPU that is always true except during floating long instructions
OPRQ	A latch set true in PRE1 by either signal PREOPRQ/1 or PREOPRQ/2, and held true during the preparation phases (figure 3-142). Indicates that an operand is required
/ORIL/	A signal generated from the PCP when the INTERLEAVE SELECT switch is in DIAGNOSTIC. When this signal is true, the interleaving feature is not in effect
/PEC/	Parity error signal generated in memory port C when a parity error is detected during a read or partial write cycle
/POKC/	Parity OK signal generated in memory port C. True when no parity error is detected during a read or partial write cycle
PREDO	A signal generated in the CPU that is true for doubleword instructions

(Continued)

Table 3-23. Memory Request Timing and Control Signals (Cont.)

Signal	Definition
PREFADO	A signal generated in the CPU that is true for doubleword instructions
PREIM	A buffered gate true for all immediate operand type of instructions (table 3-24)
PREOPRQ/1 and PREOPRQ/2	Buffered gates that are decoded from the C-register during preparation state PRE1 to set OPRQ latch (table 3-24)
PRERQ	A gate true for those instructions in which flip-flop RQ must be set during the preparation phases PRE1, PRE2, or PRE4 (table 3-24). Indicates that the next instruction may be accessed early
RIDL	This signal, when true, denotes that a memory request is current in the memory request delay line (MRDL) (figure 3-140)
RIP	This signal, when true, denotes that a memory request is in progress (see figure 3-140)
RQ	A memory request flip-flop in the CPU normally set at the beginning of the last operand request of most instructions. If RQ is true when the address release signal (AR) associated with that operand request is received, another memory request is initiated with the Q-register specifying the address. Since the Q-register contains the address of the next instruction, signal RQ true causes the next instruction to be fetched and placed in the C-register (see figure 3-143)
RQC	RQC is a buffered gate in the CPU that initiates a memory request in either preparation states PRE1 or PRE4 when the memory address must be specified from the contents of the C-register (see figure 3-144)
/SRAC/	Second request allowed signal generated in memory port C. Becomes signal SRA in the CPU after the cable receiver. This signal is true before the memory data has been strobed and specifies that a second memory request can be made
SRAF	Second memory request latch set by SRA and reset by AR or RESETM
/ST/	This signal, when true, inhibits drive currents in the memory and resets the control flip-flops. Signal ST is true when power is first applied and remains true until memory voltages are stabilized. This signal also sets the CPU to initial conditions. When power fails, ST goes true when power falls below a preset level

Table 3-24. Instruction Decoding for PREOPRQ/1, PREOPRQ/2, PRERQ, PREFADO, PREDO, and PREIM

PREOPRQ/1	PREOPRQ/2	PRERQ	PREFADO	PREDO	PREIM	Code	Mnemonic	Instruction
					x	02	LCFI	Load Conditions and Floating Control Immediate
	x		x	x		10	AD	Add Doubleword
	x		x	x		11	CD	Compare Doubleword
	x		x	x		12	LD	Load Doubleword
	x					13	MSP	Modify Stack Pointer

(Continued)

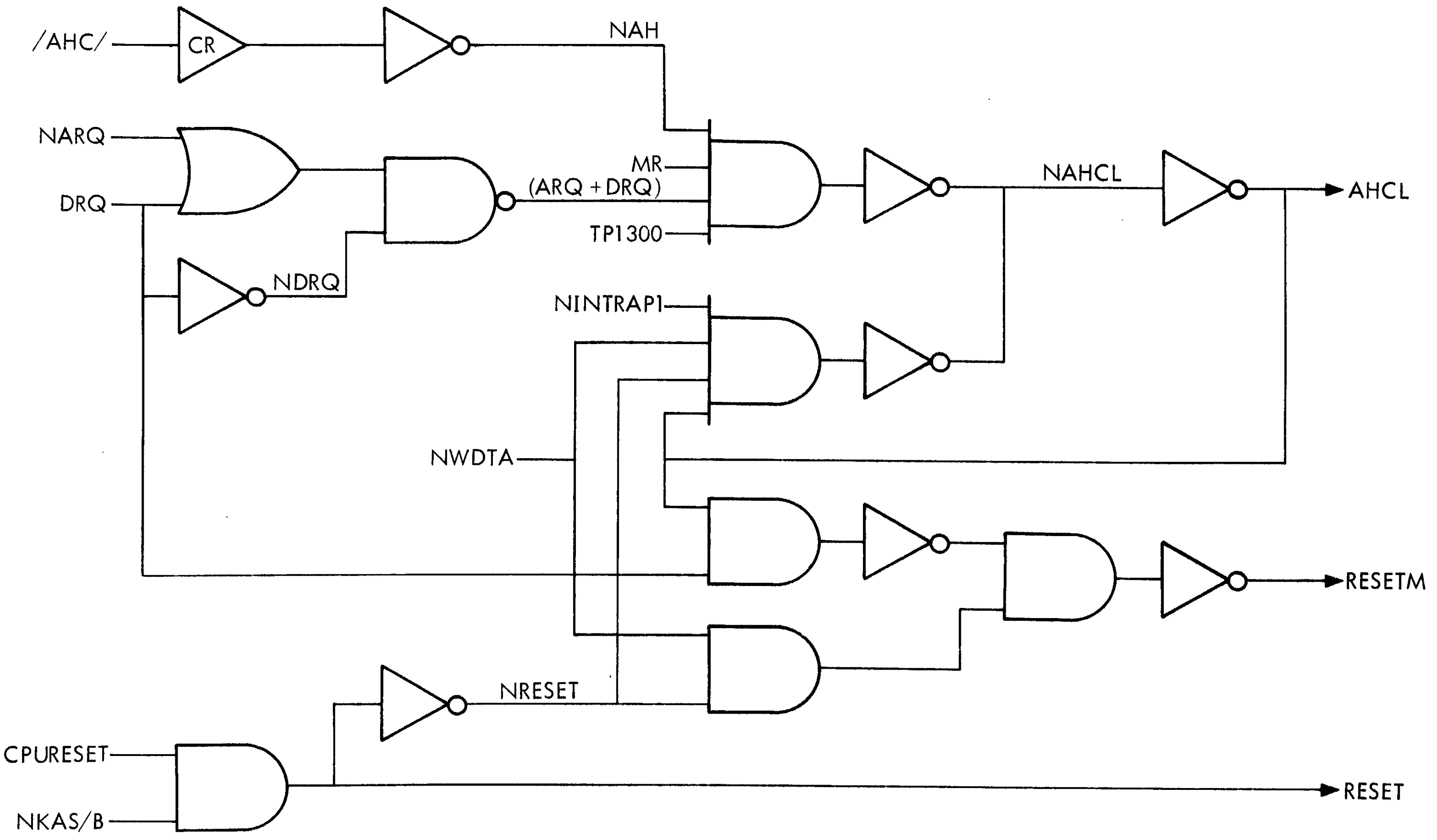
Table 3-24. Instruction Decoding for PREOPRQ/1, PREOPRQ/2, PRERQ, PREFADO, PREDO, and PREIM (Cont.)

PREOPRQ/1	PREOPRQ/2	PRERQ	PREFADO	PREDO	PREIM	Code	Mnemonic	Instruction
x			x	x		18	SD	Subtract Doubleword
x			x	x		19	CLM	Compare Limits in Memory
x			x	x		1A	LCD	Load Complement Doubleword
x			x	x		1B	LAD	Load Absolute Doubleword
x						1C	FSL	Floating Subtract Long
x						1D	FAL	Floating Add Long
x						1E	FDL	Floating Divide Long
x						1F	FML	Floating Multiply Long
					x	20	AI	Add Immediate
					x	22	LI	Load Immediate
		x				2F	LRP	Load Register Pointer
	x	x				30	AW	Add Word
x	x	x				31	CW	Compare Word
x	x	x				32	LW	Load Word
x						33	MTW	Modify and Test Word
x						36	DW	Divide Word
x						37	MW	Multiply Word
x		x				38	SW	Subtract Word
x		x				39	CLR	Compare with Limits in Register
x		x				3A	LCW	Load Complement Word
x		x				3B	LAW	Load Absolute Word
x						3C	FSS	Floating Subtract Short
x						3D	FAS	Floating Add Short
x						3E	FDS	Floating Divide Short
x						3F	FMS	Floating Multiply Short
	x					40	TTBS	Translate and Test Byte String

(Continued)

Table 3-24. Instruction Decoding for PREOPRQ/1, PREOPRQ/2, PRERQ, PREFADO, PREDO, and PREIM (Cont.)

PREOPRQ/1	PREOPRQ/2	PRERQ	PREFADO	PREDO	PREIM	Code	Mnemonic	Instruction
	x					41	TBS	Translate Byte String
	x					44	ANLZ	Analyze
	x					45	CS	Compare Selective
	x					46	XW	Exchange Word
	x					47	STS	Store Selective
x		x				48	EOR	Exclusive OR
x		x				49	OR	OR
x						4A	LS	Load Selective
x		x				4B	AND	AND
		x				50	AH	Add Halfword
		x				51	CH	Compare Halfword
		x				52	LH	Load Halfword
	x					53	MTH	Modify and Test Halfword
x		x				58	SH	Subtract Halfword
x		x				5A	LCH	Load Complement Halfword
x		x				5B	LAH	Load Absolute Halfword
					x	60	CBS	Compare Byte String
					x	61	MBS	Move Byte String
					x	63	EBS	Edit Byte String
	x					64	BDR	Branch on Decrementing Register
	x					65	BIR	Branch on Incrementing Register
	x					66	AWM	Add Word to Memory
	x					67	EXU	Execute
x						68	BCR	Branch Condition Reset
x						69	BCS	Branch Condition Set
x						6A	BAL	Branch Area Link
x						6B	INT	Interpret
	x					56	DH	Divide Halfword
	x					57	MH	Multiply Halfword
	x	x				70	LCF	Load Conditions and Floating Control
	x	x				71	CB	Compare Byte
	x	x				72	LB	Load Byte
	x					73	MTB	Modify and Test Byte



901060B. 3612

Figure 3-138. AHCL and RESTM, Logic Diagram

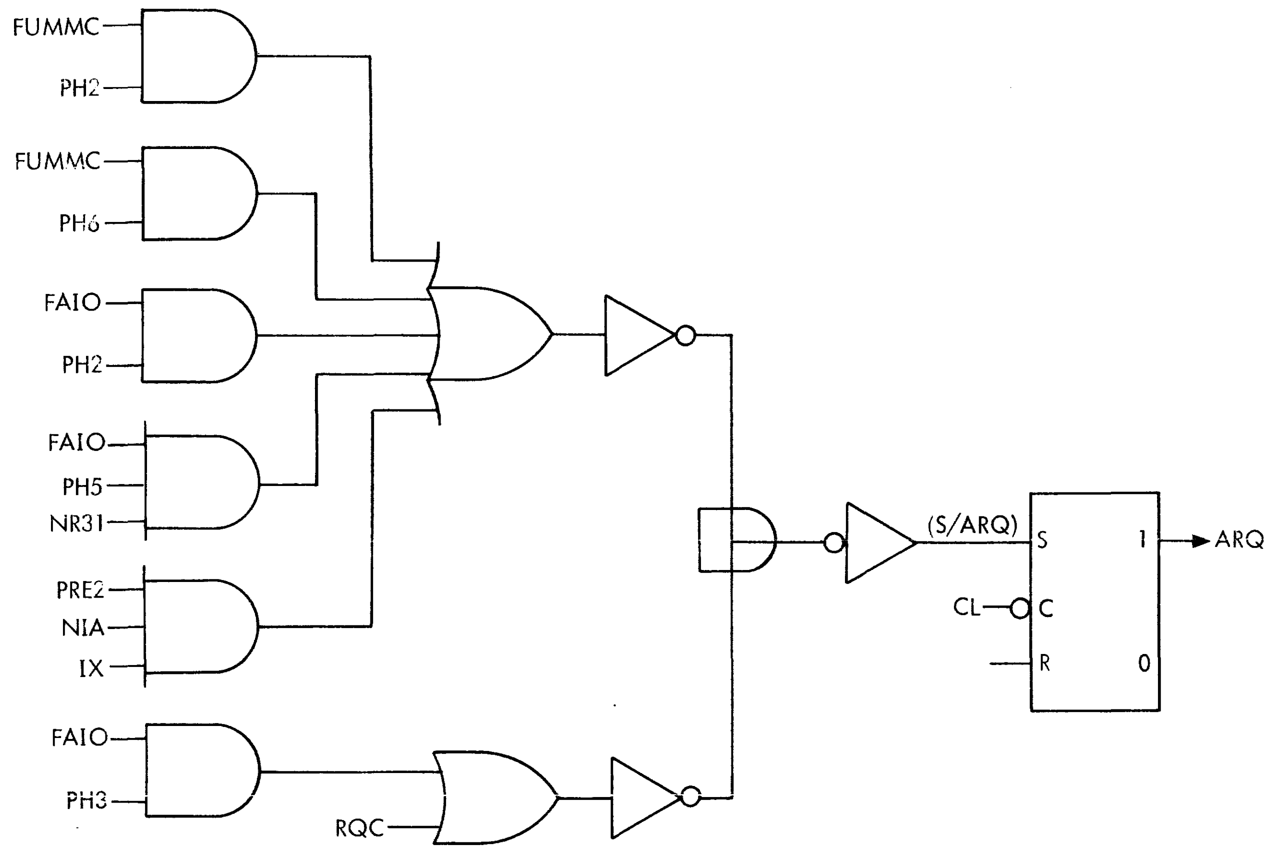


Figure 3-139. Address Release Request Signal (ARQ), Logic Diagram

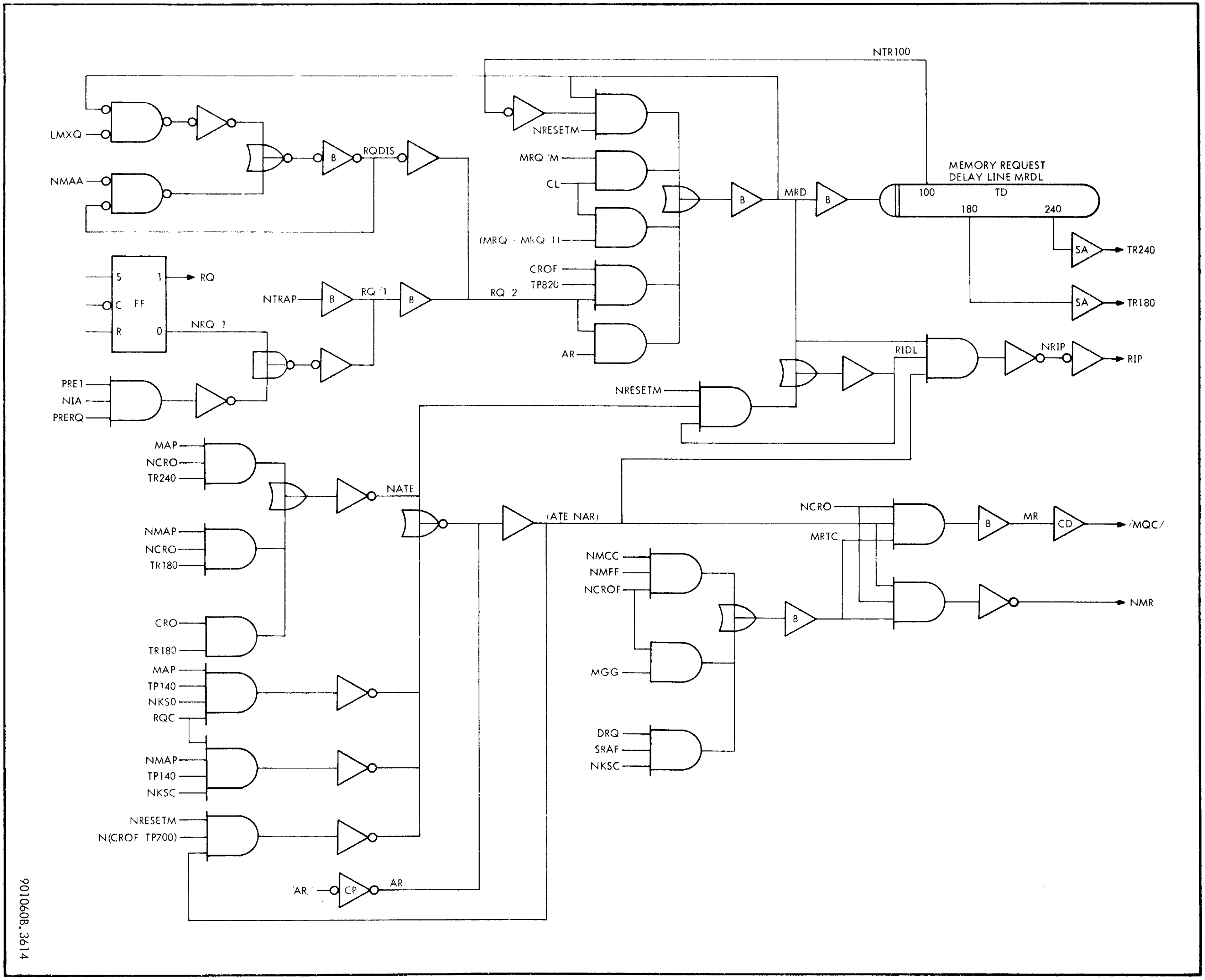


Figure 3-140. Memory Request Timing, Logic Diagram

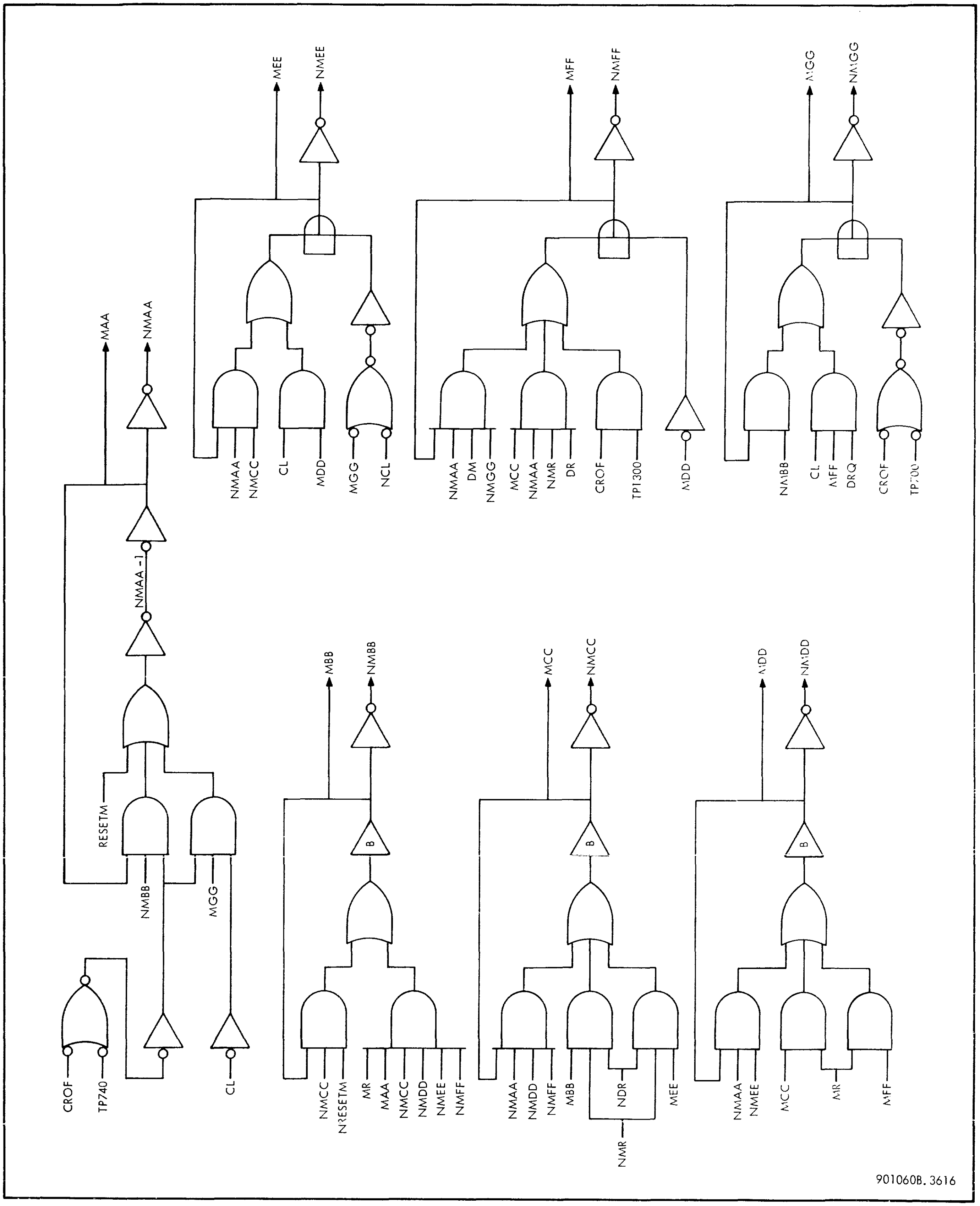


Figure 3-141. Memory Request Phase Flip-Flops, Simplified Logic Diagram

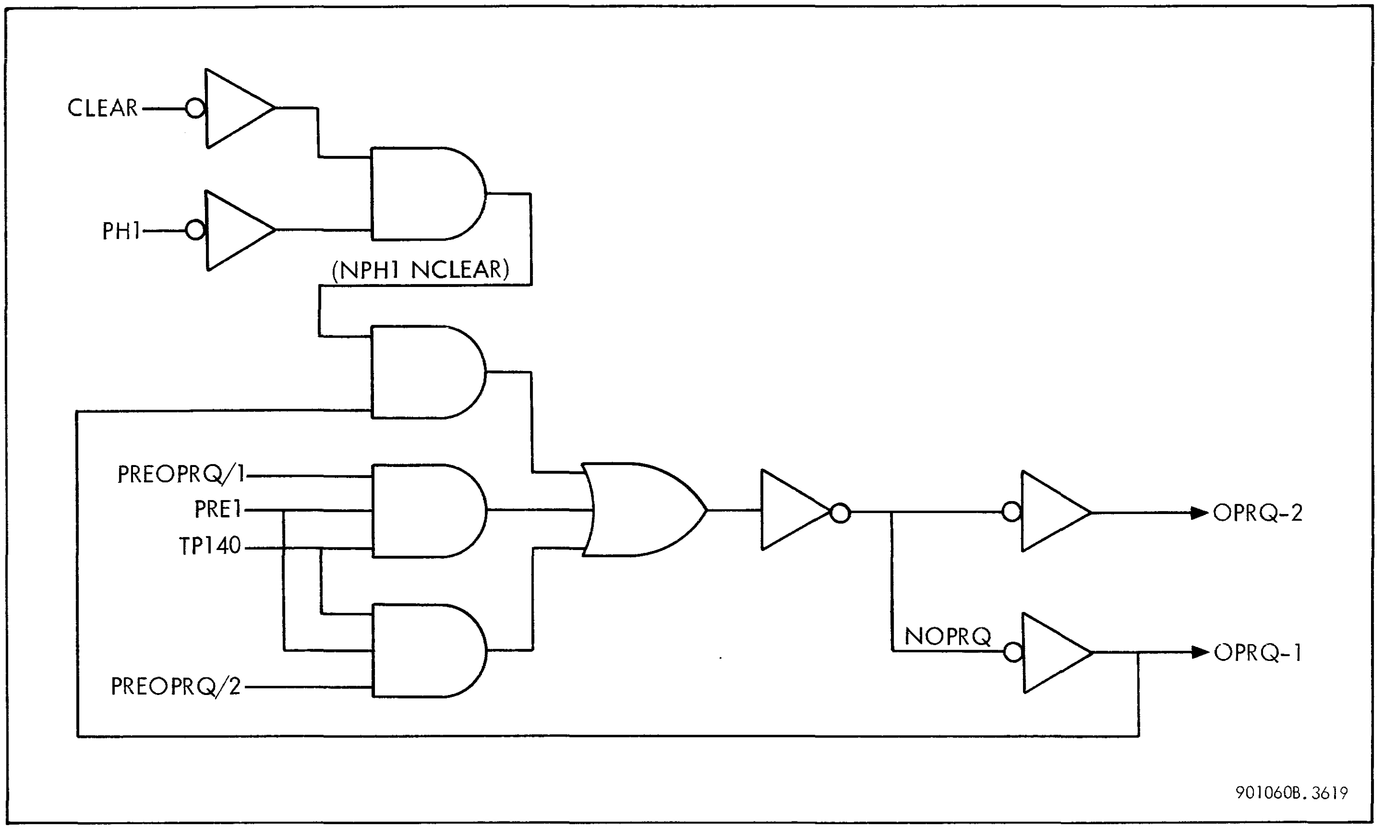


Figure 3-142. OPRQ Signal Gating, Simplified Logic Diagram

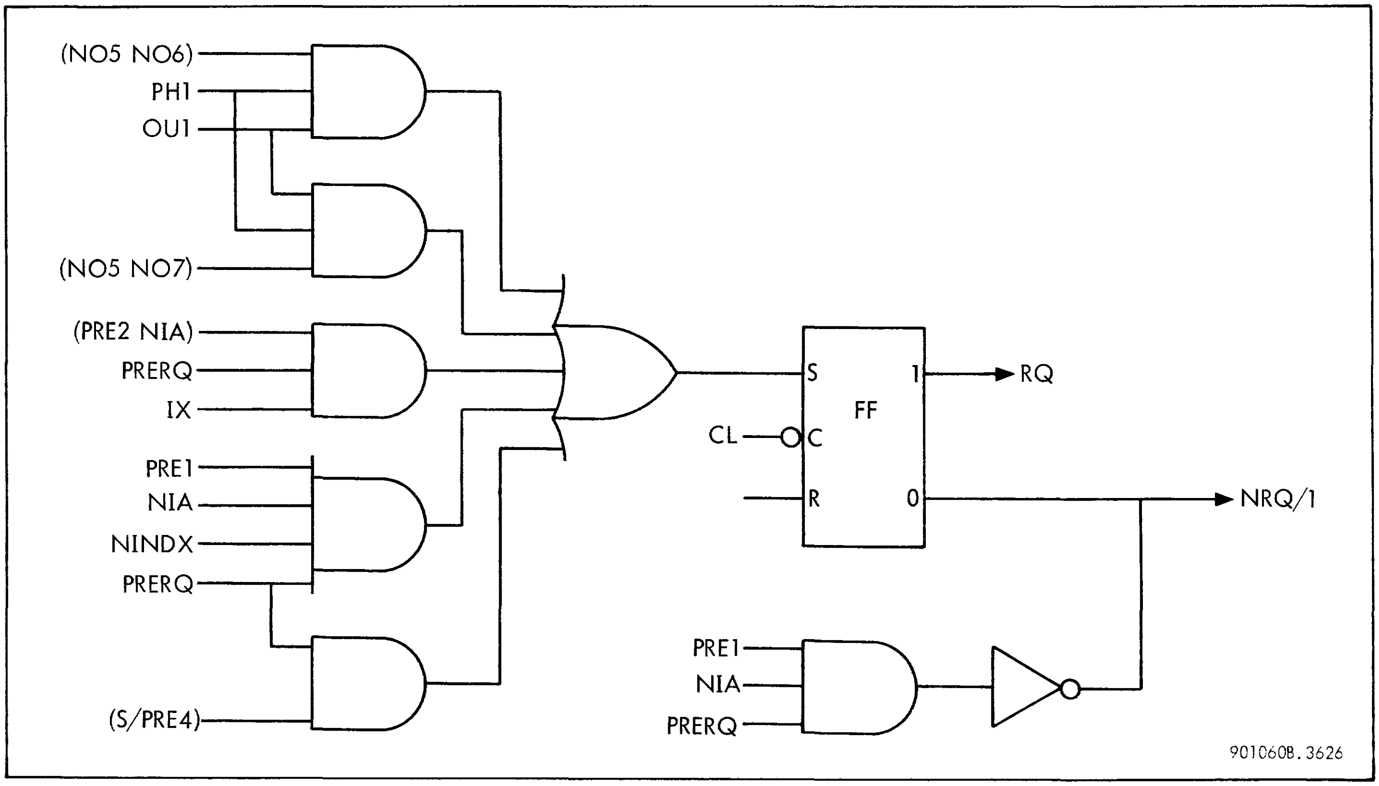


Figure 3-143. RQ Signal Gating, Simplified Logic Diagram

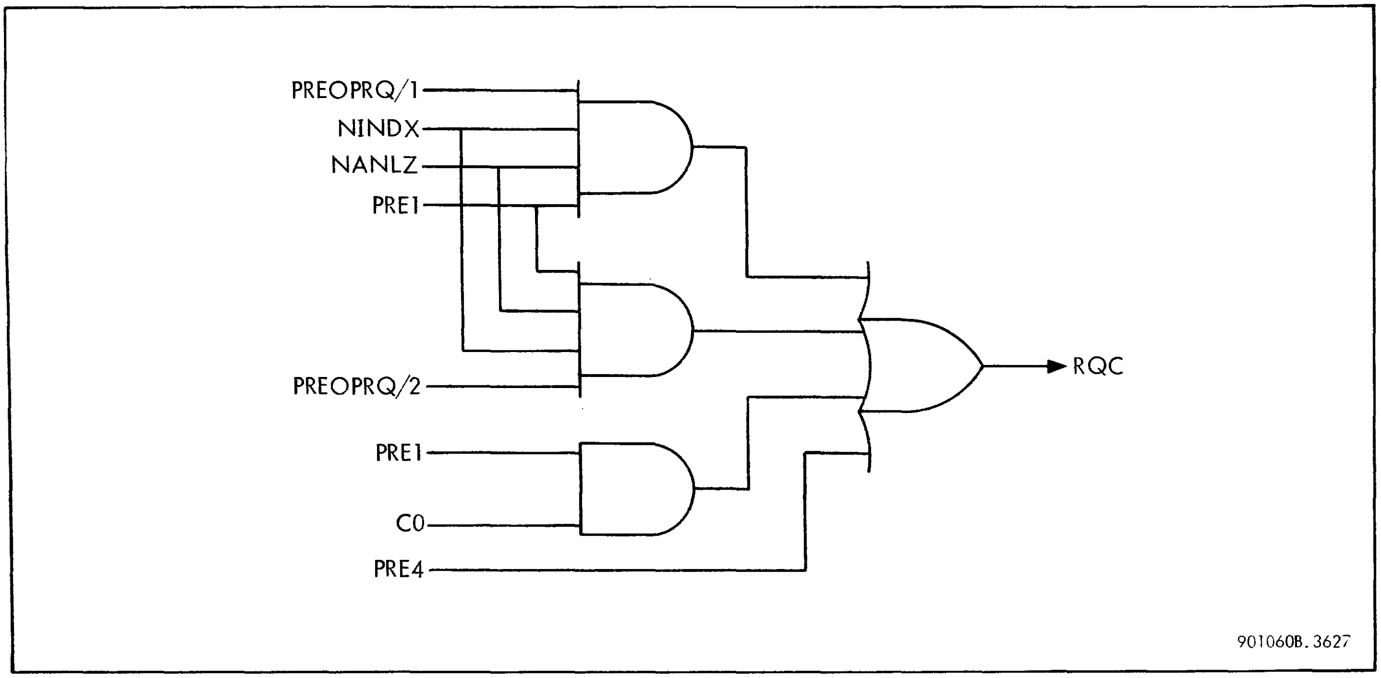


Figure 3-144. RQC Signal Gating, Simplified Logic Diagram

3-155 MEMORY ADDRESSING

3-156 Core Memory

Core memory is addressed by the address placed on the LB lines, LB15 through LB31. The LB lines receive memory addresses directly or indirectly from either the Q-register, the C-register, or the P-register. If the memory map is present and in effect (flip-flop MAP true), the eight most significant LB lines (LB15-LB22) receive their page addresses from the appropriate memory map register. Whether the LB address lines take their data from the C-, Q-, or P-registers depends on the status of the terms LMXC and LMXQ. See figure 3-149.

3-157 MEMORY ADDRESSING WITHOUT MAPPING. When the memory map option is either not present or not in effect (NMAP), the core memory address lines are affected in the following manner when LMXQ is true.

Q15-Q22 → LM15-LM22 → LB15-LB22

When the memory map option is either not present or not in effect (NMAP), the core memory address lines are affected in the following manner when LMXC is true.

C15-C22 → LM15-LM22 → LB15-LB22

C23-C30 → LB23-LB30

If (LMXC C0 NAG C31), 1 → LB31

If (LMXC AG PREDO), 1 → LB31

If (LMXC NPREFLL C0 C31), 1 → LB31

If (LMXC NC0 PREFADO), 1 → LB31

If (PRE3 NIA), 1 → LB31

If (LB31/1), 1 → LB31

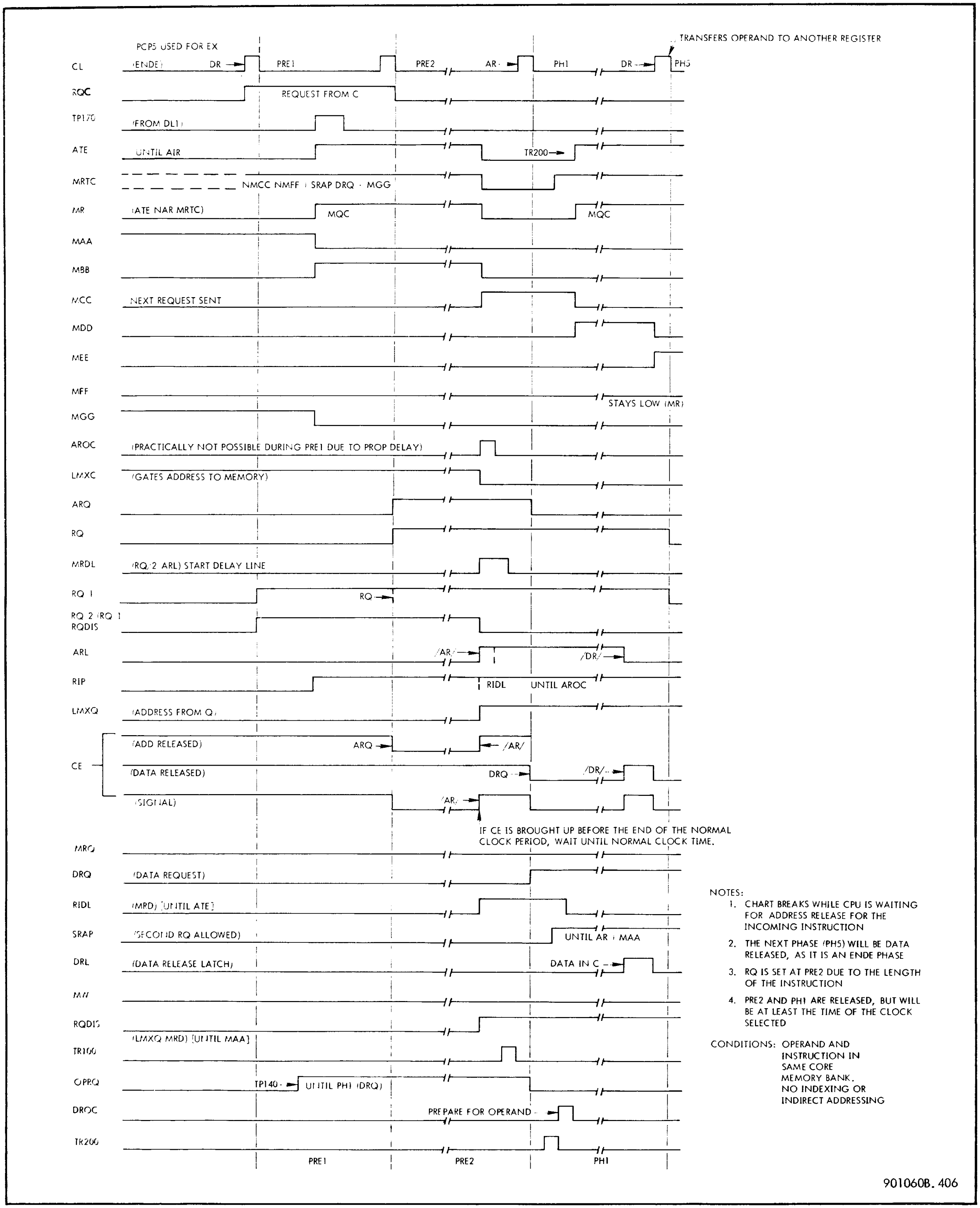
The low order address bit (LB31) is handled in a manner different from the high order bits (LB15-LB30). The LB31 address line represents the difference between an even numbered address and the next higher odd numbered address. For doubleword operations or for multiple word operations where successive odd even numbered locations are addressed, the flip-flop LB31/1 is toggled to provide the successive even odd numbered operand locations.

A memory request MRQ is initiated, and flip-flop ARQ is set inhibiting transmission of another clock until address release signal AR is received. Flip-flop RQ is set indicating that another memory request can be generated as soon as the operand is received with the address specified in the Q-register. Flip-flop PH1 is set, and clock T4R1 is enabled which indicates the first of the execution phases of the instruction.

If the memory map option is not present, or is not in effect (NMAP), the core memory address lines are affected in the following manner when neither LMXQ nor LMXC is true (NLMXQ NLMXC):

P15-P22 → LM15-LM22 → LB15-LB22

P23-P31 → LB23-LB31

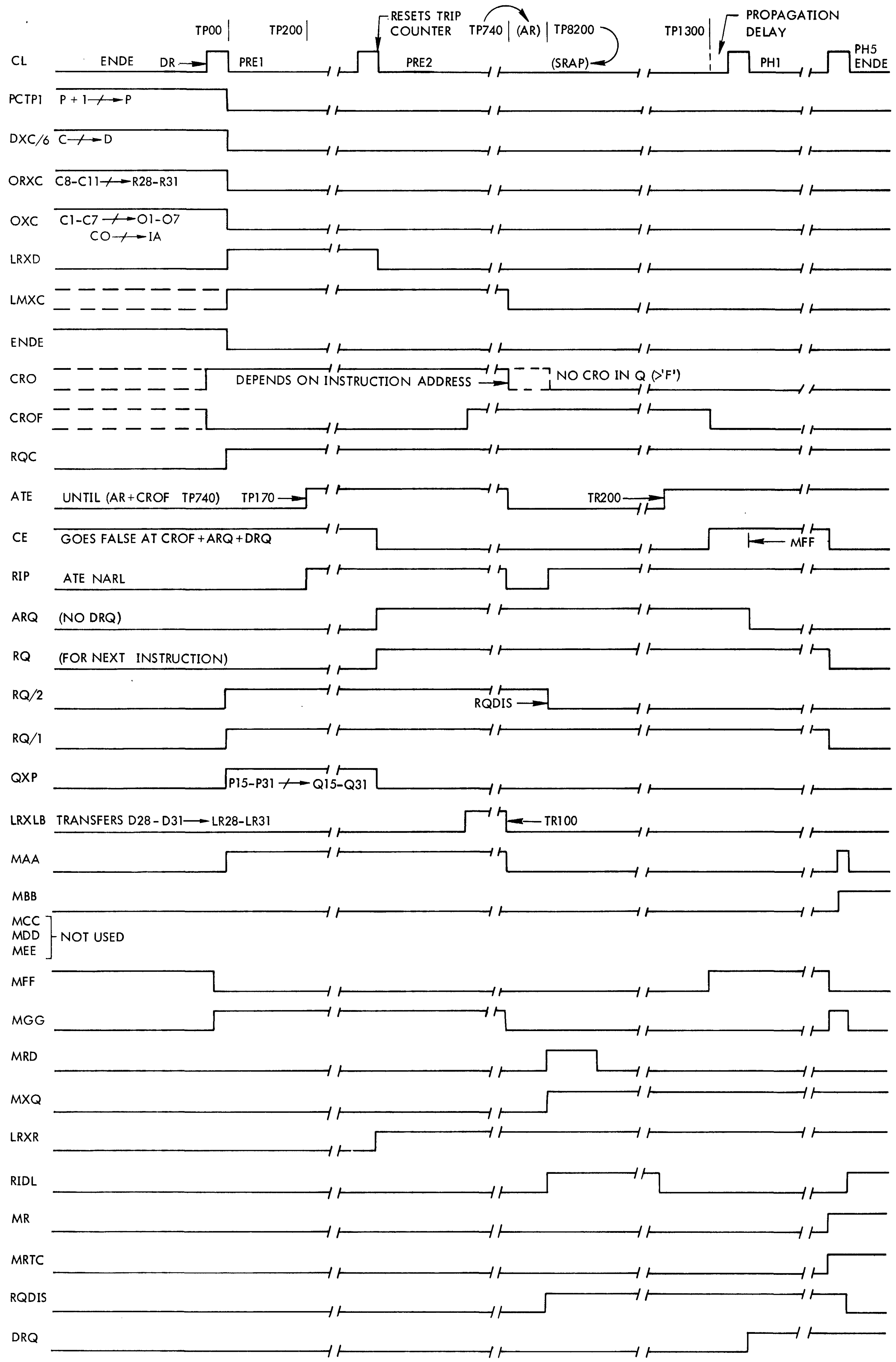


NOTES:

1. CHART BREAKS WHILE CPU IS WAITING FOR ADDRESS RELEASE FOR THE INCOMING INSTRUCTION
2. THE NEXT PHASE (PH5) WILL BE DATA RELEASED, AS IT IS AN ENDE PHASE
3. RQ IS SET AT PRE2 DUE TO THE LENGTH OF THE INSTRUCTION
4. PRE2 AND PH1 ARE RELEASED, BUT WILL BE AT LEAST THE TIME OF THE CLOCK SELECTED

CONDITIONS: OPERAND AND INSTRUCTION IN SAME CORE MEMORY BANK.
NO INDEXING OR INDIRECT ADDRESSING

Figure 3-145. Request from Port C During Load Word Instruction, Timing Diagram (Typical)



PERTINENT EQUATIONS

CRO = NLM15-NLB27 NCLEARMEM (ADDRESS < X'10')

CROF = LRXLB+CROF NMFF NRESETM

LRXLB = CRO ATE AR MAA GARQDRQ NTP740

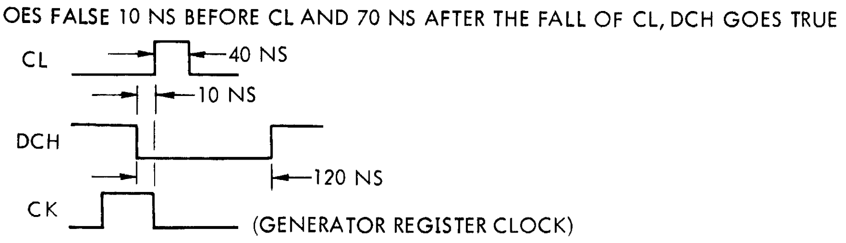
GARQDRQ = (ARQ+DRQ) TP170+GARQDRQ NCL

MFF = CROF TP1300

CXRR/0 = CXRR/D (NHOLDC-0+LRXLB) NIA (TRANSFERS GENERAL REGISTER TO C)

HOLD-C = NCX1 NEDR (NDCH+NCXS NCXRR) (BREAKS LATCH ON C-REGISTER)

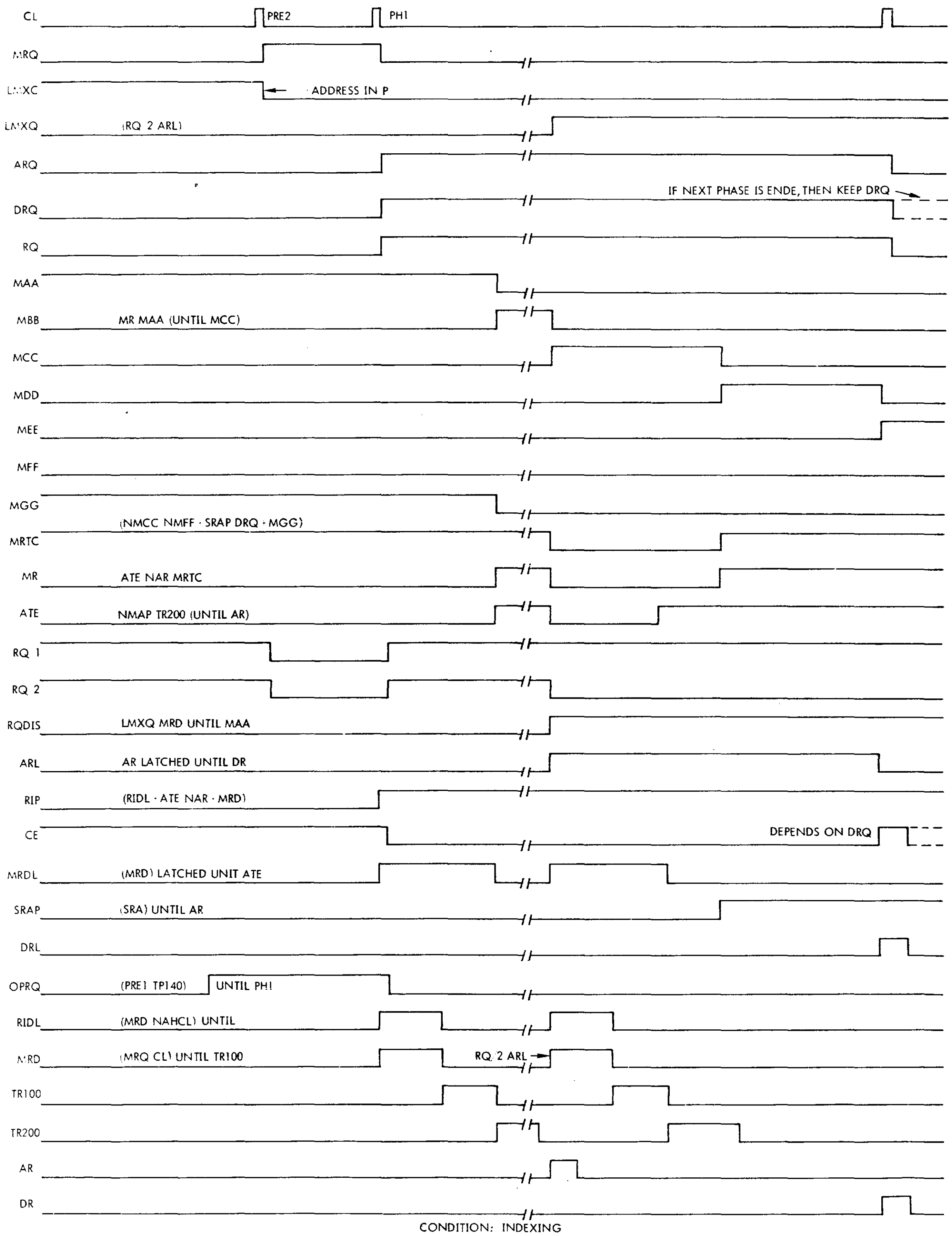
DCH = DCH GOES FALSE 10 NS BEFORE CL AND 70 NS AFTER THE FALL OF CL, DCH GOES TRUE



CONDITIONS: NO
INDIRECT ADDRESSING, INDEXING, OR MAPPING

Figure 3-146. Memory Request During Load Word Instruction With Crossover Timing Diagram (Typical)

901060B. 407



CONDITION: INDEXING

NOTES:

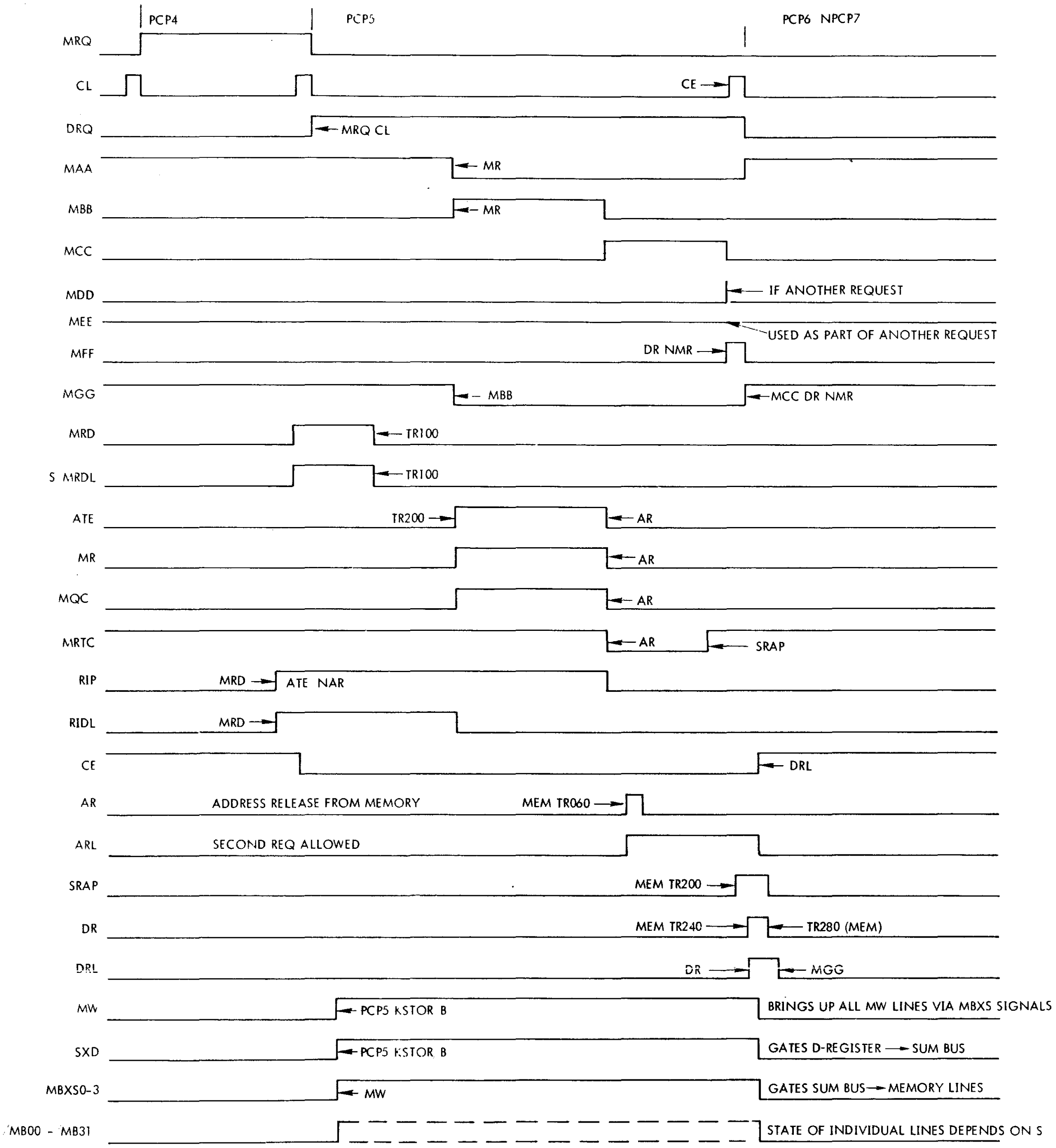
1. THE END OF THE TIMING DIAGRAM REPRESENTS WAITING FOR ADDRESS RELEASE, FOR THE NEXT INSTRUCTION

2. LMXQ APPLIES THE ADDRESS IN Q(IA) TO THE ADDRESS LINES AT ADDRESS RELEASE FOR THE OPERAND

3. THE NEXT M PHASE WOULD BE MCC AT THE RECEIPT OF AR FOR THE INSTRUCTION ACCESS

901060B.408

Figure 3-147. Load Word Instruction, Request from P-Register Followed by Request from Q-Register, Timing Diagram (Typical)



PERTINENT EQUATIONS

MRQ = PCP (PCP KSTOR B)
S DRQ = PCP4 KSTOR B
MAA = (MGG CL + MAA NMBB)(NCROF + NTP740)
MBB = NMDD NMEE (MR MAA NMCC NMFF + MBB NMCC NRESETM)
MDD = (MCC + MFF) MR + MDD NMEE NMAA
MCC = (MBB + MEE) MR NDR + MCC NMDD NMFF NMAA
MEE = [(NMGG + CL) (NMDD CL + MEE NMCC NMAA)]

MFF = MDD (MCC DR NMR NMAA + CROF TP1300 + MFF NMGG NMAA)
MGG = (NCROF + NTP740) (MFF DRQ CL + MGG NMBB)
MRD = (MRQ CL) + (MRD NTR100 NRESETM)
S/MRDL = MRD NAHCL
ATE = NMAP CRO TR200 + ATE NAR RESETM (NCROF + NTP740)
MR = ATE AR NCRO MRTC
MRTC = NMCC NMFF NCROF + SRAP DRQ NKSC + MGG NCROF

901060B.409

Figure 3-148. Store Display Operation, Timing Diagram (Typical)

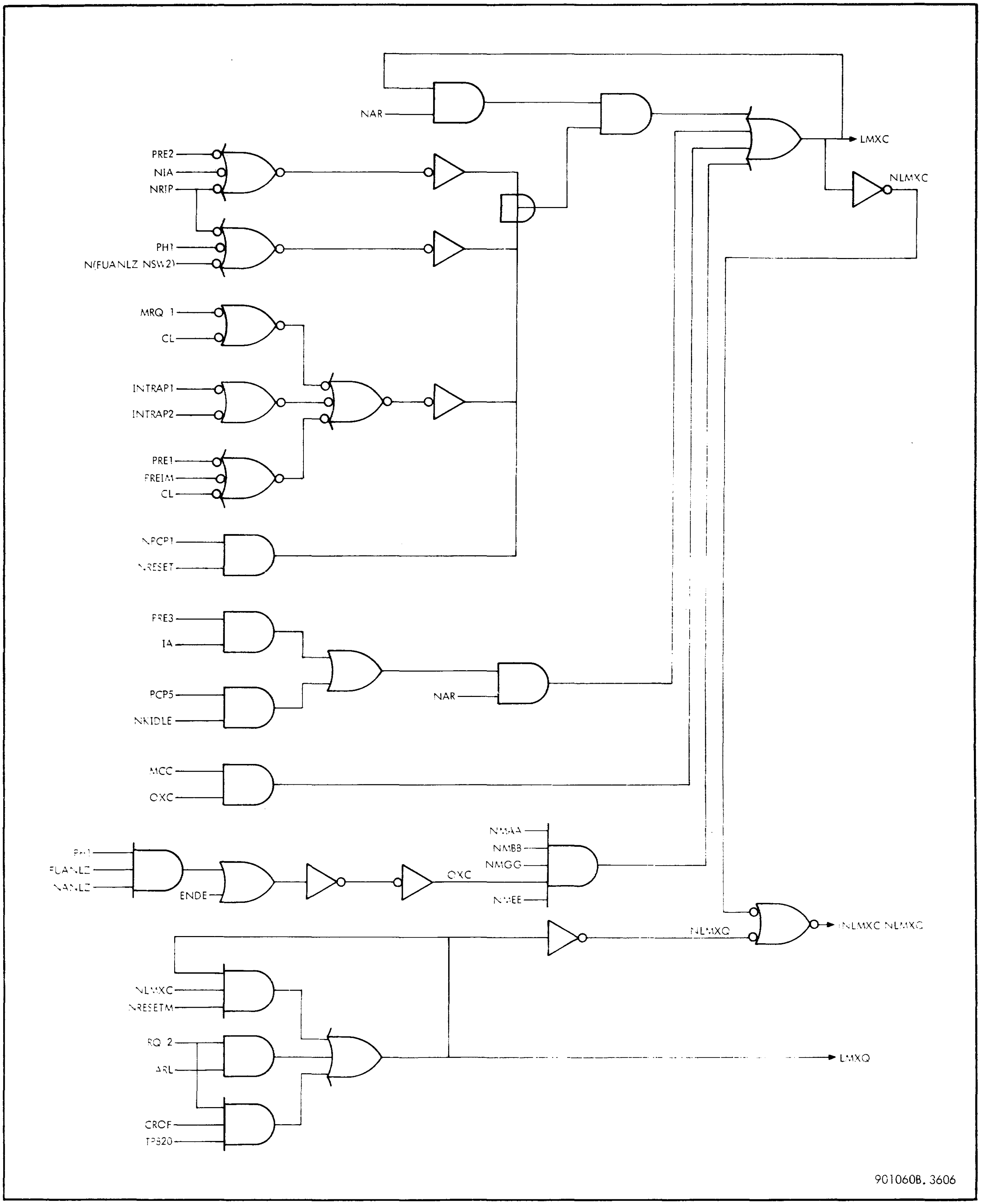


Figure 3-149. LMXC and LMXQ Signals Gating, Logic Diagram

3-158 MEMORY ADDRESSING WITH MAPPING. If the memory map option (MAP) is present and in effect, the eight high order LB lines do not accept the address from the LM lines as is the case when mapping is not in effect. Instead, the eight high order LM lines address a map register, and the information (page address) located in this addressed map register is placed on the LB lines, LB15-LB22. When the map mode is in effect, the eight high order LB lines are affected in the following manner.

MAP15/1 MAP15/2-MAP22/1 MAP22/2 →
LB15-LB22

Mapping does not affect the manner in which the LM lines, LM15-LM22, and the LB lines, LB23-LB31, are addressed.

The fast-access map addressing registers that address each LB line, LB15-LB22, are divided in two groups for each LB line, MAPn/1 and MAPn/2, where n 15 through 22. Either one or the other can be addressed at any time; both cannot be addressed simultaneously. Since the paralleled output of an unaddressed fast-access memory is always true, MAPn/1 is true when MAPn/2 is addressed, and MAPn/2 is true when MAPn/1 is addressed. Figure 3-150 shows how LB15 receives its data from the memory map high-speed memory elements. LB16 through LB22 operate in a similar manner. Refer to the memory map detailed theory of operation in this section.

3-159 Private Memory

The private memory registers are addressed on lines LR23 through LR31 which go to cable drivers /LR23/ through /LR31/ and then through cables to the private memory fast-access registers. Address lines /LR23/ through /LR27/ receive their inputs directly from the register pointer (RP) register. These lines address one of 32 possible private memory banks. Address lines LR29 through LR31 address one of 16 registers within the private memory bank currently selected by the register pointer.

The individual private memory registers 0 through F of the current private memory bank are addressed under the following conditions:

a. Before reading the contents of the register designated by the R-field (bits 8 through 11) of the instruction word. This address is taken from the R-register when signal LRXR is true, and when crossover does not exist. When signal LRXR is true and signal LRXLB is false, the LR lines are addressed in the following manner.

RP23-RP27 → /LR23/-/LR27/

R28-R31 → LR28-LR31 → /LR28/-/LR31/

b. Before reading the contents of one of the index registers (1 through 7) during the preparation phases of an indexed instruction, halfword, or byte addressed instructions. This address is taken from the D-register (D12 through D14). When signal LRXD is true, the LR lines are addressed in the following manner.

RP23-RP27 → /LR23/-/LR27/

0 → LR28 → /LR28/

D12-D14 → LR29-LR31 → /LR29/-/LR31/

c. When a crossover condition exists while addressing either an operand or an instruction (NLMXC). Crossover occurs when the address on the LB lines is X'0000F' or less (figure 3-151). When crossover occurs, the address on core memory address lines LB28-LB31 is transferred to private memory address lines LR28-LR31. The LR lines are addressed

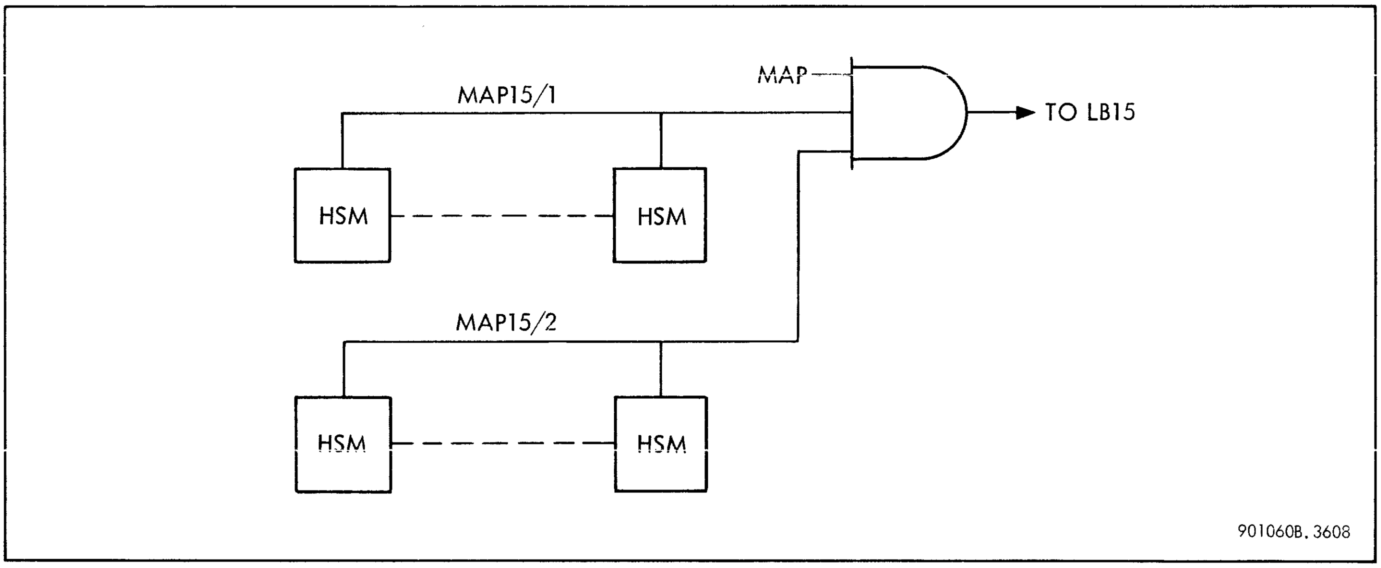


Figure 3-150. Fast-Access Memory Map Output Distribution

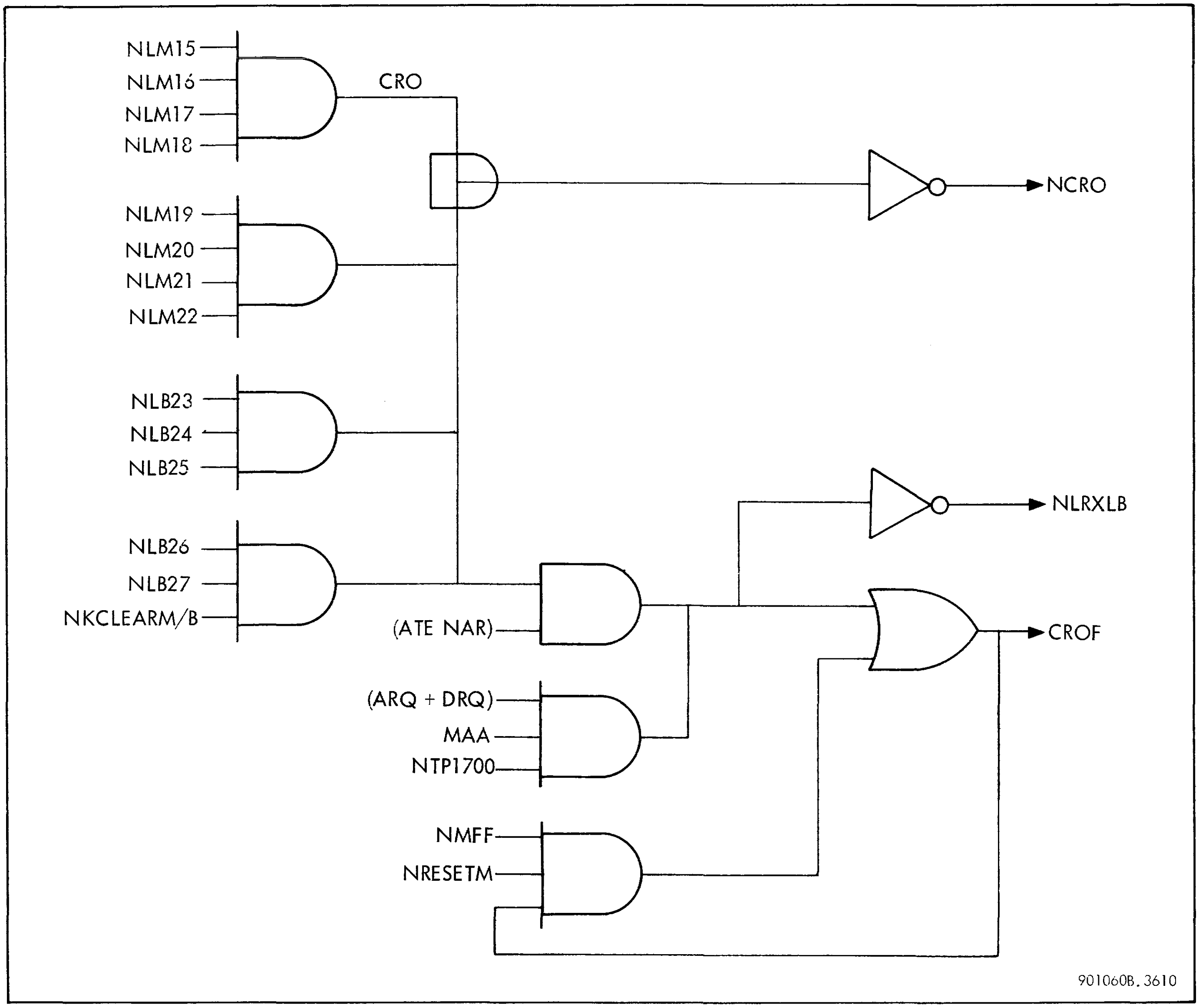


Figure 3-151. Crossover, Logic Diagram

in the following manner, when the term (LRXLB NLMXC) is true.

RP23-RP27 → /LR23/-/LR27/

LB28-LB31 → LR28-LR31 → /LR28/-/LR31/

d. When a crossover condition exists while addressing an operand. The LR lines are addressed in the following manner when signal LMXC is true and signal LRXLB is true.

RP23-RP27 → /LR23/-/LR27/

D28-D31 → LR28-LR31 → /LR28/-/LR31/

When an address for core memory is taken from the contents of the C-register, and the address causes crossover from core memory to private memory, the private memory address is taken from the D-register which contains the same information as the C-register.

Flip-flop LR31/2 forces a one into the least significant private memory register line LR31 for doubleword and for some multioperand instructions to provide the successive odd even, even odd register addresses required for these instructions. Input logic for flip-flop LR31/2 is described under each appropriate instruction.

When PCP switches CPU-RESET-CLEAR and SYS-RESET-CLEAR are pressed simultaneously, the general storage memory is cleared to zeros. During this memory clear operation, crossover cannot occur with an address less than X'0000F'; therefore, memory locations X'00000' through X'0000F' are cleared to zeros. The private memory registers are not affected. See figure 3-151 for the gating for CRO.

3-160 INSTRUCTION DESCRIPTION

3-161 Preparation Sequence

3-162 GENERAL PREPARATION. Nearly all instructions require the same preparation sequence. Sequence for those instructions requiring special preparation sequences is discussed in the applicable instruction description. General preparation sequences are covered in the following paragraphs.

During the last execution phase of an instruction, the next instruction address contained in the Q-register is gated into memory on memory address lines LB and LM (modified by MAP, if indicated). The instruction word contained in that memory address is loaded into the C-register.

From the C-register, the opcode (bits C1 through C7) is gated into bit positions 1 through 7 of the opcode register (O-register). The R-field (bits C8 through C11) is gated into bit positions R28 through R31 of the R-register; and, during the time when signal ENDE is true, the entire word, bits C0 through C31, of the C-register is gated into the D-register. The address of the desired operand, bits C15 through C31, is placed in the memory address lines LM15 through LM23 and LB24 through LB31 by signal LMXC. When the clock occurs, phase flip-flop PRE1 is set, clock T6L is enabled, and the address contained in the P-register (program register) is increased by one count.

When the PRE1 clock occurs, the content of the P-register is transferred into the Q-register. The Q-register then contains the address of the next instruction. During phase PRE1, signal RQC is generated for the instructions that require an operand during the preparation phase. Signal RQC generates memory request /MQC/ for data from the memory address placed on the LB and LM lines from the C-register. Registers A, B, and E are cleared; flip-flop ARQ is set which indicates that there is a memory address request that has not received response and that another clock is not be to transmitted until address-release signal AR is received from memory. Phase flip-flop PRE2 is set, and clock T4RL is enabled.

During phase PRE2, the contents of the A-, D-, and CS- (carry save) registers are added, and the result is placed on the sum bus. Since the A-register was cleared during phase PRE1, and the CS-register is not used during phase PRE2, both contain all zeros. Therefore, the data transferred to the sum bus is the contents of the D-register. If address-release signal AR has been received from memory, a PRE2 clock occurs, and bits S15 through S31 from the sum bus (address of the operand) are gated into the P-register. Registers A and D are cleared, and flip-flop DRQ is set. This inhibits transmission of another clock until data-release signal DR is received from memory if there is a memory request out. Phase 1 (PH1) flip-flop, the first of the execution phases, is set and clock T4RL is enabled. The execution phase of the instruction begins with phase PH1.

A sequence chart of the general preparation phases is shown in table 3-25.

3-163 INDIRECT ADDRESSING. Any Sigma instruction, except immediate and byte string instructions, can be indirectly addressed. The indirect addressing operation is limited to one level and is indicated by a one in bit position zero in the instruction word. Indirect addressing does not proceed to further levels regardless of the contents of bit position zero of the word addressed by the instruction address field.

Indirect addressing is performed during the preparation phases of the instruction. In indirect addressing, the 17-bit word address field of the instruction, referenced as the program address, is used to obtain a second word from memory. The 17-bit address field of the second word, referenced as the indirect program address, replaces the initial word address field and is used as the address to obtain the operand from memory. Because indirect addressing is the same for all instructions, the logical operation is being presented in the following paragraphs and will not be repeated in the individual instruction discussions.

During the execution phases of an instruction, the next instruction is read from memory and is gated into the C-register as described under general preparation phases. If bit C0 is a one, indicating indirect addressing, indirect address flip-flop IA is set at the following clock. The program address, bits C15 through C31, is gated onto the memory address lines by signal LMXC. The preparation phase one (PRE1) flip-flop is set at the clock pulse, and clock T6L is enabled.

Signal RQC is generated during phase PRE1 and causes a memory request with the address as specified by the C-register. When the PRE1 clock occurs, flip-flop ARQ is set. This inhibits transmission of another clock until address release signal AR is received from memory. Phase PRE2 flip-flop is set, and T4RL is enabled. When address release signal AR is received from memory, clock T4RL is allowed to occur. At clock T4RL, flip-flop DRQ is set. This inhibits transmission of another clock until a data release signal is received from memory. Repeater flip-flop AG is set, indicating the first pass through PRE2 for indirect addressing. Phase PRE3 is generated, and clock T4RL is enabled.

When data release signal DGC is received from memory, gating signal CXMB is generated and the C-register is enabled. As soon as the data is available on the MB lines from memory, signal CXMB gates the data into the C-register. Data release signal DR from memory enables clock PRE3 to occur. When the clock occurs, data from the C-register is transferred into the D-register, and bits C15 through C31 are loaded onto the LM and LB memory address lines by signal LMXC to obtain the operand from memory. Indirect address flip-flop IA is reset, repeater flip-flop AG is held set, phase PRE4 flip-flop is set, and clock T4RL is enabled.

Table 3-25. Preparation Sequence (No Indexing, No Indirect Addressing)

Phase	Function Performed	Signals Involved		Comments
ENDE (Last phase of previous instruction)	C1-C7 -/→ O1-O7	OXC	= ENDE	Opcode → O-register
	C8-C11 -/→ R28-R31	ORXC	= ENDE + . . .	R-field → R-register
	C0-C31 -/→ D0-D31	DXC	= ENDE + . . .	
	C15-C31 → LM15-LB31	LMXC	= OXC (. . .) + . . . (latched)	Program address for required operand
	Set phase PRE1 flip-flop	S/PRE1	= ENDE (NHALT + FUEXU) (S/INTRAPF) + . . .	Set at clock following ENDE
	Enable clock T6L for next phase	T6L	= NT1L NT4L NT8L NT10L NRESET	
	Add one to address in P-register	PCTP1	= ENDE + . . .	Address of next instruction
PRE1	P -/→ Q	QXP	= PRE1 NANLZ + . . .	Address of next instruction
T6L	Clear A-, B-, and E-registers	NAX/1 NBX/1 NEX/1	= PRE1 + . . . = PRE1 NINTRAPF + . . . = PRE1 + . . .	
	Memory request for operand from address specified in C-register	RQC	= PRE1 NINDX PREOPRQ NANLZ + . . .	Generates /MQC/ for transmission to memory
	Set flip-flop ARQ	(S/ARQ)	= RQC + . . .	Set at T6L. Inhibits another clock until AR received from memory
	Set Phase PRE2 flip-flop	S/PRE2	= NPREIM PRE1 N(S/INTRAPF)	
	Enable clock T4RL for next phase	T4RL	= PREP + . . .	
PRE2 T4RL	D → S	SXADD	= PRE2 NIA + . . .	Contents of A-, D-, and CS-registers added and placed on sum bus. A- and CS-registers contain zeros
	S -/→ P	PXS	= PRE2 NIA + . . .	
	Reset A- and D-registers	NAX/1	= PRE2 NIA + . . .	Clear A- and D-registers
		NDX/1	= PRE2 NIA + . . .	
	Set flip-flop DRQ	S/DRQ	= PRE2 NIA OPRQ N(PREDO IX + NFABRANCH) + . . .	Set at T4RL. Inhibits another clock until data release signal received
	Set phase 1 flip-flop	S/PH1/1	= NPREDO PRE2 NIA + . . .	
AR Received	Enable clock T4RL	T4RL	= PREP + . . .	
PH1 T4RL	Execution phase of instruction			

Signal RQC is generated during PRE4 and causes memory request /MQC/ for the data word stored in the location specified on the LB and LM memory address lines. When the clock occurs, flip-flop ARQ is set to inhibit another clock until the address release signal is received. Repeater flip-flop AG is held set, indicating that a second pass through PRE2 has not yet been performed. Phase PRE2 flip-flop is set, and clock T4RL is enabled.

During the second PRE2, the contents of the A-, D-, and CS- (carry save) registers are added, and the result is placed on the sum bus. The A-register was cleared during phase PRE3 and contains all zeros. The CS-register has not been set and also contains all zeros. The data transferred to the sum bus is, therefore, the contents of the D-register. If the address release signal has been received from memory, a T4RL clock is enabled, and bits S15 through S31 from the sum bus are gated into the P-register. The P-register then contains the address of the present operand and the Q-register contains the address of the next instruction. Registers A and D are cleared, and flip-flop DRQ is set inhibiting transmission of another clock until a data release signal is received. Phase PH1 flip-flop is set, the first of the execution phases for the instruction, and clock T4RL is enabled. The logic for the PRE2 sequence is described in the PRE2 discussion under the general preparation sequence.

A sequence chart of the preparation sequence when indirect addressing is specified is shown in table 3-26.

3-164 INDEXING. The three-bit X-field of a normal instruction format allows any one of the registers, 1 through 7 in private memory, to be used as an index register.

The indexing technique provides for operation on bytes, halfwords, words, and doublewords. Indexing for byte operations, halfwords, and doublewords will be discussed under the applicable instructions. Indexing for word operations (FAW) is discussed in the following paragraphs. See figure 3-152 which illustrates all possible preparation phase sequences.

As described under indirect addressing, during a previous instruction, the next instruction in the sequence is read from memory into the C-register, and the appropriate bits are gated into the O-, and R-registers. The entire word is gated into the D-register. If bit C0 of the instruction is a one, indirect addressing is performed prior to any indexing. When the clock occurs after occurrence of signal ENDE, flip-flop PRE1 is set, and clock T6L is enabled.

During PRE1, bits D12 through D14 (X-field of the instruction) are gated onto the private memory address lines LR29 through LR31. The address contained on the LR lines selects one of seven registers in private memory for use as an index register. A one in bits C12, C13, or C14 causes signal INDX to be generated. The PRE1 clock gates the address contained in the P-register into the Q-register, resets registers B and E, and clocks the word on the RR lines from private memory into the A-register. Index flip-flop IX is set by signal INDX. Phase PRE2 flip-flop is set, and clock T4RL is enabled.

During PRE2, the index data contained in the A-register, and the address contained in the D-register are added and are placed on the sum bus. Clock T4RL clocks the modified program address contained on the sum bus into the P-register. Index flip-flop IX is reset, and registers A and D are cleared. When the modified program address is in the P-register, the address is immediately gated onto the LM and LB lines to obtain the operand from memory. A memory request MRQ is initiated, and flip-flop ARQ is set which inhibits transmission of another clock until address release signal AR is received. Flip-flop RQ is set indicating that another memory request can be generated as soon as the operand is received with the address as specified in the Q-register. Flip-flop PH1 is set, and clock T4R1 is enabled which indicates the first of the execution phases of the instruction.

Table 3-27 is a sequence chart of the preparation phases when indexing is specified.

3-165 Load Immediate (LI 22, 62)

This instruction extends the sign bit of a 20-bit operand 12 positions to the left, then loads the new 32-bit word into the private memory register specified by the R-field. The operand, including the sign bit contained in bit 12, occupies bit positions 12 through 31. At the end of the instruction, the word loaded in the private memory register is tested for a positive, a negative, or a zero value. The results of this test are stored in flip-flops CC3 and CC4. A sequence chart of the Load Immediate instruction is given in table 3-28.

3-166 Load Byte (LB 72, F2)

The Load Byte instruction is used to transfer a byte from core memory into a private memory register specified by the R-field. Initially, the four bytes of the referenced word are transferred from core memory into the C-register. From here, the appropriate byte determined by the states of flip-flops P32 and P33 is clocked into D24-D31. At the end of the instruction, the byte is tested for a zero condition. Results of the test are stored in flip-flop CC3. A sequence chart of the Load Byte instruction is given in table 3-29.

3-167 Load Halfword (LH 52, D2)

The Load Halfword instruction is used to transfer a halfword from core memory into a private memory register specified by the R-field. Initially, both halves of the word, which are specified by the reference address, are transferred from core memory into the C-register. The appropriate halfword determined by the state of flip-flop P32 is then clocked into D16-D31. The sign bit of the halfword (MB0 if NP32, MB16 if P32) is extended into D0-D15. Next, the contents of the D-register are loaded into the private memory register specified by the R-field. At the end of the instruction, the word loaded in the private memory register is tested for a positive, a negative, or a zero value. Results of this test are stored in flip-flops CC3 and CC4. A sequence chart of the Load Halfword instruction is given in table 3-30.

Table 3-26. Preparation Sequence (Indirect Addressing, No Indexing)

Phase	Function Performed	Signals Involved	Comments
ENDE (Last phase of previous instruction)	C1-C7 —/→ O1-O7	OXC = ENDE + . . .	Opcode → O-register
	C8-C11 —/→ R28-R31	ORXC = ENDE + . . .	R-field → R-register
	C15-C31 → LM15-LB31	LMXC = MCC OXC + . . . (latched)	Program address to obtain second word from memory
	C0-C31 —/→ D0-D31	DXC = ENDE + . . .	
	Set Indirect Address flip-flop, IA	S/IA = C0 OXC + . . .	
	Set Phase PRE1 flip-flop	S/PRE1 = ENDE NHALT N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PRE1 T6L	Memory request for next word from address specified in C-register	RQC = C0 PRE1 + . . .	
	Set flip-flop ARQ	S/ARQ = RQC + . . .	Set at T6L. Inhibits another clock until AR signal received
	Set Phase PRE2 flip-flop	S/PRE2 = NPREIM PRE1 N(S/INTRAPF) + . . .	
	Enable clock T4RL	T4RL = PREP + . . .	
PRE2 T4RL	Set flip-flop DRQ	S/DRQ = IA PRE2 + . . .	After signal AR received, inhibits another clock until data received
	Set flip-flop AG	S/AG = PRE2 IA NIX OPRQ NANLZ + . . .	Indicates first pass through phase PRE2
	Set Phase PRE3 flip-flop	S/PRE3 = PRE2 IA N(S/INTRAPF) + . . .	
	Enable clock T4RL	T4RL = PREP + . . .	
AR Received			
PRE3 T4RL	MB0-MB31 → C0-C31	CXMB = DGC	Second memory word gated into C-register when data release signal received
	C0-C31 —/→ D0-D31	DXC = IA PRE3 + . . .	
	Reset indirect address flip-flop IA	R/IA = PRE3 + . . .	
	Set repeater flip-flop AG	S/AG = PRE3 AG + . . .	Indicates 2nd pass through PRE2 has not been made. Program address to obtain operand from memory
	C15-C31 → LM15-LB31	LMXC = (PRE3 IA + . . .) NAR	
	Set PRE4 flip-flop	S/PRE4 = PRE3 AG N(S/INTRAPF)	

(Continued)

Table 3-26. Preparation Sequence (Indirect Addressing, No Indexing) (Cont.)

Phase	Function Performed	Signals Involved	Comments
DR Received	Enable clock T4RL	T4RL = PREP + . . .	
PRE4 T4RL	Memory request for operand from address specified in C-register	RQC = PRE4 + . . .	
	Set flip-flop AG	S/AG = PRE4 + . . .	Indicates 2nd pass through PRE2 has not been made
	Set flip-flop ARQ	S/ARQ = RQC + . . .	
	Set Phase PRE2 flip-flop	S/PRE2 = PRE4 N(S/INTRAPF) + . . .	
	Enable clock T4RL	T4RL = PREP + . . .	
PRE2 T4RL	$D \longrightarrow S$	SXADD = PRE2 NIA + . . .	Contents of A-, D-, and CS-registers added and placed on sum bus. A- and CS-registers contain zeros
	$S \nrightarrow P$	PXS = PRE2 NIA + . . .	Address of present operand
	Reset A- and D-registers	NAX/1 = PRE2 NIA + . . . NDX/1 = PRE2 NIA + . . .	Clear A- and D-registers
	Set flip-flop DRQ	S/DRQ = PRE2 NIA OPRQ N(PREDO IX + NFABRANCH) + . . .	Inhibits another clock until data received
	Set phase 1 flip-flop	S/PH1 NPREDO PRE2 NIA + . . .	
AR Received	Enable clock T4RL	T4RL = PREP + . . .	
PH1 T4RL	Execution phase of instruction		

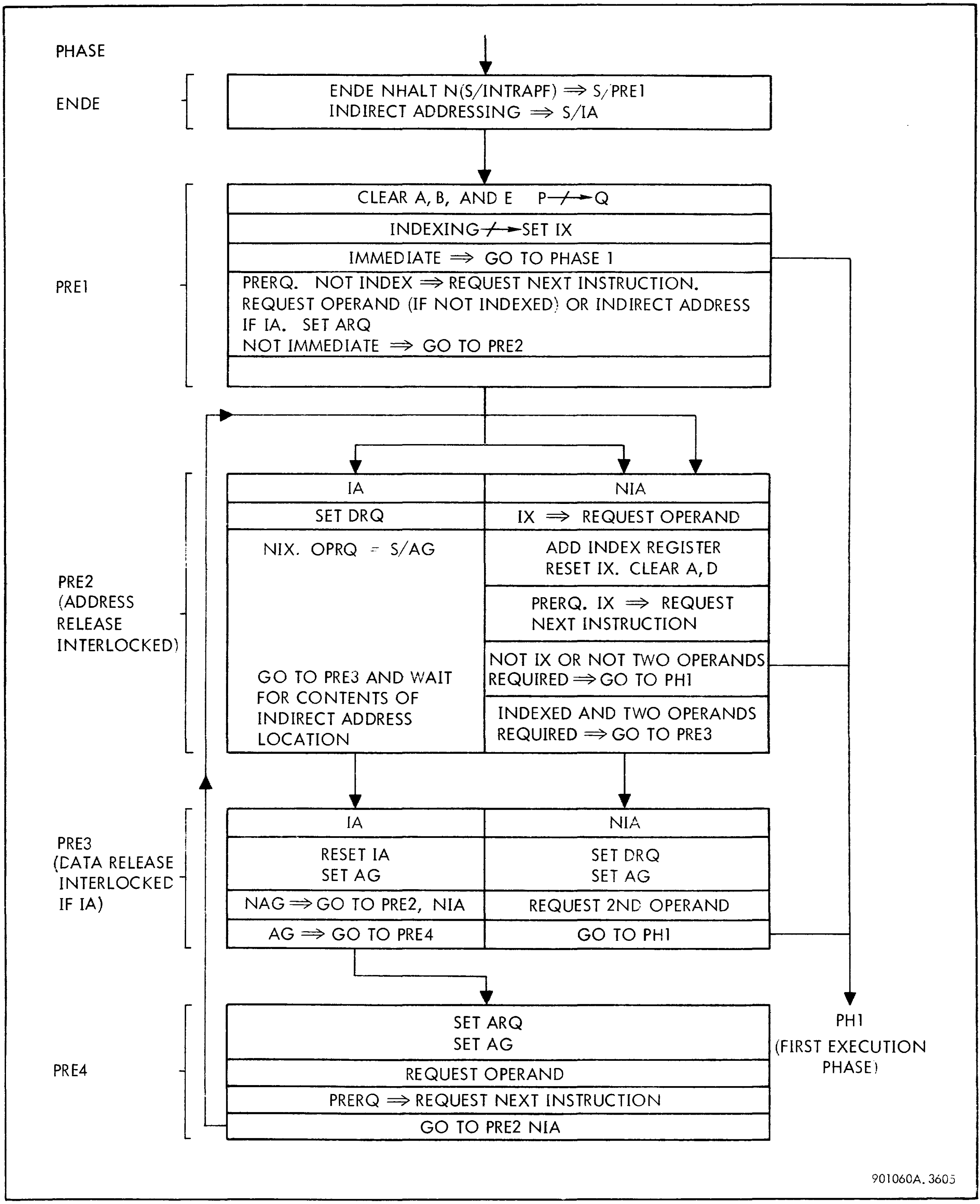


Figure 3-152. Preparation Sequence, Flow Diagram

Table 3-27. Preparation Sequence (Indexing, No Indirect Addressing)

Phase	Function Performed	Signals Involved	Comments
ENDE (Last phase of previous instructions)	C1-C7 —/→ O1-O7	OXC = ENDE + . . .	Opcode → O-register
	C8-C11 —/→ R28-R31	ORXC = ENDE + . . .	R-field → R-register
	C0-C31 —/→ D0-D31	DXC = ENDE + . . .	
	Set flip-flop LRXD	(S/LRXD) = OXC + . . .	X-field onto private memory address lines for PRE1
	Add one to address in P-register	PCTP1 = ENDE + . . .	
	Set phase PRE1 flip-flop	S/PRE1 = ENDE NHALT N(S/INTRAPF)	
	Enable clock T6L for next phase	T6L = NT1L NT4L NT8L NT10L NRESET	
PRE1 T6L	Generate index signal INDX	INDX = (C12 + C13 + C14) (C3 + C4 + C5)	Select index register
	P —/→ Q	QXP = PRE1 NANLZ + . . .	Address of next instruction
	Clear B- and E-registers	NBX/1 = PRE1 NINTRAPF + . . .	
		NEX/1 = PRE1 + . . .	
	Clear A-register	NAX/1 = PRE1 + . . .	
	RR0-RR31 —/→ A0-A31	AXRR = FAW INDX PRE1 + . . .	Word from selected index register read into A-register
	Set index flip-flop IX	S/IX = INDX PRE1	
	Set phase PRE2 flip-flop	S/PRE2 = NPREIM PRE1 N(S/INTRAPF)	
	Enable clock T4RL	T4RL = PREP + . . .	
PRE2 T4RL	A + D + CS → Sum bus	SXADD = PRE2 NIA + . . .	Modified program address transferred to sum bus
	S15-S31 —/→ P15-P31	PXS = PRE2 NIA + . . .	Modified program address gated into P-register
	Index flip-flop reset	R/IX = PRE2 NIA + . . .	
	Clear A-, and D-register	NAX/1 = PRE2 NIA + . . .	
		NDX/1 = PRE2 NIA + . . .	
	P15-P31 → LM15-LB31	(NLMXC NLMXQ) = PRE2 NIA NRIP	New program address gated onto address lines
	Generate memory request MRQ	MRQ = PRE2 NIA IX OPRQ NANLZ + . . .	Generate memory request for operand
	Set flip-flop RQ	S/RQ = (PRE2 NIA) IX PRERQ + . . .	Memory address from Q-register
	Set flip-flop ARQ	S/ARQ = PRE2 NIA IX + . . .	Inhibits transmission of another clock until address release signal received
	Set phase 1 flip-flop	S/PH1 = NPREDO PRE2 NIA	
	Enable clock T4RL	T4RL = PREP + . . .	

Table 3-28. Load Immediate, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP: (O) Opcode (P) Next instruction address			
	(D0-D11) Value of bit 12 of the instruction word			Sign extended 12 positions to the left
	(D12-D31) Value field of instruction word (R) R-field			
PH1 T4RL	Set flip-flop DRQ	S/DRQ	= FAS23 PH1 + . . .	Inhibits clock until data release signal received from memory
		R/DRQ	= . . .	
	Set flip-flop PH5	S/PH5	= BRPH5 NCLEAR + . . .	Advance to PH5
		BRPH5	= FAS23 PH1 + . . .	
		R/PH5	= . . .	
	Enable signal (S/T10L)	(S/T10L)	= FAS23 PH1 + . . .	Select clock T10L in PH5
PH5 T10L	D0-D31 → ADDER → S0-S31	SXADD	= FAS21 PH5 + . . .	Sign-extended value field gated into sum bus
	S0-S31 → RW0-RW31	RW	= FAS21 PH5 + . . .	Data from sum bus gated into scratchpad memory
	S0-S31 —/→ A0-A31	AXS	= FAS21 PH5 + . . .	Data gated from sum bus into A-register
	Set flip-flop TESTA	S/TESTA	= FAS21 PH5 + . . .	Prepare to test result after the next clock
		R/TESTA	= . . .	
	Enable signal ENDE	ENDE	= FAS21 PH5 + . . .	
	Set flip-flop PRE1 for next instruction	S/PRE1	= ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . .	
Next clock	Test word in A-register and, accordingly, code condition-code flip-flops CC3 and CC4:			
	If word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3	= NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z	= A-register does contain all zeros	
		R/CC4	= TESTA + . . .	
	If word is negative (A0), set flip-flop CC4 and reset flip-flop CC3	S/CC4	= A0 TESTA NTESTA/1 + . . .	
		R/CC3	= TESTA + . . .	
	If word contains all zeros, reset flip-flops CC3 and CC4	R/CC3	= TESTA + . . .	
		R/CC4	= TESTA + . . .	

Mnemonic: LI22

Table 3-29. Load Byte, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (R) R-field (P) Effective word address (P32, P33) Byte select bits		
PH1 T4RL	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
	MB0-MB31 ——→ C0-C31	CXMB = DG	Operand gated into the C-register from memory bus when data gate signal received from memory

	If NP32 NP33
	C0-C7 —/→ D24-D31
		DXCR24 = DXCBP NP32 NP33 + . . .
		DXCBP = OU7 NO4 NO5 PH1 + . . .
			Load byte 0 into the D-register

	If NP32 P33 C8-C15 —/→ D24-D31	DXCR16/13 = DXCBP NP32 P33 + . . .	Load byte 1 into the D-register
	If P32 NP33 C16-C23 —/→ D24-D31	DXCR8 = DXCBP P32 NP33 + . . .	Load byte 2 into the D-register
	If P32 P33 C24-C31 —/→ D24-D31	DXC/13 = DXC/1 + . . . DXC/1 = DXCBP P32 P33 + . . .	Load byte 3 into the D-register
	Enable signal DX	DX = DXCR24 + DXCR16/1 + DXCR8 + DXC/1	Clear the D-register
	Set flip-flop DRQ	S/DRQ = FAS23 PH1 + . . . R/DRQ = . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . . BRPH5 = FAS23 PH1 + . . . R/PH5 = . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L) = FAS23 PH1 + . . .	Select clock in PH5
PH5 T10L	D0-D31 ——→ ADDER ——→ S0-S31	SXADD = FAS21 PH5 + . . .	Data from D-register gated onto sum bus

Mnemonic: LB (72, F2)

(Continued)

Table 3-29. Load Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T10L (Cont.)	S0-S31 → RW0-RW31	RWXS/0 - RWXS/3 = RW RW = FAS21 PH5 + . . .	Load data from sum bus into private memory register specified by the R-field
	S0-S31 ↛ A0-A31	AXS = FAS21 PH5 + . . .	Load data from sum bus into A-register
	Set flip-flop TESTA	S/TESTA = FAS21 PH5 + . . . R/TESTA = . . .	
	Enable signal ENDE	ENDE = FAS21 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . R/PRE1 . . .	
Next clock	Test word in A-register and, accordingly, code condition-code flip-flop CC3		
	If word is not zero (NA0031Z), set flip-flop CC3	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . . NA0031Z = A-register does not contain all zeros	
	If word is zero (A0031Z), reset flip-flop CC3	R/CC3 = TESTA + . . .	

Mnemonic: LB (72, F2)

Table 3-30. Load Halfword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (R) R-field (P) Effective word address (P32) Halfword select bit		
PH1 T4RL	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
	MB0-MB31 ⟶ C0-C31	CXMB = DGC	Operand gated into the C-register from memory bus when data gate signal received from memory
	If NP32		
	C0-C15 ⟶ D16-D31	DXCR16 = DXC/5 NP32 + . . . DXC/5 = OU5 NO4 NO5 PH1 + . . .	Load most significant half of operand word into the D-register

	MB0 —/→ D0-D11
	MB0 —/→ D12-D15
		COC16C12 = MB0 NP32 CXMB (. . .) + . . .
		COC16 = MB0 NP32 CXMB (. . .) + . . .
			Extend sign bit of most significant halfword in the D-register

	If P32		
	C16-C31 —/→ D16-D31	DXC/2 = DXC/5 P32	Load least significant half of operand word into the D-register

	MB16 —/→ D0-D11
	MB16 —/→ D12-D15
		COC16C12 = MB16 P32 CXMB (. . .) + . . .
		COC16 = MB16 NP32 CXMB (. . .) + . . .
			Extend sign bit of least significant half of operand word into the D-register

	Set flip-flop DRQ	S/DRQ = FAS23 PH1 + . . . R/DRQ = . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . . BRPH5 = FAS23 PH1 + . . . R/PH5 = . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L) = FAS23 PH1 + . . .	Select clock in PH5
PH5 T10L	D0-D31 ⟶ ADDER ⟶ S0-S31	SXADD = FAS21 PH5 + . . .	Data from D-register gated onto sum bus
	S0-S31 ⟶ RW0-RW31	RWXS/0 - RWXS/3 = RW RW = FAS21 PH5 + . . .	Load data from sum bus into private memory register specified by the R-field

Mnemonic: LH (52, D2)

(Continued)

Table 3-30. Load Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T10L (Cont.)	S0-S31 —/→ A0-A31	AXS = FAS21 PH5 + . . .	Load data from sum bus into A-register
	Set flip-flop TESTA	S/TESTA = FAS21 PH5 + . . . R/TESTA = . . .	
	Enable signal ENDE	ENDE = FAS21 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . R/PRE1 = . . .	
Next clock	Test word in A-register and, accordingly, code condition-code flip-flops CC3 and CC4		
	If word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . . NA0031Z = A-register does not contain all zeros	
	If word is negative (A0), set flip-flop CC4 and reset flip-flop CC3	R/CC3 = TESTA + . . . S/CC4 = A0 TESTA NTESTA/1 + . . .	
	If word contains all zeros, reset flip-flops CC3 and CC4	R/CC4 = TESTA + . . .	

Mnemonic: LH (52, D2)

3-168 Load Word (LW 32, B2)

The Load Word instruction is used to read a word from a referenced core memory address, and to load this word into a private memory register specified by the R-field. At the end of the instruction, the word is tested for a positive, a negative, or a zero value. Results of this test are stored in flip-flops CC3 and CC4. A sequence chart of the Load Word instruction is given in table 3-31.

3-169 Load Doubleword (LD 12, 92)

The Load Doubleword instruction is used to transfer two words from consecutive locations in core memory into two consecutive private memory registers specified by the R-field. The low order word is loaded first, followed by the high order word. The sequence of word transfer is from core memory via the C-register to the D-register, then via the sum bus to the private memory register. A sequence chart of the Load Doubleword instruction is given in table 3-32.

3-170 Load Complement Halfword (LCH 5A, DA)

The Load Complement Halfword instruction is used to transfer the two's complement of a halfword from core memory into a private memory register specified by the R-field. Initially, both halves of the word, specified by the reference address, are transferred from core memory into the C-register. The one's complement of the appropriate halfword determined by the state of flip-flop P32 is clocked into D16-D31. The complement of the sign bit of the appropriate halfword (NMB0 if NP32, NMB16 if P32) is extended into D0-D15. Next, a one is added to the contents of the D-register in the adder, and the new sum placed on the sum bus. The two's complement formed from the appropriate halfword is loaded into the private memory register. At the end of the instruction, the new word (a complemented halfword with extended sign) is tested for a positive, a negative, or a zero value. Results of this test are stored in flip-flops CC3 and CC4.

Table 3-33 is a sequence chart of the Load Complement Halfword instruction.

3-171 Load Absolute Halfword (LAH 5B, DB)

During the Load Absolute Halfword instruction, a halfword is read from core memory. The sign bit is extended, and the new 32-bit word is tested. If it is positive, the new word is loaded into a private memory register specified by the R-field. If it is negative, the two's complement of the new word is loaded into the specified private memory register. Initially, both halves of the word specified by the reference address are transferred from core memory into the C-register. The halfword specified by P32 is clocked into D16-D31, and its sign is extended into D0-D15. Next, depending on whether the new word is positive or negative, the CS-register is either reset or set in preparation for the adder operation. The adder places on the sum bus either the two's complement of the contents of the D-register (CS-register set) or the contents of the D-register unaltered (CS-register reset). From the sum bus, the result is loaded into the private memory register specified by the R-field. At the end of the instruction, the word that is loaded in the private memory register is tested for zero. The result is stored in flip-flop CC3.

A sequence chart of the Load Absolute Halfword instruction is given in table 3-34.

3-172 Load Complement Word (LCW 3A, BA)

The Load Complement Word instruction is used to transfer the two's complement of a word from core memory into a private memory register specified by the R-field. First, the word obtained from a reference address in core memory is transferred to the C-register. The one's complement of the word is then clocked into the D-register. A one is added to the contents of the D-register in the adder, and the new sum placed on the sum bus. The two's complement formed is loaded into the private memory register, and the complemented word is first tested for overflow at the end of the instruction. Flip-flop CC2 is set, if overflow occurs and the trap mask is present. Next, the complemented word is tested for a positive, a negative, or a zero value. Results of the second test are stored in flip-flops CC3 and CC4.

A sequence chart of the Load Complement Word instruction is given in table 3-35.

3-173 Load Absolute Word (LAW 3B, BB)

During the Load Absolute Word instruction, a word is read from a reference address in core memory, and is tested. If it is positive, the word is loaded into a private memory register specified by the R-field. If it is negative, the word is first two's complemented and then is loaded into the specified private memory register. Initially, the word is read from the referenced core memory location and is clocked, via the C-register, into the D-register. Then, depending on the value of the sign bit D0, all flip-flops of the CS-register are either set (negative sign) or reset (positive sign). In addition, if D0 is true, the term K31 is also driven true. During the next operation, the contents of the D-register are either two's complemented (CS-register set) in the adder or are maintained in their original form (CS-register reset) and are placed on the sum bus. The absolute value of the original word is loaded into the private memory register. At the end of the instruction, the word now contained in the private memory register is tested for overflow. Flip-flop CC2 is set if overflow occurs and the trap mask is present. Next, the word is tested for a zero value. Results of this test are stored in flip-flop CC3.

A sequence chart of the Load Absolute Word instruction is given in table 3-36.

3-174 Load Complement Doubleword (LCD 1A, 9A)

During the Load Complement Doubleword instruction, two words are read from consecutive locations in core memory.

Table 3-31. Load Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (P) Effective word address (R) R-field		
PH1 T4RL	MB0-MB31 ⟶ C0-C31	CXMB = DGC	Operand gated into C-register from memory bus when data gate signal received from memory
	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
	C0-C31 —/→ D0-D31	DXC/6 = OU3 NO4 NO5 PH1 + . . .	Operand clocked into D-register
	Set flip-flop DRQ	S/DRQ = FAS23 PH1 + . . . R/DRQ = . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . . BRPH5 = FAS23 PH1 + . . . R/PH5 = . . .	Advance to PH5
	Enable signal (S/T10L)	(S/10L) = FAS23 PH1 + . . .	Select clock T10L in PH5
PH5 T10L	D0-D31 ⟶ ADDER ⟶ S0-S31	SXADD = FAS21 PH5 + . . .	Data from D-register gated onto sum bus
	S0-S31 ⟶ RW0-RW31	RWXS/0 - RWXS/3 = RW RW = FAS21 PH5 + . . .	Load data from sum bus into private memory register specifield by R-field
	S0-S31 —/→ A0-A31	AXS = FAS21 PH5 + . . .	Data gated from sum bus into A-register
	Set flip-flop TESTA	S/TESTA = FAS21 PH5 + . . . R/TESTA = . . .	Prepare to test result after the next clock
	Enable signal ENDE	ENDE = FAS21 PH5 + . . .	
	Set flip-flop PRE1 for next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . R/PRE1 = . . .	
Next clock	Test word in A-register and, accordingly, code condition-code flip-flops CC3 and CC4		

Mnemonic: LW (32, B2)

(Continued)

Table 3-31. Load Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
Next clock (Cont.)	If word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . . NA0031Z = A-register does not contain all zeros R/CC4 = TESTA + . . .	
	If word is negative (A0), set flip-flop CC4 and reset flip-flop CC3	S/CC4 = A0 TESTA NTESTA/1 + . . . R/CC3 = TESTA + . . .	
	If word contains all zeros, reset flip-flops CC3 and CC4	R/CC3 = R/CC4 = TESTA + . . .	

Mnemonic: LW (32, B2,

Table 3-32. Load Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP:		
	(O) Opcode		
	(Q) Next instruction address		
	(R) R-field		
	(P) Effective doubleword address		
	(LB31) Logical one		Selects 32 low order bits of doubleword to be loaded
	Two memory requests pending, one for each word of the doubleword		
PH1 T4RL	MB0-MB31 → C0-C31	CXMB = DG	Low order word gated into the C-register from memory bus when data gate signal received from memory
	C0-C31 —/→ D0-D31	DXC/6 = NO1 O3 NO4 NO5 NO7 PH1 + . . .	Clock low order word into the D-register
	Enable signal (S/LR31/2)	(S/LR31/2) = OU1 NO5 NO7 PH1 + . . .	Force a one into address line LR31 for selection of odd numbered fast memory register
	Set flip-flop RQ	S/RQ = OU1 NO5 NO7 PH1 + . . .	Generate memory request for next instruction
		R/RQ = CLEAR + DRQ	
	Set flip-flop DRQ	S/DRQ = PREDO PH1 + . . .	Inhibits clock until data release signal received from memory
		PREDO = OU1 NO5 NO7 + . . .	
		R/DRQ = . . .	
	Enable signal (S/T8L)	(S/T8L) = FULD PH1 + . . .	Select clock in PH3
		FULD = OU1 OL2 + . . .	
	Set flip-flop PH3	S/PH3 = BRPH3 NCLEAR + . . .	Advance to PH3
		BRPH3 = FULD PH1 + . . .	
		R/PH3 = . . .	
PH3 T8L	Q15-Q31 —/→ P15-P31	PXQ = PREDO PH3 + . . .	Load next instruction address into the P-register
		PREDO = OU1 NO5 NO7 + . . .	
	D0-D31 → S0-S31	SXD = FULD PH3 + . . .	Place low order word onto the sum bus
		FULD = OU1 OL2 + . . .	
	S0-S31 → RW0-RW31 [R + 1]	RWXS/0 - RWXS/3 = RW	Load low order word in odd numbered private memory register
		RW = FULD PH3 + . . .	

Mnemonic: LD (12, 92)

(Continued)

Table 3-32. Load Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3 T8L (Cont.)	S0-S31 —/→ A0-A31	AXS	= FULD PH3 + . . .	Load low order word in A-register in preparation for zero-testing in PH5
	MB0-MB31 ⟶ C0-C31	CXMB	= DG	High order word gated into the C-register from memory bus when data gate signal received from memory
	C0-C31 —/→ D0-D31	DXC	= FULD PH3 + . . .	Load high order word into the D-register. Memory request for high order word was made during PREP phases
	Set flip-flop DRQ	S/DRQ	= FAS16 PH3 + . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH5	S/PH5	= BRPH5 NCLEAR + . . .	Advance to PH5
		BRPH5	= FAS16 PH3 + . . .	
		R/PH5	= . . .	
	Enable signal (S/T8L)	(S/T8L)	= FULD PH3 + . . .	Select clock in PH5
PH5 T8L	D0-D31 ⟶ S0-S31	SXD	= FULD PH5 + . . .	Place high order word onto the sum bus
	S0-S31 ⟶ RW0-RW31 [R]	RWXS/0 - RWXS/3	= RW	Load high order word in even numbered private memory register
		RW	= FAS16 PH5 + . . .	
	If A-register does not contain all zeros, set flip-flop A31	S/A31	= A31X1 + . . .	Store nonzero condition of low order word
		A31X1	= FULD NA0031Z PH5 + . . .	
	S0-S31 —/→ A0-A31	AXS	= FAS16 PH5 + . . .	Load high order word in A-register
	Set flip-flop TESTA	S/TESTA	= FAS16 PH5 + . . .	
		R/TESTA	= . . .	
	Enable signal ENDE	ENDE	= FAS16 PH5 + . . .	
	Set flip-flop PRE1	S/PRE1	= ENDE N(S/INTRAPF) (NHALT + NFUEXU) + . . .	
		R/PRE1	. . .	
Next clock	Test doubleword for positive, negative, or zero condition and accordingly, code condition-code flip-flops CC3 and CC4			

Mnemonic: LD (12, 92)

(Continued)

Table 3-32. Load Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
Next clock (Cont.)	If doubleword is positive, set flip-flop CC3; if doubleword is negative, set flip-flop CC4; if doubleword is zero, reset flip-flops CC3 and CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . . NA0031Z = A-register does not contain all zeros R/CC3 = TESTA S/CC4 = A0 TESTA NTESTA/1 + . . . R/CC4 = TESTA	

Mnemonic: LD (12, 92)

Table 3-33. Load Complement Halfword, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (R) R-field (P) Effective word address (P32) Halfword select bit			
PH1 T4RL	Q15-Q31 —/→ P15-P31	PXQ	= PRERQ PH1 + . . .	Load next instruction address into the P-register
	MB0-MB31 ——→ C0-C31	CXMB	= DG	Operand gated into the C-register from memory bus when data gate signal received from memory
	If NP32			
	NMB0 —/→ D0-D11	C0C16C12	= (DXNCR16 NMB0 NP32 DXNC/3) (. . .) + . . .	Extend sign bit of most significant halfword in the D-register
	NMB0 —/→ D12-D15	C0C16/1	= (DXNCR16 NMB0 NP32 DXNC/3) (. . .) + . . .	
		DXNCR16	= DXNC/3 NP32	
		DXNC/3	= OU5 O4 NO5 NO7 PH1 DXNCR16	
	NC0-NC15 D16-D31			Load most significant half of operand word into the D-register
	If P32			
	NMB16 —/→ D0-D11	C0C16C12	= (DXNC/2 NMB16 P32 DXNC/3) (. . .) + . . .	Extend sign bit of least significant halfword in the D-register
	NMB16 —/→ D12-D16	C0C16/1	= (DXNC/2 NMB16 P32 DXNC/3) (. . .) + . . .	
		DXNC/2	= DXNC/3 P32	
	NC16-NC31 —/→ D16-D31	DXNC/12, DXNC/13	= DXNC/2 + . . .	Load one's complement of least significant halfword into the D-register
	Set flip-flop CS31	S/CS31	= CSX1/31 + . . .	Force a one into CS31 in preparation for two's complementing of data in PH5
		CSX1/31	= CSX1/8 + . . .	
		CSX1/8	= OU5 O4 NO5 NO7 PH1 + . . .	
		R/CS31	= (R/CS31)	
	Set flip-flop DRQ	S/DRQ	= FAS23 PH1 + . . .	Inhibits clock until data release signal received from memory

Mnemonic: LCH (5A, DA)

(Continued)

Table 3-33. Load Complement Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL (Cont.)	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . .	Advance to PH5
		R/PH5 = . . .	
		BRPH5 = FAS23 PH1 + . . .	
	Enable signal (S/T10L)	(S/10L) = FAS23 PH1 + . . .	Select clock in PH5
PH5	D0-D31 + CS31 → ADDER → S0-S31	SXADD = FAS21 PH5 + . . .	Gate two's complement of data onto the sum bus
	S0-S31 → RW0-RW31	RWXS/0- RWXS/3 = RW	Load data from sum bus into private memory register specified by R-field
		RW = FAS21 PH5 + . . .	
	S0-S31 —/→ A0-A31	AXS = FAS21 PH5 + . . .	Load data from sum bus into A-register
	Set flip-flop TESTA	S/TESTA = FAS21 PH5 + . . .	
		R/TESTA = . . .	
	Enable signal ENDE	ENDE = FAS21 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + NFUEXU) + . . .	
		R/PRE1 . . .	
Next clock	Test word in A-register and code condition-code flip-flop CC3 and CC4, accordingly		
	If word is positive, set flip-flop CC3 and reset flip-flop CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z = A-register does not contain all zeros	
	If word is negative, set flip-flop CC4 and reset flip-flop CC3	R/CC3 = TESTA + . . .	
		S/CC4 = A0 TESTA NTESTA/1 + . . .	
	If word is zero, reset flip-flops CC3 and CC4	R/CC4 TESTA	

Mnemonic: LCH (5A, DA)

Table 3-34. Load Absolute Halfword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (R) R-field (P) Effective word address (P32) Halfword select bit (A) All zeros		
PH1 T4RL	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
	MB0-MB31 ——→ C0-C31	CXMB = DGC	Operand gated into the C-register from memory bus when data gate signal received from memory
	If NP32		
	C0-C15 —/→ D16-D31	DXCR16 = DXC/5 NP32 + . . . DXC/5 = OU5 NO5 O6 O7 PH1 + . . .	Load most significant half of operand word into the D-register
	MB0 —/→ D0-D11	C0C16C12 = MB0 NP32 CXMB (. . .) + . . .	Extend sign bit of most significant halfword in the D-register
	MB0 —/→ D12-D15	C0C16 = MB0 CXMB NP32 (. . .) + . . .	
	If P32		
	C16-C31 —/→ D16-D31	DXC/2 = DXC/5 P32	Load least significant half of operand word into the D-register
	MB16 —/→ D0-D11	C0C16C12 = MB16 P32 CXMB (. . .) + . . .	Extend sign bit of least significant halfword in the D-register
	MB16 —/→ D12-D15	C0C16 = MB16 P32 CXMB (. . .) + . . .	
	Enable signal (S/T4L)	(S/T4L) = FAS10 PH1 + . . .	Select clock in PH2
PH2 T4L	If data word is negative (D0) Enable signal (S/NGX) Force ones into the CS-register	(S/NGX) = FAS10 PH2 D0 + . . . CSX/9, CSX/10 (S/NGX) + . . .	Prepare to place two's complement of word stored in D-register onto the sum bus during PH5
	Set flip-flop DRQ	S/DRQ = FAS10 PH2 + . . . R/DRQ = . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . . BRPH5 = FAS10 PH2 + . . . R/PH5 = . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L) = FAS10 PH2 + . . .	Select clock in PH5

Mnemonic: LAH (5B, DB)

(Continued)

Table 3-34. Load Absolute Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH5 T10L	If word stored in D-register is negative (NGX)			
	Enable signal K31	K31	= NGX + . . .	Used for changing one's complement number to two's complement
	NA0-NA31 ↓ CS0-CS31 ↓ N(D0-D31) → ADDER → S0-S31 K31 ↑	SXPR/10-13 SXADD	= SXADD + . . . = FAS21 PH5 + . . .	Place two's complement of word stored in the D-register onto the sum bus
	If word stored in D-register is positive			
	NA0-NA31 ↓ NCS0-NCS31 → ADDER → S0-S31 D0-D31 ↑	SXPR/10-13 SXADD	SXADD + . . . = FAS21 PH5 + . . .	Place word contained in D-register onto the sum bus
	S0-S31 → RW0-RW31	RWXS/0-RWXS/3 RW	= RW = FAS21 PH5 + . . .	Load data from sum bus into private memory register specified by R-field
	S0-S31 —/→ A0-A31	AXS	= FAS21 PH5 + . . .	Load data from sum bus into A-register
	Set flip-flop TESTA	S/TESTA R/TESTA	= FAS21 PH5 + ...	
	Enable signal ENDE	ENDE	= FAS21 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 R/PRE1	= ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . = . . .	
Next clock	Test word in A-register and code condition-code flip-flop CC3			
	If word is not zero, set flip-flop CC3.	S/CC3 NA0031Z	= NA0 NA0031Z TESTA (. . .) + . . . = A-register does not contain all zeros	
	If word is zero, reset flip-flop CC3	R/CC3	= TESTA ,	

Mnemonic: LAH (5B, DB)

Table 3-35. Load Complement Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	<u>At the end of PREP</u>: (O) Opcode (Q) Next instruction address (R) R-field (P) Effective word address		
PH1 T4RL	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
	MB0-MB31 ——→ C0-C31	CXMB = DG	Operand gated into the C-register from memory bus when data gate signal received from memory
	NC0-NC31 —/→ D0-D31	DXNC/1 = NO1 O3 O4 NO5 NO7 PH1 + . . .	Load one's complement of data in C-register into the D-register
	Set flip-flop CS31	S/CS31 = CSX1/31 CSX1/31 = DXNC/1 + . . . R/CS31 = (R/CS31)	Force a one into CS31 in preparation for two's complementing of data in PH5
	Set flip-flop DRQ	S/DRQ = FAS23 PH1 + . . . R/DRQ = . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . . BRPH5 = FAS23 PH1 + . . . R/PH5 = . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L) = FAS23 PH1 + . . .	Select clock in PH5
PH5 T10L	D0-D31 + CS31 ——→ ADDER ——→ S0-S31	SXADD = FAS21 PH5 + . . .	Gate two's complement of data onto the sum bus
	S0-S31 ——→ RW0-RW31	RWXS/0- RWXS/3 = RW RW = FAS21 PH5 + . . .	Load data from sum bus into private memory register specified by the R-field
	S0-S31 —/→ A0-A31	AXS = FAS21 PH5 + . . .	Load data from sum bus into A-register
	Set flip-flop TESTA	S/TESTA = FAS21 PH5 + . . . R/TESTA = . . .	
	Enable signal ENDE	ENDE = FAS21 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . R/PRE1 = . . .	

Mnemonic: LCW (3A, BA)

(Continued)

Table 3-35. Load Complement Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments

PH5
T10L
(Cont.)
	If overflow, set condition-code flip-flop CC2	S/CC2 = OVER PROBEOVER + . . .
		OVER = D0 NK0 NPR0 NFAS10 + . . .
		PROBEOVER = FAS26 PH5 + . . .
		R/CC2 = PROBEOVER + . . .
			During an LCW instruction fixed-point overflow occurs if the original data word in the C-register was X'80000000', since the two's complement of this number is also X'80000000'

	If overflow and bit 11 of the PSW is a one (trap mask), enable signal TROVER	TROVER OVER AM PROBEOVER + . . . AM = Function of bit 11 of the PSW	Trap to location X'43'
Test clock	Test word in A-register and code condition-code flip-flops CC3 and CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . . NA0031Z = A-register does not contain all zeros	
	If word is positive, set flip-flop CC3 and reset flip-flop CC4	R/CC3 = TESTA + . . . S/CC4 = A0 TESTA NTESTA/1 + . . .	
	If word is negative, set flip-flop CC4 and reset flip-flop CC3	R/CC4 = TESTA + . . .	
	If word is zero, reset flip-flops CC3 and CC4		

Mnemonic: LCW (3A, BA)

Table 3-36. Load Absolute Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (R) R-field (P) Effective word address (A) All zeros		
PH1 T4RL	Q15-Q31 ⇸ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
	MB0-MB31 → C0-C31	CXMB = DG	Operand gated into the C-register from memory bus when data gate signal received from memory
	C0-C31 → D0-D31	DXC/6 = NO1 O3 OLB PH1 + . . .	Operand clocked into the D-register
	Set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ) NANLZ) + . . .	Advance to PH2
		R/PH2 = . . .	
	Enable signal (S/T4L/2)	(S/T4L/2) = FAS10 PH1 + . . .	Select clock T4L in PH2
PH2 T4L	If data word is negative (D0)		
	Enable signal (S/NGX)	(S/NGX) = FAS10 D0 PH2 + . . .	Prepare to place two's complement of word stored in D-register onto the sum bus during PH5
		CSX1/9, CSX1/10 = (S/NGX) + . . .	
	Set flip-flop DRQ	S/DRQ = FAS10 PH2 + . . .	Inhibits clock until data release signal received from memory
		R/DRQ = . . .	
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . .	Advance to PH5
		BRPH5 = FAS10 PH2 + . . .	
		R/PH5 = . . .	
	Enable signal (S/T10L)	(S/T10L) = FAS10 PH2 + . . .	Select clock in PH5
PH5 T10L	If word stored in D-register is negative (NGX)		
	Enable signal K31	K31 = NGX + . . .	Used for changing one's complement number to two's complement
	NA0-NA31 CS0-CS31 → ADDER → S0-S31 N(D0-D31) K31	SXPR/10-SXPR/13 = SXADD + . . .	Place two's complement of word contained in the D-register onto the sum bus
		SXADD = FAS21 PH5 + . . .	

Mnemonic: LAW (3B, BB)

(Continued)

Table 3-36. Load Absolute Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH5 T10L (Cont.)	If word stored in D-register is positive			
	NA0-NA31 ↓ NCS0-NCS31 → ADDER → S0-S31 D0-D31 ↑	SXPR/10- SXPR/13	= SXADD + . . .	Place word contained in the D-register onto the sum bus
	S0-S31 → RW0-RW31	RWXS/0- RWXS/3	= RW	Load data from sum bus into private memory register specified by R-field
		RW	= FAS21 PH5 + . . .	
	S0-S31 —/→ A0-A31	AXS	= FAS21 PH5 + . . .	Load data from sum bus into A-register
	Set flip-flop TESTA	S/TESTA	= FAS21 PH5 + . . .	
		R/TESTA	= . . .	
	Enable signal ENDE	ENDE	= FAS21 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1	= ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . .	
		R/PRE1	= . . .	
	If overflow, set condition-code flip-flop CC2	S/CC2	= FAS10 PH5 K0 + . . .	During an LAW instruction fixed-point overflow occurs if the original data word in the C-register was X'80000000'
		R/CC2	= PROBEOVER + . . .	
		PROBEOVER	= FAS26 PH5 + . . .	
	If overflow, and bit 11 of the PSW is a one (trap mask), enable signal TROVER, and set flip-flops TRAP, TR30 and TR31	TROVER	= FAS10 PH5 K0 AM + . . .	Trap to location X'43'
		AM	= function of bit 11 of the PSW	
		S/TRAP	= TROVER + . . .	
		TRAP forces a one into P25 via PXTR		
		R/TRAP	= RESET + . . .	
		S/TR30	= TROVER NSTRAP N(S/TRACC4/1) + . . .	
		TR30 forces a one into P30 via PXTR		
		R/TR30	= (R/TR)	
		S/TR31	= TROVER N(S/TRACC4/1) NSTRAP NTRAP + . . .	
		TR31 forces a one into P31 via PXTR		
		R/TR31	= (R/TR)	
		PXTR	INTRAP1 INTRAP2 TRAP	

Mnemonic: LAW (3B, BB)

(Continued)

Table 3-36. Load Absolute Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
Next clock	Test word in A-register and code condition-code flip-flop CC3			
	If word is not zero, set flip-flop CC3	S/CC3	= NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z	= A-register does not contain all zeros	
	If word is zero, reset flip-flop CC3	R/CC3	= TESTA	

Mnemonic: LAW (3B, BB)

The two words are taken together as a 64-bit unit and two's complemented. The complemented doubleword is loaded into two consecutive private memory registers specified by the R-field. The 32 low order bits occupy the odd private memory location. If the address specified by the R-field is odd, only the 32 high order bits of the doubleword are stored.

The low order word is read from core memory into the C-register. The contents of the C-register is one's complemented and is clocked into the D-register. In the adder, one is added to the contents of the D-register. The two's complement of the low order word formed is loaded via the sum bus into the odd private memory register specified by the R-field. The complemented low order word is also tested for zero and for end carry conditions, and the results are stored in flip-flops BWZ and CS31, respectively.

The high order word is then read from core memory into the C-register. Again the one's complement of the C-register is clocked into the D-register and is placed in the adder. Bit CS31, if true, is added to the contents of the D-registers. The entire sum is placed on the sum bus and is loaded into the even numbered private memory register. In addition, this new high order word is clocked into the A-register, and bit A31 is merged with the contents of flip-flop BWZ. This information is used for zero-testing the complemented doubleword. At the end of the instruction the complemented doubleword is tested for overflow. Flip-flop CC2 is set, if overflow occurs and the trap mask is present. The complemented doubleword is tested for positive, negative, and zero conditions. Results of this test are stored in flip-flops CC3 and CC4.

A sequence chart of the Load Complement Doubleword instruction is given in table 3-37.

3-175 Load Absolute Doubleword (LAD 1B, 9B)

The Load Absolute Doubleword instruction is used to transfer the absolute value of a doubleword from two consecutive word locations in core memory to two consecutive fast memory registers specified by the R-field. Initially, the low order word is read from core memory and is gated, via the C-register, into the D-register. The two's complement of the low order word is clocked via the adder and sum bus into the A-register, while the D-register retains the original low order word. If end carry is detected in the two's complement of the low order word, it is stored in flip-flop K00H (reset condition). The high order word is read from core memory into the C-register, and is tested for a positive or a negative value. If it is positive, the low order word is transferred from the D-register into the odd private memory register. If it is negative, the two's complement of the low order word is transferred from the A-register into the odd private memory register. Also, if the value is positive, the high order word is clocked into the D-register; if negative, the one's complement of the high order word is gated into the D-register. The two's complement of the low order word is also tested for zero conditions, and the result is stored in flip-flop BWZ (reset condition). In the adder, the contents of the D-register is combined with the end carry, if any, and the new sum is loaded via the sum bus into the even numbered fast memory register. The absolute value of the doubleword is tested for overflow condition. If overflow occurs and the trap mask is present, it is stored in flip-flop CC2. Next, the absolute value of the doubleword is tested for a positive, a negative, and a zero condition. Results of this test are stored in flip-flops CC3 and CC4.

A sequence chart of the Load Absolute Doubleword instruction is given in table 3-38.

3-176 Load Selective (LS 4A, CA)

The LS instruction causes storing of selected bits from the word contained in the effective memory location, as determined by a 32-bit mask in private memory, into the private memory register addressed in the R-field of the instruction word.

If the private memory register addressed in the R-field is an even numbered register, the 32-bit mask contained in the odd numbered private memory register is compared with the data word from core memory. Wherever a one is contained in the mask, the corresponding bit of the core memory data word is stored in the even numbered private memory register. Wherever a zero exists in the mask, the corresponding bit in the private memory register is left unchanged.

If the private memory register addressed in the R-field is an odd numbered register (the same register containing the mask), a logical AND operation is performed with the mask and the data word from core memory. The result is stored in the odd numbered private memory register.

The core memory operand is loaded into the D-register, and, at the same time, the contents of the odd numbered private memory register are loaded into the A-register. Address line LR31 has been set previously to address the odd numbered register. A logical AND operation is performed in the adder between the mask in the A-register and the data word in the D-register. The result is loaded into the B-register. Next an Exclusive OR operation is performed between the mask in the A-register and the ones in the CS-register to obtain the complement of the mask. The result is stored in the D-register. Flip-flop LR31 is reset, and the data from the private memory register selected by the R-field of the instruction, whether even or odd, is clocked into the A-register. A logical AND operation is performed on the data in the A-register, the complement of the mask in the D-register, and the ones in the CS-register. The adder output is merged on the sum bus with the output of the B-register, and the result is stored in the private memory register addressed by the R-field of the instruction. The data on the sum bus is tested for all zeros and for the state of bit 0, and the condition-code flip-flops are set accordingly.

Table 3-39 is an example of the Load Selective operation for both an even and an odd numbered private memory register. A sequence chart for the instruction is given in table 3-40.

Table 3-37. Load Complement Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP:		
	(O) Opcode		
	(Q) Next instruction address		
	(R) R-field		
	(P) Effective doubleword address		
	(LB31) Logical one		Selects 32 low order bits of doubleword to be loaded
	Two memory requests pending, one for each word of the doubleword		
PH1 T4RL	MB0-MB31 → C0-C31	CXMB = DG	Low order word gated into the C-register from memory bus when data gate signal received from memory
	NC0-NC31 —/→ D0-D31	DXNC/10- DXNC/13 = DXNC/1 + . . . DXNC/1 = NO1 O3 O4 NO5 NO7 PH1 + . . .	Clock one's complement of low order word into the D-register
	1 —/→ LR31	(S/LR31/2) = OU1 NO5 NO7 PH1 + . . .	Force a one into address line LR31 for selection of odd numbered fast memory register
	1 —/→ CS31	CSX1/31 = DXNC/1 + . . . DXNC/1 = NO1 O3 O4 NO5 NO7 PH1 + . . .	Preset adder for two's complement of low order word → S in PH2
	Set flip-flop RQ	S/RQ = OU1 NO5 NO7 PH1 + . . . R/RQ = CLEAR DRQ	Generate memory request for next instruction
	Set flip-flop DRQ	S/DRQ = PREDO PH1 + . . . PREDO = OU1 NO5 NO7 + . . . R/DRQ = . . .	Inhibits clock until data release signal received from memory
	Set flip-flop PH2	S/PH2 = PH1 NBR (. . .) + . . . R/PH2 = . . .	Advance to PH2
	Enable signal (S/T10L)	(S/T10L) = FULCD PH1 + . . . FULCD = OU1 OLA	Select clock in PH2
PH2 T10L	CS31 ─┐ D0-D31 → ADDER → S0-S31	SXPR/10- SXPR/13 = SXADD + . . . SXADD = FAS16 PH2 + . . .	Place two's complement of low order word onto the sum bus

Mnemonic: LCD (1A, 9A)

(Continued)

Table 3-37. Load Complement Doubleword Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T10L (Cont.)	S0-S31—/→RW0-RW31 [R + 1]	RWXS/0- RWXS/3 = RW	Load low order word into the odd numbered private memory register specified by the R-field
		RW = OU1 O4 NO5 NO7 PH2 + . . .	
	S0-S31—/→A0-A31	AXS = FAS16 PH2 + . . .	Load low order word in A-register in preparation for zero-testing in PH5
	MB0-MB31——→C0-C31	CXMB = DG	High order word gated into the C-register from memory bus when data gate signal received from memory
	NC0-NC31—/→D0-D31	DXNC = FULCD PH2 + . . .	Clock one's complement of high order word into the D-register
		FULCD = OU1 OLA	
	If end carry is detected in low order word, reset flip-flop K00H	S/K00H = S00 + . . .	Store carry from low order word
		S00 Function of NK00	
		K00 = End carry present	
		R/K00H = . . .	
	Set flip-flop PH3	S/PH3 = PH2 NBR (. . .) + . . .	Advance to PH3
	Enable signal (S/T4RL)	(S/T4RL) FULCD PH2 + . . .	Select clock in PH3
PH3 T4RL	If A-register does not contain all zeros, set flip-flop BWZ	S/BWZ = NA0031Z FAS16 N0L9 PH3 + . . .	
		R/BWZ = CLEAR	
	Clear A-register	R/A0-A/31 = AX/1 + . . .	
		AX/1 = FAS19 PH3 + . . .	
	If flip-flop K00H was reset in PH2, set flip-flop CS31	S/CS31 = CSX1/8 + . . .	Store end carry from low order word in flip-flop CS31
		CSX1/8 = NK00H (S/BWZ/1) NOLB	
		R/CS31 (R/CS31)	
	Q15-Q31—/→P15-P31	PXQ PREDO PH3 + . . .	Load next instruction address into the P-register
		PREDO = OU1 NO5 NO7 + . . .	
	Set flip-flop PH5	S/PH5 = BRPH5 NCLEAR + . . .	Advance to PH5
		BRPH5 = FAS16 PH3 + . . .	
	Enable signal (S/T10L)	(S/T10L) = FAS19 PH3 + . . .	Select clock in PH5
	Enable signal (S/DRQ)	(S/DRQ) FAS16 PH3 + . . .	Inhibits transmission of another clock until data release signal received from core memory

Mnemonic: LCD (1A, 9A)

(Continued)

Table 3-37. Load Complement Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T10L	NA0-NA31 D0-D31 → ADDER → S0-S31 CS31	SXPR/10-13 SXADD + . . . SXADD = FAS19 PH5 + . . .	Place high order word together with carry from low order word onto the sum bus
	S0-S31 → RW0-RW31 [R]	RWXS/0-RWXS/3 = RW RW = FAS16 PH5 + . . .	High order word gated into even numbered private memory register specified by the R-field
	S0-S31 —/→ A0-A31	AXS = FAS16 PH5 + . . .	Load high order word from sum bus into the A-register
	BWZ —/→ A31	S/A31 = A31X1 + . . . A31X1 = FAS16 BWZ + . . .	Store not-all-zeros condition of low order word in flip-flop A31
	Set flip-flop TESTA	S/TESTA = FAS16 PH5 + . . . R/TESTA = . . .	
	Enable signal ENDE	ENDE = FAS16 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . R/PRE1 = . . .	

	If overflow, set condition-code flip-flop CC2	S/CC2 = OVER PROBEOVER + . . .
		OVER = D0 NK0 NPR0 NFAS10 + . . .
		PROBEOVER = FAS19 PH5 + . . .
		R/CC2 = PROBEOVER + . . .
			During an LCD instruction fixed-point overflow occurs if the original doubleword was X'8000000000000000', since the two's complement of this number is also X'8000000000000000'

	If overflow, and bit 11 of the PSW is a one (trap mask), enable signal TROVER	TROVER = OVER AM PROBEOVER
		AM = Function of bit 11 of the PSW
			Trap to location X'43'

Next clock	Test word in A-register and code condition-code flip-flops CC3 and CC4		
	If word is positive, set flip-flop CC3 and reset flip-flop CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . . NA0031Z = A-register does not contain all zeros	
	If word is negative, set flip-flop CC4 and reset flip-flop CC3	R/CC3 = TESTA S/CC4 = A0 TESTA NTESTA/1 + . . .	
	If word is zero, reset flip-flops CC3 and CC4	R/CC4 = TESTA	

Mnemonic: LCD (1A, 9A)

Table 3-38. Load Absolute Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP:			
	(O) Opcode			
	(Q) Next instruction address			
	(R) R-field			
	(P) Effective doubleword address			
	(LB31) Logical one			Selects 32 low order bits of doubleword to be loaded
	Two memory requests pending, one for each word of the doubleword			
PH1 T4RL	MB0-MB31 ⟶ C0-C31	CXMB	= DG .	Low order word gated into the C-register from memory bus when data gate signal received from memory
	C0-C31 —/→ D0-D31	DXC/10- DXC/13	= DXC/6 + . . .	Clock low order word into the D-register
		DXC/6	NO1 O3 OLB PH1 + . . .	
	Enable signal (S/NGX)	(S/NGX)	FULAD PH1 + . . .	Preset for -D ⟶ S in PH2
		FULAD	OU1 OLB	
	Set flip-flop DRQ	S/DRQ	= PREDO PH1 + . . .	Inhibits clock until data release signal received from memory
		PREDO	= OU1 NO5 NO7 + . . .	
		R/DRQ	= . . .	
	Force ones into the CS-register	CSX1/9, CSX1/10	= (S/NGX) + . . .	Prepare to place two's complement of word stored in the D-register onto the sum bus in PH2
	Set flip-flop PH2	S/PH2	= PH1 NBR (. . .) + . . .	Advance to PH2
		R/PH2	= . . .	
	Enable signal T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	Select clock in PH2
PH2 T6L	NA0-NA31 CS0-CS31 ⟶ ADDER ⟶ S0-S31 N(D0-D31) K31	NA0-NA31	Reset at end of PREP	
		CS0-CS31	= Set at PH1 clock	Place two's complement of low order word onto the sum bus
		SXPR/10-13	= SXADD + . . .	
		SXADD	= NGX NFAMDSF + . . .	
		K31	= NGX	
	S0-S31 —/→ A0-A31	AXS	FAS16 PH2 + . . .	Load two's complement of low order word from sum bus into the A-register

Mnemonic: LAD (1B, 9B)

(Continued)

Table 3-38. Load Absolute Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)	Reset flip-flop K00H if end carry is detected in complement of low order word	S/K00H = S00 + . . .	Store carry from low order word
		S00 = Function of NK00	
		K00 = End carry present	
		R/K00H = . . .	
	MB0-MB31 → C0-C31	CXMB = DG	High order word gated into the C-register from memory bus when data gate signal received from memory
	1 -/→ LR31	(S/LR31/2) = FULAD PH2 + . . .	Selects odd numbered private memory register
		FULAD = OU1 OLB	
	Set flip-flop PH3	S/PH3 = PH2 NBR (. . .) + . . .	Advance to PH3
		R/PH3 = . . .	
	Enable signal (S/T8L)	(S/T8L) = FULAD PH2 + . . .	Select clock in PH3
PH3 T8L	If NC0 D0-D31 → S0-S31	SXD/10- SXD/13 = SXD + . . .	If operand is positive, place low order word onto the sum bus
		SXD = FULAD NC0 PH3 + . . .	
	If C0 A0-A31 → S0-S31	SXA/10- SXA/13 = SXA	If operand is negative, place two's complement of low order word onto the sum bus
		SXA = FULAD C0 PH3 + . . .	
	S0-S31 → RW0-RW31 [R + 1]	RWXS/0- RWXS/3 = RW	Load absolute value of low order word into the odd numbered private memory register specified by the R-field
		RW = FULAD PH3 + . . .	
	Set flip-flop BWZ if two's complement of low order word does not contain all zeros	S/BWZ = NA0031Z FAS16 NOL9 PH3 + . . .	
		R/BWZ = CLEAR	
	If NC0 C0-C31 -/→ D0-D31	DXC/10- DXC/13 = DXC/6	If operand is positive, clock high order word into the D-register
	If C0	DXC/6 = FULAD NC0 PH3 + . . .	
	NC0-NC31 -/→ D0-D31	DXNC/10- DXNC/13 = DXNC	If operand is negative, clock one's complement of high order word into the D-register
		DXNC = FULAD C0 PH3 + . . .	

Mnemonic: LAD (1B, 9B)

(Continued)

Table 3-38. Load Absolute Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3 T8L (Cont.)	If C0 and flip-flop K00H was reset in PH2, set flip-flop CS31	S/CS31	= CSX1/8 + . . .	Store end carry from low order word in flip-flop CS31
		CSX1/8	= FULAD NK00H C0 PH3 + . . .	
	Q15-Q31 ↛ P15-P31	PXQ	= PREDO PH3 + . . .	Redundant Q → P is for mechanization convenience
		PREDO	= OU1 O4 NO5 + . . .	
	Enable signal MRQ/1	MRQ/1	= FULAD PH3 + . . .	Generate memory request for next instruction
	Set flip-flop PH5	S/PH5	= BRPH5 NCLEAR + . . .	Advance to PH5
		BRPH5	= FAS16 PH3 + . . .	
	Enable signal (S/T10L)	(S/T10L)	= FAS19 PH3 + . . .	Select clock in PH5
PH5 T10L	NA0-NA31 → ADDER DA0-DA31 → ADDER → S0-S31 CS31 → ADDER	SXPR/10-13	= SXADD + . . .	Place absolute value of high order word, together with carry from low order word, onto the sum bus
		SXADD	= FAS19 PH5 + . . .	
	S0-S31 → RW0-RW31 [R]	RWXS/0-RWXS/3	= RW	High order word gated into even numbered private memory register specified by the R-field
		RW	= FAS16 PH5 + . . .	
	S0-S31 ↛ A0-A31	AXS	= FAS16 PH5 + . . .	Load high order word from sum bus into the A-register
	BWZ → A31	S/A31	= A31X1 + . . .	Store not-all-zeros condition of low order word in flip-flop A31
		A31X1	= FAS16 BWZ + . . .	
	Set flip-flop TESTA	S/TESTA	= FAS16 PH5 + . . .	
		R/TESTA	= . . .	
	Enable signal ENDE	ENDE	= FAS16 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1	= ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . .	
		R/PRE1	= . . .	
	If overflow, set condition-code flip-flop CC2	S/CC2	= OVER PROBEOVER + . . .	During a LAW instruction fixed-point overflow occurs if the original value of the doubleword was X'8000000000000000'
		OVER	= DO NKO NPRO NFAS10 + . . .	
		PROBEOVER	= FAS19 PH5 + . . .	
		R/CC2	= PROBEOVER	

Mnemonic: LAD (1B, 9B)

(Continued)

Table 3-38. Load Absolute Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	
PH5 T10L (Cont.)	If overflow and bit 11 of the PSW is a one (trap mask), enable signal TROVER, and set flip-flops TRAP, TR30, and TR31	TROVER = OVER AM PROBEOVER	Trap to location X'43'
		AM = Function of bit 11 of the PSW	
		S/TRAP = TROVER + . . .	
		TRAP forces a one into P30 via PXTR	
		R/TRAP = RESET + . . .	
		S/TR30 = TROVER N(S/TRACC4/1) NSTRAP + . . .	
		TR30 forces a one into P30 via PXTR	
		R/TR30 = (R/TR)	
		S/TR31 = TROVER N(S/TRACC4/1) NSTRAP NTRAP + . . .	
		TR31 forces a one into P31 via PXTR	
		R/TR31 = (R/TR)	
		PXTR = INTRAP1 INTRAP2 TRAP	
Next clock	Test word in A-register and set flip-flops CC3 and CC4		
	If word is positive, set flip-flop CC3; if word is negative, set flip-flop CC4; if word is zero, reset flip-flops CC3 and CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z = A-register does not contain all zeros	
		R/CC3 = TESTA	
		S/CC4 = A0 TESTA NTESTA/1 + . . .	
		R/CC4 = TESTA	

Mnemonic: LAD (1B, 9B)

Table 3-39. Load Selective Operation Example with Even and Odd Numbered R-Field Address

A. R-Field Even Address		
Contents of registers:		
R (odd) (mask)	00101100111010101110100011011101	2CEAE8DD
Effective word	01010001110110010111101011010011	51D97AD3
R (even)	01100111001110100010101100110111	673A2B37
PH2 $A.D \rightarrow S$	$S \not\rightarrow B$	
Mask (A-register)	00101100111010101110100011011101	2CEAE8DD
Effective word (D)	01010001110110010111101011010011	51D97AD3
Result in B	00000000110010000110100011010001	00C868D1
PH4 $A \oplus CS \rightarrow S$	$S \rightarrow C \not\rightarrow D$	
Mask (A-register)	00101100111010101110100011011101	2CEAE8DD
CS-register	11111111111111111111111111111111	FFFFFFFFF
Result in D-register (NRodd)	11010011000101010001011100100010	D3151722
PH5 $A.D \rightarrow S$	$B \rightarrow S$	
R (even) (A-register)	01100111001110100010101100110111	673A2B37
D-register (NRodd)	11010011000101010001011100100010	D3151722
Result on sum bus	01000011000100000000001100100010	43100322
Sum bus	01000011000100000000001100100010	43100322
B-register	00000000110010000110100011010001	00C868D1
Merge	01000011110110000110101111110011	43D83BF3
Operation:	Wherever a one is contained in the mask, the corresponding bit of the effective word is stored in the even numbered private memory register. When a zero exists in the mask, the corresponding bit in the private memory register is left unchanged.	
B. R-Field Odd Address		
Contents of registers:		
R (odd) (mask)	00101100111010101110100011011101	2CEAE8DD
Effective word	01010001110110010111101011010011	51D97AD3
PH2 $A.D \rightarrow S$	$S \not\rightarrow B$	
Mask (A-register)	00101100111010101110100011011101	2CEAE8DD
Effective word (D)	01010001001010010111101011010011	51D97AD3
Result in B-register	00000000110010000110100011010001	00C868D1
PH4 $A \oplus CS \rightarrow S$	$S \rightarrow C \not\rightarrow D$	
Mask (A-register)	00101100111010101110100011011101	2CEAE8DD
CS-register	11111111111111111111111111111111	FFFFFFFF
Result in D-register	11010011000101010001011100100010	D3151722
PH5 $A.D \rightarrow S$	$B \rightarrow S$	
R (odd) A-register	00101100111010101110100011011101	2CEAE8DD
D-register (NRodd)	11010011000101010001011100100010	D3151722
Result on sum bus	00000000000000000000000000000000	00000000
Sum bus	00000000000000000000000000000000	00000000
B-register	00000000110010000110100011010001	00C868D1
Result of merge	00000000110010000110100011010001	00C868D1
Operation:	A logical AND operation is performed with the mask and the effective word. The result is stored back in the odd numbered private memory register.	

Table 3-40. Load Selective, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (P) Effective word address (Q) Next instruction address (R) R-field (LR31) True		Selects odd private memory register
PH1 T4RL	MB0-MB31 → C0-C31	CXMB = DG	Operand gated from core memory into C-register when data gate signal received from memory
	C0-C31 -/→ D0-D31	DXC/10 - DXC/13 = DXC/6 + . . . DXC/6 = FAS6 PH1 + . . . FAS6 = OU4 OLA + . . .	Clock operand into the D-register
	Ones -/→ CS0-CS31	CSX1/9 = (S/NPRX) + . . . (S/NPRX) = FAS6 PH1 + . . .	Used for logical operation in PH2
	Enable signal (S/T4L)	(S/T4L) = FULS PH1 + . . . FULS = OU4 OLA	Select clock in PH2

	RR0-RR31 -/→ A0-A31 [R + 1]	AXRR = FAS15 PH1 + . . .
		FAS15 = OU4 OLA + . . .
			Transfer mask from odd fast memory register into the A-register

PH2 T4L	A0-A31, CS0-CS31, D0-D31 → ADDER → S0-S31	SXPR = NPRX/1 NSDIS + . . . (S/NPRX/1) = (S/NPRX) = set in PH1	Perform an AND operation between the mask (A-register) and the operand (D-register) and place the result on sum bus
	S0-S31 -/→ B0-B31	BXS = FAS6 PH2 + . . .	Clock masked bits of operand into the B-register
	Ones -/→ CS0-CS31	CSX1 = FULS PH2 + . . .	Used for logical operation in PH4
	Enable signal BRPH4	BRPH4 = FULS PH2 + . . .	Advance to PH4
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH4
	Enable signal (S/CXS)	(S/CXS) = FULS PH2 + . . .	Used in PH4

Mnemonic: LS (4A, CA)

(Continued)

Table 3-40. Load Selective,Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH4 T6L	A0-A31 ⊕ CS0-CS31 → ADDER → S0-S31	SXPR	= FULS PH4 + . . .	Place complement of mask (A-register) via adder on sum bus, and then in the C-register
	S0-S31 → C0-C31	CXS	Set at PH2 clock	
	C0-C31 $\nrightarrow$ D0-D31	DXC/10 - DXC/13 DXC/6	= DXC/6 + . . . = FULS PH4 + . . .	Clock complement of mask in D-register
	RR0-RR31 $\nrightarrow$ A0-A31 [R]	AXRR	FULS PH4 + . . .	Transferred word to be masked from even numbered private memory register into the A-register
	Enable signal (S/NPRX)	(S/NPRX)	= FULS PH4 + . . .	Used for logical operation in PH5
	Ones $\nrightarrow$ CS0-CS31	CSX1/9	(S/NPRX) + . . .	
	Enable signal MRQ/1	MRQ/1	= FAS6 PH4 + . . .	Generate memory request for next instruction, as specified by the Q-register
	Q15-Q31 $\nrightarrow$ P15-P31	PXQ	= MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal (S/DRQ)	(S/DRQ)	= FAS6 PH4 + . . .	Inhibits clock until data release signal received from core memory
	Enable signal (S/T8L)	(S/T8L)	= FULS PH4 + . . .	Select clock in PH5
PH5 T8L	CS0-CS31, A0-A31, D0-D31 → ADDER → S0-S31 B0-B31 → S0-S31	SXPR (S/NPRX/1)	= NPRX/1 NSDIS + . . . = (S/NPRX) = Set in PH4	Perform an AND operation between the complement of the mask (D-register) and the word to be masked (A-register). Merge result on sum bus with masked bits of operand (B-register)
	S0-S31 → RW0-RW31	RWXS/0 - RWXS/3 RW	= RW = O1 NO2 NO5 O6 PH5 + . . .	Store result in even numbered private memory register
	Enable signal ENDE	ENDE	= FAS15 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1	= ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . .	
		R/PRE1	= . . .	

Mnemonic: LS (4A, CA)

(Continued)

Table 3-40. Load Selective, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH5 T8L (Cont.)	Set flip-flop TESTA	S/TESTA	= FAS15 PH5 + . . .	
		R/TESTA	= . . .	
	S0-S31 —/→ A0-A31	AXS	= FAS15 PH5 + . . .	Clock result in A-register; used for testing result stored in private memory register
Next clock	Test word in A-register, and set flip-flops CC3 and CC4			
	If word is positive, set flip-flop CC3; if word is negative, set flip-flop CC4; if word is zero, reset flip-flops CC3 and CC4	S/CC3	= NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z	= A-register does not contain all zeros	
		R/CC3	= TESTA	
		S/CC4	= A0 TESTA NTESTA/1 + . . .	
		R/CC4	= TESTA	

Mnemonic: LS (4A, CA)

3-177 Load Multiple (LM 2A, AA)

During the Load Multiple instruction, a sequential set of words is read from core memory and is loaded into a sequential set of private memory registers specified by the R-field. The number of words to be loaded (word count) is first stored in flip-flops CC1-CC4 and then is clocked into flip-flops E4-E7. An initial count of 0000 is interpreted as a word count of 16.

The instruction is executed in a sequence of iterations. During each iteration, a word is transferred from core memory, via the C- and D-registers, and sum bus into the private memory register. The P- and R-registers are incremented by one, and the E-register is decremented by one. Iterations stop when the word count in the E-register reaches one.

A sequence chart of the Load Multiple instruction is given in table 3-41.

3-178 Load Conditions and Floating Control Immediate (LCFI 02, 82)

This instruction is used to load the condition codes and floating control bits obtained during the previous instruction into CPU registers reserved for this purpose. The condition codes occupy bit positions 24 through 27 of the instruction word, and are loaded into flip-flops CC1-CC4 if bit position 10 of the instruction word is a one. The floating control bits occupy bit positions 29 through 31 of the instruction word, and are loaded into flip-flops FS, FZ, and FNF if bit position 11 of the instruction word is a one.

A sequence chart of the Load Conditions and Floating Control Immediate instruction is given in table 3-42.

3-179 Load Conditions and Floating Control (LCF 70, F0)

The Load Conditions and Floating Control instruction is used to load the condition codes and floating control bits, obtained from a specific byte in core memory, into CPU registers reserved for this purpose. The condition codes occupy bit positions 0 through 3 of the effective byte, and are loaded into flip-flops CC1-CC4 if bit position 10 of the instruction word is a one. The floating control bits occupy bit positions 5 through 7 of the effective byte, and are loaded into flip-flops FS, FZ, and FNF if bit position 11 of the instruction word is a one.

A sequence chart of the Load Conditions and Floating Control instruction is given in table 3-43.

3-180 Exchange Word (XW 46, C6)

The Exchange Word instruction is used to exchange a word in core memory with a word in a private memory register specified by the R-field. The original core memory word is held temporarily in the D-register and then is loaded into the private memory register. The original private memory word is held temporarily in the A-register and then is stored in core memory. At the end of the instruction, the new word contained in the private memory register is tested for zero, positive, and negative conditions. The results of this test are stored in flip-flops CC1-CC4.

A sequence chart of the Exchange Word instruction is given in table 3-44.

3-181 Store Byte (STB 75, F5)

The Store Byte instruction is used to transfer one byte from a fast memory register specified by the R-field into a core memory location specified by the effective address.

The byte to be stored is obtained from bits 24-31 of the affected fast memory register and is clocked into A24-A31. It is transferred to the appropriate byte on the sum bus (S0-S7, S8-S15, S16-S23, or S24-S31) as specified by P32 and P33 and is stored at the effective byte address in core memory.

A sequence chart of the Store Byte instruction is given in table 3-45.

3-182 Store Halfword (STH 55, D5)

The Store Halfword instruction is used to transfer one halfword from a fast memory register specified by the R-field into a core memory location specified by the effective address.

The halfword to be stored is obtained from bits 16-31 of the affected fast memory register and is clocked into A16-A31. It is transferred to the appropriate halfword on the sum bus (S0-S15 or S16-S31) as specified by P32 and is stored at the effective halfword address in core memory.

A sequence chart of the Store Halfword instruction is given in table 3-46.

3-183 Store Word (STW 35, B5)

The Store Word instruction is used to transfer one word from a private memory register specified by the R-field, via the A-register and sum bus, into the effective core memory location.

A sequence chart of the Store Word instruction is given in table 3-47.

3-184 Store Doubleword (STD 15, 95)

The Store Doubleword instruction is used to transfer a doubleword from two consecutive private memory registers into two consecutive core memory locations. The R-field and the effective address point to the even numbered private memory register and to the even numbered core memory locations, respectively. Initially, bits LR31 and LB31 are driven true, causing the low order word to be transferred first. The low order word is obtained from the odd private memory register and is stored in the odd core memory location.

Table 3-41. Load Multiple, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (P) Effective address of first word (R) Address of first general register (CC) Word count		
PH1 T4RL	CC1-CC4 —/→ E4-E7	EXCC = FAST PH1 O4 NO5 O6	Load word count into the E-register
	Enable signal BRPH3	BRPH3 = FAST PH1 OLA O2 + . . .	Advance to PH3A
	Set flip-flop PHA	S/PHA = FAST PH1 O2 + . . .	
		R/PHA = CLEAR	
	Enable signal MRQ	MRQ = FAST PH1 NO2 NO7 + . . .	Request first data word from memory
	Enable signal (S/DRQ)	(S/DRQ) = FAST EXU + . . .	Inhibits clock until data release signal received from memory
	Set flip-flop IEN	S/IEN = FAST PH1 O4 NO5 O6 + . . . R/IEN = (R/IEN)	Allows interrupt to be accepted at any clock pulse
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH3A
PH3A T6L	MB0-MB31 ——→ C0-C31	CXMB = DGC	First word loaded into the C-register from memory bus when data gate signal received from memory
	C0-C31 —/→ D0-D31	DXC/6 = FASTA PH3 + . . .	Clock first word into the D-register
	If word count does not equal 1 P + 1 —/→ P	PCTP1 = FASTA PH3A OLA N(E=1) + . . . (E=1) = Word count equals 1	Increment P-register to obtain the next word address
	Enable signal MRQ	MRQ = FASTA PH3 OLA N(E=1) + . . .	Request next data word from memory
	Reset flip-flop IEN	R/IEN = FASTA PH3 + . . .	Prevents interrupt from being accepted
	Enable signal BRPH5	BRPH5 = FASTA PH3 OLA + . . .	Advance to PH5A
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH5A

Mnemonic: LM (2A, AA)

(Continued)

Table 3-41. Load Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5A T6L	D0-D31 → S0-S31	SXD = FASTA PH5 NSDIS + . . .	Transfer first data word from the D-register via the sum bus into the private memory register specified by the R-field
	S0-S31 → RW0-RW31	RWXS/0- RWXS/3 = RW RW = FASTA PH5 + . . .	
	MB0-MB31 → C0-C31	CXMB = DGC	Next word loaded into the C-register from memory bus when data gate signal received from memory
	C0-C31 -/→ D0-D31	DXC/6 = FASTA PH5 + . . .	Load next data word into the D-register
	R + 1 -/→ R	PCTR = FASTA PH5 + . . .	Increment private memory register address in preparation for the next iteration
	E-1 -/→ E	MCTE1 = FASTA PH5 + . . .	Decrement the word count by one after each iteration
	If count in E-register does not equal 1, enable signal BRPH5	BRPH5 = FASTA PH5 OLA N(E=1) + . . .	Sustain PH5A until word count is reduced to one
	If count in E-register does not equal 1 or 2		Memory request is one clock ahead of the word count, and anticipates a reduction in the word count at the end of this phase
	Enable signal MRQ	MRQ = FASTA PH5 N(E=1 or 2) OLA + . . .	
	P + 1 → P	PCTP1 = FASTA PH5 N(E=1 or 2) OLA + . . .	
	If count in E-register equals 1		
	Enable signal MRQ/1	MRQ/1 = FASTA PH5 (E=1) O2 NO7 + . . .	Generate memory request for next instruction, with the address as specified by the Q-register
	Q15-Q31 -/→ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal BRPH13	BRPH13 = FASTA PH2 O2 (E=1) + . . .	Advance to PH13A
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH13A
PH13A T6L	Enable signal ENDE	ENDE = FASTA PH13 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: LM (2A, AA)

Table 3-42. Load Conditions and Floating Control Immediate, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP:		
	(O) Opcode		
	(D) Opcode		
	(R30) D10		
	(R31) D11		
PH1 T4RL	Enable signals CSX1/9 and CSX1/10	CSX1/9 = (S/NPRX) + . . .	Force ones into the CS-register, in preparation for D ⟶ S in PH2
		CSX1/10 = (S/NPRX) + . . .	
		(S/NPRX) = FAS17 PH1 + . . .	
	Set flip-flop DRQ	S/DRQ = FAS17 PH1 + . . .	Inhibits transmission of another clock until data received
		R/DRQ = . . .	
	Set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	Advance to phase 2
		R/PH2 = . . .	
	Reset flip-flop NT8L	R/NT8L = . . .	Select clock T8L in PH2
		S/NT8L = N(S/T8L)	
		(S/T8L) = FAS17 PH1 + . . .	
PH2 T8L	D24-D31 ⟶ ADDER ⟶ S0-S7	SXUAB = FAS17 PH2/1 + . . .	
	If R31	PH2/1 = PH2 NSDIS	R31 is derived from bit 11 of the instruction word which specifies floating mode control
	S5 ↛ FS	S/FS = FSZNXS S5	
		FSZNXS = FAS17 R31 PH2 + . . .	
		R/FS = FSZNXS	
	S6 ↛ FZ	S/FZ = FSZNXS S6	
		R/FZ = FSZNXS	
	S7 ↛ FNF	S/FNF = FSZNXS S7	
	If R30	R/FNF = FSZNXS	R30 is derived from bit 10 of the instruction word which specifies condition code setting
	S0-S3 ↛ CC1-CC4	S/CC1-CC4 = (S0-S3) CCXS + . . .	
		CCSX = FAS17 R30 PH2 + . . .	
		R/CC1-CC4 = (R/CC1)-(R/CC4)	
	Enable signal ENDE	ENDE = FAS17 PH2 + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . .	
		R/PRE1 = . . .	
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: LCFI (02)

Table 3-43. Load Conditions and Floating Control, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (P) Operand address (P32, P33) Byte address			
PH1 T4RL	MB0-MB31 → C0-C31	CXMB	= DGC	Operand word gated into C-register from memory bus when data gate signal received from memory
	If P32 P33			
	C24-C31 –/→ D24-D31	DXC/13	= DXCBP P32 P33 + . . .	Byte 3 of operand word gated into D24-D31
		DXCBP	= OU7 NO4 NO5 PH1 + . . .	
	If P32 NP33			
	C16-C23 –/→ D24-D31	DXCR8	= DXCBP P32 NP33 + . . .	Byte 2 of operand word gated into D24-D31
	If NP32 P33			
	C8-C15 –/→ D24-D31	DXCR16	= DXCBP NP32 P33 + . . .	Byte 1 of operand word gated into D24-D31
	If NP32 NP33			
	C0-C7 –/→ D24-D31	DXCR24	= DXCBP NP32 NP33 + . . .	Byte 0 of operand word gated into D24-D31
	Enable signals CSX1/9 and CSX1/10	CSX1/9	= (S/NPRX) + . . .	Force ones into the CS-register in preparation for D → ADDER in PH2
		CSX1/10	= (S/NPRX) + . . .	
		(S/NPRX)	= FAS17 PH1 + . . .	
	Set flip-flop PH2	S/PH2	= PH1 NBR N(FNANLZ NANLZ) + . . .	Advance to PH2
		R/PH2	= . . .	
	Reset flip-flop NT8L	R/NT8L	= . . .	Select clock T8L in PH2
		S/NT8L	= FAS17 PH1 + . . .	
PH2	D24-D31 → ADDER → S0-S7	PRX	= Was set at PH1 clock and	
		SXUAB	= FAS17 PH2/1 + . . .	
		PH2/1	= PH2 NSDIS	
	If R31			R31 is derived from bit 11 of the instruction word which specifies floating mode control
	S5 –/→ FS	S/FS	= FSZNXS S5	
		FSZNXS	= FAS17 R31 PH2 + . . .	
		R/FS	= FSZNXS	

Mnemonic: LCF (70)

(Continued)

Table 3-43. Load Conditions and Floating Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 (Cont.)	S6 $\not\rightarrow$ FZ	S/FZ = FSZNXS S6	
		R/FZ = FSZNXS	
	S7 $\not\rightarrow$ FNF	S/FNF = FSZNXS	
	If R30	CCXS	R30 is derived from bit 10 of the instruction word which specifies condition code setting
	S0-S3 $\not\rightarrow$ CC1-CC4	S/CC1-CC4 = (S0-S3) CCXS + . . .	
		CCXS = FAS17 R30 PH2 + . . .	
		R/CC1-CC4 = (R/CC1)-(R/CC4)	

Mnemonic: LCF (70)

Table 3-44. Exchange Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	<u>At the end of PREP</u>: (O) Opcode (Q) Next instruction address (P) Effective address of data word in core memory (R) Address of data word in fast memory		
PH1 T4RL	MB0-MB31 → C0-C31	CXMB = DGC	Data word from core memory loaded into the C-register upon receipt of data gate signal from core memory
	C0-C31 $\nrightarrow$ D0-D31	DXC/6 = FAS9 PH1 + . . .	Clock data word from core memory into the D-register
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FAS9 PH1	Clock data word from fast memory into the A-register
	Enable signal (S/T8L)	(S/T8L) = FAS9 PH1 + . . .	Select clock in PH2

	Enable signal MRQ	MRQ = FUXW PH1 + . . .
		FUXW = OU4 OL6
			Generate memory request for writing fast memory data word into core memory

	Enable signal (S/DRQ)	(S/DRQ) = FAS9 PH1 + . . .	Inhibits clock until data release signal received from core memory

PH2
T8L
	A0-A31 → S0-S31
	S0-S31 → MB0-MB31
		SXA = FAS9 PH2 NSDIS + . . .
		MBXS/0-MBXS/3 = MW (. . .) + . . .
		MW = FAS9 PH2 + . . .
			Transfer data word originally contained in fast memory into core memory location specified by the P-register

	Enable signal MRQ/1	MRQ/1 = FAS9 PH2 + . . .	Generate memory request for next instruction as specified by the Q-register
	Enable signal (S/DRQ)	(S/DRQ) = FAS9 PH2 + . . .	Inhibits clock until data release signal received from core memory
	Q15-Q31 $\nrightarrow$ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal (S/T8L)	(S/T8L) = FAS9 PH2 + . . .	Select clock in PH3

Mnemonic: XW (46, C6)

(Continued)

Table 3-44. Exchange Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3 T8L	D0-D31 → S0-S31	SXD	= FUXW PH3 NSDIS + . . .	Transfer data word originally contained in core memory into the fast memory register specified by the R-field
	S0-S31 → RW0-RW31	RWXS/0- RWXS/3	= RW	
		RW	= FAS9 PH3 + . . .	
	S0-S31 $\xrightarrow{/}$ A0-A31	AXS	= FAS9 PH3 + . . .	Clock data word originally contained in core memory into the A-register. Word will be tested after PH3 clock for positive, negative or zero value
	Set flip-flop TESTA	S/TESTA	= FAS9 PH3 + . . .	
		R/TESTA	= . . .	
	Enable signal ENDE	ENDE	= FAS9 PH3 + . . .	
Next clock	Test word now contained in both A-register and fast memory register for positive, negative, and zero values. Code condition-code flip-flops accordingly			
	If word is positive, set flip-flop CC3; if word is negative, set flip-flop CC4; if word is zero, reset flip-flops CC3 and CC4	S/CC3	= NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z	= A-register does not contain all zeros	
		R/CC3	= TESTA	
		S/CC4	= A0 TESTA NTESTA/1 + . . .	
		R/CC4	= TESTA	
				Mnemonic: XW (46, C6)

Table 3-45. Store Byte,Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (P15-P31) Effective word address (P32, P33) Byte address (A) Byte to be stored		
PH1 T4RL	A24-A31 → ADDER → S0-S7	SXUAB = OU7 NO4 O5 NO6 PH1 + . . .	Transfer byte from A-register via adder to each byte of the sum bus
	A24-A31 → ADDER → S8-S15	SXUAB	
	A24-A31 → ADDER → S16-S23	SXUAB	
	A24-A31 → ADDER → S24-S31	SXUAB	
	If NP32 NP33		
	S0-S7 → MB0-MB7	MBXS/0 = NP32 NP33 MWB (. . .) + . . . MWB = OU7 NO4 O5 NO6 PH1 + . . .	Store byte to the byte location specified by flip-flops P32 and P33
	If NP32 P33		
	S8-S15 → MB8-MB15	MBXS/1 = NP32 P33 MWB (. . .) + . . .	
	If P32 NP33		
	S16-S23 → MB16-MB23	MBXS/2 = P32 NP33 MWB (. . .) + . . .	
	If P32 P33		
	S24-S31 → MB24-MB31	MBXS/3 = P32 P33 MWB (. . .) + . . .	
	Enable signal MRQ/1	MRQ/1 = FAS18 PH1 + . . .	Generate memory request for next instruction with address as specified by the Q-register
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal (S/DRQ)	(S/DRQ) = FAS18 PH1 + . . .	
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH2
PH2 T6L	Enable signal ENDE	ENDE = FAS18 PH2 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE NHALT N(S/INTRAPF) (NHALT + FUEXU) + . . .	

Mnemonic: STB (75, F5)

Table 3-46. Store Halfword Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP:			
	(O) Opcode			
	(Q) Next instruction address			
	(P15-P31) Effective word address			
	(P32) Halfword address			
	(A) Contents of general register specified by R-field, including halfword to be stored			
PH1 T4RL	A16-A31 → ADDER → S0-S15	SXUAH	= FUSTH PH1 NSDIS + . . .	
		FUSTH	= OU5 OL5	
	A16-A31 → ADDER → S16-S31	SXUAH		
	If NP32			
	S0-S15 → MB0-MB15	MBXS/0, MBXS/1	= NP32 MWH (. . .) + . . .	Store halfword in the halfword location specified by flip-flop P32
		MWH	= FUSTH PH1 + . . .	
	If P32			
	S16-S31 → MB16-MB31	MBXS/2, MBXS/3	= P32 MWH (. . .) + . . .	
	Enable signal /MRQ/1	/MRQ/1	= FAS18 PH1 + . . .	Generate memory request for next instruction as specified by Q-register
	Enable signal (S/DRQ)	(S/DRQ)	= FAS18 PH1 + . . .	Inhibits clock until data release signal received from core memory
	Q15-Q31 —/→ P15-P31	PXQ	= /MRQ/1	Load next instruction address into the P-register
	Set flip-flop CS15	S/CS15	= FUSTH PH1 + . . .	Used in PH2 to generate carry bit K00 if high order halfword contains the extended sign of the low order halfword
	Enable signal (S/T8L)	(S/T8L)	= FUSTH PH1 + . . .	Select clock in PH2
PH2 T8L	Test for end carry K00	K00	= CS15 (A0-A15 are all ones)	
	Set flip-flop CC2 under one of the following conditions	CS15	= Was set in PH1	
	If A16 and NK00	S/CC2	= FUSTH PH2 A16 NK00 + . . .	Setting of CC2 indicates that high order halfword contains data and not the extended sign of the low order word
		A16	= Indicates that halfword stored is negative	

Mnemonic: STH (55, D5)

(Continued)

Table 3-46. Store Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T8L (Cont.)	If NA16 and A0-A15 are not zeros	S/CC2 = FUSTH PH2 NA16 NA0015Z + . . .	
	If neither of the above, reset flip-flop CC2	R/CC2 = FUSTH PH2 + . . .	NCC2 indicates that high order halfword contains extended sign of low order halfword
	Generate signal ENDE	ENDE = FAS18 PH2 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: STH (55, D5)

Table 3-47. Store Word, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (P) Effective word address (A) Contents of general register specified by R-field			
PH1 T4RL	A0-A31 → S0-S31	SXA	= NO1 O3 OL5 PH1 NSDIS	Store word obtained from fast memory in core memory
	S0-S31 → MB0-MB31	MBXS/0- MBXS/3	= MW (. . .) + . . .	
		MW	= NO1 O3 OL5 PH1 + . . .	
	Enable signal MRQ/1	MRQ/1	= FAS18 PH1 + . . .	Generate memory request for next instruction as specified by Q-register
	Enable signal (S/DRQ)	(S/DRQ)	= FAS18 PH1 + . . .	Inhibits clock until data release signal received from core memory
	Q15-Q31 –/→ P15-P31	PXQ	= (S/MRQ/1) + . . .	Load next instruction address into the P-register
	Enable signal T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	Select clock in PH2
PH2 T6L	Enable signal ENDE	ENDE	= ENDE = FAS18 PH2 + . . .	
	Set flip-flop PRE1	S/PRE1	= ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: STW (35, B5)

Bits LR31 and LB31 are driven false, and the contents of the even private memory register are stored in the even core memory location.

A sequence chart of the Store Doubleword instruction is given in table 3-48.

3-185 Store Selective (STS 47, C7)

The Store Selective instruction is used to alter or to write over selected bits of a core memory word. If the R-field is even, the even numbered private memory register contains the operand, and the odd numbered private memory register contains a 32-bit mask. When ones appear in the mask, the corresponding bits of the operand are stored in the effective core memory word. When zeros appear in the mask, the corresponding bits of the core memory word remain unchanged.

Initially, the complement of the mask is clocked in the A-register, and the effective word is clocked in the D-register. Then, the contents of the A-register are AND-gated with the contents of the D-register, and the result stored in the B-register. This result is the Exclusive OR between the mask and the core memory word.

Next, the mask is placed in the A-register; the operand is placed in the D-register. The two are AND-gated in the adder, and are merged on the sum bus with the contents of the B-register. Finally, the contents of the sum bus are stored in the effective core memory location.

If the R-field is odd, the contents of the private memory register are OR-gated with the contents of the effective core memory word. The new word is stored in the effective core memory location.

An example of the Store Selective instruction is shown in table 3-49. A sequence chart of the instruction is given in table 3-50.

3-186 Store Multiple (STM 2B, AB)

During the Store Multiple instruction, a group of words is obtained from a sequential set of private memory registers specified by the R-field and is stored in a sequential set of core memory locations. The number of words to be stored (word count) is first stored in flip-flops CC1-CC4 and then is clocked into flip-flops E4-E7. An initial count of 0000 is interpreted as a word count of 16.

The instruction is executed in a sequence of iterations. During each iteration, a word is transferred from a fast memory register, via the A-register and sum bus, into the effective core memory location. The P- and E-registers are incremented by one, and the E-register is decremented by one. Iterations stop when the word count in the E-register reaches one.

A sequence chart of the Store Multiple instruction is given in table 3-51.

3-187 Store Conditions and Floating Control (STCF 74, F4)

This instruction is used to store the current condition codes and floating control bits of the program status doubleword in byte 0 of the effective core memory location.

A sequence chart of the Store Conditions and Floating Control instruction is given in table 3-52.

3-188 Analyze (ANLZ 44, C4)

The ANLZ instruction causes a word to be read from the effective core memory address. The operand read into the C-register is treated as an instruction, and either the reference address is modified by indexing or indirect addressing is performed, as required, by the instruction being analyzed. If the operation code (opcode) portion of the word specifies an immediate addressing type, the condition code register is set to indicate the addressing type, and the execution proceeds to the next instruction. If the opcode portion of the instruction being analyzed is a non-immediate addressing type, the condition code register is set to indicate the type of addressing.

The effective address of the analyzed instruction is computed and is aligned as an integer displacement value (as illustrated in figure 3-153). The displacement value is then loaded into the private memory register indicated in the R-field of the ANLZ instruction word.

The preparation sequence for an ANLZ instruction is the same as the general preparation sequence. Since the ANLZ flip-flop has not been set, normal indexing and indirect addressing are performed as required.

The instruction word being analyzed is loaded into the D-register, and its opcode is clocked into the O-register. Signal LMXC is inhibited so that the reference address of the new instruction is not placed on the core memory address lines. A normal preparation sequence is entered to compute the effective address of the instruction being analyzed. Condition code flip-flops are set to indicate whether the instruction is in the immediate, the byte, the halfword, or the word class (see table 3-53). The instructions for each class are shown in table 3-54.

If the instruction is not an immediate type, the effective address in the P-register is gated onto bits 15 through 31 of the sum bus, and the byte or the halfword addressing bits in flip-flops P32 and P33 are gated into S0 and S1. The contents of the sum bus are clocked into the A-register.

If the instruction is of the byte addressing type, the A-register contents are shifted left two bit positions, and the byte selection bits in S0 and S1 are shifted into A30 and A31. The A-register contents are shifted left one bit position, and the halfword selection bit in S0 is shifted into A31, if the instruction is of the halfword addressing type.

Table 3-48. Store Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP:		
	(O) Opcode		
	(Q) Next instruction address		
	(P15-P30) Effective doubleword address		
	(P31) False		
	(LB31) True		Selects odd memory location
	(A) Low order word to be stored		

PH1
T4RL
	A0-A31 → S0-S31
	S0-S31 → MB0-MB31
		SXA = NO1 O3 OL5 PH1 NSDIS + . . .
		MBXS/0 - MBXS/3 = MW (. . .) + . . .
		MW = NO1 O3 OL5 PH1 + . . .
			Store low order word in odd memory location

	Enable signal MRQ	MRQ = FUSTD PH1 + . . .
		FUSTD = OU1 OL5
			Generate memory request for writing high order word

	RR0-RR31 —/→ A0-A31	AXRR = FUSTD PH1 + . . .	Transfer high order word from even location in fast memory register to the A-register
	Enable signal (S/DRQ)	(S/DRQ) = O3 OL5 PH1 + . . .	Inhibits clock until data release signal received from core memory
	Enable signal BRPH4	BRPH4 = FUSTD PH1 + . . .	Advance to PH4
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH4

PH4
T6L
	A0-A31 → S0-S31
	S0-S31 → MB0-MB31
		SXA = FUSTD PH4 NSDIS + . . .
		MBXS/0 - MBXS/3 = MW (. . .) + . . .
		MW = FAS14 PH4 + . . .
			Store high order word in even memory location

	Enable signal MRQ/1	MRQ/1 = FUSTD PH4 + . . .	Generate memory request for next instruction, as specified by the Q-register
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal (S/DRQ)	(S/DRQ) = FUSTD PH4 + . . .	Inhibits clock until data release signal received from core memory

Mnemonic: STD (15, 95)

(Continued)

Table 3-48. Store Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH4 T6L (Cont.)	Enable signal T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	Select clock in PH5
PH5 T6L	Enable signal ENDE Set flip-flop PRE1 for the next instruction	ENDE S/PRE1	= FAS14 PH5 + . . . = ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: STD (15, 95)

Table 3-49. Store Selective Operation Example with Even and Odd Numbered R-Field Address

A. R-Field Even Address

Contents of registers:

R (odd) (mask)	00101100111010101110100011011101	2CEAE8DD
R (even)	01100111001110100010101100110111	673A2B37
Effective word	01010001110110010111101011010011	51D97AD3
PH1 A ⊕ CS → S	S ↛ A	
R (odd) A-register	00101100111010101110100011011101	2CEAE8DD
CS-register (NPRX)	11111111111111111111111111111111	FFFFFFFF
NR (odd) A-register	11010011000101010001011100100010	D3151722
PH2 A.D → S	S ↛ B	
Effective word (D-reg.)	01010001110110010111101011010011	51D97AD3
NR (odd) from A-register	11010011000101010001011100100010	D3151722
Result in B-register	01010001000100010001001000000010	51111202
PH4 A.D → S	S → S	
R (even) D-register	01100111001110100010101100110111	673A2B37
R (odd) A-register	00101100111010101110100011011101	2CEAE8DD
Result on sum bus	00100100001010100010100000010101	242A2815
Sum bus	00100100001010100010100000010101	242A2815
B-register	01010001000100010001001000000010	51111202
Result of merge on sum bus	01110101001110110011101000010111	753B3A17

Operation: Wherever a one is contained in the mask, the corresponding bit of the even numbered private memory register is stored in the core memory word. The corresponding bit in the core memory word is left unchanged, when a zero exists in the mask.

B. R-Field Odd Address

Contents of registers:

R (odd) (mask)	00101100111010101110100011011101	2CEAE8DD
Effective word	01010001110110010111101011010011	51D97AD3
PH1 A + CS → S	S ↛ A	
R (odd) (mask) A-reg.	00101100111010101110100011011101	2CEAE8DD
CS-register (NPRX)	11111111111111111111111111111111	FFFFFFFF
NR (odd) A-register	11010011000101010001011100100010	D3151722
PH2 A.D → S	S ↛ B	
Effective word (D-reg.)	01010001110110010111101011010011	51D97AD3
NR (odd) A-register	11010011000101010001011100100010	D3151722
Result in B-register	01010001000100010001001000000010	51111202
PH4 A.D → S	B → S	
R (odd) D-register	00101100111010101110100011011101	2CEAE8DD
R (odd) A-register	00101100111010101110100011011101	2CEAE8DD
Result on sum bus	00101100111010101110100011011101	2CEAE8DD
Sum bus	00101100111010101110100011011101	2CEAE8DD
B-register	01010001000100010001001000000010	51111202
Result of merge on sum bus	01111101111110111111101011011111	7DFBFADF

Operation: An inclusive OR operation is performed with the word from the odd numbered private memory register and the core memory word. The result is stored in the core memory word location.

Table 3-50. Store Selective, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (P) Effective word address (Q) Next instruction address (A) Contents of odd fast memory register (mask) (CS) All ones		
PH1 T4RL	A0-A31 ⊕ CS0-CS31 → S0-S31	SXPR = FUSTS PH1 NSDIS + . . .	Load complement of mask in A-register
		FUSTS = OU4 OL7	
	S0-S31 —/→ A0-A31	AXS = FUSTS PH1 + . . .	
	MB0-MB31 → C0-C31	CXMB = DGC	Load effective word into the D-register
	C0-C31 —/→ D0-D31	DXC/10-DXC/13 = DXC/6 + . . .	
		DXC/6 = FAS6 PH1 + . . .	
		FAS6 = OU4 NO4 O5 O7 + . . .	
	Ones —/→ CS0-CS31	CSX1 = (S/NPRX) + . . .	Used for logical operation in PH2
		(S/NPRX) = FAS6 PH1 + . . .	
	Enable signal (S/CXRR)	(S/CXRR) = OU4 NO4 O5 O7 PH1 + . . .	Preset for RR → C in PH2
	Enable signal (S/T10L)	(S/T10L) = (S/CXRR) + . . .	Select clock in PH2
PH2 T10L	CS0-CS31, A0-A31, D0-D31 → ADDER → S0-S31	SXPR = NPRX/1 NSDIS + . . .	Complement of mask AND-gated with effective core memory word, and loaded in B-register via sum bus
		NPRX = Set at PH1 clock	
	S0-S31 —/→ B0-B31	BXS = FAS6 PH2 + . . .	
	RR0-RR31 → C0-C31	CXRR = Set at PH1 clock	Transfer word from fast memory register (even numbered word if R-field is even) to the C-register
	Set flip-flop LR31/2	S/LR31/2 = FUSTS PH2 + . . .	Select odd fast memory register in PH3
	Enable signal T4RL	T4RL = FUSTS PH2 + . . .	Select clock in PH3
PH3 T4RL	RR0-RR31 —/→ A0-A31	AXRR = FUSTS PH3 + . . .	Transfer mask from the odd numbered fast memory register into the A-register
		LR31 = Was set in PH2	

Mnemonic: STS (47, C7)

(Continued)

Table 3-50. Store Selective, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T4RL (Cont.)	C0-C31 —/→ D0-D31	DXC/6 = FUSTS PH3 + . . .	Load data word into the D-register (word was originally contained in even numbered fast memory register if R-field is even)
	Ones —/→ CS0-CS31	CSX1/9 = (S/NPRX) + . . . (S/NPRX) = FUSTS PH3 + . . .	Used for logical operation in PH4
	Enable signal MRQ	MRQ = FUSTS PH3 + . . .	Generate memory request for writing result in PH4
	Enable signal (S/DRQ)	(S/DRQ) = FUSTS PH3 + . . .	Inhibits clock until data release signal received from memory
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH4

PH4
T6L
	A0-A31
	D0-D31 → ADDER → S0-S31
	CS0-CS31
	B0-B31 → S0-S31
		SXPR = NPRX/1 NSDIS + . . .
		NPRX/1 = Set at PH3 clock
		SXB = OU4 NO4 O5 O7 PH4 + . . .
			AND mask in A-register with data word in D-register and place result on sum bus; merge result with contents of B-register

	S0-S31 → MB0-MB31	MBXS/0-MBXS/3 = MW (. . .) + . . . MW = FAS14 PH4 + . . .	Store result in core memory
	Enable signal MRQ/1	MRQ/1 = FAS6 PH4 + . . .	Generate memory request for next instruction, as specified by the Q-register
	Enable signal (S/DRQ)	(S/DRQ) = FAS6 PH4 + . . .	Inhibit clock until data release signal is received from memory
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH5
PH5 T6L	Enable signal ENDE	ENDE = FAS14 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: STS (47, C7)

Table 3-51. Store Multiple, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (P) Storage address of first word (Q) Next instruction address (R) First word address in private memory register (CC1-CC4) Word count		

PH1
T4RL
	CC1-CC4 —/→ E4-E7	EXCC = FAST PH1 O4 NO5 O6
		FAST = OU2 OLB NPHA + . . .
			Transfer word count into the E-register

	Set flip-flop PHA	S/PHA = FAST PH1 O2 + . . .	Advance to PH1A
	Enable signal BRPH1	BRPH1 = FAST PH1 OLB O2 + . . .	
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH1A
	Set flip-flop IEN	S/IEN = FAST PH1 O4 NO5 O6 + . . .	Allows interrupt to be accepted at any clock pulse

PH1A
T6L
	RR0-RR31 —/→ A0-A31	AXRR = FASTA PH1 O4 NO5 O7 + . . .
		FASTA = OU2 OLB PHA + . . .
			Transfer first word from fast memory register into the A-register

	R + 1 —/→ R	PCTR = FASTA PH1 O7 + . . .	Increment fast memory address by one
	Enable signal MRQ	MRQ = FASTA PH1 + . . .	Generate memory request for writing first word
	Enable signal (S/DRQ)	(S/DRQ) = FASTA EXU + . . . EXU = Was set during PREP phases	Inhibits clock until data release signal received from core memory
	Reset flip-flop IEN	R/IEN = FASTA PH1 + . . .	Prevents interrupt from being accepted
	Enable signal (S/T8L)	(S/T8L) = FASTA PH1 + . . .	Select clock in PH2A
PH2A T8L	A0-A31 ——→ S0-S31	SXA = FASTA PH2 NSDIS + . . .	Store word in core memory
	S0-S31 ——→ MB0-MB31	MBXS/0-MBXS/3 = MW (. . .) + . . . MW = FASTA PH2 + . . .	
	If word count does not equal one [N(E=1)]		
	RR0-RR31 —/→ A0-A31	AXRR = FASTA PH2 N(E=1) NOL9 + . . .	Transfer next word from fast memory register into the A-register

Mnemonic: STM (2B, AB)

(Continued)

Table 3-51. Store Multiple, Phase Sequence (Cont.)

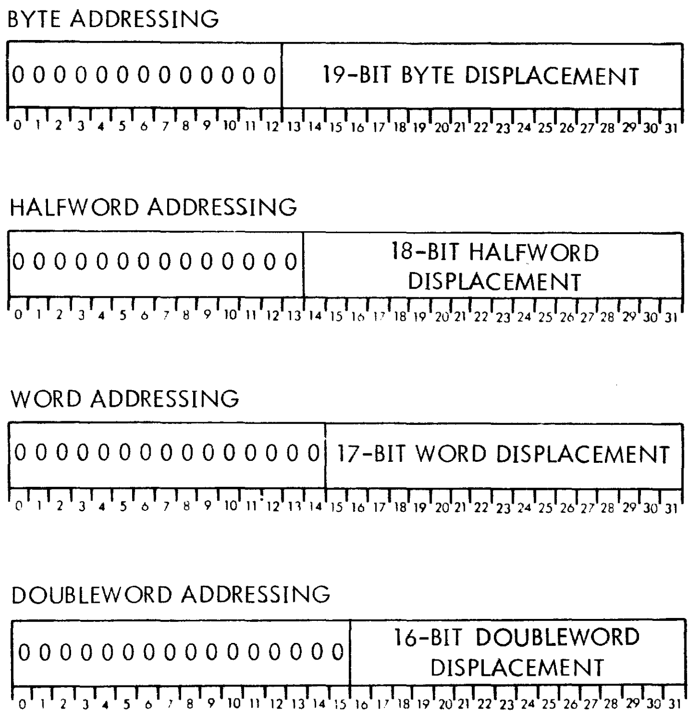
Phase	Function Performed	Signals Involved	Comments
PH2A T8L (Cont.)	Enable signal MRQ	MRQ = FASTA PH2 N(E=1) OLB + . . .	Generate memory request for next word to be stored
	Enable signal (S/DRQ)	(S/DRQ) = FASTA EXU + . . .	Inhibits clock until data release signal received from core memory
	P + 1 ⇸ P	CPTP1 = FASTA PH2 N(E=1) OLB + . . .	Increment P-register by one
	Enable signal BRPH2	BRPH2 = FASTA PH2 N(E=1) NOL9 + . . .	Sustain PH2 until word count is reduced to one
	R + 1 ⇸ R	PCTR = FASTA PH2 + . . .	Increment fast memory address by one
	E - 1 ⇸ E	MCTE1 = FASTA PH2 + . . .	Decrement word count by one
	If word count equals one		
	Enable signal MRQ/1	MRQ/1 = FASTA PH2 (E=1) O2 + . . .	Generate memory request for next instruction as specified by the Q-register
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal BRPH13	BRPH13 = FASTA PH2 O2 (E=1) + . . .	Advance to PH13A
	Enable signal (S/T8L)	(S/T8L) = FASTA PH2 + . . .	Select clock in PH13A
PH13A T8L	Enable signal ENDE	ENDE = FASTA PH13 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: STM (2B, AB)

Table 3-52. Store Conditions and Floating Control, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (Q) Next instruction address (P15-P31) Effective address (P32, P33) Zeros (D24) CC1 (D25) CC2 (D26) CC3 (D27) CC4 (D29) Floating significance (FS) (D30) Floating zero (FZ) (D31) Floating normalize (FN)		
PH1 T4RL	D24-D31 ⟶ ADDER ⟶ S0-S7	SXUAB = OU7 NO4 O5 NO6 PH1 + . . .	Transfer byte from D-register via adder to the high order byte of sum bus
	S0-S7 ⟶ MB0-MB7	MBXS/0 = NP32 NP33 MWB (. . .) + . . . MWB = OU7 NO4 O5 NO6 PH1 + . . . NP32, NP33 = Came true at the end of PREP	Store byte in core memory location specified by the P-register. Byte contains condition codes and floating control bits
	Enable signal MRQ/1	MRQ/1 = FAS18 PH1 + . . .	Generate memory request for next instruction, as specified by the Q-register
	Enable signal (S/DRQ)	(S/DRQ) = FAS18 PH1 + . . .	Inhibits transmission of another clock until data release signal is received from core memory
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal T6L	T6L = NT1L NT4L NT8L NT10L NRESET	Select clock in PH2
PH2 T6L	Enable signal ENDE	ENDE = FAS18 PH2 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE NHALT N(S/INTRAPF) + . . .	

Mnemonic: STCF (74, F4)



901060A. 3633

Figure 3-153. Instruction Addressing, Integer Displacement

Table 3-53. Condition Code Settings

CONDITION CODE FLIP-FLOPS				TYPE OF ADDRESSING*
CC1	CC2	CC3	CC4	
0	0	0	0	Byte ⓪
0	0	0	1	Immediate Byte ①
0	1	0	0	Halfword ④
1	0	0	0	Word ⑧
1	0	0	1	Immediate Word ⑨
1	1	0	0	Doubleword ⑫

*Circled numbers indicate instruction groups listed in table 3-54

Table 3-54. Analyze Table for Opcode Groups

X	00 + X	20 + X	40 + X	60 + X
00	——	AI	TTBS	CBS
01	——	CI	TBS ①	MBS
02	LCFI ⑨	LI	——	——
03	——	MI	——	EBS
04	CAL1	SF	ANLZ	BDR
05	CAL2	S	CS	BIR
06	CAL3	——	XW	AWM
07	CAL4	——	STS	EXU
08	PLW	CVS	EOR	BCR
09	PSW	CVA ⑧	OR	BCS
0A	PLM	LM	LS	BAL
0B	PSM	STM	AND	INT
0C	——	——	SIO	RD
0D	——	——	TIO	WD
0E	LPSD ⑫	WAIT	TDV	AIO
0F	XPSD	LRP	HIO	MMC
10	AD	AW	AH	LCF
11	CD	CW	CH	CB
12	LD	LW	LH	LB
13	MSP	MTW	MTH	MTB
14	——	——	——	STCF
15	STD	STW	STH ④	STB ⓪
16	——	DW	DH	PACK
17	——	MW	MH	UNPK
18	SD	SW	SH	DS
19	CLM	CLR	——	DA
1A	LCD	LCW	LCH	DD
1B	LAD	LAW	LAH	DM
1C	FSL	FSS	——	DSA
1D	FAL	FAS	——	DC
1E	FDL	FDS	——	DL
1F	FML	FMS	——	DST

Note

Circled numbers indicate type of addressing on table 3-53

If the instruction is of the doubleword type, the A-register contents are shifted one bit position to the left and then two bit positions to the right. This is an effective shift of one bit position to the right. The result of these shifts is shown in figure 3-153. The modified word is stored in the addressed private memory register.

A sequence chart of the Analyze instruction is given in table 3-55.

3-189 Interpret (INT 6B, EB)

INT instruction causes bits zero through three of the core memory effective word to be loaded into the condition code flip-flops. Bits 4 through 15 are loaded into bit positions 20 through 31 of the private memory register addressed in the R-field of the instruction word, and bits 16 through 31 of the effective word are loaded into bit positions 16 through 31 of the odd numbered private memory register. Zeros are loaded into the remaining bit positions.

If the private memory register addressed in the R-field is an odd numbered register, bits 16 through 31 of the core memory effective word are loaded into bit positions 16 through 31 of the odd numbered register. Zeros are loaded into bit positions 0 through 15 of the private memory register.

Bits 4 through 15 of the core memory word are loaded into bit positions 20 through 31 of the D-register, and bits 0 through 19 are forced to 0. The result is stored in the private memory register specified in the R-field of the instruction. The entire memory word is then clocked into the D-register, and the odd numbered private memory register is cleared which may or may not be the register addressed above. Bits 0 through 3 of the operand in the D-register are transferred to condition code flip-flops CC1 through CC4. Bits 16 through 31 of the core memory word are loaded into the odd numbered private memory register by setting the private memory write byte signals for bytes 2 and 3.

A sequence chart of the Interpret instruction is given in table 3-56.

3-190 Add Immediate (AI 20)

Add Immediate is an immediate addressing type instruction in which bits 12 through 31 of the instruction word are treated as a 20-bit two's complement integer.

The AI instruction causes the sign bit of the integer (bit 12) to be extended 12 bit positions to the left which forms a 32-bit addend. This addend is added to the 32-bit word contained in the private memory register addressed in the R-field of the instruction word. The sum is stored in the addressed private memory register.

The sign in the D-register is extended into bits 0 through 11 during the preparation phases. The contents of the addressed private memory register are loaded into the A-register. This value is gated into the adder with the contents of the D-register. The sum on the sum bus is stored in the selected private memory register, and the contents of the sum bus are also clocked into the A-register to allow the setting of appropriate condition code flip-flops.

The instruction traps to location X'43', if condition code flip-flop CC2 is set and if the trap mask flip-flop in the PSD contains a one. The instruction traps in the preparation phases, if indirect addressing is attempted (see paragraph 3-141 for trap sequencing).

Table 3-57 is a sequence chart of the Add Immediate instruction.

3-191 Add Halfword (AH 50, D0)

The AH instruction selects the desired halfword from the effective core memory location and extends the sign bit 16 bit positions to the left. The resulting 32-bit addend is then added to the data word contained in the private memory register addressed in the R-field. The sum is stored in the private memory register.

During the preparation phases, the least significant bit (LSB) of the index register contents is moved into flip-flop P32 to select the desired halfword in the memory location. The indexing data is read into the A-register. The contents are shifted one bit position to the right. The operand address in the D-register and the indexing data in the A-register are added. This modified program address is transferred to the P-register. The selected halfword from memory is loaded into the D-register, and the value from the selected private memory register is clocked into the A-register. These two values are gated into the adder. The sum is transferred to the selected private memory register and to the A-register. The contents of the adder and the A-register are tested for condition code and trap information. The program traps to location X'43' after loading the results into private memory, if overflow has occurred and the fixed-point arithmetic trap mask bit in the PSD is set.

