INTRODUCTION

The ND812 Computer, part number ND88-0397, manufactured by Nuclear Data Incorporated, is a general purpose computer which may consist of the ND812 Processor, ASR33 Teletype and a variety of peripheral equipment, such as high speed readers and punches, line printers, plotters, magnetic tape and disk storage devices, and CRT's. The ND812 is a 12-bit computer with 8K memory (basic), expandable to 16K in 4K increments. Memory cycle time is 2 microseconds. Standard features of the ND812 are hardware multiply and divide.

This publication consists of one volume covering operation and maintenance information for the ND812 Computer System. The information contained in this volume covers only the basic system consisting of the ND812 central processor unit. This publication is presented with the assumption that the reader has had experience with small computers. To effectively use the material in this manual for repair of the system, maintenance personnel should have a fair working knowledge of machine language programming. (Refer to the Principles of Programming the ND812 Computer in Assembly Language.)

This publication is divided into seven sections.

Section 1 Contains general information about the system, such as its purpose, description and general specifications. It also contains descriptions of the various options and peripherals that are available to expand the basic system.

Section 2 Covers instructions for site planning, unpacking and inspection, interconnection cabling and other information required to effect installation of a system.

Section 3 Describes the various controls and indicators of the system, and provides manual operating instructions essential for preparation of the system in performing a programmed function, for troubleshooting, and for maintenance

Section 4 Contains the theory of operation of the basic system.

Section 5 Covers preventive maintenance, and corrective maintenance.

Section 6 Provides a replaceable parts list.

Section 7 Contains diagrams necessary for maintenance of the ND812.

Related Publications
Principles of Programming the ND812 Computer in Assembly Language, IM41-0000-00
ND812 Diagnostics, IM41-8001-01
ASR33 Teletype Instruction Manual
Applicable Peripheral and Interface Manuals

TABLE OF CONTENTS

SECTION TITLE PAGE

SECTION TITLE PAGE

SECTION TITLE PAGE

LIST OF ILLUSTRATIONS

FIGURE TITLE PAGE

FIGURE TITLE PAGE

LIST OF TABLES

SECTION I
DESCRIPTION

1.1 PURPOSE OF EQUIPMENT

The ND812 Computer is a binary, parallel, synchronous computer specifically designed to be used in dedicated computer-based systems. It can also be used as a general purpose computer.

1.2 PHYSICAL DESCRIPTION

The ND812 (see Figure 1-1) is contained in an aluminum housing designed to fit into a standard 19-inch rack with other equipment. The ND812 is supplied with aluminum panels on all sides (except the operator console which is plastic) which can be easily removed for maintenance purposes. The front panel contains clearly marked controls and indicators for manual operation.

The computer is a solid state device employing medium scale integrated circuits (MSI). As shown, all boards are easily accessible for maintenance. The ND812 is prewired and mechanically designed to accommodate two I/O control cards for optional or peripheral equipment. Additional interfaces to other peripheral devices can be contained in a separate I/O expansion cage.

1.3 FUNCTIONAL DESCRIPTION

To introduce the reader to the major functional units of the ND812 Computer (see Figure 1-2), the following circuit grouping and discussions are presented as a guide:

1) Manual Start and Control (19)
2) Clock Generation and Timing (23, 24)
3) Control Registers (1, 2,3, 4, 6, 7, 18, 20 and 25)
4) Accumulator Registers J, K, R, and S (1, 2, 3, and 4)
5) Multiplexers (9, 15 and 17)
6) Arithmetic Unit (11)
7) Memory Unit (26)
8) I/O Processor
9) Power Supply Unit

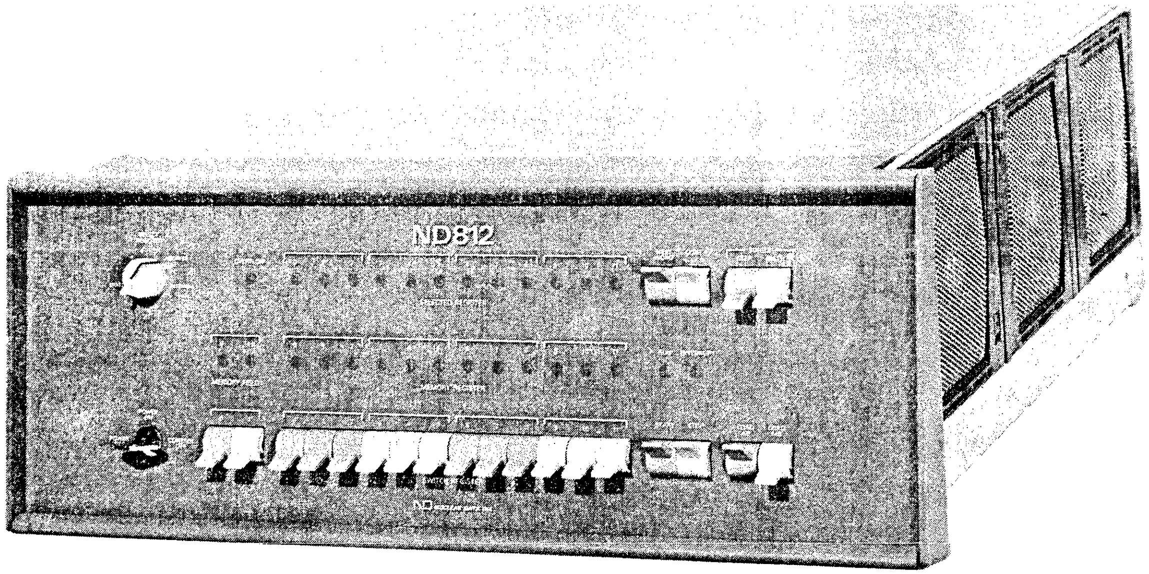


Figure 1-1. ND812 Computer.

1.3.1 MANUAL START AND CONTROL

The ND812 system is comprised of its front panel and a Teletype (TTY) keyboard (Type ASR-33 Automatic Send Receive Model) together with other peripheral equipment. The Teletype provides a means of communicating with the ND812 in its automatic or run mode. The front panel, on the other hand, provides communication with the ND812 in a manual mode. Normally, the ND812 operates automatically through program control. It examines successive locations in memory and performs indicated operations. However, manual operation is necessary for many tasks. For example, the binary loader, which permits the ND812 to load a program into core memory, must be manually loaded into the computer (if the hardware auto-load interface (which reads in binary) is not used).

NOTE

> "The Binary Loader" is a program consisting of 176_8 instructions. (ND41-0005)

The front panel provides means to store a program (although the process is extremely laborious), to initiate operation of the ND812, and to examine the contents of various memory locations and other ND812 registers. It is also possible to change the contents of memory. The operator console is described in Section 3 of this manual.

Once a program is assembled into machine language (binary format) it may be either loaded through the binary loader from pre-punched paper tape or cassette, or it may be manually loaded through the front panel switch register. When the machine POWER switch is placed in the POWER ON position, a power ready condition is established and the operator console is conditioned for manual operation. Thus, the operator, or field service engineer may manually load his program instructions into memory. Once a program is loaded into memory, either manually or automatically through one of the hardware loader interfaces, or through program control by the Binary Loader, the machine "run" condition may be implemented by depressing the front panel START switch, if the starting address has been initialized by a LOAD AR to establish a starting address.

1.3.2 CLOCK GENERATION AND TIMING

Machine cycle timing is produced by a 16 MHz crystal-controlled oscillator, a pulser, and logic circuits (Figure 1-2) which generate three discrete sets of timing pulses for control of machine cycle timing. The combined operation of these units produces a parallel train of continuous keying pulse units (PU0 through PU7) that are segmented into continuous groups of eight pulses. These PU pulses are used to generate eight basic phase (BP0 through BP7) and seven execute phase (EP0 through EP6) pulses together with four peripheral control pulses (PCP0* through PCP3*). Peripheral control pulses (PCPs) are only generated when the instruction decoder senses an I/O instruction.

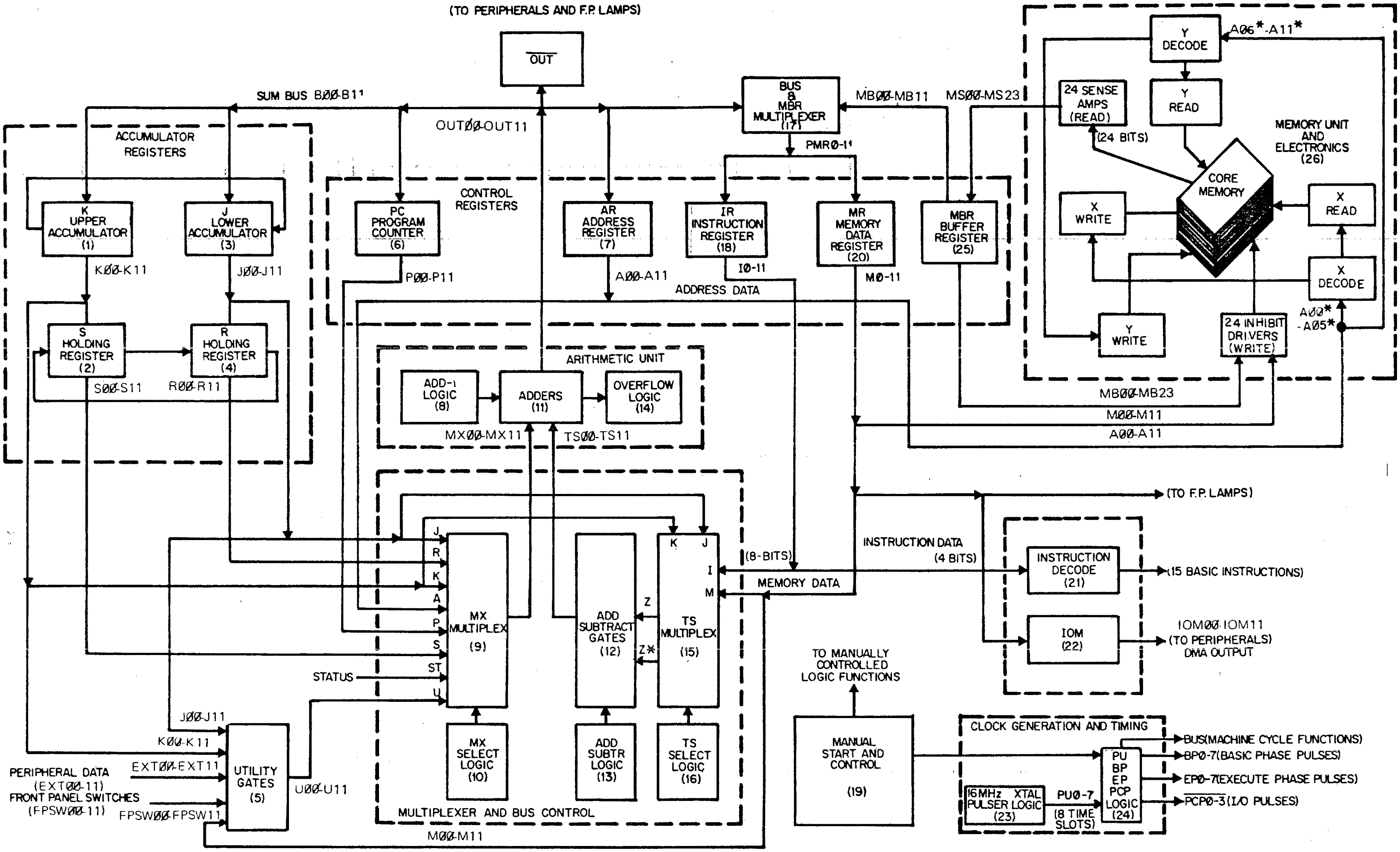


Figure 1-2. ND812 Block Diagram

NOTE

Signals followed by an * indicates that the signal is active in its low-level state.

Fundamental pulses (PUs) are generated in continuous trains of 8 pulses. These pulses are generated by a pulser that is automatically recycled at the end of a train. The PU pulses are used to key logic that controls the generation of the basic phase and execute phase pulses; they also synchronize various control functions within the processor.

Instructions that are processed by the ND812 are segmented by time periods called phases. Operate instructions, for example, generally require only one phase for their execution, although there are exceptions. Memory reference instructions, on the other hand, may require two or more phases. There are two types of phases, the basic and execute. At the end of any instruction, whether it requires one or more phases, a register called the done latch is set. The done latch enables a new machine cycle for the execution of the next instruction in the program. A machine cycle can consist of one or more phases, but always begins with a basic phase. The execute phase is used only during the execution of a memory reference instruction. When the basic phase for the new machine cycle has been entered, the done latch is cleared, but is set again at the conclusion of the instruction. Thus a machine cycle is defined as the period between the set condition of the done latch at the beginning of the instruction, and the set condition of the done latch at the end of the instruction.

1.3.2.1 BASIC PHASE. When the ND812 is in the run state the processor is continuously recycling through the basic and execute phases of the fundamental machine cycle, or, in the case of an I/O instruction, deferring a machine cycle to an I/O cycle.

At the beginning of a machine cycle, the contents of the program counter (PC) are transferred to the address register (AR). Later in the cycle the PC is incremented by one and the result transferred back into the program counter. The contents of the address register is the core location from which a 12 bit binary word is obtained. Memory can only contain a binary word 12 bits long. This word may be an instruction, or half of a two word instruction; an effective or intermediate (indirect) address; or data which may be formatted in any possible combination of 12 bits. Then the memory is read and the contents of the core from the location specified by the address appears at the read outputs of the memory and are set into the memory buffer register. The Memory Buffer is then transferred to the memory data register. Before the memory buffer transfer, the twelve bits of the instruction are transferred to the instruction register and decoded.

1.3.2.2 EXECUTE PHASE. The execute phase is entered for all memory reference instructions except JMP and XCT. For instructions that affect the contents of memory such as increment, decrement, and load-store, the contents of the core location established by the address are read into the memory buffer and the operation specified by the contents of the instruction register is executed. During a store instruction, the contents of the J or K registers are transferred into the memory data register and written into memory at the address specified.

1.3.3 CONTROL REGISTERS

The ND812 has two instruction decoders and four control registers. The control registers store data during the execution of an instruction. These registers do not include the core memory which stores the instructions and operands to be processed when a machine cycle is executed. The five control registers and instruction decoders are used to store data which is processed during the execution of an instruction. These are:

a. Instruction Decoders (21, Figure 1-2)

b. Instruction Register (18)

c. Address Register (7)

d. Program Counter (6)

e. Memory Buffer Register (25)

f. Memory Data Register (20)

1.3.3.1 INSTRUCTION DECODER. The instruction decoder (21) receives its inputs directly from the instruction register (IR). The instruction decoder receives the first four bits of any instruction during the basic phase. It decodes these four bits to determine the type and class of instruction. The four types of instructions that are decoded are the memory reference instruction (MRI), the operate instruction (OPI), the literal instruction (LIT), and the input-output (I/O) instruction. The operate instructions are further divided into two classes or groups of instructions, class 1 (OP1) and class 2 (OP2).

1.3.3.2 INSTRUCTION REGISTER. The IR (18) holds a 12-bit instruction word and receives its input from the memory buffer multiplexer (17). The data held in the IR is retained during the time the instruction is processed because bits 5 through 11 of any instruction word are used to obtain various functions of the logic during each machine cycle. The IR is cleared at the beginning of each machine cycle. Two-word instructions are shifted 3 bits left during the second basic phase.

1.3.3.3 ADDRESS REGISTER. The AR receives its 12-bit inputs from the adder. The contents of the AR specifies the memory location from which a word is fetched and restored during each memory cycle. Any data loaded into the AR is retained until it is replaced by new data. Twelve-bit AR data is outputted to the address decoders in the memory unit so that when a word is fetched or is rewritten into memory it is always from or to the location specified by the content of the AR. AR outputs are also supplied to the MX multiplexer so that any current address may be summed with the relative address in an instruction word.

1.3.3.4 PROGRAM COUNTER. ThePC is a 12-bit register that holds the location of the next instruction that will be fetched from memory during program execution. The PC receives its inputs from the adders. Outputs are provided to the MX multiplexer.

1.3.3.5 MEMORY BUFFER REGISTER. The MBR receives all data read from memory during a memory cycle. This is a word that can either be an instruction, an address, or pure data. This 24-bit register receives its inputs directly from the outputs of the memory sense amplifiers. Any data read into the MBR is retained until it is cleared at the beginning of a phase. The 24-bit output of the MBR is supplied to the Bus and memory buffer multiplexer. Twelve bits are switched to the MDR and, under appropriate circumstances, to the instruction register.

1.3.3.6 MEMORY DATA REGISTER. The memory data register holds all information that is read from, or is written into memory, transferred as outputs to peripheral equipment, or examined via the memory register display on the ND812 front panel. Any information read into the memory data register is retained until replaced by new data. The 12-bit output of the memory data register is also supplied to the TS multiplexer so that the contents of the address register can be replaced when indirect addresses are specified by the instruction currently being executed.

1.3.4 ACCUMULATOR REGISTERS (J, K, R, and S)

The accumulators consist of four 12-bit registers designated as the J, K, R, and S registers. These registers are also referred to as the accumulators (J, K registers), and sub-accumulators (R and S registers). The J and K registers are capable of direct storage to, and loading from, memory. Their respective contents may be summed, subtracted, ANDed, shifted, rotated, and exchanged. All arithmetic results appear in the J or K registers, except in the case of multiplications where the product appears in the R and S sub-accumulators. The sub-accumulators are not directly addressable (loaded from or stored in memory), but can be loaded from, or to, or exchanged with the contents of the J and K registers.

The J and K registers are parallel-loaded with data from the adder bus. Each register delivers its contents to the TS and MX multiplexers (9 and 15). In addition to its MX multiplexer outputs, the K register is connected (bit-parallel) to the S register and the J register is connected (bit-parallel) to the R register.

1.3.5 MULTIPLEXERS

The ND812 has three multiplexers, each of which receives various source inputs. These multiplexers are actually high-speed electronic switches and output any one of its several inputs. The three multiplexers are the bus and memory buffer multiplexer, the TS multiplexer, and the MX multiplexer.

1.3.5.1 BUS AND MEMORY BUFFER MULTIPLEXER. The bus and memory buffer multiplexer receives two inputs, one from the sum bus and the other from the MBR. Thus, during a machine cycle, data from either source can be switched to the IR and to the memory data register for subsequent processing. Information can be outputted to peripheral equipment through the memory data register, and the IOM gates. The MDR contents is displayed at all times on the front panel MR indicators.

1.3.5.2 TS MULTIPLEXER. The TS multiplexer receives four different inputs. Three of these inputs are: The J register, the K register and the memory data register. The fourth input is a partial word and consists of bits 6 through 11 of the IR. Any one of these four data words can be switched as an output from the TS multiplexer by the TS select logic. There are actually two output paths from the TS multiplexer which are supplies to the Add-Subtract gates. These gates select either the data word (Z) or its complement (Z*) for a subsequent summing operation. When its output data is to be added to the output from the MX multiplexer, the data word selected by the gates (12) is the complement of the selected input data word (Z*), because the Add-Subtract Gates invert their inputs. However, subtraction requires the true (Z) of the data word taken from the TS multiplexer. The outputs from these gates are supplied directly to the adders where the actual summing operation takes place. During addition the output from the add-subtract gates is summed with the data from the MX multiplexer resulting in simple binary addition. During subtraction the resulting complement of the data word is taken and incremented by a count of 1 by the add-1 logic resulting in a 2's complement addition (binary subtraction). When 2's complement addition takes place, a carry input is propagated through the adder together with the complement to provide the resultant subtraction.

1.3.5.3 MX MULTIPLEXER. The MX multiplexer receives eight 12-bit input data words from various sources. These sources include the following registers:

a. K Register (1)
b. J Register (3)
c. S Register (2)
d. R Register (4)
e. Address Register (7)
f. Program Counter (6)
g. Status
h. Utility Gates (5)

Depending on the state of the MX select logic (10), the data word contained in any one of the registers listed in a through f above can be switched through the MX multiplexer (9) for output to the adders (11). The status inputs are derived from several registers including the overflow logic (14), the flag flip/flop, the current memory field, the jump to subroutine memory field, and the interrupt memory field status. These data are normally loaded into the J register where they are then transferred into the memory data register through the buss and memory buffer register multiplexer (11) for storage somewhere in memory (20) when a powerfail condition causes an interrupt. The utility gates (5) also permit one of several inputs to be loaded into the MX Multiplexer. The utility gates accept data from either the J or the K registers (1 and 3), data from a peripheral device (which is transferring a word into one of the ND812 registers), or from the front panel switch register switches. The output from the MX multiplexer is applied to the adders (11) and a summing operation occurs whenever data is outputted from either the TS multiplexer or the MX multiplexer. However, in the absence of a TS subtract or TS add function from the TS add subtract logic (10), the output from the TS multiplexer is all zeroes. Output from the MX multiplexer, however, can be derived from any one of its eight input data words.

1.3.6 ARITHMETIC UNIT

The arithmetic unit is a high-speed full-binary adder (11) operating in conjunction with the add-subtract logic (13) and the TS multiplexer (15) output data which is summed with the MX multiplexer (9) outputs. The adder features 12-bit ripple carry addition. The adder output data is applied to the sum buss where it is made available to various registers. The adder performs all summing operations through straight binary addition or through 2's complement addition for subtraction. Hardware multiplication is carried out through successive binary addition and hardware division is carried out through successive 2's complement addition.

1.3.7 MEMORY UNIT

The memory unit (standard 8K configuration) consists of a 8K (4096 x 12 bit locations) core stack and associated read/write electronics. Control registers external to the memory include Memory Buffer Register, Memory Address Register, and Memory Data Register.

The memory unit is organized around two 4K fields (4096 x 12 each) to simplify programming. All locations within a memory field are directly accessible to the program or programmer (and operations that involve data outside the current field can be implemented through program control. Hardware return from the called field to the main field is standard with the 8K or larger memory configurations. A two-word MRI permits selection of any memory field for the location of a word or the address of a JMP instruction. Additionally a two-word instruction can directly address any location within any field, but, a single word instruction only directly addresses memory locations relative to the indicated (current) value of the PC (128 locations). The single-word instruction can indirectly address any location within the current field.

1.3.8 INPUT/OUTPUT (I/O) UNIT

The I/O unit interfaces with the central processor and the external interface controller logic for all peripheral units. It consists of gating and control logic which enables program-controlled data transfers to and from peripheral devices. In addition to the data transfers, the I/O unit also enables hardware interrupts (excluding an internal power-fail interrupt), a direct-memory access (DMA), and a real-time-clock (RTC) option (when included).

When data is transmitted to an I/O device by a program-controlled input/output transfer, the J and K registers place data on the sum buss. This data appears on 12-parallel lines (OUT 00 thru OUT 11*) through output gating and is available to all peripheral devices. On interrupt operations, input EXT 00 thru EXT 11 data lines transfer a trap address on the DMA (direct memory access) Controller. These lines transfer the desired memory address to the AR. Peripheral control operations such as device select codes are outputted to the peripheral device through a second group of 12 output lines (IOMs).

1.3.9 POWER SUPPLY UNIT

The power supply unit basically consists of a 115/230 vac power transformer, bridge rectifier, +5 Vdc 10A regulator and 30 Vdc 4A regulator. The power supply unit converts 115 Vac or 230 Vac into regulated +5 Vdc for operation of all ND812 computer logic circuitry, and regulated 30 Vdc for operation of the Memory Unit.

The power supply unit also includes power sense logic circuitry. The power sense circuitry operates two latches in the power supply unit. In the event of internal or external power failure, one latch provides a bad power signal to the computer timing circuits. The other latch provides a delayed power on signal to various logic elements throughout the ND812 computer to initially clear pertinent registers and latches when power is turned on.

1.4 GENERAL SPECIFICATIONS

TYPE

Digital stored-program general-purpose computer.

MEMORY

Magnetic core, 8192 words, 12 bits, 2 µsec cycle time. Memory options: Minimum 4K, expandable in the field to 16K in 4K increments.

ADDRESSING

Relative, indirect, and direct. Hardware multiple field control.

ARITHMETIC

Parallel, binary, fixed point, 2's complement. Hardware multiply and divide are standard features.

INSTRUCTIONS

Single and double word instructions which include 16 memory reference instructions, three literals, and more than 50 arithmetic and register control instructions.

INPUT/OUTPUT

(1) Interrupt: Programmable 4-level priority interrupt. Trap to any core location in first 4K of memory.

(2) Programmed I/O transfer: Capability per single I/O instruction:
Transmit 12 or 24 bits
Receive 12 or 24 bits
Transmit 12 and receive 12 bits
Receive 12 and transmit 12 bits

I/O instruction includes four (4) microprogrammable pulses for multi-function operation with a singl e instruction.

With single-word instructions there are 256 possible I/O commands at 3 microseconds per instruction. With two-word instructions there are 4096 possible I/O commands at 5 microseconds per instruction.

Total of 78 control, data, and sense lines available on a single connector. Direct Memory Access (DMA); 6 megabits per second/read, load, increment or decrement on DMA, in a single cycle.

ACCUMULATORS

Dual accumulators with individual sub-accumulators.

CONTROL PANEL

Constant display of memory register. Switch-selected display of six other registers and two busses when not in run state.

Front panel removable key lock. Power off, on, control off (panel lock).

Removable front panel for dedicated O.E.M. applications.

TIMING

16 MHz crystal controlled clock.

SOFTWARE

Includes Assembler, Editor,diagnostics, Utilities (Integer Arithmetic Package, Floating-Point Package), and NUTRAN interpreter.

1.5 EFFECTIVITY

This instruction manual is effective for ND812 Central Processors bearing serial numbers 70-350 and above, and supercedes all previous editions. Changes to this instruction manual are included in the front of the manual as applicable.

1.6 ORDERING ADDITIONAL MANUALS

One manual is shipped with each ND812 Central Processor. Additional manuals may be purchased through your nearest Nuclear Data representative (refer to rear cover of this manual for addresses). Specify the equipment model number, serial number, and IM number shown on the title page.

SECTION II
INSTALLATION

2.1 SITE PLANNING

2.1.1 SPACE REQUIREMENTS AND INSTALLATION CONSTRAINTS

Normal heat generated by the ND812 will not hamper its operation. However, it should not be located over, or in proximity to radiators or systems using vacuum tubes since the high ambient heat of such devices may affect the operation. Do not cover the left side ventilation inlet or the fans which are located on the right side of the ND812.

Figure 2-1 shows the overall dimensions of the ND812. The system should be situated so that space is available for maintenance. If a Teletype is used in the system, its signal cable must not exceed 100 feet in length.

2.1.2 POWER SOURCE

The ND812 is powered by a power supply which requires a nominal 115/230 Vac 50-60 Hz source, free of excessive noise or fluctuations. A voltage stabilizing transformer can be inserted between the AC source and the power supply where available power is subject to large fluctuations. Noise produced by various types of electrical equipment can be eliminate(or greatly reduced by connecting a suitable filter between the AC source and the interfering equipment.

2.2 UNPACKING AND INSPECTION

Carefully unpack the ND812 and ASR-33 Teletype, saving the shipping cartons for possible reshipment. Check all items, such as program tapes and mounting hardware, cited on the packing slip. Thoroughly inspect the units for damage in shipping. If damage is apparent or parts missing, notify your nearest Nuclear Data Sales office or the Nuclear Data home office and the discrepancy will be promptly adjusted.

2.3 INTERCONNECTION CABLING

The following procedure includes optional equipment interconnection cabling, disregard devices that are not applicable. (Refer to Volume II for board and connector locations.)

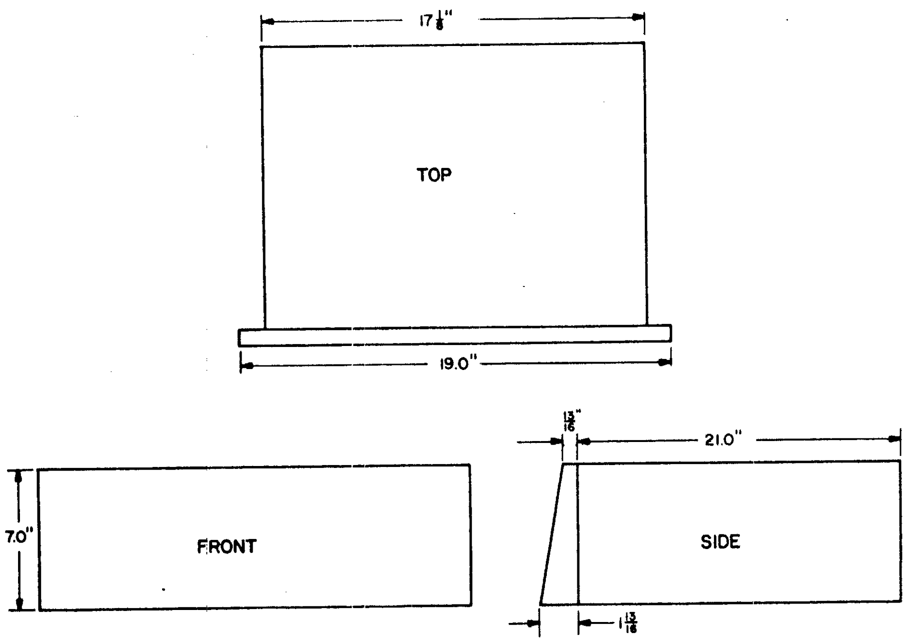


Figure 2-1. ND812 Dimensions

a. Be sure that the front panel POWER ON/POWER OFF/CONTROL OFF switch is in the POWER OFF position.

b. Connect the gray ribbon cable and IC connector originating from the ASR-33 Teletype to the TTY integrated circuit (IC) socket on the teletype interface board using care that cable connector pins correspond with receptacle pins.

NOTE

The IC connectors are not idiot proof and can easily be reversed in the IC socket.

c. Connect the gray ribbon cable and IC connector originating from the high speed punch to the IC socket (labeled PUNCH) on the high speed Punch/Reader interface board.

e. Connect the 86 pin multicolored (or white) ribbon cable and card edge I/O connector to the peripheral (CMTC, TCS, Disc) that is to be interfaced to the ND812.

f. Connect the AC line cord to the 115-Vac source.

NOTE

Only the ASR-33 Teletype should be powered through the 115-Vac outlet provided on the rear panel of the ND812.

2.4 CHECKOUT

There is no particular turn on sequence for the ND812 system, either the peripherals or the central processor can be turned on first. However, no peripheral should ever be turned on or off while the central processor is in the run state.

Upon completion of installation, refer to Section 3 and load the Binary Loader program and run the diagnostic routine to ascertain that equipment is functioning properly.