Table 3-58 is a sequence chart of the Add Halfword instruction.

3-192 Add Word (AW 30, B0)

The AW instruction causes the data word from the reference address location in core memory to be added to the data word from the private memory register addressed in the R-field of the instruction word. The sum is stored in the addressed private memory register.

The operand from core memory is transferred to the D-register, and the value in the addressed private memory register is transferred to the A-register. These two values are gated into the adder, and the sum is loaded into the selected private memory register. The sum is also stored in the A-register. The adder and the A-register contents are tested for condition code and overflow information.

Table 3-55. Analyze, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PH1 T4RL	C0-C31 ⇸ D0-D31	DXC	= FUANLZ NANLZ PH1 + . . .	Instruction to be analyzed
	C1-C7 ⇸ O1-O7	OXC	= FUANLZ NANLZ PH1 + . . .	Opcode of instruction to be analyzed
	Inhibit generation of signal LMXC	LMXC	= NAR (NPH1 + NFUANLZ + SW2) + . . .	To ensure that reference address of new instruction is not read onto LM and LB address lines
	Set flip-flop ANLZ	S/ANLZ	= FUANLZ PH1	
	Set flip-flop PRE1	S/PRE1	= FUANLZ NANLZ PH1 N(S/INTRAPF) + . . .	To perform preparation sequence for new instruction
	Inhibit setting of flip-flop PH2	S/PH2	= BRPH2 NCLEAR + PH1 NBR	
		BRPH2	= PH1 NBR	
		NBR	= NFUANLZ + ANLZ (. . .)	
	Enable clock T8L	S/T8L	= FUANLZ PH1 NANLZ + . . .	
PRE1 T8L	Normal preparation sequence except as follows:			To compute effective address
	Inhibit signal QXP	QXP	= PRE1 NANLZ + . . .	Prohibit clocking operand address into Q-register
	Inhibit memory requests except for indirect addressing	S/RQ	= PRERQ PRE1 NIA NINDXX	
		PRERQ	= NOL3 NANLZ + . . .	
		RQC	= PRE1 NINDX PREOPRQ/2 NANLZ + . . .	Allowed for indirect addressing
	Set flip-flop SW2	S/SW2	= ANLZ PRE1 + . . .	To control address gating term LMXC
	Set condition code flip-flops to indicate opcode class	S/CC1	= FADW (ANLZ PRE1) + ANLZ PRE1 FAIM (NO1 NO3) (NO4 NO5) + (ANLZ PRE1) FAW + . . .	Immediate word type, doubleword type, and word type
		S/CC2	= OU5 (ANLZ PRE1) + FADW ANLZ PRE1 + . . .	Doubleword and halfword class
		S/CC4	= FAIM ANLZ PRE1 + . . .	Immediate class
	If flip-flop is not set, generate signal ANLZ/1	ANLZ/1	= ANLZ NCC4	
END PREP	Generate signal (S/PH1/1)	(S/PH1/1)	= NPREDO (PRE2 NIA) + PREDO NIX (PRE2 NIA) + PRE3 NIA	
T4RL	Set flip-flop PH1	S/PH1	= (S/PH1/1) NCLEAR + . . .	
	Force ANLZ opcode into O-register	OX/1	= (S/PH1/1) ANLZ + . . .	
	Enable T4RL	T4RL	PREP + . . .	

Mnemonic: ANLZ
(44, C4)

(Continued)

Table 3-55. Analyze, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1 (Cont.)	If signal ANLZ/1 is true, gate P15-P31 ——→ S15-S31		
T4RL		SXP = ANLZ/1 PH1 + . . .	Address gated onto sum bus if not immediate addressing
	If a byte addressing class, gate signals	S0 = P32 SXP + . . .	Byte addressing bits
	P32 and P33 into S0 and S1	S1 = P33 SXP + . . .	Byte or halfword addressing bit
	Clock S0-S31 —/→ A0-A31	AXS = ANLZ/1 PH1 + . . .	If not immediate addressing
	Clear D-register	DX = ANLZ PH1 + . . .	
	Generate memory request for next instruction	MRQ/1 = ANLZ PH1 + . . .	
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	Address of next instruction
	P15-P31 ——→ LM15-LB31	(NLMXC NLMXQ)	
	Set flip-flop PH2	S/PH2 = PH1 NBR + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH2 T6L	If signal ANLZ/1 is true, A0-A31 ——→ S0-S31	SXA = (NCC1 + CC2) PH2 ANLZ/1 + . . .	Byte, halfword, or doubleword
	S1-S31 —/→ A0-A30	AXSL1 · (NCC1 + CC2) PH2 ANLX/1 + . . .	Cyclic left shift into A-register
	S0 —/→ A31	A31EN/2 = S0 ALCYC	
		ALCYC = NFAMDSF	
	Set flip-flop PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH3 T6L	If a byte class instruction, A0-A31 ——→ S0-S31	SXA = NCC1 NCC2 PH3 ANLZ/1 + . . .	
	S1-S31 —/→ A0-A30 S0 —/→ A31 If doubleword class, A0-A31 ——→ PR0-PR31	AXSL1 = NCC1 NCC2 PH3 ANLZ/1 + . . .	Cyclic left shift into A-register. Byte address now in bits 30 and 31
	PR0-PR29 —/→ A2-A31	AXPRR2 = CC1 CC2 PH3 ANLZ/1 + . . .	Doubleword class shifted two bit places to the right (16 bit displacement value)
	Set flip-flop DRQ	S/DRQ = PH3 ANLZ + . . .	Inhibits transmission of another clock until signal received

Mnemonic: ANLZ (44, C4)

(Continued)

Table 3-55. Analyze, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3 T6L (Cont.)	Set flip-flop PH4	S/PH4	= PH3 NBR + . . .	Halfword not shifted, leaving 18-bit displacement value. Word not shifted during PH1 and PH2, leaving 17-bit displacement value
	Enable clock T10L	S/T10L	= ANLZ PH3 + . . .	
PH4 T10L	A0-A31 ⟶ S0-S31	SXA	= ANLZ NCC4 PH4 + . . .	
	S0-S31 ⟶ RW0-RW31	RWXS	= RWB0 through RWB3	Write modified address into addressed private memory register
	Generate write byte signals RWB0-RWB3	RWB0-RWB3	= ANLZ/1 PH4 + . . .	
	Reset flip-flop ANLZ	R/ANLZ	= PH4 + CLEAR	
	Generate signal ENDE	ENDE	= ANLZ PH4 + . . .	
	Set flip-flop PRE1	S/PRE1	= ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	

Mnemonic: ANLZ
(44, C4)

Table 3-56. Interpret,Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	At the end of PREP: (O) Opcode (P) Effective word address (Q) Next instruction address (R) R-field		
PH1 T4RL	MB0-MB31 → C0-C31	CXMB = DG	Operand gated from core memory into the C-register when data gate signal received from core memory
	C4-C15 ⇸ D20-D31	DXCR16/2 = DXC/9 + . . . DXC/9 = FUINT PH1 FUINT = OU6 OLB	Bits 4 through 15 are loaded into the even private memory register during PH2
	Enable signal (S/T8L)	(S/T8L) = FUINT EXU NEDNE + . . .	Select clock in PH2
	Zeros ⇸ D0-D19	DX = DXC/9	
PH2 T8L	D0-D31 → S0-S31	SXD = FUINT 07 PH2 + . . .	Load bits 4 through 15 in the even numbered private memory register
	S0-S31 → RW0-RW31 [R]	RWXS/0-RWXS/3 = RW + . . . RW = FUINT PH2 + . . .	
	C0-C31 ⇸ D0-D31	DXC/6 = FUINT PH2 + . . .	Clock operand into the D-register
	Enable signal MRQ/1	MRQ/1 = FUINT PH2 + . . .	Generate memory request for next instruction, as specified by the Q-register
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + . . .	Load next instruction address into the P-register
	Enable signal (S/LR31/2)	(S/LR31/2) = FUINT PH2 + . . .	Used in PH3 for selecting odd numbered private memory register
	Enable signal (S/T8L)	(S/T8L) = FUINT EXU NENDE + . . .	Select clock in PH3
PH3 T8L	S0-S31 → RW0-RW31 [R + 1]	RWXS/0-RWXS/3 = RW + . . . RW = FUINT PH3 + . . .	S0-S31 presently contain all zeros, causing the odd numbered private memory register to be cleared. During PH4, data is written into bits 16 through 31 of the odd numbered private memory register, leaving bits 0 through 15 clear

Mnemonic: INT
(6B, EB)

(Continued)

Table 3-56. Interpret, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T8L (Cont.)	Enable signal (S/LR31/2)	(S/LR31/2) = FUINT PH3 + . . .	Used in the PH4 for selecting odd numbered private memory register
	Enable signal (S/DRQ)	(S/DRQ) = FUINT PH3 + . . .	Inhibits transmission of another clock until data release signal received from core memory
	Enable signal (S/T8L)	(S/T8L) = FUINT EXU NENDE + . . .	Select clock in PH4
PH4 T8L	D0-D31 → S0-S31	SXD = FUINT PH4 NSDIS + . . .	Place operand on the sum bus
	S0 ⇸ CC1	S/CC1 = FUINT PH4 S0 + . . . R/CC1 = (R/CC1)	Load bits 0 through 3 of operand into flip-flops CC1 through CC4, respectively
	S1 ⇸ CC2	S/CC2 = FUINT PH4 S1 + . . . R/CC2 = (R/CC2)	
	S2 ⇸ CC3	S/CC3 = FUINT PH4 S2 + . . . R/CC3 = (R/CC3)	
	S3 ⇸ CC4	S/CC4 = FUINT PH4 S3 + . . . R/CC4 = (R/CC4)	
	S16-S31 → RW16-RW31 [R + 1]	RWXS/2 = RWB2 RWXS/3 = RWB3 RWB2, RWB3 = FUINT PH4 + . . .	Load bits 16 through 31 of the operand into the odd numbered private memory register. Leave bits 0 through 15 of the odd numbered private memory register clear
	Enable signal ENDE	ENDE = FUINT PH4 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + FUEXU) + . . . R/PRE1 = . . .	

Mnemonic: INT
(6B, EB)

Table 3-57. Add Immediate, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP: (O) Opcode (P) Next instruction address (R) R-field (D12-D31) Value field (D0-D11) Extended sign			
	Instruction traps if IA is set	FAILL	= IA NO3 (NO4 NO5) + . . .	
PH1 T4RL	RR0-RR31 ⇸ A0-A31	AXRR FAS12	= FAS12 PH1 + . . . = NO1 O2 OL0 + . . .	Transfer addend from private memory register into the A-register
	Enable signal (S/DRQ)	(S/DRQ)	= FAS12 PH1 + . . .	Inhibits transmission of another clock until data release signal received from core memory
	Enable signal BRPH5	BRPH5	= FAS12 PH1 + . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L)	= FAS12 PH1 + . . .	Select clock in PH5
PH5 T10L	A0-A31 → ADDER CS0-CS31 → ADDER → S0-S31 D0-D31 → ADDER	SXPR/10- SXPR/13 SXADD	 = SXADD + . . . = FAS12 PH5 NSDIS + . . .	Add value field to addend and place result on sum bus
	S0-S31 → RW0-RW31	RWXS/0- RWXS/3 RW	 = RW + . . . = FAS12 PH5 + . . .	Store result in private memory register
	S0-S31 ⇸ A0-A31	AXS	= FAS12 PH5 + . . .	Store result in A-register. Used during next phase for testing result
	If end carry, set flip-flop CC1	S/CC1 (S/CC1/1) K00 R/CC1	= (S/CC1/1) + . . . = FAS12 PH5 K00 = End carry = FAS12 PH5 + . . .	
	If overflow, set flip-flop CC2	S/CC2 OVER PROBEOVER R/CC2	= OVER PROBEOVER + . . . = D0 NK0 NPR0 NFAS10 + . . . = FAS12 PH5 + . . . = PROBEOVER	
	If overflow, and bit 11 of the PSW is a one (trap mask), enable signal TROVER, and set flip-flops TRAP, TR30, TR31, and BRQ	TROVER AM	= AM OVER PROBEOVER + . . . = function of bit 11 of the PSW	Trap to location X'43'

Mnemonic: AI (20)

(Continued)

Table 3-57. Add Immediate,Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T10L (Cont.)		S/TRAP = TROVER + . . .	
		TRAP forces a one into P25 via PXTR	
		R/TRAP = RESET + . . .	
		S/TR30 = TROVER NSTRAP N(S/TRACC4/1)	
		TR30 forces a one into P30 via PXTR	
		R/TR30 = (R/TR)	
		S/TR31 = TROVER N(S/TRACC4/1) NSTRAP NTRAP + . . .	
		TR31 forces a one into P31 via PXTR	
		R/TR31 = (R/TR)	
		S/BRQ = TROVER	BRQ causes the return address to reflect the address of the instruction being executed at the time of the trap
	Set flip-flop TESTA	S/TESTA = FAS12 PH5 + . . .	
		R/TESTA = . . .	
	Enable signal ENDE	ENDE = FAS12 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + NFUEXU) + . . .	
		R/PRE1 = . . .	
Next clock	Test word in A-register, and set flip-flops CC3 and CC4		
	If word is positive, set flip-flop CC3; if word is negative, set flip-flop CC4; if word is zero, reset flip-flops CC3 and CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z = A-register does not contain all zeros	
		R/CC3 = TESTA	
		S/CC4 = A0 TESTA NTESTA/1 + . . .	
		R/CC4 = TESTA	

Mnemonic: AI (20)

Table 3-58. Add Halfword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation phases except as follows:		
	RR14-RR30 —/→ A15-A31 in PRE1	AXRRR1 = OU5 INDX PRE1	Index value moved one bit position to the right
	RR31 —/→ P32	S/P32 = RR31 AXRRR1	Halfword address bit
PH1	MB0-MB31 ——→ C0-C31	CXMB = DG	Operand gated from core memory into the C-register when data gate signal received from core memory
	If NP32		
	C0-C15 —/→ D16-D31	DXCR16 = DXC/5 NP32 + . . .	Bits 0 through 15 of operand clocked into D16-D31
		DXC/5 = OU5 NO4 NO5 PH1 + . . .	
	If P32		
	C16-C31 —/→ D16-D31	DXC/12, DXC/13 = DXC/7 + . . .	Bits 16 through 31 of the operand gated into D16-D31
		DXC/7 = DXC/2 + . . .	
		DXC/2 = DXC/5 P32 + . . .	
	If NP32		
	MB0 —/→ D0-D11	C0C16C12 = MB0 NP32 CXMB/4 (. . .) + . . .	Extend sign bit (MB0) into D0-D15
		CXMB/4 = DGC	
	MB0 —/→ D12-D15	C0C16/1 = MB0 NP32 CXMB/4 (. . .) + . . .	
	If P32		
	MB16 —/→ D0-D11	C0C16C12 = MB16 P32 CXMB/4 (. . .) + . . .	Extend sign bit (MB16) into D0-D15
	MB16 —/→ D12-D15	C0C16/1 = MB16 P32 CXMB/4 (. . .) + . . .	
	RR0-RR31 —/→ A0-A31	AXRR = FAS12 PH1 + . . .	Transfer augend from the fast memory register to the A-register
		FAS12 = OU5 NO5 NO6 NO7 + . . .	
	Enable signal (S/DRQ)	(S/DRQ) = FAS12 PH1 + . . .	Inhibits transmission of another clock until data release signal received from core memory
	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Load next instruction address into the P-register
		PRERQ = O1 O3 NO4 NO5 (. . .) + . . .	
	Enable signal BRPH5	BRPH5 = FAS12 PH1 + . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L) = FAS12 PH1 + . . .	Select clock in PH5

Mnemonic: AH (50, D0)

(Continued)

Table 3-58. Add Halfword Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH5 T10L	A0-A31 ↓ CS0-CS31 → ADDER → S0-S31 D0-D31 ↑	SXPR/10- SXPR/13	= SXADD + . . .	Add halfword (D-register) to augend (A-register) and place result on sum bus
		SXADD	= FAS12 PH5 NSDIS + . . .	
	S0-S31 → RW0-RW31	RWXS/0- RWXS/3	= RW + . . .	Store result in private memory register
		RW	= FAS12 PH5 + . . .	
	S0-S31 —/→ A0-A31	AXS	= FAS12 PH5 + . . .	Store result in A-register. Used during next phase for testing result
	If end carry, set flip-flop CC1	S/CC1	= FAS12 PH5 K00 + . . .	
		K00	= End carry	
		R/CC1	= FAS12 PH5 + . . .	
	If overflow, set flip-flop CC2	S/CC2	= OVER PROBEOVER + . . .	
		OVER	= D0 NK0 NPR0 NFAS10 + . . .	
		PROBEOVER	= FAS12 PH5 + . . .	
		R/CC2	= PROBEOVER	
	If overflow, and bit 11 of the PSW is a one (trap mask), enable signal TROVER, and set flip-flops TRAP, TR30, TR31, and BRQ	TROVER	= AM OVER PROBEOVER + . . .	Trap to location X'43'
		AM	= Function of bit 11 of the PSW	
		S/TRAP	= TROVER + . . .	
		TRAP forces a one into P25 via PXTR		
		R/TRAP	= RESET + . . .	
		S/TR30	= TROVER NSTRAP N(S/TRACC4/1) + . . .	
		TR30 forces a one into P30 via PXTR		
		R/TR30	= (R/TR)	
		S/TR31	= TROVER N(S/TRACC4/1) NSTRAP NTRAP + . . .	
		TR31 forces a one into P31 via PXTR		
		R/TR31	= (R/TR)	
		S/BRQ	= TROVER	
	Set flip-flop TESTA	S/TESTA	= FAS12 PH5 + . . .	
		R/TESTA	= . . .	
	Enable signal ENDE	ENDE	· FAS12 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1	= ENDE N(S/INTRAPF) (NHALT + NFUEXU) + . . .	
		R/PRE1	= . . .	

Mnemonic: AH (50, D0)

(Continued)

Table 3-58. Add Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
Next clock	Test word in A-register, and set flip-flops CC3 and CC4		
	If word is positive, set flip-flop CC3; if word is negative, set flip-flop CC4; if word is zero, reset flip-flops CC3 and CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z = A-register does not contain all zeros	
		R/CC3 = TESTA	
		S/CC4 = A0 TESTA NTESTA/1 + . . .	
		R/CC4 = TESTA	

Mnemonic: AH (50, D0)

The program traps to location X'43' after loading the results into private memory, if overflow has occurred, and the fixed-point arithmetic trap mask bit of the PSD is set.

A sequence chart of the Add Word instruction is given in table 3-59.

3-193 Add Doubleword (AD 10, 90)

Instruction AD causes the effective doubleword from core memory to be added to the contents of an even and an odd numbered private memory register. The low order 32 bits of the sum are stored in the odd numbered private memory register, and the 32 high order bits are stored in the even numbered private memory register.

The one's complement of low order half of the core memory doubleword is loaded into the D-register. The low order half of the doubleword in the selected private memory registers is loaded into the A-register. Flip-flop CS31 is set to form the two's complement of the word in the D-register. The A-, D-, and CS-register outputs are gated into the adder, and the addition is an effective subtraction of the core memory word from the private memory word. The result is loaded into the odd numbered private memory register, and into the A-register to be tested for a zero condition. The one's complement of the high order half of the core memory doubleword is loaded into the D-register, and the high order half of the private memory doubleword is loaded into the A-register. The A-, the D-, and the CS-register outputs are gated into the adder, and the addition completes the effective subtraction mentioned above. The result is transferred to the even numbered private memory register.

A sequence chart of the Add Doubleword instruction is given in table 3-60.

3-194 Subtract Halfword (SH 58, D8)

The SH instruction causes a word to be read from a referenced memory address. The instruction selects the desired halfword and extends the sign bit 16 bit positions to the left. The two's complement is obtained; the resulting 32-bit word is added to the data word contained in the addressed private memory register. The sum is stored in the private memory register.

The index value is read into the A-register, and is shifted one bit position to the right, during the preparation phases. The halfword select bit in the least significant position is clocked into flip-flop P32. The operand address in the D-register and the indexing data in the A-register are placed in the adder, and the modified program address is clocked into the P-register.

During the execution phases, the one's complement of the most or the least significant halfword, depending on the state of flip-flop P32, is clocked into the D-register. Flip-flop CS31 is set to form the two's complement. The data word in the selected private memory register is loaded into the A-register. These values are added in the adder which results in algebraically subtracting the core memory data word from the data word in private memory. The result is transferred to the selected private memory register, and to the A-register to test for a non-zero value and for condition code information. The program traps to location X'43' after loading the contents of the sum bus into private memory, if the sign bit changes, which indicates fixed-point overflow, and if the fixed-point arithmetic trap mask bit in the PSD is set.

A sequence chart of the Subtract Halfword instruction is given in table 3-61.

3-195 Subtract Word (SW 38, B8)

The SW instruction causes a word to be read from a referenced memory address. The two's complement of the word is obtained and is added to the data word contained in the addressed private memory register. The sum is stored in the private memory register.

The one's complement of the word in memory is clocked into the D-register. Flip-flop CS31 is set to form the two's complement in the adder. The data word from the selected private memory register is loaded into the A-register. The outputs of the A- and D-registers are gated into the adder where the addition performed results in the effective algebraic subtraction of the core memory word from the word in private memory. The result is transferred to the selected private memory register, and to the A-register to be tested for overflow and condition code information. The program traps to location X'43' after loading the result into the private memory register, if the sign bit changes, which indicates fixed-point overflow, and the fixed-point arithmetic trap mask bit in the PSD is set.

Table 3-62 is a sequence chart of the Subtract Word instruction.

3-196 Subtract Doubleword (SD 18, 98)

The SD instruction causes two data words to be read from successive core memory locations. The two's complement of the 64-bit data word is obtained and is added to the contents of an even and an odd numbered private memory register. The low order 32-bits of the sum are then stored in the memory register, and also in the A-register where they are tested for overflow and condition code information. If fixed-point overflow occurs, and if the fixed-point arithmetic trap mask bit in the PSD is set, the program traps to location X'43' after loading the result into the private memory registers.

A sequence chart of the Subtract Doubleword is given in table 3-63.

The low order half of the addend from core memory is loaded into the D-register, and the low order half of the augend from the odd numbered private memory register is loaded into the A-register. The two values are gated into the adder, and the sum is loaded into the odd numbered private memory register, and into the A-register to be tested for a nonzero condition.

Table 3-59. Add Word, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	At the end of PREP: (O) Opcode (P) Effective address (Q) Next instruction address (R) R-field			
PH1 T4RL	MB0-MB31 ⟶ C0-C31	CXMB	= DG	Addend gated from core memory into the C-register when data gate signal received from core memory.
	C0-C31 –/→ D0-D31	DXC/10- DXC/13	= DXC/6 + . . .	Clock addend into the D-register
		DXC/6	= OU3 NO4 NO5 PH1 + . . .	
	RR0-RR31 –/→ A0-A31	AXRR-0, AXRR-1, AXRR/12, AXRR/13	= AXRR + . . .	Transfer augend from fast memory register into the A-register
		AXRR	= FAS12 PH1 + . . .	
		FAS12	= OU3 NO5 NO6 NO7 + . . .	
	Enable signal (S/DRQ)	(S/DRQ)	= FAS12 PH1 + . . .	Inhibits transmission of another clock until data release signal received from core memory
	Q15-Q31 –/→ P15-P31	PXQ	= PRERQ PH1 + . . .	Load next instruction address into the P-register
		PRERQ	= OU3 NO5 (. . .) + . . .	
	Enable signal BRPH5	BRPH5	= FAS12 PH1 + . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L)	= FAS12 PH1 + . . .	Select clock in PH5
PH5 T10L	A0-A31 ↓ CS0-CS31 ⟶ ADDER ⟶ S0-S31 D0-D31 ↑	SXPR/10- SXPR/13	= SXADD + . . .	Add addend (D-register) to augend (A-register) and place result on sum bus
		SXADD	= FAS12 PH5 NSDIS	
	S0-S31 ⟶ RW0-RW31	RWXS/10- RWXS/13	= RW + . . .	Store result in fast memory register
		RW	= FAS12 PH5 + . . .	
	S0-S31 ⟶ A0-A31	AXS	= FAS12 PH5 + . . .	Store result in A-register. Used during next phase to test result
	Set flip-flop TESTA	S/TESTA	= FAS12 PH5 + . . .	
		R/TESTA	= . . .	

Mnemonic: AW (30, B0)

(Continued)

Table 3-59. Add Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T10L (Cont.)	Enable signal ENDE	ENDE = FAS12 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1 = ENDE N(S/INTRAPF) (NHALT + NFUEXU) + . . .	
		R/PRE1 = . . .	
	If end carry, set flip-flop CC1	S/CC1 = FAS12 PH5 K00 + . . .	
		K00 = End carry	
		R/CC1 = FAS12 PH5 + . . .	
	If overflow, set flip-flop CC2	S/CC2 = OVER PROBEOVER + . . .	
		OVER = D0 NK0 NPR0 NFAS10 +. . .	
		PROBEOVER = FAS12 PH5 + . . .	
		R/CC2 = PROBEOVER	
	If overflow and bit 11 of the PSW is a one (trap mask), enable signal TROVER and set flip-flops TRAP, TR30, TR31, and BRQ	TROVER = AM OVER PROBEOVER + . . .	Trap to location X'43'
		AM = Function of bit 11 of the PSW	
		S/TRAP = TROVER + . . .	
		TRAP forces a one into P25 via PXTR	
		R/TRAP = RESET + . . .	
		S/TR30 = TROVER NSTRAP N(S/TRACC4/1) + . . .	
		TR30 forces a one into P30 via PXTR	
		R/TR30 = (R/TR)	
		S/TR31 = TROVER NSTRAP NTRAP N(S/TRACC4/1) + . . .	
		TR31 forces a one into P31 via PXTR	
		R/TR31 = (R/TR)	
		S/BRQ = TROVER	
Next clock	Test word in A-register, and set flip-flops CC3 and CC4		
	If word is positive, set flip-flop CC3; if word is negative, set flip-flop CC4; if word is zero, reset flip-flops CC3 and CC4	S/CC3 = NA0 NA0031Z TESTA (. . .) + . . .	
		NA0031Z = A-register does not contain all zeros	
		R/CC3 = TESTA	
		S/CC4 = A0 TESTA NTESTA/1 + . . .	
		R/CC4 = TESTA	

Mnemonic: AW (30, B0)

Table 3-60. Add Doubleword Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence except		
	Force a one on LB31 address line to select odd numbered core memory word location	LB31 = PREFADO LMXC NC0	
	During PRE2, force a zero on LB31 address line to select even numbered core memory word by inhibiting flip-flop P31 from setting	S/P31 = S31 PXS NP31Z	
	Generate a memory request for second half of doubleword	MRQ = PRE2 NIA PREDO NANLZ	
	Force a one on LR31 address line to select odd numbered private memory register	S/LR31/2 = (S/PH1/1) (FAS3 N04 N05 N06)	
PH1 T4RL	C0-C31—/→D0-D31	DXC = (N01 03) (N04 N05 N07) PH1	Low order half of doubleword from core memory
	RR0-RR31—/→A0-A31	AXRR = FAS22 PH1	Low order half of doubleword from private memory
	Set flip-flop DRQ	S/DRQ = PREDO PH1	Inhibits transmission of another clock until signal received from memory
	Force a one on LR31 address line	S/LR31/2 = OU1 (N05 N07) PH1	Selects odd numbered private memory register
PREP	<u>At the end of PREP</u>:		
	(O) Opcode		
	(P) Effective address		
	(P31) Reset		
	LB31 = 1 during PRE1	LB31 = PREFADO LMXC NC0 + . . .	Memory request for low order word was made in PRE1, for high order word in PRE2 by inhibiting setting of P31
	LB31 = 0 during PRE2	S/P31 = S31 PXS NP31Z	
	(Q) Next instruction address	P31Z = PRE2 NIA FADW	
	(R) R-field		
	(LR31) Set		Selects R + 1 in PH1
PH1 T4RL	MB0-MB31——→C0-C31	CXMB = DG	Low order half of addend doubleword gated into the C-register from core memory when data gate signal received from core memory

Mnemonic: AD (10, 90)

(Continued)

Table 3-60. Add Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL (Cont.)	C0-C31 —/→ D0-D31	DXC/10- DXC/13 = DXC/6 + . . .	Clock low order half of addend doubleword into the D-register
		DXC/6 = NO1 O3 NO4 NO5 NO7 PH1 + . . .	
	RR0-RR31 —/→ A0-A31 [R + 1]	AXRR = FAS22 PH1 + . . .	Transfer low order half of augend doubleword from the odd fast memory register into the A-register
		FAS22 = OU1 NO5 NO6 + . . .	
	Set flip-flop RQ	S/RQ = OU1 NO5 NO6 PH1 + . . .	Memory request for next instruction
	Enable signal (S/DRQ)	(S/DRQ) = PREDO PH1 + . . .	Inhibits transmission of another clock until data release signal received from core memory
		PREDO = OU1 NO5 NO7 + . . .	
	Enable signal (S/LR31/2)	(S/LR31/2) = OU1 NO5 NO7 PH1 + . . .	Used in PH2 for selecting odd numbered fast memory register
	Enable signal (S/T10L)	(S/T10L) = FAS3 PH1 + . . .	Select clock in PH1
		FAS3 = OU1 NO5 NO6 NO7	
PH2	A0-A31 ↓ CS0-CS31 → ADDER → S0-S31 D0-D31 ↑	SXPR/10- SXPR/13 = SXADD + . . .	Add low order half of addend doubleword to low order half of augend doubleword, and place result on sum bus
		SXADD = FAS22 PH2 NSDIS + . . .	
	If end carry is detected as a result of the low order addition, reset flip-flop K00H	S/K00H = S00 + . . .	
		S00 = Function of NK00	
		K00 = End carry present	
		R/K00H = . . .	
	S0-S31 —/→ A0-A31	AXS = FAS22 PH2 + . . .	Store result in A-register. Used in PH3 for testing result
	S0-S31 → RW0-RW31 [R + 1]	RWXS/0- RWXS/3 = RW + . . .	Store result in odd fast memory register
		RW = FAS3 PH2 + . . .	
	MB0-MB31 → C0-C31	CXMB = DG	High order half of addend doubleword gated into the C-register from core memory when data gate signal received from core memory

Mnemonic: AD (10, 90)

(Continued)

Table 3-60. Add Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH2 (Cont.)	C0-C31 ─/→ D0-D31	DXC/10-DXC/13	= DXC/6 + . . .	Clock high order half of addend doubleword into the D-register
		DXC/6	= FUAD PH2 + . . .	
		FUAD	= OU1 OL0	
	Enable signal T6RL	T6RL	= FAS3 PH2 + . . .	Select clock in PH3
PH3 T6RL	Test contents of A-register. If nonzero, set flip-flop BWZ	S/BWZ	= (S/BWZ/1) NA0031Z + . . .	Store nonzero condition in flip-flop BWZ
		(S/BWZ/1)	= FAS22 NOL9 PH3	
		NA0031Z	= nonzero condition	
		R/BWZ	= CLEAR	
	RR0-RR31 ─/→ A0-A31 [R]	AXRR	= FAS22 PH3 + . . .	Transfer high order half of augend doubleword from the even fast memory register to the A-register
	If flip-flop K00H was reset in PH2, set flip-flop CS31	S/CS31	= CSX1/8 + . . .	Store end carry from low order addition in flip-flop CS31
		CSX1/8	= (S/BWZ/1) NK00H NOLB + . . .	
		(S/BWZ/1)	= FAS22 NOL9 PH3 + . . .	
		R/CS31	= (R/CS31)	
	Enable signal (S/DRQ)	(S/DRQ)	= FAS22 PH3 + . . .	Inhibits transmission of another clock until data release signal received from core memory
	Q15-Q31 ─/→ P15-P31	PXQ	= PREDO PH3 + . . .	Load next instruction address into the P-register
	Enable signal BRPH5	BRPH5	= FAS22 PH3 + . . .	Advance to PH5
	Enable signal (S/T10L)	(S/T10L)	= FAS3 PH3 + . . .	Select clock in PH5
PH5 T10L	CS31 ─┐ A0-A31 ──→ ADDER ──→ S0-S31 D0-D31 ─┘	SXPR/10-13	= SXADD + . . .	Add high order half of addend doubleword to high order half of augend doubleword; incorporate end carry if any, and place result on sum bus
		SXADD	= FAS22 PH5 NSDIS + . . .	
	S0-S31 ──→ RW0-RW31 [R]	RWXS/0-RWXS/3	= RW + . . .	Store result in even fast memory register
		RW	= FAS3 PH5 + . . .	
	S0-S31 ─/→ A0-A31	AXS	= FAS22 PH5 + . . .	Used for testing result during next clock
	If BWZ, 1 ─/→ A31	S/A31	= S31 AXS-3 + A31X1 + . . .	
		A31X1	= FAS22 BWZ + . . .	

Mnemonic: AD (10, 90)

(Continued)

Table 3-60. Add Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH5 T10L (Cont.)	Set flip-flop TESTA	S/TESTA	FAS22 PH5 + . . .	
		R/TESTA	. . .	
	Enable signal ENDE	ENDE	= FAS22 PH5 + . . .	
	Set flip-flop PRE1 for the next instruction	S/PRE1	ENDE N(S/INTRAPF) (NHALT + NFUEXU) + . . .	
	If end carry, set flip-flop CC1	S/CC1	FAS3 PH5 K00 + . . .	
		K00	= End carry	
		R/CC1	FAS3 PH5 + . . .	
	If overflow, set flip-flop CC2	S/CC2	OVER PROBEOVER + . . .	
		OVER	NPRO NFAS10 (D0 NK0 + NA0 ND0 K0)	
		PROBEOVER	= FAS3 PH5 + . . .	
		R/CC2	= PROBEOVER	
	If overflow and bit 11 of the PSW is a one (trap mask), enable signal TROVER and set flip-flops TRAP, TR30, and TR31	TROVER	AM OVER PROBEOVER + . . .	Trap to location X'43'
		AM	Function of bit 11 of the PSW	
		S/TRAP	TROVER + . . .	
		TRAP forces a one into P25 via PXTR		
		R/TRAP	RESET + . . .	
		S/TR30	= TROVER NSTRAP N(S/TRACC4/1) + . . .	
		TR30 forces a one into P30 via PXTR		
		R/TR30	= (R/TR)	
		S/TR31	TROVER NSTRAP NTRAP N(S/TRACC4/1) + . . .	
		TR31 forces a one into P31 via PXTR		
		R/TR31	= (R/TR)	
Next clock	Test word in A-register, and accordingly, set flip-flops CC3 and CC4			
	If word is positive (indicates that doubleword sum in private memory register is positive), set flip-flop CC3; if word is negative (indicates that doubleword sum in private memory register is negative), set flip-flop CC4; if word is zero (indicates that doubleword sum in private memory register is zero), reset flip-flops CC3 and CC4	S/CC3	= NA0 NA0031Z TESTA (. . .)	
		NA0031Z	= A-register does not contain all zeros	
		R/CC3	= TESTA	
		S/CC4	= A0 TESTA NTESTA/1 + . . .	
		R/CC4	= TESTA	

Mnemonic: AD (10, 90)

Table 3-61. Subtract Halfword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1 T6L	D12-D14 → LR29-LR31	LRXD = ENDE	Selects index register containing halfword select bit
	Generate signal INDX	INDX = (C12 + C13 + C14) (C3 + C4 + C5)	
	P15-P31 ⇸ Q15-Q31	QXP = PRE1 NANLZ + . . .	
	RR14-RR30 ⇸ A15-A31	AXRRR1 = OU5 INDX PRE1	Index data gated into A-register shifted one bit right
	RR31 ⇸ P32	S/P32 = RR31 AXRRR1	Halfword select bit
	Set flip-flop IX	S/IX = INDX PRE1	
	Set flip-flop PRE2	S/PRE2 = NPREIM PRE1 N(S/INTRAPF)	
	Enable T4RL	T4RL = PREP + . . .	
PRE2 T4RL	A15-A31 + D15-D31 → S15-S31	SXADD = PRE2 NIA + . . .	Modified program address
	S15-S31 ⇸ P15-P31	PXS = PRE2 NIA + . . .	
	P15-P31 → LM15-LB31	(NLMXC NLMXQ) = PRE2 NIA NRIP + . . .	Address of operand from core memory
	R28-R31 → LR28-LR31	(LRXR NLRXLB) = NENDE	Address of operand from private memory
	Generate memory request for next instruction	S/RQ = PRE2 NIA IX PRERQ + . . .	
	Set flip-flop PH1	S/PH1 = NPREDO PRE2 NIA +. . .	
	Enable T4RL	T4RL = PREP + . . .	
	Reset IX flip-flop	R/IX PRE2 NIA	
PH1 T4RL	Q15-Q31 ⇸ P15-P31	PXQ = PRERQ PH1 + . . .	Address of next instruction
	If NP32 is true, NC0-NC15 ⇸ D15-D31 and NMB0 ⇸ D0-D15	DXNCR16-2 = OU5 (04 N05 N07) PH1 NP32 C0C16C12 NMB0 NP32 CXMB/4	Most significant halfword Extend sign
	If P32 is true, NC16-NC31 ⇸ D16-D31 and NMB16 ⇸ D0-D15	DXNC OU5 (04 N05 N07) PH1 P32 C0C16C12 NMB16 P32 CXMB/4	Least significant halfword Extend sign D-register contains inverse of selected halfword with inverted sign intended 16 places to the left
	Force a one into flip-flop CS31	S/CS31 CSX1 = OU5 (04 N05 N07) PH1	To obtain two's complement
	RR0-RR31 ⇸ A0-A31	AXRR = FAS12 PH1 + . . .	Addend from private memory

Mnemonic: SH (58, D8)

(Continued)

Table 3-61. Subtract Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH1 T4RL (Cont.)	Set flip-flop DRQ	S/DRQ	= FAS12 PH1 + . . .	Inhibits transmission of another clock until data release signal received
	Set flip-flop PH5	S/PH5	= BRPH5 = FAS12 PH1 + . . .	
	Enable clock T10L	S/T10L	= FAS12 PH1 + . . .	
PH5	A0-A31 + D0-D31 + CS31 ⟶ S0-S31	SXPR	= SXADD = FAS12 PH5 + . . .	Addition operation performed
T10L	S0-S31 ⟶ RW0-RW31	RWXS	= FAS12 PH5	
	Generate write byte signals RWB0-RWB3	RWB0-RWB3	= RW = FAS12 PH5	Sum stored in addressed private memory register
	S0-S31 ↛ A0-A31	AXS	= FAS12 PH5 + . . .	Result into A-register for testing
	Set flip-flop TESTA	S/TESTA	= FAS12 PH5 + . . .	
	Generate signal ENDE	ENDE	= FAS12 PH5 + . . .	
	If word in A-register is all zeros, reset flip-flops CC3 and CC4; if word is negative (A0), set flip-flop CC4 and reset flip-flop CC3; if word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3	= NTESTA/1 TESTA NA0 + . . . NA0031Z	
		R/CC3	= TESTA + . . .	
		S/CC4	= NTESTA/1 TESTA A0 + . . .	
		R/CC4	= TESTA + . . .	
	If fixed-point overflow occurs, set flip-flop CC2; if a carry exists from bit position zero, set flip-flop CC1	S/CC2	= PROBEOVER OVER + . . .	
		R/CC2	= PROBEOVER + . . .	
		S/CC1	= FAS12 PH5 K00 + . . .	
		R/CC1	= FAS12 PH5 + . . .	
	If fixed-point overflow occurs and fixed-point arithmetic trap mask (bit 11 of PSD) is a one, computer traps to location X'43'	S/TRAP	= TROVER + . . .	
		R/TRAP	= RESET + . . .	
		TROVER	= PROBEOVER AM OVER + . . .	
		S/AM	= PSW1XS S11	
	Set flip-flop PRE1	S/PRE1	= ENDE (NHALT + FUEXU) N(S/INTRAPF) + . . .	
	Enable clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
		PROBEOVER	= FAS12 PH5	
		OVER	= NA0 ND0 K0 + D0 NK0	
		S/BRQ	= TROVER	

Mnemonic: SH (58, D8)

Table 3-62. Subtract Word Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence except that during PRE1 a memory request is generated for the next instruction	S/RQ = NIA PRE1 NINDX PRERQ + . . .	
PH1 T4RL	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	Address of next instruction
	NC0-NC15 —/→ D0-D15	DXNC/1 = (N01 03) (04 N05 N07) PH1	One's complement of data word from core memory
	Force a one into flip-flop CS31	S/CS31 = DXNC/1	To obtain two's complement
	RR0-RR31 —/→ A0-A31	AXRR = FAS12 PH1 + . . .	Data word from private memory
	Set flip-flop DRQ	S/DRQ = FAS12 PH1 + . . .	Inhibits transmission of another clock until data signal received from memory for next instruction
	Set flip-flop PH5	S/PH5 = BRPH5 = FAS12 PH1 + . . .	
	Enable clock T10L	S/T10L = FAS12 PH1 + . . .	
PH5 T10L	A0-A31 + D0-D31 + CS31 ——→ S0-S31	SXADD = FAS12 PH5 + . . .	Algebraic subtraction of data word from core memory from data word from private memory
	S0-S31 ——→ RW0-RW31	RWXS = FAS12 PH5 + . . .	
	Generate write byte signals RWB0-RWB3	RWB0-RWB3 = RW FAS12 PH5 + . . .	Result of subtraction stored in private memory register
	S0-S31 —/→ A0-A31	AXS = FAS12 PH5 + . . .	For condition code and overflow tests
	Generate signal ENDE	ENDE = FAS12 PH5 + . . .	
	Set flip-flop TESTA	S/TESTA = FAS12 PH5 + . . .	
	If word in A is all zeros, reset flip-flops CC3 and CC4; if word is negative (A0), set flip-flop CC4 and reset flip-flop CC3; if word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3 = NTESTA/1 TESTA NA0 + . . . NA0031Z	
		R/CC3 = TESTA + . . .	
		S/CC4 = NTESTA/1 TESTA A0 + . . .	
		R/CC4 = TESTA + . . .	
	If fixed-point overflow occurs, set flip-flop CC2; if a carry exists from bit position zero, set flip-flop CC1	S/CC2 = PROBEOVER OVER + . . .	
		R/CC2 = PROBEOVER + . . .	
		S/CC1 = FAS12 PH5 K00 + . . .	
		R/CC1 = FAS12 PH5 + . . .	

Mnemonic: SW (38, B8)

(Continued)

Table 3-62. Subtract Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved			Comments
PH5 T10L (Cont.)	If fixed-point overflow occurs and fixed-point arithmetic trap mask is a one, computer traps to location X'43'	S/TRAP	-	TROVER + . . .	
		TROVER	-	PROBEOVER AM OVER + . . .	
		S/AM		PXW1XS S11	

	Set flip-flop PRE1	S/PRE1	=	ENDE (NHALT + FUEXU)
				N(S/INTRAPF)	

	Enable clock T6L	T6L		NT1L NT4L NT8L NT10L NRESET	
		PROBEOVER		FAS12 PH5 + . . .	
		OVER		NA0 ND0 K0 + D0 NK0	
		S/BRQ	-	TROVER	

Mnemonic: SW (38, B8)

Table 3-63. Subtract Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence except		
	Force a one onto LB31 address lines	LB31 = PREFADO LMXC NC0 + . . .	Selects address for low order half of doubleword
	Force a one onto LR31 address line	S/LR31/2 = (S/PH1/1) OU1 NO4 NO5 NO6 + . . .	Selects low order half of doubleword from private memory

	During PRE2, force a zero onto LB31 address line by inhibiting flip-flop P31 from setting	NP31Z = NPRE2 + IA + NFADW
		P31Z = PRE2 NIA FADW
			Selects even numbered core memory location for high order half of doubleword

	Generate a memory request for high order half of doubleword	MRQ = PRE2 NIA PREDO NANLZ + . . .	
PH1 T4RL	NC0-NC31 —/→ D0-D31	DXNC = FAS2 PH1 + . . .	Complement of low order half of data word
	RR0-RR31 —/→ A0-A31	AXRR = FAS22 PH1 + . . .	Low order half of data word from private memory
	Set flip-flop DRQ	S/DRQ = PREDO PH1 + . . .	Inhibits transmission of another clock until data signal received
	Force a one onto LR31 address line	S/LR31/2 = OU1 (NO5 NO7) PH1 + . . .	Odd numbered private memory register to store low order half of sum
	Generate memory request for next instruction	S/RQ = OU1 (NO5 NO7) PH1 + . . .	
	Force a one into flip-flop CS31	S/CS31/1 = DXNC	To form two's complement of memory word
	Set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
	Enable clock T10L	S/T10L = FAS3 PH1 + . . .	
PH2	A0-A31 + D0-D31 + CS31 ——→ S0-S31	SXPR = SXADD = FAS22 PH2 + . . .	Addition of low order half of doubleword
	If end carry occurs, reset flip-flop K00H	S/K00H = NK00 + . . .	
	S0-S31 ——→ RW0-RW31	RWXS = FAS3 PH2 + . . .	
	Generate write byte signals RWB0-RWB3	RWB0-RWB3 = RW = FAS3 PH2 + . . .	Sum of low order addition stored in odd numbered private memory register
	S0-S31 —/→ A0-A31	AXS = FAS22 PH2 + . . .	For zero testing
	MB0-MB31 —/→ C0-C31	CXMB = DGC	High order half of doubleword from core memory

Mnemonic: SD (18, 98)

(Continued)

Table 3-63. Subtract Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH2 (Cont.)	NC0-NC31 —/→ D0-D31	DXNC	= FAS2 PH2 + . . .	Complement of high order half of doubleword
	Set flip-flop PH3	S/PH3	= PH2 NBR + . . .	
	Enable T6RL	T6RL	= FAS3 PH2 + . . .	
PH3 T6RL	Check A-register for nonzero condition. If nonzero, set flip-flop BWZ	S/BWZ	= NA0031Z FAS22 NOL9 PH3 + . . .	
	RR0-RR31 —/→ A0-A31	AXRR	= FAS22 PH3 + . . .	High order half of doubleword from private memory
	If NK00H in PH2, force a one into flip-flop CS31	S/CS31	= NK00H FAS22 NOL9 PH3 + . . .	End carry from previous addition
	Set flip-flop DRQ	S/DRQ	= FAS22 PH3 + . . .	Inhibits transmission of another clock until data signal received from core memory for next instruction
	Q15-Q31 —/→ P15-P31	PXQ	= PREDO PH3 + . . .	Address of next instruction
	Set flip-flop PH5	S/PH5	= BRPH5 = FAS22 PH3 + . . .	
	Enable clock T10L	S/T10L	= FAS3 PH3 + . . .	
PH5 T10L	A0-A31 + D0-D31 + CS31 ——→ S0-S31	SXPR	= SXADD = FAS22 PH5 + . . .	Addition of high order half of doubleword
	S0-S31 ——→ RW0-RW31	RWXS	= FAS3 PH5 + . . .	
	Generate write byte signal RWB0-RWB3	RWB0-RWB3	= RWXS	Sum of high order addition stored in even numbered private memory register
	S0-S31 —/→ A0-A31	AXS	= FAS22 PH5 + . . .	For overflow and condition code testing
	Force a one into flip-flop A31 if flip-flop BWZ was set during PH3	S/A31	= FAS22 BWZ + . . .	
	Generate signal ENDE	ENDE	= FAS22 PH5 + . . .	
	Set flip-flop TESTA	S/TESTA	= FAS22 PH5 + . . .	
	If word in A is all zeros, reset flip-flops CC3 and CC4; if word is negative (A0), set flip-flop CC4 and reset flip-flop CC3; if word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3 R/CC3 S/CC4 R/CC4	= NTESTA/1 TESTA NA0 + . . . NA0031Z = TESTA + . . . = NTESTA/1 TESTA A0 + . . . = TESTA + . . .	

Mnemonic: SD (18, 98)

(Continued)

Table 3-63. Subtract Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T10L (Cont.)	If fixed-point overflow occurs, set flip-flop CC2; if an end carry occurs, set flip-flop CC1	S/CC2 = OVER PROBEOVER + . . . R/CC2 = PROBEOVER + . . . S/CC1 = FAS3 PH5 K00 + . . . R/CC1 = FAS3 PH5 + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE NHALT N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

	If fixed-point overflow occurs and fixed-point arithmetic trap mask (bit 11 of PSD) is a one, trap to location X'43'	S/TRAP = TROVER + . . .
		TROVER = PROBEOVER AM OVER + . . .
		S/AM = PSW1XS S11
		PROBEOVER = FAS3 PH5 + . . .
		OVER = NA0 ND0 K0 + D0 NK0	

Mnemonic: SD (18, 98)

The high order half of the addend is then loaded into the D-register, and the high order half of the augend from the even numbered private memory register is clocked into the A-register. These two values are gated into the adder, and the sum is clocked into the even numbered private memory register, and into the A-register to be tested for condition code and trap information. The program traps to location X'43' after loading the result into private memory if fixed-point overflow occurs, and if the fixed-point arithmetic trap mask bit in the PSD is set.

3-197 Multiply Word (MW 37, B7), Multiply Immediate (MI 23), and Multiply Halfword (MH 57, D7)

Instructions MW, MI, and MH are similar in implementation and result and are discussed as a group. Specific differences are noted in the discussion.

Instruction MW multiplies the contents of an odd numbered private memory register by the effective word from core memory. If the R-field of the instruction word is an even value, the 32 lower order bits of the product are stored in the odd numbered private memory register, and the 32 high order bits of the product are stored in the even numbered private memory register. If the R-field is an odd value, only the 32 lower order bits of the product are stored in the odd numbered private memory register.

Instruction MI extends the sign of the value field (bit position 12) of the instruction word 12 bit positions to the left. The resulting 32-bit value is then multiplied by the contents of an odd numbered private memory register in the same manner as instruction MW.

Instruction MH multiplies the contents of bit positions 16 through 31 of an even or odd numbered private memory register by the effective halfword from core memory. The 32-bit product is stored back in the odd numbered register if the R-field of the instruction word is odd. If the R-field is even, the 32-bit product is stored in the odd numbered private memory register; the even numbered register retains the original multiplier.

Instructions MW and MI utilize condition code bits 2 through 4. Instruction MH uses only the third and fourth bits of the condition code. Table 3-64 lists the condition codes for the three instructions.

3-198 MULTIPLICATION BY BIT-PAIRS AND CARRY-SAVE ADDITION.

The Sigma 7 computer uses bit-pair multiplication and carry-save addition for implementation of instructions MW, MI, and MH. A review of single-bit multiplication may help in understanding bit-pair multiplication techniques.

Single-bit multiplication by repeated addition is illustrated in figure 3-154. In this example, four-bit registers are used for both the multiplicand (the number to be multiplied) and the multiplier. An eight-bit register is used to hold the product. (The number of bits in the product is never greater than the total of the number of multiplicand bits, and multiplier bits.) Sign bits are not used in this example.

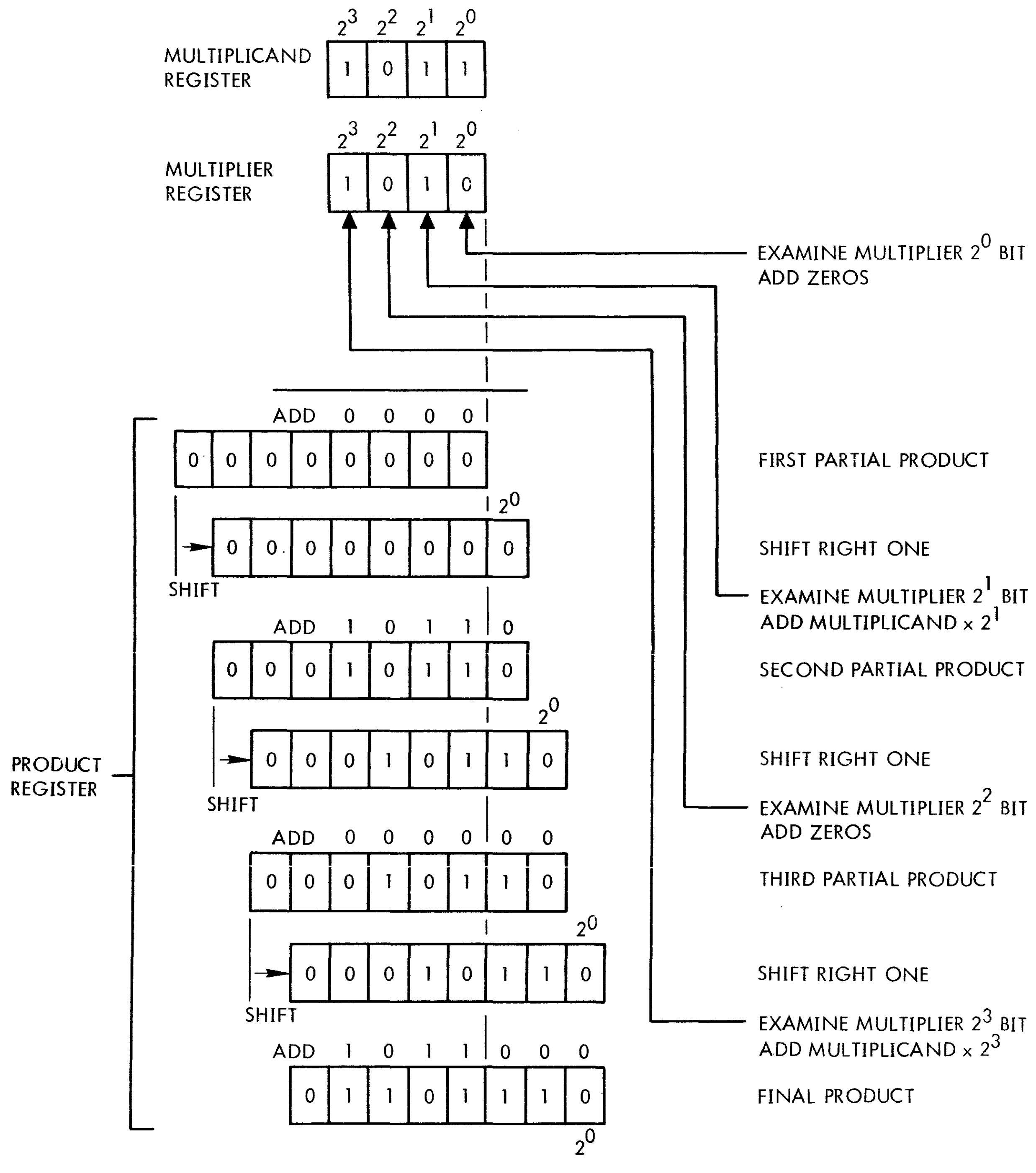
The multiplier is examined one bit at a time, starting with the least significant digit (LSD). If the bit examined is a zero, all zeros are added to the partial product in the product register. If the bit examined is a one, the multiplicand or the appropriate multiple of the multiplicand (two times the multiplicand, four times the multiplicand, and so forth) is added to the partial product. After each examination and addition, the multiplicand is shifted left one bit position. Alternately, the partial product is shifted one bit position to the right. The previous example and the Sigma 7 computer use the latter type of shift. One iteration is required for each bit of the multiplier.

Bit-pair multiplication reduces the number of iterations required by examining two bits of the multiplier at one time and by performing one addition or one subtraction for that bit pair.

There are four permutations of two bits; 00, 01, 10, and 11. Multiplying by the first three types of bit pairs is done by normal shift and add operations: Multiplying by bit pair 00 is done by adding zeros to the partial product; multiplying by bit pairs 01 and 10 is done by adding the multiplicand, or two times the multiplicand, respectively, to the partial product. Multiplying by bit pair 11 is a special case. Multiplication by 3 cannot be represented by a factor of 2; simply shifting the multiplicand and adding is not possible in this case. To multiply by this bit pair, one times the multiplicand is subtracted from the partial product during

Table 3-64. Instruction MW, MI, and MH Condition Codes

Description	CC1	CC2	CC3	CC4	Meaning
Instructions MW, MI, MH	-	-	0	0	Result is zero
	-	-	0	1	Result is negative
	-	-	1	0	Result is positive
	-	0	-	-	Result can be held in odd numbered private memory register only
Instructions MW, MI only	-	1	-	-	Result cannot be held in odd numbered private memory register only



901060A. 3655

Figure 3-154. Single-Bit Multiplication

one iteration, and four times the multiplicand is added to the partial product during the next iteration. Adding four times the multiplicand is accomplished by adding a one to the next bit pair at the time that the bit pair is under examination. The next bit pair becomes a 01 (if it was 00), a 10 (if it was 01), an 11 (if it was 10), or a 00 with a one to be added to the next bit pair (if it was 11). Any one, or carry, to the next higher bit pair is saved until that bit pair comes under examination. Table 3-65 lists bit pair combinations and multiplications.

Table 3-65. Bit-Pair Multiplier Logic

Bit Pair	Carry From Lower Order Bit Pair	Bit Pair Becomes	Carry to Higher Order Bit Pair	Multiplicand Multiplied By
00	0	00	0	0
01	0	01	0	1
10	0	10	0	2
11	0	11	1	-1
00	1	01	0	1
01	1	10	0	2
10	1	11	1	-1
11	1	00	1	0

An example of multiplication by bit pairs is shown in figure 3-155. Sign bits are not used in this example. After each partial product is found, the product register is shifted two places to the right, since the multiplicand has been multiplied by two bit positions.

One addition to the partial sum occurs during each iteration in a multiply operation. The Sigma 7 computer uses carry-save addition to eliminate the need (and time required) for propagating carries as each partial product is formed. Carry propagation is needed only after the last iteration with this technique. Carry-save addition resolves three quantities to be added into the two quantities, sums and carries. An example of carry-save addition is shown in figure 3-156.

The sum for any bit position is a one when one or three of the three bits in a column is a one. The carry from any bit position is a one when two of the three bits are ones. The carry from any bit position is shifted one bit position to the left to provide a normal carry to the next higher order. Figure 3-157 shows how carry-save addition is used in the previous multiplication example. Quantity A (A-register) is the sums portion of the previous partial product. Quantity D (D-register) is the number produced when the bit pair is examined (0, 1, 2, or -1 times the multiplicand), and quantity CS (Carry-Save register) is the carries portion of the previous partial product. Register nomenclature and organization in this example are similar to the Sigma 7 computer multiply process. Sign bits are not used. The three quantities at the beginning of each iteration are resolved to two quantities at the end, which represent the partial sum for that iteration. The final two quantities are added by conventional methods to assimilate the complete product.

3-199 DESCRIPTION OF MULTIPLY WORD AND MULTIPLY IMMEDIATE INSTRUCTIONS. Instructions Multiply Word (MW) and Multiply Immediate (MI) FAMULNH are almost identical in implementation. Figure 3-158 shows the register arrangement during the two instructions. The C-register holds the complete or the inverted multiplicand from core memory. (During instruction MI, the sign bit of the value field of the instruction word, bit position 12, is extended to bit positions 0 through 11 to make up a 32-bit multiplicand.) The multiplicand is inverted if the multiplier is negative to ensure a correct product. The D-register holds the multiple of the multiplicand to be added to the partial product. The contents of the D-register, under control of the B control (BCON) logic, vary with each iteration. BCON logic, in turn, is controlled by the bit pair under examination in bit position 30 and 31 of the B-register and the contents of flip-flop BC31, the carry from the previous bit pair. The A- and CS-registers taken together contain the partial product for each iteration. The A-register holds the sums, and the CS-register holds the carries. The B-register initially holds the multiplier from private memory. With each iteration, the contents of the B-register are shifted two places to the right; the two least significant bits are lost; and a new bit pair comes under examination in B30 and B31. The four most significant bit positions in the B-register, 0 through 3, receive four sum bits of the partial product for each iteration. The sums portion of the partial product is constantly expanding into the higher order bit positions of the B-register which are vacated as the multiplier is shifted out. At the end of the iterations (MITEX: multiply iteration exit), the A- and B-registers contain the 64 bits of the sums portion of the final product, and the CS-register holds the final product carries. The sums and carries are added to produce the 64-bit product. The 32 high order bits are placed in the even numbered private memory register, and the 32 lower order bits are placed in the odd numbered private memory register.

The following paragraphs describe the phases of instructions MW and MI. Figure 3-158 and the sequence chart (table 3-70) supplement the discussion.

During the phase preceding PRE2 NIA, a one is forced onto private memory address line LR31 to obtain the multiplier contained in the odd numbered private memory register. During PRE2, the multiplier is clocked into the A-register. If the multiplier is negative, flip-flop RN (register negative) is set. One's are forced into the CS-register by signal CSX1 in PRE2 in preparation for a possible multiplier sign adjustment in PH1. The remainder of the preparation sequence is the same as the general preparation sequence.

At the time of the PH1 clock, the multiplicand from core memory has been read into the C-register. If instruction MI is being performed, the sign-padded multiplicand is clocked into the D-register by signal DXC/4, and flip-flop CSX is set for use in PH2. Sign manipulation of the operand is started in PH1.

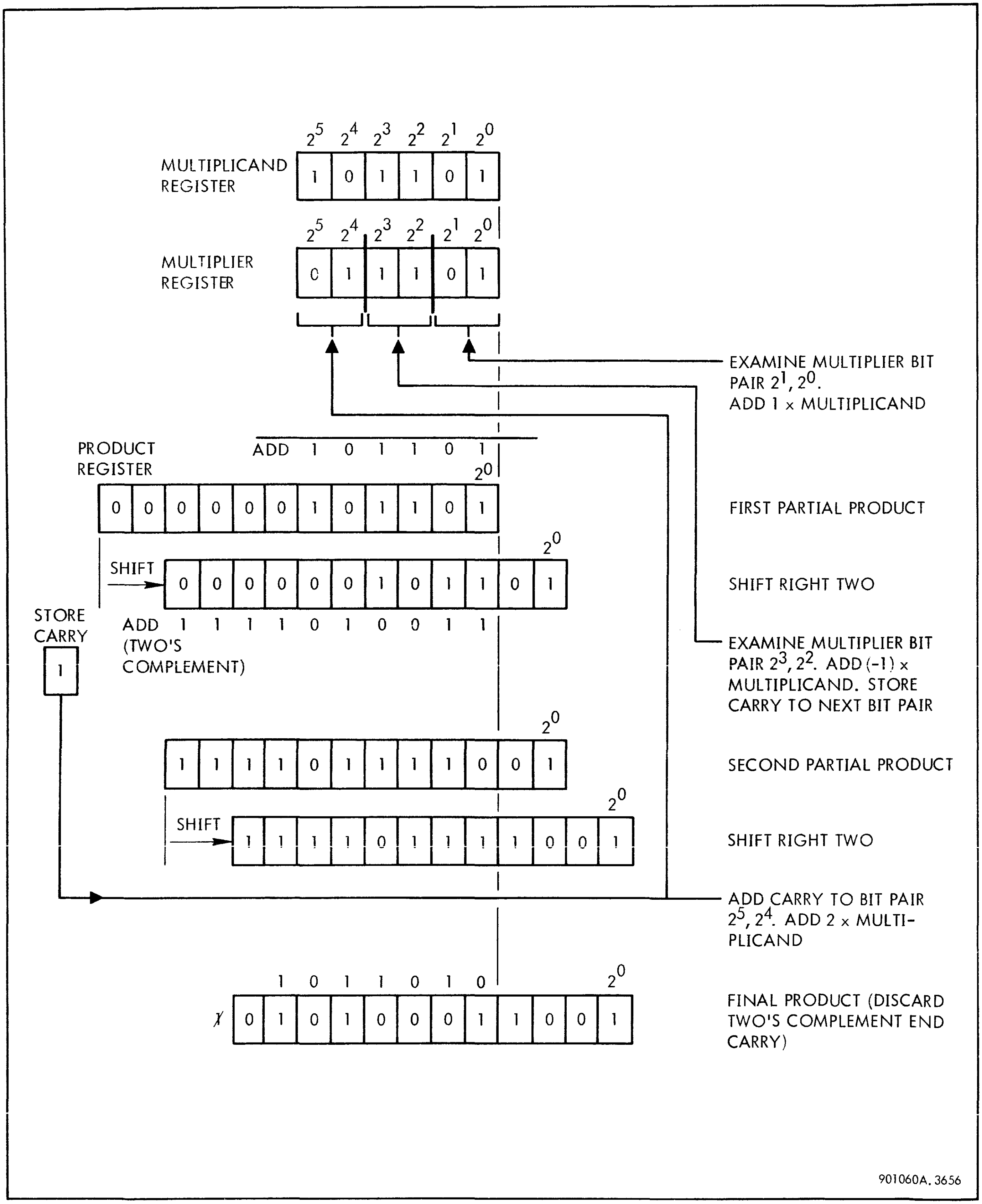


Figure 3-155. Bit-Pair Multiplication

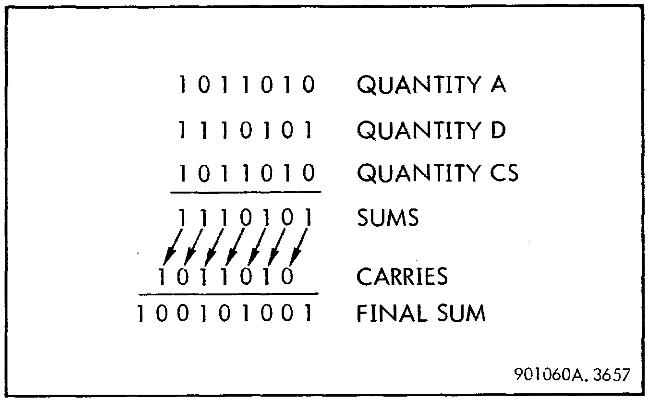


Figure 3-156. Carry-Save Addition

This is necessary to ensure that the multiplier is always positive; multiplying by a negative, as shown in table 3-66, would produce an incorrect result.

Table 3-66. Multiplication Sign Logic

Multiplication	Result	Validity
(+5) x (+4)	(0101) x (0100) = 00010100	Correct
(+5) x (-4)	(0101) x (1100) = 00111100	Incorrect
(-5) x (+4)	(1011) x (0100) = 11101100	Correct
(-5) x (-4)	(1011) x (1100) = 10000100	Incorrect

If the multiplier is positive, the A-register is gated to the sum bus by signal SXA during PH1. At the time of the PH1 clock, the multiplicand (sum bus) is clocked into the B-register by signal BXS. The multiplier remains in its original form, positive or negative.

If the multiplier is negative, it is inverted by an exclusive OR operation with the CS-register (all ones). The inverted (one's complemented) multiplier is gated to the sum bus from the adder by signal SXPR. At the PH1 clock, the inverted multiplier is clocked into the B-register by signal BXS, and the BC31 (carry into B31) flip-flop is set. The contents of the B-register and flip-flop BC31 make up the two's complement of the multiplier. The multiplier sign has changed, therefore the multiplicand sign must also change. This is accomplished in PH9 by combining the D-register (temporary multiplicand storage) with the CS-register (all ones) in an exclusive OR operation. The result is gated to the sum bus by signal SXPR and is clocked into the C-register at the time of the PH9 clock by signal CXS. Signal CCWCM denotes that C contains one's complement and was previously enabled in PRE2. The contents of the C-register and signal CCWCM constitute the two's complement of the multiplicand. Table 3-67 shows the sign manipulation for all possible multiply cases.

Other operations occurring at the PH1 clock include clearing the A-register, clearing the P-register, resetting flip-flop CC2, and enabling clock T4L for PH2.

At the PH2 clock, the multiplicand in the C-register is clocked into the D-register if instruction MW is being performed. If instruction MI is being performed, the sign-padded multiplicand in the D-register is gated to the sum bus by signal SXD, is transferred to the C-register by signal CXS, and is transferred to the D-register by signal DXC/6. The multiplicand is now in the C- and D-registers. If the multiplier is negative (flip-flop RN is set), the D contents is inverted and clocked into the C-register in PH9 for sign manipulation, as discussed in previous paragraphs.

Coincident with sign manipulation in PH2 through PH9, BCON logic is set up and performed. Signal BCON is high during the time that the multiplier bit pairs are examined and is an enable signal to the examination logic. BCON logic in PH2 through PH9 is described below.

$$\begin{aligned}\text{BCON} = \ & \text{FAMDSF/M}\ \overset{\text{floating multiply}}{\text{NFLMC}}\ \overset{\text{multiply last clock}}{\text{NMULC}} \\ & \text{N(FAMUL MITEX) (FAMUL PH2} \\ & + \text{FAMDSF/M PH9} + \text{MIT} + \ldots)\end{aligned}$$

The absolute value of the multiplier is contained in the B-register in PH2. The first bit pair of the multiplier, B30-B31 (multiplier 2^1, 2^0), and flip-flop BC31 are examined at this time. Flip-flop BC31 saves the carry to the next bit pair (multiply by -1 case). If the multiplier was originally negative, BC31 is set at this time. B30, B31, and BC31 control the same functions as shown in table 3-65.

Table 3-67. Multiply Sign Manipulation

Original Multiplicand Sign	Original Multiplier Sign	RN	Sign of Multiplicand in C-register (PH9)	CCWCM f(RN)	Sign of Multiplier in B-register (PH1)	BC31 f(RN)	Sign of Product (PH11)
+	+	0	+	0	+	0	+
+	-	1	-	1	+	1	-
-	+	0	-	0	+	0	-
-	-	1	+	1	+	1	+

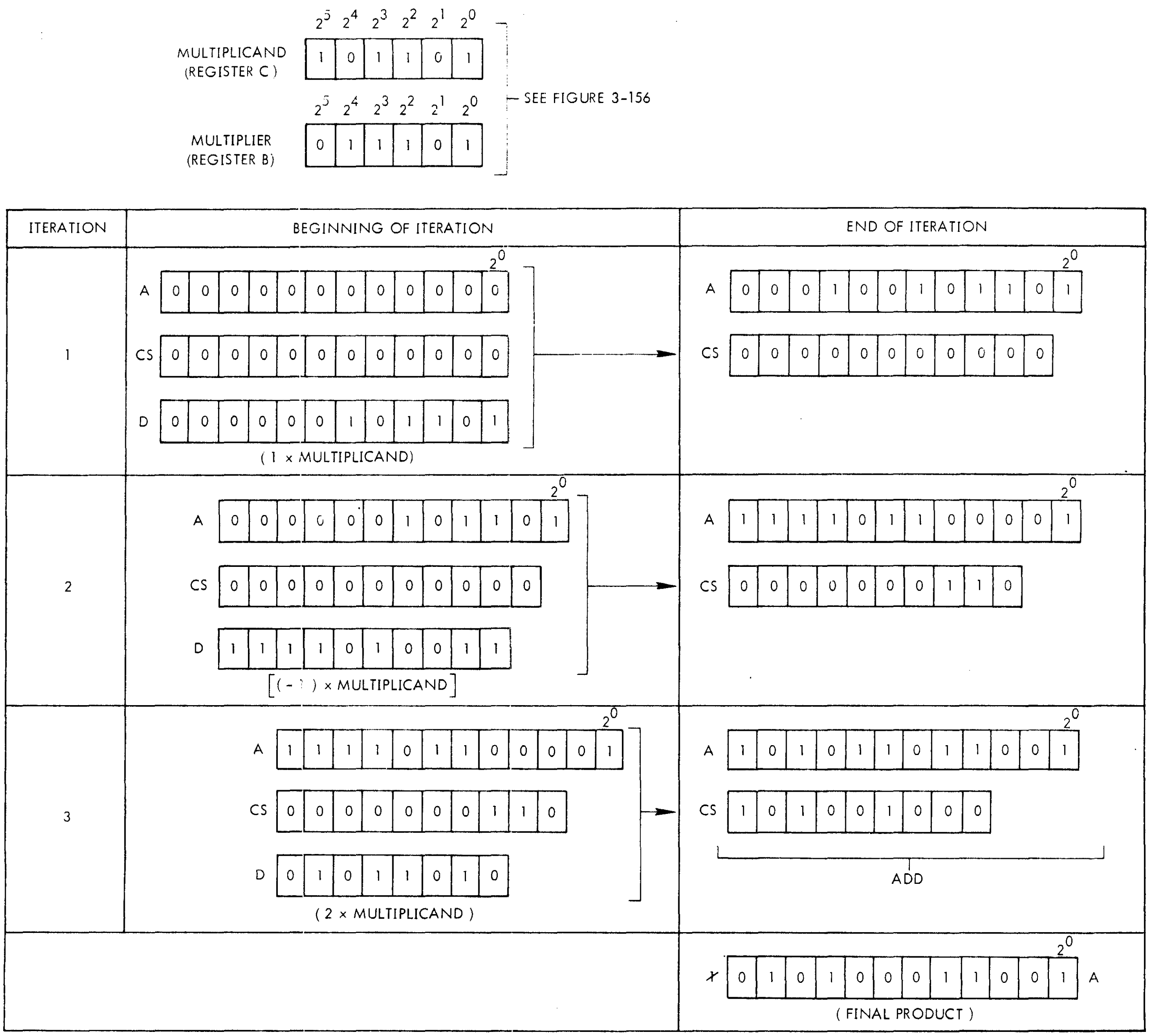


Figure 3-157. Carry-Save Addition in Multiplication

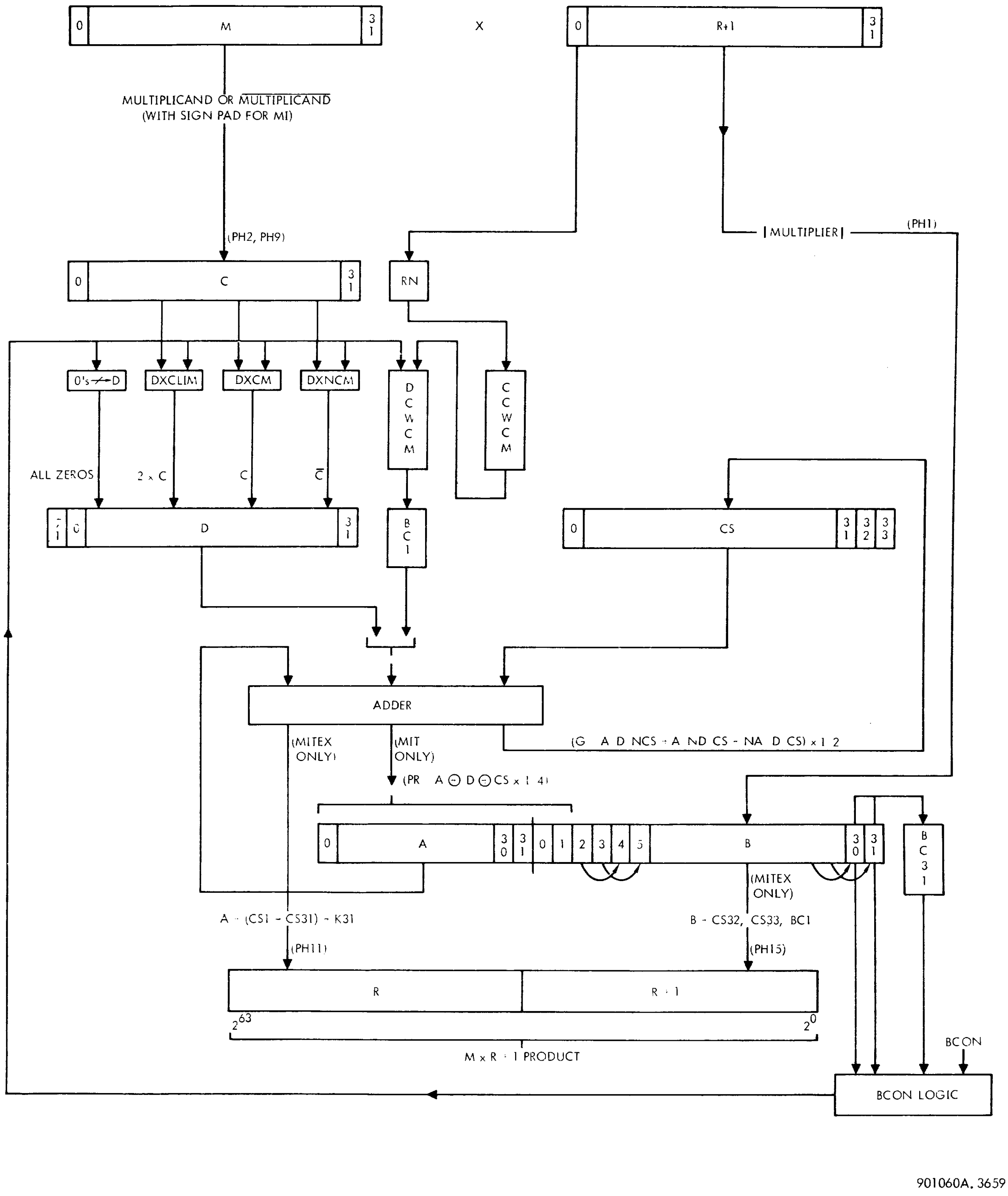


Figure 3-158. Register Organization and Flow Chart for Multiply Word and Multiply Immediate Instructions

The multiplicand is multiplied by either 0, 1, 2, or (-1) and the resulting quantity is transferred to the D-register. Multiplier control flip-flops DXCM, DXCLIM, and DXNCM control the multiplication and transfer and are, in turn, controlled by B30, B31, and BC31. Setup of the multiplier control flip-flops occurs one clock before the transfer into the D-register; bit pair 2^1, 2^0, therefore, controls the transfer to the D-register in PH9. Table 3-68 shows the multiplier logic. Note that BC31 may be set for the next bit pair examination.

$$S/DXCM = BCON\ (NB30\ \overset{\text{(bit pair 01, no carry)}}{B31}\ NBC31 + NB30\ \overset{\text{(bit pair 00, carry)}}{NB31}\ BC31)$$

$$S/DXNCM = BCON\ (B30\ \overset{\text{(bit pair 11, no carry)}}{B31}\ NBC31 + B30\ \overset{\text{(bit pair 10, carry)}}{NB31}\ BC31)$$

$$S/DXCLIM = BCON\ (B30\ \overset{\text{(bit pair 10, no carry)}}{NB31}\ NBC31 + NB30\ \overset{\text{(bit pair 01, carry)}}{B31}\ BC31)$$

$$S/BC31 = BCON\ (B30\ B31\ \overset{\text{(bit pair 11, no carry)}}{NBC31} + B30\ \overset{\text{(bit pair 10, or 11, carry)}}{BC31} + \overset{\text{(PH1)}}{FAMUL\ PH1}\ RN)$$

At the same time that the multiplier control flip-flops are set up, the B-register shifts two bit positions to the right, bringing the second multiplier bit pair (2^3, 2^2) into B30 and B31. Zeros are clocked into B0 and B1. At the end of PH2, the multiplier control flip-flops have been set up by the first bit pair, and the next bit pair is in position for examination.

At the PH9 clock, the following operations occur simultaneously:

a. The contents of the C-register are clocked into the D-register as a function of the multiplier control flip-flop set by multiplier bit pair 2^1, 2^0 in PH2. If no multiplier control flip-flops are set, zeros are clocked into the D-register. If multiplier control flip-flop DXCM is set, the contents of C0 through C31 are clocked into D0 through D31, respectively. If the multiplicand of the C-register is the one's complement of the original multiplicand (flip-flop CCWCM set), flip-flop DCWCM (D contains one's complement) is set. If multiplier control flip-flop DXCLIM is set, the contents of C0 through C31 are clocked into D71 and D0 through D30, respectively. A one is forced into D31 to maintain the one's complement condition, and flip-flop DCWCM is set. The contents of C0 through C31 are inverted and clocked into D0 through D31, respectively, if multiplier control flip-flop DXNCM is set. If flip-flop CCWCM is set, the multiplicand is in its original form, and flip-flop DCWCM is not set. The one's complement is in the D-register and flip-flop DCWCM is set, if the C-register does not hold the one's complement. The various transfers are shown in table 3-69.

b. The multiplier control flip-flops are set up again by bit pair 2^3, 2^2 of the multiplier.

c. The B-register shifts two bit positions to the right by signal BXBR2, bringing bit pair 2^5, 2^4 of the multiplier into B30, B31. Zeros are clocked into B0 and B1.

$$BXBR2 = FAMDSF/M\ PH9 + \ldots$$

d. Flip-flop MIT (multiply iterations) is set. Signal MIT controls the number of multiply iterations.

e. Clock T10 is enabled for PH10.

$$S/T1L = FAMSDF/M\ PH9\ \overset{\text{not single clocking}}{NKSC}$$
$$\overset{\text{floating multiply}}{N(S/FLMC/1}\ P18)$$

$$R/T1L = (MP19\ S/MULC/1) + \ldots$$

The D-register now contains the first partial sum. The A-register contains all ones (cleared in PH1), and the CS-register all zeros.

There are 16 clocks in PH10 for the 16 multiply iterations. At each clock, the following events occur:

a. The D-register is combined with the A- and CS-registers in the adder to produce the partial product for that iteration. Figure 3-159 shows the addition process and signal routing. The sums portion of the product is clocked into A0 through A31 and the first four bit positions of the B-register, B0 through B3. Depending upon the number of iterations previously performed, B4 through B31 contain 0 to 27 lower-order bits of the partial product; however, these do not enter into the addition. Because of their position in the product, these bits remain constant as higher-order bits of the multiplier are examined, and the partial product is shifted. The carries portion of the partial product is clocked into the CS-register, bit positions 0 through 33. Generally, the sum for any bit position in the sums register (A-register) is set by a propagate signal from the adder, defined by $PR_n = A_n\ NCS_n\ ND_n + NA_n\ NCS_n\ D_n$

Table 3-68. Bit-Pair Multiplier Logic

B30	B31	BC31	S/BC31	S/DXCM	S/DXNCM	S/DXCLIM	Transfer
0	0	0	0	0	0	0	0 into D
0	1	0	0	1	0	0	C into D
1	0	0	0	0	0	1	2 x C into D
1	1	0	1	0	1	0	NC into D
0	0	1	0	1	0	0	C into D
0	1	1	0	0	0	1	2 x C into D
1	0	1	1	0	1	0	NC into D
1	1	1	1	0	0	0	0 into D

Table 3-69. Multiplier Control Flip-Flop Logic

CCWCM	Multiplier Control Flip-Flop Set	Transfer	DCWCM
0	None	Zeros into D71, D0-D31	0
0	DXCM	C0-C31 into D0-D31	0
0	DXCLIM	C0 into D71, C1-C31 into D0-D30	0
0	DXNCM	NC0-NC31 into D0-D31	1
1	None	Zeros into D71, D0-D31	0
1	DXCM	C0-C31 into D0-D31	1
1	DXCLIM	C0 into D71, C1-C31 into D0-D30, one into D31	1
1	DXNCM	NC0-NC31 into D0-D31	0

+ $NA_n\ CS_n\ ND_n + A_n\ CS_n\ D_n$. PR_n is true when one or three bits of the three to be added are ones. The propagate signals are shifted two bit positions to the right in prepreparation for the next addition. The carry to any bit position is generally set by generate signals from the adder, defined by $G_n = A_n\ D_n + A_n\ CS_n\ ND_n + NA_n\ CS_n\ D_n$. G_n is true whenever two of the three bits added are ones. A true G_n signal for any bit position provides a carry signal to the next higher order; the generate signals are shifted right only one bit position for this reason (two right shifts in preparation for the next addition minus one left shift for a carry to the next higher order). A new set of sums and carries is generated with each iteration. The only bits changing in the addition are the 36 higher order sum bits (A0-A31, B0-B3) and the 33 carry bits (CS1-CS33).

b. The extreme higher order and lower order bits of the partial product are formed by special logic. Figure 3-160 shows the possible configurations of the high order bits to be added in an iteration (CS0 is always a 0), together with the resulting sums and carries. A0 of the sum is used only for sign-padding the partial product and is a one when there is a one in A0 or D71 or both. A1 would normally be the sign bit in a straight addition, but it is a sum bit in carry-save addition, since CS1 may hold a carry after the addition. Flip-flops A2 through A31 are set two orders to the left by the propagate signal. Flip-flop B3 is set by signal B1S, equivalent to a propagate signal from bit position 33 and true when there is a total of one or three ones in B1, BC1, and CS33. Flip-flop BC1 is set by signal DCWCM and contains a one if the one's complement of the multiplicand was put in the D-register at the previous iteration. By adding a one to B3 of the partial product one clock after the addition of the A-, D-, and CS-registers, the D-register is paid the 1 needed to make a 2's complement. The partial product has been shifted so that the order of B3 is the same that would have received the one, if it had been initially added to the D-register. Figure 3-161 illustrates the timing for this operation. Flip-flop B2 is set by the propagate signal, PR32, for bit position 32; this is the same as the normal propagate signal except that only the CS- and B-registers are combined. There is no CS32 or CS33. B2 is not set, however, when there are both a PR32 signal and a generate signal from the thirty-third bit position. G33 amounts to a carry from bit position 33. The carry from bit position 33, if not resolved by setting flip-flop B2, is stored in CS33. Flip-flops CS1 through CS32 are set by G0 through G31, respectively.

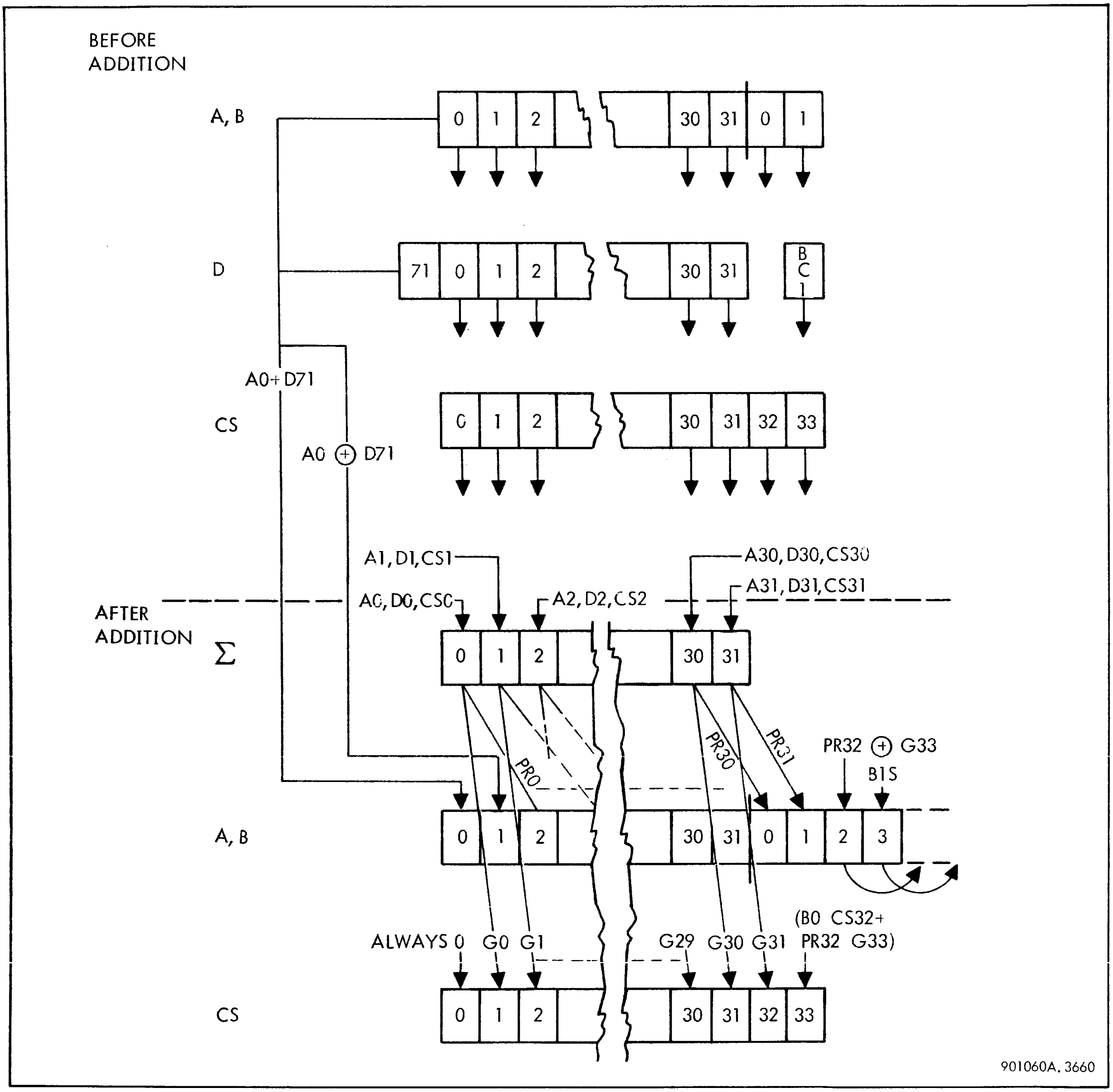


Figure 3-159. Multiply Iteration Signal Routing

PR0 = A0 NCS0 ND0 + NA0 NCS0 D0 + NA0 CS0 ND0 + A0 CS0 D0

. .
. .
. .

PR31 = A31 NCS31 ND31 + NA31 NCS31 D31 + NA31 CS31 ND31 + A31 CS31 D31

PR32 = B0 NCS32 + NB0 CS32

G0 = A0 D0 + A0 CS30 ND0 + NA0 CS0 D0

. .
. .
. .

G31 = A31 D31 + A31 CS31 ND31 + NA31 CS31 D31

G33 = B1 BC1 + B1 NBC1 CS33 + NB1 BC1 CS33

S/BC1 = DCWCM CSXGR1 = DCWCM MIT

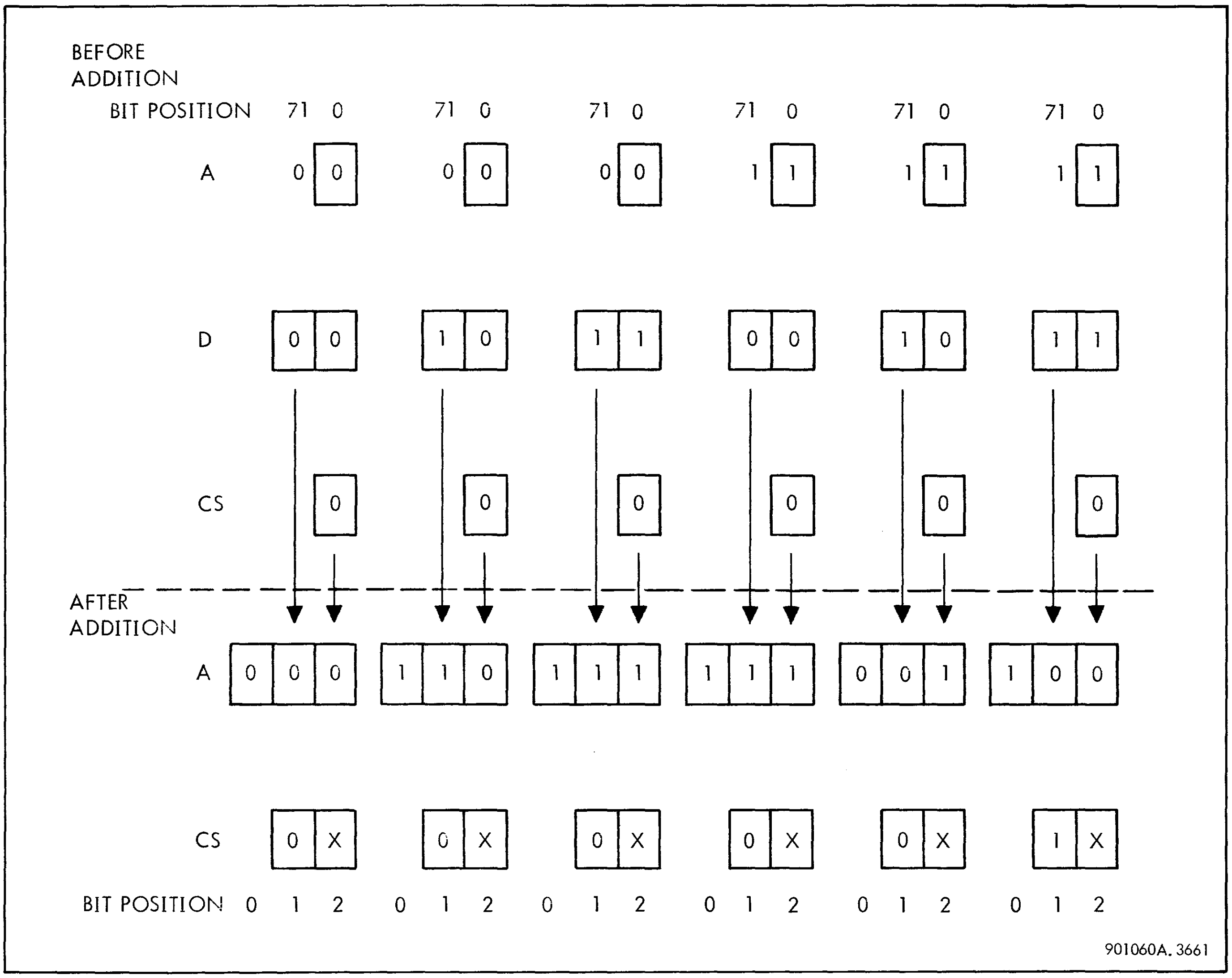

BEFORE ADDITION
BIT POSITION 71 0 71 0 71 0 71 0 71 0 71 0
A 0 0 0 0 0 0 1 1 1 1 1 1
D 0 0 1 0 1 1 0 0 1 0 1 1
CS 0 0 0 0 0 0
AFTER ADDITION
A 0 0 0 1 1 0 1 1 1 1 1 1 0 0 1 1 0 0
CS 0 X 0 X 0 X 0 X 0 X 1 X
BIT POSITION 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
901060A. 3661

Figure 3-160. High-Order Bit Logic

S/A0 = (A0 + D71) MIT + . . .

S/A1 = (A0 + D71) N(A0 D71) MIT + . . .

S/A2 = PR0 MIT + . . .

. .
. .
. .

S/A31 = PR29 MIT + . . .

S/B0 = PR30 MIT + . . .

S/B1 = PR31 MIT + . . .

S/B2 = B0S MIT

S/B3 = B1S MIT

B0S = (PR32 NG33 + NPR32 G33) MIT

B1S = B1 NBC1 NCS33 + NB1 BC1 NCS33 + NB1 NBC1 CS33 + B1 BC1 CS33

S/CS1 = G0 MIT + . . .

. .
. .
. .

S/CS32 = G31 MIT + . . .

S/CS33 = (B0 CS32 + PR32 G33) MIT + . . .

c. The contents of the C-register are clocked into the D-register as a function of the multiplier control flip-flops that were set during the previous clock.

d. The next bit-pair of the multiplier is examined, and the multiplier control flip-flops are set.

e. The B-register shifts two bit positions to the right by signal BXBR2, bringing a new bit-pair into B30 and B31.

f. The contents of B2 and B3 are clocked into bit positions 2 and 3, respectively, of the register. If E2 and E3

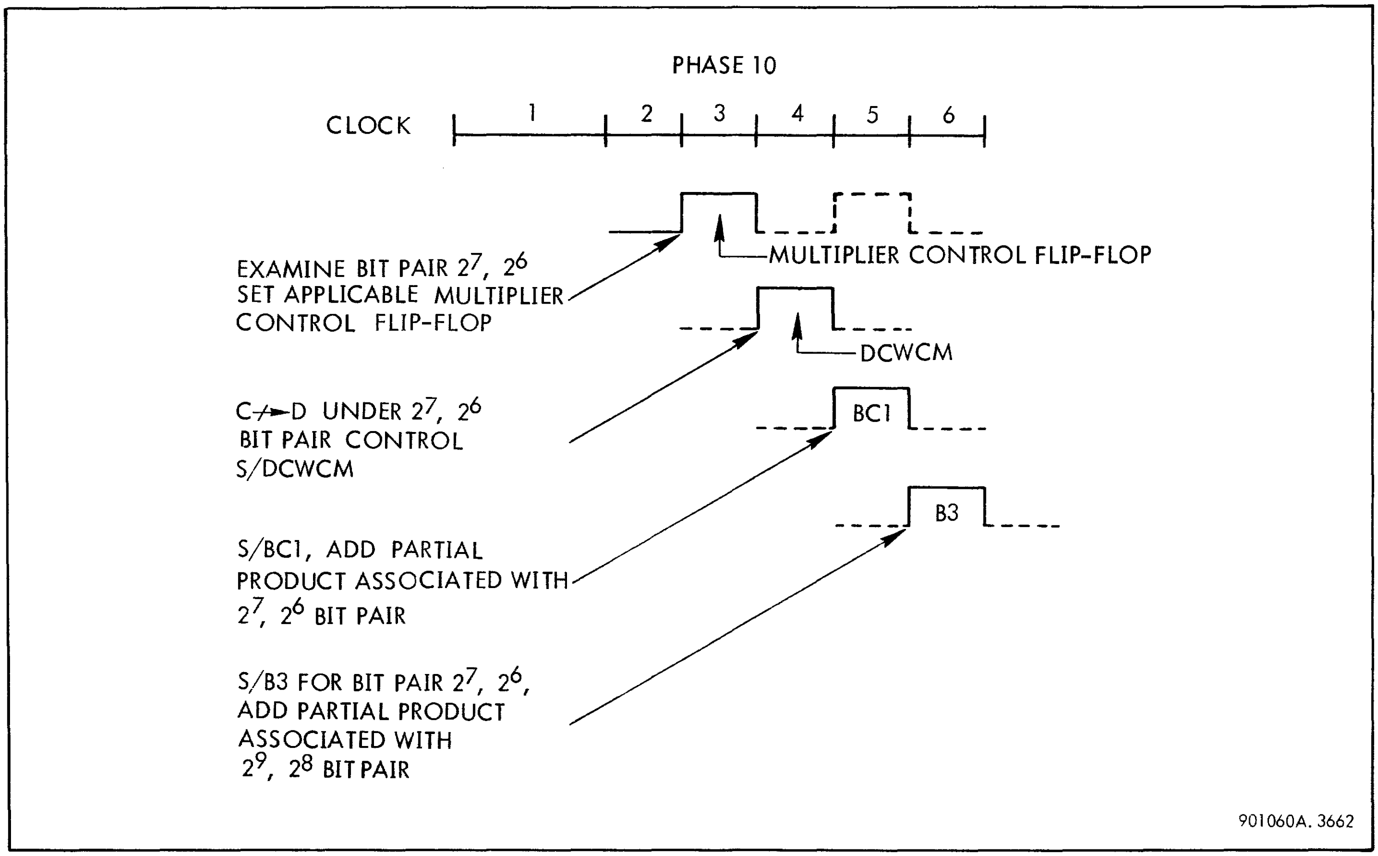

PHASE 10
CLOCK
1
2
3
4
5
6
MULTIPLIER CONTROL FLIP-FLOP
DCWCM
BC1
B3
EXAMINE BIT PAIR 2⁷, 2⁶ SET APPLICABLE MULTIPLIER CONTROL FLIP-FLOP
C→D UNDER 2⁷, 2⁶ BIT PAIR CONTROL S/DCWCM
S/BC1, ADD PARTIAL PRODUCT ASSOCIATED WITH 2⁷, 2⁶ BIT PAIR
S/B3 FOR BIT PAIR 2⁷, 2⁶, ADD PARTIAL PRODUCT ASSOCIATED WITH 2⁹, 2⁸ BIT PAIR
901060A. 3662

Figure 3-161. BC1 Timing

contain 0's at the end of 16 iterations, the product bits 2^{27} through 2^0 are 0's, since all of these bits have been shifted in and out of E2 and E3 during the iterative process.

S/E2 = B2 (FAMUL MIT) + . . .

S/E3 = B3 (FAMUL MIT) + . . .

g. The number of the iteration is counted. The iteration counter is made up of flip-flops P15 through P18 (cleared in PH1 by signal PX) and flip-flop MP19. These five flip-flops form a five-bit binary up-counter that counts iterations during instructions MW, MI, MH, and floating point instructions. Figure 3-162 shows the counting configurations and the signals enabled by the counter for instructions MW and MI. The counter is used only through the 14th clock for the two instructions. At the first clock of PH10, signal PCTP5 (plus count control for P15 through P19) goes true as flip-flop MP19 is set. Flip-flop MP19 takes the place of P19 in the iteration counting and is set at every even clock except the last (clock 16). At the 14th clock, flip-flop MULC is set. A true MULC signal sets flip-flop MITEX at clock 15; MITEX resets flip-flop MIT at clock 16. At clock 14, also, flip-flop T1L is reset to initiate the end of T1L (T1L timing lasts through PH10) and memory request flip-flop MRQ/M is set, requesting the next instruction. At clock 15, the P-register is cleared (signal PX), and the address of the next instruction is transferred from Q15 through Q31 to P15 through P31 by signal PXQ. Clock 15 is the last transfer of the C-register contents to the D-register under control of the multiplier control flip-flops. Other functions at clock 15 include clearing the D-register (signal DX/1\ and resetting flip-flop MIT to signify the end of the multiply iterations. The last carry-save addition of the A-, D-, and CS-registers occurs during clock 16; the product of the multiplication is contained in A0 through A31 and B0 through B31 (sums) and CS1 through CS33 and BC1 (carries). If the R-field of the instruction word is even, clock T10L is enabled. If the R-field is odd, clock T6L is enabled. PH11 is set.

During PH11, the final product of the multiplication is assimilated from the results of the last iteration in PH10 (see figure 3-163). The most significant word of the product, bits 2^{63} through 2^{32}, is obtained by adding the contents of the A-register to the contents of the CS-register. Any carry from bits 2^{31} through 2^0 of the product, designated K31, is also added to the sum. The sum, S0 through S31, is clocked into the A-register by signal AXS and, if R is even, it is clocked into the even private memory register by signal RW.

The least significant word of the product, bits 2^{31} through 2^0, is obtained by adding the contents of B0 and B1 to the contents of flip-flops CS32, CS33, and BC1. Flip-flop BC1 may contain a one if the one's complement was put into the

CLOCK (PH10)	P15	P16	P17	P18	MP 19	
1	0	0	0	0	0	
2					1	PCTP5
3				1	0	
4				1	1	
5			1	0	0	
6			1	0	1	
7			1	1	0	
8			1	1	1	
9		1	0	0	0	
10		1	0	0	1	
11		1	0	1	0	
12		1	0	1	1	
13		1	1	0	0	
14		1	1	0	1	S/MULC, R/T1L, S/MRQ/M
15	NOT APPLICABLE					S/MITEX, MRQ, Q→P BY PX
16	NOT APPLICABLE					0→D, R/MIT, S/T10L IF R EVEN

901060A.3663

Figure 3-162. Iteration Counting for Multiply Word and Multiply Immediate Instructions

D-register at the fifteenth iteration of PH10. CS32 and CS33 contain the remaining carries to be added. The sum of B0, B1, and CS32, CS33, and BC1 is clocked back into bit positions B0 and B1 of the B-register by signal BXB. The remainder of the bits in the B-register remain unchanged.

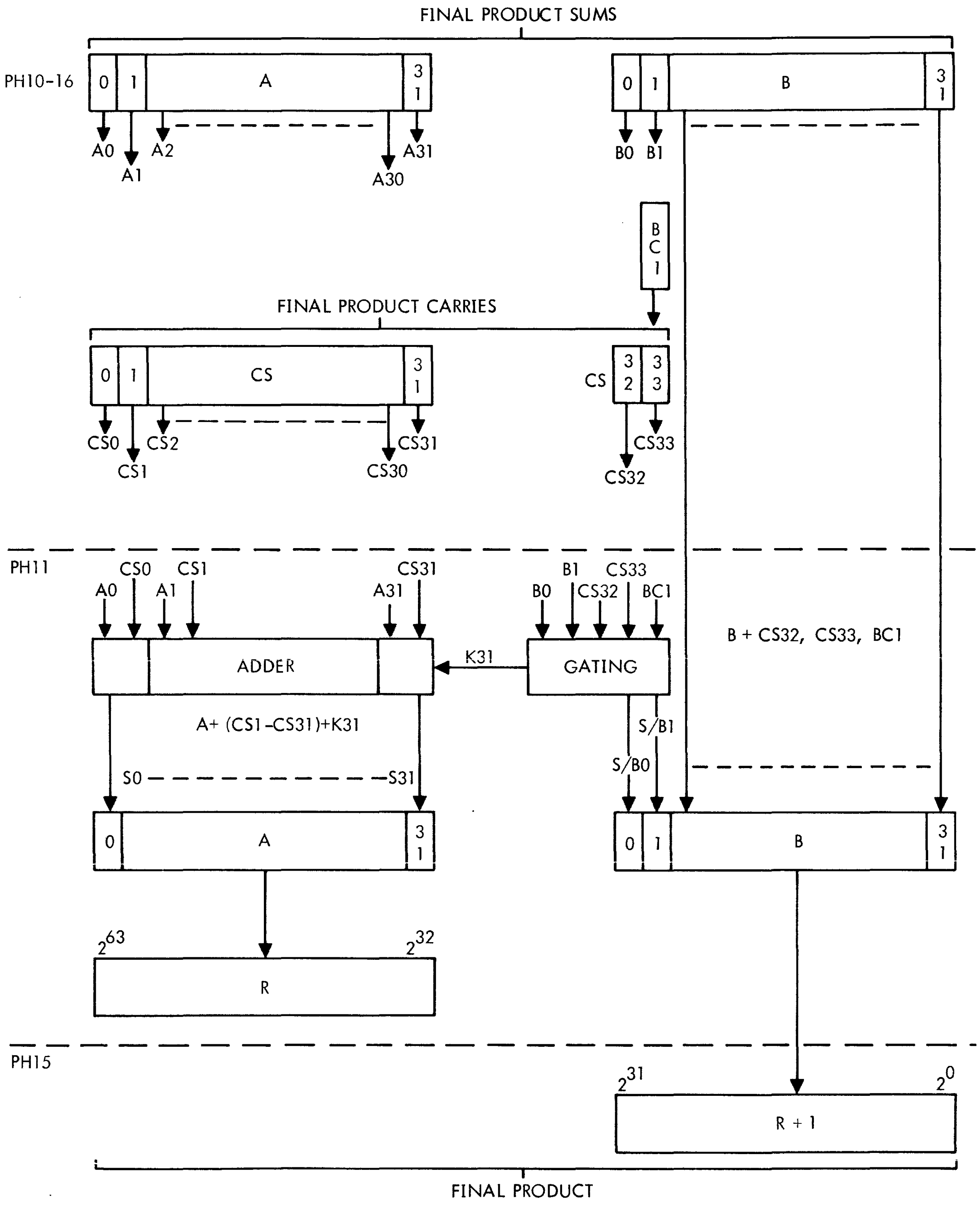
If either bit positions 2 or 3 of the E-register contain a one at the beginning of PH11, the product 2^{27} through 2^0 is nonzero. During PH11, any ones in the product bits 2^{31} through 2^{28} cause one or more ones to be clocked into E0 through E3. If E0 through E3 contain all zeros at PH15, product bits 2^{31} through 2^0 are all zeros.

At the PH11 clock, a one is forced on private memory address line LR31 to select the odd numbered private memory register for storage of the least significant word of the product, bits 2^{31} through 2^0, in PH15. A one is forced into CS31 to enable K00 in PH15 for magnitude testing. Flip-flop DRQ is set which inhibits transmission of another clock until the memory data release signal is received. PH15 is enabled by setting flip-flop PH15. Flip-flop T8L is set for PH15 clock.

During PH15, the least significant word of the product, bits 2^{31} through 2^0, is gated to the sum bus by signal SXB. At the PH15 clock, the least significant word of the product is read into the odd numbered private memory register by signal RW at the PH15 clock. If the E-register contents are not equal to zero, the least significant word of the product is not zero. A one is forced into A31 in this case. Flip-flop CC3 is set if there is a one in A1 through A31 and if A0 is a zero which indicates a positive product. Flip-flop CC4 is set if A0 is a one, indicating a negative product. Flip-flop CC2 is set if the 2^{31} (B0) bit of the product is a zero and if bits 2^{63} through 2^{32} are nonzero, or if the 2^{31} bit is a one and bits 2^{63} through 2^{32} are not all ones. (K00 goes true if bits 2^{63} through 2^{32} are all ones and if CS31 contains a one.) A one was forced into CS31 in PH11 for this test. Signal ENDE is generated.

A sequence chart for Multiply Immediate and Multiply Word is given in table 3-70.

3-200 DESCRIPTION OF MULTIPLY HALFWORD INSTRUCTION. Instruction Multiply Halfword (MH), FAMULH, is very similar to instructions MW and MI. However, there are eight iterations rather than 16 in MH, and there are 32 product bits. Figure 3-164 shows the register arrangement during instruction MH. The C-register holds the complete or inverted halfword multiplicand from core memory in bit positions 0 through 15. The multiplicand is inverted if the multiplier is negative. Bit positions 16 through 31 contain all zeros if the multiplicand is not inverted or all ones if the multiplicand is inverted.



901060A. 3664

Figure 3-163. Multiply Word and Multiply Immediate Final Product Assimilation

Table 3-70. Multiply Word, Multiply Immediate, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence except that a one is forced onto LR31 address line and PRE2 NIA is as follows	S/LR31/2 = (PRE1 NIA + PRE3 IX + PRE4) FAFRR/1 + (PRE1 NIA + PRE3 IA) FAFRR	Selects odd numbered private memory register
PRE2 NIA	RR0-RR31 —/→ A0-A31	AXRR = FAMDSF PRE2 NIA + . . .	Multiplier —/→ A-register
	Set flip-flop RN if multiplier negative	S/RN = RR0 RNXRR0 + . . . RNXRR0 = FAMULNH + . . .	Stores multiplier sign
	Force ones into CS0-CS31	CSX1 = PRE2 NIA FAMUL + . . .	To produce one's complement of multiplier in PH1 if RN
PH1 T4RL	MB0-MB31 —/→ C0-C31	Preparation control	Multiplicand —/→ C-register
	C12 —/→ D0-D11	C0C16 = C0C16 C12 + DXC/4 C12 + ...	Sign pad
	C12-C31 —/→ D12-D31 } FAMULI only	DXC/4 = FAMULI PH1 + . . .	Value field
	Set flip-flops CXS	S/CXS = FAMULI PH1 + . . .	For PH2 use
	A0-A31 ——→ S0-S31 if A is positive	SXA = FAMUL PH1 NRN + . . .	Absolute value of multiplier transferred to the B-register and BC31
	A0-A31 ⊕ CS0-CS31 → S0-S31 } If A is negative	SXPR = FAMUL PH1 RN + . . .	
	Set flip-flop BC31	S/BC31 = FAMUL PH1 RN + . . .	Carry into B31
	S0-S31 —/→ B0-B31	BXS = FAMDSF PH1 + . . .	Multiplier or multiplicand
	Clear A-register	AX/1 = FAMDSF PH1 + . . .	
	Clear P-register	PX = FAMUL PH1 + . . .	
	Reset flip-flop CC2	R/CC2 = FAMDSF NFAMULH PH1 + . . .	
	Enable clock T4L if FAMULW	S/T4L = FAMUL OU3 PH1 + . . .	
PH2 T4L or T6L	Timing: T4L if FAMULW, T6L if FAMULI		
	D0-D31 ——→ S0-S31 } if FAMULI	SXPR = FAMULI PH2 + ...	PR = D since A ⊕ CS = 0. 32-bit multiplicand transferred to D-register. Final multiplicand sign adjustment occurs in PH9
	S0-S31 ——→ C0-C31	CXS = Previously set in PH1	
	C0-C31 —/→ D0-D31	DXC/6 = FAMUL PH2 + . . .	
	Force ones into CS0-CS31 for PH9 if multiplier is negative	CSX1 = FAMUL PH2 RN + . . .	For PH9 multiplicand sign adjustment
	Set flip-flop CXS for PH9 if multiplier is negative	S/CXS = FAMUL PH2 RN + . . .	For PH9 multiplicand sign adjustment

Mnemonic: MW (37, B7), MI (23, 17)

(Continued)

Table 3-70. Multiply Word, Multiply Immediate, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments

PH2
T4L
or
T6L
(Cont.)
	B0-B29 —/→ B2-B31
	Enable BCON
		BXBR2 = FAMUL PH2 + . . .
		BCON = BXBR2 FAMDSF/M + . . .
			Examine multiplier bit-pair 2^1, 2^0. Set multiplier control flip-flops to this bit-pair. Shift multiplier to bring next bit-pair into B30, B31

	Set up flip-flops DXCM, DXNCM, DXCLIM, BC31 as functions of B30, B31, and BC31		
	Set flip-flop PH9	BRPH9 = FAMUL PH2 + . . .	

PH9
T6L
	D0-D31 ⊕ CS0-CS31 ——→ S0-S31
	S0-S31 ——→ C0-C31
		SXPR = FAMDSF/M PH9 + . . .
		CXS = Previously set in PH2
			Final multiplicand sign adjustment in case of negative multiplier. C already contains multiplicand if multiplier is positive

	C0-C31 —/→ D0-D31
	or
	NC0-NC31 —/→ D0-D31
	or
	C0-C31 —/→ D71-D30
	or
	Zeros transferred to D71-D30 if none of the above
		DXCM = Previously set in PH2
		DXNCM = Previously set in PH2
		DXCLIM = Previously set in PH2
			First transfer of multiplicand under control of bit pair 2^1, 2^0

	Force a one into flip-flop DCWCM if one's complement is transferred to D	S/DCWCM = CCWCM (DXCM + DXCLIM) + NCCWCM DXNCM CCWCM = FAMUL RN	Records one's complement condition so that a one can be added later to make up two's complement Implies one's complement in C-register

	B0-B29 —/→ B2-B31
	Sustain BCON
		BXBR2 = FAMDSF/M PH9 + . . .
		BCON = FAMDSF/M BXBR2 + . . .
			Examine multiplier bit-pair 2^3, 2^2. Set multiplier control flip-flops to this bit-pair. Shift multiplier to bring next bit-pair into B30, B31

	Set up flip-flops DXCM, DXNCM, DXCLIM, B31 as functions of B30, B31, BC31		
	Set flip-flop MIT	S/MIT = FAMDSF/M PH9 NCLEAR + . . .	
	Enable clock T1L if not a single clocking	S/T1L = FAMDSF/M PH9 NKSC N(S/FLMC/1 P18) + . . .	
PH10 T1L	Sixteen clocks. During the first 13 clocks, the following events occur:		
	A ⊕ D ⊕ CS ——→ PR0-PR31 PR0-PR31 —/→ A2-A31, B0, B1	AXPRR2 = MIT + . . .	Sums of the partial products

Mnemonic: MW (37, B7),
MI (23, 17)

(Continued)

Table 3-70. Multiply Word, Multiply Immediate, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T1L (Cont.)	A D + A CS + D CS ——→ G0-G31		
	G0-G31 —/→ CS1-CS32	CSXGR1 = MIT + . . .	Carries of the partial products
	Contents of C are transferred to D under control of the multiplier control flip-flops DXCM, DXNCM, DXCLIM, set during previous clock		
	DCWCM set as before if one's complement is transferred to D		
	B0-B29 —/→ B2-B31	BXBR2 = MIT + . . .	
	PR30, PR31 —/→ B0, B1	BXBR2 = MIT + . . .	Least significant bits of partial product
	Sustain BCON	BCON = BXBR2 N(MULC + MITEX) + . . .	
	Set up flip-flops DXCM, DXNCM, DXCLIM, B31 as functions of B30, B31, BC31		
	Merge contents of B2, B3 into contents of E2, E3	Path enabled by FAMUL MIT	For detection of a 0 product in product bits 2^{27} - 2^{0}
	Count iterations using P15-P18 and MP19	S/MP19 = MIT N(MULC + MITEX) + . . . NMP19 BCON	High on all even clocks but the last
		R/MP19 = .	
		PCTP5 = MP19 + . . .	Count up P15-P18
	On the 14th clock:		
	Other events listed above		
	Set flip-flop MULC	S/MULC = MP19 FAMULNH P16 P17 + . . .	
	Reset clock T1L	R/T1L = MIT NMP19 + . . .	T1L timing lasts through PH10
	Set flip-flop MRQ/M	S/MRQ/M = (S/MULC/1) MP19	Request the next instruction
	On the 15th clock:		
	Other events listed above		
	Set flip-flop MITEX	S/MITEX = FAMUL MULC + . . .	To signal end of multiply iterations

	Enable MRQ
	Clear the P-register
	Q15-Q31 —/→ P15-P31
		MRQ = MRQ/M + . . .
		PX = PXQ
		PXQ = MULC + . . .
			No longer needed for iteration counting. Transfer next instruction address

	On the 16th clock:		
	Other events listed above		

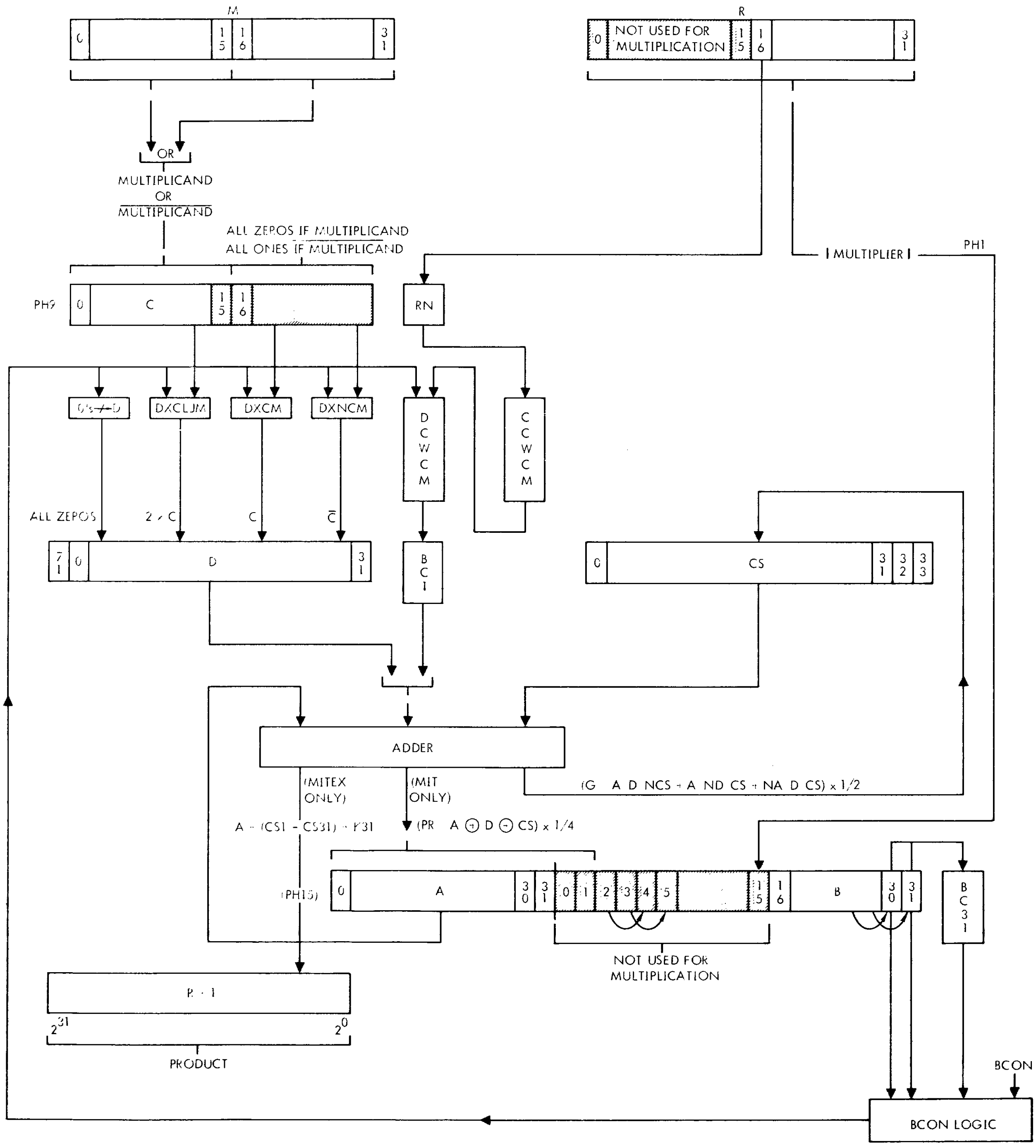
Mnemonic: MW (37, B7)
MI (23, 17)

(Continued)

Table 3-70. Multiply Word, Multiply Immediate, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T1L (Cont.)	Clear the D-register	DX/1 = MIT + . . .	
	Reset flip-flop MIT	R/MIT = .	
	Enable clock T10L if R is even	S/T10L = FAMULNH MITEX NR31 + . . .	
	Stop sustaining PH10	BRPH10 = MIT NMITEX + . . .	
	Set flip-flop PH11	S/PH11 = PH10 N(MIT NMITEX) + . . .	
PH11 T6L or T10L	Timing: T6L if R is odd, T10L if R is even		
	A + CS + K31 ⟶ S0-S31	SXADD = FAMDSF/M PH11 + . . .	
		SXK = FAMDSF/M PH11 + . . .	
	S0-S31 —/→ A0-A31	AXS = FAMDSF/M PH11 + . . .	Retained for PH15 testing
	S0-S31 —/→ RW0-RW31 if R is even	RW = FAMULNH PH11 NR31 + . . .	Store product bits 2^{63} - 2^{32}
	B + CS32, CS33, BC1 —/→ B0-B7	BXB = FAMDSF/M PH11 + . . .	B8-B31 remain unchanged
	Merge the contents of B0 through B3 into the contents of E0-E3	Path enabled by FAMUL PH11	For check of nonzero product in bits 2^{31} - 2^{0} in PH15
	Force a one into CS31 to enable K00 in PH15 for magnitude test	CSX1 = FAMUL PH11 + . . .	
	Force a one on private memory address line LR31	S/LR31/2 = FAMUL PH11 + . . .	To select the odd numbered private memory register for storage of bits 2^{31} - 2^{0} of the product
	Set flip-flop DRQ	S/DRQ = FAMUL PH11 + . . .	
	Set flip-flop PH15	BRPH15 = FAMUL PH11 + . . .	
	Enable clock T8L	S/T8L = FAMUL PH11 + . . .	
PH15	B0-B31 ⟶ S0-S31	SXB = FAMULNH PH15 + . . .	
	S0-S31 ⟶ RW0-RW31	RW = FAMDSF PH15 + . . .	Store product bits 2^{31} - 2^{0}
	Set flip-flop TESTA	S/TESTA = FAMDSF NFASHFX PH15 + . . .	
	Force a one into A31 if the contents of E are not equal to zero	A31X1 = FAMULNH PH15 NEZ + . . .	If E ≠ 0, the product bits 2^{31} - $2^{0} \neq 0$
	Set CC2, CC3, CC4 as applicable	S/CC2 = FAMULNH PH15 (NB0 NA0031Z + B0 NK00) + . . .	
		S/CC3 = TESTA NA0 NA0031Z + . . .	
		S/CC4 = TESTA A0 + . . .	
	Enable ENDE	ENDE = FAMDSF PH15 + . . .	

Mnemonic: MW (37, B7)
MI (23, 17)



901060A.3665

Figure 3-164. Multiply Halfword Register Organization and Flow Chart

In either case, this portion of the multiplicand (shaded area) produces zeros in the lower order bits of the product; the lower order bits are then discarded. The effect obtained is the same as by using a 16-bit multiplicand. As with instructions MW and MI, the D-register holds the multiple of the multiplicand, which has been transferred under control of the BCON logic. The iteration process is also the same. The A-register holds the sums, and the CS-register holds the carries. The B-register is shifted as before and initially holds the multiplier in bit positions 16 through 31. Bit positions 0 through 15 hold the other halfword from the private memory register. These bits do not take part in the multiplication. At the end of eight iterations, the A-register contains the 32 sum bits of the final product, and the CS-register holds the final carries. The sums and carries are added to produce the 32-bit product which is placed in the odd numbered private memory register.

The following paragraphs describe the phases of instruction MH. Figure 3-164 and table 3-71 supplement the discussion.

The preparation sequence is the same as the general preparation sequence, with the following exceptions: During PRE2, the contents of the even numbered private memory register are clocked into the A-register by signal AXRR. The multiplier consists of bit positions 16 through 31; bit positions 0 through 15 are clocked into the A-register, but are not used in the multiplication. The sign of the multiplier, contained in bit position 16, is stored in flip-flop RN. Ones are clocked into the CS-register by signal CSX1 in preparation for a possible multiplier sign adjustment.

During the preparation phase of the instruction, the word containing the effective halfword is read into the C-register from core memory. At the time of the PH1 clock, the effective halfword is clocked by signal DXC/5 into bit positions 16 through 31 of the D-register from either the least significant halfword or the most significant halfword location in the C-register. Choice of halfword location is dependent upon the state of P32. Also dependent upon P32, signal C0C16 goes true if C0 (P32 true) or C16 (P32 false) is true. If C0C16 is true, ones are clocked into bit positions 0 through 15 of the D-register (temporary multiplicand storage) to sign-pad the negative halfword multiplicand.

Sign manipulation identical to instructions MW and MI takes place in PH1. If the multiplier is positive, the multiplier is gated to the sum bus by signal SXA and is clocked into the B-register by signal BXS. The multiplicand is not inverted in the following phases. If the multiplier is negative, it is inverted by an exclusive OR operation with the CS-register (all ones). The inverted multiplier is clocked into the B-register, and flip-flop BC31 is set. The contents of the B-register and BC31 make up the two's complement of the original multiplier. The multiplier is inverted in PH9 to complete the sign manipulation. Other operations during PH1 include clearing the A-register by signal AX/1, clearing the P-register by signal PX, and enabling clock T8L for PH2. Flip-flops CXS and NPRX are also set at this time in preparation for PH2.

During PH2, the halfword multiplicand in D16 through D31 is upward aligned into bit positions 0 through 15 of the C-register by signals SXUAH/1 and CXS. Zeros are present in bit positions 16 through 31 of the C-register. The C-register contents are transferred to the D-register at the PH2 clock by signal DXC/6. The D-register now contains the multiplicand in bit positions 0 through 15, followed by zeros in bit positions 16 through 31.

The remainder of PH2 is identical to instructions MW and MI. Ones are clocked into the CS-register by signal CSX1, and if the multiplier is negative, the CXS flip-flop is set for inversion of the multiplicand in PH9. The first bit-pair of the multiplier is examined, and the multiplier control flip-flops are set. The B-register shifts two bit positions to the right, bringing the next bit-pair into B30, B31. Zeros are clocked into B0 and B1. PH9 is enabled.

PH9 of instruction MH is the same as instructions MW and MI. In summary:

a. The contents of the C-register are clocked into the D-register as a function of the multiplier control flip-flops set by bit-pair 2^1, 2^0 in PH2.

b. Bit-pair 2^3, 2^2 is examined. The multiplier control flip-flops are set again.

c. The B-register shifts two bit positions to the right, bringing bit-pair 2^5, 2^4 of the multiplier into B30 and B31. Zeros are clocked into B0 and B1.

d. Flip-flop MIT is set.

The D-register now contains the first partial product, the A-register contains all zeros, and the CS-register contains all zeros.

PH10 is identical to instructions MW and MI except that eight iterations are performed, and iteration counting is modified. Eight iterations are involved, since only eight bit-pairs in the multiplier are examined. At each of the eight clocks in PH10, the following operations occur:

a. The D-register is combined with the A- and CS-registers in the adder to produce the partial product for the iteration.

b. The C-register is clocked into the D-register as a function of the multiplier control flip-flops set during the previous clock.

c. The next bit-pair of the multiplier is examined, and the multiplier control flip-flops are set.

d. The B-register shifts two bit positions to the right, bringing a new bit-pair into B30 and B31. The last bit-pair examined is bit-pair 2^{16}, 2^{15}.

e. The number of the iteration is counted. The iteration counting scheme is identical to instructions MW and

Table 3-71. Multiply Halfword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence except for PRE2 NIA		
PRE2 NIA	RR0-RR31 $\not\rightarrow$ A0-A31	AXRR = FAMDSF PRE2 NIA + . . .	Multiplier is in RR16-RR31
	Set flip-flop RN if halfword multiplier negative	S/RN = RR16 FAMULH PRE2 NIA + . . .	Store multiplier sign
	Force ones into CS0-CS31	CSX1 = PRE2 NIA FAMUL + . . .	To produce one's complement of multiplier in PH1 if RN
PH1	MB0-MB31 → C0-C31	Preparation control	Multiplicand $\not\rightarrow$ C-register
	C0-C16 $\not\rightarrow$ D0-D15	C0-C16 = RR0 NP32 + RR16 P32	Sign pad
	C0-C16 $\not\rightarrow$ D16-D31 if NP32	DXC/5 = OU5 (NO4 O5 O6) PH1 + . . .	Downward aligned halfword multiplicand $\not\rightarrow$ C-register
	C16-C31 $\not\rightarrow$ D16-D31 if P32		
	A0-A31 → S0-S31 if A positive	SXA = FAMUL PH1 NRN + . . .	Absolute value of multiplier transferred to the B-register and BC31
	A0-A31 $\oplus$ CS0-CS31 → S0-S31 if A negative	SXPR = FAMUL PH1 RN + . . .	
	Set flip-flop BC31	S/BC31 = FAMUL PH1 RN + . . .	
	S0-S31 $\not\rightarrow$ B0-B31	BXS = FAMDSF PH1 + . . .	
	Clear A-register	AX/1 = FAMDSF PH1 + . . .	
	Clear P-register	PX = FAMUL PH1 + . . .	
	Set flip-flop CXS	S/CXS = FAMULH PH1 + . . .	For PH2 upward alignment of halfword
	Force a one into NPRX	S/NPRX = FAMULH PH1 + . . .	For PH2 upward alignment of halfword
	Enable clock T8L	S/T8L = FAMULH PH1 + . . .	
PH2 T8L	D0-D31 $\oplus$ CS0-CS31 → K0- K31		
	K15-K30 → S0-S15	SXUAH/1 = FAMULH PH2 + . . .	Zeros on S16-S31
	S0-S31 → C0-C31	CXS = Previously set in PH1	
	C0-C31 $\not\rightarrow$ D0-D31	DXC/6 = FAMUL PH2 + . . .	
	Force ones into CS0-CS31 for PH9 if multiplier is negative	CSX1 = FAMUL PH2 RN + . . .	For PH9 multiplicand sign adjustment
	Set flip-flop CXS for PH9 if multiplier is negative	S/CXS = FAMUL PH2 RN + . . .	For PH9 multiplicand sign adjustment
	B0-B29 $\not\rightarrow$ B2-B31	BXBR2 = FAMUL PH2 + . . .	Examine multiplier bit-pair 2^1, 2^0. Set multiplier control flip-flops to this bit-pair
	Enable BCON	BCON = BXBR2 FAMDSF/M + . . .	

Mnemonic: MH (57, D7)

(Continued)

Table 3-71. Multiply Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH2 T8L (Cont.)	Set up flip-flops DXCM, DXNCM, DXCLIM, BC31 as functions of B30, B31, and BC31			Shift multiplier to bring next pair into B30, B31
	Set flip-flop PH9	BRPH9	= FAMUL PH2 + . . .	
PH9 T6L	D0-D31 $\oplus$ CS0-CS31 $\longrightarrow$ S0-S31	SXPR	= FAMDSF/M PH9 + . . .	Final multiplicand sign adjustment in case of negative multiplier. C already contains multiplicand if multiplier is positive
	S0-S31 $\longrightarrow$ C0-C31	CXS	= Previously set in PH2	
	C0-C31 —/→ D0-D31 or	DXCM	= Previously set in PH2	
	NC0-NC31 —/→ D0-D31 or	DXNCM	= Previously set in PH2	
	C0-C31 —/→ D71-D30 or Zeros transferred to D71-D30 if none of the above	DXCLIM	= Previously set in PH2	First transfer of multiplicand under control of bit-pair 2^1, 2^0.
	Force a one into flip-flop DCWCM if one's complement is transferred to D	S/DCWCM	= CCWCM (DXCM + DXCLIM) + NCCWCM DXNCM	Records one's complement condition so that a one can be added later to make up two's complement
	B0-B29 —/→ B2-B31	BXBR2	= FAMDSF/M PH9 + . . .	Examine multiplier bit-pair 2^3, 2^2. Set multiplier control flip-flops to this bit-pair. Shift multiplier to bring next bit-pair into B30, B31
	Sustain BCON	BCON	= FAMDSF/M BXBR2 + . . .	
	Set up flip-flops DXCM, DXNCM, DXCLIM, B31 as functions of B30, B31, BC31			
	Set flip-flop MIT	S/MIT	= FAMDSF/M PH9 NCLEAR + . . .	
	Enable clock T1L if not a single clocking	S/T1L	= FAMDSF/M PH9 NKSC N(S/FLMC/1 P18) + . . .	
PH10 T1L	Eight clocks. During the first 6 clocks, the following events occur:			
	A $\oplus$ D $\oplus$ CS $\longrightarrow$ PR0-PR31			
	PR0-PR31 —/→ A2-A31, B0, B1	AXPRR2	= MIT + . . .	Sums of the partial products
	A D + A CS + D CS $\longrightarrow$ G0-G31			
	G0-G31 —/→ CS1-CS32	CSXGR1	= MIT	Carries of the partial products
	Contents of C are transferred to D under control of the multiplier control flip-flops DXCM, DXNCM, DXCLIM, set during previous clock			

Mnemonic: MH (57, D7)

(Continued)

Table 3-71. Multiply Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T1L (Cont.)	DCWCM set as before if one's complement is transferred to D		
	B0-B29 $\nrightarrow$ B2-B31	BXBR2 = MIT + . . .	
	PR30, PR31 $\nrightarrow$ B0, B1	BXBR2 = MIT + . . .	Least significant bits of partial product
	Sustain BCON	BCON = BXBR2 N(MULC + MITEX)	
	Set up flip-flops DXCM, DXNCM, DXCLIM, B31 as functions of B30, B31, BC31		
	Merge contents of B2, B3 into contents of E2, E3	Path enabled by FAMUL MIT	For detection of a zero product in product bits 227 - 2^0
	Count iterations using P15-P18 and MP19	S/MP19 = MIT N(MULC + MITEX) NMP19 BCON	High on all even clocks but the last
		R/MP19 = .	
		PCTP5 = MP19	
	<u>On the 6th clock</u>:		
	Other events listed above		
	Set flip-flop MULC	S/MULC = FAMULH P17 + . . .	
	Reset clock T1L	R/T1L = MIT NMP19	T1L timing lasts through PH10
	Set flip-flop MRQ/M	S/MRQ/M = (S/MULC/1) MP19	Request the next instruction
	<u>On the 7th clock</u>:		
	Other events listed above		
	Set flip-flop MITEX	S/MITEX = FAMUL MULC + . . .	To signal end of multiply iterations

	Enable MRQ
	Clear the P-register
	Q15-Q31 $\nrightarrow$ P15-P31
		MRQ = MRQ/M + . . .
		PX = PXQ
		PXQ = MULC + . . .
			No longer needed for iteration counting. Transfer next instruction address

	<u>On the 8th clock</u>:		
	Other events listed above		
	Clear the D-register	DX/1 = MIT + . . .	
	Reset flip-flop MIT	R/MIT = .	
	Stop sustaining PH10	BRPH10 = MIT NMITEX + . . .	
	Set flip-flop PH11	S/PH11 = PH10 N(MIT NMITEX) + . . .	

Mnemonic: MH (57, D7)

(Continued)

Table 3-71. Multiply Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH11 T6L	A + CS + K31 → S0-S31	SXADD	= FAMDSF/M PH11 + . . .	
		SXK	= FAMDSF/M PH11 + . . .	
	S0-S31 —/→ A0-A31	AXS	= FAMDSF/M PH11 + . . .	
	B + CS32, CS33, BC1 —/→ B0-B7	BXB	= FAMDSF/M PH11 + . . .	B0-B19 = 0, B20-B31 is meaningless
	Merge the contents of B0 through B3 into the contents of E0 through E3	Path enabled by FAMUL PH11		For check of nonzero product in bits $2^{31} - 2^{0}$ in PH15
	Force a one on private memory address line LR31	S/LR31/2	= FAMUL PH11 + . . .	To select the odd numbered private memory register for storage of the product
	Set flip-flop DRQ	S/DRQ	= FAMUL PH11 + . . .	
	Set flip-flop PH15	BRPH15	= FAMUL PH11 + . . .	
	Enable clock T8L	S/T8L	= FAMUL PH11 + . . .	
PH15	A0-A31 → S0-S31	SXB	= FAMULNH PH15 + . . .	
	S0-S31 → RW0-RW31	RW	= FAMDSF PH15 + . . .	Store product
	Set flip-flop TESTA	S/TESTA	= FAMDSF NFASHFX PH15 + . . .	
	Set CC3, CC4 as applicable	S/CC3	= TESTA NA0 NA0031Z + . . .	
		S/CC4	= TESTA A0 + . . .	
	Enable ENDE	ENDE	= FAMDSF PH15 + . . .	

Mnemonic: MH (57, D7)

MI except that terminal signals are detected at clocks six through eight rather than through 16. Figure 3-165 shows the counting configurations and the signals enabled for instruction MH. At clock 6, flip-flop MULC is set. A true MULC signal sets flip-flop MITEX at clock 7; MITEX resets flip-flop MIT at clock 8. At the sixth clock, also, flip-flop T1L is reset to initiate the end of T1L (T1L timing lasts through PH10), and memory request flip-flop MRQ/M is set to request the next instruction. At clock 7, the P-register is cleared (signal PX), and the address of the next instruction is clocked into the P-register by signal PXQ. The seventh clock is the last transfer of the C-register contents to the D-register under control of the multiplier control flip-flops. At the eighth clock, the D-register is cleared by signal DX/1 and flip-flop MIT is reset. This signifies the end of the multiply iterations. The last addition of the A-, D-, and CS-registers occurs at clock 8; the product of the multiplication is now contained in A0 through A31, B0 through 31 (sums), CS1 through CS33, and BC1 (carries). The product bits in B4 through B31 are irrelevent; the final product is contained in A0 through A31 after the final addition of the sums and carries. PH11 is set by MITEX.

During PH11, the final product of the multiplication is assimilated from the results of the last iteration in PH10. This is done by adding the contents of the A-register to the contents of CS1 through CS31 and K31 (carry from the lower-order bits). The addition is shown in figure 3-166. The sum, S0 through S31, is clocked back into the A-register, which now contains the complete product of the halfword multiplication. K31 is produced by the addition of B0 and B1 to CS32, CS33, and BC1 at the PH11 clock. Flip-flop BC1 contains a one if the one's complement was put into the D-register on the seventh iteration of PH10. The sum of B0 and B1, CS32 and CS33, and BC1 for this last addition is zero. Zeros are clocked into B0 and B1 by signal BXB. The quantity in B2 through B19 is also zero; the contents of B20 through B31 are meaningless.

At the PH11 clock, a one is forced on private memory address line LR31 to select the odd numbered private memory register for storage of the product. Flip-flop DRQ is set, inhibiting transmission of another clock until the memory data release signal is received. PH13 is enabled by setting flip-flop PH15. Flip-flop T8L is set for PH15.

During PH15, the product in the A-register is gated to the sum bus by signal SXA. At the PH15 clock, the product is read into the odd numbered private memory register. Flip-flop CC3 is set if bits 2^{31} through 2^{0} of the product are non-zero and if the 2^{31} bit is not a one. Flip-flop CC4 is set if the 2^{31} bit is a one. Signal ENDE is generated.

A sequence chart for Multiply Halfword is given in table 3-71.

3-201 Divide Word (DW 36, B6) and Divide Halfword (DH 56, D6)

Instructions DW and DH are similar in implementation and result. Both instructions are discussed together; however, specific differences between instruction DW and DH are noted.

Instruction DW divides the contents of an even and an odd numbered private memory register by the effective word from core memory if the R-field of the instruction word is an even value. The resulting 32-bit quotient is stored in the odd numbered private memory register, and a 32-bit remainder is stored in the even numbered private memory register. If the R-field of the instruction word is an odd value, the contents of the odd numbered private memory register are divided by the effective word from core memory, and the 32-bit quotient is loaded back into the odd numbered

CLOCK (PH10)	P15	P16	P17	P18	MP 19	
1	0	0	0	0	0	
2					1	
3				1	0	
4				1	1	
5			1	0	0	
6			1	0	1	S/MULC, R/T1L, S/MRQ/M
7	NOT APPLICABLE					S/MITEX, MRQ, Q→P BY PX
8	NOT APPLICABLE					0→D, R/MIT

901060A.3666

Figure 3-165. Multiply Halfword Iteration Counting

private memory register. The remainder is lost in this case. When the R-field is odd, the result of the instruction is the same as if instruction DH had been performed.

Instruction DH divides the contents of the even or the odd numbered private memory register by the effective halfword from core memory, and loads the quotient back into the same private memory register. The remainder is lost.

Instructions DW and DH use condition code bits 2 through 4. Fixed-point overflow may occur in instruction DW if the quotient cannot be represented correctly in 32 bits, or the overflow may occur in instruction DH if division is by -2^{31}, -1, or 0. The condition codes shown in table 3-72 apply to the two instructions.

Table 3-72. Instruction DW and DH Condition Codes

CC1	CC2	CC3	CC4	Meaning
-	0	0	0	Quotient zero, no overflow
-	0	0	1	Quotient negative, no overflow
-	0	1	0	Quotient positive, no overflow
-	1	-	-	Fixed-point overflow

3-202 NONRESTORING DIVISION. The Sigma 7 computer uses the method of nonrestoring division for instructions DW and DH. A review of division techniques may help in understanding nonrestoring methods.

Restoring division is the most basic type of division. Long division is an example of restoring type. Subtractions are repeated to obtain a quotient. An example of restoring binary division is shown in figure 3-167. Sign bits are not used in this example.

Multiples of the divisor (hereafter called denominator) are subtracted from the dividend (hereafter called numerator) or partial dividend (hereafter called residue). The first subtraction in this example is ($1011_2 \times 2^5$) or 101100000_2. Successive subtractions involve the demoninator times 2^4, 2^3, and so forth, for as many times as necessary. Each subtraction yields either a one or a zero in the quotient, depending upon whether the residue is positive or negative. A negative residue signifies that the denominator multiple is larger than the previous residue and could not be contained in it. The previous positive residue, therefore, is restored. A 0 is put in the quotient for that order, and the next lower order multiple is subtracted. Figure 3-168 is a representation of restoring division using the same numbers as in figure 3-167. At no time may the residue be negative. The zero point (residue = 0) is always approached from the positive side. The total of all multiples subtracted and the remainder (the last residue) equals the numerator.

Nonrestoring division is similar to restoring division in that repeated subtractions of multiples of the denominator are used; however, it differs from the nonrestoring type because the residue may be negative. Figure 3-169 is an example of nonrestoring binary division, using the same numerator and denominator as before. Sign bits are not used. Successive subtractions of multiples of the denominator are performed as before. However, if the residue is negative, the previous positive residue is not restored. A negative residue signifies that a zero should be placed in the quotient for that order and that the next denominator multiple should be added to, rather than subtracted from, the residue. Every time the residue goes positive, a one is added to the appropriate order of the quotient and the next denominator multiple is subtracted. Every time the residue goes negative, a zero is added to the appropriate order of the quotient, and the next denominator multiple is added. The result is exactly the same as restoring division. The numerator is equal to the total of all of the multiples subtracted, minus the total of all the multiples added, plus the remainder. The zero point (residue = 0) is approached from both sides. Figure 3-170 illustrates how this type of division works.

Nonrestoring division using complementing techniques for subtractions is used in Sigma 7. The denominator multiple to be subtracted can then be added to the residue. Figure 3-171 shows a division of the previous numbers, using sign bits (MSD) and two's complementing. The division process in this example is the same as before. Each time a one appears in the sign bit of the residue, a zero is added to the appropriate order of the quotient, and the next denominator multiple is added (remaining in uncomplemented form). Each time that a zero appears in the sign bit of the residue, a one is added to the appropriate order of the quotient, and the next denominator multiple is subtracted (two's complement form is added). The end carry bit is a one each time that a positive residue is reached. Normally, in two's complement additions, this end carry is discarded. In Sigma 7 division, the end carry, designated K00, is used to signify that a positive partial dividend has been obtained. Nonrestoring division, complementing, and the end carry are used in all division cases. There are four different divisions in regard to sign - a positive divided by a positive $(\frac{+}{+})$, a positive divided by a negative $(\frac{+}{-})$, a negative divided by a negative $(\frac{-}{-})$, and a negative divided by a positive $(\frac{-}{+})$. In Sigma 7 division, special logic is used to produce the correct quotient in the $(\frac{+}{-})$, $(\frac{-}{+})$, and $(\frac{-}{-})$ cases.

Figure 3-171 is an example of the $(\frac{+}{+})$ case. Each time that the end carry is a one, a one is placed in the quotient for that order. Each time that there is no end carry, a zero is placed in the quotient. The final quotient is in the correct form with proper sign and magnitude.

Figure 3-172 shows the case of $(\frac{+}{-})$. If a one is put in the quotient for every end carry, the result is the two's complement of the correct quotient as shown, with the opposite sign of the correct quotient. To produce the correct quotient, therefore, a one is put in the quotient when there is no end carry.

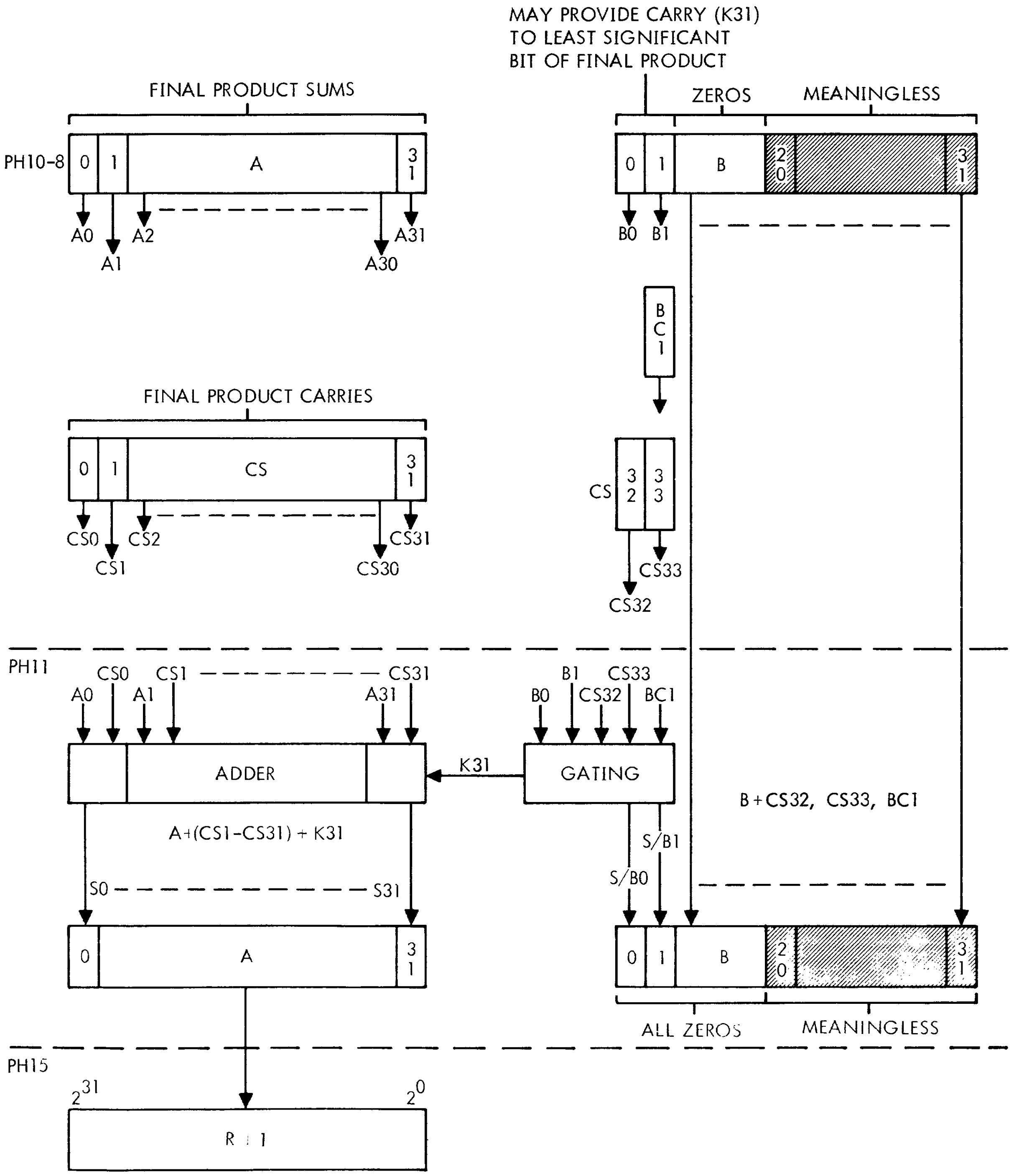


Figure 3-166. Multiply Halfword Final Product Assimilation

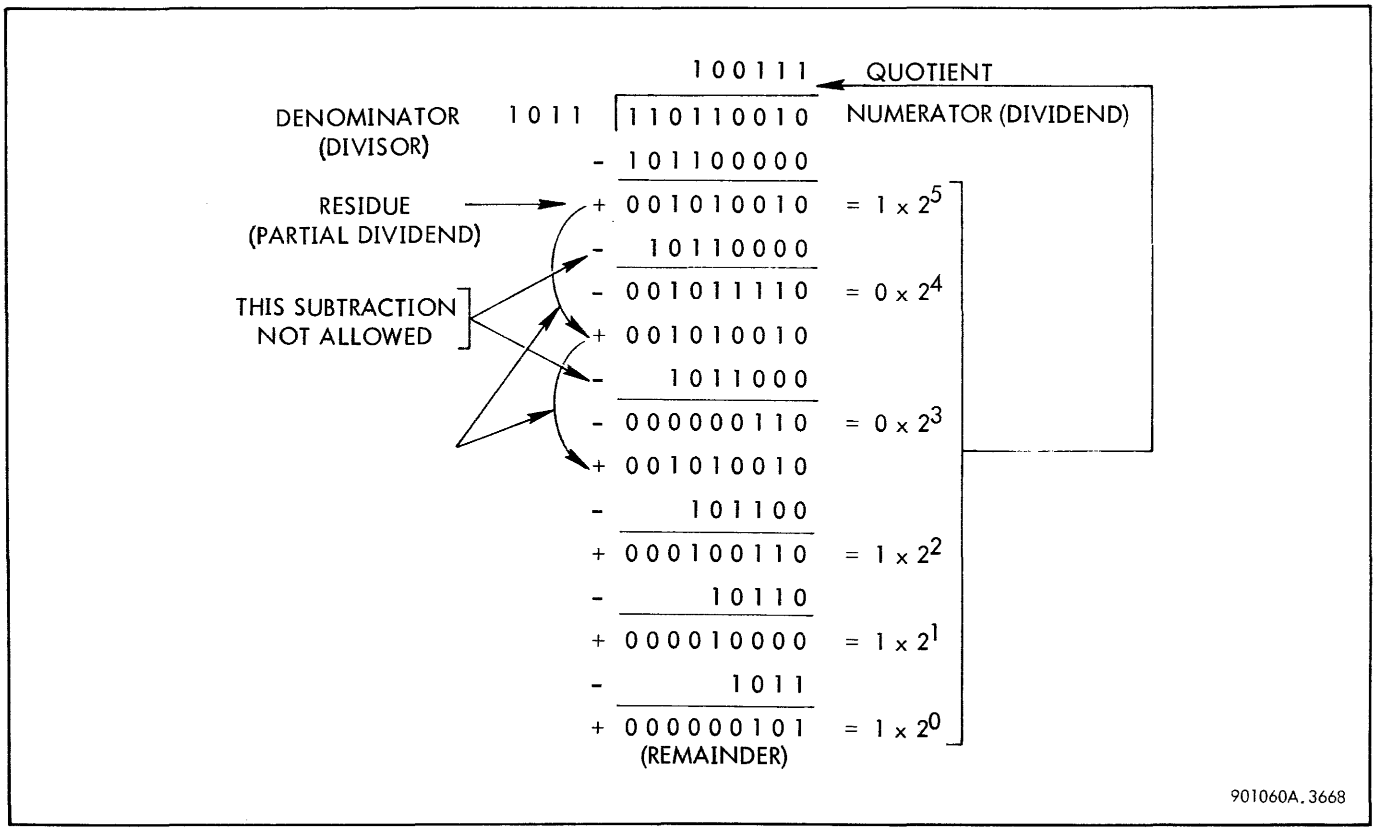


Figure 3-167. Restoring Division

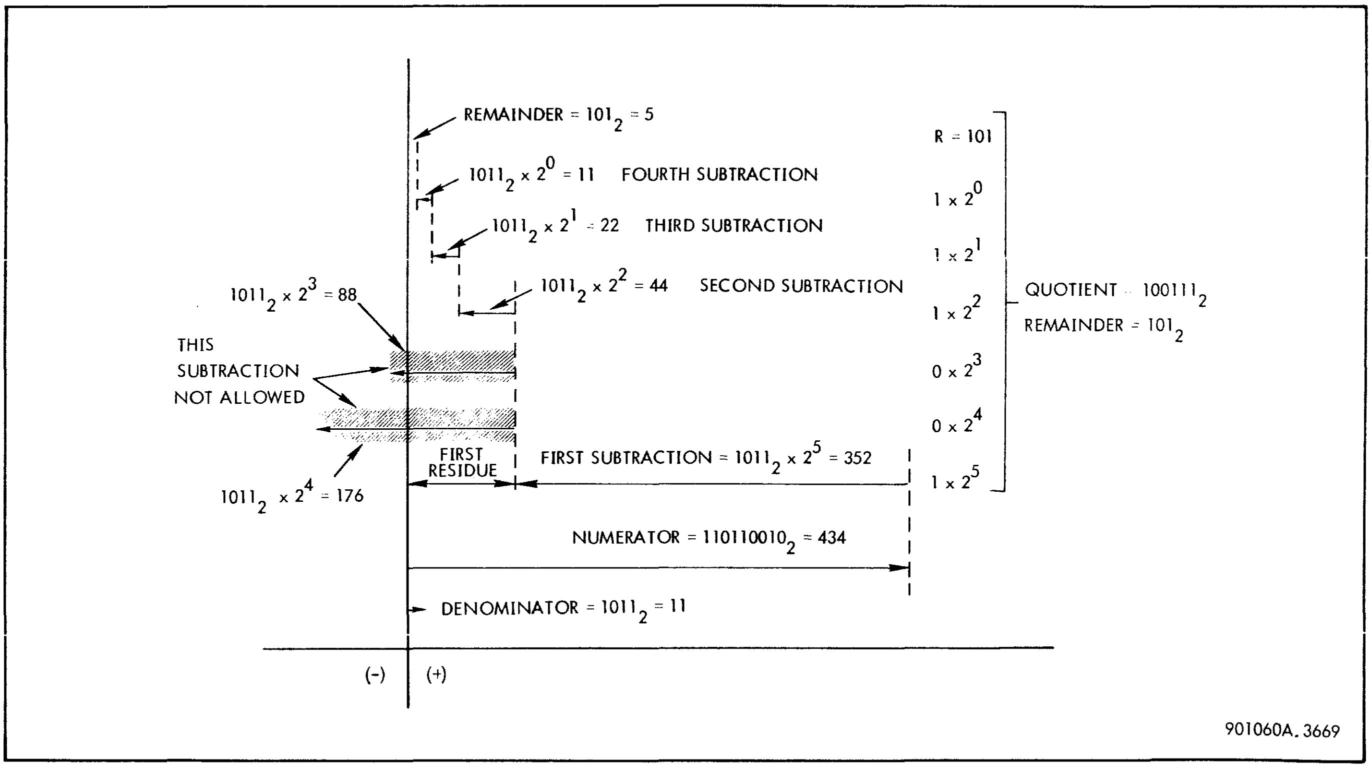

REMAINDER = 101₂ = 5
1011₂ x 2^0 = 11 FOURTH SUBTRACTION
1011₂ x 2^1 = 22 THIRD SUBTRACTION
1011₂ x 2^2 = 44 SECOND SUBTRACTION
1011₂ x 2^3 = 88
THIS SUBTRACTION NOT ALLOWED
1011₂ x 2^4 = 176
FIRST RESIDUE
FIRST SUBTRACTION = 1011₂ x 2^5 = 352
NUMERATOR = 110110010₂ = 434
DENOMINATOR = 1011₂ = 11
(-)
(+)
R = 101
1 x 2^0
1 x 2^1
1 x 2^2
0 x 2^3
0 x 2^4
1 x 2^5
QUOTIENT = 100111₂
REMAINDER = 101₂
901060A.3669

Figure 3-168. Restoring Division, Graphic Representation

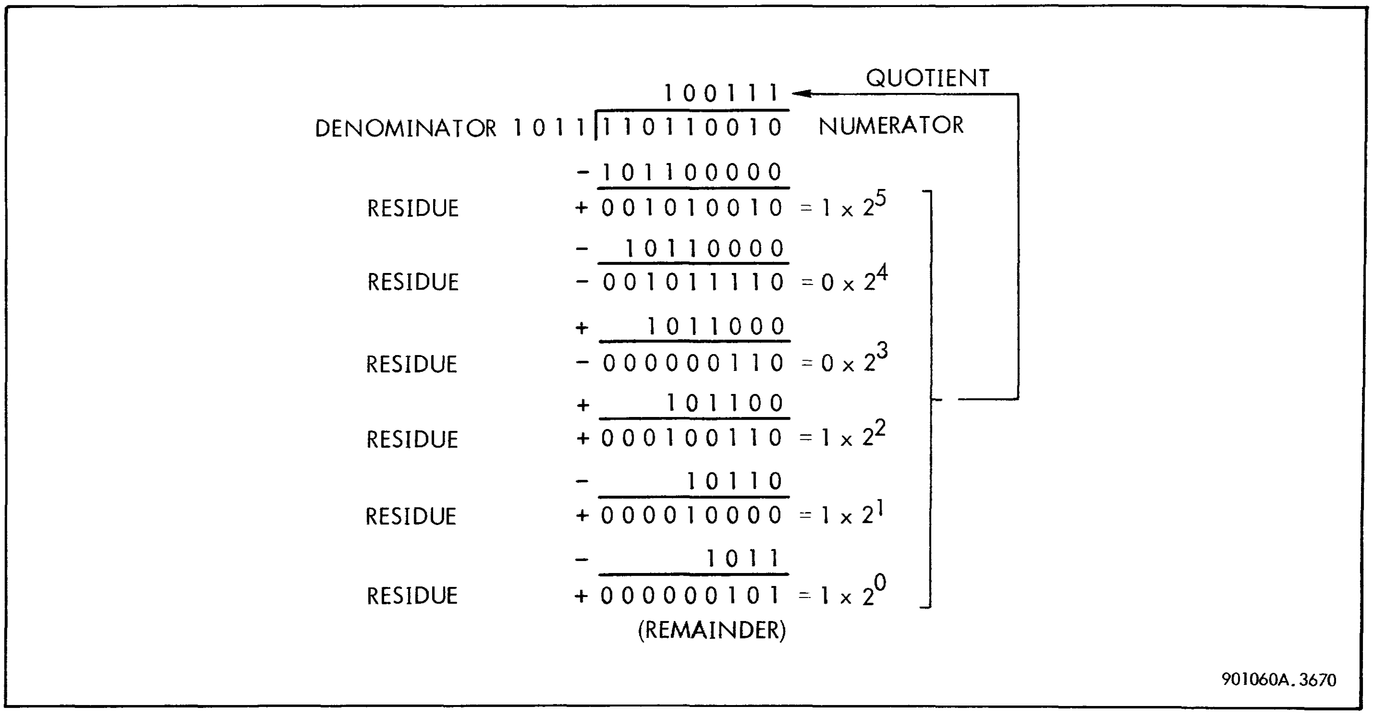


Figure 3-169. Nonrestoring Division

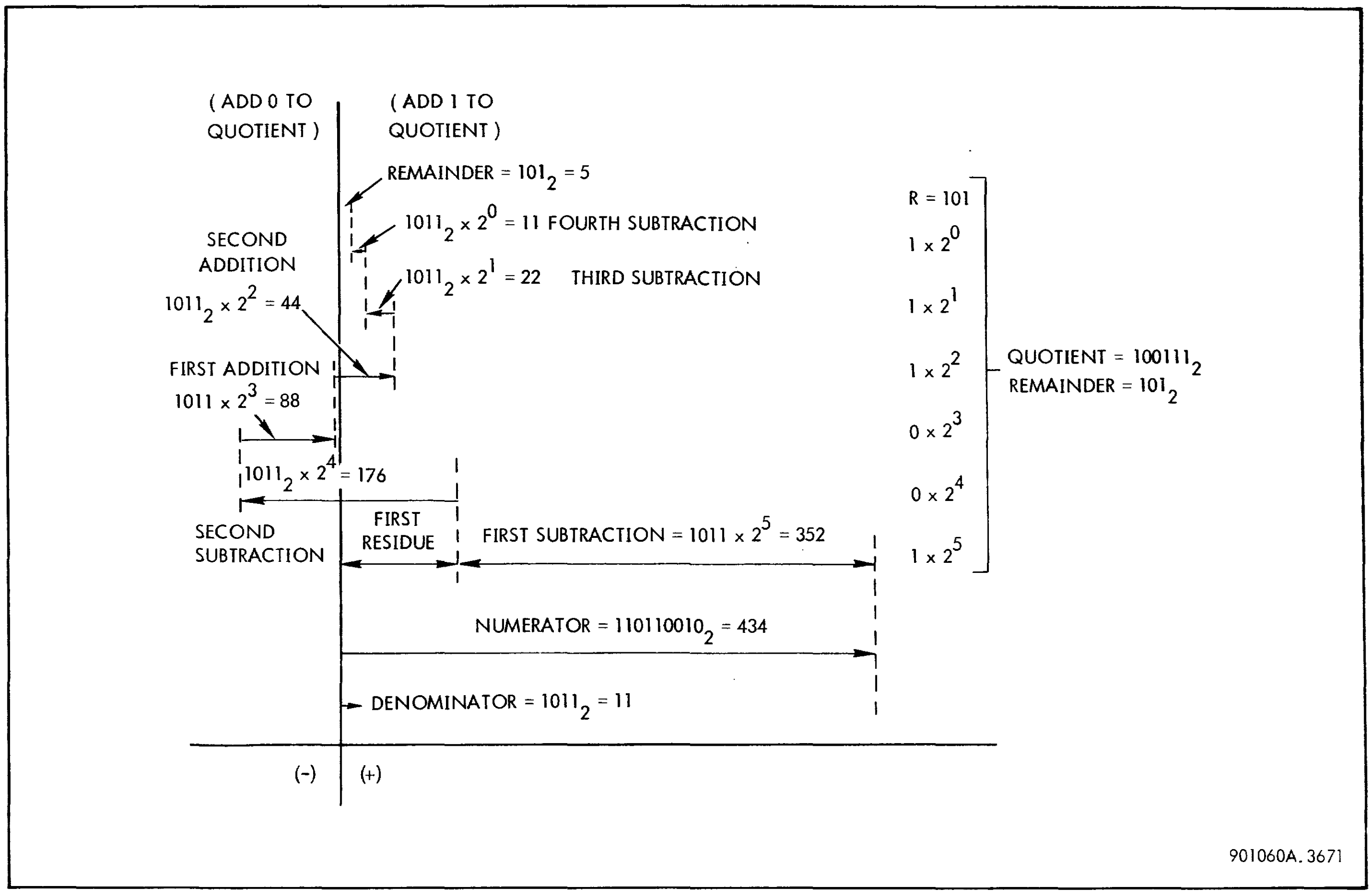


Figure 3-170. Nonrestoring Division, Graphic Representation

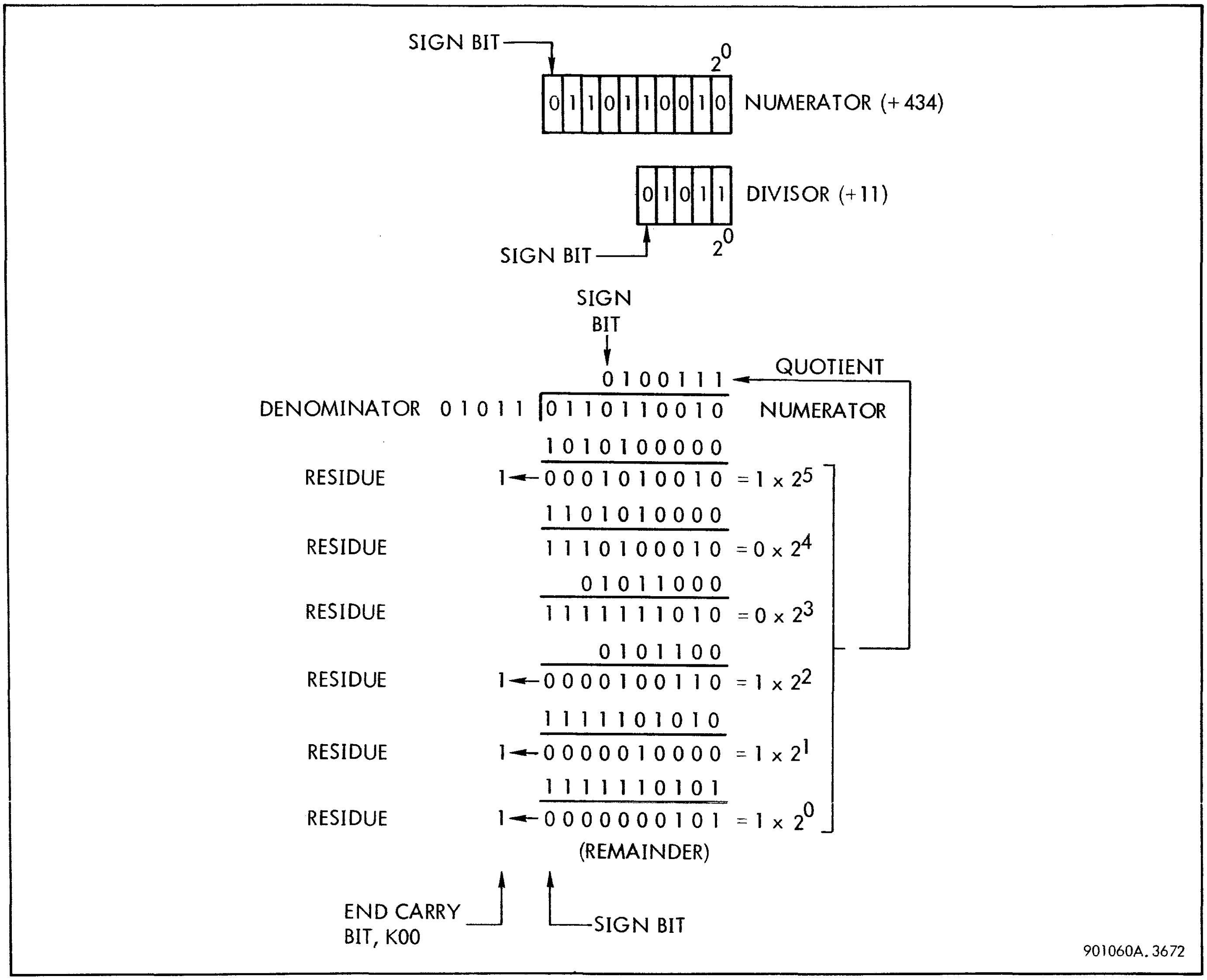


Figure 3-171. Nonrestoring Division with Two's Complement Addition

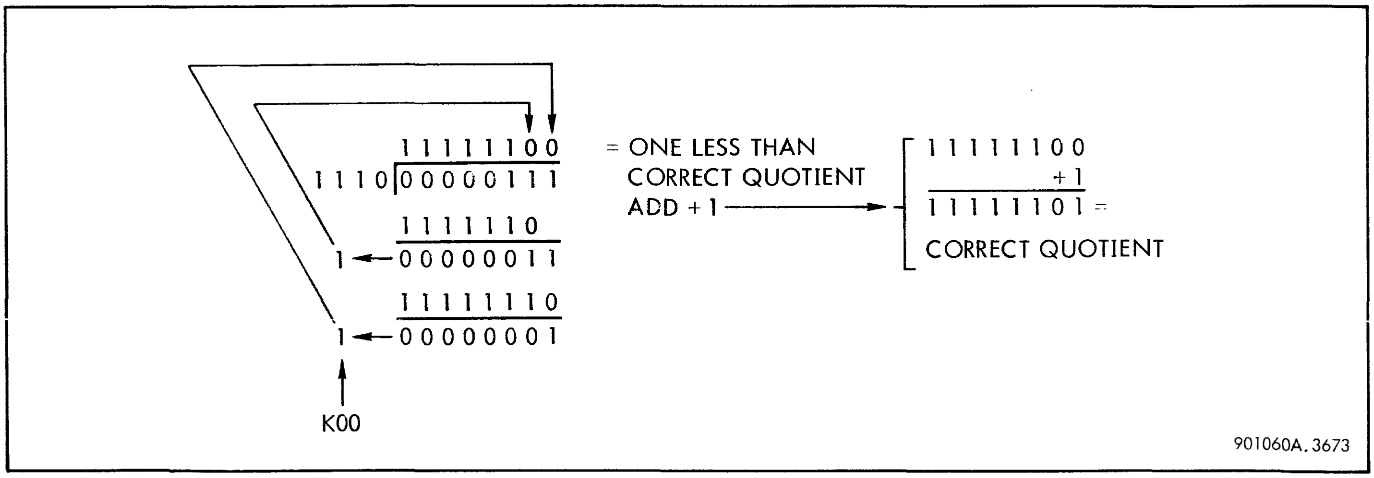


Figure 3-172. Plus/Minus Case in Division

A zero is put in the quotient every time that there is an end carry. Adding a one to this quantity produces the correct result.

Figure 3-173 shows the case of $(\frac{-}{+})$ which includes two subcases. If a zero residue is produced in iterations, the answer is correct as it stands. If, at the end of the division process, the residue has never been nonzero, the correct quotient is found by adding one to the quantity produced by the division. This difference in methods results from zero being a positive number; the nonrestoring division process becomes nonsymmetrical at the zero point. Both subcases produce a quotient bit of one when the end carry is one and produce a quotient bit of zero when the end carry is zero. The process used in both subcases is shown in the figure.

Figure 3-174 shows the $(\frac{-}{-})$ case. The subcase of the nonzero remainder and the subcase of a zero remainder are shown. In both cases, putting a one in the quotient when there is an end carry results in a quantity with an incorrect sign. A one is put in the quotient, therefore, when there is no end carry. A zero is used when there is an end carry. In the nonzero residue case, the result is the correct quotient. A one must be added to the quantity produced to obtain the correct remainder, in the zero residue case. All division cases and accompanying logic for producing the correct quotient are shown in table 3-73.

An eight-bit numerator divided by a four-bit denominator is shown in figure 3-175. Register nomenclature and division methods are similar to Sigma 7 division. The eight-bit numerator is held in the A- and the B-registers. The four-bit denominator is contained in the C-register.

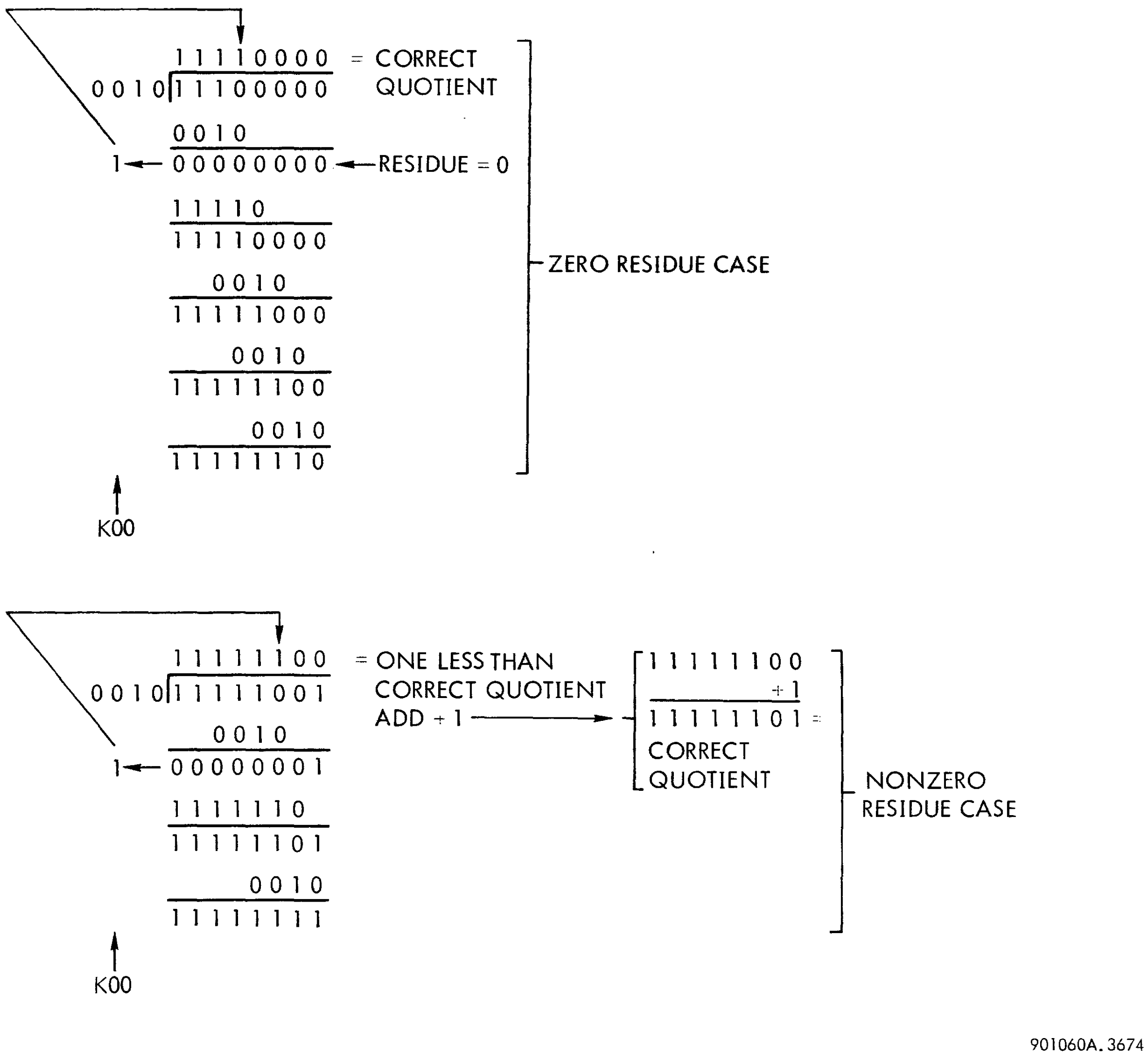


Figure 3-173. Minus/Plus Case in Division

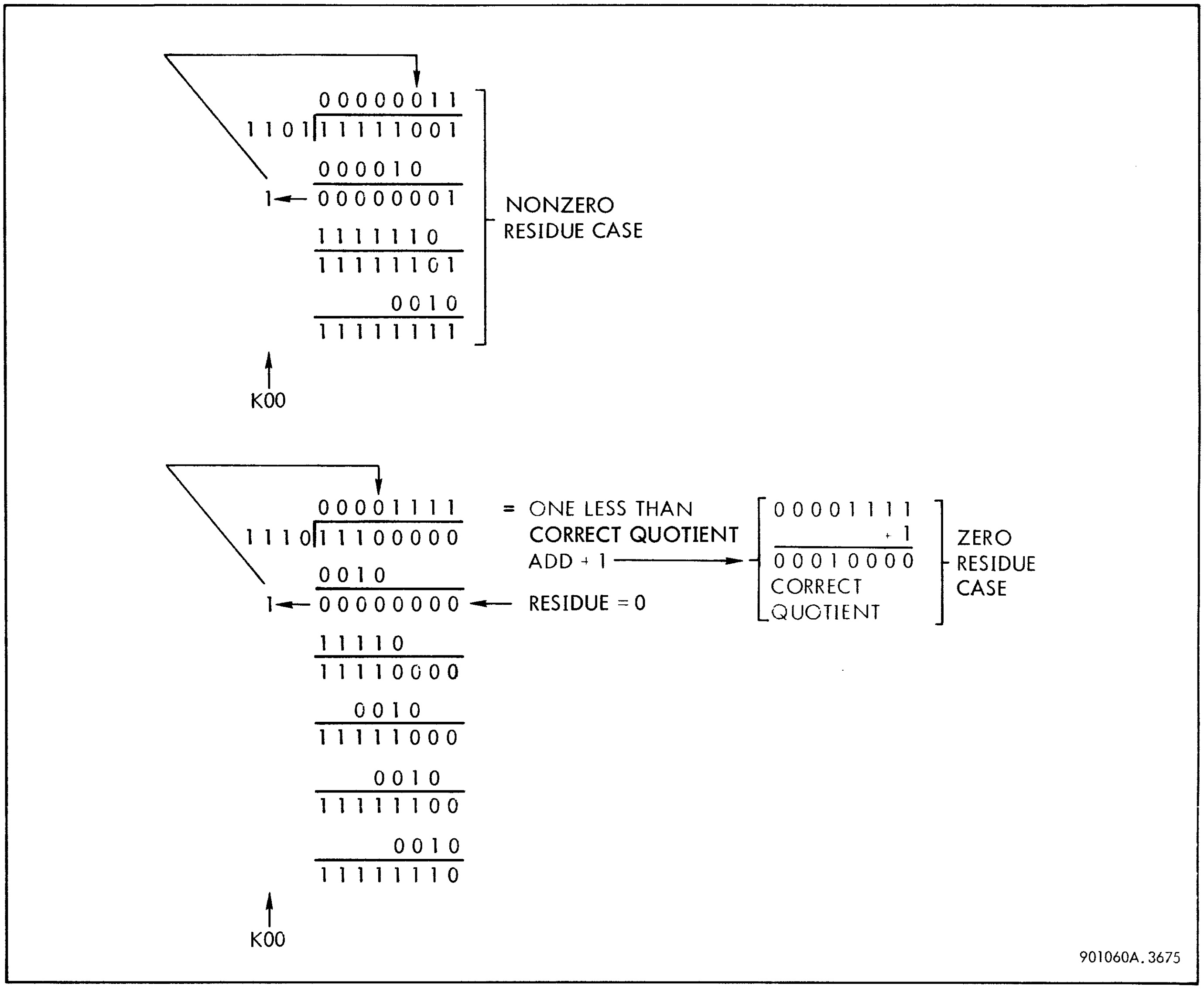


Figure 3-174. Minus/Minus Case in Division

Table 3-73. Quotient Logic

CASE		QUOTIENT BIT		FORM AT END OF DIVISION
		1	0	
Plus/Plus		K00	NK00	Correct; no changes necessary
Plus/Minus		NK00	K00	One less than correct quotient; add + 1 to form correct quotient
Minus/Plus	Zero residue	K00	NK00	Correct; no changes necessary
	Nonzero residue	K00	NK00	One less than correct quotient; add + 1 to form correct quotient
Minus/Minus	Zero residue	NK00	K00	One less than correct quotient; add + 1 to form correct quotient
	Nonzero residue	NK00	K00	Correct; no changes necessary

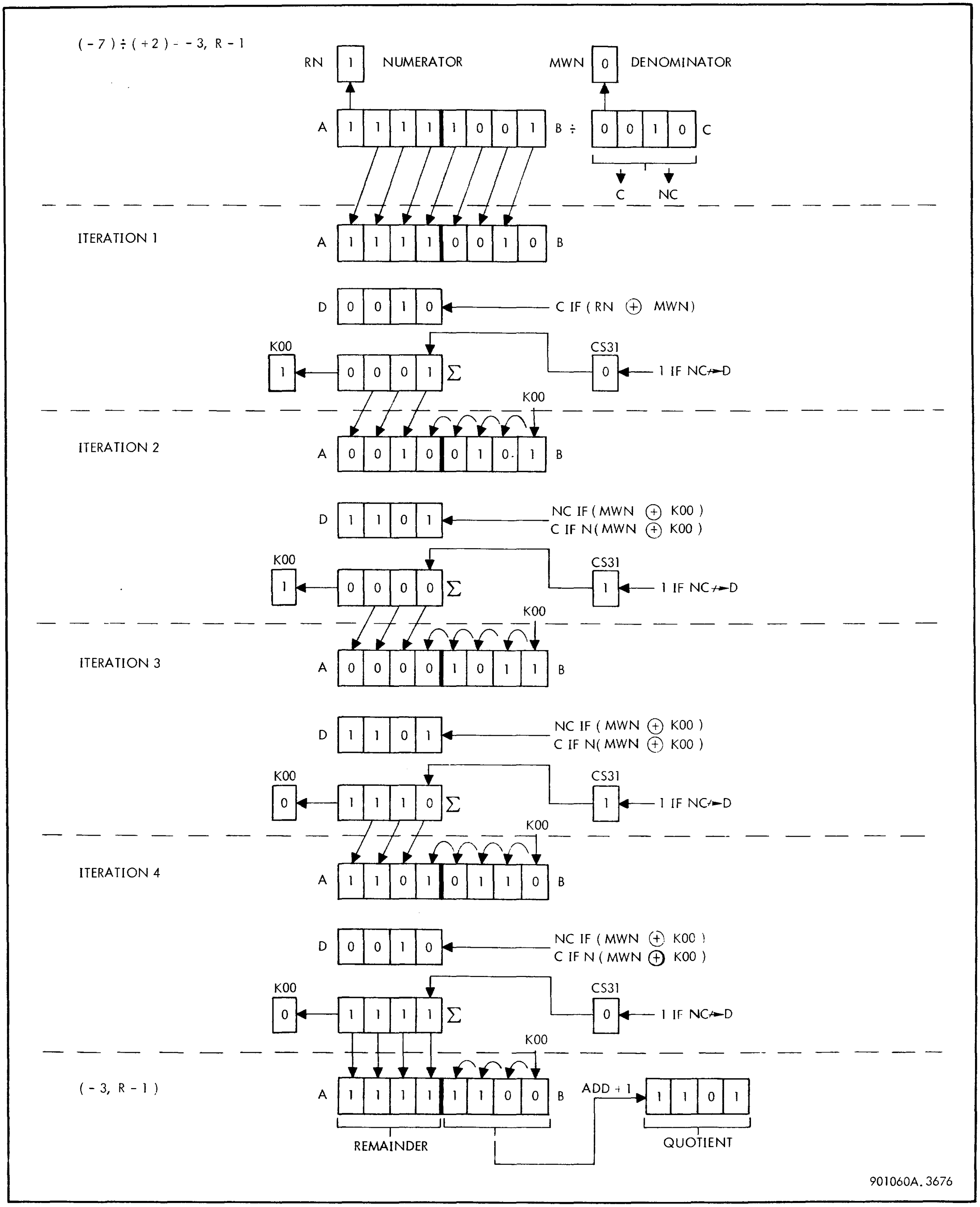


Figure 3-175. Sigma 7 Division

The signs of the numerator and the denominator are held in flip-flops RN and MWN, respectively. The size of the numerator and the demoninator are selected in such a way that the quotient may be held in a four-bit register ($|\frac{N}{d}| < 2^3$) The numerator is shifted one bit position to the right to bring the 2^3 bit of the numerator (the maximum bit position of the quotient) over the 2^0 denominator bit before the first iteration.

Either the true denominator (C) or the inverted denominator (NC) may be transferred to the D-register. If NC is transferred, a one is forced into flip-flop CS31 to make up the two's complement of the true denominator. The transfer to the D-register is always made so that the D-register holds the opposite sign to the numerator (or residue) in the A- and B-registers. As each denominator multiple is added to the numerator or residue, one quotient bit is produced. During each iteration, the residue from the addition is shifted one bit position to the left so that the next addition produces the quotient bit for the next lower order. Quotient bits are produced according to the method of the case; each quotient bit is transferred to the least significant bit position of the B-register. After the last iteration, the residue is transferred directly to the A-register; and the quotient is adjusted (see table 3-73).

3-203 DESCRIPTION OF DIVIDE WORD INSTRUCTION. Figure 3-176 shows the register arrangement during instruction Divide Word (DW), FADIVW. Division is similar to the example in figure 3-175. The numerator from the even and odd numbered private memory registers is contained in flip-flop RN (numerator sign), the A-register (numerator bits 2^{62} through 2^{31}), and the B-register bit positions 0 through 30 (numerator bits 2^{30} through 2^0). The C-register holds the true denominator from core memory. Flip-flop MWN holds the sign of the denominator. Either the true denominator or the inverted denominator may be transferred from the C-register to the D-register. The registers are aligned so that the 2^{31} bit is the first quotient bit produced. After each iteration except the last, the residue is shifted one place to the left and is transferred back to the A-register. At the same time, the B-register is shifted one place to the left; the contents of B0 are transferred to A31. Quotient bits are clocked into B31. During the iterations, the quotient fills up more of the B-register as the A- and B-registers shift left, and the quotient bits are fed into B31 (the residue becomes smaller). After the 32nd iteration, the B-register holds the 32-bit quotient and the A-register holds the last residue. The quotient is corrected, if necessary, and the remainder is restored from the residue. The quotient is clocked into the odd numbered private memory register, and the remainder is clocked into the even numbered private memory register.

A fixed-point overflow condition exists, if the absolute value of the numerator is relatively large compared with the absolute value of the denominator, and the quotient is larger or equal to 2^{31} (too large to be contained in a 32-bit register if a sign bit is to be present. This condition is detected after the first division iteration and results in a trap to memory location 67.

During PRE1, a one is forced on private memory address line LR31 to obtain the least significant word of the numerator in the odd numbered private memory register. During PRE2, the odd numbered private memory register is clocked into the A-register. The remainder of the preparation sequence is the same as the general preparation sequence.

The following paragraphs describe the phases of instruction DW. Figure 3-176 and the sequence chart, table 3-74, supplement the discussion.

During PH1, the A-register contents are gated to the sum bus by signal SXA. At the PH1 clock, the sum bus output is clocked into the B-register by signal BXS. Flip-flop BWZ (B was zero) is set if bit positions 1 through 31 of the A-register contain zeros. Signal BWZ is used in PH11 for the overflow test. At the PH1 clock, the even numbered private memory register contents are transferred to the A-register. The A-register now contains the most significant word of the numerator and the B-register contains the least significant word. Flip-flop RN is set if RR0 is true. The denominator from core memory has been read into the C-register at this time; if the denominator is negative, flip-flop MWN (memory was negative) is set. Flip-flop CC2 is reset; the P-register is cleared by signal PX; and interruptibility is started by setting flip-flop IEN (interrupt enable).

During PH2, the contents of the A-register (most significant word of the numerator) and the B-register (least significant woid of the numerator) are shifted one bit position to the left by signals AXSL1 and BXBL1, respectively. The sign bit of the numerator is still retained in flip-flop RN. B0 is clocked into A31. Flip-flop B0 is not set. The numerator magnitude is held in A0 through A31 (numerator bits 2^{62} through 2^{31}, respectively) and B0 through B31 (numerator bits 2^{30} through 2^0, respectively). The numerator is scaled in this fashion so that the first quotient bit produced is the 2^{31} bit (the numerator 2^{31} bit is in the same relative position as the denominator 2^0 bit).

At the PH2 clock, the first transfer of the denominator from the C- to the D-register occurs at the PH2 clock. The denominator in the D-register must be of the opposite sign to the numerator to accomplish the first nonrestoring division iteration. Comparison of the numerator and the denominator signs is made by comparing flip-flops RN (numerator sign) and MWN (denominator sign). If only one of the flip-flops contains a one, the numerator and denominator signs are unlike, and the contents of register C are clocked into register D by signal DXC/6. If both the flip-flops contain ones or if both contain zeros, the signs of the numerator and denominator are the same. The contents of C are inverted and clocked into the D-register by signal DXNC/1, and a one is forced into CS31 in the case of like signs. The inverted denominator in the D-register and the one in CS31 make up the two's complement of the denominator.