SECTION III
OPERATION

3.1 CONTROLS AND INDICATORS

3.1.1 ND812 OPERATOR CONTROL PANEL

The ND812 Operator Control Panel, shown in Figure 3-1, contains control switches and indicators that provide the means for local operator control of the system. These controls and indicators are used in the normal operation of the computer and in the step mode of operation for program debugging and troubleshooting of the system. The lower set of indicators display the contents of the memory register continually. Upon selection, if not in run, the upper set displays the contents of any one of the operational registers, S (upper sub-accumulator), R (lower sub-accumulator), K (upper accumulator), J (lower accumulator), address register, PC (program counter), status lines, and EXT lines. Data entry into a memory is accomplished while the system is in a stop mode using the panel switches. When the system is in a run mode, these switches may be sensed under software control.

The function of each of the switches and indicators of the front panel is described in Table 3-1.

3.1.2 ASR-33 TELETYPE

The ASR-33 Teletype, shown in Figure 3-2, provides for on line operator communication with the software being executed by the computer. The function of the Teletype keys is described in Table 3-2.

3.2 MANUAL OPERATION AT THE OPERATOR CONTROL PANEL

The operator control panel (Figure 3-1) permits programmers and maintenance personnel to control functions of the computer manually. Using the controls and indicators for programming, data can be entered as an address, or as an operand, through the switch register switches. Thus, any program can be manually loaded into the ND812. In addition to the controls that permit loading of data, this panel has a number of switch controlled display indicators that permit maintenance personnel to examine the content of various registers of the ND812. Other display indicators indicate the current operating state of the ND812.

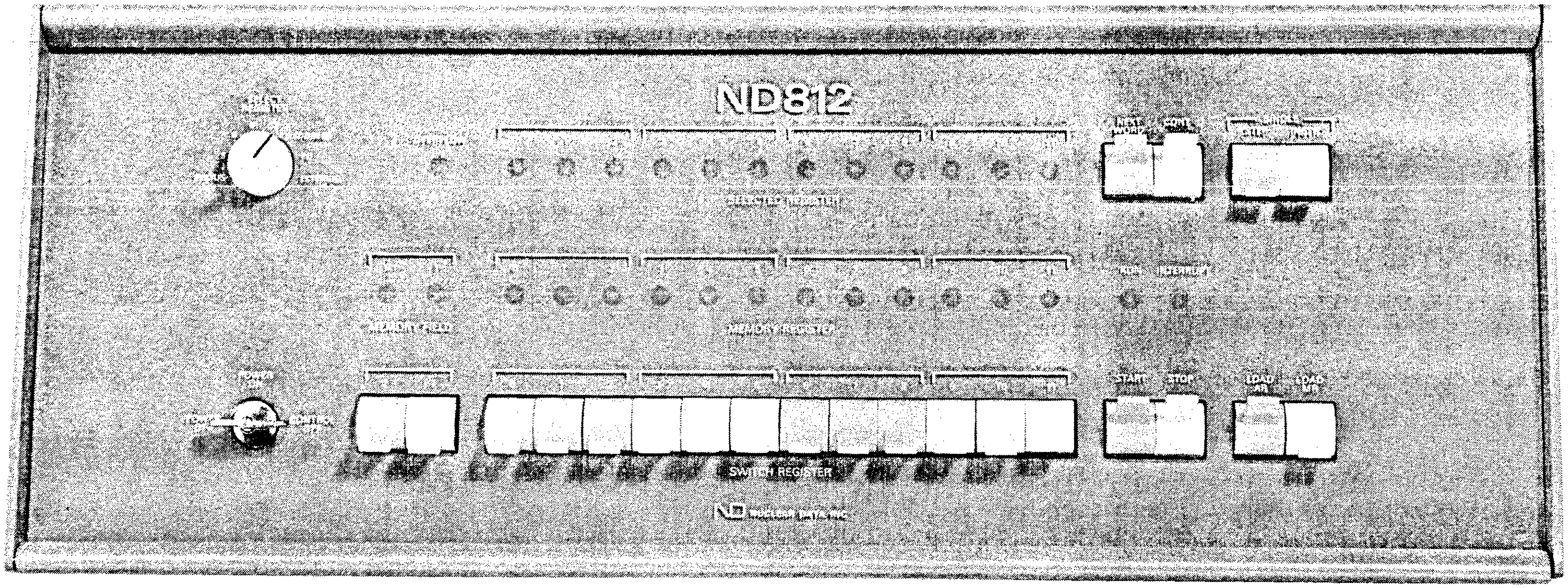


Figure 3-1. ND812 Operator Control Panel.

Table 3-1. Function of ND812 Operator Control Panel Controls And Indicators

Control	Function
POWER/CONTROL Key Switch	Controls application of primary power. In the POWER OFF position, all primary power is removed from the processor. In the POWER ON position, power is applied to all circuits and manual program control switches are enabled. In the CONTROL OFF position, power is maintained but all front panel switches are disabled except the SWITCH REGISTER, which can be monitored only by execution of a program.
Start Switch	When this switch is depressed, the interrupt is turned off, the overflow bit is set to 0, the flag is set to 0 and program execution is initiated at the memory location specified by the Program Counter. When the switch is released, the processor enters the run state, the contents of the Program Counter are transferred to the Address Register, and the contents of the Address Register are then used as the address of the first instruction of the program.
LOAD AR Switch	Depressing this switch loads the contents of the Switch Register into the Program Counter and Address Register, and updates the Memory Register and Memory Register indicator lamps to reflect the contents of core at the address contained in the Address Register. The Memory Field switches are loaded into the Memory Field bits as an extension of the Program Counter.
LOAD MR Switch	Depressing this switch loads the contents of the Program Counter into the Address Register, initiates a memory cycle that loads the contents of the Switch Register into the Memory Address specified by the updated Memory Register and transfers the Memory Register into core at the address in the Address Register. The Program Counter is then incremented by one. Memory Register indicator lamps will then display the deposit, and the Address Register indicator the deposit address. In this way it is possible to load consecutive core locations without continually resetting the Address Register from the Switch Register.

Table 3-1. Function of ND812 Operator Control Panel Controls and Indicators (Cont'd)

Control	Function
STOP Switch	Depressing this switch causes the processor to stop at the end of the current instruction. Program Counter contains the address of the next instruction after program termination.
SINGLE STEP Switch	When this switch is set up, the run mode is terminated and the timing circuits are disabled at the completion of one cycle (step) of the current instruction. Depressing CONTINUE causes the program to advance one additional cycle of the current instruction. Interrupt circuitry is disabled when a Single Step operation is performed.
SINGLE INSTRUCTION Switch	When this switch is set (up), execution is stopped at the end of each complete instruction. Depressing CONTINUE executes the next instruction in the logical sequence. DMA circuitry is disabled when a Single Instruction operation is performed.
NEXT WORD Switch	Depressing this switch sets the contents of the address specified by the Program Counter into the Address Register. The contents of the Program Counter are incremented by one. Reloading the Address Register generates a memory cycle which updates the Memory Register to equal the contents of memory at the address now contained in the Address Register. In this way it is possible to display the contents of consecutive memory locations without continually reloading the Address Register from the Switch Register.
CONTINUE Switch	Depressing and releasing this switch begins execution of the program at the address specified by the Program Counter. Start clear is not generated. NEXT WORD and CONTINUE switches are disabled when the processor is in the run mode.
SWITCH REGISTER Switches	These switches perm it manual loading of a 12-bit word into the registers. Words are arranged in an octal format with bit 0 representing the most significant bit. Switches in the up position correspond to binary "1's", down to "0's". The contents of the Switch Register are loaded into the Program Counter and Address Register by the LOAD AR Switch, into the memory by the LOAD MR Switch, or into the J Register during program execution with a LJSW instruction.

Table 3-1. Function of ND812 Operator Control Panel Controls and Indicators (Cont'd)

<table>
<tr><th>Control</th><th>Function</th></tr>
<tr><td>MEMORY FIELD Switches</td><td>These switches determine the Memory Field to be loaded as an extension of the Program Counter and Address Register. These switches affect only the Hardware Loader and "LOAD AR" operation. Each Memory Field is a 4K core memory and is numbered from 0 through 3, and represents 4096_{10} or 10000_8 memory locations (0000-7777_8).
<table>
<tr><th><i>Memory Field</i></th><th><i>Switch 0</i></th><th><i>Switch 1</i></th></tr>
<tr><td>0</td><td>off</td><td>off</td></tr>
<tr><td>1</td><td>off</td><td>on</td></tr>
<tr><td>2</td><td>on</td><td>off</td></tr>
<tr><td>3</td><td>on</td><td>on</td></tr>
</table></td></tr>
<tr><td>MEMORY FIELD Indicator Lamps</td><td>These lamps indicate the Memory Field in which the program is currently being executed. Lamps are numbered in a form identical to the Memory Field Switches.
<table>
<tr><th><i>Memory Field</i></th><th><i>Lamp 0</i></th><th><i>Lamp 1</i></th></tr>
<tr><td>0</td><td>off</td><td>off</td></tr>
<tr><td>1</td><td>off</td><td>on</td></tr>
<tr><td>2</td><td>on</td><td>off</td></tr>
<tr><td>3</td><td>on</td><td>on</td></tr>
</table></td></tr>
<tr><td>MEMORY REGISTER Indicator Lamps</td><td>These lamps indicate the 12-bit contents of the Memory Register at the location specified by the Address Register. A lamp "on" indicates a "1" and "off" indicates a "0", with "0" representing the most significant bit.</td></tr>
<tr><td>RUN Indicator Lamp</td><td>If this lamp is "on", it indicates that a program is in process.</td></tr>
<tr><td>INTERRUPT INDICATOR Lamp</td><td>If this lamp is "on", it indicates that one, or more, of the priority interrupt levels is enabled.</td></tr>
<tr><td>SELECTED REGISTER Indicator Lamps</td><td>These lamps indicate the contents of the register selected by the SELECT REGISTER Switch.</td></tr>
<tr><td>OVERFLOW INDICATOR Lamp</td><td>An overflow condition created by either a J or K Register arithmetic operation causes the overflow bit to be complemented. The overflow indicator lamp will illuminate when the overflow bit is non-zero.</td></tr>
</table>

Table 3-1. Function of ND812 Operator Control Panel Controls and Indicators (Cont'd)

Control	Function
SELECT REGISTER Switch	This switch selects the register to be displayed on the SELECTED REGISTER Indicator Lamps. Positions S, R, K, and J select the respective registers. Position PC selects the Program Counter; ADDRESS selects the Address Register; and STATUS selects the circuitry and signals listed below.

Indicator Lamp Number	*Logic Circuit Signal Name*	*Status*
0	FLAG	If this lamp is "on", it indicates that the flag is set.
1	OV	If this lamp is "on", it indicates that the overflow is set.
2 3	JPSMF0 JPSMF1	These lamps indicate the Memory Field in which is located the last JPS instruction which caused the program to branch to another memory field.

Memory Field	*Lamp 3*	*Lamp 2*
0	off	off
1	on	off
2	off	on
3	on	on

4 5	INTMF0 INTMF1	These lamps indicate the Memory Field in which execution was taking place at the time the last interrupt occurred.

Memory Field	*Lamp 5*	*Lamp 4*
0	off	off
1	on	off
2	off	on
3	on	on

Table 3-1. Function of ND812 Operator Control Panel Controls and Indicators (Cont'd)

Control	Function		
	6	IONN	If this lamp is "on", it indicates that the lowest level priority interrupt circuitry is activated, as well as the A, B, and highest level interrupt circuitry.
	7	IONB	If this lamp is "on", it indicates that the B level and highest level priority interrupt circuitry is activated.
	8	IONA	If this lamp is "on", it indicates that the A level and highest level priority interrupt circuitry is activated.
	9	IONH	If this lamp is "on", it indicates that the highest level priority interrupt circuitry is activated.
	10 11	MF0 MF1	These lamps indicate the Memory Field in which the program is currently being executed, (actual extension of the Program Counter and Address Register).

Memory Field	*Lamp 10*	*Lamp 11*
0	off	off
1	on	off
2	off	on
3	on	on

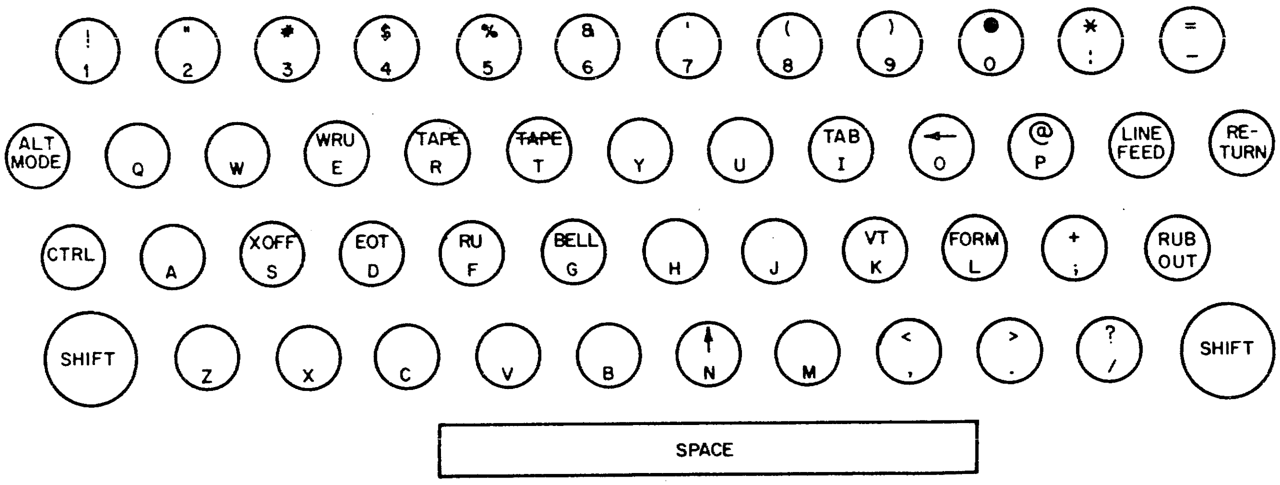


Figure 3-2. ASR-33 Teletype Keyboard.

Table 3-2. Function of ASR33 Teletype Controls

Control	Function
REL. pushbutton	Disengages the punch mechanism allowing tape removal or tape loading.
B. SP. pushbutton	Backspaces the tape in the punch by one space, allowing manual correction or rub out of the character just punched.
OFF and ON pushbuttons	Control use of the tape punch with operation of the Teletype keyboard printer.
START/STOP/FREE switch	Controls use of the tape reader with operation of the Teletype. In the FREE (lowest) position the reader is disengaged and can be loaded or unloaded. In the STOP (center) position the reader mechanism is engaged but de-energized. In the START (highest) position the reader is engaged and operated under program control.
Keyboard	Provides a means for printing on paper; similar in operation to a typewriter. Also perforates tapes when the punch ON button is depressed, and supplies input data to the computer when the LINE/OFF/LOCAL switch is in the LINE position. For a complete description of the keyboard refer to ASR-33 manual. (See Figure 3-2.)
LINE/OFF/LOCAL switch	Controls application of primary power in the Teletype and data connection to the processor. In the LINE position the Teletype is energized and connected as an I/O device of the computer. In the OFF position the Teletype is de-energized. In the LOCAL position the Teletype is energized for off-line operation, and signal connections to the processor are broken. Both LINE and LOCAL use of the Teletype require that the computer be energized through the POWER switch, if the TTY is powered through the utility outlets on the rear of the computer.

Two controls are extremely important to the maintenance technician. These permit partial machine cycles to manually execute, or permit instructions to be processed one at a time on command from the operator.

3.3 BINARY LOADER

One of the possible configurations (Figure 3-3) for the ND812 may include a high speed paper tape reader (HSR), a low speed teletype paper tape reader (TTY) and three magnetic tape cassette readers (Cass).

Each paper tape contains only one program. Each cassette can contain many programs. Each program in a cassette is addressed by a unique 7-bit identifying tag word.

To instruct the ND812 as to the mode of loading used, a 12-bit status word is entered into the switch register early in the loading procedure. The first two bits of the status word (0, 1) indicate which device is to be used. If the content of the first two bits is 01, the high speed reader is selected. If the content is 00 or 10, one of the cassettes will be used. If 11, a low speed reader is selected.

The remaining 10 bits are applicable only when a cassette program is selected for loading into the ND812. Bits 2, 3 and 4 determine which of the three cassettes is selected. If the content of bits 2, 3 and 4 is 001, cassette drive 1 is selected. If it is 010, drive 2 is selected, and if it is 100, drive 3 is selected. The remaining bits (5-11) name any one of 128 possible programs on the selected tape cassette. This tagword has an octal value ranging from 0000 to 0177. The status word is manually loaded into the switch register and is held there until required by the loader program. There are two loader programs, the bootstrap and the binary loader.

The Binary Loader (ND41-0005) is a short program designed to load binary formatted programs or data into the computer. The software loader used is the Binary Loader. An optional hardware method of loading binary formatted paper tapes is also available.

The Binary Loader reads binary formatted paper tape or cassette stored program records into the Memory Field specified by field change bits on the input source tape. Each Memory Field consists of 4096 core locations and is defined as Memory Field 0 core locations 00_2 0000_8 through 00_2 7777_8. Memory Field 1 core locations 01_2 0000_8 through 01_2 7777_8, Memory Field 2 core locations 10_2 0000_8 through 10_2 7777_8 and Memory Field 3 core locations 11_2 0000_8 through 11_2 7777_8. The Binary Loader may reside in any Memory Field, but unless the binary input tape has field change bits it will only read into the field in which it resides. Since the Binary Loader is itself a program, some means must be provided to load it into memory. This may be accomplished with a Bootstrap, or the Teletype/Auto Loader. The Bootstrap is a very short program which may be manually loaded from the Switch Register. When executed, the Bootstrap loads enough of the Binary Loader to allow the Binary Loader to complete loading itself. The bootstrap is destroyed in the process so that if it is necessary to reload the Binary Loader, the Bootstrap must first be reloaded.

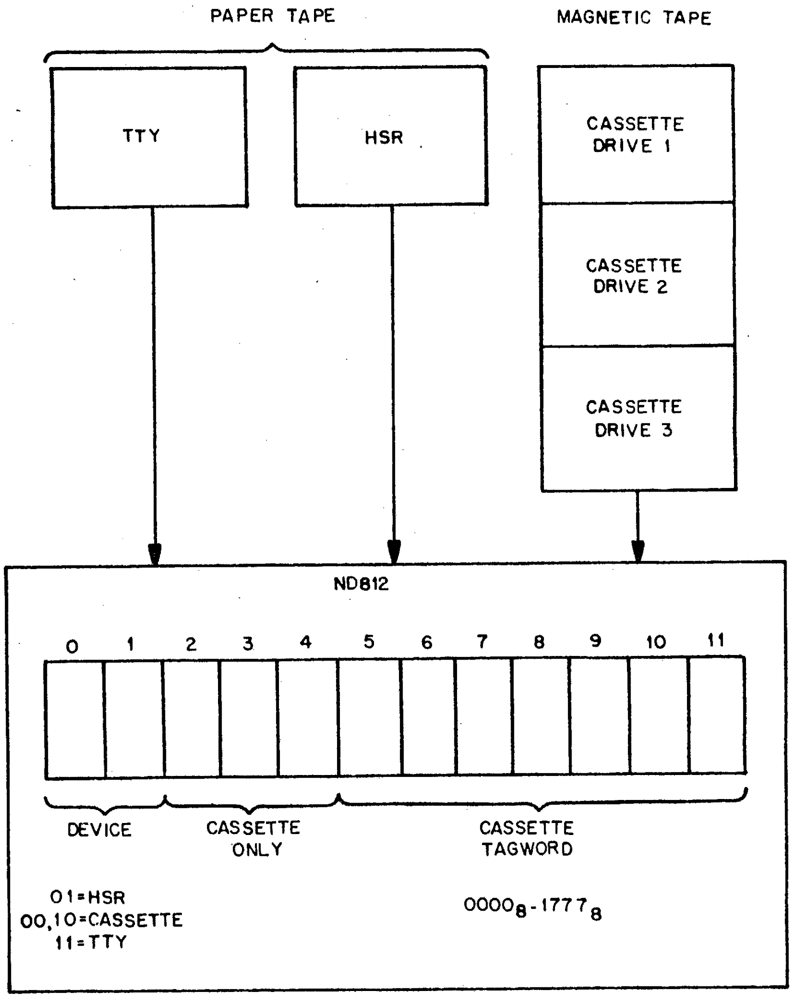


Figure 3-3. Switch Register Status-Word Format

3.4 MANUAL LOADING

In loading a program from the paper tape reader, the Binary Loader begins by reading the leader. Actual loading of the processor begins when the Binary Loader detects the first character different from 0200_8 (eight level punch only). For this reason, it is essential that a program tape be placed in the reader with the leader at the read station. Should the program tape be placed in the reader with blank tape at the read station, the Binary Loader will begin loading zeros into memory beginning with location 0000_8. The actual leader of the program record will be interpreted as trailer.

The loading process consists of assembling consecutive pairs of frames using levels one through six as upper and lower halves of 12-bit words. The assembled 12-bit words are stored in memory in consecutive locations, beginning at address, as determined by the presence, and interpretation of the origins, on the paper tape. An origin is a 12-bit word punched in two frames of six bits each which is interpreted by the binary loader as an address at which to begin storing programs. An origin is distinguished from program words on the binary tape by the presence of the seven level punch. Field change characters, special characters used to indicate the field in which a program is to be stored as it is loaded, cause the storage to take place in memory fields specified by the field change characters. When a binary tape is created, the last two frames are written as a pseudo origin in such a way as to make the sum of all the 12-bit words (including the last) on the tape equal to zero. The binary loader keeps a running sum or checksum of the 12-bit words on the tape. When the loader detects the trailer it halts. If the checksum is zero, the loading process is assumed to be correct and K register equal to zero. If the K register is non-zero, the loading process may be assumed to be in error.

3.4.1 AUTO-START

There are three alternative methods of using the Binary Loader. In all cases the format of the record being loaded remains the same. The differences lie in the manner of entering the Binary Loader and exiting once the loading process has been completed. In all three of the cases to be described, location 7751_8 contains the exit address. That is, instead of stopping when the loading process is complete, the Binary Loader performs a jump to the address contained in location 7751_8, provided the contents of location 7751_8 is not zero. This feature will be described as the Auto-start feature of the Binary Loader.

Manual Load with Auto-start is the simplest use of the Auto-start feature of the Binary Loader. The program record being loaded includes the necessary coding to cause location 7751_8 to be set equal to some non-zero address. When the Binary Loader completes the loading process, it will jump to this address with the checksum, (normally zero), in the K register. The exit address should be the starting address of the program being loaded. The program making use of the Auto-start feature should check the K register to determine if the loading process was correct and take appropriate action should the J register be non-zero.

Second use of the Auto-start involves performing a JPS to the Binary Loader from a program outside the Binary Loader. The JPS is performed to location 7751_8. When the loading process is completed the Binary Loader will return to the calling location plus one with the checksum in the K register, as though the Binary Loader were a sub-routine.

The third method of using the Auto-start feature is very similar to the second method but rather than a JPS, the calling program performs a JMP to location 7752_8. The control word must have been loaded into the J register by the calling program. It is necessary that the calling program either set location 7751_8 to zero, thereby causing the Binary Loader to stop at the end of the loading process (provided the program being loaded does not alter the location 7751_8) or the calling program should set an appropriate exit address into location 7751_8.

3.5 OPERATOR OR USER CONTROL

The control word is the only control the user has over the loader. The bits of the control word are interpreted by the loader as follows: BITS 0 and 1 determine the input device, BITS 2, 3 and 4 determine the input cassette drive if the cassette was selected, and BITS 5 through 11 indicate the tagword when loading from magnetic tape cassette. Each one of these three functions is described in detail below.

Input Drive Selection

BITS 0 and 1 determine the device from which the record is to be read. If BIT 1 is "0", input is from cassette and BIT 0 is ignored. If BIT 1 is "1", input is from one of the paper tape readers as controlled by BIT "0". When BIT 0 is "0", the input is from the high speed reader and if BIT 0 is "1", the input is from low speed or teletype reader.

Cassette Drive Selection

BITS 2, 3 and 4 permit the user to select one of the three cassette drives for input. No two of these bits should be on together as the loader will try to read from two or more drives simultaneously.
BITS 2, 3 and 4 permit the user to select one of the three cassette drives for input. No two of these bits should be on together as the loader will try to read from two or more drives simultaneously.
BIT 2 set to "1" selects drive three, BIT 3 selects drive two, and BIT 4 selects drive one.

Tagword Selection

Any tagword from 0000_8 to 0177_8 may be selected by BITS 5 through 11, Capacity to select a tagword allows the user to select at random one of many records on a particular cassette without the need to hunt manually for the record in question. Starting with the beginning of the cassette, the Binary Loader will search for the correct tagword by reading the first character of each program record on the tape.

NOTE

Entering the Binary Loader under software control demands that the J register be set to the desired control word by the software performing the call to the Binary Loader (as described above).

3.6 TELETYPE/AUTO LOADER

The Teletype /Auto Loader is a hard-wired interface that allows loading of finary format-fed paper tapes via Teletype Reader with a minimum of manual control selections. The Binary Load (ND41-0005) is not required when using the Teletype/Auto Loader for loading paper tapes via Teletype. However, the Binary Loader is required when loading programs with the High Speed Reader or Tape Cassette.

The procedure for loading binary formatted paper tapes (including the Binary Loader) via Teletype is as follows:

a. Set Teletype LINE/OFF/LOCAL Switch in LINE position, and START/STOP/FREE switch in FREE position.

b. Place paper tape in ASR-33 reader, and set START/STOP/FREE switch in START position.

c. Simultaneously depress ND812 Computer LOAD ADDRESS and NEXT WORD switches. The paper tape is read-in and will stop on trailer.

3.7 BOOT STRAP PROGRAMS

There are two Bootstrap programs available for loading the Binary Loader into the ND812 Computer; one is used for the Teletype, and the other is used for the High Speed Reader. The Bootstrap program is toggled into the ND812 Computer via the SWITCH REGISTER as described in the following paragraphs.

3.7.1 TELETYPE BOOTSTRAP LOADING PROCEDURE

Normally, programs are loaded into the ND812 Computer utilizing the Teletype/Auto Loader procedure described in Paragraph 3.6. The following Teletype Bootstrap procedure provides a method for loading the Binary Loader in case the Teletype/Auto Loader is in-operative.

a. Set the ND812 Computer key switch in the POWER ON position.

b. Set the SWITCH REGISTER 7762_8, set both MEMORY FIELD switches in the DOWN position, and depress LOAD AR.

c. It is now necessary to load fourteen instructions from the SWITCH REGISTER, each of which is followed by lifting the LOAD MR key. The address is automatically incremented each time the LOAD MR Key is depressed.

Address	Instruction	Address	Instruction
7762	7404	7771	7404
7763	6101	7772	6101
7764	7403	7773	7403
7765	1146	7774	1122
7766	1501	7775	5700
7767	6105	7776	6114
7770	1101	7777	7745

d. Set the SWITCH REGISTER to 7773_8 and depress LOAD AR twice.

e. Place the Binary Loader Paper Tape (ND41-0005) into the ASR-33 reader on leader, not blank tape. Set Teletype LINE/OFF/LOCAL switch to LINE, and START/STOP/FREE switch to START.

f. Depress ND812 Computer START Switch. The paper tape is read and will stop on reaching trailer. Set the ND812 SELECT REGISTER switch to J position and check SELECTED REGISTER indicator lamps for zero. Repeat the above procedure from step b if J register is is non-zero.

The Binary Loader is now in memory and may be used to load programs from either paper tape or cassette.

3.7.2 HIGH SPEED READER BOOTSTRAP LOADING PROCEDURE

Normally, the Binary Loader is loaded into the ND812 Computer via the Teletype as described in Paragraph 3.6. The following High Speed Reader Bootstrap procedure provides an alternate method in case the Teletype is not available or inoperative.

a. Set the ND812 Computer key switch in the POWER ON position.

b. Set the SWITCH REGISTER to 7762_8, set both MEMORY FIELD switches in the DOWN position, and depress LOAD AR.

c. It is now necessary to load fourteen instructions from the SWITCH REGISTER, each of which is followed by lifting the LOAD MR key. The address is automatically incremented each time the LOAD MR key is depressed.

Address	Instruction	Address	Instruction
7762	7424	7771	7424
7763	6101	7772	6101
7764	7423	7773	7423
7765	1146	7774	1122
7766	1501	7775	5700
7767	6105	7776	6114
7770	1101	7777	7745

d. Set ND812 SWITCH REGISTER to 7773_8 and depress LOAD AR.

e. Place the binary paper tape of Binary Loader (ND41-0005) into the high speed reader with the leader at the read station.

NOTE

It is extremely important that the leader is positioned at the read station for a punched paper tape loading operation. The leader, in most cases, is approximately six inches in length and is set back approximately two feet from the beginning of tape .

f. Depress START key. The paper tape will move through the reader a short distance and then stop. At that moment depress STOP key.

g. Set ND812 SWITCH REGISTER to 7762_8 and depress LOAD AR.

h. Set ND812 SWITCH REGISTER to 7424_8 and raise LOAD MR.

i. Depress ND812 CONTINUE key. The paper tape is then completely read and the reader will stop when tape trailer is sensed. Set the ND812 SELECT REGISTER switch to J position and check SELECTED REGISTER indicator lamps for zero. Repeat above procedure from step b if J is non-zero.

3.8 GENERAL LOADING PROCEDURES

The following paragraphs provide procedures for loading programs into the ND812 Computer using the Teletype Low Speed Paper Tape Reader, or High Speed Paper Tape Reader, or Magnetic Tape Cassette.

3.8.1 LOADING PROCEDURE USING TELETYPE

a. Depress ND812 Computer STOP switch.

b. Set Teletype START/FREE/STOP switch in FREE position.

c. Place paper tape in ASR-33 reader, and set START/STOP/FREE switch in START position.

d. Simultaneously depress ND812 Computer LOAD ADDRESS and NEXT WORD switches. The paper tape is read-in and will stop on trailer. Set ND812 Computer SELECT REGISTER switch to K position, and check SELECTED REGISTER indicator lamps for zero. Repeat the above procedure from step a if K register is non-zero.