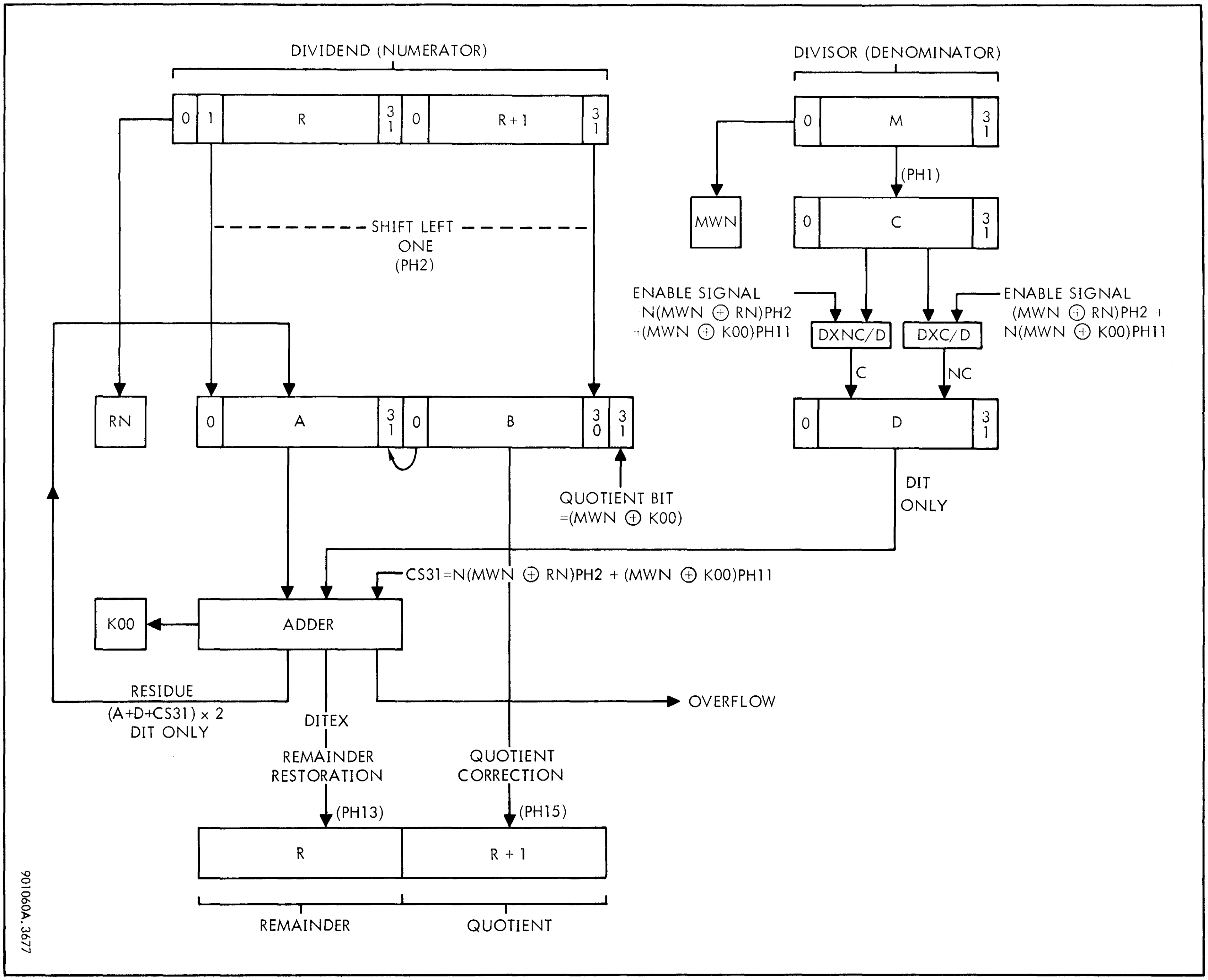


Figure 3-176. Divide Word Register Organization and Flow Chart

Table 3-74. Divide Word, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	Same as general preparation sequence except that a one is forced onto LR31 address line and PRE2 NIA is as follows	S/LR31/2	= (PRE1 NIA + PRE3 IX + PRE4) FAFRR/1 + . . .	Selects odd numbered private memory register
PRE2 NIA	RR0-RR31 —/→ A0-A31	AXRR	= FAMDSF PRE2 NIA + . . .	LSW of numerator (2^{31} - 2^0 bits) —/→ A
PH1 T4L	A0-A31 —/→ S0-S31	SXA	= FADIV PH1 + . . .	LSW of numerator —/→ B
	S0-S31 —/→ B0-B31	BXS	= FAMDSF PH1 + . . .	
	Force a one into flip-flop BWZ if the magnitude of the number in the A-register was zero	S/BWZ	= FADIV PH1 A0131Z + . . .	
	RR0-RR31 —/→ A0-A31	AXRR	= FADIVW PH1 + . . .	MSW of numerator (2^{63} - 2^{32} bits) —/→ A
	Set flip-flop RN if numerator negative	S/RN	= RR0 FADIVW PH1 + . . .	Stores numerator sign
	MB0-MB31 ——→ C0-C31	Preparation control		Denominator —/→ C
	Set flip-flop MWN if denominator negative	S/MWN	= C0C16 FAMDSF PH1 + . . .	Stores denominator sign
	Reset flip-flop CC2	R/CC2	= FAMDSF NFAMULH PH1 + . . .	
	Clear P-register	PX	= FADIV PH1 + . . .	
	Set flip-flop IEN to start interruptibility	S/IEN	= FAMDSF NFAMUL PH1 + . . .	
PH2	A0-A31 ——→ PR0-PR31	SXPR	= FADIVW PH2 + . . .	(D, CS = 0)
	S1-S31 —/→ A0-A31	AXSL1	= FADIVW PH2 + . . .	Align numerator so that 2^{31} bit is over denominator 2^0 bit. First iteration yields quotient bit 2^{31}
	B0 —/→ A31	BXBL1	= FADIV PH2 + . . .	
	B1-B31 —/→ B0-B30	BXBL1	= FADIV PH2 + . . .	
	(0 —/→ B31)			
	NC0-NC31 —/→ D0-D31	DXNC/1	= FADIV PH2 N(MWN ⊕ RN) + . . .	First transfer of denominator to D made so that D holds opposite polarity to the numerator

Mnemonic: DW (36, B6)

(Continued)

Table 3-74. Divide Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 (Cont.)	Force a one into CS31 or C0-C31 ⇸ D0-D31	S/CS31 = DXNC/1 + . . . or DXC/6 = FADIV PH2 (MWN ⊕ RN)	For two's complement
	Plus - count iteration counter	PCTP1 = FADIV PH2 + . . .	
	Set flip-flop PH11	BRPH11 = FADIV PH2 + . . .	
PH11 T6L	32 clocks if overflow does not occur, 3 clocks if overflow does occur. During the first 30 clocks (no overflow), the following events occur:		
	Enable signal DIT	DIT = FAMDSF/D PH11 + . . .	
	A + D + CS31 → S0-S31	SXADD = DIT + . . .	Most significant part of residue
	S1-S31 ⇸ A0-A31	AXSL1 = DIT N(FADIV P26) + . . .	Residue $^{\times}2$ → A, B
	B0 ⇸ A31	S/A31 = A31EN/2 AXSL1 A31EN/2 = B0 FAMDSF	
	B1-B31 ⇸ B0-B31	BXBL1 = DIT + . . .	
	Force a one into B31	S/B31 = (MWN ⊕ K00) FADIV PH11	First quotient bit
	NC0-NC31 ⇸ D0-D31	DXNC/D = FADIV PH11 (MWN ⊕ K00) + . . .	Polarity of denominator clocked into D and CS31 must be opposite to the residue clocked into A, B
	Force a one into CS31	S/CS31 = DXNC/D + . . .	
	C0-C31 ⇸ D0-D31	DXC/D = FADIV PH11 N(MWN ⊕ K00)	
	Set flip-flop SW3 if zero residue numerator is negative and zero residue is reached	S/SW3 = FADIV PH11 RN A0031Z + . . .	
	Count iterations, using P26-P31	PCTP1 = DIT NDITEX + . . .	
	Sustain PH11 until DITEX = 1	BRPH11 = DIT NDITEX + . . .	
	<u>On the 31st clock</u>:		
	Other events listed above		
	Enable DITEX	DITEX = FADIV P26 + CC2 + . . .	Initiate end of iterations

Mnemonic: DW (36, B6)

(Continued)

Table 3-74. Divide Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH11 T6L (Cont.)	Enable MRQ/1	MRQ/1 = DIT DITEX + . . .	
	Clear P-register	PX = MRQ/1 + . . .	
	Q15-Q31 $\not\rightarrow$ P15-P31	PXQ = MRQ/1 + . . .	
	On the 32nd clock:		
	Other events listed above except that residue is not shifted into A, B		
	S0-S31 $\not\rightarrow$ A0-A31	AXS = FADIV PH11 P26 + . . .	Residue to A for remainder restoration
	Reset flip-flop IEN	R/IEN = DIT DITEX + . . .	
	Set flip-flop PH13	BRPH13 = FADIV PH11 P26 + . . .	
	Enable clock T10L	S/T10L = FADIV PH11 P26 FADIVW + . . .	
	Overflow Logic		
	Set flip-flop CC2 if $\lvert N\rvert \div \lvert D\rvert \leq 2^{31}$	S/CC2 = FADIV PH11 P2629 P30 NP31 DIVOVER + . . .	Probe DIVOVER after the first iteration
		DIVOVER = RN ND0 + NRN D0 + D0 A0031Z BWZ	Negative numerator, negative residue. Positive numerator, positive residue. Residue = 0
	Enable DITEX if CC2 (3rd clock). Perform DITEX functions and inhibit NDITEX functions Also:	DITEX = CC2 + . . .	
	Set flip-flop DRQ	S/DRQ = FADIV PH11 CC2 + . . .	ENDE follows
	Set flip-flop TRAP	S/TRAP = FADIV PH11 CC2 AM + . . .	
	Set flip-flop TR30	S/TR30 = FADIV PH11 CC2 AM + . . .	Trap to location 67 if AM is true
	Set flip-flop TR31	S/TR31 = FADIV PH11 CC2 AM + . . .	
	Set flip-flop PH12 if P26 = 0	S/PH12 = PH11 NBR + . . .	
PH12 T6L	Entered only if overflow detected in PH11		
	Generate signal ENDE	ENDE = FADIV PH12 + . . .	

Mnemonic: DW (36, B6)

(Continued)

Table 3-74. Divide Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T10L	Remainder restoration phase. Four cases are possible		
	Case 1: Quotient and residue have like signs, zero residue was never achieved		Residue is true remainder
	A0-A31 $\rightarrow$ S0-S31	SXA = FADIV PH13 NSW3 (RN $\oplus$ D0) + . . .	
	Case 2: Quotient and residue have like signs, zero residue was achieved		Remainder must be equal to zero
	S0-S31 = 0		
	Case 3: Quotient and residue have unlike signs, residue = 0		Residue is true remainder
	S0-S31 = 0		
	Case 4: Quotient and residue have unlike signs, residue ≠ 0		
	A + D + CS31 $\rightarrow$ S0-S31	SXADD = FADIV PH13 N(RN $\oplus$ D0) NA0031Z + . . .	Residue converted to remainder with same sign as quotient
	S0-S31 $\nrightarrow$ A0-A31	AXS = FADIV PH13 + . . .	For quotient adjustment in PH14
	S0-S31 $\rightarrow$ RW0-RW31	RW = FADIVW PH13 + . . .	Remainder $\nrightarrow$ R
PH14 T6L	Quotient adjustment setup phase		
	B0-B31 $\rightarrow$ S0-S31	SXB = FADIV PH14 + . . .	Transfer the unadjusted quotient to the A-register
	S0-S31 $\nrightarrow$ A0-A31	AXS = FADIV PH14 + . . .	
	Clear D-register	DX/1 = FADIV PH14 + . . .	
	Force a one into CS31 in four cases		
	Case 1: ($\overset{+}{=}$)	S/CS31 = FADIV PH14 NRN MWN + . . .	Completes two's complement
	Case 2: ($\underset{+}{=}$) , remainder ≠ 0	S/CS31 = FADIV PH14 RN NMWN NA0031Z + . . .	Completes two's complement
	Case 3: ($\equiv$) , remainder = 0	S/CS31 = FADIV PH14 RN MWN A0031Z + . . .	Completes two's complement

Mnemonic: DW (36, B6)

(Continued)

Table 3-74. Divide Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH14 T6L (Cont.)	Set flip-flop DRQ	S/DRQ = FAMDSF PH14 + . . .	
	Force a one onto LR31 address line	S/LR31/2 = FADIVW PH14 + . . .	Selects odd numbered private memory register for PH15
	Enable clock T10L	S/T10L = FAMDSF PH14 + . . .	
PH15	A + CS31 → S0-S31	SXADD = FADIV PH15 + . . .	Adjusted quotient
	S0-S31 → RW0-RW31	RW = FAMDSF PH15 + . . .	Store quotient
	S0-S31 ↛ A0-A31	AXS = FADIV PH15 + . . .	
	Set flip-flop TESTA	S/TESTA = FAMDSF NFASHFX PH15 + . . .	
	Set CC3, CC4 as applicable	S/CC3 = TESTA NA0 NA0031Z + . . .	
		S/CC4 = TESTA A0 + . . .	
	Generate signal ENDE	ENDE = FAMDSF PH15 + . . .	

Mnemonic: DW (36, B6)

Other operations occurring at the PH2 clock include enabling signal PCTP1 (plus count P-register) for PH11 iteration counting and enabling PH11.

PH11 is the iteration phase of instruction DW. There is one iteration in PH11 for each clock, making a total of 32 iterations. Signal DIT (divide iterations) is set at the beginning of the phase and remains true as long as the iterations are to continue. Signal DITEX (divide iterations exit) goes true at the 32nd clock if there is no fixed-point overflow, or at the third clock if there is fixed-point overflow, and signals the end of the divide iterations.

During the first clock of PH11, the first addition of the division process occurs. The contents of the A-register (bits 2^{62} through 2^{31} of the numerator) are summed with the contents of the D-register (true or inverted form of the denominator) and CS31 (contains a one if denominator was inverted in PH2). The sum is the first residue (partial dividend). At the first clock of PH11, the following division operations occur.

a. The residue is shifted one place to the left by signal AXSL1 and is transferred back into the A-register.

b. The contents of B0 are transferred to A31.

c. The B-register is shifted one bit position to the left by signal BXBL1.

d. The first quotient bit is put into B31. The bit is a one if K00 is true and the denominator is positive or if K00 is false and the denominator is negative (see table 3-72).

e. The denominator or the inverted denominator is transferred from the C-register to the D-register. The transfer is made so that the contents of the D-register will be of opposite polarity to the residue in the A-register and will enable the next addition in the nonrestoring process to occur. The true denominator will be transferred by signal DXC/D if the residue is negative and the true denominator is positive, or if the residue is positive and the true denominator is negative. The inverted denominator will be transferred by signal DXNC/D if both the residue and true denominator have like signs. A one is clocked into CS31 in the latter case to make up the two's complement of the denominator when the next addition takes place.

f. Iteration counting begins. Flip-flops P26 through P31 make up a six-bit binary iteration up-counter for instruction DW and DH. The count is increased by one (in PH2) so that PH13 will be enabled on clock 32 of PH11 rather than on clock 33 (P26 sets flip-flop PH13). Figure 3-177 shows the counting configurations and the signals enabled for instruction DW.

When the counter indicates 31 iterations, signal DITEX goes true, enabling signal MRQ/1 to request the next instruction. At the next clock (32nd), the P-register is cleared (signal PX). Signal BRPH11 goes false as DITEX goes true (31st clock). At clock 32, PH13 and clock T10L are enabled, and flip-flop IEN is reset.

During successive clocks of PH11, another addition takes place, and the operations in steps a through f, with the exception of residue transfer on the last iteration and overflow condition, are repeated.

On the last iteration (32nd), signal AXSL1 goes false, and signal AXS is enabled. The residue, represented by sum bus outputs S0 through S31, is not shifted before transfer to the A-register; it is transferred directly to the A-register. The residue remains intact in this case to accomplish PH13 remainder restoration.

Fixed-point overflow is detected, if present at the second clock of PH11. Fixed-point overflow occurs if the quotient cannot be contained in a 32-bit register. (See figure 3-178) If the magnitude of the quotient is greater than or equal to 2^{31}, at least one 2^{31} multiple of the denominator is contained in the numerator. The quotient will be at least 33 bits long (sign bit plus 32 magnitude bits). If the magnitude of the quotient is less than 2^{31}, the 2^{31} multiple of the denominator extends over the zero-point line to make the 2^{31} magnitude bit of the quotient equal to zero. Extension to the opposite side is also true in the case of a negative numerator. After the first iteration, therefore, a residue sign opposite to the numerator sign signifies that there is no overflow and that the next denominator multiple to be added must be of the same sign as the numerator. Unlike denominator multiple and numerator signs after the first iteration indicate that the zero-point was not crossed. Divide overflow is therefore defined by flip-flop RN condition and the contents of D0. The last DIVOVER = RN ND0 + NRN D0 + D0 A0031 BWZ term defines the case of a zero residue after the first iteration (figure 3-178, second example).

Condition code 2 is set at the second clock of PH11 if signal DIVOVER is true. CC2 enables DITEX, which enables the other control functions (MRQ/1, PX, NBRPH11, R/IEN). Flip-flops PH13 and T10L are not set. CC2 also causes a trap to memory location X'43' (67) if the fixed-point arithmetic mask (AM) is a one. PH12 is entered if overflow is detected, and signal ENDE is enabled.

Flip-flop SW3 is set if the division involves a negative numerator, and a zero residue is reached after any iteration. If a zero residue is reached in this case, succeeding residues are negative and approach, but do not attain, a zero residue (see figures 3-173 and 3-174). SW3 indicates that the correct remainder of all zeros is to be restored after the division process is completed.

PH13 is the remainder restoration phase of the instruction. The last residue produced in the iteration process is not necessarily the true remainder of the division. During this phase, the true remainder is restored from the last residue, and the sign of the remainder is adjusted to agree with the sign of the quotient. There are four remainder cases at the start of the phase.

a. If the last residue is the same sign as the quotient, and flip-flop SW3 is not set, a zero residue was never present during the iterations, and the last residue is the true remainder. The last residue in the A-register is gated to the sum bus by signal SXA.

b. If the last residue is the same sign as the quotient, and flip-flop SW3 is set, the remainder should be all zeros (see figures 3-173 and 3-174). Nothing is gated to the sum bus. The sum bus outputs, S0 through S31, represent a zero remainder.

c. If the last residue is of an opposite sign to the quotient, and if the residue is equal to zero, the residue is the true remainder. Nothing is gated to the sum bus. The sum bus outputs represent a remainder of zero.

d. If the last residue is of an opposite sign to the quotient, and the residue is not equal to zero, it must be converted to a remainder that has the same sign as the quotient. This is done by restoring the next to the last residue, which has the same sign as the quotient. The last bit of the quotient, in this case, must have the absolute value of zero, so that restoring the next to last residue does not alter the quotient. Any further division of the next to the last residue by the same denominator would result in exactly the same quotient as division of the last residue. The last residue in the A-register is added to the denominator multiple in the D-register (already of opposite sign to the last residue) and CS31 (containing a zero or one, depending on the previous C to D transfer). The true remainder result is gated to the sum bus by signal SXADD. The sum bus outputs, S0 through S31, represent the 32-bit remainder. The remainder is clocked into the even numbered private memory register for storage and into the A-register for PH14 testing.

PH14 is the quotient adjustment setup phase of instruction DW. As discussed in paragraph 3-202, certain division cases other than the plus/plus case produce a quantity that is one less than the correct quotient. These cases are detected in this phase, and the one is added in PH15. The B-register, containing the unadjusted quotient, is gated to the sum bus by signal SXB. At the PH14 clock, the sum bus outputs are clocked into the A-register by signal AXS. At the same time, a one is clocked into CS31 if the plus/minus, minus/plus with nonzero residue, or minus/minus with zero residue case is being performed. The D-register is cleared by signal DX/1 at the PH14 clock. Flip-flop DRQ is set, inhibiting transmission of another clock until the data signal is received from memory. A one is forced on private memory address line LR31 to select the odd numbered register. T10L is enabled for PH15.

Plus/minus case   Minus/plus, remainder nonzero case

S/CS31 = FADIV PH14 NRN MWN + RN NMWN
NA0031Z + . . .

Minus/minus, remainder zero case

FADIV PH14 RN MWN A0031Z + . . .

CLOCK (PH11)	P26	P27	P28	P29	P30	P31	
1					1	0	
2					1	1	
3				1	0	0	
4				1	0	1	
5				1	1	0	
6				1	1	1	
31	1	0	0	0	0	0	DITEX, MRQ/1
32	NOT APPLICABLE						R/IEN, S/PH13, S/T10L

901060A.3678

Figure 3-177. Divide Word and Divide Halfword Iteration Counting

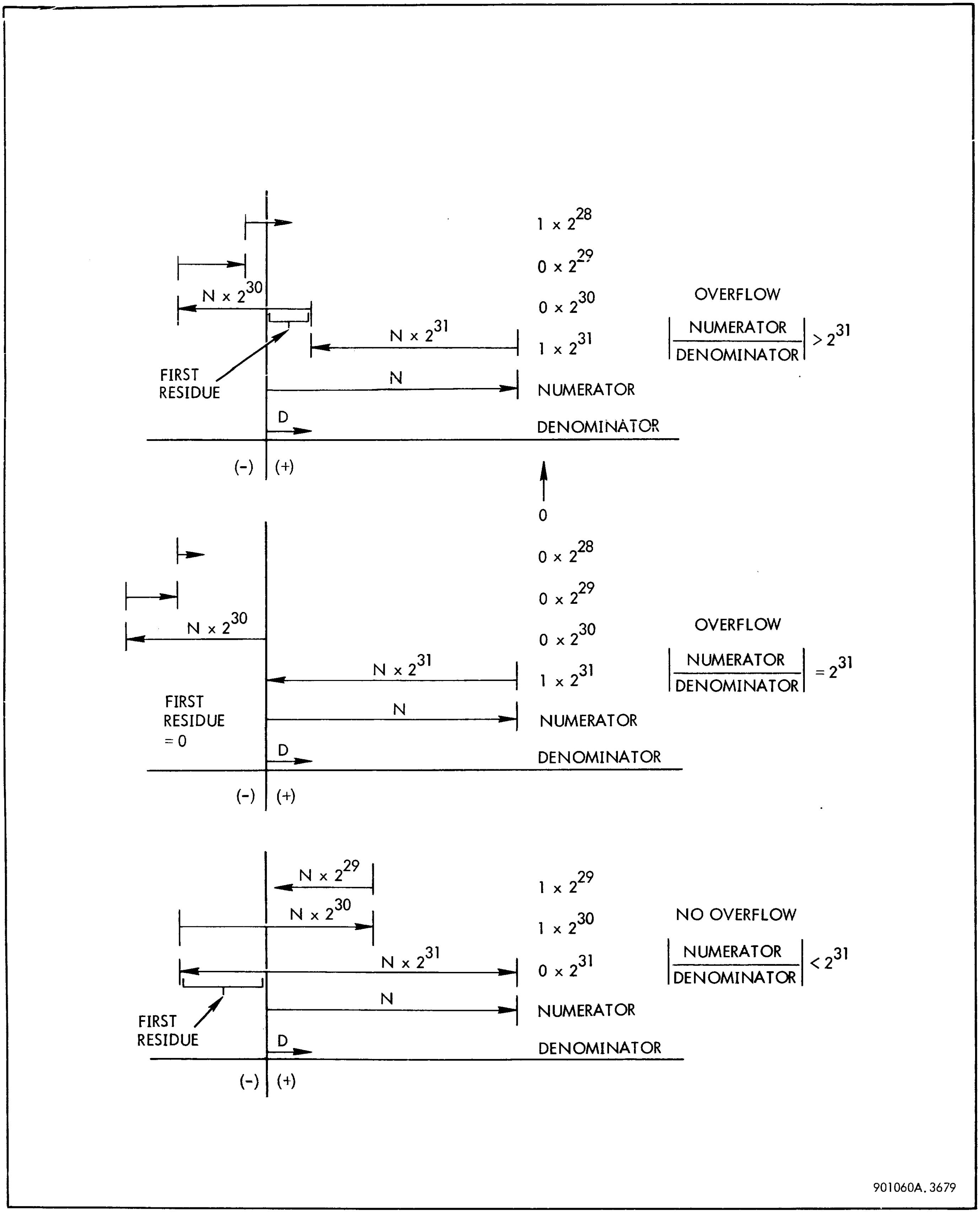


Figure 3-178. Overflow Detection

During PH15, the uncorrected quotient in the A-register is added to CS31 to produce the correct quotient. The sum is gated to the sum bus by signal SXADD and into the odd numbered private memory register by signal RW. Signal AXS also clocks the sum into the A-register at this time. Flip-flop TESTA is set. If A0 is zero and A0 through A31 is nonzero, flip-flop CC3 is set to denote a positive quotient. If A0 is a one, flip-flop CC4 is set, to denote a negative quotient. Signal ENDE is generated.

3-204 DESCRIPTION OF DIVIDE HALFWORD INSTRUCTION. Figure 3-179 shows the register arrangement during instruction Divide Halfword (DH), FADIVH. The 32-bit numerator from the even or odd numbered private memory register is initially contained in flip-flop RN (numerator sign) and bit positions 2 through 30 of the B-register. The A-register bit positions contain sign padding. The C-register holds the halfword denominator from core memory in bit positions 16 through 31. Bit positions 0 through 15 contain sign padding for the denominator. Flip-flop MWN contains the denominator sign. The nonrestoring division process is the same as instruction DW. After the 32nd iteration, the B-register holds the 32-bit quotient, and the A-register holds the last residue. The quotient is corrected if necessary and the true remainder is restored from the residue. The quotient is clocked into the even or odd numbered private memory register. The remainder is discarded. Implementation of instruction DW with an odd value R-field is identical to implementation of instruction DH.

The following paragraphs describe the phases of instruction DH. Figure 3-179 and the sequence chart in table 3-75 supplement the discussion. During PRE2, the numerator in the even or odd numbered private memory register is clocked into the A-register. Flip-flop RN is set if the numerator is negative. The remainder of the preparation sequence is the same as the general preparation sequence.

During PH1, the A-register contents are gated to the sum bus by signal SXA. The sum bus outputs are clocked into the B-register by signal BXS at the PH1 clock. Flip-flop BWZ is set if bit positions 1 through 31 of the A-register contain zeros. The word containing the halfword denominator from core memory has been read into the C-register at this time. If the effective halfword is negative, bit 0 or bit 16 of the memory word (zero if P32 is false, 16 if P32 is true) enables signal C0C16, which sets flip-flop MWN. At the PH1 clock, the halfword denominator in the C-register is downward aligned into the D-register. The halfword denominator is clocked into bit positions 16 through 31 of the D-register. If C0C16 is true, ones are clocked into bit positions 0 through 15 of the D-register to sign-pad the denominator.

Other functions that occur at the PH1 clock include resetting flip-flop CC2, clearing the P-register (signal PX), starting interruptibility by setting flip-flop IEN, and clearing the A-register (signal AX/1). Flip-flop CXS is set only if instruction DH is being implemented (not instruction DW, R31).

The downward-aligned, sign-padded denominator in the D-register is gated to the sum bus by signal SXD, during PH2. The sum bus outputs are clocked into the C-register by CXS at this clock. (If instruction DW, R31 is being performed, these two operations are not done, since the C-register already contains the 32-bit denominator.) The remainder of PH2 is identical to instruction DW, except for sign-padding of the numerator. If the numerator is negative, ones are put into bit positions 0 through 31 of the CS-register. When the A-, D-, and CS-registers are added on the first iteration of PH11, the ones in CS0 through CS31 effectively sign-pad the numerator.

The remainder of the operations in PH2 is identical to instruction DH. In summation:

a. The first denominator multiple is transferred from the C-register to the D-register so that the D-register contains a quantity of an opposite sign to the numerator (NC is transferred to D by signal DXNC/1; C is transferred to D by signal DXC/6). If the C contents are inverted and are transferred to the D-register, a one is forced into CS31 if the numerator is positive, or into A31 if the numerator is negative (a one is already in CS31 in this case). The inverted contents of the C-register and the one make up the two's complement of the denominator. For every other C to D transfer (PH11), a one is put into CS31.

b. The B-register is shifted one bit position to the left so that the numerator 2^{31} bit is in the same relative position as the denominator 2^0 bit. The first quotient bit, therefore, will be the 2^{31} bit. A zero is clocked into B31.

c. PCTP1 for PH11 iteration counting is enabled. The PH11 flip-flop is set.

PH11 is identical to instruction DW. The following operations occur for each iteration:

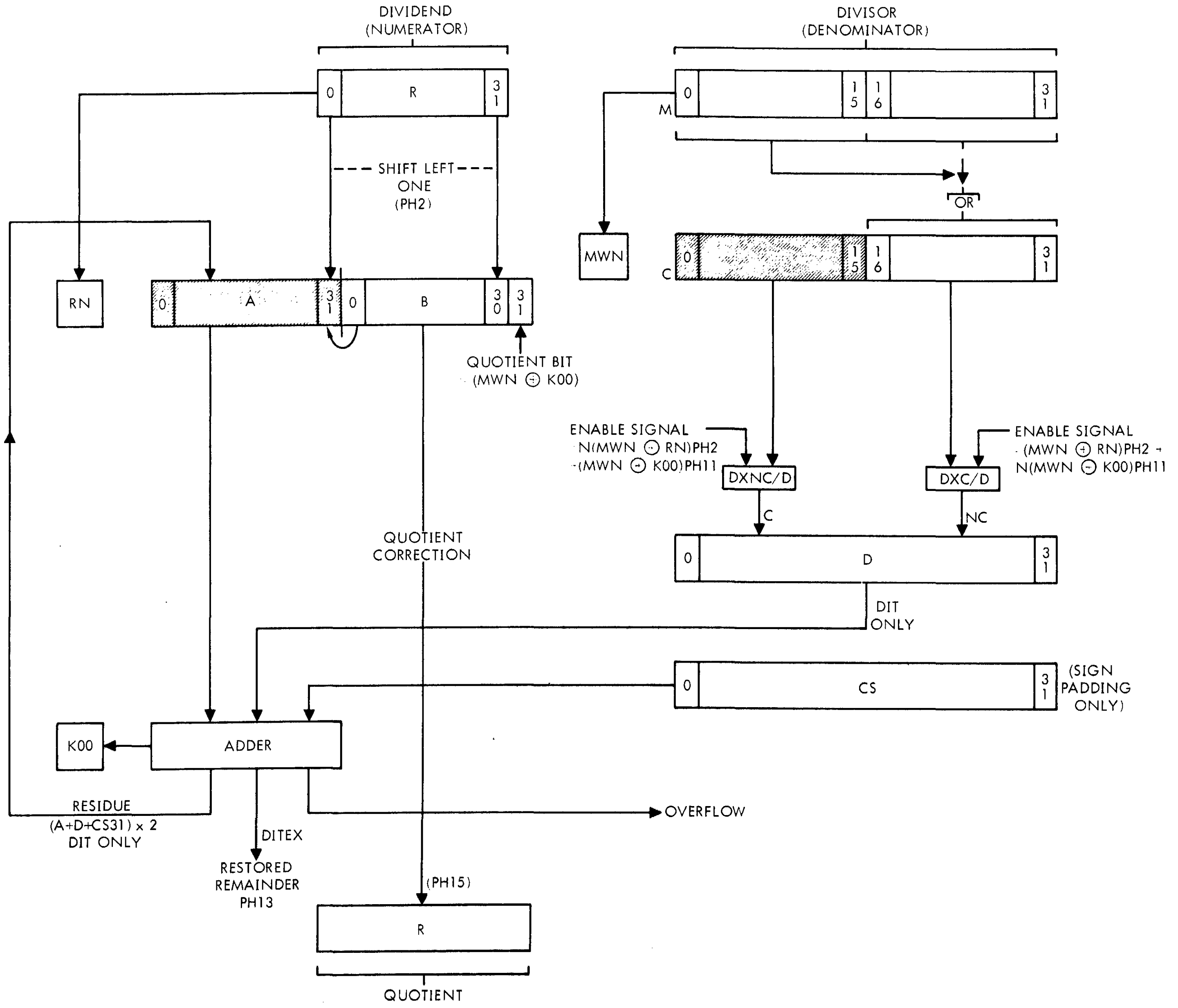
a. Signal DIT goes true, or remains true.

b. Addition of the A-, D-, and CS-registers occurs (after the first iteration, addition is of the A- and D-registers and CS31). The residue is shifted one bit position to the left by signal AXSL1 and is transferred back to the A-register.

c. The B-register is simultaneously shifted one bit position to the left by signal BXBL1. B0 is clocked into A31.

d. The first quotient bit is put into B31.

e. The denominator or inverted denominator is transferred from the C-register to the D-register so that the D-register contains a quantity of opposite polarity to the residue. A one is clocked into CS31 if the inverted denominator is transferred.



901060A. 3680

Figure 3-179. Divide Halfword Register Organization and Flow Chart

Table 3-75. Divide Halfword, Divide Word (R = 31), Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	Same as general preparation sequence except PRE2 NIA			
PRE2 NIA	RR0-RR31 -/→ A0-A31	AXRR	= FAMDSF PRE2 NIA + . . .	Numerator -/→ A
	Set flip-flop RN if numerator negative	S/RN	= RR0 RNXRR0	Stores numerator sign
		RNXRR0	= FADIVH PRE2 NIA	
PH1 T4L	A0-A31 ⟶ S0-S31	SXA	= FADIV PH1 + . . .	Numerator -/→ B
	S0-S31 -/→ B0-B31	BXS	= FAMDSF PH1 + . . .	
	Force a one into flip-flop BWZ if the magnitude of the number in the A-register was zero	S/BWZ	= FADIV PH1 A0131Z + . . .	
	MB0-MB31 ⟶ C0-C31	Preparation control		Word containing halfword denominator -/→ C
	C0-C15 or C16-C31 -/→ D16-D31	DXC/5	= OU5 (NO4 O5 O6) PH1 + . . .	Denominator downward aligned
	Set flip-flop MWN if denominator negative	S/MWN	= C0C16 FAMDSF PH1 + . . .	Stores denominator sign
		C0C16	= (RR0 NP32 + RR16 P32)	
	Reset flip-flop CC2	R/CC2	= FAMDSF NFAMULH PH1 + . . .	
	Clear P-register	PX	= FADIV PH1 + . . .	
	Set flip-flop IEN to start interruptibility	S/IEN	= FAMDSF NFAMUL PH1 + . . .	
	Set flip-flop CXS if DH (do not set in DW R31 case)	S/CXS	= FADIVH PH1 OU5 + . . .	
PH2	D0-D31 ⟶ S0-S31	SXD	= FADIVH PH2 + . . .	C already contains 32-bit denominator DW R31 case
	if DH			
	S1-S31 ⟶ C0-C31	CXS	= Previously set in PH1	
	B0 -/→ A31	BXBL1	= FADIV PH2 + . . .	Align numerator so that 2^{31} bit is over denominator 2^0 bit. First iteration yields quotient bit 2^{31}

Mnemonic: DH (56, D6)

(Continued)

Table 3-75. Divide Halfword, Divide Word (R ≠ 31), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 (Cont.)	B1-B31 ⇸ B0-B31 (0 ⇸ B31)	BXBL1 = FADIV PH2 + . . .	
	NC0-NC31 ⇸ D0-D31	DXNC/1 = FADIV PH2 N(MWN ⊕ RN) + . . .	First transfer of denominator to D made so that D holds opposite polarity to the numerator
	Force a one into CS31	S/CS31 = DXNC/1 + . . .	
	or	or	
	C0-C31 ⇸ D0-D31	DXC/6 = FADIV PH2 (MWN ⊕ RN) + . . .	
	1 ⇸ A31 if numerator negative	A31X1 = FADIVH PH2 MWN RN + . . .	
	Force ones into CS0-CS31 if numerator negative	CSX1 = FADIVH PH2 RN + . . .	To sign-pad negative numerator
	Pre-count iteration counter	PCTP1 = FADIV PH2 + . . .	
	Set flip-flop PH11	BRPH11 = FADIV PH2 + . . .	
PH11 T6L	32 clocks if overflow does not occur, 3 clocks if overflow does occur. During the first 30 clocks (no overflow), the following events occur:		
	Enable signal DIT	DIT = FAMDSF/D PH11 + . . .	
	A + D + CS31 → S0-S31	SXADD = DIT + . . .	Most significant part of residue
	S1-S31 ⇸ A0-A31	AXSL1 = DIT N(FADIV P26) + . . .	
	B0 ⇸ A31	AXSL1 = DIT N(FADIV P26) + . . .	Residue ×2 → A, B
	B1-B31 ⇸ B0-B30	BXBL1 = DIT + . . .	
	Force a 1 into B31		
	NC0-NC31 ⇸ D0-D31	DXNC/D = FADIV PH11 (MWN ⊕ K00) + . . .	Polarity of denominator clocked into D and CS31 must be opposite to the residue clocked into A, B
	Force a one into CS31		
	C0-C31 ⇸ D0-D31	DXC/D = FADIV PH11 N(MWN ⊕ K00) + . . .	

Mnemonic: DH (56, D6)

(Continued)

Table 3-75. Divide Halfword, Divide Word (R = 31), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH11 T6L (Cont.)	Set flip-flop SW3 if zero residue numerator is negative and zero residue is reached	S/SW3	= FADIV PH11 RN A0031Z + . . .	
	Count iterations, using P26-P31	PCTP1	= DIT NDITEX + . . .	
	Sustain PH11	BRPH11	= DIT NDITEX + . . .	
	<u>On the 31st clock</u>:			
	Other events listed above			
	Enable DITEX	DITEX	= FADIV P26 + CC2 + . . .	Initiate end of iterations
	Stop sustaining PH11	BRPH11	= DIT NDITEX + . . .	
	Enable MRQ/1	MRQ/1	= DIT DITEX + . . .	
	Clear P-register	PX	= MRQ/1 + . . .	
	Q15-Q31 —/→ P15-P31	PXQ	= MRQ/1 + . . .	
	<u>On the 32nd clock</u>:			
	Other events listed above except that residue is not shifted into A, B			
	S0-S31 —/→ A0-A31	AXS	= FADIV PH11 P26 + . . .	Residue to A for remainder restoration
	Reset flip-flop IEN	R/IEN	= DIT DITEX + . . .	
	Set flip-flop PH13	BRPH13	= FADIV PH11 P26 + . . .	
	Overflow Logic:			
	Set flip-flop CC2 if $\lvert N \rvert \div \lvert D \rvert \geq (\frac{N}{0} \text{ OR } \frac{-2^{31}}{-2^{0}})$	S/CC2	= FADIV PH11 P2629 P30 NP31 DIVOVER + . . .	Probe DIVOVER after the first iteration
		DIVOVER	= RN ND0 + NRN D0 + D0 A0031Z BWZ	Negative numerator, negative residue. Positive numerator, positive residue. Residue = 0
	Enable DITEX if CC2 (3rd clock) Perform DITEX functions and inhibit NDITEX functions Also:	DITEX	= CC2 + . . .	
	Set flip-flop DRQ	S/DRQ	= FADIV PH11 CC2 + . . .	ENDE follows

Mnemonic: DH (56, D6)

(Continued)

Table 3-75. Divide Halfword, Divide Word (R = 31), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH11 T6L (Cont.)	Set flip-flop TRAP	S/TRAP = FADIV PH11 CC2 AM + . . .	
	Set flip-flop TR30	S/TR30 = FADIV PH11 CC2 AM + . . .	Trap to location 67 if AM is true
	Set flip-flop TR31	S/TR31 = FADIV PH11 CC2 AM + . . .	
PH12 T6L	Entered only if overflow detected in PH11		
	Generate signal ENDE	ENDE = FADIV PH12 + . . .	
PH13 T6L	Remainder restoration phase. Four cases are possible.		
	Case 1: Quotient and residue have like signs, zero residue was never achieved		
	A0-A31 → S0-S31	SXA = FADIV PH13 NSW3 (RN ⊕ D0) + . . .	Residue is true remainder
	Case 2: Quotient and residue have like signs, zero residue was achieved		Remainder must be equal to zero
	S0-S31 = 0		
	Case 3: Quotient and residue have unlike signs, residue = 0		Residue is true remainder
	S0-S31 = 0		
	Case 4: Quotient and residue have unlike signs, residue ≠ 0		
	A + D + CS31 → S0-S31	SXADD = FADIV PH31 N(RN ⊕ D0) NA0013Z + . . .	Residue converted to remainder with some sign as quotient
	S0-S31 ↛ A0-A31	AXS = FADIV PH13 + . . .	For quotient adjustment in PH14. Remainder later discarded
PH14 T6L	Quotient adjustment setup phase		
	B0-B31 → S0-S31	SXB = FADIV PH14 + . . .	Transfer the unadjusted quotient to the A-register
	S0-S31 ↛ A0-A31	AXS = FADIV PH14 + . . .	

Mnemonic: DH (56, D6)

(Continued)

Table 3-75. Divide Halfword, Divide Word (R = 31), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH14 T6L (Cont.)	Clear D-register	DX/1 = FADIV PH14 + . . .	
	Force a one into CS31 in three cases:		
	Case 1: ($\frac{+}{-}$)	S/CS31 = FADIV PH14 NRN MWN + . . .	Completes two's complement
	Case 2: ($\frac{-}{+}$) , remainder ≠ 0	S/CS31 = FADIV PH14 RN NMWN NA0031Z + . . .	Completes two's complement
	Case 3: ($\frac{-}{-}$) , remainder = 0	S/CS31 = FADIV PH14 RN MWN A0031Z + . . .	Completes two's complement
	Set flip-flop DRQ	S/DRQ = FAMDSF PH14 + . . .	
	Enable clock T10L	S/T10L = FAMDSF PH14 + . . .	
PH15	A + CS31 → S0-S31	SXADD = FADIV PH15 + . . .	Adjusted quotient
	S0-S31 → RW0-RW31	RW = FAMDSF PH15 + . . .	Store quotient
	S0-S31 $\nrightarrow$ A0-A31	AXS = FADIV PH15 + . . .	
	Set flip-flop TESTA	S/TESTA = FAMDSF NFASHFX PH15 + . . .	
	Set CC3, CC4 as applicable	S/CC3 = TESTA NA0 NA0031Z + . . .	
		S/CC4 = TESTA A0 + . . .	
	Generate signal ENDE	ENDE = FAMDSF PH15 + . . .	

Mnemonic: DH (56, D6)

f. Iteration counting begins. Thirty-two iterations are counted, as described in the section on instruction DW, beginning in paragraph 3-203.

g. SW3 is set if the residue goes to zero after any of the 32 iterations.

h. Overflow is detected after the first iteration. Overflow can occur only if the denominator is zero or if the numerator is -2^{31} and the denominator is -2^{0}. Overflow causes a trap to memory location X'43' (67). Signal ENDE is generated in PH12.

PH13 is identical to instruction DW except that clock T6L is used instead of T10L and that signal RW is not enabled (the remainder is not stored in private memory). The remainder is restored from the last residue as before and is gated to the sum bus and into the A-register (for PH14 use).

PH14 is the quotient adjustment setup phase, as in instruction DW. The 32-bit quotient is gated from the B-register to the sum bus and is clocked into the A-register. If the quotient needs correction, a one is forced into CS31. Flip-flops DRQ and T10L are set as before. A one is not forced on private memory address line LR31/2 in this instruction.

PH15 completes the halfword division. The A-register, containing the unadjusted quotient, is added to CS31. The result is gated to the sum bus and is clocked into the even or odd numbered private memory register for storage. The condition codes are set as in instruction DW.

A sequence chart for the Divide Halfword Instruction is given in table 3-75.

3-205 Add Word to Memory (AWM 66, E6)

The AWM instruction causes a data word to be read from an effective core memory location. The contents of a private memory register addressed in the R-field of the instruction word is added to the data word, and the result is stored back in the effective core memory location. If fixed-point overflow occurs and if the fixed-point arithmetic trap mask (bit 11 of PSD) is a one, the computer traps to location X'43' (67) after loading the sum into the private memory registers. The data word from core memory is loaded into the D-register, and the data word from the selected private memory register is loaded into the A-register. The outputs of the A- and D-registers are gated into the adder, and the sum is written into the effective core memory location. The sum is also clocked into the A-register to be tested for condition code information.

If fixed-point overflow occurs, and the fixed-point arithmetic trap mask bit in the PSD is set, the program traps to location X'43' after the sum is stored in core memory. A sequence chart of the Add Word to Memory instruction is given in table 3-76.

3-206 Modify and Test Byte (MTB 73, F3)

The MTB instruction causes a data word to be read from the effective core memory address and loads the selected byte into the D-register. The desired word is determined by the byte address which is stored in the private memory register that is addressed in the X-field of the instruction word. If the R-field of the instruction word is nonzero, the high order bit (bit position 8 of the R-field) is extended to the left to form a byte. This byte is added to the byte contained in the D-register, and the sum is stored back in the effective byte location. The condition code is set according to the value of the resultant byte. The three bits of the R-field plus a sign bit allow modification of a byte by any number from -8 through +7.

If the R-field of the instruction word is zero, the effective byte is tested for a zero or nonzero value. The result sets the condition code, but the effective byte is not affected.

During the preparation sequence, the index value is shifted two bit positions to the right into the A-register, and the byte selection bits in the two least significant bit positions of the index value are loaded into flip-flops P32 and P33 for byte addressing. The index value is added to the effective address in the D-register, and the operand is read from core memory into the C-register. In the first execution phase, the selected byte is loaded from the C-register into bit positions 24 through 31 of the D-register. The value in the R-field of the instruction word is clocked into flip-flops A28 through A31, and flip-flops CS0 through CS27 are set or are cleared according to the state of R28. The remaining flip-flops in the A- and CS-registers are cleared. The contents of the A-, D-, and CS-registers are gated into the adder, and the sum is placed in the A-register. The relevant portion of the sum, in bit positions 24 through 31 of the A-register, is upward aligned into all byte positions of the sum bus, and the sum byte, with byte addressing using the outputs of P32 and P33, is written into the selected byte location in the memory word. Flip-flop CC1 is set if an end carry occurs from the byte addition.

Another execution phase is entered, if the instruction is part of an interrupt subroutine. The modified byte in the A-register is checked for a zero or nonzero condition. If the modified byte is all zeros, a counter-equals-zero interrupt is generated, and the appropriate interrupt subroutine is performed. If the modified byte is not all zeros, a priority interrupt exit signal is generated, and flip-flop INTRAPF is reset. The address of the next instruction, which had been stored in the B-register during the interrupt sequence, is gated onto the sum bus and is clocked into the P-register. This address selects the next instruction. A sequence chart for the Modify and Test Byte instruction is given in table 3-77.

3-207 Modify and Test Halfword (MTH 53, D3)

The MTH instruction causes a data word to be read from the effective core memory address and loads the selected halfword into the D-register. The desired halfword is determined by the halfword address stored in the private memory

Table 3-76. Add Word to Memory, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PH1	C0-C31 —/→ D0-D31	DXC	= FAS8 PH1 + . . .	Data word from core memory
	RR0-RR31 —/→ A0-A31	AXRR	= FUAWM PH1 + . . .	Data word from private memory
	Generate memory request to write into core memory	MRQ	= FAS8 PH1 NRZ + . . .	To write result of addition into core memory
	Set flip-flop DRQ	S/DRQ	= FAS8 PH1 + . . .	Inhibits transmission of another clock until signal received data lines have been strobed
	Set flip-flop PH2	S/PH2	= NBR PH1 N(FNANLZ NANLZ) + . . .	
	Enable clock T10L	S/T10L	= FAS24 PH1 + . . .	
PH2	A0-A31 + D0-D31 + CS0-CS31 ——→ S0-S31	SXADD	= FAS24 PH2 + . . .	Addition of data word from core memory and data word from private memory. CS = 0
	S0-S31 ——→ MB0-MB31	MB0-MB31	= S0-S31 MBXS	
	Generate memory write signal MW	MW	= FAS8 PH2 + . . .	
	Generate write byte signals MBXS/0-MBXS/3	MBXS/0-MBXS/3	= MW	Sum stored in effective core memory location
	S0-S31 —/→ A0-A31	AXS	= FAS24 PH2 + . . .	To be tested for condition code information
	If an end carry occurs, set flip-flop CC1; otherwise, reset it	S/CC1	= PH2 NINTRAPF FAS24 K00 + . . .	
		R/CC1	= PH2 NINTRAPF FAS24 + . . .	
	If fixed-point overflow occurs, set flip-flop CC2; otherwise, reset it	S/CC2	= PH2 NINTRAPF FAS8 OVER + . . .	
		OVER	= NA0 ND0 K0 + D0 NK0	
		R/CC2	= PH2 NINTRAPF FAS24 + . . .	
	Generate memory request for next instruction	MRQ/1	= NINTRAPF PH2 FAS8 + . . .	
	Q15-Q31 —/→ P15-P31	PXQ	= MRQ/1 + . . .	Address of next instruction

Mnemonic: AWM (66, E6)

(Continued)

Table 3-76. Add Word to Memory, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 (Cont.)	P15-P31 → LM15-LB31	(NLMXC NLMXQ)	
	Set flip-flop DRQ	S/DRQ = FAS24 PH2 + . . .	Inhibits transmission of another clock until data signal received
	Set flip-flop PH4	S/PH4 = BRPH4 = FAS8 PH2 NKSC + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH4 T6L	Set flip-flop TESTA	S/TESTA = FAS24 PH4 NINTRAPF + . . .	
	Generate signal ENDE	ENDE = FAS24 PH4 NINTRAPF + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If data word in A-register is all zeros, reset flip-flops CC3 and CC4; if word is negative (A0), set flip-flop CC4 and reset CC3; if word is positive (NA0), set flip-flop CC3 and reset flip-flop CC4	S/CC3 = NTESTA/1 TESTA NA0 NA0031Z + . . .	
		R/CC3 = TESTA + . . .	
		S/CC4 = NTESTA/1 TESTA A0 + . . .	
		R/CC4 = TESTA + . . .	
	If fixed-point overflow occurs and fixed-point arithmetic trap mask is a one, trap to location X'43' (67)	S/TRAP = TROVER + . . .	
		TROVER = FAS24 PH4 NINTRAPF AM CC2 + . . .	
		S/AM = PSW1XS S11	

Mnemonic: AWM (66, E6)

Table 3-77. Modify and Test Byte, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1 T6L	D12-D14 → LR29-LR31	LRXD = ENDE	
	Generate INDX signal	INDX = (C12 + C13 + C14) (C3 + C4 + C5)	
	P15-P31 -/→ Q15-Q31	QXP = PRE1 NANLZ + . . .	
	RR13-RR29 -/→ A15-A31	AXRRR2 = OU7 INDX PRE1 + . . .	Index data from private memory index register
	Reset A0-A14	AX = AXRRR2	
	RR30 -/→ P32	S/P32 = RR30 AXRRR2	Byte address
	RR31 -/→ P33	S/P33 = RR31 AXRRR2	Byte address selection
	Set flip-flop IX	S/IX = INDX PRE1	
	Set flip-flop PRE2	S/PRE2 = NPREIM PRE1 N(S/INTRAPF)	
	Enable T4RL	T4RL = PREP + . . .	
PRE2 T4RL	A0-A31 + D0-D31 + CS0-CS31 → S0-S31	SXPR = SXADD = PRE2 NIA + . . .	
	S15-S31 -/→ P15-P31	PXS = PRE2 NIA + . . .	Modified program address
	P15-P31 → LM15-LB31	(NLMXC NLMXQ) = PRE2 NIA NRIP	
	Generate memory request for operand	MRQ = PRE2 NIA IX OPRQ NANLZ	
		OPRQ = PRE1 TP140 PREOPRQ/2 latched	
		PREOPRQ/2 = C3 NC4 NC5 + . . .	
	Set flip-flop ARQ	S/ARQ = PRE2 NIA IX + . . .	Inhibits transmission of another clock until data release signal received
	Reset flip-flop IX	R/IX = PRE2 NIA + . . .	
	Set flip-flop PH1	S/PH1 = NPREDO PRE2 NIA + . . .	
	Enable clock T4RL	T4RL = PREP + . . .	

Mnemonic: MTB (73, F3)

(Continued)

Table 3-77. Modify and Test Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PRE2 T4RL (Cont.)	If bits P32 and P33 are both zeros, clock bits C0-C7 into flip-flops D24 through D31; if bit P32 is a zero and P33 is a one, clock bits C8 through C15 into flip-flops D24 through D31; if bit P32 is a one and bit P33 is a zero, clock bits C16 through C23 into flip-flops D24 through D31. If bits P32 and P33 are both ones; clock bits C24 through C31 into flip-flops D24 through D31	DXCR24 = DXCBP NP32 NP33	Selected byte down aligned into D-register
		DXCR16/13 = DXCBP NP32 NP33 + . . .	
		DXCR8 = DXCBP P32 NP33 + . . .	
		DXC/13 = DXCBP P32 P33 + . . .	
		DXCBP = OU7 (NO4 NO5) PH1 + . . .	
	R28-R31 $\not\longrightarrow$ A28-A31	AXR = FAS7 PH1 + . . .	R-field clocked into A-register
	R28 $\not\longrightarrow$ CS0-CS27	S/CS0-/S/CS27 = FAS7 PH1 R28 + . . .	Sign bit extended
	Set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ + NANLZ) + . . .	
	Enable clock T10L	S/T10L = FAS24 PH1 + . . .	
PH2 T10L	A0-A31 + D0-D31 + CS0-CS31 $\longrightarrow$ S0-S31	SXPR = SXADD = FAS24 PH2 + . . .	Add core memory byte to extended high order bit of R-field
		SXK = SXADD = FAS24 PH2 + . . .	To test for nonzero
	S0-S31 $\not\longrightarrow$ A0-A31	AXS = FAS24 PH2 + . . .	
	If end carry occurs from bit 0, set flip-flop CC1	S/CC1 = PH2 NINTRAPF FAS24 K00 + . . .	
	If end carry occurs from byte, set flip-flop MWN	S/MWN = FUMTB PH2 K23 + . . .	
	Reset flip-flop D0-D31	DX = FAS13 PH2 + . . .	
	Set flip-flop NPRX	S/NPRX = FAS13 PH2 + . . .	For byte upward alignment in PH3
	Generate a memory request to write byte	MRQ = FAS13 PH2 NRZ + . . .	
	Set flip-flop DRQ	S/DRQ = PH2 FAS24 + . . .	Inhibits transmission of another clock until signal received from memory
	Force ones into flip-flops CS0-CS31	S/CS0-S/CS31 = N(S/NPRX) + . . .	

Mnemonic: MTB (73, F3)

(Continued)

Table 3-77. Modify and Test Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH2 T10L (Cont.)	Set flip-flop PH3	S/PH3	= PH2 NBR + . . .	
	Enable clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
PH3 T6L	A24-A31 + CS24-CS31 → K23-K30			
	K23-K30 —/→ S0-S7, S8-S15, S16-S23	SXUAB	= FUMTB PH3 + . . .	Upward align byte into all bytes of sum bus
	A24-A30 —/→ S24-S30	SXA	= SXUAB + . . .	
	If NP32 NP33, generate signal MBXS/0;	MBXS/0	= NP32 NP33 MWB + . . .	
	if NP32 P33, generate signal MBXS/1;	MBXS/1	= NP32 NP33 MWB + . . .	
	if P32 NP33, generate signal MBXS/2;	MBXS/2	= P32 NP33 MWB + . . .	
	if P32 P33, generate signal MBXS/3	MBXS/3	= P32 P33 MWB + . . .	
	Generate signal MWB	MWB	= FUMTB PH3 + . . .	
	Gate selected byte onto MB lines from sum bus	MB0-MB7	= MBXS/0 S0-S7 + . . .	
		MB8-MB15	= MBXS/1 S8-S15 + . . .	Store modified byte in core memory location
		MB16-MB23	= MBXS/2 S16-S23 + . . .	
		MB24-MB31	= MBXS/3 S24-S31 + . . .	
	S8-S31 —/→ A8-A31, zeros —/→ A0-A7	AXS	= FUMTB PH3 + . . .	Resultant byte —/→ bytes 1, 2, and 3 of A-register
		AX	= AXS	
	If an end carry occurs, set flip-flop CC1	S/CC1	= PH3 NINTRAPF FUMTB MWN + . . .	If not part of an interrupt routine
	Generate memory request for next instruction if not part of interrupt sequence	MRQ/1	= PH3 NINTRAPF FAS13 + . . .	INTRAPF ⇒ interrupt sequence
	Q15-Q31 —/→ P15-P31	PXQ	= MRQ/1 + . . .	Address of next instructio
	P15-P31 → LM15-LB31			

Mnemonic: MTB (73, F3)

(Continued)

Table 3-77. Modify and Test Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (Cont.)	Set flip-flop DRQ if not part of interrupt sequence	S/DRQ = PH3 NINTRAPF FAS24 + . . .	Inhibits transmission of another clock until data signal received if not part of interrupt sequence
	Set flip-flop PH4	S/PH4 = PH3 NBR + . . .	
	Inhibit single-clocking	SCEN = FAS7 PH3 INTRAPF	Inhibits single-clock control
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH4 T6L NIN-TRAPF	If not part of interrupt subroutine		
	Generate signal ENDE	ENDE = FAS24 PH4 NINTRAPF + . . .	
	Set flip-flop TESTA	S/TESTA = FAS24 PH4 NINTRAPF + . . .	
	If data word in A-register is positive (NA0) and the rest of the word is not all zeros, set flip-flop CC3 and reset flip-flop CC4; if data word is negative (A0), flip-flop CC4 is set and flip-flop CC3 is reset	S/CC3 = NTESTA/1 TESTA NA0 NA0031Z + . . . R/CC3 = TESTA + . . . S/CC4 = NTESTA/1 TESTA A0 + . . . R/CC4 = TESTA + . . .	
	Generate memory request for next instruction if part of interrupt sequence	MRQ = FAS7 PH4 INTRAPF	
	Set flip-flop DRQ	S/DRQ = FAS7 PH4 INTRAPF	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	Go to preparation phases of next instruction
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH4 IN-TRAPF	If part of interrupt subroutine		
	Generate counter-equals-zero interrupt if modified byte = 0	CNTZREQ = FAS7 PH4 INTRAPF A0031Z	A00-31Z ⇒ contents of A-register = 0. Go to appropriate subroutine
	Generate interrupt exit signal LEVACT if modified byte ≠ 0	LEVACT = FAS7 PH4 INTRAPF + . . .	Interrupt exit signal

Mnemonic: MTB (73, F3)

(Continued)

Table 3-77. Modify and Test Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 IN-TRAPF	Generate interrupt arming signal LEVARM	LEVARM = FAS7 PH4 INTRAPF + . . .	
	B15-B31 ⟶ S15-S31 ↛ P15-P31	SXB = FAS7 PH4 + . . .	
		PXS = FAS7 PH4 INTRAPF + . . .	
	Generate memory request for next instruction	MRQ = FAS7 PH4 INTRAPF + . . .	
	Set flip-flop DRQ	S/DRQ = FAS7 PH4 INTRAPF + . . .	Inhibits transmission of another clock until data release signal received
	Reset flip-flop INTRAPF	R/INTRAPF = FAS7 PH4 INTRAPF + . . .	
	Set flip-flop PH5	S/PH5 = PH4 NBR + . . .	
PH5	Generate signal ENDE	ENDE = FAS24 PH5 + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	Go to preparation phases of next instruction
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MTB (73, F3)

register and addressed in the X-field of the instruction word. If the R-field of the instruction word is nonzero, the high order bit (bit position 8 of the R-field) is extended to the left to form a halfword. This halfword is added to the halfword contained in the D-register, and the sum is stored in the effective halfword location. The condition code register is set according to the value of the resultant halfword. The three bits of the R-field plus a sign bit allows modification of a halfword by any number from -8 to +7.

If the R-field of the instruction word is zero, the effective halfword is tested for being a zero or nonzero value. The condition code is set as a result of this test, but the effective halfword is not affected.

During the preparation sequence, the index value is shifted one bit position to the right into the A-register, and the halfword selection bit in the least significant bit position of the index value is loaded into flip-flop P32 for halfword addressing. The index value is added to the effective address in the D-register, and the operand is read from core memory into the C-register. When the data is on the memory bus, the selected halfword is loaded into flip-flops D15 through D31. The value in the R-field of the instruction word is clocked into flip-flops A28 through A31, and flip-flops CS0 through CS27 are set or are cleared according to the state of flip-flop R28. The remaining flip-flops in the A- and CS-registers are cleared. The contents of the A-, D-, and CS-registers are gated into the adder, and the sum is placed in the A-register. Flip-flop CC1 is set if an end carry occurs in this operation. If fixed-point overflow occurs, flip-flop CC2 is set. The relevant portion of the sum, in bit positions 15 through 31 of the A-register, is upward aligned into both halfword positions of the sum bus, and with halfword addressing, using the output of flip-flop P32, the sum byte is written into the selected byte location in the memory word.

If the MTH instruction is not being performed in an interrupt sequence, execution ends and the program goes to the preparation sequence for the next instruction. Condition code flip-flops are set. If overflow has occurred and if the fixed-point arithmetic trap mask bit in the PSD is set, the program traps to location X'43' after the result is stored in the effective halfword location.

If the instruction is part of an interrupt subroutine, another execution phase is entered. The modified halfword in the A-register is checked for a zero or nonzero condition. If the modified halfword is all zeros, a counter-equals-zero interrupt is generated, and the appropriate interrupt subroutine is performed. If the modified halfword is not all zeros, a priority interrupt exit signal is generated, and flip-flop INTRAPF is reset. The address of the next instruction, which had been stored in the B-register during the interrupt sequence, is gated onto the sum bus and is clocked into the P-register. This address selects the next instruction. A sequence chart of the Modify and Test Halfword is given in table 3-78.

3-208 Modify and Test Word (MTW 33, B3)

The MTW instruction causes a data word to be read from the effective core memory address. If the R-field of the instruction word is nonzero, the high order bit (bit position 8 of the R-field) is extended 28 bit positions to the left to form a 32-bit word. This word is added to the data word from core memory, and the sum is stored back in the effective core memory location. The three bits of the R-field plus a sign bit allow modification of a word by any number from -8 to +7.

If the R-field of the instruction word is zero, the effective word is tested for being zero, negative, or positive. As a result, the condition code is set but the effective word is not affected.

The core memory operand is loaded into the D-register, and the value in the R-field of the instruction word is clocked into flip-flops A28 through A31. Flip-flops CS0 through CS27 are set or are cleared according to the state of flip-flop R28. This extends the R-field 28 bit positions to the left as the contents of the A-, D-, and CS-registers are gated into the adder. The sum is clocked into the A-register for testing and onto the core memory data lines for writing into memory. If an end carry occurs, flip-flop CC1 is set; if overflow occurs, flip-flop CC2 is set.

If the MTW instruction is not being performed in an interrupt sequence, execution ends and the program goes to the preparation sequence for the next instruction. The program traps to location X'43' after the result is stored in the effective location, if overflow has occurred and if the fixed-point arithmetic trap mask bit in the PSD is set.

If the MTW instruction is part of an interrupt subroutine, a check of the A-register is made for a zero or nonzero condition. If the modified word is all zeros, a counter-equals-zero interrupt is generated, and the program branches to the appropriate subroutine. If the modified word is not all zeros, a priority interrupt exit signal is generated, and flip-flop INTRAPF is reset. A sequence chart of the Modify and Test Word instruction is given in table 3-79.

3-209 Compare Immediate (CI 21)

The Compare Immediate instruction extends the sign of the value field (bit 12) of the instruction word 12 bit positions to the left. It also compares the 32-bit result with the contents of register R and sets the condition code according to the results of the comparison.

The CI instruction sequences through the preparation phases in which the contents of the selected private memory register are transferred to the A-register. The CS-register is set to ones, and the D-register is cleared to zeros. The one's complement of the private memory word is taken by performing an exclusive-OR operation on the word in the A-register and on the ones in the CS-register.

Table 3-78. Modify and Test Halfword, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PRE1 T6L	D12-D14 → LR29-LR31	LRXD	= ENDE	Selects index register containing halfword address
	Generate signal INDX	INDX	= (C12 + C13 + C14) (C3 + C4 + C5)	
	P15-P31 -/→ Q15-Q31	QXP	= PRE1 NANLZ + . . .	Next instruction address
	RR14-RR30 -/→ A15-A31	AXRRR1	= OU5 INDX PRE1 + . . .	Index data shifted right one bit position
	RR31 -/→ P32	S/P32	= RR31 AXRRR1 + . . .	Halfword select bit
	Set flip-flop IX	S/IX	= INDX PRE1	
	Set flip-flop PRE2	S/PRE2	= NPREIM PRE1 N(S/INTRAPF)	
	Enable T4RL	T4RL	= PREP + . . .	
PRE2 T4RL	A15-A31 + D15-D31 + CS15-CS31 → S15-S31	SXPR	= SXADD = PRE2 NIA + . . .	Add indexing data and operand address
	S15-S31 -/→ P15-P31	PXS	= PRE2 NIA + . . .	
	P15-P31 → LM15-LB31	(NLMXC NLMXQ)	= PRE2 NIA NRIP + . . .	Address → memory address lines
	Generate memory request	MRQ	= PRE2 NIA IX 0PRQ NANLZ + . . .	Memory request for operand
	Set flip-flop ARQ	S/ARQ	= PRE2 NIA IX + . . .	Inhibits transmission of another clock until address release signal received
	Reset flip-flop IX	R/IX	= PRE2 NIA	
	Set flip-flop PH1	S/PH1	= NPREDO PRE2 NIA + . . .	
	Enable clock T4RL	T4RL	= PREP + . . .	
PH1 T4RL	If P32 is a zero, clock bit MB0 into flip-flops D0 through D15	C0C16C12	= MB0 NP32 CXMB + . . .	Sign bit extended
		C0C16/1	= MB0 NP32 CXMB + . . .	
	If P32 is a one, clock bit MB16 into flip-flops D0 through D15	C0C16C12	= MB16 P32 CXMB + . . .	Sign bit extended
		C0C16/1	= MB16 P32 CXMB + . . .	

Mnemonic: MTH (53, D3)

(Continued)

Table 3-78. Modify and Test Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL (Cont.)	If P32 is a zero, clock C0-C15 into D16-D31	DXCR16 = DXC/5 NP32 + . . . DXC/5 = OU5 (NO4 NO5) PH1	Halfword -/→ D-register
	If P32 is a one, clock C16-C31 into D16-D31	DXC = DXC/5 P32 + . . .	
	R28-R31 -/→ A28-A31	AXR = FAS7 PH1 + . . .	R-field of instruction word clocked into A-register
	R28 -/→ CS0-CS27	S/CS0-S/CS27 = FAS7 PH1 R28 + . . .	Sign bit extended into CS-register
	Set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
	Enable clock T10L	S/T10L = FAS24 PH1 + . . .	

PH2
T10L
	D0-D31 + A0-A31 + CS0-CS31 → S0-S31	SXPR = SXADD = FAS24 PH2 + . . .
		SXK = SXADD = FAS24 PH2 + . . .
			Halfword modified by bits in R-field

	S0-S31 -/→ A0-A31	AXS = FAS24 PH2 + . . .	
	If an end carry occurs, generate signal K00 and set flip-flop CC1	S/CC1 = PH2 NINTRAPF FAS24 K00 + . . .	
	If bit S15 is not the same as bit S16, set flip-flop CC2	S/CC2 = PH2 NINTRAPF FUMTH (S15 + S16) + . . .	
	Clear D-register	DX = FAS13 PH2 + . . .	
	Set flip-flop NPRX	S/NPRX = FAS13 PH2 + . . .	For upward alignment of A-register
	Force ones into flip-flops CS0-CS31	CSX1 = N(S/NPRX) + . . .	
	Generate a memory request to write result	MRQ = FAS13 PH2 NRZ + . . .	
	Set flip-flop DRQ	S/DRQ = FAS24 PH2 + . . .	Inhibits transmission of another clock until signal received that data lines have been strobed
	Set flip-flop PH3	S/PH3 = PH2 NBR + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MTH (53, D3)

(Continued)

Table 3-78. Modify and Test Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L	A16-A31 + CS16-CS31 ⟶ K15-K30		
	Generate signal SXUAH	SXUAH = FUMTH PH3 + . . .	Upward align halfword
	K15-K30 ⟶ S0-S15		
	A16-A31 ⟶ S16-S31	SXA = SXUAH + . . .	Sum bus contains two halfwords of the same data
	If flip-flop P32 is a zero, generate signals MBXS/0 and MBXS/1	MBXS/0-MBXS/1 = NP32 MWH + . . .	Determine which halfword to write into core memory
	If flip-flop P32 is a one, generate signals MBXS/2 and MBXS/3	MBXS/2-MBXS/3 = P32 MWH N(TRAP NINTRAPF) + . . .	
	Generate signal MWH	MWH = FUMTH PH3 + . . .	Store modified halfword in core memory location

	Gate selected halfword onto MB lines from sum bus	MB0-MB15 = MBXS/0, MBXS/1 + . . .
		MB16-MB31 = MBXS/2, MBXS/3 + . . .	

	S0-S31 ↛ A0-A31	AXS = FUMTH PH3 + . . .	To set flip-flops CC3 and CC4
	Generate memory request for next instruction if not part of interrupt sequence	MRQ/1 = PH3 NINTRAPF FAS13 + . . .	INTRAPF ⇒ interrupt sequence
	Q15-Q31 ↛ P15-P31	PXQ = MRQ/1 + . . .	Next instruction address
	Set flip-flop DRQ if not part of interrupt sequence	S/DRQ = PH3 NINTRAPF FAS24 + . . .	Inhibits transmission of another clock until data signal received
	If part of interrupt sequence, reset flip-flop MAPDIS	R/MAPDIS = FAS13 INTRAPF PH3 + . . .	Reinstates map function
	Set flip-flop PH4	S/PH4 = PH3 NBR + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Inhibit single-clock	SCEN = PH3 FAS7 INTRAPF	

Mnemonic: MTH (53, D3)

(Continued)

Table 3-78. Modify and Test Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L INTRAPF	If part of interrupt subroutine		
	If modified halfword in A-register is all zeros, generate counter-equals-zero interrupt signal CNTZREQ	CNTZREQ = FAS7 PH4 INTRAPF A0031Z	Counter-equals-zero interrupt A0031Z ⟹ contents of A-register = 0. Go to appropriate subroutine
	If modified halfword in A-register is not all zeros, generate interrupt exit signal LEVACT	LEVACT = FAS7 PH4 INTRAPF + . . .	
	Generate interrupt arming signal LEVARM	LEVARM = FAS7 PH4 INTRAPF + . . .	
	Gate address of next instruction from B-register onto sum bus	SXB = FAS7 PH4 + . . .	Address stored in B-register during interrupt subroutine
	Clock address from sum bus into P-register	PXS = FAS7 PH4 INTRAPF + . . .	
	Generate memory request for instruction if part of interrupt sequence	MRQ = FAS7 PH4 INTRAPF + . . .	
	Set flip-flop DRQ	S/DRQ = FAS7 PH4 INTRAPF + . . .	Inhibits transmission of another clock until data signal received
	Set flip-flop PH5	S/PH5 = PH4 NBR + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Reset flip-flop INTRAPF	R/INTRAPF = FAS7 PH4 INTRAPF + . . .	
PH4 T6L NINTRAPF	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	Go to PRE1 of next instruction if not interrupt sequence
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Generate signal ENDE	ENDE = FAS24 PH4 NINTRAPF + . . .	If instruction is not in an interrupt sequence
	Set flip-flop TESTA	S/TESTA = FAS24 PH4 NINTRAPF + . . .	To test condition code flip-flops

Mnemonic: MTH (53, D3)

(Continued)

Table 3-78. Modify and Test Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L NIN-TRAPF (Cont.)	If data word in A-register is positive (NA0) and the rest of the word is not all zeros, set flip-flop CC3 and reset flip-flop CC4; if the data word is negative (A0), flip-flop CC4 is set and flip-flop CC3 is reset	S/CC3 = NTESTA/1 TESTA NA0 NA0031Z + . . .	
		R/CC3 = TESTA + . . .	
		S/CC4 = NTESTA/1 TESTA A0 + . . .	
		R/CC4 = TESTA + . . .	
	If flip-flop CC2 is set and fixed-point arithmetic trap mask (bit 11 of PSD) is a one, trap to location X'43' (67)	S/TRAP = TROVER + . . .	CC2 ⇒ overflow
		TROVER = FAS24 PH4 NINTRAPF AM CC2 + . . .	
		S/TR30 = TROVER N(S/TRACC4/1) + . . .	Trap address flip-flops
		S/TR31 = TROVER N(S/TRACC4/1) NSTRAP NTRAP + . . .	
PH5 T6L	Generate signal ENDE	ENDE = FAS24 PH5 + . . .	Go to preparation phases of next instruction
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MTH (53, D3)

Table 3-79. Modify and Test Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1	C0-C31 ⇸ D0-D31	DXC = FAS8 PH1 + . . .	
	Generate memory request to write data	MRQ = (FAS8 PH1 + . . .) NRZ + . . .	
	Set flip-flop DRQ	S/DRQ = FAS8 PH1 + . . .	Inhibits transmission of another clock until signal received that data lines have been strobed
	R28-R31 ⇸ A28-A31	AXR = FAS7 PH1 + . . .	R-field of instruction word clocked in A-register
	R28 ⇸ CS0-CS27	S/CS0-S/CS27 = FAS7 PH1 R28 + . . .	Sign bit from R-field extended into CS-register
	Set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
	Enable clock T10L	S/T10L = FAS24 PH1 + . . .	
PH2 T10L	D0-D31 + A0-A31 + CS0-CS31 → S0-S31	SXPR = SXADD = FAS24 PH2 + . . .	Contents of R-field and extended sign bit added to effective word
	S0-S31 ⇸ A0-A31	AXS = FAS24 PH2 + . . .	
	S0-S31 → MB0-MB31	MBXS = MW	
	Generate core memory write byte signals /MW0/ through /MW3/	/MW0/-/MW3/ = MBXS	Store modified word in core memory
	If instruction being performed as part of interrupt subroutine, reset flip-flop MAPDIS	R/MAPDIS = FUMTW PH2 INTRAPF + . . .	Reinstates map function
	If end carry occurred during addition, set flip-flop CC1	S/CC1 = PH2 NINTRAPF FAS24 K00 + . . .	
	If overflow occurred during addition, set flip-flop CC2	S/CC2 = PH2 NINTRAPF FAS8 OVER + . . .	
	Generate memory request for next instruction if not interrupt subroutine	MRQ/1 = PH2 NINTRAPF FAS8 + . . .	INTRAPF ⇒ interrupt subroutine
	Set flip-flop DRQ	S/DRQ = FAS24 PH2 + . . .	Inhibits transmission of another clock until data signal received

Mnemonic: MTW (33, B3)

(Continued)

Table 3-79. Modify and Test Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T10L (Cont.)	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	
	P15-P31 ——→ LM15-LB31	(NLMXC NLMXQ)	Address of next instruction
	Set flip-flop PH4	S/PH4 = BRPH4 = FAS8 PH2 NKSC + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH4 T6L IN-TRAPF	If not part of interrupt subroutine		
	Generate signal ENDE	ENDE = FAS24 PH4 NINTRAPF + . . .	
	Set flip-flop TESTA	S/TESTA = FAS24 PH4 NINTRAPF + . . .	To test for condition code settings
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If data word in A-register is positive (NA0), set flip-flop CC3 and reset CC4; if data word is negative (A0), set flip-flop CC4 and reset flip-flop CC3	S/CC3 = NTESTA/1 TESTA NA0 NA0031Z + . . .	
		R/CC3 = TESTA + . . .	
		S/CC4 = NTESTA/1 TESTA A0 + . . .	
		R/CC4 = TESTA + . . .	
	If part of interrupt subroutine		
	If overflow occurred (flip-flop CC2 set) and fixed-point arithmetic trap mask (bit 11 of PSD) is a one, trap to location X'43' (67)	S/TRAP = TROVER + . . .	
		S/TR30 = TROVER N(S/TRACC4/1) + . . .	
		S/TR31 = TROVER N(S/TRACC4/1) NSTRAP NTRAP + . . .	
		TROVER = FAS24 PH4 NINTRAPF AM CC2 + . . .	
		N(S/TRACC4/1) = TRAP + . . .	

Mnemonic: MTW (33, B3)

(Continued)

Table 3-79. Modify and Test Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L INTRAPF (Cont.)	If modified word in A-register is all zeros, generate count zero interrupt CNTZREQ	CNTZREQ = FAS7 PH4 INTRAPF A0031Z	
	If modified word in A-register is not all zeros, generate interrupt exit signal LEVACT	LEVACT = FAS7 PH4 INTRAPF + . . .	
	Generate interrupt arming signal LEVARM	LEVARM = FAS7 PH4 INTRAPF + . . .	
	Reset flip-flop INTRAPF	R/INTRAPF = FAS7 PH4 INTRAPF + . . .	
	B15-B31 → S15-S31	SXB = FAS7 PH4 + . . .	Address of next instruction stored in B-register during interrupt subroutine
	S15-S31 ↛ P15-P31	PXS = FAS7 PH4 INTRAPF + . . .	
	P15-P31 → LM15-LB31	(NLMXC NLMXQ)	
	Generate memory request for next instruction if part of interrupt sequence	MRQ = FAS7 PH4 INTRAPF	INTRAPF ⇒ interrupt sequence
	Set flip-flop DRQ	S/DRQ = FAS7 PH4 INTRAPF	Inhibits transmission of another clock until data signal received
	Set flip-flop PH5	S/PH5 = PH4 NBR + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH5 T6L	Generate signal ENDE	ENDE = FAS24 PH5 + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	To begin preparation for next instruction
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
			Mnemonic: MTW (33, B3)

This complement is transferred to the B-register. The immediate value in bits 12 through 31 of the C-register is transferred to the D-register, and the value of C12 is transferred to flip-flops D0 through D11 to extend the sign. The individual bits in the A- and D-registers are compared with an AND operation, and the result is placed in the A-register. If the A-register contains zeros, indicating that no bits of the private memory word and the sign-extended immediate value were coincident, flip-flop CC2 is reset. Otherwise, CC2 is set.

The contents of the B-register are transferred to the A-register, and a one is set into flip-flop CS31 to obtain the two's complement of the private memory word. The contents of the A- and D-registers are added, subtracting the private memory word from the immediate value. The sum is placed in the A-register.

An end carry out of bit position 0 is stored in flip-flop K00H. This flip-flop and the contents of the A-register are examined in the first preparation phase of the following instruction, or in the first phase of interrupt, trap, or control panel operation. Flip-flop CC4 is set if a nonzero value in the A-register and a one in K00H indicate that the private memory word is less than the immediate value. A zero in K00H and a nonzero value in the A-register cause flip-flop CC3 to be set. This indicates that the private memory word is greater than the immediate value. A sequence chart of the Compare Immediate instruction is given in table 3-80.

3-210 Compare Byte (CB 71, F1)

The CB instruction compares the contents of bit positions 24 through 31 of register R with the effective byte. Both bytes are treated as positive magnitudes. The condition code is set according to the results of the comparison.

Because the Compare Byte instruction addresses a byte, it becomes an indexed instruction if the X-field does not equal zero. Indexing is described under preparation phases. A memory request for the effective byte is made during indexing, and flip-flop RQ is set to request the next instruction as soon as the operand is received.

Signal AXRR2 is true in the preparation sequence, and causes a downward alignment of the index register two bit positions into the A-register. At this same clock, bits 30 and 31 of the index register are transferred to P32 and P33, respectively. The configuration placed into P32 and P33 determines which core memory byte is to be clocked into the D-register.

```
S/A15   = RR13 AXRRR2
  :         :
  :         :
S/A31   = RR29 AXRRR2
 AXRRR2 = OU7 INDX PRE1
S/P32   = RR30 AXRRR2
S/P33   = RR31 AXRRR2
```

Byte 3 (bits 24 through 31) of the addressed private memory register is gated to A24 through A31. Zeros are clocked into A0 through A23, the CS-register is filled with ones and the D-register is cleared to zeros.

In PH2 an AND operation is performed on the contents of the A- and D-registers (the original contents of the private memory register word and the effective operand). The result of this operation is taken from the sum bus and is placed into the A-register at the clock that ends PH2.

The selected byte from the core memory word is loaded into bits 24 through 31 of the D-register. The one's complement of the private memory byte is taken by performing an exclusive OR operation on the byte in bits 24 through 31 of the A-register and the ones in the CS-register. This complement is transferred to the B-register. The individual bits in the bytes which are in the least significant byte positions in the A- and D-registers are compared with an AND operation, and the result is placed in the A-register. If the A-register contains zeros, indicating that no bits of the private memory byte and the effective byte are coincident, flip-flop CC2 is reset. Otherwise, CC2 is set.

The contents of the B-register are transferred to the A-register, and a one is set into flip-flop CS31 to obtain the two's complement of the private memory byte. The contents of the A- and D-registers are added, subtracting the private memory byte from the effective byte, and the sum is placed in the A-register. A carry into S00 is stored in flip-flop K00H. This flip-flop and the contents of the A-register are examined in the first preparation phase of the following instruction, or in the first phase of interrupt, trap, or control panel operation, if one of these occurs next. Flip-flop CC4 is set if a nonzero value in the A-register and a one in K00H indicate that the private memory byte is less than the private memory word. A zero in K00H and a nonzero value in the A-register cause flip-flop CC3 to be set. This indicates that the private memory byte is greater than the effective byte. A sequence chart of the Compare Byte instruction is given in table 3-81.

3-211 Compare Halfword (CH 51, D1)

The Compare Halfword instruction extends the sign of the effective halfword 16 bit positions to the left and then compares the 32-bit result with the contents of the addressed private memory register. Both words are treated as signed magnitudes. The condition code is set according to the results of the comparison.

The instruction sequences through the preparation states. Because the instruction addresses a halfword, it effectively becomes an indexed instruction if the X-field does not equal zero. Indexing is described under the preparation sequence. During indexing, flip-flop MRQ is set to request the effective halfword, and flip-flop RQ is set to request the next instruction as soon as the operand is received.

Table 3-80. Compare Immediate, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	RR0-RR31 ⇸ A0-A31	AXRR = (PRE2 NIA) FAS11 NFUCB + . . .	Private memory word into A-register
	1's ⇸ CS0-CS31	CSX1 = (PRE2 NIA) FAS11 + . . .	Set ones into CS-register
	0's ⇸ D0-D31	DX/1 = PRE2 NIA + . . .	Reset D to zeros
	Enable T4RL	S/T4RL = PREP + . . .	
	Set PH1	S/PH1 = NPRED0 (PRE2 NIA) + . . .	
PH1	C12-C31 ⇸ D12-D31	DXC/4 = FUCI PH1 + . . .	Effective word into D-register and extend sign
	C12 ⇸ D0-D11	C0C16C12 = DXC/4 + . . .	
	A0-A31 ⊕ CS0-CS31 ⟶ S0-S31	SXPR = FAS11 PH1 + . . .	Exclusive OR action places one's complement of private memory word onto sum bus
T4RL	S0-S31 ⇸ B0-B31	BXS = FAS11 PH1 + . . .	One's complement of private memory word to B-register for adding later
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + . . .	Next instruction address into P-register
		MRQ/1 = FUCI PH1 + . . .	
	1's ⇸ CS0-CS31	CSX1 = S/NPRX + . . .	Prepare for AND action in PH2
	Set NPRX	S/NPRX = FAS11 PH1 + . . .	
	Enable T4L clock	S/T4L = FAS11 PH1 + . . .	
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ ANLZ) + . . .	
PH2	A0-A31 ∧ D0-D31 ⟶ S0-S31	SXPR = NPRX + . . .	Compare coincident 1-bits in private memory and core memory words
T4L	S0-S31 ⇸ A0-A31	AXS = FAS11 PH2 + . . .	Place results into A-register
	Enable T4L clock	S/T4L = FAS11 PH2 + . . .	
	Set PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	

Mnemonic: CI (21)

(Continued)

Table 3-80. Compare Immediate, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3	0 ⇸ CC2	R/CC2	= FAS11 PH3 + . . .	If no bits in private memory and core memory words are coincident, reset CC2; otherwise, set CC2
	If A0-A31 ≠ 0, 1 ⇸ CC2	S/CC2	= FAS11 PH3 NA0031Z + . . .	
T4L	B0-B31 → S0-S31	SXB	= FAS11 PH3 + . . .	Place one's complement of private memory word onto sum bus and clock into A-register
	S0-S31 ⇸ A0-A31	AXS	= FAS11 PH3 + . . .	
	1 ⇸ CS31	CSX1/8	= FAS11 PH3 + . . .	For two's complement
	0 ⇸ CS0-CS30	R/CS0-CS30	= .	
	Set DRQ	S/DRQ	= FAS11 PH3 + . . .	Inhibits another clock until data release received from memory
	Enable T6L clock	T6L	= NT1L NT4L NT8L NT10L NRESET	
	Set PH4	S/PH4	= FAS11 PH3 + . . .	
PH4	Generate ENDE	ENDE	= FAS11 PH4 + . . .	
T6L	A0-A31 + D0-D31 + CS1 → S0-S31	SXADD SXK	= FAS11 PH4 + . . . = SXADD	Subtract private memory word from core memory word
	S0-S31 ⇸ A0-A31	AXS	= FAS11 PH4 + . . .	Difference into A-register
	If D0-D31 ≤ A0-A31, 1 ⇸ K00H	S/K00H	= S00	

		S00	= K00 S00X + NK00 NS00X N(. . .)	S00X = 1 if A0 ≠ D0
				NS00XN = 1

		S00X	= FAS11 N[NS00XN (A0 . D0 + NA0 . ND0)]	
	Set TESTA	S/TESTA	= FAS11 PH4 + . . .	
	Set TESTA/1	S/TESTA/1	= S/TESTA NO4 NO6 O7 + . . .	
	Enable T6L clock	T6L	= NT1L NT4L NT8L NT10L NRESET	

Mnemonic: CI (21)

(Continued)

Table 3-80. Compare Immediate, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PRE1 Next in-struc-tion	Set flip-flop CC3 if K00H = 0	S/CC3 = TESTA/1 NK00H NA0031Z + . . . R/CC3 = TESTA	Private memory word > immediate value
	Set flip-flop CC4 = K00H = 1	S/CC4 = TESTA/1 K00H NA0031Z + . . . R/CC4 = TESTA	Private memory word > immediate value

Mnemonic: CI (21)

Table 3-81. Compare Byte, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	RR24-RR31 -/→ A24-A31	AXRR/3 = PRE2 NIA FUCB + . . .	Private memory byte 3 into A-register
	1's -/→ CS0-CS31	CSX1 = PRE2 NIA FAS11 + . . .	Set all ones into CS-register
	0's -/→ D0-D31	DX/1 = PRE2 NIA	Reset D to zeros
	Enable T4RL	S/T4RL = PREP + . . .	
	Set PH1	S/PH1 = NPREDO (PRE2 NIA) + . . .	
PH1 T4RL	P32 P33		
	MB0-MB31 ——→ C0-C31	CXMB = DGC	

	If 0 0, C0-C7 -/→ D24-D31
	If 0 1, C8-C15 -/→ D24-D31
	If 1 0, C16-C23 -/→ D24-D31
	If 1 1, C24-C31 -/→ D24-D31
		DXCR24 = NP32 NP33 DXCBP + . . .
		DXCR16/1 = NP32 P33 DXCBP + . . .
		DXCR8 = P32 NP33 DXCBP + . . .
		DXC/1 = P32 P33 DXCBP + . . .
		DXCBP = OU7 NO4 NO5 PH1 + . . .
			Effective byte into C-register depending on configuration of P32 and P33

	A0-A31 ⊕ CS0-CS31 ——→ S0-S31	SXPR = (FAS11 PH1) + . . .	Exclusive OR action places one's complement of A onto sum bus
	S0-S31 -/→ B0-B31	BXS = FAS11 PH1 + . . .	One's complement of private memory into B-register for adding in PH4
	1's -/→ CS0-CS31	CSX1 = S/NPRX + . . .	
	Set NPRX	S/NPRX = FAS11 PH1 + . . .	Prepare for AND action in PH2
	Enable T4L clock	S/T4L = FAS11 PH1 + . . .	
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
PH2	A0-A31 D0-D31 ——→ S0-S31	SXPR = NPRX NSDIS + . . .	Compare coincident 1-bits in private memory byte and effective byte
T4L	S0-S31 -/→ A0-A31	AXS = FAS11 PH2 + . . .	Put results into A
	Enable T4L clock	S/T4L = FAS11 PH2 + . . .	

Mnemonic: CB (71, F1)

(Continued)

Table 3-81. Compare Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH2 T4L (Cont.)	Set PH3	S/PH3	= PH2 NBR N(FNANLZ NANLZ) + . . .	
PH3	0 $\nrightarrow$ CC2	R/CC2	= FAS11 PH3 + . . .	If no bits of private memory byte and effective byte are coincident, reset CC2; otherwise, set CC2
	If A0-A31 ≠ 0, 1 $\nrightarrow$ CC2	S/CC2	= FAS11 PH3 NA0031Z + . . .	
	B0-B31 → S0-S31	SXB	= FAS11 PH3 + . . .	One's complement of private memory word
	S0-S31 $\nrightarrow$ A0-A31	AXS	= FAS11 PH3 + . . .	
	1 $\nrightarrow$ CS31	CSX1/8	= FAS11 PH3 + . . .	For two's complement
	Set DRQ for next instruction	S/DRQ	= FAS11 PH3 + . . .	Inhibits clock until data release received from memory
	Enable T6L clock	T6L	= NT1L NT4L NT8L NT10L NRESET	
	Set PH4	S/PH4	= FAS11 PH3 + . . .	
PH4	Generate ENDE	ENDE	= FAS11 PH4 + . . .	
T6L	A0-A31 + D0-D31 + CS31 → S0-S31	SXADD	= FAS11 PH4 + . . .	Subtract effective byte from private memory byte
	S0-S31 $\nrightarrow$ A0-A31	AXS	= FAS11 PH4 + . . .	Difference into A
	If D0-D31 ≥ A0-A31, 1 $\nrightarrow$ K00H	SXK	= SXADD	
	Set flip-flop K00H if S00 = 1	S/K00H	= S00	

		S00	= K00 S00X + NK00 NS00X N(. . .)	S00X = 1 if A0 ≠ D0
				NS00XN = 1

		S00X	= FAS11 N[NS00XN (A0 D0 + NA0 ND0)]	
	Set flip-flop TESTA	(S/TESTA)	= FAS11 PH4 + . . .	To test for condition codes
	Set flip-flop TESTA/1	S/TESTA/1	= (S/TESTA) NO4 NO6 O7 + . . .	
	Enable T6L clock	T6L	= NT1L NT4L NT8L NT10L NRESET	
	Set flip-flop PRE1	S/PRE1	= ENDE NHALT N(S/INTRAPF)	

Mnemonic: CB (71, F1)

(Continued)

Table 3-81. Compare Byte, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PRE1 (next instruction)	Set flip-flop CC3 if K00H = 0	S/CC3 = TESTA/1 NK00H NA0031Z + . . .	Private memory byte > effective byte
		R/CC3 = TESTA + . . .	
	Set flip-flop CCH if K00H = 1	S/CC4 = TESTA/1 K00H NA0031Z + . . .	Private memory byte > effective byte
		R/CC4 = TESTA + . . .	

Mnemonic: CB (71, F1)

Signal AXRR1 is true causing a one-bit downward alignment of the index register into the A-register at the clock ending PRE1. At this same clock, bit 31 of the index register is transferred to P32. The status of P32 will later determine which core memory halfword is to be clocked into the D-register.

At the clock pulse ending PRE2, the halfword in private memory is clocked into the A-register. The CS-register is filled with ones and the D-register is reset to zeros.

If halfword 0 were addressed, RQ would have been set at the end of PRE1, because the Compare Halfword instruction would not have been indexed. Flip-flop NPRX is reset at the end of PRE2 for the exclusive OR operation in the first execution phase.

The one's complement of the private memory halfword is taken by performing an exclusive OR operation on the halfword in the A-register and the ones in the CS-register, and this complement is transferred to the B-register. The selected halfword in the C-register is transferred to bits 16 through 31 in the D-register, and the sign bit is extended to bits 0 through 15 of the D-register. The individual bits in the A- and D-registers are compared with an AND operation, and the result is placed in the A-register. Flip-flop CC2 is reset if the A-register contains zeros, indicating that no bits of the private memory halfword and the sign-extended halfword are coincident. Otherwise CC2 is set.

The contents of the B-register are transferred to the A-register, and a one is set into flip-flop CS31 to obtain the two's complement of the private memory halfword. The contents of the A- and D-registers are added, subtracting the private memory halfword from the effective halfword, and the result is placed in the A-register. An end carry from bit 0 is stored in flip-flop K00H. This flip-flop and the contents of the A-register are examined in the first preparation phase of the following instruction, or, at times, in the first phase of interrupt, trap, or control panel operation. Flip-flop CC4 is set if a nonzero value in the A-register and a one in K00H indicate that the private memory halfword is less than the effective halfword. A zero in K00H and a nonzero value in the A-register cause flip-flop CC3 to be set, indicating that the private memory halfword is greater than the effective halfword. A sequence chart of the Compare Halfword instruction is given in table 3-82.

3-212 Compare Word (CW 31, B1)

The Compare Word instruction compares the contents of the register specified by the R-field with the contents of the effective word. Both words are treated as signed quantities. The condition code is set according to the results of the comparison.

The CW instruction sequences through the preparation states. If the instruction is not indirectly addressed, a memory request is generated for the next instruction. The address of the next instruction is taken from the address contained in the Q-register:

S/RQ = NIA PRE1 NINDX (NANLZ PRERQ)

The contents of the addressed private memory register are transferred to the A-register, ones are placed into the CS-register, and the D-register is cleared to zeros. The one's complement of the private memory word is taken by performing an exclusive-OR operation on the word in the A-register and the ones in the CS-register, and this complement is transferred to the B-register. The effective word in the C-register is transferred to the D-register. The individual bits in the A- and D-registers are compared with an AND operation, and the result is placed in the A-register. If the A-register contains zeros, indicating that no bits of the private memory word and the effective word were coincident, flip-flop CC2 is reset. Otherwise, CC2 is set.

The contents of the B-register are transferred to the A-register, and a one is set into flip-flop CS31 to obtain the two's complement of the private memory word. The contents of the A- and D-registers are added, subtracting the private memory word from the effective word. The sum is placed in the A-register. The presence or absence of an end carry, depending on the sign, is stored in flip-flop K00H. This flip-flop and the contents of the A-register are examined in the first preparation phase of the following instruction or in the first phase of interrupt, trap, or control panel operation. Flip-flop CC4 is set if a nonzero value in the A-register and a one in K00H indicate that the private memory word is less than the effective word. A zero in K00H and a nonzero value in the A-register cause flip-flop CC3 to be set, indicating that the private memory word is greater than the effective word. A sequence chart of the Compare Word instruction is given in table 3-83.

3-213 Compare Doubleword (CD 11, 91)

The Compare Doubleword instruction compares the effective doubleword with the contents of register R and R + 1. Both doublewords are treated as signed magnitudes. The condition code bits CC3 and CC4 are set according to the results of the comparison.

During PRE1, the address of the low order half of the doubleword in memory is gated onto the core memory lines from the C-register by forcing a one onto address line LB31 to select the odd numbered memory location.

The low order half of the data doubleword is read into the C-register during PRE2, and a memory request is generated for the high order half of the memory doubleword. Since the high order half of the doubleword is in an even numbered location, a zero is forced onto address line LB31 by inhibiting flip-flop P31 from setting regardless of the state of S31. At the clock ending PRE2, flip-flop LR31/2 is set to force a one onto private memory address line LR31 to obtain the low order half of the private memory register doubleword. The remainder of the preparation sequence is the same as the general preparation sequence.

Table 3-82. Compare Halfword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	RR0-RR31 —/→ A0-A31	AXRR = PRE2 NIA FAS11 NFUCB + . . .	Private memory halfword into A-register
	1's —/→ CS0-CS31	CSX1 = FAS11 PRE2 NIA + . . .	Set ones into CS-register
	0's —/→ D0-D31	DX = PRE2 NIA + . . .	Clear D-register to zeros
	Enable T4RL clock	S/T4RL = PREP + . . .	
	Set PH1	S/PH1 = PRE2 NIA NPREDO + . . .	
PH1 T4RL	If NP32, C0-C15 —/→ D16-D31	DXCR16 = NP32 DXC/5	Effective halfword into C-register, depending on state of P32
	C0 —/→ D0-D15	DXC/5 = OU5 NO4 NO5 PH1 + . . .	
	If P32, C16-C31 —/→ D16-D31	DXC/12 = DXC/7 = DXC/2 = DXC/5	
	C16 —/→ D0-D15	C0C16 = CXRR (RR0 NP32 + RR16 P32)	Extend sign left
	A0-A31 ⊕ CS0-CS31 ——→ S0-S31	SXPR = FAS11 PH1 + . . .	Exclusive OR action places one's complement of A onto sum bus. One's complement of private memory halfword into B-register for addition
	S0-S31 —/→ B0-B31	BXS = FAS11 PH1 + . . .	
	1's —/→ CS0-CS31	CSX1 = S/NPRX + . . .	
		NPRX = FAS11 PH1 + . . .	Prepare for AND action in PH2
	Enable T4L clock	S/T4L = FAS11 PH1 + . . .	
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
PH2	A0-A31 ∧ D0-D31 ——→ S0-S31	SXPR = NPRX	Compare coincident 1-bits in private memory halfword and effective halfword. Put results into A
	S0-S31 —/→ A0-A31	AXS = FAS11 PH2 + . . .	
	Enable T4L clock	S/T4L = FAS11 PH2 + . . .	
	Set PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	

Mnemonic: CH (51, D1)

(Continued)

Table 3-82. Compare Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T4L	0 ⇸ CC2	R/CC2 = FAS11 PH3 + . . .	If no 1-bits of private memory halfword and effective halfword are coincident, reset CC2; otherwise, set CC2
	If A0-A31 = 0, 1 ⇸ CC2	S/CC2 = FAS11 PH3 NA0031Z + . . .	
	B0-B31 → S0-S31	SXB = FAS11 PH3 + . . .	One's complement of private memory halfword ⇸ B
	S0-S31 ⇸ A0-A31	AXS = FAS11 PH3 + . . .	
	1 ⇸ CS31	CSX1/8 = FAS11 PH3 + . . .	
	Set DRQ for next instruction	S/DRQ = FAS11 PH3 + . . .	Inhibits clock until data release received from memory
	Enable T6L clock	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set PH4	S/PH4 = FAS11 PH3 + . . .	
PH4 T6L	Generate ENDE	ENDE = FAS11 PH4 + . . .	
	A0-A31 + D0-D31 + CS31 → S0-S31	SXADD = FAS11 PH4 + . . .	Subtract private memory halfword from effective halfword
	S0-S31 ⇸ A0-A31	AXS = FAS11 PH4 + . . .	Difference into A
	Set flip-flop K00H	SXK = SXADD	

	Set flip-flop K00H if S00 = 1	S/K00H = S00	S00X = 1 if A0 ≠ D0
			NS00XN = 1

		S00 = K00 S00X + NK00 NS00X N(. . .)	
		S00X = FAS11 N[NS00XN (A0 D0 + NA0 ND0)]	
	Set TESTA	S/TESTA = FAS11 PH4 + . . .	
	Set TESTA/1	S/TESTA/1 = (S/TESTA) NO4 NO6 O7 + . . .	
	Enable T6L clock	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set PRE1	S/PRE1 = ENDE NHALT N(S/INTRAPF)	

Mnemonic: CH (51, D1)

(Continued)

Table 3-82. Compare Halfword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PRE1 (next instruction)	Set flip-flop CC3 if K00H = 0	S/CC3	= TESTA/1 NK00H NA0031Z + . . .	Private memory halfword > effective halfword
		R/CC3	= TESTA + . . .	
	Set flip-flop CC4 if K00H = 1	S/CC4	= TESTA/1 K00H NA0031Z + . . .	Private memory halfword > effective halfword
		R/CC4	= TESTA + . . .	

Mnemonic: CH (51, D1)

Table 3-83. Compare Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	RR00-RR31 $\nrightarrow$ A00-A31	AXRR = (PRE2 NIA) FAS11 NFUCB + . . .	Private memory word into A-register
	1's $\nrightarrow$ CS00-CS31	CSX1 = (PRE2 NIA) FAS11 + . . .	Set all ones in CS-register
	0's $\nrightarrow$ D0-D31	DX/1 = PRE2 NIA + . . .	Reset D to zeros
	Set PH1	S/PH1 = NPREDO (PRE2 NIA) + . . .	
	Enable T4RL	S/T4RL = PREP + . . .	
PH1	C0-C31 $\nrightarrow$ D0-D31	DXC/6 = OU3 (NO4 NO5) PH1 + . . .	Effective word into D-register
	A0-A31 $\oplus$ CS0-CS31 $\longrightarrow$ S0-S31	SXPR = FAS11 PH1 + . . .	Exclusive OR action places one's complement of A onto sum bus
T4RL	S0-S31 $\nrightarrow$ B0-B31	BXS = FAS11 PH1 + . . .	One's complement of (R) into B-register for adding later
	Q15-Q31 $\nrightarrow$ P15-P31	PXQ = PH1 PRERQ + . . .	Next instruction address into P-register
	Set NPRX	S/NPRX = FAS11 PH1 + . . .	
	1's $\nrightarrow$ CS0-CS31	CSX1 = S/NPRX + . . .	Prepare for AND action in PH2
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
	Enable T4L clock	S/T4L = FAS11 PH1 + . . .	
PH2	A0-A31 $\wedge$ D0-D31 $\longrightarrow$ S0-S31	SXPR = NPRX NSDIS + . . .	Compare coincident 1-bits in private memory word and effective word
T4L	S0-S31 $\nrightarrow$ A0-A31	AXS = FAS11 PH2 + . . .	Put result into A-register
	Set PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	
	Enable T4L clock	S/T4L = FAS11 PH2 + . . .	
PH3	0 $\nrightarrow$ CC2	R/CC2 = FAS11 PH3 + . . .	If no bits in the two words are coincident, reset CC2; otherwise set CC2

Mnemonic: CW (31, B1)

(Continued)

Table 3-83. Compare Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 (Cont.) T6L	If A0-A31 ≠ 0, 1 ⇸ CC2	S/CC2 = FAS11 PH3 NA0031Z + . . .	
	B0-B31 → S0-S31	SXB = FAS11 PH3 + . . .	One's complement of private memory register contents
	S0-S31 ⇸ A0-A31	AXS = FAS11 PH3 + . . .	
	1 ⇸ CS31	CSX1/8 = FAS11 PH3 + . . .	For two's complement
	0 ⇸ CS0-CS30	R/CS0-CS30 = .	
	Set DRQ	S/DRQ = FAS11 PH3 + . . .	Inhibits another clock until data release received from memory
	Set PH4	S/PH4 = FAS11 PH3 + . . .	
PH4	Generate ENDE	ENDE = FAS11 PH4 + . . .	
	A0-A31 + D0-D31 + CS31 → S0-S31	SXADD = FAS11 PH4 + . . .	Add effective word and two's complement of private memory word
	S0-S31 ⇸ A0-A31	AXS = FAS11 PH4 + . . .	Difference into A-register
		SXK = SXADD	
	If D0-D31 ≥ A0-A31, 1 ⇸ K00H	S/K00H = S00	

		S00 = (S00X K00) + (NS00X NK00) N(. . .)	S00X = 1 if A0 ≠ D0
			NS00XN = 1

		S00X = FAS11 N[NS00XN (A0 D0 + NA0 ND0)]	
	Set TESTA	S/TESTA = FAS11 PH4 + . . .	To test A-register for condition codes
	Set TESTA/1	S/TESTA/1 = S/TESTA NO4 NO6 O7	
	Set PRE1	S/PRE1 = ENDE NHALT N(S/INTRAPF)	
	Enable T6L clock	T6L = NT1L NT4L NT8L NT10L NRESET	
PRE1 next instruction	Set flip-flop CC3 if K00H = 0	S/CC3 = TESTA/1 NK00H NA0031Z + . . .	Private memory word > effective word
		R/CC3 = TESTA + . . .	

Mnemonic: CW (31, B1)

(Continued)

Table 3-83. Compare Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PRE1 next instruction (Cont.)	Set flip-flop CC4 if K00H = 1	S/CC4 = TESTA/1 K00H NA0031Z + . . . R/CC4 = TESTA + . . .	Private memory word < effective word

Mnemonic: CW (31, B1)

The low order word of the effective memory doubleword is read into the C-register, and into the one's complement as it is transferred into the D-register. A one is forced into flip-flop CS31 to obtain the two's complement during addition. The low order word of the private memory doubleword is loaded into the A-register. The A- and D-registers contents are gated into the adder, which subtracts the effective word from the private memory word. The sum is transferred to the A-register, and flip-flop K00H is set if an end carry occurs. Flip-flop CS31 is set to reflect the carry if signal K00H is true. The high order word of the effective memory doubleword is read into the C-register, and into the one's complement as it is transferred into the D-register. Flip-flop BWZ is set if the A-register does not contain all zeros, this indicates a nonzero difference between the low order words. The contents of the A- and D-registers are again gated into the adder, subtracting the high order effective word from the high order private memory word. The result is placed in the A-register. Flip-flop K00H is set if no end carry results from bit 0. The A-register and flip-flop K00H are examined in the first phase of the instruction or interrupt, trap, or control sequence that follows. Flip-flop CC3 is set if a nonzero value in the A-register and a zero in flip-flop K00H indicate that the private memory doubleword was greater than the effective doubleword. Flip-flop CC4 is set if a one in flip-flop K00H and a nonzero value in the A-register indicate that the private memory doubleword was less than the effective doubleword. A sequence chart of the Compare Doubleword instruction is given in table 3-84.

3-214 Compare Selective (CS 45, C5)

The Compare Selective instruction compares the contents of register R with the effective word in only those bit positions selected by a one in corresponding bit positions of the mask in register R + 1. The contents of register R and the effective word are ignored in those bit positions designated by a zero in corresponding bit positions of register R + 1. The selected contents of register R and the effective word are treated as positive magnitudes, and the condition codes CC3 and CC4 are set according to the results of the comparison. If the R-field of the instruction is an odd value, Compare Selective compares the contents of the register with the logical product (AND) of the effective word and the contents of register R. A sequence chart of the Compare Selective instruction is given in table 3-85.

The normal preparation sequence for the CS instruction is described under the preparation sequence. During PRE2, the odd numbered private memory register is addressed by setting LR31/2. In the execution phases, the effective word in the C-register is clocked into the D-register. The mask is read from private memory register R + 1 into the A-register. With ones in the CS-register, and NPRX true, an AND operation is performed in the adder between the A- and D-registers to extract ones from the effective word wherever a one appears in the corresponding bit of the mask. The result is placed in the B-register.

The contents of private memory register R are loaded into the D-register by way of the C-register. An AND operation is again performed between the A- and D-registers. This extracts ones from the private memory word wherever a one appears in the corresponding bit of the mask. The result is placed in the A-register.

The contents of the B-register, representing the extracted bits from the effective word, are one's complemented by way of the C-register and are placed in the D-register. Flip-flop CS31 is set to form the two's complement. The contents of the A- and D-registers are gated into the adder, subtracting the selected bits of the effective word from the selected bits of the private memory word. The result is placed in the A-register. Flip-flop K00H is set if no end carry results out of bit 0.

The A-register and flip-flop K00H are examined in the first phase of the instruction or interrupt, trap, or control panel operation that follows. Flip-flop CC3 is set if a nonzero value in the A-register and a zero in flip-flop K00H indicate that the private memory word bit combination was greater than the effective word bit combination. Flip-flop CC4 is set if a one in flip-flop K00H and a nonzero value in the A-register indicate that the private memory bit combination was less than the effective word bit combination.

3-215 Compare With Limits in Registers (CLR 39, B9)

The Compare with Limits in Registers instruction compares the contents of the effective word with the contents of both private memory registers R + 1 and R, in that order. All three words are treated as signed magnitides. The condition codes are set according to the results of the comparison, CC1 and CC2 representing the results of the comparison of register R + 1 and the effective word. CC3 and CC4 represent the results of the comparison of register R and the effective word.

The normal preparation sequence for the instruction is described under preparation sequence. In PRE2, the least significant LR line is set true to address the odd numbered private memory register, R + 1.

S/LR31/2 = FUCLR PRE2 NIA

The low order effective word is read from memory and one's complemented by way of the C-register into the D-register. Flip-flop CS31 is set to form the two's complement in the adder. The contents of private memory register R + 1 are loaded into the A-register. The A- and D-register contents are gated into the adder, subtracting the effective word from the private memory word. Flip-flop K00H and the A-register are examined to determine which value was larger. If the private memory word is greater than the effective word, flip-flop CC3 is set; otherwise, flip-flop CC4 is set.

The contents of private memory register R are gated into the A-register, and the effective word is subtracted from this value by again gating the A- and D-register values into the adder with CS31 set for the two's complement.

Table 3-84. Compare Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence except		
	During PRE1		
	C15-C31 → LM15-LB30	LMXC = ENDE	
	1 → LB31/2	LB31/2 = PREFADO LMXC NC0 + . . .	Address low order half of effective doubleword
	During PRE2		
	Force P31 to zero	S/P31 = S31 PXS NP31Z + . . .	Inhibit P31 from setting. NP31Z cannot be true in PRE2
		NP31Z = NPRE2 + IA + NFADW	
	Set flip-flop MRQ	MRQ = PRE2 NIA PREDO NANLZ + . . .	
	Set LR31/1	S/LR31/1 = (S/PH1/1) OU1 NO4 NO5 NO6 + . . .	Address low order half of private memory doubleword
PH1 T4RL	MB0-MB31 → C0-C31	CXMB = DGC	
	NC0-NC31 ↛ D0-D31	DXNC/1 = FAS2 PH1 + . . .	One's of low order memory word complement into D-register
	1 ↛ CS31	S/CS31 = CSX1/31	For two's complement
		CSX1/31 = DXNC/1	
	RR0-RR31 ↛ A0-A31	AXRR = FAS22 PH1 + . . .	Low order private memory word
	Set flip-flop RQ	S/RQ = OU1 (NO5 NO6) PH1 + . . .	Request for next instruction
	Set flip-flop DRQ	S/DRQ = PREDO PH1 + . . .	Inhibits next clock until signal received from memory
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: CD (11, 91)

(Continued)

Table 3-84. Compare Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)	A0-A31 + D0-D31 + CS31 ⟶ S0-S31	SXPR = SXADD = FAS22 PH2 + . . .	Subtract low order effective word from low order private memory word
	S0-S31 —/→ A0-A31	AXS = FAS22 PH2 + . . .	Transfer to A-register
	Set flip-flop K00H if S00 = 1	S/K00H = S00	
		R/K00H = . . .	

		S00 = K00 S00X + NK00 NS00X N(. . .)	S00XN = 0 in PH2
			S00X = 0

		S00X = S00XN + . . .	S00 ⟹ no end carry out of bit position 0
	MB0-MB31 ⟶ C0-C31	CXMB = DGC	
	NC0-NC31 —/→ D0-D31	DXNC = FAS2 PH2 + . . .	One's of high order memory word complement into D-register
	Set flip-flop PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If A ≠ 0, set flip-flop BWZ	S/BWZ = NA0031Z (S/BWZ/1)	BWZ = 0 ⟹ low order words equal
		(S/BWZ/1) = FAS22 NOL9 PH3 + . . .	
	NK00H —/→ CS31	CSX1/8 = NK00H (S/BWZ/1)	End carry out of bit position 0
	RR0-RR31 —/→ A0-A31	AXRR = FAS22 PH3 + . . .	High order private memory word
	Q —/→ P	PXQ = PREDO PH3 + . . .	Next instruction address
	Set flip-flop DRQ for next instruction	S/DRQ = FAS22 PH3 + . . .	
	Set flip-flop PH5	S/PH5 = BRPH5 + . . .	
		BRPH5 = FAS22 PH3 + . . .	
PH5 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: CD (11, 91)

(Continued)

Table 3-84. Compare Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T6L (Cont.)	Generate ENDE	ENDE = FAS22 PH5 + . . .	
	A0-A31 + D0-D31 + CS31 → S0-S31	SXADD = FAS22 PH5 + . . .	Difference high order halves of doublewords
	S0-S31 $\nrightarrow$ A0-A31	AXS = FAS22 PH5 + . . .	Place in A-register
	Set flip-flop K00H if S00 = 1	S/K00H = S00	
		S00 = K00 NS00X + NK00 S00X N(. . .)	
		S00X = S00XN (A0 D0 + NA0 ND0) N(. . .)	
		S00XN = FAS22 PH5 + . . .	
	If BWZ, merge 1 $\nrightarrow$ A31	A31X1 = FAS22 BWZ + . . .	To indicate difference between low order words ≠ 0
	Set flip-flop TESTA	(S/TESTA) = FAS22 PH5 + . . .	For condition code test
	Set flip-flop TESTA/1	S/TESTA/1 = (S/TESTA) NO4 NO6 O7 + . . .	
PRE1 (next instruction)	Set flip-flop CC3 if K00H = 0	S/CC3 = TESTA/1 NK00H NA0031Z + . . .	Private memory doubleword > effective doubleword
		R/CC3 = TESTA	
	Set flip-flop CC4 if K00H = 1	S/CC4 = TESTA/1 K00H NA0031Z + . . .	Private memory doubleword < effective doubleword
		R/CC4 = TESTA	

Mnemonic: CD (11, 91)

Table 3-85. Compare Selective, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
ENDP	S/LR31/2	S/LR31/2 = PRE2 NIA FUCS + . . .	Merge a one onto address to select mask
PH1 T4RL	C0-C31 —/→ D0-D31	DXC/6 = FAS6 PH1 + . . .	Effective word to D-register
	RR0-RR31 —/→ A0-A31	AXRR = FAS15 PH1 + . . .	Mask from private memory register R + 1
	1 —/→ NPRX	S/NPRX = FAS6 PH1 + . . .	To prepare for AND operation in PH2
	1's —/→ CS0-CS31	CSX1 = (S/NPRX) + . . .	
	Set flip-flop CXRR	S/CXRR = OU4 NO4 O5 O7PH1	For transfer of contents of selected private memory register to C-register
	Enable T10L clock	S/T10L = FUCS PH1 + . . .	
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
PH2 T10L	A0-A31 + D0-D31 ——→ S0-S31	SXPR = NPRX + . . .	Ones extracted from effective word under control of mask
	S0-S31 —/→ B0-B31	BXS = FAS6 PH2 + . . .	Contents of selected private memory register
	RR0-RR31 ——→ C0-C31	CXRR set in PH1	
	C0-C31 —/→ D0-D31	DXC/6 = (FUCS PH2) + . . .	Transfer to D-register
	S/NPRX	S/NPRX = FUCS PH2 + . . .	For AND operation
	1's —/→ CS0-CS31	CSX1 = S/NPRX + . . .	
	Set T4L clock	S/T4L = FUCS PH2 + . . .	
	Set PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	
PH3 T4L	A0-A31 ∧ D0-D31 ——→ S0-S31	SXPR = NPRX + . . .	Ones extracted from private memory register under control of mask
	S0-S31 —/→ A0-A31	AXS = FUCS PH3 + . . .	Transfer to A-register

Mnemonic: CS (45, C5)

(Continued)

Table 3-85. Compare Selective, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T4L (Cont.)	Set CXS	S/CXS = FUCS PH3 + . . .	For transfer of sum bus contents to C-register
	Set PH4	S/PH4 = PH3 NBR + . . .	
PH4 T6L	B0-B31 ——→ S0-S31	SXB = OU4 NO4 O5 O7 PH4 + . . .	Ones extracted from effective word to C-register via sum bus
	S0-S31 —/→ C0-C31	CXS true in PH4	
	NC0-NC31 —/→ D0-D31	DXNC/1 = FUCS PH4 + . . .	One's complement of extracted bits of effective word into D-register
	1 —/→ CS31	S/CSX1/31 = DXNC/1 + . . .	For two's complement
	Set MRQ/1 and DRQ for next instruction	MRQ/1 = FAS6 PH4 + . . . S/DRQ = FAS6 PH4 + . . .	
	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set PH5	S/PH5 = PH4 NBR	
PH5 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Enable ENDE	ENDE = FAS15 PH5 + . . .	
	A0-A31 + D0-D31 + CS31 ——→ S0-S31	SXADD = FUCS PH5 + . . .	Difference selected bits of effective word and register R and transfer to A-register
	S0-S31 —/→ A0-A31	AXS = FAS15 PH5 + . . .	
	Set flip-flop K00H if S00 = 1	K00H = S00	FAS11 = 0
		S00 = (S00X + NK00) (NS00X + K00)	S00X = 0
		S00X = (A0 ND0 + NA0 D0) FAS11 NS00XN	S00 ⟹ no end carry out of bit position 0
	Set test for A	S/TESTA = FAS15 PH5 + . . .	
		S/TESTA/1 = (S/TESTA) (NO4 NO6 O7) + . . .	

Mnemonic: CS (45, C5)

(Continued)

Table 3-85. Compare Selective, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PRE1 next instruction	Set condition codes	S/CC3	= (TESTA/1 NA0031Z) NK00H + . . .	Extracted bits of private memory word > extracted bits of effective word
		S/CC4	= (TESTA/1 NA0031Z) K00H + . . .	Extracted bits of private memory word less than extracted bits of effective word
				Mnemonic: CS (45, C5)

The contents of flip-flops CC3 and CC4 are transferred to CC1 and CC2. In the first phase of the instruction, or interrupt, trap, or control panel operation that follows, flip-flops CC3 and CC4 are set by checking the A-register and the state of flip-flop K00H. Flip-flop CC3 is set if the word from private memory register R is greater than the effective word; otherwise, flip-flop CC4 is set. A sequence chart of the Compare with Limits in Register instruction is given in table 3-86.

3-216 Compare With Limits in Memory (CLM 19, 99)

The Compare with Limits in Memory instruction compares the contents of register R with the effective memory doubleword. First the low order effective doubleword is compared, and the high order effective word is then compared. All three words are treated as signed magnitudes. The condition codes are set according to the results of the comparison.

The normal preparation sequence for the CLM instruction is described under the preparation sequence. During the preparation sequence, a one is forced onto the least significant memory address line LB31 to address the low order effective word.

LB31/2 = LMXC NC0 PREFADO + . . .

The low order effective word is read from memory and one's complemented into the D-register by way of the C-register. Flip-flop CS31 is set to form the two's complement in the adder. The contents of the selected private memory register are loaded into the A-register. The A- and D-register contents are gated into the adder, subtracting the low order effective word from the private memory word. Flip-flop K00H and the A-register are examined to determine which value is larger. Flip-flop CC3 is set if the private memory word is greater than the effective word; otherwise, flip-flop CC4 is set.

The contents of the selected private memory register are again loaded into the A-register. The high order effective word is read from memory and one's complemented into the D-register by way of the C-register. Flip-flop CS31 is again set to form the two's complement in the adder. The contents of flip-flops CC3 and CC4 are transferred to flip-flops CC1 and CC2. The A- and D-register outputs are gated into the adder, subtracting the high order effective word from the private memory word. In the first phase of the next operation, whether instruction, interrupt, trap, or control panel operation, flip-flop K00H and the A-register are examined to determine which value is larger. Flip-flop CC3 is set, if K00H contains a zero, indicating that the private memory word was greater than the effective word; otherwise, flip-flop CC4 is set.

A sequence chart of the Compare with Limits in Memory instruction is given in table 3-87.

3-217 OR Word (OR 49, C9)

The OR Word instruction merges the individual bits of the contents of the private memory register R and the effective word. The result, which contains 1-bits in those bit positions where ones appear in either R or EW, is transferred to private memory register R.

The OR Word instruction can be indirectly addressed, or indexed, or both. The normal preparation sequence, as well as the sequencing for all modes and conditions of the OR instruction, is described under the preparation sequence.

At PH1 clock, EW in the C-register is transferred to D, and the private memory word is clocked into the A-register.

Both signals SXA and SXD come true during PH3. This allows the sum bus to follow the contents of both the A-register and the D-register, effecting a merge or OR operation. The contents of the sum bus are placed on the RW lines to be stored back into private memory register R. The contents of the sum bus are also transferred to the A-register in PH3 to be tested by CC3 and CC4. Flip-flop TESTA is set at the PH3 clock pulse.

Condition code bits CC3 and CC4 are not affected until the clock at the end of the phase following PH3 of the OR instruction. This normally is at the end of PREI, although it could also be at the end of the first phase of an interrupt or trap operation or at the clock of phase PCP1 of a control panel operation.

A sequence chart of the OR Word instruction is given in table 3-88.

3-218 Exclusive OR Word (EOR 48, C8)

The EOR instruction performs a logical addition on the contents of private memory register R and the effective word. The result contains one-bits in those bit positions where ones appear in either R or EW, and zeros in those bit positions where either ones or zeros appear in both R and EW. The result is transferred to private memory register R.

The EOR instruction can be indexed, indirectly addressed, or both. The normal preparation sequence, as well as the sequencing for all modes and conditions of the EOR instruction, is described under the preparation sequence.

The effective word in the C-register is transferred to D at the PH1 clock, and the private memory word R is clocked into the A-register.

During PH3, the signal SXPR is true, allowing the sum bus to follow the propagate signals. The contents of the sum bus, which contain the exclusive-OR information, are then placed onto the RW lines. The contents of the sum bus are also transferred in PH3 to the A-register to be tested by CC3 and CC4. Flip-flop TESTA is set at the PH3 clock pulse.

Table 3-86. Compare With Limits in Register, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
ENDP	Set LR31	S/LR31/2 = FUCLR PRE2 NIA + . . .	Addresses R + 1
PH1 T4RL	NC0-NC31 —/→ D0-D31	DXNC/1 = FAS1 PH1 + . . .	One's complement of effective word into D-register
	Set flip-flop CS31	S/CS31 = CSX1/31 = DXNC/1 + . . .	For two's complement. Contents of private memory register
	RR0-RR31 —/→ A0-A31	AXRR = FAS22 PH1 + . . .	R + 1 into A-register
	Q15-Q31 —/→ P15-P31	PXQ = PRERQ PH1 + . . .	
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	A0-A31 + D0-D31 + CS31 ⟶ S0-S31	SXADD = FAS22 PH2 + . . .	Private memory register R + 1 - effective word
	S0-S31 ⟶ A0-A31	AXS = FAS22 PH2 + . . .	Difference into A-register
	Set flip-flop K00H if S00 = 1	S/K00H = S00	
		S00 = (S00X K00 = NS00X NK00) NFASHFL	
		S00X = (A0 D0 + NA0 ND0) S00XN	

		S00XN = FAS1 PH2 + . . .	S00XN = 1
			S00X = 1, if A0 = D0
			S00 ⇒ no end carry from bit position 0, if signs are alike; carry if signs are unlike

	Set A test	S/TESTA = FAS1 PH2 + . . .	
		S/TESTA/1 = (S/TESTA) FAS1 + . . .	
	Enable T4L	S/T4L = FAS1 PH2 + . . .	
	Set PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + . . .	

Mnemonic: CLR (39, B9)

(Continued)

Table 3-86. Compare With Limits in Register, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3 T4L	Set condition codes CC3, CC4	R/CC3	= TESTA + . . .	
		R/CC4	= TESTA + . . .	
	Set flip-flop CC3, if K00H = 0	S/CC3	= TESTA/1 NA0031Z NK00H + . . .	Contents of R + 1 > effective word
	Set flip-flop CC4, if K00H = 1	S/CC4	= TESTA/1 NA0031Z K00H + . . .	Contents of R + 1 < effective word
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR	= FAS22 PH3 + . . .	Contents of private memory register R
	Set CS31	S/CS31	= CSX1/31 = CSX1/8 + . . .	For two's complement of effective word
		CSX1/8	= FAS1 PH3 + . . .	
	Set DRQ for next instruction	S/DRQ	= FAS22 PH3 + . . .	Inhibits clock until data release from memory
	Set PH5	S/PH5	= BRPH5 NCLEAR + . . .	
		BRPH5	= FAS22 PH3 + . . .	
PH5 T6L	Enable T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
	Enable ENDE	ENDE	= FAS22 PH5 + . . .	
	A0-A31 + D0-D31 + CS31 $\rightarrow$ S0-S31	SXADD	= FAS22 PH5 + . . .	Contents of private memory register R - effective word
	S0-S31 $\nrightarrow$ A0-A31	AXS	= FAS22 PH5 + . . .	Difference into A
	A0 $\oplus$ D0 $\oplus$ K00 $\nrightarrow$ K00H	S00XN	= FAS22 PH5 + . . .	
	Set A test	S/TESTA	= FAS22 PH5 + . . .	
		S/TESTA/1	= (S/TESTA) FAS22 PH5 + . . .	
	0 $\nrightarrow$ CC1	R/CC1	= FAS1 PH5 + . . .	Transfer contents of CC3 and CC4 to CC1 and CC2
	CC3 $\nrightarrow$ CC1	S/CC1	= FAS1 PH5 CC3 + . . .	
	0 $\nrightarrow$ CC2	R/CC2	= FAS1 PH5 + . . .	
	CC4 $\nrightarrow$ CC2	S/CC2	= FAS1 PH5 CC4 + . . .	

Mnemonic: CLR (39, B9)

(Continued)

Table 3-86. Compare With Limits in Register, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PRE1 Next instruction	If private memory word > effective word, $1 \nrightarrow$ CC3	R/CC3 = TESTA + . . . S/CC3 = TESTA/1 NA0031Z NK00H + . . .	This action occurs at the clock of the phase following ENDE
	If private memory word < effective word, $1 \nrightarrow$ CC4	R/CC4 = TESTA + . . . S/CC4 = TESTA/1 NA0031Z K00H + . . .	

Mnemonic: CLR (39, B9)

Table 3-87. Compare With Limits in Memory, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
ENDP	1 ——→ LB31	LB31/2 = LMXC NC0 PREFADO	Address EW + 1 if EW is even
PH1 T4RL	NC0-NC31 —/→ D0-D31	DXNC = FAS1 PH1 + . . .	One's complement of low order effective word into D-register
	Set CS31	S/CS31 = DXNC/1 = FAS1 PH1 + . . .	For two's complement
	RR0-RR31 —/→ A0-A31	AXRR = FAS22 PH1 + . . .	Contents of selected private memory register
	Set DRQ for operand	S/DRQ = PREDO PH1 + . . .	Inhibits bits clock until data release received from memory
	Set PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	
PH2	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	A0-A31 + D0-D31 + CS31 ——→ S0-S31	SXADD = FAS22 PH2 + . . .	Private memory word, low order effective word
	S0-S31 —/→ A0-A31	AXS = FAS22 PH2 + . . .	Difference into A-register
		S00XN = FAS1 PH2 + . . .	
	Set flip-flop K00H if S00 = 1	S00X = S00XN (A0 D0 + NA0 ND0)	
		S00 = S00X K00 + NS00X NK00 N(. . .)	
		S/K00H = S00	
	MB0-MB31 —/→ C0-C31	CXMB = DGC	
	NC0-NC31 —/→ D0-D31	DXNC/1 = OU1 OL9 PH2 + . . .	One's complement of high-order effective word into D-register
	Set A test	S/TESTA = FAS1 PH2 + . . .	For condition code test
		S/TESTA/1 = (S/TESTA) FAS1 + . . .	
	Set T4L	S/T4L = FAS1 PH2 + . . .	

Mnemonic: CLM (39, 99)

(Continued)

Table 3-87. Compare With Limits in Memory, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH2 (Cont.)	Set PH3	S/PH3	= PH2 NBR N(FNANLZ NANLZ) + . . .	
PH3 T4L	Set condition codes CC3, CC4			
	Set flip-flop CC3 if K00H = 0	S/CC3	= TESTA/1 NA0031Z NK00H	
		R/CC3	= TESTA	
	Set flip-flop CC4 if K00H = 1	S/CC4	= TESTA/1 NA0031Z K00H	Private memory word↛ A-register
		R/CC4	= TESTA	
	RR0-RR31↛A0-A31	AXRR	= FAS22 PH3 + . . .	Private memory word A-register
	Set CS31	S/CS31	= CSX1/31 = CSX1/8 + . . .	For two's complement of effective word
		CSX1/8	= FAS1 PH3 + . . .	
	Q15-Q31↛P15-P31	PXQ	= PREDO PH3 + . . .	
	Set DRQ for next instruction	S/DRQ	= FAS22 PH3 + . . .	Inhibits clock until data release received from memory
	Set PH5	S/PH5	= PH3 BRPH5 + . . .	
		BRPH5	= FAS22 PH3 + . . .	
PH5 T6L	Enable T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
	A0-A31 + D0-D31 + CS31 → S0-S31	SXADD	= FAS22 PH5 + . . .	Private memory word, high order effective word
	S0-S31↛A0-A31	AXS	= FAS22 PH5 + . . .	Difference into A-register
	Set flip-flop K00H if S00 = 1	S00XN	= FAS22 PH5 + . . .	
	Set A test	S/TESTA	= FAS22 PH5 + . . .	
		S/TESTA/1	= (S/TESTA) FAS1 + . . .	
	0↛CC1	R/CC1	= FAS1 PH5	

Mnemonic: CLM (39, 99)

(Continued)

Table 3-87. Compare With Limits in Memory, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH5 T6L (Cont.)	CC3 $\nrightarrow$ CC1	S/CC1	= FAS1 PH5 CC3	Transfer CC3, CC4 $\longrightarrow$ CC1, CC2
	0 $\nrightarrow$ CC2	R/CC2	= FAS1 PH5	
	CC4 $\nrightarrow$ CC2	S/CC2	= FAS1 PH5 CC4	
PRE1 (next in-struction)	Set flip-flop CC3 if K00H = 0	S/CC3	= TESTA/1 NA0031Z NK00H + . . .	Private memory word > effective word
		R/CC3	= TESTA + . . .	
	Set flip-flop CC4 if K00H = 1	S/CC4	= TESTA/1 NA0031Z K00H + . . .	Private memory word < effective word
		R/CC4	= TESTA + . . .	

Mnemonic: CLM (39, 99)

Table 3-88. OR Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	MB0-MB31 $\nrightarrow$ C0-C31	CXMB = /DGC/	EW into C-register
	Set RQ	S/RQ = PRE1 NIA NINDX PRERQ + . . .	
PH1 T4RL	Set T8L	S/T8L = FAS9 PH1 + . . .	
	C0-C31 $\nrightarrow$ D0-D31	DXC/6 = FAS9 PH1 + . . .	EW into D-register
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FAS9 PH1 + . . .	R into A-register
	Q15-Q31 $\nrightarrow$ P15-P31	PXQ = PH1 PRERQ + . . .	
	Set DRQ	S/DRQ = FAS9 PH1 + . . .	
	Go to PH3	BRPH3 = FAS9 PH1 O4 + . . .	
PH3 T8L	End	ENDE = FAS9 PH3 + . . .	Final phase
	A0-A31 → S0-S31	SXA = FUOR PH3 + . . .	A OR D into sum bus
	D0-D31 → S0-S31	SXD = FUOR PH3 + . . .	
	S0-S31 → RW0-RW31	RW = FAS9 PH3 + . . .	Store result into R
	S0-S31 $\nrightarrow$ A0-A31	AXS = FAS9 PH3 + . . .	Put result in A for test
	Set test for A	S/TESTA = FAS9 PH3 + . . .	
PRE1 (next instruction)	Test contents of A-register	R/CC3 = TESTA	
		R/CC4 = TESTA	
	If A0 = 0 and A1-A31 > 0, set CC3	S/CC3 = NTESTA/1 TESTA NA0 NA0031Z	
	If A0 = 1, set CC4	S/CC4 = NTESTA/1 TESTA A0	

Mnemonic: OR (49, C9)

Condition code bits CC3 and CC4 are not affected until the clock at the end of the phase following PH3 of the EOR instruction. This normally is at the end of PRE1, although it could also be at the end of the first phase of an interrupt or a trap operation or at the clock of phase PCP1 of a control panel operation. If A0 = 0 and A1-31 ≠ 0, set CC3; if A0 = 1, set CC4.

A sequence chart of the Exclusive OR Word instruction is given in table 3-89.

3-219 AND Word (AND 4B, CB)

The AND Word instruction performs a logical multiplication of the individual bits of the contents of private memory register R and the effective word. The result, which contains one-bits in bit positions where both R and EW contain ones and zero-bits in all other bit positions, is transferred to private memory register R.

The AND instruction can be indirectly addressed, or indexed, or both. The normal preparation sequence, as well as the sequencing for all modes and conditions of the AND instruction, is described under the preparation sequence.

At the PH1 clock, the EW in the C-register is transferred to D, and the private memory word R is clocked into the A-register. The CS-register is filled with ones at the PH1 clock and NPRX is set to prepare for the AND operation in PH2.

During PH3, SXPR is enabled. This allows PR0 through PR31 to be true wherever the corresponding bits of the A-, D-, and CS-registers are ones. The data in PR0 through PR31 is transferred to the sum bus, and the sum bus contents are placed on the RW lines to be stored in private memory register R. The contents of the sum bus are also transferred in PH3 to the A-register to be tested by CC3 and CC4. Flip-flop TESTA is set at the PH3 clock pulse.

Condition code bits CC3 and CC4 are not affected until the clock at the end of the phase following PH3 of the AND instruction. This normally is at the end of PRE1, although it could also be at the end of the first phase of an interrupt or trap operation or at the clock of phase PCP1 of a control panel operation. If A0 = 0 and A1-A31 ≠ 0, set CC3; if A0 = 1, set CC4.

A sequence chart of the AND Word instruction is given in table 3-90.

3-220 Shift (S 25, A5)

The shift instruction shifts the contents of the private memory register selected by the R-field of the instruction or shifts register R and register R + 1. This type of shift is determined as follows: If neither indirect addressing nor indexing is called for in the S-instruction, bit positions 21 through 31 of the reference address field determine the type, direction, and amount of the shift. If indirect addressing is indicated in the instruction (where bit 0 is a one), bits 15 through 31 of the instruction word are used to fetch the indirect word. Bits 21 through 31 of the indirect word determine the type, the direction, and the amount of shift. If only indexing is indicated in the instruction word, bits 21 through 23 of the instruction word determine the type of shift. The direction and amount of shift are determined by the sum of bits 25 through 31 of the instruction and bits 25 through 31 of the index register. If both indirect addressing and indexing are indicated, bits 15 through 31 of the reference address are used as the address to fetch the new word. Bits 21 through 23 of the indirect word are used to determine the type of shift. The direction and the amount of the shift are determined by bits 25 through 31 of the indirect word plus bits 25 through 31 of the index register.

Bits 21 through 23 of the effective address determine the type of shift as listed in table 3-91.

Bit positions 25 through 31 of the effective address contain a shift count that determines the direction and the amount of the shift. The shift count is treated as a seven-bit signed binary integer with bit 25 as the sign bit. A one in bit 25 indicates a right shift with the count given in two's complement form. A zero in bit 25 indicates a left shift. The value of the count can be within the range -64 to +63.

Overflow occurs on a left shift whenever the value of the sign bit changes. At the completion of the shift operation, the condition code register is set to indicate the results of the shift; the result is then stored in the private memory registers.

The preparation sequence is the same as the general preparation sequence except that data is loaded from the odd numbered private memory register into the A-register in case the instruction calls for a double-register shift. This data is loaded into the B-register in the first execution phase, and then is cleared if the instruction calls for a single-register shift.

3-221 SHIFT PHASE. In the shift phase, the data in the A-register or in the A- and B-registers is shifted as follows:

a. Left shift. In a single-register arithmetic left shift or a single-register logical left shift, the data in the A-register is shifted left the number of bit positions indicated by the count, and zeros are placed in the vacated bit positions on the right. The bits shifted past A0 are lost. In a single-register circular left shift, the bits shifted past position A0 are clocked into position A31. In a double-register arithmetic left shift or a double-register logical left shift, the contents of the A- and B-registers are shifted left the number of bit positions indicated by the count. Zeros are placed in the vacated bit positions on the right in the B-register. Bits shifted past bit 0 in the B-register are copied into bit 31 of the A-register. Bits shifted past bit 0 of the A-register are lost. In a double-register circular left shift, bits shifted past A0 are clocked into B31.

Table 3-89. Exclusive OR Word, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	MB0-MB31 → C0-C31	CXMB	= /DGC/ + . . .	EW into C-register
	Set RQ	S/RQ	= PRE1 NIA NINDX PRERQ + . . .	
PH1 T4RL	Set T8L	S/T8L	= FAS9 PH1 + . . .	
	C0-C31 —/→ D0-D31	DXC/6	= FAS9 PH1 + . . .	EW into D-register
	RR0-RR31 —/→ A0-A31	AXRR	= FAS9 PH1 + . . .	R into A-register
	Q15-Q31 —/→ P15-P31	PXQ	= FAS9 PH1 + . . .	
	Go to PH3	BRPH3	= FAS9 PH1 O4 + . . .	
PH3 T8L	End	ENDE	= FAS9 PH3 + . . .	Final phase
	PR0-PR31 → S0-S31	SXPR	= FUEOR PH3 + . . .	
	S0-S31 → RW0-RW31	RW	= FAS9 PH3 + . . .	Store EOR result into R
	S0-S31 —/→ A0-A31	AXS	= FAS9 PH3 + . . .	EOR result into A for test
	Set test for A	S/TESTA	= FAS9 PH3 + . . .	
PRE1 (next instruction)	Test contents of A-register	R/CC3	= TESTA	
		R/CC4	= TESTA	
	If A0 = 0 and A1-A31 > 0, set CC3	S/CC3	= NTESTA/1 TESTA NA0 NA0031Z	
	If A0 = 1, set CC4	S/CC4	= NTESTA/1 TESTA A0	

Mnemonic: EOR (48, C8)

Table 3-90. AND Word, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	MB0-MB31 → C0-C31	CXMB	= /DGC/	EW into C-register
	Set RQ	S/RQ	= PRE1 NIA NINDX PRERQ	
PH1 T4RL	C0-C31 ⇸ D0-D31	DXC/6	= FAS9 PH1 + . . .	EW into D-register
	RR0-RR31 ⇸ A0-A31	AXRR	= FAS9 PH1 + . . .	R into A-register
	Set NPRX	S/NPRX	= OU4 (O4 NO5 O6) + . . .	
	1's ⇸ CS0-CS31	CSX1	= (S/NPRX) + . . .	
	Q15-Q31 ⇸ P15-P31	PXQ	= PH1 PRERQ + . . .	
	Set DRQ	S/DRQ	= FAS9 PH1 + . . .	
	Set T8L	S/T8L	= FAS9 PH1 + . . .	
	Go to PH3	BRPH3	= FAS9 PH1 O4 + . . .	
PH3 T8L	End	ENDE	= FAS9 PH3 + . . .	Final phase
	PR0-PR31 → S0-S31	SXPR	= NPRX + . . .	
	S0-S31 → RW0-RW31	RW	= FAS9 PH3 + . . .	Store AND result into R
	S0-S31 ⇸ A0-A31	AXS	= FAS9 PH3 + . . .	AND result into A for test
	Set test for A	S/TESTA	= FAS9 PH3 + . . .	
PRE1 (next in-struc-tion)	Test contents of A register	R/CC3	= TESTA	
		R/CC4	= TESTA	
	If A0 = 0 and A1-A31 > 0, set CC3	S/CC3	= NTESTA/1 TESTA NA0 NA0031Z	
	If A0 = 1, set CC4	S/CC4	= NTESTA/1 TESTA A0	

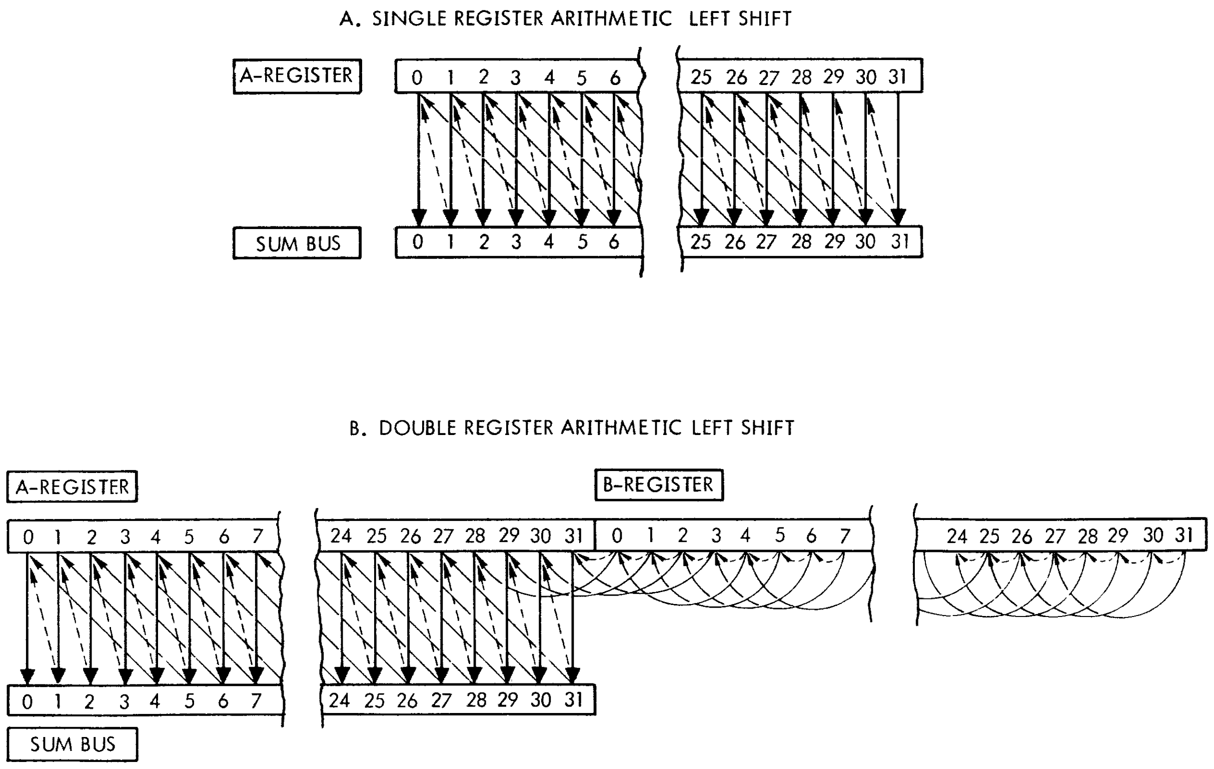
Mnemonic: AND (4B, CB)

Table 3-91. Shift Determination

Bit Positions			Type of Shift
21	22	23	
0	0	0	Single register logical
0	0	1	Double register logical
0	1	0	Single register circular (cyclic)
0	1	1	Double register circular (cyclic)
1	0	0	Single register arithmetic
1	0	1	Double register arithmetic
1	1	0	Not used
1	1	1	Not used

b. Right shift. In a single-register arithmetic right shift, the contents of the A-register are shifted right the number of bit positions indicated by the count. The sign bit contained in bit 0 is copied into the vacated bit positions. Bits shifted past A31 are lost. A single-register circular right shift is the same except that bits shifted past A31 are clocked into A0. In a single-register logical right shift, bits shifted past A31 are lost, and zeros are placed in the vacated bit positions. In a double-register arithmetic right shift, the sign bit is copied into the vacated bit positions in the A-register, bits shifted past A31 are clocked into B0, and bits shifted past B31 are lost. A double-register circular right shift is the same except that bits shifted past B31 are clocked into flip-flop A0. A double-register logical right shift is the same as a double-register arithmetic right shift except that zeros are clocked into the vacated bit positions in the A-register. During a right shift, the A-register is shifted right two bits at a time throughout the shift phase. At the end of the phase, if the shift count is odd, the A-register is shifted right one more bit position.

During a left shift operation, the bits are shifted one bit position at a time until flip-flops P30 and P31 contain zeros. From that point, the A-register is shifted left four bits at a time. Diagrams of single and double-register arithmetic left shift, arithmetic right shift, circular left shift, and circular right shift are shown in figures 3-180 through 3-183.



NOTE: LIGHT SOLID TRANSFER LINES INDICATE 4-BIT LEFT SHIFT WHEN NP30 NP31.
DASHED TRANSFER LINES INDICATE 1-BIT LEFT SHIFT WHEN P30 + P31

901060A. 3634

Figure 3-180. Arithmetic Left Shift, Flow Diagram

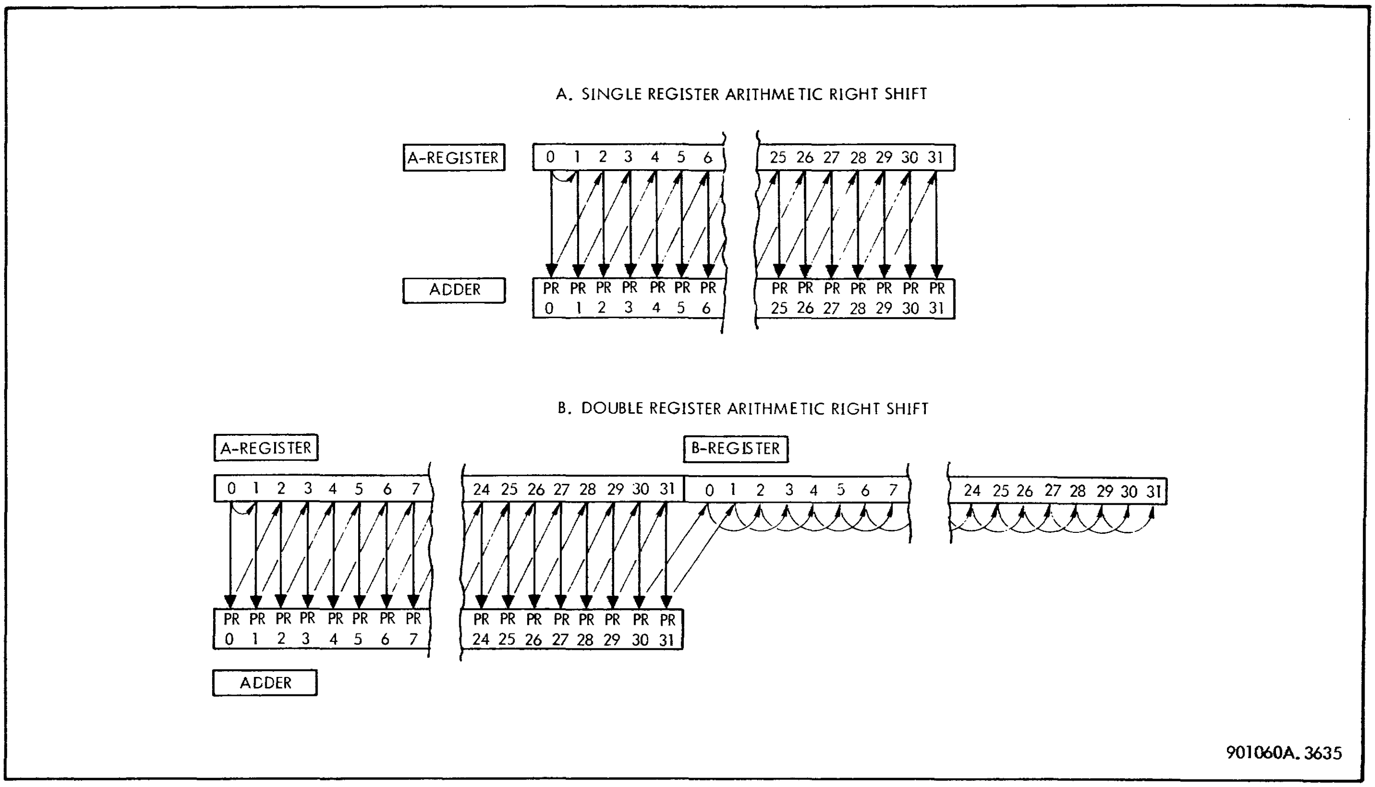
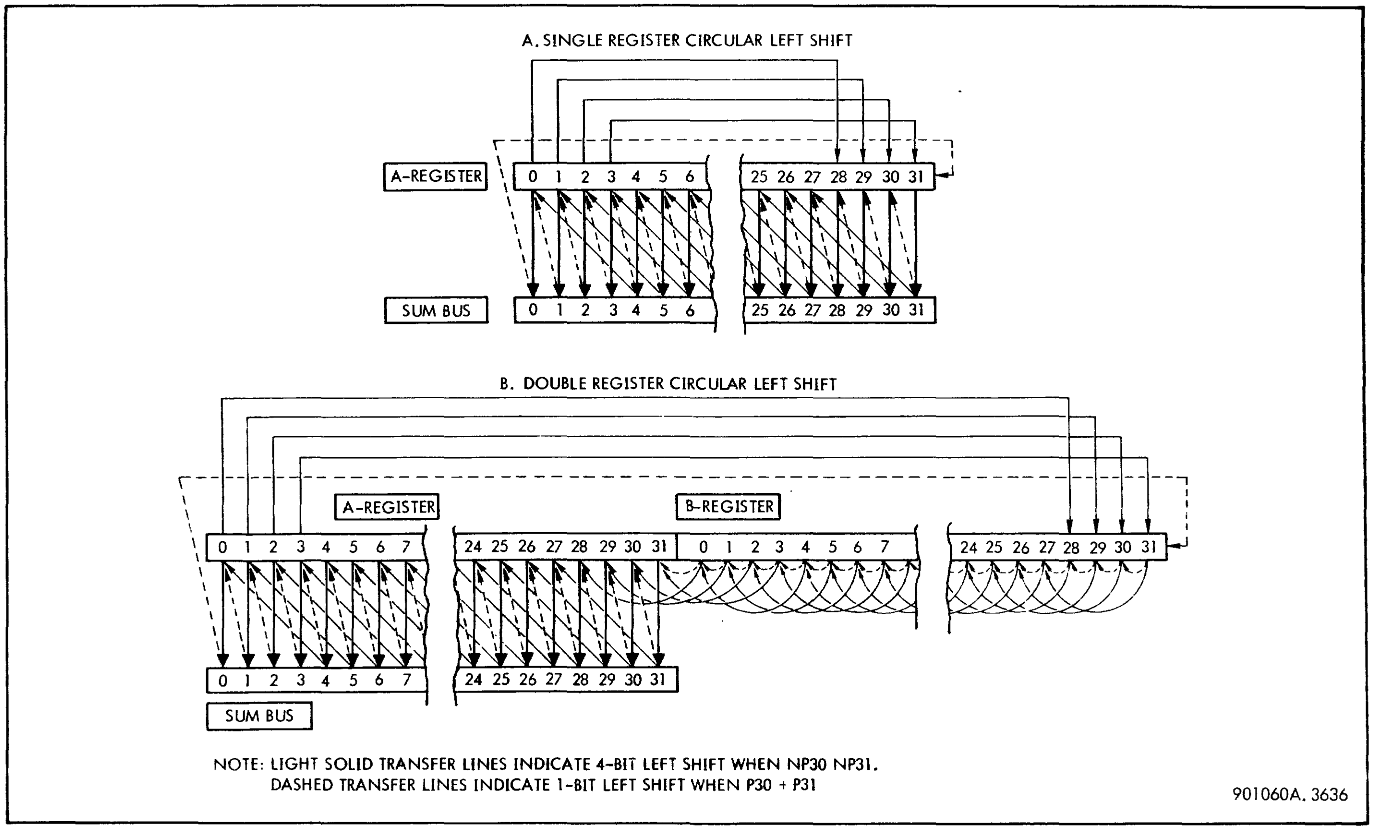


Figure 3-181. Arithmetic Right Shift, Flow Diagram



NOTE: LIGHT SOLID TRANSFER LINES INDICATE 4-BIT LEFT SHIFT WHEN NP30 NP31.
DASHED TRANSFER LINES INDICATE 1-BIT LEFT SHIFT WHEN P30 + P31

Figure 3-182. Circular Left Shift, Flow Diagram

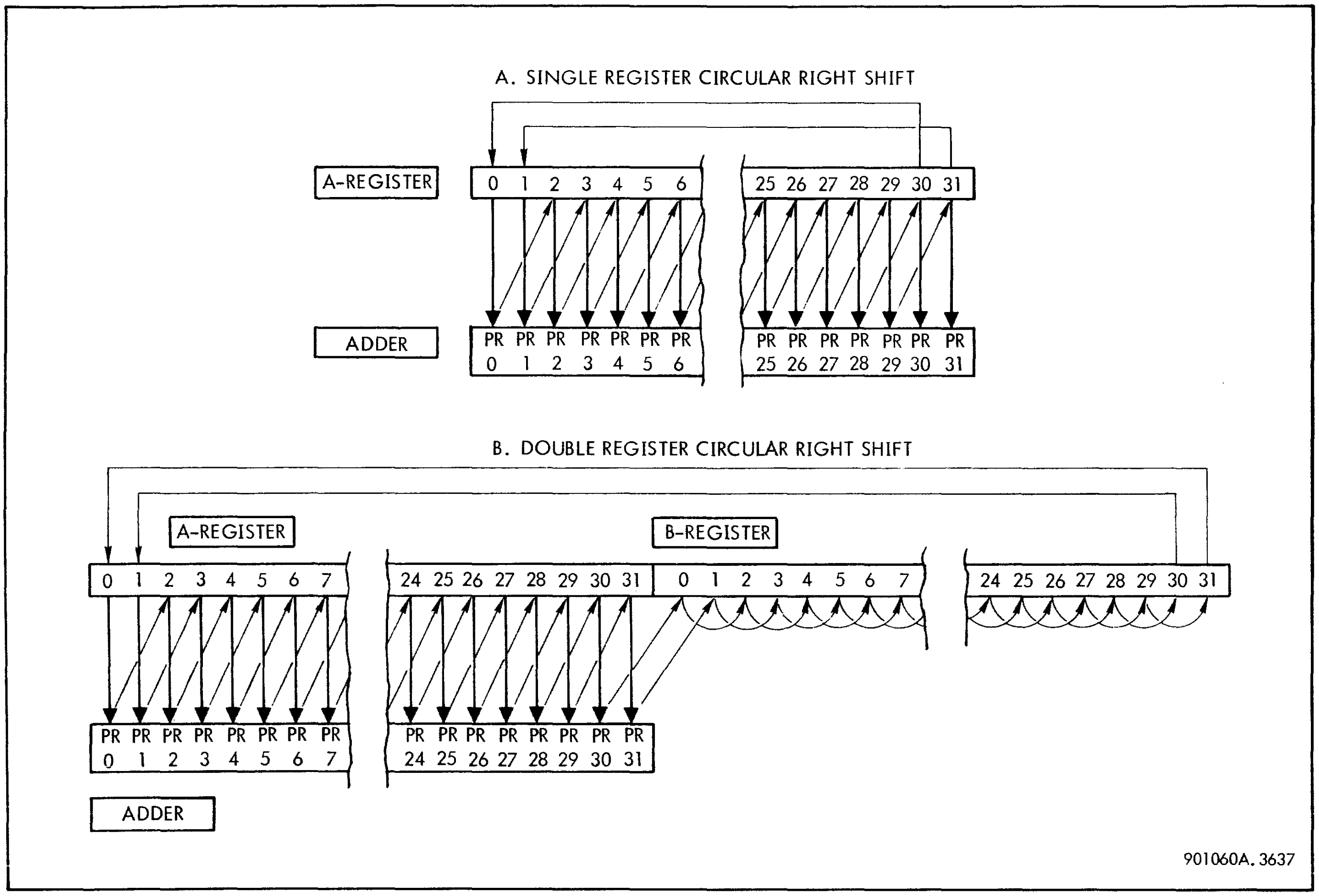


Figure 3-183. Circular Right Shift, Flow Diagram

3-222. SHIFT COUNT. In left shift operation, the P-register counts down by ones until P30 and P31 contain zeros and then counts down by fours, as shown in table 3-92. In a right shift, flip-flops P30 and P31 are not used in the count, but flip-flops P26 through P29 are counted up by one every other clock time until they all contain ones. Up-counting is used because the count is in the two-complement form. When flip-flops P26 through P29 contain ones, flip-flops P15 through P18 are counted down in one clock time from all zeros to all ones. At this time, flip-flop P31 is checked to determine if the shift is odd and a one-bit right shift should take place. The P-register count during a right shift count of 17 is shown in table 3-93.

After the shift is finished, the modified data in the A-register or in the A- and B-registers is loaded into the selected private memory register for a single-register shift, or into registers R and R + 1 for a double-register shift.

A sequence chart of the Shift instruction is given in table 3-94.

3-223 Shift Floating (SF 24, A4)

If either indirect addressing or indexing is called for in the instruction word, the address is computed as described under the shift instruction. Bit position 23 of the address determines the type of shift. If bit 23 is a zero, the contents of the addressed private memory register are treated as a short format floating point number. If bit 23 is a one, the contents of the even or odd numbered private memory registers are treated as a long format floating point number.

Table 3-92. Left Shift Count Sequence Example

P25	P26	P27	P28	P29	P30	P31	Decimal Count
0	0	1	1	0	0	1	+25
0	0	1	1	0	0	0	+24
0	0	1	0	1	0	0	+20
0	0	1	0	0	0	0	+16
0	0	0	1	1	0	0	+12
0	0	0	1	0	0	0	+ 8
0	0	0	0	1	0	0	+ 4
0	0	0	0	0	0	0	0

Table 3-93. Right Shift Count Example

SHPC	P25	P26	P27	P28	P29	P30	P31	Shift Action
1	1	1	0	1	1	1	1	Shift 2
0	1	1	1	0	0	1	1	Shift 2
1	1	1	1	0	0	1	1	Shift 2
0	1	1	1	0	1	1	1	Shift 2
1	1	1	1	0	1	1	1	Shift 2
0	1	1	1	1	0	1	1	Shift 2
1	1	1	1	1	0	1	1	Shift 2
0	1	1	1	1	1	1	1	Shift 2
1	1	1	1	1	1	1	1	Shift 1
			P15	P16	P17	P18		
			0	0	0	0		
			1	1	1	1		

The shift count, contained in bit positions 25 through 31 of the instruction word, determines the amount and direction of the shift. The shift count is treated as a 7-bit signed binary integer, with bit position 25 as the sign bit. Negative numbers are in two's complement form. The absolute value of the shift count determines the number of hexadecimal bit positions that the floating point number is to be shifted. If the shift count is positive, the floating point number is shifted to the left; if the count is negative, the number is shifted to the right.

The SF instruction loads the floating point number from the register specified by the R-field of the instruction word, if it is in the short format or from registers R and R + 1, if the instruction is in the long format, and loads into the A-register if the number is in the short format, or the A- and B-registers, if it is in the long format. If the number is negative, it is changed to a two's complement. A record of the original sign is retained. The floating point number is then separated into a characteristic and a fraction.

A positive shift count (NP25) produces the following left-shift operations:

a. If the fraction field is all zeros, the entire floating point number is set to all zeros (true zero) and flip-flop CC1 is set.

b. If the fraction is normalized (less than one and equal to or greater than 1/16), flip-flop CC1 is set.

c. If the fraction is not normalized, the fraction field is shifted four bit positions to the left, and the characteristic field is decreased by one. Vacated bit positions at the right are filled with zeros.

d. If the characteristic field underflows (that is, if it is all ones as the result of being down counted), flip-flop CC2 is set. If the characteristic field does not underflow, the shift process continues until the fraction is normalized, until underflow occurs, until the shift count is reached, or until 14 shifts have taken place.

e. At the completion of the left-shift operation, the floating point result is stored back in the private memory registers. If the number was originally negative, the two's complement of the result is stored in private memory.

A negative shift count produces the following right-shift operations:

a. The fraction field is shifted two digit positions to the right, and the characteristic field is increased by one. Vacated bit positions on the left are filled with zeros.

b. Flip-flop CC2 is set if the characteristic field overflows (that is, if it becomes all zeros from being counted up). If the characteristic field does not overflow, the shift process continues until the characteristic field overflows or until the shift count is reached.

c. If the resultant fraction field is all zeros, the entire floating point number is set to all zeros (true zero).

d. Upon completion of the right-shift operation, the floating point result is loaded back into the private memory registers. If the number was originally negative, the two's complement of the result is loaded into private memory.

The preparation sequence for an SF instruction is the same as the general preparation sequence except that, if the instruction is not indirectly addressed, the sign bit of the data word from private memory is stored in flip-flop RN. If bit C23 is a one, indicating a long floating point format, a one is forced on private memory address line LR31 to select the odd numbered private memory register.

If the odd numbered private memory register was addressed, the contents of the private memory register are clocked into the A-register. If a one was not forced on the LR31 address line, the contents of the private memory register addressed in the R-field of the instruction word are read into the A-register.

3-224 SHORT FORMAT. The data word, if positive, or the two's complement of the data word, if negative, is loaded into the A- and E-registers with the fraction in bits 8 through 31 of the A-register and the characteristic in the E-register. The shift count is loaded into flip-flops P25 through P31. If flip-flop P25 contains a zero, indicating a left shift, the data in the A-register is shifted four bit positions to the left at each clock time. While the shift count in bits 25 through 31 of the P-register is counted down by ones, the characteristic in the E-register is counted down by ones, and the limit count in P15 through P18 is counted up by ones.

Table 3-94. Shift, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PREP	Same as general preparation sequence except			
	During PRE1, force a one on private memory address line LR31	R/NLR31/2	= PRE1 NIA + PRE3 IA OU2 OL5 + . . .	Select odd numbered private memory register
	At PRE2 clock, RR0-RR31 —/→ A0-A31	AXRR	= FAMDSF PRE2 NIA + . . .	Data from odd numbered private memory register
PH1 T4RL	If bit C23 is a one, A0-A31 ——→ S0-S31	SXA	= FASHFX C23 PH1 + . . .	
	S0-S31 —/→ B0-B31	BXS	= FAMDSF PH1 + . . .	
		BX	= BXS	
	RR0-RR31 —/→ A0-A31	AXRR	= FASHFX PH1 + . . .	Data from even numbered private memory register
	Reset flip-flops CC1 and CC2	R/CC1	= FASH PH1 + . . .	
		R/CC2	= FAMDSF NFAMULH PH1 + . . .	
	Set interrupt enable flip-flop IEN	S/IEN	= FAMDSF NFAMUL PH1 + . . .	Allows instruction to be interrupted
	If P25 is a one, set flip-flop BWZ	S/BWZ	= FASH PH1 P25 + . . .	
	Force zeros into P15 through P22	PX/2	= FASH PH1 + . . .	
	If P30 is a one, set flip-flop SHPC	S/SHPC	= FASHFX PH1 P30 + . . .	
	Set flip-flop PH9	S/PH9	= BRPH9 = FASHFX PH1 + . . .	
	Enable clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
PH9	Single register shift			
	Set flip-flop CC2 if overflow	S/CC2	= FASHFX SFTL4 N(A0004Z + A0004W) + SFTL1 A0 NBWZ	
T6L	If P25 is a zero (NBWZ), generate signal SFTL	SFTL	= FASH PH9 NBWZ + . . .	Indicates left shift
	If bits P30 or P31 are ones, generate signal SFTL1	SFTL1	= FASHFX (NFASHFX + P30 + P31) SFTL + . . .	Shift one bit left until bits P30 and P31 are zeros

Mnemonic: S (25, A5)

(Continued)

Table 3-94. Shift, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH9 T6L (Cont.)	A0-A31 → S0-S31	SXA	= FASH PH9 + . . .	
	S1-S31 ↛ A0-A30	AXSL1	= SFTL1 + . . .	
	P-1 ↛ P	MCTP1	= SFTL NSHEX	Count down P by ones
	When bits P30 and P31 are not ones, generate signal SFTL4	SFTL4	= SFTL NSFTL1 NSHEX/1 + . . .	Shift four bits left until signal SHEX
		SFTL	= FASH PH9 NBWZ	
		SHEX/1	= FASH PH9 (P2629Z NP25 NP30 NP31) + . . .	
	A0-A31 → S0-S31	SXA	= FASH PH9 + . . .	
	S4-S31 ↛ A0-A27	AXSL4	= SFTL4 + . . .	
	P-4 ↛ P	MCTP2	= MCTP1 NP30 NP31	Count down P by fours
	Sustain PH9 until shifting completed	S/PH9	= BRPH9 = FASH PH9 NSHEX + . . .	
	If bit CC2 is a one, generate signal ALCYC	ALCYC	= FASHFX C22 NC23 + . . .	Indicates circular left shift
	A0 ↛ A31 if AXSL1 ALCYC	S/A31	= A31EN/2 AXSL1 + A31EN/1 AXSL4 + . . .	Single bit circular left
		⋮		
	A0-A3 ↛ A28-A31 if AXSL4 ALCYC	S/A28	= S29 AXSL1 + A28EN/1 AXSL4 + . . .	Four bit circular left shift
		A31EN/1	= A3 ALCYC + . . .	
		A31EN/2	= S0 ALCYC + . . .	
		A28EN/1	= A0 ALCYC + . . .	
	As bits are shifted left, toggle flip-flop CC1 if bit A0 is a one	S/CC1	= SFTPAR NCC1 + . . .	Check for odd or even number of ones
		R/CC1	= SFTPAR + . . .	
		SFTPAR	= SFTL4 FASHFX (A0 ⊕ A1 ⊕ A2 ⊕ A3) + SFTL1 A0 NBWZ + . . .	
	If bit P25 is a one, generate signal SFTR	SFTR	= FASH PH9 BWZ + . . .	Indicates right shift

Mnemonic: S (25, A5)

(Continued)

Table 3-94. Shift, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)	A0-A31 → PR0-PR31	PR0-PR31 = A0-A31 NCS0-NCS31 ND0-ND31	
	PR0-PR29 -/→ A2-A31	AXPRR2 = SFTR2 + . . .	Right shift data two bit places
		SFTR2 = SFTR NSHEX	
	A0 -/→ A0 and A1	A0EN/2, A1EN/2 = A0 FASHFX C21 + . . .	Extend sign bit in vacated bit positions
	(P26-P29) +1 -/→ P26-P29 until SHEX = 1	PCTP2 = FASHFX SHPC P25 + . . .	Add four to two's complement of shift count
		S/SHPC = SFTR2 NSHPC	
		SFTR2 = SFTR NSHEX	
	(P15-P18) -1 → P15-P18 when P19-P31 are all ones	MCTP5 = FASHFX SHPC P25 P2629W	Downcount P15-P18 from zeros to ones
	Shift left one bit position if odd shift count	SFTL1 = FASHFX P18 P31 SFTR + . . .	
	If bit CC2 is a one, generate signal ARCYC	ARCYC = ALCYC NANLZ	Right circular shift
	If ARCYC SFTR, A30 and A31 -/→ A0 and A1	S/A0 = A0EN/2 AXPRR2 + . . .	0 → vacated bit positions if logical right shift (C21 = 0)
		S/A1 = A1EN/2 AXPRR2 + . . .	
		A0EN/2 = A30 ARCYC + A0 FASHFX C21 + . . .	
		A1EN/2 = A31 ARCYC + A0 ND71 FASHFX C21 + . . .	
	Sustain PH9 until P15 through P17 are all ones	S/PH9 = BRPH9 = FASH PH9 NSHEX + . . .	Repeat shifting until signal SHEX is generated
	Generate signal SHEX	SHEX = P2629Z NP25 NP31 + P15 P16 P17 + . . .	End of shift
	Reset flip-flop IEN	R/IEN = SHEX + . . .	
	Generate memory request for next instruction	MRQ/1 = SHEX (NFASHFL + NC23) + . . .	

Mnemonic: S (25, A5)

(Continued)

Table 3-94. Shift, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)	Set flip-flop DRQ	S/DRQ = SHEX FASHFX NC23 + . . .	Inhibits transmission of another clock until data signal received
	Set flip-flop PH15	S/PH15 = SHEX FASHFX NC23 + . . .	
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + . . .	Address of next instruction
	P15-P31 → LM15-LB31	(NLMXC NLMXQ)	
	Enable clock T10L	S/T10L = SHEX FASHFX + . . .	
	Double register shift		
	If bit P25 is a zero, generate signal SFTL	SFTL = FASH PH9 NBWZ + . . .	Indicates left shift
	If bits P30 or P31 are ones, generate signal SFTL1	SFTL1 = FASHFX (NFASHFX + P30 + P31) SFTL + . . .	Shift one bit left until bits P30 and P31 are zeros
	A0-A31 → S0-S31	SXA = FASH PH9 + . . .	
	S1-S31 ⇸ A0-A30	AXSL1 = SFTL1 + . . .	
	B0 ⇸ A31	S/A31 = A31EN/2 AXSL1 + A31EN/1 AXSL4 + . . .	
		A31EN/2 = B0 FAMDSF + . . .	
		A31EN/1 = B3 FASH C23 + . . .	
	B1-B31 ⇸ B0-B30	BXBL1 = SFTL1	Shift B-register contents left one bit position
	When bits P30 and P31 are not ones, generate signal SFTL4	SFTL4 = SFTL NSFTL1 NSHEX/1 + . . .	
		SHEX/1 = P2629Z NP25 NP30 NP31 + . . .	
	A0-A31 → S0-S31	SXA = FASH PH9 + . . .	
	S4-S31 ⇸ A0-A27	AXSL4 = SFTL4 + . . .	Shift four bit positions left until signal SHEX/1
	B0-B3 ⇸ A28-A31	AXSL4 = SFTL4 + . . .	
	B4-B31 ⇸ B0-B27	BXBL4 = SFTL4 + . . .	

Mnemonic: S (25, A5)

(Continued)

Table 3-94. Shift, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH9 T6L (Cont.)	Sustain PH9 until bits P25 through P31 are zeros	S/PH9	= BRPH9 = FASH PH9 NSHEX + . . .	
	If bit CC2 is a one			
	S0 ⇸ B31	S/B31	= B31EN/1 BXBL1 + S3 BXBL4/1	Indicates circular left shift
		⋮		
	S0-S3 ⇸ B28-B31	S/B28	= B29 BXBL1 + S0 BXBL4/1	
		B31EN/1	= S0 FASHFX C22 C23	
		BXBL4/1	= FASHFX C22 C23	
	If bit P25 is a one, generate SFTR	SFTR	= FASH PH9 BWZ	Indicates right shift
	A0-A31 → PR0-PR31	PR0-PR31	= A0-A31 NCS0-NCS31 ND0-ND31 + . . .	Shift right two positions
	PR0-PR29 ⇸ A2-A31	AXPRR2	= SFTR2 + . . .	
	PR30-PR31 ⇸ B0 and B1	BXBR2	= FAMDSF SFTR2 + . . .	
	B0-B29 ⇸ B2-B31	BXBR2	= FAMDSF SFTR2 + . . .	
	A0 ⇸ A0, A1	A0EN/2, A1EN/2	= A0 FASHFX C21 + . . .	Sign → vacated positions, 0 → vacated positions if logical right shift (C21 = 0)
	Count P-register and shift left one extra bit position if odd shift count (see single-register right shift)			
	If bit CC2 is a one			
	B30 ⇸ A0, B31 ⇸ A1	A0EN/2, A1EN/2	= B30 B31 FASHFX CC2 CC3 + . . .	Right circular shift
	Sustain PH9 until P15 through P17 are all ones	S/PH9	= BRPH9 = FASH PH9 NSHEX + . . .	Repeat shifting until signal SHEX is generated
	Generate signal SHEX	SHEX	= P2629Z NP25 NP30 NP31 + P15 P16 P17	
	Reset flip-flop IEN	R/IEN	= SHEX + . . .	Terminate interrupt enable

Mnemonic: S (25, A5)

(Continued)

Table 3-94. Shift, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)	Generate memory request for next instruction	MRQ/1 = SHEX (NFASHFL + NC23) + . . .	
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	Address of next instruction
	P15-P31 → LM15-LB31	(NLMXC NLMXQ)	
	Set flip-flop PH14	S/PH14 = BRPH14 = SHEX FASHFX C23 + . . .	
	Force a one on address line LR31	R/NLR31/2 = SHEX FASHFX C23 + . . .	Selects odd numbered private memory register
	Enable clock T10L	S/T10L = SHEX FASHFX + . . .	
PH14	(Double register only)		
T10L	B0-B31 → S0-S31	SXB = FASHFX PH14 + . . .	
	S0-S31 → RW0-RW31	RWXS = FASHFX PH14 + . . .	
	Generate write byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	Store modified data in B-register in odd numbered private memory register
	Set flip-flop DRQ	S/DRQ = FAMDSF PH14 + . . .	Inhibits transmission of another clock until data signal received
	Set PH15	S/PH15 = PH14 NBR + . . .	
	Enable clock T10L	S/T10L = FAMDSF PH14 + . . .	
PH15	A0-A31 → S0-S31	SXPR = SXADD = FASH PH15 NRTZ + . . .	Modified contents of A-register
T10L	S0-S31 → RW0-RW31	RWXS = FAMDSF PH15 + . . .	
	Generate write byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	Modified contents of A-register stored in addressed private memory register
	Generate signal ENDE	ENDE = FAMDSF PH15 + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF) + . . .	
	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: S (25, A5)

Zeros are placed in the vacated bit positions at the least significant end of the A-register. Shifting stops when one of the following events takes place:

a. Zeros in P25 through P31 indicate that the shift count has been reached.

b. Zeros in bits 8 through 11 of the A-register indicate that the number has been normalized.

c. Ones in P15 through P17 of the limit counter indicate that 14 shifts have taken place, and that the number is shifted 56 bit positions.

d. A one in bit 0 of the E-register indicates characteristic underflow.

If flip-flop P25 contains a one, indicating a right shift, the data in the A-register is shifted two bit positions to the right at each clock time, and zeros are copied into the vacated bit positions at the most significant end of the A-register. The shift count in flip-flops P25 through P31, the limit count in flip-flops P15 through P18, and the characteristic in the E-register are counted up by one at every other clock time. Shifting is stopped by the same events as in left shift except that normalizing does not apply in the right shift, and a one in bit 0 of the E-register indicates characteristic overflow instead of underflow.

3-225 LONG FORMAT. The data words, if positive, or the two's complement of the data words, if negative, are loaded into the A-, B-, and E-registers with the most significant part of the fraction in bits 8 through 11 of the A-register, the remainder of the fraction in the B-register, and the characteristic in the E-register. The shift count is loaded into flip-flops P25 through P31, as in the short format operation. Shifting and counting takes place in the same manner as in short format operation, except that during left shift the data in flip-flops B0 through B3 is shifted into flip-flops A28 through A31, and during right shift, the data in flip-flops A30 and A31 is shifted into flip-flops B0 and B1. Vacated positions in the A-register are filled with zeros during right shift, and vacated positions in the B-register are filled with zeros during left shift.

3-226 SHORT AND LONG FORMAT. When shifting is complete, the shifted data or the two's complement of the shifted data, if originally negative, is loaded into the selected private memory register if it is in short format or into registers R and R + 1 if it is in long format.

Examples of left and right shift counts less than 14 and exceeding 14 are shown in tables 3-95 through 3-98.

A sequence chart of the Shift Floating instruction is given in table 3-99.

Table 3-95. Left Shift Count Less Than 14

P15	P16	P17	P18	Decimal Count	P26	P27	P28	P29	P30	P31	Decimal Count
0	0	0	0	+ 0	0	0	1	1	0	0	+12
0	0	0	1	+ 1	0	0	1	0	1	1	+11
0	0	1	0	+ 2	0	0	1	0	1	0	+10
0	0	1	1	+ 3	0	0	1	0	0	1	+ 9
0	1	0	0	+ 4	0	0	1	0	0	0	+ 8
0	1	0	1	+ 5	0	0	0	1	1	1	+ 7
0	1	1	0	+ 6	0	0	0	1	1	0	+ 6
0	1	1	1	+ 7	0	0	0	1	0	1	+ 5
1	0	0	0	+ 8	0	0	0	1	0	0	+ 4
1	0	0	1	+ 9	0	0	0	0	1	1	+ 3
1	0	1	0	+10	0	0	0	0	1	0	+ 2
1	0	1	1	+11	0	0	0	0	0	1	+ 1
1	1	0	0	+12	0	0	0	0	0	0	+ 0

Table 3-96. Left Shift Count Exceeding 14

P15	P16	P17	P18	Decimal Count	P26	P27	P28	P29	P30	P31	Decimal Count
0	0	0	0	+ 0	0	1	0	0	0	0	+16
0	0	0	1	+ 1	0	0	1	1	1	1	+15
0	0	1	0	+ 2	0	0	1	1	1	0	+14
0	0	1	1	+ 3	0	0	1	1	0	1	+13
0	1	0	0	+ 4	0	0	1	1	0	0	+12
0	1	0	1	+ 5	0	0	1	0	1	1	+11
0	1	1	0	+ 6	0	0	1	0	1	0	+10
0	1	1	1	+ 7	0	0	1	0	0	1	+ 9
1	0	0	0	+ 8	0	0	1	0	0	0	+ 8
1	0	0	1	+ 9	0	0	0	1	1	1	+ 7
1	0	1	0	+10	0	0	0	1	1	0	+ 6
1	0	1	1	+11	0	0	0	1	0	1	+ 5
1	1	0	0	+12	0	0	0	1	0	0	+ 4
1	1	0	1	+13	0	0	0	0	1	1	+ 3
1	1	1	1	+14	0	0	0	0	1	0	+ 2

Table 3-97. Right Shift Count Less Than 14

P15	P16	P17	P18	Decimal Count	P25	P26	P27	P28	P29	P30	P31	Decimal Count	SHPC
0	0	0	0	+ 0	1	1	1	0	1	0	1	-11	1
0	0	0	1	+ 1	1	1	1	0	1	1	0	-10	0
0	0	0	1	+ 1	1	1	1	0	1	1	0	-10	1
0	0	1	0	+ 2	1	1	1	0	1	1	1	- 9	0
0	0	1	0	+ 2	1	1	1	0	1	1	1	- 9	1
0	0	1	1	+ 3	1	1	1	1	0	0	0	- 8	0
0	0	1	1	+ 3	1	1	1	1	0	0	0	- 8	1
0	1	0	0	+ 4	1	1	1	1	0	0	1	- 7	0
0	1	0	0	+ 4	1	1	1	1	0	0	1	- 7	1
0	1	0	1	+ 5	1	1	1	1	0	1	0	- 6	0
0	1	0	1	+ 5	1	1	1	1	0	1	0	- 6	1
0	1	1	0	+ 6	1	1	1	1	0	1	1	- 5	0
0	1	1	0	+ 6	1	1	1	1	1	1	1	- 5	1
0	1	1	1	+ 7	1	1	1	1	1	0	0	- 4	0
0	1	1	1	+ 7	1	1	1	1	1	0	0	- 4	1
1	0	0	0	+ 8	1	1	1	1	1	0	1	- 3	0
1	0	0	0	+ 8	1	1	1	1	1	0	1	- 3	1
1	0	0	1	+ 9	1	1	1	1	1	1	0	- 2	0
1	0	0	1	+ 9	1	1	1	1	1	1	0	- 2	1
1	0	1	0	+10	1	1	1	1	1	1	1	- 1	0
1	0	1	0	+10	1	1	1	1	1	1	1	- 1	1
1	0	1	1	+11	0	0	0	0	0	0	0	- 0	0

Table 3-98. Right Shift Count Exceeding 14

P15	P16	P17	P18	Decimal Count	P25	P26	P27	P28	P29	P30	P31	Decimal Count	SHPC
0	0	0	0		1	1	1	0	0	0	1	-15	1
0	0	0	1	+ 1	1	1	1	0	0	1	0	-14	0
0	0	0	1	+ 1	1	1	1	0	0	1	0	-14	1
0	0	1	0	+ 2	1	1	1	0	0	1	1	-13	0
0	0	1	0	+ 2	1	1	1	0	0	1	1	-13	1
0	0	1	1	+ 3	1	1	1	0	1	0	0	-12	0
0	0	1	1	+ 3	1	1	1	0	1	0	0	-12	1
0	1	0	0	+ 4	1	1	1	0	1	0	1	-11	0
0	1	0	0	+ 4	1	1	1	0	1	0	1	-11	1
0	1	0	1	+ 5	1	1	1	0	1	1	0	-10	0
0	1	0	1	+ 5	1	1	1	0	1	1	0	-10	1
0	1	1	0	+ 6	1	1	1	0	1	1	1	- 9	0
0	1	1	0	+ 6	1	1	1	0	1	1	1	- 9	1
0	1	1	1	+ 7	1	1	1	1	0	0	0	- 8	0
0	1	1	1	+ 7	1	1	1	1	0	0	0	- 8	1
1	0	0	0	+ 8	1	1	1	1	0	0	1	- 7	0
1	0	0	0	+ 8	1	1	1	1	0	0	1	- 7	1
1	0	0	1	+ 9	1	1	1	1	0	1	0	- 6	0
1	0	0	1	+ 9	1	1	1	1	0	1	0	- 6	1
1	0	1	0	+10	1	1	1	1	0	1	1	- 5	0
1	0	1	0	+10	1	1	1	1	0	1	1	- 5	1
1	0	1	1	+11	1	1	1	1	1	0	0	- 4	0
1	0	1	1	+11	1	1	1	1	1	0	0	- 4	1
1	1	0	0	+12	1	1	1	1	1	0	1	- 3	0
1	1	0	0	+12	1	1	1	1	1	0	1	- 3	1
1	1	0	1	+13	1	1	1	1	1	1	0	- 2	0
1	1	0	1	+13	1	1	1	1	1	1	0	- 2	1
1	1	1	0	+14	1	1	1	1	1	1	1	- 1	0

Table 3-99. Shift Floating, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence in tables 3-24 through 3-26 except:		
	If the instruction is not indirectly addressed, during PRE2 the sign bit of the data word is stored in flip-flop RN	S/RN = RR0 RNXRR0 + . . .	Stores sign bit of floating point number. RN $\Rightarrow$ negative number
	If bit C23 is a one, force a one on address line LR31 to select odd numbered private memory register	R/NLR31/2 = LR31/2 = FASHFL PRE2 NIA C23 + . . .	
PH1	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FASHFL PH1 + . . .	Number to be shifted (from register R + 1 if long format)
	Reset condition code flip-flops CC1 and CC2	R/CC1 = FASH PH1 + . . . R/CC2 = FAMDSF NFAMULH PH1 + . . .	
	Set interrupt enable flip-flop IEN	S/IEN = FAMDSF NFAMULH PH1 + . . .	Allow interrupts
	If bit P25 is a one, set flip-flop BWZ	S/BWZ = FASH PH1 P25 + . . .	Negative shift count
	Clear flip-flops P15 through P22	PX/2 = FASH PH1 + . . .	Clears limit counter
	If flip-flop RN is set, set flip-flop NGX	S/NGX = FASHFL PH1 RN + . . .	Negative number
	Force ones into CS-register is signal (S/NGX) is generated	CSX1 = (S/NGX) + . . .	For two's complement operation
	Reset flip-flop K00H	R/K00H = NK00 (FASHFL PH1) + . . .	Generates signal K31 for two's complement operation
PH1 T4RL	If NC23, set flip-flop PH3	S/PH3 = BRPH3 = FASHFL PH1 NC23 + . . .	For short format
	If C23, set flip-flop PH2	S/PH2 = PH1 NBR N(FNANLZ NANLZ) + . . .	For long format

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL (Cont.)	If NC23, enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	For short format
	If C23, enable T6RL	T6RL = FASHFL PH1 C23 + . . .	For long format
PH2	(Long format only)		
T6RL	If NGX is true, A0-A31 ⊕ CS0-CS31 plus K31 → S0-S31	K31 = NGX (NK00H + NFASHFL) + . . .	Exclusive OR operation performed to obtain one's complement. K31 added to obtain two's complement
	CS0-CS31 plus K31 → S0-S31	SXK = SXADD + FASHFL PH2 + . . .	
	If GX is true, A0-A31 → S0-S31		
	S0-S31 -/→ B0-B31	BXS = FASHFL PH2 + . . .	
	If end carry occurs, reset K00H; otherwise, set K00H	S/K00H = S00 = NK00 N(FASHFL PH1) + . . .	Inverted end carry
		R/K00H = .	Inverted end carry
	If NGX is true, hold NGX set	S/NGX = FASHFL PH2 RN + . . .	To hold ones in CS-register for negation of remainder of word
	RR0-RR31 -/→ A0-A31	AXRR = FASHFL PH2 + . . .	From even numbered private memory register
	Set flip-flop PH3	S/PH3 = PH2 NBR N(FNANLZ ANLZ) + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH3 T6L	(Short and long format)		
	If NGX is true, A0-A31 ⊕		
	CS0-CS31 plus K31 if an end carry exists from PH2	SXK = SXADD = FASHFL PH3 + . . .	Obtain two's complement for short format operation. Continue obtaining two's complement if long format
	If signal GX is true, A0-A31 → S0-S31	SXK = SXADD = FASHFL PH3 + . . .	

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (Cont.)	S0-S7 —/→ E0-E7	EXS = FASHFL PH3 + . . .	Characteristic read into E-register
	S8-S31 —/→ A8-A31	AXS/3 = FASHFL PH3 + . . .	Fraction read into A-register
	Set flip-flop PH9	S/PH9 = BRPH9 = FASHFL PH3 + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH9	(Short and long format)		
T6L	If NC23 NP25, A0-A31 → S0-S31	SXA = FASH PH9 + . . .	Short format left shift
	S4-S31 —/→ A0-A27	AXSL4 = SFTL4 + . . .	Left shift four bit positions
		SFTL4 = SFTL NSFTL1 NSHEX + . . .	Zeros —/→ vacated positions
		SFTL = PH9 NBWZ + . . .	
	P26-P31 minus one, P15-P17 plus one, E0-E7 minus one	MCTP1 = SFTL NSHEX + . . .	Shift count decreased by one, limit count increased by one, and characteristic decreased by one. Limit counter limits shifts to 14. Signal SHEX is generated if 14 shifts take place and shift count is not reached
		MCTP2 = MCTP1 NP30 NP31	
		PCTP5 = FASHFL SFTL4 + . . .	
		MCTE1 = SFTL4 FASHFL + . . .	
		MCTE2 = MCTE1 NE4 NE5 (NE6 NE7 + NES4)	
	Hold flip-flop PH9 set	S/PH9 = BRPH9 = FASH PH9 NSHEX + . . .	
	If a one is detected in A8 through A11, generate signal SHEX	SHEX = NA0811Z NP25 FASHFL + . . .	Indicates number is normalized
	If a one is detected in flip-flop E0, generate signal SHEX	SHEX = FASH PH9 E0 + . . .	Indicates characteristic underflow

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)	If bits P25 through P31 are all zeros or P15 through P17 are all ones, generate signal SHEX	SHEX = FASH PH9 (P2629Z NP25 NP30 NP31 + P15 P16 P17) + . . .	Indicates count has been reached or register is all zeros
	If C23 NP25, A0-A31 → S0-S31	S/A28 = A28EN/1 AXSL4 + . . . ⋮	Long format left shift
		S/A31 = A31EN/1 AXSL4 + . . .	
		A28EN/1 = B0 FASH C23 + . . . ⋮	
		A31EN/1 = B3 FASH C23 + . . .	
		SXA = FASH PH9 + . . .	
	S4-S31 ⇸ A0-A27	AXSL4 = SFTL4 + . . .	Shift left four bit positions
	B0-B3 ⇸ A28-A31	BXBL4 = SFTL4 + . . .	Zeros ⇸ vacated bit positions in B-register
	B4-B27 ⇸ B0-B27		
	If NC23 P25		
	A0-A31 → PR0-PR31 PR0-PR29 ⇸ A2-A31	AXPRR2 = SFTR2 + . . .	Short format right shift. Shift right two bit positions. Zeros ⇸ vacated bit positions
		SFTR2 = SFTR NSHEX + . . .	
		SFTR = FASH PH9 BWZ + . . .	
	PR30, PR31 ⇸ B0, B1 B0-B29 ⇸ B2-B31	BXBR2 = SFTR2 + . . .	
	Toggle flip-flop SHPC	S/SHPC = SFTR2 NSHPC	Enable counting up shift count and characteristic every other clock time
		R/SHPC = •	
	(P25-P31) +1 ⇸ P25-P31	PCTP1 = SHPC FASHFL + . . .	Add one to two's complement of shift count
		PCTP2 = NPA33 P30 P31 + . . .	
		PCTP3 = PCTP2 P2629W	
	(P15-P18) +1 ⇸ P15-P18	PCTP5 = FASHFL SHPC + . . .	Add one to limit counter
	E + 1 ⇸ E	PCTE1 = FASHFL SHPC + . . .	Add one to characteristic

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)		PCTE2 = PCTE1 E4 E5 E6 E7 (NF EOF + NE2)	
	Hold flip-flop PH9 set until shift exit signal SHEX	S/PH9 = BRPH9 = FASH PH9 NSHEX + . . .	
	Generate signal SHEX		See short format left shift. E0 ⇒ characteristic overflow
	Reset flip-flop IEN	R/IEN = SHEX + . . .	Disable interruptibility
	If NC23, generate memory request for next instruction	MRQ/1 = SHEX (NFASHFL NC23) + . . .	Short format
	Q15-Q31 $\nrightarrow$ P15-P31	PXQ = MRQ/1 + . . .	Address of next instruction
	P15-P31 → LM15-LM31		Address → core memory address lines
	Set flip-flop PH14 if short format	S/PH14 = SHEX FASHFL NC23 + . . .	Branch to phase 14
	Set flip-flop PH10 if long format	S/PH10 = PH9 NBR + . . .	Go to phase 10
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set flip-flop CXS	S/CXS = SHEX FASHFL C23 + . . .	For use in phase 10
PH10	(Long format only)		
T6L	A0-A31 → S0-S31	SXA = FAMDSF PH10 + . . .	
	S0-S31 → C0-C31	CXS = SHEX FASHFL C23 + . . .	
	If NA0031Z, reset flip-flop RTZ	R/RTZ = FASHFL NA0031Z + . . .	NA0031Z ⇒ A-register not all zeros
	Set flip-flop PH11	S/PH11 = PH10 NBR + . . .	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH11	(Long format only)		
T6L	B0-B31 → S0-S31	SXB = FASHFL PH11 + . . .	
	S0-S31 $\nrightarrow$ A0-A31	AXS = FASHFL PH11 + . . .	