3.8.2 LOADING PROCEDURE USING HIGH SPEED PAPER TAPE READER.

NOTE

The Binary Loader (ND41-0005) must be loaded into the ND812 Computer before binary formatted programs can be loaded using the High Speed Reader. Load the Binary Loader via Teletype per paragraph 3.8.2. If the Teletype is not available, use the bootstrap procedure described in paragraph 3.7.2.

a. Set the ND812 Computer SWITCH REGISTER to 7700_8 and depress LOAD AR twice.

b. Set control word 3700_8 into ND812 SWITCH REGISTER.

c. Place the binary formatted program tape to be read into the High Speed Reader with the leader at the read station.

d. Depress ND812 Computer start switch. The Binary Loader will read the program tape into the computer, and the reader will stop when tape trailer is sensed. Set the ND812 SELECT REGISTER switch to K position, and check SELECTED REGISTER lamps for zero. Repeat the above procedure from step a if K register is non-zero.

3.8.3 LOADING PROCEDURE USING MAGNETIC TAPE CASSETTE

NOTE

The Binary Loader (ND41-0005) must be loaded into the ND812 Computer before binary formatted programs can be loaded from the Magnetic Tape Cassette. Load the Binary Loader via Teletype per paragraph 3.8.2.

a. Set the ND812 Computer SWITCH REGISTER to 7700_8 and depress LOAD AR twice.

b. Set Switch Register BITS 0 and 1 to "0", set BITS 2, 3 and 4 for the desired cassette drive, set BITS 5 through 11 to desired tagword (refer to paragraph 3.5).

c. Depress START on the computer.

d. The Binary Loader will read from the cassette and conduct a tagword search. When the tagged record is reached, the Binary Loader will read the record and stop at completion. If the content of the K register is zero, the loading process was correct. If the content of the K register is non-zero, re-start from Step a.

NOTE

If the Binary Loader proceeds to the end of tape without stopping, the user has specified a tagword which does not exist on the cassette. To recover from this condition, depress STOP and restart from Step a, and ascertain that the tagword specified exists on the cassette.

3.9 ERROR DIAGNOSTICS

The checksum is stored in the J register at the completion of a hardware autoload from ASR-33 or in the K register for the binary loader controlled program loading procedure. A NON-ZERO J or K register indicates an erroneous load. Refer to the appropriate section of the OPERATIONAL PROCEDURE and re-load the program. If an error is encountered under binary loader autostart, check the calling program and the exit address of the Binary Loader.

If a non-existent input device is specified, the Binary Loader will enter an endless loop. The processor must be stopped with the front panel STOP switch and the loader restarted.

Failure to position a tape with leader over the read station will cause the leader to stop on reaching the program's leader, if it is positioned on blank tape preceeding the leader. The J register will be zero, but since the program was not read, it will not be loaded. The user is particularly warned against placing blank tape at the read station when loading overlays, as part of the background program is likely to be lost when the loader tries to load the blank tape.

SECTION IV
THEORY OF OPERATION

4.1 GENERAL

The various functional units of the ND812 have been described in the overview in Section I. Each of these functional units is further detailed in the following descriptions. In some cases a functional unit may consist of a method of performing some function such as the execution of an instruction, or the method employed in obtaining a run condition. In other cases a functional unit may comprise a unique functioning logic group such as the pulser control and pulser logic which is described under clock generation and timing. Each of the following descriptions has a block diagram explanation, a fundamental operation table, and timing diagrams.

4.1.1 BLOCK DIAGRAM DESCRIPTIONS

The block diagram descriptions present an overview at the functional level of operation of a circuit or circuit group. Throughout these descriptions the ND812 logic diagrams contained in Section VII of this manual are called out by sheet number and zone location. References to these diagrams are simply identified by a number-letter-number combination, 9B4, for example. The first number references the sheet number of the 28-sheet logic diagram. If the first number is preceded by the letter M, the reference is to the 6-sheet or the 10-sheet memory-control logic diagrams, 6M referring to the 6-sheet memory logic diagram and 10M referring to the 10-sheet memory logic diagram; thus, the reference 10M7A4 refers to the 10-sheet memory logic diagram, the seventh sheet, zone A4. The reference 18B1 refers to the 28-sheet ND812 logic diagram, sheet 18, zone B1.

4.1.2 SIMPLIFIED LOGIC TABLES

The simplified logic is presented in the fundamental operation tables so that all logical functions needed to assert a described function or operation can be easily traced without having to search all the logic diagrams. They are work savers so that needless time is not spent in searching for all the logical functions required to obtain a given operation. For example, when an instruction is being processed, certain events occur in a precisely ordered sequence during the basic phase and other events occur during the execute phase. The simplified logic for the instruction indicates all the signals needed to obtain the end function. For example, often when a basic phase is processed the content of the MX multiplexer is transferred through the adders to the AR at period PU0B. The event is symbolized in the following manner.

1. ↑PU0B→↓PEA*→↑PEA/9B1
2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = MX→ADDER→AR

The symbological shorthand should be interpreted as follows: a high-level PU0B signal produces a low-level PEA* signal, which in turn produces a high-level PEA signal shown at 9B1 of the logic diagrams in Volume II. The second line should be interpreted in a similar manner: the high-level PEA signal is ANDed with the next high-level REGCLK signal to produce the low-level CPA* signal at 9B2. This results in the output of the MX multiplexer being switched through the adders to the AR. That these signals are required to obtain the specified logical function of switching MX multiplexer output data into the AR can be confirmed simply by tracing through the logic. Notice that the symbology does not indicate which MX multiplexer input is selected as the switched output. This is because a different pulse is used to select the MX multiplexer input data although both events may occur during the same time period, PU0, BP0, or EP0. However, this event is also described symbologically (refer to Table 4-2). Notice also that no distinction is made between levels and pulses. In the example, the REGCLK signal is the next register clock pulse produced by the clock logic during the time signal PEA is high. The output, low level signal CPA*, is shaped by the REGCLK signal because REGCLK has the shortest duration. Familiarity with the clock generator and timing circuits will eliminate any confusion about pulse inputs to the various logic elements.

4.1.3 TIMING DIAGRAMS

The timing diagrams are idealized waveforms that illustrate the signals required, during each discrete time period, to obtain the end function described in the narration or in each of the simplified logic tables. Both the simplified logic tables and the timing diagrams should be used when analyzing any of the logical functions that must be obtained during a given operation or process. That the various waveforms are produced as the result of a given operation can be verified by sampling signal outputs at the indicated logical outputs.

4.1.4 DISCRETE LOGIC ELEMENTS

With the exception of discrete component circuits on the power supply and the memory boards, the smallest complete logic element in the ND812 is an integrated circuit (IC). Each of the various integrated circuits combines one or more logic functions in replaceable dual in-line packages. These various logic functions include standard NAND, NOR, and inverter gates, shift registers, binary to octal decoders, adders, multiplexers, counters, latches, flip-flops, and one shots. Functional description for medium scale integrated circuits (MSI' s) used in the ND812 Computer, are provided in Section VII as a reference. It is assumed that the reader is conversant with all these various logic elements and the purpose for which they are included to obtain various functions. It is further assumed that the reader is familiar with binary arithmetic, 2's complement notation, and that machine language programming is no deep mystery.

4.2 ENERGIZING THE ND812

When the ND812 is turned on, the key-lock switch is placed in the POWER ON or the CONTROL OFF position. When the switch is placed in the POWER ON position, the ND812 power circuits are energized and the power sense circuits operate. When the key-lock switch is placed in the CONTROL OFF position, those controls that permit program modification from the front panel control switches are disabled by the removal of a switch ground signal (KEY SW GND). As a condition of power application the power sense logic operates two latches in the power supply circuits. One of these senses any power loss whether they are caused by an external or internal failure. The other latch provides a delayed power-on signal to various logic elements throughout the ND812 to obtain an initial register and latch clearing operation.

4.2.1 POWER SENSE

Table 4-1 lists the simplified logic for the power sense and power fail circuits. When power is first turned on, +5-volt and +20-volt power (sheet 19) is applied to the power sense circuit. The power sense transistor bias circuit, consisting of a 2.32K and 1K resistor, places a slightly positive voltage at the base of transistor Q6(19A6) with respect to +5-volt level at the emitter. This bias condition assures that transistor Q6 is cut off and the input to the NAND gate 3A is a low-level signal. When a loss in power occurs, the +20-volt power source loses its voltage level before the +5-volt power source, and the bias voltage at the base of transistor Q6 becomes more negative than its emitter. Transistor Q6 saturates and the signal at the input to NAND gate 3A becomes high-level.

Table 4-1. Latch and Register Clearing, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	T'ON	PWR SENSE	
		1. ↓BDPWR, ↑BDPWR*/19A4	Power up condition delayed 1 second after turn-on to stabilize logic.
		2. ↓BDPWR + ↓RUN*→↑PWRDY/19A4, ↓PWRDY (initial condition)	
		3. ↑GO→↓RUN*/19A4	
		4. ↓BDPWR + ↓RUN*→↑ENM1/19B4	
		5. ↑ENM1→↓PWRDY/19A4, ↑PWRDY* (delayed → 1 second)	
B.	ANY	PWR FAIL	
		1. ↓BDPWR*→↑BDPWR/4A1	Power loss inhibits clock and halts computer. Halt occurs in less than 1 microsecond.
		2. (↑BDPWR) (↑GO*)→↓BLKMC*/4A1 = STOP→CLOCK GEN	
		3. (↑BDPWR) (↑PRINT*)→↓RGPWR*/3A2	
		4. ↓RGPWR*→↑RGO/3A2	
		5. (↑CPPU) (↑DONE) (↑PU6B) (↑GO)→ ↓FFCLK*→↑FCCLK/3A2	
		6. (↑FFCLK) (↑RGO)→↓RGO*/3A3 = CLR→GO (↓GO, ↑GO*/3A3)	

Table 4-1. Latch and Register Clearing, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
C.	T'ON	CLEAR LATCHES AND REGISTERS 1. ↓PWRDY*→↑PWRDY/3B2 2. ↑PWRDY→↓STCLR*/3B2	The start-clear signal is produced at turnon and start.
D.	T'ON START, LOAD AR, LOAD MR, NEXT WORD PU1	CLR→CYCLE STEAL AND INTERRUPT REGS 1. ↓START*→↑START/3A1 2. (↑START) (↑GO*)→↓STCLR*/3B2 3. ↓STCLR* + ↑EXPMR* + ↑PU1*→↑STPU1/3A3 4. ↑STPU1→↓STPU1*/3A3 5. ↓STPU1* = CLR→CYCLE STEAL REG (↓CSP/4B1, ↑CSP*) 6. ↓STPU1* = CLR→INTERRUPT REG (↓INTP/3B4, ↑INTP*)	Cycle steal and interrupt registers closed.
E.	T'ON START	CLR→POWER INTERRUPT 1. ↓STCLR*→↑STCLR/3B3 2. (↑STCLR) (↑PULSKP)→↓RPRIN* = CLR→POWER INTERRUPT LATCH (↑PRINT/3A1↓PRINT*)	Enables low power condition to stop ND812 when in run state.
F.	T'ON START	CLR→DOUBLE TIME PERIOD REG 1. ↓STCLR*→↑STCLR/3B3 2. ↑STCLR→↓RDBLE*/4A4 = CLR→ START DOUBLE and DOUBLE REGS (↓STDBL/4A4, ↑STDBL*) and (↓DBLE/4A4, ↑DBLE*)	Clears PCP control registers.
G.	T'ON, START	CLR→INSTRUCTION REG 1. ↓STCLR*→↑STCLR/3B3 2. ↑STCLR→↓MRI* = CLR→IR (↓I00 - ↓I11/17B1, 2, 3)	Instruction register cleared.
H.	T'ON, START	CLR→FLAG REG 1. ↓STCLR*→↑STCLR/3B3 2. ↑STCLR→↓RFLAG* = CLR→FLAG REG (↓FLAG/10B4, ↑FLAG*)	Flag register cleared.
I.	T'ON, START	CLR→PULSER 1. ↓STCLR*/5A1 = CLR→PULSER (↓PU0 - ↓PU7)	Pulser cleared.
J.	T'ON, START	CLR→EXECUTE PHASE 1. ↓STCLR*/3B3 = CLR→EXECUTE PHASE (↓EPH/5B2)	Execute phase disabled.

Table 4-1. Latch and Register Clearing, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
K.	T'ON, START	CLR→IND LATCH 1. ↓STCLR*/3B3 = CLR→INDIRECT LATCH (↓IND/6A2, ↑IND*)	Indirect latch
L.	T'ON, START	CLR→OVERFLOW REG 1. ↑CLOV* + ↓STCLR*→↑ROV/11B4 2. ↑ROV→↓ROV* = CLR OV (↓OV/11B4, ↑OV*)	Overflow register cleared.
M.	T'ON, START	CLR→PRIORITY REG 1. ↓STCLR*/3B3 = CLR→PRIORITY REG (↓ION/11B3, ↓IONA, ↓IONB, ↓IONN)	Priority structure disabled.
N.	T'ON, START	CLR→INTERRUPT ENABLE LATCH 1. ↓STCLR* = CLR→INTERRUPT ENABLE (↓ENINT/11B2, ↑ENINT*)	Interrupt enable cleared.
P.	T'ON, START, DONE	ENABLE INSTRUCTION-CLEARED LATCHES 1. ↓STCLR*→↑STCLR/3B3 2. ↑STCLR + (↑DONE) (↑PU7B)→↓STDN7*/3B3	All instruction-cleared registers and latches require that the done latch is set and pulse PU7 has occurred.
Q.	T'ON, START, DONE	ENABLE→J REG LATCH 1. ↓STDN7* = CLR→J, K LATCH (↑JCOM/3A4, ↓KCOM)	Sets J register for two-word data transfer.
R.	T'ON, START, DONE	CLR→FIRST/LAST MEMORY LATCH 1. ↓STDN7*= CLR FLMEM LATCH (↑FLMEM*, ↓FLMEM/3B4)	Clears first/last memory latch.
S.	T'ON, START, DONE	CLR→FIRST/LAST LOCATION LATCH 1. ↓STDN7* = CLR→FLLOC LATCH (↑FLLOC*/3B4, ↓FLLOC)	Sets auto-index for first location.
T.	T'ON, START, DONE	CLR→EXECUTE LATCH 1. ↓STDN7* = CLR→XCTFF (↑XCTFF*/6A3, ↓XCTFF)	Clears execute latch.
U.	T'ON, START, DONE	CLR→INDIRECT LATCH 1. ↓STDN7* = CLR→INDFF (↑INDFF*/6A3, ↓INDFF)	Clear indirect latch.

Table 4-1. Latch and Register Clearing, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
V.	T'ON, START, DONE	CLR→INDIRECT MEMORY LATCH 1. ↓STDN7* = CLR→INDM (↑INDM*/6A3, ↓INDM)	Clear indirect memory latch.
W.	T'ON, START, DONE	CLR→KOM REG 1. ↓STDN7* = CLR→KOM REG (↓KOM/8B1)	Clear K00 set register (multiply logic).
X.	T'ON, START, DONE	CLR→WAIT REG 1. ↓STDN7* = CLR→WAIT REG (↑WAIT*/8B3, ↓WAIT)	Clear wait register (divide logic).
Y.	T'ON, START, DONE	CLR→ONCE REG 1. ↓STDN7* = CLR→ONCE REG (↑ONCE*/8B3, ↓ONCE)	Clear once register (divide logic).

With a low-level input signal to NAND gate 3A (19A4), the BDPWR latch provides the outputs listed in reference A of Table 4-1. These levels tend to make the output of NAND gate 4B (19A4) and 3C (19A4) low level and high level, respectively. However, the large capacitor connected to output of NAND gate 3C tends to keep its output at a low level and thus, for approximately 1 second, the outputs listed in entry A2 of Table 4-1 are obtained. When the capacitors in these NAND gates are charged and discharged, the outputs listed in A5 of Table 4-1 are obtained. This is the normal state of these latches.