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH11 T6L (Cont.)	If original operand negative, set flip-flop NGX	S/NGX = FASHFL RN PH11 + . . .	For two's complement operation
	Reset flip-flop K00H to generate signal K31	R/K00H = .	
	Force a one onto private memory address line LR31	R/NLR31/2 = LR31/2 = FASHFL PH11 + . . .	To select odd numbered private memory register
	Set flip-flop PH12	S/PH12 = PH11 NBR + . . .	
	Enable clock T10L	S/T10L = FASH PH11 + . . .	
PH12	(Long format only)		
T10L	If GX is true, A0-A31 → S0-S31	SXK = FASHFL PH12 + . . .	GX ⇒ positive sign
	If NGX is true, A0-A31 ⊕ CS0-CS31 → S0-S31		Forms one's complement
	If NGX is true and NK00H is true from PH11, A0-A31 ⊕ CS0-CS31 plus K31 → S0-S31	SXK = SXADD = FASHFL PH12 + . . .	Forms two's complement
		K31 = N(K00H FASHFL) NGX + . . .	
	Set K00H unless an end carry occurs during the two's complement operation	S/K00H = S00	
	S0-S31 → RW0-RW31	RWXS = FASHFL PH12 + . . .	Store lower half of floating point word in odd numbered private memory register
	Generate write byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	
	If NA0031Z, reset flip-flop RTZ	R/RTZ = FASHFL NA0031Z + . . .	
		S/A0, A1 = A0EN/1, A1EN/1 AXE	
		A0EN/1 = E0 NFAMDSF	
		A1EN/1 = E1 NFAFL	
	E0-E7 $\not\longrightarrow$ A0-A7	AXE = FASHFL PH12 + . . .	Modified characterisitc
	C0-C31 $\not\longrightarrow$ D0-D31	DXC = FASHFL PH12 + . . .	Modified fraction. Bits D0 through D7 are zeros

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH12 T10L (Cont.)	Generate memory request for next instruction	MRQ/1 = FASHFL PH12 + . . .	
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	Address of next instruction
	Set flip-flop DRQ	S/DRQ = FASH PH12 + . . .	Inhibits transmission of another clock until data signal received
	Set flip-flop PH15	S/PH15 = BRPH15 = FASHFL PH12 + . . .	
	Enable clock T10L	S/T10L = FASH PH12 + . . .	
PH14	(Short format only)		
T6L	E0-E7 —/→ A0-A7	AXE = FASHFL PH14 + . . .	Characteristic clocked into A-register
	If NA0031Z, reset flip-flop RTZ	R/RTZ = FASHFL NA0031Z + . . .	
	If original operand was negative, set flip-flop NGX	S/NGX = FASHFL PH14 RN + . . .	Force one's into CS-register for two's complement
	Reset flip-flop K00H to generate K31	R/K00H = .	
	Set flip-flop DRQ	S/DRQ = FAMDSF PH14 + . . .	Inhibits transmission of another clock until data signal received
	Set flip-flop PH15	S/PH15 = PH14 NBR + . . .	
	Enable clock T10L	S/T10L = FAMDSF PH14 + . . .	
PH15	(Short and long format)		
T10L	If original operand was positive		
	A0-A31 ——→ S0-S31 (if short)		
	A0-A31 + D0-D31 ——→ S0-S31 (if long)	SXK = SXADD = FASH PH15 NRTZ + . . .	8-bit characteristic and 4-bit fraction. Modified 32-bit floating point word
	If original operand was negative		
	A0-A31 ⊕ CS0-CS31 plus K31 —/→ S0-S31	SXK = SXADD = FASH PH15 NRTZ + . . .	Two's complement for short format

Mnemonic: SF (24, A4)

(Continued)

Table 3-99. Shift Floating, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH15 T10L (Cont.)	A0-A31 ⊕ CS0-CS31 ⊕ D0-D31 plus K31 → S0-S31 if long format and an end carry exists			One's complement plus carry from PH12 for long format
	A0-A31 ⊕ D0-D31 ⊕ CS0-CS31 → S0-S31			One's complement for long format without carry
	S0-S31 → RW0-RW31	RWXS	= FAMDSF PH15 + . . .	Store floating point word in addressed private memory register
	Generate write byte signals RWB0-RWB3	RWB0-RWB3	= RWXS	
	S0-S31 $\not\rightarrow$ A0-A31	AXS	= FASHFL PH15 + . . .	To set condition code flip-flops
	Force a one into A31 if NRTZ	A31X1	= FASHFL PH15 NRTZ + . . .	Result of shift $\neq$ 0
	Set flip-flop TESTA	S/TESTA	= FAMDSF NFAHSFX PH15 + . . .	For condition code testing
	Generate signal ENDE	ENDE	= FAMDSF PH15 + . . .	
	Set flip-flop PRE1	S/PRE1	= ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
	If result of shift is zero or fraction was normalized, set flip-flop CC1	S/CC1	= FASHFL PH15 NBWZ RTZ + FASHFL SHEX NA0811Z + . . .	
	If characteristic under or overflow occurs, set flip-flop CC2	S/CC2	= FASHFL PH15 NRTZ E0 + . . .	
PRE1 next instruction	If the sign bit is positive, set flip-flop CC3; if the sign bit is negative, flip-flop CC4 is set	S/CC3	= TESTA NTESTA/1 NA0 NA0031Z + . . .	
		S/CC4	= NTESTA/1 TESTA A0 + . . .	

Mnemonic: SF (24, A4)

3-227 Convert by Addition (CVA 29, A9)

The CVA instruction converts positive binary-coded decimal (BCD) integers to their positive binary equivalents or converts positive binary fractions to their positive BCD equivalents. The value in R + 1 is the number converted, and the converted binary equivalent is stored into private memory register R. The effective address is the first address of a 32-word table that contains descending conversion values. Since all values are positive, bit 0 of each word is the most significant magnitude bit.

Conversion is carried out during PH3 of the CVA instruction by adding those values of the conversion table that correspond to the one-bits of the word to be converted. The B-register contains the number to be converted and B shifts end-around left each iteration of PH3. As each one-bit in B is detected, the corresponding table word value is read from memory and added to the A-register.

During the preparation sequence of the CVA instruction, the D- and E-registers are cleared to zeros, and LR31 is set for the transfer of R + 1 to the A-register in PH1. During all execution phases, the DRQ flip-flop is set. In PH2 the contents of the private memory register R + 1 are transferred to the A-register from the RR lines.

The number to be converted is placed on the sum bus and is then clocked into the B-register. The A- and C-registers are cleared to zeros and CC1 is reset. The CPU enters into PH3 and remains in PH3 for 34 iterations which are counted by the E-register. When the E-register reaches a count of 33, the CPU sequences from PH3 to PH4. During each iteration of PH3 except the last two, MRQ is enabled whenever the most significant bit (MSB) in the B-register is a one. This requests the appropriate conversion word to be read from memory so that it can be added to the A-register when the one-bit in B has been shifted to bit position 30.

After the conversion value has been transferred from C into the D-register, the contents of A and D are added. The sum is placed on the sum bus, and, if B30 is a one, it is transferred to A. CC1 will set if the sum bus generates an end carry during PH2. The SW1 signal is true during each PH3 iteration except the first. The P-register counts up by one each PH3 when SW1 is true. The B-register, which contains the value being converted during the PH3 iteration, shifts one bit position to the left in each iteration. At the clock of the final PH3 iteration (E = 33), MRQ/1 is enabled to request the next instruction.

During PH4 the converted number in the A-register is placed on the sum bus and at the next clock pulse is transferred to the RW lines to be stored in private memory R. In PH5 the A test flip-flop TESTA is set, and ENDE is enabled.

Condition code bits CC3 and CC4 are not affected until the clock at the end of the phase following PH5 of the CVA instruction. This is normally at the end of PRE1 of the next instruction, although it could also be during the first phase of an interrupt or trap operation or during phase PCP1 of a control panel operation.

A sequence chart of the Convert by Addition instruction is given in table 3-100.

The number to be converted is placed on the sum bus and is clocked into the B-register. The A- and C-registers are cleared to zeros, and CC1 is reset. The CPU enters into PH3 and remains in PH3 for 34 iterations counted by the E-register. When the E-register reaches a count of 33, the CPU sequences from PH3 to PH4.

3-228 Convert by Subtraction (CVS 28, A8)

The Convert by Subtraction instruction is normally used to convert positive binary integers to their positive BCD equivalents or positive BCD fractions to their positive binary equivalents. The R-field of the instruction points to the number to be converted. The effective address is the first address of a 32-word table that contains descending conversion values. Since all values are positive, bit 0 of each word is the most significant magnitude bit.

Conversion is carried out by successive subtractions of the table values from the number to be converted. If the result of the subtraction is a positive difference, a one-bit is placed in the least significant bit position of the B-register, the B-register contents are shifted one binary place to the left, and the difference is placed in the A-register for the succeeding subtraction. If the result of the subtraction is a negative difference, a zero bit is placed into the low order bit of the B-register, and the B-register contents are shifted one binary place to the left. The contents of A are unchanged.

The conversion phase lasts for 33 iterations to generate a 32-bit conversion (the first operation does not generate a bit into the B-register). Figure 3-184 is a flow chart describing the Convert by Subtraction instruction operation.

The normal preparation sequence is described under preparation sequence. In the execution sequence, the number to be converted is taken from the RR lines and is transferred to the A-register. The B-register is cleared to zeros, and a one is clocked into E7. The first operand request is made by MRQ, and PH3 is set.

During a CVS instruction, the CPU remains in PH3 for 33 iterations counted by the E-register. When the E-count reaches 33 (E2 E7), the CPU sequences to PH4. MRQ is enabled during each iteration except the last. At the final PH3 sequence, MRQ/1 is enabled. Also during each iteration, the P-register counts up by one to address the next operand of the conversion table.

During each iteration of PH3, the new memory table word in the C-register is one's complemented, and is clocked into the D-register. CS31 is set. The contents of the

Table 3-100. Convert by Addition, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	0's -/→ D0-D31	DX = ENDE + . . .	
	0's -/→ E0-E7	EX/1 = PRE1 + . . .	
	Set LR31/2	S/LR31/2 = FACV O7 (PRE2 NIA) + . . .	To address R + 1
PH1 T4RL	RR0-RR31 -/→ A0-A31	AXRR = FACV PH1 + . . .	Number to be converted → A-register
	0's -/→ B0-B31	BX/1 = FACV PH1 + . . .	
	Go to PH2	S/PH2 = PH1 NBR + . . .	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	A0-A31 → S0-S31	SXA = FACV PH2 + . . .	
	S0-S31 -/→ B0-B31	BXS = FACV PH2 + . . .	Number to be converted into B-register
	0's -/→ A0-A31	AX/1 = FACV PH2 + . . .	
	Reset CC1	R/CC1 = FACV PH2 + . . .	
	Set CX/1	S/CX/1 = FACV PH2 + . . .	Clear C-register
	Set T10L	S/T10L = FACV PH2 + . . .	
	Go to PH3	S/PH3 = PH2 NBR (FNANLZ NANLZ) + . . .	
PH3 T10L	Set SW1	S/SW1 = FACV PH3 + . . .	
	Shift B0-B31 left one place	BXBL1 = FACV PH3 + . . .	End-around shift
		S/B31 = B31EN BXBL1	
		B31EN = B0 NFAMDSF	
	E + 1 -/→ E	PCTE1 = FACV PH3 + . . .	
		PCTE2 = PCTE1 E4-E7	
	If B0 = 1, enable MRQ	MRQ = FACV PH3 NE2 B0 + . . .	Request conversion value, but not in last two iterations
	Set flip-flop DRQ	S/DRQ = FACV BRPH3 + . . .	
	C0-C31 -/→ D0-D31	DXC/1 = FACV PH3 O7 + . . .	

Mnemonic: CVA (29, A9)

(Continued)

Table 3-100. Convert by Addition, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T10L (Cont.)	A0-A31 + D0-D31 + CS0-CS31 $\longrightarrow$ S0-S31	SXADD = FACV PH3 + ...	CS always zero
	If B0 = 1, S0-S31 $\not\longrightarrow$ A0-A31	AXS = FACV PH3 O7 B30 + ...	
	If B30 = 1, K00 = 1; set CC1	S/CC1 = FACV PH3 O7 K00 + ...	Sum is greater than $2^{32} - 1$
	P + 1 $\not\longrightarrow$ P	PCTP1 = FACV PH3 SW1 N(E2 E7) + ...	
		PCTP2 = PCTP1	
	If E $\neq$ 33, remain in PH3	BRPH3 = FACV PH3 N(E2 E7) + ...	
	If E = 33, go to PH4	S/PH4 = PH3 NBR + ...	
	Enable MRQ/1	MRQ/1 = FACV PH3 E2 E7	
	Set T10L	S/T10L = FACV PH3 + ...	
PH4 T10L	A0-A31 $\longrightarrow$ S0-S31	SXA = FACV PH4 + ...	
	S0-S31 $\longrightarrow$ RW0-RW31	RW = FACV PH4 + ...	Converted number to R
	Set flip-flop DRQ for next instruction	S/DRQ = FACV PH4 + ...	
	Set T8L	S/T8L = FACV PH4 + ...	
	Go to PH5	S/PH5 = PH5 NBR + ...	
PH5 T8L	Set test for A	S/TESTA = FACV PH5 + ...	
	Enable ENDE	ENDE = FACV PH5 + ...	
PRE1 (next instruction)	Test contents of A	R/CC3 = TESTA + ...	
		R/CC4 = TESTA + ...	
		S/CC3 = NTESTA/1 TESTA NA0 NA0031Z + ...	If A0 $\neq$ 1, and A1-A31 < 0
		S/CC4 = NTESTA/1 TESTA A0 + ...	If A0 = 1

Mnemonic: CVA (29, A9)

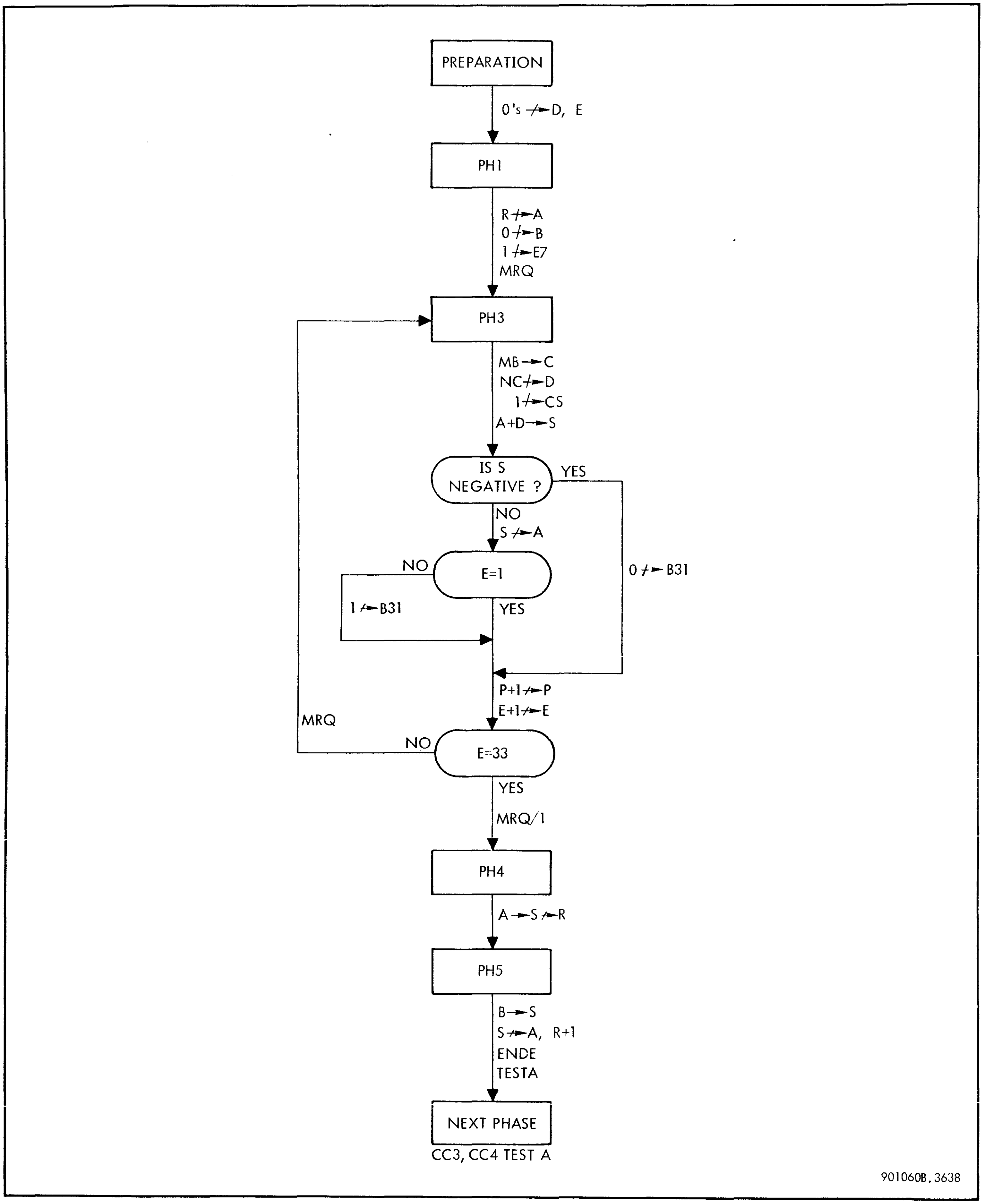


Figure 3-184. Convert by Subtraction, Logic Sequence Diagram

A- and D-registers are gated into the adder, and the sum is placed on the sum bus. If the sum is positive, a one is inserted into the low order bit of B. B is shifted one position to the left, and the contents of the sum bus are gated back into the A-register. If the sum is negative, a one is not inserted into the low order bit of B. B is shifted one position to the left, and the contents of the A-register are left unchanged.

In PH4 the remainder (if a remainder exists) in A is placed on the sum bus and is then transferred to the RW lines to be written into private memory register R. LR31 flip-flop is set to address R + 1 in PH5.

In PH5 the converted word in the B-register is placed on the sum bus and is clocked into the A-register for testing purposes. The converted word on the sum bus is also placed on the RW lines and is written into private memory register R + 1. The test for A flip-flop is set.

Condition code bits CC3 and CC4 are not affected until the clock at the end of the phase following PH5 of the CVS instruction. This is normally at the end of PRE1 of the next instruction, although it could also be during the first phase of an interrupt or a trap operation or during phase PCP1 of a control panel operation.

A sequence chart of the Convert by Subtraction instruction is given in table 3-101.

3-229 Move Byte String (MBS 61)

The MBS instruction is an immediate type instruction with a displacement value contained in bit positions 12 through 31. If the address in the R-field of the instruction word is an even numbered address, the contents of the even numbered private memory register are clocked into the A-register. This data word contains a source address in bit positions 13 through 31. The displacement number is added to this source address as the address where the data readout will begin.

A one is forced on the private memory address lines to select the odd numbered private memory register. The data word in the register contains a destination address in bit positions 13 through 31 and a byte count in bit positions 0 through 7.

The MBS instruction copies the contents of a byte string beginning with the modified source address into byte string locations beginning with the destination address. The number of bytes to be transferred are determined by the count; the bytes to be transferred are determined by the byte address in flip-flops P32 and P33. Each time a byte is transferred, the count is decreased by one. The byte source address and the byte destination address are increased by one.

If the MBS instruction is indirectly addressed, the instruction is treated as a nonexistent instruction. The computer aborts execution of the instruction at the time of opcode decoding and traps to location X'40'.

The MBS instruction word is clocked into the D-register at the clock following signal ENDE. The opcode portion is clocked into the O-register for decoding; the R-field is clocked into the R-register; and the address contained in the P-register is increased by one. If bit C0 is a one, indicating indirect addressing, signal FAILL is generated, causing the computer to trap to location X'40'.

During PRE1, signal PROTECTDIS is generated to inhibit trapping if an attempt is made to read out from or store into a protected area of core memory. At the PRE1 clock the A-, B- and E-registers are cleared, and the address of the next instruction contained in the P-register is clocked into the Q-register. Flip-flop PH1 is set to begin execution of the instruction. Flip-flop EXU is set, indicating the execution phases of the instruction, and clock T4RL is enabled.

During PH1, the address contained in flip-flops R28 through R31 is gated onto the LR lines to select the private memory register. At the PH1 clock, the contents of the addressed private memory register are clocked into the A-register. The displacement value contained in bits 12 through 31 of the C-register is clocked into the D-register, and the sign bit (bit 12) is extended 12 bit positions to the left to form a 32-bit data word. A one is forced onto private memory address line LR31 to select the odd numbered private memory register. Flip-flop DRQ is set, but the clock transmission is not affected because a memory request has not been initiated. At the PH1 clock, flip-flop PH2 is set, and clock T6RL is enabled.

During PH2, the source address contained in the A-register and the displacement contained in the D-register are gated into the adder, and an addition operation is performed. The result is gated onto the sum bus and is clocked into the B-register at the PH2 clock. When the data from the odd numbered private memory register is available, clock PH2 clocks the data into the A-register. The data word from the odd numbered register contains the count in bit positions 0 through 7 and the destination address in bit positions 13 through 31. Bit positions 30 and 31 contain the byte selection data. A one is forced on private memory address line LR31 to select the odd numbered private memory register for use during PH3. Flip-flop DRQ is set, but clock transmission is not inhibited since a memory request has not been generated. The instruction sequences to PH3, and clock T10L is enabled.

The 32-bit displacement value contained in the D-register is gated onto the sum bus during PH3 and is then gated onto private memory data lines RW0 through RW31. Write-byte signals RWB0 through RWB3 are generated, and the data word is stored in the odd numbered private memory register. At the PH3 clock, the D-register is cleared, and flip-flop DRQ is set. Since no memory request has been generated, transmission of the clock is not inhibited by flip-flop DRQ.

Table 3-101. Convert by Subtraction, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
ENDP	0's ⇸ D0-D31	DX = ENDE + ...	
	0's ⇸ E0-E7	EX/1 = PRE1 + ...	
PH1 T4RL	RR0-RR31 ⇸ A0-A31	AXRR = FACV PH1 + ...	Number to be converted into A-register
	0's ⇸ B	BX/1 = FUCS PH1 + ...	Clear B-register
	1 ⇸ E7	EX7X1 = FUCS PH1 + ...	Set one count into E
	Enable MRQ	MRQ = FUCVS PH1 + ...	Requests first operand conversion value
	Set flip-flop DRQ	S/DRQ = FACV BRPH3 + ...	Inhibits clock until data release received from memory
	Set T10L	S/T10L = FUCS PH1 + ...	
	Go to PH3	S/PH3 = PH1 BRPH3 + ...	
		BRPH3 = FUCVS PH1 + ...	
PH3 T10L	NC0-NC31 ⇸ D0-D31	NDXNC = FUCVS PH3 + ...	One's complement of EW into D-register
	1 ⇸ CS31	CX1/8 = FUCVS PH3 + ...	For two's complement
	A0-A31 + D0-D31 + CS1→ S0-S31	SXADD = FACV PH3 + ...	Private memory word (R)-(EW) 2 -effective word
	S0-S31 ⇸ A0-A31	AXS = FACV PH3 K00 N(E + 1) + ...	Difference into A if not negative
	If K00, 1 ⇸ B31	B31X1 = FUCVS PH3 N(E = 1) K00 + ...	But not if first time in PH3
	Shift B left	BXBL1 = FACV PH3 + ...	
	P + 1 ⇸ P	PCTP1 = FUCVS PH3 N(E2 E7) + ...	Increase memory address by one
		PCTP2 = PCTP1	
	E + 1 ⇸ E	PCTE1 = FACV PH3 + ...	Add one to the E-register
		PCTE2 = PCTE1 E4 E5 E6 E7 + ...	
	Set T10L	S/T10L = FACV PH3 + ...	
	Enable MRQ/1	MRQ/1 = FACV PH3 (E2 E7) + ...	Request next instruction, final iteration

Mnemonic: CVS (28, A8)

(Continued)

Table 3-101. Convert by Subtraction, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T10L (Cont.)	If E ≠ 33, remain in PH3	S/PH3 = BRPH3 NCLEAR + ...	
		BRPH3 = FACV PH3 N(E2 E7) + ...	
	If E = 33, go to PH4	S/PH4 = PH3 NBR + ...	
		NBR = NBRPH1 NBRPH2 NBRPH3	
	Enable MRQ	MRQ = FUCVS PH3 NE2 + ...	Request next conversion value
	Set flip-flop DRQ for operand	S/DRQ = FACV BRPH3	Inhibits clock until data release received from memory
PH4 T10L	A0-A31 → S0-S31	SXA = FACV PH4 + ...	Transfer remainder into private memory register
	S0-S31 → RW0-RW31	RW = FACV PH4 + ...	
	Set LR31/2	S/LR31/2 = FUCVS PH4 + ...	Private memory register to address R + 1
	Set flip-flop DRQ for next instruction	S/DRQ = FACU PH4 + ...	Inhibits clock until data release received from memory
	Enable T8L	S/T8L = FACV PH4 + ...	
	Go to PH5	S/PH5 = FACV PH4 + ...	
PH5 T8L	B0-B31 → S0-S31	SXB = FUCVS PH5 + ...	Transfer converted number to R + 1 and to A for test in next phase
	S0-S31 ↛ A0-A31	AXS = FUCVS PH5 + ...	
	S0-S31 → RW0-RW31	RRWXS = FUCVS PH5 + ...	
	Set A test	S/TESTA = FACV PH5 + ...	
	ENDE	ENDE = FACV PH5 + ...	
PRE1 (next instruction)	Test A	R/CC3 = TESTA + ...	
		R/CC4 = TESTA + ...	
		S/CC3 = NTESTA/1 TESTA NA0 NA0031Z + ...	If remainder is not zero
		S/CC4 = NTESTA/1 TESTA A0 + ..	If remainder is negative

Mnemonic: CVS (28, A8)

Flip-flop PH5 is set at the PH3 clock, and clock T8L is enabled.

The byte address is contained in bits 30 and 31 of both the source address and the destination address. During PH5, both the source address contained in the B-register and the destination address contained in the A-register are circular right shifted two bit positions. The operation places the desired core memory addresses in bit positions 15 through 31 and the byte selection bits in bit positions 0 and 1. If flip-flop PHA is not set, flip-flop PH6 is set, and clock T8L is enabled. Flip-flop DRQ is set, but clock transmission is not inhibited since a memory request has not been generated.

The modified source address and byte selection bits contained in the B-register are gated onto the sum bus. At the PH6 clock, the source address is clocked from the sum bus into the P-register, and byte selection bits S0 and S1 are clocked into flip-flops P32 and P33. The source address is gated from the P-register onto core memory address lines LM15 through LB31.

Word mode flip-flop SW1 is set if the byte selection bits in the source address (B0 B1) and the destination address (A0 A1) are all zeros and if the count in the E-register is four or larger, which indicates that there is to be a transfer of at least one word (four bytes).

If signal SW1 is generated, which indicates a word mode, and if the R-field is not all zeros, a memory request is generated for the first word from the address contained on the LM and the LB lines. Flip-flop DRQ is set. This inhibits transmission of another clock until the data release signal is received. Word mode flip-flop SW1 is reset at the PH6 clock if the set input is not true; the instruction sequences to PH7; and clock T6L is enabled.

The first word is gated into the C-register when it is available from core memory. If signal SW1 is true, indicating a full word, bits C0 through C31 are clocked into flip-flops D0 through D31. If signal NSW1 is true, flip-flops P32 and P33 are read to determine the byte selections. If both flip-flop P32 and flip-flop P33 are zeros, signal DXCR24 is generated, which gates bits C0 through C7 into flip-flops D24 through D31. If flip-flop P32 is a zero and P33 is a one, signal DXCR16/1 is generated, gating bits C8 through C15 into flip-flops D24 through D31. Signal DXCR8 is generated, causing bits C16 through C23 to be clocked into flip-flop D24 through D31 if flip-flop P32 is a one and P33 is a zero. Bits C24 through C31 are clocked into flip-flops D24 through D31 if flip-flops P32 and P33 are both ones.

A word transfer is indicated, if signal SW1 is true, and the contents of the P-register are increased by one count. If signal NSW1 is true, indicating a byte transfer, a one is added to P33 to increase the byte address by one. When the byte address has been counted to three, a one is added to the source address in the P-register.

If the memory map and the memory protection options are installed and if the modified source address points to a protected area of core memory, flip-flop SW2 is set, indicating that a memory protection violation has occurred. Flip-flop PH8 is set, and clock T4L is enabled.

Since the first word or byte to be transferred is now in the D-register, the modified source address and byte selection contained in flip-flops P15 through P33 are gated onto the sum bus. Bits P32 and P33 are gated onto bit positions S0 and S1. At the PH8 clock, the contents of the sum bus are clocked into the B-register. Flip-flop DRQ is set, but clock transmission is not inhibited since a memory request has not been generated. The instruction sequences to PH10, and clock T4L is enabled.

The destination address stored in the A-register is gated onto the sum bus and is then clocked into the P-register. The address is gated onto the LM and the LB address lines to select the memory location for storage of the first word or byte. The byte selection bits contained in bit locations 0 and 1 are clocked into flip-flops P32 and P33. At the PH9 clock, the A-register is cleared. If signal NSW2 is true, indicating that the source address did not point to a protected area of memory, a memory request is generated to store the word or byte in the location pointed by the destination address. Flip-flop NPRX is set for use during PH10 if a byte transfer is indicated by SW1. Signal NPRX forces ones into the CS-register at the PH9 clock. Flip-flop PH10 is set, and clock T6L is enabled.

During PH10, storage of the first word or byte is performed. A word mode is indicated if signal SW1 is true, and the contents of the D-register are gated onto the sum bus and then onto data lines MB0 through MB31. Write-byte signals MW0 through MW3 are generated; WRITE signal is generated; and the first word is stored in the core memory location pointed by the destination address.

A byte selection is indicated if signal NSW1 is true, and the byte previously clocked into flip-flops D24 through D31 is gated onto the sum bus by means of the adder. Bits D24 through D31 are gated into positions S0 through S7, S8 through S15, S16 through S23, and S24 through S31. The byte selected by flip-flops P32 and P33 is gated onto the applicable MB0 through MB31 data lines. The appropriate write-byte signal is generated; WRITE signal is generated; and the byte is stored in the byte location pointed by the destination address.

After a word has been stored in core memory, the byte count in the E-register is decreased by four, and the destination address in the P-register is increased by one. If a byte has been stored in core memory, the byte count in the E-register is decreased by one, and the byte address in flip-flops P32 and P33 is increased by one for the next byte. The destination address is not increased until byte 3 has been transferred.

If the destination address points to a protected area of memory, signal PROTECTD is generated. The word or byte is not stored, and the destination address is not counted up. As the trap function is disabled by signal PROTECTDIS, flip-flop SW2 is set to indicate an attempted memory protection violation. The instruction sequences to PH13, and clock T6L is enabled.

After the destination address and byte selection bits have been modified, they are gated onto the sum bus. Bits P15 through P31 are gated onto S15 through S31 and P32 and P33 are gated onto S0 and S1. The modified destination address and byte address are clocked into the A-register of the PH13 clock. If a memory protection violation has not occurred and the byte count has not been counted down to zero, the instruction branches back to PH6 and the operation is repeated until either flip-flop SW2 is set, indicating a memory violation, or the byte count is zero.

During the shift of the destination address, the count contained in bits 0 through 7 is clocked into flip-flops E0 through E7. If the byte count is not all zeros, the word mode flip-flop SW1 is set. As the memory protection trapping function is disabled, flip-flop SW2 is reset to allow an indication when a memory protection violation occurs. The instruction branches to PH6A if the byte count in bit positions A0 through A7 is all zeros, or if an interrupt has occurred, flip-flop PHA is set. If SW2 is set, which indicates a memory protection violation, an interrupt occurs or the byte count is counted down to zero. Flip-flop PHA is set; the instruction branches to PH6A; and clock T8L is enabled.

The modified destination address contained in the A-register is gated onto the sum bus and is shifted left one bit position. Bit A0 is clocked into flip-flop A31. At the same PH6A clock, the source address contained in the B-register is shifted one bit position to the left, with bit B0 clocked into flip-flop B31. A one is forced onto private memory address line LR31 to select the odd numbered private memory register. This register contains the displacement value that was stored in it during PH3. Signal CXRR is generated to gate the data word into the C-register when it is available on the RR lines. The instruction sequences to PH7, and clock T4RL is enabled.

During PH7A, the destination address contained in the A-register is gated onto the sum bus and is shifted one bit position to the left into the A-register. Bit A0 is clocked into flip-flop A31. The destination address and the byte address are then contained in flip-flops A13 through A31. The modified byte count contained in the E-register is clocked into flip-flops A0 through A7. A one is forced on private memory address line LR31/2 to select the odd numbered private memory register. When the data is available from private memory, signal CXRR gates it into the C-register. The instruction sequences to PH8A, and clock T8L is enabled.

During PH8A, the modified destination address and the byte count contained in the A-Register are gated onto the sum bus and then onto private memory data lines RW0 through RW31. Write-byte signals RWB0 through RWB3 are generated, and the data word is stored in the odd numbered private memory register. Flip-flop PH9 is set, and clock T8L is enabled.

During PH9, the modified source address contained in the B-register is gated onto the sum bus and is shifted one bit position to the left and is clocked into the A-register. Bit S0 is clocked into flip-flop A31. The source address and the byte selection address are contained in flip-flops A13 through A31 at the PH9 clock, and the displacement value contained in the C-register is inverted and is clocked into the D-register.

A one is forced into flip-flop CS31, if signals EZ and NSW2 are true, indicating that the byte count was counted down to zero and a memory protection violation did not occur. The remainder of the CS-register is cleared. A memory request is generated for the next instruction. When memory request MRQ/1 is generated, the address of the next instruction is clocked from the Q-register into the P-register and is gated into the LM and the LB core memory address lines. Flip-flop PH10 is set for the next phase, and clock T6L is enabled.

During PH10, the one's complement of the displacement value contained in the D-register, the source address contained in the A-register, and the contents of the CS-register are gated into the adder. An addition operation is performed the result of which is the source address minus the displacement value. The result is gated onto the sum bus and is clocked into the A-register. The D-register is cleared at the PH10 clock. Flip-flop DRQ is set which inhibits transmission of another clock until the data is received from memory.

One's are forced into flip-flops CS0 through CS29 (-4) if the transfer operations were in the word mode (SW1), and if a memory protection violation occurred (SW2). Interrupt enable flip-flop IEN is set which allows interrupts to occur. Flip-flop PH11 is set, and clock T10L is enabled.

If bit R31 is not a one, but if there are ones in R28, R29, or R30, the source address contained in the A-register and the contents of the D- and CS-registers are gated into the adder, and an addition operation is performed. The D-register contains all zeros because it was cleared at the PH10 clock. The CS-register contains a -4 if the transfer operation was a word mode and a memory protection violation occurred. Adding this value to the source address causes the source address to point to the address of the last byte transferred before the violation occurred.

The result of the addition operation is gated onto the sum bus and then onto private memory data lines RW0 through

RW31. Write-byte signals RWB0 through RWB3 are generated, and the source address is stored in the even numbered private memory register. If the transfer operation was completed and the count in the E-register was all zeros, signal ENDE is generated. Flip-flop PRE1 is set, and clock T6L is enabled.

Signal ENDE is not generated if a memory protection violation occurred (SW2 is true). Signal (S/TRACC4/1) is generated, and the trap flip-flop is set which causes entrance into the trap sequence.

A sequence chart of the Move Byte String instruction is given in table 3-102.

3-230 Compare Byte String (CBS 60)

The CBS instruction compares the contents of the source byte string with the contents of the destination byte string, byte by corresponding byte, beginning with the first byte ofeach string. The comparison continues until the specified number of bytes has been compared or until an inequality is found.

When the CBS instruction terminates, flip-flops CC3 and CC4 are set to indicate the result of the last comparison. Flip-flop CC4 is set if the source byte string is less than the destination byte string. Flip-flop CC3 is set, if the source byte string is greater than the destination byte string. If the CBS instruction terminates because of inequality, the byte count is one greater than the number of bytes remaining to be compared. The source address and the destination address indicate the locations of the unequal bytes.

If the CBS instruction is indirectly addressed, it is treated as a nonexistent instruction, and the computer aborts execution of the instruction at the time of opcode decoding and traps to location X'40'.

The preparation phase and phases PH1 through PH3 are the same as the sequences given for the Move Byte String instruction except that, during PH3, condition code flip-flops CC3 and CC4 are reset.

Phases PH5 through PH9 are the same as PH5 through PH9 of the Move Byte String instruction except that word mode flip-flop SW1 is not set during PH6 and, during PH9, signal (S/NGX) is generated instead of signal NPRX. Signal (S/NGX) forces ones into the CS-register and forces a one into K31 for the two's complement operation to be performed during PH10. During PH10, the source byte contained in the D-register, the ones in the CS-register, and the one in K31 are gated into the adder, and a two's complement operation is performed. The result is gated onto the sum bus and, at the PH10 clock, is clocked into the A-register.

When the byte from the destination address location is available on the MB data lines, signal CXMB gates the data into the C-register. At the PH10 clock, the selected byte is downward aligned into flip-flops D24 through D31. Flip-flop SW2 is set if a memory protection violation occurs. The instruction sequences to PH11, and clock T6L is enabled.

During PH11, the two's complement of the source byte contained in the A-register and the destination byte contained in the D-register are gated into the adder, and an addition operation is performed. The result is gated onto the sum bus and is clocked into the A-register at the PH11 clock. The instruction sequences to PH12, and clock T8L is enabled.

If the source byte is larger than the destination byte, flip-flop A0 is a one, and flip-flop CC3 is set. If the source byte is smaller than the destination byte, flip-flop A0 is zero, but the A-register does not contain all zeros, and flip-flop CC4 is set. If the two bytes are equal, the contents of the A-register are all zeros and flip-flops CC3 and CC4 remain reset. If the contents of the A-register are all zeros, indicating the bytes were equal, the byte count in the E-register is decreased by one and the byte address in flip-flops P32 and P33 is increased by one. Flip-flop PH13 is set, and clock T6L is enabled. If the bytes do not compare, the count in the E-register and the byte address in the P-register remain unchanged.

Operation during phases PH13 through PH11A is the same as the operation described for the Move Byte String instruction, except that if, during PH13, the bytes compared and the E-register are not all zeros, flip-flop PHA is not set. The sequence repeats from PH6 through PH13 until unequal bytes occur (NA0031Z) or the count in the E-register is counted down to zeros. Flip-flop PHA is then set, and the operation proceeds as described in the Move Byte String description.

During PH11A, signal ENDE is generated if flip-flops CC3 or CC4 have been set or if the E-register has been counted down to all zeros. If a trap does not occur, flip-flop PRE1 is set and clock T6L is enabled.

Signal ENDE is generated if the transfer operation is completed and the count in the E-register is all zeros. Flip-flop PRE1 is set, and clock T6L is enabled.

If a memory protection violation occurred (SW2 is true), signal ENDE is not generated. Signal (S/TRACC4/1) is generated, and the trap flip-flop is set which causes entrance into the trap sequence.

A sequence chart of the Compare Byte String instruction is given in table 3-103.

3-231 Translate Byte String (TBS 41)

The TBS instruction replaces each byte of the destination byte string with a source byte located in a translation table. The destination byte string begins with the byte locations pointed by the destination address in an odd numbered private memory register.

Table 3-102. Move Byte String, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1	P15-P31 $\not\longrightarrow$ Q15-Q31	QXP = PRE1 NANLZ + . . .	Next instruction address
	Reset A- and E-registers	AX/1 = PRE1 + . . .	
		EX/1 = PRE1 + . . .	
	Reset B-register	BX/1 = PRE1 NINTRAPF + . . .	
	Generate signal PROTECTDIS	PROTECTDIS = FAB0 + . . .	Disables trapping feature of memory protection
	Set flip-flop PH1	S/PH1 = (S/PH1/1) NCLEAR NBR + . . .	
		(S/PH1/1) = PREIM PRE1 + . . .	
	Enable clock T4RL	T4RL = PREP	
	If bit C0 is a one, generate signal FAILL and set flip-flop TRAP	FAILL = IA N03 (N04 N05) + . . .	
		S/TRAP = FAILL (PRE1 NANLZ) + . . .	
	Generate signal EXU	S/EXU = PREIM PRE1 NCLEAR + . . .	
PH1 T4RL	C12-C31 $\not\longrightarrow$ D12-D31	DXC/4 = FABS PH1 + . . .	Displacement value clocked into D-register
	C12 $\not\longrightarrow$ D0-D11		Sign bit extended to form 32-bit word
	R28-R31 $\longrightarrow$ LR28-LR31	S/LRXR = 0XC + . . .	
	If R28-R31 are not all zeros, clock RR0-RR31 $\not\longrightarrow$ A0-A31	AXRR = FABS PH1 NRZ + . . .	Contents of R-register (contains source address)
		AX/1 = FABS PH1 + . . .	
	Force a one onto private memory address line LR31 to address odd numbered private memory register	R/NLR31/2 = LR31/2 = FABS PH1 + . . .	
	Set flip-flop PH2	S/PH2 = PH1 NBR + . . .	
	Enable clock T6RL	T6RL = FAB0/1 PH1 + . . .	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + . . .	No function. Does not inhibit transmission of clock
PH2 T6RL	A0-A31 + D0-D31 $\longrightarrow$ S0-S31	SXADD = FAB0/2 PH2 + . . .	Adds displacement value to source address

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6RL (Cont.)	S0-S31 ↛ B0-B31	BXS = FABO PH2 + ...	Store modified source address in B-register
	RR0-RR31 ↛ A0-A31	AXRR = FABO/2 PH2 + ...	Word from odd numbered private memory register (contains count and destination address)
	Force a one on private memory address line LR31	R/NLR31/2 = LR31/2 = FABO PH2 + ...	For use in PH3
	Set flip-flop PH3	S/PH3 = PH2 NBR + ...	
	Enable clock T10L	S/T10L = FABO PH2 + ...	
	Set flip-flop DRQ	S/DRQ = FABO/1 EXU + ...	No function. Does not inhibit transmission of clock
PH3 T10L	R28-R30 → LR28-LR30 LR31/2 → LR31	LRXR = LRXD + LRXR	Address of odd numbered private memory register
	D0-D31 → S0-S31	SXD = FABO/2 PH3 + ...	Displacement gated onto sum bus
	S0-S31 → RW0-RW31	RWXS = FABO PH3 + ...	
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RW = FABO PH2 + ...	Store 32-bit displacement value in odd numbered private memory register
	Disable memory protection	PROTECTDIS = FABOX + ...	
	Clear D-register	DX/1 = FABS PH3 + ...	
	Set flip-flop PH5	S/PH5 = BRPH5 = FABO/1 PH3 + ...	
	Enable clock T8L	R/NT8L = FABO/1 PH3 + ...	
	Set flip-flop DRQ	S/DRQ = FABO/1 EXU + ...	No function. Does not inhibit transmission of clock
PH5 T8L	Circular shift B-register right two bit positions with B30 ↛ B0 and B31 ↛ B1	BXBR2 = FABS PH5 + ... S/B0 = B30 BXBR2 + ... S/B1 = B31 BXBR2 + ...	Modified source address shifted right to place byte address bits in positions B0 and B1 and address in bit positions 15 through 31
	Circular shift A-register right two bit positions through adder with A30 ↛ A0 and A31 ↛ A1	AXPRR2 = FABS PH5 + ... S/A0 = A30 ARCYC AXPRR2 + ... S/A1 = A31 ARCYC AXPRR2 + ...	Destination address circular shifted right to place byte address bits in positions A0 and A1

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T8L (Cont.)		ARCYC = ALCYC NANLZ	
		ALCYC = NFAMDSF + ...	
	A0-A31 ——→ S0-S31	SXA = FAB0/2 PH5 + ...	
	S0-S7 —/→ E0-E7	EXS = FAB0/2 PH5 + ...	Count clocked into E-register
	If the count is not all zeros and if there is no interrupt occurring, set word mode flip-flop SW1	S/SW1 = FABS PH5 NA0007Z NINT + ...	
	Reset flip-flop SW2 to use for memory protection violation	R/SW2 = FABS PH5 + ...	
	Set flip-flop PHA if count is all zeros or INT is true	S/PHA = FABS PH5 A0007Z + FABS PH5 INT + ...	
	If PHA is true, branch to PH6A; otherwise, sequence to PH6	S/PH6 = PH5 NBR + ...	
		R/NT8L = T8L = FAB0 PH5 + ...	
	Enable clock T8L		
	If flip-flop PHA is not set, set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of clock
PH6 T8L	B0-B31 ——→ S0-S31	SXB = FABS PH6 + ...	Modified source address gated onto sum bus
	S0 —/→ P32 S1 —/→ P33	S/P32 = S0 PXS/1 + ... S/P33 = S1 PXS/1 + ...	Byte address clocked into flip-flop P32 and P33
	S15-S31 —/→ P15-P31	PXS/1 = FAB0/2 PH6 + ...	Modified source address clocked into P-register
	P15-P31 ——→ LM15-LB31		Modified source address gated onto core memory address lines
	If the R-field is not all zeros and byte 0 has been selected both in the source and destination addresses and the count in the E-register is 4 or more, set word mode flip-flop SW1; otherwise, reset it	S/SW1 = (FABS 07) PH6 (NB0 NB1 NA0 NA1) NRZ NE05Z + ... R/SW1 = FABS PH6 + ...	

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH6 T8L (Cont.)	If signal SW1 is generated, indicating a word mode, or the R-field is not all zeros and byte 0 has been selected, generate a memory request	MRQ	= FABS PH6 SW1 + FABS PH6 NRZ NB0 NB1 + ...	Generate memory request for first data word
	Set flip-flop DRQ	S/DRQ	= FAB0/1 EXU + ...	If MRQ is generated, inhibits transmission of another clock until data release signal is received
	Set flip-flop PH7	S/PH7	= PH6 NBR + ...	
	Enable clock T6L	T6L	= NT1L NT4L NT8L NL10L NRESET	
PH7 T6L	MB0-MB31 → C0-C31	CXMB	= DGC	First data word gated into C-register
	If SW1 is true, C0-C31 -/→ D0-D31	DXC/6	= FABS PH7 SW1 + ...	
		DXC/10, DXC/11	= DXC/6 + ...	
		DXC/12	= DXC/7 + ...	
		DXC/13	= DXC/7 + DXC/1 + ...	
		DXC/7	= DXC/6 + ...	
		DXC/1	= DXCBP P32 P33	
		DXCR8	= DXCBP P32 NP33	
		DXCR16/1	= DXCBP NP32 P33	
		DXCR24	= DXCBP NP32 NP33	
	If NSW1 is true, byte selected by P32 and P33 clocked into flip-flops D24-D31	DXCBP	= FABS PH7 NSW1 + ...	
	If signal SW1 is true, add one to address contained in P-register	PCTP1	= FABS PH7 SW1 + ...	
	If NSW1 is true, add one to byte selection bits, P + 1/4 -/→ P	PCTP1	= FABS PH7 NRZ + ...	
		PA33	= FABS PH7 NRZ + ...	
	If modified source address points to an address located in protected memory, set flip-flop SW2	S/SW2	= FAB0/2 PH7 PROTECTD + ...	
	Set flip-flop PH8	S/PH8	= PH7 NBR + ...	

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T6L (Cont.)	Enable clock T4L	S/T4L = FAB0/1 PH7 + ...	
PH8 T4L	P15-P31 ⟶ S15-S31, P32 ⟶ S0, P33 ⟶ S1	SXP = FABS PH8 + ...	Modified source address and byte selection bits gated onto sum bus
	S0-S31 $\not\longrightarrow$ B0-B31	BXS = FABS PH8 + ...	Modified source address stored in B-register to allow destination address into P-register
	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T4L	S/T4L = FAB0/1 PH8 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of next clock
PH9 T4L	A0-A31 ⟶ S0-S31	SXA = FAB0/1 PH9 + ...	
	S15-S31 $\not\longrightarrow$ P15-P31	PXS/1 = FAB0/1 PH9 + ...	Destination address clocked into P-register
	P15-P31 ⟶ LM15-LB31		Destination address gated onto core memory address lines
	Clear A-register	AX/1 = FABS PH9 + ...	
	If signal NSW2 is true, generate memory request to store first word or byte	MRQ = FABS PH9 NSW2 + ...	
	If signal NSW1 is true, set flip-flop NPRX to use for upward alignment	S/NPRX = FABS 07 NSW1 PH9 + ...	
	Set flip-flop PH10	S/PH10 = PH9 NBR	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU	If MRQ is generated, inhibits transmission of next clock until signal is received that data lines have been strobed; otherwise, no function
PH10 T6L	If signal SW1 is true, D0-D31 ⟶ S0-S31	SXD = FABS 07 PH10 SW1 + ...	Word mode

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T6L (Cont.)	S0-S31 → MB0-MB31	MBXS = FABS O7 PH10 SW1 + ...	Store word in destination address location of core memory
	Generate write-byte signals /MW0/ - /MW3/	/MW0/ - /MW3/ = MBXS	
	If signal NSW1 is true, upward align byte selected by bits P32 and P33 onto S0-S31. Byte selected by bits P32 and P33 gated onto data lines MB0-MB31 as determined by write-byte signal	SXUAB = FABS O7 PH10 NSW1	Selected byte → S0-S7, S8-S15, S16-S23, and S24-S31. Sum bus output selected by MBXS
		MBXS/0 = NP32 NP33 MWB + ...	
		MBXS/1 = NP32 P33 MWB + ...	
		MBXS/2 = P32 NP33 MWB + ...	
		MBXS/3 = P32 P33 MWB + ...	
		WRITE = MWB + MW + ...	
		MWB = FABS O7 PH10 NSW1	
		MW = FABS O7 PH10 SW1	
	If signal SW1 is true, count in E-register is decreased by four, indicating four bytes were transferred. Count in P-register is increased by one	ES4 = FABS/10 SW1 + ...	
		PCTP1 = FABS/10 + ...	
		MCTE1 = FABS/10 + ...	
	If signal NSW1 is true, count in E-register is decreased by one and address in P-register is increased by 1/4	MCTE2 = MCTE1 NE4 NE5 NE6 NE7	
		PCTP1 = FABS/10 + ...	
		PA33 = FABS/10 NSW1 + ...	
		FABS/10 = FABS PH10 NSW2 NPROTECTD + ...	
	If signal PROTECTD received, set flip-flop SW2, indicating a memory protection violation occurred	S/SW2 = FAB0 PH10 PROTECTD + ...	
	Set flip-flop PH13	S/PH13 = BRPH13 = FABS O7 PH10 + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH13 T6L	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FABS PH13 + ...	Modified destination address and byte address

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T6L (Cont.)	S0-S31 ⇸ A0-A31	AXS = FABS PH13 + ...	Modified destination address stored in A-register
	If signal SW2 is true, INT is true, or the E-register contains all zeros flip-flop PHA is set	S/PHA = FABS PH13 (SW2 + INT + NA0031Z + EZ) + ...	
	Branch to PH6. If PHA is not set, phases PH6 through PH13 are repeated until flip-flop PHA is set	S/PH6 = BRPH6 = FABS PH13 + ...	
	Enable clock T8L	R/NT8L = T8L = FABS PH13 + ...	
PH6A T8L	A0-A31 → S0-S31	SXA = FABSA PH6 + ...	Destination address shifted left once to place byte address from P32 into A31
	Circular left shift one digit position with S1-S31 ⇸ A0-A30, S0 ⇸ A31	AXSL1 = FABSA PH6 + ... A31EN/2 = S0 ALCYC	
	Circular left shift one digit position with B1-B31 ⇸ B0-B30, B0 ⇸ B31	BXB1L = FABSA PH6 B31EN/1 = B0 NFAMDSF	Source address shifted left once to place byte address from B0 into B31
	Force a one onto private memory address line LR31 to select odd numbered private memory register	R/NLR31/2 = LR31/2 = FABSA PH6 + ...	
	Generate signal (S/CXRR)	(S/CXRR) = FABSA PH6 + ...	For use in PH7A
	Set flip-flop PH7	S/PH7 = PH6 NBR	
	Enable clock T10L	S/T10L = (S/CXRR) + ...	
PH7A T10	A0-A31 → S0-S31	SXA = FABSA PH7 + ...	Destination address shifted left one bit position to place address in bit positions 13 through 31 with byte address in bits 30 and 31
	Circular shift left one digit position with S1-S31 ⇸ A0-A30, S0 ⇸ A31	AXSL1 = FABSA PH7 + ... A31EN/1 = S0 ALCYC	
	E0-E7 ⇸ A0-A7	AXE = FABS PH7 + ... A0EN/1 = E0 NFAMDSF	Modified byte count clocked into flip-flops A0-A7
	RR0-RR31 → C0-C31	CXRR preset in PH6	Displacement bits read back into C-register. Stored in private memory register during PH3

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7A T10L (Cont.)	Force a one on LR31 private memory address line to select odd numbered private memory register	R/NLR31/2 = LR31/2 = FABSA PH7 + ...	
	Set flip-flop PH8	S/PH8 = PH7 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FABSA PH7 + ...	
PH8A T8L	A0-A31 → S0-S31	SXA = FAB0A/2 PH8 + ...	Modified destination address and count
	S0-S31 → RW0-RW31	RWXS = FAB0A/2 PH8 + ...	
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	Destination address and count stored in odd numbered private memory register
	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FAB0A/2 PH8	
PH9A T8L	B0-B31 → S0-S31	SXB = FAB0A/2 PH9 + ...	Source address
	Shift left one bit position into A-register with S1-S31 ⇸ A0-A30, S0 ⇸ A31	AXSL1 = FAB0A PH9 + ...	Source address shifted left one bit position to place address into bit positions 13 through 31 with byte address in bits 30 and 31
	NC0-NC31 ⇸ D0-D31	DXNC = FABSA PH9 + ...	Displacement value inverted into D-register to obtain one's complement
	1 ⇸ CS31 under conditions specified	CSX1/8 = FABSA PH9 (SW1 + NSW2)	For two's complement SW1 ⇒ word mode NSW2 ⇒ no memory protection violation
	Generate a memory request for the next instruction	MRQ/1 = FAB0A/1 PH9 + ...	
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + ...	Address of next instruction
	P15-P31 → LM15-LB31		
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MBS (61)

(Continued)

Table 3-102. Move Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10A T6L	A0-A31 + D0-D31 + CS0-CS31 → S0-S31	SXK = SXADD = FAB0A/2 PH10 + ...	Subtraction of displacement value from source address
	S0-S31 –/→ A0-A31	AXS = FABSA PH10 + ...	Result –/→ A-register
	Clear D-register	DX/1 = FABSA PH10 + ...	
	If signals SW1 and SW2 are true, indicating a word transfer and a memory protection violation, ones are forced into flip-flops CS0-CS29	CSX1/1 = FABSA PH10 SW1 SW2 + ...	Places -4 in CS-register
	Set flip-flop DRQ	S/DRQ = FAB0A PH10 + ...	Inhibits transmission of another clock until data release signal is received
	Set flip-flop IEN	S/IEN = FAB0A PH10 + ...	Enable interrupt
	Set flip-flop PH11	S/PH11 = PH10 NBR + ...	
	Enable clock T10L	S/T10L = FAB0A/2 PH10 + ...	
PH11A T10L	R28-R31 → LR28-LR31	Always occurs unless signal LRXD is true	Address of private memory register
	If R31 is not a one, but the R-register is not all zeros A0-A31 + CS0-CS29 → S0-S31	SXK = SXADD = FABSA PH11 NR31 NRZ + ...	Four subtracted from address to correct for increase due to 4-byte transfer. D-register contains zeros
	S0-S31 → RW0-RW31	RWXS = FABSA PH11 NR31 NRZ + ...	Store source address in even numbered private memory register
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	
	Generate signal ENDE	ENDE = FAB0A PH11 EZ + ...	
	If signal SW2 is true, indicating a memory protection violation, generate signal (S/TRACC4/1) and set TRAP flip-flop	(S/TRACC4/1) = FAB0A PH11 SW2 NTRAP + ... S/TRAP = (S/TRACC4/1) + ...	
	If flip-flop INTRAPF is not set, set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MBS (61)

Table 3-103. Compare Byte String, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1	P15-P31 ⇸ Q15-Q31	QXP = PRE1 NANLZ + ...	
	Reset A-, B-, and E-registers	AX/1 = PRE1 + ...	
		BX/1 = PRE1 NINTRAPF + ...	
		EX/1 = PRE1 + ...	
	Generate signal PROTECTDIS	PROTECTDIS = FABO + ...	Disables trapping feature of memory protection
	C0-C31 ⇸ D0-D31	DXC = ENDE + ...	
	Set flip-flop PH1	S/PH1 = PREIM PRE1 NCLEAR NBR + ...	
	Enable clock T4RL	T4RL = PREP	
	If bit C0 is a 1, generate signal FAILL and set flip-flop TRAP	FAILL = IA N03 (N04 N05) + ...	
		S/TRAP = FAILL (PRE1 NANLZ) + ...	
	Generate signal EXU	S/EXU = PREIM PRE1 NCLEAR + ...	
PH1 T4RL	C12-C31 ⇸ D12-D31	DXC/4 = FABS PH1 + ...	Displacement value clocked into D-register
	C12 ⇸ D0-D11		Sign bit extended to form 32-bit word
	R28-R31 → LR28-LR31	S/LRXR = OXC + ...	
	If R28-R31 are not all zeros, clock RR0-RR31 ⇸ A0-A31	AXRR = FABS PH1 NRZ + ...	Contents of register R (contains source address)
	Force a one onto private memory address line LR31 to address odd numbered private memory register	R/NLR31/2 = LR31/2 = FABS PH1 + ...	
	Set flip-flop PH2	S/PH2 = PH1 NBR + ...	
	Enable clock T6RL	T6RL = FABO/1 PH1 + ...	
	Set flip-flop DRQ	S/DRQ = FABO/1 EXU	No function. Does not inhibit transmission of clock
PH2 T6RL	A0-A31 + D0-D31 → S0-S31	SXADD = FABO/2 PH2 + ...	Adds displacement value to source address
	S0-S31 ⇸ B0-B31	BXS = FABO PH2 + ...	Store modified source address in B-register

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6RL (Cont.)	RR0-RR31 —/→ A0-A31	AXRR = FAB0/2 PH2 + ...	Word from odd numbered private memory register (contains count and destination address)
	Force a one on private memory address line LR31	R/NLR31/2 = LR31/2 = FAB0 PH2 + ...	For use in PH3
	Set flip-flop PH3	S/PH3 = PH2 NBR + ...	
	Enable clock T10L	S/T10L = FAB0 PH2 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of clock
PH3 T10L	R28-R30 → LR28-LR30	S/LRXR = LRXD + LRXR	Address of odd numbered private memory register
	LR31/2 → LR31		
	D0-D31 → S0-S31	SXD = FAB0/2 PH3 + ...	Displacement gated onto sum bus
	S0-S31 → RW0-RW31	RWXS = FAB0 PH3 + ...	
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RW = FAB0 PH2 + ...	Store 32-bit displacement value in odd numbered private memory register
	Reset flip-flops CC3 and CC4	R/CC3 = FABS PH3 NO7 + ...	
		R/CC4 = FABO/1 PH3 NO7 + ...	
	Clear D-register	DX/1 = FABS PH3 + ...	
	Disable memory protection	PROTECTDIS = FABOX + ...	
	Set flip-flop PH5	S/PH5 = BRPH5 = FAB0/1 PH3 + ...	
	Enable clock T8L	R/NT8L = T8L = FAB0/1 PH3 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of clock
PH5 T8L	Circular shift B-register right two bit positions with B30 —/→ B0 and B31 —/→ B1	BXBR2 = FABS PH5 + ... S/B0 = B30 BXBR2 + ...	Modified source address shifted right to place byte address bits in positions B0 and B1 and address in B15 through B31
		S/B1 = B31 BXBR2 + ...	
	Circular shift A-register right two bit positions through adder with A30—/→A0 and A31—/→A1	AXPRR2 = FABS PH5 + ... S/A0 = A30 ARCYC AXPRR2 + ...	Destination address circular shifted right to place byte address in positions A0 and A1

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T8L (Cont.)		S/A1 = A31 ARCYC AXPRR2 + ...	
		ARCYC = ALCYC NANLZ	
		ALCYC = NFAMDSF + ...	
	A0-A31 ──→ S0-S31	SXA = FAB0/2 PH5 + ...	
	S0-S7 ─/→ E0-E7	EXS = FAB0/2 PH5 + ...	Count clocked into E-register
	Reset flip-flop SW2 to use for memory protection violation	R/SW2 = FABS PH5 + ...	
	Set flip-flop PHA if count is all 0's or if signal INT is true	S/PHA = FABS PH5 A0007Z + FABS PH5 INT + ...	
	If PHA is true, branch to PH6A; otherwise,sequence to PH6	S/PH6 = PH5 NBR + ...	
	Enable clock T8L	R/NT8L = FAB0 PH5 + ...	
	If flip-flop PHA is not set, set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of clock
PH6 T8L	B0-B31 ──→ S0-S31	SXB = FABS PH6 + ...	Modified source address gated onto sum bus
	S0 ─/→ P32 S1 ─/→ P33	S/P32 = S0 PXS/1 + ...	Byte address clocked into flip-flops P32 and P33
		S/P33 = S1 PXS/1 + ...	
	S15-S31 ─/→ P15-P31	PXS = FAB0/2 PH6 + ...	Modified source address clocked into P-register
	P15-P31 ──→ LM15-LB31		Modified source address gated onto core memory address lines
	Generate memory request for first byte	MRQ = FABS PH6 N07 + ...	Memory request for first byte
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	Inhibits transmission of another clock until data release signal is received
	Set flip-flop PH7	S/PH7 = PH6 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T6L	MB0-MB31 → C0-C31	CXMB = DGC	First data word gated into C-register
	Byte selected by bits P32 and P33 are clocked into flip-flop D24-D31	DXCBP = FABS PH7 NSW1 + ...	See equations in PH7 of Move Byte String
		PCTP1 = FABS PH7 NRZ + ...	
	Add one to byte selection bits for selection of next byte	PA33 = FABS PH7 NRZ + ...	
	If source address pointed to an address located in protected memory, set flip-flop SW2	S/SW2 = FAB0/2 PH7 PROTECTD + ...	
	Set flip-flop PH8	S/PH8 = PH7 NBR + ...	
	Enable clock T4L	S/T4L = FAB0/1 PH7 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of next clock
PH8 T4L	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FABS PH8 + ...	Modified source address and byte selection bits gated onto sum bus
	S0-S31 –/→ B0-B31	BXS = FABS PH8 + ...	Modified source address stored in B-register to allow destination address in P-register
	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T4L	S/T4L = FAB0/1 PH8 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Does not inhibit transmission of next clock
PH9 T4L	A0-A31 → S0-S31	SXA = FAB0/1 PH9 + ...	
	S15-S31 –/→ P15-P31	PXS/1 = FAB0/1 PH9 + ...	Address of destination byte clocked into P-register
	P15-P31 → LM15-LB31		Address of destination byte gated onto core memory address lines
	Clear A-register	AX/1 = FABS PH9 + ...	

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T4L (Cont.)	Generate memory request for byte located in destination address location	MRQ = FABS PH9 NSW2 + ...	
	Generate signal (S/NGX)	(S/NGX) = FABS PH9 N07 + ...	
	Force ones into CS-register	CSX1/1 = (S/NGX) + ...	
	Force a one into K31	K31 = (S/NGX) + ...	For 2's complement in PH10
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	Inhibits transmission of another clock until data release signal is received

PH10
T6L
	Gate source destination byte from D-register into adder with ones in CS-register and one in K31 for two's complement operation	SXPR = SXADD/1 = NGX NFAMDSF + ...
		SXK = SXADD/1 = NGX NFAMDSF + ...
			Generate two's complement of source byte for subtraction from destination byte during PH11

	Gate two's complement of source byte onto sum bus	SXPR and SXK as above	
	Clock contents of sum bus into flip-flops A0-A31	AXS = FABS PH10 N07 + ...	
	When byte from destination address is available, MB0-MB31 ⟶ C0-C31	CXMB = DGC	Destination byte gated into C-register, and then selected byte clocked into flip-flops D24-D31
	Destination byte selected by bits P32 and P33 clocked into flip-flops D24-D31	DXCBP = 0U6 0L0 PH10 NPHA + ...	See equations in PH7 of Move Byte String
	If address of destination byte pointed to a protected area of memory, set flip-flop SW2	S/SW2 = FAB0 PH10 PROTECTD + ...	
	Set flip-flop PH11	S/PH11 = PH10 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments

PH11
T6L
	Gate two's complement of source byte into adder with destination byte, and perform an addition (subtraction of source byte from destination byte)	SXPR = SXADD
		SXK = SXADD	

	Result gated onto sum bus	SXADD = FABS PH11 + ...	
	Clock result from sum bus into flip-flops A0-A31	AXS = FABS PH11 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU	No function. Does not inhibit transmission of next clock
	Set flip-flop PH12	S/PH12 = PH11 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FABS PH11 + ...	
PH12 T8L	If bit A0 is a one, set flip-flop CC3	S/CC3 = FABS PH12 NSW2 A0 + ...	Indicates source byte is larger than destination byte
	If bit A0 is not a one, but there are ones contained in the A-register, set flip-flop CC4	S/CC4 = FABS PH12 NSW2 NA0 NA0031Z + ...	Indicates source byte is smaller than destination byte

	Subtract one from byte count in E-register and add one to byte address in bits P32 and P33 if the bytes compare (A0031Z)	MCTE1 = FABS PH12 NSW2 A0031Z + ...
		ES4 = MCTE1
		PCTP1 = FABS PH12 NSW2 A0031Z + ...
		PA33 = FABS PH12 NSW2 A0031Z + ...	

	Set flip-flop PH13	S/PH13 = PH12 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH13 T6L	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FABS PH13 + ...	Modified destination address and byte address
	S0-S31 ↛ A0-A31	AXS = FABS PH13 + ...	Modified destination address stored in A-register

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T6L (Cont.)	If signal SW2 is true, INT is true, or the E-register contains all zeros, flip-flop PHA is set. If the two bytes do not compare (NA0031Z), flip-flop PHA is set	S/PHA = FABS PH13 (SW2 + INT + NA0031Z + EZ) + ...	
	Branch to PH6. If flip-flop PHA is not set, phases PH6 through PH13 are repeated until flip-flop PHA is set		
	Enable clock T8L	R/NT8L = T8L = FABS PH13 + ...	
PH6A T8L	A0-A31 ——→ S0-S31	SXA = FABSA PH6 + ...	Destination address shifted left once to place byte address into A31
	Circular left shift one digit position with S1-S31 —/→ A0-A30, S0 ——→ A31	AXSL1 = FABSA PH6 + ... A31EN/2 = S0 ALCYC	
	Circular left shift B-register one digit position with B1-B31 —/→ B0-B30, B0 —/→ B31	BXBL1 = FABSA PH6 + ... B31EN/1 = B0 NFAMDSF	Source address shifted left one bit position to place byte address into flip-flop B31
	Force a one onto private memory address line LR31 to select odd numbered private memory register	R/NLR31/2 = LR31/2 = FABSA PH6 + ...	
	Generate signal (S/CXRR)	(S/CXRR) = FABSA PH6 + ...	For use in PH7A
	Set flip-flop PH7	S/PH7 = PH6 NBR + ...	
	Enable clock T10L	S/T10L = (S/CXRR) + ...	
PH7A T10L	A0-A31 ——→ S0-S31	SXA = FABSA PH7 + ...	Destination address shifted left one bit position to place address in bit positions 13 through 31 with byte address in bits 30 and 31
	Circular shift left one digit position with S1-S31 —/→ A0-A30, S0 —/→ A31	AXSL1 = FABSA PH7 + ... A31EN/1 = S0 ALCYC	
	E0-E7 —/→ A0-A7	AXE = FABS PH7 + ... A0EN/1 = E0 NFAMDSF	Modified byte count clocked into flip-flops A0-A7

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7A T10L (Cont.)	RR0-RR31 → C0-C31	CXRR preset in PH6A	Displacement bits read back into C-register. Stored in private memory register during PH3
	Force a one on 0R31 private memory address line to select odd numbered private memory register	R/NLR31/2 = LR31/2 = FABSA PH7 + ...	
	Set flip-flop PH8	S/PH8 = PH7 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FABSA PH7 + ...	
	A0-A31 → S0-S31	SXA = FAB0A/2 PH8 + ...	Modified destination address and count
	S0-S31 → RW0-RW31	RWXS = FAB0A/2 PH8 + ...	
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	Destination address and count stored in odd numbered private memory register
	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FAB0A/2 PH8	
PH9A T8L	B0-B31 → S0-S31	SXB = FAB0A/2 PH9 + ...	Source address
	Shift left one bit position into A-register with S1-S31 ⇸ A0-A30 S0 ⇸ A31	AXSL1 = FAB0A PH9 + ...	Source address shifted left one bit position to place address into bit positions 13 through 31 with byte address in bits 30 and 31
	NC0-NC31 ⇸ D0-D31	DXNC = FABSA PH9	Displacement value inverted into D-register to obtain one's complement

	1 → CS31 under specified conditions	CSX1/8 = FABSA PH9 NSW2 N(NO7 CC3 + NO7 CC4) + ...	For two's complement.
			NSW2 => no memory protection violation
			CC3 => source byte > destination byte
			CC4 => source byte < destination byte

	Generate a memory request for next instruction	MRQ/1 = FAB0A/1 PH9 + ...	
	Q15-Q31 ⇸ P15-P31	PXQ = MRQ/1 + ...	Address of next instruction

Mnemonic: CBS (60)

(Continued)

Table 3-103. Compare Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9A T8L (Cont.)	P15-P31 ⟶ LM15-LB31		
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH10A T6L	A0-A31 + D0-D31 + CS0-CS31 ⟶ S0-S31	SXK = SXADD = FAB0A/2 PH10 + ...	Subtraction of displacement value from source address
	S0-S31 $\not\longrightarrow$ A0-A31	AXS = FABSA PH10 + ...	Result $\not\longrightarrow$ A-register
	Clear D-register	DX/1 = FABSA PH10 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0A PH10 + ...	Inhibits transmission of another clock until data release signal is received
	Set flip-flop IEN	S/IEN = FAB0A PH10 + ...	
	Set flip-flop PH11	S/PH11 = PH10 NBR + ...	
	Enable clock T10L	S/T10L = FAB0A/2 PH10 + ...	
PH11A T10L	R28-R31 ⟶ LR28-LR31	Always occurs unless signal LRXD is true	Address of private memory register
	If R31 is not a one, but the R-register is not all zeros, A0-A31 + CS0-CS31 ⟶ S0-S31	SXK = SXADD = FABSA PH11 NR31 NRZ	Source address gated onto sum bus
	S0-S31 ⟶ RW0-RW31	RWXS = FABSA PH11 NR31 NRZ	Store source address in even numbered private memory register
	Generate write byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	
	Generate signal ENDE	ENDE = FABSA PH11 N07 CC3 + FAB0A/1 PH11 N07 CC4 + FAB0A PH11 EZ	If CC3 or CC4 has been set or E-register contains zeros
	If signal SW2 is true, indicating a memory protection violation, generate signal (S/TRACC4/1) and set TRP flip-flop	(S/TRACC4/1) = FAB0A PH11 SW2 NTRAP + ... S/TRAP = (S/TRACC4/1) + ...	
	If flip-flop INTRAPF is not set, set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: CBS (60)

The translation table consists of up to 256 consecutive byte locations. The first byte location of the table is pointed by the displacement in the TBS instruction word plus the source address contained in the even numbered private memory register.

A source byte is the byte in the location pointed to by the 19 low order bits of the sum of the following:

a. The displacement in bit positions 12 through 31 of the TBS instruction word

b. The current contents of bit positions 13 through 31 of the even numbered private memory register

c. The numeric value of the current destination byte

If the TBS instruction is indirectly addressed, it is treated as a nonexistent instruction, and the computer traps to location X'40' at the time of opcode decoding.

The TBS instruction word is clocked into the D-register at the clock following the generation of signal ENDE. The opcode portion is clocked into the O-register for decoding, the R-field is clocked into the R-register, and the address contained in the P-register is increased by one. Indirect addressing is indicated if bit C0 is a one. Signal FAILL is generated, which causes the computer to trap to location X'40'.

During PRE1, signal PROTECTDIS is generated to inhibit trapping if an attempt is made to read out from or store into a protected area of core memory. At the PRE1 clock, the A-, B-, and E-registers are cleared, and the address of the next instruction contained in the P-register is clocked into the Q-register. Flip-flop PRE2 is set, and clock T4RL is enabled.

At the PRE2 clock, flip-flop NLR31/2 is reset, thereby forcing a one on the LR31 address line. Signal (S/PH1/1) is generated, and flip-flops PH1 and EXU are set. Clock T4RL is enabled.

The displacement value contained in bits C12 through C31 is clocked into the D-register at the PH1 clock. Flip-flops D0 through D11 are cleared. When the data word from the odd numbered private memory register is available on the RR lines, it is read into the A-register at the PH1 clock. Flip-flop PH2 is set, and clock T6RL is enabled.

The destination address and the count are gated from the A-register onto the sum bus during PH2. The count is read from bits S0 through S7 into the E-register at the PH2 clock, and the entire word, bits S0 through S31, is clocked into the B-register. When the data word from the even numbered private memory register is available on the RR lines, it is read into the A-register at the PH2 clock. A one is forced on LR31 address line to select odd numbered private memory register. If the R-field of the instruction word is all zeros, ones are forced into flip-flops CS0 through CS7. Flip-flop DRQ is set, but transmission of the next clock is not inhibited, because a memory request has not been generated. The instruction sequences to PH3, and clock T10L is enabled.

During PH3, the source address contained in the A-register, the displacement value contained in the D-register, and the contents of the CS-register (all zeros if the R-field is nonzero) are gated into the adder. An addition operation is performed, and the result is gated onto the sum bus. If the R-field is all zeros, the result of the addition places the displacement value in bit positions 12 through 31 and the ones in bit positions 0 through 7. If the R-field is nonzero, the displacement value is added to the source address, and the result is gated onto the sum bus.

The modified source address is gated from the sum bus onto private memory data lines RW0 through RW31. Write-byte signals RWB0 through RWB3 are generated, and the source address is stored in the odd numbered private memory register.

The address contained in the B-register has the destination byte address in bits 30 and 31 and the destination word address in bit positions 13 through 29. At the PH3 clock, the contents of the B-register are circular shifted two bit positions to the right to place the destination address in bit positions 15 through 31 and the byte address in bit positions 0 and 1. First pass flip-flop SW1 is set, indicating the first pass through the translation sequence. Protect fail flip-flop SW2 is reset. Because the memory protection trapping feature is disabled, flip-flop SW2 is used to indicate a memory protection violation. Flip-flop DRQ is set, but clock transmission is not inhibited because a memory request has not been generated. The instruction branches to PH5, and clock T8L is enabled.

If first pass flip-flop SW1 is set during PH5, the destination address is gated from the B-register onto the sum bus and is then clocked into the P-register at the PH5 clock. S15 through S31 are clocked into flip-flops P15 through P31 and byte address bits S0 and S1 are clocked into flip-flops P32 and P33, respectively. First pass flip-flop SW1 is then reset.

A memory request is generated for the first byte from the destination address that is contained in the P-register if the first pass flip-flop SW1 is set, or for the next byte if the first pass is not set. If the address is in a protected area of memory and if signal PROTECTD is received, protect fail flip-flop SW2 is set, flip-flop PHA is set, and the instruction sequences to PH9A without performing any additional transfers.

If a memory protection violation did not occur (NSW2 + NPROTECTD), a one is added to the byte address in bits P32 and P33 to select the next byte. A one is forced on private memory address line LR31 to select the odd numbered private memory register where the source address is stored. Clock T8L is enabled, and the instruction sequences to PH6.

If this is not the first pass through PH5 (that is, if the instruction has proceeded through PH10 and has returned to PH5), the byte read in from the source address during PH10 is gated into the adder and is upward aligned into bit positions 0 through 7, 8 through 15, 16 through 23, and 24 through 31 on the sum bus. The byte selected by bits P32 and P33 is gated onto the appropriate MB data lines. Signal MWB is generated, and the byte is stored in the addressed core memory destination address location. A one is added to the byte address in bit positions P32 and P33 to select the next byte. A one is forced onto private memory address line LR31 to select the odd numbered private memory register. A memory request is generated for the next byte; clock T8L is enabled; and the instruction sequences to PH6. When the count in the E-register finally becomes all zeros, flip-flop PHA is set, and the instruction branches to PH9A.

When the word pointed by the destination address is available on the MB data lines, signal CXMB gates the word into the C-register. At the PH6 clock, the byte selected by bits P32 and P33 is clocked into flip-flops D24 through D31. The destination address is gated from the P-register into the sum bus, and byte address bits P32 and P33 are gated into positions S0 and S1. At the PH6 clock, the entire destination address is clocked into the B-register. When the modified source address is on the RR lines from the odd numbered private memory register, signal AXRR is generated which clocks the address into the A-register at the PH6 clock. The instruction sequences to PH7, and clock T6L is enabled.

The modified source address contained in the A-register and the selected byte contained in the D-register from the destination address are gated into the adder during PH7 and an addition operation is performed. The result is gated onto the sum bus and is clocked from the sum bus into the A-register at the PH7 clock. The D-register is cleared, clock T4L is enabled, and the instruction sequences to PH8.

The modified source address is gated from the A-register into the adder during PH8 and is circular shifted two digit positions to the right. Bit A30 is clocked into flip-flop A0, and bit A31 is clocked into flip-flop A1. This places the source address in bit positions A15 through A31 and the byte selection bits in position A0 and A1. Clock T4L is enabled, and the instruction sequences to PH9.

During PH9 the modified source address is gated from the A-register onto the sum bus and is then clocked into the P-register. The byte address bits S0 and S1 are clocked into flip-flops P32 and P33. A memory request is generated for the byte pointed to by the source address. The source address is gated onto the LM and LB address lines to core memory. The byte count in the E-register is decreased by one count. Clock T6L is enabled, and the instruction sequences to PH10.

When the data word is available on the MB lines from the source address, signal CXMB is generated to gate the word into the C-register. The byte selected by bits P32 and P33 is clocked into flip-flops D24 through D31 at the PH10 clock. The destination address contained in the B-register is gated onto the sum bus and, at the PH10 clock, is clocked into the P-register. Byte selection bits S0 and S1 are clocked into flip-flops P32 and P33, respectively.

If a memory protection violation occurred during the source addressing, flip-flop SW2 is set and no further transfers are made. If flip-flop SW2 is not set, a memory request is generated to store the first byte in the location pointed to by the destination address. The A-register is cleared at the PH10 clock, thereby clearing the source address used for the byte. Flip-flop NPRX is set, forcing ones into the CS-register for use during the byte upward alignment in PH5. Flip-flop PH5 is set to return the instruction to PH5, and clock T8L is enabled.

Sequences PH5 through PH10 and back to PH5 are repeated until all of the destination bytes have been replaced with bytes from the translation table (that is, E-register contains all zeros). A memory protection violation (that is, PROTECTD is received or flip-flop SW2 is set) or an interrupt occurs. Flip-flop PHA is set, clock T8L is enabled, and the instruction branches to PH9A.

During PH9A, the destination address is gated from the P-register onto the sum bus. Bits P32 and P33 are gated onto bit positions S0 and S1. The contents of the sum bus are then circular shifted one bit position to the left and are clocked into the A-register which places byte address bit S0 in flip-flop A31. A memory request is generated for the next instruction. When memory request MRQ/1 is generated, the address of the next instruction is clocked from the Q-register into the P-register and is then gated onto the LM and LB address lines.

If a memory protection violation occurs, one is added to the count in the E-register to leave the count pointing to the last byte transferred. Clock T6L is enabled, and the instruction sequences to PH10A. During PH10A, the destination address contained in the A-register is gated onto the sum bus, is circular shifted one bit position to the left and is clocked back into the A-register. The destination address is then contained in bit positions A13 through A29, and the byte selection bits are contained in A30 and A31. At the same clock, the count from the E-register is clocked into flip-flops A0 through A7. The modified destination address and byte count are then ready for storing in private memory. A one is forced on address line LR31 to select the odd numbered private memory register. Flip-flop DRQ is set which inhibits transmission of another clock until the data release signal is received. Interrupt enable flip-flop IEN is set, clock T8L is enabled, and the instruction proceeds to PH11A.

The modified destination address and byte count from the A-register are gated onto the sum bus during PH11A and then onto the data lines RW0 through RW31. Write-byte signals RWB0 through RWB3 are generated, and the data word is stored in the odd numbered private memory register.

If flip-flop SW2 is set, which indicates a memory protection violation, signal (S/TRACC4/1) is generated. Flip-flop TRAP is set, which branches the instruction into the trap sequence. If flip-flop SW2 is not set and if the byte count in the E-register is all zeros, signal ENDE is generated, flip-flop PRE1 is set for the next instruction, and clock T6L is enabled.

A sequence chart of the Translate Byte String instruction is given in table 3-104.

3-232 Translate and Test Byte String (TTBS 40)

The TTBS instruction compares a mask contained in bit positions 0 through 7 of the private memory register addressed in the R-field with source bytes contained in a byte translation table. The destination byte string begins with the byte location pointed by the destination address in an odd numbered private memory register. The destination byte string is examined (without being changed) until a source byte is found that contains a one in any location comparing with the mask. When a comparison is found, the mask is replaced with the logical AND result of the source byte and the mask. Condition code flip-flop CC4 is set to a one.

If the instruction terminates because of a comparison match, the byte count is one greater than the number of bytes remaining to be compared, and the destination address indicates the location of the destination byte that caused the termination. If no comparison match is found, after the number of bytes that were indicated in the count have been compared, the instruction terminates with condition code flip-flop CC4 reset to zero. In no case is the source byte string changed.

If the TTBS instruction is indirectly addressed, it is treated as a nonexistent instruction. The computer aborts execution of the instruction at the time of opcode decoding and traps to location X'40'.

Operation during the preparation phases and PH1 through PH9 is the same as the operation for these phrases described in the Translate Byte String instruction.

If the R-field of the instruction word is all zeros, ones are forced into flip-flops CS0 through CS7 during PH2. This byte is added to the word which contains the displacement value and forms a mask for later use.

The word from the location pointed by the source address is gated into the C-register during PH10. At this clock, the byte selected by bits P32 and P33 is clocked into flip-flops D24 through D31. The destination address is gated from the B-register onto the sum bus. The destination address, bits S15 through S31, is clocked into the P-register of the PH10 clock, and the byte selection bits S0 and S1 are clocked into flip-flops P32 and P33. The A-register is cleared, thereby clearing the source address and mask. If a memory protection violation occurs (signal PROTECTD received), flip-flop SW2 is set and no further translation of bytes is performed. Flip-flop NPRX is set at the PH10 clock. This forces one's into the CS-register for use during upward alignment in PH11. Signal (S/CXS) is generated for use during PH11. Since the source address and mask were cleared from the A-register, a one is forced on the LR31 address line to read them back from the odd numbered private memory register. Flip-flop PH11 is set for the next phase, and clock T10L is enabled.

The source byte contained in bits D24 through D31 is gated into the adder during PH11 and is upward aligned into bit positions 0 through 7 of the sum bus. The byte is then gated from the sum bus into bit positions 0 through 7 of the C-register and is clocked into flip-flops D0 through D7 of the PH11 clock. Since the mask was cleared from the A-register during PH10, the data word containing the mask is read out of the private memory register and is clocked into the A-register at the PH11 clock. Flip-flop NPRX is set to force ones into the CS-register for use during the logical AND operation in PH12. Clock T4RL is enabled, and the instruction sequences to PH12.

The mask contained in the A-register and the source byte contained in the D-register are gated into the adder during PH12, and a logical AND operation is performed. This result is gated onto the sum bus and is clocked into the A-register at the PH12 clock. If no ones are compared in the mask and the source byte, the result in the A-register is all zeros. If any ones are compared, the result in the A-register is the byte formed as the result of the logical AND. Clock T8L is enabled, and the instruction sequences to PH13.

Flip-flop CC4 is set if any ones are contained in the A-register which indicates a comparison. No further byte comparisons are made. If a comparison has been found, and bits R28 through R31 are not all zeros, and if bit R31 is a zero, the byte stored in the A-register is gated onto the sum bus and then onto data lines RW0 through RW7. Write-byte signal RWB0 is generated, and the byte is stored in the even numbered private memory register in place of the mask.

Clock T8L is enabled, and the instruction branches to PH5. Phases PH5 through PH13 are repeated until a byte comparison is made (that is, flip-flop CC4 is set), the total number of bytes have been compared (signal EZ is true), a memory violation occurs (flip-flop SW2 is set), or an interrupt occurs (INT is true). Flip-flop PHA is then set and the instruction branches to PH9A.

Operation during PH9A through PH11A is the same as the operation for Translate Byte instruction, except that during PH11A flip-flop CC4 is reset if the count in the E-register is all zeros, and that signal ENDE is generated if flip-flop CC4 is set or if the count in the E-register is all zeros.

A sequence chart of the Translate and Test Byte String instruction is given in table 3-105.

Table 3-104. Translate Byte String, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1 T6L	Generate signal PROTECTDIS	PROTECTDIS = FABO + ...	Disables trapping function for memory protection

	Clear A-, B-, and E-registers	AX = PRE1 + ...
		BX = PRE1 NINTRAPF + ...
		EX = PRE1 + ...	

	P15-P31 $\nrightarrow$ Q15-Q31	QXP = PRE1 NANLZ + ...	Next instruction address
	Set flip-flop PRE2	S/PRE2 = NPREIM PRE1 N(SNTRAPF) + ...	
	Enable clock T4RL	T4RL = PREP + ...	
PRE2 T4RL	Force a one on LR31	R/NLR31/2 = LR31/2 = FATR (PRE2 NIA) + ...	Selects odd numbered private memory register
	Set flip-flop EXU	S/EXU = (S/PH1/1) NCLEAR + ...	
	Set flip-flop PH1	S/PH1 = (S/PH1/1) NCLEAR NBR + ... (S/PH1/1) = NPREDO PRE2 NIA + ...	
	Enable clock T4RL	T4RL = PREP + ...	
PH1 T4RL	C12-C31 $\nrightarrow$ D12-D31, 0 $\nrightarrow$ D0-D11	DXC/3 = FATR PH1 + ...	Displacement value
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FATR PH1 + ...	Destination address and count
	Set flip-flop PH2	S/PH2 = PH1 NBR + ...	
	Enable clock T6RL	T6RL = FABO/1 PH1 + ...	
PH2 T6RL	A0-A31 ⟶ S0-S31	SXA = FATR PH2 + ...	
	S0-S7 $\nrightarrow$ E0-E7	EXS = FATR PH2 + ...	Byte count
	S0-S31 $\nrightarrow$ B0-B31	BXS = FABO PH2 + ...	Destination address

Mnemonic: TBS (41)

(Continued)

Table 3-104. Translate Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6RL (Cont.)	RR0-RR31 ⇸ A0-A31	AXRR = FATR PH2 NRZ + ...	Source address
	Force a one on LR31 address line	R/NLR31/2 = LR31/2 = FAB0 PH2 + ...	Selects odd numbered private memory register
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Memory request has not been made
	Set flip-flop PH3	S/PH3 = PH2 NBR + ...	
	Enable clock T10L	R/NT10L = T10L = FAB0 PH2 + ...	
PH3 T10L	A0-A31 + D0-D31 + CS0-CS31 → S0-S31	SXADD = FATR PH3 + ...	Source address plus displacement value
	S0-S31 → RW0-RW31	RWXS = FAB0 PH3 + ...	Modified address ⇸ R-register
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	Store modified source address in odd numbered private memory register
	Circular shift B-register right two bit positions	BXBR2 = FATR PH3 + ...	Places destination address in bit positions 15 through 31 and byte selection bits in positions 0 and 1
	If first translation, set flip-flop SW1	S/SW1 = FATR PH3 + ...	First pass
	Disable memory protection	PROTECTDIS = FAB0X + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. Memory request has not been made
	Set flip-flop PH5	S/PH5 = BRPH5 = FAB0/1 PH3 + ...	
	Enable clock T8L	R/NT8L = T8L = FAB0/1 PH3 + ...	
PH5 T8L	If first pass (SW1 set), B0-B31 → S0-S31	SXB = FATR5 SW1 + ...	
	S15-S31 ⇸ P15-P31	PXS/1 = FATR5 SW1 + ...	Destination address
	S0 ⇸ P32, S1 ⇸ P33	PXS/1 = FATR5 SW1 + ...	Byte selection bits
	Reset flip-flop SW1	R/SW1 = FATR PH5 + ...	
	Generate memory request for byte	MRQ = FATR5 NFATRFINISH + ...	

Mnemonic: TBS (41)

(Continued)

Table 3-104. Translate Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T8L (Cont.)	If address was in a protected area of memory, set flip-flop SW2	S/SW2 = FATR PH5 PROTECTD + ...	
	If flip-flop SW2 is set, signal PROTECTD is received, the E-register has been counted down to zero, or an interrupt occurs, set flip-flops PHA and PH9	S/PHA = FATR5 FATRFINISH + ... S/PH9 = BRPH9 = FATR5 FATRFINISH + ... FATRFINISH = (SW2 + PROTECTD) + EZ + INT +...	

	If flip-flop SW2 is not set, add one to byte selection bits P32 and P33to select next byte	PCTP1 = FATR5 NSW1 (NSW2 NPROTECTD) + ...
		PA33 = FATR5 NSW1 (NSW2 NPROTECTD) + ...
			Selects new byte

	Force a one on address line LR31	R/NLR31/2 = LR31/2 = FAB0 NFABS PH5 + ...	Selects register where source address was stored
	Set flip-flop PH6	S/PH6 = PH5 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FAB0 PH5 + ...	
	If not first pass (NSW1), D24-D31 → K23-K30		
	K23-K30 → S0-S7, S8-S15, S16-S23		
	D24-D31 → S24-S31	SXUAB = FATR5 NSW1 + ...	Byte selected by P32 and P33 gated onto data lines. See equations in PH10 of Move Byte String
	Byte selected by P32 and P33 → MB0-MB7, or MB8-MB15, or MB15-MB23, or MB24-MB31	MBXS/0, MBXS/1, MBXS/2, or MBXS/3	
	Generate memory write-byte signal MWB	MWB = FATR NSW1 PH5 + ...	Store byte in destination address location of core memory
	Set flip-flop PH6	S/PH6 = PH5 NBR + ...	
	Enable clock T8L	R/NT8L = S/T8L = FAB0 PH5 + ...	Inhibits transmission of another clock
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	

Mnemonic: TBS (41)

(Continued)

Table 3-104. Translate Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6 T8L	MB0-MB31 → C0-C31	CXMB = DGC	Word from destination address
	Byte selected by P32 and P33 ↛ D24-D31	DXCBP = FATR PH6 + ...	Byte from destination address downward aligned into D-register. See equations in PH7 of Move Byte String
	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FATR PH6 + ...	
	S0-S31 ↛ B0-B31	BXS = FATR PH6 + ...	Destination address stored in B-register
	RR0-RR31 ↛ A0-A31	AXRR = FATR PH6 + ...	Modified source address clocked into A-register from private memory
	Set flip-flop PH7	S/PH7 = PH6 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
PH7 T6L	A0-A31 + D0-D31 → S0-S31	SXADD = FATR PH7 + ...	Source address and byte added
	S0-S31 ↛ A0-A31	AXS = FATR PH7 + ...	Result ↛ A-register
	Clear D-register	DX/1 = FATR PH7 + ...	
	Set flip-flop PH8	S/PH8 = PH7 NBR + ...	
	Enable clock T4L	R/NT4L = S/T4L = FAB0/1 PH7 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function
PH8 T4L	A0-A31 → PR0-PR31		Address shifted into bit positions 15 through 31 and byte selection bits into positions 0 and 1

	Circular shift right two digit places with A30 ↛ A0, and A31 ↛ A1	AXPRR2 = FATR PH8 + ...
		A0EN/2 = A30 ARCYC
		A1EN/2 = A31 ARCYC
		ARCYC = ALCYC = NFAMDSF	

Mnemonic: TBS (41)

(Continued)

Table 3-104. Translate Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH8 T4L (Cont.)	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T4L	R/NT4L = S/T4L = FAB0/1 PH8 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function
PH9 T4L	A0-A31 → S0-S31	SXA = FAB0/1 PH9 + ...	
	S15-S31 $\nrightarrow$ P15-P31, S0 $\nrightarrow$ P32, S1 $\nrightarrow$ P33	PXS/1 = FATR PH9 + ...	Source address clocked into P-register. Byte selection bits clocked into flip-flops P32, P33
	Generate a memory request for byte from source address	MRQ = FATR PH9 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	Inhibits transmission of another clock until data signal is received
	Subtract one from byte count in E-register	MCTE1 = FATR PH9 + ...	
		ES4 = NFABS NSW1	
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L	
PH10 T6L	MB0-MB31 → C0-C31	CXMB = DGC	Data word from modified source address
	Byte selected by bits P32 and P33, C $\nrightarrow$ D24-D31	DXCBP = FATR PH10 + ...	See equations in PH7 of Move Byte String
	B0-B31 → S0-S31	SXB = FATR PH10 + ...	Destination address
	S15-S31 $\nrightarrow$ P15-P31, S0 $\nrightarrow$ P32, S1 $\nrightarrow$ P33	PXS/1 = FATR PH10 + ...	Destination address clocked into P-register. Byte selection bits clocked into flip-flops P32, P33
	Clear A-register	AX/1 = FATR PH10 + ...	
	If memory protection violation occurs, set flip-flop SW2	S/SW2 = FAB0 PH10 PROTECTD + ...	
	Memory request generated to store first byte	MRQ = FATR 07 PH10 NPROTECTD NSW2 + ...	
	Set flip-flop NRPX	S/NRPX = FATR PH10 + ...	

(Continued)

Table 3-104. Translate Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T6L (Cont.)	Force ones into flip-flops CS0-CS31	CSX1 = NPRX	For use during upward byte alignment in PH5
	Branch to PH5	S/PH5 = BRPH5 = FATR 07 PH10 + ...	Cycle repeated until flip-flop PHA is finally set during PH5
	Enable clock T8L	R/NT8L = T8L = FATR 07 PH10 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	Inhibits transmission of another clock until signal is received that data lines have been strobed
PH9A T8L	(Entered from PH5 when flip-flop PHA is set)		
	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FATRA PH9 + ...	Destination address and byte selection bits
	Circular shift one bit position to the left with S0 ↛ A31, S1-S31 ↛ A0-A30	AXSL1 = FAB0A PH9 + ...	Places address in bit positions 14 through 30 and byte selection bits in 0 and 31
	Generate memory request for next instruction	MRQ/1 = FAB0A/1 PH9 + ...	
	Q15-Q31 ↛ P15-P31	PXQ = MRQ/1 + ...	Address of next instruction
	P15-P31 → LM15-LB31		
	If flip-flop SW2 is set, add one to count in E-register	PCTE1 = FATRA PH9 SW2 + ...	Count points to last byte transferred
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH10A T6L	A0-A31 → S0-S31	SXA = FATRA PH10 + ...	Destination address and byte selection bits
	Circular shift one bit position to the left with S0 ↛ A31, S1-S31 ↛ A0-A30	AXSL1 = FATRA PH10 + ...	Places address in bit positions 13 through 29 and byte selection bits in positions 30 and 31
	E0-E7 ↛ A0-A7	AXE = FATRA PH10 + ...	Modified byte count clocked into A-register

(Continued)

Table 3-104. Translate Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10A T6L (Cont.)	Set flip-flop IEN	S/IEN = FATRA PH10 + ...	Enables interrupt to be acknowledged
	Set flip-flop DRQ	S/DRQ = FATRA PH10 + ...	Inhibits transmission of another clock until signal is received from memory
	Force a one on address line LR31	R/NLR31/2 = LR31/2 = FATRA PH10 + ...	Select odd numbered private memory register
	Set flip-flop PH11	S/PH11 = PH10 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FATRA PH10 + ...	
PH11A T8L	A0-A31 → S0-S31	SXA = FATRA PH11 + ...	
	S0-S31 → RW0-RW31	RWXS = FATRA PH11 + ...	Modified destination address and byte count
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	Store modified destination address and byte count in odd numbered private memory register
	If flip-flop SW2 is true, generate signal (S/TRACC4/1) and set TRAP flip-flop	(S/TRACC4/1) = FAB0A PH11 SW2 NTRAP + ... S/TRAP = (S/TRACC4/1) + ...	Trap because of attempted memory protection violation
	If SW2 is not set and byte count in E-register is all zeros, generate signal ENDE	ENDE = FAB0A PH11 EZ + ...	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	For next instruction
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: TBS (41)

Table 3-105. Translate and Test Byte String, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1 T6L	Generate signal PROTECTDIS	PROTECTDIS = FAB0 + ...	Disables trapping function for memory protection
	Clear A-, B-, and E-registers	AX = PRE1 + ...	
		BX = PRE1 NINTRAPF + ...	
		EX = PRE1 + ...	
	P15-P31 $\nrightarrow$ Q15-Q31	QXP = PRE1 NANLZ + ...	
	Set flip-flop PRE2	S/PRE2 = NPREIM PRE1 N(S/INTRAPF) + ...	
	Enable clock T4RL	T4RL = PREP + ...	
PRE2 T4RL	Force a one on LR31	R/NLR31/2 = LR31/2 = FATR (PRE2 NIA) + ...	Selects odd numbered private memory register
	Set flip-flop EXU	S/EXU = (S/PH1/1) NCLEAR	
	Set flip-flop PH1	S/PH1 = (S/PH1/1) NCLEAR NBR	
		(S/PH1/1) = NPREDO (PRE2 NIA) + ...	
	Enable clock T4RL	T4RL = PREP + ...	
PH1 T4RL	C12-C31 $\nrightarrow$ D12-D31, 0 $\nrightarrow$ D0-D11	DXC/3 = FATR PH1 + ...	Displacement value
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FATR PH1 + ...	Destination address and count
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Set flip-flop PH2	S/PH2 = PH1 NBR + ...	
	Enable clock T6RL	T6RL = FAB0/1 PH1	
PH2 T6RL	A0-A31 $\longrightarrow$ S0-S31	SXA = FATR PH2 + ...	Destination address and count
	S0-S7 $\nrightarrow$ E0-E7	EXS = FATR PH2 + ...	Byte count
	S0-S31 $\nrightarrow$ B0-B31	BXS = FAB0 PH2 + ...	Destination address
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FATR PH2 NRZ + ...	Source address and mask
	If R-field is zero, force ones into flip-flops CS0-CS7	CSX1/5 = FATR PH2 RZ + ...	Generate a mask

Mnemonic: TTBS (40)

(Continued)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6RL (Cont.)	Force a one on LR31 address line	R/NLR31/2 = LR31/2 = FABO PH2 + ...	Selects odd numbered private memory register
	Set flip-flop DRQ	S/DRQ = FABO/1 EXU + ...	No function. Memory request has not been made
	Set flip-flop PH3	S/PH3 = PH2 NBR + ...	
	Enable clock T10L	R/NT10L = FABO PH2 + ...	
PH3 T10L	A0-A31 + D0-D31 + CS0-CS31 → S0-S31	SXADD = FATR PH3 + ...	Source address plus displacement value plus mask if RZ
	S0-S31 → RW0-RW31	RWXS = FABO PH3 + ...	Store modified source address and mask in odd numbered private memory register
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = RWXS	
	Circular shift B-register left two bit positions	BXBR2 = FATR PH3 + ...	Places destination address in bit positions 15 through 31 and byte selection bits in positions 0 and 1
	If first translation of byte, set flip-flop SW1	S/SW1 = FATR PH3 + ...	
	Set flip-flop DRQ	S/DRQ = FABO/1 EXU + ...	No function. Memory request has not been made
	Reset flip-flop CC4	R/CC4 = FABO/1 PH3 NO7 + ...	
	Set flip-flop PH5	S/PH5 = BRPH5 = FABO/1 PH3 + ...	
	Enable clock T8L	R/NT8L = T8L = FABO/1 PH3 + ...	
PH5 T8L	If first pass (SW1 set), B0-B31 → S0-S31	SXB = FATR5 SW1 + ...	
	S15-S31 -/→ P15-P31, S0 -/→ P32, S1 -/→ P33	PXS/1 = FATR5 SW1 + ...	Destination address and byte selection bits
	Reset flip-flop SW1	R/SW1 = FATR PH5 + ...	
	Generate memory request for byte	MRQ = FATR5 NFATRFINISH + ...	
	If address was in a protected area of memory, set flip-flop SW2	S/SW2 = FATR PH5 PROTECTD + ...	

Mnemonic: TTBS (40)

(Continued)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T8L (Cont.)	If flip-flop SW2 is set, compare flip-flop CC4 is set; if E-register has been counted down to zero, or an interrupt occurs, set flip-flops PHA and PH9	S/PHA = FATR5 FATRFINISH + ...	FATRFINISH = SW2 + PROTECTD + CC4 + EZ + INT
		S/PH9 = BRPH9 = FATR5 FATRFINISH + . . .	
		FATRFINISH = (SW2 + PROTECTD) + EZ + INT	If signal FATRFINISH is true, instruction branches to PH9A
	If flip-flop SW2 is not set, add one to byte selection bits P32 and P33	PCTP1 = FATR5 NSW1 (NSW2 NPROTECTD) + ...	Selects next byte
		PA33 = FATR5 NSW1 (NSW2 NPROTECTD) + ...	
	Force a one on address line LR31	R/NLR31/2 = LR31/2 = FAB0 NFABS PH5 + ...	Selects register where source address and mask were stored
	Set flip-flop PH6	S/PH6 = PH5 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FAB0 PH5 + ...	
	If not first pass (NSW1), D24-D31 ⟶ K23-K30		
	K23-K30 ⟶ S0-S7, S8-S15, S16-S23		
	D24-D31 ⟶ S24-S31	SXUAB = FATR NSW1 PH5 + ...	
	Byte selected by P32 and P33 ⟶ MB0-MB7, or MB8-MB15, or MB15-MB23, or MB24-MB31	MBXS/0, MBXS/1, MBXS/2, or MBXS/3	Byte selected by P32 and P33 gated onto data lines. See equations in PH10 of Move Byte String
	Generate memory write-byte signal MWB	MWB = FATR NSW1 PH5 + ...	Store byte in destination address location of core memory
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	Inhibits transmission of another clock until signal received from memory
	Set flip-flop PH6	S/PH6 = PH5 NBR + ...	
	Enable clock T8L	R/NT8L = S/T8L = FAB0 PH5	

Mnemonic: TTBS (40)

(Continued)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6 T8L	MB0-MB31 → C0-C31	CXMB = DGC	Word from destination address
	Byte selected by P32 and P33 -/→ D24-D31	DXCBP = FATR PH6 + ...	Byte from destination address downward aligned into D-register. See equations in PH7 of Move Byte String
	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FATR PH6 + ...	
	S0-S31 -/→ B0-B31	BXS = FATR PH6 + ...	Destination address stored in B-register
	RR0-RR31 -/→ A0-A31	AXRR = FATR PH6 + ...	Modified source address clocked into A-register from private memory
	Set flip-flop PH7	S/PH7 = PH6 NBR + ...	
	Set flip-flpp DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH7 T6L	A0-A31 + D0-D31 → S0-S31	SXADD = FATR PH7 + ...	Source address and byte added
	S0-S31 -/→ A0-A31	AXS = FATR PH7 + ...	Result -/→ A-register
	Clear D-register	DX/1 = FATR PH7 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Set flip-flop PH8	S/PH8 = PH7 NBR + ...	
	Enable clock T4L	R/NT4L = S/T4L = FAB0/1 PH7	
PH8 T4L	A0-A31 → PR0-PR31		
	Circular shift right two digit places with A30 -/→ A0, and A31 -/→ A1	AXPRR2 = FATR PH8 + ...	Address shifted into bit positions 15 through 31 and byte selection bits into positions 0 and 1
		A0EN/2 = A30 ARCYC	
		A1EN/2 = A31 ARCYC	

Mnemonic: TTBS (40)

(Continued)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH8 T4L (Cont.)		ARCYC = ALCYC = NFAMDSF	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T4L	R/NT4L = S/T4L = FAB0/1 PH8 + ...	
PH9 T4L	A0-A31 $\longrightarrow$ S0-S31	SXA = FAB0/1 PH9 + ...	
	S15-S31 $\nrightarrow$ P15-P31, S0 $\nrightarrow$ P32, S1 $\nrightarrow$ P33	PXS/1 = FATR PH9 + ...	Source address clocked into P-register. Byte selection bits clocked into flip-flops P32, P33
	Generate memory request for byte from source address	MRQ = FATR PH9 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	Inhibits transmission of another clock until data signal received
	Subtract one from count in E-register	MTCE1 = FATR PH9 + ...	
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH10 T6L	MB0-MB31 $\longrightarrow$ C0-C31	CXMB = DGC	Data word from modified source address
	Byte selected by bits P32 and P33, C $\nrightarrow$ D24-D31	DXCBP = FATR PH10 + ...	See equations in PH7 of Move Byte String
	B30-B31 $\longrightarrow$ S0-S31	SXB = FATR PH10 + ...	Destination address
	S15-S31 $\nrightarrow$ P15-P31, S0 $\nrightarrow$ P32, S1 $\nrightarrow$ P33	PXS/1 = FATR PH10 + ...	Destination address clocked into P-register. Byte selection bits clocked into flip-flops P32, P33
	Clear A-register	AX/1 = FATR PH10 + ...	
	If memory protection violation occurs, set flip-flop SW2	S/SW2 = FAB0 PH10 PROTECTD + ...	
	Set flip-flop NPRX	S/NPRX = FATR PH10 + ...	

(Continued)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T6L (Cont.)	Force ones into flip-flops CS0-CS31	CSX1 = NPRX	For upward alignment during PH11
	Force a one onto address line LR31	R/NLR31/2 = LR31/2 = 0U4 0L0 NPHA PH10 + ...	Selects odd numbered private memory register to obtain mask
	Generate signal (S/CXS)	(S/CXS) = 0U4 0L0 NPHA PH10 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Set flip-flop PH11	S/PH11 = PH10 NBR + ...	
	Enable clock T10L	R/NT10L = T10L = 0U4 0L0 NPHA PH10 + ...	
PH11 T10L	D24-D31 $\longrightarrow$ K23-K30		Source byte
	K23-K30 $\longrightarrow$ S0-S7	SXUAB = FATR PH11 + ...	See equations in PH10 of Move Byte String
	S0-S7 $\longrightarrow$ C0-C7	CXS = (generated during PH10)	Source byte upward aligned into C-register and clocked into D-register
	C0-C7 $\not\longrightarrow$ D0-D7	DXC = FATR PH11 + ...	
	RR0-RR31 $\not\longrightarrow$ A0-A31	AXRR = FATR PH11 + ...	Source address and mask
	Set flip-flop NPRX	S/NPRX = FATR PH11 + ...	For AND operation during PH12
	Force ones into flip-flops CS0-CS31	CSX1 = NPRX + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Set flip-flop PH12	S/PH12 = PH11 NBR + ...	
	Enable clock T4RL	T4RL = FATR PH11 + ...	
PH12 T4RL	A0-A31 + D0-D31 + CS0-CS31 $\longrightarrow$ S0-S31	SXPR = NPRX NSDIS + ...	Logical AND operation between mask and source byte
	S0-S31 $\not\longrightarrow$ A0-A31	AXS = FATR PH12 + ...	Result $\not\longrightarrow$ A-register
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made

(Continued)

Mnemonic: TTBS (40)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH12 T4RL (Cont.)	Set flip-flop PH13	S/PH13 = PH12 NBR + ...	
	Enable clock T8L	R/NT8L = T8L = FATR PH12	
PH13 T8L	If any ones are compared, ones are contained in A-register and flip-flop CC4 is set	S/CC4 = FATR NSW2 NA007Z PH13 + ...	
	If a comparison has been found and bits R28 through R30 are not all zeros, and R31 is a zero A0-A31 → S0-S31	SXA = FATR NSW2 NA0007Z NRZ NR31 PH13 + ...	
	S0-S7 → RW0-RW7	RWXS/0 = RWB0 RWB0 = FATR NSW2 NA0007Z PH13 NRZ NR31 + ...	Store new byte in even numbered private memory register in place of mask
	Set flip-flop PH5	S/PH5 = BRPH5 = FATR PH13 + ...	
	Set flip-flop DRQ	S/DRQ = FAB0/1 EXU + ...	No function. No memory request made
	Enable clock T8L	R/NT8L = T8L = FAB0/1 PH13 + ...	
	Phases PH5 through PH13 are repeated until flip-flop PHA is set, and the instruction then branches to PH9A		
PH9A T8L	P15-P31 → S15-S31, P32 → S0, P33 → S1	SXP = FATRA PH9 + ...	Destination address and byte selection bits
	Circular shift left one bit position with S0 ↛ A31, S1-S31 ↛ A0-A30	AXSL1 = FAB0A PH9 + ...	Place address in bit positions 14 through 30 and byte selection bits in positions 0 and 31
	Generate memory request for next instruction	MRQ/1 = FAB0A/1 PH9 + ...	
	Q15-Q31 ↛ P15-P31	PXQ = MRQ/1 + ...	Address of next instruction
	P15-P31 → LM15-LB31		
	If flip-flop SW2 is set, add one to count in E-register	PCTE1 = FATRA PH9 SW2 + ...	Count points to last byte compared

Mnemonic: TTBS (40)

(Continued)

Table 3-105. Translate and Test Byte String, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH9A T8L (Cont.)	Set flip-flop PH10	S/PH10	= PH9 NBR + . . .	
	Enable Clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
PH10A T6L	A0-A31 → S0-S31	SXA	= FATRA PH10 + . . .	Destination address and byte selection bits
	Circular shift left one bit position with S0 ↛ A31, S1-S31 ↛ A0-A30	AXSL1	= FATRA PH10 + . . .	Places address in bit positions 13 through 29 and byte selection bits in positions 30 and 31
	E0-E7 ↛ A0-A7	AXE	= FATRA PH10 + . . .	Modified byte count clocked into A-register
	Set flip-flop IEN	S/IEN	= FATRA PH10 + . . .	Enables interrupt to be acknowledged
	Set flip-flop DRQ	S/DRQ	= FATRA PH10 + . . .	Inhibits transmission of another clock until signal is received from memory
	Force a one on address line LR31	R/NLR31/2	= LR31/2 = FATRA PH10 + . . .	Select odd number private memory register
	Set flip-flop PH11	S/PH11	= PH10 NBR + . . .	
	Enable clock T8L	R/NT8L	= T8L = FATRA PH10 + . . .	
PH11A T8L	A0-A31 → S0-S31	SXA	= FATRA PH11 + . . .	
	S0-S31 → RW0-RW31	RWXS	= FATRA PH11 + . . .	Modified destination address and byte count
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3	= RWXS	Stored modified destination address and byte count in odd numbered private memory register
	If E-register is all zeros, reset flip-flop CC4	R/CC4	= FATRA NO7 PH11 EZ + . . .	
	If flip-flop SW2 is true, generate signal (S/TRACC4/1) and set flip-flop TRAP	(S/TRACC4/1)	= FAB0A PH11 SW2 NTRAP + . . .	Trap because of attempted memory protection violation
		S/TRAP	= (S/TRACC4/1) + . . .	
	If byte count in E-register is all zeros or a byte comparison was made (CC4 set), generate signal ENDE	ENDE	= FAB0A PH11 EZ + FAB0A/1 PH11 CC4 N07 + . . .	
		S/PRE1	= ENDE (NHALT + FUEXU) N(S/INTRAPF)	For next instruction
	Enable clock T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	

Mneumonic: TTBS (40)

3-233 Push Word (PSW, 09, 89)

The PSW instruction pushes the contents of register R into the location that is one greater than the location addressed by the current TSA defined by the stack pointer doubleword. Figure 3-185 describes the purposes and the geometry of the push-down stack as they relate to the PSW instruction.

The effective doubleword addresses (70C in the example in figure 3-185) in the PSW instruction addresses TSA which, in turn, addresses the last word (1503) that was pushed into the stack. A one is added to this address, and the contents of the R-register are pushed into location TSA + 1 (1504) by the PSW instruction. The count word, CW (the least significant half of the effective doubleword), contains two counts — a space count that defines the number of spaces remaining in the stack and a word count that defines the number of words currently in the stack. The sum of these two counts is equal to the length of the stack.

Words are pushed into the stack in numerically ascending order and are pulled from the stack in numerically descending order. At the end of the PSW instruction described in figure 3-185, location 1504 will contain the contents of register R, the top of stack address in 70C will be 1504, and the space and word counts in the count word 70D will be 1B and 05, respectively.

The preparation sequence for the PSW instruction is the same as the general preparation sequence given under the Preparation Sequence. The execution sequence for the Push Word instruction is described in figure 3-186.

In PH1 the A-register is set to equal X'00000001'. The contents of the A-register are two's complemented and are transferred to the B-register in PH2, B0-B15, which are reset to zeros, are the only exception. B equals X'0000FFFF'. In PH3 the A-register is halfword aligned upward, and A equals X'00010000'. The contents of A and B are merged (OR-gated) in PH4, and the result is placed into A. The A-register now contains X'0001FFFF'. MRQ is enabled to read the contents of the count word. In PH5 carry bit K15 is forced true, and the A-register is again two's complemented, which effectively reverses the two halves of the A-register. A now equals X'FFFF0001'. A0-A15 contains a one's decrement value, and A16-A31 contains a one's increment value.

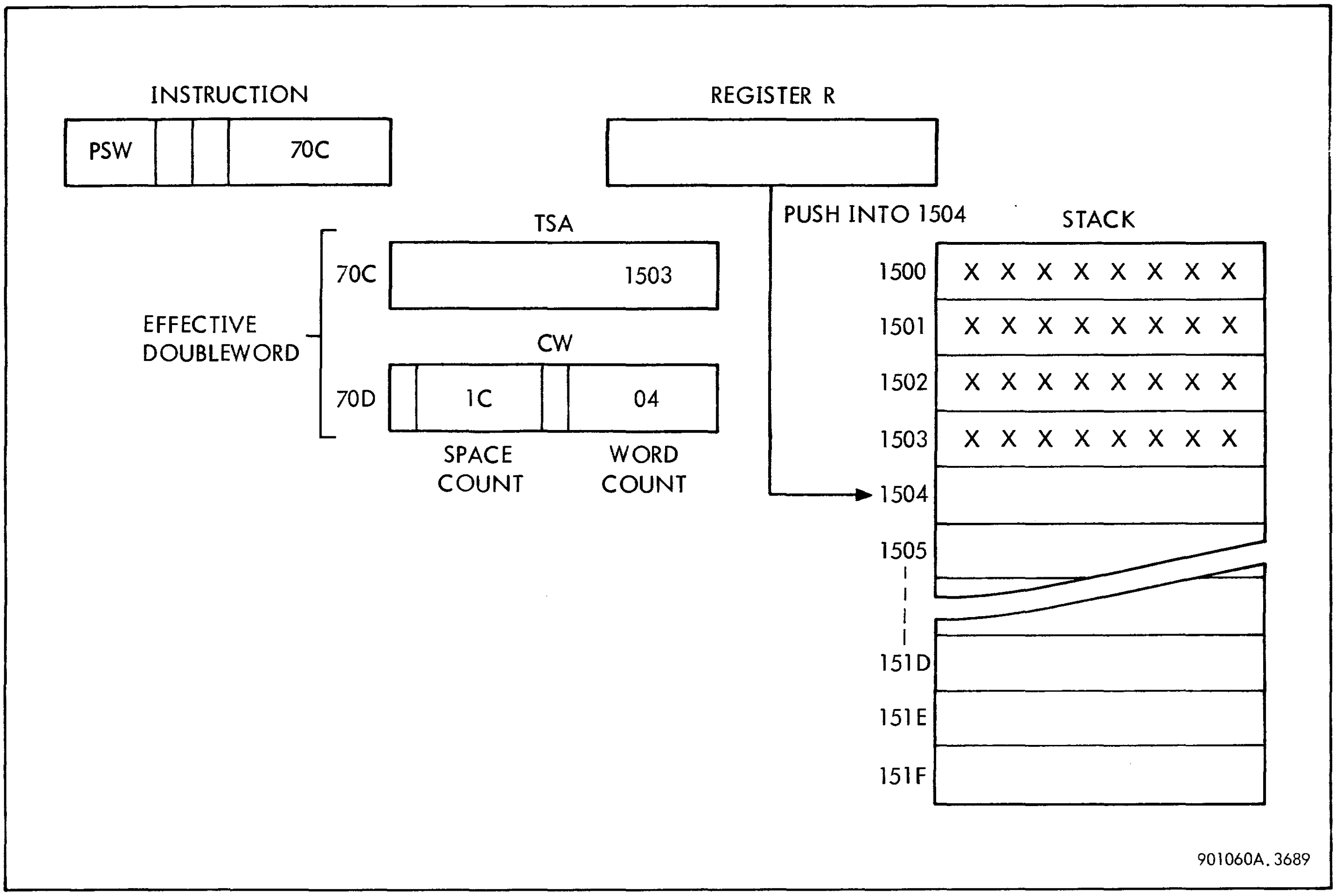


Figure 3-185. Doubleword-Stack Relationship for PSW Instruction

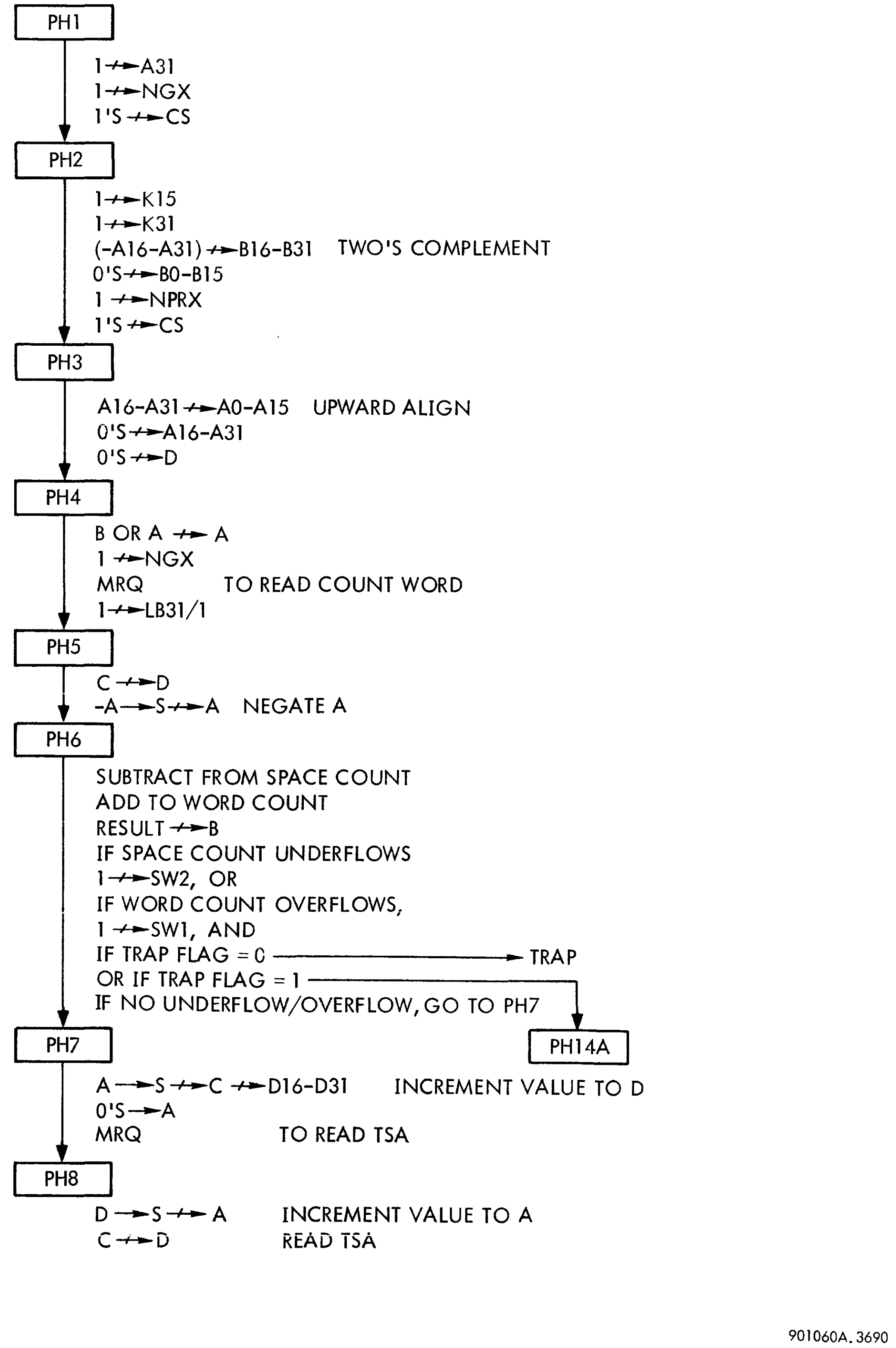


Figure 3-186. Push Word, Logic Sequence Diagram (Sheet 1 of 2)

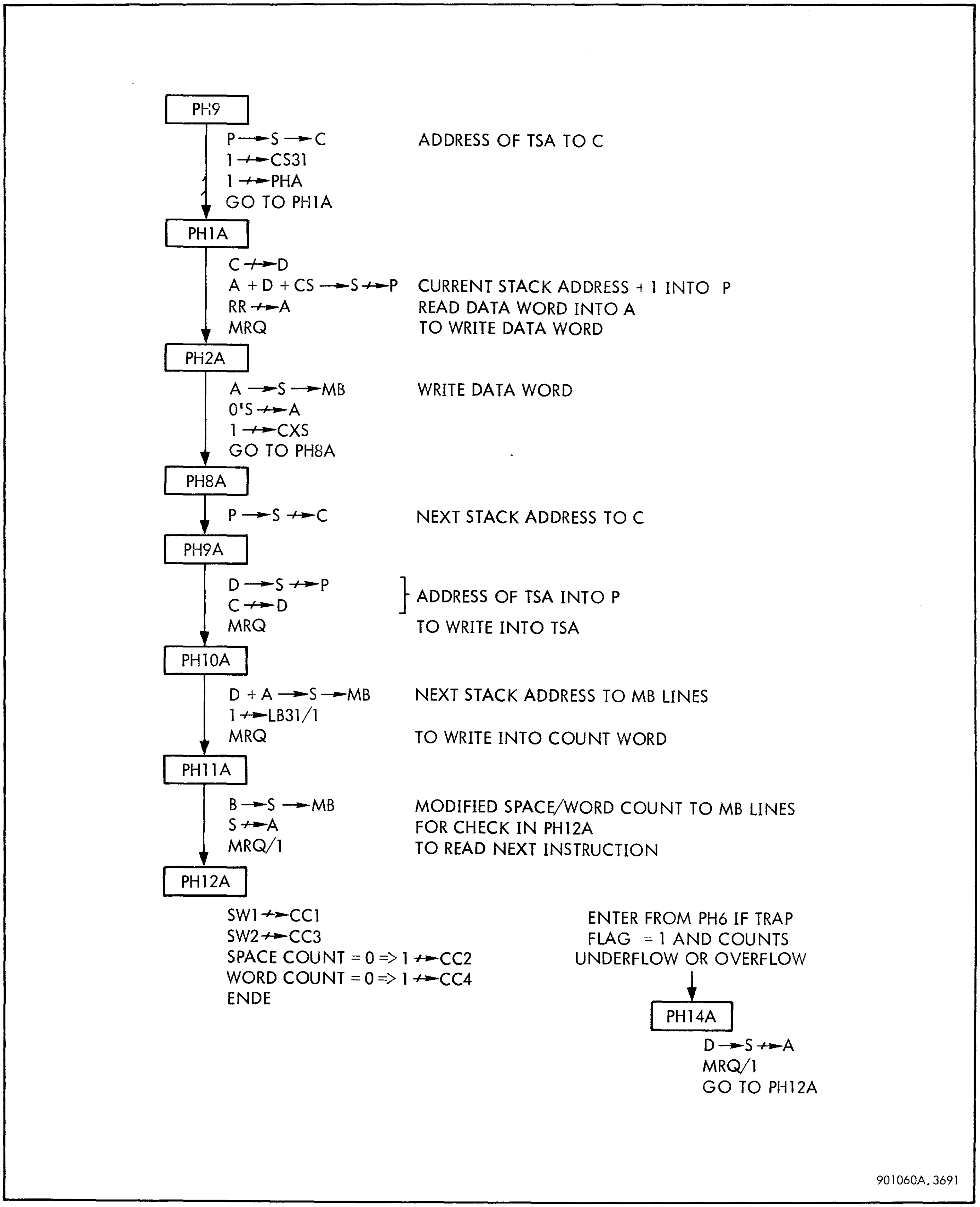


Figure 3-186. Push Word, Logic Sequence Diagram (Sheet 2 of 2)

In PH6 the count word is in D. The contents of A and D are added, which effectively adds one to the word count and subtracts one from the space count. The result, which is the modified count word, is transferred to the B-register and is held there until it is used in PH11A. Carry bit K15 is inhibited during this addition. This prevents any carry from the word count from affecting the space count addition.

SW1 is set if the space count underflows. SW2 is set if the word count overflows. Space count underflow and word count overflow can occur simultaneously only if the stack contains 32,768 words. Space count underflow and word count overflow are detected by comparing bit 0 of the original space count in D with the result in sum bus bit 0 and by comparing bit 16 of the original word count in D with the result in sum bus bit 16. If bit 0 of the count word TS (trap on space underflow) is a zero, a trap occurs because of a space count underflow. If bit 16 of the count word TW (trap on word overflow) is a zero, á trap occurs because of a word count overflow. Flip-flop SW1 is not set if a trap occurs because of a space count underflow since D0 must be a one to set SW1 or a zero to cause a trap. Flip-flop SW2 is not set if a trap occurs because of a word count overflow, since D16 must be a zero to cause a trap or a one to set SW2.

If bi! 0 of the count word TS (trap on space underflow) is a one and if the space count underflowed, the CPU aborts the PSW instruction by branching to PH14A. If bit 16 of the count word TW (trap on word overflow) is a one and if the word count overflowed, the CPU aborts the PSW instruction by branching to PH14A. If no underflow/overflow occurs, the instruction sequences from PH6 to PH7. A memory request is made to read TSA in PH7.

Starting with PH7 and ending with PH12A, four pieces of data (two addresses and two words) are entered and are saved in the various registers. The following list will aid in keeping track of the data, where: ATSA is the address of TSA; TSA is the address of the current stack word; SW is the contents of the current stack word; and CW is the contents of the count word.

PHASE	ATSA	TSA	SW	CW
PH7	P			S B
PH8	P	C↛D		B
PH9	P→C	D		B
PH1A	C↛D	D+1↛P	RR→A	B
PH2A	D	P	A→MB	B
PH8A	D	P→C		B
PH9A	D↛P	C↛D		B
PH10A	P	(TSA+1)→MB		B
PH11A	P+1			B→MB, B↛A
PH12A				A
PH14A				D↛A

A one is added to TSA in PH1A, and the result is placed in the P-register. The contents of register R are clocked into A, and a memory request is made to write into the top of stock location (TSA+1).

In PH10A the new top of the stack address, which has been transferred to D, is written into the TSA location. The modified space and word count is transferred from B to the MB lines in PH11A and is written into the count word location.

In PH12A the condition codes are set. CC1 is set if the space count underflowed in PH6; CC3 is set if the word count overflowed in PH6; CC2 is set if the modified space count is currently equal to zero; and CC4 is set if the modified word count is equal to zero.

A sequence chart of the Push Word instruction is given in table 3-106.

3-234 Pull Word (PLW 08, 88)

The Pull Word instruction loads the private memory register specified by the R-field of the instruction with the contents of the word whose address currently represents the top of the push-down stack defined by the stack pointer doubleword located at the effective doubleword address of the instruction. Figure 3-187 describes the purposes of the effective doubleword and the geometry of the push-down stack as they related to the PLW instruction.

The effective doubleword address (70C in the example in figure 3-187) in the PLW instruction addresses TSA (top of stack address) which, in turn, addresses the last word (1503) that was pushed into the stack. This is the word that is pulled out of the stack and is placed into register R by the PLW instruction. The count word CW the least significant half of the effective doubleword) contains two counts, a space count that defines the number of spaces remaining in the stack, and a word count that defines the number of words currently in the stack.

Table 3-106. Push Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	1 -/→ DRQ	S/DRQ = FAST EXU + FASTA EXU + ...	DRQ is set for all execution phases
PH1 T4RL	1 -/→ A31	A31X1 = FAST PH1 NO6 + ...	Set one in A-register
	1 -/→ NGX	(S/NGX) = FAST PH1 NO2 + ...	Prepare for negation
	1's -/→ CS0-CS31	CSX1 = (S/NGX)	
PH2 T6L	1 → G1619	G1619 = FAST PH2 + ...	Causes zeros in B0-B15
	1 → K15	K15 = G1619/1	
	-(A16-A31) → S16-S31 0's → S0-S15	SXADD/1 = NGX NFAMDSF + ...	Two's complement
	S0-S15 -/→ B0-B31	BXS = FAST PH2 + ...	Results in B = 0000FFFF
	1 -/→ NPRX	CSX1 = (S/NPRX)	Prepare for upward alignment
	1 → K31	K31 = NGX	
PH3 T6L	A16-A31 → S0-S15	SXUAH = FAST PH3 + ...	Upward align halfword. Results in A = 00010000
	S0-S15 -/→ A0-A15	AXS/2 = FAST PH3 + ...	
	0's -/→ D0-D31	DX/1 = FAST PH3 + ...	
PH4 T6L	B0-B31 → S0-S31	SXB = FAST PH4 + ...	Increment-decrement values. This OR function results in A = 0001FFFF. Combines positive and negative halves of modifier. Positive in A0-A15. Negative in A16-A31
	A0-A31 → S0-S31	SXA = FAST PH4 + ...	
	S0-S31 -/→ A0-A31	AXS = FAST PH4 + ...	
	Enable memory request	MRQ = FAST PH4 + ...	To read space and word count
	1 -/→ LB31/1	(S/LB31/1) = FAST PH4 + ... R/LB31/1 = •	Address low order half of stack pointer doubleword
PH5 T6L	C0-C31 -/→ D0-D31	DXC/6 = FAST PH5 + ...	Read space and word count
	(-A0-A31) → S0-S31	AXS = FAST PH5 O7 + ...	Negate A-register. Reverses two halves of A-register
	S0-S31 -/→ A0-A31	G1619 = FAST PH5 + ...	

Mnemonic: PSW (09, 89)

(Continued)

Table 3-106. Push Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T6L (Cont.)		K15 = G1619	
	Reset flip-flop NT10L	(S/T10L) = FAST PH5 + ...	
PH6 T10L	A0-A31 + D0-D31 → S0-S31	SXADD = FAST PH6 + ...	Add one to word count, subtract one from space count. Modifier is in A-register
	Force 0 → K15	K15 = N(FAST PH) (...)+...	Prevent carry from word field to space field
	S0-S31 ↛ B0-B31	BXS = FAST PH6 + ...	Update count word to B-register
	D16 NS16 => 1 ↛ SW2, or	S/SW2 = FAST PH6 D16 NS16 + ...	Word count overflow
	D0 NS0 => 1 ↛ SW1	S/SW1 = FAST PH6 D0 NS0 + ...	Space count underflow
	Trap, or	(S/TR30/1) = FAST PH6 ND16 S16 + FAST PH6 ND0 S0 + ...	If trap flag = 1, trap with condition code unchanged
		S/TRAP = (S/TR30/1) + ...	
	Go to PH14A	BRPH14 = FAST PH6 D16 NS16 + FAST PH6 D0 NS0 + ...	If trap flag = 0, go to phase 14A to abort instruction
	1 ↛ PHA	S/PHA = FAST PH6 D0 NS0 + ...	Bit 16 or bit 0 has changed because of an underflow or overflow
	1 ↛ CXS	S/CXS = FAST PH6 + ...	
		S/T8L = (S/CXS) + ...	
PH7 T8L	0's ↛ A0-A31	AX/1 = FAST PH7 + ...	
	Enable memory request	MRQ = FAST PH7 + ...	To read top-of-stack address
PH8 T6L	MB0-MB31 → C0-C31	CXMB = DGC	
	C0-C31 ↛ D0-D31	DXC/6 = FAST PH8 + ...	Read top-of-stack address D-register via C-register
	1 ↛ CXS	S/CXS = FAST PH8 NOL3 + ...	Preset for PH9
		S/T8L = (S/CXS) + ...	

Mnemonic: PSW (09, 89)

(Continued)

Table 3-106. Push Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T8L	P16-P31 → S16-S31	SXP = FAST PH9 + ...	Effective address → C-register
	S0-S31 → C0-C31		CXS set in PH8
	1 -/→ CS31	CSX1/8 = FAST PH9 NOL8 NOL3 + ...	To add one to top-of-stack address
	Go to PH1A	BRPH1 = FAST PH9 NOL3 + ...	
	1 -/→ PHA	S/PHA = FAST PH9 + ...	
PH1A T6L	A0-A31 + D0-D31 + CS0-CS31 → S0-S31	SXADD = FASTA PH1 + ...	Add one to top-of-stack address
		FASTA = FAST PHA + ...	
	S16-S31 -/→ P16-P31	PXS = FASTA PH1 NO2 + ...	Transfer to P-register
	RR0-RR31 -/→ A0-A31	AXRR = FASTA PH1 (O4 NO5 O7) + ...	Read data word from private memory
	C0-C31 -/→ D0-D31	DXC/6 = FASTA PH1 + ...	Effective address -/→ D-register
	Enable memory request	MRQ = FASTA PH1 + ...	To write data word into stack
		S/T8L = FASTA PH1 + ...	
PH2A T8L	A0-A31 → S0-S31	SXA = FASTA PH2 + ...	
	S0-S31 → MB0-MB31	MW = FASTA PH2 + ...	Write data word in core memory
	0's -/→ A0-A31	AX/1 = FASTA PH2 OL9 + ...	
	1 -/→ CXS	S/CXS = FASTA PH2 OL9 + ...	Preset for PH8A
	Go to PH8A	BRPH8 = FASTA PH2 NO2 OL9 + ...	
		S/T8L = FASTA PH2 + ...	
PH8A T8L	P16-P31 → S16-S31	SXP = FASTA PH8 + ...	New top-of-stack address into C-register
	S0-S31 → C0-C31		CXS set true in PH2A

Mnemonic: PSW (09, 89)

(Continued)

Table 3-106. Push Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9A T6L	D0-D31 → S0-S31	SXD = FASTA PH9 + ...	Effective address ↛ P-register
	S16-S31 ↛ P16-P31	PXS = FASTA PH9 + ...	
	C0-C31 ↛ D0-D31	DXC/6 = FASTA PH9 + ...	Top-of-stack address ↛ D-register
	Enable memory request	MRQ = FASTA PH9 + ...	To write top-of-stack address
PH10A T6L	D0-D31 + A0-A31 → S0-S31	SXADD = FASTA PH10 + ...	A = 0's, top-of-stack address → core memory in effective word location
	S0-S31 → MB0-MB31	MW = FASTA PH10 + ...	
	1 ↛ LB31/1	(S/LB31/1) = FASTA PH10 + ...	To address low order half of stack pointer doubleword
	Enable memory request	MRQ = FASTA PH10 + ...	To write count word
PH11A T6L	B0-B31 → S0-S31	SXB = FASTA PH11 + ...	Write updated count word in effective address + 1
	S0-S31 → MB0-MB31	MW = FASTA PH11 + ...	
	S0-S31 ↛ A0-A31	AXS = FASTA PH11 + ...	Updated count word to A-register for check in PH12A
	Enable memory request	MRQ/1 = FASTA PH11 + ...	To read next instruction
PH12A T6L	If SW1 = 1, 1 ↛ CC1	S/CC1 = FASTA PH12 SW1 + ...	Space count underflow
	If SW2 = 1, 1 ↛ CC3	S/CC3 = FASTA PH12 SW2 + ...	Word count overflow
	If (A1-A15 = 0), 1 ↛ CC2	S/CC2 = FASTA PH12 A0115Z + ...	Space count zero
	If (A17-A31 = 0), 1 ↛ CC4	S/CC4 = FASTA PH12 A1731Z + ...	Word count zero
		(R/CC) = FASTA PH12 + ...	
	End	ENDE = FASTA PH12 + ...	
	Reset PHA	R/PHA = CLEAR = ENDE	
PH14A T6L	Entered from PH6 if trap has occurred		

Mnemonic: PWS (09, 89)

(Continued)

Table 3-106. Push Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH14A T&L (Cont.)	D0-D31 → S0-S31	SXD = FASTA PH14 + ...	Word and space count ⇸ A-register to check for zero
	S0-S31 ⇸ A0-A31	AXS = FASTA PH14 + ...	
	Enable memory request	MRQ/1 = FASTA PH14 + ...	To read next instruction
	Go to PH12A	BRPH12 = FASTA PH14 + ...	

Mnemonic: PSW (09, 89)

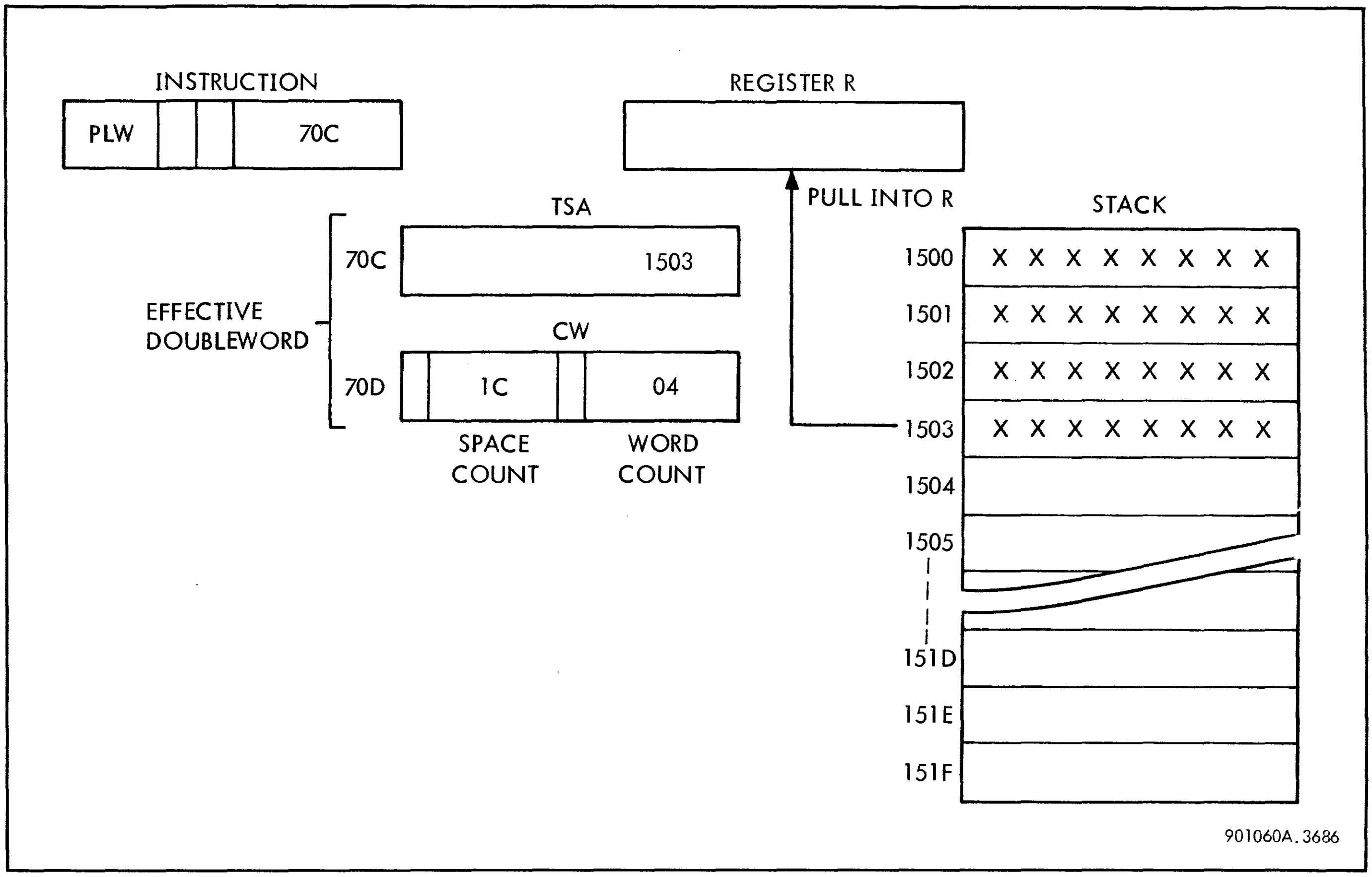


Figure 3-187. Doubleword-Stack Relationship for PLW Instruction

The sum of these two counts is equal to the length of the stack. Words are pushed into the stack in numerically descending order, and are pulled from the stack in numerically descending order. At the end of the PLW instruction described in figure 3-187, register R contains the contents of word location 1503; the top of stack address in 70C is 1502; and space and word counts in the count word 70D are 1D and 03, respectively.

The preparation sequence for the PLW instruction is the same as the general preparation sequence given under the Preparation Sequence. The execution sequence for the Pull Word instruction is described in figure 3-188.

In PH1 the A-register is set to equal X'00000001'. In PH2 the contents of the A-register are two's complemented and are transferred to the B-register. B equals X'0000FFFF'. In PH3 the A-register is half-word upward aligned, and A equals X'00010000'. The contents of A and B are merged (OR-gated) in PH4, and the result is placed in A. The A-register now contains X'0001FFFF', which represents the increment/decrement values to be added to the contents of the count word in PH6.

The count word is read into D at the PH5 clock and in PH6 the contents of A and D are added, effectively adding one to the space count and subtracting one from the word count. The result, which is the modified count word, is transferred to the B-register and is held there for later use. Carry bit K15 is inhibited during this addition to prevent any carry from the word count from affecting the space count addition.

SW1 is set if the space count overflows. If the word count underflows, SW2 is set. Space count overflow and word count underflow can occur simultaneously only if the stack contains 32,768 words. Space count overflow and word count underflow are detected by comparing bit 0 of the original space count in D with the result in sum bus bit 0, and by comparing bit 16 of the original word count in D with the result in sum bus bit 16.

If bit 0 of the count word TS (trap on space overflow) is a zero, a trap occurs because of a space count overflow. If bit 16 of the count word TW (trap on word underflow) is a zero, a trap occurs because of a word count underflow. If a trap occurs because of a space count overflow, flip-flop SW1 is not set, since D0 must be a one to set SW1 or a zero to cause a trap. If a trap occurs because of a word count

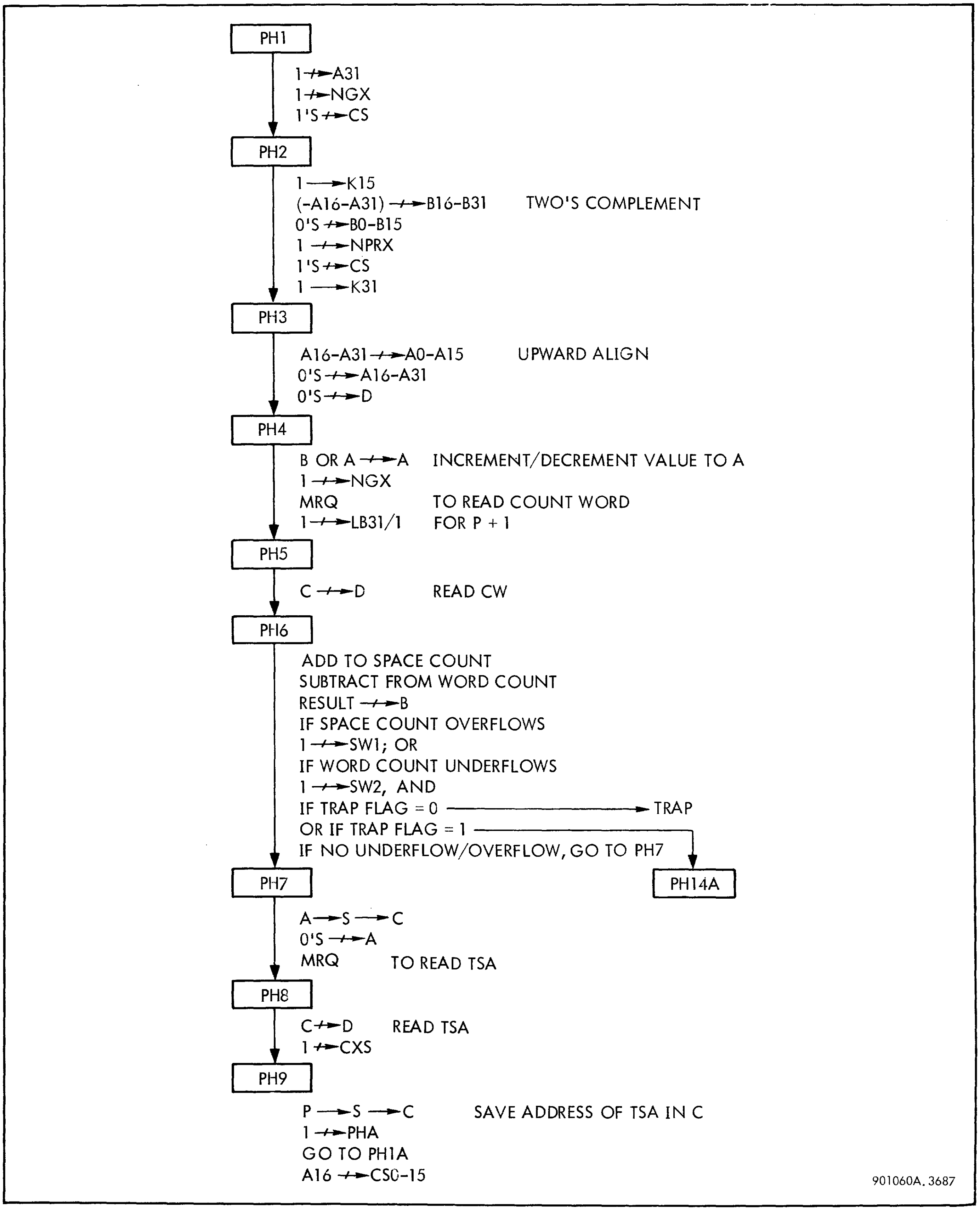
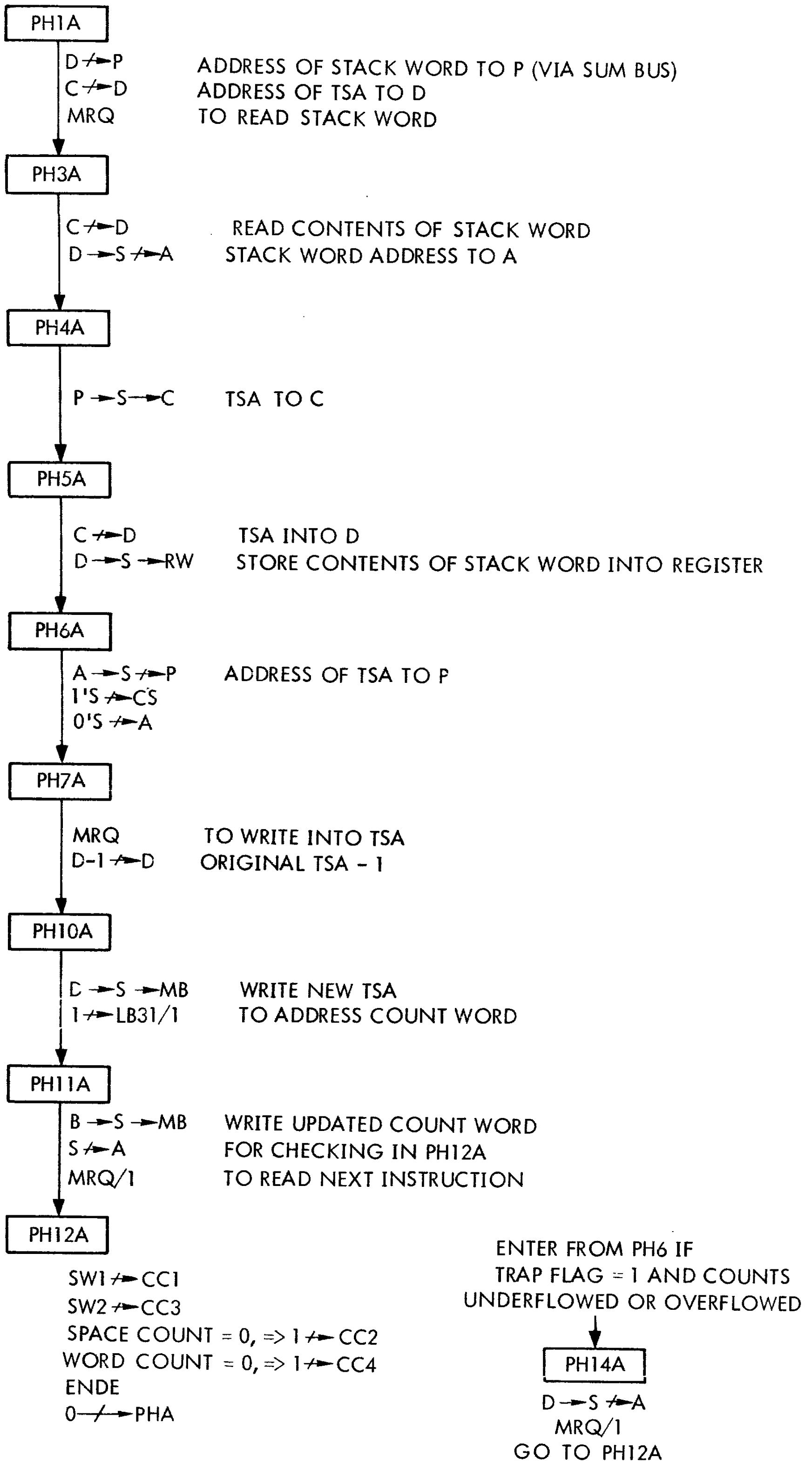


Figure 3-188. Pull Word, Logic Sequence Diagram (Sheet 1 of 2)



901060A.3688

Figure 3-188. Pull Word, Logic Sequence Diagram (Sheet 2 of 2)

underflow, flip-flop SW2 is not set, since D16 must be a zero to cause a trap or a one to set SW2.

If bit 0 of the count word TS (trap on space overflow) is a one and the space count overflowed, the CPU aborts the PLW instruction by branching to PH14A. If bit 16 of the count word TW (trap on word underflow) is a one and the word count underflowed, the CPU aborts the PLW instruction by branching to PH14A. If no underflow/overflow occurs the instruction sequences from PH6 to PH7.

In PH7 a memory request is made to read TSA. Starting with PH7 and ending with PH12A, four pieces of data (two addresses and two words) are entered and are saved in the various registers. The following sequence list will aid in keeping track of the data, where: ATSA is the address of TSA; TSA is the address of the current stack word; SW is the contents of the current stack word; and CW is the contents of the count word.

PHASE	ATSA	TSA	SW	CW
PH7	P			S⇸B
PH8	P	C⇸D		B
PH9	P→C	D		B
PH1A	C⇸D	D⇸P		B
PH3A	D⇸A	P	C⇸D	B
PH4A	A	P→C	D	B
PH5A	A	C⇸D	D⇸RW	B
PH6A	A⇸P	D		B
PH7A	P	D		B
PH10A	P	(TSA-1)→MB		B
PH11A	P+1			B→MB, B⇸A
PH12A				A
PH14A				D⇸A

During PH7A the CS-register contains all ones (-1) and is added to the TSA, effectively subtracting one from the TSA. The modified TSA is written into the TSA location in PH10A. The modified count word, which has been held in the B-register, is written into the count word location in PH11A. The count word is also clocked into the A-register for testing in PH12A.

In PH12A the condition code flip-flops are set. Flip-flop CC1 is set if the space count overflowed in PH6; flip-flop CC3 is set if the word count underflowed in PH6. Flip-flop CC2 is set if the modified space count is currently equal to zero; and flip-flop CC4 is set if the modified word count is equal to zero.

A sequence chart of the Pull Word instruction is given in table 3-107.

3-235 Push Multiple (PSM 0B, 8B)

Before the PSM instruction is executed, the condition code must contain a value equal to the number of words to be pushed into the pushdown stack. Since the condition code consists of four flip-flops, CC1-CC4, no more than 16 words can be pushed into a stack during the execution of any one PSM instruction regardless of the stack length. If the condition code is equal to zero, 16 words will be pushed into the stack.

During the execution of the PSM instruction, the value of the condition code is transferred to the E-register, and a one is added to the contents of TSA to become the address of the first word to be pushed into the stack. Words are pushed into the stack in ascending order. For example, if CC equals three, and the top of stack address is 1517, the stack words will be pushed from private memory into the stack in the following order:

R to (1518)

R+1 to (1519)

R+2 to (151A)

As each word is pushed into the stack, the contents of E are counted down. The words to be pushed have been loaded into the stack when E reaches a count of one. The contents of TSA are modified by adding the value of the condition code to the current TSA and by restoring it to memory. The value of the condition code is added to the word count and subtracted from the space count. The result is stored to the count word in memory.

Table 3-107. Pull Word, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	1 -/→ DRQ	S/DRQ = FAST EXU + FASTA EXU + ...	DRQ is set for all execution phases
PH1 T4RL	1 -/→ A31	A31X1/1 = FAST PH1 NO6 + ...	Set one in A-register
	1 -/→ NGX	(S/NGX) = FAST PH1 NO2 + ...	Prepare for negation
	1's -/→ CS0-CS31	CSX1 = (S/NGX)	
PH2 T6L	1 → G1619/1	G1619/1 = FAST PH2 + ...	Causes zeros in B0-B15
	1 → K15	K15 = G1619/1	
	-(A16-A31) → S16-S31	SXADD/1 = NGX NFAMDSF + ...	Two's complement
	0's → S0-S15		
	S0-S31 -/→ B0-B31	BXS = FAST PH2 + ...	Results in B = 0000FFFF
	1 -/→ NPRX	(S/NPRX) = FAST PH2 + ...	Prepare for upward alignment
	1's -/→ CS0-CS31	CSX1 = (S/NPRX)	
	1 → K31	K31 = NGX	
PH3 T6L	A16-A31 → S0-S15	SXUAH = FAST PH3 + ...	Upward align halfword Results in A = 00010000
	S0-S15 -/→ A0-A15	AXS/2 = FAST PH3 + ...	
	0's -/→ D0-D31	DX/1 = FAST PH3 + ...	
PH4 T6L	B0-B31 → S0-S31	SXB = FAST PH4 + ...	Increment-decrement values -/→ A-register. This OR function results in A = 0001FFFF. Combines positive and negative halves of modifier. Positive in A0-A15,negative in A16-A31
	A0-A31 → S0-S31	SXA = FAST PH4 + ...	
	S0-S31 -/→ A0-A31	AXS = FAST PH4 + ...	
	Enable memory request	MRQ = FAST PH4 + ...	To read space and word count
	1 -/→ LB31/1	(S/LB31/1) = FAST PH4 + ... R/LB31/1 =	Address low order half of stack pointer doubleword

Mnemonic: PLW (08, 88)

(Continued)

Table 3-107. Pull Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T6L	C0-C31 ↛ D0-D31	DXC/6 = FAST PH5 + ...	Read space and word count
	Reset flip-flop NT10L	(S/T10L/1) = FAST PH5	
PH6 T10L	A0-A31 + D0-D31 → S0-S31	SXADD = FAST PH6 + ...	Add one to space count; subtract one from word count. Modifier is in A-register
	Force 0 → K15	K15 = N(FAST PH6) (...)	Inhibit carry to prevent carry from word count from affecting space count
	S0-S31 ↛ B0-B31	BXS = FAST PH6 + ...	Updated count word to B-register
	D16 NS16 => 1 ↛ SW2, or	S/SW2 = FAST PH6 D16 NS16 + ...	Word underflow
	D0 NS0 => 1 ↛ SW1	S/SW1 = FAST PH6 D0 NS0 + ...	Space overflow
	Trap if overflow or underflow	(S/TR30/1) = FAST PH6 ND16 S16 + FAST PH6 ND0 S0 + ...	If trap flag = 0, trap with condition code unchanged
		S/TRAP = (S/TR30/1) + ...	
	Go to PH14A	BRPH14 = FAST PH6 D16 NS16 + FAST PH6 D0 NS0 + ...	If trap flag = 1, go to phase 14A to abort instruction
	1 ↛ PHA	S/PHA = FAST PH6 D0 NS0 + FAST PH6 D16 NS16 + ...	Bit 16 or bit 0 has changed because of an underflow or overflow
	1 ↛ CXS	S/CXS = FAST PH6 + ...	
		S/T8L = (S/CXS) + ...	
PH7 T8L	0's ↛ A0-A31	AX/1 = FAST PH7 + ...	
	Enable memory request	MRQ = FAST PH7 + ...	To read top-of-stack address
PH8 T6L	MB0-MB31 → C0-C31	CXMB = DGC	Read top-of-stack address → C-register
	C0-C31 ↛ D0-D31	DXC/6 = FAST PH8 + ...	↛ D-register
	1 ↛ CXS	S/CXS = FAST PH8 NOL3 + ...	Preset for PH9
		S/T8L = (S/CXS) + ...	

Mnemonic: PLW (08, 88)

(Continued)

Table 3-107. Pull Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T8L	P16-P31 → S16-S31	SXP = FAST PH9 + ...	Effective address → C-register
	S16-S31 ⇸ C16-C31	CSX31 = CSX1/8	
	1 ⇸ CS31	CSX1/8 = FAST PH9 NOL3 NOL8	TSA + 1
	1 ⇸ PHA	S/PHA = FAST PH9 + ...	
	Go to PH1A	BRPH1 = FAST PH9 NOL3 + ...	
PH1A T6L	C0-C31 ⇸ D0-D31	DXC/6 = FASTA PH1 + ...	Effective address → D-register
		FASTA = FAST PHA + ...	
	D0-D31 → S0-S31	SXADD = FASTA PH1 + ...	Top-of-stack address ⇸ P-register
	S16-S31 ⇸ P16-P31	PXS = FASTA PH1 NO2 + ...	
	Enable memory request	MRQ = FASTA PH1 + ...	To read data word
		S/T8L = FASTA PH1 + ...	
	Go to PH3A	BRPH3 = FASTA PH1 (O4 NO5 NO7) + ...	
PH3A T8L	C0-C31 ⇸ D0-D31	DXC/6 = FASTA PH3 + ...	Read data word from core memory
	D0-D31 → S0-S31	SXD = FASTA PH3 + ...	
	S0-S31 ⇸ A0-A31	AXS = FASTA PH3 + ...	Effective address ⇸ A-register
	1 ⇸ CXS	S/CXS = FASTA PH3 OL8 + ...	For S → C in next phase
		S/T8L = FASTA PH4 + ...	
PH4A T8L	P16-P31 → S16-S31	SXP = FASTA PH4 + ...	Top-of-stack address → C-register
	S16-S31 → C16-C31	S/T8L = FASTA PH4 + ...	CXS set in PH3A
PH5A T8L	C0-C31 ⇸ D0-D31	DXC/6 = FASTA PH5 + ...	Top-of-stack address ⇸ D-register
	D0-D31 → S0-S31	SXD = FASTA PH5 + ...	

Mnemonic: PLW (08, 88)

(Continued)

Table 3-107. Pull Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5A T8L (Cont.)	S0-S31 → RW0-RW31	RW = FASTA PH5 + ...	Store data word in private memory register
		S/T8L = FASTA PH5 + ...	
PH6A T6L	A0-A31 → S0-S31	SXA = FASTA PH6 + ...	Effective address ⇸ P-register
	S16-S31 ⇸ P16-P31	PXS = FASTA PH6 + ...	
	1's ⇸ CS0-CS31	CSX1 = FASTA PH6 + ...	To subtract one from top-of-stack address
	0's ⇸ A0-A31	AX/1 = FASTA PH6 + ...	
	1 ⇸ CXS	S/CXS = FASTA PH6 OL8 + ...	For subtraction in nextphase
PH7A T6L	(D0-D31)-1 → S0-S31	SXADD = FASTA PH7	Subtract one from top-of-stack address. Transfer to D-register
	S0-S31 → C0-C31	CSX set in PH6A	
	C0-C31 ⇸ D0-D31	DXC/6 = FASTA PH7	
	Enable memory request	MRQ = FASTA PH7 + ...	To write new top-of-stack address
	Go to PH10A	BRPH10 = FASTA PH7 + ...	
PH10A T6L	D0-D31 + A0-A31 → S0-S31	SXADD = FASTA PH10 + ...	Top-of-stack address minus one → effective memory location
	S0-S31 → MB0-MB31	MW = FASTA PH10 + ...	
	1 ⇸ LB31/1	(S/LB31/1) = FASTA PH10 + ...	To address low order half of stack pointer doubleword
	Enable memory request	MRQ = FASTA PH10 + ...	To write new count word
PH11A T6L	B0-B31 → S0-S31	SXB = FASTA PH11 + ...	Write updated count word in effective address plus one
	S0-S31 → MB0-MB31	MW = FASTA PH11 + ...	
	S0-S31 ⇸ A0-A31	AXS = FASTA PH11 + ...	Updated count word to A-register to check in PH12A
	Enable memory request	MRQ/1 = FASTA PH11 + ...	To read next instruction

Mnemonic: PLW (08, 88)

(Continued)

Table 3-107. Pull Word, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH12A T6L	SW1 = 1 => 1 —/→ CC1	S/CC1 = FASTA PH12 SW1 + ...	Indicates space count overflow
	SW2 = 1 => 1 —/→ CC3	S/CC3 = FASTA PH12 SW2 + ...	Indicates word count underflow
	If (A1-A15 = 0), 1 —/→ CC2	S/CC2 = FASTA PH12 A0115Z + ...	Space count = 0
	If (A17-A31 = 0), 1 —/→ CC4	S/CC4 = FASTA PH12 A1731Z + ...	Word count = 0
		(R/CC) = FASTA PH12 + ...	
	End	ENDE = FASTA PH12 + ...	
PH14A T8L	Entered from PH6 if trap has occurred		
	D0-D31 ⟶ S0-S31	SXD = FASTA PH14 + ...	Word and space count —/→ A-register to check for zero
	S0-S31 —/→ A0-A31	AXS = FASTA PH14 + ...	
	Enable memory request	MRQ/1 = FASTA PH14 + ...	To read next instruction
	Go to PH12A	BRPH12 = FASTA PH14 + ...	

Mnemonic: PLW (08, 88)

The preparation sequence for the PSM instruction is the same as the general preparation sequence described in the paragraphs entitled Preparation Sequence. The execution sequence for the Push Multiple instruction is described in figure 3-189.

In PH1 the contents of condition code bits CC1-CC4 are transferred to E4-E7 and to D28-D31. A one is set into D27 if the value of CC1-CC4 is zero. Flip-flop NGX is set, and the CS-register is set to ones to prepare for the negation of D in PH2.

The D-register is two's complemented in PH2, and the result is transferred to B16-B31. B16-B31 now contains the word count increment value, and B0-B15 is cleared to zeros.

D16-D31 is upward aligned into A0-A15 in PH3. A0-A16 contains the space count decrement value. In PH4 the contents of A and B are merged (OR-gated), and the result, which is both the space count decrement value and the word count increment value, is transferred to A. The count word is read into D in PH5, and in PH6 the count word and the increment/decrement values are added. The modified count word is then transferred into the B-register and is held until PH11A. During the addition in PH6, carry bit K15 is inhibited to prevent any carry from the word count from affecting the space count addition.

If the space count underflows, SW1 is set; if the word count overflows, SW2 is set. Space count underflow and word count overflow can occur simultaneously only if the stack contains 32,768 words. Space count underflow and word count overflow are detected by comparing bit 0 of the original space count in D with the result in sum bus bit 0, and by comparing bit 16 of the original word count in D with the result in sum bus bit 16.

If bit 0 of the count word TS (trap on space underflow) is a zero, a trap occurs because of a space count underflow. If bit 16 of the count word TW (trap on word overflow) is a zero, a trap occurs because of a word count overflow. If a trap occurs because of a space count underflow, flip-flop SW1 is not set, since D0 must be a one to set SW1 or a zero to cause a trap. If a trap occurs because of a word count overflow, flip-flop SW2 is not set, since D16 must be a zero to cause a trap or a one to set SW2.

It bit 0 of the count word TS (trap on space underflow) is a one and if the space count underflowed, the CPU aborts the PSM instruction by branching to PH14A. If bit 16 of the count word TW (trap on word overflow) is a one and if the word count overflowed, the CPU aborts the PSM instruction by branching to PH14A. If no underflow/overflow occurs, the instruction sequences from PH6 to PH7. In PH7 a memory request is made to read TSA.

The contents of TSA are read into the D-register during PH8. At PH9 clock one is set into CS1, and in PH1A this one is added to TSA and is transferred to the P-register. The address in the P-register is the address of the first word to be pushed into the stack. A memory request is enabled in PH1A to write the first stack word in PH2A.

The CPU remains in PH2A until the E-register has counted down to one. The R- and P-registers are counted up by one at each iteration of PH2A. When the contents of the E-register have counted down to one, the P-register contains the address of the last word pushed into the stack. This address becomes the new top-of-stack address. The contents of P are transferred to C in PH8A and to D in PH9A. The new modified TSA is written into memory in PH10A. The modified count word, which has been saved in the B-register, is written into memory in PH11A. The modified count word is also transferred to the A-register in PH11A and is tested in PH12A.

In PH12A the condition bits are set. Flip-flop CC1 is set if the space count underflowed in PH6. Flip-flop CC3 is set if the word count underflowed in PH6; flip-flop CC2 is set if the modified space count is currently equal to zero; and flip-flop CC4 is set if the modified word count is zero.

A sequence chart of the Push Multiple instruction is given in table 3-108.

3-236 Pull Multiple (PLM 0A, 8A)

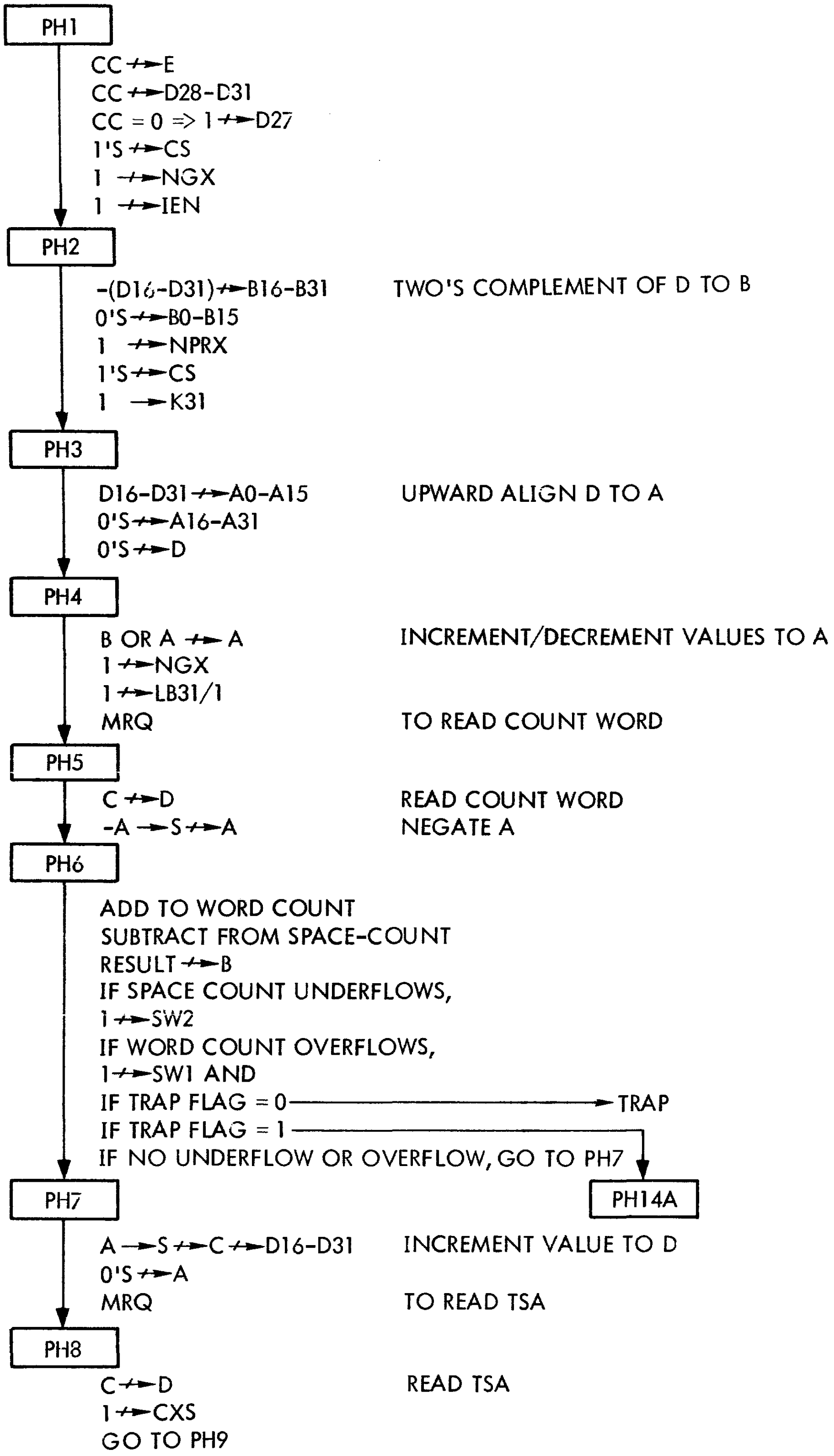
Before the PLM instruction is executed, the condition code must contain a value equal to the number of words to be pulled from the pushdown stack. Since the condition code consists of four flip-flops, CC1-CC4, not more than 16 words can be pulled from a stack during the execution of any one PLM instruction regardless of the stack length. If the condition code is equal to zero, 16 words are pulled from the stack.

During the execution of the PLM instruction, the value of the condition code less one is subtracted from the top of stack address so that the stack words are loaded into private memory in ascending order. For example, if CC is equal to 3, and the top of stack address is 1517, the stack words will be loaded into private memory in the following order:

(1515) to R

(1516) to R+1

(1517) to R+2



901060A.3695

Figure 3-189. Push Multiple, Logic Sequence Diagram (Sheet 1 of 2)

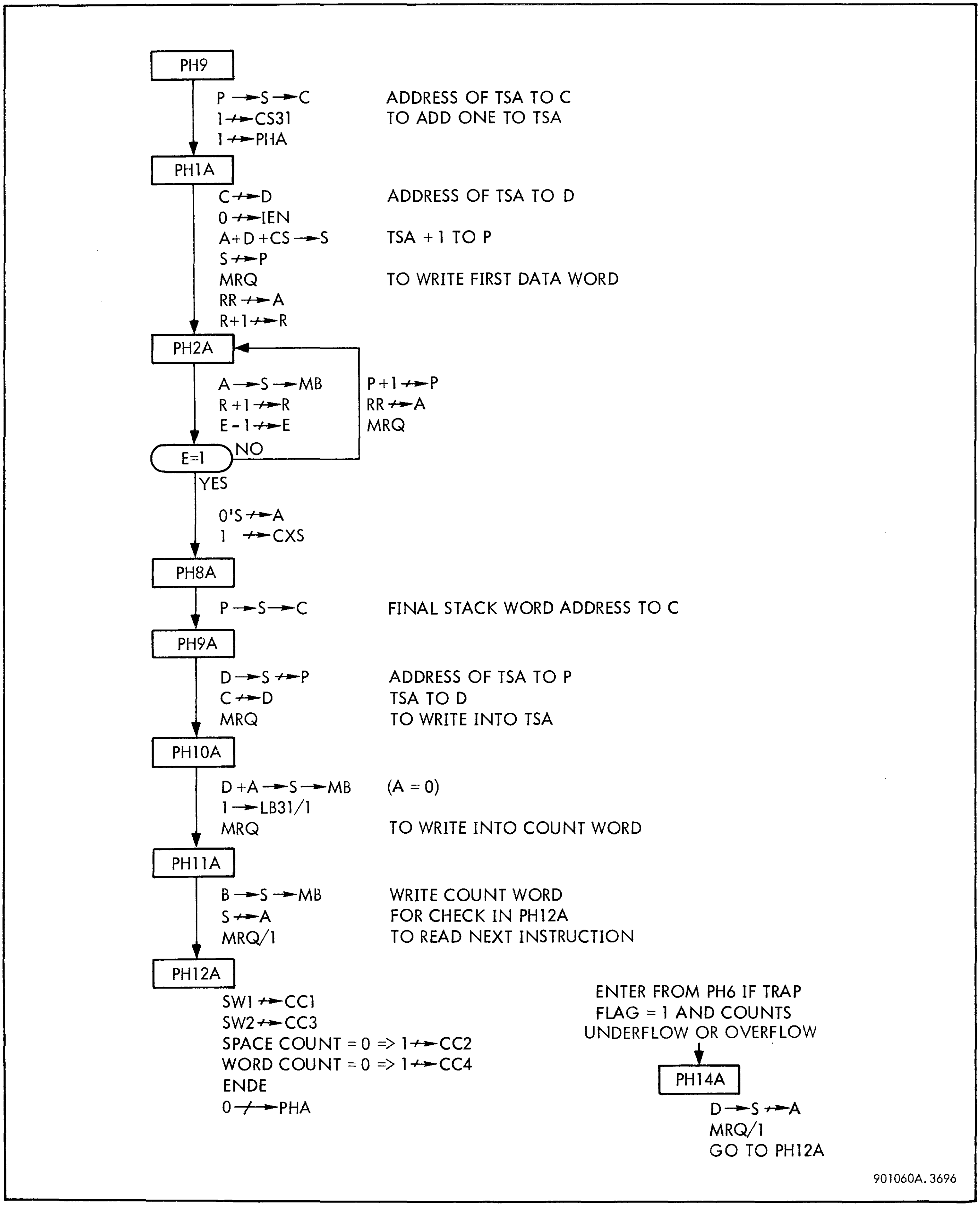


Figure 3-189. Push Multiple, Logic Sequence Diagram (Sheet 2 of 2)

Table 3-108. Push Multiple, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	1 —/→ DRQ	S/DRQ = FAST EXU + FASTA EXU + ...	DRQ is set for all execution phases
PH1 T4RL	CC1-CC4 —/→ E4-E7	EXCC = FAST PH1 (O4 NO5 O6) + ...	Number of words —/→ E-register and D-register
	CC1-CC4 —/→ D28-D31	DXCC = FAST PH1 (O4 NO5 O6) + ...	
	1 —/→ IEN	S/IEN = FAST PH1 (O4 NO5 O6) + ...	Allow interrupts
	1 —/→ NGX	(S/NGX) = FAST PH1 NO2 + ...	Prepare for negation
	1's —/→ CS0-CS31	CSX1 = (S/NGX) + ...	
PH2 T6L	1 ——→ G1619/1	G1619/1 = FAST PH2 + ...	Causes zeros in B0-B15
	1 ——→ K15	K15 = G1619/1	
	-(D16-D31) ——→ S16-S31	SXADD/1 = NGX NFAMDSF + ...	Negated word count —/→ B16-B31
	0's ——→ S0-S15		
	S0-S31 —/→ B0-B31	BXS = FAST PH2 + ...	(Word count increment value)
	1 —/→ NPRX	(S/NPRX) = FAST PH2 + ...	
	1's —/→ CS0-CS31 1 ——→ K31	CSX1 = (S/NPRX) + ...	Prepare for upward alignment
PH3 T6L	D16-D31 ——→ S0-S15	SXUAH = FAST PH3 + ...	Upward align negated count —/→ A0-A15
	S0-S15 —/→ A0-A15	AXS/2 = FAST PH3 + ...	
	0's —/→ D0-D31	DX/1 = FAST PH3 + ...	Clear D-register
PH4 T6L	B0-B31 ——→ S0-S31	SXB = FAST PH4 + ...	Effectively OR-gates A and B into A. A contains increment and decrement values
	A0-A31 ——→ S0-S31	SXA = FAST PH4 + ...	
	Set flip-flop NGX	(S/NGX) = FAST PH4	Prepare to negate A-register contents
	S0-S31 —/→ A0-A31	AXS = FAST PH4 + ...	

Mnemonic: PSM (0B, 8B)

(Continued)

Table 3-108. Push Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	Enable memory request	MRQ = FAST PH4 + ...	To read count word
	1 → LB31/1	(S/LB31/1) = FAST PH4 + ...	Address low order half of stack pointer doubleword
PH5 T6L	MB0-MB31 → C0-C31	CXMB = DGC	
	C0-C31 ⇸ D0-D31	DXC/6 = FAST PH5 + ...	Read count word from effective address plus one
	-(A0-A31) → S0-S31		Negate A-register
	S0-S31 ⇸ A0-A31	AXS = FAST PH5 O7 + ...	
	1 → G1619/1	G1619/1 = FAST PH5 + ...	
	1 → K15	K15 = G1619 + ...	
		S/T10L = FAST PH5 + ...	
PH6 T10L	A0-A31 + D0-D31 → S0-S31	SXADD = FAST PH6 + ...	Subtract decrement value from space count; add increment value to word count
	Force 0 → K15	K15 = N(FAST PH6) (...) + ...	
	S0-S31 ⇸ B0-B31	BXS = FAST PH6 + ...	Updated count word to B-register
	D16 NS16 => 1 ⇸ SW2	S/SW2 = FAST PH6 D16 NS16 + ...	Word count overflow
	D0-NS0 => 1 ⇸ SW1	S/SW1 = FAST PH6 D0 NS0 + ...	Space count underflow
	Trap, or	(S/TR30/1) = FAST PH6 ND16 S16 + FAST PH6 ND0 S0 + ...	If trap flag = 0, trap with condition code unchanged
		S/TRAP = (S/TR30/1) + ...	
	Go to PH14A	BRPH14 = FAST PH6 D16 NS16 + FAST PH6 D0 NS0 + ...	If trap flag = 1, go to PH14A to abort instruction
	1 ⇸ PHA	S/PHA = FAST PH6 D0 NS0 + FAST PH6 D16 NS16 + ...	Bit 16 or bit 0 has changed because of an underflow or overflow
	1 ⇸ CXS	S/CXS = FAST PH6 + ...	
		S/T8L = (S/CXS) + ...	Preset for PH7

Mnemonic: PSM (0B, 8B)

(Continued)

Table 3-108. Push Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T8L	A0-A31 → S0-S31	SXA = FAST PH7 + ...	Decrement and increment values -/→ D-register
	S0-S31 → C0-C31		
	C16-C31 -/→ D16-D31	DXS/2 = FAST PH7 + ...	
	0's -/→ A0-A31	AX/1 = FAST PH7 + ...	Clear A-register
	Enable memory request	MRQ = FAST PH7 + ...	To read top-of-stack address
PH8 T6L	MB0-MB31 → C0-C31	CXMB = DGC	
	C0-C31 -/→ D0-D31	DXC/6 = FAST PH8 + ...	Read top-of-stack address from effective memory location
	1 -/→ CXS	S/CXS = FAST PH8 NOL3 + ...	Preset for PH9
		S/T8L = (S/CXS) + ...	
PH9 T8L	P16-P31 → S16-S31	SXP = FAST PH9 + ...	Effective address → C-register
	S16-S31 → C16-C31		CXS set true in PH8
	1 -/→ CS31	CSX1/8 = FAST PH9 NOL8 NOL3 + ...	To add one to top-of-stack address
	1 -/→ PHA	S/PHA = FAST PH9 + ...	
	Go to PH1A	BRPH1 = FAST PH9 NOL3 + ...	
PH1A T6L	C0-C31 -/→ D0-D31	DXC/6 = FASTA PH1 + ...	Effective address → D-register
		FASTA = FAST PHA + ...	
	A0-A31 + D0-D31 + CS0-CS31 → S0-S31	SXADD = FAST PH1 + ...	Top-of-stack address plus 1 -/→ P-register
	S16-S31 -/→ P16-P31	PXS = FASTA PH1 NO2 + ...	
	RR0-RR31 -/→ A0-A31	AXRR = FASTA PH1 (O4 NO5 O7) + ...	First data word from private memory -/→ A-register
	R+1 -/→ R	PCTR = FASTA PH1 O7 + ...	Increment R-register for next data word
	0 -/→ IEN	S/T8L = FASTA PH1 + ...	

Mnemonic: PSM (0B, 8B)

(Continued)

Table 3-108. Push Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1A T6L (Cont.)		R/IEN = FASTA PH1 + ...	Disallow interrupts
	Enable memory request	MRQ = FASTA PH1 + ...	To write first data word

PH2A
T8L
	A0-A31 → S0-S31 → MB0-MB31	SXA = FASTA PH2
		MW = FASTA PH1 + ...
			Write first data word into core memory

	R+1 -/→ R	PCTR = FASTA PH2 + ...	Increment R-register
	E-1 -/→ E	MCTE = FASTA PH2 + ...	Decrement word count in E-register
	E ≠ 1 => Enable memory request, and	MRQ = FASTA PH2 N(E=1) NO2 + ...	Request next data word
	P+1 -/→ P, and	PCTP1 = FASTA PH2 N(E=1) NO2 + ...	Increment P-register
	RR0-RR31 -/→ A0-A31, and	AXRR = FASTA PH2 N(E=1) NOL9 + ...	Data word from private memory -/→ A-register
	Repeat PH2A	BRPH2 = FASTA PH2 N(E=1) NO2 + ...	If word count ≠ 1
		S/T8L = FASTA PH2 + ...	
	E = 1 => 0's -/→ A0-A31, and	AX/1 = FASTA PH2 (E=1) NO2 + ...	
	1 -/→ CXS, and	S/CXS = FASTA PH2 (E=1) NO2 + ...	
	Go to PH8	BRPH8 = FASTA PH2 (E=1) NO2 + ...	If word count = 1
PH8A T8L	P16-P31 → S16-S31	SXP = FASTA PH8 + ...	Final top-of-stack address → C-register
	S0-S31 → C0-C31		CXS set true in PH2A
PH9A T6L	D0-D31 → S0-S31	SXD = FASTA PH9 + ...	Effective address -/→ P-register
	S16-S31 -/→ P16-P31	PXS = FASTA PH9 + ...	
	C0-C31 -/→ D0-D31	DXC/6 = FASTA PH9 + ...	Final top-of-stack address → D-register
	Enable memory request	MRQ = FASTA PH9 + ...	To write final top-of-stack address in core memory

Mnemonic: PSM (0B, 8B)

(Continued)

Table 3-108. Push Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10A T6L	D0-D31+A0-A31 → S0-S31	SXADD = FASTA PH10 + ...	Write final top-of-stack address in effective memory location. Address low order half of stack pointer double word
	S0-S31 → MB0-MB31	MW = FASTA PH10 + ...	To write count word
	1 ↛ LB31/1	(S/LB31/1) = FASTA PH10 + ...	
	Enable memory request	MRQ = FASTA PH10 + ...	
PH11A T6L	B0-B31 → S0-S31	SXB = FASTA PH11 + ...	Write count word in effective address plus one
	S0-S31 → MB0-MB31	MW = FASTA PH11 + ...	
	S0-S31 ↛ A0-A31	AXS = FASTA PH11 + ...	Count word to A-register for check in PH12A
	Enable memory request	MRQ/1 = FASTA PH11 + ...	To read next instruction
PH12A T6L	If SW1 = 1, 1 ↛ CC1	S/CC1 = FASTA PH12 SW1 + ...	Space count underflow, word count overflow
	If SW2 = 1, 1 ↛ CC3	S/CC3 = FASTA PH12 SW2 + ...	
	If (A1-A15 = 0), 1 ↛ CC2	S/CC2 = FASTA PH12 A0115Z + ...	Space count zero
	If (A17-A31 = 0), 1 ↛ CC4	S/CC4 = FASTA PH12 A1731Z + ...	Word count zero
		(R/CC) = FASTA PH12 + ...	
	End	ENDE = FASTA PH12 + ...	
PH14A T6L	Entered from PH6 if trap has occurred		
	D0-D31 → S0-S31	SXD = FASTA PH14 + ...	Word and space count into A-register to check for zero
	S0-S31 ↛ A0-A31	AXS = FASTA PH14 + ...	
	Enable memory request	MRQ/1 = FASTA PH14 + ...	
	Go to PH12A	BRPH12 = FASTA PH14 + ...	

Mnemonic: PSM (0B, 8B)

The value of the condition code is also transferred to the E-register, which counts the number of words to be pulled from the stack.

The preparation sequence for the PLM instruction is the same as the general preparation sequence given under the Preparation Sequence. The execution sequence for the Pull Multiple instruction is described in figure 3-190. In PH1 the contents of the condition code bits, CC1-CC4, are transferred to E4-E7 and to D28-D31. If the value of CC1-CC4 is zero, a one is set into D27. Flip-flop NGX is set and the CS-register is set to ones to prepare for the negation of D in PH2.

The D-register is two's complemented in PH2, and the result is transferred to B16-B31. B16-B31 now contain the word count decrement value, and B0-B15 are cleared to zeros.

In PH3, D16-D31 is upward aligned into A0-A15. A0-A16 contains the space count increment value. In PH4 the contents of A and B are merged (OR-gated), and the result, which is both the space count increment value and the word count decrement value, is transferred to A. The count word is read into D in PH5. In PH6 the count word and the increment/decrement values are added. The modified count word is then transferred into the B-register and is held there until PH11A. During the addition in PH5, carry bit K15 is inhibited to prevent any carry from the word count from affecting the space count addition.

SW1 is set if the space count overflows; SW2 is set if the word count underflows. Space count overflow and word count underflow can occur simultaneously only if the stack contains 32,768 words. Space count overflow and word count underflow are detected by comparing bit 0 of the original space count in D with the result in sum bus bit 0 and by comparing bit 16 of the original word count in D with the result in sum bus bit 16.

If bit 0 of the count word TS (trap on space overflow) is a zero, a trap occurs because of a space count overflow. If bit 16 of the count word TW (trap on word underflow) is a zero, a trap occurs because of a word count underflow. If a trap occurs because of a space count overflow, flip-flop SW1 is not set, since D0 must be a one to set SW1 or a zero to cause a trap. If a trap occurs because of a word count underflow, flip-flop SW2 is not set, since D16 must be a zero to cause a trap or a one to set SW2.

If bit 0 of the count word TS (trap on space overflow) is a one and if the space count overflowed, the CPU aborts the PLM instruction by branching to PH14A. If bit 16 of the count word TW (trap on word underflow) is a one and if the word count underflowed, the CPU aborts the PLM instruction by branching to PH14A. If no underflow/overflow occurs, the instruction sequences from PH6 to PH7. In PH7 a memory request is made to read TSA.

The contents of TSA are read into the D-register during PH8. In PH9, a one is placed in CS31 to add one to the top-of-stack address in PH1A. The value in A16 is loaded into CS0-CS15 to extend the sign of the word count decrement value. In PH1A the two's complement of the original contents of the condition codes, less one, is added to TSA. The result is clocked into the P-register. This address in P is the address of the first word to be pulled from the stack. A memory request is made in PH1A to read the first stack word.

The first stack word is read into the D-register in PH3A. If the value in E does not equal one, the P-register counts up by one, and the CPU requests the next word from the memory stack as it sequences to PH5A.

The CPU remains in PH5, sequentially reading the stack words until the count in E equals one. During each iteration of PH5A, the E-register counts down by one, and the R- and P-registers count up by one. When the E count reaches one, however, the P-register does not count, and the CPU sequences to PH6A. The top-of-stack address in the A-register is transferred to the P-register in PH6A. Signal MRQ is enabled to read the unmodified top-of-stack address from memory. The condition code is again transferred to D28 through D31.

The contents of the D-register are negated and are transferred to the A-register in PH7A. The top-of-stack address is read into the D-register and PH10A is entered. The decrement value in the D-register is added to the top-of-stack address in the A-register, and the modified top-of-stack address is written into the top-of-stack address location in core memory.

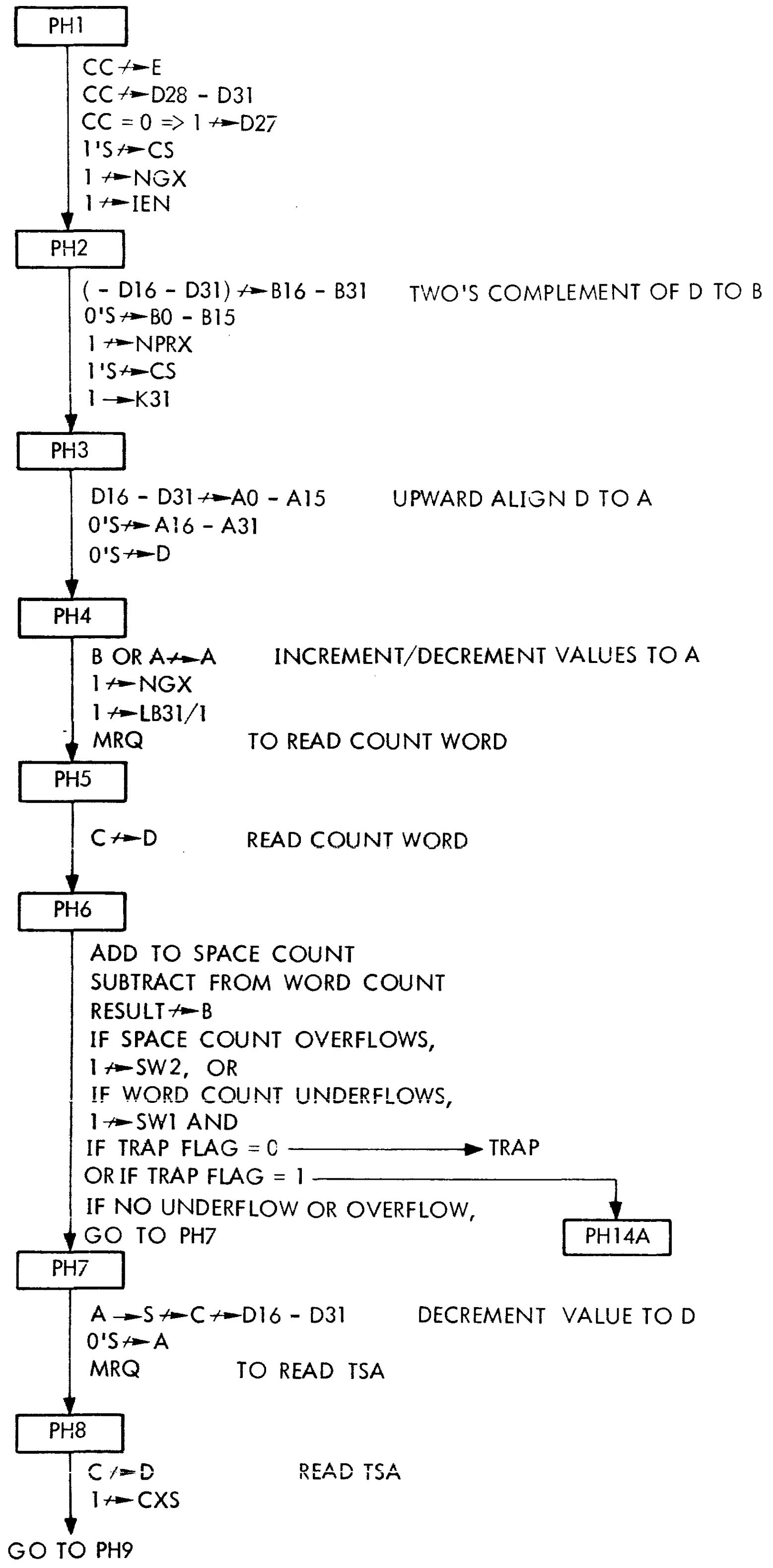
In PH11A the modified count word in the B-register is written into the count word location in memory. The count word is also transferred to the A-register for testing in PH12A.