At the outset of power turn on, the high-level PWRDY signal provides initial latch and register clearing signals. These are listed in entries A and B, of Table 4-1. These same clearing signals are obtained when the ND812 is placed in its run state by depressing and releasing the START switch.

4.2.2 MANUAL START AND CONTROL

The ND812 front panel is shown in Figure 3-1. Its controls and indicators provide a means to manually load a program into core memory, examine the contents of various locations in memory, alter memory contents, and to determine the current status of a program. The ND812 can be started and stopped through its front panel control switches, and, after stopping, can be restarted. Before beginning the descriptions in this paragraph, the reader should study Section III of this manual.

4.2.2.1 SWITCH REGISTER. The outputs of the switch register switches (2A2/2B2) are applied directly to the utility gates (sheets 14, 15 and 16) and through the utility gates to the MX multiplexer so that any setting of the switch register switches are applied directly to the utility gates.

4.2.2.2 GO CONTROL LOGIC. The GO control logic permits the ND812 run condition to be obtained. When manual operation such as loading registers or examining registers is occurring, the GO condition is temporarily inhibited until the START switch is operated. The GO control logic has two states, initialize and run. The initialize condition is dependent on the operation of the power sense and power delay latches.

4.2.2.2.1 Initialize Condition. The GO control logic consists of the go and the start pulser registers. (The start pulser register is described in subsequent paragraphs.) Operation of the ND812 is initialized when the power sense and power delay logic (19A4 and Table 4-1) is exercised. When the front panel power switch is turned on the power sense logic detects +20 Vdc power through transistor Q6 on the power supply and oscillator board. When the +20 Vdc is sensed, the power sense latch is disabled. If a low-power condition exists, the bias voltage on the sensing transistor drops causing it to conduct and set the bad power BDPWR latch. With the power sense latch in its unset condition the power delay latch normally provides a high-level PWRDY* signal and a low-level PWRDY signal. After about a one-second delay this circuit is set to its normal state. The delay permits the power supply outputs to stabilize and the GO control register (3A3) to be reset through logic by the temporarily high-level PWRDY signal applied through the gate logic. When the PWRDY latch is set to its normal operation the PWRDY signal becomes low level together with all the other gated signals, and the GO control register is initialized with the GO signal low-level and the GO* signal high level (Table 4-2).

Table 4-2. GO Control, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	T'ON	GO CONTROL, INITIALIZE→↓GO/3A3,↑GO*/3A3	
		1. ↓PWRDY*→↑PWRDY/3B2	Occurs at turnon and start.
		2. ↑PWRDY→STCLR*/3B2	
		3. ↑PWRDY→↓RGO*/3A3	
		4. (↓RGO*) (↑CNTRP)→↓GO/3A3, ↑GO*	
B.	T'ON	GO CONTROL, RUN→↑GO/3A3,↓GO*/3A3	
		1. ↑XLDPR*→↓EXLDPR/6B1	Occurs at start and continue.
		2. (↓EXLDPR) (↑GO*)→↑LDPR*/6B1	
		3. ↑XLDMR*→↓EXLDMR/6B1	
		4. (↓EXLDMR) (↑GO*)→↑LDMR*/6B1	
		5. ↑EEXAM*→↓EXEXAM/6B1	
		6. (↓EXEXAM) (↑GO*)→↑EXAM*/6B1	
		7. (↑LDPR*) (↑LDMR*) (↑EXAM*)→ ↓EXPMR→↑EXPMR*/6B2	
C.	T'ON	ENABLE TOGGLE MODE	
		1. (↑EXPMR*) (↑CNTRP) (↑ALODT*) (↑RGO*)→TOGGLE IndRGO*/3A3	Occurs at start and continue.
		2. ↓START*→↑STCNT/3A3	
		3. ↑STCNT→↓CLKGO*→↑GO/3A3,↓GO*	

Table 4-2. GO Control, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
D.	PU0	EXECUTE AND BASIC PHASES BLOCKED 1. ↓GO + ↓RDBLE*→↑RDBLE/4B1 2. (↑RDBLE) (↑CSP*) (↑SDCS*)→ ↓RCSFF*/4B2 3. (↓RCSFF*) (↑CSP*) = CLR→CYCLE STEAL FF (↓CSFF/4B2, ↑CSFF*) 4. (↑INTP*) (↑RDBLE)→↓RITFF*/4B4 5. (↓RITFF*) (↑INTP*) = CLR→INTFF (↓INTFF/4B4 ↑ INTFF*) 6. (↑INTFF*) (↑CSFF*)→↓INTCSF/4B4 7. ↓INTCSF→↑INTCS*/5B1 8. (↑INTCS*) (↑PU0) (↓GO)→↑RBLK*/5B2 9. (↑GO*) (↑PU0B)→↓SBLK*/5B1 10. (↓SBLK*) (↑RBLK*) = SET→BLOCK BPEP REGISTER (↑BLKBEP/5B2)	Occurs at PU0 during cycle steal, interrupt, or when GO control logic is disabled.

4.2.2.2.2 Run Condition. The GO control register is set when the operator depresses and releases the front panel START switch. When this switch is operated, the GO signal becomes high level and the GO* signal becomes low-level. The high-level GO signal is coupled to the power delay latch through the power sense gate so that when low power is sensed, the power delay latch will be immediately reset and cause a halt in the operation of the clocking circuits at the conclusion of the instruction.

With the establishment of the run condition, a register clearing pulse is generated through the start-clear logic (3A1 and 3A2) so that various registers are cleared (refer to Table 4-1). The run condition is disabled whenever the STOP switch, or certain other halt conditions have been detected, but only after the current machine cycle has been completed. After the ND812 has been set into operation and stopped, the CONT switch should be depressed to obtain a new run condition. This prevents registers from being cleared again by the START switch. A high-level GO* signal is required to enable the front panel switch control logic; however, this signal is only present during the initialize state of the GO control register. The front panel switch control logic is also disabled when the NEXT WORD, LOAD MR, and LOAD AR switches on the front panel are in their normally off positions. Depressing any of these switches after a go condition has been established will have no effect because they are inhibited by the go-logic disabling signal.

4.2.2.3 MANUAL LOAD ADDRESS REGISTER (AR). The LOAD AR switch (2A4) transfers the switch register settings into the AR, the PC, and the memory field register (paragraph 4.5.3.5). To do this, power must be turned on. When power is sensed, the ND812 clock circuits are operating. The reader should refer to paragraph 4.2.1 and then return to this paragraph.

Figure 4-1 is the manual load-address block diagram. When the front panel LOAD AR switch is depressed, the load AR latch (3) provides a signal that enables the front-panel switch control logic. Prior to the application of PU timing pulses to the time control logic (9), the front-panel switch control logic (5) enables the utility gates (6), the MX multiplexer (7), and the adders (10). With the utility gates, the MX multiplexer and adders enabled, front panel switch register data (signals SW00 through SW11) are applied through the utility gates (signals U00 through U11) to the MX multiplexer, and from the enabled MX multiplexer (signals MX00 through MX11) to the adders. Because there is no other input to the adders at this time, the adder outputs (signals B00 through B11) are identical to the MX multiplexer inputs and are applied to both the AR (13) and the PC(12). (These signals from the adder are also applied to the bus and memory buffer multiplexer (4) which permits its outputs to come from either the adder bus of from the memory buffer register.)

The logic signals developed by the front panel control logic (5) are also applied to the time control logic (9) which develops its various enabling signals in synchronism with applied PU pulses. At the occurrence of timing pulse PU0 the output from the MX multiplexer/adders (7 and 10) is enabled to the AR (13). At the occurrence of timing pulse PU1, these same outputs are enabled to the PC (12) so that at the end of the first two timing pulses, both the address register and the program counter are loaded. The AR output signals (A00 through A11) are applied to the memory (1) for decoding when a read signal is sensed.

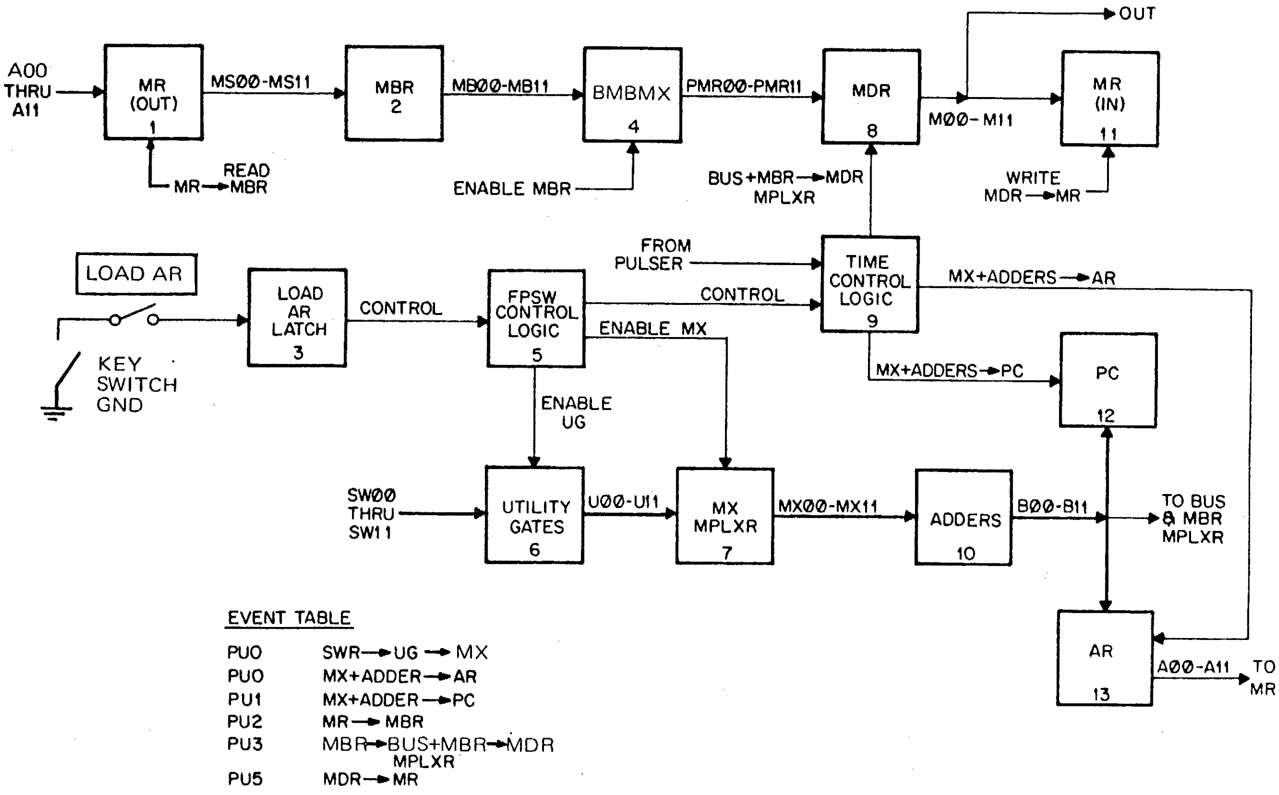


Figure 4-1. Manual Load Add·ess Register, Block Diagram

When pulse PU2 occurs, the location in memory whose address is specified by the AR outputs sends its signals (MS00 through MS11) to the memory buffer register (2). This sequence is enabled by memory control logic. Memory control logic also selects the memory buffer register outputs (signals MB00 through MB11) for input to the bus and memory buffer multiplexer (4), which in turn, applies its outputs (signals PMR00 through PMR11) to the memory data register (8). When pulse PU3 occurs, the outputs from the bus and memory buffer multiplexer are loaded into the memory data register.

When pulse PU5 occurs the outputs from the memory data register are applied to output I/O lines and to the memory (1) where they are written back into the same location in memory from which they were read. The read/write sequence permits the data at the addressed memory location to be loaded into the memory data register for display by front panel indicators.

Table 4-3 is the simplified logic for the manual load address sequence. This table together with the logic and timing diagrams will aid in understanding the following description.

When the LOAD AR switch on the front panel is depressed the XLDPR* signal (2A4) latch provides a low-level XLDPR* signal (Figure 4-2, wavefom A). Simultaneously both the LDPR* and the EXPMR* go to their low-level state (waveforms B and C). The switch control logic which permits these signals to be generated is enabled only in the initialize state of the go control because of a high-level GO* signal (6B1).

Ultimately these signals produce the high-level LDFPSW signal (6B3 and waveform D) which is applied to the utility gates and through them to the MX multiplexer. This data is entered into the MX multiplexer through the SELU gate (8A3) and develops a low-level SLUMX* (9A1) signal which sets MXEN* low and enters switch register data into the MX multiplexers through the utility gates.

Multiplexer select logic (9B1) accepts the low-level SLUMX* signal and develops a high-level MXS0 signal (9B1 and waveforms E and F, Figure 4-2_ which selects the utility gates for output from the multiplexers (Table 4-2) and enables signals MX00 through MS11 which are applied to the adders (sheets 14, 15 and 16). The outputs from the MX multiplexer are summed with the outputs from the TS multiplexer (sheet 13) when enabled; however, because the add-subtract output gates are not enabled, the sum output when TS data is not called for is the selected MX multiplexer output. The adder outputs (sheets 14, 15 and 16) are applied parallel to the inputs of the address register, the program counter and the J and K registers. The address register is enabled when pulse PEA* (waveform I, Figure 4-2) goes to its low-level state and the program register is enabled when pulse PEP* (waveform K) goes to its low-level state.

Because the go control logic has not been set to its run state, the pulser control logic (3A4) is in its initialized state and configured for toggle operation. With the go control logic in its initialized state pulse PU7B (waveform D, Figure 4-10) cannot recycle the pulser so that only a single string of PU pulses is generated by the EXPMR* signal. This string of PU pulses permits other enabling signals to be generated.

If the ND812 is in its run state when data is to be entered into the address register through the switch register switches, the ND812 STOP switch must first be operated to obtain the initialized condition of the go-control register. One of the important functions obtained when the go-control logic is in its initialized state is that when the pulser is operated, only a single machine cycle will be developed and the basic and execute phases will be blocked so that BP and EP pulses cannot be generated. This is because the low-level GO signal applied through the RBLK* NAND gate (5B2 and E8 and E9, Table 4-3) and the high-level GO* signal applied through the SBLK* NAND gate produce the set condition of the BLKBEP flip-flop register. The high-level BLKBEP signal is inverted and applied to the basic and execute phase NAND gates as an inhibiting signal.

Table 4-3. Manual Load Address Register, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	LOAD AR	SWR→MX 1. ↓XLDPR*→↑EXLDPR/6B1 2. (↑EXLDPR*) (↑GO*)→↓LDPR*/6A1 3. ↓LDPR* + ↑LDMR*→↑LDPRMR/6B2 4. ↓LDPR* + ↓LDMR* + ↓EXAM*→↑EXPMR→ ↓EXPMR*/6B2 5. ↓LDPR*→↑LDPR/6A2 6. (↑LDPRMR) (↑ALODT*)→↓LDPSW*/6B2 7. ↓LDPSW*→↑LDFPSW/6B3 8. (↑LDFPSW) (↑SW00* - ↑SW11*) = ↑U00-U11/14, 15, 16→MX	Occurs when LOAD AR switch is depressed.
B.	LOAD AR	MX→ADDERS 1. ↑LDPR→↓SELU6*→↑SELU→↓SLUMX*/8A4 2. ↓SLUMX*→↑MXEN→↓MXEN*/9A2 3. ↓SLUMX*→↑MXS0 = U00-U11→ MX→ADDER	Occurs when LOAD AR switch is depressed.
C.	LOAD AR	START→PULSER 1. ↓EXPMR*/6B2 = DISABLE GO CONTROL TOGGLE, 3A3 2. ↓EXPMR*/6B2 = ↑CKST1*/3A4→↑ST1/3A4 = PULSER ON (Pulser Cycle off by PU0)	AR switch is depressed.
D.	PU0 AR	CLR→LATCHES AND REGISTERS 1. ↓EXPMR*→↑STPU1→↓STPU1*/3A3 2. (↓STPU1*) (↑CNTRP)→CLR CYCLE STEAL PERMIT 3. (↓CSP, ↑CSP*/4B2) 4. (↓STPU1*) (↑CNTRP)→CLR INTERRUPT PERMIT 5. (↓INTP, ↑INTP*/4B4)	PULSER AR switch.

Table 4-3. Manual Load Address Register, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
E.	PU0	PULSER→1 CYCLE 1. ↓PU0*→↑RST1→↓RST1*/3A4 2. (↓RST1*) (↓PU7B) + ↓GO + ↓DBLE*→ ↑SST1*/3A4 3. (↑SST1*) (↓RST1*)→↓ST1/3A4 = PULSER→1 CYCLE	Permits only 1 cycle to occur because GO control is disabled. Pulser zeroed by PU0.
F.	PU0	ADDER→AR 1. ↑PU0B→↓PEA→↑PEA/9B1 2. (↑PEA) (↑REGCLK)→↓CPA* = ADDER→AR	Adder data unconditionally to AR at this time.
G.	PU1	CLR→MBR 1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	Memory buffer register cleared unconditionally at PU1.
H.	PU1	ADDER→PC 1. (↑PU1B) (↑EXPMR)→↓EXM1B*/9B2→ ↑MPEP/9B3→↓PEP*→↑PEP/9B1 2. (↑PEP) (↑REGCLK)→↓CPP*/9B2 = ADDER→PC	Adder to program counter.
I.	PU2	MR→MBR 1. (↑ADDF0*) (↑MCRT)→↓MCIR*/8A4 = MR→MBR	Read memory data occurs unconditionally at PU2.
J.	PU3	BMBMX→MDR 1. ↓PU3*→↑PEM/12B2 2. (↑PEM) (↑REGCLK)→↓CPM*/12B2 = MBR→BUS + MBR MPLXER→MDR 3. ↑ADDF1* = MBR→BMBMX	Memory buffer to memory data register.
K.	PU5	PU5B, MDR→MR 1. (↑WAIT*) (↑RLOOP*)→↓WTRL→↑WTRL*/8B2 ↑WTRL*/8B2 2. (↑WTRL*) (↑PU5B)→↓PU5W*→ ↑PU5W/8B3 3. (↑PU5W) (↑ADDF0*)→ ↓MCIW*/8B4 = BMBMX→MDR	Write memory data occurs at PU5 if multiply or divide are not in effect.

Table 4-3. Manual Load Address Register, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
L.	PU0	EXECUTE AND BASIC PHASES BLOCKED 1. ↓GO + ↓RDBLE*→↑RDBLE/4B1 2. (↑RDBLE) (↑CSP*) (↑SDCS*)→ ↓RCSFF*/3B2 3. (↓RCSFF*) (↑CSP*) = CLR→CYCLE STEAL FF (↓CSFF/3B2, ↑CSFF*) 4. (↑INTP*) (↑RDBLE)→↓RITFF*/4B4 5. (↓RITFF*) (↑INTP*) = CLR→INTERRUPT FF (↓INTFF/4B4, ↑INTFF*) 6. (↑INTFF) (↑CSFF*)→↓INTCSF/4B4 7. ↓INTCSF→↑INTCS*/5B1 8. (↑INTCS*) (↑PU0) (↓GO)→↑RBLK*/5B2 9. (↑GO*) (↑PU0B)→↓SBLK*/5B1 10. (↓SBLK*) (↑RBLK*) = SET→ BLOCK BPEP REGISTER (↑BLKBEP/5B2)	Occurs at PU0 during cycle steal, interrupt, or when GO control logic is disabled.

Pulses PEA* and PEP* (sheet 9) are generated through their respective control logic circuits/pulse PEA* goes low when pulse PU0B (Figure 4-10) is generated and pulse PEP* when signal EXPMR goes high level concurrently with pulse PU1B. With the occurrence of these pulses both the address register and program counter register are enabled, however, this data is not strobed into these registers until the next REGCLK pulse occurs and pulses CPA* and CPP* (waveforms J and L, Figure 4-2 and 9B2) are generated.

With address register output signals A00 through A11 applied to the memory (core) register address decoders (6M2 and 10M3), the proper X-Y-select lines are selected from the memory matrix and the data stored at the indicated address is read into the memory buffer register (18A1, B1 and A2, B2) from the memory register when pulse MCIR* (8B4 and waveform M, Figure 4-2) occurs. This takes place when pulse PU2 (8B4) is generated by the pulser (refer to time-state PU2, Table 4-3, I). Pulse MCIR* commands the memory control logic to read from memory.

Outputs MB00 through MB23 of the memory buffer register are applied to the bus and memory buffer multiplexer (sheet 17) together with the bus (adder) outputs B00 through B11. Because both signals ADDF1* (sheet 17) and RDB* (12B3) are high level at this time, the memory buffer register (signals MB00 through MB11) is selected for output from the bus and memory buffer multiplexer. Thus when pulse PU3 (12B2) occurs, the memory buffer inputs are loaded into the memory data register (sheet 17) by signals PEM* and CPM*, the former enabling the inputs (signals PMR00 through PMR11) and the latter clocking them. Thus, the content of the memory register is loaded into the memory data register.

When pulse PU5 occurs, pulse MCIW* (8B4 and waveform Q, Figure 4-2) goes low level and the data contained in the memory data register is written back into the memory

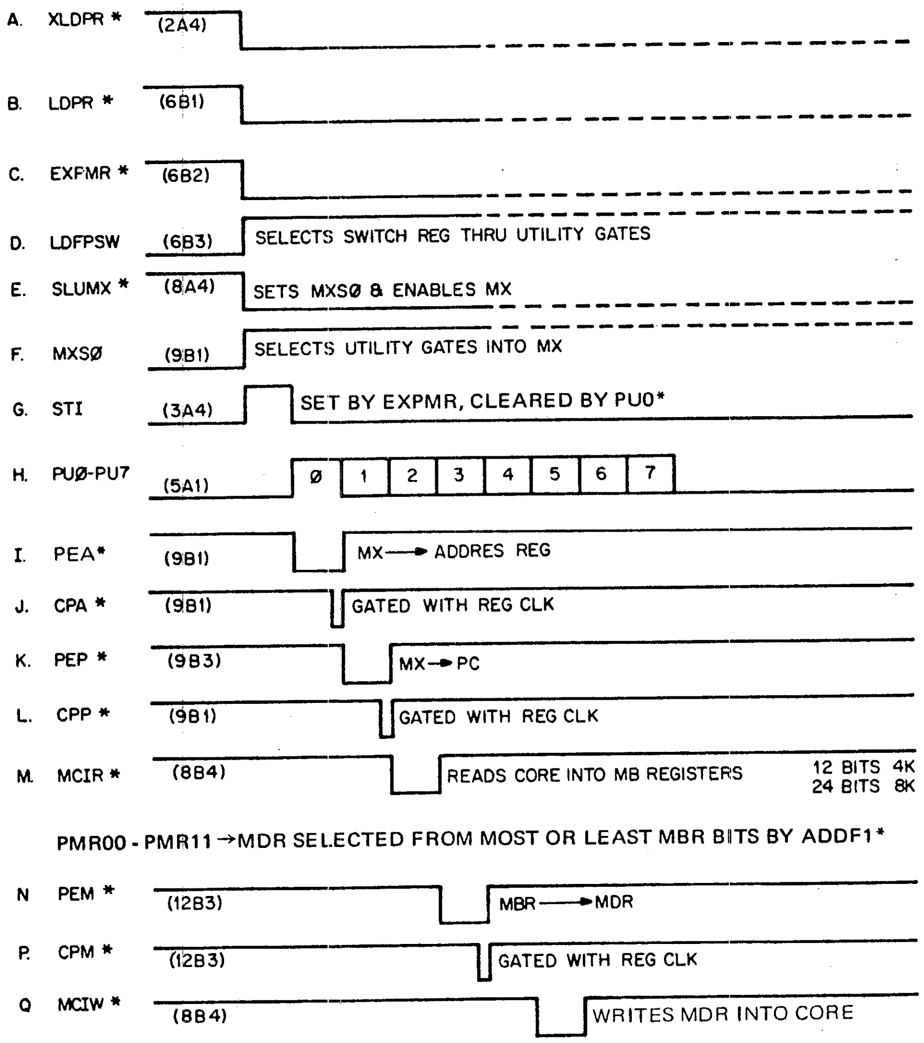

A. XLDPR * (2A4)
B. LDPR * (6B1)
C. EXFMR * (6B2)
D. LDFPSW (6B3) SELECTS SWITCH REG THRU UTILITY GATES
E. SLUMX * (8A4) SETS MXSØ & ENABLES MX
F. MXSØ (9B1) SELECTS UTILITY GATES INTO MX
G. STI (3A4) SET BY EXPMR, CLEARED BY PU0*
H. PUØ-PU7 (5A1)
Ø
1
2
3
4
5
6
7
I. PEA* (9B1) MX——▶ ADDRES REG
J. CPA * (9B1) GATED WITH REG CLK
K. PEP * (9B3) MX—▶PC
L. CPP * (9B1) GATED WITH REG CLK
M. MCIR * (8B4) READS CORE INTO MB REGISTERS
12 BITS 4K
24 BITS 8K
PMR00 - PMR11 →MDR SELECTED FROM MOST OR LEAST MBR BITS BY ADDF1*
N PEM * (12B3) MBR——▶MDR
P. CPM * (12B3) GATED WITH REG CLK
Q MCIW * (8B4) WRITES MDR INTO CORE

Figure 4-2. Manual Load Address Register, Timing Diagram

register (core). The result of this operation is that the front panel MEMORY REGISTER lamps indicate the content of the addressed memory data register.

4.2.2.4 MANUAL LOAD MEMORY REGISTER (MR). A word can be data or an instruction or an address. It is the information stored at a uniquely addressed location in core. The LOAD MR switch on the front panel is used to load a word into memory at the location held by the AR. When the LOAD MR switch is operated, the content of the PC is transferred into the AR. The program counter is incremented and now contains the next address to be accessed after this operation is over. The address register contains the address currently accessed.

Figure 4-3 is the manual load memory block diagram. Except for the events that occur, this block diagram is identical to that for the manual load address, (Figure 4-2). When the front panel LOAD MR switch is raised, the load memory register latch (3) is set. Although this logic is similar to the load address register logic, a different sequence of events is ordered.

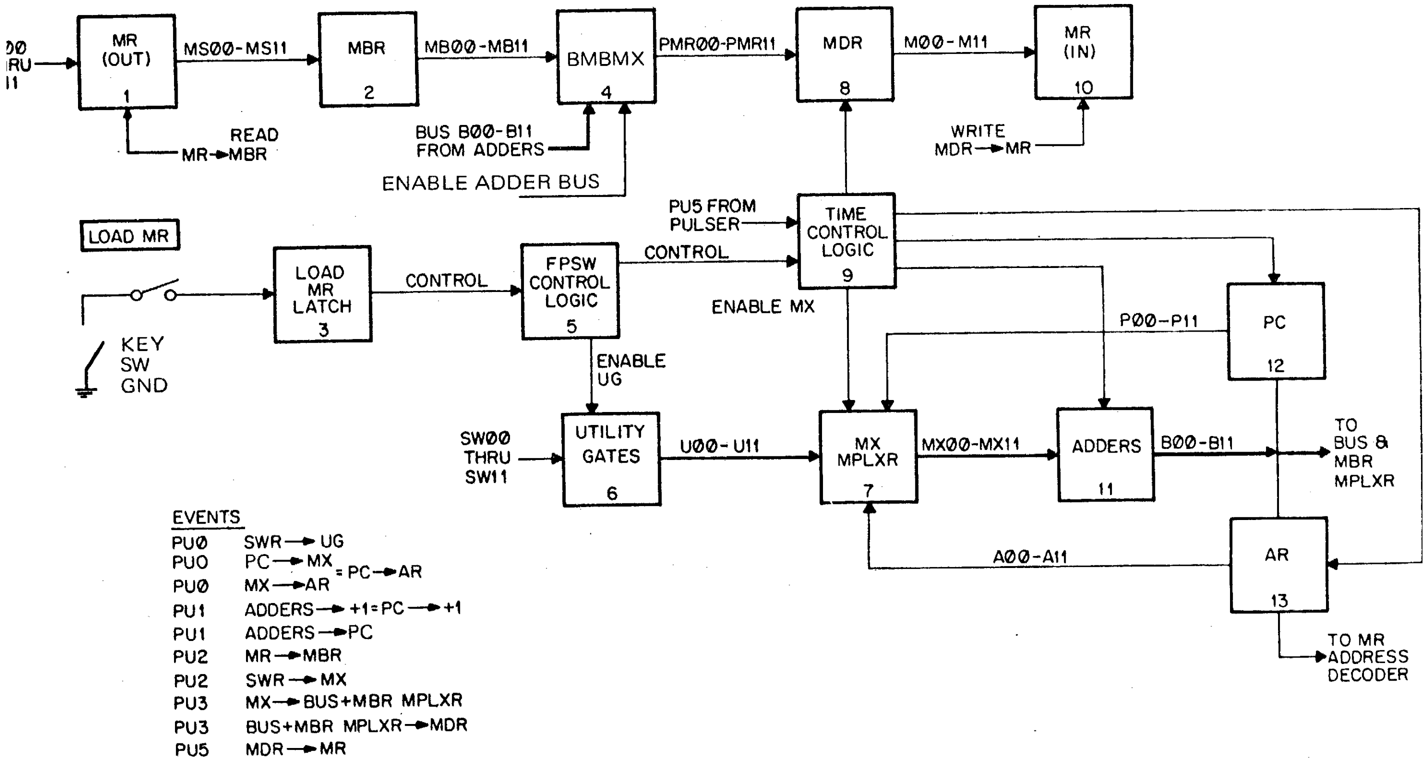


Figure 4-3. Manual Load Memory Register, Block Diagram

Prior to the application of PU timing pulses to the time control logic (9) the front panel switch control logic (5) enables the utility gates (6) so that switch register data (signals SW00 through SW11) is ready to be switched through the MX multiplexer (7).

The logic signals developed by the front panel switch control logic (5) are also applied to the time control logic (9) which develops the various enabling signals in synchronism with the applied PU pulses. When pulse PU0 occurs, two events take place simultaneously; the content of the PC (12) is switched through the MX multiplexer (7) and the MX multiplexer outputs are applied through the adders (11) to the AR (13). Thus, signals P00 through P11 are loaded into the address register through multiplexer signals, MX00 through MX11, and adder signals B00 through B11.

When pulse PU1 occurs two more events take place; a carry input is propagated through the adders so that their content, which is still the output of the PC, is incremented by 1 and the output of the adders is returned to the PC.