In PH12A the condition code flip-flops are set. Flip-flop CC1 is set if the space count overflowed in PH6; flip-flop CC3 is set if the word count underflowed in PH6. Flip-flop CC2 is set if the modified space count is currently equal to zero; flip-flop CC4 is set if the modified word count is zero.

A sequence chart of the Pull Multiple instruction is given in table 3-109.

3-237 Modify Stack Pointer (MSP 13, 93)

The MSP instruction modifies the stack pointer doubleword by the contents of register R, bits R16-R31. If bit 16 of R is a zero, the modifier value is a positive number; if bit 16 of R is a one, the modifier value is a negative number. If the modifier value in R is a positive number, the stack pointer doubleword is affected in the following manner: the modifier value increases numerically the TSA and the



901060A.3692

Figure 3-190. Pull Multiple, Logic Sequence Diagram (Sheet 1 of 3)

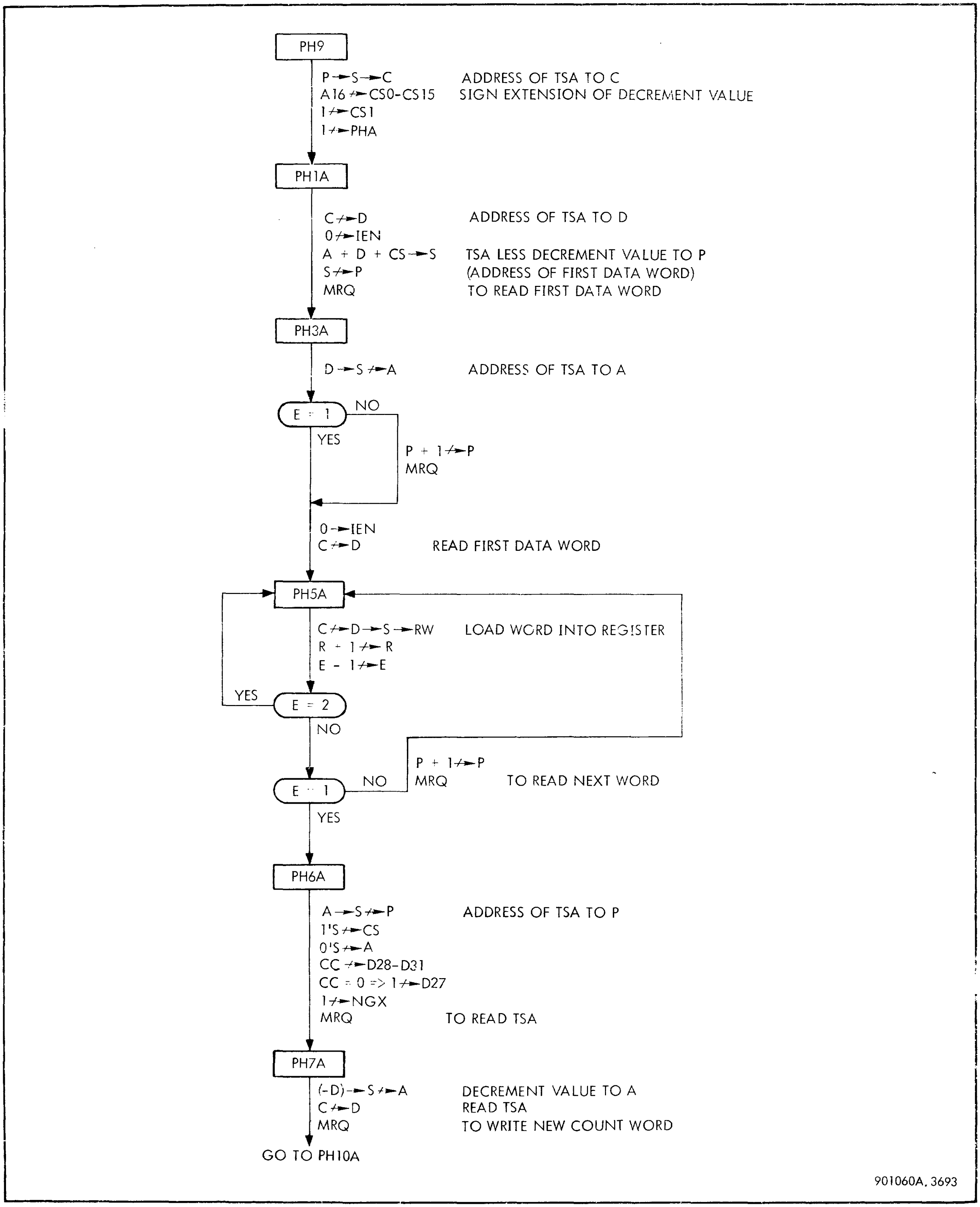


Figure 3-190. Pull Multiple, Logic Sequence Diagram (Sheet 2 of 3)

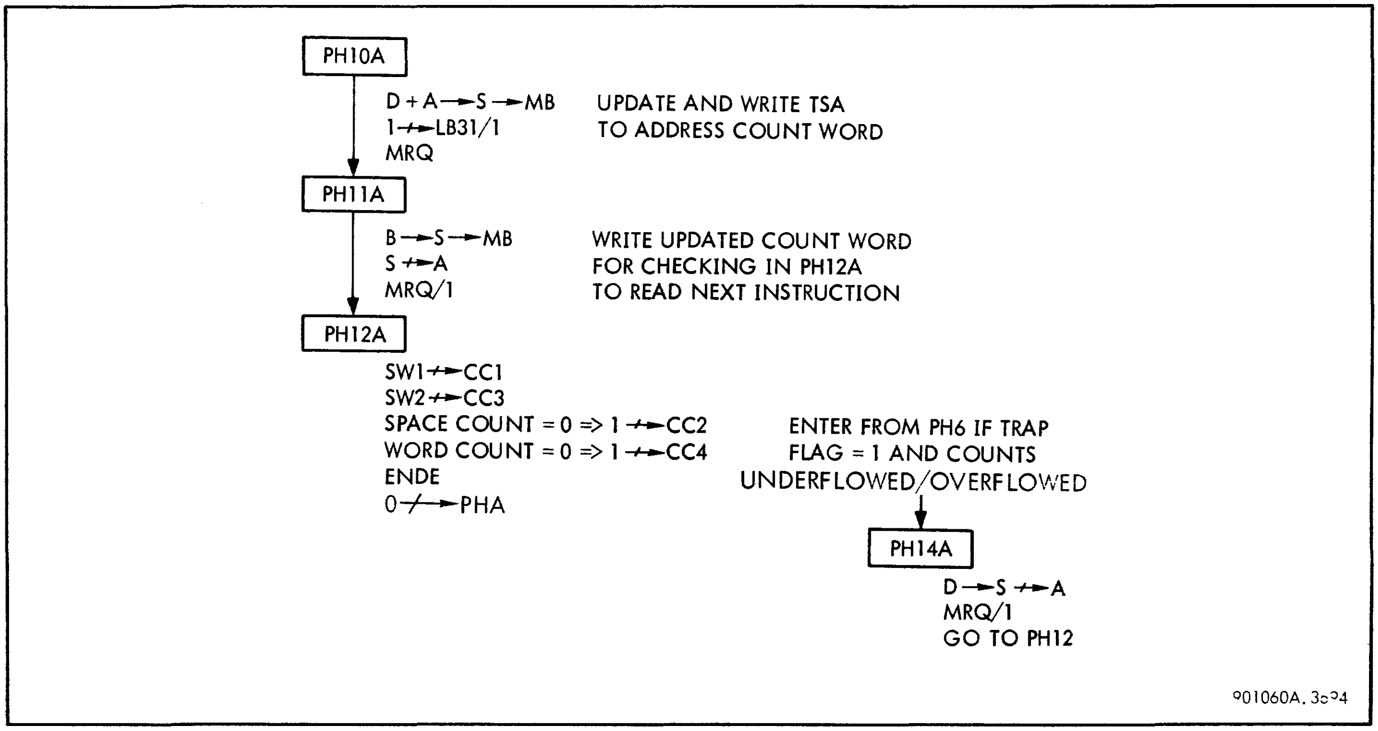


Figure 3-190. Pull Multiple, Logic Sequence Diagram (Sheet 3 of 3)

word count, but decreases the space count. If the modifier value in R is a negative number, the stack pointer doubleword is affected in the following manner: the modifier value decreases numerically the TSA and the word count, but increases the space count. No words are pushed into or pulled from the push-down stack during the execution of an MSP instruction.

The preparation sequence for the MSP instruction is the same as the general preparation sequence described under the Preparation Sequence. The execution sequence for the MSP instruction is described in figure 3-191.

In PH1 the least significant half of register R, R16-R31, is clocked into the A-register, A16-A31. In PH2 the A-register is two's complemented and is transferred to B. Carry K15 is forced true during the two's complement action, which effectively places zeros in B0-B15.

In PH3 the least significant half of A, A16-A31, is upward aligned into the most significant half of A, A0-A15, and A16-A31 is cleared to zeros. In PH4 the A- and B-registers are merged (OR-gated), and the result is placed in A. In PH5 the A-register is again two's complemented. The A-register now contains the increment and decrement values that are used to modify the stack pointer doubleword. If the R modifier value is a positive number, A0-A15 contain the decrement value, and A16-A31 contain the increment value. Conversely, if the R modifier value is a negative number, A0-A15 contain the increment value, and A16-A31 contain the decrement value.

In PH5 the count word is read into the D-register, and, in PH6, the modifier value in A is added to the count word in D. The modified count word is transferred to the B-register and is saved until PH11A. If either the space count or the word count overflows or underflows, the CPU either traps (if the corresponding trap flag, bit 0 or bit 16 of the count word, is a zero) or aborts the MSP instruction by branching to PH14A (if the corresponding trap flag, bit 0 or bit 16 of the count word, is a one).

During the addition in PH6 carry bit K15 is inhibited to prevent any carry from the word count from affecting the space count addition. SW1 is set if the space count either overflows or underflows. SW2 is set if the word count either overflows or underflows. Space count underflow or overflow is detected by comparing the original contents of bit 0 of the original count word in D with the modified contents of bit 0 of the sum bus. If they are different, an underflow or overflow has occurred. Word count underflow or overflow is detected by comparing the contents of bit 16 of the original word count in D with the modified contents of bit 16 of the sum bus. If they are different, an underflow or an overflow has occurred.

If bit 0 of the count word TS (trap on space underflow or overflow) is a zero, a trap occurs because of a space count underflow or overflow. If bit 16 of the count word TW (trap on word underflow or overflow) is a zero, a trap occurs because of a word count underflow or overflow.

Table 3-109. Pull Multiple, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	1 —/→ DRQ	S/DRQ = FAST EXU + FASTA EXU + ...	DRQ is set for all execution phases
PH1 T4RL	CC1-CC4 —/→ E4-E7	EXCC = FAST PH1 (O4 NO5 O6) + ...	Number of words —/→ E-register and D-register
	CC1-CC4 —/→ D28-D31	DXCC = FAST PH1 (O4 NO5 O6) + ...	
	1 —/→ IEN	S/IEN = FAST PH1 (O4 NO5 O6) + ...	Allow interrupts
	1 —/→ NGX	(S/NGX) = FAST PH1 NO2 + ...	Prepare for negation
	1's —/→ CS0-CS31	CSX1 = (S/NGX) + ...	
PH2 T6L	1 ⟶ G1619/1	G1619/1 = FAST PH2 + ...	Causes zeros in B0-B15
	1 ⟶ K15	K15 = G1619/1	
	-(D16-D31) ⟶ S16-S31	SXADD/1 = NGX NFAMDSF + ...	Negated word count ⟶ B16-B31 (word count decrement value)
	0's ⟶ S0-S15		
	S0-S31 —/→ B0-B31	BXS = FAST PH2 + ...	Prepare for upward alignment
	1 —/→ NPRX	(S/NPRX) = FAST PH2 + ...	
	1's —/→ CS0-CS31	CSX1 = (S/NPRX) + ...	
PH3 T6L	D16-D31 ⟶ S0-S15	SXUAH = FAST PH3 + ...	Upward align word count —/→ A0-A15 (increment value)
	S0-S15 —/→ A0-A15	AXS/2 = FAST PH3 + ...	
	0's —/→ D0-D31	DX/1 = FAST PH3 + ...	Clear D-register
PH4 T6L	B0-B31 ⟶ S0-S31	SXB = FAST PH4 + ...	Effectively OR-gates A and B to A
	A0-A31 ⟶ S0-S31	SXA = FAST PH4 + ...	A now contains increment and decrement values
	S0-S31 —/→ A0-A31	AXS = FAST PH4 + ...	

Mnemonic: PLM (0A, 8A)

(Continued)

Table 3-109. Pull Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	Enable memory request	NRQ = FAST PH4 + ...	To read count word
	1 —/→ LB31/1	(S/LB31/1) = FAST PH4 + ...	Address low order half of space count doubleword from core memory
PH5 T6L	MB0-MB31 → C0-C31	CXMB = DGC	
	C0-C31 —/→ D0-D31	DXC/6 = FAST PH5 + ...	Read count word
		(S/T10L/1) = FAST PH5 + ...	
PH6 T10L	A0-A31+D0-D31 → S0-S31	SXADD = FAST PH6 + ...	Add increment value to space count; subtract decrement value from word count
	Force 0 → K15	K15 = N(FAST PH6) (...) + ...	
	S0-S31 —/→ B0-B31	BXS = FAST PH6 + ...	Updated count word to B-register
	D16 NS16 => 1 —/→ SW2	S/SW2 = FAST PH6 D16 NS16 + ...	Word count underflow
	D0-NS0 => 1 —/→ SW1	S/SW1 = FAST PH6 D0 NS0 + ...	Space count overflow
	Trap, or	(S/TR30/1) = FAST PH6 ND16 S16 + FAST PH6 ND0 S0 + ...	If trap flag = 0, trap with condition code unchanged
		S/TRAP = (S/TR30/1) + ...	
	Go to PH14A	BRPH14 = FAST PH6 D16 NS16 + FAST PH6 D0 NS0 + ...	If trap flag = 1, go to phase 14A to abort instruction
	1 —/→ PHA	S/PHA = FAST PH6 D0 NS0 + FAST PH6 D16 NS16 + ...	Bit 16 or bit 0 has changed because of an underflow or overflow
	1 —/→ CXS	S/CXS = FAST PH6 + ...	Preset for PH7
		S/T8L = (S/CXS) + ...	
PH7 T8L	A0-A31 → S0-S31	SXA = FAST PH7 + ...	Increment and decrement values —/→ D-register
	S0-S31 → C0-C31		
	C16-C31 —/→ D16-D31	DXC/2 = FAST PH7 + ...	Sign in D16

Mnemonic: PLM (0A, 8A)

(Continued)

Table 3-109. Pull Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T8L (Cont.)	0's ↛ A0-A31	AX/1 = FAST PH7 + ...	
	Enable memory request	MRQ = FAST PH7 + ...	To read top-of-stack address
PH8 T6L	MB0-MB31 → C0-C31	CXMB = DGC	
	C0-C31 ↛ D0-D31	DXC/6 = FAST PH8 + ...	Read top-of-stack address into D-register via C-register
	D0-D31 → S0-S31	SXD = FAST PH8 + ...	Word count decrement value ↛ A-register
	S0-S31 ↛ A0-A31	AXS = FAST PH8 (OLA + OL3) + ...	
	1 ↛ CXS	S/CXS = FAST PH8 NOL3 + ...	Preset for PH9
		S/T8L = (S/CXS) + ...	
PH9 T8L	P16-P31 → S16-S31	SXP = FAST PH9 + ...	Effective address → C-register
	S16-S31 → C16-C31		CXS set true in PH8
	1 → CS31	CSX1/8 = FAST PH9 NOL8 NOL3 + ...	For two's complement
	A16 ↛ CS0-CS15	CSX1/3 = FAST PH9 A16 + ...	For sign extension in word count decrement value
	1 ↛ PHA	S/PHA = FAST PH9 + ...	
	Go to PH1A	BRPH1 = FAST PH9 NOL3 + ...	
PH1A T6L	C0-C31 ↛ D0-D31	DXC/6 = FASTA PH1 + ...	Effective address ↛ D-register
		FASTA = FAST PHA + ...	
	A0-A31+D0-D31+CS0-CS31 → S0-S31	SXADD = FASTA PH1 + ...	Top-of-stack address minus decrement value ↛ P-register. P-register now contains address of first word to be pulled
	S16-S31 ↛ P16-P31	PXS = FASTA PH1 NO2 + ...	
	Enable memory request	MRQ = FASTA PH1 + ...	To read first data word from core memory
	Set flip-flop T8L	S/T8L = FASTA PH1 + ...	

Mnemonic: PLM (0A, 8A)

(Continued)

Table 3-109. Pull Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1A T6L (Cont.)	0 —/→ IEN	R/IEN = FASTA PH1 + ...	Disable interrupt
	Go to PH3A	BRPH3 = FASTA PH1 (O4 NO5 NO7) + ...	
PH3A T8L	MB0-MB31 ⟶ C0-C31	CXMB = DGC	
	C0-C31 —/→ D0-D31	DXC/6 = FASTA PH3 + ...	Read data word. Place in D-register
	D0-D31 ⟶ S0-S31	SXD = FASTA PH3 + ...	
	S0-S31 —/→ A0-A31	AXS = FASTA PH3 + ...	Effective address —/→ A-register
	P+1 —/→ P	PCTP1 = FASTA PH3 N(E=1) OLA + ...	Increment P-register
	Enable memory request	MRQ = FASTA PH3 N(E=1) OLA + ...	Request for next data word
		S/T8L = FASTA PH3 + ...	
	Go to PH5A	BRPH5 = FASTA PH3 OLA + ...	
PH5A T8L	MB0-MB31 ⟶ C0-C31	CXMB = DGC	
	C0-C31 —/→ D0-D31	DXC/6 = FASTA PH5 + ...	Read next data word from memory
	D0-D31 ⟶ S0-S31	SXD = FASTA PH5 + ...	Load previous data word ⟶ private memory register R
	S0-S31 ⟶ RW0-RW31	RW = FASTA PH5 + ...	
	R+1 —/→ R	PCTR = FASTA PH5 + ...	Increment R-register
	E-1 —/→ E	MCTE1 = FASTA PH5 + ...	Decrement word count
	E=1 or 2 => P+1 —/→ P	PCTP1 = FASTA PH5 N(E=1 or 2) OLA + ...	Increment P-register
	If E≠1, remain in PH5A	BRPH5 = FASTA PH5 N(E=1) OLA + ...	
	Enable memory request	MRQ = FASTA PH5 N(E=1 or 2) OLA + ...	To read next data word
	If E=1, go to PH6A	S/PH6 = PH5 NBR + ...	
		S/T8L = FASTA PH5 + ...	

Mnemonic: PLM (0A, 8A)

(Continued)

Table 3-109. Pull Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6A T8L	CC1-CC4 -/→ D28-D31	DXCC = FASTA PH6 OLA NO2 + ...	Word count back into D-register
	CC = 0 => 1 -/→ D27	D27X1 = FASTA PH6 OLA CCZ + ...	Zero word count => Pull 16 words
	1 -/→ NGX	S/NGX = FASTA PH6 OLA NO2 + ...	Prepare for negation
	A0-A31 ⟶ S0-S31	SXA = FASTA PH6 + ...	Effective address to P-register
	S16-S31 -/→ P16-P31	PXS = FASTA PH6 + ...	
	1's -/→ CS0-CS31	CSX1 = FASTA PH6 + ...	Prepare for negation
	0's -/→ A0-A31	AX/1 = FASTA PH6 + ...	
	Enable memory request	MRQ = FASTA PH6 OLA + ...	To read top-of-stack address
		S/T8L = FASTA PH6 + ...	
PH7A T8L	-(D0-D31) ⟶ S0-S31	SXADD = FASTA PH7 + ...	Negate word count in D-register. Transfer to A-register
	S0-S31 -/→ A0-A31	AXS = FASTA PH7 OLA + ...	
	MB0-MB31 ⟶ C0-C31	CXMB = DGC	
	C0-C31 -/→ D0-D31	DXC/6 = FASTA PH7 + ...	Read top-of-stack address
	Enable memory request	MRQ = FASTA PH7 + ...	To write new top-of-stack address
	Go to PH10A	BRPH10 = FASTA PH7 + ...	
PH10A T6L	D0-D31 + A0-A31 ⟶ S0-S31	SXADD = FASTA PH10 + ...	Update and write top-of-stack address into core memory
	S0-S31 ⟶ MB0-MB31	MW = FASTA PH10 + ...	
	1 -/→ LB31/1	(S/LB31/1) = FASTA PH10 + ...	Address low order half of stack pointer doubleword
	Set memory request	MRQ = FASTA PH10 + ...	To write new count word
PH11A T6L	B0-B31 ⟶ S0-S31	SXB = FASTA PH11 + ...	Write updated count word in effective location plus one

Mnemonic: PLM (0A, 8A)

(Continued)

Table 3-109. Pull Multiple, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH11A T6L (Cont.)	S0-S31 → MB0-MB31	MW = FASTA PH11 + ...	
	S0-S31 ⇸ A0-A31	AXS = FASTA PH11 + ...	Updated count word to A-register for check in PH12A
	Enable memory request	MRQ/1 = FASTA PH11 + ...	To read next instruction
PH12A T6L	SW1 = 1 => 1 ⇸ CC1	S/CC1 = FASTA PH12 SW1 + ...	Space count overflow
	SW2 = 1 => 1 ⇸ CC3	S/CC3 = FASTA PH12 SW2 + ...	Word count underflow
	If (A1-A15=0), 1 ⇸ CC2	S/CC2 = FASTA PH12 A0115Z + ...	Space count = 0
	If (A17-A31=0), 1 ⇸ CC4	S/CC4 = FASTA PH12 A1731Z + ...	Word count = 0
		(R/CC) = FASTA PH12 + ...	
	End	ENDE = FASTA PH12 + ...	
PH14A T8L	Entered from PH6 if trap has occurred		
	D0-D31 → S0-S31	SXD = FASTA PH14 + ...	Word and space count → A-register to check for zero
	S0-S31 ⇸ A0-A31	AXS = FASTA PH14 + ...	
	Enable memory request	MRQ/1 = FASTA PH14 + ...	To read next instruction
	Go to PH12A	BRPH12 = FASTA PH14 + ...	

Mnemonic: PLM (0A, 8A)

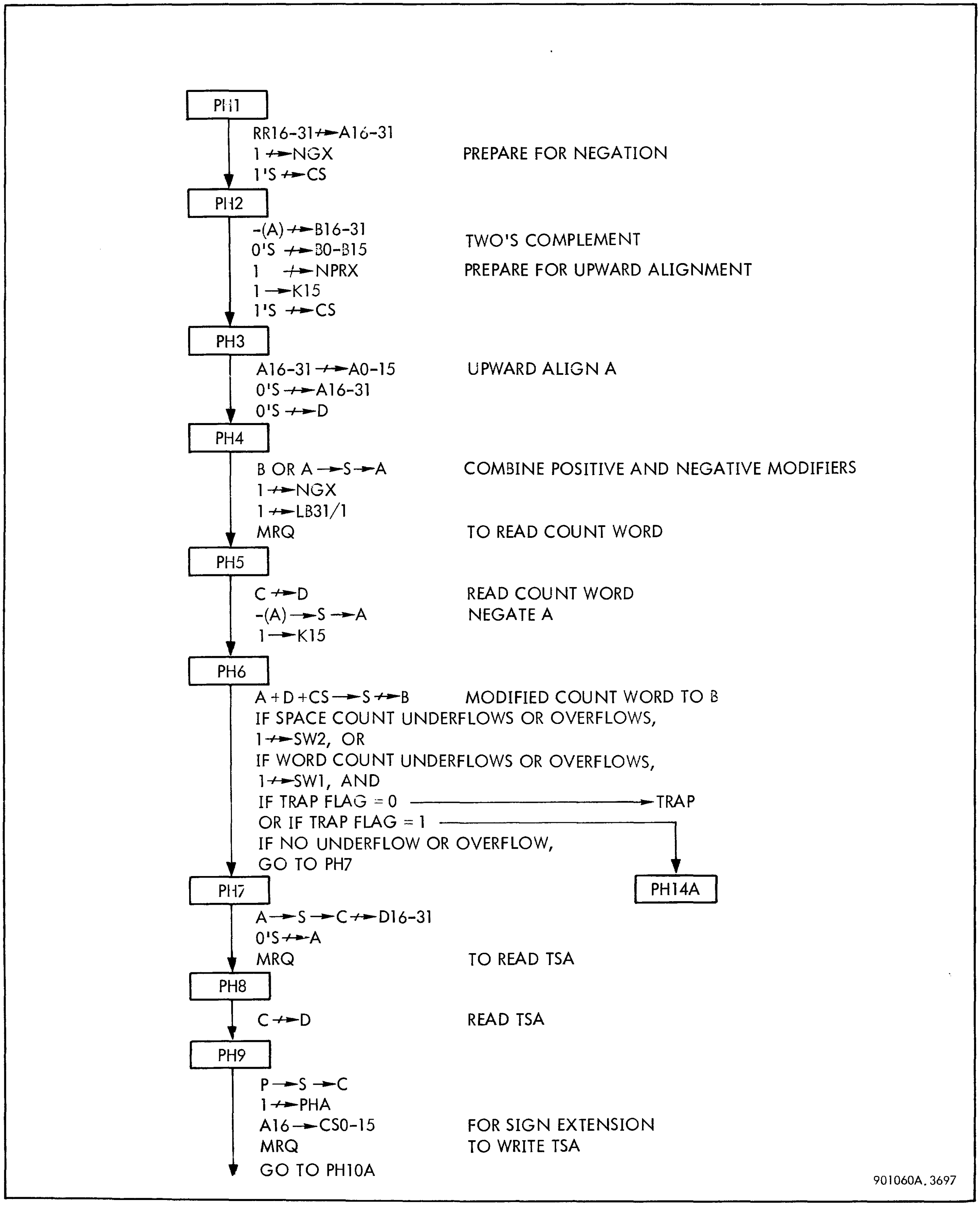


Figure 3-191. Modify Stack Pointer, Logic Sequence Diagram (Sheet 1 of 2)

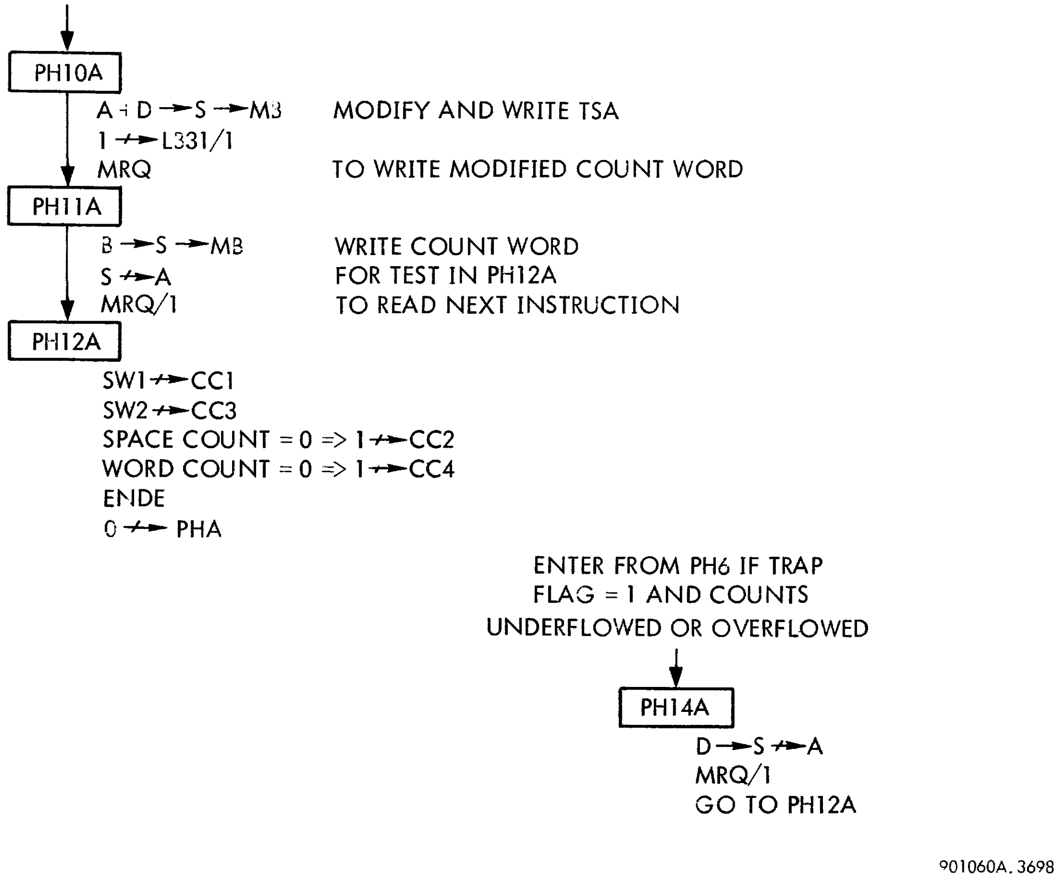


Figure 3-191. Modify Stack Pointer, Logic Sequence Diagram (Sheet 2 of 2)

If a trap occurs because of a space count underflow or overflow, flip-flop SW1 cannot be set, since D0 must be a one to set SW1 or a zero to cause a trap. If a trap occurs because of a word count underflow or overflow, flip-flop SW2 will not be set because D16 must be a zero to cause a trap or a one to set SW2. If bit 0 of the count word TS (trap on space underflow or overflow) is a one and if the space count underflowed or overflowed, the CPU aborts the MSP instruction by branching to PH14A. If bit 16 of the count word TW (trap on word underflow or overflow) is a one and if the word count underflowed or overflowed, the CPU aborts the MSP instruction by branching to PH14A. If no space count or word count underflow or overflow occurs, the instruction sequences from PH6 to PH7. In PH7 a memory request is made to read TSA.

The contents of TSA are read into the D-register during PH8. In PH10A the contents of D (which contains TSA) and the contents of A (which contains the positive or negative modifier value) are added. The result, which is the modified TSA, is written into the top-of-stack address in memory.

During PH11A the contents of B (which contains the modified count word) are written into memory. The modified count word is also transferred into the A-register in PH11A for testing in PH12A.

The condition code bits are set in PH12A. CC1 is set if the space count underflowed or overflowed in PH6; CC3 is set if the word count underflowed or overflowed in PH6; CC2 is set if the modified space count is currently equal to zero; and CC4 is set if the modified word count is currently equal to zero.

A sequence chart of the Modify Stack Pointer instruction is given in table 3-110.

3-238 Execute (EXU 67, F7)

The reference address field of the EXU instruction points to the address of an operand. This operand is taken to be the next instruction to be executed by the CPU following the EXU instruction. If the subject instruction is not a branch (or another EXU), the next instruction to be executed is the instruction that normally would follow the initial EXU instruction. If the subject instruction is located in a memory area under memory protection, the memory protection feature is disabled for that instruction access only.

Table 3-110. Modify Stack Pointer, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	1 ⇸ DRQ	S/DRQ = FAST EXU + FASTA EXU + ...	DRQ is set for all execution phases
PH1 T4RL	RR16-RR31 ⇸ A16-A31	AXRR/2 = FAST PH1 OL3 + ...	Modify value to A
	1 ⇸ NGX	(S/NGX) = FAST PH1 NO2 + ...	
	1's ⇸ CS0-CS31	CSX1/20 = (S/NGX) + ...	
PH2 T6L	-(A0-A31) ⇸ B16-B31	BXS = FAST PH2 + ...	Zeros to B0-B15
		G1619/1 = FAST PH2 + ...	
		K15 = G1619	
	1 ⇸ NPRX	S/NPRX = FAST PH2 + ...	
	1's ⇸ CS0-CS31	CSX1/20 = (S/NPRX) + ...	
PH3 T6L	Upward align halfword, A → S	SXUAH = FAST PH3 + ...	Results in A16-A31 ⇸ A0-A15, and 0's ⇸ A16-A31
	S0-S31 ⇸ A0-A31	AXS/2 = FAST PH3 + ...	
	0's ⇸ D0-D31	DX/1 = FAST PH3 + ...	
PH4 T6L	B0-B31 or A0-A31 → S0-S31 ⇸ A0-A31	SXB = FAST PH4 + ...	Combine positive and negative halves of modifier in A
		SXA = FAST PH4 + ...	
	1 ⇸ NGX	(S/NGX) = FAST PH4 + ...	
	1 ⇸ LB31/1	(S/LB31/1) = FAST PH4 + ...	
	Enable memory request	MRQ = FAST PH4 + ...	To read count word
PH5 T6L	C0-C31 ⇸ D0-D31	DXC/6 = FAST PH5 + ...	Read count word
	-(A0-A31) → S0-S31 ⇸ A0-A31	AXS = FAST PH5 O7 + ...	Negate A-register
	1 → G1619/1	G1619/1 = FAST PH5 + ...	
		K15 = G1619	
		(S/T10L) = FAST PH5 + ...	

Mnemonic: MSP (13, 93)

(Continued)

Table 3-110. Modify Stack Pointer, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6 T10L	A0-A31+D0-D31 → S0-S31	SXADD = FAST PH6 + ...	Add/subtract to space count; subtract/add to word count
	Force 0 → K15	K15 = N(FAST PH6) (...)	
	S0-S31 ↛ B0-B31	BXS = FAST PH6 + ...	Modified count word to B-register
	D16 NS16 => 1 ↛ SW2, or	S/SW2 = FAST PH6 D16 NS16 + ...	Word underflow or overflow
	D0 NS0 => 1 ↛ SW1	S/SW1 = FAST PH6 D0 NS0 + ...	Space underflow or overflow
	Trap, or	(S/TR30) = FAST PH6 ND16 S16 + FAST PH6 ND0 S0 + ...	If trap flag = 0
	Go to PH14A	BRPH14 = FAST PH6 D16 NS16 + FAST PH6 D0 NS0 + ...	If trap flag = 1
	1 ↛ PHA	S/PHA = FAST PH6 D0 NS0 + FAST PH6 D16 NS16 + ...	Bit 16 or bit 0 has changed because of an underflow or overflow
	1 ↛ CXS	S/CXS = FAST PH6 + ...	
		S/T8L = (S/CXS) + ...	
PH7 T8L	A0-A31 → S0-S31	SXA = FAST PH7 + ...	
	S0-S31 → C0-C31		CXS set true in PH6
	C16-C31 ↛ D16-D31	DXC/2 = FAST PH7 + ...	
	0's ↛ A0-A31	AX/1 = FAST PH7 + ...	
	Enable memory request	MRQ = FAST PH7 + ...	To read TSA
PH8 T6L	C0-C31 ↛ D0-D31	DXC/6 = FAST PH8 + ...	Read TSA
	D0-D31 → S0-S31 ↛ A0-A31	SXD = FAST PH8 + ...	Increment/decrement values to A-register
		AXS = FAST PH8 (OLA + OL3) + ...	
PH9 T6L	P16-P31 → S16-S31 → C16-C31	SXP = FAST PH9 + ...	Hold TSA address in C
	1 ↛ PHA	S/PHA = FAST PH9 + ...	
	A16 ↛ CS0-CS15	CSX1/3 = FAST PH9 A16 + ...	For sign extension
	Enable memory request	MRQ = FAST PH9 OL3 + ...	To write TSA

Mnemonic: MSP (13, 93)

(Continued)

Table 3-110. Modify Stack Pointer, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10A T6L	D0-D31+A0-A31 → S0-S31	SXADD = FASTA PH10 + ...	Modify and write TSA
	S0-S31 → MB0-MB31	MW = FASTA PH10 + ...	
	1 ↛ LB31/1	(S/LB31/1) = FASTA PH10 + ...	
	Enable memory request	MRQ = FASTA PH10 + ...	To write count word
PH11A T6L	B0-B31 → S0-S31	SXB = FASTA PH11 + ...	Write modified count word
	S0-S31 → MB0-MB31	MW = FASTA PH11 + ...	
	S0-S31 ↛ A0-A31	AXS = FASTA PH11 + ...	Modified count word to A-register for test in PH12A
	Enable memory request	MRQ/1 = FASTA PH11 + ...	To read next instruction
PH12A T6L	SW1 = 1 => 1 ↛ CC1	S/CC1 = FASTA PH12 SW1 + ...	
	SW2 = 1=> 1 ↛ CC3	S/CC3 = FASTA PH12 SW2 + ...	
	If (A1-A15 = 0), 1 ↛ CC2	S/CC2 = FASTA PH12 A0115Z + ...	
	If (A17-A31 = 0), 1 ↛ CC4	S/CC4 = FASTA PH12 A1731Z + ...	
		(R/CC) = FASTA PH12 + ...	
	End	ENDE = FASTA PH12 + ...	
PH14A T8L	D0-D31 → S0-S31	SXD = FASTA PH14 + ...	D to A for test in PH12A
	S0-S31 ↛ A0-A31	AXS = FASTA PH14 + ...	
	Enable memory request	MRQ/1 = FASTA PH14 + ...	To read next instruction
	Go to PH12A	BRPH12 = FASTA PH14 + ...	

Mnemonic: MSP (13, 93)

The EXU instruction can be indexed, indirectly addressed, or both. The normal preparation sequence, as well as the sequencing for all modes and conditions of the EXU instruction, is described under Preparation Sequence.

The operand (the next instruction to be executed) is loaded into the D-register at PH1 clock. DRQ is set, and the CPU sequences to PH2.

During PH2 of the EXU instruction, ENDE is enabled, which causes the normal action of transferring C0-C31 to D0-D31, C0-C7 to O1-O7, and C8-C11 to R28-R31. The normal action of permitting the P-register to increase its count by one is suppressed; thus, P contains an address one greater than the address of the EXU instruction. The P + 1 count and the memory protection trap are disabled.

A sequence chart of the Execute instruction is given in table 3-111.

3-239 Branch on Conditions Set (BCS 69, E9)

The Branch on Conditions Set instruction forms the logical product of the R-field of the instruction word and the current condition code. If the logical product is nonzero, the branch condition is satisfied and the program proceeds with the instruction pointed to by the effective address of the instruction. If the logical product is zero, the branch condition is unsatisfied, and the instruction execution proceeds with the next instruction in sequence.

The execution sequence for the BCS instruction begins with PH3. During the preparation sequence the contents of the reference address are read into the D-register. In PH3 the D-register holds the next instruction to be performed if a branch is to be executed. Whether or not a branch is to be made depends on the result of the comparison of bits R28-R31 with bits CC1-CC4. The BCS instruction ends at PH3 if one bit of R28-R31 and the corresponding bit of the condition code compares.

A trap occurs if the CPU is denied access to the branch address because of a nonexistent memory address or because of a memory protection violation. In this case, the trap occurs, and the return address must be taken from the Q-register. Flip-flop BRQ is set to ensure that the return address is taken from Q during the trap operation.

If no bits of the R-field compare with the corresponding bits of the condition code, the CPU sequences to PH4. MRQ/1 is enabled in PH3 if no branch condition exists and the address of the next instruction is taken from the Q-register and is transferred into the P-register.

Phase PH5 is the final phase of the BCS instruction if no branch occurred. If the R-field of the BCS instruction contains all zeros, the term (R.CC) cannot come true and the BCS instruction acts as a no-operation type of instruction.

A sequence chart of the Branch on Conditions Set instruction is given in table 3-112.

3-240 Branch on Conditions Reset (BCR 68, E8)

The Branch on Conditions Reset instruction forms the logical product of the R-field of the instruction word and the current condition code. If the logical product is zero, the branch condition is satisfied, and the instruction proceeds with the instruction pointed to by the effective address of the instruction. If the logical product is nonzero, the branch condition is unsatisfied, and instruction execution proceeds with the next instruction in the sequence.

The preparation sequence for the BCR instruction is the same as that described under Preparation Sequence, except that the instruction branches from PRE2 to PH3 for the first execution phase. During the preparation sequence, the contents of the reference address are read into the D-register. In PH3, the D-register holds the next instruction to be performed if a branch is to be made. Whether or not a branch is to be made depends on the result of the comparison of bits R28 through R31 with bits CC1 through CC4. The logical product of each pair of corresponding bits is taken in the two registers. If all of the results are zero, signal ENDE is generated and the instruction in the D-register is executed. This is, in effect, a branch operation.

A trap occurs if the CPU is denied access to the branch address either because of a nonexistent memory address or because of a memory protection violation. In case of a trap, the return address must be taken from the Q-register. Flip-flop BRQ is set to ensure that, during the trap operation, the return address is taken from the Q-register.

If the logical product of any two corresponding bits is a one, the instruction goes to PH4 to access the next instruction in sequence. Signal MRQ/1 is enabled in PH3 if no branch condition exists, and the address of the next instruction is taken from the Q-register and is transferred to the P-register. If no branch has occurred, PH5 is the final phase of the instruction.

If the R-field of the instruction contains all zeros, the instruction acts as an unconditional branch instruction.

A sequence chart of the Branch on Conditions Reset instruction is given in table 3-113.

3-241 Branch on Incrementing Register (BIR 65, E5)

The BIR instruction counts up the private memory register R by one. If R is negative after the R-register has been counted up, the contents of the reference address location are taken as the next instruction. If R is not negative after the R-register has been counted up, the next instruction is taken from the location whose address is one greater than the address of the BIR instruction address.

The preparation sequence for the BIR instruction is the same as that described under Preparation Sequence, except that the CPU branches from PRE2 to PH2 for the first execution phase. Also during PRE2, the CS-register is set to equal a one to prepare for the incrementing action in PH2.

Table 3-111. Execute, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	C0-C31 ↛ D0-D31	DXC/6 = FUEXU PH1 + ...	Instruction to be executed into D
	Q15-Q31 ↛ P15-P31	PXQ = FUEXU PH1 + ...	
	Set DRQ	S/DRQ = FUEXU PH1 + ...	
	Go to PH2	S/PH2 = PH1 NBR + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	End	ENDE = FUEXU PH2 + ...	ENDE causes C ↛ O R, D
	Suppress P + 1 ↛ P	PCTP1DIS = ENDE (FUEXU + ...)	P now contains address of instruction after EXU address rather than the address of the instruction following the instruction read in PH1
	Suppress Instruction Protect	S/TRACC4 = NRESET (ENDE PROTECTI NFEUXU NTRAP) + ...	

Mnemonic: EXU (67, F7)

Table 3-112. Branch on Conditions Set, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	Branch to PH3 and enable T4RL	BRPH3 = FABC (PRE2 NIA)/1 + ...	
		FABC = OU6 O4 NO5 NO6	
		T4RL = N[FABC (PRE2 NIA)/1] + ...	
PH3 T4RL	Enable ENDE if branch	ENDE = FABC PH3 O7 (R.CC) + ...	
		(R.CC) = R28 CC1 + R29 CC2 + R30 CC3 + R31 CC4	
	If trap and branch, set BRQ	S/BRQ = (S/BRQ/1) (S/TRAP) ENDE + ...	Return address in Q-registe
	If branch only, go to PRE1	S/PRE1 = ENDE + ...	
		(S/BRQ/1) = FABC PH3 + ...	
	If no branch, 1 → MRQ/1	MRQ/1 = FABR + FABCNBRANCH + ...	
		FABCNBRANCH = FABC O7 N(R.CC) PH3	
	Suppress protect fail	(S/TRACC4/1) = PROTECTD NPROTECTDIS + ...	
		NPROTECTDIS = N(FABR + FABCNBRANCH)	
PH4 T6L	1 ⇸ DRQ	S/DRQ = FABC PH4 + ...	If no branch, next instruction address is in Q-register
	Q ⇸ P	PXQ = MRQ/1 + ...	
PH5 T6L	End of no branch	ENDE = FABC PH5 + ...	

Mnemonic: BCS (69, E9)

Table 3-113. Branch on Conditions Reset, Phase Sequence

Phase	Function Performed	Signals Involved	Comments

PRE2	Go to PH3 and enable T4RL	BRPH3 = FABC (PRE2 NIA)/1 + ...
		FABC = OU6 O4 NO5 NO6
		T4RL = N FABC (PRE2 NIA)/1 + ...	

PH3
T4RL
	Enable ENDE if branch	ENDE = FABC PH3 NO7 N(R.CC) + ...
		N(R.CC) = N(R28 CC1 + R29 CC2 + R30 CC3 + R31 CC4)	

	If trap and branch, set BRQ	S/BRQ = (S/BRQ/1) = (S/TRAP) ENDE + ...	Return address is in Q-register
	If branch only, go to PRE1	S/PRE1 = ENDE + ... (S/BRQ/1) = FABC PH3 + ...	
	If no branch, 1 → MRQ/1	MRQ/1 = FABR + FABCNBRANCH + ... FABCNBRANCH = FABC NO7 (R.CC) PH3 + ...	
	Suppress protect fail	(S/TRACC4/1) = PROTECTD NPROTECTDIS + ... NPROTECTDIS = N(FABR + FABCNBRANCH)	
PH4 T6L	1 $\not\longrightarrow$ DRQ	S/DRQ = FABC PH4 + ...	If no branch, next instruction address is in Q-register
	Q $\not\longrightarrow$ P	PXQ = MRQ/1 + ...	
PH5 T6L	End of no branch	ENDE = FABC PH5 + ...	

Mnemonic: BCR (68, E8)

In PH2 the A-register, which contains the contents of private memory register R, is added to the contents of the CS-register which contains +1. The result, (R) + 1, is clocked back into the A-register. The D-register contains zeros during this addition and does not pick up the contents of the branch instruction until the PH3 clock.

In PH3 the contents of A are stored back in register R unless the branch location is either nonexistent or memory protected. If the contents of the A-register are negative, PH3 is the final phase of the BIR instruction, and the data clocked from the C-register into D at the next clock pulse is the next instruction to be executed.

If the contents of the A-register are not negative, or are equal to zero, ENDE cannot come true during PH3, and the branch does not occur. MRQ/1 is enabled in PH3, and the CPU sequences to PH4. In PH4, DRQ is set, and the CPU goes to PH5 for the end of the BDR sequence.

A sequence chart of the Branch on Incrementing Register instruction is given in table 3-114.

3-242 Branch on Decrementing Register (BDR 64, E4)

The BDR instruction counts down the private memory register R by one. If, after the R-register has been counted down, R is positive, but not zero, the contents of the reference address location are taken as the next instruction. If, after the R-register has been counted down, R is negative or is equal to zero, the next instruction is taken from the location whose address is one greater than the address of the BDR instruction address.

The preparation sequence for the BDR instruction is the same as that described under Preparation Sequence, except that the CPU branches from PRE2 to PH2 for the first execution phase. Also during PRE2, the CS-register is set to all ones to prepare for the decrementing action in PH2.

In PH2 the A-register which contains the contents of private memory register R is added to the contents of the CS-register, which contains -1. The result, (R) - 1, is clocked back into the A-register. The D-register contains zeros during this addition and does not pick up the contents of the branch instruction until the PH3 clock.

In PH3 the contents of A are stored back into register R unless the branch location is either nonexistent or memory protected. If the contents of the A-register are positive, but are not equal to zero, PH3 is the final phase of the BDR instruction, and the data clocked from the C-register into D at the next clock pulse is the next instruction to be executed.

If the contents of the A-register are negative, ENDE cannot come true during PH3 and the branch does not occur. MRQ/1 is enabled in PH3, and the CPU sequences to PH4. In PH4, DRQ is set, and the CPU goes to PH5 for the end of the BDR execution sequence.

A sequence chart of the Branch on Decrementing Register instruction is given in table 3-115.

3-243 Branch and Link (BAL 6A, EA)

The BAL instruction loads the address of the BAL instruction plus one into register R and transfers the effective address into the P-register. The next instruction to be executed after BAL is the instruction located in the effective address of the BAL instruction.

The preparation sequence for the BAL instruction is the same as that described under Preparation Sequence.

In PH1 the effective address of the BAL instruction is transferred into the P-register from Q, and the current instruction address plus one is transferred from the Q-register into P. The CPU sequences from PH1 to PH2.

In PH2 the current instruction address in the P-register is transferred to register R. The Q-register, which contains the effective address, is clocked into P. This is the address to which the CPU unconditionally branches. The current instruction address plus one is also transferred from P to Q, since a trap occurs if the branch location in the P-register is either nonexistent or memory protected. MRQ/1 is enabled in PH2 to read the next instruction.

PH3 is the final phase of the instruction. Flip-flop BRQ is set in the event of a trap. This indicates that the return address is in the Q-register.

A sequence chart of the Branch and Link instruction is given in table 3-116.

3-244 Call 1 (CAL1, 04), Call 2 (CAL2, 05), Call 3 (CAL3, 06), Call 4 (CAL4, 07)

The Call instructions initiate programmed trap operations. The instructions CAL1, CAL2, CAL3, and CAL4 trap to locations 48, 49, 4A, and 4B, respectively. Normally, each of these trap locations contains an exchange program status doubleword instruction.

All Call instructions can be indirectly addressed, indexed, or both. The normal preparation sequence, as well as the sequencing for all modes and all conditions, is described under Preparation Sequence.

In PH1 the contents of the R-register (R-field of the Call instruction word) are transferred to flip-flops TRACC1-TRACC4. These flip-flops hold this data until it is used during the XPSD instruction following the trap sequence. The flip-flops TR28, TR30, and TR31 are set in different configurations, depending on whether the instruction is CAL1, CAL2, CAL3, CAL4. The flip-flops TR28, TR30, and TR31 are used during the trap sequence to determine the trap address 48, 49, 4A, or 4B. Flip-flop TRAP is set ct the PH1 clock.

Table 3-114. Branch on Incrementing Register, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	RR0-RR31 -/→ A0-A31	AXRR = FABR (PRE2 NIA)/1 + ...	Read register
	1 -/→ CS31	CSX1/8 = FABR (PRE2 NIA)/1 + ...	CS-register = +1
	Go to PH2	BRPH2 = FABR (PRE2 NIA)/1 + ...	
		T4RL = FABR (PRE2 NIA)/1 + ...	
PH2	A0-A31 + CS0-CS31 → S0-S31	SXADD = FABR PH2 + ...	A+1 -/→ A (D = 0 at this time)
	S0-S31 -/→ A0-A31	AXS = FABR PH2 + ...	
	1 -/→ DRQ	S/DRQ = FABR PH2 + ...	
	1 → T10L	S/T10L = FABR PH2 + ...	
PH3	A0-A31 → S0-S31 → RW0-RW31	SXA = FABR PH3 + ...	
		RW = FABR PH3 + ...	
		RWDIS = FABR ENDE PROTECTD + FABR ENDE PROTECTI	Disable A → RW if trap
	Branch if A is negative	ENDE = FABR PH3 O7 A0 + ...	Next instruction to D-register
	C0-C31 -/→ D0-D31	DXC = ENDE + ...	
		S/PRE1 = ENDE + ...	
	1 -/→ BRQ	S/BRQ = (S/BRQ/1) + ...	
		(S/BRQ/1) = FABR PH3 + ...	
	1 → MRQ/1	MRQ/1 = (FABR + FABCNBRANCH) + ...	
	Q15-Q31 -/→ P15-P31	PXQ = MRQ/1 + ...	
	Suppress Protect Fail	(S/TRACC4/1) = PROTECTD NPROTECTDIS + ...	
		NPROTECTDIS = N(FABR + FABCNBRANCH)	
PH4	1 -/→ DRQ	S/DRQ = FABR PH4 + ...	
PH5	End of no branch	ENDE = FABR PH5 + ...	

Mnemonic: BIR (65, E5)

Table 3-115. Branch on Decrementing Register, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE2	RR0-RR31 —/→ A0-A31	AXRR = FABR (PRE2 NIA)/1 + . . .	Read register
	1's —/→ CS0-CS31	CSX1/8 = FABR (PRE2 NIA)/1 + . . .	CS-register = -1
		CSX1 = FABR NO7 (PRE2 NIA)/1 + . . .	
	Go to PH2	BRPH2 = FABR (PRE2 NIA)/1 + . . .	
		T4RL = FABR (PRE2 NIA)/1 + . . .	
PH2	A0-A31 + CS0-CS31 → S0-S31	SXADD = FABR PH2 + . . .	A-1—/→A (D = 0 at this time)
	S0-S31 —/→ A0-A31	AXS = FABR PH2 + . . .	
	1 —/→ DRQ	S/DRQ = FABR PH2 + . . .	
	1 —/→ T10L	S/T10L = FABR PH2 + . . .	
PH3	A0-A31 → S0-S31 → RW0-RW31	SXA = FABR PH3 + . . .	
		RW = FABR PH3 + . . .	
		RWDIS = FABR ENDE PROTECTD + FABR ENDE PROTECTI	Disable A → RW if trap
	Branch if A is positive but not zero	ENDE = FABR PH3 NO7 NA0 NA0031Z + . . .	
	C0-C31 —/→ D0-D31	S/PRE1 = ENDE + . . .	
		DXC = ENDE + . . .	Next instruction to D-register
	1 —/→ BRQ	S/BRQ = (S/BRQ/1) = (S/TRAP) ENDE + . . .	
		(S/BRQ/1) = FABR PH3 + . . .	
	1 —/→ MRQ/1	MRQ/1 = (FABR + FABCN BRANCH) + . . .	
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + . . .	
	Suppress Protect Fail	(S/TRACC4/1) = PROTECTD NPROTECTDIS + . . .	
		NPROTECTDIS = N(FABR + FABCN BRANCH)	

Mnemonic: BDR (64, E4)

(Continued)

Table 3-115. Branch on Decrementing Register, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4	$1 \not\rightarrow$ DRQ	S/DRQ = FABR PH4 + ...	
PH5	End of no branch	ENDE = FABR PH5 + ...	

Mnemonic: BDR (64, E4)

Table 3-116. Branch and Link, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PH1	P15-P31 $\not\longrightarrow$ Q15-Q31	QXP	= FUBAL PH1 + ...	
	Q15-Q31 $\not\longrightarrow$ P15-P31	PXQ	= FUBAL PH1 + ...	Current instruction address +1 to P
		S/T8L	= FUBAL PH1 + ...	
PH2	P15-P31 → S15-S31	SXP	= FUBAL PH2 + ...	
	S0-S31 → RW0-RW31	RW	= FUBAL PH2 + ...	Current instruction address +1 to R
	P15-P31 $\not\longrightarrow$ Q15-Q31	QXP	= FUBAL PH2 + ...	P to Q in case of trap
	Q15-Q31 $\not\longrightarrow$ P15-P31	PXQ	= FUBAL PH2 + ...	Effective address to P
	Enable memory request	MRQ/1	= FUBAL PH2 + ...	
		S/DRQ	= FUBAL PH2 + ...	
PH3	End of BAL	ENDE	= FUBAL PH3 + ...	
	If trap, 1 $\not\longrightarrow$ BRQ	S/BRQ	= ENDE (S/TRAP) (S/BRQ/1) + ...	BRQ indicates that the return address is in the Q-register
	(Memory protection violation)	(S/BRQ/1)	= FUBAL + ...	
		(S/TRAP)	= PROTECTI ENDE NFUEXU	

Mnemonic: BAL (6A, EA)

In PH2 of a Call instruction, the flip-flops INTRAPF, INTRAP1, CX/1, and BRQ are set. The term ENDE does not come true during the final phase of a Call instruction. The address of the next instruction is in the Q-register as the Call instruction sequences to the trap phase sequence.

Trap sequencing is discussed under Trap Operation. For details of the XPSD instruction, see the instruction description.

A sequence chart of the Call instructions is given in table 3-117.

3-245 Load Program Status Doubleword (LPSD 0E, 8E)

The LPSD instruction loads the effective doubleword (EW and EW +1) in memory into the program status doubleword; EW is loaded into PSW1, and EW + 1 is loaded into PSW2. If bit position 8 of the LPSD instruction is a one, bits 23 through 27 of EW + 1 are loaded into the register pointer RP23 through RP27. If bit position 8 is not a one, the register pointer is not changed.

During the execution of the LPSD instruction, the highest priority interrupt level that is currently active is affected by the status of bit positions 10 and 11 of the LPSD instruction.

The LPSD instruction can be indirectly addressed, but not indexed. The normal preparation sequence, as well as the sequencing for the indirect addressing mode of this instruction, is described under Preparation Sequence.

If the CPU is in the slave mode and if the privileged LPSD instruction is not being executed as the result of a trap or an interrupt (INTRAPF false), the LPSD aborts and the CPU traps to location 40 at the end of preparation state PRE2.

A memory request is made for EW and then for EW + 1. The effective word EW in the C-register is transferred to D.

The effective word EW in D is transferred to the sum bus, and the contents of the sum bus are transferred to the A-register. All bits of PSW1 except the P-register are loaded from the sum bus. The following flip-flops are unconditionally reset: MAPDIS, CI, II, EI, and CIF. INTRAPF is reset if the watchdog timer has not run down. The effective word plus one is transferred from the C-register into D.

The effective word in the A-register is transferred to the sum bus, and the 16 least significant bits of the sum bus are clocked into the P-register. The significant bits of EW + 1 (PSW2) in the D-register are clocked into the current program status word 2 as the D-register is cleared to zeros. If R28 is true, the register pointer is changed; otherwise, it is not affected. If R30 is true, the highest priority interrupt level currently active is cleared. If R30 and R31 are both true, this interrupt level is cleared but is armed.

If R30 is true because the highest priority interrupt currently active is to be disabled, the CPU clock enable signal CE must be synchronized with the interrupt clock.

The effective word EW in the A-register is added to the D-register (which now contains zeros), and the result on the sum bus is clocked into the P-register.

A sequence chart of the Load Program Status Doubleword instruction is given in table 3-118.

3-246 Exchange Program Status Doubleword (XPSD 0F, 8F)

The XPSD instruction stores the current PSD into the memory doubleword location specified by the contents of the instruction reference address field, and loads the contents of the successive doubleword in memory into the current PSD. Four memory words are involved in this operation (two doublewords), and they are referred to as EW, EW + 1, EW + 2 and EW + 3, where EW represents the effective word addressed by the reference address field. The current PSD is stored into EW and EW + 1, and the contents of EW + 2 and EW + 3 are loaded into the current PSD.

Because the XPSD instruction is privileged, an attempt to execute this instruction in the slave mode causes a trap to location 40. However, if the CPU is in the slave mode and an interrupt or trap occurs for any reason, the XPSD instruction that immediately follows the interrupt or trap sequence is allowed.

An XPSD instruction can be executed as the result of any of the following conditions:

a. During the course of a program or subroutine in the master mode but not as the result of an interrupt or trap (NINTRAPF NTRAP).

b. Immediately following an interrupt sequence (INTRAPF NTRAP).

c. A trap to location 40 as the result of a nonexistent instruction, a nonexistent memory address, an attempt to execute a privileged instruction in the slave mode, or as the result of a memory protection violation (INTRAPF TRAP TRACC1 + TRACC2 + TRACC3 + TRACC4 = 1).

d. A trap to location 40 as the result of an interrupt system fault, or a trap to trap locations 41 through 46 [INTRAPF TRAP [(TRACC1 + TRACC2 + TRACC3 + TRACC4 = 0)]

e. A trap resulting from any Call instruction [INTRAPF TRAP (TRACC1 + TRACC2 + TRACC3 + TRACC4 = 1)].

Depending upon previous conditions, special actions described in the following paragraph are taken during the execution of the XPSD instruction.

Table 3-117. Call 1, Call 2, Call 3, Call 4, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	R28-R31 ⇸ TRACC1-TRACC4	R/TRACC1-TRACC4 = NTRAP	
		S/TRACC1 = FACAL PH1 NTRAP NSTRAP R28 + ...	
		S/TRACC2 = FACAL PH1 NTRAP NSTRAP R29 + ...	Hold R-field for use later during XPSD
		S/TRACC3 = FACAL PH1 NTRAP NSTRAP R30 + ...	
		S/TRACC4 = FACAL PH1 NTRAP NSTRAP R31 + ...	
	Set TR28	S/TR28 = FACAL PH1 NTRAP NSTRAP + ...	TR28, TR30, TR31 are used for trap address into P-register during trap phase
	If CAL3 or CAL4, set TR30	S/TR30 = FACAL PH1 O6 + ...	
	If CAL2 or CAL4, set TR31	S/TR31 = FACAL PH1 O7 NSTRAP NTRAP + ...	
	Set TRAP	S/TRAP = (S/TRAP) NRESET	
		(S/TRAP/1) = FACAL PH1 + ...	
	Set PH2	S/PH2 = PH1 NBR + ...	
PH2 T6L	T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set INTRAPF	S/INTRAPF = (S/INTRAPF) NRESET	
		(S/INTRAPF) = TRAP NINTRAPF + ...	
	Set INTRAP1	S/INTRAP1 = (S/INTRAPF) NRESET	(INTRAP1 CX/1) true for the first phase of trap sequence
		S/CX/1 = (S/INTRAPF) + ...	
	Set BRQ	S/BRQ = (S/INTRAPF) NENDE N(PRE1 NANLZ) + ...	
	Clear	CLEAR = (S/INTRAPF) + ...	

Mnemonic: CAL1-2-3-4, 04, 05, 06, 07

Table 3-118. Load Program Status Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	PSW1 —/→ D0-D31	DXPSW1 = FAPSD PH1 + ...	Not meaningful for LPSD
	Request EW	MRQ = FAPSD NO7 PH1 + ...	
	Set data release request	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
	Go to PH5	S/PH5 = BRPH5 + ...	
		BRPH5 = FAPSD PH1 NO7 + ...	
PH5 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	P + 1 —/→ P	PCTP1 = FAPSD PH5 + ...	Increment P-register
	C0-C31 —/→ D0-D31	DXC/6 = FAPSD PH5 + ...	EW into D
	Request EW + 1	MRQ = FAPSD PH5 + ...	
	Set data release request	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
	Go to PH6	S/PH6 = PH5 NBR + ...	
PH6 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	D0-D31 ——→ S0-S31	SXD = FAPSD PH6 + ...	
	S0-S31 —/→ PSW1	PSW1XS = (FAPSD PH6 + ...) N(S/INTRAPF)	EW into PSW1
	S0-S31 —/→ A0-A31	AXS = FAPSD PH6 + ...	EW into A
	C0-C31 —/→ D0-D31	DXC/6 = FAPSD PH6 + ...	EW + 1 into D
	Reset MAPDIS	R/MAPDIS = FAPSD PH6 + ...	
	0 —/→ CIF	R/CIF = FAPSD NO7 PH6	
	0 —/→ II	R/II = FAPSD NO7 PH6 + ...	
	0 —/→ EI	R/EI = FAPSD NO7 PH6 + ...	
	S0-S3 —/→ CC1-CC4	CCXS = FAPSD PH6 + ...	
	S5 —/→ FS	S/FS = FSZNXS S5	Floating significance bit
	S6 —/→ FZ	S/FZ = FSZNXS S6	Floating zero bit

Mnemonic: LPSD (0E, 8E)

(Continued)

Table 3-118. Load Program Status Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6 T6L (Cont.)	S7 —/→ FNF	S/FNF = FSZNXS S7 FSZNXS = PSW1XS	Floating normalize bit
	S8 —/→ MASTERF	S/MASTERF = PSW1XS S8	Master-slave bit
	S9 —/→ MAPF	S/MAPF = PSW1XS S9	Map control bit
	S10 —/→ DM	S/DM = PSW1XS S10	Decimal mask bit
	S11 —/→ AM	S/AM = PSW1XS S11	Arithmetic mask bit
	Set data release request	S/DRQ = FAPSD NPH7 NPH8 EXU + . . .	
	Inhibit single clock enable signal SCEN	SCEN = N(FAPSD PH6 NO7 R30)	Synchronizes ac clock with 1-MHz clock during single clock operation by disabling latch SCD (see Single Clock Operation in section on Basic Timing)
	Go to PH7	S/PH7 = PH6 NBR + . . .	
PH7 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	D2-D3 —/→ Write Keys	S/WK0 = FAPSD PH7 D2 + . . . S/WK1 = FAPSD PH7 D3 + . . .	
	D5 —/→ CIF	S/CIF = FAPSD PH7 D5 + . . .	EW + 1 into PSW2
	D6 —/→ II	S/II = FAPSD PH7 D6 + . . .	
	D7 —/→ EI	S/EI = FAPSD PH7 D7 + . . .	
	If R28 = 1, D23-D27 —/→ RP23-RP27	RPXD = FAPSD PH7 R28 + . . .	If R28, change register pointer
	If R30 = 1, 1 —/→ LEVACT	LEVACT = FAPSD NO7 PH7 R30 + . . .	Disable highest priority interrupt level currently active
	If R30 and R31 = 1, 1 —/→ LEVARM	LEVARM = FAPSD NO7 PH7 R31R30 + . . .	
		CE = NCEINT + ARE	

Mnemonic: LPSD (0E, 8E)

(Continued)

Table 3-118. Load Program Status Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T6L (Cont.)	Synchronize clock	CEINT = FAPSD NO7 PH7 R30 + ...	Synchronizes next clock with interrupt 1 -MHz clock by disabling clock enable signal CE until 1 MHz clock occurs (see Basic Timing)
	Reset flip-flop INTRAPF	R/INTRAPF = FAPSD PH7 + ...	
	A0-A31 → S0-S31	SXA = FAPSD PH7 + ...	
	S15-S31 ⇸ P15-P31	PXS = FAPSD PH7 + ...	Next instruction address into P and again in PH8
	Reset TRAP	R/TRAP = FAPSD PH7 + ...	
	0's ⇸ D0-D31	DX/1 = FAPSD PH7 + ...	
	Go to PH8	S/PH8 = PH7 NBR + ...	
PH8 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	A0-A31 + D0-D31 → S0-S31	SXADD = FAPSD PH8 + ...	Effectively, A to sum bus, since D is cleared
	S15-S31 ⇸ P15-P31	PXS = FAPSD PH8 + ...	Next instruction address to P
	Request next instruction	MRQ = FAPSD PH8 + ...	
	Go to PH9	S/PH9 = PH8 NBR + ...	
PH9 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	0's ⇸ TRACC1-TRACC4	R/TRACC = FAPSD PH9 + ...	
	Set data release request	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
	Go to PH10	S/PH10 = PH9 NBR + ...	
PH10 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	End of LPSD	ENDE = FAPSD PH10 + ...	
	Set data release request	S/DRQ = FAPSD NPH7 NPH8 NRESET + ...	

Mnemonic: LPSD (0E, 8E)

Instruction bits 8, 9, and 10 (which are loaded into R28, R29, and R30 as the instruction is read from memory) have special meanings and affect the execution of the instruction in the following ways:

Instruction Bits			
8	9	10	
(R28	R29	R30)	
x	x	1	If the memory map is in effect (bit 9 of the current PSD = 1), it remains enabled
x	x	0	If the memory map is in effect (bit 9 of the current PSD = 1), it is disabled for the duration of the XPSD instruction execution
0	x	x	The XPSD instruction does not change the contents of the register pointer (RP-register) of the current PSD
1	x	x	The XPSD instruction loads the contents of bits 23 through 27 of EW + 3 into the RP-register of the current PSD
x	1	x	If the trap has occurred to trap location X'40', merge the contents of TRACC1-TRACC4 into the condition code bits CC1-CC4 and into the next instruction address. If the trap occurred because of the execution of a Call instruction, add the contents of the R-field of the Call instruction to the next instruction address. (The R-field of the Call instruction is placed in TRACC1 through TRACC4 during the execution of the Call instruction.)

The XPSD instruction can be indirectly addressed but not indexed. The normal preparation sequence, as well as the sequencing for the indirect addressing mode of this instruction, is described under Preparation Sequence. The XPSD aborts and the CPU traps to location X'40' at the end of preparation state PRE2 if the CPU is in the slave mode and the privileged XPSD instruction is not being executed as the result of a trap or an interrupt (INTRAPF false).

S/TRAP = FAPRIV (PRE1 NANLZ) NMASTER

(S/INTRAPF) = TRAP NINTRAPF

S/PH1 = (S/PH1/1) NCLEAR

CLEAR = (S/INTRAPF)

S/INTRAP1 = NRESET (S/INTRAPF)

S/CX/1 = (S/INTRAPF)

A flow chart of the instruction is shown in figure 3-192. The current PSW1 (except for the contents of the P-register) is transferred to the D-register, and if R31 is not true, the memory map is disabled. If INTRAPF is not true, the operand address in the P-register is transferred to Q, and the next instruction address in the Q-register is transferred to P. If INTRAPF is true, the contents of the P- and Q-registers are not exchanged, and MRQ is enabled.

If the instruction is not part of an interrupt or a trap sequence, the next instruction address in the P-register is placed on the sum bus and is then clocked into the B-register. At the same time, the address of EW is returned to the P-register where it had been temporarily stored in the Q-register during PH1.

The sum bus performs an OR function on the contents of D (which contains the current PSW1) and B (which contains the next instruction address). The next instruction address was placed into B during an interrupt or trap sequence, and the result is written into memory at the operand address location. The current PSW2 is gated into the D-register, a second memory write cycle is requested, DRQ is set, P is counted up by one, and the CPU sequences to PH4.

The PSW2 in D is written into memory at the effective word location plus one. The P-register is counted up by one. The effective word plus two from memory, which has been read into the C-register, is clocked into D and the P-register counts up by one. The contents of the D-register are placed on the sum bus, and the sum bus is read into PSW1 and into the A-register. If flip-flop TRAP is true, TRACC1-TRACC4 are gated into CC1-CC4. Since S0-S3 are also gated to CC1-CC4, TRACC1-TRACC4 are effectively merged into CC1-CC4.

The contents of A (EW + 2) are placed on the sum bus, and the 16 least significant bits of the sum bus are clocked into the P-register. The D-register (which has picked up [EW + 3]) is clocked into PSW2. The D-register is reset to zeros unless TRAP is true, in which case the contents of TRACC1-TRACC4 are clocked into D28 through D31, provided that R29 is also true. If R29 is not true, the CPU requests the next instruction from memory and sequences to PH9; otherwise, it goes to PH8.

If R29 is false, the contents of the A- and D-registers are added, and the result on the sum bus is transferred to the P-register. This operation, in effect, adds the state of the TRACC1-TRACC4 flip-flops to the next instruction address before the request is made for the next instruction.

A sequence chart of the Exchange Program Status Doubleword instruction is given in table 3-119.

3-247 Load Register Pointer (LRP 2F, AF)

The LRP instruction loads bits 23 through 27 of the effective word into the RP-register, RP23 through RP27. The RP-register is part of the current program status doubleword. No other portion of the PSDW is affected.

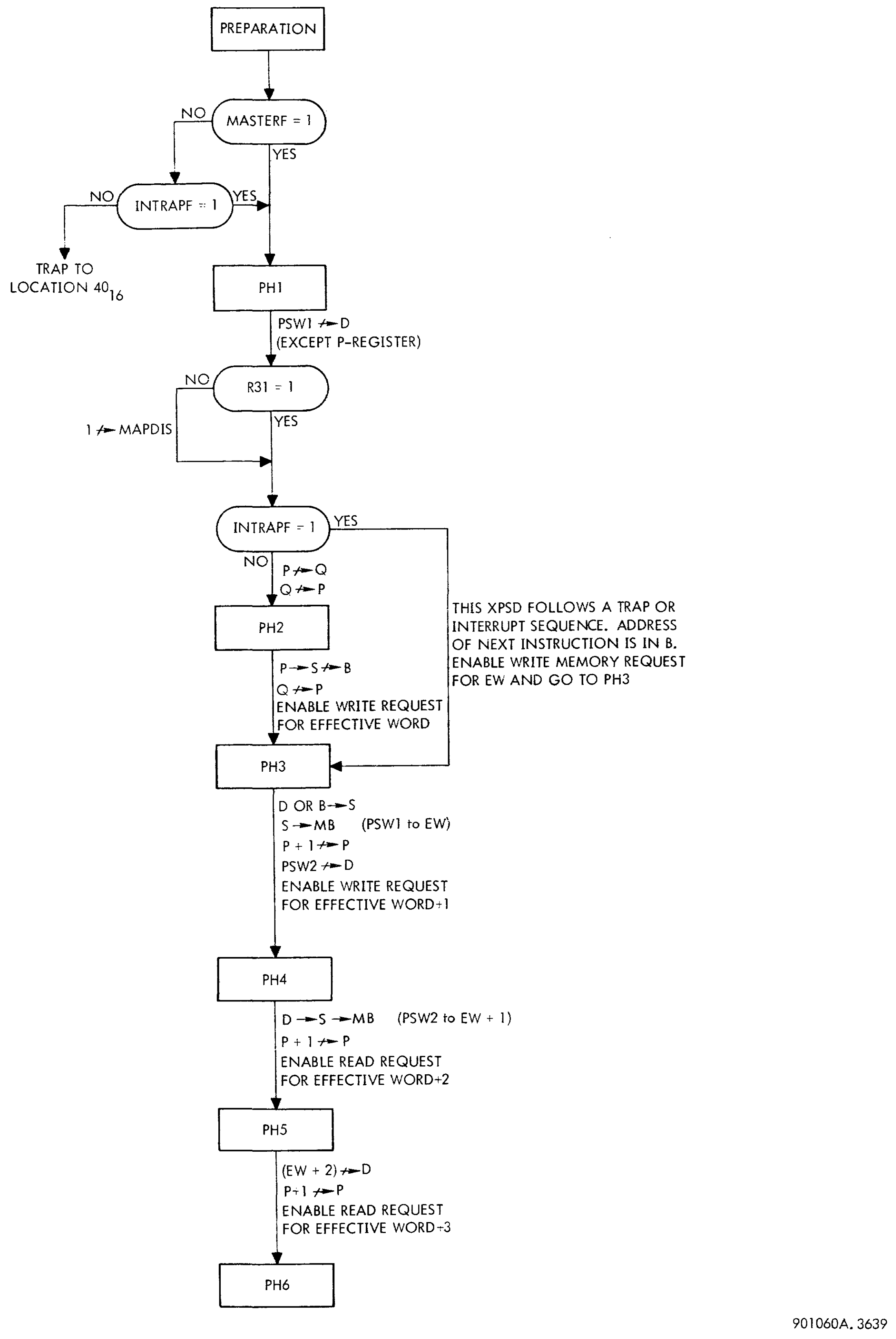


Figure 3-192. Exchange Program Status Doubleword, Logic Sequence Diagram (Sheet 1 of 2)

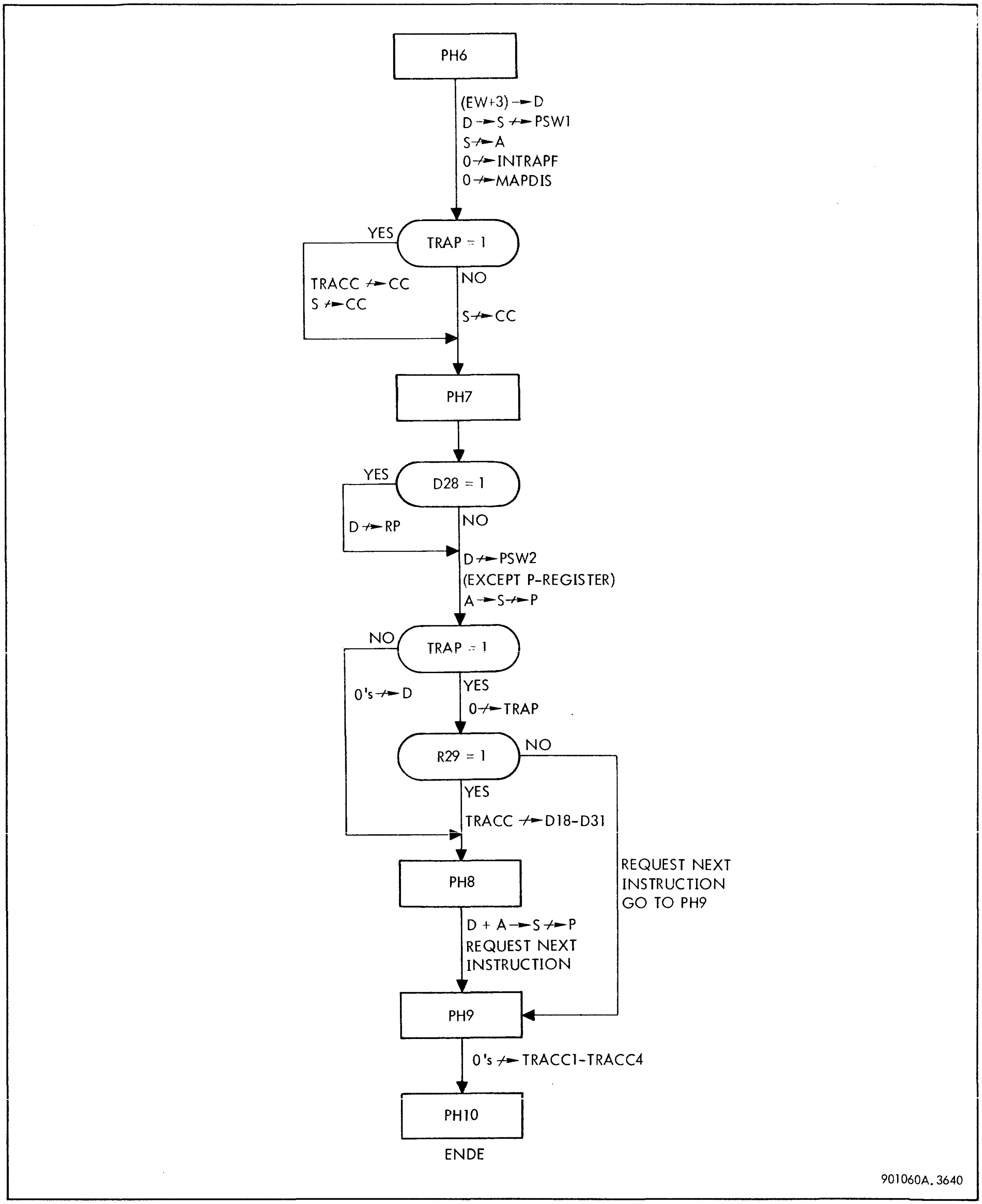


Figure 3-192. Exchange Program Status Doubleword, Logic Sequence Diagram (Sheet 2 of 2)

Table 3-119. Exchange Program Status Doubleword, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	PSW1 —/→ D0-D11	DXPSW1 = FAPSD PH1 + ...	0's —/→ D12-D31
	CC1-CC4 —/→ D0-D3	S/D0-D3 = CC1-CC4 DXPSW1 + ...	Condition code
	FS —/→ D5	S/D5 = FS DXPSW1 + ...	Floating significance bit
	FZ —/→ D6	S/D6 = FZ DXPSW1 + ...	Floating zero bit
	FNF —/→ D7	S/D7 = FNF DXPSW1 + ...	Floating normalize bit
	NMASTERF —/→ D8	S/D8 = NMASTERF DXPSW1 + ...	Master-slave bit
	MAPF —/→ D9	S/D9 = MAPF DXPSW1 + ...	Map control bit
	DM —/→ D10	S/D10 = DM DXPSW1 + ...	Decimal mask bit
	AM —/→ D11	S/D11 = AM. DXPSW1 + ...	Arithmetic mask bit
	0 —/→ MAPDIS	R/MAPDIS = FAPSD O7 PH1 + ...	
	If R30 = 1, 1 —/→ MAPDIS	S/MAPDIS = FAPSD O7 PH1 NR30 + ...	NR30 => MP bit = 0
	If INTRAPF = 1, set memory request and go to PH3	BRPH3 = FAPSD O7 PH1 INTRAPF + ...	Part of interrupt or trap sequence
		MRQ = FAPSD O7 PH1 INTRAPF + ...	
	If INTRAPF = 0, exchange P and Q and go to PH2	PXQ = FAPSD PH1 O7 NINTRAPF + ...	Next instruction address
		QXP = FAPSD PH1 O7 NINTRAPF + ...	Operand address
		S/PH2 = PH1 NBR N(FNANLZ NANLZ) + ...	If not part of interrupt or trap sequence
		S/PH3 = PH1 BRPH3 + ...	
		S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	P15-P31 ——→ S15-S31	SXP = FAPSD PH2 + ...	Next instruction address into B
	S0-S31 —/→ B0-B31	BXS = FAPSD PH2 + ...	
	Q15-Q31 —/→ P15-P31	PXQ = FAPSD PH2 + ...	Address of EW into P
	P15-P31 —/→ Q15-Q31	QXP = FAPSD PH2 + ...	Next instruction address
	Write memory request	MRQ = FAPSD PH2 + ...	

Mnemonic: XPSD (0F, 8F)

(Continued)

Table 3-119. Exchange Program Status Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)		S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
	Go to PH3	S/PH3 = PH2 NBR N(FNANLZ NANLZ) + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	D or B → S	SXD = FAPSD PH3 + ...	Current PSW1 or next instruction address
		SXB = FAPSD PH3 + ...	
	S0-S31 → MB0-MB31	MBXS/0 = MW	PSW1 or next instruction address → effective word location
		MW = FAPSD PH3 + ...	
	P + 1 ↛ P	PCTP1 = FAPSD PH3 + ...	Address of EW + 1 into P
	PSW2 ↛ D	DXPSW2 = FAPSD PH3 + ...	Program status doubleword 2
	WK0, WK1 ↛ D2, D3	S/D2, D3 = WK0, WK1 DXPSW2 + ...	Write keys
	CIF ↛ D5	S/D5 = CIF DXPSW2 + ...	Counter inhibit bit
	II ↛ D6	S/D6 = II DXPSW2 + ...	Interrupt inhibit bit
	EI ↛ D7	S/D7 = EI DXPSW2 + ...	External interrupt inhibit bit
	RP23-RP27 ↛ D24-D27	S/D24-D27 = (RP23-RP27) DXPSW2 + ...	Register pointer
	Write memory request	MRQ = FAPSD PH3 + ...	
		S/DRQ = FAPSD NPH7 NPH8 EXU + ...	Inhibits clock until data release received from memory
	Go to PH4	S/PH4 = PH3 NBR + ...	
PH4 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	D0-D31 → S0-S31	SXD = FAPSD PH4 + ...	
	S0-S31 → MB0-MB31	MBXS/0 = MW	PSW2 → EW + 1 location
		MW = FAPSD PH4 + ...	

Mnemonic: XPSD (0F, 8F)

(Continued)

Table 3-119. Exchange Program Status Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	Read memory request	MRQ = FAPSD PH4 + ...	
	Set data request	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
	P + 1 -/→ P	PCTP1 = FAPSD PH4 + ...	Address of EW + 2 into P
PH5 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	C0-C31 -/→ D0-D31	DXC/6 = FAPSD PH5 + ...	EW + 2 -/→ D
	P + 1 -/→ P	PCTP1 = FAPSD PH5 + ...	Address of EW + 3 into P
	Read memory request	MRQ = FAPSD PH5 + ...	
	Set data request	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
PH6 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	C0-C31 -/→ D0-D31	DXC/6 = FAPSD PH6	EW + 3 -/→ D
	D0-D31 → S0-S31	SXD = FAPSD PH6	
	S0-S11 -/→ PSW1	CCXS = PSW1XS	EW + 2 -/→ PSW1
		PSW1XS = FAPSD PH6 N(S/INTRAPF) + ...	
	S0-S3 -/→ CC1-CC4	S/CC1-CC4 = (S0-S3) CCXS + ...	Condition code
	S5 -/→ FS	S/FS = S5 FSZNXS + ...	Floating significance bit
	S6 -/→ FZ	S/FZ = S6 FSZNXS + ...	Floating zero bit
	S7 -/→ FNF	S/FNF = S7 FSZNXS	Floating normalize bit
		FSZNXS = PSW1XS	
	S8 -/→ MASTERF	S/MASTERF = S8 PSW1XS	Master-slave bit
	S9 -/→ MAPF	S/MAPF = S9 PSW1XS	Map control bit
	S10 -/→ DM	S/DM = S10 PSW1XS	Decimal mask bit
	S11 -/→ AM	S/AM = S11 PSW1XS	Arithmetic mask bit
	S0-S31 -/→ A0-A31	AXS = FAPSD PH6 + ...	EW + 2
	0 -/→ MAPDIS	R/MAPDIS = FAPSD PH6 + ...	

Mnemonic: XPSD (0F, 8F)

(Continued)

Table 3-119. Exchange Program Status Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6 T6L (Cont.)	If TRAP = 1, TRACC1-4 ⇸ CC1-4	CCXTRACC = FAPSD O7 PH6 TRAP + ...	Merge CC1-CC4 and TRACC1-TRACC4
	Go to PH7	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
		S/PH7 = PH6 NBR + ...	
PH7 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If D28 = 1, D23-D27 ⇸ RP23-RP27	RPXD = FAPSD PH7 R28 + ...	Bits 55-59 of second effective doubleword ⇸ register pointer
		DX/1 = FAPSD PH7 + ...	
	D2-D3, D5-D7 ⇸ PSW2	S/WK0 = FAPSD PH7 D2 + ...	Write keys
	(EW + 3)	S/WK1 = FAPSD PH7 D3 + ...	
		S/CIF = FAPSD PH7 D5 + ...	Counter interrupt inhibit bit
		S/II = FAPSD PH7 D6 + ...	Input/output interrupt inhibit bit
		S/EI = FAPSD PH7 D7 + ...	External interrupt inhibit bit
	If TRAP R29, TRACC ⇸ D28-D31	DXTRACC = FAPSD PH7 TRAP R29 + ...	If part of trap sequence, request next instruction
	If TRAP NR29, 1 → MRQ and go to PH9	MRQ = FAPSD PH7 O7 NR29 + ...	
		BRPH9 = FAPSD PH7 O7 NR29 + ...	
	Reset flip-flop INTRAPF	R/INTRAPF = FAPSD PH7	
	A0-A31 → S0-S31	SXA = FAPSD PH7 + ...	EW + 2
	S15-S31 ⇸ P15-P31	PXS = FAPSD PH7 + ...	Address in EW + 2
	If NTRAP, go to PH8	S/PH8 = PH7 NBR + ...	
PH8 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: XPSD (0F, 8F)

(Continued)

Table 3-119. Exchange Program Status Doubleword, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH8 T6L (Cont.)	A0-A31 + D0-D31 ⟶ S0-S31	SXADD = FAPSD PH8 + ...	Adds states of trace flip-flops to next instruction address. Transfer to P
	S15-S31 ↛ P15-P31	PXS = FAPSD PH8 + ...	
	Request next instruction	MRQ = FAPSD PH8 + ...	
	Go to PH9	S/PH9 = PH8 NBR + ...	
PH9 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	0's ↛ TRACC1-TRACC4	R/TRACC1-TRACC4 = FAPSD PH9 + ...	
	Go to PH10	S/DRQ = FAPSD NPH7 NPH8 EXU + ...	
PH10 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	End of execution	ENDE = FAPSD PH10 + ...	
		S/DRQ = FAPSD NPH7 NPH8 EXU + ...	

Mnemonic: XPSD (0F, 8F)

The LRP instruction can be indexed, indirectly addressed, or both. The normal preparation sequence, as well as the sequencing for all modes and conditions of the LRP instruction, is described under Preparation Sequence.

At the PH1 clock pulse, the effective word in the C-register is transferred to D.

At the PH2 clock pulse, bit positions 23 through 27 of the D-register are transferred to RP23 through RP27. MRQ/1 is enabled to start the memory request for the next instruction, and DRQ is set.

A sequence chart of the Load Register Pointer instruction is given in table 3-120.

The P + 1 count and the memory protection trap are disabled.

3-248 Move to Memory Control (MMC 6F, EF)

MMC loads a string of one or more words into one of the three banks of high speed memory registers. Bit positions 12 through 14 of the MMC instruction word are not used as an index register address; instead, they are used to specify which bank of high speed memory registers is to be loaded, as indicated in table 3-121.

The instruction produces an undefined result if bit positions 12 through 14 contain either all zeros or more than a single one. Bit positions 15 through 31 (reference address field) are ignored insofar as the operation of the instruction is concerned, and the results of the instruction are the same whether or not the instruction is indirectly addressed.

The R-field of the instruction word designates an even-odd pair of private memory registers that are used to control the loading of the specified bank of high speed memory registers. Bit positions 15 through 31 of the even numbered private memory register contain the address of the first word of the data to be loaded into the specified bank of high speed registers. Bit positions 0 through 7 of the odd numbered private memory register contain a count of the number of words to be loaded. If bits 0 through 7 are initially all zeros, a word count of 256 is implied.

Bit positions 15 through 22 (15 through 20 for load program control registers or load write lock registers) of the odd numbered private memory register indicate the particular fast access memory module half and bits to be written into.

During PRE1, the address of the next instruction contained in the P-register is clocked into the Q-register, and the A-, B-, and E-registers are cleared.

Since there must be a one in bit position D12, D13, or D14 to specify the bank of high speed memory registers to be loaded, signal INDX is generated, flip-flop IX is set, and the bit is gated onto the LR address lines as though the instruction had been indexed. The data contained in the addressed index register is clocked into the A-register at the PRE1 clock. Flip-flop PRE2 is set, and clock T4RL is enabled.

During PRE2, the contents of the D-register and the contents of the A-register are gated into the adder, and an addition operation is performed. The result is gated onto the sum bus and is clocked into flip-flops P15 through P31 at the PRE2 clock. This address is not used by the MMC instruction and is cleared during PH2. The A- and D-registers are cleared, and flip-flop IX is reset. If bit D12 is a one, indicating a load map function, flip-flop SW1 is set; if bit D13 is a one, indicating a load program control function, flip-flop SW2 is set; if bit D14 is a one, indicating a load write lock function, flip-flop SW3 is set.

Flip-flop SW4 is set. If the instruction is indirectly addressed, flip-flop SW4 inhibits changing SW1, SW2, or SW3 during the second PRE2 from the new information contained in the D-register. Signal (S/PH1/1) is generated, flip-flops PH1 and EXU are set, and clock T4RL is enabled.

The even numbered private memory register contains an image address in bit positions 15 through 31. This address points to the core memory location containing the first word to be read out. At the PH1 clock the contents of the even numbered private memory register are clocked into the A-register. Flip-flops P32 and P33 are cleared to select byte 0 as the first byte to be stored.

A one is forced on address line LR31 to select the odd numbered private memory register. Flip-flop PH2 is set, and clock T4RL is enabled. The data word containing the image address is gated from the A-register onto the sum bus. At the PH2 clock, the address contained in S15 through S31 is clocked into flip-flops P15 through P31 and is immediately gated onto the LM and LB address lines to read the first word from memory. A memory request is generated for the first word, and flip-flop ARQ is set. Setting of flip-flop ARQ inhibits transmission of another clock until the address release signal is received.

When the data from the odd numbered private memory register is available on the RR lines, it is clocked into the A-register at the PH2 clock. The data word from the odd numbered register contains the word count and control start address. The instruction sequences to PH3, and clock T4RL is enabled.

During PH3, the data from the A-register is gated onto the sum bus. The word count contained in bit positions S0 through S7 is clocked into flip-flops E0 through D7, and the control start address contained in bit positions S15 through S22 (or S20) is clocked into flip-flops P15 through P22 (or P20). Flip-flop DRQ is set, inhibiting transmission of another clock until the data release signal is received. When the first word is available from core memory, signal CXMB is generated and gates the data into the C-register. At the PH3 clock, the image address from the even numbered private memory register is clocked back into the

Table 3-120. Load Register Pointer, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	C0-C31 —/→ D0-D31	DXC/6 = FULRP PH1 + ...	
	Go to PH2	S/PH2 = PH1 NBR + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	D23-D27 —/→ RP23-RP27	RPXD = FULRP PH2 + ...	
	Enable MRQ/1	MRQ/1 = FULRP PH2 + ...	
	Set DRQ	S/DRQ = FULRP PH2 + ...	
	Go to PH3	S/PH3 = PH3 NBR + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	End	ENDE = FULRP PH3 + ...	

Mnemonic: LRP (2F, AF)

Table 3-121. Bit Position Functions

BIT POSITIONS			FUNCTION
12	13	14	
1	0	0	Load memory map registers
0	1	0	Load program control registers
0	0	1	Load write lock registers

A-register. The instruction sequences to PH4, and clock T6L is enabled.

Phase PH4 is repeated four times. During the first pass through PH4, flip-flops P32 and P33 are both zeros. Signal DXCR24 is generated and, at the PH4 clock, bits C0 through C7 are downward aligned into flip-flops D24 through D31. A one is added to bit P33 to select the next byte. During the second pass through PH4, signal DXCR16 is generated, and bits C8 through C15 are downward aligned into flip-flops D24 through D31. Flip-flops P32 and P33 are cleared, and a one is added to P32. During the third pass through PH4, signal DXCR8 is generated, and bits C16 through C23 are downward aligned into flip-flops D24 through D31. A one is then added to bit P33, making P32 and P33 both true. During the fourth pass through PH4, signal DXC is generated, and bits C24 through C31 are clocked into flip-flops D24 through D31. Flip-flops P32 and P33 are then reset to zero for the next word.

The address contained in the P-register is gated onto the LM and LB lines. If flip-flop SW1 is true, indicating load memory map registers, an address is contained only on address lines LM15 through LM22. The address contained on LM15 through LM19 is gated onto map address lines L/MP0/1 through L/MP15/1 and L/MP0/2 through L/MP15/2. These bits select the fast-access memory module in which the page addresses are to be stored and the alternate halves of the module as determined by bit position LM19. The bits contained on address lines LM20 through LM22 are gated onto map address lines L/MP0/3, L/MP0/4, L/MP0/5 through L/MP15/3, L/MP15/4, L/MP15/5. These bits select the bit positions to be written into. Signal MAPWXD is generated, gating the byte from bit positions D24 through D31 onto data lines MAPW15 through MAPW22. A one is then added to bits P20-22 to select the next bit position to be written into. During the second, third, and fourth passes through PH4, signals MMCW and MAPW are generated. During the second pass, byte 0 is stored in the selected map register, and during passes 3 and 4, bytes 1 and 2 are stored in their addressed register locations. After a byte has been stored in bit position 7 (that is, P20, P21, P22), a one is added to P19 to select the alternate half of the module to store the next four bytes.

If flip-flop SW2 is true, indicating load program control registers, an address is contained on only address lines LM15 through LM20. The address contained on LM15 through LM17 is gated onto program control address lines L/PB0/1 through L/PB0/2 and L/PB0/2 through L/PB3/2. These bits select the module in which the access codes are to be stored and the alternate halves of the module as determined by bit position LM17. The bits contained on address lines LM18 through LM20 are gated onto program control address lines L/PB0/3, L/PB0/4, L/B0/5 through L/PB3/3, L/PB3/4, and L/P3/5. These bits select the bit positions to be written into. Signal MAPWXD is generated, gating the byte from bit positions D24 through D31 onto data lines MAPW15 through MAPW22. The inverse of this data is gated onto data lines W/PB0/15 through W/PB0/22, W/PB1/15 through W/PB1/22, W/PB2/15 through W/PB2/22, and W/PB3/15 through W/PB3/22. A one is then added to bit position P20 to select the next bit position to be written into. During the second, third, and fourth passes through PH4, signals MMCW, PCBW and clocks K/PB0 through K/PB3 are generated. During the second pass, the one's complement of byte 0 is stored in the addressed PB-register and, during passes 3 and 4, the one's complement of bytes 1 and 2 are stored in their addressed register locations. After a byte has been stored in bit position 7 (that is, P18, P19, P20), a one is added to P17 to select the alternate half of the module to store the next four bytes.

If flip-flop SW3 is true, indicating load write lock registers, an address is contained only on address lines LM15 through LM20. The address contained on LM15 through LM17 is gated onto address lines LB15 through LB17 and from LB15 through LB17 onto write lock address lines L/LK0/1 through L/LK3/1 and L/LK0/2 through L/LK3/2. These bits select the module in which the write locks are to be stored and the alternate halves of the module as determined by bit position LB17. The bit contained on address lines LM18 through LM20 are gated onto address lines LB18 through LB20 and then onto write lock register address lines L/LK0/3, L/LK0/4, L/LK0/5 through L/LK3/3, L/LK3/4, and L/LK3/5. These bits select the bit positions to be written into. Signal MAPWXD is generated, gating the byte from bit positions D24 through D31 onto data lines MAPW15 through MAPW22. The inverse of this data is gated onto data lines W/LK0/15 through L/LK0/22 and L/LK3/15 through L/LK3/22. A one is then added to bit position P20 to select the next bit position to be written into. During the second, third, and fourth passes through PH4, signals MMCW, LOCKW and clocks K/LK0 through K/LK3 are generated. During the second pass, the one's complement of byte 0 is stored in the addressed LK-register and, during passes 3 and 4, the one's complement of bytes 1 and 2 is stored in their addressed register locations. After a byte has been stored in bit position 7 (that is, P18, P19, P20) a one is added to P17 to select the alternate half of the module to store the next four bytes. When flip-flops P32 and P33 are both ones, indicating that three bytes have been stored, flip-flop PH5 is set, clock T8L is enabled, and flip-flop SW4 is reset.

During PH5 the last byte is gated onto data lines MAPW15 through MAPW22, signal MMCW is generated, and the appropriate write signal is determined by flip-flop SW1, SW2, or SW3. The fourth byte is then stored in the addressed register. The address contained in the P-register is gated onto the sum bus and, at the PH5 clock, is stored in the B-register. The D-register is cleared for the next word. A one is subtracted from the count in the E-register, indicating that a complete word has been loaded. A one is forced into flip-flop CS31 for use during PH6. The instruction sequences to PH6, and clock T6L is enabled.

During PH6, the image address contained in the A-register and the one contained in flip-flop CS31 are gated into the adder, and an addition operation is performed to select the next word. The result is gated onto the sum bus and, at the PH6 clock, bits S15 through S31 are clocked into flip-flops P15 through P31. The address is then gated onto the LM and LB lines to address the next word. The modified image address is also clocked back into the A-register. A memory request is generated for the next word. Flip-flop ARQ is set, inhibiting transmission of another clock until an address release signal is received. If the E-register has not been counted down to zero and if an interrupt request occurs, set flip-flop SW4 to inhibit branching during PH7. The instruction sequences to PH7, and clock T6L is enabled.

The modified control start address stored in the B-register during PH5 is gated onto the sum bus and, at the PH7 clock, is clocked into flip-flops P15 through P17. Flip-flop DRQ is set, inhibiting transmission of another clock until the data signal is received from memory. If flip-flop SW4 is not set, and if the E-register is not all zeros, the instruction branches to PH4. Phases PH4 through PH7 are then repeated until flip-flop SW4 is set or until the count in the E-register reaches zero, indicating that all the words have been loaded. Flip-flop PH8 is then set, and clock T8L is enabled. When the next data word is available from core memory, signal CXMB is generated, gating the word into the C-register.

The image address contained in the A-register is gated onto the sum bus and from the sum bus onto data lines RW0 through RW31. Write-byte signals RWB0 through RWB3 are generated, and the modified image address is stored in the even numbered private memory register. If flip-flop SW1 is true, a one is added to bits P19 through P22. A one is added to bits P17 through P20 if either flip-flop SW2 or SW3 is true. The control start address then points to the next module and the bit to be loaded. Flip-flop IEN is set to allow any interrupt requests to be recognized. Flip-flop MAPDIS is reset to reinstate the mapping function. The instruction sequences to PH9, and clock T6L is enabled.

The modified control start address is gated from the P-register onto the sum bus and, at the PH9 clock, is clocked into flip-flops B15 through B31. The modified count in bit positions E0 through E7 is clocked into flip-flops A0 through A7. A memory request is generated for the next instruction. Memory request MRQ/1 gates the address of the next instruction for the Q-register into the P-register, and it is immediately gated onto the LM and LB address lines. Flip-flop DRQ is set, inhibiting transmission of another clock until the data release signal is received from memory. A one is forced on address line LR31 to select the odd numbered private memory register. Flip-flop PH10 is set, and clock T8L is enabled.

The modified word count contained in bit positions A0 through A7 is gated onto bit positions S0 through S7 of the sum bus. The control start address contained in bit positions B15 through B31 is gated onto bit positions S15 through S31 of the sum bus. The entire word is then gated from the sum bus onto data lines RW0 through RW31. Write-byte signals RWB0 through RWB3 are generated, and the data word is stored in the odd numbered private memory register. If flip-flop SW4 is not set, signal ENDE is generated, flip-flop PRE1 is set, and clock T6L is enabled for the next instruction.

A sequence chart of the Move to Memory Control instruction is shown in table 3-122.

3-249 Wait (WAIT 2E, AE)

The WAIT instruction halts the sequential operation of the CPU until the computer operator manually moves the COMPUTE switch on the Processor Control Panel from RUN to IDLE and then back to RUN. The CPU then proceeds to execute the instruction that sequentially follows the WAIT instruction.

Input/output operations can be performed while the CPU is halted. If an interrupt is activated while the CPU is halted, the interrupt subroutine is carried out and, at the end of the subroutine, executes the instruction following the WAIT. It then continues its normal execution sequence.

The preparation sequence for the WAIT instruction is the same as that described under Preparation Sequence.

During PH1, MRQ/1 is enabled to read the next instruction following the WAIT, the contents of the Q-register are transferred to the P-register, and flip-flop HALT is set. The next instruction is clocked into the D-register at PH2 clock, and signal ENDE is enabled. The CPU cannot sequence to PRE1 since HALT is true. Instead, it sequences to PCP1, and then to PCP2. After the COMPUTE switch has been set to IDLE and then back to RUN, flip-flop HALT resets, and the CPU sequences to phases PCP3, PCP4, PCP5, and then to PRE1 to execute the instruction following the WAIT.

A sequence chart of the WAIT instruction is given in table 3-123.

3-250 Read Direct (RD 6C, EC)

The Read Direct instruction can be executed in either one of two modes: the internal control mode (table 3-124) or the external mode (table 3-125). When the RD instruction is executed in the internal control mode, the condition code bits CC1-CC4 are set to the status of the control panel sense switches KSS1-KSS4.

Table 3-122. Move to Memory Control, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PRE1 T6L	P15-P31 $\nrightarrow$ Q15-Q31	QXP = PRE1 NANLZ + ...	Next instruction address
	Clear A-, B-, and E-registers	AX/1 = PRE1 + ...	
		BX/1 = PRE1 NINTRAPF + ...	
		EX/1 = PRE1 + ...	
	D12-D14 $\longrightarrow$ LR29-LR31	LRXD = OXC	No function
	Generate signal INDX	INDX = (C12 + C13 + C14) (C3 + C4 + C5)	No function. Mechanization convenience
	Set flip-flop IX	S/IX = INDX PRE1	No function. Mechanization convenience
	RR0-RR31 $\nrightarrow$ A0-A31	AXRR = FAW INDX PRE1 + ...	No function. Mechanization convenience
	Disable RW lines	RWDIS = PRE1	
	Set flip-flop PRE2	S/PRE2 = NPREIM PRE1 N(S/INTRAPF)	
	Enable clock T4RL	T4RL = PREP + ...	
PRE2 T4RL	D0-D31 + A0-A31 $\longrightarrow$ S0-S31	SXADD = PRE2 NIA NSDIS + ...	No function. Mechanization convenience
	S15-S31 $\nrightarrow$ P15-P31	PXS = PRE2 NIA + ...	No function. Mechanization convenience
	Reset flip-flop IX	R/IX = PRE2 NIA	No function. Mechanization convenience
	Clear A- and D-registers	AX/1 = PRE2 NIA + ...	
		DX/1 = PRE2 NIA + ...	
	If bit D12 is a one, set flip-flop SW1	S/SW1 = FUMMC NSW4 NANLZ PRE2 D12 + ...	Indicates load map function
	If bit D13 is a one, set flip-flop SW2	S/SW2 = FUMMC NSW4 NANLZ PRE2 D13 + ...	Indicates load program control registers
	If bit D14 is a one, set flip-flop SW3	S/SW3 = FUMMC NSW4 NANLZ PRE2 D14 + ...	Indicates load write lock register
	Set flip-flop SW4	S/SW4 = FUMMC PRE2 + ...	Inhibits changing of flip-flops SW1, SW2 or SW3 if instruction is indirectly addressed

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PRE2 T4RL (Cont.)	Set flip-flop PH1	S/PH1 = (S/PH1/1) NCLEAR NBR	
		(S/PH1/1) = NPREDO (PRE2 NIA) + ...	
	Enable clock T4RL	T4RL = PREP + ...	
	Set flip-flop EXU	S/EXU = (S/PH1/1) NCLEAR	
PH1 T4RL	RR0-RR31 ↛ A0-A31	AXRR = FUMMC PH2 + ...	Even numbered private memory register. Contains image address. Points to memory location containing first word to be read out
	Force a one onto address line LR31	R/NLR31/2 = LR31/2 = FUMMC PH1 + ...	
	Clear P32 and P33	PX/1 = FUMMC PH1 + ...	Clear P32 and P33 for byte 0
	Set flip-flop PH2	S/PH2 = PH1 NBR + ...	
	Enable clock T4RL	T4RL = FUMMC PH1 + ...	
PH2 T4RL	A0-A31 → S0-S31	SXA = FUMMC PH2 + ...	
	S15-S31 ↛ P15-P31	PXS = FUMMC PH2 + ...	Image address for first word from core memory
	P15-P31 → LM15-LB31		Address for first word
	Generate memory request	MRQ = FUMMC PH2 + ...	Memory request for first word
	Set flip-flop ARQ	S/ARQ = FUMMC PH2 + ...	Inhibits transmission of another clock until address release signal is received
	RR0-RR31 ↛ A0-A31	AXRR = FUMMC PH2 + ...	Data from odd numbered private memory register. Contains count and control start addresses
	Set flip-flop PH3	S/PH3 = PH2 NBR + ...	
	Enable clock T4RL	T4RL = FUMMC PH2 + ...	

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T4RL	A0-A31 → S0-S31	SXA = FUMMC PH3 + ...	
	S0-S7 ⇸ E0-E7	EXS = FUMMC PH3 NPRE1 + ...	Word count stored in E-register
	S15-S31 ⇸ P15-P31	PXS = FUMMC PH3 + ...	Control start address clocked into P-register
	MB0-MB31 → C0-C31	CXMB = DGC	First data word
	RR0-RR31 ⇸ A0-A31	AXRR = FUMMC PH3 + ...	Image address clocked back into A-register from even numbered private memory register
	Set flip-flop DRQ	S/DRQ = FUMMC PH3 + ...	Inhibits transmission of another clock until data signal is received
	Set flip-flop PH4	S/PH4 = PH3 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH4 T6L	If bits P32 and P33 are 0's, generate signal DXCR24 and clock C0-C7 ⇸ D24-D31	DXCR24 = NP32 NP33 DXCBP + ...	True first pass through PH4. Downward align byte 0
	Reset flip-flop SW4	R/SW4 = FUMMC PH4	Reset during first pass
	If NP32 and NP33, generate signal DXCR16 and clock C8-C15 ⇸ D24-D31	DXCR16 = NP32 P33 DXCBP + ...	True second pass through PH4. Downward align byte 1
	If P32 and NP33, generate signal DXCR8 and clock C16-C23 ⇸ D24-D31	DXCR8 = P32 NP33 DXCBP + ...	True third pass through PH4. Downward align byte 2
	If P32 and P33, generate signal DXC and clock C24-C31 ⇸ D24-D31	DXC = P32 P33 DXCBP + ...	True fourth pass through PH4. Transfer byte 3
		DXCBP = FUMMC PH4 + ...	
	If flip-flop SW1 is true, gate address P15-P22 → LM15-LM22		
	LM15-LM19 → L/MP0/1 - L/MP15/1 and L/MP0/2- L/MP15/2		LM15 through LM18 select module in which page addresses are to be stored. LM19 selects alternate halves of module

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	LM20-LM22 → L/MP0/3, L/MP0/4, LMP0/5-L/MP15/3, L/MP15/4, L/MP15/5		LM20 through LM22 select bit positions to be written into
	D24-D31 → MAPW15-MAPW22	MAPWXD = FUMMC EXU + ...	Byte gated onto MAP data lines
	Add 1 to P32, P33	PCTP1 = FUMMC PH4 (NP32 + NP33) + ...	Selects next byte to be downward aligned
		PA33 = FUMMC PH4 (NP32 + NP33) + ...	
	Set flip-flop MAPDIS	S/MAPDIS = FUMMC BRPH4 + ...	Disable memory map function
	Repeat PH4 until P32 and P33 are both ones	S/PH4 = BRPH4 = FUMMC PH4 (NP32 + NP33) + ...	
	During passes 2, 3, and 4, generate signal MMCW and clock bytes 0, 1, and 2 into MAP registers	MMCW = FUMMC PH4 (P32 + P33) + ...	
		MAPW = MMCW SW1 + ...	SW1 => load map page registers
		K/MP0-K/MP15 = MAPW CK-24U08	Clocks bytes into selected bits of MAP modules
	Add one to P19-P22	PA22 = N(FUMMC PH4 NSW1)	Select next bit position to be written into
		PCTP4 = FUMMC PH4 NSW4	
	If flip-flop SW2 is true, gate address P15-P20 → LM15-LM20		SW2 => load program control registers
	LM15-LM17 → L/PB0/1-L/PB3/1 and L/PB0/2-L/PB3/2		LM15 and LM16 select fast-access memory module in which program control bits are to be stored. LM17 selects alternate halves of module
	LM18-LM20 → L/PB0/3, L/PB0/4, L/PB0/5-L/PB3/3, L/PB3/4, L/PB3/5		LM18 through LM22 select bit positions to be written into
	D24-D31 → MAPW15-MAPW22	MAPWXD = FUMMC EXU	Selected byte gated onto MAP lines
	NMAPW15-NMAPW22 → W/PB0/15-W/PB0/22 through W/PB3/15-W/PB3/22		One's complement of data gated from MAP data lines onto PB data lines

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	Add a one to P32, P33	PCTP1 = FUMMC PH4 (NP32 + NP33) + ...	Selects next byte to be downward aligned
		PA33 = FUMMC PH4 (NP32 + NP33)	
	Add a one to P17-P20	PCTP4/1 = FUMMC NSW4 NANLZ PH4 + ...	Selects next bit position to be written into. PA22 = 0
		PCTP5 = P19 P20 NPA22 PCTP4	
	Repeat PH4 until P32 and P33 are both ones	S/PH4 = BRPH4 = FUMMC PH4 (NP32 + NP33) + ...	
	During passes 2, 3, and 4, generate signal MMCW and clock bytes 0, 1, and 2 into PCB-registers	MMCW = FUMMC PH4 (P32 + P33)	Clock bytes into selected bits of PCB-register
		PCBW = MMCW SW2	
		K/PB0-K/PB3 = PCBW CK-24U06	
	If flip-flop SW3 is true, gate address		SW3 => load write lock registers
	P15-P20 → LM15-LM20		
	LM15-LM17 → LB15-LB17		LB15 and LB16 select module in which write locks are to be stored. LB17 selects alternate halves of module
	LB15-LB17 → L/LK0/1 - L/LK3/1 and L/LK0/2-L/LK3/2		
	LB18-LB20 → L/LK0/3, L/LK0/4, L/LK0/5-L/LK3/3, L/LK3/4, L/LK3/5		LB18 through LB20 select bit positions to be written into
	D24-D31 → MAPW15-MAPW22	MAPWXD = FUMMC EXU	Selected byte gated onto MAP lines
	NMAP15-NMAP22 → W/LK0/15-W/LK0/22 through W/LK3/15-W/LK3/22		One's complement of byte gated from MAP data lines onto write lock data lines
	Add a one to P32, P33	PCTP1 = FUMMC PH4 (NP32 + NP33) + ...	Selects next byte to be downward aligned
		PA33 = FUMMC PH4 (NP32 + NP33)	

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	Add a one to P17-P20	PCTP4/1 = FUMMC NSW4 NANLZ PH4	PA22 = 0
	Repeat PH4 until P32 and P33 are both ones	PCTP5 = P19 P20 NPA22 PCTP4	
		S/PH4 = BRPH4 = FUMMC PH4 (NP32 + NP33) + ...	
	During passes 2, 3, and 4, generate signal MMCW and clock bytes 0, 1, and 2 into write lock registers	MMCW = FUMMC PH4 (P32 + P33)	
		LOCKW = MMCW SW3	
		K/LK0-K/LK2 = LOCKW CK-24U04	Clock bytes into selected bits of write lock registers
	Reset flip-flop SW4	R/SW4 = FUMMC PH4 + ...	Reset during first pass through PH4
	When flip-flops P32 and P33 are both ones, set flip-flop PH5	S/PH5 = PH4 NBR + ...	
	Enable clock T8L	R/NT8L = S/T8L = FUMMC PH4	
PH5 T8L	P15-P31 → S15-S31	SXP = FUMMC PH5 + ...	Store control start address
	S15-S31 ↛ B15-B31	BXS = FUMMC PH5 + ...	
	D24-D31 → MAPW15-MAPW22	MAPWXD = FUMMC EXU	
	Generate signal MMCW	MMCW = FUMMC PH5	
	Store byte 3 into register as determined by SW1, SW2, or SW3		Last byte stored in addressed register
	Reset flip-flop MAPDIS	R/MAPDIS = FUMMC PH5 + ...	
	Clear D-register	DX/1 = FUMMC PH5 + ...	
	Force a one into flip-flop CS31	CSX1/8 = FUMMC PH5 + ...	For use during PH6
	Subtract one from count in E-register	MCTE1 = FUMMC PH5 + ...	Indicates one word loaded
	Set flip-flop PH6	S/PH6 = PH5 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont)

Phase	Function Performed	Signals Involved	Comments
PH6 T6L	A0-A31 + CS0-CS31 → S0-S31	SXADD = FUMMC PH6 + ...	Add one to image address to select next word
	S15-S31 -/→ P15-P31	PXS = FUMMC PH6 + ...	
	S15-P31 → LM15-LB31		Address of next word from image address
	S0-S31 -/→ A0-A31	AXS = FUMMC PH6 + ...	Image address clocked back into A-register
	Generate a memory request for next word	MRQ = FUMMC PH6 NEZ + ...	
	Set flip-flop ARQ	S/ARQ = FUMMC PH6 + ...	Inhibits transmission of another clock until address release signal is received
	If the E-register has not been counted down to zero and an interrupt occurs set flip-flop SW4	S/SW4 = FUMMC NEZ PH6 INT + ...	Inhibits branching to PH4 during PH7
	Set flip-flop PH7	S/PH7 = PH6 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH7 T6L	B0-B31 → S0-S31	SXB = FUMMC PH7 + ...	
	S15-S31 -/→ P15-P31	PXS = FUMMC PH7 + ...	Control start addressing information
	If flip-flop SW4 is not set, and the count in the E-register is not all zeros, branch to PH4	S/PH4 = BRPH4 = FUMMC PH7 NSW4 NANLZ NEZ + ...	
	Disable MAP	S/MAPDIS = FUMMC BRPH4	
	Repeat phases PH4 through PH7 until an interrupt occurs (that is, flip-flop SW4 is set), or the count in the E-register reaches zero, indicating that all the words have been loaded. Flip-flop PH8 is then set	S/PH8 = PH7 NBR + ...	
	Enable clock T8L	R/T8L = S/T8L = FUMMC PH7 + ...	

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T6L (Cont.)	Set flip-flop DRQ	S/DRQ = FUMMC PH7 + ...	Inhibits transmission of another clock until data release signal is received for memory request generated during PH6
	MB0-MB31 → C0-C31	CXMB = DGC	When data release signal is received, next word is gated into C-register
PH8 T8L	A0-A31 → S0-S31	SXA = FUMMC PH8 + ...	
	S0-S31 → RW0-RW31	RWXS = FUMMC PH8 + ...	Image address
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = FUMMC PH8 + ...	Stores modified image address in even numbered private memory register
		RW = FUMMC PH8 + ...	
	Add a one to P19 through P22 if SW1 is true or to P17 through P20 if SW2 or SW3 is true	PA22 = N(FUMMC PH8 NSW1)	Control start address points to next module and bit to be loaded
		PCTP4/1 = FUMMC PH8	
	Set flip-flop IEN	S/IEN = FUMMC PH8 + ...	Enable interrupt
	Set flip-flop PH9	S/PH9 = PH8 NBR + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH9 T6L	P15-P31 → S15-S31	SXP = FUMMC PH9 + ...	
	S15-S31 ↛ B15-B31	BXS = FUMMC PH9 + ...	Modified control start address stored in B-register
	Set flip-flop IEN	S/IEN = FUMMC PH9 + ...	Allows interrupt to be recognized
	Force a one on address line LR31/2	R/NLR31/2 = LR31/2 = FUMMC PH9 + ...	
	E0-E7 ↛ A0-A7	AXE = FUMMC PH9 + ...	Count stored in A-register
	Generate memory request for next instruction	MRQ/1 = FUMMC PH9 + ...	

Mnemonic: MMC (6F, EF)

(Continued)

Table 3-122. Move to Memory Control, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)	Q15-Q31 ↛ P15-P31	PXQ = MRQ/1 + ...	Address of next instruction
	P15-P31 → LM15-LB31		
	Set flip-flop DRQ	S/DRQ = FUMMC PH9 + ...	Inhibits transmission of another clock until data signal is received
	Set flip-flop PH10	S/PH10 = PH9 NBR + ...	
	Enable clock T8L	R/NT8L = S/T8L = FUMMC PH9 + ...	
PH10 T8L	A0-A7 → S0-S7	SXA = FUMMC PH10 + ...	Modified word count gated onto sum bus
	B15-B31 → S15-S31	SXB = FUMMC PH10 + ...	Control start address gated onto sum bus
	S0-S31 → RW0-RW31	RWXS = FUMMC PH10 + ...	
	Generate write-byte signals RWB0-RWB3	RWB0-RWB3 = FUMMC PH10 + ...	Store count and control start address in odd numbered private memory register
	Generate signal ENDE	ENDE = (FUMMC NSW4 NANLZ) PH10 + ...	
	Set flip-flop PRE1	S/PRF1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: MMC (6F, EF)

Table 3-123. Wait, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	Q15-Q31 -/→ P15-P31	PXQ = MRQ/1 + ...	
	Enable memory request	MRQ/1 = FUWAIT PH1 + ...	To read next instruction
		S/DRQ = FUWAIT PH1 + ...	
	1 -/→ HALT	S/HALT = FUWAIT PH1 + ...	
PH2 T6L	End of Wait	ENDE = FUWAIT PH2 + ...	
	C0-C31 -/→ D0-D31	DXC = ENDE + ...	Next instruction to D-register
	Suppress PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF) + ...	
	Go to PCP1	S/PCP1 = (S/PCP1) NPCP3 + ...	
		(S/PCP1) = ENDE HALT/1 N(S/INTRAPF) NFUEXU	
PCP1	Go to PCP2	S/PCP2 = (S/PCP2) NPCP3	
		(S/PCP2) = PCP1 NCLEAR	
PCP2	Halt in PCP2 until COMPUTE switch is set to IDLE and then back to RUN	S/PCP3 = (S/PCP3) NPCP3	
		(S/PCP3) = PCP2 NHALT NCLEAR KAS/1 KAS/2	
		R/HALT = PCP2 NKAS/B + ...	
PCP3	Go to PCP4	S/PCP4 = PCP3 (NPCP7 NENDE)	
PCP4	Go to PCP5	S/PCP5 = PCP4 (NPCP7 NENDE)	
PCP5	Go to PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF) + ...	
		ENDE = PCP5 NKIDLE + ...	

Mnemonic: WAIT (2E, AE)

Table 3-124. Read Direct Internal Control Mode, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	Enable memory request for next instruction	MRQ/1 = FARWD PH1 + ...	
	Q15-Q31 -/→ P15-P31	PXQ = MRQ/1 + ...	Next instruction address
	Go to PH2	S/PH2 = PH1 NBR N(...) + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Go to PH3	S/PH3 = PH2 NBR N(...) + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	1 -/→ SW1	S/SW1 = (FARWD PH3) B1619Z NSW1 + ...	Differentiates between two PH3 intervals
		R/SW1 = FADIO + ...	
	Sense switches -/→ CC1-CC4	BRPH3 = FARWD PH3 NSW1 + ...	
		R/CC1 = FARWD SW1 + ...	
		S/CC1 = FARWD SW1 B1619Z KSS1 + ...	
		S/CC2 = FARWD SW1 B1619Z KSS2 + ...	
		S/CC3 = FARWD SW1 B1619Z KSS3 + ...	
		S/CC4 = FARWD SW1 B1619Z KSS4 + ...	
	If NRZ B27, MFL0-MFL7 -/→ D24-D31	DXPARITY = (FARWD B1619Z NO7 NRZ B27 PH3) + ...	Memory fault indicators
		S/D24-D31 = (MFL0-MFL7) DXPARITY	
	If NRZ B27, 0's -/→ MFL0-MFL7	PFSR = (FARWD SW1 B1619Z NRZ B27) + ...	Reset memory fault indicators
	Go to PH4	S/PH4 = PH3 NBR + ...	
		R/SW1 = FADIO + ...	
PH4 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	End RD instruction if DRQ = 1	ENDE = FARWD PH4 DRQ + ...	
		S/DRQ = FARWD PH4 B1619Z + ...	

Mnemonic: RD (6C, EC)

(Continued)

Table 3-124. Read Direct Internal Control Mode, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	If R28-R31 ≠ 0, D0-D31 ⟶ S0-S31	SXD = OU6 OLC PH4 NRZ + ...	Status of memory fault indications into private memory

	S0-S31 ⟶ RW0-RW31	RW = OU6 OLC PH4 NRZ + ...
		BRPH4 = FARWD PH4 NDRQ + ...	

Mnemonic: RD (6C, EC)

Table 3-125. Read Direct External Mode, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	Enable memory request for next instruction	MRQ/1 = FARWD PH1 + ...	
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + ...	Address of next instruction to P
	P15-P31 ⟶ S15-S31	SXP = FADIO PH1 + ...	
	S0-S31 —/→ B0-B31	BXS = FADIO PH1 + ...	Effective address to B
	Set SW3	S/SW3 = FADIO PH1 + ...	For use in PH2
	Go to PH2	S/PH2 = PH1 NBR N(...) + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	B16-B31 ⟶ DIO32-DIO47	DIOXB = FARWD SW3 + ...	Device address to direct input/output address lines
	Go to PH3	S/PH3 = PH2 NBR N(...) + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set SW4	S/SW4 = FARWD PH3 NSW1 + ...	
		R/SW4 = FADIO SW1 + ...	
	Repeat PH3 until SW1 = 1	BRPH3 = FARWD PH3 NSW1 + ...	SW1 => device response received
		S/PH3 = BRPH3 + ...	
		S/PH4 = PH3 NBR + ...	
	Set SW1 when device response received	S/SW1 = FARWD PH3 DIO49 NSW1 + ...	DIO49 => device response
		R/SW1 = FADIO SW1 + ...	
	DIO0-DIO31 —/→ D0-D31	DXDIO = FARWD SW1 NB1619Z + ...	Input data from device
		S/D0-D31 = (DIO0-DIO31) DXDIO	
	Set CC3-CC4 from device	S/CC3 = FARWD SW1 NB1619Z DIO51 + ...	Condition code information from device
		S/CC4 = FARWD SW1 NB1619Z DIO52 + ...	

Mnemonic: RD (6C, EC)

(Continued)

Table 3-125. Read Direct External Mode, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (Cont.)	Strobe data	/DIO48/ = DIO48X1 DIO48X1 = FADIO SW1 SW4 IARWD + ...	Strobe signal to device
PH4 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Remain in PH4 until device response signal drops and sets data request	BRPH4 = FARWD PH4 NDRQ + ...	
		S/DRQ = FARWD PH4 NDIO49 + ...	Final PH4 sequence
	D0-D31 → S0-S31	SXD = OU6 OLC PH4 NRZ + ...	Read input data into private memory register unless address is zero
	S0-S31 → RW0-RW31	RW = OU6 OLC PH4 NRZ + ...	
	End of RD instruction sequence	ENDE = FARWD PH4 DRQ + ...	
		R/SW3 = CLEAR + ...	

Mnemonic: RD (6C, EC)

If the R-field of the RD instruction is not zero, the states of the memory fault indicators, MFL0-MFL7, are transferred to the addressed private memory register R, bit positions 24 through 31. Bit positions 0 through 23 of the private memory register are cleared to zeros, and the memory fault indicators are reset.

When the RD instruction is executed in the external mode, an external device is addressed. The RD instruction then waits until the device responds with input data to the CPU and then loads the data into the addressed private memory register.

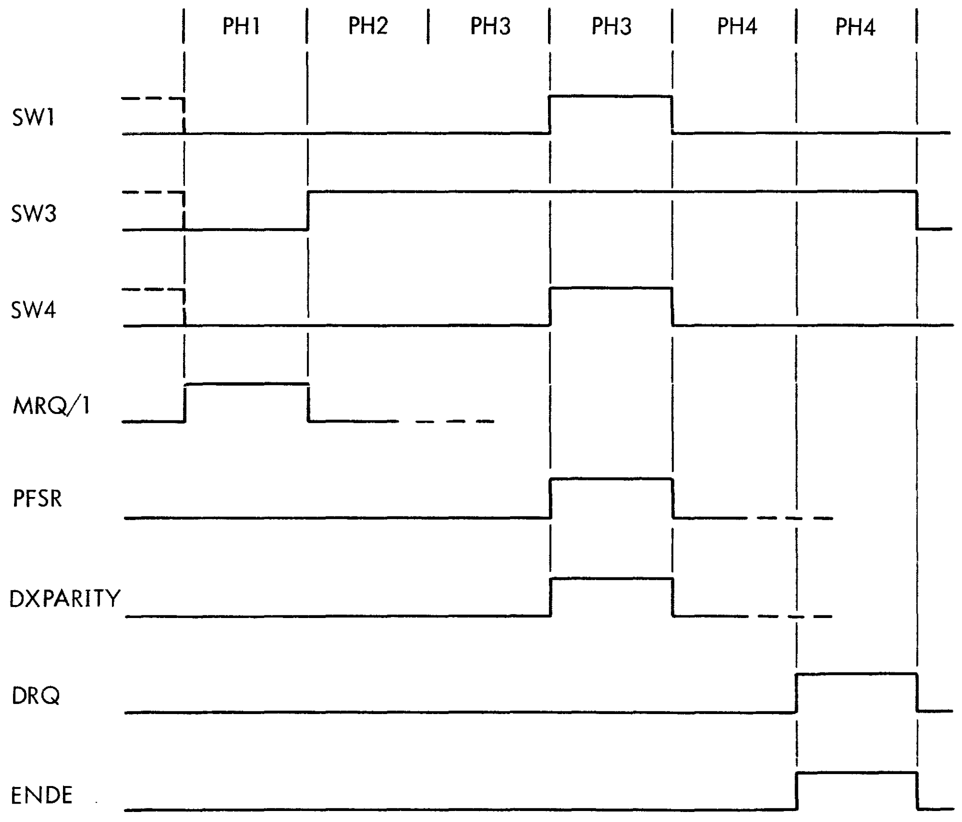
Timing for the RD instruction internal control mode is shown in figure 3-193; timing for the RD instruction external mode is shown in figure 3-194.

Fifty-five separate input/output lines associated with the read direct instruction are available. The 32 data lines DIO0 through DIO31 are bidirectional lines used as input lines for the read direct instruction. The input data is accepted from the data lines by the D-register. (See figure 3-195.) The 16 address lines, /DIO32/ through /DIO47/, are used to address the input device. (See figure 3-196.)

There are 7 direct input control lines, /DIO48/ through /DIO54/. (See figure 3-197.) /DIO48/ is a strobe signal to the addressed device. /DIO49/ is a response signal from the device. Signal /DIO50/ is false for read-direct operation. Signals /DIO51/ and /DIO52/ are input lines from the device and can be used for miscellaneous purposes, for example, for temperature out of limits in some applications, or for A/D off scale readings. The status of lines /DIO51/ and /DIO52/ is transferred into condition code bits CC3 and CC4. Line /DIO53/ is used for the resetting of all inputs when the I/O RESET or SYS RESET switch-indicators are pressed or when power is first applied to the computer system. A 1-MHz signal to all devices associated with the read-direct operations is provided on line/DIO54/.

The RD instruction can be indirectly addressed, indexed, or both. The normal preparation sequence, as well as the sequencing for all modes and conditions of the RD instruction, is described under Preparation Sequence.

The effective address in P is transferred to the sum bus and is clocked into the B-register. The next instruction address in Q is clocked into P; SW3 and PH2 are set. The address of the device is taken from the B-register (B16-B31) and is placed on the input/outout address lines.



901060A. 3641

Figure 3-193. Read Direct Internal Control Mode, Timing Diagram

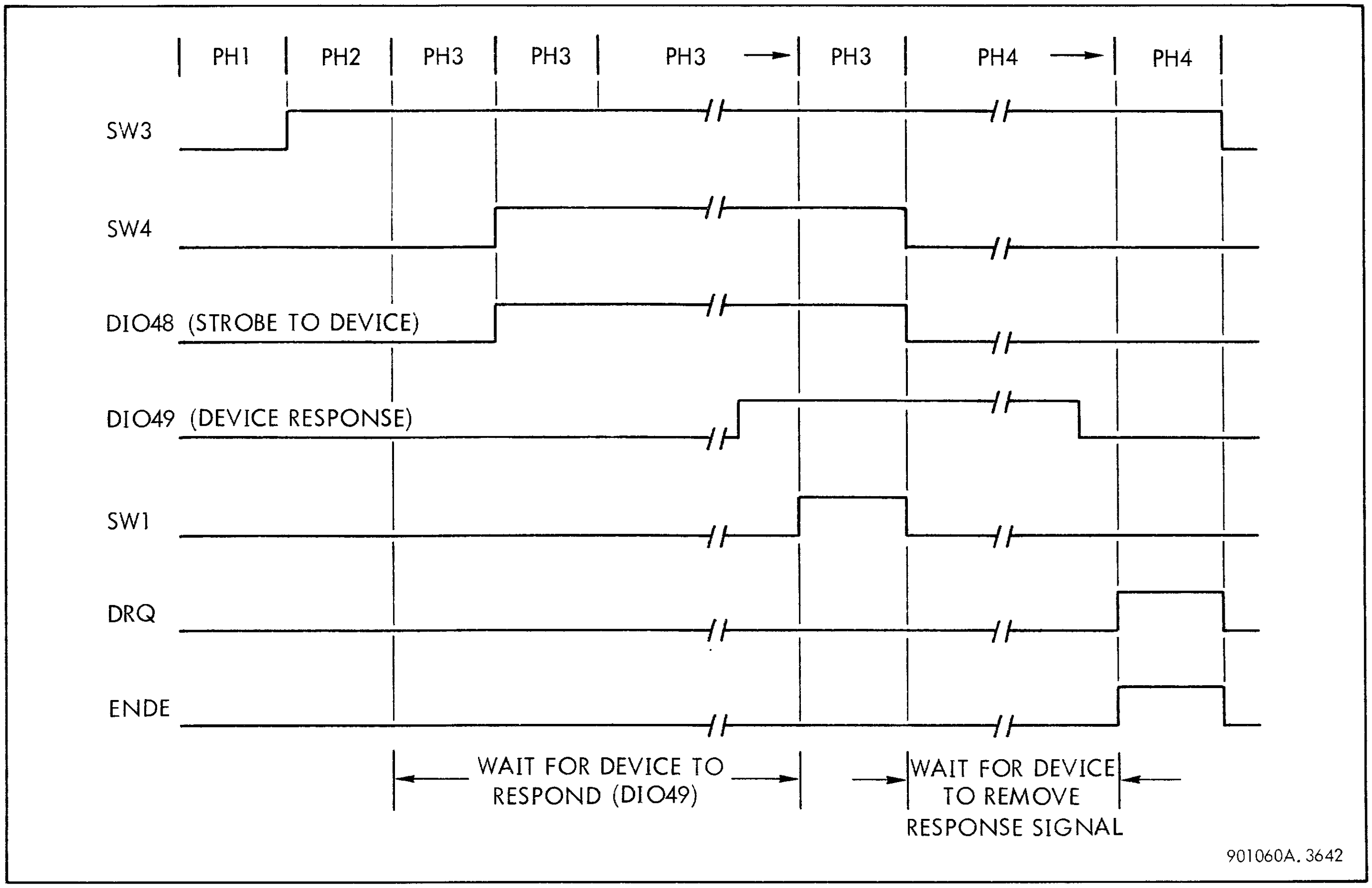


Figure 3-194. Read Direct External Mode, Timing Diagram

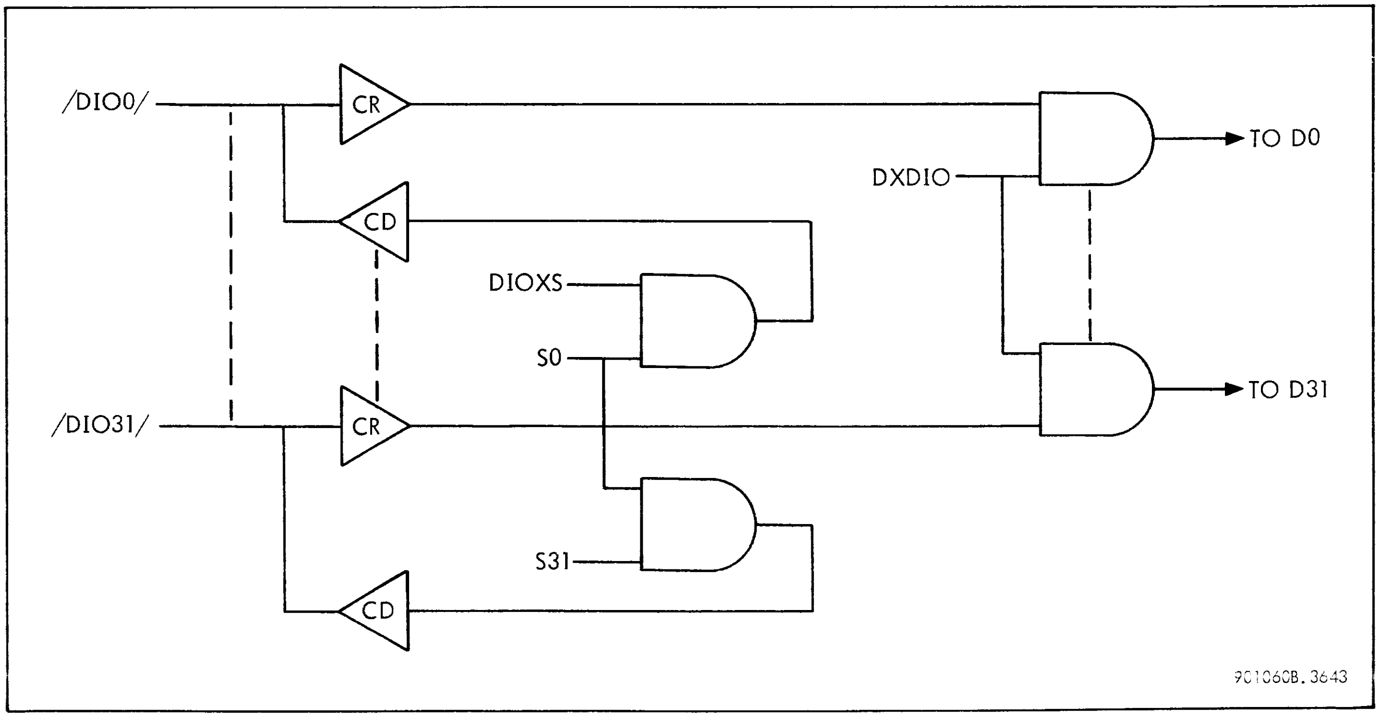


Figure 3-195. Read/Write Direct Input/Output Lines DIO0-DIO31

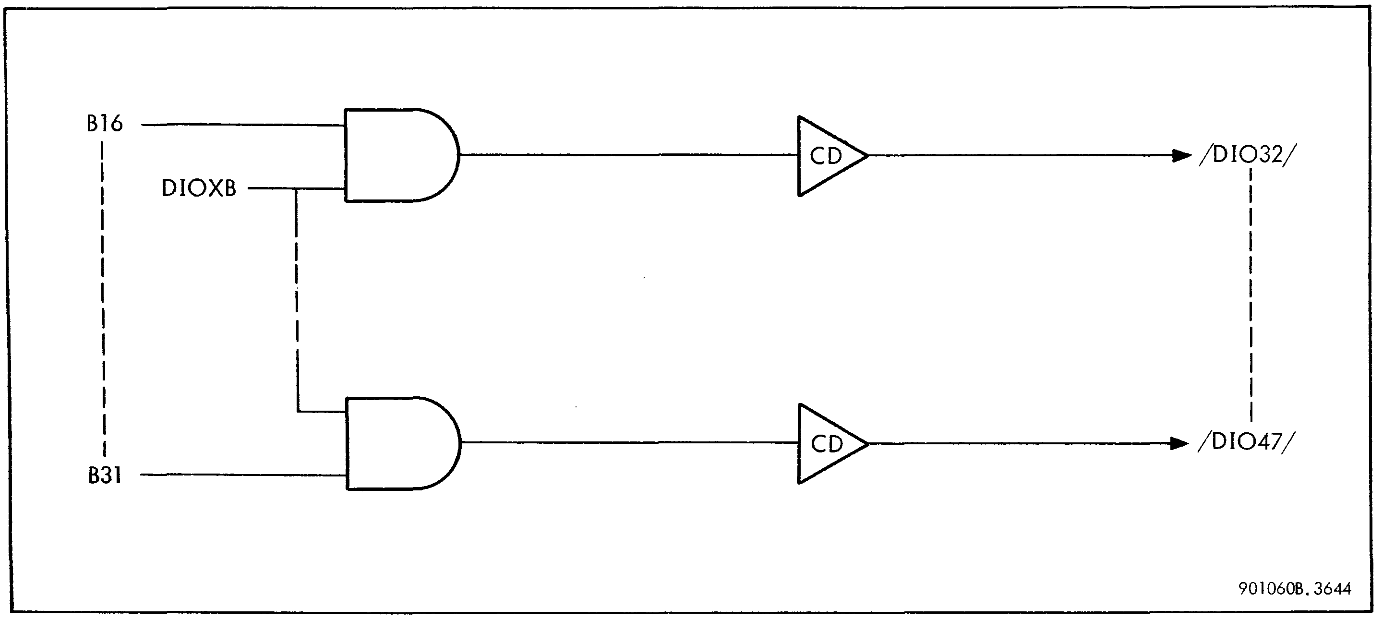


Figure 3-196. Read/Write Direct Address Lines /DIO32/-/DIO47/

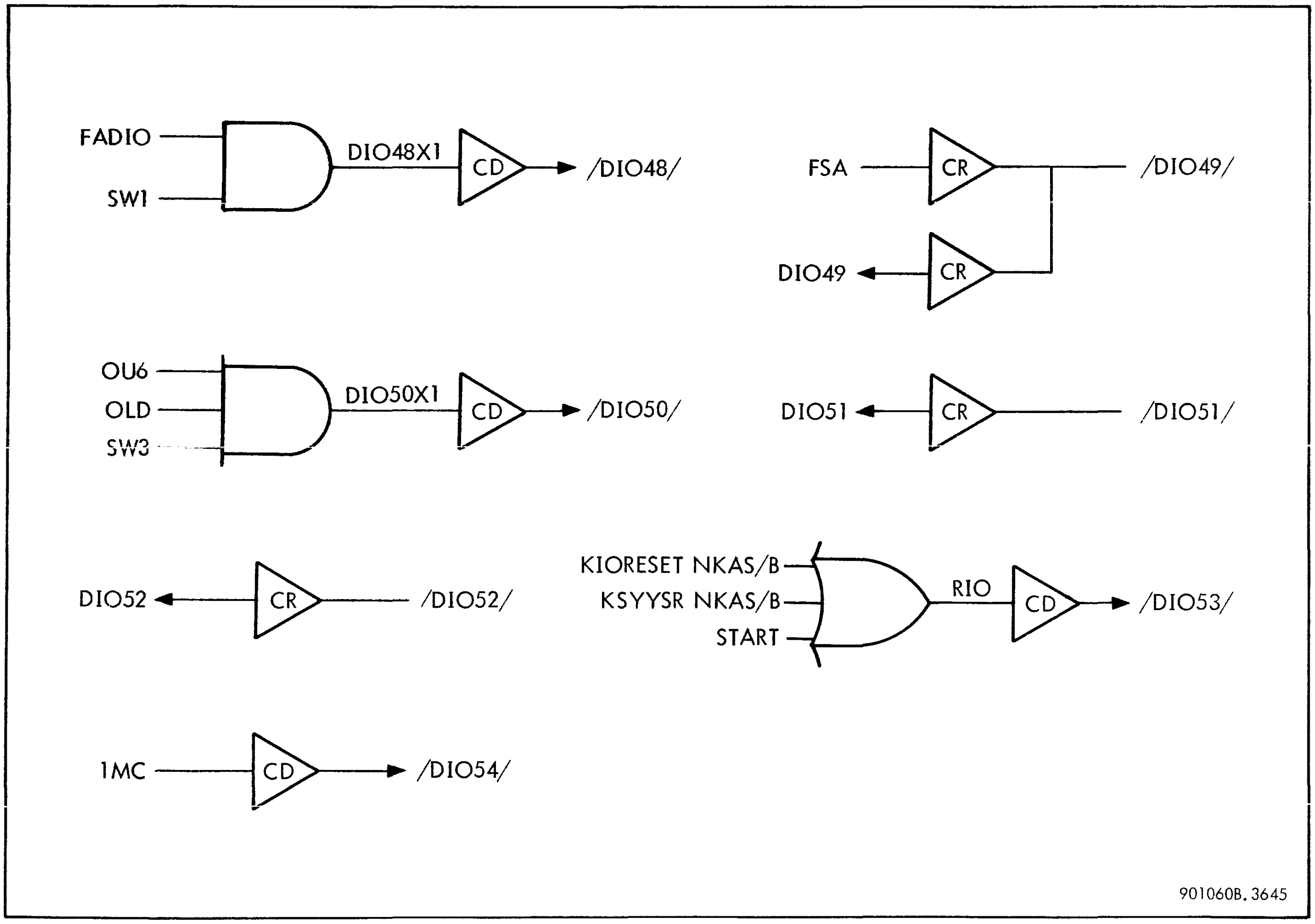


Figure 3-197. Read/Write Direct Control Lines /DIO48/-/DIO54/

The PH3 sequence of the RD instruction operates differently in the internal control mode (B1619Z) and in the external mode (NB1619Z). Each mode will be discussed separately in the following paragraphs.

3-251 PH3 INTERNAL MODE (B1619Z). The CPU leaves PH2 and enters PH3 for two clock periods. SW1 differentiates between the two PH3 intervals by setting at the end of the first PH3 interval and resetting at the end of the second PH3 interval. The sense switches are strobed into CC1 through CC4 when SW1 is high. If the R-field of the RD instruction word is not zero (NRZ), and if B27 is a one, the memory fault indicators are strobed into D24 through D31, and the memory fault indicators are reset.

3-252 PH3 EXTERNAL MODE (NB1619Z). The CPU leaves PH2 and enters PH3 to wait for the device response signal /DIO49/. Flip-flop SW4 is set after the first PH3 sequence. When SW4 is true, the strobe signal /DIO48/ is sent to the device. The CPU is forced to remain in PH3 by the signal BRPH3 until the device responds with the signal /DIO49/. When /DIO49/ is received, SW1 is set, and the CPU prepares to sequence to PH4. CC3 and CC4 are set if signals /DIO51/ and /DIO52/ are received from the input device. Flip-flop SW1 true marks the final PH3 sequence. At the clock of this final sequence in PH3, the input data from the device is read into the D-register. The CPU remains in PH4 until the device response signal /DIO49/ drops. During PH4, if the R-field of the RD instruction was not zero (NRZ), the contents of D are placed on the sum bus and are then clocked into the addressed private memory register. When the device response signal drops, data release flip-flop DRQ is set, marking the final PH4 sequence.

A sequence chart of the Read Direct Internal Control Mode is shown in table 3-124. The External Mode Phase Sequence is shown in table 3-125.

3-253 Write Direct (WD 6D, ED)

Fifty-five separate input/output lines are associated with the write direct instruction. The 32 data lines, /DIO0/ through /DIO31/, are bidirectional lines used for data output during the write direct instruction. The output data is placed on the data lines from the sum bus.

Sixteen address lines, /DIO32/ through /DIO47/, are used to address the output device. There are seven control lines, /DIO48/ through /DIO54/. Line /DIO48/ carries a strobe that informs the addressed device that data is present on the data lines. Line /DIO49/ carries a response signal from the addressed device. The signal on line /DIO50/ is always true during write direct operations. Lines /DIO51/ and /DIO52/ are input lines from the device that can be used for miscellaneous purposes. The status of the signals on these lines is transferred to condition code flip-flops CC3 and CC4. Line /DIO53/ is used to reset all output functions when the I/O RESET or SYS RESET switch-indicators are pressed or when power is first applied to the computer system. A 1-MHz signal to all devices associated with the write direct operations is provided on line /DIO54/.

The WD instruction can be executed in one of three modes: internal control (table 3-126), interrupt control (table 3-127), or output (table 3-128). The mode of execution is determined by the contents of bit positions 16 through 19 of the effective word. If these four bits contain a hexadecimal zero, the internal control mode is specified; if they contain a hexadecimal one, the interrupt control mode is specified; if they contain any hexadecimal number 2 through F, the output mode is specified. These three modes of WD execution are described in the following paragraphs.

When the WD instruction is executed in the internal control mode (B1619Z), bits 25 through 31 of the effective word determine the functions to be performed. (See figure 3-198.) The contents of the effective word are transferred to the B-register during the execution of the instruction and remain in the B-register until the end of the instruction; thus, the signal B1619Z represents the internal control mode of operation. Bits 25 through 31 of the B-register represent the function to be executed.

The interrupt inhibits can be set (if B27 = 1) or reset (if B27 = 0) either individually or in any combination if a 1 bit is placed in bits 29, 30, or 31. Bit 29 addresses the counter interrupt inhibit flip-flop CI; bit 30 addresses the input/output interrupt inhibit flip-flop II, and bit 31 addresses the priority interrupt inhibit flip-flop EI. A 1 bit in bit positions 25 and 31 sets the ALARM indicator on the processor control panel, and a 1 bit in bit position 25 and a 0 bit in bit position 31 resets the ALARM indicator. The AUDIO ALARM can be toggled by the flip-flop MUSIC if bit positions 25 and 30 both contain a one. Whenever the ALARM indicator is set and the CPU is in the RUN state (KRUN/B), the PCP speaker emits a 1-kHz signal to the panel speaker.

When the WD instruction is executed in the interrupt control mode (B1619ONE), bits 21 through 23 of the effective word (which is transferred to the B-register) determine the interrupt action code, and bits 28 through 31 address the interrupt group 0 through F. (See figure 3-199.)

The individual interrupt levels of the addressed interrupt group are addressed by 1 bits contained in bit positions 16 through 31 of the private memory register addressed by the R-field of the WD instruction word. In all code conditions except code 4 above, zeros in the addressed private memory register do not affect the current status of the associated interrupt levels.

The WD instruction operates in the output mode if bits 16 through 19 of the effective word represent a hexadecimal 2 through F as shown in figure 3-200. Bits 20 through 31 of the effective word represent the address of the output device (such as a digital to analog converter or digital display) to which the output data is to be transferred. The output data is represented by the contents of the private memory register addressed by the R-field of the WD instruction word.

The write direct instruction in any of the three modes of operation can be indexed, indirectly addressed, or both.

Table 3-126. Write Direct Internal Control Mode, Phase Sequence

Phase	Function Performed	Signals Involved		Comments
PH1 T4RL	Enable memory request for next instruction	MRQ/1	= FARWD PH1 + ...	
	P16-P31 → S16-S31	SXP	= FADIO PH1 + ...	
		FADIO	= FARWD + FAIO + ...	
	S16-S31 ↛ B16-B31	BXS	= FADIO PH1 + ...	Contains mode of operation (zeros in the case of internal mode) and function to be performed
	0's ↛ B0-B31			
	1 ↛ SW3	S/SW3	= FADIO PH1 + ...	
	Go to PH2	S/PH2	= PH1 NBR N(...) + ...	
PH2 T6L	Enable T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
	1 → /DIO50/	DIO50X1	= OU6 OLD SW3 + ...	
	1 → WDINTL	WDINTL	= OU6 OLD PH2 B1619Z + ...	
	If (WDINTL B25 B31) = 1, 0 ↛ ALARM	S/ALARM	= WDINTL B25 B31	
		ALARM/L	= ALARM	Alarm light on PCP
	If (WDINTL B25 NB31) = 1, 1 ↛ ALARM	R/ALARM	= WDINTL B25 + RESET	
	If (WDINTL B25 B30) = 1, toggle MUSIC	S/MUSIC	= WDINTL B25 B30 NMUSIC	
		R/MUSIC	= WDINTL B25 B30	
		WDINTL	= OU6 OLD PH2 B1619Z	
	If (MUSIC NALARM) = 1, or (ALARM KRUN/B) = 1, 1 ↛ AUDIO	AUDIO	= MUSIC NALARM + (ALARM KRUN/B) 1KC	
	If B26 = 1, 1 → INHXWD	INHXWD	= OU6 OLD PH2 B1619Z B26	
	If (INHXWD B27 B29) = 1, 1 ↛ CIF	S/CIF	= INHXWD B27 B29 + ...	Inhibit counter interrupts
	If (INHXWD NB27 B29) = 1, 0 ↛ CIF	R/CIF	= INHXWD B29 + ...	Allow counter interrupts

Mnemonic: WD (6D, ED)

(Continued)

Table 3-126. Write Direct Internal Control Mode, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)	If (INHXWD B27 B30) = 1, 1 $\nrightarrow$ II	S/II = INHXWD B27 B30 + ...	Inhibit I/O interrupts
	If (INHXWD NB27 B30) = 1, 0 $\nrightarrow$ II	R/II = INHXWD B30 + ...	Allow I/O interrupts
	If (INHXWD B27 B31) = 1, 1 $\nrightarrow$ EI	S/EI = INHXWD B27 B31 + ...	Inhibit external interrupts
	If (INHXWD NB27 B31) = 1, 0 $\nrightarrow$ EI	R/EI = INHXWD B31 + ...	Allow external interrupts
	Go to PH3	S/PH3 = PH2 NBR + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If SW1 = 0, remain in PH3	BRPH3 = FARWD PH3 NSW1 + ...	
	1 → /DIO50/	DIO50X1 = OU6 OLD SW3 + ...	Identifies write direct
	1 $\nrightarrow$ SW1	S/SW1 = (FARWD PH3) B1619Z NSW1 + ...	
		R/SW1 = FADIO + ...	
	Read sense switches into condition codes	S/CC1 = (FARWD SW1) B1619Z KSS1+... S/CC2 = (FARWD SW1) B1619Z KSS2+... S/CC3 = (FARWD SW1) B1619Z KSS3+... S/CC4 = (FARWD SW1) B1619Z KSS4+...	
	If SW1 = 1, go to PH4	S/PH4 = PH3 NBR + ...	
PH4 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If DRQ = 0, remain in PH4 and set data request	BRPH4 = FARWD PH4 NDRQ + ...	
		S/DRQ = FARWD PH4 B1619Z + ...	
		R/SW3 = CLEAR + ...	
		CLEAR = ENDE + ...	
		ENDE = FARWD PH4 DRQ + ...	

Mnemonic: WD (6D, ED)

Table 3-127. Write Direct Interrupt Control Mode, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	Enable memory request for next instruction	MRQ/1 = FARWD PH1 + ...	
	P16-P31 $\longrightarrow$ S16-S31	SXP = FADIO PH1 + ...	Contains mode (one in the case of interrupt) function and group address
	S16-S31 $\not\longrightarrow$ B16-B31	BXS = FADIO PH1 + ...	
	0's $\not\longrightarrow$ B0-B15		
	R28-R31 $\longrightarrow$ LR28-LR31	Normal action	
	RR0-RR31 $\not\longrightarrow$ A0-A31	AX/1 = FARWD PH1 + ...	Interrupt selection bits
	1 $\not\longrightarrow$ SW3	S/SW3 = FADIO PH1 + ...	
	Go to PH2	S/PH2 = PH1 NBR + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	B16-B31 $\longrightarrow$ /DIO32/-/DIO47/	DIOXB = FARWD SW3 + ...	Mode function and group address to address lines
	If $R \neq 0$, A0-A31 $\longrightarrow$ S0-S31	SXA = OU6 OLD SW3 NRZ + ...	Selection bits to data lines. Bits 16-31 are used
	S16-S31 $\longrightarrow$ /DIO00/-/DIO15/	DIOXS = OU6 OLD SW3 NRZ + ...	
	Go to PH3	S/PH3 = PH2 NBR + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If SW1 = 0, remain in PH3	BRPH3 = (FARWD PH3) NSW1 + ...	
	1 $\not\longrightarrow$ SW4	S/SW4 = (FARWD PH3) NSW1 + ...	
		R/SW4 = FADIO SW1 + ...	
		S/SW1 = FARWD PH3 DIO49	DIO49 => response from interrupt chassis
		/DIO48/ = DIO48X1	
		DIO48X1 = FARWD SW4 + ...	
		WD = /DIO50/	

Mnemonic: WD (6D, ED)

(Continued)

Table 3-127. Write Direct Interrupt Control Mode, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (Cont.)		/DIO50/ = DIO50X1	True during write direct
		DIO50X1 = OU6 OLD SW3 + ...	
		/DIO49/ = FSA	
		FSA = NCNA CNB + FSA FS	Timing for inhibiting interrupts during WD, generated in interrupt circuits
		FS = /DIO48/	
		S/CNA = NCNA NCNB SCNA	
		R/CNA = CNB	
		S/CNB = CNA	
		R/CNB = ...	
		SCNA = EWDM FS + ENFNL RCENFNL NCPURESET	
		EWDM = WD B1619ONE	
		R/SW4 = FADIO SW1 + ...	
	If SW1 = 1, go to PH4	S/PH4 = PH3 NBR + ...	
PH4 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If /DIO49/ = 1, remain in PH4	BRPH4 = FARWD PH4 NDRQ + ...	
		S/DRQ = FARWD PH4 NDIO49 + ...	
		R/SW3 = CLEAR + ...	
		CLEAR = ENDE + ...	
		ENDE = FARWD PH4 DRQ + ...	

Mnemonic: WD (6D, ED)

Table 3-128. Write Direct Output Mode, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	Enable memory request for next instruction	MRQ/1 = FARWD PH1 + ...	
	P16-P31 → S16-S31	SXP = FADIO PH1 FADIO = FARWD + FAIO	Address of output device
	S16-S31 -/→ B16-B31	BXS = FADIO PH1 + ...	
	0's -/→ B0-B15		
	R28-R31 → LR28-LR31	Normal action	
		AXRR = FARWD PH1 + ...	Output data
	RR0-RR31 -/→ A0-A31	AX/1 = FARWD PH1 + ...	Register into A
	1 -/→ SW3	S/SW3 = FADIO + ...	
	Go to PH2	S/PH2 = PH1 NBR + ...	
PH2 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	B16-B31 → /DIO32/-/DIO47/	DIOXB = FARWD SW3 + ...	Device address to DIO address lines
	If R ≠ 0, A0-A31 → S0-S31	SXA = OU6 OLD SW3 NRZ + ...	
	S0-S31 → /DIO00/-/DIO31/	DIOXS = OU6 OLD SW3 NRZ + ...	Data into DIO data lines
	Go to PH3	S/PH3 = PH2 NBR + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If SW1 = 0, remain in PH3	BRPH3 = (FARWD PH3) NSW1 + ...	
	1 -/→ SW4	S/SW4 = (FARWD PH3) NSW1 + ...	
		/DIO48/ = DIO48X1	Strobe to device
		DIO48X1 = FARWD SW4 + ...	
		/DIO50/ = DIO50X1	
		DIO50X1 = OU6 OLD SW3 + ...	
		WD = /DIO50/	

Mnemonic: WD (6D, ED)

(Continued)

Table 3-12E. Write Direct Output Mode, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (Cont.)	When /DIO49/ = 1, 1 ⇸ SW1	S/SW1 = (FARWD PH3) DIO49 + ...	DIO49 is response from device
		R/SW1 = FADIO + ...	
		R/SW4 = FADIO SW1 + ...	
		R/CC = (FARWD SW1) NB1619Z + ...	Set condition codes from device
		S/CC3 = (FARWD SW1) NB1619Z DIO51 + ...	
		S/CC4 = (FARWD SW1) NB1619Z DIO52 + ...	
	If SW1 = 1, go to PH4	S/PH4 = PH3 NBR + ...	
PH4 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If DRQ = 0, remain in PH4	BRPH4 = FARWD PH4 NDRQ + ...	
	Set data received request	S/DRQ = FARWD PH4 NDIO49 + ...	When device response goes low
		R/SW3 = CLEAR + ...	
		CLEAR = ENDE + ...	
		ENDE = FARWD PH4 DRQ + ...	

Mnemonic: WD (6D, ED)

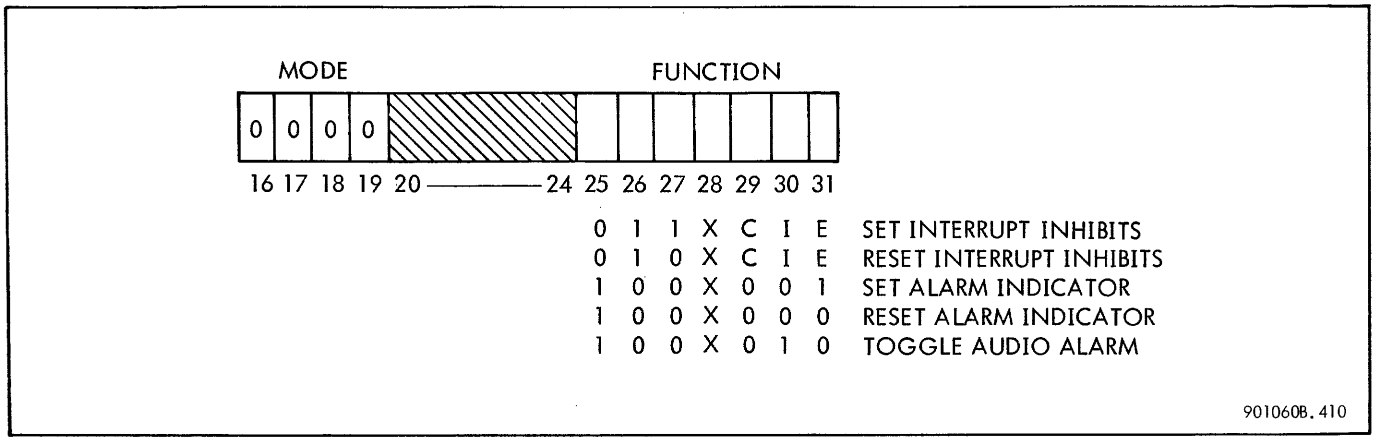

MODE
FUNCTION
0 0 0 0
16 17 18 19 20 —— 24 25 26 27 28 29 30 31
0 1 1 X C I E SET INTERRUPT INHIBITS
0 1 0 X C I E RESET INTERRUPT INHIBITS
1 0 0 X 0 0 1 SET ALARM INDICATOR
1 0 0 X 0 0 0 RESET ALARM INDICATOR
1 0 0 X 0 1 0 TOGGLE AUDIO ALARM
901060B.410

Figure 3-198. Write Direct Function Decoding in Internal Control Mode

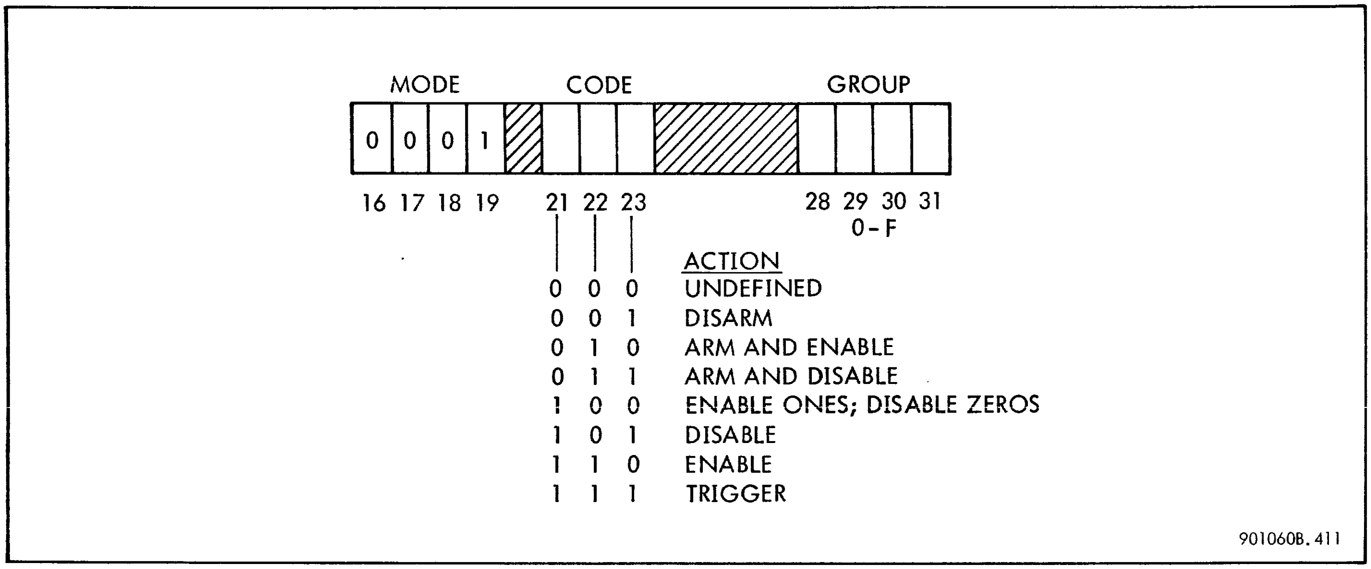


Figure 3-199. Write Direct Interrupt Control Mode Action Decoding

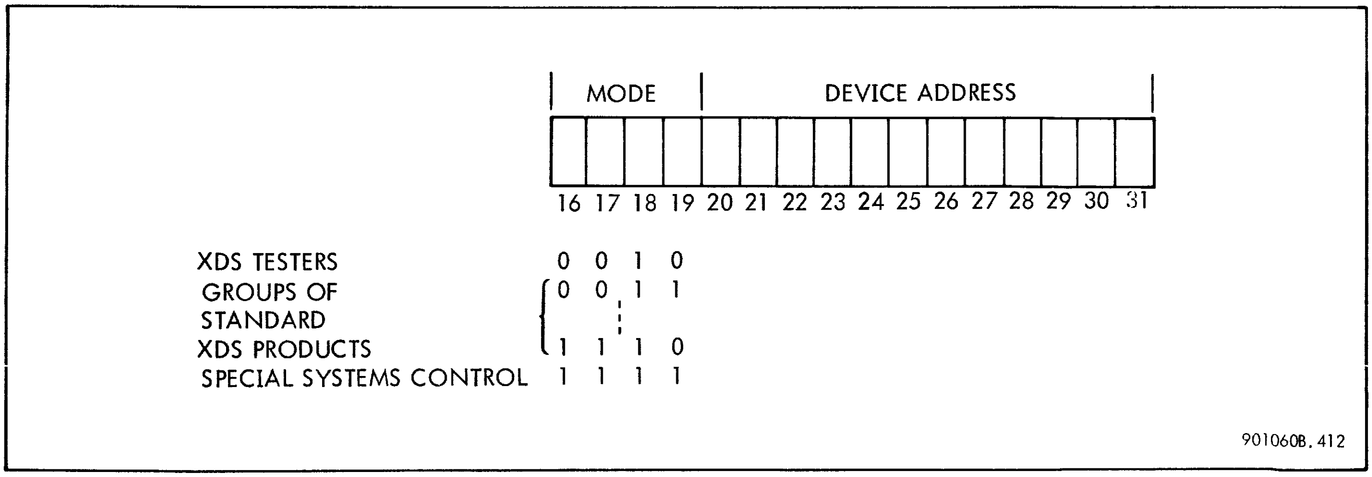


Figure 3-200. Write Direct Output Mode Decoding

The normal preparation sequence, as well as the sequencing for all modes and conditions of the WD instruction, is described under Preparation Sequence.

3-254 INTERNAL CONTROL MODE. Timing for the WD instruction, internal control mode, is shown in figure 3-201. During PH1, the memory request for the next instruction is enabled. The contents of the P-register are transferred to the sum bus and are then clocked into B16 through B31. During the remaining sequences of the WD instruction execution the B-register contains the mode (B1619Z) and the function to be performed. Flip-flop SW3 is set at the PH1 clock. The following flip-flops are set or are reset according to the contents of B25 through B31: ALARM, MUSIC, CIF, II, and EI.

The CPU sequences through two PH3 states. Flip-flop SW1 is true during the second PH3 sequence and, at the next clock, the condition codes read the PCP sense switches.

The CPU sequences from PH3 to PH4 for two clock times. DRQ is set, and SW3 is reset.

A sequence chart of the Write Direct internal control mode is shown in table 3-126.

3-255 INTERRUPT CONTROL MODE. Timing for the WD instruction execution, interrupt control mode, is shown in figure 3-202. During PH1 the memory request for the next instruction is enabled, the contents of the P-register are transferred to the sum bus and are then clocked into the B-register, B16 through B31. During the remaining part of the execution, the B-register contains the mode (B1619ONE), the function code in B21 through B23, and the interrupt group address in B28 through B31. Flip-flop SW3 is set.

The selection bits in the addressed private memory register are clocked into the A-register, and the contents of B16 through B31 are placed on DIO address lines /DIO31/ through /DIO47/. If the R-field of the WD instruction word did not address register 0, the contents of A are transferred to the sum bus, and sum bus bits S0 through S31 are placed on the DIO data lines /DIO00/ through /DIO31/. Only bits 15 through 31 contain the selection bits. Flip-flop SW4 is set.

The write direct output lines, /DIO00/ through /DIO31/ and WD address lines, /DIO32/ through /DIO47/, are shown in figures 3-203 and 3-204, respectively. Figure 3-205 is a logic diagram showing the interrupt group and function decoding for the WD instruction in the interrupt control mode. The CPU sequences to PH3 and remains in PH3 until SW1 is set. SW1 does not set until response signal DIO49 is received from the interrupt circuits. /DIO49/ is delayed for a minimum of 3 but not more than 4 microseconds, and is timed by flip-flops CNA and CNB in the interrupt chassis, which are clocked by the 1-MHz clock.

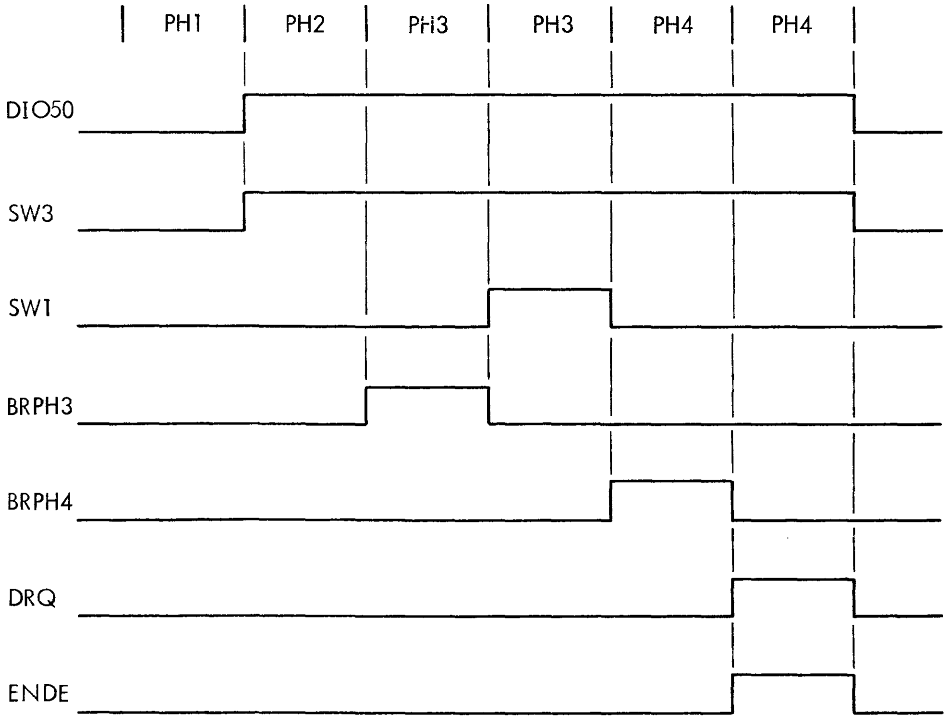


Figure 3-201. Write Direct Internal Mode, Timing Diagram

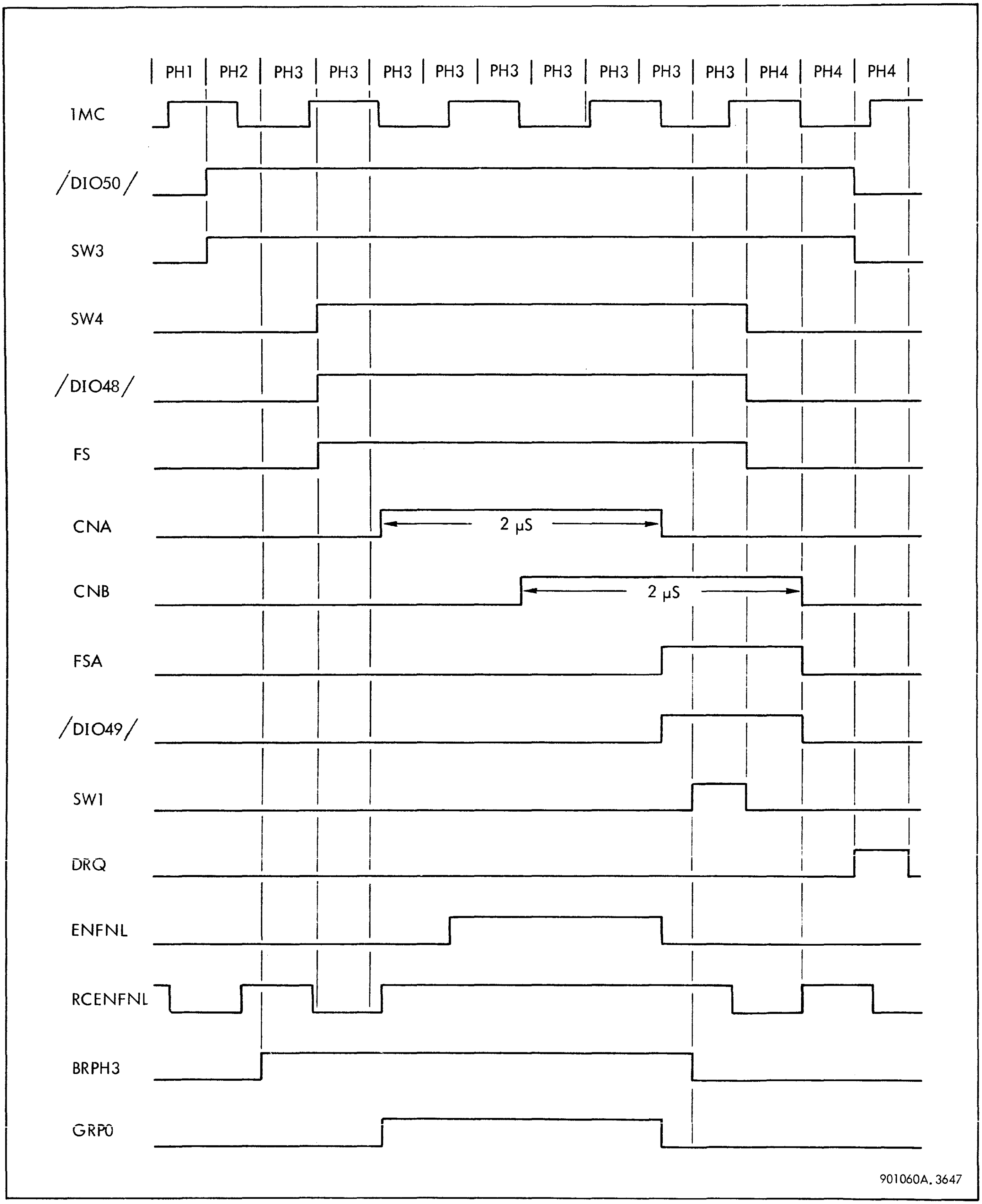


Figure 3-202. Write Direct Interrupt Mode, Timing Diagram

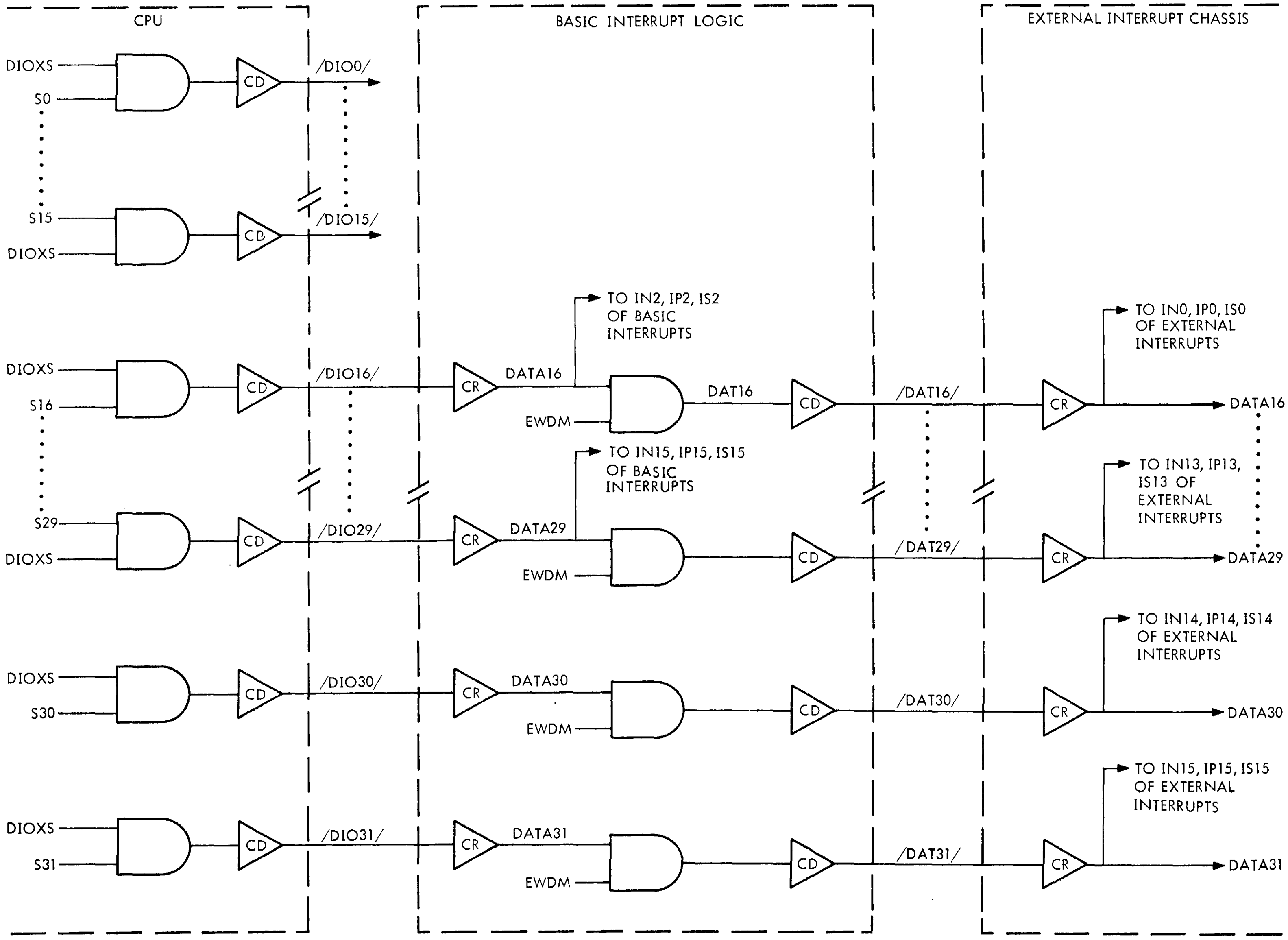

CPU
BASIC INTERRUPT LOGIC
EXTERNAL INTERRUPT CHASSIS
DIOXS
S0
S15
S16
S29
S30
S31
CD
CR
/DIO0/
/DIO15/
/DIO16/
/DIO29/
/DIO30/
/DIO31/
DATA16
DATA29
DATA30
DATA31
EWDM
DAT16
/DAT16/
/DAT29/
/DAT30/
/DAT31/
TO IN2, IP2, IS2 OF BASIC INTERRUPTS
TO IN15, IP15, IS15 OF BASIC INTERRUPTS
TO IN0, IP0, IS0 OF EXTERNAL INTERRUPTS
TO IN13, IP13, IS13 OF EXTERNAL INTERRUPTS
TO IN14, IP14, IS14 OF EXTERNAL INTERRUPTS
TO IN15, IP15, IS15 OF EXTERNAL INTERRUPTS
901060B.3648

Figure 3-203. Write Direct Data Lines

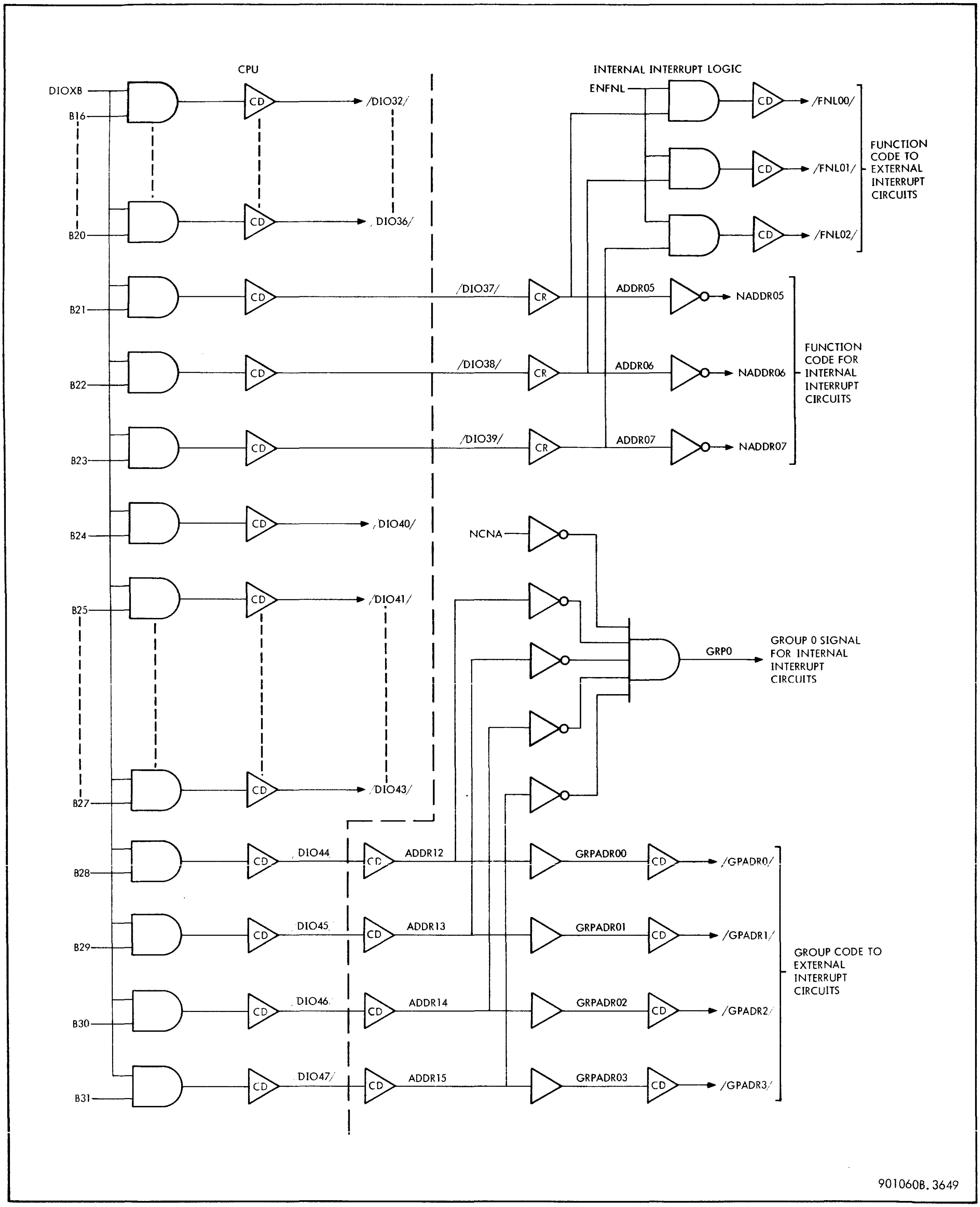


Figure 3-204. Write Direct Address Lines

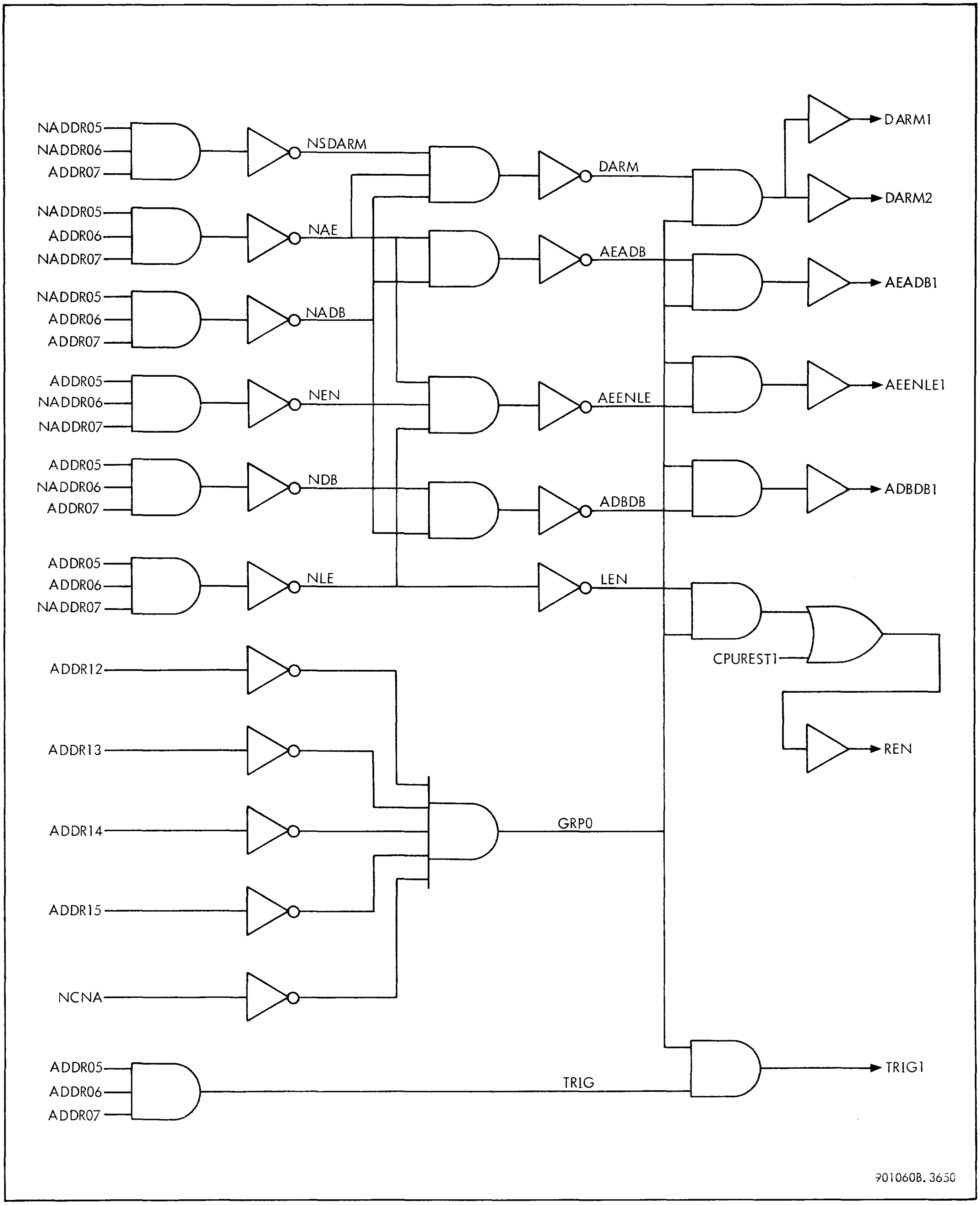


Figure 3-205. Write Direct Interrupt Mode Function Decoding

Signal /DIO49/ is generated in the internal interrupt chassis for both internal and external interrupts.

The internal interrupts levels are enabled, disabled, armed, disarmed, or triggered when CNA is true, since the addressed interrupt group gate can be true only when CNA is true. For example:

GRP0 = NADDR12 NADDR13 NADDR14
NADDR15 CNA

The addressed group gate in the external interrupt chassis is selected by the group code from the CPU and switch settings for each chassis:

```
GRPSEL = N(GPADR0  NSWTH0
          + GPADR1  NSWTH1
          + GPADR2  NSWTH2
          + GPADR3  NSWTH3
          + NGPADR0  SWTH0
          + NGPADR1  SWTH1
          + NGPADR2  SWTH2
          + NGPADR3  SWTH3)
```

The individual interrupt levels within a group are selected by signals DATA16 through DATA31, which are generated from the selection bits on the CPU data lines. The equations for the internal interrupt circuits are as follows:

```
S/IN2   = AEENLE1  DATA16 + ...

R/IN2   = ADBDB1  DATA16 + REN
   .
   .
   .
S/IN15  = AEENLE1  DATA29 + ...

R/IN15  = ADBDB1  DATA29 + REN


S/SP2   = AEADB1  DATA16 + ...

R/IP2   = DARM  DATA16 + ...
   .
   .
   .
S/IP15  = AEADB1  DATA29 + ...

R/IP15  = DARM  DATA29 + ...


S/IS2   = TRIG1  IP2  DATA16 + ...

R/IS2   = DARM  DATA16 + ...
   .
   .
   .
S/IS15  = TRIG1  IP15  DATA29 + ...

R/IS15  = DARM  DATA29 + ...
```

The first two interrupt levels of the basic interrupts (power on, power off) are never addressed by the WD instruction. The data lines, DATA16 through DATA31, are offset by two when addressing group 0 interrupts compared to all other interrupt groups. For example:

	Basic Interrupt Group	External Interrupt Group
DATA16	2	0
DATA17	3	1
DATA18	4	2
.	.	.
.	.	.
.	.	.
DATA28	14	12
DATA29	15	13
DATA30		14
DATA31		15

When /DIO49/ comes true, SW1 is set and the CPU sequences to PH4. The CPU remains in PH4 until /DIO49/ drops.

The number of PH3 and PH4 iterations for any WD instruction execution depends on the time relationship of the T6L clock and the 1-MHz clock as the instruction sequences out of PH2 into PH3.

A sequence chart of the Write Direct interrupt control mode is given in table 3-127.

3-256 OUTPUT MODE. Timing for the WD instruction, output mode, is shown in figure 3-206. During PH1 the memory request for the next instruction is enabled, the contents of the P-register are transferred to the sum bus and are clocked into B16 through B31. During the remaining sequences of the WD instruction, the B-register contains the address of the output device. SW3 is set at the PH1 clock.

The output data in the addressed private memory register is transferred from the RR lines to the A-register. The device address in B is placed on the direct output address lines, /DIO32/ through /DIO47/. If the R-field of the WD instruction word was not zero, the output data is taken from the A-register, is placed onto the sum bus, and then is transferred to the output data lines, /DIO00/ through /DIO31/.

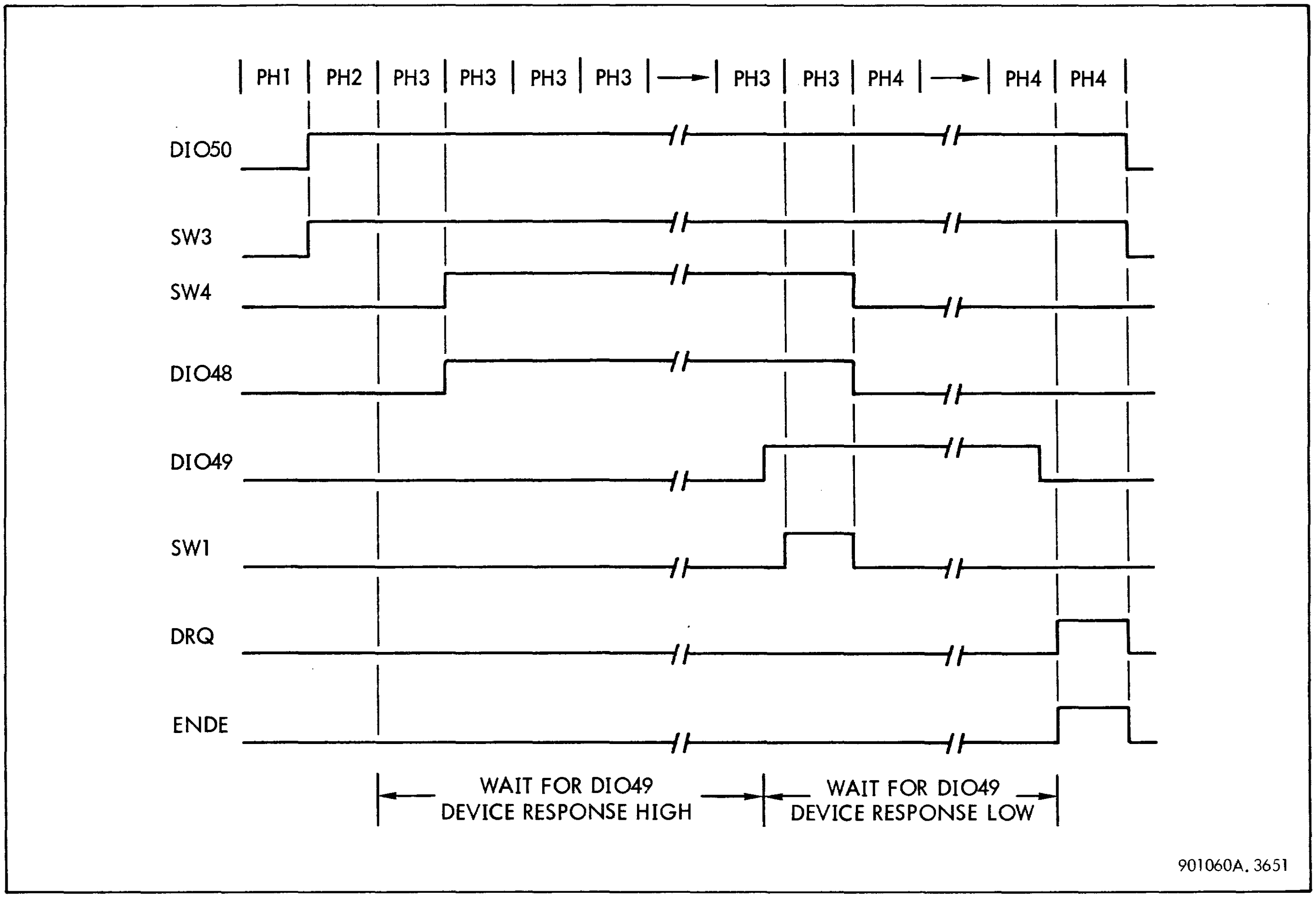


Figure 3-206. Write Direct External Mode, Timing Diagram

During PH3 the strobe signal /DIO48/ is sent to the output device. The CPU remains in PH3 until the response signal from the device /DIO49/ is received. When /DIO49/goes true, SW1 is set and, at the next clock condition, codes CC3 and CC4 are set according to the states of the device signals /DIO51/ and /DIO52/.

In PH4 the CPU waits for the device response signal /DIO49/ to fall. Flip-flop DRQ is set, and ENDE comes true to mark the final phase of the instruction.

A sequence chart of the Write Direct output mode is given in table 3-128.

3-257 Start Input/Output (SIO 4C, CC)

Before the SIO instruction is executed, private memory register R0 must contain, in bit positions 16 through 31, the first command address for the IOP operation. This address is placed into the A-register together with the IOP, controller and device addresses and the status (zero, odd or even) of the R-field of the instruction word. This data in the A-register is stored into memory location X'20' as shown in figure 3-207.

After the CPU stores this data into location X'20', the IOP reads this same data. If the IOP, controller or device address is nonexistent, CC1 is set and the SIO instruction execution is aborted. If these addresses are valid, the IOP performs the following operations:

a. If the R-field of the SIO instruction word is zero, the IOP attempts to start the addressed device. If the start is successful, CC2 is reset; if the start is not successful, CC2 is set.

b. If the R-field of the SIO instruction word is even, but is not zero, the IOP stores the current status and byte count of the IOP operation into memory location X'20', and stores the current command address into location X'21'.

c. If the R-field of the SIO instruction word is odd, the IOP stores the status and byte count of the current IOP operation into memory location X'21'.

If the contents of the R-field of the SIO instruction word are even but not zero, the CPU reads the contents of core memory location X'21' (current command address) into private memory register R, and reads the contents of X'20' (status and byte count) into private memory register R+1.

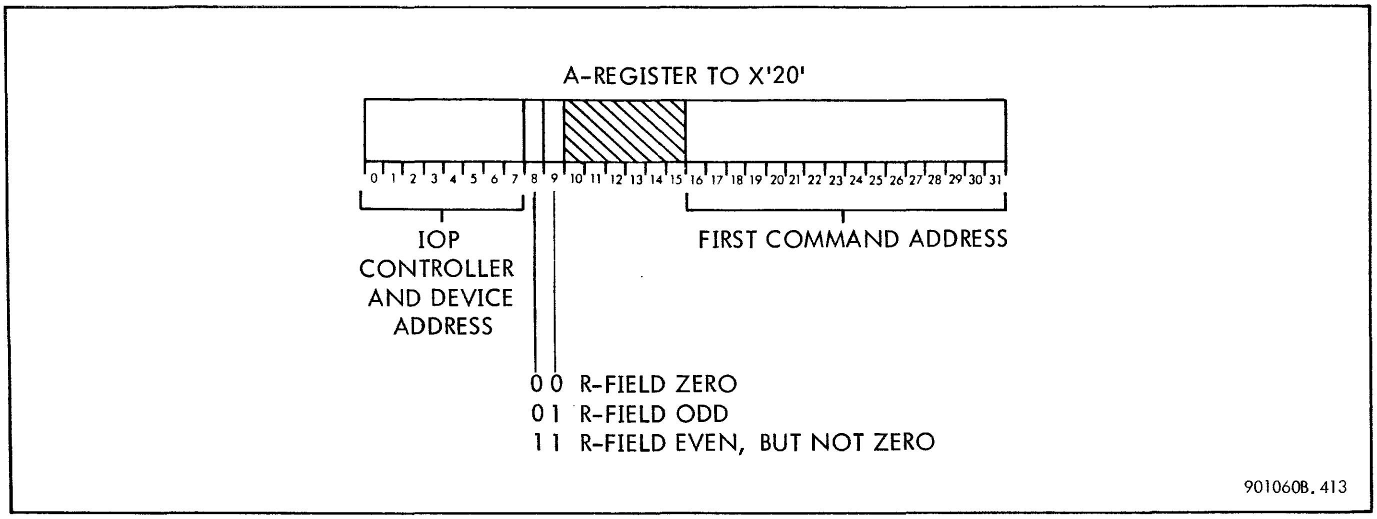


Figure 3-207. Start Input/Output Information Stored in Location X'20'

If the contents of the R-field of the SIO instruction word are odd, the CPU reads the contents of X'21' (status and byte count) into private memory register R. The meaning of each of the various status bits for all input/output instructions (SIO, HIO, TIO, TDV, AIO) is given in table 3-129.

The preparation sequence for the SIO instruction is the same as the general preparation sequence described under the Preparation Sequence.

All conditions of the SIO instruction execution are described by the logic sequence diagram, figure 3-208. This diagram also describes the execution of all other input/output instructions (HIO, TIO, TDV, AIO). Figure 3-209 is a simplified timing diagram that applies to an SIO instruction when the R-field of the instruction word is even but is not zero. Figure 3-210 shows the timing for the same instruction when the R-field of the instruction word is zero, or when the R-field is not zero, but the IOP made an unsuccessful start.

In PH1 the contents of the P-register (which contains the first command address) are transferred to the sum bus and are then clocked into the least significant halves of both the A- and B-registers. The indicator flip-flop SW1 is reset, the data flip-flop SW3 is reset, and ones are clocked into the CS-register for the upward alignment of A to the sum bus during PH2. The memory map is disabled, since locations X'20' and X'21' must always be actual addresses.

At PH1 clock flip-flop NLRXR is set to force zeros onto the LR lines during PH2. During PH2 of an SIO instruction, the function code lines, FNC0 through FNC2, are gated to represent the type of input/output instruction, where:

FNC0	FNC1	FNC2	
0	0	0	Represents SIO (4C)
0	0	1	Represents TIO (4D)
0	1	0	Represents TDV (4E)
0	1	1	Represents HIO (4F)
1	1	0	Represents AIO (6E)

The LR lines are forced to an address of zero by NLRXR true. The 17 least significant bits of register R0 are read and are clocked into A16 through A31, but only if the instruction is an SIO. Bits A24 through A31 are realigned and are transferred to S0 through S7 and are clocked back into A0 through A7. A9 is set if the R-field of the instruction word is not zero. A8 is set if the R-field is an even number but is not zero. For all I/O instructions except AIO, MRQ is enabled in PH2. The address release request flip-flop ARQ is set and the P-register is set to an address of 20_{16}.

In PH3 the strobe flip-flop SW4, the address release request flip-flop ARQ, and the data request flip-flop DRQ are set. Phase 3 repeats until the proceed signal PR is received from the IOP. The indicator flip-flop SW1 sets at the next clock after the proceed signal is received. The contents of the A-register are read into memory location 20_{16} in PH3, and condition codes CC1 and CC2 are set or reset according to the IOP signals COND1 and COND2.

At this point in PH3, two conditions can exist that sequence the CPU to PH8 to read the next instruction: COND1 true

Table 3-129. Status Bits for Input/Output Instructions

Position and State in Register R+1				Meaning (SIO, HIO, TIO)
0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	
1 - - -	- - - -	- - - -	- - - -	Device interrupt pending
- 0 0 -	- - - -	- - - -	- - - -	Device ready
- 0 1 -	- - - -	- - - -	- - - -	Device not operational
- 1 0 -	- - - -	- - - -	- - - -	Device unavailable
- 1 1 -	- - - -	- - - -	- - - -	Device busy
- - - 0	- - - -	- - - -	- - - -	Device manual
- - - 1	- - - -	- - - -	- - - -	Device automatic
- - - -	1 - - -	- - - -	- - - -	Device unusual end
- - - -	- 0 0 -	- - - -	- - - -	Device controller ready
- - - -	- 0 1 -	- - - -	- - - -	Device controller not operational
- - - -	- 1 0 -	- - - -	- - - -	Device controller unavailable
- - - -	- 1 1 -	- - - -	- - - -	Device controller busy
- - - -	- - - 0	- - - -	- - - -	Unassigned
- - - -	- - - -	1 - - -	- - - -	Incorrect length
- - - -	- - - -	- 1 - -	- - - -	Transmission data error
- - - -	- - - -	- - 1 -	- - - -	Transmission memory error
- - - -	- - - -	- - - 1	- - - -	Memory address error
- - - -	- - - -	- - - -	1 - - -	IOP memory error
- - - -	- - - -	- - - -	- 1 - -	IOP control error
- - - -	- - - -	- - - -	- - 1 -	IOP halt
- - - -	- - - -	- - - -	- - - 1	IOP busy (SIOP only)

Position and State in Register R				Meaning (AIO)	Meaning (TDV)
0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15		
1 - - -	- - - -	- - - -	- - - -	Data overrun	Data overrun
- 1 - -	- - - -	- - - -	- - - -	Device end interrupt	Device end
- - 1 -	- - - -	- - - -	- - - -	Unique to the device and the device controller	
- - - 1	- - - -	- - - -	- - - -		
- - - -	1 - - -	- - - -	- - - -		
- - - -	- 1 - -	- - - -	- - - -		
- - - -	- - 1 -	- - - -	- - - -		
- - - -	- - - 1	- - - -	- - - -		
- - - -	- - - -	1 - - -	- - - -	Incorrect length	Incorrect length
- - - -	- - - -	- 1 - -	- - - -	Transmission data error	Transmission data error
- - - -	- - - -	- - 1 -	- - - -	Zero byte count interrupt	Transmission memory error
- - - -	- - - -	- - - 1	- - - -	Channel end interrupt	Memory address error
- - - -	- - - -	- - - -	1 - - -	Unusual end interrupt	IOP memory error
- - - -	- - - -	- - - -	- 0 - -	Unassigned	IOP control error
- - - -	- - - -	- - - -	- - 0 -		IOP halt
- - - -	- - - -	- - - -	- - - 0		IOP busy

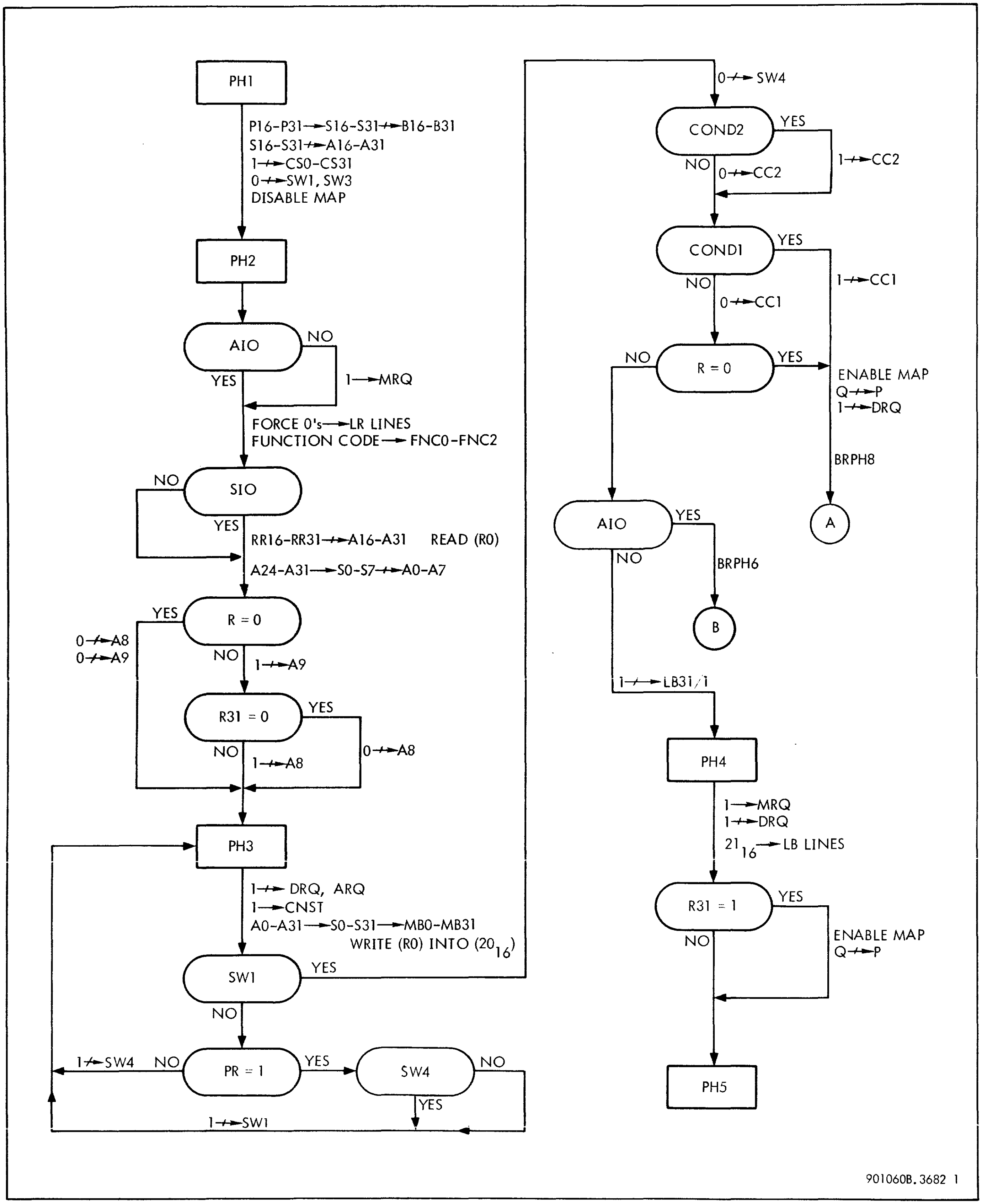


Figure 3-208. SIO, HIO, TIO, TDV, AIO, Logic Sequence Diagram (Sheet 1 of 2)

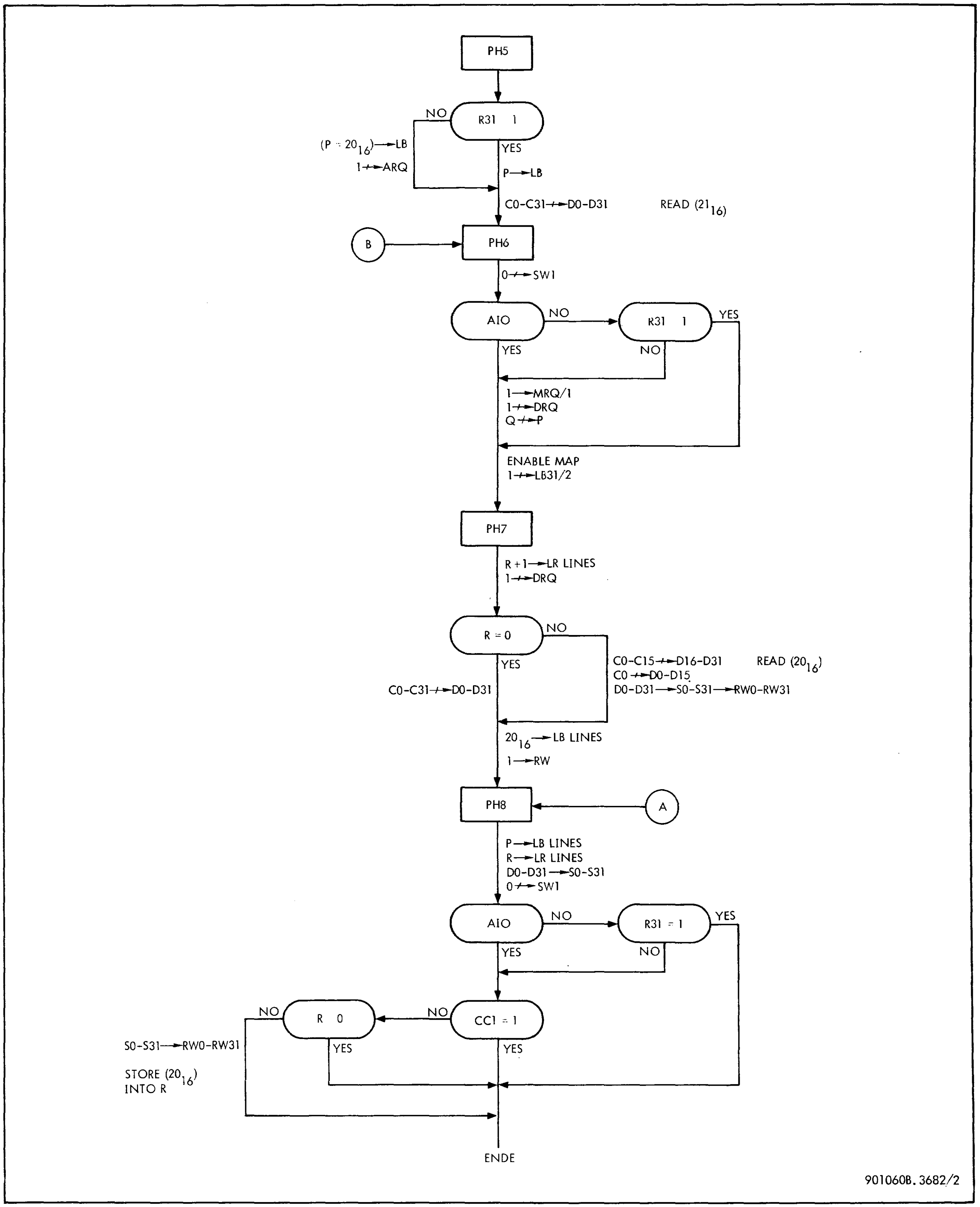


Figure 3-208. SIO, HIO, TIO, TDV, AIO, Logic Sequence Diagram (Sheet 2 of 2)

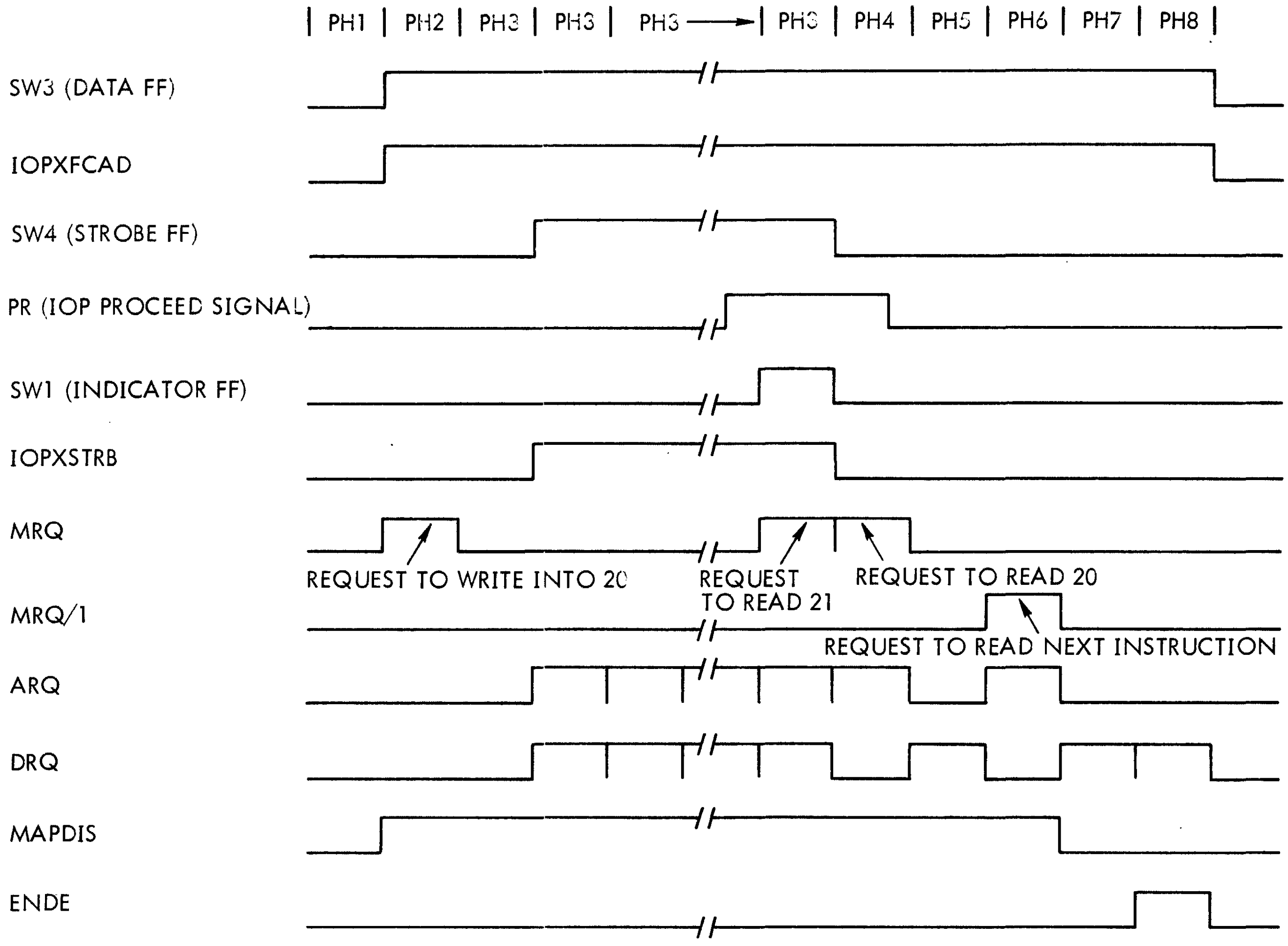


Figure 3-209. SIO, R-Field Even But Not Zero, Simplified Timing Diagram

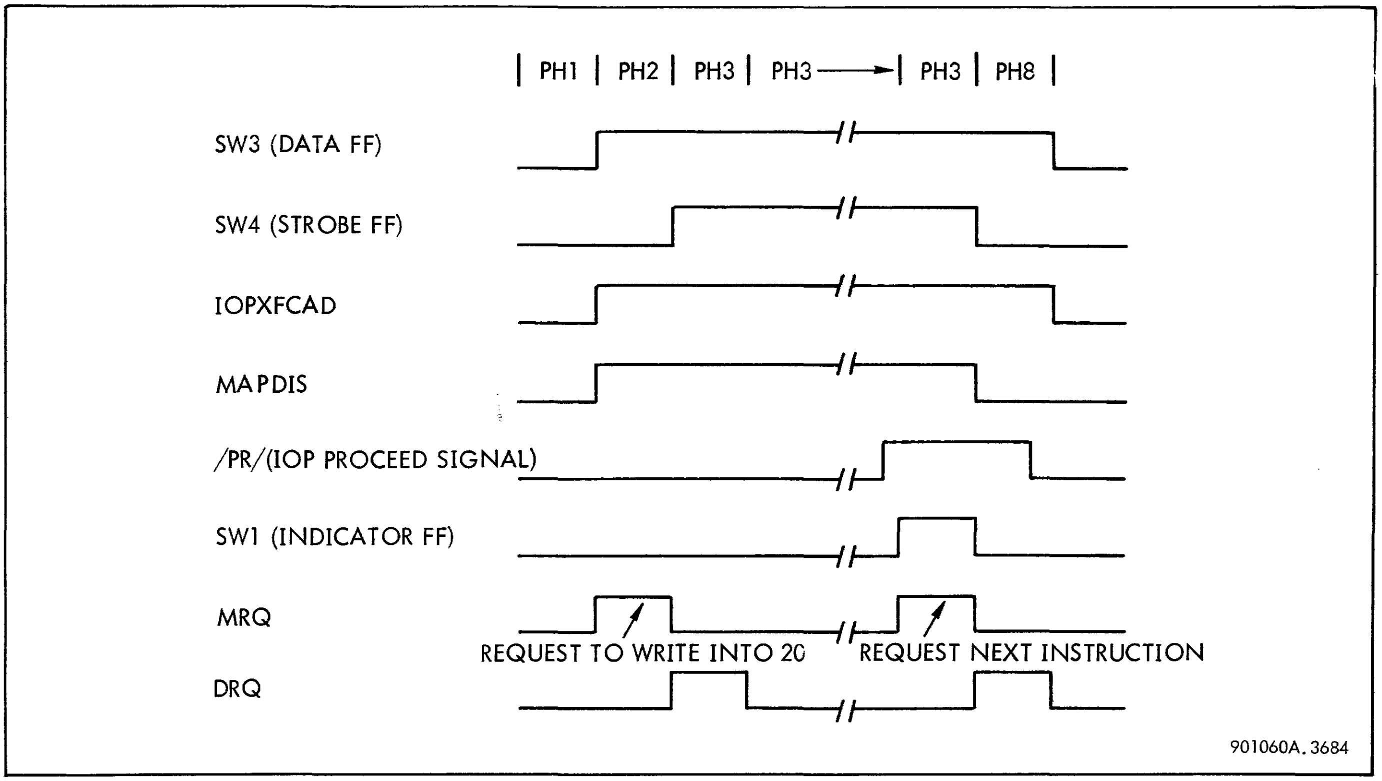


Figure 3-210. SIO, R-Field Zero or R-Field Not Zero But Start Unsuccessful, Simplified Timing Diagram

(nonexistent IOP address), or the R-field of the instruction word was zero. In either case, the contents of memory locations X'20' and X'21' are not read into private memory. The memory map is enabled; DRQ is set; and the next instruction address in Q is transferred to the P-register.

If the R-field in the instruction word is not zero, flip-flop LB31/1 is set in order to address memory location 21_{16} in PH4.

S/LB31/1 = FAIO PH3 NBR

If the instruction is an AIO with the R-field zero and the COND1 signal from the IOP true, the memory map is enabled, the next instruction address in Q is transferred to P, and the CPU branches to PH6.

In PH5, the contents of the P-register are placed on the memory address lines. P equals 20_{16} if the R-field of the instruction word is even, and ARQ is set; if the R-field of the instruction word is odd, the contents of P are the next instruction address. At PH5 clock, the contents of location 21_{16} are transferred from the C-register into D.

In PH6, if the instruction is an AIO or if the R-field of the instruction word is even, MRQ/1 is enabled to request the next instruction from memory. The data request flip-flop DRQ is set, and the next instruction address in the Q-register is clocked into P. The memory map is enabled, and flip-flop LR31/2 is set to address R+1.

In PH7, if the instruction is not an AIO, the contents of memory location 20_{16} are gated into the C-register and are then clocked into D. This data in D is not stored into register R0, however, until PH8. If the instruction is an SIO, HIO, TIO, or TDV, the 16 high order bits of C are transferred to the 16 low order bits of D. D0 through D15 extend the status of C0. The contents of D are transferred to the sum bus, are placed onto the RW lines, and are then transferred to register R+1.

In PH8, if the R-field of the instruction word is even, but is not zero, and if CC1 is not true, the contents of the sum bus (the original contents of memory location 20_{16}) are transferred to register R.

A sequence chart of the Start Input/Output instruction is given in table 3-130.

3-258 Halt Input/Output (HIO 4F, CF)

The HIO instruction causes the addressed device to halt immediately its current operation. If the R-field of the HIO instruction is zero, condition codes CC1 and CC2 are affected, but no data is transferred into private memory registers. If the R-field of the HIO instruction is an even value but is not zero, condition codes CC1 and CC2 are affected, private memory register R is loaded with the current command address, and register R+1 is loaded with the current status and the byte count of the IOP operation being halted. If the R-field of the HIO instruction word is an odd

Table 3-130. Start Input/Output, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PH1 T4RL	P16-P31 → S16-S31	SXP = FAIO PH1 + ...	I/O address
	S16-S31 ↛ A16-A31	AXS = FAIO PH1 + ...	
	S16-S31 ↛ B16-B31	BXS = FAIO PH1 + ...	
	1's ↛ CS0-CS31	S/NPRX = FAIO PH1 + ...	Ones into CS-register for upward alignment in PH2
		CSX1 = (S/NPRX)	
	1 ↛ SW3	S/SW3 = FAIO PH1 + ...	Data flip-flop
	0 ↛ SW1	R/SW1 = FAIO PH1 + CLEAR + ...	Indicator flip-flop
	Disable MAP	S/MAPDIS = FAIO PH1 + ...	Since locations X'20' and X'21' must be actual addresses
	Prepare for 0's → LR in PH2	S/NLRXR = FAIO PH1 + ...	
	Select T6RL for PH2	T6RL = FAIO PH1 + ...	
	Go to PH2	S/PH2 = PH1 NBR + ...	
PH2 T6RL	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	A24-A31 → K23-K30 K23-K30 → S0-S7 K23-K30 → S8-S15 K23-K30 → S16-S23 K23-K30 → S24-S31	SXUAB = FAIO PH2 + ...	Upward align byte
	S0-S7 ↛ A0-A7	AXS/1 = FAIO PH2 + ...	
		AX/1 = FAIO PH2 + ...	
	0's → LR lines		Address private memory register D because NLRXR is true
	RR16-RR31 ↛ A16-A31, if SIO	AXRR/2 = OU4 OLC PH2 + ...	Command address SIO only
	If $R \neq 0$, 1 ↛ A9	A9X1 = FAIO PH2 NRZ + ...	
	If $R \neq 0$, but even, 1 ↛ A8	A8X1 = FAIO PH2 NRZ NR31 + ...	
	20_{16} ↛ P16-P31	PX20 = FAIO PH2 + ...	Memory address 20_{16}

Mnemonic: SIO (4C, CC)

(Continued)

Table 3-130. Start Input/Output, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6RL (Cont.)		S/P26 = PX20 + ...	
	Enable function codes	/FNC0/ = IOPXFCAD O2	
		/FNC1/ = IOPXFCAD O6	
		/FNC2/ = IOPXFCAD O7	
	B21-B23 —/→ IOPA0-IOPA3	IOPXFCAD = FAIO SW3	IOP address to IOP's
	Enable memory request	MRQ = FAIO/1 PH2 + ...	But not if AIO
		FAIO/1 = FAIO OU4	
	Set address release request	S/ARQ = FAIO PH2 + ...	
	Go to PH3	S/PH3 = PH2 NBR + ...	
PH3 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	Set address release request	S/ARQ = FAIO PH3 + ...	
	A0-A31 ⟶ S0-S31	SXA = FAIO PH3 + ...	Command address ⟶ location 20_{16}
	S0-S31 ⟶ MB0-MB31	MW = FAIO PH3 + ...	
	20_{16} ⟶ LB lines		To write into 20_{16}
	1 —/→ SW4	S/SW4 = FAIO3 NPR NSW1 + ...	Set strobe flip-flop if proceed signal from IOP has not arrived
	Set data request	S/DRQ = FAIO3 NSW1 + ...	
	Repeat PH3 until SW1 = 1	BRPH3 = FAIO3 NSW1 + ...	
	1 —/→ SW1	S/SW1 = FAIO3 PR NSW1 SW4 + ...	After proceed signal PR arrives, set indicator flip-flop, and then reset at next clock
	Enable memory request	MRQ = FAIO3 SW1 + ...	
	If R = 0, or COND1 = 1,		
	Q16-Q31 —/→ P16-P31	PXQ = (FAIO RZ + FAIO COND1) SW1 +...	Next instruction address

Mnemonic: SIO (4C, CC)

(Continued)

Table 3-130. Start Input/Output, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH3 T6L (Cont.)	Enable MAP	R/MAPDIS	= (FAIO RZ + FAIO COND1) SW1 + ...	
	Set data release request	S/DRQ	= (FAIO RZ + FAIO COND1) SW1 + ...	
	Go to PH8	BRPH8	= (FAIO RZ + FAIO COND1) SW1 + ...	
	Set condition codes	(S/CC1/1)	= FAIO3 COND1 SW1 + ...	COND1 => nonexistent IOP address
		(S/CC2/1)	= FAIO3 COND2 SW1 + ...	
	If (AIO, R = 0, COND1 = 1),			
	Enable MAP	R/MAPDIS	= (FAIO RZ + FAIO COND1) SW1 + ...	
	Go to PH6, if AIO	BRPH6	= (OU6 OLE NRZ NCOND1) SW1 + ...	
	Strobe IOP via CNST line	CNST	= IOPXSTRB	
		IOPXSTRB	= FAIO3 SW4	
	If no branch to PH6 or PH8,			
	$1 \nrightarrow$ LB31/1	(S/LB31/1)	= FAIO PH3 NBR + ...	
	Go to PH4	S/PH4	= PH3 NBR + ...	
PH4 T6L	Enable T6L	T6L	= NT1L NT4L NT8L NT10L NRESET	
	$21_{16} \longrightarrow$ LB lines			To read location X'21'$_{16}$. P26 set by PX20 in PH2. LB31 set in PH3
	Enable memory request	MRQ	= FAIO PH4 + ...	
	Set data release request	S/DRQ	= FAIO PH4 + ...	
	If R31 = 1,			
	Q16-Q31 $\nrightarrow$ P16-P31	PXQ	= FAIO PH4 R31 + ...	If R-field was odd
	Go to PH5	S/PH5	= PH4 NBR + ...	

(Continued)

Table 3-130. Start Input/Output, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	C0-C31 —/→ D0-D31	DXC/6 = FAIO PH5 + ...	Contents of location X'21'
	If R31 = 1,		
	Set address release request	S/ARQ = FAIO PH5 NR31 + ...	
PH6 T6L	Enable T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
	If AIO, or R31 = 1		
	Enable memory request	MRQ/1 = FAIO PH6 (NR31 + O2) + ...	For next instruction
	Set data release request	S/DRQ = FAIO PH6 (NR31 + O2) + ...	
	Q15-Q31 —/→ P15-P31	PXQ = MRQ/1 + ...	Next instruction address
	Enable MAP	R/MAPDIS = FAIO PH6 + ...	
	1 —/→ LR31/2	S/LR31/2 = FAIO PH6 + ...	
	1 —/→ T8L	S/T8L = FAIO PH6 + ...	
	Go to PH7	S/PH7 = PH6 NBR + ...	
PH7 T8L	If AIO, C0-C31 —/→ D0-D31	DXC/6 = OU6 OLE PH7 + ...	
	If not AIO, C0-C15 —/→ D16-D31 C0 —/→ D0-D15	DXCR16 = FAIO/1 PH7 + ...	Contents of location 20_{16}
	D0-D31 —/→ S0-S31	SXD = FAIO/1 PH7 + ...	
	S0-S31 ——→ RW0-RW31	RW = FAIO/1 PH7 + ...	Transfer contents of location X'21' to odd numbered private memory register
	Set data request	S/DRQ = FAIO PH7 + ...	
	Set T8L	(S/T8L) = FAIO PH7 + ...	
	Go to PH8	S/PH8 = PH7 NBR + ...	

Mnemonic: SIO (4C, CC)

(Continued)

Table 3-130. Start Input/Output, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH8	R28-R31 → LR lines		
T&L	P16-P31 → LB lines		
	Reset data flip-flop	R/SW3 = CLEAR + ...	
		CLEAR = ENDE + ...	
		ENDE = FAIO PH8 + ...	
	D0-D31 → S0-S31	SXD = FAIO PH8 O2 + FAIO/1 PH8 + ...	
	If (AIO + R31 = 0) (R ≠ 0, CC1 = 1)		
	S0-S31 → RW0-RW31	RW = FAIO NRZ NCC1 (NR31 + O2) PH8 + ...	

Mnemonic: SIO 4C, CC)

value, condition codes CC1 and CC2 are affected, and register R is loaded with the current status and the byte count of the IOP operation is halted. Register R+1 is not affected.

Condition codes CC1 and CC2 have the following meaning after the HIO instruction has been executed:

CC1	CC2	
0	-	I/O address recognized
1	-	I/O address not recognized
-	0	Device not operating at time of the halt
-	1	Device operating at time of the halt

The preparation sequence for the HIC instruction is the same as the general preparation sequence described under Preparation Sequence.

All conditions of the HIO instruction execution are described by the Logic Sequence Diagram, figure 3-208, and by the SIO execution sequence logic. An HIO instruction is identical to an SIO instruction in its execution except that register R0 is not used, and therefore the contents of register R0 are inhibited from entering the A-register in PH2 by the equation

$$AXRR/2 = OU4\ OLC\ PH2$$

which applies only to an SIO instruction.

The meaning of the status bits in register R+1 after the HIO instruction has been executed is described in table 3-129.

3-259 Test Input/Output (TIO 4D, CD)

The TIO instruction interrogates the addressed IOP, controller, and device for its current status without affecting these units in any way. No input/output operations are initiated or are terminated by this instruction.

If the R-field of the TIO instruction is zero, condition codes CC1 and CC2 are affected but no data is transferred into private memory registers. If the R-field of the TIO instruction is an even value but is not zero, condition codes CC1 and CC2 are affected, private memory register R is loaded with the current command address, and register R+1 is loaded with the current status and the remaining byte count of the IOP operation. If the R-field of the TIO instruction word is an odd value, condition codes CC1 and CC2 are affected, and register R is loaded with the current status and the remaining byte count of the IOP operation; register R+1 is not affected. The meaning of the various status bits for the TIO instruction is given in table 3-129.

Condition codes CC1 and CC2 have the following meanings after the TIO instruction has been executed:

CC1	CC2	
0	-	I/O address recognized
1	-	I/O address not recognized
-	0	Successful SIO is currently possible
-	1	Successful SIO is not currently possible

The preparation sequence for the TIO instruction is the same as the general preparation sequence described under Preparation Sequence.

All conditions of the TIO instruction execution are described by the Logic Sequence Diagram, figure 3-208, and by the SIO execution sequence logic described in table 3-130. A TIO instruction is identical to an SIO instruction in its execution except that register R0 is not used. Therefore, the contents of register R0 are inhibited from entering the A-register in PH2 by the equation which applies only to an SIO instruction.

3-260 Test Device (TDV 4E, CE)

The TDV instruction obtains information about the addressed device other than that obtainable by means of a TIO instruction. The operation of the selected IOP, controller, and device are not affected. No operations are initiated or are terminated by this instruction. Responses to TDV provide the program with information giving details on the condition of the selected device, the number of bytes remaining to be transmitted in the current operation, and the present point at which the IOP is operating in a command list.

If the R-field of the TDV instruction is zero, the condition code flip-flop CC1 is affected but no private memory registers are affected. If the R-field of the instruction word is even, but not zero, the condition code flip-flop CC1 is affected, register R is loaded with the current command address, and register R+1 is loaded with the current status and the remaining byte count of the IOP operation. If the R-field of the instruction word is odd, condition code flip-flop CC1 is set, and the device status and the remaining byte count are loaded into register R.

After the TDV instruction has been executed, condition code flip-flop CC1 has the following meaning:

CC1	
0	I/O address recognized
1	I/O address not recognized

The preparation sequence for the TDV instruction is the same as the general preparation sequence described under Preparation Sequence.

All conditions of the TDV instruction execution are described by the logic sequence diagram, figure 3-208 and by the SIO execution sequence logic described in table 3-130. A TDV instruction is identical to an SIO instruction in its execution except that register R0 is not used. Therefore, the contents of register R0 are inhibited from entering the A-register in PH2 by the equation

AXRR/2 = OU4 OLC PH2

which applies only to an SIO instruction.

3-261 Acknowledge Input/Output (AIO 6E, EE)

The AIO instruction is used to acknowledge an I/O interrupt and to identify the source and the reason for the interrupt. Bits 21 through 23 of the effective program address of the AIO instruction word (the IOP address portion of the I/O selection code field) specify the type of interrupt being acknowledged. For the standard I/O system interrupt acknowledgement, these bits should be coded 000. Other codings are reserved for use with special I/O systems.

If the R-field of the AIO instruction word is zero, only condition code flip-flops CC1 and CC2 are affected. CC1 and CC2 are affected if the R-field of the AIO instruction word is not zero. Register R is loaded with status identification bits and with the address of the interrupting I/O unit. The status bits for acknowledging an I/O interrupt are described in table 3-129.

After the TDV instruction has been executed, condition code flip-flops CC1 and CC2 have the following meanings:

CC1	CC2	
0	-	I/O interrupt recognition
1	-	No I/O interrupt is present
-	0	Normal interrupt
-	1	Unusual condition interrupt

The preparation sequence for the AIO instruction is the same as the general preparation sequence described under Preparation Sequence.

All conditions of the AIO instruction execution are described by the logic sequence diagram, figure 3-208, and by the SIO execution sequence logic described in table 3-130. The execution of an AIO instruction is similar to the execution of an SIO instruction except that register R0 is not used. Therefore, the contents of register R0 are inhibited from entering the A-register in PH2 by the equation

AXRR/2 = OU4 OLC PH2

which applies only to an SIO instruction.

3-262 FLOATING POINT FEATURE

3-263 General

Implemented floating point instructions can be used to add, subtract, multiply, and divide floating point numbers. Floating point instructions are implemented by the addition of the floating point option to the Sigma 7 computer system. If the floating point option is not implemented in the computer and if execution of a floating point instruction is attempted, the computer will abort execution of the instruction and will trap to location X'41'(65), the unimplemented instruction trap location.

The following floating point instructions are included in the floating point option:

Instruction	Mnemonic	Basic Family
Floating Add Short	FAS	FAFLAS
Floating Add Long	FAL	FAFLAS
Floating Subtract Short	FSS	FAFLAS
Floating Subtract Long	FSL	FAFLAS
Floating Multiply Short	FMS	FAFLM
Floating Multiply Long	FML	FAFLM
Floating Divide Short	FDS	FAFLD
Floating Divide Long	FDL	FAFLD

3-264 Floating Point Formats

There are short and long floating point number formats possible with the Sigma 7 computer. Both are shown in figure 3-211. The short format is made up of a sign bit (bit 0), a biased base-16 exponent (bits 1 through 7), and a 24-bit fraction (bits 8 through 31).

The long format floating point number adds 32 bits of lower significance to the fraction. The short format floating point number can be contained in a single word of memory or in one general register; the long format floating point number occupies a doubleword memory location or an even-odd pair of general registers.

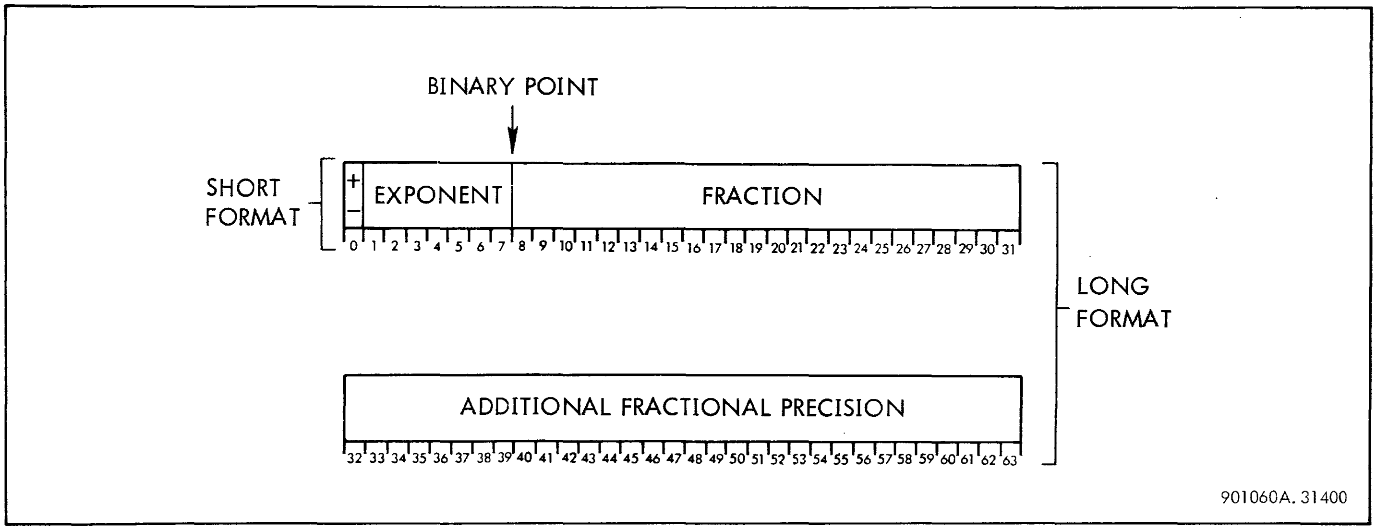


Figure 3-211. Short and Long Floating Point Formats

The range of numbers that can be expressed in both the short and the long floating point formats and the register for sample floating point numbers are shown in figure 3-212. The floating point number contained in a register is expressed by the equation

$$\text{Floating point number} = F \times 16^{E}$$

The fraction of the floating point number, F, is contained in bit position 0 (sign) and in bit positions 8 through 31 (short format) or 8 through 63 (long format). The binary point is before bit position 8. The fraction is in true form if the sign bit is zero (positive floating point number). It is in two's complement form if the sign bit is a one (negative floating point number).

Bit positions 1 through 7 contain a quantity that is 64 more than the exponent, E, of the floating point number. This is the biased exponent (excess 64). Subtracting 64 from this quantity produces the actual or unbiased exponent of the floating point number. If the floating point number is positive (sign bit zero), the exponent is in true form; if the floating point number is negative (sign bit one), the exponent is in one's complement form. The term inverted exponent refers to the one's complemented (negated) form; the term uninverted exponent refers to the true form of the exponent of a floating point number.

Biasing the exponent by 64 and inverting the biased exponent of a negative number enables processing of floating point numbers with fixed-point instructions, making possible, for example, the performance of a Compare instruction.

The range of Sigma 7 floating point numbers is from approximately $+16^{63}$ to approximately -16^{63} for both short and long formats. (The most positive and most negative long format floating point numbers are slightly more positive or negative than the short format because of the extra bits of significance of the fraction.) There are several types of floating point numbers which equal zero. A number with a zero fraction and a zero-biased exponent (shown in figure 3-212), for example, is a true zero. A number with a fraction of zero and a nonzero, biased exponent is an abnormal zero and may or may not be treated as a true zero.

3-265 Implementation of Floating Point Instructions

3-266 MODULE ADDITIONS FOR FLOATING POINT OPTION. Forty-eight modules are added to the Sigma 7 system to implement the floating point arithmetic instructions. In addition to providing floating point logic circuits, these modules contain the circuitry required to extend five of the CPU registers by 25 bit positions. The CPU registers and their floating point extensions are shown in figure 3-213. The S-bus extension is also shown.

3-267 FLOATING POINT REGISTER NOMENCLATURE. In the discussion of floating point instructions, the following nomenclature will be used:

a. The letters A, B, C, CS, and D refer to bit positions 0 (MSB) through 31 of the respective register (32 bits). The letter S refers to bit positions 0 through 31 of the sum bus.

b. The designations A', B', C', CS', and D' refer to bit positions 47 (MSB) through 71 of the respective register (25 bits). The designation S refers to bit positions 47 through 71 of the sum bus.

c. The designations A", B", C", CS", and D" refer to bit positions 47 (MSB) through 31 (57 bits) of the respective registers; S" refers to bit positions 47 through 31 of the sum bus.

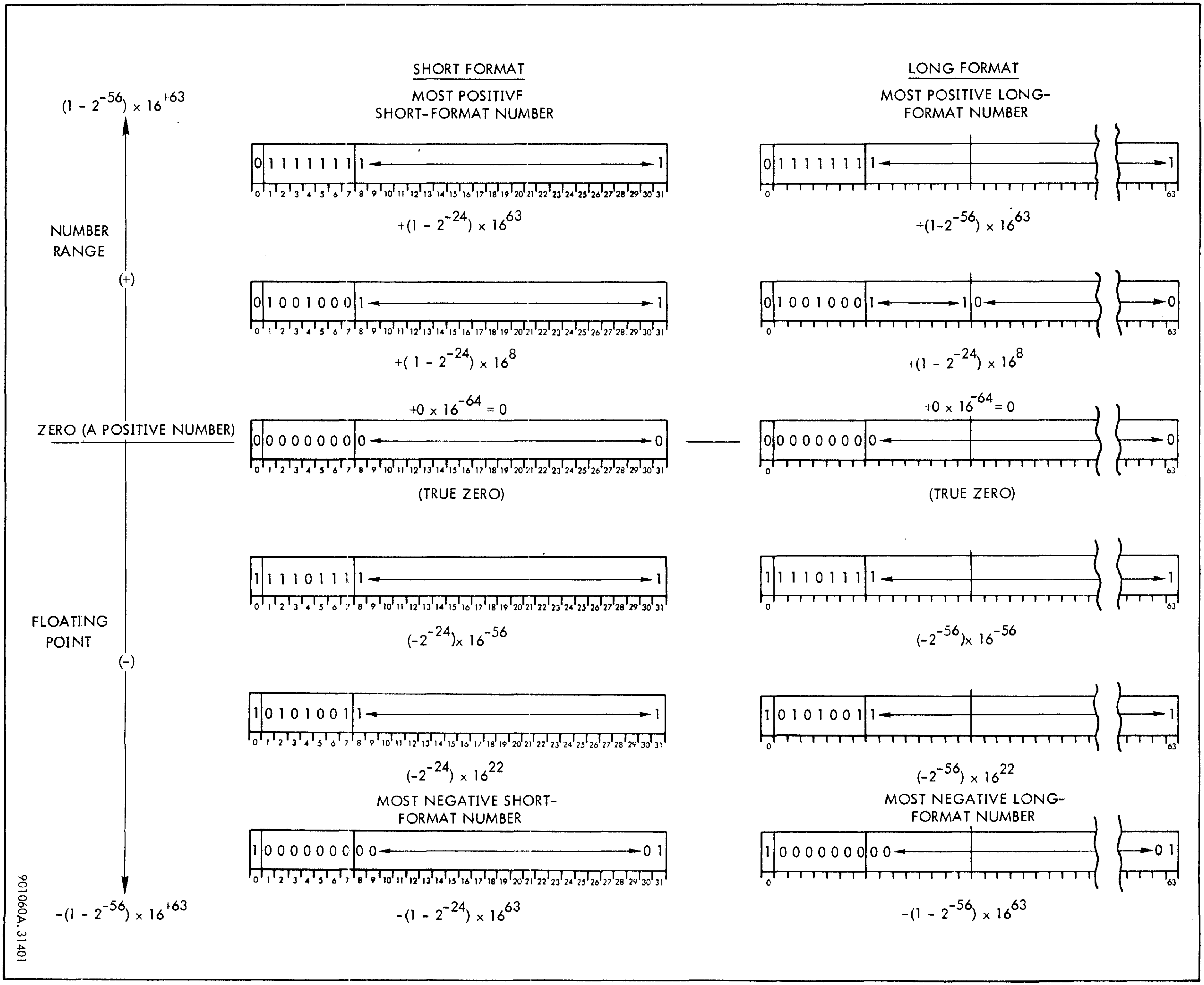
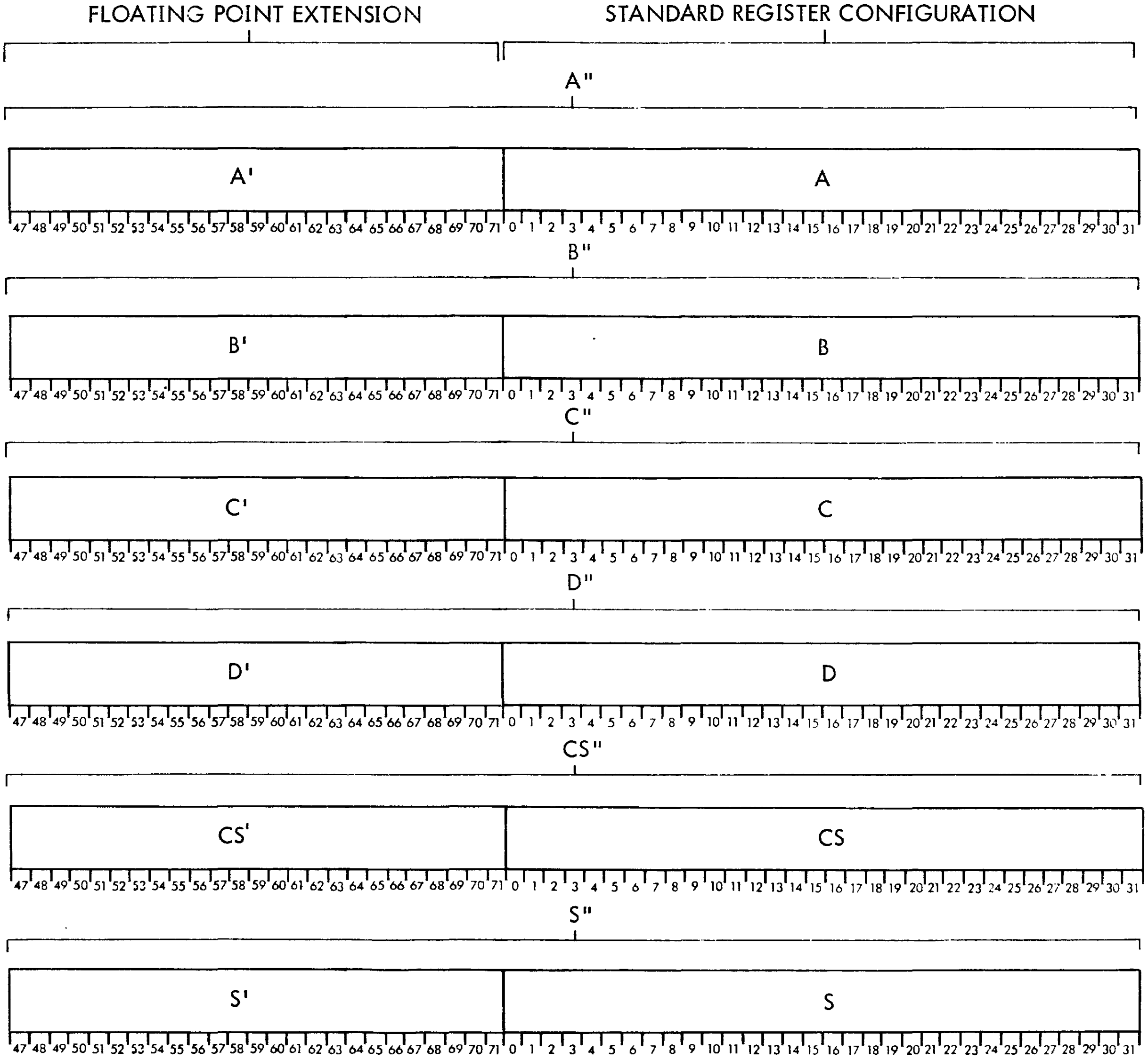

SHORT FORMAT
LONG FORMAT
$(1 - 2^{-56}) \times 16^{+63}$
MOST POSITIVF SHORT-FORMAT NUMBER
MOST POSITIVE LONG-FORMAT NUMBER
0 1 1 1 1 1 1 1 1 ← → 1
$+(1 - 2^{-24}) \times 16^{63}$
$+(1-2^{-56}) \times 16^{63}$
NUMBER RANGE
(+)
0 1 0 0 1 0 0 0 1 ← → 1
0 1 0 0 1 0 0 0 1 ← → 1 0 ← → 0
$+(1 - 2^{-24}) \times 16^{8}$
$+(1 - 2^{-24}) \times 16^{8}$
ZERO (A POSITIVE NUMBER)
$+0 \times 16^{-64} = 0$
0 0 0 0 0 0 0 0 0 ← → 0
(TRUE ZERO)
(TRUE ZERO)
FLOATING POINT
(-)
1 1 1 1 0 1 1 1 1 ← → 1
$(-2^{-24}) \times 16^{-56}$
$(-2^{-56}) \times 16^{-56}$
1 0 1 0 1 0 0 1 1 ← → 1
$(-2^{-24}) \times 16^{22}$
$(-2^{-56}) \times 16^{22}$
MOST NEGATIVE SHORT-FORMAT NUMBER
MOST NEGATIVE LONG-FORMAT NUMBER
1 0 0 0 0 0 0 0 0 0 ← → 0 1
$-(1 - 2^{-24}) \times 16^{63}$
$-(1 - 2^{-56}) \times 16^{63}$
$-(1 - 2^{-56}) \times 16^{+63}$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 63
901060A. 31401

Figure 3-212. Short and Long Floating Point Number Ranges



901060A. 31402

Figure 3-213. Floating Point Register Extensions

3-268 Description of Floating Point Instructions

3-269 FLOATING ADD SHORT (FAS 3D, BD). Instruction FAS adds the effective word from core memory and the contents of an odd or even numbered private memory register. If no floating point arithmetic fault occurs, the sum is stored back in the same odd or even numbered private memory register as a short format floating point number.

3-270 FLOATING ADD LONG (FAL 1D, 9D). Instruction FAL adds the effective doubleword from core memory and the contents of an odd and even numbered private memory register. If no floating point arithmetic fault occurs, the sum is stored back in the same two registers as a long format floating point number. The R-field of the instruction word must be an even value for a correct result.

3-271 FLOATING SUBTRACT SHORT (FSS 3C, BC). Instruction FSS is identical to instruction FAS except that the effective word from core memory is subtracted from the contents of the private memory register.

3-272 FLOATING SUBTRACT LONG(FSL 1C, 9C). Instruction FSL is identical to instruction FAL except that the effective doubleword from core memory is subtracted from the contents of the odd and even numbered private memory registers.

3-273 FLOATING MULTIPLY SHORT (FMS 3F, BF). Instruction FMS multiplies the effective word from core memory by the contents of an odd or even numbered private memory register, after effective prenormalization of both operands. The product is postnormalized. If the R-field of the instruction word is an even value and if there is no fault condition, the product is loaded into the odd and even numbered private memory registers as a long format floating point number. If the R-field is an odd value and there is no fault condition, the product is loaded into the odd numbered private memory register as a truncated short format floating point number.

3-274 FLOATING MULTIPLY LONG (FML 1F, 9F). Instruction FML multiplies the effective doubleword from core memory by the contents of the odd and even numbered private memory registers after prenormalization of both operands. The product is postnormalized and, if no floating point arithmetic fault occurs, the product is truncated to a long format floating point number and is loaded into the odd and even numbered private memory registers. The R-field of the instruction word must be an even value for correct results.

3-275 FLOATING DIVIDE SHORT (FDS 3E, BE). Instruction FDS divides the contents of an odd or even numbered private memory register by the effective word from core memory after prenormalization of both operands. If no floating point arithmetic fault occurs, the quotient is loaded back into the odd or even numbered private memory register as a short format floating point number.

3-276 FLOATING DIVIDE LONG (FDL 1E, 9E). Instruction FDL divides the contents of the odd and even numbered private memory registers by the effective doubleword from core memory after prenormalization of both operands. If no floating point arithmetic fault occurs, the quotient is loaded back into the same two registers as a long format floating point number. The R-field of the instruction word must be even for the correct result.

3-277 FLOATING POINT CONDITION CODES AND MODE CONTROL BITS. Three mode control bits, FS (floating significance), FN (floating normalize), and FZ (floating zero), in the program status doubleword, are used to qualify floating point operations. The floating mode control bits and condition code settings are described under the discussion of the individual floating point operation codes.

3-278 Floating Point Concepts

3-279 SHIFTING FLOATING POINT NUMBERS. Shifting floating point numbers is accomplished by adjusting both the fraction and the exponent of the floating point number. Changing the exponent by one changes the size of the number by a factor of 16 (2^4). The minimum shift in the fraction of a floating point number, consequently, is four bit positions (one hexadecimal place, shortened to one hex). A right shift of one hex necessitates adding one to the exponent in addition to shifting the fraction; a left shift of one hex necessitates subtracting one from the exponent. Each successive shift of one changes the exponent by one bit. A right shift is usually used during overflow adjustment and is used to equalize exponents of two numbers. A left shift is usually used during normalization procedures. Examples of floating point shifts are shown in figure 3-214.

In the following discussion of floating point operation codes, shifting will always refer, unless otherwise noted, to shifting the fraction of the floating point number one hexadecimal place (four bit positions) and changing the exponent by one bit (one power of 16).

3-280 GENERAL NORMALIZATION. A floating point number is normalized if the absolute value of the fraction is less than one but is equal to or is greater than 1/16. The fraction of a positive floating point number must have a one somewhere in the four most significant bits for the number to be normalized. The fraction of a negative floating point number is in two's complement form. A negative floating point number, therefore, must have a zero somewhere in the four most significant bits or have all ones in the four most significant bits and all zeros in the remaining digits. The following lists all possible cases.

```
  Normalized Positive               Normalized Negative
 Floating Point Numbers            Floating Point Numbers

0 | xxxxxxx | 0001xx←→x        1 | xxxxxxx | 111100←→0
                               1 | xxxxxxx | 1110xx←→x
0 | xxxxxxx | 0010xx←→x
       .          .            1 | xxxxxxx | 1101xx←→x
       .          .                   .
       .          .                   .
0 | xxxxxxx | 1111xx←→x        1 | xxxxxxx | 0000xx←→x
```

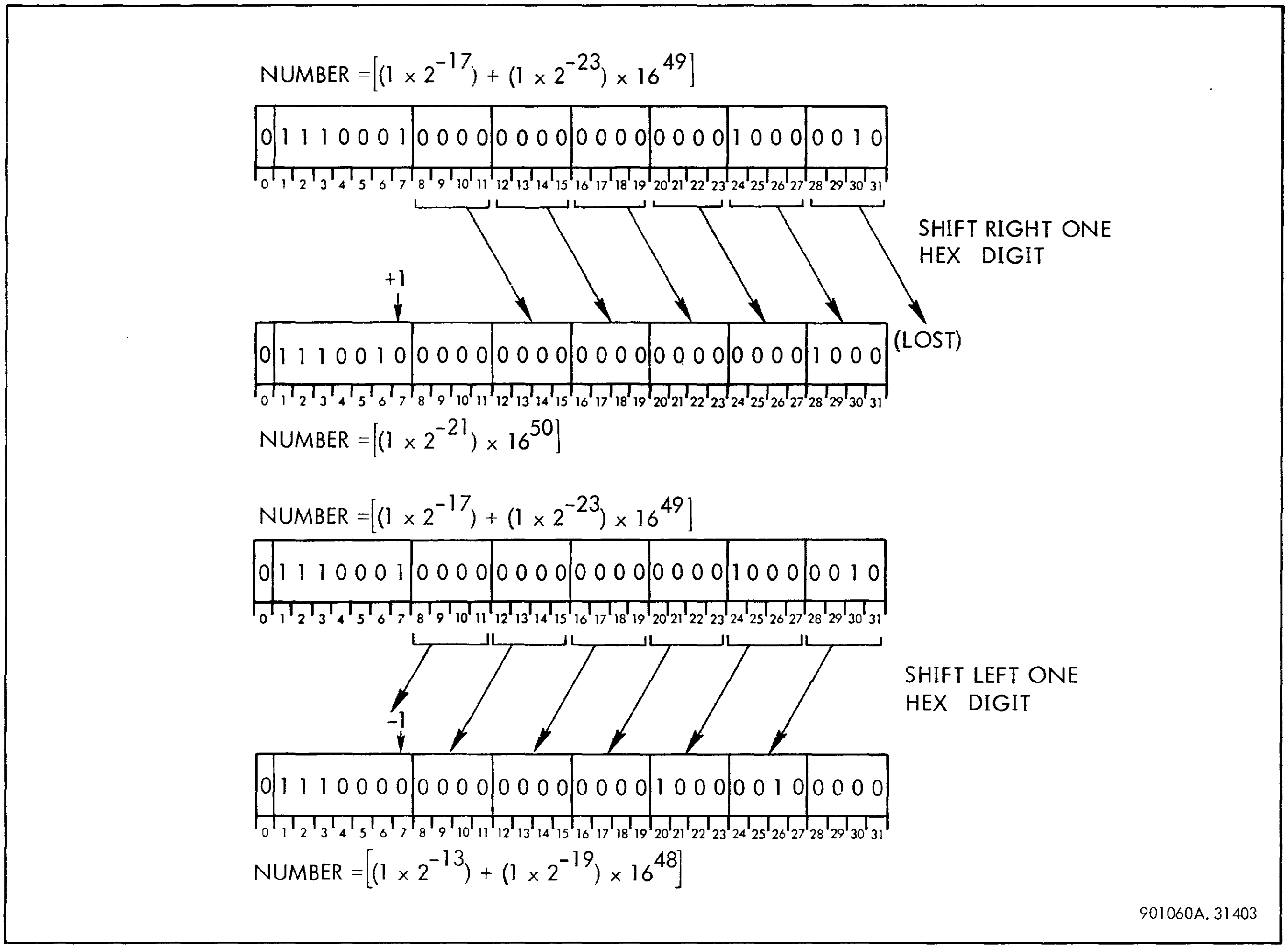


Figure 3-214. Shifting Floating Point Numbers

The negative floating point number 1xxxxxxx0000←→0 is illegal, since the absolute value of the fraction is greater than the number of bit positions allocated for the fraction. This number is always changed to an equal number whose exponent is one greater than the original and whose fraction is 1/16 of the original.

$$1 \mid 0000010 \mid 0 \longleftrightarrow 0 \quad = \quad -1 \times 16^{61}$$

$$1 \mid 0000001 \mid 11110 \longleftrightarrow 0 \quad = \quad -1/16 \times 16^{62}$$

$$-1 \times 16^{61} \quad = \quad -1/16 \times 16^{62}$$

Normalization of floating point numbers in implementation of floating point instructions may occur before the arithmetic operation takes place (prenormalization) or after the arithmetic operation (postnormalization). Normalization is usually performed on absolute numbers. Both prenormalization and postnormalization are accomplished by left shifting operations. Left shifting continues until the absolute value of the fraction is greater than or is equal to 1/16. Normalization is illustrated in figure 3-215.

3-281 SIMPLE NORMALIZATION. Valid floating point numbers in the Sigma 7 memory are numbers whose fractions are in the range $1/16 \leq |N| < 1$. In the hardware, however, the computer tolerates another range of floating point numbers while arithmetic operations are performed. Floating point number, valid in the execution of an instruction, may be in the ranges $+1/16 \leq N < 1$ and $-1 \leq N < -1/16$. Numbers in these ranges are called simple normalized numbers.

3-282 TRUNCATION. A floating point number may be truncated when it is necessary to discard some of the least significant bits of the number. Multiplication of two long format floating point numbers, for example, produces a fraction product of 112 bits, of which it is convenient to retain only 56.

Truncation takes the place of rounding off a number and is accomplished simply by discarding the unwanted bits. The error in the truncated number may range from 0 (when 0.00100000 is truncated to 0.0010, for example) to almost one bit of the lowest order in the truncated number (when 0.00101111 is truncated to 0.0010). Since it is more

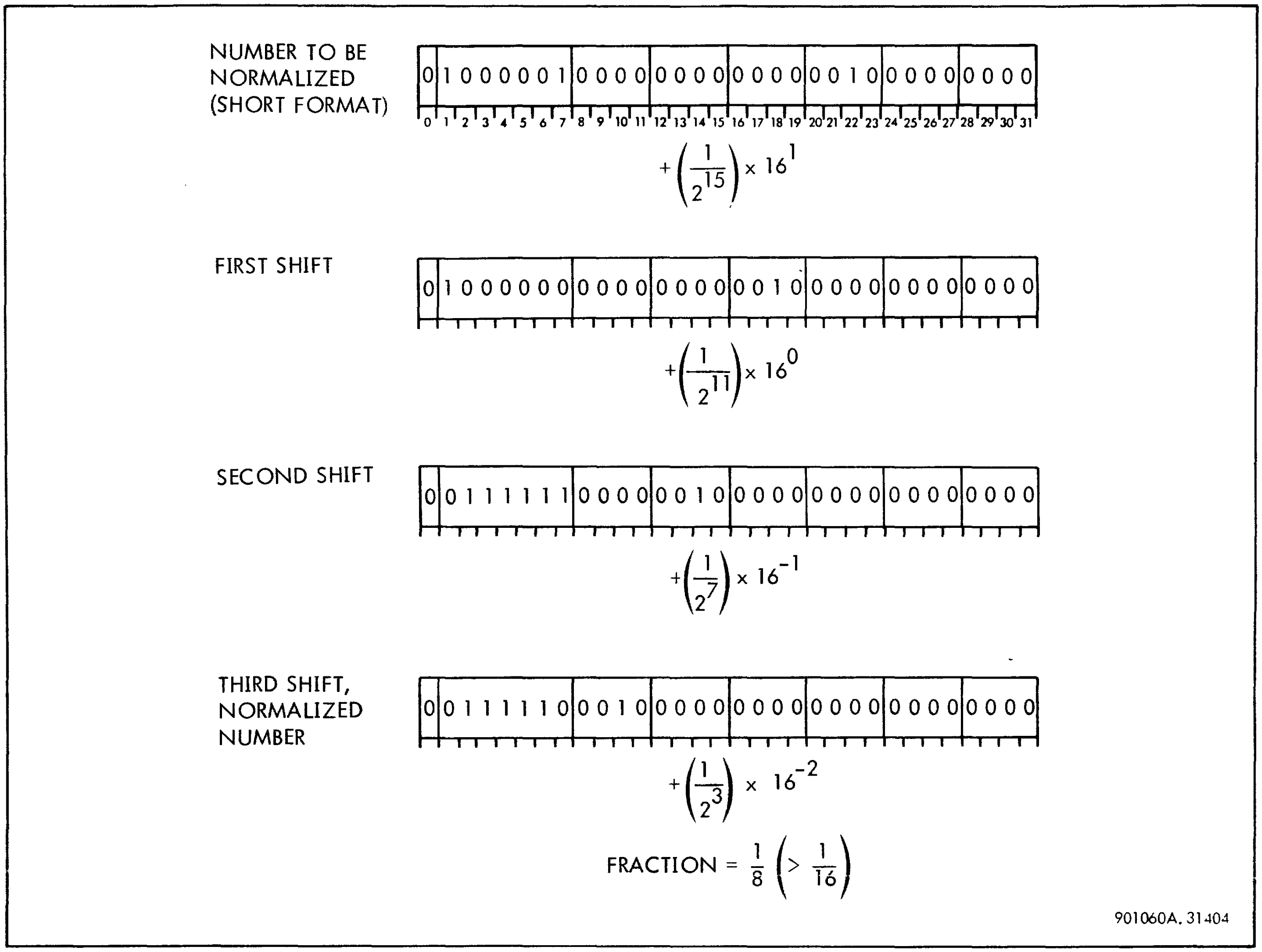


Figure 3-215. Normalization of Floating Point Numbers

accurate to truncate absolute values than negative numbers, negative numbers are always put in the absolute form in the Sigma 7 operations where truncation may occur.

3-283 OVERFLOW AND UNDERFLOW. There are two types of overflow in floating point operations: fraction (mantissa) and exponent (characteristic). When fraction overflow occurs, the fault is easily corrected by shifting the absolute value of the floating point number one hex to the right. There may be some error in the adjustment process because of truncation. Fraction overflow is illustrated in figure 3-216.

When exponent overflow occurs during any floating point instruction, a trap to memory location X'44' (68) always occurs.

Underflow occurs if the final result of an addition, subtraction, multiplication, or division cannot be postnormalized because the result is smaller than the smallest possible quantity which can be represented in the computer. Underflow may cause a trap to location X'44' (68); it may cause a setting of the result to true zero; or it may be ignored, depending on the floating point mode control bits.

3-284 Floating Point Addition and Subtraction (FAFLAS)

3-285 GENERAL. The implementation of addition and subtraction of floating point numbers is nearly identical since, for example, subtracting (A) from (B) is identical to adding (-A) to (B). Operations with short and long format numbers are also very similar; there are only differences in the number of digits to be operated upon. Because of the similarities between operations and format, all FAFLAS operation codes (FAS, FAL, FSS, FSL) are discussed as a group, with differences between them noted in the discussion. The floating point number which is operated upon is called the augend whether it is the actual augend (addition) or the minuend (subtraction), and the floating point number which is added or is subtracted is called the addend (whether

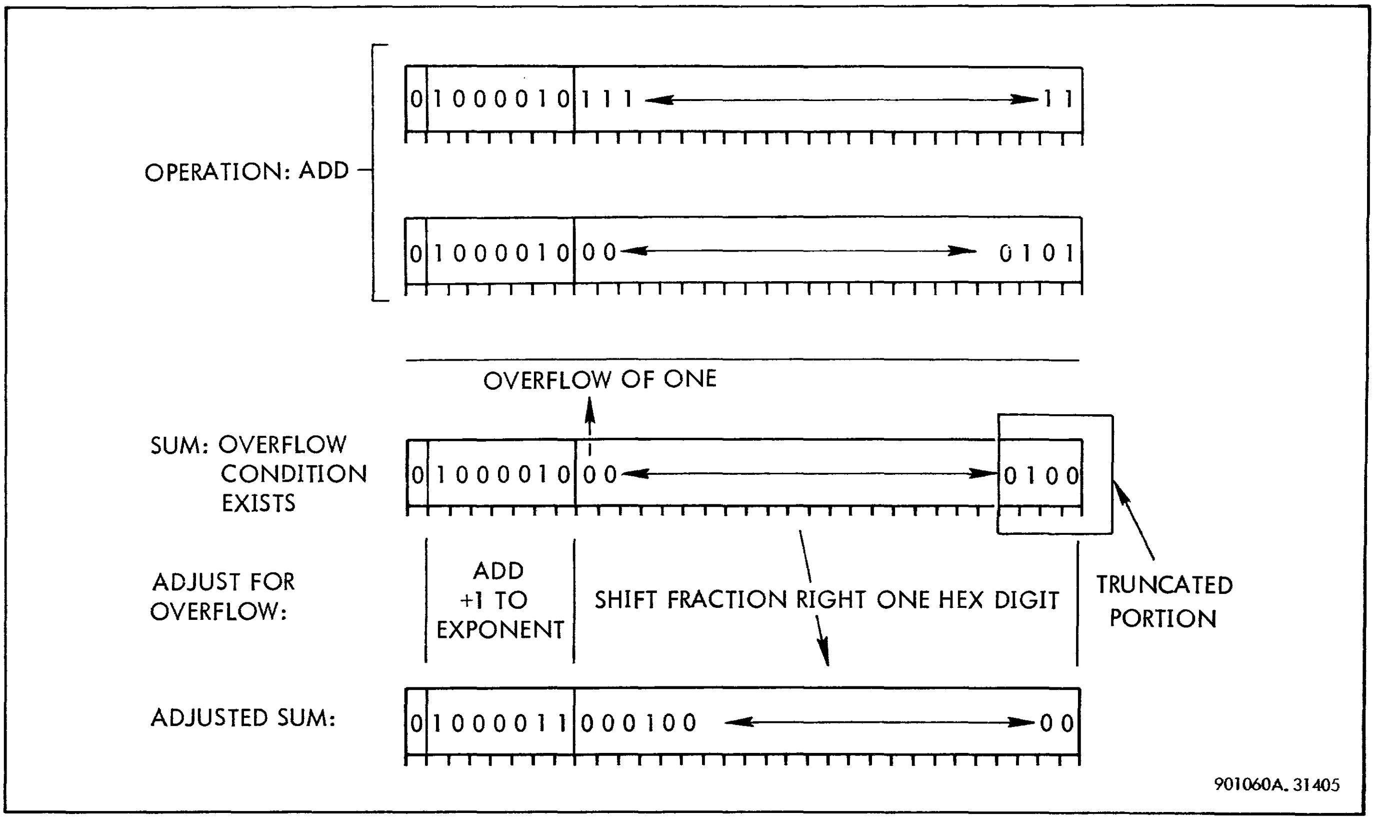


Figure 3-216. Fraction Overflow and Adjustment

it is the addend or subtrahend). The result of the operation is always called the sum.

3-286 FLOATING POINT MODE CONTROL BITS. The three floating point mode control bits, FN, FZ, and FS, determine the operation of floating point addition and subtraction. The three are interrelated and the setting of one bit may affect the other two.

If mode control bit FN, floating normalize, equals zero, the results of the addition or subtraction are postnormalized. If underflow occurs, result is zero, or if more than two postnormalization shifts (4 bit positions each) are required, the settings for FZ and FS determine the resultant action. If FN = 1, postnormalization of the result is inhibited, and the settings of FZ and FS have no effect on the instruction operation.

FZ, floating zero, is effective only when FN = 0. If FN = 0 and FZ = 0, underflow causes the results to be set equal to true zero. The one exception to this is if FS = 1 and a trap occurs. In this case, the underflow generated in the process of postnormalization is ignored. If FN = 0 and FZ = 1, underflow causes the result to trap to location X'44' (68).

FS, floating significance, is valid only when FN = 0. If FN = 0 and FS = 1, and if more than two postnormalizing shifts are required or if the result is zero, the computer traps to location X'44' (68). If FN = 0 and FS = 0, the significance trap is inhibited. If the result of an addition or subtraction is zero, the result is set equal to true zero, and no trap occurs. No trap occurs if more than two postnormalizing shifts are required and underflow does not occur.

3-287 EXPONENT OVERFLOW. If exponent overflow occurs, the computer always traps to location X'44' (68).

3-288 CONDITION CODE SETTINGS. The condition code settings and their meanings for floating point addition and subtraction are shown in table 3-131.

3-289 BASIC STEPS IN FAFLAS IMPLEMENTING. To add or to subtract two floating point numbers in Sigma 7, the following basic steps are performed:

a. Transfer of operands (PH1 and PH3): The augend (from private memory) and the addend (from core memory) are transferred to the arithmetic unit registers.

b. Exponent differencing (PH2 and PH3): The uninverted biased addend exponent is subtracted from the uninverted biased augend exponent.

c. Equalization of exponents (PH4 through PH8): The exponent difference is examined.

Table 3-131. Condition Codes for Floating Point Addition and Subtraction

CONDITION CODE				IF NO TRAP TO LOCATION X'44' OCCURS	IF TRAP TO LOCATION X'44' OCCURS
1	2	3	4		
0	0	0	0	-A + A with FN = 1, result set to true zero	Impossible
0	0	0	1	Result negative	
0	0	1	0	Result positive and nonzero	
0	1	0	0	Impossible	
0	1	0	1		Overflow, result negative
0	1	1	0		Overflow, result positive and nonzero
1	0	0	0	-A + A, result set to true zero	-A + A
1	0	0	1	Result negative, more than two postnormalizing shifts, FS = 0, FN = 0, no underflow	Result negative, more than two postnormalizing shifts, FS = 1, FN = 0, no underflow with FZ = 1
1	0	1	0	Result positive and nonzero, more than two postnormalizing shifts, FS = 0, FN = 0, no underflow	Result positive and nonzero, more than two postnormalizing shifts, FS = 1, FN = 0, no underflow with FZ = 1
1	1	0	0	Underflow with FZ = 0, no trap by FS = 1, result set to true zero	Impossible
1	1	0	1	Impossible	Underflow, result negative, FZ = 1
1	1	1	0		Underflow, result positive and nonzero, FZ = 1

1. If the difference is zero, the exponents of the augend and the addend are the same and the two floating point numbers can be added or subtracted.

2. If the difference is not zero, the exponents of the augend and the addend are different, and the smaller floating point number must be adjusted (shifted right) so that its exponent equals the exponent of the other. When the exponents of both floating point numbers are equal or when the fraction of the smaller number is found to be zero, the numbers are ready for addition or subtraction.

d. Addition (PH9): The augend and addend fractions are added. The exponent of the floating point number that was not adjusted (originally the larger number) is unbiased by subtracting 64 from it.

e. Overflow detection and postnormalization (PH12): The sum of the fractions is changed to its absolute value and examined.

1. If fraction overflow has occurred, the absolute value of the sum is shifted one hex to the right to correct the overflow condition.

2. If fraction overflow has not occurred, if the sum is not simple normalized, and if FN = 0, the absolute value of the sum is shifted one hex to the left for the first try at normalization.

3. The sum is left unchanged, if fraction overflow has not occurred, and if the sum is simple normalized or if normalization is to be inhibited (FN = 1).

f. Postnormalization (PH13): The adjusted sum is examined.

1. The result is ready for storage if the sum is normalized or if it is not normalized and FN = 1, or if the sum is zero. Sixty-four is added to the exponent to bias it once again.

2. If the sum is not normalized, if FN = 0, and if the result is not zero, the sum is shifted left one hex at a time until it is normalized. Step 1 above is then performed.

g. Storage (PH13 and PH15): The biased exponent and the fraction result are assimilated and are stored in memory in proper form if there is no trap condition.

An example of FAFLAS implementation is shown in figure 3-217. The lettered steps correspond to the basic steps discussed.

3-290 DETAILED STEPS IN FAFLAS IMPLEMENTATION. Figure 3-21E shows the FAFLAS phases and the operations performed for each phase. The following discussion is supplemented by table 3-132 and by figure 3-218.

The preparation sequence is the same as the general preparation sequence.

a. PH1. At the PH1 clock, the most significant word (MSW) of the augend is transferred to the A-register from the even numbered private memory register by signal AXRR. The MSW of the addend has already been transferred to the C-register from core memory during the preparation phases. Flip-flop RN is set at the PH1 clock if the augend is negative; flip-flop MWN is set if the addend is negative. Flip-flop CXCL32 is set for use in PH2. Flip-flops CC1 and CC2 are reset for underflow and overflow tests in PH15, and flip-flop IEN is set to start interruptibility. Ones are forced into flip-flops CS0 through CS7 by signal CSX1/5 to invert the augend exponent to its true form if the augend is negative.

S/CXCL32 = FAFL PH1 + ...

R/CC1 = FAFL PH1 + ...

R/CC2 = FAMDSF PH1 NFAMULH + ...

b. PH2. At the PH2 clock, the fraction part of the augend MSW (or the complete fraction if FAS or FSS is being performed) is transferred by signal AXAL32 from flip-flops A0 and A8 through A31 to A47 and A48 through A71, respectively. The fraction part of the addend MSW is transferred to the C-register extension (C') in the same manner by signal CXCL32. The augend and addend exponents are set up in PH2 for exponent differencing. The uninverted (absolute) augend exponent either is present in flip-flops A1 through A7 or is found by combining bits A1 through A7 with bits CS0 through CS7 (all ones) in an exclusive OR operation. The uninverted augend exponent is gated to the sum bus during PH2 and, at the PH2 clock, is transferred from the sum bus to two locations: flip-flops A0 through A7 and B0 through B7 by signals AXS/1 and BXS, respectively. Flip-flops B0 through B7 retain the uninverted exponent for later use.

The uninverted addend exponent must be subtracted in exponent differencing. The inverted addend exponent is therefore transferred to flip-flops D0 through D7 (signals DXC/6 or DXNC). A one is forced into flip-flop CS7 by signal CS7X1 in preparation for exponent differencing. (The contents of flip-flops A0 through A7, D0 through D7, and CS7 are later added to find the difference of exponents.)

Ones are forced on core memory address line LB31 and on private memory line LR31 for possible use in PH3 (with FAL or FSL). If FAL or FSL are being implemented, signal MRQ (memory request) is enabled and flip-flop DRQ (data release) is set to transfer the least significant word (1SW) of the augend to the C-register at the PH3 clock. Clock T6RL is enabled for PH3 if FAL is being performed. If FAS or FSS is being implemented, flip-flop CX/1 is set at the PH2 clock to force all zeros into the C-register in PH3.

c. PH3. During PH3, the uninverted exponent augend (in flip-flops A0 through A7) and the two's complemented addend exponent (in flip-flops D0 through D7 and CS7) are added, and the result gated to the sum bus. At the PH3 clock, the sum bus outputs on S0 through S7 are transferred to flip-flops E0 through E7 by signal EXS. Flip-flops E0 through E7 now hold the difference of exponents, which may be zero, positive, or negative. E0 is the sign bit of the difference. If E0 is a zero, the difference is positive or zero and is in true form. If E0 is a one, the difference is negative and in two's complemented form.

If FAL or FSL is being implemented, the 1SW of the augend is clocked into flip-flops A0 through A31 from the odd numbered private memory register by signal AXRR. The 1SW of the addend is clocked into C0 through C31. If FAS or FSS is in effect, zeros are clocked into flip-flops A0 through A31 (no gating term is present) and are forced into C0 through C31 (signal CX/1 is set in PH2).

d. PH4 through PH8. Phases 4 through 8 are the phases in which the exponents of the augend and the addend are compared and are equalized, if necessary.

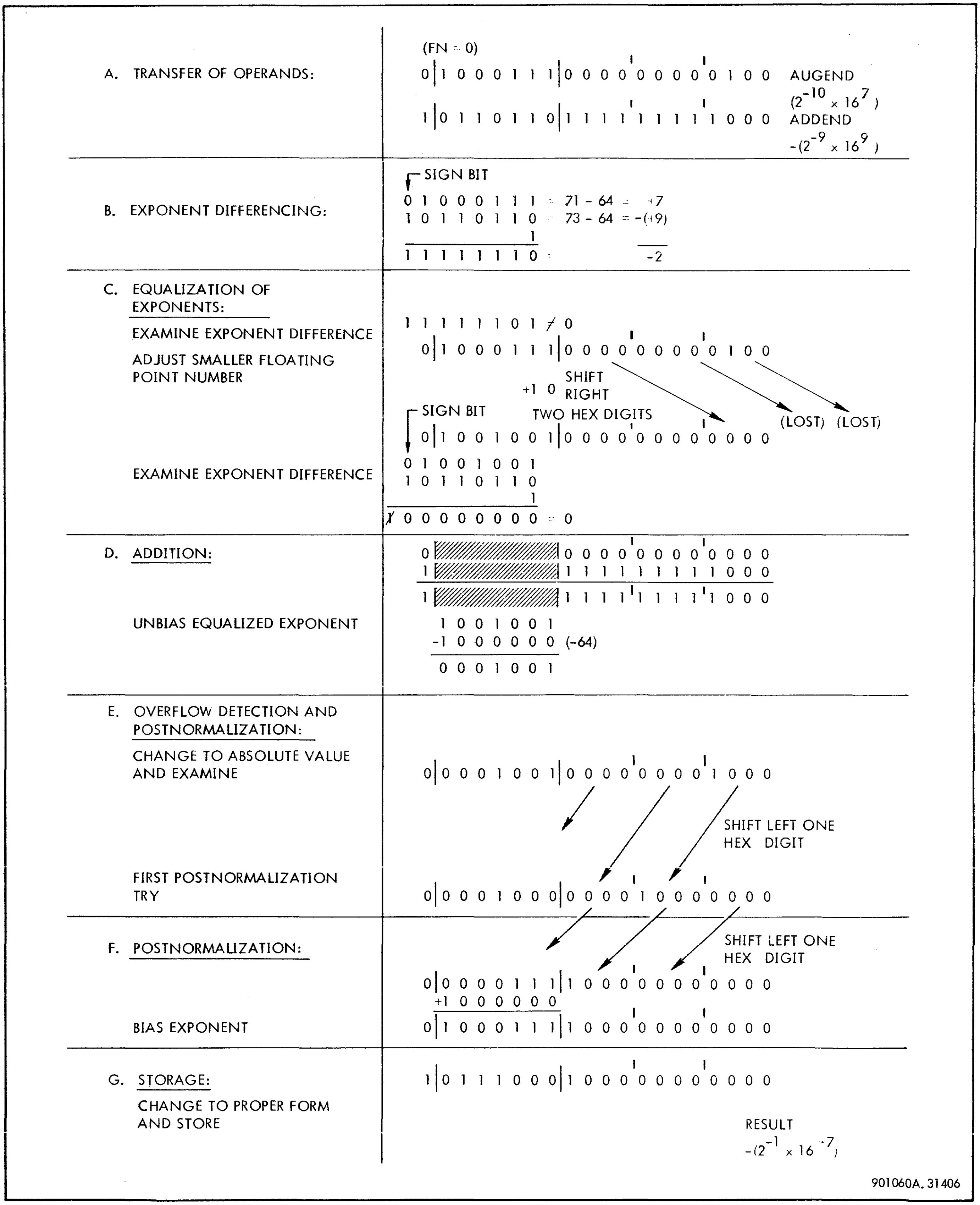


Figure 3-217. Example of FAFLAS Implementation (Floating Add)

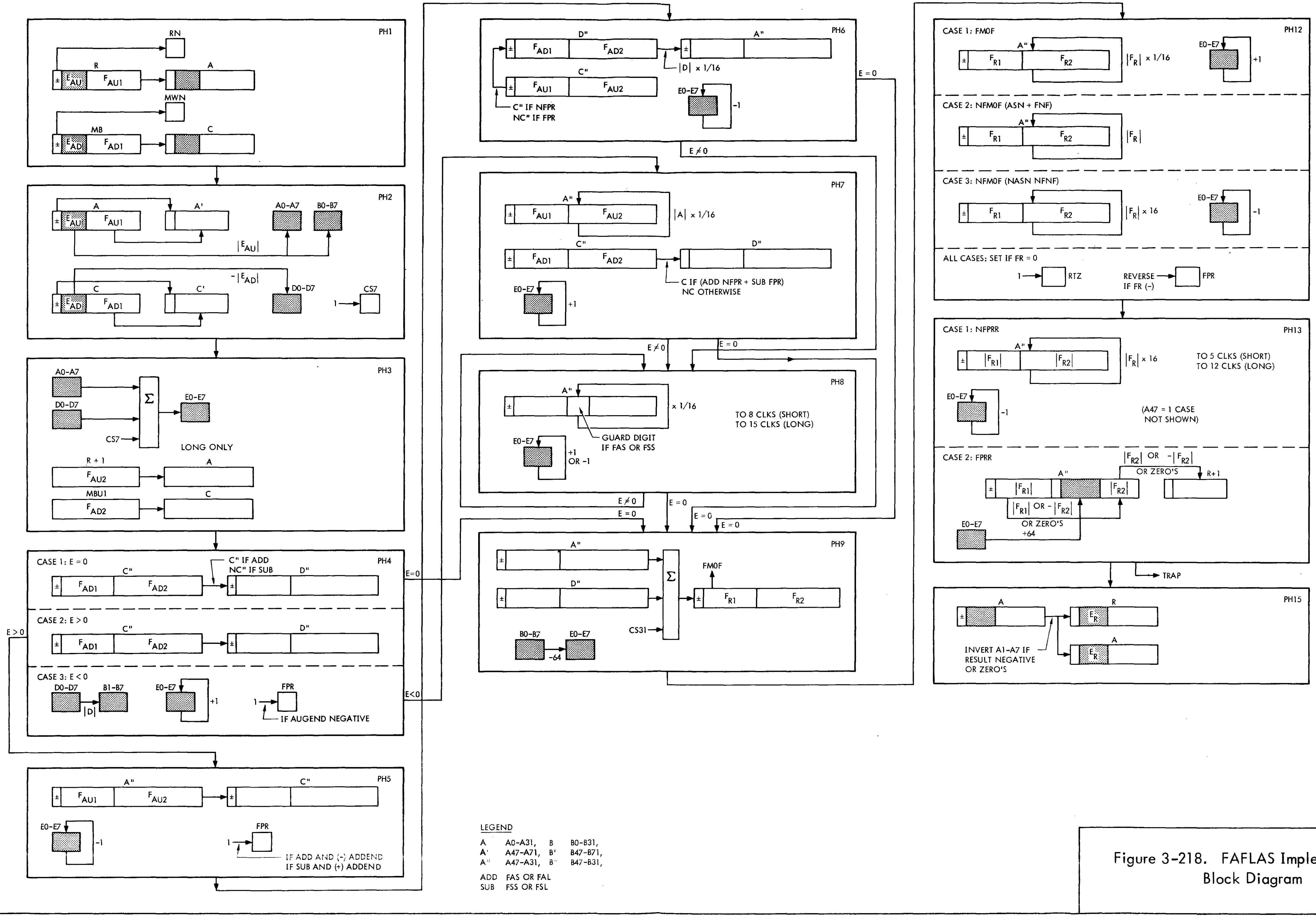


Figure 3-218. FAFLAS Implementation, Block Diagram

901060A. 31407

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence		
PH1 T4RL	RR0-RR31 -/→ A0-A31	AXRR = RNRXRR0/2 PH1 + ... = FAFL PH1 + ...	MSW of augend -/→ A
	Set flip-flop RN if augend negative	S/RN = RR0 RNRXRR0 + ... = RR0 PH1 FAFL + ...	Stores augend sign
	MB0-MB31 -/→ C0-C31	Preparation control	MSW of addend -/→ C
	Set flip-flop MWN if addend negative	S/MWN = C0C16 FAMDSF PH1 + ... = MB0 NP32 CXMB + ...	Stores addend sign
	Set flip-flop CXCL32	S/CXCL32 = FAFL PH1 + ...	For PH2 use
	Reset flip-flops CC1 and CC2	R/CC1 = FAFL PH1 + ...	For PH15 use
		R/CC2 = FAMDSF PH1 NFAMULH + ...	
	Force ones into CS0-CS7	CSX1/5 = FAFL PH1 + ...	To invert the augend exponent if augend negative
	Set flip-flop IEN	S/IEN = FAMDSF NFAMUL PH1 + ...	Starts interruptibility
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH2 T6L	A0, A8-A31 -/→ A47, A48-A71	AXAL32 = FAFL PH2	MSW of augend -/→ A'
	C0, C8-C31 → C47, C48-C71	CXCL32 = Set in PH1	MSW of addend → C'
	A0-A7 + CS0-CS7 → S0-S7 if augend negative	SXPR = FAFL PH2 RN + ...	
	or		
	A0-A7 → S0-S7 if augend positive	SXA = FAFL PH2 NRN + ...	
	S0-S7 { -/→ A0-A7 -/→ B0-B7	AXS/1 = FAFL PH2 + ... = FAFL PH2 N(FAFLM NO2)	Uninverted augend exponent
	C0-C7 -/→ D0-D7 if addend negative	DXC/6 = FAFL PH2 NMWN + ...	Inverted addend exponent
	or		
	NC0-NC7 -/→ D0-D7 if addend positive	DXNC = FAFL PH2 NMWN + ...	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)	Force a one into CS7	CS7X1 = FAFL PH2 NFAFLM	For two's complement subtraction (PH3)
	Force a one onto LB31 address line	S/LB31/1 = FAFL PH2 + ...	Selects LSW of addend
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = FAFL PH2 + ...	Selects LSW of augend
	Enable MRQ if FAL or FSL	MRQ = FAFL PH2 NO2 + ...	
	Set flip-flop DRQ if FAL or FSL	S/DRQ = FAFL PH2 NO2 + ...	Inhibits transmission of another clock until data release signal is received
	Set flip-flop CX/1 if FAS or FSS	S/CX/1 = FAFL PH2 O2 + ...	To clear C0-C31 in PH3
	Enable clock T6RL if FAL or FSL	T6RL = FAFL PH2 NO2 + ...	
PH3 T6L if short T6RL if long	A0-A7 + D0-D7 + CS7 → S0-S7 —/→ E0-E7	SXK/1 = FAFL PH3 + ...	Unbiased exponent difference —/→ E0-E7
		SXPR/1 = FAFL PH3 + ...	
		EXS = FAFL PH3 + ...	
	RR0-RR31 —/→ A0-A31 } if long	AXRR = FAFL PH3 N(FAFLM ASN) NO2 + ...	LSW of augend —/→ A
	MB0-MB31 → C0-C31 } if long	See PH2	LSW of addend → C
	Clear A47-A31	AX/1 = FAFL PH3 + ...	A47-A71 remains the same
	S47-S71 —/→ A47-A71	AXS/4 = FAFL PH3 N(FAFLM ASN) + ...	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	
PH4 T6L	Case 1: E0-E7 = 0		
	C47-C71 —/→ D47-D71 if FAS or FAL	DXC/6 = FAFLAS PH4 EZ O7 + ...	
	NC47-NC71 —/→ D47-D71 } if FSS	DXNC/1 = FAFLAS PH4 EZ NO7 + ...	
	Force 1 into CS31 } if FSS or FSL	DXNC/1 = Same	For two's complement
	Set flip-flop T8L	R/NT8L = T8L = FAFLAS PH4 EZ + ...	
	Enable PH9	BRPH9 = FAFLAS PH4 EZ + ...	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T6L (Cont.)	Case 2: E0-E7 > 0		
	C47-C31 —/→ D47-D31	DXC/6 = FAFLAS PH4 NE0 NEZ + ...	
	Set flip-flop CXS	S/CXS = FAFLAS PH4 NE0 NEZ + ...	For PH5 use
	Case 3: E0-E7 < 0		
	D1-D7 —/→ B1-B7	BXND = FAFLAS PH4 E0	Augend exponent —/→ B1-B7
	Clear D-register	DX/1 = FAFLAS PH4 + ...	
	Increment E-register by one	PCTE1 = FAFLAS PH4 E0 + ...	To record PH7 shift
	Set flip-flop NGX } if augend negative	S/NGX = FAFLAS PH4 E0 RN + ...	For PH7 use
	Set flip-flop FPR	S/FPR = FAFLAS PH4 E0 RN + ...	Result must be reversed
	Set clock T8L	R/NT8L = T8L = FAFLAS PH4 E0 + ...	
	Enable PH7	BRPH7 = FAFLAS PH4 E0 + ...	
PH5 T6L	(Entered from PH4, case 2)		
	A47-A31 → S47-S31	SXA = FAFL PH5 + ...	Augend fraction → S47-S31
	S47-S31 → C47-C31	CXS = Set in PH4	
	Clear A-register	AX/1 = FAFLAS PH5 + ...	
	Set flip-flop NGX if addend negative	S/NGX = FAFLAS PH5 MWN + ...	
	Decrement E-register by one	MCTE1 = FAFLAS PH5 + ...	To record PH6 shift
	Set FPR if absolute value of the addend is opposite to form required for addition or subtraction	S/FPR = FAFLAS PH5 (MWN O7 + NMWN NO7) + ...	Result must be reversed
	Set clock T8L	R/NT8L = T8L = FAFLAS PH5 + ...	
PH6 T8L	(Entered from PH5)		
	| D47-D31 | → S47-S31	SXADD = FAFLAS PH6 + ...	Shift addend to the right one hex
	S47-S31 x 1/16 —/→ A51-A31	AXSR4 = FAFLAS PH6 + ...	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH6 T8L (Cont.)	C47-C71 —/→ D47-D71 if NFPR	DXC/6 = FAFLAS PH6 NFPR + ...	Augend D", sign reversed if sign of addend was reversed
	or		
	NC47 -NC71 —/→ D47-D71 } if FPR	DXNC/1 = FAFLAS PH6 FPR + ...	
	Force one into CS31	DXNC/1 = Same	For two's complement
	Decrement E-register by one	MCTE1 = FAFLAS PH6 + ...	To record next shift
	If E0-E7 is zero, enable PH9 and set flip-flop T8L	BRPH9 = FAFLAS PH6 EZ + ...	
	or	R/NT8L = T8L = FAFLAS PH8 EZ + ...	
	If E0-E7 is not zero, enable PH8	BRPH8 = FAFLAS PH6 NEZ + ...	
PH7 T8L	(Entered from PH4, case 3)		
	|A47-A31| → S47-S31	SXADD = FAFLAS PH7 + ...	Shift absolute value of augend one hex to the right
	S47-S31 x 1/16 —/→ A51-A31	AXSR4 = FAFLAS PH7 + ...	
	Increment E-register by one	PCTE1 = FAFLAS PH7 + ...	
	C47-C31 —/→ D47-D31 if performing addition with NFPR or subtraction with FPR	DXC/6 = FAFLAS PH7 (O7 ⊕ FPR) + ...	Change addend in same sense as augend was changed
	or		
	NC47-NC31 —/→ D47-D31 if not the above	DXNC/1 = FAFLAS PH7 N(O7 ⊕ FPR) + ...	
	If E0-E7 is zero, enable PH9 and set flip-flop T8L	BRPH9 = FAFLAS PH7 EZ + ...	
		R/NT8L = T8L = FAFLAS PH7 EZ + ...	
PH8 T6L	(Entered from PH6 or PH7)		
	0-8 clocks if FAS or FSS 0-15 clocks if FAL or FSL		
	A47-A71 → S47-S71 if FAS or FSS	SXA/3 = FAFLAS PH8 O2 + ...	Zeros to S0-S31
	A47-A31 → S47-S31 if FAL or FSL	SXA = FAFLAS PH8 NO2 + ...	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH8 T6L (Cont.)	S47-S31 x 1/16 $\not\longrightarrow$ A47-A31	AXSR4 = FAFLAS PH8 + ...	Shift augend or addend right one hex. A0-A3 is guard digit for FAS and FSS
	Increment E-register by one if E is negative	PCTE1 = FAFLAS PH8 E0 + ...	Count towards zero
	or		
	Decrement E-register by one if E is positive	MCTE1 = FAFLAS PH8 NEZ	
	Retain state of CS31 for PH9	R/CS31 = N(FAFLAS PH8) + ...	Remains set for two's complement
	Sustain PH8 if E0-E7 is nonzero and if A47-A31 is nonzero	BRPH8 = FAFLAS PH8 NEZ NA4731Z + ...	
	or		
	Advance to PH9 if not above and set flip-flop T8L	R/NT8L = T8L = FAFLAS PH8 EZ + ...	
PH9 T8L or T6L	(Entered from PH4 case 1, PH6, PH7, or PH8)		
	A47-A31 + D47-D31 + CS31 $\longrightarrow$ S47-S31	SXADD = FAFLAS PH9 + ...	
	S47-S31 $\not\longrightarrow$ A47-A31	AXS = FAFLAS PH9 NFMOF (ASN + FNF)	Fraction sum $\longrightarrow$ A47-A31 (Inverted if FPR). Does not include case 1
	Set flip-flop FMOF if fraction overflow exists	S/FMOF = FAFLAS PH9 (A47 D47 NK47 + NA47 ND47 K47)	
	(B1-B7) -64_{10} $\not\longrightarrow$ E0-E7	EXB = FAFLAS PH9 + ...	Unbiased exponent of sum $\not\longrightarrow$ E0-E7
	Clear B-register	BX/1 = FAFLAS PH9 + ...	
	Clear D-register	DX/1 = FAFLAS PH9 + ...	
	Set flip-flop NGX	S/NGX = FAFLAS PH9 + ...	
	Enable MRQ/1	MRQ/1 = FAFLAS PH9 + ...	Memory request for next instruction

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T8L or T6L (Cont.)	Reset flip-flop IEN	R/IEN = FAFLAS PH9 + ...	Stop interruptibility
	Enable PH12	BRPH12 = FAFLAS PH9 + ...	
	Set flip-flop T8L	R/NT8L = T8L = FAFLAS PH9 + ...	
PH12 T8L	NA47-NA31 + 1 $\longrightarrow$ S47-S31 if sum is negative	SXADD = (FAFLAS PH12) (A47 ⊕ FMOF)	NGX set in PH9
	or		|Sum|
	A47-A31 $\longrightarrow$ S47-S31 if sum is positive	SXA = (FAFLAS PH12) N(A47 ⊕ FMOF)	
	Reverse state of flip-flop if sum is negative	S/FPR = FAFLAS PH12 NFPR (A47 ⊕ FMOF)	Taking the absolute value of the sum has changed the sum
		R/FPR = FAFLAS PH12 (A47 ⊕ FMOF) + CLEAR	
	Case 1: FMOF $(1 \leq \text{sum} < 2$ $-\bar{2} < A$ $< -\bar{1})$		
	S47-S31 x 1/16 $\not\longrightarrow$ A47-A31	AXSR4 = FMOF + ...	Shift |sum| to correct overflow
	Increment E0-E7 by one	PCTE1 = FMOF + ...	
	Force a one into A50 if A47-A31 are all zeros	S/A50 = FMOF A4731Z + ...	
	Case 2: NFMOF (ASN + FNF)		
	S47-S31 $\not\longrightarrow$ A47-A31	AXS = FAFLAS PH9 NFMOF (ASN + FNF) + ...	|Sum| $\longrightarrow$ A"
	Case 3: NFMOF NASN NFNF		
	S47-S31 x 1/16 $\not\longrightarrow$ A47-A31	AXSL4 = FPPN = FAFLAS PH12 NFMOF NASN NFNF NRTZ + ...	|Sum| x 1/16 $\not\longrightarrow$ A"
	Decrement E0-E7 by one	MCTE1 = FPPN + ...	
	Set flip-flop BC31	S/BC31 = FPPN + ...	For postnormalization counting
	Set flip-flop RTZ if sum is zero	S/RTZ = FAFLAS PH12 NFMOF NA4731Z + ...	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating Subtract (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH12 T8L (Cont.)	Clear D-register	DX/1 = FAFL PH12 + ...	
	Set flip-flop NGX	S/NGX = FAFL PH12 + ...	For PH13 use
	Set flip-flop G0003/1 if FAS or FSS	(S/G0003/1) = FAFL PH12 NFPRD	Causes K71 in PH13
	Set flip-flop T10L if FAL or FSL	S/T10L = FPRD PH12 + ..	
		FPRD = FAFL NO2 +...	
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = FAFL PH12 + ...	
PH13 T6L or T10L	1-6 clocks if FAS or FSS 1-13 clocks if FAL or FSL		
	Case 1: NFPRR - Result Not Ready for Storage	FPRR = FAFLAS PH13 NA47 N(NRTZ NFNF A4751Z)	
	A47-A31 → S47-S31	SXA = FAFL PH13 NFPRR + ...	
	Sustain PH13	BRPH13 = FAFL PH13 NFPRR + ...	
	Set flip-flop T10L if FAL or FSL and more than two postnormalizing shifts are required	S/T10L = FAFL PH13 NFPRR N(FPPN A5255Z) FPRD + ...	
	Force a one on LR31 address line	R/NLR31/2 = LR31/2 = FAFL PH13 NFPRR + ...	
	Set flip-flop NGX	S/NGX = FAFL PH13 NFPRR + ...	
	S47-S31 x 1/16 ↛ A51-A31 If A47 is a one	AXSR4 = FAFL PH13 A47 + ...	Absolute value of sum equals + case 1. |Sum| x 1/16 → A". Result now ready for storage
	Increment E0-E7 by one	PCTE1 = FAFL PH13 A47 + ...	
	S47-S31 x 16 → A47-A27	FPPN = FAFL PH13 A4751Z NRTZ NFNF	0-5 clocks if FAS or FSS, 0-12 clocks if FAL or FSL. After last clock result is ready for storage
	if FPPN	AXSL4 = FPPN + ...	
	Decrement E0-E7 by one	MCTE1 = FPPN + ...	
	Set flip-flop BC31	S/BC31 = FPPN + ...	
	B31 ↛ B30	BXBL1 = FAFL PH13 NFPRR	
		FPRR = Same as above	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T6L or T10L (Cont.)	Case 2: FPRR - Result Ready for Storage		
	NA47-NA31 + K → S47-S31 if sum must be negative	SXADD = FPRR N(RTZ + FEUF NFZ) FPR + ...	K = K31 because of NGX or K71 if G0003/1 was set in PH12
	A47-A31 → S47-S31 if sum must be positive	SXA = FPRR N(RTZ + FEUF NFZ) NFPR + ...	
	All zeros onto S47-S31 if result is zero or if exponent underflow with FZ = 0	No gating term enabled	
	S0-S31 → RW0-RW31 if FAL or FSL	RW = FPRR FPRD NFTRAP + ...	Store LSW of result
	S48-S71 ↛ A8-A31	AXSR32 = FPRR N(FAFLD O2)	Most significant fraction sum bits
	E1-E7 + 64 ↛ A1-A7	AXE = FPRR + ...	(NE1 ↛ A1, E2-E7 ↛ A2-A7) uninverted exponent plus bias
	Force ones into CS0-CS7 if result is negative	CSX1/5 = FPRR FPR + ...	To invert exponent
	Set flip-flop DRQ	S/DRQ = FPRR + ...	Inhibits transmissions of another clock until data release signal is received
	Set flip-flop T8L	R/NT8L = T8L = FPRR + ...	
	Set flip-flop RTZ if FAS or FSS and A47-A71 = 0	S/RTZ = FPRR NFPRD NO6 A4771Z	Needed for PH15 use
	Set flip-flops TRAP and TR29 for trap to 68 if trap conditions exist	S/TRAP = FPRR FTRAP + ... S/TR29 = FPRR FTRAP + ...	
	Go to phase 15	BRPH15 = FPRR + ...	
		FTRAP = FEOF = NEO E1 NRTZ FAFL	Exponent overflow with result not equal to zero
		+ FEUF FZ = FZ E0 NE1 NRTZ FAFL	Exponent underflow with FZ = 1
		+ FRNSIG FS = FS (FAFLAS B30	More than two postnormalizing shifts (FN = 0, FS = 1)
		+ FAFLAS NFNF RTZ)	Result is zero (FN = 0, FS = 1)

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

(Continued)

Table 3-132. Floating Add, Floating (FAFLAS), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH15 T8L	A0-A31 ⊕ CS0-CS7 → S0-S31 if result does not equal zero	SXPR = FAFL PH15 N(RTZ + FEUF NFZ) + ...	Invert exponent if result negative
	S0-S31 → RW0-RW31 if NTRAP	RW = FAMDSF PH15 + ...	Store MSW of result
		RWDIS = TRAP NINTRAPF + ...	Disable storage
	Set flip-flop TESTA	S/TESTA = FAMDSF NFASHFX PH15 + ...	For CC3 and CC4 use
	Merge a one into CS31 if result is nonzero	A31X1 = FAFL PH15 N(RTZ + FEUF NFZ) + ...	For CC3 test
	Set flip-flop CC1, CC2, CC3 and CC4 as applicable	S/CC1 = FAFL PH15 FEUF + FAFL PH15 FRINSIG + ...	Exponent underflow Insignificant result, FN = 0
		S/CC2 = FAFL PH15 FEUF + FAFL PH15 FEOF + ...	Exponent underflow Exponent overflow
		S/CC3 = TESTS NA0 NA0031Z + ...	Result greater than zero
		S/CC4 = TESTA A0 + ...	Result less than zero
	Enable ENDE	ENDE = FAMDSF PH15 + ...	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: FAS (3D, BD), FAL (1D, 9D), FSS (3C, BC), FSL (1C, 9C)

There are three possible quantities that flip-flops E0 through E7 may contain at the beginning of PH4: all zeros, a positive number, or a negative number. If E0 through E7 are all zeros, signal EZ goes true. If a positive number is present, NEZ is true and E0 goes false. A negative quantity is defined by a one in E0. If flip-flops E0 through E7 are zeros, the augend and addend exponents are equal, the augend and addend are set up for addition or subtraction, and a branch to PH9 occurs. If flip-flops E0 through E7 are positive, the augend exponent is larger than the addend exponent. The augend and addend are interchanged, and the addend is shifted to the right until the addend exponent equals the augend exponent (flip-flops E0 through E7 = 0). The shifting and the examination process occurs in PH5, PH6, and possibly in PH8. PH8 is used if the original exponent difference was greater than one. If flip-flops E0 through E7 are negative, the addend exponent is larger than the augend exponent. The augend is shifted to the right until the augend exponent equals the addend exponent. The shifting and the examination process occurs in PH7 and possibly is PH8.

EZ = NE0 NE1 NE2 NE3 NE4 NE5 NE6 NE7

1. PH4, flip-flops E0 through E7 = 0. If the augend and the addend exponents are equal, PH4 is used to set up the addend for the addition or the subtraction of the two floating point numbers. If FAS or FAL is being performed, the complete fraction of the addend in bits C47 through C31 is transferred to flip-flops D47 through D31 at the PH4 clock (signal DXC/6). Bits C0 through C31 contain zeros if FAS is being implemented. If FSS or FSL is being performed, the inverted added fraction is transferred to flip-flops D47 through D31, and a one for the two's complement is forced into flip-flop CS31 by signal DXNC/1. PH9 and clock T8L are enabled.

2. PH4, flip-flops E0 through E7 > 0. If the augend exponent is greater than the addend exponent, PH4 is used to set up the registers for the interchange of the addend and the augend. The complete fraction of the addend in bits C47 through C31 is transferred to flip-flops D47 through D31 at the PH4 clock by signal DXC/6. The addend fraction is held temporarily in D" register until it can be transferred to the A-register in PH6. Flip-flop CXS is set for use in PH5.

3. PH4, flip-flops E0 through E7 > 0. If the addend exponent is larger than the augend exponent, PH4 is used to set up the augend for right-shifting in PH7 and PH8. The inverted addend exponent is in flip-flops D0 through D7. The absolute value of the addend exponent is taken by transferring bits ND0 through ND7 to flip-flops B1 through B7. (Flip-flops B1 through B7 always hold the absolute value of the larger of the original exponents.) The D-register is cleared at the PH4 clock by signal DX/1. The E-register contents are incremented by one to record the right shift which occurs in PH7 (E-register is counting toward zero). PH7 and T8L are enabled.

If the augend is negative, flip-flops NGX and FPR are set. NGX is used in the right shift during PH7. Signal FPR signifies that the floating point polarity of the result of the addition or the subtraction should be reversed. If the augend is negative, it is changed to positive before the addition for shifting purposes. The addend is also inverted to retain the same relationship to the augend. The result of the addition is therefore the inverse of the proper result.

e. PH5. PH5 is entered only if the E-register contained a positive number in PH4 (the augend was greater than the addend). PH5 is used to transfer the augend fraction to the C"-register. The augend fraction in flip-flops A47 through A31 is gated to the sum bus by signal SXA and is transferred at the PH5 clock to the C"-register by signal CXS. The A"-register is cleared by signal AX/1. The E-register contents are decremented by one to record the first right shift of the addend in PH6. (The E-register is counting toward zero.) Flip-flop NGX is set for use in PH6 if the addend is negative. The absolute value of the addend must be taken for shifting purposes. Flip-flop FPR is consequently set if the absolute value of the addend is opposite to the form required for addition or subtraction (if an addition is being performed with a negative addend or if a subtraction is being performed with a positive addend). Clock T8L is enabled for PH6.

f. PH6. PH6 is entered only from PH5 (the augend was greater than the addend in PH4). PH6 is used to complete the interchange of augend and addend and to perform the first shift of the addend. The absolute value of the addend is gated to the sum bus by signal SXADD. (NGX was set in PH5 if addend was negative.) The sum bus outputs are shifted one hex to the right and are transferred to the A"-register at the PH6 clock. The E-register contents are decremented one count to record a possible PH8 shift.

The augend was put into C"-register in PH5. If flip-flop FPR has been set, the sign of the addend is opposite to what it should be, and likewise, the augend sign must be reversed. The augend is transferred to the D"-register by signal DXC/6 at the PH6 clock if signal FPR is false. The inverted augend is transferred to the D"-register, and a one is forced into flip-flop CS31 for the two's complement by signal DXNC/1 if FPR is true. If the E-register contains a zero at the PH6 clock, PH9 and T8L are enabled. If flip-flops E0 through E7 are not all zeros, more shifting of the addend must be done, and PH8 is enabled.

g. PH7. PH7 is entered only if the E-register contained a negative number in PH4 (the addend was greater than the augend). PH7 is used to perform the first shift of the augend. The absolute value of the augend is gated to the sum bus from the A"-register by signal SXADD. (NGX was set in PH5 if the addend was negative.) The sum bus outputs are shifted one hex to the right and are transferred back to the A"-register at the PH6 clock. The E-register contents are incremented by one count.

Since the absolute value of the augend has been taken, the sign of the addend must also be reversed either if FPR is true and an addition is being performed or if FPR is false and a subtraction is being performed. Either the addend or the inverted addend is transferred to the D"-register at the PH6 clock (signals DXC/6 and DXNC/1, respectively). A one is forced into flip-flop CS31 if the inverted addend is transferred to complete the two's complemented addend. If the E-register contains a zero at the PH7 clock, PH9 and T8L are enabled. If flip-flops E0 through E7 are not all zeros, more shifting of the augend must be done, and PH8 is entered.

h. PH8. PH8 is entered from either PH6 (the augend was greater than the addend) or PH7 (the addend was greater than the augend). If PH8 is entered, it signifies that the difference of augend and addend exponents was greater than one and that more shifting is required before the addition can take place. If FAS or FSS is being performed, the augend or the addend in flip-flops A47 through A71 is gated to sum bus outputs S47 through S71 by signal SXA/3. If FAL or FSL is in effect, flip-flops A47 through A31 are gated to S47 through S31 by signal SXA. The sum bus outputs are shifted one hex to the right and are transferred back into the A-register at the PH8 clock. If FAS or FSS is being implemented, only bits A47 through A31 are significant. The bits in flip-flop A0 through A3 are the guard digits which may be retained in the final answer if postnormalization is needed. The count in the E-register is counted toward zero by incrementing the contents by one (if the contents are negative) or by decrementing the contents by one (if the contents are positive). Flip-flop CS31 remains set for complement for an inverted operand in the D-register. This must be held until the addition or subtraction takes place; repeater flip-flop CS31 is therefore not reset.

If the E-register is nonzero at the PH8 clock, PH8 is sustained and another shift is performed. If the E-register contains zero at any PH8 clock, the addend and augend exponents are equal and the addition or the subtraction can be performed. PH9 and clock T8L are enabled in this case. PH9 may also be enabled if a zero fraction in flip-flops A47 through A31 is detected. PH9 is enabled in this case but clock T8L is not, since carry propagation time in the addition is not required.

i. PH9 is the addition phase of FAFLAS, entered from PH4, PH6, PH7, or PH8. At this point in the implementation, the exponents of the augend and the addend are equalized and the addition or subtraction can take place. The augend and the addend in the A- and D-registers (not necessarily in that order) and bit CS31 are added and the fraction result (sum) gated to the sum bus by signal SXADD. At the PH9 clock, the result on S47 through S31 is transferred from the sum bus outputs to flip-flops A47 through A31. If signal FPR is true at this time, the result obtained in the addition process has the opposite polarity of the correct result. The result will be inverted in a later phase.

Fraction (mantissa) overflow may occur with the addition. When two positive or two negative numbers are added and their absolute sum is greater than or is equal to one, the usual overflow that occurs is a carry into the sign bit position. Maximum overflow into bit position 46 exists when the sum of the addition is -2. Overflow is corrected in all cases by a right shift of one hex. Flip-flop FMOF (floating point mantissa overflow) is set if overflow exists, but not if the overflow is caused by a sum of -1. Overflow is corrected in PH12 if FMOF is set, or in PH13 if caused by a sum of -1.

Flip-flops B1 through B7 contain the biased uninverted exponent of the sum (put in the register in PH2 or PH4). Sixty-four is subtracted from this by inverting the content of flip-flop B1 and by transferring B1 through B7 to E0 through E7. The result is the unbiased exponent of the sum. Bit E1 is now used as the sign bit of the exponent and the seven exponent bits may represent a positive or negative number, depending upon the state of bit E1. If bits E1 through E7 are a negative number (bit E1 = 1), it is in two's complement form.

The B- and D-registers are cleared by signals BX/1 and DX/1, respectively, at the PH9 clock. Flip-flop NGX is set for PH12 use. Memory request signal MRQ/1 is enabled to request the next instructions; interruptibility is stopped by resetting flip-flops IEN; and PH12 and T8L are enabled.

j. PH12. PH12 is used for overflow adjustments and first postnormalization attempts. The absolute value of the fraction result is gated to the sum bus by signal SXADD (if the result is negative) or by signal SXA (if the result is positive). Fraction overflow causes the state of bit position 47 of the result to be reversed. A negative result is detected either when there has been no overflow and the sign bit is true or when there has been overflow and the sign bit is false. A positive result is detected when the opposite conditions are true. The state of flip-flop FPR is reversed if the result generated in PH9 was negative, since the taking of the absolute value has changed the result from negative to positive.

There are three possible sets of conditions at the beginning of PH12: overflow, no overflow with postnormalization not necessary, and no overflow with postnormalization necessary.

When overflow has occurred, the absolute value of the result must be shifted one hex to the right. If FMOF is true, the sum bus outputs S47 through S27 are shifted this amount to the right and are transferred back into the A-register at the PH12 clock. The unbiased exponent in the E-register is incremented by one as the fraction result is shifted. A special case of overflow occurs if FMOF is true and if flip-flops A47 through A31 contain all zeros. This is the case when the addition or the subtraction has yielded -2. A one is forced into flip-flop A50 as the number is shifted in this case.

If overflow has not occurred and the result is simple normalized, or if the result is not simple normalized and postnormalization is to be inhibited (mode control bit FN = 1), the absolute value of the result is transferred to the A"-register at the PH12 clock by signal AXS.

If overflow has not occurred, and if the result is not simple normalized (postnormalization is to be performed), the sum bus outputs S51 through S31 are shifted one hex to the left and are transferred into flip-flops A47 through A27 at the PH12 clock. The unbiased exponent in the E-register is decremented by one count as the shift is made. This is the first try at postnormalization; if other shifts are required, they are performed in PH13. Also at the PH12 clock, a one is forced into flip-flop BC31. Flip-flops BC31, B31, and B30 form the postnormalization counter. When the first postnormalization attempt is made (PH12 or PH13), flip-flop BC31 is set. The B-register was cleared in PH9; it shifts left one bit position (B30 $\nleftarrow$ B31 $\nleftarrow$ BC31) for each postnormalization attempt. If flip-flop B31 contains a one and if postnormalization is still in progress or if flip-flop B30 contains a one when the result is ready for storage, more than two postnormalization shifts have taken place.

If overflow has not been detected and if flip-flops A47 through A31 contain all zeros, the result is equal to zero. Flip-flop RTZ is set in this case for use in following phases. The D-register is cleared by signal DX/1, and flip-flop NGX is set for PH13 use. If FAS or FSS is being performed, flip-flop G0003/1 is set for PH13 use. If FAL or FSL is being performed, clock T10L is enabled. A one is forced on private memory address line LR31 for transfer of the LSW of the result into the odd numbered private memory register in PH13.

k. PH13. PH13 is the postnormalization phase of FAFLAS. The result at this time is either ready for storage or not ready for storage. The result is not ready for storage if, at the beginning of PH13 or after adjustment, bit A47 is true or if there is a nonzero result that must be postnormalized. The result is ready for storage if bit A47 is not true, and there is a zero result, if the result is postnormalized, or if postnormalization is to be inhibited.

If the result is not ready for storage (NFPRR true), it is gated to the sum bus by signal SXA during PH13. PH13 is sustained as long as signal NFPRR is true. At the first PH13 clock, clock T10L is enabled if FAL or FSL is in effect and if more than two postnormalization shifts are required. A one is again forced on private memory address line LR31, and flip-flop NGX is set.

One of two courses is then followed:

1. If bit A47 is true, it signifies that the absolute value of the sum is +1. This can happen if the result in PH9 was -1/16 and FN = 1, or if the result in PH9 was -1. The result must be shifted to the right to correct the overflow. The S-bus outputs are shifted one hex to the right and the shifted result is transferred into the A"-register at the PH13 clock. The exponent in the E-register is increased by one. The result is now ready for storage.

2. If bit A47 is false, the result is nonzero, and if the result is not yet normalized and is to be, five postnormalization tries are made if FAS or FSS is in effect or twelve tries are made if FAL or FSL is being performed. The first clock is T10L; all others are T6L. The result on the sum bus is shifted one hex to the left and clocked into the A"-register. The E-register is decreased by one count to adjust the exponent. A one is forced into flip-flop BC31 and the contents of the B-register and bit BC31 are shifted one bit position to the left for postnormalization counting logic. Since the result is nonzero, no more than four additional clocks normalize the result if FAS or FSS is in effect, and no more than eleven additional clocks normalize the result if FAL or FSL is being implemented. This step is repeated for each clock until the result is normalized.

Underflow or exponent overflow is detected during PH13. If the exponent of the result in the E-register is decremented so often by postnormalization shifts that it becomes negative, underflow has occurred and signal FEUF goes true. The underflow indication is killed if more than two postnormalizing shifts are required, FZ = 0, and FS = 1 (trap on significance but not an underflow). A one is merged into flip-flop E1 to accomplish this. Exponent overflow is detected when the exponent is changed so that the quantity it holds exceeds +63 or is less than -63. Signal FEOF goes true with exponent overflow. If an overflow condition exists, adding 64 to bias the exponent would result in a biased exponent exceeding +127 or a negative exponent, neither of which is possible.

FEUF = E0 NE1 FAFL NRTZ

E1X1 = FAFLAS FPPN FS NFZ E0 B31

S/E1 = E1X1 + ...

FEOF = NE0 E1 FAFL NRTZ

When the result is ready for storage, one of three actions is taken

1. The result is negated, and a one is added for the two's complement if FPR is true. The one for two's complementing is provided by signal K31 (if FAL or FSL is in effect) or signal K71 (if FAS or FSS is being performed). The negated result is gated to the sum bus by signal SXADD.

2. The result remains unaltered if signal FPR is false; it is gated to the sum bus by signal SXA.

3. Zeros are gated to the sum bus if the result is zero or if there was exponent underflow with FZ = 0.

If FAL or FSL is being performed, and if there is no trap condition (signal RWDIS is false), sum bus outputs S0 through S31, LSW of the result, are transferred to the odd numbered private memory register at the PH13 clock by signal RW. Sum bus outputs S48 through S71 are transferred to flip-flops A8 through A31 at the clock by signal AXSR32. The exponent result in flip-flops E1 through E7 is biased by adding 64, and is clocked into flip-flops A1 through A7. Bias is achieved by reversing the state flip-flop of E1. Ones are clocked into flip-flops CS0 through CS7 to invert the exponent if the final result is negative. The A-register now holds a zero in flip-flop A0, the uninverted exponent in flip-flops A1 through A7, and the most significant part of the fraction result in flip-flops A8 through A31.

Flip-flops DRQ, T8L, and PH15 are set at the PH13 clock. Flip-flop DRQ inhibits transmission of another clock until a data release signal is received from memory. If FAS or FSS are being performed and if the result is zero, flip-flop RTZ is set for PH15 use.

If trap conditions exist, flip-flops TRAP and TR29 are set to provide a trap to memory location X'44' (68). Trap conditions are the following:

1. Exponent overflow. If the result is not equal to zero, exponent overflow causes an unconditional trap.

2. Underflow. Underflow causes a trap if the floating zero mode control bit, FN, is a one, if the result is not zero, and if signal FEUF is true.

3. Insignificant result. If FN = 0 (result is to be postnormalized), and FS = 1, a trap results if more than two postnormalizing shifts are required or if the result is zero.

l. PH15. PH15 is used to store the MSW of the result and to set the condition codes. If the result is not equal to zero, the MSW of the result is summed with the CS-register in an exclusive OR operation (bits CS0 through CS7 are all ones if the result is negative and invert the exponent in this case). If the result is equal to zero and if there is no trap condition, zeros are on sum bus outputs S0 through S31. The sum bus outputs, S0 through S31, are transferred to the even numbered private memory register if no trap condition exists. Signal RWDIS disables the transfer if a trap exists. The sum bus outputs are also transferred to flip-flops A0 through A31 for condition code use. A one is merged into flip-flop A31 by signal A31X1 if the result was not equal to zero. This action is necessary because a portion of the result has already been stored (PH13), and the most significant part may be zero.

Flip-flop TESTA is set to enable condition code bits 3 and 4. Condition code 1 is set if underflow has occurred or if there is an insignificant result with FN = 0. Condition code 2 is set if there is underflow with FZ = 0 (inhibited if FS = 1 and significance trap exists). Condition code 3 is set if the result is positive and nonzero; condition code 4 is set if the result is negative. Signal ENDE is generated. Flip-flop PRE1 is set to begin the preparation sequence for the next instruction. Clock T6L is enabled.

3-291 Floating Point Multiplication (FAFLM)

3-292 GENERAL. The implementation of FML is identical to the implementation of FMS with the exception of the number of fractional digits to be operated upon. The two instructions will be discussed together and differences between them will be noted in the discussion.

3-293 FLOATING POINT MODE CONTROL BITS. Mode control bit FZ, floating zero, determines the operation of floating point multiplication. Mode control bits FN and FS are not in effect.

If mode control bit FZ = 0, underflow or a zero result causes the result to be set equal to true zero. If FZ = 1, underflow causes the computer to trap to memory location X'44' (68). The contents of the general registers remain unchanged if the trap occurs.

If exponent overflow occurs, the computer always traps to location X'44' (68).

3-294 CONDITION CODE SETTINGS. The condition code settings and their meanings for floating point multiplication are shown in table 3-133.

3-295 BASIC STEPS IN FAFLM IMPLEMENTATION. To multiply two floating point numbers in Sigma 7, the following basic steps are performed:

a. Transfer of operands (PH1 and PH3). The multiplier (from private memory) and the multiplicand (from core memory) are transferred to the arithmetic unit registers.

b. Exponent summing (PH2 and PH3). The uninverted biased multiplicand exponent is added to the uninverted

Table 3-133. Condition Codes for Floating Point Multiplication

CONDITION CODE 1 2 3 4	IF NO TRAP TO LOCATION X'44' OCCURS	IF TRAP TO LOCATION X'44' OCCURS
0 0 0 0	A x 0	Impossible
0 0 0 1	Result negative	
0 0 1 0	Result positive and nonzero	
0 1 0 0	Impossible	
0 1 0 1		Overflow, result negative
0 1 1 0		Overflow, result positive and nonzero
1 0 0 0		▨
1 0 0 1		
1 0 1 0		
1 1 0 0		
1 1 0 1		Underflow, result negative, FZ = 1
1 1 1 0		Underflow, result positive and nonzero, FZ = 1

biased multiplier exponent. Also, -128 is subtracted from the exponent sum to remove the bias (since the exponent sum is biased by twice the normal amount after the addition of the two biased numbers). The unbiased exponent sum of the two floating point numbers represents the exponent of the product. This product exponent may be corrected as prenormalizing shifts are performed on the fractions of the two operands, or as postnormalizing shifts are performed on the product.

c. Prenormalization (PH3-PH7). Before the two floating point numbers are multiplied, both are simple normalized. After the exponents are added (step b), the multiplier and the multiplicand are examined. The following conditions are possible.

1. The multiplier and the multiplicand are both simple normalized. In this case, the multiplication may be performed without adjustment of either operand.

2. The multiplier is simple normalized but the multiplicand is not. The multiplicand is shifted to the left one hex at a time until it is simple normalized and is ready for multiplication. With each shift, the exponent sum is decremented by one to compensate for the shift.

3. The multiplicand is simple normalized but the multiplier is not. The multiplier is shifted to the left one hex at a time until it is simple normalized and is ready for multiplication. With each shift, the exponent sum is decremented by one.

4. Neither the multiplicand nor the multiplier are simple normalized. The multiplier is shifted to the right until it is simple normalized, and is followed by the simple normalization of the multiplicand. The exponent sum is decremented by an amount equal to the total number of multiplicand and multiplier shifts. Prenormalization of the multiplier is done only to record the number of insignificant digits in the multiplier. The prenormalized multiplier is discarded before the multiplication. When the multiplication is effected, the original multiplier is used; the insignificant digits of the original multiplier are ignored in the multiplication.

5. The multiplier or multiplicand or both are equal to zero. The multiplication is not performed and the product is set equal to zero.

d. Multiplication (PH10). After prenormalization of both operands, the fractions of the prenormalized floating point multiplicand and the significant bits of the original multiplier are multiplied. Multiplication methods are almost identical to the bit-pair methods used for fixed-point numbers (MW, MI, and MH opcodes).

e. Postnormalization (PH13). The product is examined.

1. If the product is normalized or is zero, and if there is no overflow, the product is ready for storage; 64 is added to the exponent to bias it once again.

2. If the product is not normalized, it is shifted to the left one hex to perform the postnormalization (since both operands were prenormalized, only one postnormalization shift is required). Step 1 is then performed.

3. If there is overflow in the product, it is shifted right one hex to correct the overflow condition. Step 1 is then performed.

f. Storage (PH13 and PH15). The biased exponent and the fraction result of the product are assimilated and are stored in memory in proper form, if there is no trap condition.

An example of FAFLM is shown in figure 3-219. The lettered steps correspond to the basic steps discussed.

3-296 DETAILED STEPS IN FAFLM IMPLEMENTATION. Figure 3-220 shows the FAFLM phases and the operations performed for each phase. The following discussion is supplemented by table 3-134 and by figure 3-220.

If FML is being performed, a one is forced on private memory address line LR31 during the phase preceding PRE2 NIA. The least significant word of the multiplier (LSW) is transferred from the odd numbered private memory register to the A-register during PRE2 NIA.

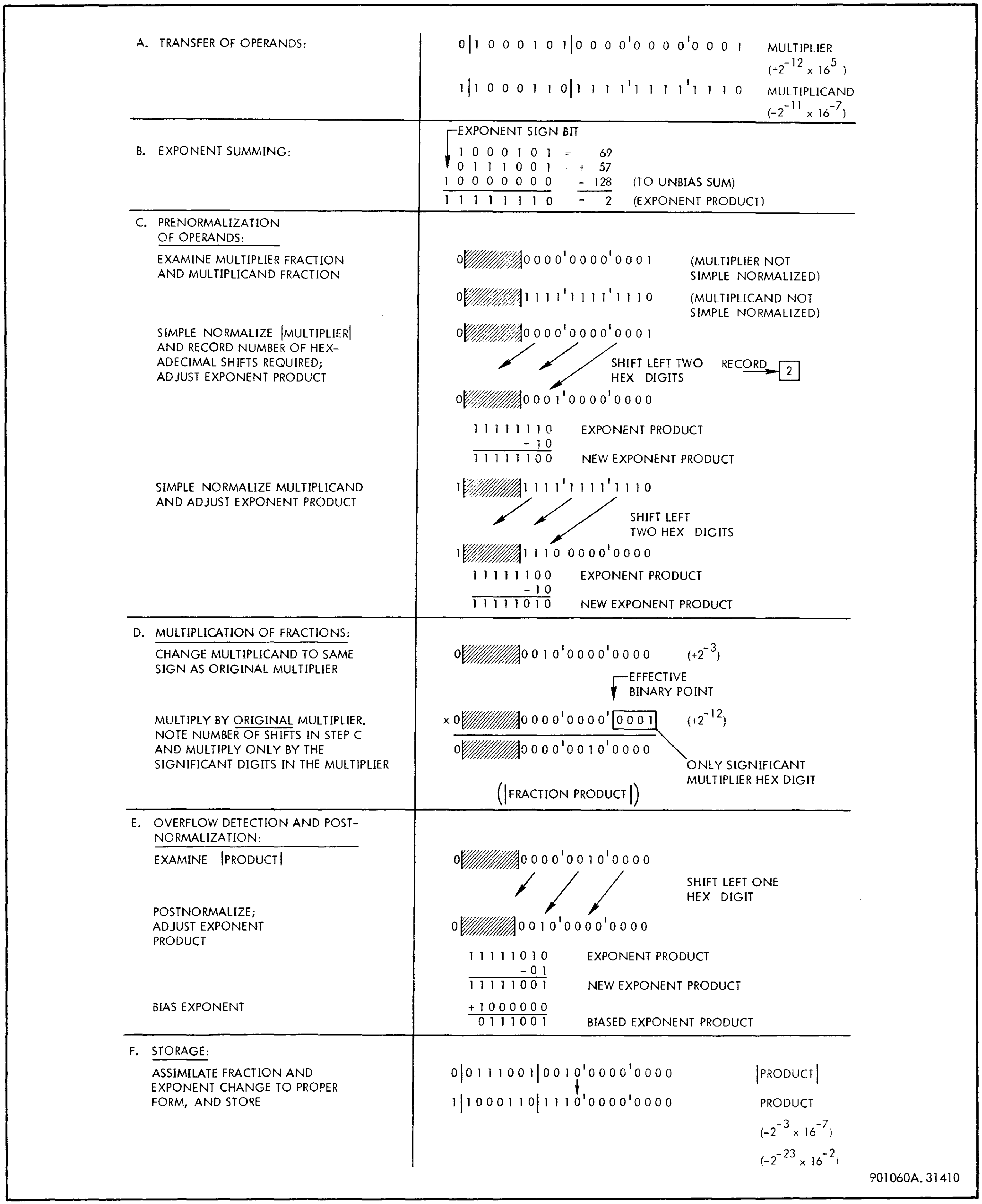

A. TRANSFER OF OPERANDS:
0|1 0 0 0 1 0 1|0 0 0 0'0 0 0 0'0 0 0 1 MULTIPLIER
(+2^-12 x 16^5)
1|1 0 0 0 1 1 0|1 1 1 1'1 1 1 1'1 1 1 0 MULTIPLICAND
(-2^-11 x 16^-7)
B. EXPONENT SUMMING:
EXPONENT SIGN BIT
1 0 0 0 1 0 1 = 69
0 1 1 1 0 0 1 + 57
1 0 0 0 0 0 0 0 - 128 (TO UNBIAS SUM)
1 1 1 1 1 1 1 0 - 2 (EXPONENT PRODUCT)
C. PRENORMALIZATION OF OPERANDS:
EXAMINE MULTIPLIER FRACTION AND MULTIPLICAND FRACTION
0 0 0 0 0'0 0 0 0'0 0 0 1 (MULTIPLIER NOT SIMPLE NORMALIZED)
0 1 1 1 1'1 1 1 1'1 1 1 0 (MULTIPLICAND NOT SIMPLE NORMALIZED)
SIMPLE NORMALIZE |MULTIPLIER| AND RECORD NUMBER OF HEXADECIMAL SHIFTS REQUIRED; ADJUST EXPONENT PRODUCT
0 0 0 0 0'0 0 0 0'0 0 0 1
SHIFT LEFT TWO HEX DIGITS
RECORD 2
0 0 0 0 1'0 0 0 0'0 0 0 0
1 1 1 1 1 1 1 0 EXPONENT PRODUCT
- 1 0
1 1 1 1 1 1 0 0 NEW EXPONENT PRODUCT
SIMPLE NORMALIZE MULTIPLICAND AND ADJUST EXPONENT PRODUCT
1 1 1 1 1'1 1 1 1'1 1 1 0
SHIFT LEFT TWO HEX DIGITS
1 1 1 1 0 0 0 0 0'0 0 0 0
1 1 1 1 1 1 0 0 EXPONENT PRODUCT
- 1 0
1 1 1 1 1 0 1 0 NEW EXPONENT PRODUCT
D. MULTIPLICATION OF FRACTIONS:
CHANGE MULTIPLICAND TO SAME SIGN AS ORIGINAL MULTIPLIER
0 0 0 1 0'0 0 0 0'0 0 0 0 (+2^-3)
EFFECTIVE BINARY POINT
MULTIPLY BY ORIGINAL MULTIPLIER. NOTE NUMBER OF SHIFTS IN STEP C AND MULTIPLY ONLY BY THE SIGNIFICANT DIGITS IN THE MULTIPLIER
x 0 0 0 0 0'0 0 0 0'0 0 0 1 (+2^-12)
0 0 0 0 0'0 0 1 0'0 0 0 0
ONLY SIGNIFICANT MULTIPLIER HEX DIGIT
(|FRACTION PRODUCT|)
E. OVERFLOW DETECTION AND POST-NORMALIZATION:
EXAMINE |PRODUCT|
0 0 0 0 0'0 0 1 0'0 0 0 0
SHIFT LEFT ONE HEX DIGIT
POSTNORMALIZE; ADJUST EXPONENT PRODUCT
0 0 0 1 0'0 0 0 0'0 0 0 0
1 1 1 1 1 0 1 0 EXPONENT PRODUCT
- 0 1
1 1 1 1 1 0 0 1 NEW EXPONENT PRODUCT
BIAS EXPONENT
+ 1 0 0 0 0 0 0
0 1 1 1 0 0 1 BIASED EXPONENT PRODUCT
F. STORAGE:
ASSIMILATE FRACTION AND EXPONENT CHANGE TO PROPER FORM, AND STORE
0|0 1 1 1 0 0 1|0 0 1 0'0 0 0 0'0 0 0 0 |PRODUCT|
1|1 0 0 0 1 1 0|1 1 1 0'0 0 0 0'0 0 0 0 PRODUCT
(-2^-3 x 16^-7)
(-2^-23 x 16^-2)
901060A.31410

Figure 3-219. Example of FAFLM Implementation

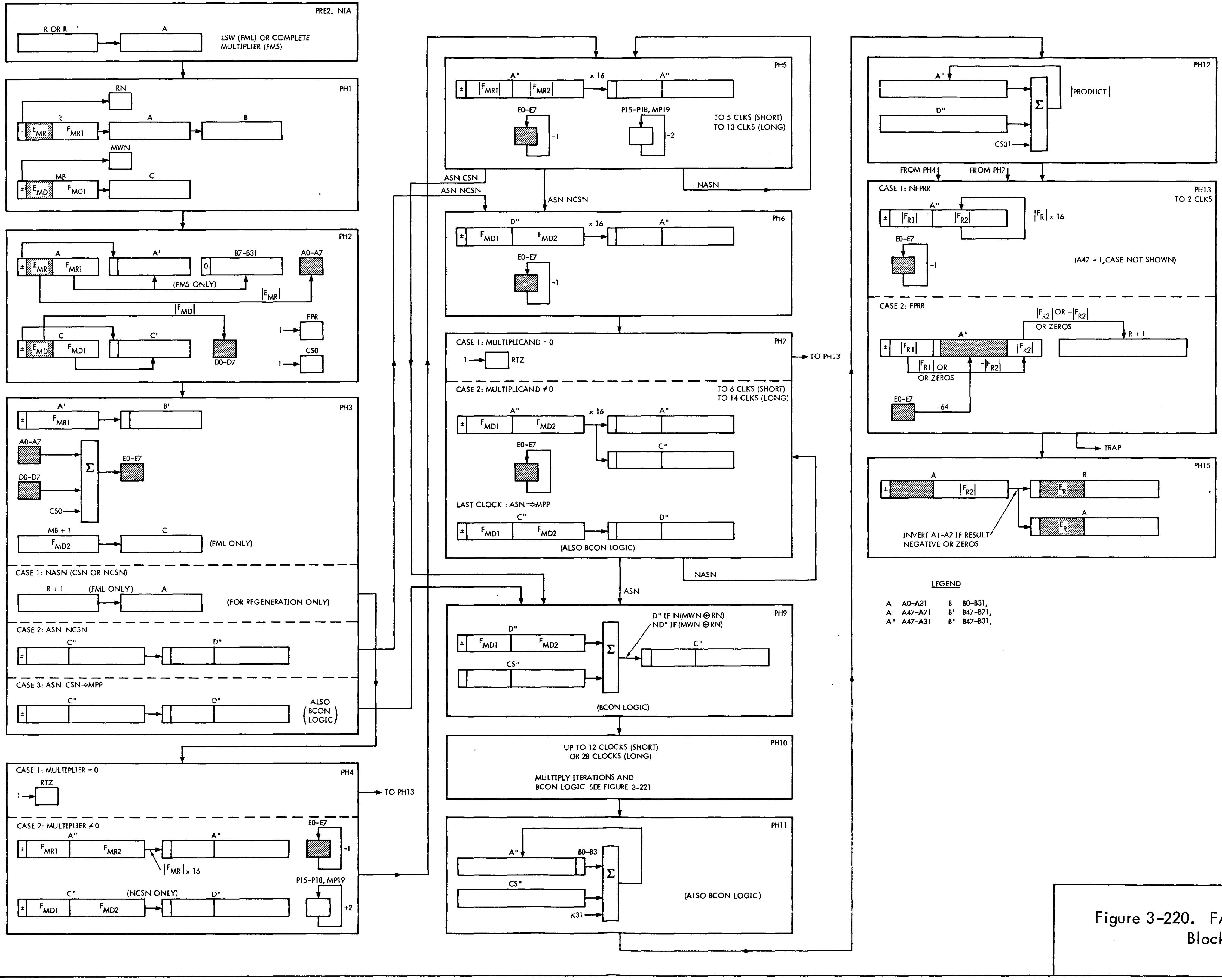


Figure 3-220. FAFLM Implementation, Block Diagram

901060A. 31413

Table 3-134. Floating Multiply (FAFLM), Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation except for		
Phase Pre-ceding PRE2, NIA	Force one onto LR31 address line if FML	R/NLR31/2 = (PRE1 NIA + PRE3 IX + PRE4) OUI OUF + ...	To select odd numbered private memory register
PRE2 NIA T4RL	RR0-RR31 —/→ A0-A31	AXRR = FAMDSF PRE2 NIA + ...	LSW of multiplier (FML) or complete multiplier (FMS)
PH1 T4RL	A0-A31 ——→ S0-S31	SXA = FAFL PH1 + ...	
	S0-S31 —/→ B0-B31	BXS = FAMDSF PH1 + ...	
	RR0-RR31 —/→ A0-A31	AXRR = RNXRR0/2 PH1 + ... = FAFL PH1 + ...	Redundant if FMS; MSW of multiplier if FML
	Set flip-flop RN if multiplier negative	S/RN = RR0 RNRXRR0 + ... = RR0 PH1 FAFL + ...	Stores multiplier sign
	MB0-MB31 ——→ C0-C31	Preparation control	MSW of multiplicand
	Set flip-flop MWN if multiplicand negative	S/MWN = C0C16 FAMDSF	Stores multiplicand sign
	Set flip-flop CXCL32	S/CXCL32 = FAFL PH1 + ...	For PH2 use
	Reset flip-flop CC1	R/CC1 = FAFL PH1 + ...	
	Reset flip-flop CC2	R/CC2 = FAMDSF NFAMULH	
	Force all ones into CS0-CS7	CSX1/5 = FAFL PH1 + ...	To invert multiplier exponent if multiplier negative
	Set flip-flop IEN	S/IEN = FAMDSF NFAMUL + ...	Start interruptibility
PH2 T6L	A0, A8-A31 —/→ A47, A48-A71	AXAL32 = FAFL PH2	MSW of multiplier —/→ A'
	C0, C8-C31 ——→ C47, C48-C71	CXCL32 = set in PH1	MSW of multiplicand —/→ C'
	A0-A31 + CS0-CS7 ——→ S0-S31 if multiplier negative or	SXPR = FAFL PH2 RN + ...	Uninverted multiplier exponent
	A0-A31 ——→ S0-S31 if multiplier positive	SXA = FAFL PH2 NRN + ...	
	S0-S7 —/→ A0-A7	AXS/1 = FAFL PH2 + ...	

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)	NC0-NC7 -/→ D0-D7 if multiplicand negative	DXNC = FAFL PH2 MWN + ...	Uninverted multiplicand exponent
	or		
	C0-C7 -/→ D0-D7 if multiplicand positive	DXC/6 = FAFL PH2 NMWN + ...	
	Force a one into CS0	S/CS0 = FAFLM PH2 + ...	To remove 2X bias from the exponent sum
	S8-S31 -/→ B8-B31 if FMS	BXS = FAFL PH2 N(FAFLM NO2) + ...	Multiplier -/→ B8-B31
	Set flip-flop FPR if final product is to be negative	S/FPR = FAFLMD PH2 (MWN ⊕ RN) + ...	Result must be reversed
	Force a one onto LB31 address line	S/LB31/1 = FAFL PH2 + ...	
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = FAFL PH2 + ...	
	Enable MRQ	MRQ = FAFL PH2 + ...	
	Set flip-flop DRQ if FML	S/DRQ = FAFL PH2 NO2 + ...	Inhibits transmission of another clock until data release signal received
	Enable T6RL	T6RL = FAFL PH2 O2 + ...	
	Set flip-flop CX/1 if FMS	S/CX/1 = FAFL PH2 O2 + ...	
PH3 T6L (FML) T6RL	A47-A71 → S47-S71	SXA/3 = FAFL PH3 + ...	
	S47-S71 -/→ B47-B71	BXS/1 = FAFL PH3	Insignificant if FMS
	A0-A7 + D0-D7 + CS0 → S0-S7	SXK/1 = FAFL PH3 + ...	Exponent product
		SXPR/1 = FAFL PH13 + ...	
	S0-S7 -/→ E0-E7	EXS = FAFL PH3 + ...	Exponent product
	Clear P-register	PX = FAFL PH3 + ...	
	MB0-MB31 → C0-C31 if FML	See PH2	Zeros → C0-C31 if FMS
	Clear A"-register	AX/1 = FAFL PH3 + ...	May be regenerated (see case 1 below)

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (FMS) T6RL (Cont.)	Case 1: NASN	ASN = (A47 + A48 + A49 + A50 + A51) (NA47 + NA48 + NA49 + NA50 + NA51)	A, not simple normalized case
	S47-S71 -/→ A47-A71	AXS/4 = FAFL PH3 N(FAFLM ASN)	Regenerate A'
	RR0-RR31 → A0-A31 if FML	AXRR = FAFL PH3 N(FAFLM ASN) NO2 + ...	LSW of multiplier (zeros → A if FMS)
	Set flip-flop NGX if multiplier negative	S/NGX = FAFL PH3	For PH4 use
	Clear D'-register	DX/1 = FAFLMD PH3 + ...	
	Set flip-flop T8L	R/NT8L = T8L = FAFL PH3 N(FAFLM ASN) FAFLMD + ...	
	Case 2: ASN NCSN	CSN = (C47 + C48 + C49 + C50 + C51) (NC47 + NC48 + NC49 + NC50 + NC51)	ASN and C, not simple normalized case
	C47-C31 -/→ D47-D31	DXC/6 = FAFLM PH3 ASN + MPP + ...	Multiplicand -/→ D"
	Clear A"-register	AX/1 = See above, PH3	
	Enable PH6	BRPH6 = FAFLM PH3 ASN + ...	
	Case 3: ASN CSN => MPP		A and C, simple normalized case
	C47-C31 -/→ D47-D31	DXC/6 = FAFLM PH3 ASN + MPP + ...	Multiplicand -/→ D"
	Enable MPP	MPP = FAFLM PH3 ASN CSN + ...	Prepare for multiply iterations
	Clear A"-register	AX/1 = MPP + ...	Redundant in this phase
	C47-C31 -/→ D47-D31	DXC/6 = MPP + ...	Redundant in this phase
	Force ones into CS47-CS31 if signs of multiplier and multiplicand opposite	CSX1 = MPP (MWN ⊕ RN)	To generate absolute value of product
	B0-B29 -/→ B2-B31	BXBR2 = MPP + ...	Examine multiplier bit pair 2^1, 2^0. Set multiplier control flip-flops to this bit-pair. Shift multiplier to bring next bit-pair into B30, B31
	Enable BCON	BCON = BXBR2 FAMDSF/M + ... = MPP	
	Set up flip-flops DXCM, DXNCM, DXCLIM, BC31 as functions of B30, B31, and BC31		
	Set flip-flop CXS	S/CXS = MPP + ...	For PH9 use

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH3 T6L (FMS) T6RL (Cont.)	Reset flip-flop IEN	R/IEN = MPP + ...	Stop interruptibility
	Enable PH9	BRPH9 = MPP + ...	Pre-iterations phase
PH4 T8L	(Entered only if NASN in PH3)		
	Case 1: A" = 0		
	Set flip-flop RTZ	S/RTZ = FAFLM PH4 A4731Z + ... = (S/RTZ/1) + ...	Product = 0
	Enable MRQ/1	MRQ/1 = FAFLMD (S/RTZ/1) + ...	Memory request for next instruction
	Reset flip-flop IEN	R/IEN = (S/RTZ/1) + ...	Stop interruptibility
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = (S/RTZ/1) + ...	
	Set flip-flop T10L if FML or on even R-field	S/T10L = (S/RTZ/1) FPRD + ... = (S/RTZ/1) FAFLM NR31 NO2 + ...	
		FPRD = FAFLM NR31 NO2	
	Enable PH13	BRPH13 = (S/RTZ/1) + ...	
	Case 2: A" ≠ 0		
	| A47-A31 | → S47-S31	SXADD = FAFLMD PH4 + ...	NGX set in PH3 if multiplier negative
	Enable FPRENR	FPRENR = FAFLMD PH4 + NASN + ...	Prenormalize R operand
	S51-S31 x 16 —/→ A47-A27	AXSL4 = FPRENR + ...	Shift multiplier one hex to the left
	Increment E0-E7 by one	MCTE1 = FPRENR + ...	
	Up-count P15-P18	PCTP5/1 = FPRENR NA4731Z + ...	Delete two multiply iterations
	C47-C31 —/→ D47-D31 if multiplicand not simple normalized	DXC/6 = FAFLMD PH4 NCSN + ...	
PH5 T6L	(Entered only from PH4 case 2)		
	1-5 clocks if FMS		
	1-13 clocks if FML		

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T6L	Case 1: NASN		
	Enable FPRENR	FPRENR = FAFLMD PH5 NASN + ...	
	A47-A31 → S47-S31	SXA = FAFL PH5 + ...	
	S51-S31 x 16 ↛ A47-A27	AXSL4 = FPRENR + ...	Shift multiplier one hex to the left
	Increment E0-E7 by one	MCTE1 = FPRENR + ...	
	Up-count P15-P18	PCTP5 = FPRENR NA4731Z + ...	Delete two multiply iterations
	Sustain PH5 if NASN	BRPH5 = FPRENR PH5 + ...	
	Case 2: ASN NCSN		
	Advance to PH6		
	Case 3: ASN-CSN => MPP	Same as PH3, case 3	Prepare for multiply iterations (enable PH9)
		MPP = FAFLM PH5 ASN CSN + ...	
PH6 T6L	(Entered only from PH3, case 2 or PH5, case 2)		
	D47-D31 → S47-S31	SXD = FAFLMD PH6 + ...	Multiplicand in D"
	Enable FPRENM	FPRENM = FAFLMD PH6 N(PH7 ASN) + ...	Prenormalize M operand
	S51-S31 x 16 ↛ A47-A27	AXSL4 = FPRENM + ...	Shift multiplicand one hex to the left
	Decrement E0-E7 by one	MCTE1 = FPRENM FAFLM + ...	
	Enable S/CXS	S/CXS = FAFLMD PH6 + FPRENM NA4731Z	For PH7 use
PH7 T6L	(Entered from PH6)		
	1-6 clocks if FMS		
	1-14 clocks if FML		
	Case 1: A = 0	Same as PH4, case 1	Product = 0 (enable PH13)
		(S/RTZ/1) = FAFLD PH7 NA4731Z + ...	

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T6L (Cont.)	Case 2: A ≠ 0, NASN		
	Enable FPRENM	FPRENM = FAFLMD PH7 N(PH7 ASN)	
	A47-A31 → S47-S31	SXA = FAFLMD PH7 + ...	
	S51-S31 ↛ A47-A27	AXSL4 = FPRENM + ...	Shift multiplicand left one hex
	Decrement E0-E7 by one	MCTE1 = FPRENM FAFLM + ...	
	S47-S31 → C47-C31	CXS = See PH6	To save adjusted multiplicand
	Enable S/CXS again	S/CXS = FPRENM NA4731Z + ...	To save adjusted multiplicand
	Sustain PH7 is NASN	BRPH7 = FPRENM NA4731Z + ...	
	Case 3: A ≠ 0, ASN => MPP	Same as PH3, case 3 and	Prepare for multiply iterations (enable PH9)
		MPP = FAFLM PH7 ASN + ...	
	A47-A31 → S47-S31	SXA = Same as above, case 2	
	S47-S31 → C47-C31	CXS = PH6 or PH7, case 2	
PH9 T6L	(Entered from PH3, case 3; PH5, case 2; or PH7, case 3)		
	D47-D31 ⊕ CS47-CS31 → S47-S31	SXPR = FAMDSF/M PH9 + ...	Match multiplicand to multiplier for | product |
	S47-S31 → C47-C31	S/CXS = Enabled by MPP	PH3, PH5, or PH7
	C47-S31 ↛ D47-D31	DXCM = Previously set by MPP	First C ↛ D transfer as f(BCON)
	or		
	NC47-NC31 ↛ D47-D31	DXNCM = Previously set by MPP	
	or		
	C47-C31 ↛ D46-D30	DXCLIM = Previously set	
	or		
	Zeros ↛ D47-D31 if none of above		

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T6L (Cont.)	Force a one into flip-flop DCWCM if ones complement transferred to D"	S/BC31 = BCON (B30 B31 NBC31 + B30 BC31 + ...) S/DCWCM = CCWCM (DXCM + DXCLIM) + NCCWCM DXNCM	One added later to make up two's complement
	B0-B29 $\nrightarrow$ B2-B31	BXBR2 = FAMDSF/M PH9 + ...	Examine bit-pair 2^3, 2^2. Set multiplier control flip-flops to this bit-pair. Shift multiplier to bring next bit-pair into B30, B31
	Sustain BCON	BCON = FAMDSF/M PH9 + ...	
	Set up flip-flops DXCM, DXWCM, DXCLIM, B31 as functions of B30, B31, BC31		
	Set flip-flop MIT	S/MIT = FAMDSF/M PH9 + ...	
	Enable clock T1L if not a single clocking	S/T1L = FAMDSF/M PH9 NKSC N(S/FLMC/1 P18)	
	Set flip-flop FLMC if significance in original multiplier is only in the least significant hex	S/FLMC = PH9 P18 FAFLM P16(P15 + O2)	Only two PH10 clocks necessary
PH10 T1L	12 clocks (FMS) or 28clocks (FML) - two times number of prenormalization shifts. For each clock the following events occur:		
	A" ⊕ D" ⊕ CS" ⟶ PR47-PR31		
	PR47-PR31 $\nrightarrow$ A48-A31, B0, B1	AXPRR2 = MIT	Sums of the partial products
	A" D" + A" CS" + D" CS" $\nrightarrow$ G47-G31		Carries of the partial products
	G47-G31 $\nrightarrow$ CS48-CS32	CSXGR1 = MIT	
	Contents of C" are transferred to D" under control of the multiplier control. Flip-flops DXCM, DXNCM, DXCLIM, set during previous clock		
	DCWCM set as before		

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH10 T1L (Cont.)	B47-B69 ⇸ B49-B71 B70, B71, B4-B29 ⇸ B4-B71	BXBR2 = MIT + ...	
	Sustain BCON	BCON = BXBR2 NFLMC + ...	
	Set up flip-flops DXCM, DXNCM, DXCLIM, B31 as functions of B30, B31, BC31		
	Count iterations using P15-P18 and MP19	PCTP5 = MP19 + ...	
		S/MP19 = MIT N(FLMC + MITEX) NMP19	
	<u>On the 26 count</u>:		
	Other events listed above		
	Set flip-flop FLMC	S/FLMC = MP19 FAFLM P16 (P15 + O2) + ...	
	Reset flip-flop T1L	R/T1L = MP19 (S/FLMC/1) + NMIT	
	<u>On the 27 count</u>:		
	Other events listed above		
	Set flip-flop MITEX	S/MITEX = FLMC + ...	
	<u>On the 28 count</u>:		
	Other events listed above		
	Reset flip-flop MIT	R/MIT =	
	Set flip-flop T8L	R/NT8L = T8L = MITEX FAFLM + ...	
	Advance to PH11	BRPH10 = MIT NMITEX + ...	Signal BRPH10 disabled
PH11 T8L	A47-A31 + CS47-CS31 + K31 → S47-S31	SXADD = FAMDSF/M PH11	K31 is end carry from B
	S47-S31 ⇸ A47-A31	AXS = FAFLM PH12 + ...	Last partial product
	B0-B1 + CS32, CS33, BC1 ⇸ B0-B1	BXB = FAMDSF/M PH11 + ...	

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PH11 T8L (Cont.)	Contents of C" are transferred to D" under control of the multiplier control flip-flops DXCM, DXNCM, DXCLIM, which were set during last clock of PH10			
	Force a one into CS31 if one's complement transferred to D"	CSX1/8	= PH11 (S/DCWCM) + ...	Used in lieu of DCWCM
	Enable MRQ/1	MRQ/1	= FAFLM PH11 + ...	Memory request for next instruction
	Set flip-flop T8L	R/NT8L	= T8L = FAFLMD PH11	
PH12 T8L	A47-A31 + D47-D31 + CS31 ⟶ S47-S31	SXADD	= FAFLM PH12	$\lvert$Product$\rvert$
	S47-S31 $\not\longrightarrow$ A47-A31	AXS	= FAFLM PH12 + ...	
	Clear D"-register	DX/1	= FAFL PH12 + ...	
	Set flip-flop NGX	S/NGX	= FAFL PH12 + ...	For PH13 use
	Set flip-flop G0003/1 if FMS and R-field is odd	(S/G0003/1)	= FAFL PH12 NFPRD	Causes K71 in FPRR case
	Force a one on address line LR31	R/NLR31/2	= LR31/2 = FAFL PH12 + ...	
	Set flip-flop T10L if not FMS or the R-field is not odd	S/T10L	= FPRD PH12 + ...	
PH13 T6L or T10L	One to two clocks, both T6L if FMS and R are odd; otherwsie, both are T10L			
	Case 1: NFPRR - Result Not Ready for Storage	FPRR	= FAFLAS PH13 NA47 N(NRTZ NFNF A4751Z)	
	A47-A31 ⟶ S47-S31	SXA	= FAFL PH13 NFPRR	
	Sustain PH13	BRPH13	= FAFL PH13 NFPRR + ...	
	Set flip-flop T10L if not FMS or R-field is not odd	S/T10L	= FAFL PH13 NFPRR N(FPPN A5255Z) FRD	
	Force a one on LR31 address line	R/NLR31/2	= LR31/2 = FAFL PH13 NFPRR + ...	
	Set flip-flop NGX	S/NGX	= FAFL PH13 NFPRR + ...	

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T6L or T10L (Cont.)	S47-S31 x 1/16 → A51-A31 } If A47 is a one	AXSR4 = FAFL PH13 A47 + ...	
	Increment E0-E7 by one	PCTE1 = FAFL PH13 A47 + ...	
		FPPN = FAFL PH13 A4751Z NRTZ NFNF	
	S47-S31 x 16 –/→ A47-A27 } if FPRN	AXSL4 = FPPN + ...	
	B0-B3 –/→ A28-A31	MCTE1 = FPPN + ...	
	Decrement E0-E7 by one		
	Case 2: FPRR - Result Ready for Storage	FPRR = Same as above, case 1	
	NA47-NA31 + K → S47-S31 if product must be negative	SXADD = FPRR N(RTZ + FEUF NFZ) FPR + ...	
	A47-A31 → S47-S31 if product must be positive	SXA = FPRR N(RTZ + FEUF NFZ) NFPR + ...	
	All zeros into S47-S31 if result is zero or if exponent underflow with FZ = 0	No gating term enabled	
	S0-S31 –/→ RW0-RW31 if not FMS or R-field is not odd (and no trap)	RW = FPRR FPRD NFTRAP + ...	Store LSW of result
	S48-S71 –/→ A8-A31	AXSR32 = FPRR N(FAFLD O2)	Most significant fraction product bits
	E1-E7 + 64 –/→ A1-A7	AXE = FPRR + ...	NE1 –/→ A1, E2-E7 –/→ A2-A7 uninverted exponent plus bias
	Force ones into CS0-CS7 if result negative	CSX1/5 = FPRR FPR + ...	
	Set flip-flop DRQ	S/DRQ = FPRR + ...	Inhibits transmission of another clock until data release signal received
	Set flip-flop T8L	R/NT8L = T8L = FPRR + ...	
	Set flip-flops TRAP and TR29 for trap to 68 if trap conditions exist	S/TRAP = FPRR FTRAP + ...	

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(Continued)

Table 3-134. Floating Multiply (FAFLM), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T6L or T10L (Cont.)		FTRAP = FEOF + FEUF FZ	
		FEOF = NE0 E1 NRTZ FAFL	Exponent overflow
		FEUF = E0 NE1 NRTZ FAFL	Exponent underflow with FZ = 1
		S/TR29 = FPRR FTRAP + ...	
	Enable PH15	BRPH15 = FPRR + ...	
PH15 T8L	A0-A31 ⊕ CS0-CS7 ——→ S0-S31 if result does not equal zero	SXPR = FAFL PH15 N(RTZ + FEUF NFZ) + ...	Invert exponent if result negative
	S0-S31 —/→ RW0-RW31 if NTRAP	RW = FAMDSF NFASHFX PH15 + ...	Store MSW of result
	S0-S31 —/→ A0-A31	AXS = FAFL PH15 + ...	
		RWDIS = TRAP NINTRAPF + ...	Disable storage
	Set flip-flop TESTA	S/TESTA = FAMDSF NFASHFX PH15 + ...	For CC3 and CC4 use
	Merge a one into CS31 if result is nonzero	A31X1 = FAFL PH15 N(RTZ + FEUF NFZ) + ...	For CC3 test
	Set flip-flops CC1, CC2, CC3, and CC4 as applicable	S/CC1 = FAFL PH15 FEUF + ...	Exponent underflow
		S/CC2 = FAFL PH15 (FEUF + FEOF) + ...	Exponent underflow or overflow
		S/CC3 = TESTA NA0 NA0031Z + ...	Result greater than zero
		S/CC4 = TESTA A0 + ...	Result less than zero
	Enable ENDE	ENDE = FAMDSF PH15 + ...	

Mnemonic: FMS (3F, BF), FML (1F, 9F)

(If FMS is being implemented, the entire 32 bits of the multiplier is transferred from the even numbered private memory register to the A-register.)

a. PH1. At the PH1 clock, the contents of the A-register are transferred to the B-register by signal BXS, and the even numbered private memory register contents are transferred to the A-register. (If instruction FMS is being performed, this operation is redundant; if instruction FML is being implemented, the most significant word, MSW, of the multiplier is now in flip-flops A0 through A31 and the LSW of the multiplier is now in flip-flops B0 through B31.) Flip-flop RN is set at the PH1 clock if the multiplier is negative.

The MSW of the multiplicand has already been transferred to the C-register during the preparation phases. Flip-flop MWN is set if the multiplicand is negative.

Signal CXCL32 is generated for use in PH2. Flip-flops CC1 and CC2 are reset for underflow and overflow tests in PH15, and flip-flop IEN is set to start interruptibility. Ones are forced into flip-flops CS0 through CS7 by signal CSX1/5 to invert the multiplier exponent to its true form if the multiplier is negative.

b. PH2. At the PH2 clock, the fraction part of the multiplier MSW (or the complete fraction if FMS is being performed) is transferred from flip-flops A0 and A8 through A31 to flip-flops A47 and A48 through A71, respectively, by signal AXAL32. The fraction part of the multiplicand MSW is transferred to the C-register extension in the same manner by signal CXCL32.

The multiplier and the multiplicand exponents are set up in PH2 for exponent addition. The uninverted multiplier exponent is either present in flip-flops A1 through A7 or is found by combining bits A1 through A7 with bits CS0 through CS7 (all ones) in PH3 in an exclusive OR operation. The uninverted multiplier exponent is gated to the sum bus during PH2 and, at the PH2 clock, is transferred from the sum bus to flip-flops A0 through A7 by signal AXS/1.

The uninverted multiplicand exponent must be added to the uninverted multiplier exponent. The uninverted multiplicand is transferred to flip-flops D0 through D7 by signal DXNC (if the multiplicand is negative) or by signal DXC/6 (if the multiplicand is positive).

A one is forced into flip-flop CS0 at the PH2 clock. When the exponents are added in PH3, the one in flip-flop CS0 subtracts 128 from the exponent sum, removing the bias.

S/CS0 = FAFLM PH2 + ...

If FMS is being performed, the complete multiplier consists of only 32 bits. The fraction of the multiplier is transferred from the sum bus outputs S8 through S31 to flip-flops B8 through B31 by signal BXS. Flip-flop FPR is set if the product of the multiplication is to be negative (that is, if a positive floating point number is to be multiplied by a negative floating point number).

Ones are forced on core memory address line LB31 and on private memory line LR31 for possible use in PH3 (with FML). If FML is being implemented, flip-flops MRQ (memory service request) and DRQ (inhibits transmission of another clock until data release signal is received from memory) are set to transfer the least significant word of the multiplicand to the C-register at the PH3 clock. Clock T6RL is enabled for PH3. If FMS is being implemented, flip-flop CX/1 is set at the PH2 clock to force all zeros into flip-flops C0 through C31 in PH3.

c. PH3. During PH3, the most significant part of the multiplier fraction in flip-flops A47 through A71 is gated to the sum bus outputs S47 through S71 by signal SXA/3. At the PH3 clock, S47 through S71 are transferred to flip-flops B47 through B71 by signal BXS/1. (The transfer is insignificant if FMS is being performed.) The complete multiplier is held in flip-flops B47 through B31 if FML is being performed or in flip-flops B7 through B31 if FMS is in effect. This multiplier is the one involved in the actual multiply process in PH10 and is saved in the B-register until that phase. The multiplier in the A"-register is the one involved in any prenormalization and is later discarded.

During PH3, the uninverted multiplier exponent in flip-flops A0 through A7, the uninverted multiplicand exponent in flip-flops D0 through D7, and the 1 in flip-flop CS0 are added, and their sum is gated to the sum bus. At the PH3 clock, the sum bus outputs on S0 through S7 are transferred to flip-flops E0 through E7 by signal EXS. Flip-flops E0 through E7 now hold the unbiased exponent sum.

At the PH3 clock, the P-register is cleared by signal PX. The P-register later serves as an iteration counter for the multiply iterations. If FML is being implemented, the LSW of the multiplicand is gated into C0 through C31 from the odd numbered private memory register. If FMS is being performed, zeros are read into C0 through C31 (no gating term enabled). The A-register is cleared by signal AX/1. The contents of the A-register may be regenerated if the multiplier is not simple normalized.

The multiplier and multiplicand are examined in PH3. If the multiplier is simple normalized, signal ASN goes true. If the multiplicand is simple normalized, signal CSN goes true.

There are three normalization combinations in PH3. Each causes a different sequence of events to take place.

1. Case 1. The multiplier is not simple normalized. If signal ASN is false, sum bus outputs S47 through S71 are transferred to flip-flops A47 through A71, regenerating the contents of the A-register extension cleared by signal AX/1. To regenerate this portion of the multiplier, the contents of the odd numbered private memory register are again

transferred to flip-flops A0 through A31 if FML is being performed. If FMS is in effect, zeros are clocked into flip-flops A0 through A31. Flip-flop NGX is set if the multiplier is negative. The D"-register is cleared and clock T8L is enabled for PH4.

2. Case 2. The multiplier is simple normalized, but the multiplicand is not simple normalized. Preparations are made to normalize the multiplicand. If signal ASN is true and if signal CSN is false, the multiplicand is transferred from the C"-register to the D"-register by signal DXC/6. The A-register is cleared by signal AX/1. PH6 is enabled.

3. Case 3. Both the multiplier and the multiplicand are simple normalized. Preparations are made to multiply the two operands. If signals ASN and CSN are both true, the multiplicand is transferred from the C-register to the D-register by signal DXC/6. The A-register is cleared by signal AX/1. Signal MPP (multiply preparation) is enabled. Ones are clocked into the CS"-register if the multiplier and the multiplicand signs are of opposite polarity. Buffer latch CXS is set for PH9 use. Flip-flop IEN is reset to stop interruptibility. Signal BCON (B control) goes true for the multiplication process. (The PH3 BCON functions and other BCON functions are described in PH9.) PH9 is enabled.

S/D47	=	C47 DXC/6 + ...
⋮		⋮
S/D31	=	C31 DXC/6 + ...
R/D47 through D31	=	DX = DXC/6 + ...
C/D47 through D41	=	CL-32P28 (typical)

d. PH4. PH4 is entered only if the multiplier is not simple normalized. The multiplicand may or may not be normalized at this point.

If the multiplier is zero, the product of the floating point multiplication is zero. Multiply iterations do not have to take place in this case and the first storage phase (PH13) of the instruction may be enabled. If the multiplier is zero, flip-flop RTZ (result zero) is set. Signal MRQ/1 (memory request) is enabled and flip-flop IEN is reset. A one is forced on private memory address line LR31 to select the odd numbered private memory register in the storage phase. Clock T10L is enabled if FMS is not being performed or if the R-field is not odd. PH13 is enabled.

If the multiplier is not zero, the first simple normalization attempt is made in this phase. The absolute value of the multiplier in flip-flops A47 through A71 (FMS) or A47 through A31 (FML) is gated to the sum bus by signal SXADD during the phase. The multiplier is shifted one hex to the left by signal AXSL4, and the exponent sum is decremented by one to compensate for the shift. Flip-flops P15 through P18 and flip-flop MP19 are used during the multiply iterations to record the number of iterations performed (the description of iteration counting may be found in the multiply word opcode description). Since performing a prenormalization shift means that there are effective zeros in the four most significant bits of the multiplier, two multiply iterations may be discarded; the iteration counter is incremented (from zero count) by two to record this fact (P18 is the 2^1 bit position). If the multiplicand is not simple normalized, it is transferred from the C"-register to the D"-register by signal DXC/6 to prepare for simple normalization in PH6. PH5 is entered even if the multiplier is now simple normalized.

e. PH5. PH5 is entered only if the multiplier was not simple normalized in PH3, and if it is nonzero. If FMS is being performed, up to four normalization shifts may be used. If FML is being implemented, up to 12 normalization shifts are used. Since the multiplier is nonzero if this phase is entered, simple normalization is always accomplished at the end of the phase.

The absolute value of the multiplier (found in PH4) is gated to the sum bus by signal SXA. The prenormalization shift is performed by signal AXSL4. The exponent sum is decremented by one to record the shift. The iteration counter is incremented by two to delete two multiply iterations. If signal FPRENR is still true, PH5 is sustained and another prenormalization shift occurs.

If the multiplier is simple normalized after any PH5 clock, signal ASN goes true and signal FPRENR goes false. PH5 is no longer sustained. If the multiplicand is not simple normalized, PH6 is enabled; if the multiplicand is also simple normalized, signal MPP goes true and PH9 is enabled. Signal MPP true causes the same actions that are described in case 3 of PH3; that is, the operands are prepared for the multiply iterations.

f. PH6. PH6 is entered from PH3 or PH6 if the multiplicand is not simple normalized. The multiplier is nonzero and simple normalized at this point.

The multiplicand in the D"-register is gated to the S-bus by signal SXD. Signal FPRENM is true if PH6 is entered. The multiplicand is shifted left one hex by signal AXSL4, and the exponent sum is decremented by one to compensate for the shift. The iteration counter remains unchanged since the number of significant multiplier digits, and not the multiplicand digits, determine the number of iterations. Buffer latch CXS is set for PH7 use. The above shift is insignificant if the multiplicand is all zeros.

g. PH7. PH7 is entered from PH6 only; the multiplicand was not simple normalized in PH3 or PH5. If the multiplicand is still not simple normalized and is nonzero, up to 13 prenormalization shifts may be used (up to five if FMS is being performed). The multiplicand is always simple normalized at the end of PH7. Clock T8L is used only for the first clock of PH7.

If the multiplicand is zero, flip-flop RTZ is set and PH13 is enabled. Signal (S/RTZ/1) causes the same actions to occur as described in PH4.

If the multiplicand is not zero, it is gated to the sum bus by signal SXA. The first prenormalization shift of PH7 is performed by signal AXSL4. The exponent sum is decremented by one to compensate for the shift. Flip-flop CXS is set for the next PH7 clock. The adjusted multiplicand is transferred to the C"-register by signal CXS at each PH7 clock. Signal CXS is true during all the clocks of PH7.

At the end of PH7, the normalized multiplicand is in the C-register. If FPRENM is still true after the first shift, PH7 is sustained and another prenormalization shift occurs.

If the multiplicand is simple normalized after any PH7 clock, signal ASN goes true and signal FPRENM goes false. PH7 is no longer sustained. Signal MPP goes true and PH9 is enabled. Signal MPP causes the same actions described in PH3 in preparation for the multiply iterations.

h. PH9. PH9 is entered from PH3, PH5, or PH7. To enter PH9, the multiplicand must be simple normalized and nonzero. The multiplier is nonzero and is in the B"-register. The prenormalized multiplier has been discarded, and the iteration counter has been incremented by two counts for every multiplier prenormalization shift. The CS-register is loaded with all ones if the original multiplier and multiplicand were of different signs (signal CSX1). The multiplicand in the D"-register is added to the contents of the CS"-register in an exclusive OR operation, and the result transferred from the sum bus to the C"-register. The C"-register now holds the multiplicand with a polarity such that the product of the multiplication is the absolute value of the true product. The absolute value of the product is needed for the later truncation of the product.

Signal BCON was set during the same phase in which signal MPP went true. Signal BCON is high during the time that the multiplier bit-pairs are examined and is an enable signal to the examination logic. When BCON first went true, bit-pair 2^1, 2^0 of the multiplier was examined and the multiplier control flip-flops DXCM, DXCLIM, and DXNCM were set up. At the same time, the B"-register shifted two bit positions to the right, bringing the second multiplier bit-pair into flip-flops B30 and B31. At the beginning of PH9, the multiplier control flip-flops have set up by the first bit-pair, and the next bit-pair is in position for examination.

Signal BCON is again enabled in PH9, and the following operations occur simultaneously at the PH9 clock:

1. The contents of the C"-register are clocked into the D"-register as a function of the multiplier control flip-flop set in the previous phase. If no multiplier control flip-flops are set, zeros are clocked into the D"-register.

2. The multiplier control flip-flops are again set up by bit-pair 2^3, 2^2 of the multiplier.

3. The B"-register shifts right two bit positions by signal BXBR2, bringing bit-pair 2^5, 2^4 of the multiplier into flip-flops B30, B31.

4. Flip-flop MIT is set. Signal MIT controls the number of multiply iterations.

5. Clock T1L is enabled for PH10.

6. Flip-flop FLMC is set if the original multiplier had significance only in the least significant hex (only two clocks of T10 are to be used).

i. PH10. PH10 is the multiply iterations phase of FAFLM. Figure 3-221 shows the register arrangement during the implementation of FMS and FML. The implementation is very similar to the implementation of opcodes MW and MI. A detailed description of the multiply process of this phase is therefore not given; only difference between FAFLM and FAMULNH are noted.

At the beginning of PH10, the D"-register holds the first partial sum in flip-flop D46 through D31, the A'-register (flip-flops A47 through A31) is all zeros (cleared in phase preceding PH9), and the CS-register holds all zeros. The C"-register holds the multiplicand in bits C47 through C31 (FML) or bits C47 through C71 (FMS). The multiplier has been shifted so that it is now in flip-flops B47 through B71 and in B4 through B31 (FML) or B11 through B31 (FMS); the four least significant bits of the multiplier have been examined and discarded.

The maximum number of multiply iterations in PH10 is 12, if no prenormalization multiplier shifts have been performed, or if FMS is being performed. The maximum number is 28 if FML is being implemented. Since each prenormalization shift of the multipliers means that there are four binary zeros in the more significant part of the multiplier, two times the number of prenormalization multiplier shifts is subtracted from 12 or 28 to give the number of multiply iterations that must be performed in PH10. Each prenormalization multiplier shift has already been recorded by up-counting the iteration counter.

For each clock (iteration) of PH10, the following events occur:

1. The D"-register is combined with A"- and CS"-registers to produce the partial product for that iteration. Figure 3-222 shows the addition process and signal routing. The sums portion of the product is clocked into flip-flops A47 through A31 and B0 through B3. As the multiply iterations progress, the less significant bits of the partial product are lost (truncated) as they spill over to the right. The carries portion of the partial product is clocked into bit positions 47 through 33 of the CS"-register. The sums and carries are generated in the same manner as in instructions MW, MI, and MH.

2. The extreme higher-order and lower-order bits of the partial product are formed by special logic, logic identical to instructions MW, MI, and MH except for signal names. Bit A47 of the sum is used only for sign-padding the partial product and is a one whenever there is a one in bits A47 or D46 or both. Bit A48 would normally be the sign bit in a straight addition, but it is a sum bit in carry-save addition since flip-flop CS48 may hold a carry after the addition. Flip-flops A49 through A31 and B0 and B1 are set by the propagate signal two orders to the left. Flip-flop B3 is set by signal B1S, equivalent to a propagate signal from bit position 33, and is true when there is a total of

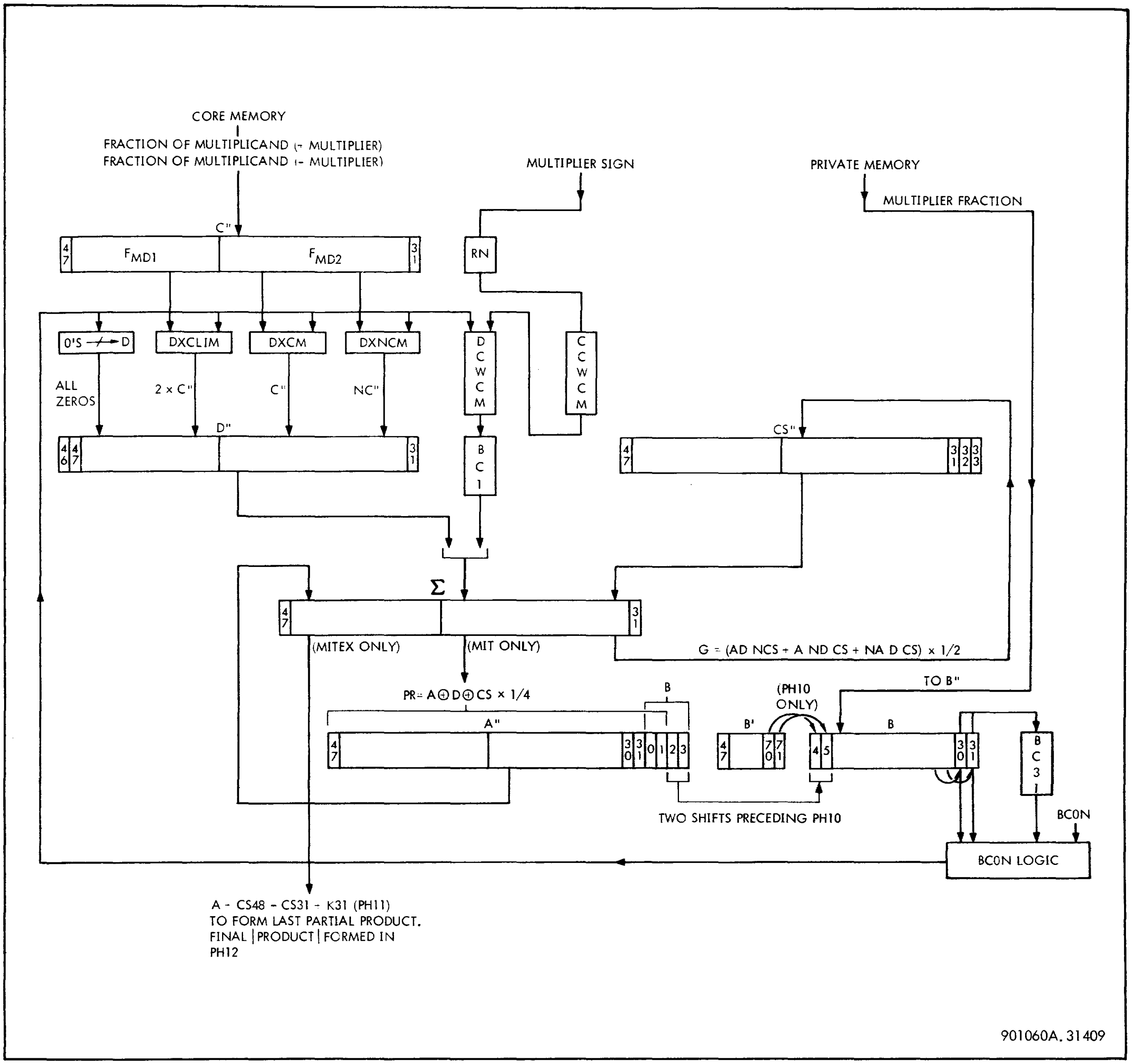


Figure 3-221. FAFLM Register Organization (Phase 10)

1 or 3 ones in bits B1, BC1, and CS33. Flip-flop BC1 is set by signal DCWCM and contains a one if the one's complement of the multiplicand was put in the D-register at the previous iteration. Paying the D"-register the one needed to make a two's complement is done by adding a one to bit B3 of the partial product one clock after the addition of the A"-, D"-, and CS"-registers. The partial product has been shifted so that bit B3 is in the same order that would have received the one if it had been added to the D"-register initially. Flip-flop B2 is set by the propagate signal for bit position 32, PRE32; this is the same as the normal propagate signal except that only the CS"- and B"-registers are combined. Flip-flop B2 is not set when there are both a PR32 signal and a generate signal from the 33rd bit position. Signal G33 amounts to a carry from bit position 33. The carry from bit position 33, if not resolved by setting flip-flop B2, is stored in flip-flop CS33. Flip-flops CS1 through CS32 are set by G0 through G31, respectively.

3. The contents of the C"-register are clocked into the D"-register as a function of the multiplier control flip-flops set during the previous clock.

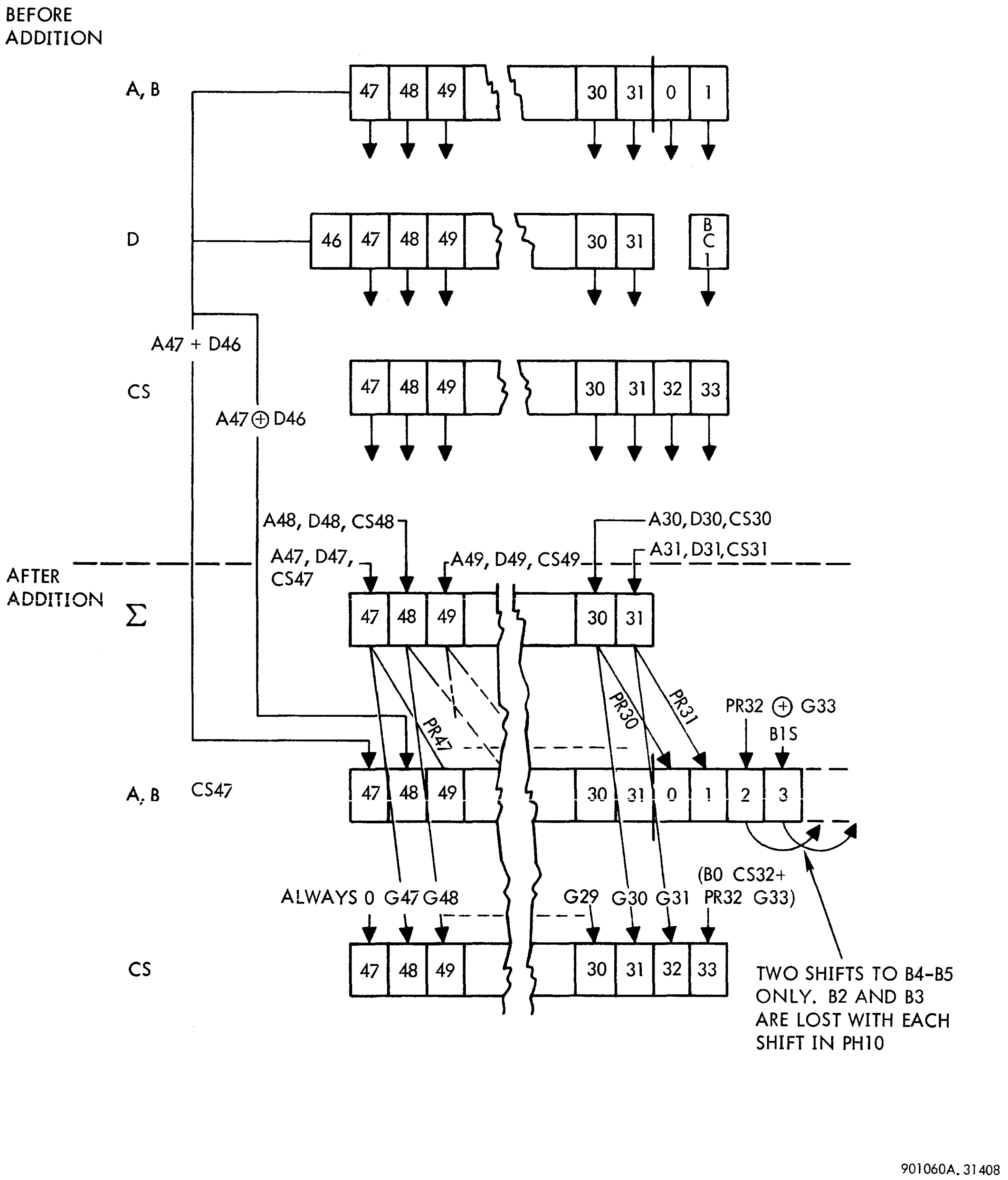


Figure 3-222. Floating Point Multiply Iteration Signal Routing

4. The next bit-pair of the multiplier is examined, and the multiplier control flip-flops are set.

5. The B"-register shift right two bit positions by signal BXBR2 which brings a new bit pair into bit positions B30 and B31.

6. The number of the iteration is counted. The iteration counter is made up of five flip-flops, P15 through P18 (cleared in PH3 by signal PX), and flip-flop MP19. The counter operates exactly as in instruction MW except that flip-flop FLMC is set on the 10th count for instruction FMS and the 28th count for instruction FML. The counter may have been up-counted before PH10 if the multiplier was prenormalized (see PH4 and PH5).

At the 26th count (10th if FMS), signal FLMC goes true. A true FLMC signal sets flip-flop MITEX at the 27th (11th) count. MITEX resets flip-flop MIT at the 28th (12th) count. At the 26 (10th) count, also, flip-flop T1L is reset to initiate the end of T1L (T1L timing lasts through PH10). The 27th (11th) count is the second-to-the-last transfer of the C"-register contents to the D"-register under control of the multiplier control flip-flop. One more transfer occurs during PH11. The last carry-save addition of the A-, D-, and CS-register occurs during the 28th (12th) clock. Two more additions occur: in PH11 a partial product is assimilated and is put in the A-register; in PH12, this partial product is added to the last multiplicand transfer in the D-register to form the final product of the multiplication. Clock T8L is enabled.

j. PH11. PH11 is used to assimilate the partial product of the multiplication from the results of the last iteration in PH10. The sum portion of this partial product is contained in flip-flops A47 through A31 and in B0 through B3. The carries are contained in flip-flops CS47 through CS33. The contents of flip-flops A47 through A71, the contents of flip-flops CS47 through CS31, and K31 (carry from the lower order bits) are added by signal SXADD and are clocked into flip-flops A47 through A31 by signal AXS. The contents of flip-flops B0 through B3, CS32, CS33, and BC1 are added by signal BXB and clocked into flip-flops B0 through B3. Flip-flops A47 through A31 and B0 through B3 now hold the last partial product; the final product is found in PH12.

At the PH11 clock, the last transfer to the D"-register takes place. The multiplier control flip-flops for this transfer were set during the last clock of PH10. A one is forced into flip-flop CS31 if the one's complement is transferred to the D"-register. Signal MRQ/1 (memory request) is enabled. Clock T8L is enabled.

k. PH12. During PH12, the final product is assimilated. This final product is the absolute value of the actual product (see PH9). The partial product in the A-register is added to the last transfer of the multiplicand in the D"-register (the transfer took place in PH11). Bit CS31 is also added, since it may contain a one if the one's complement were placed in the D"-register in PH11. The result of the addition, on S47 through S31, is clocked into the A-register. The D"-register is cleared by signal DX/1. Flip-flop NGX is set for use in PH13. Signal (S/G0003/1) is enabled if FMS is being performed and the R-field of the instruction word is odd; flip-flop G0003/1 generates signal K71 for truncation of the short product. A one is forced on private memory address line LR31 to select the odd numbered private memory register in PH13. Clock T10L is enabled if FMS is not being performed or if the R-field of the instruction word is not odd.

l. PH13. PH13 is the postnormalization phase of FAFLM. PH13 is entered from a phase preceding PH12 if the result of the multiplication is to be zero because of a zero multiplier or multiplicand; or from PH12 if the multiplier is not zero. The result, at this time, is either ready for storage or not ready for storage. The result is not ready for storage if, at the beginning of PH13 or after adjustment, bit A47 is true or there is a nonzero result that must be postnormalized.

If the result is not ready for storage (NFPRR true), it is gated to the sum bus by signal SXA during PH13. PH13 is sustained as long as signal NFPRR is true. At the first PH13 clock, clock T10L is enabled if FMS is not being performed or if the R-field of the instruction word is not odd. A one is again forced on private memory address line LR31, and flip-flop NGX is set.

One of two courses is then followed:

1. If bit A47 is true, it signifies that the absolute value of the product is +1. This can happen if the original operands were both -1/16. If both operands were -1/16, they are prenormalized to -1, and changed to +1 for the multiplication, resulting in a product of +1. The result must be shifted to the right to correct the overflow condition. The S-bus outputs are shifted one hex to the right, and the shifted result is transferred back into the A"-register at the PH13 clock. The exponent in the E-register is increased by one. The result is now ready for storage.

2. If bit A47 is false, if the result is nonzero, and if the result is not normalized, one postnormalization shift will normalize the product (only one postnormalization shift is required since the operands were both prenormalized, and there must be a one at least in the second most significant hex of the product). The result on the sum bus is shifted to the left one hex by signal AXSL4 and is clocked into the A-register. The E-register is decremented by one to adjust the exponent. The four least significant bits of the product, flip-flops in B0 through B3, are also shifted left by signal AXSL4.

When the result is ready for storage, one of three actions is taken:

1. The result is negated, and a one added for two's complement if FPR is true. The one for two's complementing

is provided by signals K31 or K71 (if FMS is being performed and if the R-field is odd). The negated result is gated to the sum bus by signal SXADD.

2. The result remains unaltered if FPR is false, and is gated to the sum bus by signal SXA.

3. Zeros are gated to the sum bus if the result is zero or if there was exponent underflow with FZ = 0.

If FML is being performed, there is no trap condition, and the R-field is not odd. Sum bus outputs S0 through S31, the LSW of the result, are transferred to the odd numbered private memory register at the PH13 clock (first or second). Sum bus outputs S48 through S71 are transferred to flip-flops A8 through A31 at the clock by signal AXAR32. The exponent result flip-flops in E1 through E7 is biased by adding 64 and is clocked into flip-flops A1 through A7. Bias is achieved by reversing the state of flip-flop E1. Ones are clocked into flip-flops CS0 through CS7 to invert the exponent if the final result is to be negative. The A-register now holds a zero in bit A0, the uninverted exponent in flip-flops A1 through A7, and the most significant part of the fraction product in flip-flops A8 through A31.

Flip-flops DRQ, T8L, and PH15 are set at the PH13 clock. Flip-flop DRQ inhibits transmission of another clock until a data release signal is received from memory.

If trap conditions exist, flip-flops TRAP and TR29 are set to provide a trap to memory location X'44' (68). Trap conditions are the following:

1. Exponent overflow. If the result is not equal to zero, exponent overflow causes an unconditional trap.

2. Underflow. Underflow causes a trap if the floating zero mode control bit, FZ, is a one.

m. PH15. PH15 is used to store the MSW of the result and to set the condition codes. If the result is not equal to zero, the MSW of the result is added to the contents of the CS-register in an exclusive OR operation (bit positions CS0 through CS7 are all ones if the result is negative and invert the exponent and the sign bit in this case). The sum bus outputs, S0 through S31, are transferred to the even numbered private memory register if no trap condition exists. Signal RWDIS disables the transfer if a trap condition exists. The sum bus outputs are also transferred to flip-flops A0 through A31 for condition code use. A one is merged into bit A31 by signal A31X1 if the result was not equal to zero. This action is necessary because a portion of the result has already been stored (PH13),and the most significant part may be zero.

Flip-flop TESTA is set to enable condition code bits 3 and 4 at the next clock. Condition code 1 is set if underflow has occurred, and if the result is not equal to zero. Condition code 2 is set if the overflow or underflow occurs, and the result is nonzero. Condition code 3 is set if the result is positive and nonzero; condition code 4 is set if the result is negative. Signal ENDE is generated.

Flip-flop PRE1 is set to begin the preparation sequence for the next instruction, and clock T6L is enabled.

3-297 Floating Point Division (FAFLD)

3-298 GENERAL. The implementation of FDL is identical with the implementation of FDS with the exception of the number of fractional digits to be operated upon. The two instructions will be discussed together, with their differences noted in the discussion.

3-299 FLOATING POINT MODE CONTROL BITS. Mode control bit FZ, floating zero, is the only control bit influencing floating point division.

If mode control bit FZ = 0, underflow or a zero result causes the result to be set equal to true zero. If FZ = 1, underflow causes the computer to trap to memory location X'44' (68). The contents of the general registers remain unchanged if the trap occurs.

If division by zero is attempted, or if exponent overflow occurs, the computer unconditionally traps to location X'44' (68).

3-300 CONDITION CODE SETTINGS. The condition code settings and their meanings for floating point division are shown in table 3-135.

3-301 BASIC STEPS IN FAFLD IMPLEMENTATION. To divide two floating point numbers in Sigma 7, the following basic steps are performed:

a. Transfer of operands (PH1 and PH3). The numerator (from private memory) and the denominator (from core memory) are transferred to the arithmetic unit registers.

b. Exponent differencing (PH2 and PH3). The uninverted biased denominator exponent is subtracted from the uninverted biased numerator exponent. The difference is the exponent of the quotient; bias has been removed by the subtraction process. This quotient exponent may be corrected as prenormalizing or postnormalizing shifts are performed on the fractions of the two operands.

c. Prenormalization (PH4-PH10). Before the two floating point numbers are divided, both are simple normalized. After the exponents are differenced (step b), the numerator and the denominator are examined; the following conditions are possible:

1. The numerator is zero, and the denominator is not simple normalized. The denominator is prenormalized to detect a zero value. The denominator is shifted right one hex at a time until it is simple normalized or is found

Table 3-135. Condition Codes for Floating Point Division

CONDITION CODE				IF NO TRAP TO LOCATION X'44' OCCURS	IF TRAP TO LOCATION X'44' OCCURS
1	2	3	4		
0	0	0	0	0 ÷ A	Impossible
0	0	0	1	Result negative	
0	0	1	0	Result positive and nonzero	
0	1	0	0	Impossible	
0	1	0	1		Overflow, result negative
0	1	1	0		Overflow, result positive and nonzero
1	0	0	0		▨
1	0	0	1		
1	0	1	0		
1	1	0	0		
1	1	0	1		Underflow, result negative, FZ = 1
1	1	1	0		Underflow, result positive and nonzero, FZ = 1

to be zero. If the denominator is nonzero, a zero quotient is produced without going through the division process. If the denominator is zero, an unconditional trap results.

2. The numerator is zero, and the denominator is simple normalized. The division is not performed, and the result is set equal to true zero.

3. The numerator is nonzero and is not simple normalized (denominator may or may not be simple normalized). The numerator is shifted right one hex at a time until it is simple normalized. The exponent difference is decremented one with each shift. The denominator is then shifted right one hex at a time until it is simple normalized or is found to be zero. The exponent difference is incremented one with each shift. If the denominator is nonzero, the division of the two operands occurs. If the denominator is zero, an unconditional trap results.

4. The numerator is nonzero and is simple normalized (denominator may or may not be simple normalized). The denominator is shifted to the right one hex at a time until it is simple normalized or is found to be zero. The exponent difference is incremented one with each shift. If the denominator is nonzero, the division of the two operands occurs. If the denominator is zero, an unconditional trap occurs.

d. Division (PH11 and PH12). The floating point numbers are divided after prenormalization of the fractions of both operands and provided that neither the denominator nor the numerator is zero. (Division methods are almost identical to the nonrestoring division methods used for fixed-point numbers (DW and DH opcodes).

e. Storage (PH13 and PH15). To bias it again, 64 is added to the exponent result. The biased exponent and the fraction result of the quotient are assimilated and are stored in memory in proper form, provided that there is no trap condition.

An example of FAFLD is shown in figure 3-223. The lettered steps correspond to the basic steps discussed.

3-302 DETAILED STEPS IN FAFLD IMPLEMENTATION. Figure 3-224 shows the FAFLD phases and the operations performed for each phase. The following discussion is supplemented by table 3-136 and by figure 3-224.

a. PH1. Signal CXCL32 is generated for use in PH2. Flip-flops CC1 and CC2 are reset for underflow and overflow tests in PH15, and flip-flop IEN is set to start interruptibility. Ones are forced into flip-flops CS0 through CS7 by signal CSX1/5 to invert the numerator exponent to its true form if the numerator is negative.

b. PH2. At the PH2 clock, the fraction part of the numerator MSW (or the complete fraction if FDS is being performed) is transferred from flip-flops A0 and A8 through A31 to A47 and A48 through A71, respectively, by signal AXAL32. The fraction part of the denominator MSW is transferred to the C-register extension in the same manner by signal CXCL32.

The numerator and denominator exponents are set up in PH2 for exponent differencing. The uninverted numerator exponent is either present in flip-flops A1 through A7 or is found by combining bits A1 through A7 with CS0 through CS7 (all ones) in PH3 in an exclusive OR operation. The uninverted numerator exponent is gated to the sum bus during PH2 and, at the PH2 clock, is transferred from the sum bus to flip-flops A0 through A7 by signal AXS/1.

The uninverted denominator exponent must be subtracted from the uninverted numerator exponent. The inverted denominator is therefore transferred to flip-flops D0 through D7 by signals DXNC (if the denominator is negative) or to DXC/6 (if the denominator is positive).

A one is forced into flip-flop CS7 at the PH2 clock by signal CS7X1 to complete the two's complemented form of the denominator exponent (needed for the subtraction). Ones are forced on core memory address line LB31 and on private memory line LR31 for possible use in PH3 (with FDL). If FDL is being implemented, signal MRQ (memory request) is enabled and flip-flop DRQ (data release) is set to transfer the least significant word of the denominator to the C-register during PH3. Clock T6RL is enabled for PH3. If FDS is being implemented, flip-flop CS/1 is set at the PH2 clock to force all zeros into the C-register in PH3.

The division is performed such that the absolute value of the quotient is generated. Flip-flop FPR (floating result polarity reversed) is set in PH2 if the final (true) quotient is to be negative.

c. PH3. During PH3, the uninverted numerator exponent in flip-flops A0 through A7, the inverted denominator exponent in flip-flops D0 through D7, and the one in flip-flops CS7 are added; their sum is gated to the sum bus. At the PH3 clock, the sum bus outputs on S0 through S7 are transferred to flip-flops E0 through E7 by signal EXS. Flip-flops E0 through E7 now hold the unbiased difference of exponents.

At the PH3 clock, the P-register is cleared by signal PX. The P-register later serves as an iteration counter for the multiply iterations. The A"-register is cleared by signal AX/1 but the contents are immediately regenerated by signals SXA/3 and AXS/4.

If FDL is being implemented, the LSW of the denominator is clocked into C0 through C31, and the LSW of the numerator is clocked into flip-flops A0 through A31. If FDS is being implemented, zeros are transferred to the C- and A-register (no gating term enabled). Flip-flop NGX is set if the numerator is negative for use in PH4. The D"-register is cleared by signal DX/1, and clock T8L is enabled.

d. PH4. PH4 is used to examine the numerator and the denominator and to initiate simple normalization for either or both. There are four conditions possible at the beginning of PH4.

1. The numerator is zero and the denominator is not simple normalized. The denominator is prenormalized in this case to detect a possible zero value. The denominator in bits C47 through C31 is transferred to flip-flops D47 through D31 at the PH4 clock. The absolute value of the numerator (in this case, zeros) is transferred to the sum bus outputs by signal SXADD and into flip-flops B47 through B31 by signal BXS (for PH8 use). PH6 is enabled.

2. The numerator is zero, and the denominator is simple normalized (and cannot be zero). The multiplication is disabled and zeros are produced for the quotient result. Flip-flop RTZ is set, and flip-flop IEN is reset to stop interruptibility. Signal MRQ/1 is generated to request the next instruction. A one is forced on private memory address line LR31 for PH13 use. Clock T10L is enabled if FDL is being performed, and PH13 is enabled.

3. The numerator is nonzero and is not simple normalized (the denominator may or may not be simple normalized). The first prenormalization attempt is made in PH4. Further prenormalization attempts are made in PH5 if necessary. Signal FPRENR is true if the numerator is not normalized. The absolute value of the numerator fraction is gated to S47 through S31 by signal SXADD and is shifted one hex to the left into flip-flops A47 through A27 by signal AXSL4.

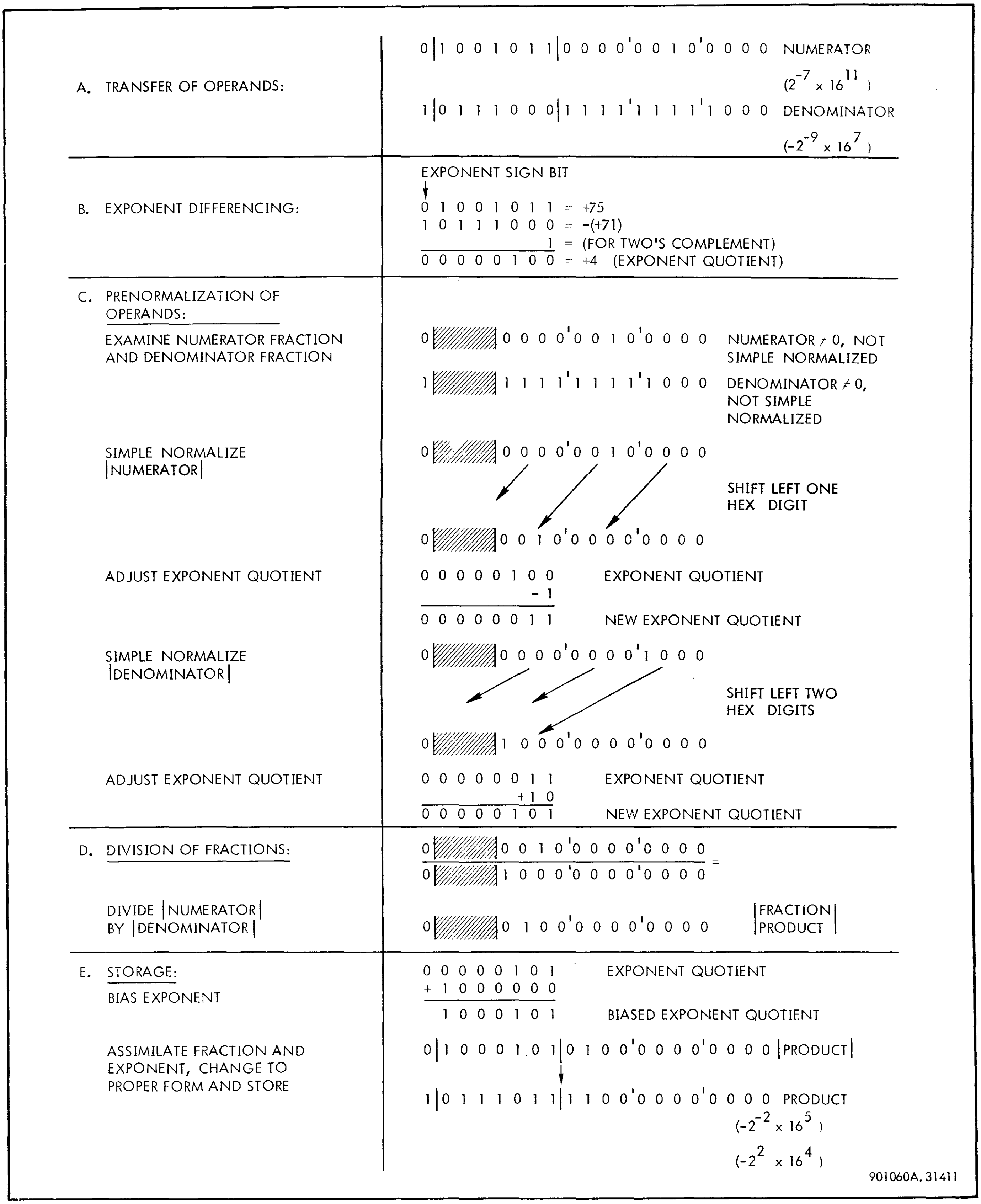

A. TRANSFER OF OPERANDS:
0|1 0 0 1 0 1 1|0 0 0 0'0 0 1 0'0 0 0 0 NUMERATOR
(2^-7 x 16^11)
1|0 1 1 1 0 0 0|1 1 1 1'1 1 1 1'1 0 0 0 DENOMINATOR
(-2^-9 x 16^7)
B. EXPONENT DIFFERENCING:
EXPONENT SIGN BIT
0 1 0 0 1 0 1 1 = +75
1 0 1 1 1 0 0 0 = -(+71)
1 = (FOR TWO'S COMPLEMENT)
0 0 0 0 0 1 0 0 = +4 (EXPONENT QUOTIENT)
C. PRENORMALIZATION OF OPERANDS:
EXAMINE NUMERATOR FRACTION AND DENOMINATOR FRACTION
0 0 0 0 0'0 0 1 0'0 0 0 0 NUMERATOR ≠ 0, NOT SIMPLE NORMALIZED
1 1 1 1 1'1 1 1 1'1 0 0 0 DENOMINATOR ≠ 0, NOT SIMPLE NORMALIZED
SIMPLE NORMALIZE |NUMERATOR|
0 0 0 0 0'0 0 1 0'0 0 0 0
SHIFT LEFT ONE HEX DIGIT
0 0 0 1 0'0 0 0 0'0 0 0 0
ADJUST EXPONENT QUOTIENT
0 0 0 0 0 1 0 0 EXPONENT QUOTIENT
- 1
0 0 0 0 0 0 1 1 NEW EXPONENT QUOTIENT
SIMPLE NORMALIZE |DENOMINATOR|
0 0 0 0 0'0 0 0 0'1 0 0 0
SHIFT LEFT TWO HEX DIGITS
0 1 0 0 0'0 0 0 0'0 0 0 0
ADJUST EXPONENT QUOTIENT
0 0 0 0 0 0 1 1 EXPONENT QUOTIENT
+1 0
0 0 0 0 0 1 0 1 NEW EXPONENT QUOTIENT
D. DIVISION OF FRACTIONS:
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
=
DIVIDE |NUMERATOR| BY |DENOMINATOR|
0 0 1 0 0'0 0 0 0'0 0 0 0
|FRACTION PRODUCT|
E. STORAGE:
BIAS EXPONENT
0 0 0 0 0 1 0 1 EXPONENT QUOTIENT
+ 1 0 0 0 0 0 0
1 0 0 0 1 0 1 BIASED EXPONENT QUOTIENT
ASSIMILATE FRACTION AND EXPONENT, CHANGE TO PROPER FORM AND STORE
0|1 0 0 0 1 0 1|0 1 0 0'0 0 0 0'0 0 0 0 |PRODUCT|
1|0 1 1 1 0 1 1|1 1 0 0'0 0 0 0'0 0 0 0 PRODUCT
(-2^-2 x 16^5)
(-2^2 x 16^4)
901060A. 31411

Figure 3-223. Example of FAFLD Implementation

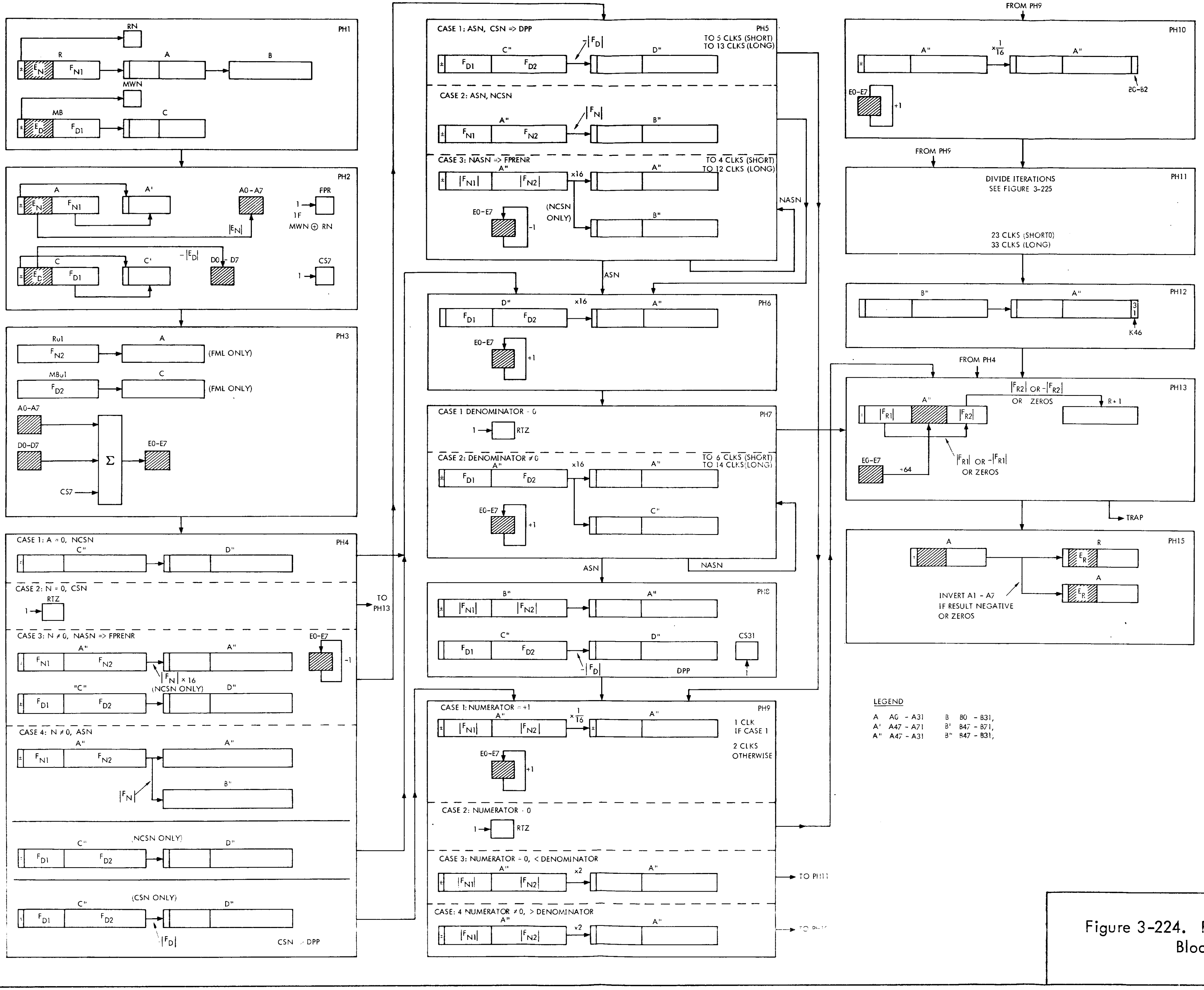


Figure 3-224. FAFLD Implementation, Block Diagram

901060A. 31414

Table 3-136. Floating Divide (FAFLD), Phase Sequence

Phase	Function Performed	Signals Involved	Comments
PREP	Same as general preparation sequence		
PH1 T4RL	RR0-RR31 ↛ A0-A31	AXRR = RNXRR/2 PH2 + ... = FAFL PH1 + ...	MSW of numerator
	Set flip-flop RN if numerator negative	S/RN = RR0 RNRXRR0 + ... = RR0 PH1 FAFL + ...	Stores numerator sign
	MB0-MB31 → C0-C31	Preparation control	MSW of denominator
	Set flip-flop MWN if denominator negative	S/MWN = C0C16 FAMDSF PH1 + ... = MB0 NP32 CXMB + ...	Stores denominator sign
	Set flip-flop CXCL32	S/CXCL32 = FAFL PH1 + ...	
	Reset flip-flop CC1	R/CC1 = FAFL PH1 + ...	
	Reset flip-flop CC2	R/CC2 = FAMDSF PH1 NFAMULH + ...	
	Force all ones into CS0-CS7	CSX1/5 = FAFL PH1 + ...	To invert numerator exponent if numerator negative
	Set flip-flop IEN	S/IEN = FAMDSF NFAMUL PH1 + ...	Start interruptibility
PH2 T6L	A0, A8-A31 ↛ A47, A48-A71	AXAL32 = FAFL PH2	MSW of numerator ↛ A'
	C0, C8-C31 → C47, C48-C71	CXCL32 = Set in PH1	MSW of denominator ↛ C'
	A0-A31 + CS0-CS7 → S0-S31 if numerator negative or	SXPR = FAFL PH2 RN + ...	Uninverted numerator exponent
	A0-A31 → S0-S31 if numerator negative	SXA = FAFL PH2 NRN + ...	
	S0-S7 ↛ A0-A7	AXS/1 = FAFL PH2 + ...	
	NC0-NC7 ↛ D0-D7 if denominator positive or	DXNC = FAFL PH2 MWN + ...	Inverted denominator exponent
	C0-C7 ↛ D0-D7 if denominator negative	DXC/6 = FAFL PH2 + NMWH + ...	

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH2 T6L (Cont.)	Force a one into CS7	CS7X1 = FAFL PH2 + NAFAFLM	For two's complement subtraction (PH3)
	Set flip-flop FPR if final quotient is to be negative	S/FPR = FAFLMD PH2 (MWN ⊕ RN) + ...	Result must be reversed
	Force a one onto LB31 address line	S/LB31/1 = FAFL PH2 + ...	
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = FAFL PH2 + ...	
	Enable MRQ }	MRQ = FAFL PH2 + ...	Memory request for LSW
	Set flip-flop DRQ } if FDL	S/DRQ = FAFL PH2 NO2 + ...	Inhibits transmission of another clock until data release signal received
	Enable T6RL }	T6RL = FAFL PH2 + NO2 + ...	
	Set flip-flop CX/1 if FDS	S/CX/1 = FAFL PH2 O2 + ...	
PH3 T6L (FDS) T6RL (FDL)	A0-A7 + D0-D7 + CS0 → S0-S7	SXK/1 = FAFL PH3 + ...	
		SXPR/1 = FAFL PH3 + ...	
	S0-S7 ⇸ E0-E7	EXS = FAFL PH3 + ...	Exponent quotient
	Clear P-register	PX = FAFL PH3 + ...	
	Clear A"-register	AX/1 = FAFL PH3 + ...	
	A47-A71 ⇸ S47-S71	SXA/3 = FAFL PH3 + ...	
	S47-S71 ⇸ A47-A71	AXS/4 = FAFL PH3 (NFAFLM ASN)	A"-regenerated
	RR0-RR31 ⇸ A0-A31 } if FDL	AXRR = FAFL PH3 (NFAFLM ASN) NO2	LSW of numerator (zeros if FDS)
	MB0-MB31 → C0-C31 }	See PH2	LSW of denominator (zeros if FDS)
	Set flip-flop NGX if numerator is negative	S/NGX = FAFL PH3 N(FAFLM ASN) RN + ...	For PH4 use
	Clear D"-register	DX/1 = FAFLMD PH3 + ...	
	Set flip-flop T8L	R/NT8L = T8L = FAFL PH3 N(FAFLM ASN) FAFLMD + ...	

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4	Case 1: Numerator = 0, denominator not simple normalized		
	C47-C31 —/→ D47-D31	DXC/6 = FAFLMD PH4 NCSN + ...	
	| A47-A31| ——→ S47-S31	SXADD = FAFLMD PH4 + ...	NGX set in PH3
	S47-S31 —/→ B47-B31	BXS = FAFLD PH4 NCSN + ...	Save numerator for later use
	Enable PH6	BRPH6 = FAFLD PH4 NCSN (ASN + A4731Z) + ...	
	Case 2: Numerator = 0, denominator is simple normalized		
	Set flip-flop RTZ	S/RTZ = (S/RTZ/1) + ... = FAFLD PH4 CSN NA4731Z + ...	
	Enable MRQ/1	MRQ/1 = (S/RTZ/1) + ...	Memory request for next instruction. Also gates Q —/→ P
	Reset flip-flop IEN	R/IEN = (S/RTZ/1) + ...	
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = (S/RTZ/1) + ...	
	Set flip-flop T10L	S/T10L = (S/RTZ/1) + ...	
	Enable PH13	BRPH13 = (S/RTZ/1) + ...	Division need not be performed
	Case 3: Numerator ≠ 0 and is not simple normalized	FPRENR = FAFLM PH4 + NASN + ...	Prenormalize R-operand
	|A47-A31| ——→ S47-S31	SXADD = FAFLMD PH4 + ...	
	S51-S31 x 16 ——→ S51-S31	AXSL4 = FPRENR + ...	Shift numerator one hex to the right
	Decrement E0-E7 by one	MCTE1 = FPRENR + ...	
	C47-C31 —/→ D47-D31 if NCSN	DXC/6 = FAFLMD PH4 NCSN + ...	To prepare for PH6
	Case 4: Numerator ≠ 0 and simple normalized		
	|A47-A31| ——→ S47-S31	SXADD = FAFLMD PH4 + ...	} |Numerator| ——→ A"
	S47-S31 —/→ A47-A31	AXS = FAFLMD PH4 ASN + ...	

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH4 T8L (Cont.)	C47-C31 ⇸ D47-D31 NCSN only	DXC/6 = FAFLMD PH4 NCSN + ...	
	S47-S31 ⇸ B47-B31	BXS = FAFLD PH4 NCSN + ...	
	Enable PH6	BRPH6 = FAFLD PH4 NCSN (ASN + A4731Z) + ...	
	Enable DPP	DPP = FAFLD PH4 ASN CSN	
	DPP only C47-C31 ⇸ D47-D31 if denominator is negative	DXC/6 = DPP MWN + ...	|Denominator| ⇸ D"
	NC47-NC31 ⇸ D47-D31 if denominator is positive	DXNC/1 = DPP NMWN + ...	
	Force a one into CS31 if denominator is positive	S/CS31 = DXNC/1	
	Clear B" register	BX/1 = FPP + ...	
	Set flip-flop T8L	RNT8L = T8L = DPP + ...	
	Enable PH9	BRPH9 = DPP + ...	Division may be performed
PH5 T6L	(Entered from PH4, case 3) 1-5 clocks (FMS) 1-13 clocks (FML)		
	Case 1: ASN CSN => DPP	Same actions as PH4, case 4 (enable PH9)	
		DPP = FAFLD PH5 ASN CSN + ...	
	Case 2: ASN NCSN	NFPRENR = FAFLMD PH5 ASN	
	A47-A31 → S47-S31	SXA = FAFL PH5 + ...	
	S47-S31 ⇸ B47-B31		
	Advance to PH6	BRPH5 = FPRENR PH5 + ...	
	Case 3: NASN	FPRENR = FAFLMD PH5 NASN	
	A47-A31 → S47-S31	SXA = FAFL PH5 + ...	
	(S41-S31) x 16 ⇸ A47-A27	AXSL4 = FPRENR + ...	Shift absolute value of numerator one hex to the left

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH5 T6L (Cont.)	Decrement E0-E7 by one	MCTE1 = FPRENR + ...	
	Sustain PH5	BRPH5 = FPRENR PH5 + ...	
	S47-S31 $\nrightarrow$ B47-B31	BXS = FAFLD PH5 NCSN + ...	Save normalized | numerator |
PH6 T6L	(Entered from PH4, cases 1 and 4, PH5 case 2)	FPRENM = FAFLMD PH6 + ...	
	D47-D31 $\longrightarrow$ S47-S31	SXD = FAFLMD PH6 + ...	
	S51-S31 x 16 $\nrightarrow$ A47-A27	AXSL4 = FPRENM + ...	Shift denominator one hex to the left
	Increment E0-E7 by one	PCTE1 = FPRENM FAFLD + ...	
	Enable S/CXS	S/CXS = FPRENM NA4731Z + FAFLMD PH6 + ...	
PH7 T6L	(Entered from PH6 only) 1-6 clocks (FMS) 1-14 clocks (FML)		
	Case 1: Denominator = 0		
	Set flip-flop CC2	S/CC2 = FAFLD PH7 A4731Z	(N ÷ 0)
	Set flip-flop RTZ	(S/RTZ/1) = FAFLMD PH7 NA4731Z + ...	
		Other signals same as PH4, case 2	
	Case 2: Denominator ≠ 0, NASN	FPRENM = FAFLM PH7 NASN + ...	
	A47-A31 $\longrightarrow$ S47-S31	SXA = FAFLMD PH7 + ...	
	S51-S31 x 16 $\nrightarrow$ A47-A31	AXSL4 = FPRENM + ...	Shift denominator one hex to the left
	S47-S31 $\longrightarrow$ C47-C31	CXS = See PH6 and PH7	
	Increment E0-E7 by one	PCTE1 = FPRENM FAFLD + ...	
	Set flip-flop CXS	S/CXS = FPRENM NA4731Z + ...	
	Sustain PH7	BRPH7 = FPRENM NA4731Z + ...	

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH7 T6L (Cont.)	Case 3: Denominator ≠ 0, ASN	NFPRENM = FAFLM PH7 ASN + ...	
	Advance to PH8	BRPH7 = FPRENM NA4731Z + ...	
PH8 T6L	(Entered from PH7 only)		
	B47-B31 → S47-S31	SXB = FAFLD PH8 + ...	
	S47-S31 $\nrightarrow$ A47-A31	AXS = FAFLD PH8 + ...	Retrieve normalized |numerator|
	Enable DPP	DPP = FAFLD PH8 + ...	
		Other signals same as PH4, case 4	
PH9 T8L or T6L	(Entered from PH4, PH5 or PH8 as the function of DPP) 1-2 clocks, 2 clocks only if |numerator| = 1 (in which case second clock is T6L)		
	Case 1: Numerator = +1		
	A47-A31 → S47-S31	SXA = FAFLD PH9 + ...	
	S47-S27 x 1/16 $\nrightarrow$ A51-A31	AXSR4 = FAFLD PH9 A47 + ...	Shift absolute value of numerator one hex to the right to avoid overflow in product
	Increment E0-E7 by one	PCTE1 = FAFLD PH9 A47 + ...	
	Sustain PH9 (advance to next PH9 clocked)	BRPH9 = FAFLD PH9 A47 + ...	
	Case 2: Numerator = 0		
	Set flip-flop RTZ	(S/RTZ/1) = FAFLD PH9 A4731Z + ...	
		Other signals same as PH4, case 2	Enable PH13
	Case 3: Numerator ≠ 0, numerator < denominator		
	A47-A31 → S47-S31	SXA = FAFLD PH9 + ...	
	S47-S31 x 2 $\nrightarrow$ A47-A31	AXSL1 = FAFLD PH9 NA47 + ...	Scale for division

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH9 T8L or T6L (Cont.)	Set flip-flop T8L	R/NT8L = T8L = FAFLD PH9 NK46 NA4731Z + ...	
	Enable PH11	BRPH11 = FAFLD PH9 NK46 NA4731Z + ...	
	Case 4: Numerator $\neq$ 0, numerator $\geq$ denominator		
	A47-A31 $\longrightarrow$ S47-S31	SXA = FAFLD PH9 + ...	
	S47-S31 x 2 $\nrightarrow$ A47-A30	AXSL1 = FAFLD PH9 NA47 + ...	Scale for division
	Advance to PH10		
	All cases:		
	Hold CS31 for first iteration in division (PH11)	R/CS31 = N(FAFLD PH9) + ...	
PH10 T6L	(Entered only from PH9, case 4)		
	A47-A31 $\longrightarrow$ S47-S31	SXA = FAMDSF PH10 + ...	
	S47-S31 x 1/16 $\nrightarrow$ A51-A31	AXSR4 = FAFLD PH10 + ...	Shift numerator right one hex
	S28-S30 $\nrightarrow$ B0-B2	Gated by FAFLD PH10 + ...	
	Increment E0-E7 by one	PCTE1 = FAFLD PH10 + ...	
	Hold CS31 for first PH11 iteration	R/CS31 = N(FAFLD PH10) + ...	
PH11 T6L	23 clocks if FDS, 35 clocks if FDL. During each clock, the following events occur		
	Enable signal DIT	DIT = FAMDSF/D PH11 + ...	
	A47-A31 + D47-D31 + CS31 $\longrightarrow$ S47-S31	SXADD = DIT + ...	
	S48-S31 $\nrightarrow$ A47-A31	AXSL1 = DIT + ...	
	B0 $\nrightarrow$ A31	S/A31 = B0 FAMDSF/D N [FAFLD (PH11 P26 + PH12)] + ...	
	B47-B31 x 2 $\nrightarrow$ B47-B30	BXBL1 = DIT + ...	

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH11 T6L (Cont.)	Clock a quotient bit into B31	S/B31 = K46 FAFLD PH11 + ...	
	NC47-NC31 —/→ D47-D31	DXNC/D = FAFLD PH11 (MWN ⊕ K46) + ...	Polarity of denominator clocked into D" and CS31 must be opposite to the residue clocked into A"
	Force a 1 into CS31	S/CS31 = DXNC/D + ...	
	C47-C31 —/→ D47-D31	DXC/D = FADIV PH11 N(MWN ⊕ K00) + ...	
	Count iterations, using P26-P31	PCTP1 = DIT NDITEX + ...	
	Sustain PH11	BRPH11 = DIT NDITEX + ...	
	<u>On the 22nd clock (FDS) or the 55th clock (FDL):</u>		
	Other events listed above		
	Enable DITEX	DITEX = FAFLD O2 (P27 P29 P30) + FAFLD P26 P27 P29 P30 + ...	Initiate end of iterations
	Stop sustaining PH11	BRPH11 = DIT NDITEX + ...	
	Enable MRQ/1	MRQ/1 = DIT DITEX + ...	Memory request for next instruction
	<u>On the 23rd clock (FDS) or the 55th clock (FDL):</u>		
	Set flip-flop T8L	R/NT8L = T8L = FAFLMD PH11 + ...	
	Reset flip-flop IEN	R/IEN = DIT DITEX + ...	Stop interruptibility
PH12 T8L	B47-B31 ——→ S47-S31	SXB = FAFLD PH12 + ...	} Completes quotient
	S49-S31 x 2 —/→ A48-A30	AXSL1 = FAFLD PH12 + ...	
	K46 —/→ A31	S/A31 = K46 FAFLD PH12 + ...	
	Clear D-register	DX/1 = FAFL PH12 + ...	
	Set flip-flop NGX	S/NGX = FAFL PH12 + ...	
	Force a one onto LR31 address line	R/NLR31/2 = LR31/2 = FAFL PH12 + ...	
	Set flip-flop T10L if FDL	S/T10L = FPRD PH12 + ... = FAFL NO2 PH12 + ...	

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH13 T10L if FDL T6L (Cont.)	One clock only	FPRR = FAFLD PH13 + ...	
	NA47-NA31 + K → S47-S31 if product must be negative	SXADD = FPRR N(RTZ + FEUF NFZ) FPR + ...	
	A47-A31 → S47-S31 if product must be positive	SXA = FPRR N(RTZ + FEUF NFZ) NFPR + ...	
	All zeros into S47-S31 if result is zero or if exponent underflow with FZ = 0	No gating term enabled	
	S0-S31 ↛ RW0-RW31 if FML and no trap	RW = FPRR FPRD NFTRAP + ...	Store LSW of result
	S48-S71 ↛ A8-A31	AXSR32 = FPRR N(FAFLD O2)	Most significant fraction quotient bits
	E1-E7 + 64 ↛ A1-A7	AXE = FPRR + ...	(NE1 ↛ A1, E2-E7 A2-A7) uninverted exponent plus bias
	Force ones into CS0-CS7 if result negative	CSX1/5 = FPRR FPR + ...	
	Set flip-flop DRQ	S/DRQ = FPRR + ...	Inhibit transmission of another clock until data release signal received
	Set flip-flop T8L	R/NT8L = T8L = FPRR + ...	
	Set flip-flops TRAP and TR29 for trap to X'44' (68) if trap conditions exist	FTRAP = FEOF + FEUF FZ + CC2	Exponent overflow
		FEOF = FAFL NRTZ NE0 E1	Exponent overflow
		FEUF = E0 NE1 NRTZ FAFL	Exponent underflow with FZ = 1. See PH7
		S/TR29 = FPRR FTRAP + ...	
	Enable PH15	BRPH15 = FPRR + ...	
PH15 T8L	A0-A31 ⊕ CS0-CS7 → S0-S31 if result does not equal zero	SXPR = FAFL PH15 N(RTZ + FEUF NFZ) + ...	Invert exponent if result negative
	S0-S31 → RW0-RW31 if NTRAP	RW = FAMDSF NFASHFX PH15 + ...	Store MSW of result

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

(Continued)

Table 3-136. Floating Divide (FAFLD), Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PH15 T8L (Cont.)	S0-S31 —/→ A0-A31	AXS = FAFL PH15 + ...	
		RWDIS = TRAP NINTRAPF + ...	Disable storage
	Set flip-flop TESTA	S/TESTA = FAMDSF NFASHFX PH15 + ...	For CC3 and CC4 use
	Merge a one into CS31 if result is nonzero	A31X1 = FAFL PH15 N(RTZ + FEUF NFZ) + ...	For CC3 test
	Set flip-flops CC1, CC2, CC3, and CC4 as applicable	S/CC1 = FAFL PH15 FEUF + ...	Exponent underflow
		S/CC2 = FAFL PH15 (FEUF + FEOF) + ...	Exponent underflow or overflow
		S/CC3 = TESTA NA0 NA0031Z + ...	Result greater than zero
		S/CC4 = TESTA A0 + ...	Result less than zero
	Enable ENDE	ENDE = FAMDSF PH15 + ...	
	Set flip-flop PRE1	S/PRE1 = ENDE (NHALT + FUEXU) N(S/INTRAPF)	
	Enable clock T6L	T6L = NT1L NT4L NT8L NT10L NRESET	

Mnemonic: FDS (3E, BE), FDL (1E, 9D)

The exponent difference in the E-register is decremented by one to compensate for the shift. If the denominator is not simple normalized, it is transferred to flip-flops D47 through D31 by signal DXC/6 in preparation for normalization in PH6. PH5 is entered.

4. The numerator is nonzero and is simple normalized (the denominator may or may not be simple normalized). The absolute value of the numerator is gated from flip-flops A47 through A31 to the sum bus outputs S47 through S31 by signal SXADD and back into flip-flops A47 through A31 by AXS. If the denominator is not simple normalized, it will be prenormalized to detect a possible zero value. The denominator is transferred to the D"-register, the absolute value of the numerator is transferred to the B"-register, and PH6 is enabled as in case 1.

If the denominator is simple normalized (the denominator is nonzero), the division may take place. In this case, signal DPP (divide preparation) goes true, signifying that the two operands are being set up for the division. When signal DPP is true, one of two actions occur. If the denominator in the C"-register is negative, it is transferred to the D"-register by signal DXC/6. If the denominator is positive, the inverted denominator is transferred to the D"-register by signal DXNC/1, and a one is forced into flip-flop CS31 to make up the two's complement of the inverted denominator. The B-register is cleared by signal BX/1. Clock T8L and PH9 are enabled.

e. PH5. PH5 is entered only if the numerator was not simple normalized at the beginning of PH4. At this point, the numerator may be simple normalized because of the one prenormalizing shift in PH4 or it may need further prenormalization shifting. Up to five clocks (FDS) or thirteen clocks (FDL) are used in PH5 to perform the prenormalization. There are three possible conditions at the start of PH5:

1. Numerator is simple normalized and denominator is simple normalized. If both ASN and CSN are true, signal DPP goes true and initiates the actions described in PH4, case 4.

2. Numerator is simple normalized and denominator is simple normalized. A true ASN disables PH5 and PH6 is entered to simple normalize the denominator. The absolute value of the simple normalized numerator in the A-register is gated to the sum bus by signal SXA and into the B-register by signal BXS at the PH5 clock. The numerator must be retained for later division.

3. Numerator is not simple normalized (denominator may or may not be normalized). If signal ASN is false, signal FPRENR is true, causing another prenormalization shift to be performed on the numerator. The fraction is shifted by signal AXSL4 and the exponent difference is decremented by one to compensate for the shift. PH5 is sustained by signal FPRENR. At each PH5 clock, the contents of the A"-register are gated to the sum bus and, if CSN is false, are transferred to the B"-register by signal BXS. On the last clock of PH5, the B"-register saves the simple normalized numerator if simple normalization is to be performed on the denominator.

```
FPRENR = FAFLMD  PH5  NASN  + ...

S47    = A47  SXA  + ...
 .             .
 .             .
 .             .
S31    = A31  SXA  + ...

  SXA  = FAFL  PH5  + ...

S/A47  = S51  AXSL4  + ...
  .            .
  .            .
  .            .
S/A27  = S31  AXSL4  + ...
```

f. PH6. PH6 is used to perform the first prenormalization shift on a denominator that is not simple normalized. The denominator may be zero at this point, but this condition is not detected until PH7. PH6 is entered only from PH4 or PH5.

During PH6, the denominator in the D"-register is gated to sum bus outputs S47 through S31 by signal SXD. Signal FPRENM is true during this phase and causes a prenormalization shift to occur by enabling signal AXSL4. The denominator on the sum bus outputs are shifted one hex to the left and are clocked into flip-flops A47 through A27. The exponent difference in the E-register is incremented by one to compensate for the shift. (The shift is meaningless if the denominator is zero). Buffer latch CXS is set for PH7 use.

g. PH7. PH7 is entered only from PH6 and is used to perform simple normalization shifts of the denominator, if necessary, and to detect a zero denominator. Up to 6 clocks (FDS) or 14 clocks (FDL) may be used to perform further normalization shifts on the denominator. The following conditions may be present at the start of PH7:

1. Denominator = 0. If the denominator equals zero, division is impossible and the computer traps to location X'44' (68). Flip-flop CC2 is set at the PH7 clock. Signal (S/RTZ/1) goes true with all zeros in flip-flops A47 through A71, and initiates the actions described in PH4. Step 2 (PH13 is entered).

2. Denominator does not equal zero and is not simple normalized (another prenormalization shift is to be performed). Signal FPRENM is true and causes a prenormalizing shift to occur by enabling signal AXSL4. The denominator in flip-flops A47 through A31 is gated to the sum bus outputs S47 through S31 by signal SXA. The sum bus outputs S47 through S31 by signal SXA. The sum bus outputs are shifted one hex to the left and are transferred back to flip-flops

A47 through A27. The exponent difference in the E-register, is incremented by one to compensate for the shift. Buffer latch CXS is set for use in the next clock of PH7. PH7 is sustained.

3. Denominator is simple normalized. If the denominator is simple normalized, signal ASN goes true, and PH7 is not sustained for another clock. PH8 is then entered.

During all of the above cases, the sum bus outputs are transferred to C47 through C31 during each PH7 clock so that the simple normalized denominator is in the C"-register at the end of PH7 (or PH13 has been enabled for the zero denominator case).

h. PH8. If the denominator underwent simple normalization in PH6 and PH7, the absolute value of the normalized numerator was temporarily stored in the B"-register. During PH8, the numerator is clocked into the A"-register. Signal DPP goes true in PH8, initiating the same actions as described in PH4, step 4. PH9 is enabled.

$$\begin{array}{lcl} S47 & = & B47\ SXB + \ldots \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ S31 & = & B31\ SXB + \ldots \end{array}$$

$$SXB = FAFLD\ PH8 + \ldots$$

$$\begin{array}{lcl} S/A47 & = & S41\ AXS + \ldots \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ S/A31 & = & S31\ AXS + \ldots \end{array}$$

i. PH9. PH9 is entered from PH4, PH5, or PH8 after signal DPP goes true. At this point, the A"-register holds the absolute value of the simple normalized numerator, in the range $1/16 \leq N \leq 1$ or equal to zero. (The zero case is possible only when PH9 is entered from PH8.) The D"-register holds the negated form of the absolute value of the denominator with the absolute denominator in the range $1/16 \leq D \leq 1$. The absolute value of the quotient of the division, if performed at this point, would be $1/16 \leq N \leq 1$ divided by $1/16 \leq D \leq 1$, or in the range $1/16 \leq Q \leq 16$. PH9 and PH10 are used to adjust the numerator so that the absolute value of the quotient will always be a normalized number in the range $1/16 \leq Q < 1$. There are two clocks in PH9 if the value of the numerator is +1; the first clock in this case is T8L and the second is T6L. In all other cases, only one T8L clock occurs. The following cases are possible.

1. Numerator is +1. If the numerator is +1, the absolute quotient is in the range from 1 to 16. The numerator is shifted one hex to the right to bring the quotient into the acceptable range. The numerator in flip-flops A47 through A31 is gated to the sum bus by signal SXA. At the PH9 clock, the sum bus outputs are shifted one hex to the right into flip-flops A51 through A31 by signal AXSR4. The exponent difference in the E-register is incremented by one to compensate for the shift. PH9 is sustained, and one of cases 2, 3, or 4 is enabled.

2. Numerator equals zero. If the numerator is equal to zero, the quotient is also zero (the denominator must be nonzero for PH9 to be entered). Flip-flop RTZ is set, and PH13 is enabled in the same manner as PH4, case 2. The division does not take place.

3. Numerator is nonzero and is less than the denominator. Bit D47 is a one (denominator negated), and bit A47 is a zero (numerator in absolute form). If the numerator is smaller than the denominator, there is no carry into the sign bit position (sum of numerator and denominator in a hypothetical addition would be negative); hence, there is no carry into bit position 46. If K46 = 0, therefore, the numerator is smaller than the denominator; if NA4731Z = 1, the numerator is nonzero. If the conditions exist, the absolute value of the quotient must be in the normalized range and the division may take place. The numerator in flip-flops A47 through A31 is gated to the sum bus by signal SXA and is shifted one bit position to the left by signal AXSL1 back to the A"-register to prescale the numerator for the division (see PH11). Clock T8L and PH11 are enabled.

4. Numerator is nonzero and is greater than or equal to the denominator (but not equal to +1). If these conditions exist the absolute value of the quotient is greater than or is equal to one and is less than 16. The numerator must be shifted right to bring the quotient into the normalized range. The numerator is prescaled, as in case 3, and PH10 is entered to perform the right shift.

In cases 1 through 4, flip-flop CS31 remains set to hold the one for the two's complemented (negated) denominator found in the phase where signal DPP was enabled.

j. PH10. PH10 is entered from PH9 only when case 4 of PH9 applies. The numerator in A47 through A30 is gated to the sum bus by signal SXA. The sum bus outputs S47 through S31 are shifted one hex to the right and are clocked back into S51 through S31 by signal AXSR4. Sum bus outputs S28 through S30 are clocked into B0 through B2, respectively. The exponent difference in the E-register is incremented by one to compensate for the shift. The one in CS31 is retained by not resetting the flip-flop.

k. PH11. PH11 is the divide iteration phase of FAFLD. Figure 3-225 shows the register arrangement during the implementation of FDS and FDL. The implementation is very similar to the implementation of opcodes DW and DH. A detailed description of the division process of this phase is therefore not given; only differences between FAFLD and FADIV are noted.

At the beginning of PH11, the A"-register holds two times the adjusted numerator in absolute form. The numerator was

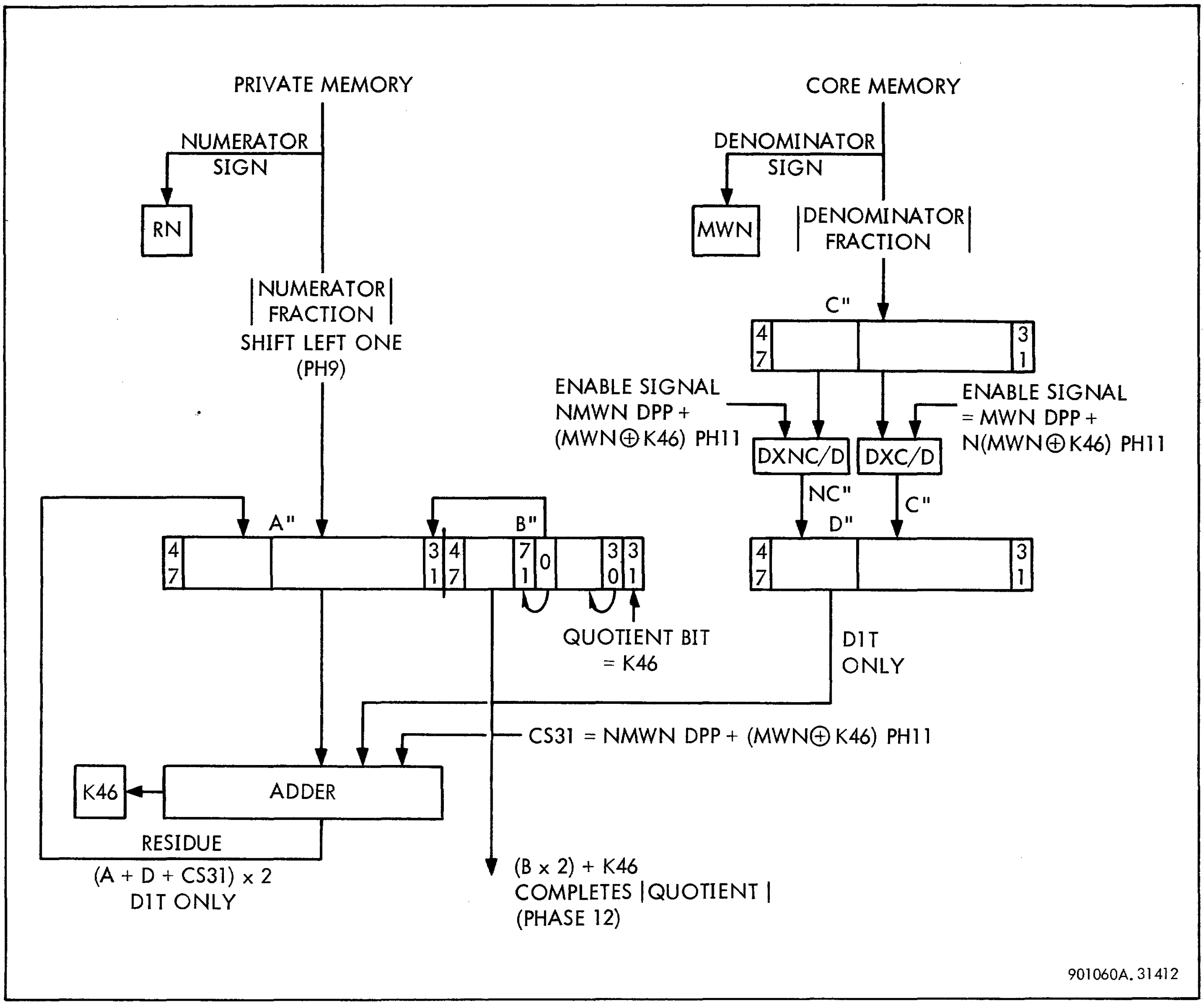


Figure 3-225. FAFLD Register Organization (Phase 11)

shifted one bit position to the right in PH9 so that the first quotient bit produced is the 2^{-1} bit. The D"-register and flip-flop CS31 contain the first denominator transfer from the C"-register. The transfer was made so that the D"-register contains the negated denominator, which is added to the numerator at the first clock of the nonrestoring division process.

There are 23 divide iterations in PH11 if FDS is being performed, or 55 divide iterations if FDL is being implemented. Signal DIT (divide iterations) is true as long as the iterations are to be performed. For each clock (iteration) of PH11, the following events occur:

1. The numerator is added to the contents of the D-register and flip-flop CS31 and the residue (see opcodes DW and DH) is shifted one bit position to the left. The residue is clocked back into the A-register.

2. The contents of flip-flop B0 are transferred to flip-flop A31. Flip-flops B0 through B2 may contain a portion of the numerator (see PH10), and these least significant bits must be shifted into the residue as the iterations progress. Bit B0 is transferred to flip-flop A31 with each clock until the 31st clock is reached, although only three transfers are actually necessary.

3. The B-register is shifted one bit position to the left by signal BXBL1. The B-register holds the partial quotients. Shifting one bit position to the left opens up B31 to receive a new quotient bit.

4. Another quotient bit is clocked into flip-flops B31. The quotient bit is a one when K46 is a one, and is a zero when K46 is a zero. K46 takes the place of K00 used in instruction DW and DH and is the end carry for the nonrestoring division process.

5. The denominator or the inverted denominator is transferred from the C"-register to the D"-register. The transfer is made so that the contents of the D"-register are opposite polarity to the residue in the A"-register and enable the next addition in the nonrestoring process to occur. The true denominator is transferred by signal DXC/D if the residue is negative and the true denominator is positive or if the residue is positive and the true denominator is negative. The inverted denominator is transferred by signal DXNC/D if both the residue and the true denominator have like signs. A one is clocked into flip-flop CS31 in the latter case to make up the two's complement of the denominator when the next addition takes place.

6. The number of the iteration is counted. Flip-flops P26 through P31 make up a six bit iteration up-counter for instruction FDS and FDL. When the counter indicates 22 for FDS or 55 for FDL, signal DITEX goes true, enabling signal MRQ/1 to request the next instruction. Signal BRPH11 goes false as DITEX is enabled. At the next clock, flip-flop T8L is set and flip-flop IEN is reset.

If FDS is being performed, bits 2^{-1} through 2^{-23} of the quotient are in flip-flops B9 through B31 at the end of PH11. If FDL is in effect, bits 2^{-1} through 2^{-55} of the quotient are in flip-flops B49 through B31. One more iteration must be performed to complete the absolute value of the product.

l. PH12. PH12 is used for the last iteration of the division process. The partial quotient in the B"-register is shifted one bit position to the left and transferred into flip-flops A47 through A30 of the A"-register by signal AXSL1. The last quotient bit (2^{-24} if FDS or 2^{-56} if FDL) is clocked into flip-flop A31. The A"-register now holds the complete absolute quotient.

Also at the PH12 clock, the D"-register is cleared by signal DX/1. Flip-flop NGX is set for possible negation of the quotient in PH13. A one is forced on private memory address line LR31 to select the odd numbered private memory register for storage of the quotient. Flip-flop T10L is set if FDL is being performed.

m. PH13. At the start of PH13, the absolute value of the quotient is in flip-flops A8 through A31 if FDS is being performed or is in flip-flops A48 through A31 if FDL is in effect. The result is ready for storage and signal FPRR is true. The quotient is normalized unless flip-flop RTZ is set. PH13 and PH15 are the two storage phases of FAFLD. One of three actions is now taken:

1. The result is negated and a one is added for two's complementing if FPR is true. The one for two's complementing is provided by K31 (NGX set in PH12). The negated result is gated to the sum bus by signal SXADD.

2. The result remains unaltered if FPR is false, and is gated to the sum bus by signal SXA.

3. Zeros are gated to the sum bus if the result is zero or if there was exponent underflow with FZ = 0.

If FDL is being performed and if there is no trap condition, sum bus outputs S0 through S31, the LSW of the result, are transferred to the odd numbered private memory register at the PH13 clock. Sum bus outputs S48 through S71 are transferred to flip-flops A8 through A31 at the clock by signal AXAR32. The exponent result flip-flops in E1 through E7 is biased by adding 64 and clocked into flip-flops A1 through A7. Bias is achieved by reversing the state of flip-flop E1. Ones are clocked into flip-flops CS0 through CS7 to invert the exponent if the final result is to be negative. The A-register now holds a zero in bit A0. It holds the uninverted exponent in bits A1 through A7 and the most significant part of the fraction product in bits A8 through A31.

Flip-flops DRQ, T8L, and PH15 are set at the PH13 clock. Flip-flop DRQ inhibits transmission of another clock until a data release signal is received from memory.

If trap conditions exist, flip-flops TRAP and TR29 are set to provide a trap to memory location X'44' (68). Trap conditions are the following:

1. Exponent overflow. If the result is not equal to zero, exponent overflow causes an unconditional trap.

2. Underflow. Underflow causes a trap if the floating zero mode control bit, FZ, is a one.

n. PH15. If the result is not equal to zero, the MSW of the result is added to the contents of CS-register in an exclusive OR operation (bits CS0 through CS7 are all ones if the result is negative and invert the exponent and sign bit in this case). The sum bus outputs, S0 through S31 are transferred to the even numbered private memory register if no trap condition exists. Signal RWDIS disables transfer if a trap condition exists. The sum bus outputs are also transferred to flip-flops A0 through A31 for condition code use. A one is merged into bit A31 by signal A31X1 if the result was not equal to zero. This action is necessary because a portion of the result has already been stored (PH13) the most significant part of which may be zero.

Flip-flop TESTA is set to enable condition code bits 3 and 4 at the next clock. Condition code 1 is set if underflow has occurred and if the result is not equal to zero. Condition code 2 is set if overflow or underflow occurs and if the result is nonzero. Condition code 3 is set if the result is positive and nonzero; condition code 4 is set if the result is negative. Signal ENDE is generated.

3-303 PROCESSOR CONTROL PANEL

The Processor Control Panel (PCP) is divided into two separate functional sections. The upper section (MAINTENANCE SECTION) is reserved for maintenance controls and indicators; the lower section, which is not labeled, contains the controls and the indicators for the computer operator. All controls and indicators appearing on the lower section of the PCP are functionally duplicated on the Free-Standing Console (FSC). The Sigma 7 Processor Control Panel is shown in figure 2-1. The PCP switches,with their designators and output signals are listed in table 3-137. The PCP indicators, with their designators and associated lamp driver origins, are listed in table 3-138. The POWER and INTERRUPT pushbuttons in the center of the panel are illuminated. The NORMAL MODE, RUN, and WAIT assemblies are indicators only.

3-304 Common Output Signals

Three logic signals are not directly associated with any single control switch, but are the result of several combinations of switch settings. These logic signals are KAS/1, KAS/2, and NKAS/B. Table 3-139 shows the conditions under which these signals are true.

3-305 POWER Switch

The POWER switch is an illuminated push on-push off button that connects ac power to the computer and to units under its control. When the switch is engaged and the ac power is available at the power source, +8 volts are supplied to the internal indicator to light the pushbutton. When the power is turned off, the indicator goes out. The events that take place in the CPU when the power goes on and off are described in the paragraphs on power on and power off sequence under Entering PCP Phases.

3-306 CONTROL MODE Switch

The CONTROL MODE switch distributes +8 and -8 volts to points in the PCP depending on whether it is set to LOCAL, REMOTE, or LOCK. These voltages can enable or disable operation of the switches to which they are connected. When the CONTROL MODE switch is in the LOCAL position, all control functions can be carried out from the PCP. When the switch is in the REMOTE position, most control functions can be carried out from the Free-Standing Console (FSC). All controls and indicators on the maintenance section except the CLOCK MODE and SENSE switches are operative; all switches on the lower portion of the PCP except the POWER and INTERRUPT switches are inoperative. When the switch is in the LOCK position, all switches on the PCP are inoperative except the POWER, INTERRUPT, SENSE, and AUDIO switches. In the LOCK position, the switch connects +8 or -8 volts to the switch circuits in such a way that the COMPUTE switch appears to be in the RUN position; the WATCHDOG TIMER and INTERLEAVE SELECT switches appear to be in the NORMAL position; and the PARITY ERROR MODE and CLOCK MODE switches appear to be in the CONT position.

3-307 CLOCK MODE Switch

When the CLOCK MODE switch is in the CONT position, signals KC and KSC are false and signal NKC/B is true. The program sequences normally and clock enable signal CE is active. When the switch is placed in the center position, signal KSC goes true, signal KC goes false, and signal NKC/B remains true. Clock enable signal CE is inhibited by the following equation.

CE = N(KSC NSC2) (...)

If the switch is held in the SINGLE STEP position, signal KSC remains true, signal KC goes true, and signal NKC/B goes false. Signal CE goes true long enough to allow a single clock pulse. Further clock pulses are inhibited by SCD as follows:

CE = N(KSC SCD) (...)

SCD = SCD SC2 + SC2 SCEN CL

S/SC2 = SC1

S/SC1 = KSC KC

When the switch is returned to the center position or is set to CONT, NKC/B goes true, SCD drops, and clock pulses are again enabled as follows:

R/SC1 = NKC/B SCD

R/SC2 = NSC1

During single step operation, the watchdog timer is inoperative because signal KSC holds signal WDTR true and the watchdog timer is held in the reset state.

WDTR = KSC + ...

This logic is explained further in the following paragraphs.

3-308 WATCHDOG TIMER Switch

When the WATCHDOG TIMER switch is in the NORMAL position, switch output signal KWDTR is false, and the watchdog timer counter is reset by signal WDTR only at interruptible points during program execution.

WDTR = IEN + ENDE + KWDTR + ...

where IEN signifies interrupt enable and ENDE is true during the last phase of instruction execution. If the counter reaches a count of 39 before an interruptible point is reached, timer runout has occurred, and the program traps to location X'46'.

Table 3-137. PCP Control Switches

Switch Name	Designator	Output Signals	Switch Position	Logic Level
CONTROL MODE	S3	None	LOCAL REMOTE LOCK	Supplies +8V and -8V voltage to switch circuits
WATCHDOG TIMER	S12	KWDTR	NORMAL OVERRIDE	False True
INTERLEAVE SELECT	S11	KINLVSEL	NORMAL DIAGNOSTIC	False True
PARITY ERROR MODE	S10	KHOP	HALT CONT	True False
SENSE 1 SENSE 2 SENSE 3 SENSE 4	S9 S8 S7 S6	KSS1 KSS2 KSS3 KSS4	ON ON ON ON	True True True True
CLOCK MODE	S5	KC	CONT Center SINGLE STEP	False False True
		NKC/B	CONT Center SINGLE STEP	True True False
		KSC	CONT Center SINGLE STEP	False True True
REGISTER DISPLAY	S4	KD	ON Off	True False
REGISTER SELECT	S18	KDSHI	CS_HE D_H S_N	True True True

(Continued)

Table 3-137. PCP Control Switches (Cont.)

Switch Name	Designator	Output Signals	Switch Position	Logic Level
			A_H	True
			B_H	True
			D_L	False
			P	False
			S_L	False
			A_L	False
			B_L	False
			CS_L	False
		KSXB	B_L	True
			B_H	True
		KSXCS	CS_L	True
			CS_HE	True
		KSXD	D_L	True
			D_H	True
		KSXP	P	True
		KSXS	S_L	True
			S_H	True
		KSXA	A_L	True
			A_H	True
AUDIO	S2	None	ON	
POWER	S19	None		
CPU RESET/ CLEAR	S18	KCPURESET NKCPURESET/ B	Pressed	True False

(Continued)

Table 3-137. PCP Control Switches (Cont.)

Switch Name	Designator	Output Signals	Switch Position	Logic Level
I/O RESET	S17	KIORESET	Pressed	True
LOAD	S16	KFILL/B	Pressed	True
UNIT ADDRESS	S15A	KUA21	0-3	False
			4-7	True
		KUA22	0, 1, 4, 5	False
			2, 3, 6, 7	True
		KUA23	0, 2, 4, 6	False
			1, 3, 5, 7	True
	S15B	KUA24	0-7	False
			8-F	True
		KUA25	0-3, 8-B	False
			4-7, C-F	True
		KUA26	0, 1, 4, 5, 8, 9, C, D	False
			2, 3, 6, 7, A, B, E, F	True
		KUA27	0, 2, 4, 6, 8, A, C, E	False
			1, 3, 5, 7, 9, B, D, F	True
	S15C	KUA28	0-7	True
			8-F	False
		KUA29	0-3, 8-B	True
			4-7, C-F	False
		KUA30	0, 1, 4, 5, 8, 9, C, D	True
			2, 3, 6, 7, A, B, E, F	False
		KUA31	0, 2, 4, 6, 8, A, C, E	True
			1, 3, 5, 7, 9, B, D, F	False
SYS RESET/ CLEAR	S14	KSYSR/B	Pressed	True
		NKSYSR		False
INTERRUPT	S13	KINTRP	Pressed	True
		NKINTRP/B		False
INSERT	S21	KPSW1/B	PSW1	True
			PSW2	False
		KPSW2/B	PSW1	False
			PSW2	True
STORE	S23	KSTORK/B	SELECT ADDR	True
			INSTR ADDR	False
		KSTORQ/B	INSTR ADDR	True
			SELECT ADDR	False

(Continued)

Table 3-137. PCP Control Switches (Cont.)

Switch Name	Designator	Output Signals	Switch Position	Logic Level
DATA	S43	KCLEARD/B	CLEAR	True
			ENTER	False
		KENTERD/B	ENTER	True
			CLEAR	False
INSTR ADDR	S20	KINCRE/B	INCREMENT	True
			HOLD	False
		NKHOLD	HOLD	True
			INCREMENT	False
DISPLAY	S22	KDISPLAK/B	SELECT ADDR	True
			INSTR ADDR	False
		KDISPLAQ/B	INSTR ADDR	True
			SELECT ADDR	False
COMPUTE	S42	KRUN/B	RUN	True
			IDLE	False
			STEP	False
		KSTEP/B	RUN	False
			IDLE	False
			STEP	True
SELECT ADDRESS	S24	KSP31	1	True
	.	.		
	.	.	Center	False
	.	.		
	S40	KSP15	0	False
ADDR STOP	S41	KADDRSTOP	ON	True
DATA 0	S75	KNC0	1	True
			Center	True
			0	False
		KS0	1	True
			Center	False
			0	False
DATA 31	S44	KNC31	1	True
			Center	True
			0	False
		KS31	1	True
			Center	False
			0	

Table 3-138. PCP Indicators

Indicator Name	Designator	Lamp Driver Origin
INSTRUCTION ADDRESS	DS39	Q31/L
	⋮	⋮
	DS55	Q15/L
TRAP		
ARITH	DS56	AM/L
DEC	DS57	DM/L
MODE		
MAP	DS58	MAPF/L
SLAVE	DS59	NMASTER/L
FLOAT MODE		
NRMZ	DS60	FNF/L
ZERO	DS61	FZ/L
SIG	DS62	FS/L
CONDITION CODE		
1	DS66	CC1/L
2	DS65	CC2/L
3	DS64	CC3/L
4	DS63	CC4/L
POINTER	DS29	RP27/L
	⋮	⋮
	DS33	RP23/L
INTRPT INHIBIT		
EXT	DS34	EI/L
I/O	DS35	II/L
CTR	DS36	CIF/L
WRITE KEY	DS37	WK0/L
	DS38	WK2/L
DISPLAY		
31	DS67	S31/L
⋮	⋮	⋮
0	DS98	S0/L
POWER	DS28	+8V
NORMAL MODE	DS25	Switch 55C-1 (on PT16 power supplies)
RUN	DS24	RUN/L
WAIT	DS23	WAIT/L

(Continued)

Table 3-138. PCP Indicators (Cont.)

Indicator Name	Designator	Lamp Driver Origin
INTERRUPT	DS22	CPILITE/L
MEMORY FAULT		
1	DS21	MF0/L
⋮	⋮	⋮
8	DS14	MF7/L
ALARM	DS13	ALARM/L
PHASES	DS12	PRE4/L
PREPARATION	DS11	PRE2/L
	DS10	PRE1/L
PCP	DS9	PCP4/L
	DS8	PCP2/L
	DS7	PCP1/L
EXECUTION	DS6	PH8/L
	DS5	PH4/L
	DS4	PH2/L
	DS3	PH1/L
INT/TRAP	DS2	INTRAP2/L
	DS1	INTRAP1/L

Table 3-139. Switch Settings for Signals KAS/1, KAS/2, and NKAS/B

SWITCH NAME	KAS/1*		KAS/2 *†	NKAS/B**
	LOAD Switch Off and Any One of Switch Positions Shown	LOAD Switch Pressed and All Switch Positions as Shown	LOAD Switch Off and Any One of Switch Settings Shown	LOAD Switch Off and All of Switch Positions as Shown
INSERT	PSW1 or PSW2	Center	PSW1 or PSW2	Center
INSTR ADDR	INCREMENT	Center or HOLD	INCREMENT	Center or Hold
STORE	INSTR ADDR or SELECT ADDR	Center	INSTR ADDR or SELECT ADDR	Center
DISPLAY	INSTR ADDR or SELECT ADDR	Center	INSTR ADDR or SELECT ADDR	Center
DATA	CLEAR or ENTER	Center	CLEAR or ENTER	Center
COMPUTE	RUN or STEP	IDLE	RUN or STEP	IDLE

* Always true with CONTROL MODE switch in LOCK position
† Always true with LOAD switch pressed
** Always false with LOAD switch pressed

Setting the switch to OVERRIDE holds WDTR true with KWDTR, and the counter is held in the reset state. Timer runout cannot occur in this condition.

3-309 INTERLEAVE SELECT Switch

When the INTERLEAVE SELECT switch is in the NORMAL position, signal KINLVSEL is false and has no effect on the memory. With the switch in the DIAGNOSTIC position, KINLVSEL is true, and memory interface signal /ORIL/ is developed to override the interleaving operation in core memory.

3-310 PARITY ERROR MODE Switch

When the PARITY ERROR MODE switch is in the CONT position, switch output signal KHOP is false. A memory parity error causes one of eight signals from memory, MFL0 through MFL7, to go true, according to the memory module in which the parity error occurs. These signals light corresponding memory fault indicators on the PCP, and interrupt signal PEI is transmitted to the memory parity interrupt level.

Signal PEI is generated from parity error signal PEC from memory as follows:

$$\mathrm{PEI} = \mathrm{PEF1} + \mathrm{PEF2}$$

$$\mathrm{S/PEF1} = \mathrm{PEL}\ \ \mathrm{NINTEN}$$

$$\mathrm{PEL} = \mathrm{PE} + \dots$$

$$\mathrm{PE} = \mathrm{/PEC/}\ \text{(memory interface)}$$

$$\mathrm{NINTEN} = \mathrm{N(KHOP}\ \ \mathrm{PEL)}$$

When the switch is in the HALT position, signal KHOP is true and computer clock enable signal CE is driven low, halting operation. The equation is as follows:

$$\mathrm{CE} = \mathrm{NINTEN}\ \ \mathrm{NCROF}\ \ \mathrm{NWDTA}\ (\dots)$$

When parity error signal PEC is received from memory, PEL goes true, and as a result, NINTEN goes low, disabling CE. The memory fault indicators are lighted in the same manner as when the switch is in the CONT position.

With KHOP true, a halt on fault signal is transmitted to memory as follows:

$$\mathrm{/HOF/} = \mathrm{KHOP}$$

Signal HOF in memory causes parity error signal PE, which is in the module where the error occurred, to latch, thereby inhibiting the memory delay line in that module and preventing further access. While PE is latched, clock enable signal CE in the CPU is also held low. The latch may be reset by pressing the SYS RESET/CLEAR button on the PCP. This switch also turns off the memory fault indicators.

3-311 AUDIO Switch

The AUDIO switch, when in the ON position, connects signal AUDIO/L to the computer speaker. Signal AUDIO/L is generated from the MUSIC or the ALARM flip-flop as follows:

$$\mathrm{AUDIO/L} = \mathrm{AUDIO}$$

$$\begin{aligned}\mathrm{AUDIO} &= \mathrm{MUSIC}\ \ \mathrm{NALARM}\\ &\quad + \mathrm{ALARM}\ \ \mathrm{KRUN/B}\ \ \mathrm{1KC}\end{aligned}$$

The MUSIC flip-flop, referred to in the Sigma 7 reference manual as the program-controlled frequency flip-flop, is toggled by a Write Direct instruction in the internal mode. The frequency, therefore, is determined by the frequency of the Write Direct instructions.

The ALARM flip-flop is set and is reset by internal mode Write Direct instructions or by a RESET signal. The AUDIO switch and the ALARM flip-flop connect a 1-kHz frequency to the computer speaker.

3-312 SENSE Switches

The four SENSE switches have outputs KSS1 through KSS4, which are true when the switches are in the 1 position and are false when the switches are in the zero position. These outputs are used to set the condition code flip-flops during a Read Direct or Write Direct internal mode instruction as follows:

$$\begin{aligned}\mathrm{(S/CC1\text{-}S/CC4)} &= \mathrm{FARWD}\ \ \mathrm{SW1}\ \ \mathrm{B1619Z}\\ &\quad \mathrm{(KSS1\text{-}KSS4)}\end{aligned}$$

If the CONTROL MODE switch is in the REMOTE position, the +8 volts are removed from the SENSE switches, holding their outputs low and making them inoperative.

3-313 REGISTER DISPLAY Switch

The REGISTER DISPLAY switch is used in connection with the REGISTER SELECT switch to display the contents of the internal CPU registers shown on the PCP panel around the REGISTER SELECT switch in the DISPLAY indicators. When the REGISTER DISPLAY switch is on, signal KD is generated unless the CLOCK MODE switch is in the CONT position, thereby removing the +8 voltage from the KD circuit. Signal KD is used to generate signal SDIS, which enables sum bus information to be placed on the DISPLAY indicator lines S0/L through S31/L. The equation for SDIS is as follows:

$$\mathrm{SDIS} = \mathrm{KD}\ \ \mathrm{KSC}\ \ \mathrm{NSC1}$$

where KSC indicates that the CLOCK MODE switch is not in the CONT position and where NSC1 indicates that the CLOCK MODE switch is not in the SINGLE STEP position.

3-314 REGISTER SELECT Switch

The REGISTER SELECT switch has 11 positions which allow the display of the contents of the registers indicated if the

REGISTER DISPLAY switch is in the ON position and if the CLOCK MODE switch is not in the CONT position. The switch positions, the logic signals generated, and the registers displayed are shown in table 3-140.

Table 3-140. REGISTER SELECT Switch Logic

Switch Position	Information Displayed	DISPLAY Indicators Lighted	Switch Output Logic Signals
A_L	A0-A31	0-31	KSXA
A_H	A47-A71	7-31	KSXA, KDSHI
B_L	B0-B31	0-31	KSXB
B_H	B47-B71	7-31	KSXB, KDSHI
CS_L	CS0-CS31	0-31	KSXCS
CS_HE	E0-E7 CS48-CS71	0-7 8-31	KSXCS, KDSHI, KSXE
D_L	D0-D31	0-31	KSXD
D_H	D47-D71	7-31	KSXD, KDSHI
P	P15-P31	15-31	KSXP
S_L	S0-S31	0-31	KSXS
S_H	S47-S71	7-31	KSXS, KDSHI

The switch outputs gate the contents of the internal registers onto the sum bus as follows:

SXA = KSXA SDIS + ...

SXB = KSXB SDIS + ...

SXCS = KSXCS SDIS + ...

SXD = KSXD SDIS + ...

SXP = KSXP SDIS + ...

Signal SDIS is false when the COMPUTE switch is in the IDLE position and the REGISTER DISPLAY switch is off or when the CLOCK MODE switch is not in the center position or when the REGISTER DISPLAY switch is at S_L or S_H and the contents of the D-register are displayed by way of the sum bus:

SXD = PCP2 NRESET/F NSDIS + ...

The sum bus outputs are gated onto display indicator lines S0/L through S31/L by KSHI or NKSHI. Signal KSHI is generated as follows:

KSHI = KD KDSHI KSC

Signal KDSHI is true when the REGISTER SELECT switch is set at A_H, B_H, CS_HE, D_H, or S_H to display bits 47 through 71 of the selected register. If KDSHI is false, bits 0 through 31 of the register are displayed. A typical equation is as follows:

S8/L = S8 NKSHI + S48 KSHI

In the case of the setting CS_HE, display indicators 0 through 7 display the contents of the E-register and indicators 8 through 31 display bits 48 through 71 of the CS-register. The following are typical equations:

S0/L = S0 NKSHI + E0 KSXE

S7/L = S7 NKSHI + S47 KSHI NKSXE
+ E7 KSXE

KSXE = KSXCS KSHI

The second two equations cause S7/L to display E7 rather than CS47 because KSXE is always true in this position.

The P-register is displayed by way of S15I through S31I as follows:

(S15-S31) = (S15I-S31I)

(S15I-S31I) = (P15-P31 SXP)

3-315 I/O RESET Pushbutton

The I/O RESET pushbutton, when pressed with the COMPUTE switch in the IDLE position, generates signal KIORESET. This signal is gated with NKAS/B to develop signal RIO and its resulting cable driver signal /RIO/ which is to be sent to the IOP. The /RIO/ signal is used in the IOP to reset to the ready condition all peripheral devices under the control of the CPU and to reset all status, interrupt, and control indicators in the input/output system. Signal KIORESET is held false when the CONTROL MODE switch is in the LOCK position. The I/O RESET switch does not affect the current operation of the CPU.

3-316 UNIT ADDRESS Thumbwheels

The three UNIT ADDRESS thumbwheels decode into binary numers the three hexadecimal numbers which represent the address of the device, the device controller, and the IOP, respectively. Signals KUA21, KUA22, and KUA23 are the outputs of the left-hand switch, signals KUA24, KUA25, KUA26, and KUA27 are the outputs of the center switch, and signals KUA28, KUA29, KUA30, and KUA31 are the outputs of the right-hand switch. These signals are used in the logic that generates the bootstrap program so that the value of the UNIT ADDRESS switches may be inserted into the appropriate memory location during load operation.

3-317 INTERRUPT Switch

The INTERRUPT switch is an illuminated pushbutton with two outputs, KINTRP and KINTRP/B. When the pushbutton is pressed, signal KINTRP goes true and signal KINTRP/B goes false. Signal KINTRP is used to generate SR13 which, in turn, is used to set interrupt flip-flop IS13. Signal NINTRP/B is used to reset service request interlock control flip-flop CNLK. When the control panel interrupt is operative, the program goes to interrupt location X'5D'. The interrupt does not occur if the interrupt inhibit bit in position 6 of program status doubleword 2 is set.

The indicator in the INTERRUPT pushbutton is lighted by signal CPILITE/L, which is the output of a lamp driver whose input is INT11. Signal INT11 is true when flip-flops IP13 and IS13 in the interrupt circuits are set, indicating that the control panel interrupt signal has been received and that the interrupt is in the waiting state. The indicator goes out when the control panel interrupt goes to the active state.

3-318 SELECT ADDRESS Switches

The SELECT ADDRESS switches have outputs KSP15 through KSP31 which are used for three purposes:

a. To select the virtual address at which the program is to be halted when the ADDR STOP switch is in the ON position. In this case, the outputs are compared with the LM address lines to generate an ADMATCH signal if the address on the core memory address lines is the same as the address in the switch settings.

b. To select the virtual address of a memory location to be altered if the STORE switch is in the SELECT ADDR position. In this case, the switch outputs are used to set the P-register flip-flops as in the following equation:

S/P18 = KSP18 PXK

PXK = PCP4 KSTORK/B + ...

c. To select the virtual address of a memory location whose contents are to be displayed if the DISPLAY switch is in the SELECT ADDR position. In this case, the switch outputs are used to set the P-register flip-flops as in the following equation:

S/P18 = KSP18 PXK

PXK = PCP4 KDISPLAK/B + ...

3-319 32 DATA Switches

The 32 DATA switches have outputs KNC0 through KNC31 and KS0 through KS31. The KNC terms are true if the corresponding switch is in the 1 or center position and false if the switch is in the 0 position. The KS terms are true if the corresponding switch is in the 1 position and are false if the switch is in the center or 0 position.

When the single DATA switch is operated, KENTERD/B is true, and, in the appropriate PCP phase, the D-register is loaded according to the 32 DATA switch settings. The logic sequence for these operations is explained in the paragraphs on the DATA ENTER/CLEAR function.

When one of the 32 DATA switches is in the 1 position, the associated KNC and KS signals are both true and the corresponding D-register flip-flop is set. When the switch is in the center position, the KNC signal is true and the KS signal is false, and the corresponding D-register flip-flop remains in its previous state. If the switch is in the 0 position, the KNC and KS signals are both false and the corresponding flip-flop is reset.

3-320 PCP Phase Sequencing

The PCP control operations described in the following paragraphs require one or more PCP phase sequences. These phase sequences are controlled by seven flip-flops, PCP1 through PCP7. The logic for the PCP phase flip-flops is shown in the PCP phase sequence charts.

3-321 ENTERING PCP PHASES (See figure 3-226.) The PCP phases are entered when signal HALT/1 is true unless a trap or interrupt is active or an Execute instruction is in process.

S/PCP1 = NPCP3 [HALT/1 ENDE N(S/INTRAPF)
NFUEXU]

Signal HALT/1 is true whenever the HALT flip-flop is set or when DCSTOP is true. This occurs under the following conditions:

a. The COMPUTE switch is set to IDLE.

S/HALT = NKRUN/B PRE1 + ...

b. A Wait instruction has been executed.

S/HALT = FUWAIT PH1 + ...

c. The ADDR STOP switch is set to ON, and the value in the SELECT ADDRESS switches is equal to the address on the memory address bus.

HALT/1 = DCSTOP

DCSTOP = KADDRSTOP MR ADMATCH

ADMATCH = (LM15 KSP15)-(LM22 KSP22)
(LB23 KSP23)-(LB31 KSP31)

d. A trap or an interrupt has occurred, and the instruction taken from the trap or interrupt location is not

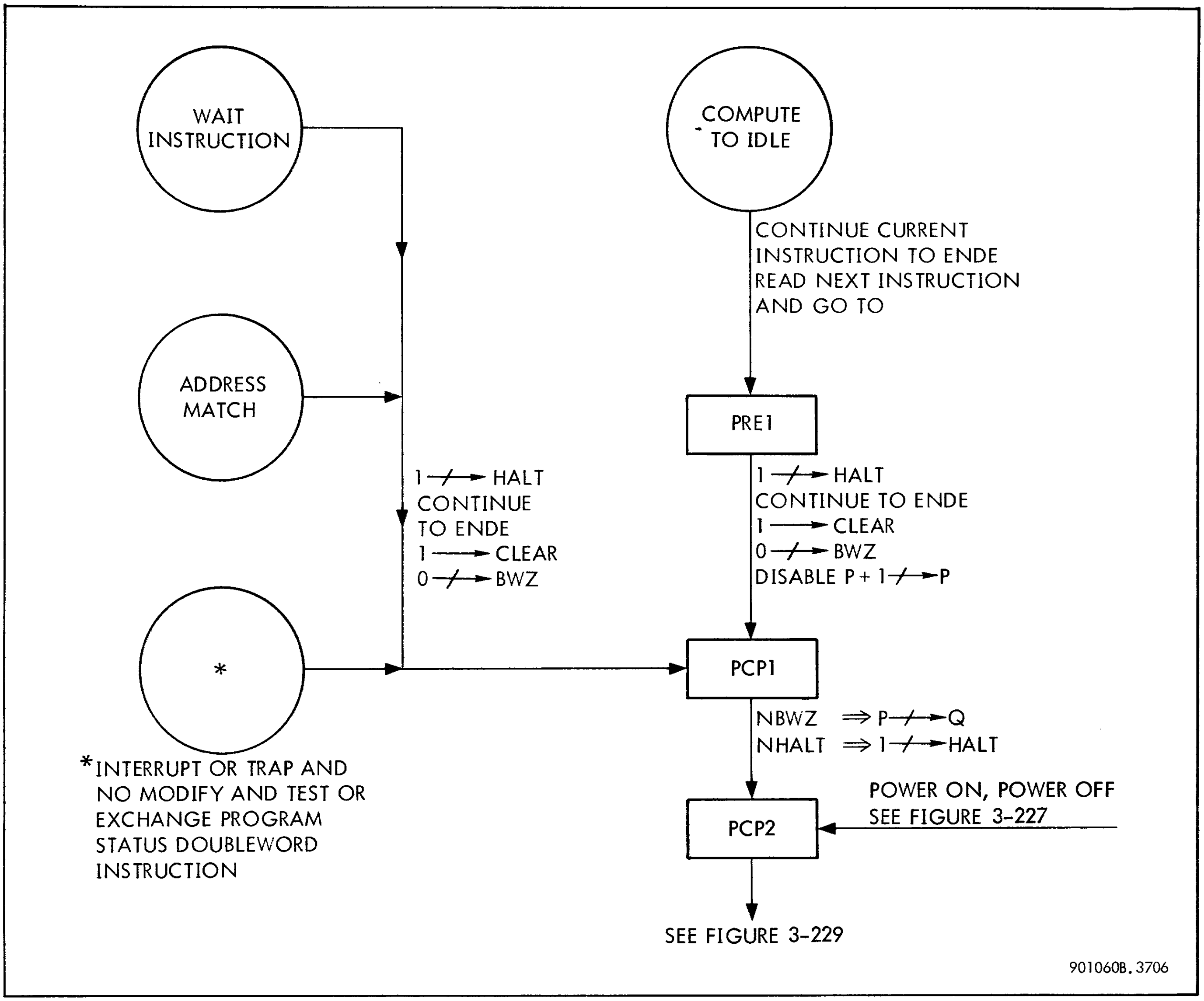


Figure 3-226. Entering PCP Phases

a Modify and Test or Exchange Program Status Doubleword instruction.

S/HALT = PH1 INTRAPF N(FAS7 + FAPSD)

e. Dc power is applied to or is removed from the system

S/HALT = RESET + ...

RESET = START + ...

START = /ST/

where /ST/ is a signal received from the power monitor when the power rises above or drops below a specified level.

3-322 COMPUTE SWITCH TO IDLE. When the PCP phases are entered as a result of setting the COMPUTE switch to IDLE, execution of the current instruction is completed as if nothing had happened. During the ENDE phase of the instruction, the next instruction (addressed by the P-register) is read into the C-register as usual, and the P-register is plus counted to obtain the next instruction address.

The CPU now goes into PRE1, and the instruction in the C-register is executed in the normal manner except that the HALT flip-flop is set in PRE1 as explained above. During the ENDE phase of this instruction, the P-register is not plus counted in the usual manner because this operation is disabled by signal PCPT1DIS:

PCTP1 = ENDE NPCTP1DIS

PCTP1DIS = ENDE (HALT + ...) (...)

(The P-register is plus counted later as explained under STEP or RUN function.)

After the second instruction has been executed, the CPU can no longer go into another execution sequence because the HALT flip-flop is set.

```
S/PRE1 = ENDE (NHALT
         + FUEXU) N(S/INTRAPF)
```

Instead, phase PCP1 is entered

```
S/PCP1 = [ENDE HALT/1 N(S/INTRAPF) NFUEXU]
          NPCP3
```

followed by PCP2

```
S/PCP2 = (PCP1 NCLEAR + ...) NPCP3
```

The LOAD button is pressed when the program is in PCP2 and when one of the six control switches on the lower right-hand side of the panel is operated. The computer advances to PCP3 when the CPU RESET/CLEAR and SYS RESET/CLEAR buttons are pressed simultaneously.

```
S/PCP3 = NPCP3 (PCP2 NHALT/1 NCLEAR
         KAS/1 KAS/2)
         + PCP2 NHALT/1 NCLEAR CLEARMEM
```

The PCP phase sequencing following PCP2 is described under the functions of the various control switches.

3-323 DCSTOP. If the CPU has been halted because of an ADMATCH signal when the ADDR STOP switch is on, the DCSTOP signal is latched by KRUN if the CONTROL MODE switch is in the LOCAL position, or by KSLOW if the CONTROL MODE switch is in the REMOTE position.

```
DCSTOP     = DCSTOP DCSTOP/H
             + KADDRSTOP MR ADMATCH

  DCSTOP/H = NRESET (KRUN + KSLOW)
```

The CPU waits in PCP2 until the COMPUTE switch is moved from RUN to IDLE to STEP and back to RUN. When the free-standing console controls the PCP (PCP CONTROL MODE switch in REMOTE), the address match function operates only when the free-standing console is in SLOW.

3-324 POWER ON OR POWER OFF SEQUENCE. (See figure 3-227.) When the CPU enters the PCP phases as the result of application or removal of power, signal PON or IOFF forces an interrupt. In the case of power on, the /ST/ signal is received immediately from the power monitor, which generates a reset and forces the computer into PCP2.

```
S/PCP2 = (RESET + ...) NPCP3
```

The RESET signal also disarms and disables all interrupts except the power interrupts and forces zeros into PSW1, PSW2, and other control flip-flops, the D-register, and Q-register. The PON interrupt signal is received from the power monitor, and /ST/ drops. This drives signal RESET low, allowing flip-flop HALT to be reset and PCP3 to be entered if the COMPUTE switch is in RUN (KAS/1 and KAS/2). The sequence goes through PCP4 to PCP5, and ENDE goes true, allowing flip-flop INTRAPF to be set and an interrupt sequence to be entered.

```
ENDE          = PCP5 NKIDLE/B

    NKIDLE/B  = KRUN + KSTEP

S/INTRAPF     = NRESET (INT ENDE NINTRAPF
                NINTEN KRUN/B + ...)
```

In the case of power off, the interrupt sequence is entered as a result of signal IOFF from the power monitor, and is allowed from 5 to 20 milliseconds (adjustable in the power monitor) before signal /ST/ is generated. The CPU then goes to PCP2 as described above.

When the PCP phases are entered, flip-flop BWZ is reset by the CLEAR signal to permit the next instruction address in the P-register to be transferred to the Q-register in PCP1.

```
QXP = PCP1 NBWZ (...)
```

Any PCP operation that subsequently sequences through PCP7 causes BWZ to be set to prevent the P- to Q-register transfer in any PCP1 phase except the first.

3-325 RESET FUNCTION (See figure 3-228.) Signal RESET is true if:

a. The CPU RESET/CLEAR switch is pressed in PCP2

b. The SYS RESET/CLEAR switch is pressed in PCP2

c. Signal START is true

In cases a and b above, the COMPUTE switch must first be set to the IDLE state in order to put the computer in PCP2. The RESET signal generated by either of the two reset switches is enabled by NKAS/B, which cannot be true with the COMPUTE switch in RUN; therefore, the reset switches do not affect the current operations of the CPU when in the run state. Signal START, however, causes a reset without placing the computer in the idle state.

The logical sequence of a reset operation when either the CPU RESET/CLEAR or the SYS RESET/CLEAR switch is pressed is shown in table 3-141.

3-326 COMPUTE STEP OR RUN FUNCTION (See figure 3-229.) When the CPU is waiting in PCP2 because the COMPUTE switch is in IDLE, setting the switch to STEP or to RUN causes the CPU to sequence to PCP3, PCP4, and PCP5. In PCP5, signal ENDE is forced true, all normal ENDE functions are performed, and the program branches

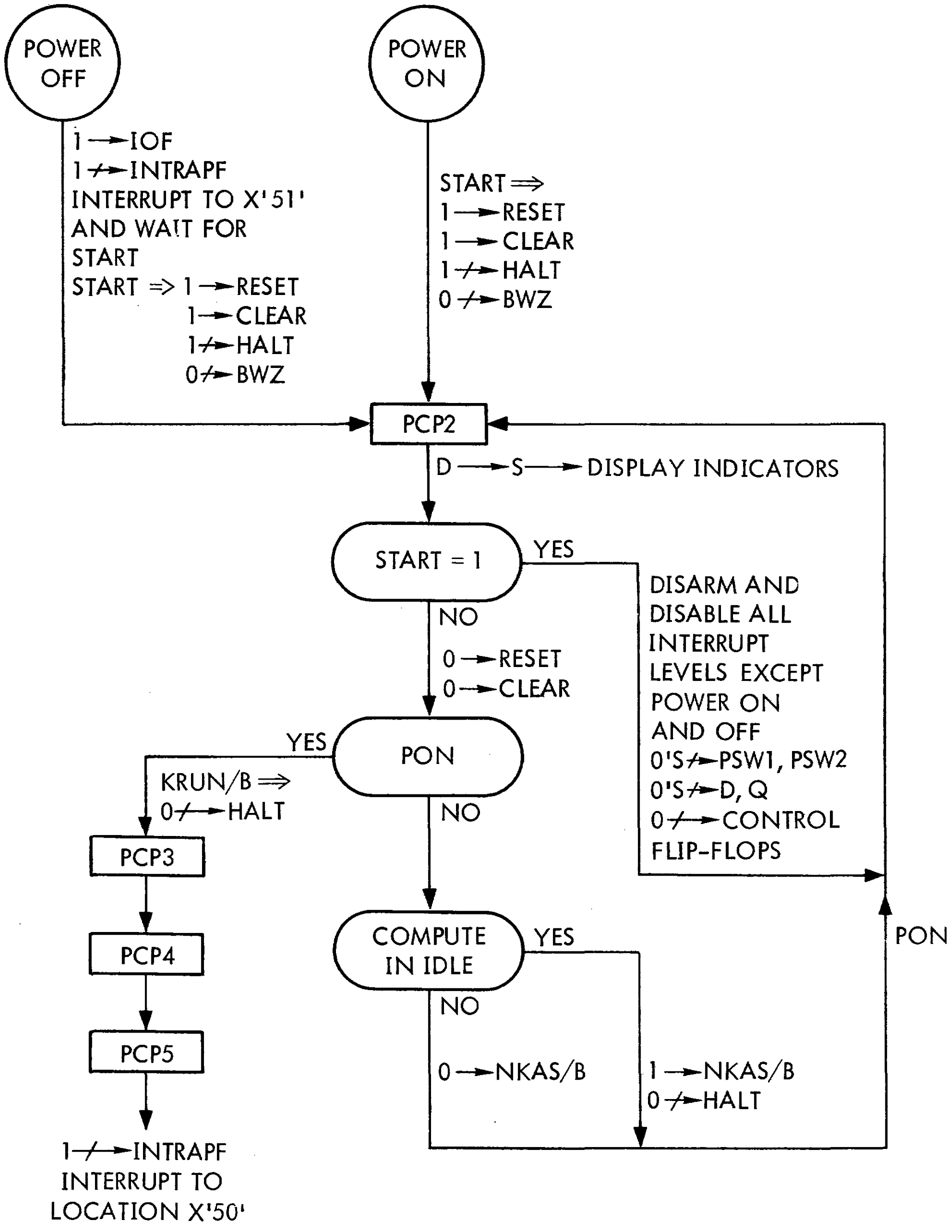


Figure 3-227. Power On or Power Off, Sequence Flow Diagram

Table 3-141. RESET Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
	Switch outputs		
	CPU RESET/CLEAR =>	KCPURESET, KCPURESET/B	
	SYS RESET/CLEAR =>	KSYSR, KSYSR/B	
	COMPUTE in IDLE =>	NKAS/B, NKRUN (necessary for operation of either reset switch	
	Power On or Off =>	START	
PCP2 or START	Idle phase - sustained until control switch operated		
	Generate RESET	RESET = KCPURESET RESET/B + KSYSR RESET/B + START	
		RESET/B = NKAS/B (KCPURESET/B + KSYSR/B)	
	Set HALT flip-flop	S/HALT = RESET + ...	
	Disarm and disable all interrupts except power on and power off	REN = CPUREST1	Interrupt reset enable
		CPUREST1 = RESET	
		R/CNA = CPUREST2 + ...	Interrupt control flip-flops
		CPUREST2 = RESET	
		R/CNB = CPUREST2 + ...	
		E/IP2-IP7 = CPUREST2	Interrupt level arm flip-flop
		E/IP8-IP15 = CPUREST1	
		E/IS2-IS7 = CPUREST2	
		E/IS8-IS15 = CPUREST1	
		R/INTRAPF = CLEAR + ...	Interrupt set - reset flip-flop
		CLEAR = NPREL NCLEARMEM (RESET + ...)	
		R/INTRAP1 = RESET + ...	Interrupt phase flip-flop
	Unlatch interrupt group active signals	CHA0 = CHA0 RCCHA0 NCPURSET + ...	CPURSET drops latch
		CPURSET = RESET	

Mnemonic: RESET

(Continued)

Table 3-141. RESET Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PCP2 or START (Cont.)		CHA1	= CHA1 RCCHA1 NCPURSET + . . .	
		CHA2	= CHA2 RCCHA2 NCPURSET + . . .	
	0 —/→ PSW1	PSW1XS	= N(S/INTRAPF) (RESET + . . .)	Clear program status doubleword
	0 —/→ PSW2	PSW2XS	= RESET + . . .	
	0 —/→ P-register	PX/4	= CLEAR + . . .	
	Reset control flip-flops			
	Index	R/IX	= CLEAR + . . .	
	Alarm	R/ALARM	= RESET + . . .	
	Analyze	R/ANLZ	= CLEAR + . . .	
	B is zero	R/BWZ	= CLEAR + . . .	
	Map disconnect	R/MAPDIS	= CLEAR + . . .	
	Memory request	R/RQ	= CLEAR + . . .	
	Switch flip-flops	R/SW1-SW4	= CLEAR + . . .	
	Trap	R/TRAP	= RESET + . . .	
	Trap condition code flip-flops	R/TRACC4	= RESET + . . .	
	Reset HALT when switch is released	R/HALT	= PCP2 NKAS/B	
	KSYSR => Reset I/O	RIO	= KSYSR NKAS/B + START	
	If reset by switches (RESET/B true) X'02000000 —/→ D	S/D6	= RESET/B	
		DX	= RESET	
	X'25' —/→ Q	S/Q26 S/Q29 S/Q31	= RESET/B	
	If start 0 —/→ Q	QX	= RESET	
	Reset memory fault indicators	/MFR/	= MFR = RESET + . . .	
	If KSYSR or /ST/, reset memory control signals to initial state via interface cables /MRS/ or /ST/	/MRS/	= Memory Reset	
		MEMORYRESET	= KSYSR NKAS/B NCLEARMEM	
		/ST/	= Signal from power monitor	

Mnemonic: RESET

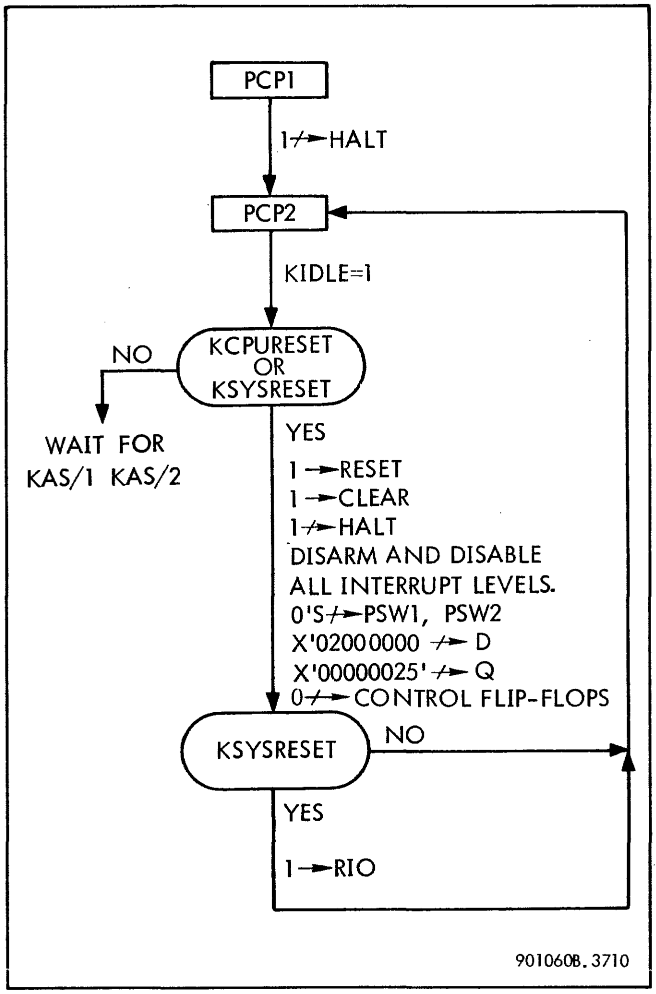


Figure 3-228. CPU RESET/CLEAR and SYS RESET/CLEAR, Sequence Flow Diagram

to PRE1. If the COMPUTE switch is in STEP, the HALT flip-flop is set in PRE1. At the end of instruction execution, the CPU returns to PCP1 and then to PCP2, the idle phase. If the COMPUTE switch is in RUN, the HALT flip-flop is not set, and the program resumes its normal sequence. The logic for the STEP and RUN functions is given in table 3-142.

3-327 INSERT FUNCTION (See figure 3-230). In the idle state, if the INSERT switch is set to PSW1, program status doubleword PSW1 is altered according to the settings of the DATA switches. If the INSERT switch is set to PSW2, program status doubleword PSW2 is altered according to the settings of the DATA switches. The entire program status doubleword is first transferred to the D-register so that in PCP5 the DATA switches can alter either the entire doubleword or only a portion of the word. In the 1 or 0 positions, the switches change the corresponding bit of the program status doubleword. In the center position, the switches cause the previous state of the PSW bit to be entered in the doubleword by transferring the corresponding bit in the D-register to PSW1 or PSW2. The next instruction in the D-register is saved in the A-register and is returned to the D-register in PCP7. A sequence chart of the INSERT function is shown in table 3-143.

3-328 DATA ENTER/CLEAR FUNCTION (See figure 3-231). When the DATA switch is set to ENTER, the states of the 32 DATA switches are transferred to the D-register and are displayed in the 32 DISPLAY indicators. A switch in the up position places a one in the corresponding D-register bit. A switch in the down position enters a zero. A switch in the center position retains the previous D-register bit. The contents of the D-register are accepted as the next instruction to be executed after data has been transferred from the DATA switches into the D-register and the COMPUTE switch is set to either RUN or STEP. When the DATA switch is set to CLEAR, zeros are placed in the D-register.

A sequence chart of the ENTER/CLEAR function is given in table 3-144.

3-329 STORE INSTR ADDR/SELECT ADDR FUNCTION (See figure 3-232). The STORE switch is operative only while the CPU is in the idle state. When the STORE switch is set to INSTR ADDR, the contents of the D-register are stored in the virtual memory address currently in the Q-register and are moved to the P-register for memory addressing. When the STORE switch is set to SELECT ADDR, the contents of the D-register are stored in the virtual address specified by setting the 17 SELECT ADDRESS switches.

A sequence chart of the store function is shown in table 3-145.

3-330 DISPLAY INSTR ADDR/SELECT ADDR FUNCTION (See figure 3-233). If the DISPLAY switch is set to INSTR ADDR in the idle phase, the CPU reads into the D-register the contents of the memory location pointed to by the Q-register, which is moved to the P-register for memory addressing. If the DISPLAY switch is set to SELECT ADDR in the idle phase, the CPU reads into the D-register the contents of the memory location whose value is equal to the contents of the 17 SELECT ADDRESS switches. The data read from memory is displayed in the 32 DISPLAY indicators.

A sequence chart of the DISPLAY INSTR ADDR/SELECT ADDR function is shown in table 3-146.

If the SELECT ADDR switch has been operated, the C-register and the D-register contain the contents of the selected location at the end of the PCP phases. In order to return the current instruction to the C- and D-registers for execution, the DISPLAY switch must be momentarily placed in the INSTR ADDR position.

3-331 INSTR ADDR HOLD/INCREMENT FUNCTION (See figure 3-234). The HOLD position of the INSTR ADDR switch in itself does not take the program into the PCP phases. When the COMPUTE switch is placed in the RUN or STEP position and when the INSTR ADDR switch is in the HOLD

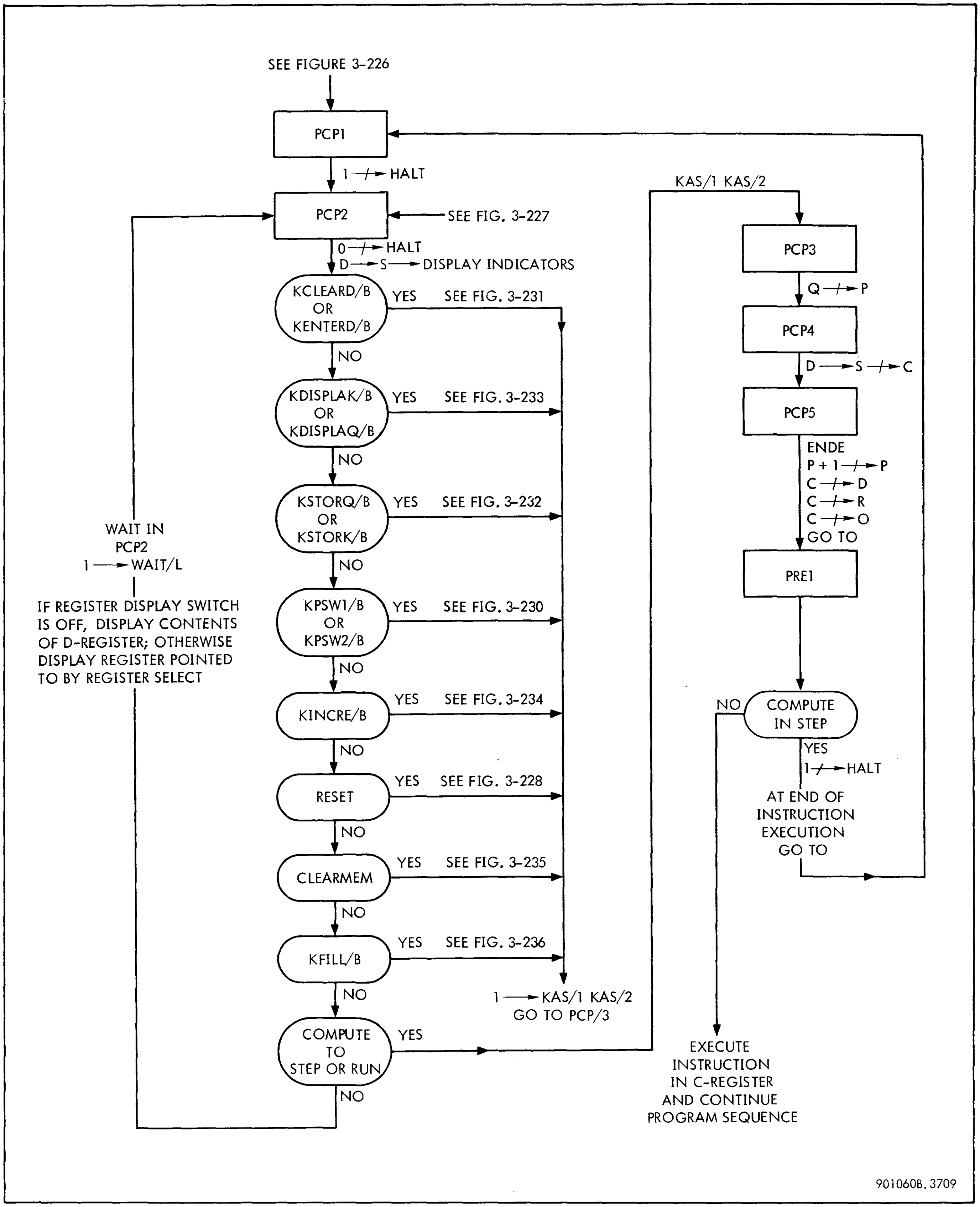


Figure 3-229. PCP Sequence Beyond Wait State

Table 3-142. STEP, RUN Function, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs COMPUTE RUN => COMPUTE STEP => Either position =>	 KRUN/B KSTEP/B KAS/1, KAS/2	
PCP2 T6L	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT NCLEAR KAS/1 KAS/2 NPCP3	Go to PCP phase 3
PCP3 T6L	One clock long (Q15-Q31) -/→ (P15-P31)	R/PCP2 = PCP3	Program address into P-register
		PXQ = PCP3 + ...	
	Set latch CXS	CXS = PCP3 NKIDLE/B + ...	Prepare for S → C in PCP4
	Set flip-flop PCP4	S/PCP4 = PCP3 (NPCP7 NENDE)	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
	Reset flip-flop PCP3	R/PCP3 = ...	
PCP4 T6L	D0-D31 → S0-S31	CXS set in PCP3	Next instruction into C-register
	S0-S31 → C0-C31	SXD = PCP4 NKIDLE/B + ...	
	Set flip-flop PCP5	S/PCP5 = PCP4 NPCP7 NENDE	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	
PCP5 T6L	Force ENDE	ENDE = PCP5 NKIDLE/B + ...	Simulate end of instruction execution to prepare for PRE1 of next instruction
	P + 1 -/→ P	PCTP1 = ENDE NPCTP1DIS	
		PCTP1DIS = ENDE (HALT + ...) (...)	Add one to program address in P-register. Inhibited by HALT in PH10 of last instruction execution

Mnemonic: STEP, RUN

(Continued)

Table 3-142. STEP, RUN Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP5 T6L (Cont.)	C0-C31 $\not\rightarrow$ D0-D31	DXC = ENDE + . . .	Repeat other ENDE functions
	C1-C7 $\not\rightarrow$ O1-O7	OXC = ENDE + . . .	
	C8-C11 $\not\rightarrow$ R28-R31	ORXC = ENDE + . . .	
	Set flip-flop PRE1	S/PRE1 = ENDE NHALT N(S/INTRAPF) + . . .	
	Reset flip-flop PCP5	R/PCP5 = . . .	
PRE1 T6L	KSTEP => Set HALT flip-flop	S/HALT = PRE1 NKRUN/B	If COMPUTE switch in STEP, return program to idle phase after one instruction execution

Menmonic: STEP, RUN

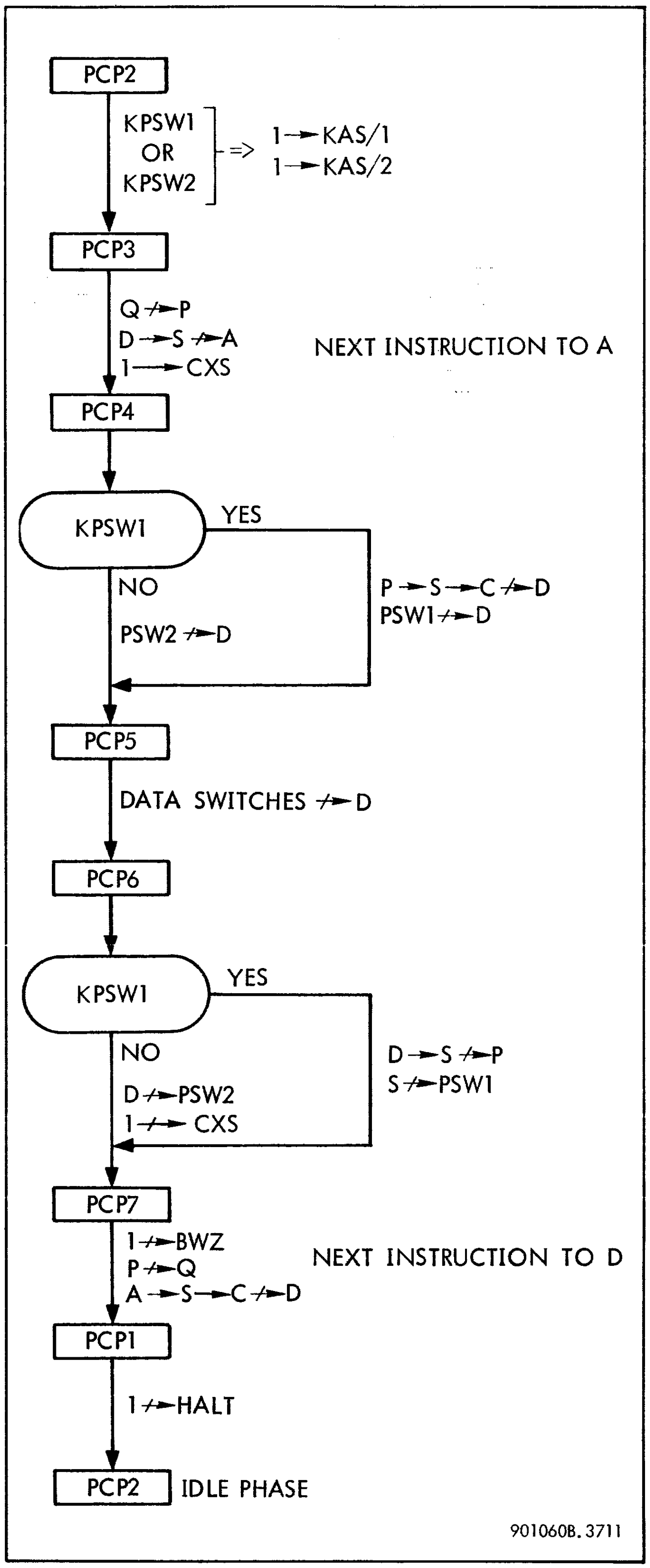


Figure 3-230. INSERT Function, Sequence Flow Diagram

position, signal NKAHOLD is false, PCPT1DIS is true, and upcounting of the P-register is inhibited during the ENDE phase of any instruction that is being executed. The equations are as follows:

PCTP1DIS = ENDE [N(NKAHOLD) + ...]

PCTP1 = NPCTP1DIS (PCTP1 + ENDE)

With the COMPUTE switch in RUN, the CPU repeatedly executes the instruction addressed by the Q-register and does not sequence to the next instruction.

With the COMPUTE switch in IDLE, moving the INSTRADDR switch to INCREMENT causes the current instruction address to be counted up by one and the contents of this updated address to be displayed in the DISPLAY indicators. Thus, the operator can display the contents of sequential memory locations by repeatedly moving the INSTR ADDR switch to INCREMENT.

A sequence chart of the INCREMENT function is given in table 3-147.

3-332 CLEAR MEMORY FUNCTION (See figure 3-235). When the CPU RESET/CLEAR and SYS RESET/CLEAR switches are closed simultaneously with the COMPUTE switch in IDLE, signal CLEARMEM is true, and all core memory locations are cleared to zero.

When CLEARMEM is true, crossover is inhibited; therefore, core memory locations X'0' through X'15' are cleared, and scratch pad memory is not affected. Since one of the two clear switches closes before the other, a memory clear operation is always preceded by a reset operation, which ensures that no memory mapping can occur while memory is being cleared.

The clearing process takes place in PCP6, which is repeated until the CLEARMEM signal is no longer true. The sum bus contents, containing zeros, are transferred to the memory bus lines each PCP6. The contents of the P-register, which are being increased by one with each repetition of PCP6, are placed on the memory address lines so that each memory location in succession is cleared.

A sequence chart of the clear memory function is shown in table 3-148.

3-333 LOAD FUNCTION (See figure 3-236). If the LOAD pushbutton is pressed with the COMPUTE switch in the IDLE position, signal KFILL/B goes true, and the CPU goes into the PCP phases. The CPU forces a bootstrap program into core memory locations X'20' through X'29' by placing the decoded outputs of the P-register onto the sum bus and by writing into memory.

A sequence chart of the LOAD function is shown in table 3-149.

The bootstrap program and its function is shown in table 3-150. Execution of the bootstrap program starts at location 26.

Table 3-143. INSERT Function, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch Outputs		
	INSERT PSW1 => KPSW1/B INSERT PSW2 => KPSW2/B Either position => KAS/1, KAS/2 DATA switches up => KNC0-KNC31, KS0-KS31 DATA switches center => KNC0-KNC31		
PCP2	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT NCLEAR KAS/1 KAS/2 NPCP3	Go to PCP phase 3
PCP3 T6L	One clock long		
	Q15-Q31 —/→ P15-P31	PXQ = PCP3 + ...	Program address into P-register
	D0-D31 —/→ S0-S31	SXD = PCP3 KPSW/B	Save current instruction in A-register
	S0-S31 —/→ A0-A31	AXS = PCP3 KPSW/B	
		KPSW/B = KPSW1/B + KPSW2/B	
	Set latch CXS	(S/CXS) = PCP3 KPSW/B	Prepare for S ——→ C in PCP4
	Set flip-flop PCP4	S/PCP4 = PCP3 NPCP7 NENDE	Go to preparation phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
	Reset flip-flop PCP3	R/PCP3 = ...	
PCP4 T6L	One clock long		
	KPSW1/B =>		
	P15-P31 ——→ S15-S31	SXP = PCP4 KPSW1/B + ...	Transfer original PSW1
	S15-S31 ——→ C15-C31	CXS set in PCP3	or
	C15-C31 —/→ D15-D31	DXC = PCP4 KPSW1/B + ...	
	PSW1 —/→ D0-D11	DXPSW1 = PCP4 KPSW1/B + ...	Floating control, modes, and masks into D-register

Mnemonic: INSERT

(Continued)

Table 3-143. INSERT Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP4 T6L (Cont.)	KPSW2/B =>		
	PSW2 —/→ D2-D7, D23-D27	DXPSW2 = PCP4 PSW2/B	Write key, interrupt inhibits, and register pointer into D-register
	Set flip-flop PCP5	S/PCP5 = PCP4 NPCP7 NENDE	Go to preparation phase 5
	Reset flip-flop PCP4	R/PCP4 = . . .	
PCP5 T6L	One clock long		
	DATA switches —/→ D	DXK = PCP5 KPSW/B S/D0-D31 = KS0-KS31 DXK + D0-D31 KNC0-KNC31 DXK R/D0-D31 = DXK	Transfer information from DATA switches into D-register. Up position transfers ones; down position transfers zeros; center position retains D-register bit.
	Set flip-flop PCP6	S/PCP6 = PCP5 in PCP7 NENDE	Go to preparation phase 6
	Reset flip-flop PCP5	R/PCP5 = . . .	
PCP6 T6L	KPSW1/B =>		
	D0-D31 —/→ S0-S31	SXD = PCP6 NPCP7 KPSW1/B NSDIS + . . .	Place contents of DATA switch on sum bus
	S15-S31 —/→ P15-P31	PXS = PCP6 NPCP7 KPSW1/B + . . .	Transfer address from sum bus to P-register
	S0-S11 —/→ PSW1	PSW1XS = PCP6 NPCP7 KPSW1/B + . . .	Transfer floating and mode control and mask bits from sum bus to appropriate PSW1 flip-flops
	KPSW2/B =>		
	D2-D7, D23-D27 —/→ PSW2	PSW2XD = PCP6 NPCP7 KPSW2/B + . . .	Transfer write key, inhibit bits, and register pointer from D-register to appropriate PSW2 flip-flops prepared for S ——→ C in PCP7
	Set latch CXS	CXS = PCP6 KPSW/B	

Mnemonic: INSERT

(Continued)

Table 3-143. INSERT Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP5 T6L (Cont.)	Set flip-flop PCP7	S/PCP7 = (S/PCP7) NPCP7 NENDE (S/PCP7) = PCP6 NCLEARMEM NKFILL/B + ...	Go to preparation phase 7
PCP7	One clock long		
	A0-A31 → S0-S31	SXA = PCP7 KPSW/B + ...	Return current instruction to C- and D-registers
	S0-S31 → C0-C31	CXS set in PCP6	
	C0-C31 $\not\rightarrow$ D0-D31	DXC = PCP7 KPSW/B + ...	
	P15-P31 $\not\rightarrow$ Q15-Q31	QXP = PCP7 KPSW/B + ...	Place new program address in Q-register
	Set flip-flop PCP1	S/PCP1 = (PCP7 + ...) NPCP3	Go to preparation phase 1
	Reset flip-flop PCP6	R/PCP6 = PCP7 + ...	
PCP1	One clock long		
	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + ...	Halt computer
	Set flip-flop PCP2	S/PCP2 = (PCP1 NCLEAR + ...) NPCP3	Go to idle phase
	Reset flip-flop PCP1	R/PCP1 = ...	

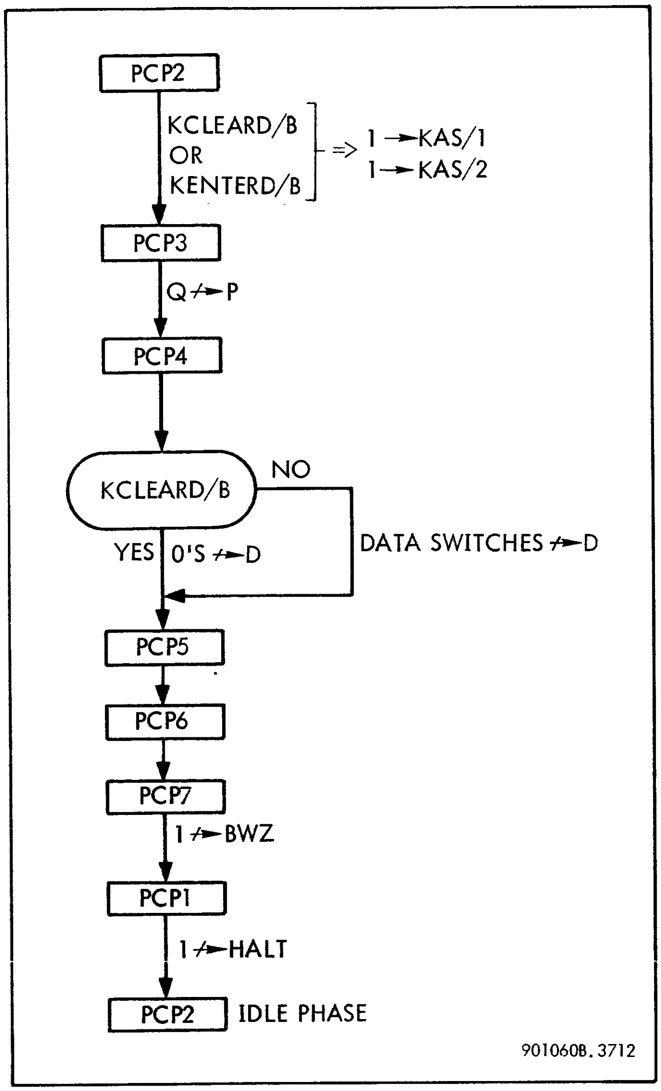


Figure 3-231. DATA ENTER/CLEAR, Sequence Flow Diagram

Address X'25'is placed in the Q-register when the SYS RESET/CLEAR switch is activated (see Reset Function). When the COMPUTE switch is set to RUN, this address is transferred to the P-register in PCP3 (see STEP, RUN Function). During PCP5, the ENDE signal forces P + 1 ⇸ P, allowing location 26 to be addressed for the bootstrap program. The instruction executed after PCP5 is meaningless because nothing is in the C- or D-register.

3-334 Indicators

The PCP indicators, their designators, function, and associated lamp driver signals are listed in tables 2-1 and 2-2. The inputs to most of the lamp drivers are the outputs of CPU flip-flops. However, in some cases, gating is used to energize the lamp drivers, as explained in the paragraphs below. The operation of the indicators in the POWER and the INTERRUPT switches is explained in the descriptions of these switches.

3-335 DISPLAY INDICATORS. The DISPLAY indicator lamp drivers, S0/L through S31/L, contain information that is gated onto the sum bus. The sum bus outputs are selected according to the state of signals KSHI and KSXE, as explained in the paragraphs on the REGISTER DISPLAY and the REGISTER SELECT switches.

3-336 NORMAL MODE INDICATOR. (See figure 3-237.) The NORMAL MODE indicator lights when the memory logic power margins are normal and when the following switches are in the positions indicated:

WATCHDOG TIMER	NORMAL
INTERLEAVE SELECT	NORMAL
PARITY ERROR MODE	CONT
CLOCK MODE	CONT

Under these conditions, with the CONTROL MODE switch in LOCAL, the MARGINS NORMAL input to the AND gate is true and the -8 local voltage cannot be connected to the second AND gate input. Therefore, the inverted output of the AND gate is low, and current flows through the indicator. With the CONTROL MODE switch in LOCK and with the MARGINS NORMAL input true, the lamp is unconditionally lighted because the switch input to the AND gate cannot be false.

3-337 RUN INDICATOR. The RUN indicator lamp driver signal, RUN/L, is true as a result of signal RUNLITE, whose equation is

RUNLITE = KRUN/B NHALT/1

where KRUN/B indicates that the COMPUTE switch is in the RUN position and where NHALT/1 indicates that the HALT flip-flop is reset and that signal DCSTOP is false.

3-338 WAIT INDICATOR. The WAIT indicator receives a lamp driver output WAIT/L, which is true if HALT/1 is true, indicating that the halt flip-flop is set or that signal DCSTOP is true (ADDR STOP switch is set to ON and SELECT ADDRESS is located).

Table 3-144. DATA ENTER/CLEAR, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs DATA ENTER => KENTER D/B DATA CLEAR => KCLEAR D/B Either position => KAS/1, KAS/2 DATA switches up => KNC0-KNC31, KS0-KS31 DATA switches center => KNC0-KNC31		
PCP2	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT KAS/1 KAS/2 NPCP3 + ...	Go to PCP phase 3
PCP3 T6L	One clock long		
	Q15-Q31 —/→ P15-P31	PXQ = PCP3 + ...	Program address into P-register
	Set flip-flop PCP4	S/PCP4 = PCP3 NPCP7 NENDE	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
PCP4 T6L	One clock long		
	KCLEARD/B => 0 —/→ D	DX = PCP4 KCLEARD/B + ...	Clear D-register
	KENTERD/B => Data switches —/→ D	DXK = PCP4 KENTRD/B + ...	Transfer information from DATA switches into D-register
		S/D0-D31 = KS0-KS31 DXK + D0-D31 KNC0-KNC31 DXK + ...	Up position transfers ones, down position transfers zeros; center position retains D-register bit
		R/D0-D31 = DXK	
	Set flip-flop PCP5	S/PCP5 = PCP4 NPCP7 NENDE	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	
PCP5 T6L	One clock long		
	Set flip-flop PCP6	S/PCP6 = PCP5 NPCP7 NENDE	Go to PCP phase 6
	Reset flip-flop PCP5	R/PCP5 = ...	

Mnemonic: DATA ENTER/CLEAR

(Continued)

Table 3-144. DATA ENTER/CLEAR, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP6 T6L	One clock long		
	Set flip-flop PCP7	S/PCP7 = PCP6 NCLEARMEM NKFILL/B + ...	Go to PCP phase 7
PCP7 T6L	One clock long		
	Set flip-flop BWZ	S/BWZ = PCP7 + ...	Prevent P → Q transfer in PCP1
	Set flip-flop PCP1	S/PCP1 = (PCP7 + ...) NPCP3	Go to preparation phase 1
	Reset flip-flop PCP6	R/PCP6 = PCP7 + ...	
	Reset flip-flop PCP7	R/PCP7 = ...	
PCP1 T6L	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + ...	
	Set flip-flop PCP2	S/PCP2 = (PCP1 NCLEAR + ...) NPCP3	Go to PCP phase 2
	Reset flip-flop PCP1	R/PCP1 = ...	
PCP2	Idle phase		
	D0-D31 → S0-S31	SXD = PCP2 NRESET/F NSDIS	Display contents of D-register
	S0-S31 → display indicators	S0L-S31L = S0-S31 NKSHI	

Mnemonic: DATA ENTER/CLEAR

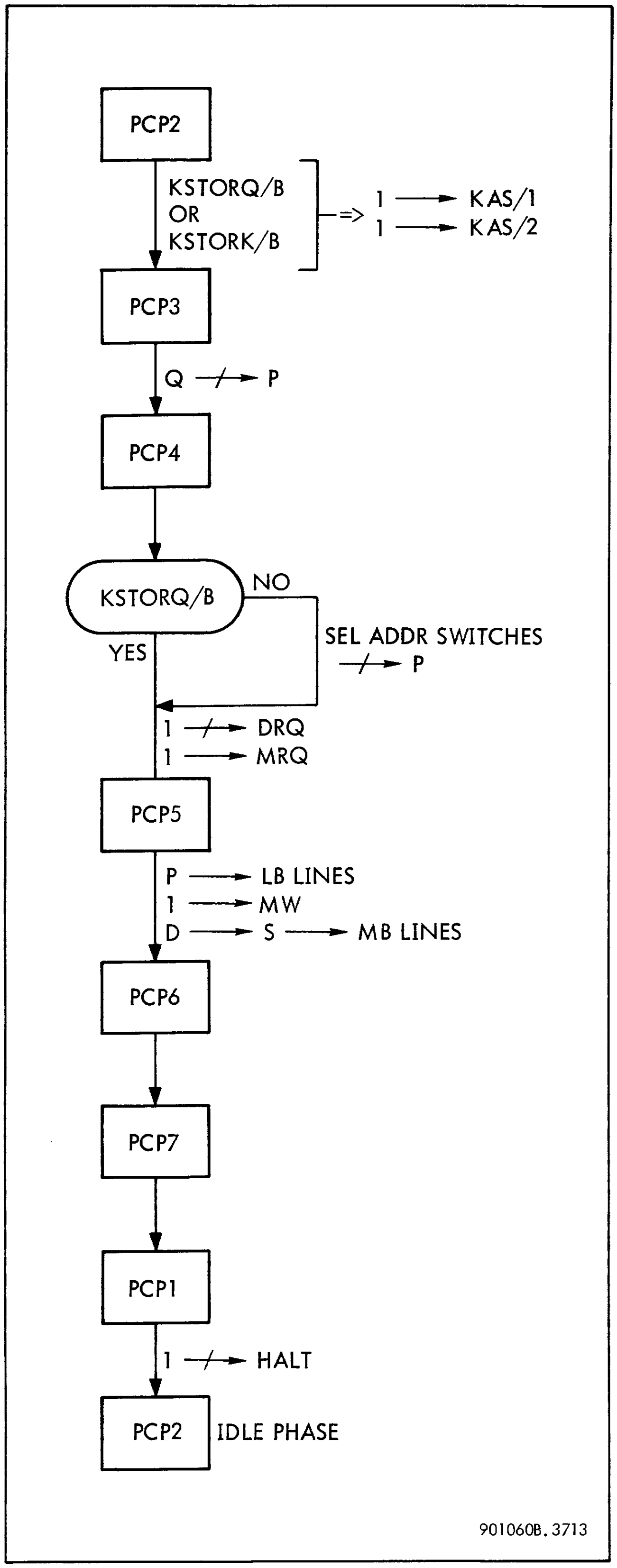


Figure 3-232. STORE INSTR ADDR/SELECT ADDR, Sequence Flow Diagram

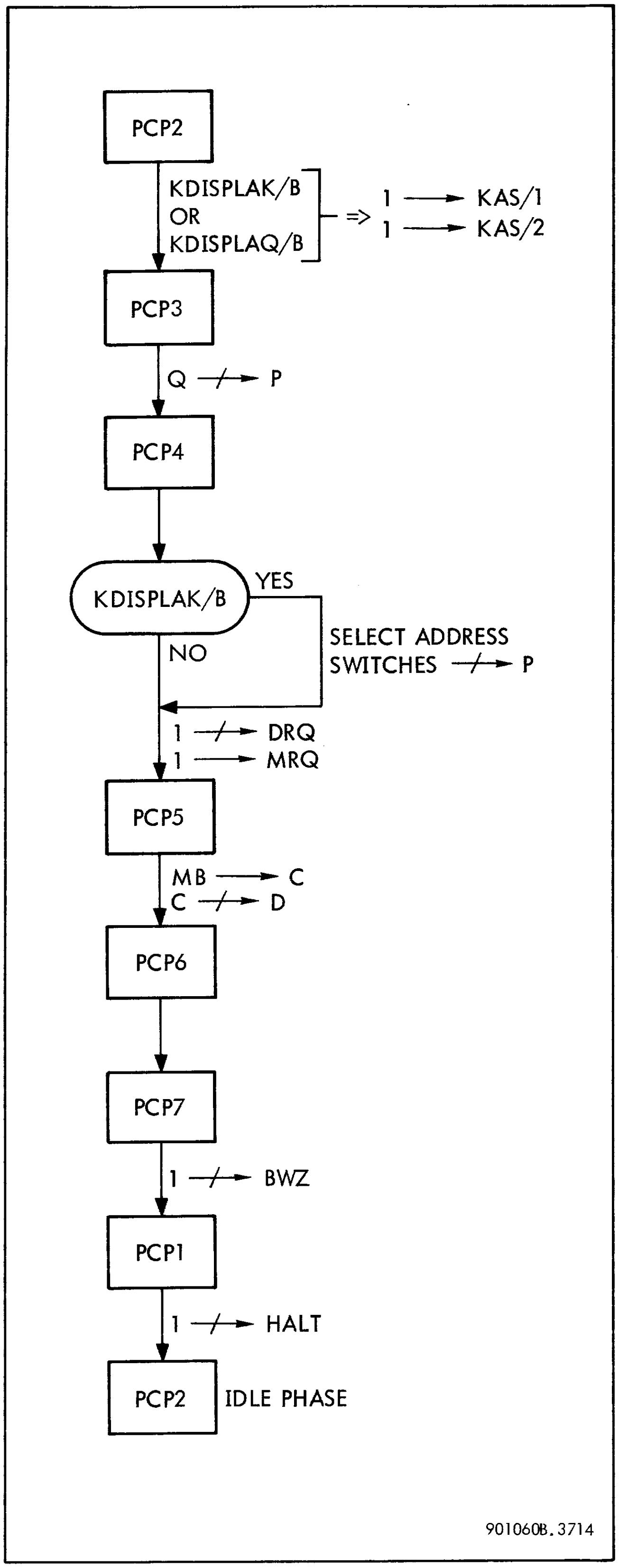


Figure 3-233. DISPLAY INSTR ADDR/SELECT ADDR, Sequence Flow Diagram

Table 3-145. STORE INSTR ADDR/SELECT ADDR, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs STORE SELECT ADDR => KSTOR K/B STORE INSTR ADDR => KSTORQ/B Either position => KAS/1, KAS/2, KSTOR/B		
PCP2 T6L	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT KAS/1, KAS/2 NPCP3	Go to PCP phase 3
PCP3 T6L	One clock long		
	Q15-Q31 —/→ P0-P31	PXQ = PCP3 + ...	Transfer INSTRUCTION ADDRESS indicator contents to P-register
	Set flip-flop PCP4	S/PCP4 = PCP3 NPCP7 NENDE	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
PCP4 T6L	One clock long		
	KSTORK/B =>		
	KSP15-KSP31 —/→ P15-P31	PXK = PCP4 KSTORK/B + ...	Place SELECT ADDRESS switch outputs in P-register
		S/P0-P31 = PXK KSP15 KSP31	
	Set flip-flop DRQ	S/DRQ = PCP4 KSTOR/B	Inhibits clock until data release signal received from memory
	Generator memory request	MRQ = PCP (S/DRQ)	Request memory cycle to store D-register data
	Set flip-flop PCP5	S/PCP5 = PCP4 NENDE NPCP7	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	
PCP5 DR	Sustained until data release		
	Inhibit memory protection	PROTD = FAILD NPCP4 NPCP6	

Mnemonic: STORE

(Continued)

Table 3-145. STORE INSTR ADDR/SELECT ADDR, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP5 DR (Cont.)	Enable core memory write	MW = PCP5 KSTOR/B	
	P15-P31 → LB15-LB31	LB15-LB31 = P15-P31 NCMXQ NCMXC	Addressed from P-register which received Q-register contents in PCP3
	D0-D31 → S0-S31	SXD = PCP5 KSTOR/B + ...	Transfer D-register contents to core memory data lines. If KSTORK/B location is contents of switches; if KSTORQ/B location is Q-register contents, either one now in P-register
	S0-S31 → MB0-MB31	MBXS = MW (...)	
	Set flip-flop PCP6	S/PCP6 = PCP5 NENDE NPCP7	Go to PCP phase 6
	Reset flip-flop PCP5	R/PCP5 = ...	
PCP6 T6L	One clock long		
	Set flip-flop PCP7	S/PCP7 = PCP6 NCLEARMEM NKFILL/B	Go to PCP phase 7
PCP7 T6L	One clock long		
	Set flip-flop PCP1	S/PCP1 = (PCP7 + ...) NPCP3	
	Reset flip-flop PCP7	R/PCP7 = ...	Go to PCP phase 1
	Reset flip-flop PCP6	R/PCP6 = PCP7 + ...	
PCP1 T6L	One clock long		
	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + ...	
	Set flip-flop PCP2	S/PCP2 = (PCP1 NCLEAR + ...) NPCP3	Go to PCP phase 2
	Reset flip-flop PCP1	R/PCP1 = ...	
PCP2	Idle phase		

Mnemonic: STORE

Table 3-146. DISPLAY INSTR ADDR/SELECT ADDR, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs DISPLAY SELECT ADDR ⇒ KDISPLAK/B DISPLAY INSTR ADDR ⇒ KDISPLAQ/B Either position ⇒ KAS/1, KAS/2 KDISPLA/B		
PCP2	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT NCLEAR KAS/1 KAS/2 NPCP3	Go to PCP phase 3
PCP3 T6L	One clock long		
	Q15-Q31 -/→ P13-P31	PXQ = PCP3 + ...	Program address into P-register
	Set flip-flop PCP4	S/PCP4 = PCP3 NENDE NPCP7	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
	Reset flip-flop PCP3	R/PCP3 = ...	
PCP4 T6L	One clock long		
	KDISPLAK/B ⇒		
	KSP15-KSP31 -/→ P15-P31	PXK = PCP4 KDISPLAK/B + ...	Contents of SELECT ADDRESS switches → P-register
		S/P15-P31 = KSP15-KSP31 PXK + ...	
	Set flip-flop DRQ	S/DRQ = PCP4 KDISPLA/B	Inhibits clock until data release signal received from memory
	Generate memory request	MRQ = PCP (S/DRQ)	Request memory cycle for data to display
	Set flip-flop PCP5	S/PCP5 = PCP4 NENDE NPCP7	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	

Mnemonic: DISPLAY

(Continued)

Table 3-146. DISPLAY INSTR ADDR/SELECT ADDR, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP5 T6L DR	Sustained until data release		
	Inhibit memory protection	PROTD = FAILD NPCP5 NPCP6	
	P15-P31 ⟶ LB15-LB31	LB15-LB31 = P15-P31 NCMXQ NLMXC	Addressed from P-register, which received Q-register contents in PCP3
	MB0-MB31 ⟶ C0-C31	CXMB = DGC	
	C0-C31 —/→ D0-D31	DXC = PCP5 KDISPLA/B	Display data gated into C- and D-registers when data gate signal received from memory. If KDISPLAK/B, location is contents of SELECT ADDRESS switches; if KDISPLAQ/B, location is contents of Q-register, either one now in P-register
	Set flip-flop PCP6	S/PCP6 = PCP5 NENDE NPCP7	Go to PCP phase 6
	Reset flip-flop PCP5	R/PCP5 = . . .	
PCP6 T6L	One clock long		
	Set flip-flop PCP7	S/PCP7 = PCP6 NCLEARMEM NKFILL/B	Go to PCP phase 7
PCP7 T6L	One clock long		
	Set flip-flop PCP1	S/PCP1 = (PCP7 + . . .) NPCP3	Go to PCP phase 1
	Reset flip-flop PCP6	R/PCP6 = PCP7 + . . .	
	Reset flip-flop PCP7	R/PCP7 = . . .	
PCP1 T6L	One clock long		
	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + . . .	
	Set flip-flop PCP2	S/PCP2 = (PCP1 NCLEAR + . . .) NPCP3	Go to PCP phase 2
	Reset flip-flop PCP1	R/PCP1 = . . .	
PCP2	Idle phase		
	D0-D31 ⟶ display indicators	SXD = PCP2 NRESET NSDIS	Display D-register contents

Mnemonic: DISPLAY

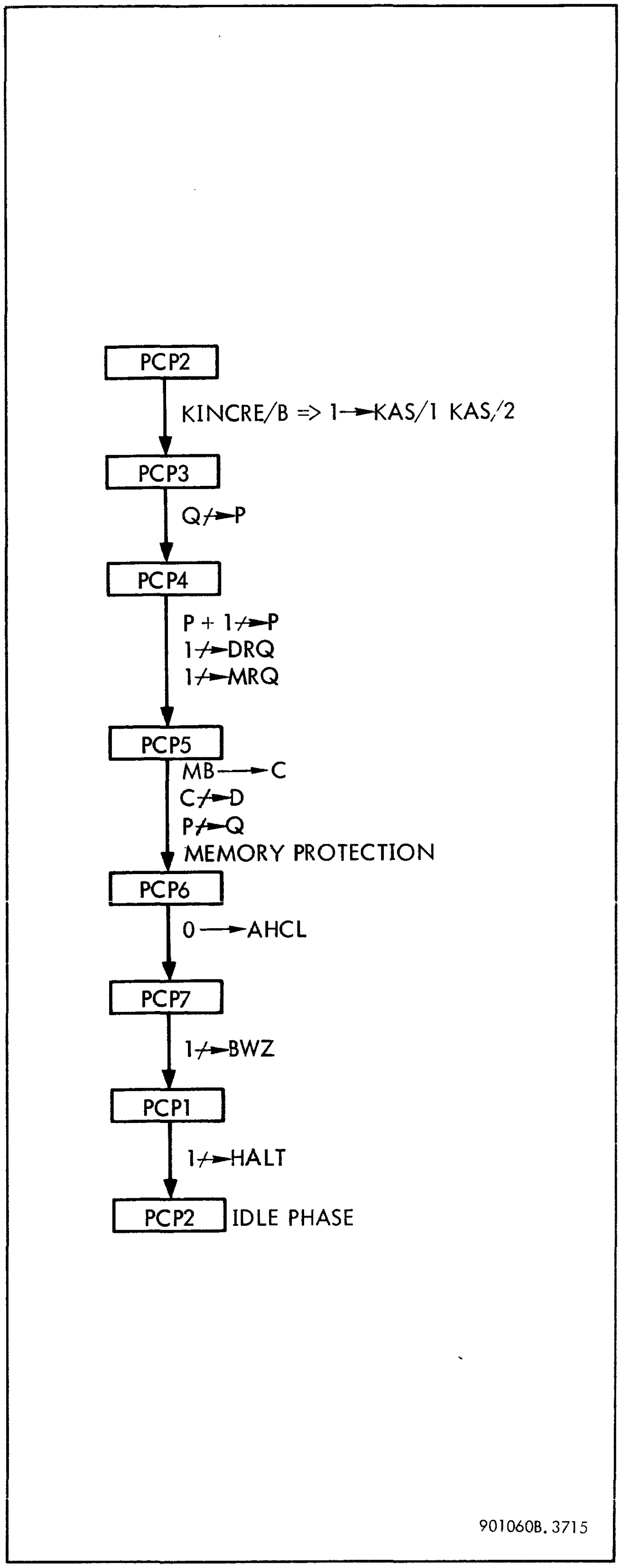


Figure 3-234. INSTR ADDR INCREMENT, Sequence Flow Diagram

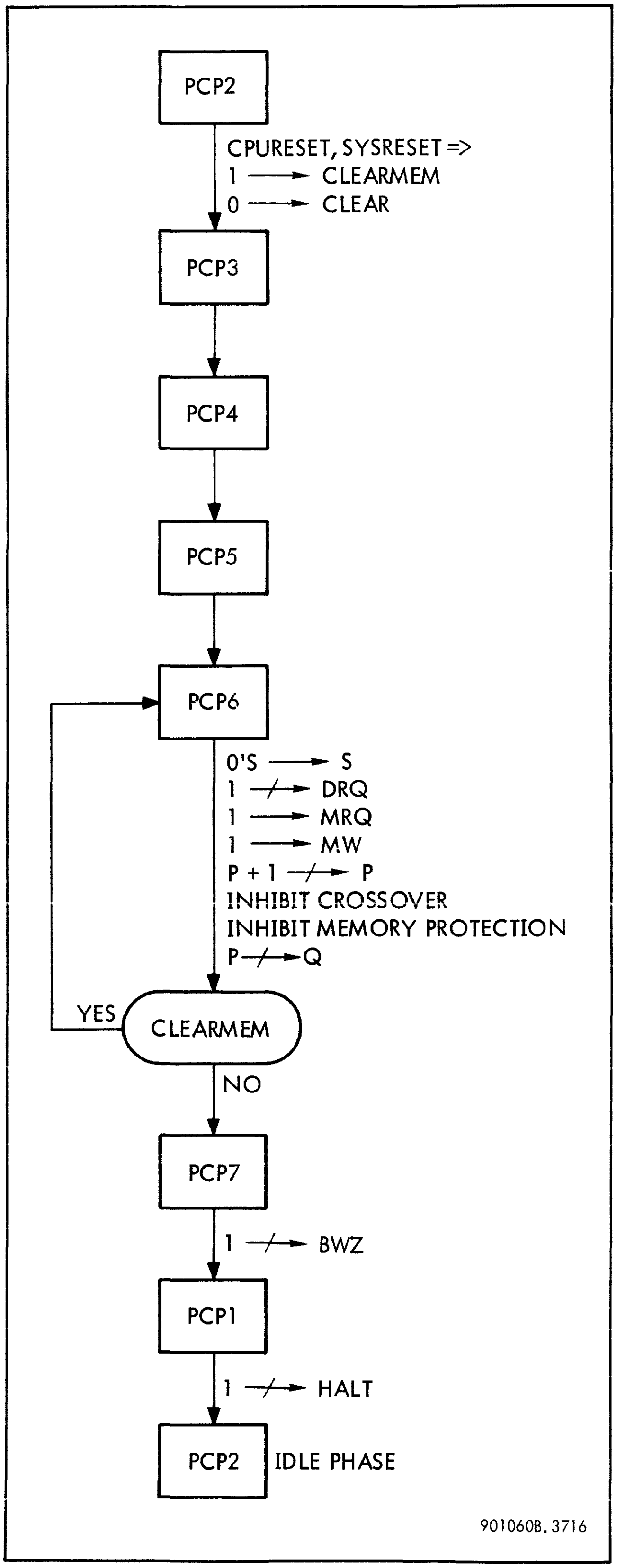


Figure 3-235. CLEARMEM Function, Sequence Flow Diagram

Table 3-147. INSTR ADDR INCREMENT, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs		
	INCREMENT => KINCRE/B, KAS/1, KAS/2		
	Center or INCREMENT => NKAHOLD		
PCP2 T6L	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT KAS/1 KAS/2 NPCP3	Go to PCP phase 3
PCP3 T6L	One clock long		
	Q15-Q31 —/→ P15-P31	PXQ = PCP3 + ...	Program address into P-register
	Set flip-flop PCP4	S/PCP4 = PCP3 NENDE NPCP7	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
	Reset flip-flop PCP3	R/PCP3 = ...	
PCP4 T6L	One clock long		
	P + 1 —/→ P	PCTP1 = PCP4 KINCRE/B + ...	Plus count the P-register by one
	Set flip-flop DRQ	S/DRQ = PCP4 KINCRE/B + ...	Inhibits clock until data release received from memory
	Generate memory request	MRQ = PCP (S/DRQ) + ...	Request contents of P-register address from memory
		PCP = PCP4 + ...	
	Set flip-flop PCP5	S/PCP5 = PCP4 NENDE NPCP7	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	
PCP5 T6L	One clock long		
	P15-P31 —/→ Q15-Q31	QXP = PC55 KINCRE/B	Return upcounted address to Q-register

Mnemonic: INSTR ADDR

(Continued)

Table 3-147. INSTR ADDR INCREMENT, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP5 T6L (Cont.)	MB0-MB31 → C0-C31	CXMB = DGC	Data gated into C-register from memory bus when data gate signal received from memory
	C0-C31 $\nrightarrow$ D0-D31	DXC = PCP5 KINCRE/B	Clock data into D-register
	Inhibit memory protection	PROTD = FAILD NPCP5	
	Set flip-flop PCP6	S/PCP6 = PCP5 NENDE NPCP7	Go to PCP phase 6
	Reset flip-flop PCP5	R/PCP5 = . . .	
PCP6 T6L	One clock long		

	Reset latch AHCL	AHCL = N(NAHCL)
		NAHCL = (PCP6 + . . .) (. . .)
			Memory address clock

	Set flip-flop PCP7	S/PCP7 = PCP6 NCLEARMEM NKFILL/B	Go to PCP phase 7
PCP7 T6L	One clock long		
	Set flip-flop PCP1	S/PCP1 = (PCP7 + . . .) NPCP3	Go to PCP phase 1
	Reset flip-flop PCP6	R/PCP6 = PCP7 + . . .	
	Reset flip-flop PCP7	R/PCP7 = . . .	
PCP1 T6L	One clock long		
	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + . . .	
	Set flip-flop PCP2	S/PCP2 = NPCP3 (PCP1 NCLEAR + . . .)	Go to PCP phase 2
	Reset flip-flop PCP1	R/PCP1 = . . .	
PCP2	Idle phase		
	Display D-register	SXD = PCP2 NRESET NSDIS	

Mnemonic: INSTR ADDR

Table 3-148. CLEARMEM Function, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs CPU RESET/CLEAR => KCPURESET SYS RESET/CLEAR => KSYSR/B COMPUTE in IDLE => NKAS/B		
PCP2 T6L	Idle phase		
	Generate RESET when first switch activated	RESET = NKAS/B (KCPURESET/B + KSYSR/B) + ...	See RESET function
	Set flip-flop PCP3 when both switches activated	S/PCP3 = PCP2 NHALT NCLEAR CLEARMEM NPCP3	
		CLEARMEM = KCPURESET KSYSR NKAS/B	
PCP3 T6L	One clock long		
	Set flip-flop PCP4	S/PCP4 = PCP3 NENDE NPCP7	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
	Reset flip-flop PCP3	R/PCP3 = ...	
PCP4 T6L	One clock long		
	Set flip-flop PCP5	S/PCP5 = PCP4 NENDE NPCP7	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	
PCP5 T6L	One clock long		
	Set flip-flop PCP6	S/PCP6 = PCP5 NENDE NPCP7	Go to PCP phase 6
	Reset flip-flop PCP5	R/PCP5 = ...	
PCP6 T6L	Repeated until NCLEARMEM		
	Set flip-flop DRQ	S/DRQ = PCP6 CLEARMEM + ...	Inhibits clock until data release received from memory

	Generate memory request	MRQ = PCP (S/DRQ) + ...
		PCP = PCP6 + ...
			Prepare to write into core memory

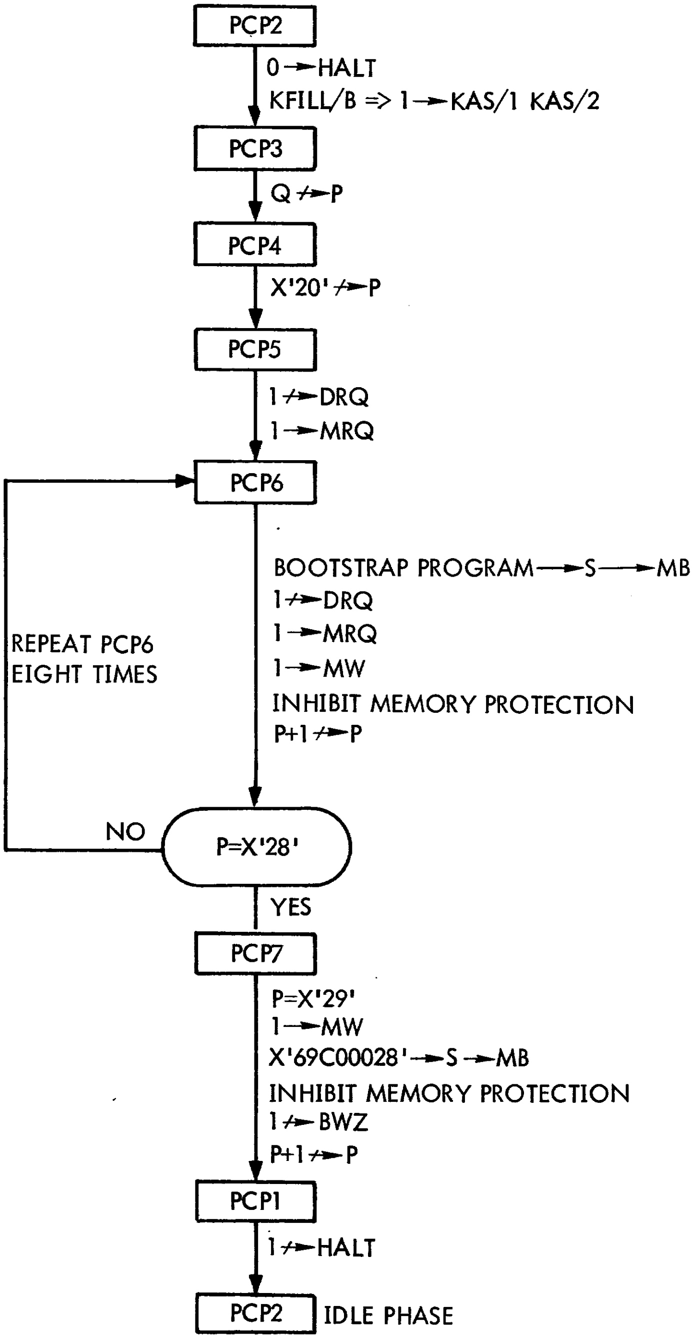
Mnemonic: CLEARMEM

(Continued)

Table 3-148. CLEARMEM Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP6 T6L (Cont.)	Generate memory write	MW = PCP6 CLEARMEM + . . .	Write zeros into addressed memory location
	(S0-S31) → (MB0-MB31)	MBXS = MW = PCP6 CLEARMEM + . . .	
	P + 1 ↛ P	PCPT1 = PCP6 CLEARMEM + . . .	Plus count the P-register by one to address next location
	Inhibit crossover	CRO = NCLEARMEM (address 0-15)	Clear locations 0-15
	Inhibit memory protection	PROTD = FAILD NPCP6 NPCP6	
	Repeat PCP6 until switches released	S/PCP7 = PCP6 NCLEARMEM NKFILL/B + . . .	
	P15-P31 ↛ Q15-Q31	QXP = CLEARMEM NINTRAP2 + . . .	Place current address in INSTRUCTION ADDRESS indicators
PCP7 T6L	One clock long		
	Set flip-flop PCP1	S/PCP1 = NPCP3 (PCP7 + ...)	Go to PCP phase 1
	Reset flip-flop PCP6	R/PCP6 = PCP7 + ...	
	Reset flip-flop PCP7	R/PCP7 = ...	
PCP1 T6L	One clock long		
	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + ...	
	Reset flip-flop PCP1	R/PCP1 = ...	
	Set flip-flop PCP2	S/PCP2 = NPCP3 (PCP1 NCLEAR + ...)	Go to PCP phase 2
PCP2	Idle phase		

Mnemonic: CLEARMEM



901060B.3717

Figure 3-236. LOAD, Sequence Flow Diagram

Table 3-149. LOAD Function, Phase Sequence

Phase	Function Performed	Signals Involved	Comments
	Switch outputs		
	LOAD => KFILL/B, KAS/1, KAS/2 UNIT ADDRESS => KUA21-KUA31		
PCP2 T6L	Idle phase		
	Set flip-flop PCP3	S/PCP3 = PCP2 NHALT NCLEAR KAS/1 KAS/2 NPCP3	Go to PCP phase 3
PCP3 T6L	One clock long		
	Set flip-flop PCP4	S/PCP4 = PCP3 NENDE NPCP7	Go to PCP phase 4
	Reset flip-flop PCP2	R/PCP2 = PCP3	
	Reset flip-flop PCP3	R/PCP3 = ...	
PCP4 T6L	One clock long		
	X'20' —/→ P	S/P26 = PX20	Bootstrap program starts at location X'20'
		PX20 = PCP4 KFILL/B + ...	
		PX/13 = PX = PX20	
	Set flip-flop PCP5	S/PCP5 = PCP4 NENDE NPCP7	Go to PCP phase 5
	Reset flip-flop PCP4	R/PCP4 = ...	
PCP5 T6L	One clock long		
	Set flip-flop DRQ	S/DRQ = PCP5 KFILL/B + ...	Inhibits clock until data release signal received from memory
	Generate memory request	MRQ = PCP (S/DRQ)	Prepare for memory write
		PCP = PCP5 + ...	
	Set flip-flop PCP6	S/PCP6 = PCP5 NENDE NPCP7	Go to PCP phase 6
	Reset flip-flop PCP5	R/PCP5 = ...	

Mnemonic: LOAD

(Continued)

Table 3-149. LOAD Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP6 T6L	Repeated until switch released		
	Set flip-flop DRQ	S/DRQ = PCP6 NPCP7 KFILL/B + ...	Inhibits clock until data release signal received from memory
	Generate memory request	MRQ = PCP (S/DRQ) + ...	
	S0-S31 → MB0-MB31	MBXS = MW = PCP6 KFILL/B + ...	Write bootstrap program
	P + 1 ⇸ P	PCTP1 = PCP6 KFILL/B + ...	Address successive memory locations
	Force bootstrap program on sum bus	S0-S31 = S0I-S31I + ...	
	P = X'20', X'21' 0 → S → MB		
	P = X'22' => X'020000A8' → S → MB	S6I, S28I = KFILL/B NP29 P30 + ...	Decode P-register to set sum bus bits
		S24I = KFILL/B NP29 P30 NP31 + ...	
		S26I = KFILL/B P30 NP31 + ...	
	P = X'23' => '0E000058' → S → MB	S4I, S5I, S27I = KFILL/B PF23 + ...	
		PF23 = KFILL/B NP29 P30 P31	
		S6I, S28I = KFILL/B NP29 P30 + ...	
		S25I = PF23	
	P = X'24' => X'00000011' → S → MB	S27I, S31I = KFILL/B PF24 + ...	
		PF24 = KFILL/B P29 NP30 NP31	
	P = X'25' => X'00000XXX' → S → MB	S21I-S31I = PF25 KUA21-KUA31 + ...	
		PF25 = KFILL/B P29 NP30 P31	
	P = X'26' => X'32000024' →	S2I, S6I, S29I = PF26 + ...	
		PF26 = KFILL/B P29 P30 NP31	
		S3I = KFILL/B PF26 + ...	
		S26I = KFILL/B P29 P30 + ...	

Mnemonic: LOAD

(Continued)

Table 3-149. LOAD Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved		Comments
PCP6 T6L (Cont.)	P = X'27' => X'CC000025'	S0I, S5I, S29I, S31I	= PF278 + ...	
		PF278, PF2789	= KFILL/B P29 P30 P31 + ...	
		S1I, S4I	= PF2789 + ...	
		S26I	= KFILL/B P29 P30 + ...	
	P = X'28' => X'CD000025'	S0I, S5I, S29I, S31I	= PF278 + ...	
		PF278	= KFILL/B P28 NP31 + ...	
		S1I, S4I	= PF2789 + ...	
		PF2789	= KFILL/B P28	
		S7I, S26I	= KFILL/B P28 + ...	
	Set flip-flop PCP7	S/PCP7	= P28 PCP6 NCLEARMEM	Go to PCP phase 7
PCP7 T6L	One clock long	S1I, S4I	= PF2789 + ...	
		PF2789	= ...	
	P = X'29' => X'69C00028' ——► S ——► MB	S2I, S8I, S9I, S28I	= KFILL/B P28 P31 + ...	Force last bootstrap word onto sum bus
		S7I, S26I	= KFILL/B P28 + ...	
	Generate memory write	MBXS	= PCP6 KFILL/B + ...	
	P + 1 $\not\rightarrow$ P	PCTP1	= PCP6 KFILL/B + ...	
	Inhibit memory protection	PROTD	= FAILD NPCP6 NPCP5 + ...	
	Set flip-flop PCP1	S/PCP1	= NPCP3 (PCP7 + ...)	Go to PCP phase 1
	Reset flip-flop PCP6	R/PCP6	= PCP7	
	Reset flip-flop PCP7	R/PCP7	= ...	

Mnemonic: LOAD

(Continued)

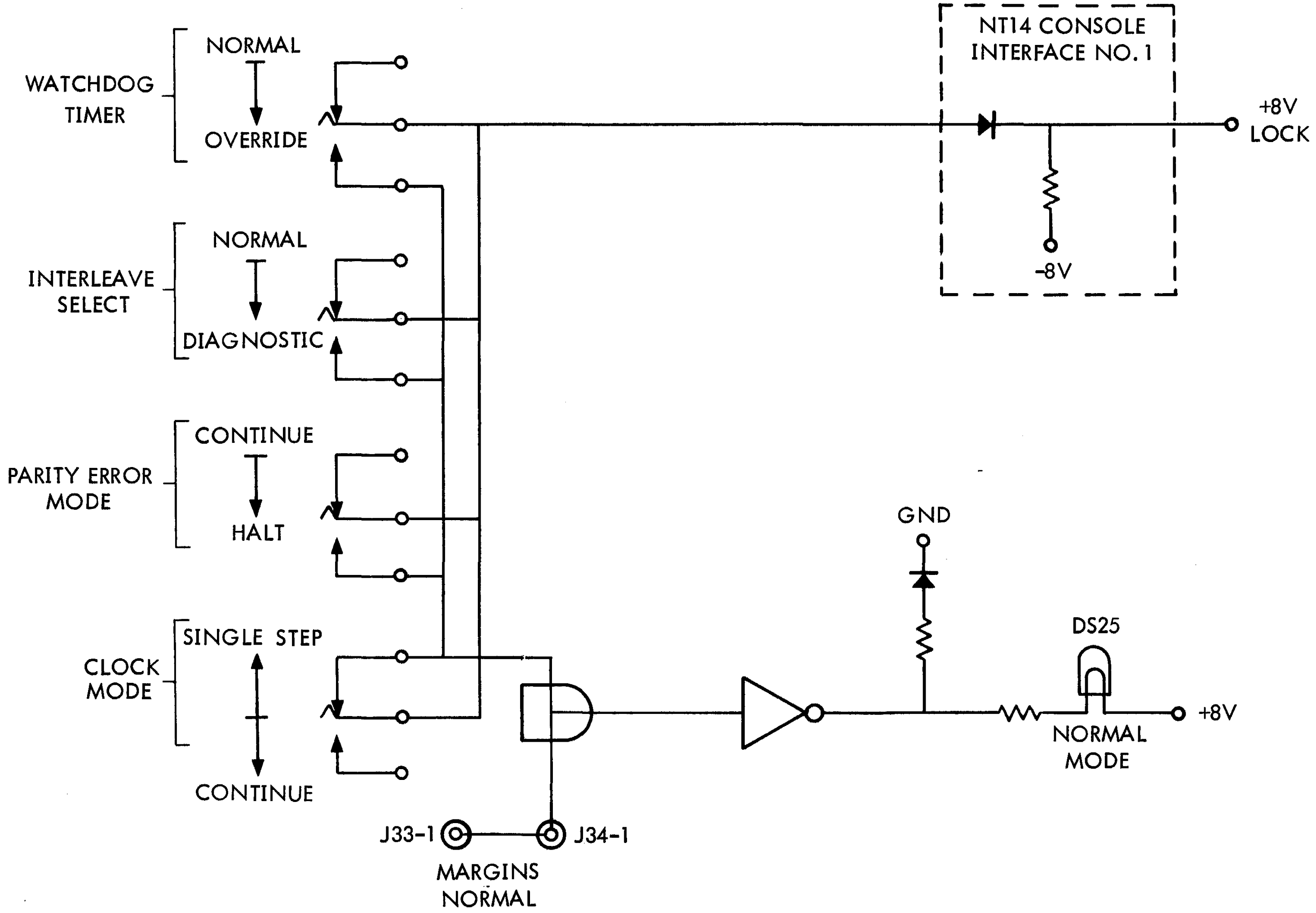
Table 3-149. LOAD Function, Phase Sequence (Cont.)

Phase	Function Performed	Signals Involved	Comments
PCP1 T6L	One clock long		
	Set HALT flip-flop	S/HALT = PCP1 NPCP3 + ...	
	Set flip-flop PCP2	S/PCP2 = NPCP3 (PCP1 NCLEAR + ...)	Go to PCP phase 2
	Reset flip-flop PCP1	R/PCP1 = ...	
PCP2	Idle phase		

Mnemonic: LOAD

Table 3-150. Bootstrap Program

Location	Contents	Significance
20	00000000	Reserved for I/O
21	00000000	Reserved for I/O
22	020000A8	Bits 0 through 31 of I/O command doubleword. Specifies read order (02) and starting byte location X'A8'. Word location is X'A8'/4, or X'2A'
23	0E000058	Bits 32 through 63 of I/O command doubleword. Contains flag bits 0E, which specify halt on transmission error, interrupt on unusual end, and suppress incorrect length. X'58' specifies reading 88 bytes into consecutive memory locations
24	00000011	Command address of I/O command doubleword (shifted one bit position to the left in IOP to specify word location X'22')
25	00000XXX	Address of I/O unit. (XXX represents address taken from UNIT ADDRESS switches)
26	32000024	Load Word instruction. Loads the contents of location X'24' (X'11') into private memory register 0
27	CC000025	Indirectly-addressed Start Input/Output instruction. Takes command doubleword from location specified in private memory register 0 and specifies device pointed to by address in location 25
28	CD000025	Indirectly-addressed Test Input/Output instruction
29	69C00028	Branch on Conditions Set (BCS) instruction. Branches to location X'28' if condition code is not X'C'. If the device address used by the SIO and the TIO instructions is not recognized by any device condition codes CC1 and CC2 are set by both instructions, causing the BCS instruction to branch continuously to X'28' until operator intervention occurs During a successful loading operation, when the TIO is executed the device should be busy because of the previous SIO instruction. As long as the device is busy, condition code CC2 is set when the TIO is executed, causing the BCS to branch continuously to X'28' The next time the TIO is executed, when the device finishes its input operation, CC2 is reset, causing the BCS not to branch, and allowing the CPU to sequence to location X'2A' for the next instruction



901060B.3718

Figure 3-237. NORMAL LOAD Indicator, Simplified Schematic Diagram

3-339 PHASE INDICATORS. The PREPARATION, PCP, and the EXECUTION phase indicators represent the binary encoding of the states of the phase flip-flops, PRE1 through PRE4, PCP1 through PCP7, and PH1 through PH15. A typical equation is as follows

$$\begin{aligned} PH1/L = {} & PH1 + PH3 + PH5 + PH7 + PH9 \\ & + PH11 + PH13 + PH15 \end{aligned}$$

The INT/TRAP phase indicators are lighted by the outputs of flip-flops INTRAP1 and INTRAP2 which define interrupt and trap phases.

3-340 MEMORY FAULT INDICATORS. MEMORY FAULT indicators 1 through 8 are lighted by lamp drivers MF0/L through MF7/L, which are activated by memory fault indicator signals MFL0 through MFL7 from the memory.

3-341 POWER DISTRIBUTION

The first five illustrations that follow give information needed to understand power distribution in Sigma 7. Figure 3-238 explains the cable designation code; figure 3-239 explains the cabinet and frame location codes; and figure 3-240 shows the standard prong shapes of connectors. Figures 3-241 and 3-242 show typical positioning of power supplies and distribution units.

A typical system power distribution is shown in figure 3-243. Figures 3-244 through 3-247 show a typical power configuration, interconnections, and functional diagram. These figures give general information; the installation conditions may make it necessary to use different cable lengths than those indicated. Specific installation information is provided by the XDS Customer Service department.

Table 3-151 lists installation drawings for peripheral equipment and power supplies. Table 3-152 lists the types of power cables used.

3-342 GLOSSARY TABLES

A glossary of signals is given as follows:

a. Table 3-153. Symbols Used in Glossaries and Equations, Miscellaneous Information.

b. Table 3-154. Glossary of CPU Signals.

Note

This glossary includes memory map, memory protect, and floating point signals, as well as certain interrupt signals.

c. Table 3-155. Glossary of Interrupt Signals. The glossary is divided into two parts, internal and external.

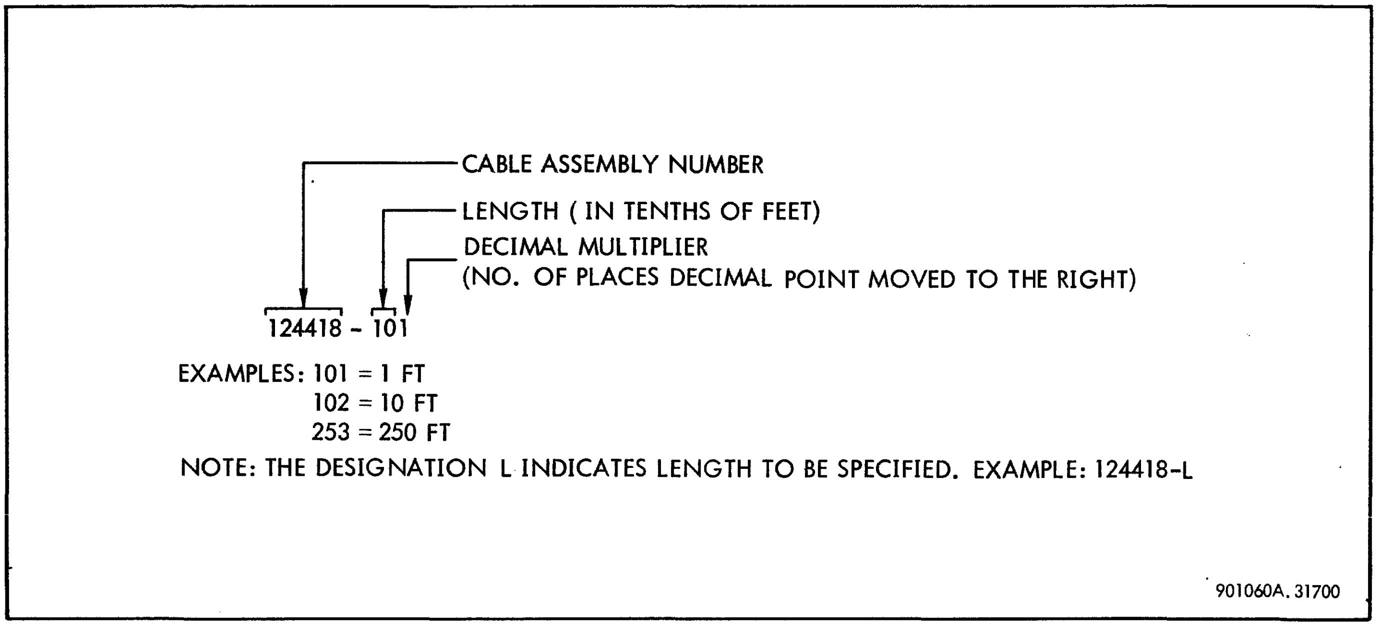
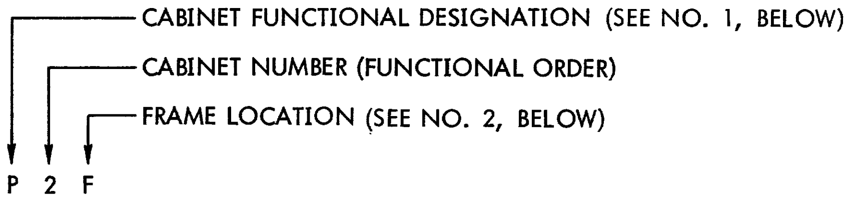

CABLE ASSEMBLY NUMBER
LENGTH (IN TENTHS OF FEET)
DECIMAL MULTIPLIER
(NO. OF PLACES DECIMAL POINT MOVED TO THE RIGHT)
124418 - 101
EXAMPLES: 101 = 1 FT
102 = 10 FT
253 = 250 FT
NOTE: THE DESIGNATION L INDICATES LENGTH TO BE SPECIFIED. EXAMPLE: 124418-L
901060A. 31700

Figure 3-238. Cable Designation Code


CABINET FUNCTIONAL DESIGNATION (SEE NO. 1, BELOW)
CABINET NUMBER (FUNCTIONAL ORDER)
FRAME LOCATION (SEE NO. 2, BELOW)
P 2 F

1. CABINET CODE

CODE	CABINET
P	PROCESSOR
M	MEMORY
R	ACCESSORY TO THE RIGHT OF CPU
L	ACCESSORY TO THE LEFT OF CPU
D	DEVICE CONTROLLER

2. FRAME LOCATION CODE

CODE	FRAME LOCATION
F	FRONT
C	CENTER
R	REAR

EXAMPLES:

M3C MEMORY, THIRD CABINET, CENTER
D5R DEVICE CONTROLLER, FIFTH CABINET, REAR
R2R ACCESSORY TO THE RIGHT OF CPU, SECOND CABINET, REAR

901060A.31701

Figure 3-239. Cabinet and Frame Location Codes

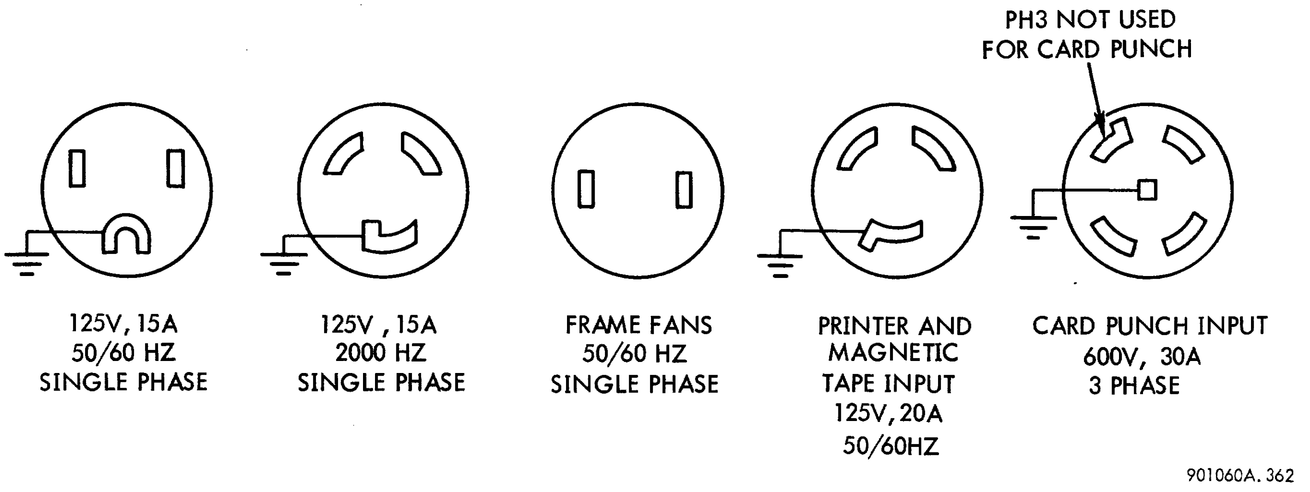


Figure 3-240. Male Connectors, Prong Shapes and Uses

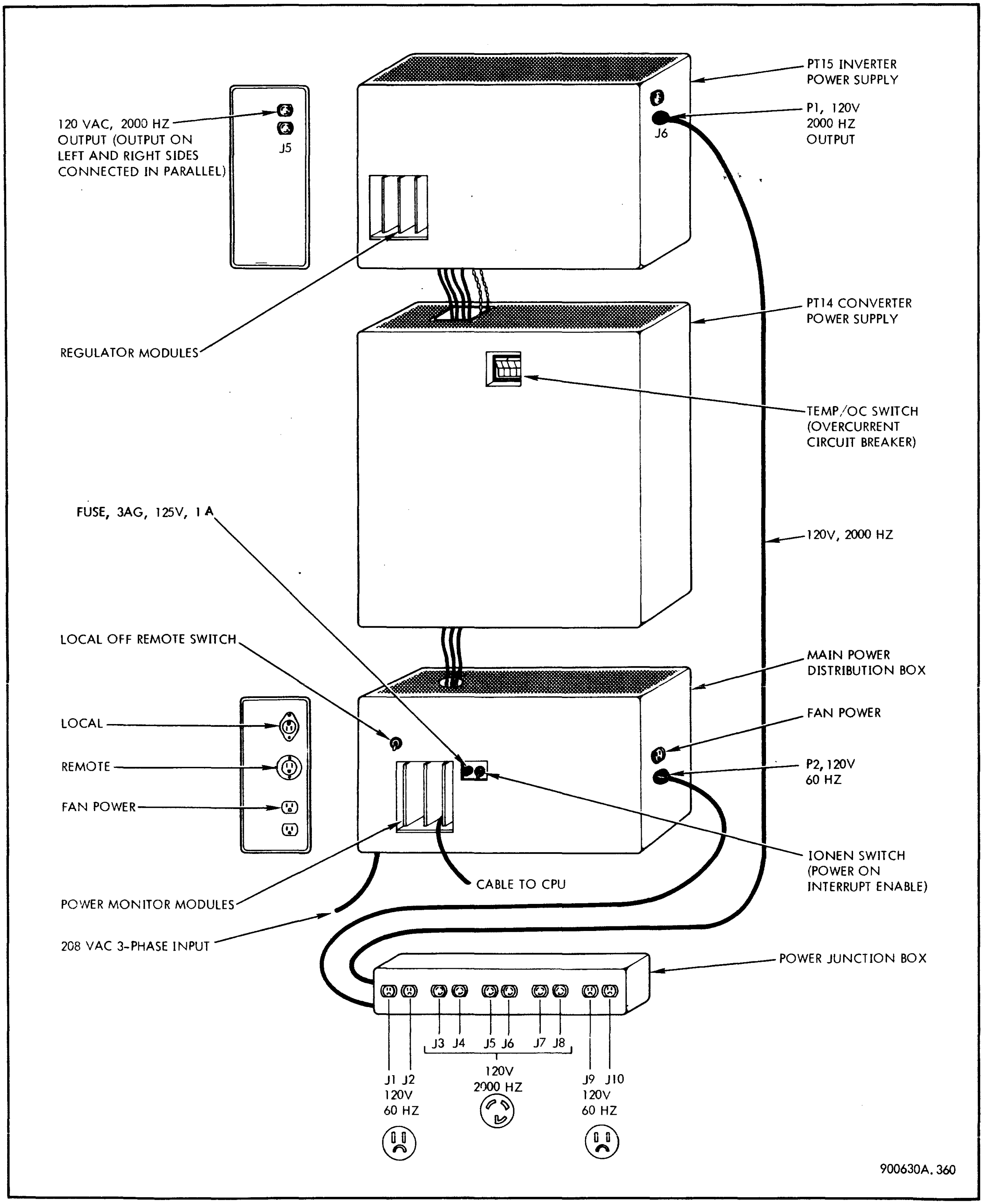
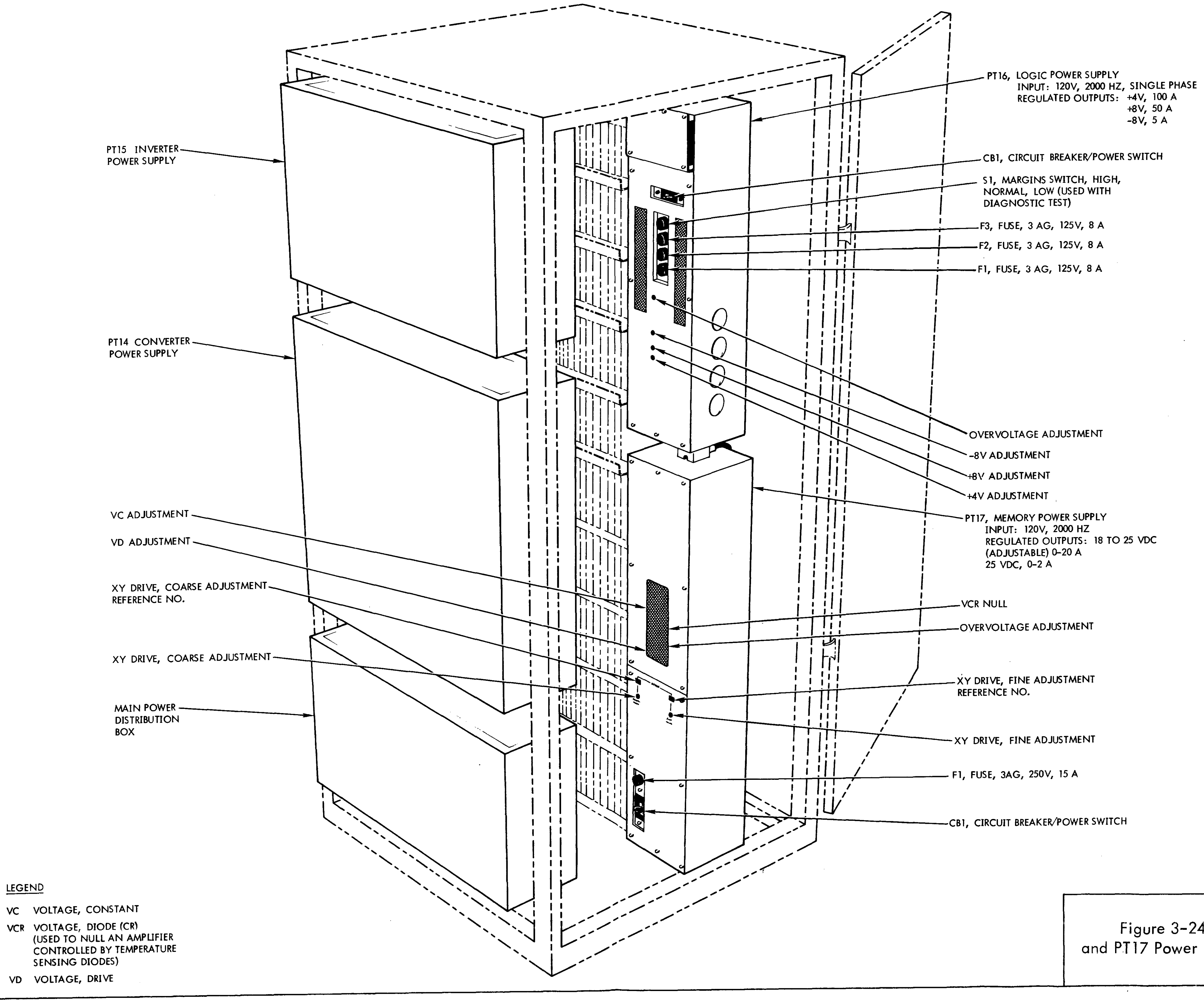


Figure 3-241. PT15 Power Supply, PT14 Power Supply, Main Power Distribution Box, and Power Junction Box (Typical Installation)


PT16, LOGIC POWER SUPPLY
INPUT: 120V, 2000 HZ, SINGLE PHASE
REGULATED OUTPUTS: +4V, 100 A
+8V, 50 A
-8V, 5 A
PT15 INVERTER POWER SUPPLY
CB1, CIRCUIT BREAKER/POWER SWITCH
S1, MARGINS SWITCH, HIGH, NORMAL, LOW (USED WITH DIAGNOSTIC TEST)
F3, FUSE, 3 AG, 125V, 8 A
F2, FUSE, 3 AG, 125V, 8 A
F1, FUSE, 3 AG, 125V, 8 A
PT14 CONVERTER POWER SUPPLY
OVERVOLTAGE ADJUSTMENT
-8V ADJUSTMENT
+8V ADJUSTMENT
+4V ADJUSTMENT
VC ADJUSTMENT
PT17, MEMORY POWER SUPPLY
INPUT: 120V, 2000 HZ
REGULATED OUTPUTS: 18 TO 25 VDC
(ADJUSTABLE) 0-20 A
25 VDC, 0-2 A
VD ADJUSTMENT
XY DRIVE, COARSE ADJUSTMENT REFERENCE NO.
VCR NULL
OVERVOLTAGE ADJUSTMENT
XY DRIVE, COARSE ADJUSTMENT
XY DRIVE, FINE ADJUSTMENT REFERENCE NO.
MAIN POWER DISTRIBUTION BOX
XY DRIVE, FINE ADJUSTMENT
F1, FUSE, 3AG, 250V, 15 A
CB1, CIRCUIT BREAKER/POWER SWITCH

LEGEND

VC VOLTAGE, CONSTANT

VCR VOLTAGE, DIODE (CR) (USED TO NULL AN AMPLIFIER CONTROLLED BY TEMPERATURE SENSING DIODES)

VD VOLTAGE, DRIVE

Figure 3-242. PT16 Power Supply and PT17 Power Supply (Typical Installation)

901060A. 31702

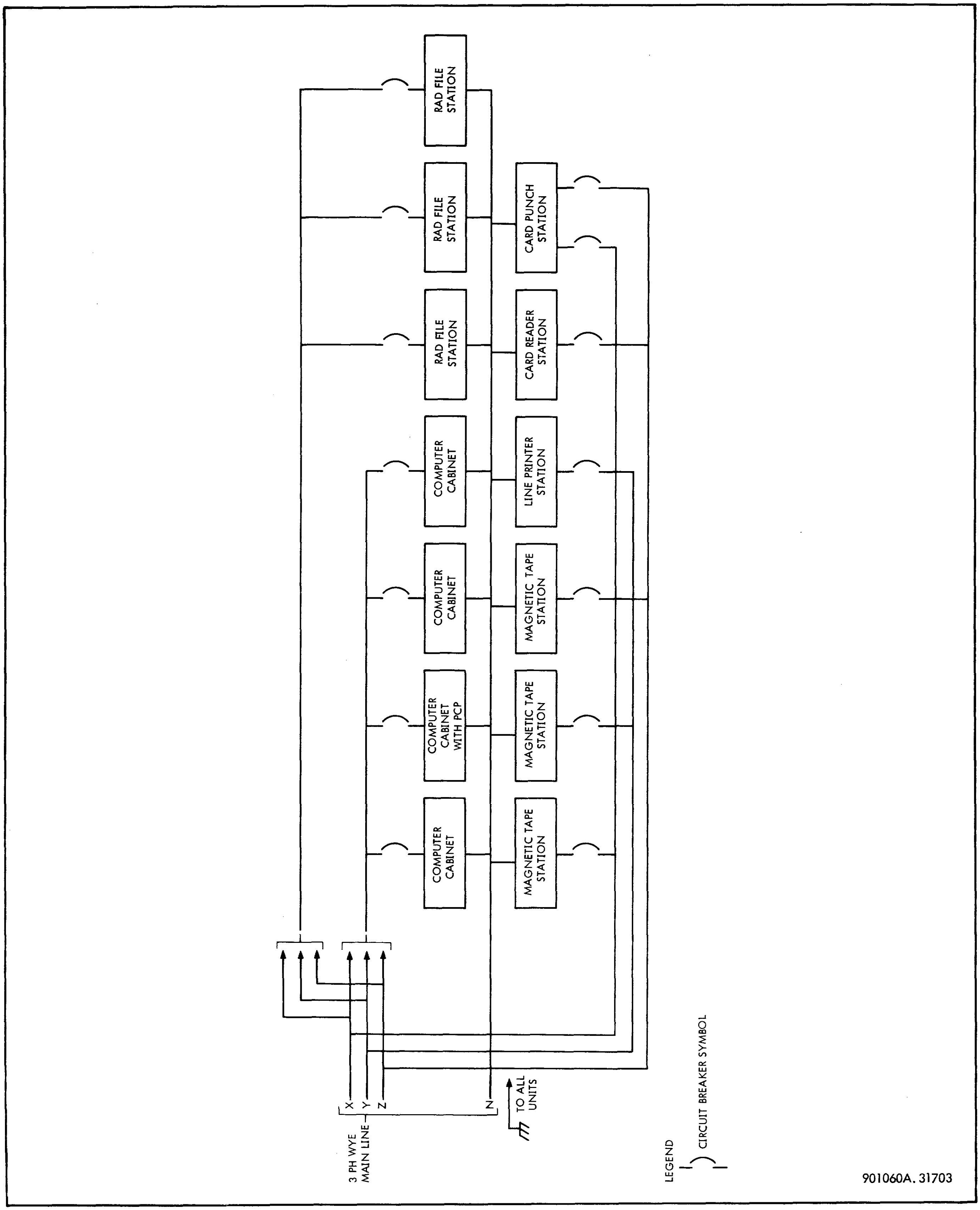
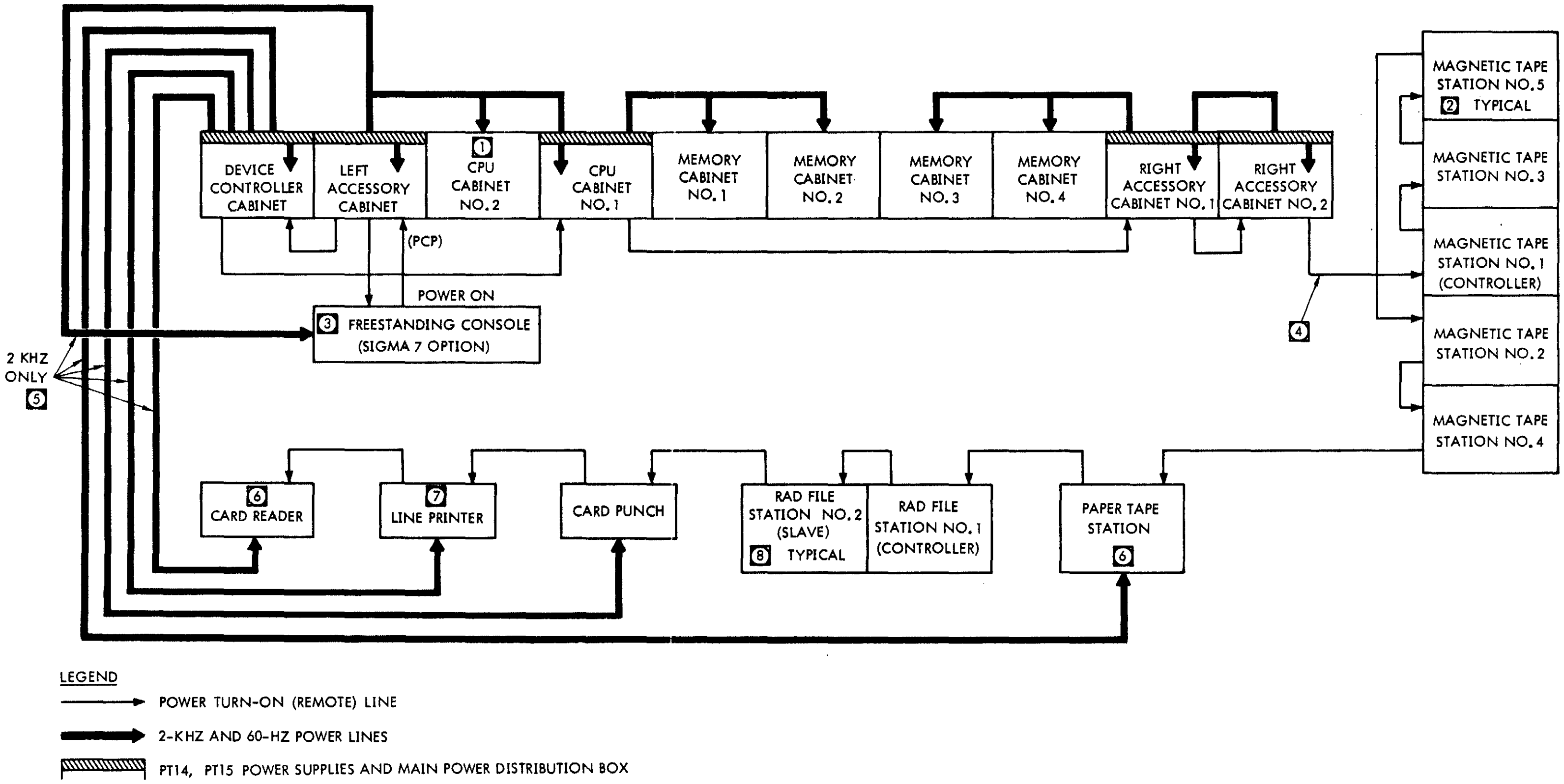


Figure 3-243. System Power Distribution, Typical


DEVICE CONTROLLER CABINET
LEFT ACCESSORY CABINET
1 CPU CABINET NO. 2
CPU CABINET NO. 1
MEMORY CABINET NO. 1
MEMORY CABINET NO. 2
MEMORY CABINET NO. 3
MEMORY CABINET NO. 4
RIGHT ACCESSORY CABINET NO. 1
RIGHT ACCESSORY CABINET NO. 2
MAGNETIC TAPE STATION NO. 5
2 TYPICAL
MAGNETIC TAPE STATION NO. 3
MAGNETIC TAPE STATION NO. 1 (CONTROLLER)
MAGNETIC TAPE STATION NO. 2
MAGNETIC TAPE STATION NO. 4
4
(PCP)
POWER ON
3 FREESTANDING CONSOLE (SIGMA 7 OPTION)
2 KHZ ONLY
5
6 CARD READER
7 LINE PRINTER
CARD PUNCH
RAD FILE STATION NO. 2 (SLAVE)
8 TYPICAL
RAD FILE STATION NO. 1 (CONTROLLER)
PAPER TAPE STATION
6
LEGEND
POWER TURN-ON (REMOTE) LINE
2-KHZ AND 60-HZ POWER LINES
PT14, PT15 POWER SUPPLIES AND MAIN POWER DISTRIBUTION BOX

NOTES:

1 THE PT14 AND PT15 POWER SUPPLIES AND THE MAIN POWER DISTRIBUTION BOX FOR CPU CABINET NO. 2 ARE USUALLY OPTIONAL. HOWEVER, IF THE SYSTEM HAS FOUR MEMORY CABINETS, THEY ARE REQUIRED. EXCEPTION: THE MEMORY CABINETS MAY OBTAIN POWER FROM RIGHT ACCESSORY CABINET NO. 1 WITH PT14 AND PT15 POWER SUPPLIES WHEN THEY ARE AVAILABLE

2 EIGHT MAGNETIC TAPE UNITS ARE A MAXIMUM CONFIGURATION FOR A MAGNETIC TAPE SUBSYSTEM. SEE INSTALLATION DRAWING NO. 137114, MAGNETIC TAPE SUBSYSTEM

3 POWER EXTENSION CORDS TO THE FREE STANDING CONSOLE CAN CONNECT TO ANY JUNCTION BOX IF CABLE LENGTH, POWER, OR CONNECTOR LIMITATIONS OCCUR. (SEE ITEM 5 AND FIGURE 3-246, ITEM 8)

4 THE REMOTE LINE TO PERIPHERAL STATIONS CONNECT IN SEQUENCE IN ORDER OF STARTING SURGE (WITH THE HIGHEST ONE FIRST), UNLESS THE SYSTEM CONTAINS A RAD SUBSYSTEM (SEE ITEM 6). THE ORDER OF PERIPHERAL STATIONS, BY DEGREE OF STARTING SURGE (FROM THE HIGHEST TO THE LOWEST), IS MAGNETIC TAPE CONTROLLER AND TRANSPORT, MAGNETIC TAPE TRANSPORT, RAD SUBSYSTEM, CARD PUNCH, LINE PRINTER, PAPER TAPE EQUIPMENT, AND CARD READER

5 THE TOTAL LENGTH OF ANY 2-KHZ LINE (FROM THE PT15 POWER SUPPLY TO THE DC POWER) MUST NOT EXCEED 60 FEET. THE PREFERRED LENGTH OF 2-KHZ EXTENSION CORD CABLE (NO. 124418) IS LESS THAN 50 FEET. IF NECESSARY, 2-KHZ POWER FOR PERIPHERAL STATIONS MAY BE OBTAINED FROM PT15 POWER SOURCES OTHER THAN THE ONE LOCATED IN THE DEVICE CONTROLLER CABINET, WITH THE EXCEPTION OF THE LINE PRINTER (SEE ITEM 7.)

6 IF THE SYSTEM CONTAINS A RAD SYSTEM, A DEVICE WITH A LOWER STARTING SURGE (SUCH AS A CARD READER OR PAPER TAPE CABINET) SHOULD BE CONNECTED IN REMOTE-SEQUENCE FASHION IMMEDIATELY BEFORE THE RAD. THE PERIPHERAL STATION AND RAD SUBSYSTEM TURN ON SIMULTANEOUSLY AFTER THE PREVIOUS LINE SURGE HAS SETTLED

7 THE 2-KHZ POWER TO THE LINE PRINTER STATION MUST ORIGINATE FROM THE SAME 2-KHZ SOURCE AS THAT TO WHICH THE LINE PRINTER CONTROLLER DC POWER SUPPLIES ARE CONNECTED

8 EIGHT RAD STORAGE UNITS ARE A MAXIMUM CONFIGURATION FOR A RAD SUBSYSTEM. SEE INSTALLATION DRAWING NO. 137115 FOR RAD SUBSYSTEM

901060A.31704

Figure 3-244. System Power Configuration, Typical

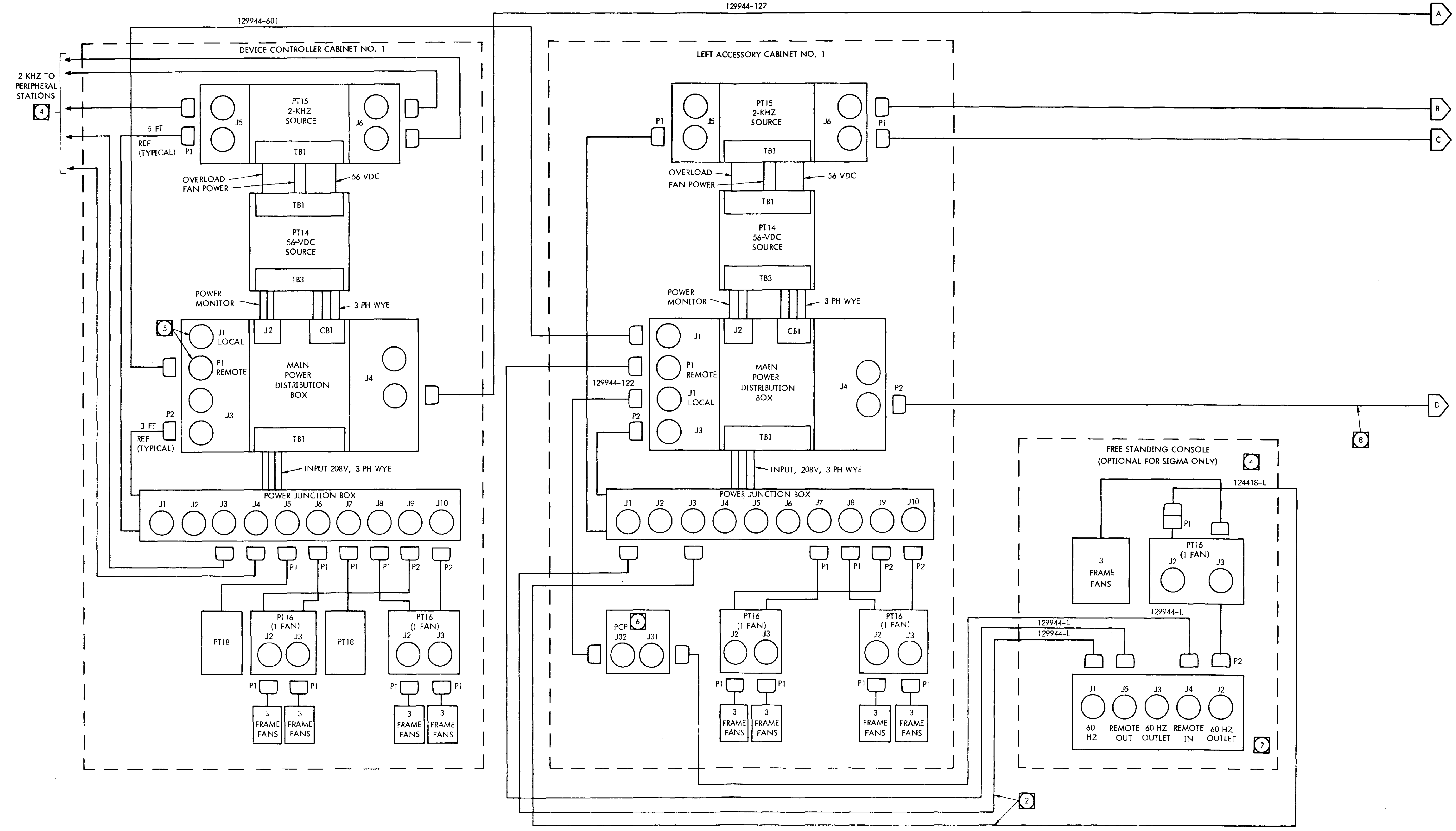


Figure 3-245. System Ac Interconnections, Typical (Sheet 1 of 3)

901060A.31705/1

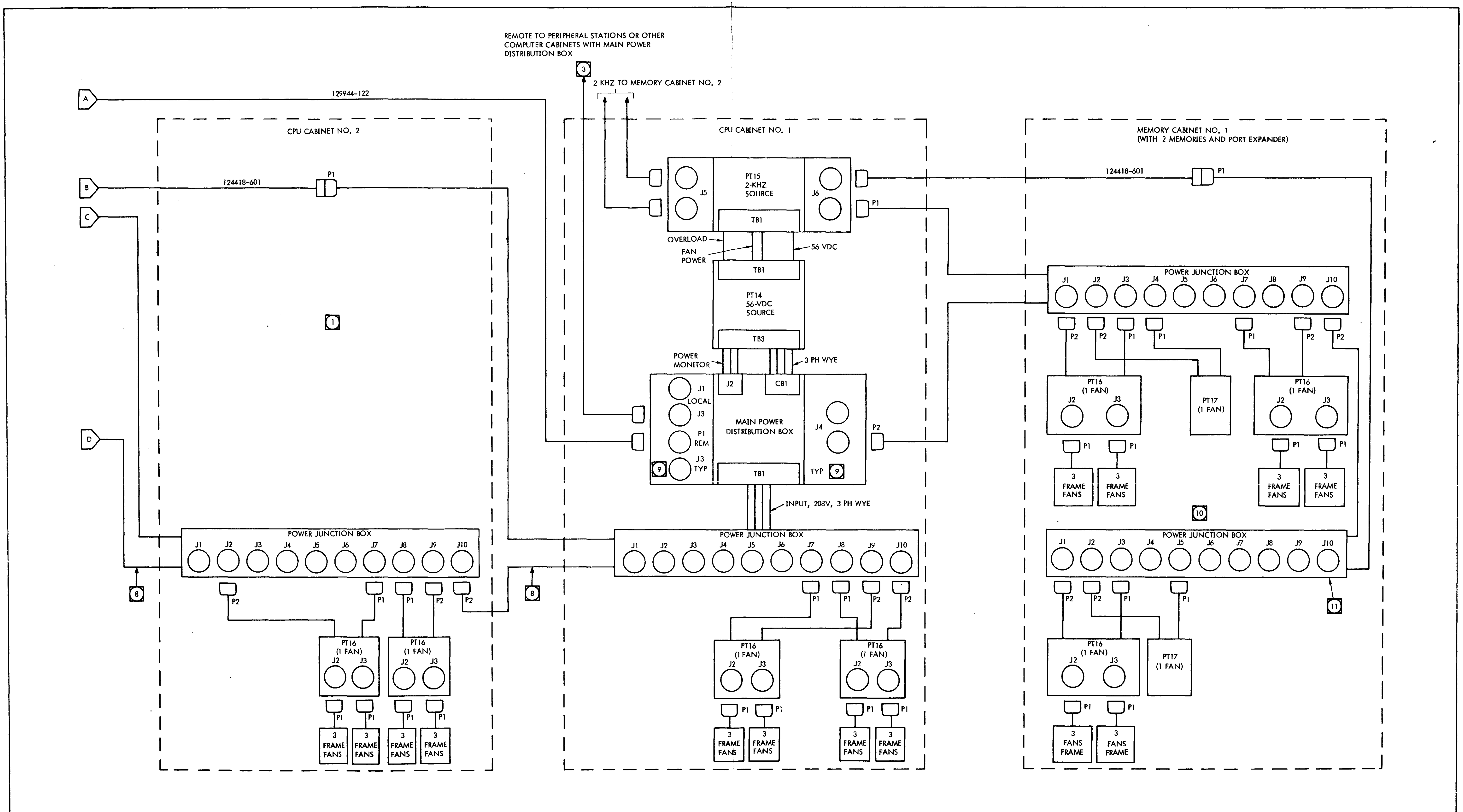


Figure 3-245. System Ac Interconnections, Typical (Sheet 2 of 3)

901060A.31705/2

NOTES:

1 THE PT14 AND PT15 POWER SUPPLIES AND THE MAIN POWER DISTRIBUTION BOX FOR CPU CABINET NO. 2 ARE USUALLY OPTIONAL. IF THE SYSTEM HAS FOUR MEMORY CABINETS, THEY ARE REQUIRED. EXCEPTION: THE MEMORY CABINETS MAY OBTAIN POWER FROM RIGHT ACCESSORY CABINET NO. 1 WITH PT14 AND PT15 POWER SUPPLIES WHEN THEY ARE AVAILABLE

2 POWER EXTENSION CORDS TO THE FREE STANDING CONSOLE CAN CONNECT TO ANY JUNCTION BOX IF CABLE LENGTH, POWER, OR CONNECTOR LIMITATIONS OCCUR. (SEE ITEMS 4 AND 8)

3 THE REMOTE LINE TO PERIPHERAL STATIONS CONNECT IN SEQUENCE IN THE ORDER OF STARTING SURGE (WITH THE HIGHEST ONE FIRST), UNLESS THE SYSTEM CONTAINS A RAD SUBSYSTEM. (SEE FIGURE 3-244, ITEM 8.) THE ORDER OF PERIPHERAL STATIONS, BY DEGREE OF STARTING SURGE (FROM THE HIGHEST TO THE LOWEST), IS: MAGNETIC TAPE CONTROLLER AND TRANSPORT, MAGNETIC TAPE TRANSPORT, RAD SUBSYSTEM, CARD PUNCH, LINE PRINTER, PAPER TAPE EQUIPMENT, AND CARD READER

4 THE TOTAL LENGTH OF ANY 2-KHZ LINE (FROM THE PT15 POWER SUPPLY TO THE DC POWER) MUST NOT EXCEED 60 FEET. THE PREFERRED LENGTH OF 2-KHZ EXTENSION CORD CABLE (NO. 124418) IS LESS THAN 50 FEET. IF NECESSARY, 2-KHZ POWER FOR PERIPHERAL STATIONS MAY BE OBTAINED FROM PT15 POWER SOURCES OTHER THAN THE ONE LOCATED IN THE DEVICE CONTROLLER CABINET, WITH THE EXCEPTION OF THE LINE PRINTER. (SEE FIGURE 3-244, ITEM 7)

5 THE LOCAL DESIGNATION INDICATES LOCAL LINE POWER. THIS REFERS TO SINGLE-PHASE POWER OBTAINED DIRECTLY FROM THE LINE BEFORE THE CONTACTOR. THE REMOTE DESIGNATION INDICATES INPUT THAT WILL ENERGIZE THE CONTACTOR

6 POWER INTERCONNECTIONS FROM THE MAIN POWER DISTRIBUTION BOX TO THE PCP ARE SHOWN BELOW. (THE CONNECTIONS FOR SIGMA 7 WITH A FREE STANDING CONSOLE APPEAR IN FIGURE 3-244)

FROM MAIN POWER DISTRIBUTION BOX	TO SIGMA 7 PCP
LOCAL J1	J31
REMOTE P1	J32

7 P1 ON ANY JUNCTION BOX (2-KHZ REMOTE IN) MUST CONNECT DIRECTLY OR BY MEANS OF AN EXTENSION CORD (NO. 124418-L) TO A PT15 POWER SUPPLY

8 WHEN CHAINING A 60-HZ LINE FROM J-BOX TO J-BOX, THE INPUT LINE ON THE FIRST J-BOX (WHERE P2 CONNECTS TO THE MAIN POWER DISTRIBUTION BOX) SHOULD NOT EXCEED 10 AMPERES. A TOTAL OF 28 FANS ARE SHOWN, DRAWING CURRENT THROUGH THE J-BOX INPUT LINE. EACH FAN DRAWS 0.16 AMPERES. THE TOTAL CURRENT (0.16 x 28) IS 4.5 AMPERES. ANY DEVICE DRAWING MORE THAN 5.5 AMPERES SHOULD NOT BE CONNECTED TO EITHER OF THE TWO J-BOXES

9 J3 AND J4 ON THE MAIN POWER DISTRIBUTION BOXES (60-HZ SWITCHED) ARE USED TO DRIVE FANS. J3 AND J4, PHASES Y AND X RESPECTIVELY, SHOULD BE CONNECTED TO DISTRIBUTE BOTH PHASES UNIFORMLY

10 AN ADDITIONAL J-BOX (NO. 117428) IS REQUIRED ONLY WHEN A REAR FIXED FRAME IS USED

11 THE 60-HZ OUTLETS ARE AVAILABLE FOR CONNECTION TO THE J-BOX (OR BOXES) IN MEMORY CABINET NO. 2, PROVIDING POWER LIMITS ARE NOT EXCEEDED. (SEE ITEM 8.)

901060A. 31705/3

Figure 3-245. System Ac Interconnections, Typical (Sheet 3 of 3)

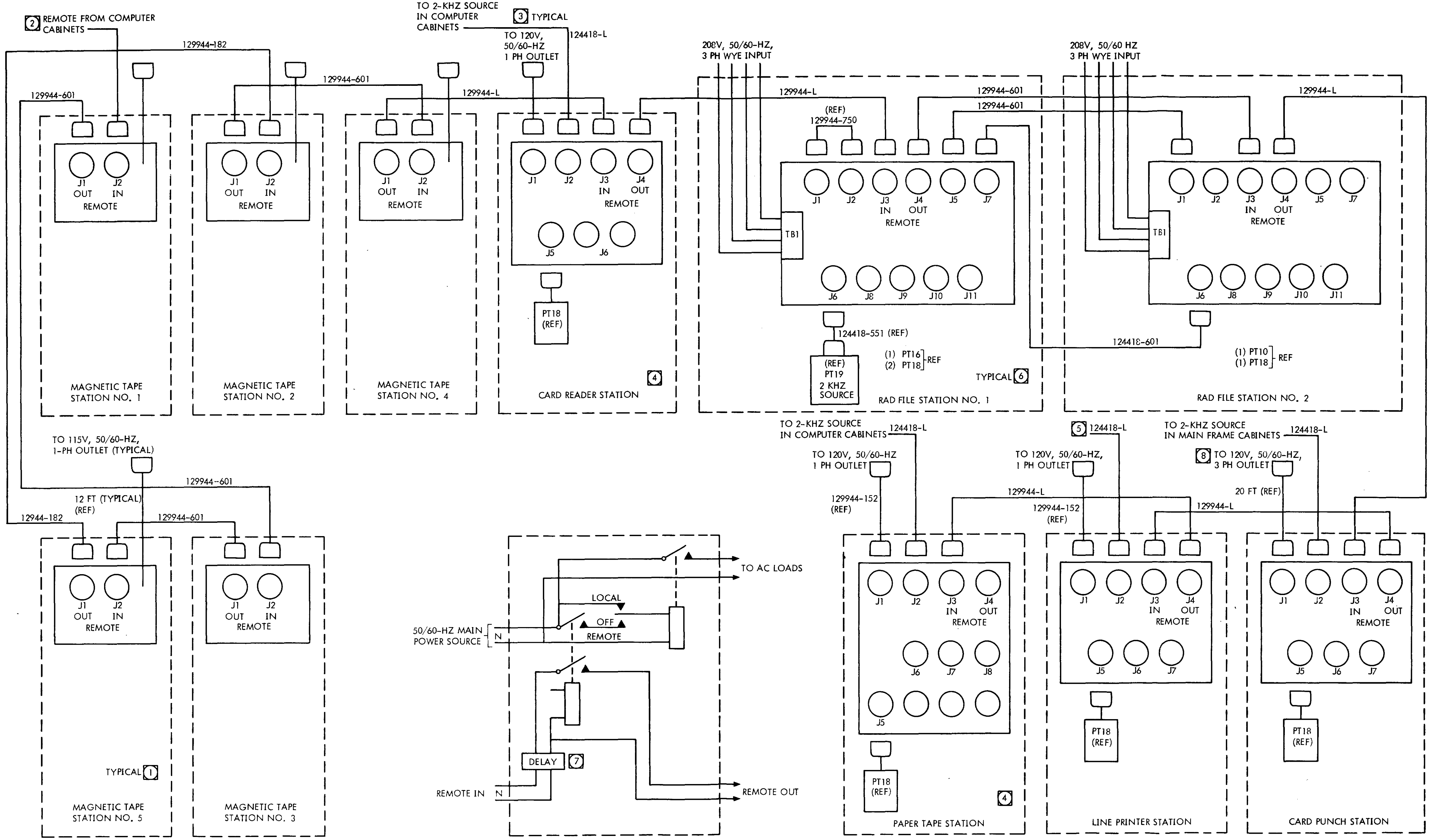

2 REMOTE FROM COMPUTER CABINETS
TO 2-KHZ SOURCE IN COMPUTER CABINETS
3 TYPICAL
TO 120V, 50/60-HZ 1 PH OUTLET
124418-L
129944-182
129944-601
129944-L
208V, 50/60-HZ, 3 PH WYE INPUT
208V, 50/60 HZ 3 PH WYE INPUT
(REF) 129944-750
J1 OUT
J2 IN
REMOTE
J3 IN
J4 OUT
J5
J6
J7
J8
J9
J10
J11
TB1
PT18 (REF)
124418-551 (REF)
(REF) PT19 2 KHZ SOURCE
(1) PT16
(2) PT18
REF
124418-601
(1) PT10
(1) PT18
TYPICAL 6
4
MAGNETIC TAPE STATION NO. 1
MAGNETIC TAPE STATION NO. 2
MAGNETIC TAPE STATION NO. 4
CARD READER STATION
RAD FILE STATION NO. 1
RAD FILE STATION NO. 2
TO 115V, 50/60-HZ, 1-PH OUTLET (TYPICAL)
12 FT (TYPICAL) (REF)
12944-182
TYPICAL 1
MAGNETIC TAPE STATION NO. 5
MAGNETIC TAPE STATION NO. 3
TO AC LOADS
LOCAL
OFF
REMOTE
50/60-HZ MAIN POWER SOURCE
N
DELAY
7
REMOTE IN
REMOTE OUT
TO 2-KHZ SOURCE IN COMPUTER CABINETS
TO 120V, 50/60-HZ 1 PH OUTLET
129944-152 (REF)
5
TO 120V, 50/60-HZ, 1 PH OUTLET
TO 2-KHZ SOURCE IN MAIN FRAME CABINETS
8 TO 120V, 50/60-HZ, 3 PH OUTLET
20 FT (REF)
PAPER TAPE STATION
LINE PRINTER STATION
CARD PUNCH STATION

Figure 3-246. Peripheral Station Power Control, Functional Diagram, Typical (Sheet 1 of 2)

901060A. 31706/1

NOTES:

1 EIGHT MAGNETIC TAPE UNITS ARE A MAXIMUM CONFIGURATION FOR A MAGNETIC TAPE SUBSYSTEM. SEE INSTALLATION DRAWING NO. 137114, MAGNETIC TAPE SUBSYSTEM

2 THE REMOTE LINE TO PERIPHERAL STATIONS CONNECT IN SEQUENCE IN ORDER OF STARTING SURGE, WITH THE HIGHEST ONE FIRST, UNLESS THE SYSTEM CONTAINS A RAD SUBSYSTEM (SEE ITEM 4). THE ORDER OF THE PERIPHERAL STATIONS, BY DEGREE OF STARTING SURGE (FROM HIGHEST TO LOWEST), IS: MAGNETIC TAPE CONTROLLER AND TRANSPORT, MAGNETIC TAPE TRANSPORT, RAD SUBSYSTEM, CARD PUNCH, LINE PRINTER, PAPER TAPE EQUIPMENT, AND CARD READER

3 THE TOTAL LENGTH OF ANY 2-KHZ LINE (FROM PT15 POWER SUPPLY TO DC POWER) MUST NOT EXCEED 60 FEET. THE PREFERRED LENGTH OF 2-KHZ EXTENSION CORD CABLE (NO. 124418) IS LESS THAN 50 FEET. IF NECESSARY, 2-KHZ POWER FOR PERIPHERAL STATIONS MAY BE OBTAINED FROM PT15 POWER SOURCES OTHER THAN THE ONE LOCATED IN THE DEVICE CONTROLLER CABINET, WITH THE EXCEPTION OF THE LINE PRINTER (SEE ITEM 5)

4 IF THE SYSTEM CONTAINS A RAD SUBSYSTEM, A DEVICE WITH A LOWER STARTING SURGE (SUCH AS CARD READER OR PAPER TAPE CABINET) SHOULD BE CONNECTED IN REMOTE-SEQUENCE FASHION IMMEDIATELY BEFORE THE RAD. THE PERIPHERAL STATION AND RAD SUBSYSTEM WILL TURN ON SIMULTANEOUSLY AFTER THE PREVIOUS LINE SURGE HAS SETTLED

5 THE 2-KHZ POWER TO THE LINE PRINTER STATION MUST ORIGINATE FROM THE SAME 2-KHZ SOURCE AS THAT TO WHICH THE LINE PRINTER CONTROLLER DC SUPPLIES ARE CONNECTED

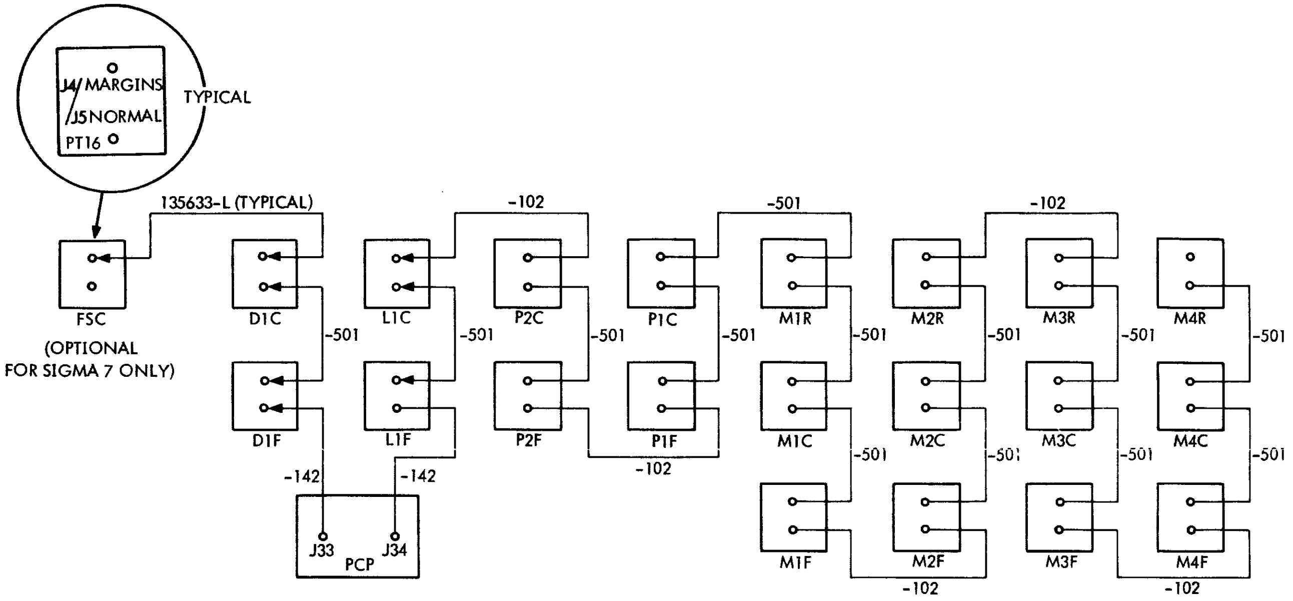
6 EIGHT RAD STORAGE UNITS ARE A MAXIMUM CONFIGURATION FOR A RAD SUBSYSTEM. SEE INSTALLATION DRAWING NO. 137115 FOR RAD SUBSYSTEM

7 WITH THE EXCEPTION OF THE RAD, PERIPHERAL STATIONS HAVE A THERMAL DELAY RELAY AT THE REMOTE INPUT. THE RELAY LIMITS POWER SURGES WHEN THE EQUIPMENT IS TURNED ON

8 ONLY TWO PHASES ARE USED BY THE CARD PUNCH. THE THIRD PHASE IS NOT WIRED TO THE CONNECTOR

901060A. 31706/2

Figure 3-246. Peripheral Station Power Control, Functional Diagram, Typical (Sheet 2 of 2)



NOTE : MARGINAL VOLTAGE CABLING MUST CONNECT TO ALL PT16 POWER SUPPLIES IN THE SYSTEM AND CAN BE CHAINED IN ANY SEQUENCE

901060A. 31707

Figure 3-247. Marginal Voltage Indicator Interconnections, Typical

Table 3-151. Peripheral Equipment and Power Supply Installation Drawings

Drawing Name	Drawing Number
Magnetic Tape Station	128336
Magnetic Tape susbystem	137114
RAD Memory	132841
RAD Memory	126657
RAD Subsystem	137115
Card Punch	124638
Card Reader	124773
Line Printer	123898
Paper Tape Station	127714
PT14 Power Supply, PT15 Power Supply, and Main Power Distribution Box	123310
PT16 Power Supply	123509
PT17 Power Supply	123636
PT18 Power Supply	127156
PT19 Power Supply	127192

Table 3-152. Types of Power Cables

Cable Description	Part Number
50-60 Hz extension cord	129944-L*
2-kHz extension cord	124418-L*
Marginal voltage indicator cable	135663-L*
*The designation -L indicates a length to be specified	

Table 3-153. Symbols Used in Glossaries, Equations, and Phase Sequence Charts

SYMBOL	EXPLANATION
	Prefixes
N	Not. The letter N, before a signal, has the same meaning as the bar that previously appeared over the signal
S/	Set input to flip-flop
R/	Reset input to flip-flop
C/	Clock input to flip-flop
E/	Erase input to flip-flop (DC reset)
F/	Force input to flip-flop (DC set)
W/	Data write input to high-speed memory
L/	Address line to high-speed memory
K/	Read/write control to high-speed memory
	Suffixes
-	Dash (-) followed by number or letter: same logic signal generated by different driver. Example: PXQ-2 = I NPXQ NMULC NMRQ/1 PXQ-3 = I NPXQ NMULC NMRQ/1 Note PXQ-3 is the output of another inverter
/	Slash (/) followed by number or letter: similar logic signal generated by different driver. Example: PX/13 = I NPX NPXS NMULC PX/12 = I NPX NPXS NMULC NPX/2
Z	Usually means 0, as in A0031Z, which means A0 through A31 = 0. Exception: P31Z means force a 0 into P31
W	Usually means a 1
F	Either floating point of flip-flop
	Miscellaneous Symbols
S/name	Flip-flop set input
→	Implies
→	Transfer to
$\nrightarrow$	Clocked transfer to (CL clock)
.	Logical AND
+	Logical OR
⊕	Exclusive OR

Table 3-154. Glossary of CPU Signals

Signal	Definition
A0004W	A0-A4 contain ones
A0004Z	A0-A4 contain zeros
A0007Z	A0-A7 contain zeros
A0015Z	A0-A15 contain zeros
A0031Z	A0-A31 contain zeros
A0107Z	A1-A7 contain zeros
A0115Z	A1-A15 contain zeros
A0131Z	A1-A31 contain zeros
A1631Z	A16-A31 contain zeros
A1731Z	A17-A31 contain zeros
A4771Z	A47-A71 contain zeros
A8X1	A8 loaded by clock with a one
A9X1	A9 loaded by clock with a one
AB0i	Where i = A, B, or C. Abort core memory port N because of memory lockout
ADMATCH	Address match between the LM and KSP lines 15-31
AH	Add halfword, hexadecimal opcode 10 or 90
AHCL	Memory address here clock
AHi	Where i = A, B, or C. Core memory address; signal sent from port i
ALARM	Causes PCP ALARM light to light, and causes a 1KC alarm to sound if the COMPUTE switch is set to RUN and the AUDIO switch is set to ON
AM	Flip-flop holding the arithmetic mask bit for program status word 1
AND	AND word, hexadecimal opcode 4B or CB
ANLZ	Flip-flop used during execution of analyze instruction
ARQ	Flip-flop indicating that the next clock will not occur until the address release signal (ARC) has been received from memory if there is a memory request out
ARi	Where i = A, B, or C. Core memory address release signal sent from port i
ASN	A simple normalized (that is, $-1 \leq A < -\frac{1}{16}$ or $\frac{1}{16} \leq A < 1$)
ATE	Memory address time elapsed

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
AUDIO	Signal sent to the PCP speaker
AX	Signal is true to load or reset the A-register. This reset term is voided if a simultaneous set occurs
AX/1	0 —/→ A0-A31, 0 —/→ A47-A71
AXAL32	A0 + A47 —/→ A47, A8 + A48 —/→ A48, ... A31 + A71 —/→ A71
AXE	E0-E7 —/→ A0-A7, 0 —/→ A8-A31, 0 —/→ A47-A71. Special action to A0 and A1 during FAMDSF
AXE/1	E0 + A0 —/→ A0, E1 + A1 —/→ A1, ... E7 + A7 —/→ A7. Special action to A0 and A1 during FAMDSF
AXPRR2	A30 —/→ A0, A31 —/→ A1, PR0-20 —/→ A2-A31, A47 + D46 —/→ A47. Special action to A0 and A1 during FAMDSF
AXR	R28-R31 —/→ A28-A31, 0 —/→ A0-A27, 0 → A57-A71
AXRR	RR0-RR31 —/→ A0-A31, 0 —/→ A47-A71
AXRR/2	RR16-RR31 —/→ A16-A31, 0 —/→ A0-A15, 0 → A47-A71
AXRR/3	RR24-RR31 —/→ A24-A31, 0 —/→ A0-A23, 0 —/→ A47-A71
AXRRL1	RR16-RR31 —/→ A15-A30, 0 —/→ A0-A14, 0 —/→ A31, 0 —/→ A47-A71
AXRRR1	RR14-RR30 —/→ A15-A31, RR31 —/→ P32, 0 —/→ P33, 0 → A47-A71
AXRRR2	RR13-RR29 —/→ A15-A31, RR30-RR31 —/→ P32-P33, 0 —/→ A47-A71
AXS	S0-S31 —/→ A0-A31, S47-S71 —/→ A47-A71
AXS/1	S0-S7 —/→ A0-A7, 0 —/→ A8-A31, 0 —/→ A47-A71
AXS/2	S0-S15 —/→ A0-A15, 0 —/→ A16-A31, 0 —/→ A47-A71
AXS/3	S8-S15 —/→ A8-A31, 0 —/→ A0-A7, 0 —/→ A47-A71
AXS/4	S47-S71 —/→ A47-A71, 0 —/→ A0-A31
AXSL1	S1-S31 —/→ A0-A30, A0 —/→ A31, S48-S71 —/→ A47-A70, S0 —/→ A71. Special action to A31 during FAMDSF
AXSL1/1	S9-S31 —/→ A8-A30, A0 —/→ A31, S48 —/→ A47, 0 —/→ A48-A71. Special action to A31 during FAMDSF
AXSL4	S4-S31 —/→ A0-A27, A0-A3 —/→ A48-A31, S51-S71 —/→ A47-A67, S0-S3 → A68-A71. Special action to A28-A31 during FAMDSF
AXSR4	S68-S71 —/→ A0-A3, S0-S27 —/→ A4-A31, S47-S67 —/→ A51-A71, 0 —/→ A47-A50. Special action to A50 during FAFLAS
AXSR32	S48 + A8 —/→ A8, S49 + A9 —/→ A9 ... S71 + A31 —/→ A31

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
B0S	B0 sum
B1S	B1 sum
B1619ONE	Flip-flops B16-B19 contain 0001
B1619Z	B16-B19 contain zeros
B31X1	Load B31 by clock with a one
BC1	B carry into position 1 (used in multiply)
BC31	B carry into position 31 (used in BCON logic; also used in shift)
BCON	B control, high during multiply when the multiplier is being interrogated
BR	BRPH1 + ... + BRPH15 + CLEAR + MITEX
BRPHi	Where i = 1-15. Branch to phase i and inhibit the nonbranch phase setting
BRQ	Causes Q ⇸ P, used during trapping or interrupting to make sure that the address stored by the XPSD instruction points at the right instruction
BWZ	B was zero. Used in divide overflow logic (fixed point)
BX/1	0 ⇸ B0-B31, 0 ⇸ B47-B71
BXB	PR32 ⊕ G33 ⇸ B0, B1 ⊕ BC1 ⊕ CS33 ⇸ B1, B2-B7 ⇸ B2-B7. Special action to B47-B71 during FAFLM
BXBL1	B1 ⊕ BC1 ⊕ CS33 ⇸ B0, B2-B31 ⇸ B1-B30, B0 ⇸ B31, 0 ⇸ B47, B49-B71 ⇸ B48-B70, B0 ⇸ B71. Special action to B31 during FAMDSF
BXBL4	B4-B31 ⇸ B0-B27, 0 ⇸ B28-B31, 0 ⇸ B47-B71. Special action to B28-B31 during FASHFX
BXBR2	B30-B31 ⇸ B0-B1, B0-B29 ⇸ B2-B31. Special action to B0-B3 during FAMDSF. Special action to B4-B11 and B48-B71 during FAFLM
BXND	ND1-ND7 ⇸ B1-B7, 0 ⇸ B0, 0 ⇸ B8-B31, 0 ⇸ B47-B71 ⇸
BXS	S0-S31 ⇸ B0-B31, S47-S71 ⇸ B47-B71 ⇸
BXS/1	S47 + B47 ⇸ B47, S48 + B48 ⇸ B48, . . . S71 + B71 ⇸ B71
C0C16	Holding C0 or C16 depending on P32
C0C16/1	7 used for sign extension in C ⇸ D transfers
C0C16, 12	5 used for sign extension in C ⇸ D transfers
CC1	Condition code 1
	Condition code 2

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
CC3	Condition code 3
CC4	Condition code 4
CCXDU	DU0 → 1 -/→ CC1, DU6 → 0 -/→ CC1, DU1 → 1 -/→ CC2, DU7 → 0 -/→ CC2
CCXDU/1	DU2 -/→ CC3, DU3 -/→ CC4
CCXS	S0-S3 -/→ CC1-CC4
CCXTRACC	TRACC1-TRACC4 -/→ CC1-CC4
CCWCM	C-register contains one's complement for multiply
CCZ	True if CC1 = CC2 = CC3 = CC4 = 0
CE	Clock enable
CEINT	Causes next clock to be synchronized with 2.048 MHz interrupt clock. Clock enable by interrupt
CK/1	Scratch pad register clock, also CK/2 - CK/12
CKT2NL	Scratch pad register clock from DL3/040 NDL3/080 where N = 2, 3, 4, or 5 for T4L, T6L, or T10L, respectively
CL/1	Ac for clock for CPU, also CL/2 - CL/12
CLEAR	True if ENDE or S/INTRAPF or RESET
CLEARMEM	True if both SYSTEM RESET/CLEAR and CPU RESET/CLEAR are set simultaneously
CLEXT	Extended clock
CLEXTE	Extended clock enable
CLT1LE	T1L even clock
CLT1LO	T1L odd clock
CLT2NL	CL clock from DL3/060 NDL3/100 where N = 2, 3, 4, or 5 for T4L, T6L, T8L, or T10L, respectively
CLT4LG	Generate T4L clock
CLT6LG	Generate T6L clock
CLT8LG	Generate T8L clock
CLT10LG	Generate T10L clock
CPULi	Where i = 1-4. Count pulses for the counter interrupts
CRO	Decoded from LM15-LM22 and LB23-LB27. Crossover from core memory address to fast memory address

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
CROF	Crossover flip-flop (actually buffer latch) indicating that crossover addressing is being used
CS32	Extension of CS-register, needed for multiply
CS33	Extension of CS-register, needed for multiply
CS7X1	1 –/→ CS7
CSi	Carry-save register. Where i = 0-33 or 47-71
CSN	C simple normalized (that is, $-1 \leq C < -\frac{1}{16}$ or $\frac{1}{16} \leq C < 1$)
CSX1	1's –/→ CS0-CS31, 1's –/→ CS47-CS71, but not CS32 or CS33
CSX1/1	1's –/→ CS0-CS29
CSX1/2	1's –/→ CS0-CS27
CSX1/3	1's –/→ CS0-CS15
CSX1/5	1's –/→ CS0-CS7
CSX1/8	1's –/→ CS31
CSX1/9	CS0-CS27 loaded by clock with ones. Intermediate control term
CSX1/10	CS28-CS31 and CS47-CS71 loaded by clock with ones. Intermediate control term
CSXGR1	G0-G31 –/→ CS1-CS32, B0 CS32 + PR32 G33 –/→ CS33, G47-G70 CS48-CS71, DCWCM => 1 –/→ BC1
CX/1	Causes 0's –/→ C0-C31 and 0 –/→ C0C16
CXCL32 dc rep	C0 –/→ C47, C8-C31 C48-C71
CXMB	C0-C31 loaded from the memory bus data lines MB0-MB31
CXMB/D	CXMB after a 50 nanosecond time delay
CXMB/i	Where i = 1-4 or 10-13. Same as CXMB/D
CXRR	Causes RR0-RR31 –/→ C0-C31, NP32 => RR0 –/→ C0C16, P32 => RR16 –/→ C0C16
CXS	Causes S0-S31 –/→ C0-C31 and S47-S71 –/→ C47-C71
D27X1	D27 loaded by clock with a one
DCH	DC hold clock for CXS and CXRR (C-register latch)
DCSTOP	Signal to halt the CPU if a core memory parity error occurs with the PARITY ERROR MODE switch in the HALT position, or if an address match occurs with the ADDR STOP switch on
DCWCM	D-register contains one's complement for multiply

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
DEOPTION	True if decimal option implemented
DIO48X1	1 ⟶ /DIO48/ (RD/WD function strobe)
DIO50X1	1 ⟶ /DIO50/ (indicates WD instruction)
DIOXB	B16-B31 ⟶ /DIO32-DIO47/(RD/WD address lines)
DIOXS	S0-S31 ⟶ /DIO0-DIO31/(RD/WD data lines)
DIT	High during divide iterations
DITEX	Exit from DIT (high last clock)
DL1	CPU delay line 1 for clock timing
DL1/D1	Positive timing pulse with delay D1 (DL1/030) with respect to CL-1 leading edge
DL1/D2	DL1 tap after delay D2 (DL1/050) with respect to CL-1 leading edge
DL1/iii	Delay line 1 tap for iii ns
DL1E	CPU DL1 enable
DL1E/S	DL1 enable set (CPU DL1/180)
DL1PC	Delay line 1 pulse cutoff (CPU DL1/080)
DL1PE	Delay line 1 pulse width expander (CPU DL1/020)
DL2	CPU delay line 2, clock timing
DL2/D1	DL2 tap after delay D1 (DL2/100)
DL2/D2	DL2 tap after delay D2 (DL2/150)
DL2I	DL2 initiate
DL2PC	Delay line 2 pulse cutoff (DL2/080)
DL2PE	Delay line 2 pulse width expander (DL2/020)
DL2R	DL2 recirculate
DL2RE	DL2 recirculate enable
DL2TC1	DL2 trip counter first flip-flop
DL2TC2	DL2 trip counter second flip-flop
DL2TCC	DL2 trip counter clock
DL3	CPU delay line 3, clock phase
DL3PC	Delay line 3 pulse width cutoff (DL3/080)

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
DL3PE	Delay line 3 pulse width expander (DL3/020)
DL4	CPU delay line 4, extended clock generator
DLSE1	CPU delay line sensor enable 1, qualifies CLT1LO, CLT4LG, CLT6LG, and CLT8LG
DLSE1/S	Set delay line sensor enable 1 (CPU DL1/120)
DLSE2	CPU delay line sensor enable 2, qualifies DL2I and CLT10LG
DLSE2/S	Set delay line sensor enable 2 (CPU DL1/240)
DLSE3	CPU delay line sensor enable 3, DL2R and (TP520 NTP560)
DLSE3/S	Set delay line sensor enable 3 (CPU DL2/150)
DPP	Divide preparation
DR	Data release signal from memory after the buffer cable receiver
DRi	Where i = A, B, or C. Core memory data release signal sent from port i
DRQ	Flip-flop indicating that the next clock will not occur until the data release signal has been received from memory if there is a memory request out
DU0-DU7	Data lines to decimal unit
DU8	DUCLOCK signal to decimal unit
DU9	DUSTART signal to decimal unit
DU10	DUEND signal from decimal unit
DU11	DUCLEAR signal to decimal unit
DU12	DUMDM signal to decimal unit
DUCLEAR	Reset signal to decimal unit (DU11 on cable)
DUCLOCK	Signal used to gate clock in decimal unit (DU8 on cable)
DUEND	Signal from decimal unit indicating the end of a particular phase of decimal unit operation (DU10 on cable)
DUMDM	Decimal trap mask or EBS mark option or interrupt request for use during DM and DD (DU12 on cable)
DUSTART	Signal indicating the beginning of a decimal instruction (DU9 on cable)
DUTI	Decimal unit trap immediately
DUTEI	Decimal unit trap at end of instruction
DUXCC	CC1-CC4 → /DU0-DU3/
DUXO	O4-O7 → /DU4-DU7/

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
DUXR	R28-R31 → /DU0-DU3/
DUXS	S24-S31 → /DU0-DU7/
DX/1	0 ⇸ D0-D31, 0 ⇸ D46-D71
DXC/1	C24-C31 ⇸ D24-D31, C47 ⇸ D46-D47, 0 ⇸ D0-D23, 0 ⇸ D48-D71
DXC/2	C0C16 ⇸ D0-D15, C16-C31 ⇸ D16-D31, C47 ⇸ D46-D47, 0 ⇸ D48-D71
DXC/3	C12-C31 ⇸ D12-D31, C47 ⇸ D46-D47, 0 ⇸ D0-D11, 0 ⇸ D48-D71
DXC/4	C12 ⇸ D0-D11, C12-C31 ⇸ D12-D31, C47 ⇸ D46-D47, 0 ⇸ D48-D71
DXC/5	Downward align halfword C to D according to P32, NP32 → DXCR16 P32 → DXC/2
DXC/6	C0-C31 ⇸ D0-D31, C47 ⇸ D46, C47-C63 ⇸ D47-D63 FAFL → C64-C71 D64-D71 NFAFL ⇸ 0 ⇸ D64-D67, 1's ⇸ D68-D70, C0 ⇸ D71
DXC/7	C16 D16 ⇸ D16, C17 D17 ⇸ D17, ... C31 D31 ⇸ D31, C47 D46 ⇸ D46, C47 D47 ⇸ D47, Intermediate control signal
DXC/8	C12 D12 ⇸ D12, C13 D13 ⇸ D13, ... C15 D15 ⇸ D15, Intermediate control signal
DXC/9	C4-C15 ⇸ D20-D31, 0 ⇸ D0-D19, 0 ⇸ D46-D71
DXC/10	DXC/6 + DXCM + DXC/D
DXC/13	DXC/1 + DXC/7 + DXC/D + DXCM
DXC/D	DXC generated by divide instructions
DXCBP	Downward align byte C to D according to P32 and P33 NP32 NP33 → NDXCR24 NP32 P33 → NDXCR16/1 P32 NP33 → NDXCR8 P32 P33 → NDXC/1
DXCC	CC1-CC4 ⇸ D28-D31, 0 ⇸ D0-D27, 0 ⇸ D46-D71
DXCL/M	Multiplicand control flip-flop (2C ⇸ D)
DXCL1	Load the D-register from the C-register with a left one shift and a clock
DXCM	Multiplicand control flip-flop (C ⇸ D)
DXCR8	C16-C23 ⇸ D24-D31, 0 ⇸ D0-D23, 0 ⇸ D46-D71
DXCR16	C0-C15 ⇸ D16-D31, C0-C16 ⇸ D0-D15, 0's ⇸ D46-D71
DXCR16/1	C8-C15 ⇸ D24-D31, 0 ⇸ D0-D23, 0 ⇸ D46-D71
DXCR24	C0-C7 ⇸ D24-D31, 0 ⇸ D0-D23, 0 ⇸ D46-D71
DXDI0	DI00-DI31 ⇸ D0-D31, 0 ⇸ D46-D71

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
DXDU	DU0-DU7 ⇸ D24-D31, 0 ⇸ D0-D23, 0 ⇸ D46-D71
DXK	KS0 + KNC0 D0 ⇸ D0, KS1 + KNC1 D1 ⇸ D1, ... KS31 + KNC31 D31 ⇸ D31, 0 ⇸ D46-D71
DXNC	NC0-NC31 ⇸ D0-D31, NC47 ⇸ D46-D47, NC48-NC63 ⇸ D48-D63 FAFL → NC64-NC71 ⇸ D64-D71 NFAFL ⇸ 0 ⇸ D64-D67, 1's ⇸ D68-D70, NC0 ⇸ D71
DXNC/1	Same as NDXNC and 1 ⇸ CS31
DXNC/2	NC0C16 ⇸ D0-D11,NC16-NC31 ⇸ D16-D31, NC47 ⇸ D46-D47 0 ⇸ D48-D71
DXNC/3	Downward align halfword NC to D according to P32 NP32 → NDXNCR16 P32 → NDXNC/2
DXNCM	Multiplicand control flip-flop (NC ⇸ D)
DXNCR1G	NC0C1G ⇸ D0-D11, NC0-NC15 ⇸ D16-D31, 0 ⇸ D46-D71
DXPARITY	MFL0-MFL7 → D24-D31, 0's → D0-D23 and D46-D71
DXPCP	PCP ⇸ D
DXPSW1	CC1-CC4 ⇸ D0-D3, FS ⇸ D5, FZ ⇸ D6, FNF ⇸ D7, MASTERF ⇸ D8, MAPF ⇸ D9, DM ⇸ D10, AM ⇸ D11, 0 ⇸ D4, 0's ⇸ D12-D31 and D46-D71
DXPSW2	WK0-WK1 ⇸ D2-D3, CIF ⇸ D5, II ⇸ D6, EI ⇸ D7, RP23-RP27 ⇸ D23-D27, 0's ⇸ D0, D1, D4, D8-D22, D28-D31 and D46-D71
DXTRACC	TRACC1 + D28 ⇸ D28, TRACC2 + D29 ⇸ D29, TRACC3 + D30 ⇸ D30, TRACC4 + D31 ⇸ D31
(E = 1)	E = 1
(E = 1 or 2)	E = 1 or 2
E05Z	E0-5 ≠ 0
E06Z	E0-6 ≠ 0
E1X1	1 ⇸ E1
E6X1	1 ⇸ E6
E7X1	1 ⇸ E7
EDR	Early data release signal from memory after the buffer cable receiver
EDRi	Where i = A, B, or C. Early data release signal in memory for port i

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition

ENDE	Signal that is true during the last execution phase of an instruction (end execution)
	ENDE → NDXC/6
	ENDE → NORXC
	ENDE → NOXC
	ENDE NHALT N(S/INTRAPF) => S/PRE1
	ENDE → (S/LRXD)
	ENDE HALT N(S/INTRAPF) => S/PCP1
	ENDE → CLEAR
	ENDE N(TRAP + HALT + FUEXU + KAHOLD) => PCTP1

ES4	If ES4 is false when MCTE1 is true, E-4 ⇸ E is performed If ES4 is true when MCTE1 is true, E-1 ⇸ E is performed
EX	0 ⇸ E0-E7, intermediate control signal
EX/1	0 ⇸ E0-E7
EXB	NB1 ⇸ E0-E1, B2-B7 ⇸ E2-E7
EXCC	CC1-CC4 ⇸ E4-E7, 0 ⇸ E0-E3
EXS	S0-S7 ⇸ E0-E7
EXU	Execute flip-flop, high from PH1 through ENDE of all instructions
FABC	Family of branch on conditions (BCR, BCS)
FABO	Family of byte operations (TTBS, TBS, CBS, MBS, EBS) with NPHA
FABO/1	Family decode for MBS, CBS, TBS, and TTBS instructions
FABO/2	Family decode for MBS, CBS, and EBS instructions
FABOA	Family of byte operations (TTBS, TBS, CBS, MBS, EBS) with PHA
FABOA/1	Same as FABO/1 with PHA
FABOA/2	Same as FABO/2 with PHA
FABOX	Family of byte operations (TTBS, TBC, CBS, MBS, EBS)
FABR	Family of branch on decrementing or incrementing register (BDR, BIR)
FABRANCH	BDR, BIR, BCR, BCS, BAL
FABS	Family decode for byte string instructions (MBS, CBS)
FABSA	Same as FABS with PHA
FACAL	CAL1, 2, 3, 4. Family of CALL instructions
FACV	Family decode for conversion (CVA, CVS)
FADE	Family decode for decimal (DL, DST, DA, DS, DM, DD, DC, DSA, PAK, UNPK)

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
FADIO	Family of direct and input/output instructions (SIO, TIO, TDV, HIO, AIO, RD, WD)
FADIV	Family of divide instructions, fixed-point (DW, DH)
FADIVH	DH
FADIVW	DW
FADW	Family of double word instructions. Consists of 19 instructions, opcodes 08 through 1F, inclusive
FAFL	Family decode for floating point arithmetic instructions (FAL, FSL, FML, FDL, FAS, FSS, FMS, FDS)
FAFLAS	Floating point add and subtract (FAS, FSS, FAL, FSL)
FAFLD	Family of floating divide instructions (FDL, FSD)
FAFLM	Family of floating multiply instructions (FML, FMS)
FAFLMD	Floating point multiply and divide (FMS, FD, FML, FDL)
FAFRR	S, STD, MI
FAFRR/1	DW, MW, STS, FML
FAILL	Family of illegal instructions. Consists of all of the following illegal opcodes, both directly and indirectly addressed, plus all indirectly addressed immediate instructions: 00, 01, 03, 0C, 0D, 14, 16, 17, 26, 27, 2C, 2D, 34, 42, 43, 54, 59, 5C, 5D, 5E, 5F, 62
FAIM	Family of immediate instructions (LCFI, AI, CI, LI, MI, TTBS, TBS, CBS, MBS, EBS)
FAIO	Family of input/output instructions (SIO, TIO, TDV, HIO, AIO)
FAIO/1	SIO, TIO, TDV, HIO
FAMDSF	Family of multiply, divide, shift, and floating instructions (MW, MH, MI, DW, HD, S, SF, FSL, FAL, FDL, FML, FSS, FAS, FD, FMS)
FAMDSF/D	Family of divide instructions (DW, DH, FDL, FDS)
FAMDSF/M	Family of multiply instructions (MW, MH, MI, FML, FMS)
FAMUL	Family of fixed-point multiply instructions (MW, MH, MI)
FAMULH	Family of multiply halfword (MH)
FAMULI	Family of multiply immediate (MI)
FAMULNH	Family of multiply not halfword (MW, MI)
FAMULW	Family of multiply word (MW)
FANIMP	Family of nonimplemented instructions. Consists of the floating point, decimal, and edit instructions, but the decoding signal is gated with either NFPOPTION or NDEOPTION

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
FAPRIV	Family of privileged instructions (LPSD, XPSD, WAIT, LRP, MMC, RD, WD, SIO, TIO, TDV, HIO, AIO)
FAPSD	Family of program status doubleword instructions (LPSD, XPSD)
FARWD	Family of read and write direct instructions (RD, WD)
FAS1	CLM, CLR
FAS2	SD, CD
FAS3	AD, SD
FAS6	CS, STS, LS
FAS7	MTW, MTH, MTB
FAS8	MTW, AWM
FAS9	XW, OR, EOR, AND
FAS10	LAW, LAH
FAS11	CI, CW, CH, CB
FAS12	AI, AW, SW, AH, SH
FAS13	MTH, MTB
FAS14	STS, STD
FAS15	CS, LS
FAS16	LCD, LAD, LD
FAS17	LCF, LCFI
FAS18	STCF, STB, STW, STH
FAS19	LCD, LAD
FAS21	LW, LH, LB, LI, LCW, LCH, LAW, LAH
FAS22	AD, CD, SD, CLM, CLR
FAS23	LW, LH, LB, LI, LCW, LCH
FAS24	AWM, MTW, MTB, MTH
FAS26	LCW, LAW
FASH	Family of shift instructions (S, SF)
FASHFL	Family term for the floating point shift instruction (SF)

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
FASHFX	Family term for the fixed-point shift instruction (S)
FASi	Families of simple opcodes where i equals a number from 1 to 26
FAST/1	Family of stack instructions (PLW, PSW, PLM, PSM, MSP, LM, STM)
FAST	PLM, PSW, PSM, MSP, LM, STM, PLW (with NPHA)
FASTA	(FAST PHA)
FATR	TBS, TTBS. Family of translate instructions (with NPHA)
FATR5	TBS, TTBS. Family of translate instructions (with NPHA, PH5)
FATRA	TBS, TTBS. Family of translate instructions (with PHA)
FAW	Family of word instructions. Consists of all of the following opcodes, whether implemented or not: 04-07, 24-3F, 44-4F, 64-6F
FEOF	Floating point exponent overflow
FEUF	Floating point exponent underflow
FLMC	On during next-to-last clock of MIT in floating multiply
FLPN	Floating point post-normalize
FLRR	Floating point result ready
FMOF	Floating point mantissa overflow
FPOPTION	Floating point option
FPR	Floating point polarity (of result) reversed
FSZNXS	S5 $\nrightarrow$ FS, S6 $\nrightarrow$ FZ, S7 $\nrightarrow$ FNF
FUAD	AD
FUAND	AND
FUANLZ	ANLZ
FUAWM	AWM
FUBAL	BAL
FUCB	CB
FUCI	CI
FUCLM	CLM
FUCLR	CLR

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
FUCS	CS
FUCVS	CVS
FUEBS	EBS (with NPHA)
FUEBS8	EBS (with NPHA PH8)
FUEBS9	EBS (with NPHA PH9)
FUEBS14	EBS (with NPHA PH14)
FUEBSA	EBS (with PHA)
FUEOR	EOR
FUEXD	EXU
FUiii	A signal decoded from the opcode register for the function of one particular instruction, iii
FUINT	INT
FULAD	LAD
FULCD	LCD
FULD	LD
FULI	LI
FULRP	LRP
FULS	LS
FUMMC	MMC
FUMTB	MTB
FUMTH	MTH
FUMTW	MTW
FUOR	OR
FUSTCF	STCF
FUSTD	STD
FUSTH	STH
FUSTS	STS
FUSTW	STW
FUWAIT	WAIT

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
FUXW	XW
GN	Where N = 0-31 or 33 or 47-71. Carry generate term activated by two or more ones in its respective binary digit
(S/NGX)	Terms for -A or -D ⟶ S (preset)
HALT	Flip-flop that causes the CPU to halt in PCP2 S/HALT = FUWAIT + NKRUN/B PH1 + PCP7 + DCSTOP PH1 + RESET R/HALT = INT NDCSTOP PCP2 KRUN/B + PCP2 NKAS/B
HOF	Halt on fault in memory
IA	Indirect address flip-flop set by bit 0 of the instruction word
IEN	Interrupt enable. An interrupt can be accepted on any clock if this flip-flop is set; otherwise, an interrupt can be accepted only at ENDE
INDX	Nonzero index field and not an immediate instruction. Indexing is to be performed
INHXWD	True during PH2 of internal mode WD which sets or resets interrupt inhibits
INT	Flip-flop indicating that an interrupt request from the interrupt logic is present. Also the mnemonic for the interpret instruction with hexadecimal opcode 6B or EB
INTRAPF	Flip-flop that indicates an XPSD, MTW, MTH, or MTB is being executed as the result of an interrupt or trap
(S/INTRAPF/1)	Signal that when true indicates that the CPU is halted by a parity error or an address stop or that the COMPUTE switch is not in RUN
INTRAPi	Where i = 1 or 2, interrupt or trap phase flip-flops
IOPXFCAD	02 ⟶ /FNC0/, 06 ⟶ /FNC1/, 07 ⟶ /FNC2/, B21-B23 ⟶ /IOPA0-2/, CPU to IOP. Function code lines and IOP address
IOPXSTRB	1 ⟶ /CNST/, CPU to IOP strobe
INTRAPF	The interrupt or trap flip-flop
IORESET	Signal that when false indicates that the I/O RESET switch is set or that power is going off or on
IX	Index flip-flop set by a nonzero index field (bits 12-14) of the instruction word.
K00H	Flip-flop for holding K00 (end-carry) for double precision floating shifts
K31	Carry into position 31 of the adder
KADDRSTOP	True if ADDR STOP switch is ON

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
KAHOLD	True if INSTR ADDR switch is in HOLD position
KAS/1 and KAS/2	If KAS/1 is true and NKAS/2 is false, one of the following PCP switches is set: DATA ENTER, DATA CLEAR, STORE SELECT ADDR, STORE INSTR ADDR, INSERT PSW2, INSERT PSW1, COMPUTE STEP, COMPUTE RUN, DISPLAY SELECT ADDR, DISPLAY INSTR ADDR, INSTR ADDR INCREMENT, FILL, or both SYSTEM RESET/CLEAR and CPU RESET/CLEAR
KAS/B	If KAS/B is true, one or more of the following PCP switches is set: DATA ENTER, DATA CLEAR, STORE SELECT ADDR, STORE INSTR ADDR, INSERT PSW2, INSERT PSW1, COMPUTE STEP, COMPUTE RUN, DISPLAY SELECT ADDR, DISPLAY INSTR ADDR, INSTR ADDR INCREMENT, FILL, or both SYSTEM RESET/CLEAR and CPU RESET/CLEAR
KC	True if CLOCK MODE switch is in SINGLE STEP position (normally open contact)
KC/B	True if CLOCK MODE switch is in SINGLE STEP position (normally closed contact generates NKC/B)
KCLEARD/B	True if DATA CLEAR switch is set
KCPURESET	True if CPU RESET switch is set (normally open contact)
KCPURESET/B	True if CPU RESET switch is set (normally closed contact generates NKCPURESET/B)
KD	True if REGISTER DISPLAY switch is in the ON position and CLOCK MODE switch is not in the CONT position
KDISPLA/B	True if either the DISPLAY SELECT ADDR switch or the DISPLAY INSTR ADDR switch is set
KDISPLAK/B	True if DISPLAY SELECT ADDR switch is set
KDISPLAQ/B	True if DISPLAY INSTR ADDR switch is set
KDSHI	True if REGISTER SELECT switch is in any of the following positions: AH, BH, CSHE, DH, SH
KENTERD/B	True if DATA ENTER switch is set
KFILL/B	True if LOAD switch is set
KHOP	True if PARITY ERROR MODE switch is in the HALT position
Kii	Where ii = 0-31 or 47-70. Carry into position NN of adder
KIDLE/B	True if COMPUTE switch is in the IDLE position
KINCRE/B	True if INSTR ADDR INCREMENT switch isset
KINLVSEL	True if INTERLEAVE SELECT switch is in the DIAGNOSTIC position
KINTRP	True if INTERRUPT switch is set (normally open contact generates NKINTRP)
KINTRP/B	False if INTERRUPT switch is not set (normally closed contact)
KIORESET	True if I/O RESET switch is set
KNC0	True if DATA 0 switch is in either center or in the 1 position

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
KNC1-31	Same as KNC0 for switch positions 1-31
KPSW/B	True if either the INSERT PSW1 switch or the INSERT PSW2 switch is set
KPSW1/B	True if INSERT PSW1 switch is set
KPSW2/B	True if INSERT PSW2 switch is set
KRUN/B	True if the COMPUTE RUN switch is set
KS0	True if DATA 0 switch is in the 1 position
KS1-31	Same KS0 for switch positions 1-31
KSC	True if CLOCK MODE switch is not in the CONT position
KSHI	True if REGISTER DISPLAY switch is in the ON position, CLOCK MODE switch is not in the CONT position, and REGISTER SELECT switch is in any of the following positions: AH, BH, CSHE, DH, SH
KSP15	True if SELECT ADDRESS switch 15 is in the 1 position
KSP16-31	Same as KSP15 for switch positions 16-31
KSS1	True if SENSE 1 switch is in the ON position
KSS2-4	Same as KSS1 for switch positions 2-4
KSTEP/B	True if COMPUTE STEP switch is set
KSTOR/B	True if either the STORE INSTR ADDR switch or the STORE SELECT ADDR switch is set
KSTORK/B	True if the STORE SELECT ADDR switch is set
KSTORQ/B	True if the STORE INSTR ADDR switch is set
KSXA	True if the REGISTER SELECT switch is either in the AL or AH position
KSXB	True if REGISTER SELECT switch is either in the BL or BH position
KSXCS	True if REGISTER SELECT switch is either in the CSL or CSHE position
KSXD	True if REGISTER SELECT switch is either in the DL or DH position
KSXP	True if REGISTER SELECTswitch is in the P position
KSXS	True if REGISTER SELECT switch is either in the SL or SH position
LB31/1	OR-gates a 1 into register address for doubleword operation
LBii	Where ii = 15-31. Address lines to memory
/LBii/	Where ii = 15-31. Core memory address lines to memory after all CPU logic

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
L/LK	Memory protection flip-flop address
LMii	Where ii = 15-31. Memory address lines into the map logic
LMXC	Transfer onto the address lines into the map logic from the C-register. C15-C22 → LM15-LM22, C23-C31 → LB23-LB31
LMXQ	Transfer onto the address lines into the map logic from the Q-register. Q15-Q22 → LM15-LM22, Q23-Q31 → LB23-LB31
LR31/2	Flip-flop to force 1 → LR31 (preset)
LRXD	Transfer onto the address lines for the registers from the D-register index bits. D12-D14 → LR29-LR31. Also resets LRXR
LRXLB	LB → LR (crossover)
/LRii/	Where ii = 23-31. The address lines for the registers in the fast memory
LRXR	Transfer onto the address lines for the registers from the flip-flops holding the R-field of the instruction. R28-R31 → LR28-LR31
MAA	Memory state flip-flop for no memory requests pending
MAP	If this signal is true, program addresses are transformed by the map
MAPDIS	Map disconnect. This flip-flo, inhibits mapping regardless of the state of MAPF
MAPF	Map flip-flop holding the mapping bit of program status word 1
MASTER	If this signal is true, the CPU operates in the master mode. MASTER will be true if either MASTERF or INTRAPF is high
MASTERF	Master flip-flop holding the master/slave bit of program status word 1
MBB	Memory state flip-flop for one memory request sent and address release not yet received
MBii	Where ii = 00-31. Memory data lines to and from CPU
MBXS/0	S0-S7 → /MB0-MB7/, 1 → /MW0/ special action if LRXLB is true
MBXS/1	S8-S15 → /MB8-MB15/, 1 → /MW1/ special action if LRXLB is true
MBXS/2	S16-S23 → /MB16-MB23/, 1 → /MW2/ special action if LRXLB is true
MBXS/3	S24-S31 → /MB24-MB31/, 1 → /MW3/ special action if LRXLB is true
MCC	Memory state flip-flop for one memory request sent and address release received
MCTE1	ES4 => E-1 ↛ E; NES4 => E-4 ↛ E; minus count exponent by one
MCTP1	Minus count the P-register (decrement), flip-flops P26-P31 minus one

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
MCTP2	Minus count control for P26-P29 (equals MCTP1 NP30 NP31)
MCTP5	Minus count the P-register (decrement). Only used to count P15-P18 down from 0000 to 1111
MCTR	Minus count the R-register, R-1 → R
MDD	Memory state flip-flop for remembering that a second memory request has been sent before the first request has received its data release
MEE	Memory state flip-flop for a satisfied first memory request, but a pending second request
MFF	Memory state flip-flop for one memory request sent and early data release received
MFLi	Where i = 0-7. Core memory fault lights
MGG	Memory state flip-flop for one memory request sent and data release received
MIT	Flip-flop high during multiply iterations
MITEX	Flip-flop exit from MIT (high during last clock)
MULC	Flip-flop, on during next to last clock if MIT in fixed multiply
MUSIC	If the alarm flip-flop is off, the music flip-flop drives the PCP speaker
MW	Write word to core memory MW => MBXS/0-3
MWB	Write byte to core memory according to P32 and P33 NP32 NP33 => MBXS/0 NP32 P33 => MBXS/1 P32 NP33 => MBXS/2 P32 P33 => MBXS/3
MWB0	Write to core memory byte 0
MWB1	Write to core memory byte 1
MWB2	Write to core memory byte 2
MWB3	Write to core memory byte 3

MWH	Write halfword to core memory according to P32
	NP32 => MBXS/0, MBXS/1
	P32 => MBXS/2, MBXS/3

MMC	Move to memory control, hexadecimal opcode 6F or EF (privileged)
MNN	Margins not normal (from PT16 logic power supplies)
MP19	Flip-flop represents P19 during multiply iterations
MPP	Multiply preparation

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
MQi	Where i = A, B, or C. Request for core memory access through port i
MR	Any CPU core memory request. MR through a cable driver becomes /MQC/
MRDL	Memory request delay line
MRTC	Memory request timing control
MRQ	Causes a request for memory service with the P-register specifying the address. Special action if P15- P22 = 0
(S/MRQ/1)	Sets MRQ and causes Q $\nrightarrow$ P. Use to initiate accessing of next instruction when set RQ is not allowed
MWH	Write to memory, halfword
MWiA	Where i = 0-3. Memory write byte lines
MWiB	Where i = 0-3. Memory write byte lines
MWiC	Where i = 0-3. Memory write byte lines
MWN	Flip-flop memory word negative
Oi	Where i = 1-7. The opcode register which remembers bits 1-7 of the instruction word
OLi	Where i = 0 through F. The decoding of the low order four bits of the opcode field, using the opcode register flip-flops O4, O5, O6, and O7
OPRQ	Dc flip-flop that is true during preparation for those instructions for which the memory request for an operand is made during preparation
ORIL	Override interleave
ORSP	Override slow sorts
ORXC	C0 $\nrightarrow$ IA, C1-C7 $\nrightarrow$ O1-O7, C8-C11 $\nrightarrow$ R28-R31
OUi	Where i = 0-7. The decoding of the high order (upper) three bits of the opcode field, using the opcode register flip-flops O1, O2, and O3
OVER	Overflow signal, true if A0 = D0 = 1 and K0 = 0 or if A0 = D0 = 0 and K0 = 1
OX/1	1 $\nrightarrow$ O1, 0 $\nrightarrow$ O2-O4, 1 $\nrightarrow$ O5, 0 $\nrightarrow$ O6-O7
OXC	Load the O-register from C. C1-C7 $\nrightarrow$ O1-O7 also load the indirect address flip-flop. C0 $\nrightarrow$ IA
P1929W	P19-29 contain ones
P2329W	P23-29 contain ones

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
P2629W	P26-29 contain ones
P2629Z	P26-29 contain zeros
P31Z	Force P31 = 0 if S $\nrightarrow$ P (used in double operand accessing for even location word)
PA22	Positive add with P22 instead of P20 as the least significant flip-flop
PA33	Positive add (or decrement) with P33 instead of P31 as the least significant flip-flop
PCB	Indicates that CPU is operating in map mode and not in master mode
PCP	True during PCP2-PCP6
PCPi	Where i = 1-7. Processor control panel phase flip-flops
PCTE1	E + 1 $\nrightarrow$ E
PCTP1	Plus count the P-register (increment): if NPA33, then P15-P31, plus one, if PA33, then P15-P33 plus one
PCTP2	Plus count control for P26-P29
PCTP2/1	PCTP1 and carry information from P30-P33
PCTP1DIS	PCTP1 disable, prevents incrementing of P-register
PCTP3	Plus count control for P23-P25 (equals PCTP2/1 P2629W)
PCTP4	Plus count control for P19-P22 (equals PCTP2/1 P2329W + PCTP4/1)
PCTP4/1	Plus count the P-register (increment): if NPA22, then P15-P20 plus one, if PA22, then P15-P22 plus one
PCTP5	Plus count control for P15-P19
PCTP5/3	Plus count the P-register flip-flops P15-P18
PCTR	R + 1 $\nrightarrow$ R
PE	Core memory parity error. /PEC/ through a cable receiver becomes PE
PEF1 and PEF2	Flip-flops used to generate 3-ns interrupt input signal when a parity error is signaled from memory
PEi	Where i = A, B, or C. Core memory parity error signal sent from port i
PEI	Parity error interrupt signal
PEL	Dc flip-flop set by parity error signal from memory
PFSR	Power fail-safe and reset signal to memory

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
PHA	Additional phase flip-flop used to generate 15 additional phases for complex instructions (that is, PH1, NPHA-PH15 NPHA, PH1 PHA-PH15 PHA). Used by stack and byte string instructions
PHi	Instruction execution phase flip-flops where i = 1-15
Pii	Where ii = 15-33. The P-register flip-flops for holding program address, and counting for various instructions
POF	Buffered power off
PON	Buffered power on
PREDO	Preparation for double operand instructions (AD, CD, LD, SD, CLM, LCD, LAD). Generated from the O-register
PREFADO	True for those instructions requiring two operands during preparation. Generated from the C-register
PREFLL	True for floating long instructions that require during preparation only, one operand which must be the low order 32 bits of a memory doubleword. Generated from the C-register
PREi	Preparation phase flip-flops where i = 1, 2, 3, or 4
PREIM	Preparation for some immediate instructions (LCFI, AI, LI, CBS, MBS, EBS) which go from PRE1 directly to PH1 during preparation. Generated from the O-register
PREOPRQ/1 and PREOPRQ/2	One of these signals will be true for all instructions requiring either one or two operands during preparation. Generated from the C-register
PREP	True during PRE1 through PRE4
PRERQ	True for those instructions for which RQ should be set during preparation
PRi	Where i = 0-32 or 47-71. Carry propagate term activated by an odd number of ones in its respective binary digit
PROTECTD	Protect or lock fail on data flip-flop to be sensed at data ready clock; sets trap unless specially prevented
PROTECTDIS	Protect disable. This signal inhibits PROTECTD from causing a trap
PROTECTI	Protect or lock fail on instruction flip-flop sensed at ENDE only
PRX	Carry propagate enable flip-flop for addition and exclusive OR functions
PSW1XS	S0-S3 -/→ CC1-CC4, S8 -/→ MASTERF, S9 -/→ MAPF, S10 -/→ DM, S11 -/→ AM
PSW2XD	D2-D3 -/→ WK0-WK1, D5 -/→ CIF, D6 -/→ II, D7 -/→ EI, D23-D27 -/→ RP23-RP27 Special action to WK0 and EI during RESET/B
PX	0 -/→ P15-P33
PX/1	0 -/→ P32-P33
PX/2	0 -/→ P15-P22

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
PX20	1 —/→ P26 for load
PXINT	INT0-INT8 —/→ P23-P31, 0 —/→ P15-P22, 0 —/→ P32-P33
PXK	KSP15-KSP31 —/→ P15-P31, 0 —/→ P32-P33
PXP3233	P32 and P33 —/→ P32 and P33 when PXS
PXQ	Q15-Q31 —/→ P15-P31, 0 —/→ P32-P33
PXS	S15-S30 —/→ P15-P30, S31 NP31Z —/→ P31, P32-P33 —/→ P32-P33
PXS/1	S15-S30 —/→ P15-P30, S31 NP31Z —/→ P31, S0-S1 —/→ P32-P33
PXTR	1 —/→ P25, TR28-TR31 —/→ P28-P31, 0 —/→ P15-P24, 0 —/→ P26-P27, 0 —/→ P32-P33
RATE/L	Signal that drives the instruction rate indicator on the free-standing console. Driven by PRE1
RESET	CPU RESET switch (normally open contact) if NKAS/B is true, or Start signal from the power monitor option
RESET/B	CPU RESET switch (normally closed contact) if NKAS/B is true
RESETCL	Reset signal for clock
Rii	Where ii = 28-31. The register address flip-flops holding the R-field from the instruction
RIDL	Memory request in delay line
RIP	Memory request in progress
RN	Flip-flop high when R is negative
W/RP0B0/0	Write control line for register page 0, byte 0, bit 0
W/RPaBb/c	Where a = 0-3, b = 0-3, and c = 0-31: similar to W/RP0B0/0
RPii	Register page flip-flops where ii equals 23-27
RPXD	D23-D27 —/→ RP23-RP27
RQ	RQ is set during certain instructions at the same time as the memory request for the last operand required by that instruction. If RQ is set when the address release signal associated with that operand access is received, another memory request is initiated with the Q-register specifying the address. Since the Q-register contains the address of the next instruction, RQ results in the next instruction being accessed
RQC	RQC causes a memory request during either PRE1 or PRE4. The C-register specifies the address
RRii	Where ii = 00-31. Register read (fast memory) data lines
/RRWii/	Where ii = 00-31. Register read and write (fast memory) data lines on bidirectional cable

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
RRWXS-0	S0-S7 → /RRW0/ through /RRW7/: byte 0
RRWXS-1	S8-S15 → /RRW8/ through /RRW15/: byte 1
RRWXS-2	S16-S23 →/RRW16/ through /RRW23/: byte 2
RRWXS-3	S24-S31 → /RRW24/ through /RRW31/: byte 3
RTZ	Flip-flop high when the result is 0
RUN/L	PCP RUN light
RW	Write word to fast memory RW => RWB0-RWB3
RWBi	Where i = 0-3. Control for RRWXS-i. Register write byte i
RWDIS	Register (fast memory) write disable. When RWDIS is true, RWB0-RWB3 do not cause a write into fast memory
RWii	Where ii = 00-31. Register write (fast memory) control lines
RWXS-0	S0-S7 → RW0-RW7: byte 0
RWXS-1	S8-S15 → RW8-RW15: byte 1
RWXS-2	S16-S23 → RW16-RW23: byte 2
RWXS-3	S24-S31 → RW24-RW31: byte 3
RX1	R28-R31 loaded by clock with ones
RZ	R = 0
QX26	Set Q = X'26'
QXP	P15-21 -/→ Q15-31
S00XK00	S00XK00 => 1 → S00X
S00XN	S00XN => N(A0 ⊕ D0) → S00X
S00XP	S00XP => (A0 ⊕ D0) → S00X
SC1	A flip-flop clocked by the 1.024 MHz clock used to detect the SINGLE STEP switch when single clocking
SC2	A flip-flop clocked by the 1.024 MHz clock. SC2 is the same as SC1 delayed by 1 μs
SCD	SCD is a buffer latch used during single clocking to disable CE after a clock has occurred as a result of setting the SINGLE STEP switch
SCEN	Single clock enable. When SCEN is false during single clocking, CE does not go false after the first clock, but stays true, allowing another clock to occur. Used when single clocking interrupt processing

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
SDIS	Select display. When SDIS is false, the register selected by the REGISTER SELECT switch is displayed in the DISPLAY REGISTER
SFTL	Shift left
SFTL1	Shift left one binary place
SFTL4	Shift left four binary places (one hexadecimal place)
SFTR	Shift right
SFTR2	Shift right two binary places
SHEX	Exit from shifting phase of shift instructions
SHPC	P control flip-flop used in shifts
SRA	Second memory request allowed signal from memory after CPU cable receiver
SRAF	Second memory request allowed flip-flop (actually buffer latch) set by SRA and reset by AR or RESETM
SRAi	Where i = A, B, or C. Second core memory access request allowed
ST	Start signal from the power monitor during power on and power off
START	Buffered ST signal
STRAP	A dc buffer latch that causes a trap to occur if WDTLATCH is set
SWi	Where i = 1-4. Four general purpose flip-flops for use during instruction execution
SXA	A0-A31 ⟶ S0-S31, A47-A71 ⟶ S47-S71
SXA/1	A ⟶ S0-S23
SXA/2	A ⟶ S24-S31
SXA/3	A47-A71 ⟶ S47-S71
SXADD	PR0 + K0 ⟶ S0, PR1 + K1 ⟶ S1, ... PR31 + K31 ⟶ S31, PR47 + K47 ⟶ S47, ... PR70 + K70 ⟶ S70, PR71 + K00 ⟶ S71
SXB	B0-B31 ⟶ S0-S31, B47-B71 ⟶ S47-S71, special action to B0 during FAMUL and FAFLM
SXCS	CS0-CS31 ⟶ S0-S31, CS47-CS71 ⟶ S47-S71
SXD	D0-D31 ⟶ S0-S31, D47-D71 ⟶ S47-S71
SXD/1	D24-D31 ⟶ S24-S31, D47 ⟶ S47
SXD/2	D16-D31 ⟶ S16-S31, D47 ⟶ S47

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
SXK	K → S
SXK/1	K → S0-S7
SXP	P15-P31 → S15-S31, P32-P33 → S0-S1
SXPR	PR0-PR31 → S0-S31, PR47-PR71 → S47-S71
SXPR/1	PR → S0-S7
SXUAB	K23-K30 → S0-S7, K23-K30 → S8-S15, K23-K30 → S16-S23, A24 + B24 → S24, A25 + B25 → S25, . . . , A31 + B31 → S31, B47 → S47. Load S with an upward aligned byte
SXUAH	K15-K30 → S0-S15, A16 + B16 → S16, A17 + B17 → S17, . . ., A31 + B31 → S31, B47 → S47. Load S with an upward aligned halfword
SXUAH/1	K15-K30 → S0-S15
T1L	T1 clock request line for 150-ns period clock for multiply inner loop
T10L	Replace T6L when addressing any private memory bank whose address is greater than three
T10LE	T10 clock request line enable
T1LEE	T1L even enable
T1LOE	T1L odd enable
T1LPE	T1L post enable; time interval between last T1L clock and next clock (T6L or T8L)
T4L	T4 clock request line for 280-ns period clock. Index add, straight forward operation without carry
T4RL	Sets T4L if RP contains 0, 1, 2, or 3; sets T10L otherwise
T6L	T6 clock request line for 320-ns period clock
T6RL	Sets T10L if PR does not contain 0, 1, 2, or 3 when reading out of fast memory
T8L	T8 clock request line for 380-ns period clock. Use for write to scratchpad when it would have been T4L, if it were S ↛ A; that is, four levels extra for writing to scratchpad. Also for controls with late inputs
TESTA and TESTA/1	TESTA NTESTA/1 => NA0 NA0031Z ↛ CC3 TESTA/1 => NK00H NA0031Z ↛ CC3 TESTA NTESTA/1 => A0 ↛ CC4 TESTA/1 => K00H NA0031Z ↛ CC4
TiL	Ti clock request line to clock generator where i = 1, 4, 6, 8, or 10
TP100	Timing point 100, 100 ns from last falling to next leading edge (CPU DL1/140)
TP140	Timing point 140, 140 ns from last falling to next leading edge (CPU DL1/180)

(Continued)

Table 3-154. Glossary of CPU Signals (Cont.)

Signal	Definition
TP520	Timing point 520 (DL2/260)
TP560	Timing point 560 (DL2/300)
TP700	Timing point 700 (DL2/140)
TP1140	Timing point 1140 (DL2/280)
TP1180	Timing point 1180 (DL2/020)
TP1300	Timing point 1300 (DL2/140)
TP1300E	TP1300 enable
TR28-TR31	Four flip-flops that are set when TRAP is set. TR28-TR31 + 64 is the trap address
TRACC1 - TRACC4	Four trap condition code flip-flops that are set when TRAP is set. TRACC1-TRACC4 are merged into the CC-register and are added to the P-register during the XPSD caused by a trap
TROVER	Trap on fixed point overflow when AM is high
WAIT/L	PCP WAIT light
WCTi	Where i = 1-6. Watchdog counter flip-flops
WDLATCH	Watchdog latch set by 40 µs count (WDTA) and reset by TRAP
WDTA	Watchdog timer activate, high when the watchdog counter has counted 40 ms
WDTR	Watchdog timer restart
WDTRAC	Watchdog timer restart and check for present count of 40 ms
WKi	Where i = 0 or 1. Write key flip-flops for protection against writing into memory
WRITE	WRITE is true whenever a write to core memory is taking place

Table 3-155. Glossary of Interrupt Signals

SIGNAL	DEFINITION
	Internal Interrupts
ADBDB	(Arm and disable) or (disable). A decode of the WD function code
NADB	Not arm and disable. A decode of the WD function code
ADDR05 through ADDR07	Function lines decoded from the CPU WD instruction
ADDR12 through ADDR15	Group lines decoded from the CPU WD instruction

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
NAE	Not arm and enable. A decode of the WD function code
AEADB	(Arm and enable) or (arm and disable)
AEENLE	(Arm and enable) or (enable) or (load enables)
AIB	Control flip-flop set by LEVACT, the leave-active-state signal from the CPU. This flip-flop, if set, remains true for one clock period. Part of the logic for turning off the arm flip-flop
AIEA	Control flip-flop set by ENTACT, the enter-active-state signal from the CPU. This flip-flop, if set, remains true for one clock period
AIEB	Control flip-flop set by AIEA. This flip-flop, if set, remains true for one clock period
ARE	Action response to the CPU. Acknowledges receipt of an enter-active-state signal or a leave-active-state signal
ARMCTR	Arm counter group signal. Sets arm flip-flop when counter group leaves active state
ARMIO	Arm input/output interrupt signal. Sets arm flip-flop when I/O group leaves active state
ARMOVD	Arm override interrupt signal. Sets arm flip-flop when override group leaves active state
BUS1	Priority chain busy signal
/BUS1/	Cable input signal to the counter group
BUS2	Priority chain busy signal
/BUS2/	Cable input signal to the I/O group
CHA0	Override group active signal
CHA1	Counter group active signal
CHA2	I/O group active signal
CI	Counter interrupt group inhibit. Bit position 37 of the WD PSD. If CI contains a one, counter interrupts are inhibited
CNA	Control flip-flop. Set by the enable WD mode signal EWDM through SCNA
CNB	Control flip-flop. Set by the true output of flip-flop CNA
CNLK	Service request interlock control flip-flop
CNLN7	Counter group requesting address line 7
CNLN8	Counter group requesting address line 8

(Continued)

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
CNRQ	Counter group service request
CNTZREQ	Counter zero request
CPUL1, CPUL2, CPUL3, and CPUL4	Counter-count-pulse 1, 2, 3, or 4 service request
CPUREST	CPU reset buffer. When this term is true, all IS and IP flip-flops are reset
/CPURST/	RESET-1 cable driver
NCPURST	RESET-1 inverter
DARM	Disarm. A decode of the WD function code. Reset term for IS and IP flip-flops
DAT16 through DAT31	Data signals transmitted to the external interrupts
DATA16 through DATA31	Buffers for data and control bits 16-31 of the WD interrupt control instruction. Inputs from the CPU to these buffers are /DIO16/ through /DIO30/
NDB	Not disable. A decode of the WD function code
/DIO37/ through /DIO39/	Direct I/O signals of the WD interrupt control instruction decoded to form function lines ADDR05, ADDR06, and ADDR07
/DIO44/ through /DIO47/	Direct I/O signals of the WD interrupt control instruction decoded to form group address lines ADDR12 through ADDR15, which become GRPADR00-GRPADR03
/DIO48/	Function strobe signal from the CPU
/DIO49/	Function strobe acknowledge signal sent to the CPU
EI	External interrupt group inhibit. Bit position 39 of the WD PSD. If EI contains a one, external interrupts are inhibited
ENCNTR	Enable counter group
ENEXSTR	(Enter strobe) or (exit strobe)
/ENEXSTR/	ENEXSTR driver
ENFNL	Enable function lines. This term goes true as the result of the EWDM mode signal having been true. See RCENFNL

(Continued)

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
ENIO	Enable I/O group
ENOVRD	Enable override group
ENTACT	Enter active state signal from the CPU
EWDM	Enable WD mode signal. This signal is true if FNL00, FNL01, or FNL02 is true
EWDM1 and EWDM2	Buffer amplifiers for EWDM
FNL00, FNL01, and FNL02	Function lines carrying bits from the WD control instruction for control of the IS, IN, and IP flip-flops
/FNL00/, /FNL01/, and /FNL02/	Input drivers for FNL00, FNL01, and FNL02
FS	Function strobe from the CPU
FSA	Function strobe acknowledge signal to the CPU
GARF	Action response flip-flop. One of the input terms to the ARE buffer
GATCLOCK	The 1-MHz clock gated by the AIEA control flip-flop.When AIEA is set, there is no clock pulse. GATCLOCK is the gated clock input to the interrupt level flip-flops IN, IS, and IP
GCLK1 and GCLK2	One megahertz gated clock signal. Same signal as GATCLOCK, but through different buffers
/GPADR0/ through /GPADR3/	Buffer cable drivers driven by GRPADR00 - GRPADR03
GRP0	Internal interrupt group select signal. This signal selects the group to be acted upon by the WD instruction. GRP0 includes all internal interrupts except PONN and POFF
GRPADR00 throgh GRPADR03	Group address lines from the WD control instruction. These lines specify which group of interrupts are acted upon
HBZ1	Higher-priority-group -busy signal received by counter group
HBZ2	Higher-priority-busy signal received by I/O group
HRQBZ1	Higher-priority-group-requesting-or-busy signal received by the counter group
HRQBZ2	Higher-priority-group-requesting-or-busy signal received by the I/O group
NHRQ1	An inverter with the input RQY1, counter group request signal

(Continued)

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
NHRQ2	An inverter with the input RQY2, the I/O group request signal
II	The I/O interrupt group inhibit. Bit position 38 of the WD PSD. If II contains a one, I/O interrupts are inhibited
IB0	Reset highest active level of the override group
IB1	Reset highest active level of the counter 0 group
IB2	Reset highest active level of the I/O group
IE0	Set highest waiting level to active for override group
IE1	Set highest waiting level to active for counter 0 group
IE2	Set highest waiting level to active for I/O group
IEN	Interrupt enable. An interrupt can be accepted on any clock if this flip-flop is true. Used to interrupt during multi-operand instructions
IN0 through IN15	Level enable flip-flops for a group of 16 interrupts. An IN flip-flop is one of three basic flip-flops for each interrupt level and is physically located on an LT16 module
INT	Set-reset flip-flop set by INT9. INT indicates that an interrupt request from the interrupt logic is present
INT0 through INT8 and /INT00/ through /INT08/	Interrupt buffer cable receivers for /INT00/ through /INT08/. These terms designate the interrupt address location in memory interrupt service routine address lines
INT9	Interrupt service request
/INT09/	Interrupt address request sent to the CPU
/INT11/	PCP interrupt light signal
/INT12/	One megacycle clock signal to the CPU
/INT13/	One kilocycle clock signal to the CPU
INT14 through INT26	Control signals from the CPU
INTRAP1	Interrupt (or trap) phase flip-flop
INTRAP2	Interrupt (or trap) phase flip-flop

(Continued)

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
INTRAPF	Set-reset flip-flop, which indicates an XPSD, MTW, MTH, or MTB is being executed as the result of an interrupt or trap
IOLN7	I/O interrupt group requesting line 7
IOLN8	I/O interrupt group requesting line 8
IOREQ	I/O interrupt group service request
IP0 through IP15	Level arm flip-flops for a group of 16 interrupts. An IP flip-flop is one of three basic flip-flops for each interrupt level and is physically located on an LT16 module
	Interrupt request from the IOP
IS through IS15	Level request flip-flops for a group of 16 interrupts. An IS flip-flop is one of three basic flip-flops for each interrupt level and is physically located on an LT16 module
NISIN0 through NISIN15	Priority control signals for a group of 16 interrupts. These terms are used for forming address lines to the CPU when an interrupt enters the waiting-enabled state. For example, if NISIN11 is true, interrupt level 11 is in the waiting-enabled state, and no higher priority flip-flops are in the active state or waiting-enabled state
NISNIP0 through NISNIP15	Priority control signals for a group of 16 interrupts. These terms are used to indicate the active state of higher priority interrupt levels. For example, if interrupt level 7 is active, NISNIP8 through NISNIP15 are true
KINTRP	Console switch signal for generating an interrupt request to memory location X'5D'
LEN	Load enables. Output term of an inverter with NLE input
LEVACT	Leave active state signal
LEVARM	Leave in armed state signal. If this signal is true, a level remains in the armed state after leaving the active state
LIN00 through LIN08	Interrupt service routine address line signals from the external interrupts
LINREQ	Interrupt service request to the CPU received from external interrupts
OF	Counter overflow flip-flop
OVLN6, OVLN7, OVLN8	Override interrupt group requesting address lines 6, 7, or 8
OVRQ	Override interrupt group service request

(Continued)

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
PEI	Parity error interrupt signal
PCBZC	Priority chain counter interrupt busy signal. CHA1 is the input signal to this buffer
PCBZ1	Priority chain I/O interrupt busy signal. CHA2 is the input signal to this buffer
PCBZO	Priority chain override interrupt busy signal. CHA0 is the input signal to this buffer
/PCBZC/	Buffer cable driver output signal indicating the counter interrupt priority chain is busy. Input to this buffer is PCBZC
/PCBZI/	Buffer cable driver output signal indicating the I/O interrupt priority chain is busy. Input to this buffer is PCBZI
/PCBZO/	Buffer cable driver output signal indicating the override interrupt priority chain is busy. Input to this buffer is PCBZO
PCRQBZC	Priority chain counter interrupt request for service or busy signal
PCRQBZI	Priority chain I/O interrupt request for service or busy signal
PCRQBZO	Priority chain override interrupt request for service or busy signal
/PCRQBZC/	Buffer cable driver output signal indicating a counter interrupt request for service or busy signal. Input to this buffer is PCRQBZC
/PCRQBZI/	Buffer cable driver output signal indicating an I/O interrupt request for service or busy signal. Input to this buffer is PCRQBZI
/PCRQBZO/	Buffer cable driver output signal indicating an override interrupt request for service or busy signal. Input to this buffer is PCRQBZO
POFF	Power-off signal to the IS0 flip-flop of the power fail-safe system
PON	Power-on signal to the IS1 flip-flop of the power fail-safe system
PXINT	Transfers the interrupt address, INT0-INT8, to the P-register, P23-P31
R0, R2, R4, R6, R8, R10, R12, R14	Part of the LT16 decode logic
NR01, NR23, NR45, NR67, NR89, NR911, NR1011, NR1213, NR1315, NR1357, NR1415	Part of the LT16 decode logic used to form interrupt subroutine addresses. For example, if NR01 is false, then interrupt level 0 or 1 is signalling for a request

(Continued)

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	Internal Interrupts
RCCHA0, RCCHA1, RCCHA2	Recirculate group 0, 1, or 2 active signal. These are the recirculate terms for the channel active signals CHA0, CHA1, CHA2
RCDAT26 RCDAT27	Recirculate terms for DAT26 and DAT27 dc latches
RCENFNL	Recirculate enable function lines. Recirculate term for the ENFNL dc latch
RCNB	Part of the reset term logic for the CNB flip-flop
REIP1, REIP3, REIP5, REIP7, REIP9, REIP11, REIP13, REIP15	Control signals which establish priority for interrupt levels in the waiting-enabled state
REN	Reset enable. (CPUREST1) or (load enable and GRP0). One of the reset logic terms for the IN flip-flops
RIJ	Reset term for OF, IJ8, IJ9, IJ10, and IJ11
RQY1	Higher priority request or busy signal received by the counter interrupts from the priority chain
RQY2	Higher priority request or busy signal received by the I/O interrupts from the priority chain
SCNA	Part of the set term logic for the CNA flip-flop
NSDARM	Not DARM
(S/INTRAPF)	This term initiates the interrupt sequence when all conditions exist for an interrupt
SR8, SR9, SR10, SR11, SR13	Part of the set term logic for request flip-flops IS8, IS9, IS10, IS11, and IS13
TNI00 through TNI08	Interrupt service routine address lines to the CPU. Input terms to /INT00/ through /INT08/
TNIREQ	Interrupt service request line to the CPU. Input term to /INT09/
TRIG	Trigger signal. Part of the logic used to set the IS request flip-flops and trigger an interrupt. Not used for power fail-safe
TRIG1	Buffer with TRIG input. Same as TRIG
WD	Write direct. Privileged instruction which can be used to control various states of the interrupts

Table 3-155. Glossary of Interrupt Signals (Cont.)

SIGNAL	DEFINITION
	External Interrupts
NR01, NR23, NR45, NR67, NR89, NR1011, NR1213, NR1415	Part of the LT16 decode logic used to form interrupt subroutine addresses. For example, if NR01 is false, the interrupt level 0 or 1 is signalling for a request
R0THR7	R0-R7. Request present from interrupt level 0, 1, 2, 3, 4, 5, 6, or 7
RCCHA	Recirculate term for the channel active signal dc latch
REIP1 through REIP15	Control signals which establish priority for interrupt levels in the waiting-enabled state
REN	Reset enable. One of the reset logic terms for the IN flip-flops
RLIN05 through RLIN07	Request signals for interrupt service routine lines, 5, 6, and 7
NRODD	Decode logic for odd numbered NR logic signals
RQY	Signal received by a lower priority group indicating that a higher priority group is requesting service
/RQY/	Driver input to RQY
SFNL00 through SFNL02	Function code bits gated to a specific chassis or group which has been selected to receive a WD instruction
SWTH0 through SWTH3	Select switches 0, 1, 2, or 3
TRIG	Trigger signal. Part of the logic used to set the IS request flip-flops and to trigger an interrupt
V8NJ	Special voltage signal

SECTION IV
INSTALLATION AND MAINTENANCE

4-1 INTRODUCTION

This section provides installation and maintenance data for the Sigma 7 computer. A basic Sigma 7 computer consists of a central processing unit (CPU), a multiplexing input/output processor (MIOP), and a 4K memory with optional features, devices, and device controllers for peripheral equipment added as applicable.

A set of documents provided with each Sigma 7 system includes an installation material list (IML), installation drawings, module location charts, assembly drawings, logic equations, and diagnostic programs (refer to table 4-1).

The Sigma 7 computer and peripheral equipment are installed by XDS personnel. To facilitate system expansion or relocation, the primary installation requirements for main units and optional features in the mainframe are included in this section.

4-2 INSTALLATION

4-3 BASIC CABINET INSTALLATION REQUIREMENTS

4-4 Floor Space

Space assignments for Sigma 7 computer cabinets must allow for the sweep of access doors and frames, as indicated on installation drawings. Space must also be allowed to permit access by maintenance personnel with test equipment while the frames are extended. A summary of space requirements is provided in table 4-2.

4-5 Power

The power requirements of each main unit, optional feature, and device controller are summarized in the related installation drawing. Power supplies are installed in each cabinet as indicated in figure 1-2. The primary reference sources

Table 4-1. Reference Documents for Sigma 7 Installation

Document	Identification Number	Contents
Installation Material List	Unique to each installation	All parts of a specific Sigma 7 installation, including layout drawing showing each major item of Sigma 7 computer and peripheral equipment. Point-to-point cabling instructions provided
Installation Drawing	134056	General information about Sigma 7 computer installation and cabling
Power Interconnection Diagram	139273	General information about power connections for Sigma 7 computer and peripheral equipment
Basic Cabinet		General information about cabinet for Sigma 7 computer
Assembly	131416	
Installation	131417	
Free Standing Console		General information about optional FSC
Assembly	127418	
Installation	127441	

Table 4-2. Space Requirements for Sigma 7 Cabinets

Equipment	Overall Dimensions	Clear Area
Basic cabinet	Width - 31 in	Front - 3 ft
	Depth - 29 in	Rear - 3 ft
	Height - 64 in	
Free standing console	Width - 60 in	Front - 4 ft
	Depth - 30 in	Rear - 3 ft
	Height - 41 in	

for power distribution are the IML and the Sigma system power interconnection installation drawing (refer to table 4-1).

4-6 Cable Connections

Cable connections consist of combinations of elements listed in table 4-3. The length of cable, number of connections, and restrictions are summarized in the installation drawings for each main unit, optional feature, and device controller.

4-7 MAIN UNIT INSTALLATION REQUIREMENTS

4-8 Central Processing Unit

The CPU occupies the forward and the center frame of one cabinet and part of the forward frame of a second cabinet. Preassigned locations in the CPU are reserved for modules for the following optional features: two additional real time clocks, power fail-safe, memory protection, memory map, private memory register extension and floating point arithmetic.

Table 4-3. Sigma 7 Cable Connections

Part No.	Function
129796	Interconnecting cable assembly. Interconnects component sides of modules
129797	Interconnecting cable assembly. Interconnects etched sides of modules
127314	Interconnecting cable assembly. Connects module etched side to component side
127315	Resistor connector assembly. Terminates cable chain at module
128047	Terminator. Terminates I/O priority cable chain.
115832	Blank cable terminal

The following engineering documents contain information required for installation of the CPU:

Document	Document No.
Assembly drawing	117307
Module location chart	127092
Installation drawing	117333
System installation drawing	134056

4-9 OPTIONAL FEATURE INSTALLATION REQUIREMENTS

4-10 Real-Time Clock

The real-time clock feature consists of two modules installed in the CPU cabinet as indicated in the CPU module location chart. Reference documents for the real-time clock feature are:

Document	Document No.
Assembly drawing	117616
Module location chart	127092
Installation drawing	117333

Two real-time clocks are included in the CPU; two optional real-time clocks may be added. Counters 1, 2, and 3 of the real-time clock feature are controlled by address selector module ST14 as illustrated in figure 4-1. Counter 4 is fixed at 400 Hz. Counters 3 and 4 are part of the CPU; counters 1 and 2 are optional.

4-11 Power Fail-Safe

The power fail-safe feature consists of one module installed in the CPU cabinet as indicated in the CPU module location chart. Reference documents for the power fail-safe feature are:

Document	Document No.
Assembly drawing	117612
Module location chart	127092
Installation drawing	117333

4-12 Memory Protection

The memory protection feature consists of five modules installed in the CPU cabinet as indicated in the CPU module location chart. Reference documents for the memory protection feature are:

Document	Document No.
Assembly drawing	117617
Module location chart	127092
Installation drawing	117333

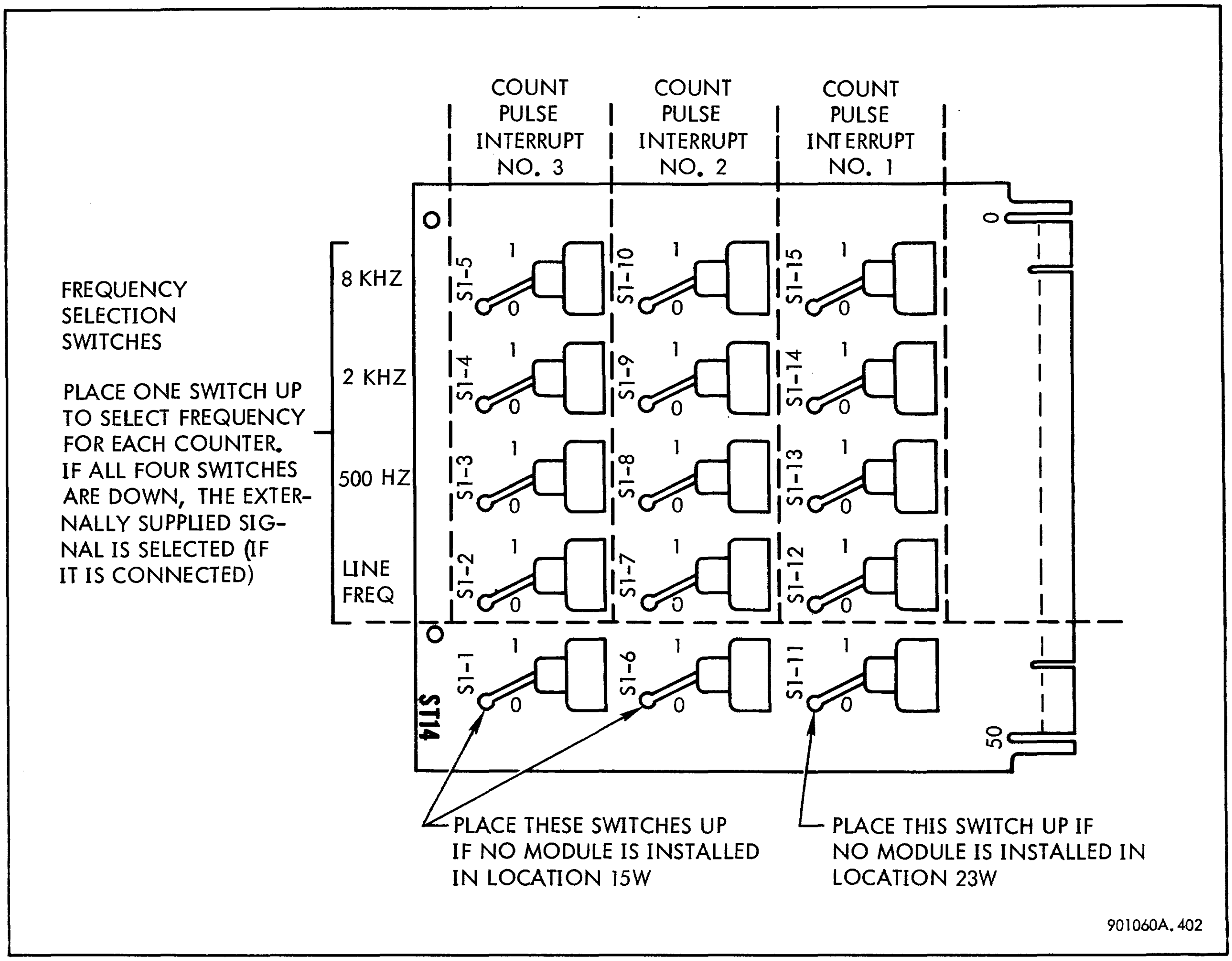
4-13 Memory Map

The memory map feature consists of 21 modules installed in the CPU cabinet as indicated in the CPU module location chart. Reference documents for the memory map feature are:

Document	Document No.
Assembly drawing	117615
Module location chart	127092
Installation drawing	117333

4-14 Private Memory Register Extension

Each private memory register extension consists of four fast-access memory 16 x 8-bit FT25 modules and additional modules as necessary. Four FT25 modules are included in the CPU; additional private memory registers may be added as an optional feature in banks of 16 registers (four FT25 modules). The first three banks of four FT25 modules may be added by inserting the modules in preassigned locations in the CPU. Additional banks are added by including a 32-module chassis in one of the available locations within a frame with the four FT25 modules and the additional modules which form the interface between the CPU and the private memory extension options. A maximum of 112 additional FT25 modules in seven 32-module chassis can be installed after the first 12 additional FT25 modules are inserted in the preassigned locations in the CPU.



PLACE ONE SWITCH UP TO SELECT FREQUENCY FOR EACH COUNTER. IF ALL FOUR SWITCHES ARE DOWN, THE EXTERNALLY SUPPLIED SIGNAL IS SELECTED (IF IT IS CONNECTED)

PLACE THESE SWITCHES UP IF NO MODULE IS INSTALLED IN LOCATION 15W

PLACE THIS SWITCH UP IF NO MODULE IS INSTALLED IN LOCATION 23W

Figure 4-1. Address Selector Module ST14

The following reference documents are required for installation of the private memory register extension:

Document	Document No.
Assembly drawing	117621
Module location chart	124819
Register extension unit assembly	130071
Register interface assembly	132208
Logic equations	124817
Installation drawing	124816
Module location chart (CPU)	127092
Installation drawing (System)	134056
Installation drawing (CPU)	117333
Modules list	129698

Interconnections between each register extension unit assembly and the CPU are indicated on Sigma 7 system installation drawing 134056. A switch comparator LT26 module is included on each register extension unit assembly to establish the address (see table 4-4 and figure 4-2).

Table 4-4. LT26 Switch Settings – Register Extension Address

Register Extension Unit Assembly	Switch Setting		
	S3-2	S3-1	S2-2
4 - 7	0	0	1
8 - 11	0	1	0
12 - 15	0	1	1
16 - 19	1	0	0
20 - 23	1	0	1
24 - 27	1	1	0
28 - 31	1	1	1

As each register extension unit assembly beyond the first is added, CPU jumper wires must be connected as follows:

Register Extension Unit Assembly	Jumper
1	None
2	28U03 to 29U17
3	28U07 to 29U19
4	28U10 to 29U21
5	28U14 to 29U23
6	28U18 to 29U25
7	28U22 to 29U27

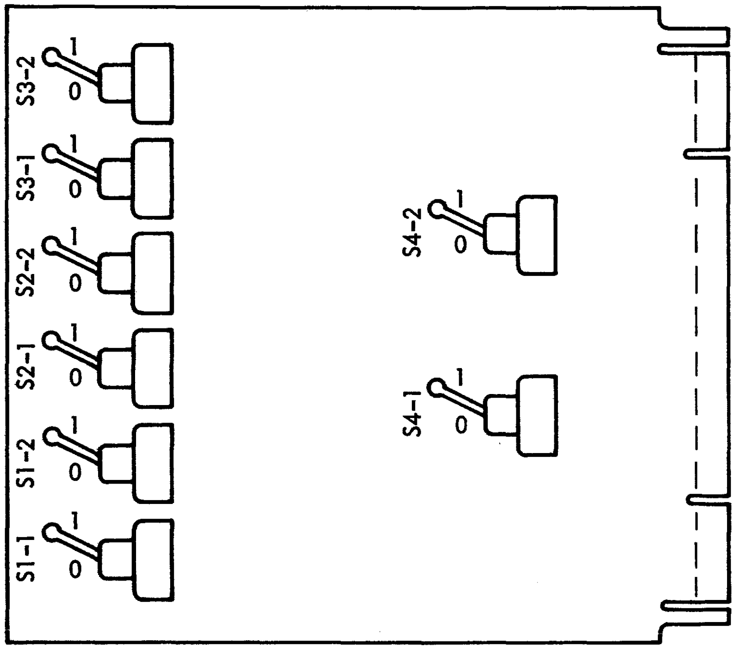


Figure 4-2. Switch Comparator LT26

4-15 Floating Point

The floating point feature consists of 41 modules installed in the CPU cabinet as indicated on the CPU module location chart. Reference documents for the floating point feature are:

Document	Document No.
Assembly drawing	117611
Module location chart	127092
Installation drawing	117333

4-16 Decimal Arithmetic

The decimal arithmetic feature consists of three 32-module chassis and the associated modules. The three chassis must be installed in one of the CPU cabinets. Additional decimal arithmetic feature modules are also placed in the CPU frames. Reference documents for the decimal arithmetic feature are:

Document	Document No.
Assembly drawing	117613
Module location chart	124823
Installation drawing	124820
Module location chart (CPU)	127092

4-17 External Interrupt Chassis

Each external interrupt chassis consists of a 32-module chassis and the associated modules. Each external interrupt chassis contains a maximum of 16 programmable interrupts and is installed within an available cabinet. The control elements of each external interrupt chassis consist of 13 modules; interrupt levels are added in increments of two by inserting up to eight LT16 priority interrupt modules. A maximum of 14 chassis, each containing 16 interrupt levels can be included to provide 224 interrupt levels. Reference documents for the external interrupt chassis are:

Document	Document No.
Assembly drawing	117330
Modules	129699
Modules	132206
Module location chart	129700
Installation drawing	124469
Logic equations	124470-001

4-18 Free-Standing Console

The free-standing console (FSC) contains three 32-module chassis with the associated modules in addition to power supplies, control panels, and indicator panels. The FSC is cable-connected to the Sigma 7 computer. Reference documents for the FSC are:

Document	Document No.
Assembly drawing	127418
Module location chart	127547
Installation drawing	127441
Logic equations	127546-001
Schematic diagrams	127439

4-19 MAINTENANCE

4-20 SPECIAL TOOLS AND TEST EQUIPMENT

Special tools and test equipment recommended for repair or maintenance of the Sigma 7 computer are listed in table 4-5.

Table 4-5. Special Tools and Test Equipment

Name	Manufacturer's Part No.	Manufacturer
P6010 IBM accessories and probe package	010-0186-00	Tektronix, Beaverton, Oregon
Oscilloscope	453	
Wirewrap tool	14XA2	Gardner-Denver, Grand Haven, Mich.
Wirewrap bit	502128	
Wirewrap sleeve	502129	
Wire unwrap tool	505084 (LH)	
Module extractor	126668	XDS, El Segundo, Calif.
Extender module ZT10	117037	
Solder sucker	(None)	

4-21 PREVENTIVE MAINTENANCE

Preventive maintenance of the Sigma 7 computer consists of scheduled diagnostic testing in addition to visual inspection and routine maintenance. Because there are no mechanical devices in the Sigma 7 computer, lubrication and mechanical adjustments are not required.

External surfaces of the Sigma 7 computer cabinets must be kept clean and dust-free. Doors and panels must close completely and must be in reasonable alignment. The topside of cabinets must remain cleared to permit fans to exhaust properly.

The interior of cabinets must be free of wire cuttings, dust, and other foreign matter. No clip leads or push-on jumpers should be in use during normal operation and all cables must be neatly dressed by clamps or routing. All chassis and frames must be properly bolted down with all hardware in place. Air filters should be checked for cleanliness and be replaced periodically.

4-22 DIAGNOSTIC TESTING

The following diagnostics are available for testing the performance of the Sigma 7 CPU and the related options. These diagnostics should be run whenever preventive maintenance is performed or when corrective maintenance is necessary. Information on loading and operating the diagnostics is included within each diagnostic program manual.

4-23 CPU Diagnostic System (SENSE)

This diagnostic locates malfunctions associated with the controls and the indicators on the Processor Control Panel (PCP). In addition, the proper operation of a basic instruction set (LPSD, XPSD, LW, STW, BCS, BCR, WAIT) is verified for use in sections of the more complex diagnostic programs. The program is written in three procedural steps. The first step checks the basic CPU controls and is entirely manual. The second step checks the controls used to enter and to retrieve data. The third step checks the operation of the Sense switches and verifies the operation of the basic instruction set.

Because of its simplicity, this diagnostic should not be run unless other CPU diagnostics fail. The first step of this diagnostic is time consuming and should not be run unless the operator is completely unfamiliar with the PCP switches.

Document No.	Description
900824	Sigma 7 CPU Diagnostic System (Sense), Diagnostic Program Manual
704041-83	Diagnostic program on absolute binary paper tape
704041-84	Diagnostic program on absolute binary cards

4-24 CPU Diagnostic System (VERIFY)

This diagnostic tests and diagnoses errors pertaining to the instructions LPSD, XPSD, LW, STW, BCS, WAIT, AND, EOR, and BIR. VERIFY assumes that all the functions tested by the SENSE diagnostic are operating correctly.

Document No.	Description
900870	Sigma 5 and 7 CPU Diagnostic Program (VERIFY), Diagnostic Program Manual
704042-83	Diagnostic program on absolute binary paper tape
704042-84	Diagnostic program on absolute binary cards

4-25 CPU Diagnostic System (PATTERN)

This diagnostic tests and diagnoses errors pertaining to the ability of the CPU to address and to access all available core memory, and private memory registers. The functions and operations tested by AUTO must be working before PATTERN is run.

Document No.	Description
900891	Sigma 7 CPU Diagnostic System (Pattern), Diagnostic Program Manual
704043-83	Diagnostic program on absolute binary paper tape
704043-84	Diagnostic program on absolute binary cards

4-26 CPU Diagnostic System (AUTO)

This diagnostic tests and diagnoses errors pertaining to all Sigma 7 instructions excluding decimal, floating point, byte string, stack, multiple, and convert instructions which are tested by other diagnostics. The functions and operations tested by VERIFY must be working before AUTO is run.

Document No.	Description
900872	Sigma 7 CPU Diagnostic-Auto, Diagnostic Program Manual
704044-83	Diagnostic program on absolute binary paper tape
704044-84	Diagnostic program on absolute binary cards

4-27 CPU Diagnostic System (SUFFIX)

The diagnostic tests and diagnoses errors pertaining to the instructions CBS, MBS, TBS, TTBS, PSW, PSM, PLW, PLM, MSP, STM, LM, CVA, and CVS.

Document No.	Description
900893	Sigma 7 CPU Diagnostic (Suffix), Diagnostic Program Manual
704045-83	Diagnostic program on absolute binary paper tape
704045-84	Diagnostic program on absolute binary cards

4-28 CPU Diagnostic System (FLOAT)

This diagnostic tests and diagnoses errors pertaining to all Sigma 7 floating point instructions excluding Floating Shift (SF). Successful operation of AUTO is a prerequisite to the running of FLOAT.

Document No.	Description
900898	Sigma 5 and 7, CPU Diagnostic-Float, Diagnostic Program Manual
704046-83	Diagnostic program on absolute binary paper tape
704046-84	Diagnostic program on absolute binary cards

4-29 CPU Diagnostic System (MAP)

This diagnostic tests and diagnoses errors pertaining to the optional memory map feature, including the instruction MMC.

Document No.	Description
900920	Sigma 7 MAP Diagnostic Program, Diagnostic Program Manual
704048-83	Diagnostic program on absolute binary paper tape
704048-84	Diagnostic program on absolute binary cards

4-30 Sigma 5 and 7 Real-Time Clock Test

This diagnostic tests and diagnoses errors pertaining to the operation of the real-time clocks by computing the time of day or elapsed time using the real-time clocks.

Document No.	Description
901136	Sigma 5 and 7 Real-Time Clock Test, Diagnostic Program Manual
704017-83	Diagnostic program on absolute binary paper tape
704017-84	Diagnostic program on absolute binary cards

4-31 Sigma 5 and 7 Memory Protect Diagnostic

This diagnostic tests and diagnoses errors pertaining to the optional memory protect feature by checking the operation of the Write Locks and Keys.

Document No.	Description
901516	Sigma 5 and 7 CPU Diagnostic Program (Memory Protect), Diagnostic Program Manual
704062-83	Diagnostic program on absolute binary paper tape
704062-84	Diagnostic program on absolute binary cards

4-32 Sigma 5 and 7 Power Fail-Safe Test

This diagnostic tests and diagnoses errors pertaining to the optional Power Fail-Safe feature by checking the operation of the power-on and power-off interrupts.

Document No.	Description
901135	Sigma 5 and 7 Power Fail-Safe Test, Diagnostic Program Manual
704122-83	Diagnostic program on absolute binary paper tape
704122-84	Diagnostic program on absolute binary cards

4-33 Sigma 5 and 7 Interrupt Test

This diagnostic tests and diagnoses errors pertaining to the interrupt system.

Document No.	Description
901134	Sigma 5 and 7 Interrupt Test, Diagnostic Program Manual
704143-83	Diagnostic program on absolute binary paper tape
704143-84	Diagnostic program on absolute binary cards

4-34 Sigma 5 and 7 Systems Test Monitor (SEVA)

This diagnostic tests and diagnoses errors pertaining to peripheral devices. The monitor program (SEVA) is loaded with individual peripheral diagnostic routines, and the program attempts to verify the operation of a number of peripherals running concurrently. The object of the program is not to isolate malfunctions within a peripheral, but rather to isolate malfunctions within the system.

Document No.	Description
901076	Sigma 5 and 7 Systems Test Monitor, Diagnostic Program Manual
704138-83	Diagnostic program on absolute binary paper tape
704138-84	Diagnostic program on absolute binary cards

Refer to the individual peripheral technical manuals for information on the individual peripheral diagnostic routines used with this monitor program.

4-35 CORRECTIVE MAINTENANCE

The following paragraphs contain information relating to some of the more common procedures encountered in troubleshooting the CPU.

4-36 Power

An electronic circuit breaker exists in each PT15 Power Supply located at the top rear of most cabinets. The 115 Vac, 2kHz output can be checked at any of the 3-prong receptacles on the side of the PT15.

A circuit breaker and three fuses are located on each PT16 Power Supply mounted on the hinged side of each frame. The PT16 plugs into a junction box on the bottom of the cabinet. The junction box is plugged into the PT15.

If a free-standing console is in the system, its power switch is in series with the power switch on the PCP. Both must be on to power-up the system.

Inability to operate the switches on the PCP may be caused by the power monitor signal ST always being true. Check its input to the CPU logic at frame 1, as noted in the logic equations.

4-37 Clocks

Pressing the CPU RESET switch with the CLOCK MODE switch in CONTINUOUS and the COMPUTE switch in IDLE should initialize and start the clock generation logic. All of the PCP phases use only the T6L clock. Other clock intervals can be checked by executing the instructions using those clock intervals as noted in the phase sequence charts. There are no clock margins other than the secondary effects, which occur when the three-position voltage margin switches on each of the PT16 Power Supplies are manipulated.

Single clock logic can be checked by moving the CLOCK MODE switch from CONTINUOUS to its center (unlabeled) position, which inhibits all clocks. With an instruction in the 32 DISPLAY indicators, and the COMPUTE switch in the RUN position, a subsequent depression of the CLOCK MODE switch to the SINGLE STEP position produces one clock and should cause the PHASE indicators to advance, depending upon the instruction.

If single clocking through a PCP phase, the appropriate PCP switch must be manually held in position and the COMPUTE switch must be in IDLE.

If single clocking through an interrupt sequence, two clocks rather than one are produced in INTRAP1 and in the phase used to clear the interrupt level during the execution of either an LPSD or Modify and Test instruction.

4-38 Memory

Memory operation can be quickly checked by using the STORE and the DISPLAY switches on the PCP using both core memory and private memory register addresses. These PCP functions use much of the memory request logic within the CPU. These functions can be repeated easily by grounding the set input to the HALT flip-flop at location 20A50, and by holding (or taping) the PCP switch in position.

If the memory request state flip-flops (MAA through MGG) seem to be out of step, it may first show up in the operation of the DISPLAY SELECT ADDRESS function. The first time that the switches are operated, the contents of the current instruction address is displayed instead of the select address. The second time that the switches are operated, the correct information is displayed.

4-39 Instructions

Instructions may be executed one at a time directly from the control panel, or a short program may be stored and executed from either core memory or the private memory registers.

To iterate any nonbranching instruction at its maximum rate, use the INSTRUCTION ADDRESS switch in the HOLD position. The instruction to be iterated (or held) must reside in memory. Use the DISPLAY switch to access it from memory and to put it into the DISPLAY indicators, then place the INSTRUCTION ADDRESS switch in the HOLD position and the COMPUTE switch to RUN.

If an instruction hangs up in a phase, place an XPSD instruction in the trap location X'46' with a new instruction address equal to the address of the instruction which is hanging up. In this way, the watchdog timer can be used automatically to cause the computer to return to that instruction for scoping purposes.

If an instruction fails during the running of the diagnostic, it is sometimes useful to place the diagnostic in a short loop using the sense switch options called out in the diagnostic program manual, and to set the SELECT ADDRESS switches to the location of the Execute instruction within the driver portion of the diagnostic. In this manner a sync pulse is generated whenever the Execute instruction is accessed. This sync pulse is the logic term ADMATCH, which can be found in frame 3 of the CPU logic equations.

If problems are being experienced in loading programs, the Bootstrap program, inserted into core memory through the LOAD switch, can be easily checked by inserting an existing teletype unit address number into the UNIT ADDRESS switches, and by attempting to load from the teletype. The Bootstrap program can also be changed quite easily to perform an output operation by changing the structure of the I/O Command Doubleword.

4-40 Replacement of Miniature Incandescent Lamps (DS1 through DS21 and DS29 through DS98) (See figure 4-3).

When a miniature lamp of the soldered lead type (P/N 123710) used for indicators DS1 through DS29 and DS29 through DS98 becomes defective, it shall be replaced with a new plug-in type lamp (P/N 108814-009). Installation of each new lamp requires the addition of one new lamp holder (P/N 180223) and one new O-ring (P/N 111344-001).

Replace the defective soldered lead type lamp with the new plug-in type lamp as follows:

a. Cut leads of defective lamp as close as possible to base of lamp. Remove lamp by releasing mounting spring clip. Keep clip to install new lamp. Discard defective lamp.

CAUTION

Do not allow physical shocks to occur to lamp during installation.

b. Insert new type lamp through mounting hole in front of panel until lamp shoulder butts against front panel surface.

c. Install O-ring over terminal end of lamp module and slide O-ring forward until it is flush with back surface of panel.

d. Using long nose pliers, open spring clip by gently squeezing two opposing tabs together. Slide spring clip over terminal end of lamp module and move clip forward until its legs fit over O-ring and contact panel. Lock spring clip in position by releasing tabs.

e. Install lampholder over lamp module by inserting lamp terminals in matching terminals of lampholder and then gently pushing lampholder forward.

f. Using long nose pliers, separately connect ends of two lamp leads to two lampholder terminals and solder connections.

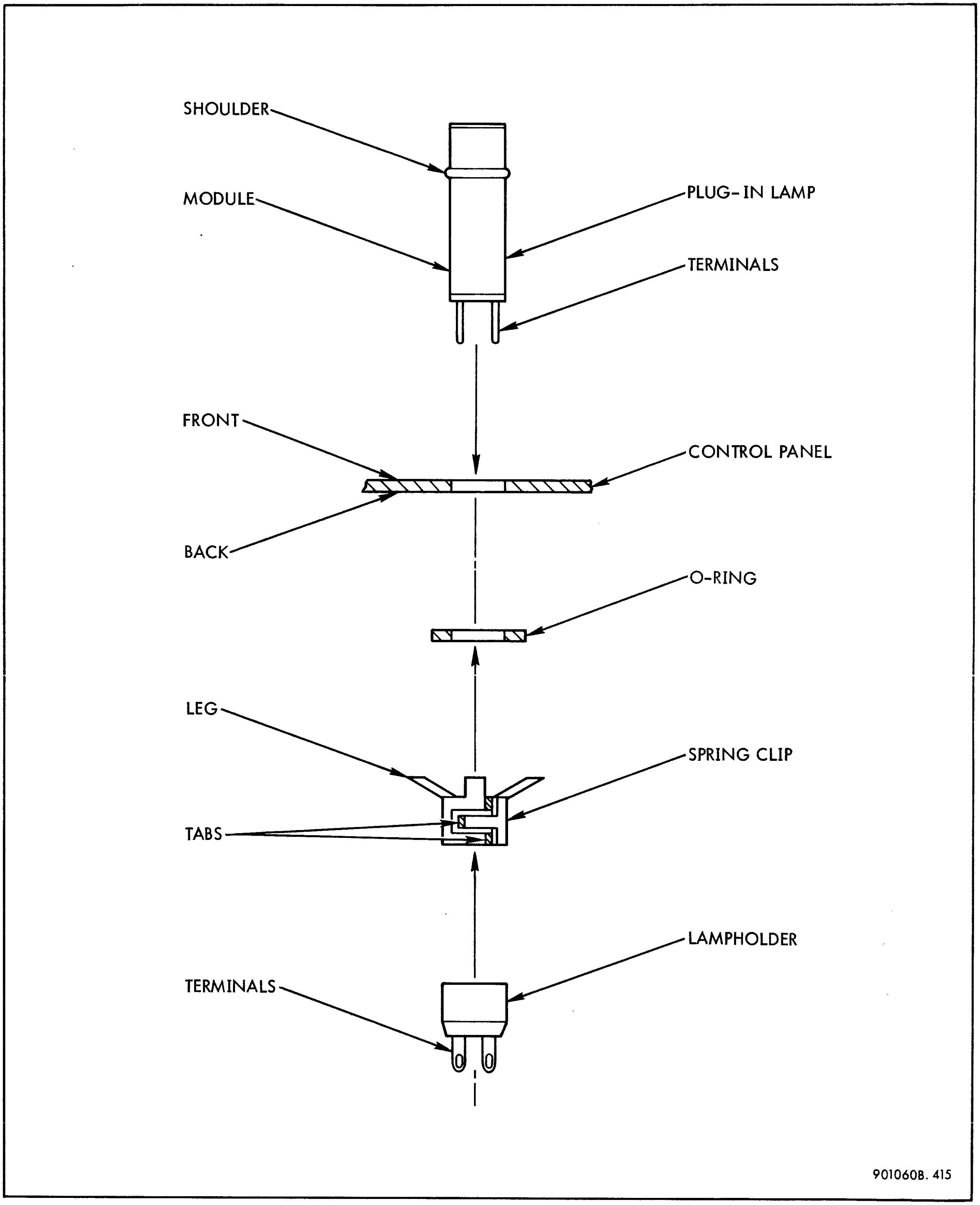


Figure 4-3. Plug-In Lamp Assembly Installation

APPENDIX A
PARTS LIST

A-1 INTRODUCTION

Appendix A lists and illustrates replaceable parts of the basic Sigma 7 CPU as well as its optional features, but does not include the Sigma 7 peripheral equipment or device controllers. Parts below the level of XDS circuit modules are not listed.

A-2 TABULAR LISTING

Appendix A is arranged in parts list of tables, starting with a listing of the main assemblies of the basic Sigma 7 Central Processing Unit Model 8401. Breakdown of the Sigma 7 CPU optional equipment by tables follows.

A-3 ILLUSTRATIONS

In some cases, the parts tables refer to accompanying illustrations to indicate the parts described in the table and to show their locations in the assembly.

A-4 REPLACEABLE PARTS

Replaceable parts are defined as parts having maintenance significance, that is, parts which can be replaced because of wear, damage, or for malfunctions caused by normal use of the equipment. Standard hardware, such as nuts, screws and washers, parts that are normally supplied in bulk, such as wire and tubing, and structural parts, such as panels and brackets, are not considered replaceable parts. In some instances, however, the parts list tables do list such nonreplaceable parts for reference purposes only.

4-5 PARTS LIST TABLES

Each parts list table is arranged in six columns as follows:

a. Figure and index number of the listed part, where applicable

b. Brief description of the part

c. Reference designator of the part, if applicable

d. Manufacturer code number

e. Manufacturer's part number

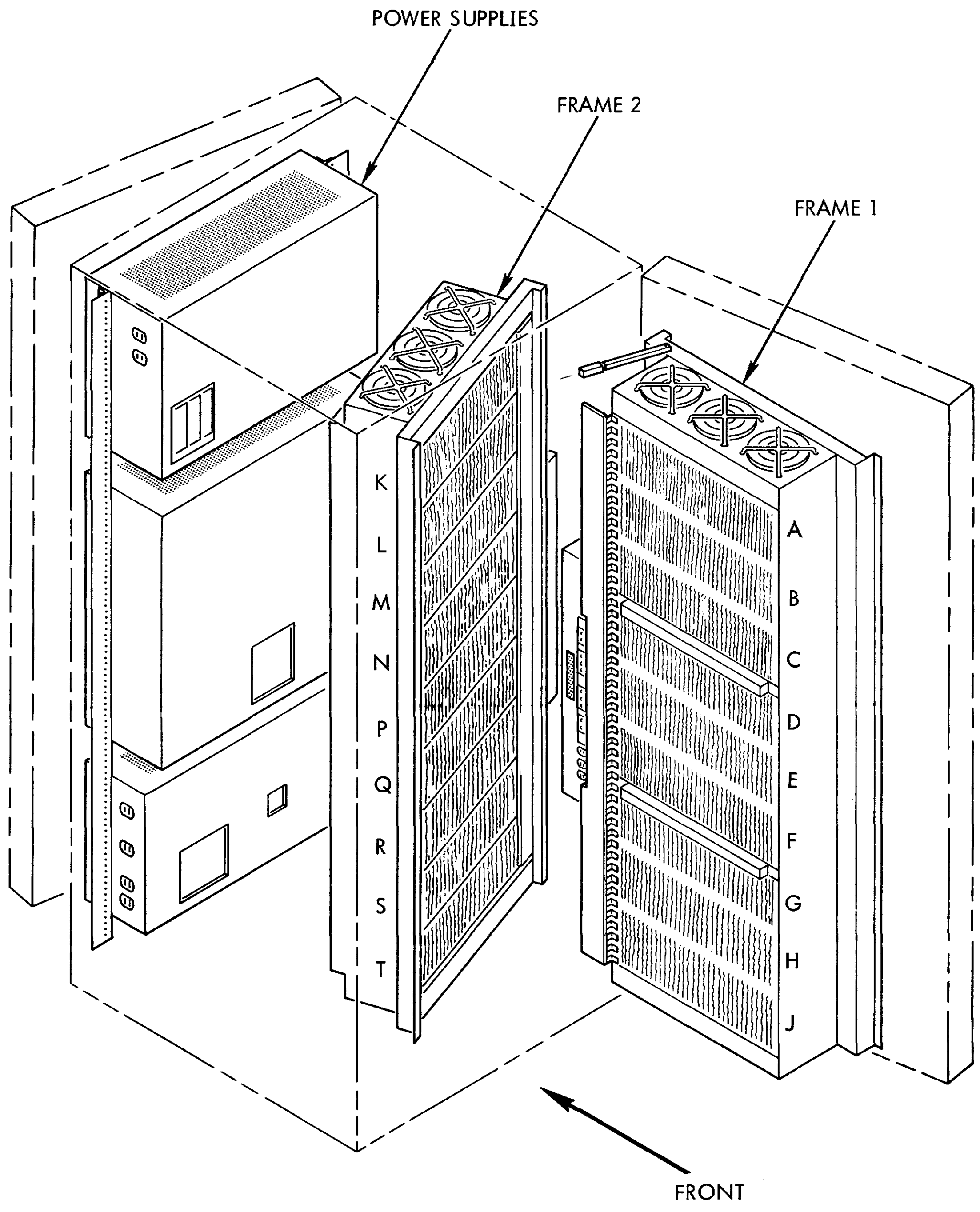
f. Quantity of the part used for each assembly

A-6 MANUFACTURERS CODE INDEX

Table 4-30 contains the names and addresses of manufacturers identified by code number given in the previous tables.

Table A-1. Sigma 7 Central Processing Unit Model 8401, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Central Processing Unit, Sigma 7		XDS	117307	1
	. Basic cabinet, Model 8900 (see table A-29 for breakdown)		XDS	131416	3
	. Power Supply Model PT14		XDS	117262	2
	. Power Supply Model PT15		XDS	117263	2
	. Power Distribution Box		XDS	117428	2
	. Quartz crystal, 2.048 MHz		XDS	128131-017	1
	. Frame No. 1, CPU cabinet No. 1 (see table A-2 for breakdown)		XDS	117309	1
	. Frame No. 2, CPU cabinet No. 1 (see table A-6 for breakdown)		XDS	117308	1
	. Frame No. 1, CPU cabinet No. 2 (see table A-10 for breakdown)		XDS	126601	1



CPU CABINET

901060A. 600

Figure A-1. CPU Cabinet No. 1

Table A-2. Frame No. 1, CPU Cabinet No. 1, Replaceable Parts

Fig. & Index No.	Description	Reference Desginator	Manufacturer	Part No.	Qty
A-1	Assembly, Frame No. 1, CPU Cabinet No. 1 (see table A-1 for next higher assembly)		XDS	117309	1
	. Power Supply PT16		XDS	117264	1
	. Cables, busbar pickup		XDS	131891-005	2
	. Cables, 33 ohm, single condition		XDS	128147-372	5
	. Cable, intraframe		XDS	128060-171	19
	. Cable, intraframe		XDS	128060-292	1
	. Cable, intraframe		XDS	128060-302	1
	. Cable, intraframe		XDS	128060-182	1
	. Chassis No. 1 (Logic), cabinet No. 1		XDS	117313	1
	. . Assembly, top fan (see table A-29 for breakdown)		XDS	123943	1
	. . Assembly, bottom fan (see table A-29 for breakdown)		XDS	117320	1
	. . Assembly, wired board A, B, C (see table A-3 for breakdown)		XDS	124729	1
	. . Assembly, wired board D, E, F (see table A-4 for breakdown)		XDS	124727	1
	. . Assembly, wired board G, H, J (see table A-5 for breakdown)		XDS	124724	1

Table A-3. Wired Board Assembly (A, B, C), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, wired board A, B, C (see table A-2 for next higher assembly)		XDS	124729	Ref
	. Cable Driver AT21		XDS	127797	1
	. Buffered AND/OR Gate BT10		XDS	116056	3
	. BAND Gate BT11		XDS	116029	3
	. Gated Buffer No. 1 BT16		XDS	125262	3
	. BAND Gate BT18		XDS	126613	2
	. Universal Flip-Flop FT22		XDS	124713	6
	. Gate Expander No. 2 GT11		XDS	124881	5
	. Gated Inverter IT16		XDS	125264	7
	. Gated Inverter IT20		XDS	126747	7
	. NAND/NOR Gate IT24		XDS	128188	1
	. NAND Gate IT25		XDS	128190	6
	. NAND Gate IT26		XDS	128192	7
	. Buffer Inverter LT13		XDS	123016	1
	. Buffer Inverter LT14		XDS	123017	12
	. Logic Element LT20		XDS	124717	3
	. Logic Element LT21		XDS	126615	2
	. Terminator XT10		XDS	116257	8
	. Cable Plug, ZT23		XDS	128164	1

Table A-4. Wired Board Assembly (D, E, F), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, wired board D, E, F (see table A-2 for next higher assembly)		XDS	124727	Ref
	. Cable Driver/Receiver AT11		XDS	123019	1
	. Buffer AND/OR Gate BT10		XDS	116056	1
	. BAND Gate BT11		XDS	116029	5
	. Gated Buffer No. 1 BT16		XDS	125262	2
	. Gated Buffer BT17		XDS	126330	1
	. Universal Flip-Flop FT22		XDS	124713	5
	. Gate Expander No. 1 GT10		XDS	124715	1
	. Gated Inverter No. 2 GT11		XDS	124881	2
	. Gated Inverter IT16		XDS	125264	7
	. Gated Inverter IT20		XDS	126747	7
	. NAND Gate IT25		XDS	128190	6
	. NAND Gate IT26		XDS	128192	4
	. Buffer Inverter LT13		XDS	123016	6
	. Buffer Inverter LT14		XDS	123017	6
	. Logic Element LT20		XDS	124717	5
	. Logic Element LT21		XDS	126615	2
	. Terminator XT10		XDS	116257	6
	. Cable Plug ZT23		XDS	128164	1

Table A-5. Wired Board Assembly (G, H, J), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Wired Board G, H, J (see table A-2 for next higher assembly)		XDS	124724	Ref
	. Buffered AND/OR Gate BT10		XDS	116056	4
	. BAND Gate BT11		XDS	116029	16
	. Gated Buffer No. 1 BT16		XDS	125262	2
	. Gated Inverter IT16		XDS	125264	8
	. Gated Inverter IT20		XDS	126747	12
	. NAND/NOR Gate IT24		XDS	128188	3
	. NAND Gate IT25		XDS	128190	4
	. NAND Gate IT26		XDS	128192	1
	. Buffer Inverter LT14		XDS	123017	3
	. Logic Element LT20		XDS	124717	10
	. Logic Element LT21		XDS	126615	6
	. Terminator XT10		XDS	116257	6

Table A-6. Frame No. 2, CPU Cabinet No. 1, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Frame No. 2, CPU Cabinet No. 1 (see table A-1 for next higher assembly)		XDS	117308	1
	. Power Supply PT16		XDS	117264	1
	. Cable, busbar pickup		XDS	131891-005	7
	. Cable, 33 ohm, single condition		XDS	128147-372	27
	. Cable, intraframe		XDS	128060-182	1
	. Cable, intraframe		XDS	128060-171	14
	. Chassis No. 2 (Logic), cabinet No. 1		XDS	117314	1
	. . Assembly, top fan (see table A-29 for breakdown)		XDS	123943	1
	. . Assembly, bottom fan (see table A-29 for breakdown)		XDS	117324	1
	. . Assembly, wired board K, L, M (see table A-7 for breakdown)		XDS	124728	1
	. . Assembly, wired board N, P, Q (see table A-8 for breakdown)		XDS	124730	1
	. . Assembly, wired board R, S, T (see table A-9 for breakdown)		XDS	124736	1

Table A-7. Wired Board Assembly (K, L, M), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Wired Board K, L, M (see table A-6 for next higher assembly)		XDS	124728	Ref
	. Cable Driver/Receiver AT11		XDS	123019	2
	. Buffer AND/OR Gate AT21		XDS	127797	2
	. Gated Buffer No. 1 BT16		XDS	125262	2
	. Gated Buffer BT17		XDS	126330	5
	. BAND Gate BT18		XDS	126613	2
	. Register Flip-Flop FT17		XDS	124628	12
	. Counter Flip-Flop FT18		XDS	124634	5
	. Universal Flip-Flop FT22		XDS	124713	6
	. Fast Access Memory FT25		XDS	126743	4
	. Gate Expander No. 1 GT10		XDS	124715	1
	. Gate Expander No. 2 GT11		XDS	124881	2
	. Gated Inverter IT16		XDS	125264	6
	. NAND Gate IT26		XDS	128192	3
	. Buffer Inverter LT13		XDS	123016	1
	. Adder No. 1 LT17		XDS	123554	4
	. Carry No. 1 LT18		XDS	123590	3
	. Logic Element LT21		XDS	126615	2
	. Terminator XT10		XDS	116257	8
	. Clock Terminator No. 1 XT18*		XDS	132009	4

*If floating point is not installed

Table A-8. Wired Board Assembly (N, P, Q), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Wired Board N, P, Q (see table A-6 for next higher assembly)		XDS	124730	Ref
	. Cable Driver/Receiver AT11		XDS	123019	6
	. Buffered AND/OR Gate AT21		XDS	127797	2
	. Gated Buffer No. 1 BT16		XDS	125262	6
	. Gated Buffer BT17		XDS	126330	3
	. Delay Element DT16		XDS	128172	1
	. Register Flip-Flop FT17		XDS	124628	8
	. Counter Flip-Flop FT18		XDS	124634	1
	. Universal Flip-Flop FT22		XDS	124713	4
	. Fast Access Memory FT25		XDS	126743	1
	. Gate Expander No. 1 GT10		XDS	124715	3
	. Gate Expander No. 2 GT11		XDS	124881	3
	. Gated Inverter IT16		XDS	125264	7
	. Gated Inverter IT17		XDS	126331	2
	. NAND Gate IT26		XDS	128192	1
	. Buffer Inverter LT13		XDS	123016	2
	. Adder No. 1 LT17		XDS	123554	5
	. Carry No. 1 LT18		XDS	123590	2
	. Logic Element LT20		XDS	124717	1
	. Logic Element LT21		XDS	126615	3
	. Terminator XT10		XDS	116257	9
	. Cable Plug ZT23		XDS	128164	1

Table A-9. Wired Board Assembly (R, S, T), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Wired Board R, S, T (see table A-6 for next higher assembly)		XDS	124736	Ref
	. Cable Driver/Receiver AT11		XDS	123019	2
	. Cable Driver AT21		XDS	127797	2
	. Gated Buffer No. 1 BT16		XDS	125262	5
	. Gated Buffer BT17		XDS	126330	1
	. BAND Gate BT18		XDS	126613	3
	. Register Flip-Flop FT17		XDS	124628	12
	. Counter Flip-Flop FT18		XDS	124634	2
	. Universal Flip-Flop FT22		XDS	124713	4
	. Fast Access Memory FT25		XDS	126743	2
	. Gate Expander No. 2 GT11		XDS	124881	4
	. Gated Inverter IT16		XDS	125264	5
	. NAND Gate IT25		XDS	128190	1
	. Adder No. 1 LT17		XDS	123554	8
	. Carry No. 1 LT18		XDS	123590	2
	. Terminator XT10		XDS	116257	5
	. Clock Terminator No. 1 XT18*		XDS	132009	2

*If floating point is not installed

Table A-10. Frame No. 1, CPU Cabinet No. 2, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Frame No. 1, CPU Cabinet No. 2 (see table A-1 for next higher assembly)		XDS	126601	1
	. Cable, 33 ohm, single condition		XDS	128147-372	6
	. Cable, intraframe		XDS	137481-171	1
	. Power Supply PT16		XDS	117264	1
	. Assembly, CPU Chassis No. 3		XDS	126593	1
	. . Assembly, top fan (see table A-29 for breakdown)		XDS	123943	1
	. . Assembly, bottom fan (see table A-29 for breakdown)		XDS	117320	1
	. . Blank Panel Simulator		XDS	129694	5
	. . Assembly, wired board U, V, W (see table A-11 for breakdown)		XDS	126594	1
	. . Assembly, wired board Y (see table A-12 for breakdown)		XDS	126596	1

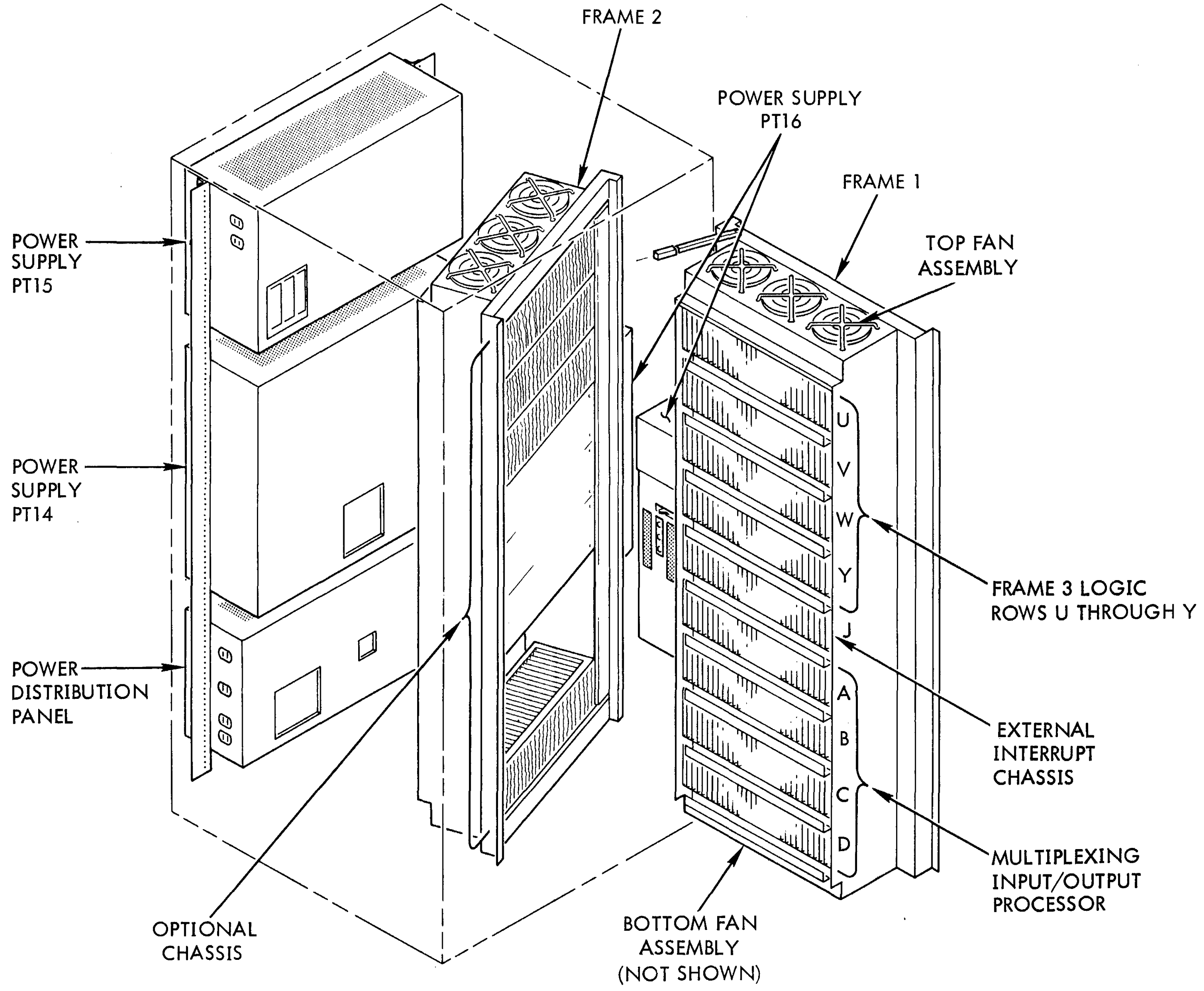


Figure A-2. CPU Cabinet No. 2 (typical)

Table A-11. Wired Board Assembly (U, V, W), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Wired Board U, V, W (see table A-10 for next higher assembly)		XDS	126594	Ref
	. Cable Receiver AT19		XDS	123018	1
	. Cable Driver/Receiver AT11		XDS	123019	2
	. Cable Driver AT12		XDS	124629	2
	. Cable Driver/Receiver AT13		XDS	125260	2
	. Clock Driver No. 1 AT23		XDS	128166	4
	. BAND Gate BT11		XDS	116029	4
	. Gated Buffer No. 1 BT16		XDS	125262	3
	. Gated Buffer BT17		XDS	126330	4
	. Medium Frequency Oscillator CT16		XDS	133694	1
	. Delay Line DT14		XDS	127319	4
	. Delay Line Expander DT15		XDS	127883	1
	. Delay Element DT16		XDS	128172	1
	. Universal Flip-Flop FT22		XDS	124713	2
	. Buffered Latch No. 3 FT26		XDS	126856	1
	. Control Flip-Flop FT28		XDS	126984	2
	. Delay Line Sensors HT15		XDS	127391	4
	. Gated Delay Line Sensor HT16		XDS	128011	3
	. Gated Inverter IT16		XDS	115264	3
	. NAND/NOR Gate IT24		XDS	128188	2
	. NAND Gate IT25		XDS	128190	7
	. Buffer Inverter LT13		XDS	123016	4
	. Buffer Inverter LT14		XDS	123017	2
	. Priority Interrupt LT16		XDS	123379	5
	. Logic Element LT21		XDS	126615	4
	. Address Selector ST14		XDS	123008	1
	. Time Base Selector ST29		XDS	129460	1
	. Terminator XT10		XDS	116357	6
	. Cable Plug ZT23		XDS	128164	3

Table A-12. Wired Board Assembly (Y), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Wired Board Y (see table A-10 for next higher assembly)		XDS	126596	Ref
	. Cable Driver AT12		XDS	124629	1
	. Buffered AND/OR Gate BT10		XDS	116056	1
	. Gated Buffer No. 1 BT16		XDS	125262	1
	. Gated Inverter IT16		XDS	125264	1
	. Terminator XT10		XDS	116257	1
	. Clock Terminator No. 2 XT19*		XDS	132015	2
	. Clock Terminator No. 2 XT19†		XDS	132015	1

*If memory map is not installed

†If memory protect is not installed

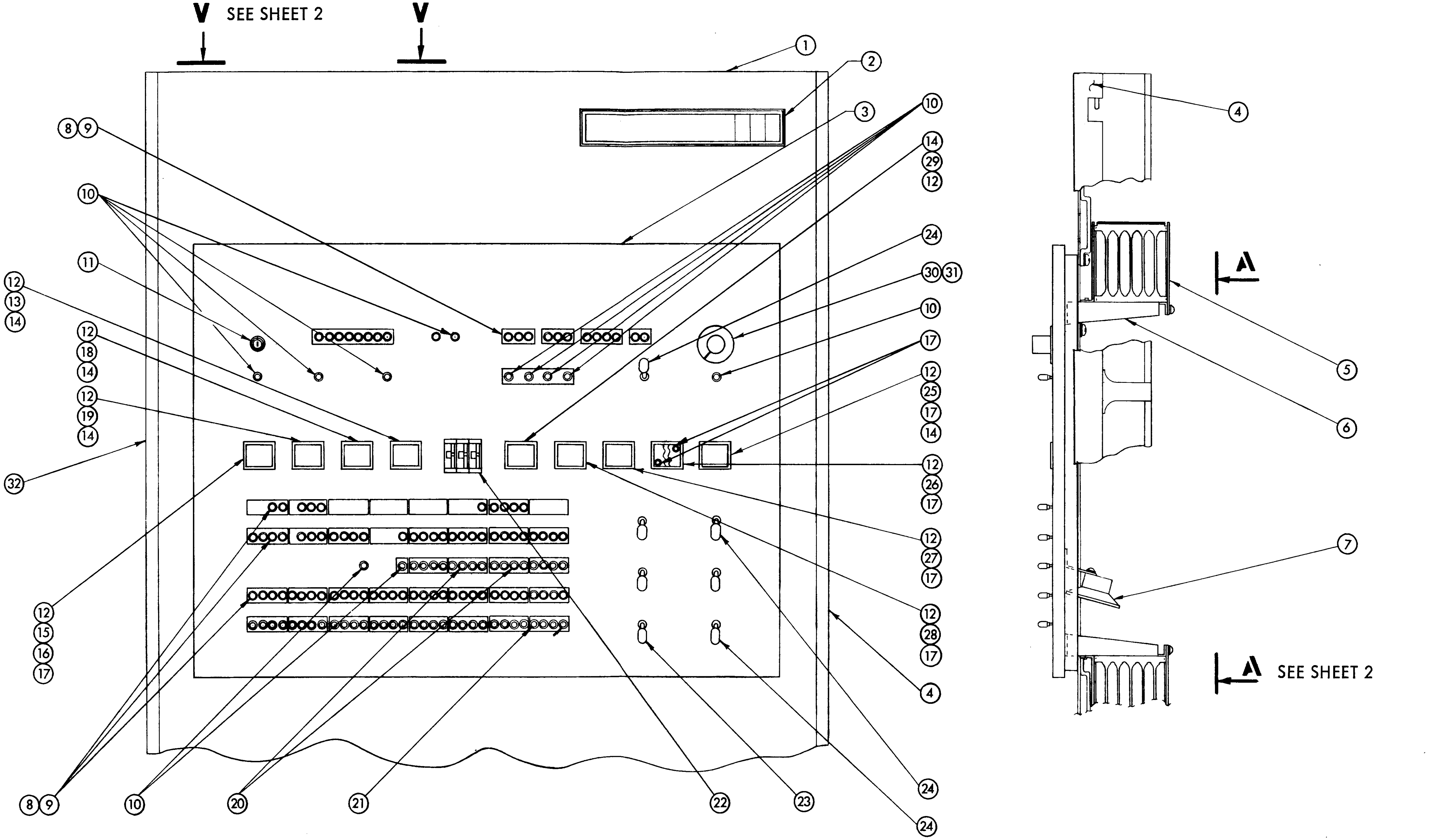
Table A-13. Processor Control Panel Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-3	Assembly, Processor Control Panel		XDS	117417	1
-1	. Door, PCP		XDS	117421	1
-2	. Nameplate, Sigma 7		XDS	126665-001	1
-3	. Panel Overlay		XDS	126388	1
-4	. Hinge, door		XDS	127335-002	1
-5	. Guide, module		XDS	126392	2
-6	. Chassis, connector, mtg		XDS	126389-002	2
-7	. Speaker, miniature		372	22A0-628	1
-8	. Clip, lamp mtg	DS1 through DS21, DS29 through DS98	56	10905-01	91
-9	. Lamp, miniature incandescent (108814-009)	DS1 through DS21, DS29 through DS98	56	CF032	91
-9A	. O-ring (111344-001)	DS1 through DS21, DS29 through DS98	333	2-X-N2197	91
-9B	. Lampholder	DS1 through DS21, DS29 through DS98	56	Q-082-2K	91
-10	. Switch Lever, single station	S2, S4, S6 through S12, S40, S41	371	18S-1021	11
-11	. Switch, keylock, 3-position	S3	XDS	132571	1
-12	. Housing, lampholder		162	2N6	9
-13	. Lampholder	DS27	XDS	116284-007	1
-14	. Switch, momentary, dpdt	S13, S14, S16, S17, S18	203	10EF1	5
-15	. Lampholder	DS28	XDS	116284-010	1
-16	. Switch, alternating action, dpdt	S19	203	10EF3	1
-17	. Lamp, miniature incandescent, 6V, 0.200A		84	328	10
-18	. Lampholder	DS99	XDS	116284-008	1
-19	. Lampholder	DS100	XDS	116284-009	1
-20	. Switch, lever, 8-station	S24 through S39	371	18S-1022	2
-21	. Switch, lever, 8-station	S44 through S75	371	18S-1023	4
-22	. Switch, thumbwheel, 16-position	S15	140	33644B-2	1
-23	. Switch, lever	S21, S22, S23, S43	371	22S-1002	4
-24	. Switch, lever	S5, S20, S42	371	22S-1001	3
-25	. Lampholder	DS22	XDS	116284-002	1
-26	. Lampholder	DS23	XDS	116284-003	1

(Continued)

Table A-13. Processor Control Panel Assembly, Replaceable Parts (Cont.)

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-3					
-27	. Lampholder	DS24	XDS	116284-004	1
-28	. Lampholder	DS25	XDS	116284-005	1
-29	. Lampholder	DS26	XDS	116284-006	1
-30	. Switch, rotary	S1	352	PS-70SL-2	1
-31	. Knob, skirted		XDS	126876	1
-32	. Trim, door		XDS	127336	1
-33	. Receptacle, male	J31	102	5287	1
-34	. Receptacle, female	J32	102	5256	1
-35	. Connector, one pin		371	3501F	2
-36	. Block, terminal	TB1	107	1492	3
	Console Interface No. 1, NT14		XDS	130666	8
	Console Interface No. 2, NT15		XDS	130674	6
	Console Interface No. 3, NT16		XDS	130682	4
	Lamp Driver, QT14		XDS	132055	1

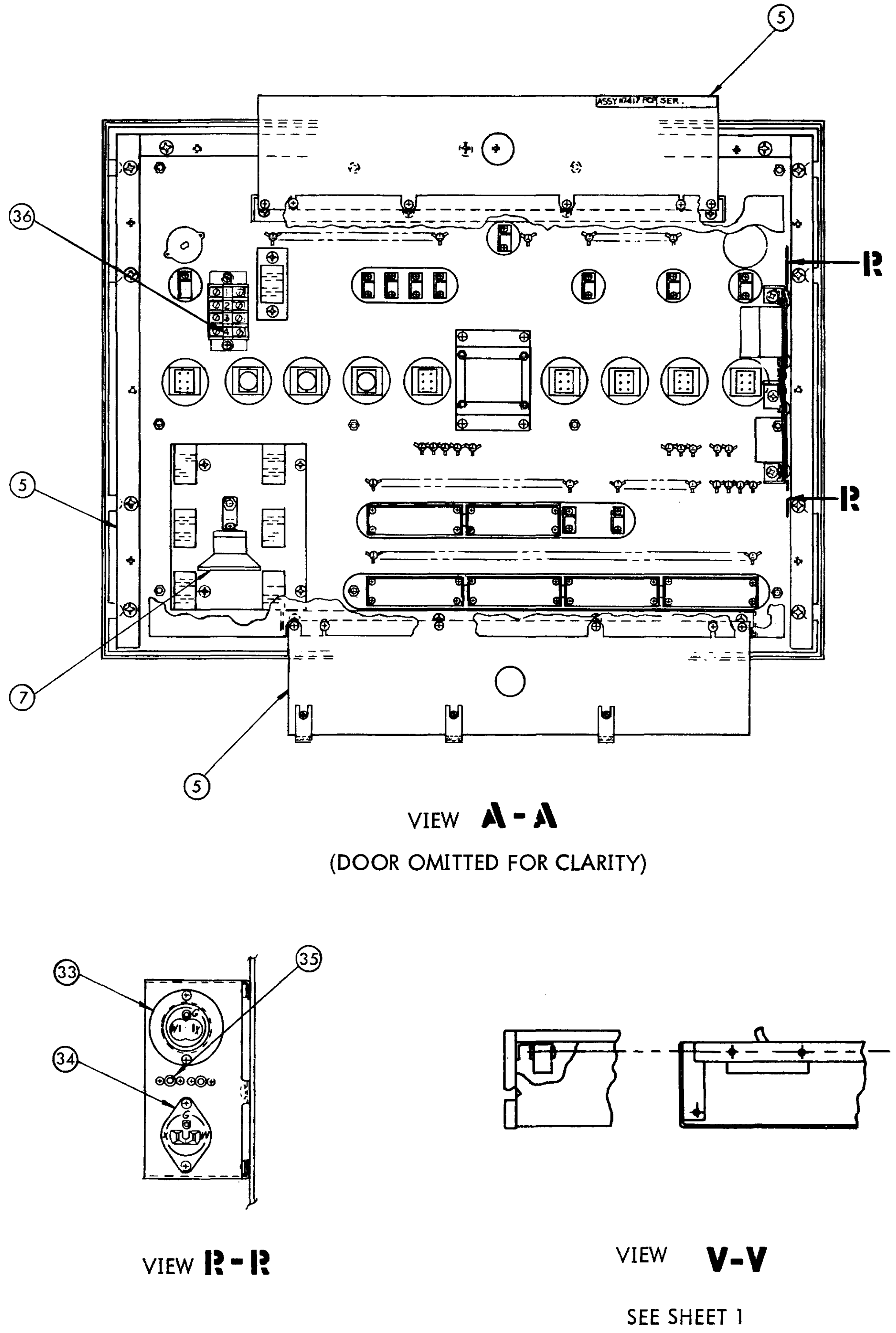


NOTE:

1. REFERENCE DRAWINGS, SCHEMATIC, P.C.P. NO. 126385.
WIRE LIST NO. 126386.
MODULE LOCATION CHART NO. 132038

2. REFERENCE DWG:117417-1L

Figure A-3. Assembly, Processor Control Panel
(Sheet 1 of 2)



NOTE: REFERENCE XDS DWG: 117417-2L

901060A.602/2

Figure A-3. Assembly, Processor Control Panel (Sheet 2 of 2)

Table A-14. Floating Point Arithmetic Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1, -2	Assembly, Floating Point Arithmetic		XDS	117611	1
	. Gated Inverter IT16		XDS	125264	4
	. Gated Inverter IT20		XDS	126747	1
	. NAND/NOR Gate IT24		XDS	128188	1
	. NAND Gate IT25		XDS	128190	4
	. Register Flip-Flop FT17		XDS	124628	17
	. Universal Flip-Flop FT22		XDS	124713	3
	. Buffer Inverter LT14		XDS	123017	1
	. Adder No. 1 LT17		XDS	123554	12
	. Carry No. 1 LT18		XDS	123590	4
	. Logic Element LT20		XDS	124717	1

Table A-15. Decimal Arithmetic Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, Decimal Arithmetic		XDS	117613	1
	. Cable Driver/Receiver AT11		XDS	123019	1
	. Gated Buffer BT17		XDS	126330	1
	. Gated Inverter IT16		XDS	125264	1
	. Logic Element LT20		XDS	124717	1
	. Cable Driver/Receiver AT11		XDS	123019	1
	. Buffered AND/OR Gate BT10		XDS	116056	5
	. BAND Gate BT11		XDS	116029	5
	. Gated Buffer No. 1 BT16		XDS	125262	1
	. BAND Gate BT18		XDS	126613	4
	. Register Flip-Flop FT17		XDS	124628	4
	. Counter Flip-Flop FT18		XDS	124634	5
	. Universal Flip-Flop FT22		XDS	124713	3
	. Fast Access Memory FT25		XDS	126743	3
	. Buffered Latch No. 3 FT26		XDS	126856	1
	. Gate Expander No. 2 GT11		XDS	124881	6
	. Gated Inverter IT15		XDS	117375	1
	. Gated Inverter IT16		XDS	125264	5
	. NAND Gate IT25		XDS	128190	1
	. Parity Generator LT12		XDS	117382	7
	. Carry No. 1 LT18		XDS	123590	2
	. Logic Element LT20		XDS	124717	1
	. Logic Element LT21		XDS	126615	24
	. Terminator XT10		XDS	116257	4
	. Cable Plug ZT23		XDS	128164	1

Table A-16. Memory Map Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Memory Map		XDS	117615	1
	. NAND/NOR Gate IT24		XDS	128188	1
	. Fast Access Memory FT25		XDS	126743	20

Table A-17. Real Time Clock Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Real Time Clock		XDS	117616	1
	. Priority Interrupt LT16		XDS	123379	2

Table A-18. Power Fail-Safe Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Power Fail-Safe		XDS	117612	1
	. Priority Interrupt LT16		XDS	123379	1

Table A-19. Additional Register Banks Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Assembly, High Speed Register Bank		XDS	117621	1
	. Fast Access Memory, FT25*		XDS	126743	4
	. Fast Access Memory, FT25†		XDS	126743	4
	. Fast Access Memory, FT25**		XDS	126743	4

*First additional register bank
†Second additional register bank
**Third additional register bank

Table A-20. Register Extension Unit Assembly and Register Extension Unit Interface, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-3	Assembly, Register Extension Unit		XDS	130071	1
	. Cable Driver/Receiver AT11		XDS	123019	5
	. Gated Buffer No. 1 BT16		XDS	125262	2
	. Fast Access Memory FT25		XDS	126743	4
	. Gated Inverter IT16		XDS	125264	1
	. Switch Comparator LT26		XDS	126982	1
	. Terminator XT10		XDS	116257	3
	. Cable Plug ZT23		XDS	128164	1
	. Fast Access Memory FT25*		XDS	126743	4
	. Fast Access Memory FT25†		XDS	126743	4
	. Fast Access Memory FT25**		XDS	126743	4
	Register Extension Unit Interface		XDS	132208	1

*Second register bank
†Third register bank
**Fourth register bank

Table A-21. Memory Protect Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Memory Protect		XDS	117617	1
	. NAND/NOR Gate IT24		XDS	128188	1
	. Fast Access Memory FT25		XDS	126743	4

Table A-22. Priority Interrupt Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Priority Interrupt		XDS	117330	1
	. Cable Driver/Receiver AT11		XDS	123019	4
	. Cable Driver/Receiver AT13		XDS	125260	1
	. Gated Buffer No. 1 BT16		XDS	125262	1
	. Gated Buffer BT17		XDS	126330	2
	. BAND Gate BT18		XDS	126613	1
	. Priority Interrupt LT16		XDS	123379	8
	. Switch Comparator LT26		XDS	126982	1
	. Address Selector ST14		XDS	123008	1
	. Terminator XT10		XDS	116357	1

Table A-23. Basic Memory, 4096 Words, Assembly, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-4	Assembly, Basic Memory, 4096 Words				
	. Basic Cabinet Model 8900 (see table A-29 for breakdown)		XDS	131416	1
	. Power Supply PT16		XDS	117264	1
	. Power Supply PT17		XDS	117265	1
	. Assembly, Basic Memory, 4K x 33 bit, Model 8251		XDS	132546	1
	. . Memory Frame		XDS	117500	1
	. . Resistor Connector		XDS	127315	8
	. . Cable Receiver AT10		XDS	123018	1
	. . Cable Driver/Receiver AT11		XDS	123019	6
	. . Rejection Gate AT16		XDS	126611	2
	. . Cable Driver/Receiver AT31		XDS	133053	1
	. . Gated Buffer BT16		XDS	125262	6
	. . Fast Buffer BT22		XDS	127393	11
	. . Buffered AND/OR Gate BT24		XDS	130967	3
	. . BAND Gate BT25		XDS	130947	1
	. . Delay Line DT11		XDS	126963	2
	. . Buffered Latch No. 2, FT37		XDS	130942	3
	. . Buffered Latch No. 3, FT38		XDS	130952	6
	. . Memory Sense Amplifier, HT11		XDS	123010	6
	. . Delay Line Sensor, HT15		XDS	127391	3
	. . Memory Preamplifier, HT26		XDS	131633	6
	. . Gated Inverter, IT14		XDS	126617	6
	. . Gated Inverter, IT16		XDS	125264	3
	. . NAND/NOR Gate IT24		XDS	128188	1
	. . NAND Gate IT25		XDS	128190	2
	. . Logic Element LT19		XDS	123915	1
	. . Logic Element w/ Inverter LT20		XDS	124717	1

(Continued)

Table A-23. Basic Memory, 4096 Words, Assembly, Replaceable Parts (Cont.)

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-4 (Cont.)	. . Logic Element w/ Buffer LT21		XDS	126615	5
	. . Parity Generator LT34		XDS	130958	9
	. . Memory Switch A ST10		XDS	123005	10
	. . Memory Switch B ST11		XDS	123006	16
	. . Toggle Switch ST14		XDS	123008	2
	. . Memory Preamplifier Selector ST15		XDS	123012	1
	. . Voltage Regulator ST17		XDS	131292	1
	. . Inhibit Driver ST21		XDS	132153	6
	. . Terminator XT10		XDS	116257	14
	. . Resistor Module C XT13		XDS	127791	9
	. . Resistor Module D XT14		XDS	127793	1
	. . Ribbon Cable ZT35		XDS	132277	3
	. . Core Diode Module, 8-bit		XDS	115549	3
	. . Core Diode Module, 9-bit		XDS	115550	1
	. . Memory Driver ST22		XDS	132159	4

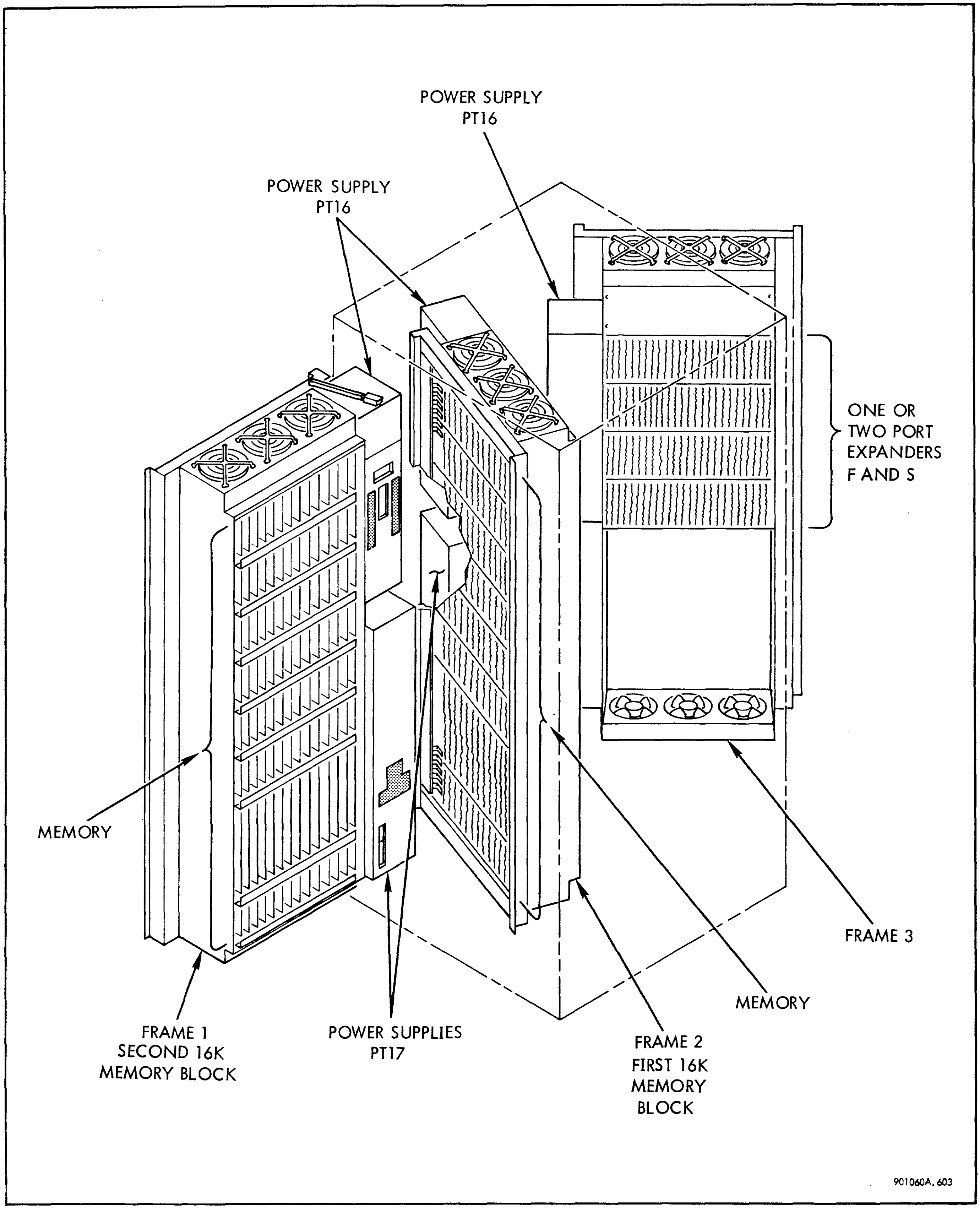


Figure A-4. Memory Cabinet

Table A-24. Memory Increment, 4096 Words, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-4	Assembly, Memory Increment, 4096 Words				
	. Memory Expansion Kit, 4K to 8K		XDS	117638	1
	. . Memory Preamplifier HT26		XDS	131633	5
	. . Memory Switch A ST10		XDS	123005	2
	. . Memory Switch B ST11		XDS	123006	16
	. . Memory Driver ST22		XDS	132159	1
	. . Core Diode Module, 8-bit		XDS	111549	3
	. . Core Diode Module, 9-bit		XDS	111550	1
	. Memory Expansion Kit 8K to 12K		XDS	117639	1
	. . Memory Preamplifier HT26		XDS	131633	6
	. . Memory Switch A ST10		XDS	123005	8
	. . Memory Driver ST22		XDS	132159	1
	. . Core Diode Module, 8-bit		XDS	111549	3
	. . Core Diode Module, 9-bit		XDS	111550	1
	. Memory Expansion Kit 12K to 16K		XDS	117640	1
	. . Memory Preamplifier HT26		XDS	131633	5
	. . Core Diode Module, 8-bit		XDS	111549	3
	. . Core Diode Module, 9-bit		XDS	111550	1

Table A-25. Three-Way Access Memory (Port A), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-4	Assembly, Port Expansion, 1 x 2		XDS	129463	1
	. Resistor Connector		XDS	127315	4
	. Interconnecting Cable		XDS	127314-102	5
	. Cable Receiver AT10		XDS	123018	1
	. Cable Driver/Receiver AT11		XDS	123019	3
	. Buffered Latch No. 2 FT37		XDS	130942	3
	. Logic Element w/Inverter LT20		XDS	124717	1
	. Logic Element w/Buffer LT21		XDS	126615	1

Table A-26. Six-Way Access Memory, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-4	Assembly, Six-Way Access Memory				
	. Memory Port Expander F		XDS	130625	1
	. . Cable Receiver AT10		XDS	123018	4
	. . Cable Driver/Receiver AT11		XDS	123019	16
	. . Rejection Gate AT16		XDS	126611	2
	. . Gated Buffer BT15		XDS	117389	1
	. . Fast Buffer BT22		XDS	127393	3
	. . Buffered AND/OR Gate BT24		XDS	130967	1
	. . Buffered Latch No. 3 FT26		XDS	126856	1
	. . Buffered Latch No. 2a FT27		XDS	126986	14
	. . Buffered Latch No. 3a FT38		XDS	130952	7
	. . Gated Inverter IT15		XDS	117375	1
	. . Gated Inverter IT16		XDS	125264	6
	. . Logic Element w/Inverter LT20		XDS	124717	4
	. . Logic Element w/Buffer LT21		XDS	126615	4
	. . Address Selector ST14		XDS	123008	2
	. . Terminator XT10		XDS	116257	7
	. . Twisted Pair Cable ZT38		XDS	133625	5
	. Memory Expander S		XDS	130626	1
	. . Rejection Gate AT16		XDS	126611	2
	. . Gated Buffer BT15		XDS	117389	1
	. . Fast Buffer BT22		XDS	127393	2
	. . Buffered AND/OR Gate BT24		XDS	130967	1
	. . Buffered Latch No. 3a FT38		XDS	130952	7
	. . Gated Inverter IT15		XDS	117375	1
	. . Gated Inverter IT16		XDS	125264	1
	. . Logic Element w/Inverter LT20		XDS	124717	4
	. . Logic Element w/Buffer LT21		XDS	126615	4
	. . Address Selector ST14		XDS	123008	2

(Continued)

Table A-26. Six-Way Access Memory, Replaceable Parts (Cont.)

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-4 (Cont.)	. . Terminator XT10		XDS	116257	5
	. . Twisted Pair Cable ZT38		XDS	133625	5
	. . Ribbon Cable ZT45		XDS	133218	4

Table A-27. Multiplexor I/O Processor, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, Multiplexer I/O Processor		XDS	117610	1
	. Cable Receiver AT10		XDS	123018	1
	. Cable Driver/Receiver AT11		XDS	123019	6
	. Cable Driver AT12		XDS	124629	2
	. Cable Driver/Receiver AT13		XDS	125260	1
	. Clock Driver No. 2 AT24		XDS	128168	1
	. BAND Gate BT11		XDS	116029	9
	. Delay Line DT11		XDS	126963	2
	. Increment/Decrement Register FT23		XDS	126749	5
	. Buffered Latch No. 1 FT24		XDS	126745	19
	. Fast Access Memory FT25		XDS	126743	5
	. Buffered Latch No. 3 FT26		XDS	126856	5
	. Buffered Latch No. 2 FT27		XDS	126986	6
	. Delay Line Sensors HT15		XDS	127391	2
	. Gated Inverter IT16		XDS	115264	5
	. NAND Gate IT25		XDS	128190	12
	. Parity Generator LT12		XDS	117382	1
	. Buffer Inverter LT13		XDS	123016	4
	. Logic Element LT21		XDS	126615	3
	. Switch Comparator LT26		XDS	126982	1
	. Terminator XT10		XDS	116257	8

Table A-28. Additional IOP/DC Expansion (Up to 8 Channels), Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-2	Assembly, IOP/DC Expansion (up to 8 channels)		XDS	117618	1
	. Fast Access Memory FT25		XDS	126743	15

Table A-29. Basic Cabinet Hardware, Replaceable Parts

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1	Basic Cabinet (see table A-1 for next higher assembly)		XDS	131416	1
	. Cabinet, basic structure		XDS	117419	1
	. Cap, cabinet top		XDS	134232	1
	. Angle, mounting		XDS	117422	4
	. Plate, bottom		XDS	130146	1
	. Assembly, power distributor box		XDS	117428	1
	. Panels, side		XDS	117338	1
	. Door, cabinet		XDS	131419	1
	. . Trim, door		XDS	127336	1
	. . Hinge, door		XDS	127335-002	1
	. . Trim, side		XDS	127384	4
	. . Catch, spring loaded		XDS	106829	2
	. Assembly, power distributor box, chassis panel		XDS	130155	1
	. Casters, full swivel, 16-32-1/2- xK		373		4
	. Frame, swing		XDS	117319	1
	. Door, chassis column		XDS	124386	1
	. Bracket, pin protect		XDS	132165-002	2
	. Block, stop		XDS	132523	2
	. Assembly, bus bar		XDS	130111	1
	. Cable Clamp		XDS	100657-002	6
	. Cable Clamp		XDS	100657-004	3

(Continued)

Table A-29. Basic Cabinet Hardware, Replaceable Parts (Cont.)

Fig. & Index No.	Description	Reference Designator	Manufacturer	Part No.	Qty
A-1 (Cont.)	. Door, Chassis column		XDS	134169	1
	. Assembly, top fan (see tables A-2, A-6, and A-10 for next higher assembly)		XDS	123943	1
	. . Bracket, fan		XDS	117281	2
	. . Cord, ac		XDS	126374-001	1
	. . Bushing, strain relief		374	SR	3
	. . Fan, electric		139	MARK 4	3
	. . Guard, fan		XDS	111187	3
	. Assembly, bottom fan (see tables A-2, A-6, and A-10 for next higher assembly)		XDS	117320	1
	. . Support, side, blower chassis		XDS	126337	2
	. . Plate, base, blower chassis		XDS	126338	1
	. . Filter Air		XDS	117427	1
	. . Cord, ac		XDS	126374-001	1
	. . Bushing, strain relief		374	SR	1
	. . Shim, sponge rubber, 34 inches		XDS	100238-001	1
	. . Fan, electric		137	MARK 4	3

Table A-30. Manufacturer's Code Index

Code No.	Name	Address
55	Centralab Electronics	900 E. Keefe Ave., Milwaukee, Wisc. 53201
56	Eldema Corp.	18345 Susana Rd., Compton, Calif. 90221
84	General Electric Co., Miniature Lamp Dept.	Nela Park, Cleveland, Ohio 44112
102	Harvey Hubbell Inc.	Harvey St. and Bostwick, Bridgeport, Conn. 06600
107	Allen-Bradley Co.	1201 Second St., Milwaukee, Wisc. 53204
139	Rotron Mfg. Co.	Woodstock, N. Y. 12498
140	The Digitran Co.	855 S. Arroyo Pkwy., Pasadena, Calif. 91105
162	Honeywell, Micro Switch Div.	11 W. Spring St., Freeport, Ill. 61033
203	Master Specialties Co.	15020 Figueroa, Gardena, Calif. 90247

(Continued)

Table A-30. Manufacturer's Code Index (Cont.)

Code No.	Name	Address
352	Buckeye Stamping Co.	555 Marion Rd., Columbus, Ohio 43207
371	Switchcraft, Inc.	5533 N. Elston Ave., Chicago, Ill. 60630
372	Quam-Nichols Corp.	Marquette Rd. at Prairie Ave., Chicago, Ill. 60637
373	Darnell Corp.	12000 S. Woodruff Ave., Downey, Calif. 90240
374	Heyman Mfg. Co.	E. Michigan Ave., Kenilworth, N.J. 07033

READER SURVEY

XDS
Xerox Data Systems

Use this postpaid form to communicate any constructive comments you may have regarding this publication. Please be specific.

PUBLICATION

NUMBER	DATE OF PUBLICATION	TITLE	REV.

HOW DID YOU USE THIS PUBLICATION

□ LEARNING □ INSTALLING □ MAINTAINING □ OPERATING □ SALES

□ OTHER ______________________

COMMENTS

YOUR NAME AND RETURN ADDRESS

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 229
EL SEGUNDO, CALIF.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

CUT ALONG LINE

POSTAGE WILL BE PAID BY

Xerox Data Systems
701 South Aviation Blvd.
El Segundo, California 90245

ATTN: FIELD ENGINEERING PUBLICATIONS

FOLD

XEROX

701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511