When pulse PU2 occurs, the output of the MR (1) is loaded into the memory buffer register (2) in a manner similar to that previously described for the manual load address mode of operation.

When pulse PU3 occurs, three events take place; the contents of the front panel switch registers are switched through the MX multiplexer and adders (7 and 11), adder outputs are switched through into the bus and memory buffer multiplexer (4), and the bus and memory buffer multiplexer outputs are loaded into the memory data register (8). In this manner, the content of the switch register (signals SW00 through SW11) are switched through the utility gates (signals U00 through U11) into the MX multiplexer and adders and the adder output (signals B00 through B11) is switched through the bus and memory buffer multiplexer into the memory data register (signals PMR00 - PMR11).

When pulse PU5 occurs, the content of the memory data register (8) is loaded into the memory (1) at the address specified by the AR (13) similar to that previously described for the manual load address mode of operation. In this case however, the memory data register (8) content was replaced by the content of the switch registers.

Table 4-4 is the simplified logic for the manual load memory sequence. This table together with the logic diagrams will aid in understanding the following description.

When the LOAD MR switch is raised the XLDMR* signal (2A4) latch provides a low-level XLDMR* signal (Figure 4-4, waveform A). This signal also enables the BLST1* one-shot gate (2B4) and produces an output from the one shot when the LOAD MR switch is released. The J-K input of the pulser control register is enabled in a manner similar to that previously described for loading an address. Similarly, because the go-control logic has not been set to its run state, the pulser output produces a string of 8 pulses identical to that previously described for loading an address. And, these signals, together with the signals enabled when the LOAD MR switch is raised, permit other enabling signals to be generated.

Table 4-4. Manual Load Memory Register, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	LOAD MR	SWR→MX 1. ↓XLDMR*→↑EXLDMR/6B1 2. (↑EXLDMR) (↑GO*)→↓LDMR*/6B1 3. ↓LDMR* + ↓LDPR*→↑LDPRMR/6B2 4. ↓LDMR* + ↓LDPR* + ↓EXAM*→↑EXPMR→↓EXPMR*/6B2 5. ↓LDMR*→↑LDMR/6B2 6. ↓LDMR* + ↓EXAM*→↑EXMR/6B2 7. (↑LDPRMR) (↑ALODT*)→↓LDPSW*/6B2 8. ↓LDPSW*→↑LDFPSW/6B3 9. (↑LDFPSW) (↑SW00* - ↑SW11*) = ↑U00-U11/14, 15, 16→MX	Occurs when LOAD MR switch is depressed.
B-E	LOAD MR	Simplified Logic is Identical to C through F of Table 4-3.	Refer to Event Column of Table 4-3.
F.	PU0	PC→MX→ADDER 1. (↑EXMR) (↑PU001)→↓CSELP→↓SLPMX*/8B4 2. ↓SLPMX*→↑MXEN/9A2 = ENABLE→MX 3. ↓SLPMX* + ↓EX1*→↑MXS1/9B1 = PC→MX	Occurs when LOAD MR switch is depressed only.
G.	PU1	+ 1→ADDER 1. (↑EXMR) (↑PU1B)→↓CINI*→↑CIN/12B2 = + 1→ADDER	Occurs only when LOAD MR switch is depressed.
H.	PU1B	CLR→MBR 1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	Occurs unconditionally during PU1.
I.	PU1B	ADDERS→PC 1. (↑PU1B) (↑EXPMR)→↓EXM1B*→↓PEP*→↑PEP/9B1 2. (↑PEP) (↑REGCLK)→↓CPP*/9B2 = ADDER→PC	Occurs only when LOAD MR switch is depressed.
J.	PU2	MR→MBR 1. (↑ADDF0*) (↑MCRT)→↓MCIR*/8B4 = MR→MBR	Read occurs unconditionally during PU2.
K.	PU3	SWR→MX→ADDER 1. ↓LDMR* + ↓CSRDB*→↑SELU3/8A3 2. (↑SELU3) (↑PU3B)→↓SLUMX*/8A4 3. ↓SLUMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 4. ↓SLUMX*→↑MXS0/9B1 = U00-U11→MX	Occurs only when LOAD MR switch is depressed.

Table 4-4. Manual Load Memory Register, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
L.	PU3B	ADDERS→BMBMX→MDR 1. ↓LDMR* + ↓INTFF* + ↓CSRDB*→ ↑RDB3/12B2 2. (↑RDB3) (↑PU3B)→↓BMEM*/12B2 3. ↓BMEM*→↑RDB→↓RDB*/12B3 = ADDER→BMBMX 4. ↓BMEM* + ↓PU3*→↑PEM/12B3 5. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = BMBMX→MDR	Occurs only when LOAD MR switch is depressed.
M.	PU5	MDR→MR 1. (↑WAIT*) (↑RLOOP*)→↓WTRL→ ↑WTRL*/8B2 2. (↑WTRL*) (↑PU5B)→↓PU5W*→ ↑PU5W/8B3 3. (↑PU5W) (↑ADDF0*) = MDR→MR	Write operation occurs at PU5 if multiply or divide not in effect.

A low-level XLDMR* signal (2A4) produces high-level LDFPSW (6B3) and EXPMR (6B2). EXMR (6B2) and LDMR (6B2) signals (Figure 4-4, waveforms A through D) and low-level EXPMR* and LDMR* signals. The high-level EXMR signal produces the SLPMX* (waveform I) signal when pulse PU001 (combines PU0 and PU1) is generated. This signal selects the PC buses (P00--P11) for input to the MX multiplexer (14, 15 and 16) (8B3) and develops signal MXS1 (9B2) which selects input line 12 (Table 4-4). Previously, the low-level EXPMR* signal produced signal MXEN* (9A1) which enabled the MX multiplexer (sheets 14, 15 and 16).

The MX multiplexer output signals (MX00 - MX11) are applied to the adders (sheets 14, 15 and 16) when the low-level SLPMX* signal is generated. During the same time interval, signals PEA* and CPA* (when the next REGCLK signal is produced) are generated (9B1). These signals enable the AR (waveforms G and H) which receives its inputs (B00 - B11) directly from the adders (sheets 14, 15 and 16).

During the subsequent time interval a high-level CIN pulse (12A2) is generated at the coincidence of the high-level EXMR signal and a PU1B pulse. This signal is a carry input to the least significant bit (B11) of the adders (sheets 14, 15 and 16) that is propagated down the adders to the most significant bit (B00) causing the output of the MX multiplexers to be incremented through the adders. Simultaneously pulses PEP* (9B3) and CPP* (waveforms K and L, Figure 4-4) permit adder outputs to be loaded into the PC (9B3 sheets 14, 15 and 16). A high-level EXPMR and PU1B generates pulse PEP*, and PEP* and REGCLK pulses generate CPP* pulse.

AR output signals A00 through A11 are applied to the memory register address decoders (sheet 3, 8K memory), the proper x- and 6-select lines are enabled in the memory

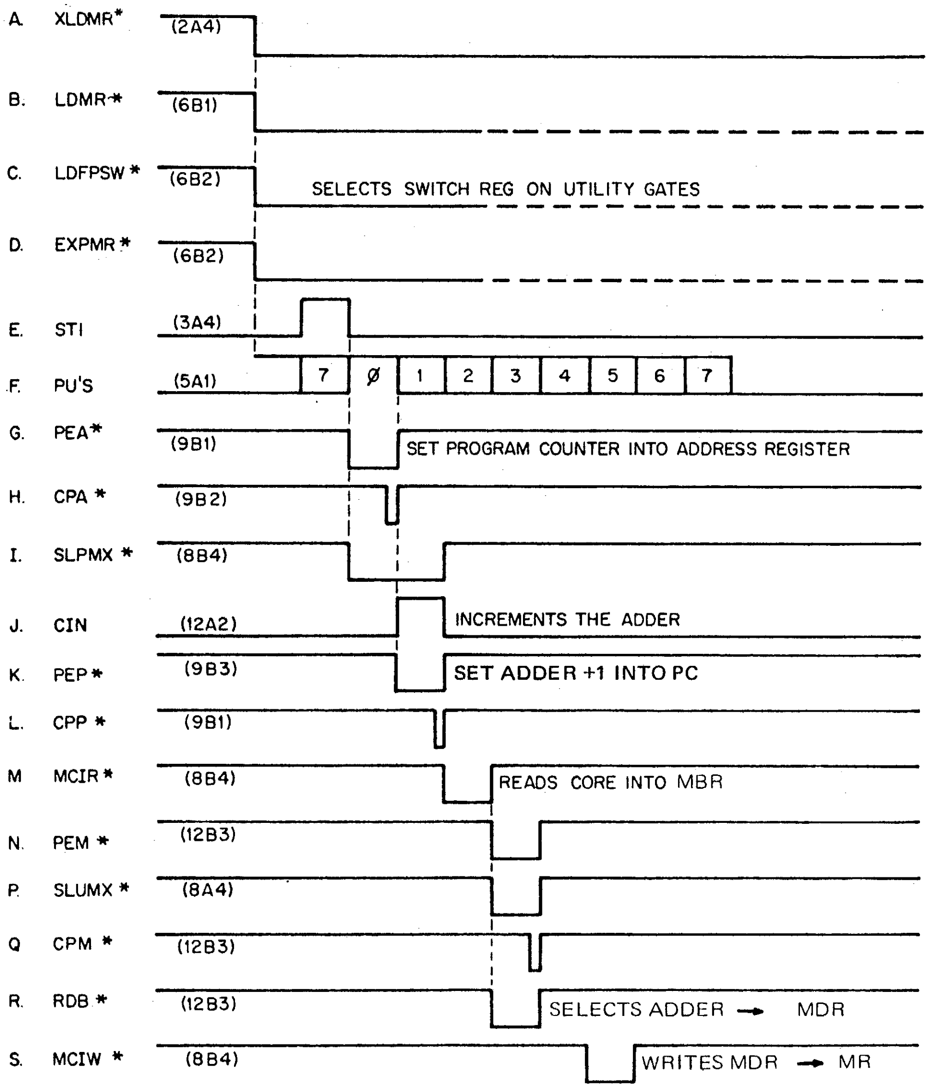


Figure 4-4. Manual Load Memory Register, Timing Diagram

address matrix and the data stored at the indicated address is read into the memory buffer register (sheet 18) from the memory when pulse MCIR* (8B4 and waveforms M, Figure 4-4) occurs. This takes place when pulse PU2 (8B4) is generated by the pulser (refer to time-state PU2, Table 4-4).

Outputs MB00 through MB23 of the memory buffer register are applied to the bus and memory buffer multiplexer (sheet 17) together with buss (adder) outputs B00 through B11. However, at this time signal ADDF1* is high level and signal RDB* (waveform R) is low level (12B3) so that when pulse PU3B occurs the adder (bus) input (signals B00 through B11) is selected for output from the memory buffer and bus multiplexer (Table 4-4). Also when pulse PU3 occurs the low-level LDMR* signal (6B2) develops a low-level SLUMX* (8A4 and waveform P, Figure 4-4) which in turn produces high-level MXSO and MXEN signals. The MXSO signal admits the content of the utility gates (switch register data) into the MX multiplexer (Table 4-4 and sheets 14, 15 and 16). Also, low-level signal PU3* (12B2) develops low-level PEM* and CPM* signals (12B3 and waveforms N and Q, Figure 4-4), the latter produced when the next REGLCK signal is generated. These signals admit the content of the bus and memory buffer multiplexer to the memory data register.

When pulse PU5 occurs a low-level MCIW* write pulse (12B3 and waveform S, Figure 4-4) causes the content of the memory data register (signals M00 through M11) to be written into memory. Pulse MCIW* is enabled through the memory control logic by the high-level ADDF0 signal. As a result of this operation the front panel MEMORY REGISTER lamps indicate that the content of the switch register settings has been loaded into the MDR and written into MR.

When successive locations in memory are to be loaded, it is only necessary to enter the data into the switch register and raise the LOAD MR switch.

4.2.2.5 SINGLE STEP OPERATION. When the SINGLE STEP switch on the front panel is raised, ND812 timing will halt at the end of the current phase. With this switch in its operate state (up), depressing the CONT switch will execute the next phase and each phase thereafter, each time the CONT switch is depressed.

Start and stop of ND812 timing is controlled through the go control register (3A3 and Table 4-2) reset logic. When the SINGLE STEPswitch is raised, signal SS* (3A2) is inverted to produce high level signal SS which is applied through AND logic together with pulse PU6B (5B2) to the set input of the go control register. Thus, if an interrupt or DMA operation is not in progress, when pulse PU6B occurs, the go control logic will be set to its initialize state. Because pulse PU6B occurs during each basic-phase and execute-phase, go control logic will be set to its initialize state at the end of either cycle. If the SINGLE STEP switch is not held in its operate state when the CONT switch is operated, pulse PU6B will not be permitted to enable the AND gate, and operation of the ND812 timing will not be halted. In single step, pulse PU7 is not permitted to recycle the pulser control register and thus timing pulses are inhibited.

Operation of timing and go control logic can be more readily understood by referring to Figure 4-5. With the SINGLE STEP switch held in its operate state, signals

INTCS* and SS (waveforms A and B) are high level and permit pulse PU6B (waveform D) to initialize the go control logic causing a low-level GO signal (wavefom C). With GO signal in its low-level state pulse PU7B, which is normally applied to the pulser control register (3A4), is inhibited and pulse ST1 (waveform E) cannot go to its normal high-level state, thus preventing the pulser from being recycled. However, pulse PU7B does enable the major-state control logic (4A2) so that either pulse BPH (waveform F) goes to its low-level state, and/or pulse EPH (waveform G) goes to its high-level state. Thus, when the CONT switch is again depressed, the major-state control will produce pulses in the next major state.

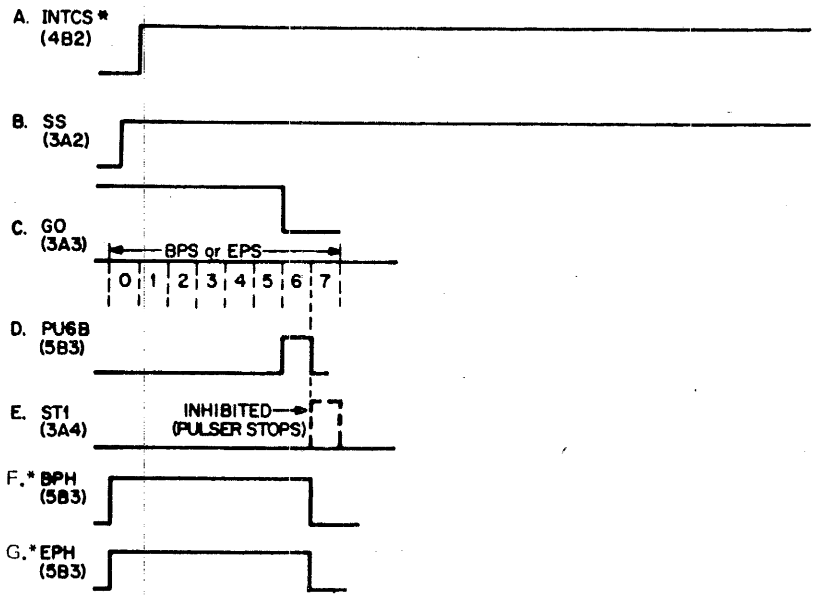
NOTE

> Although both basic- and execute-phase pulses are shown high level, indicating that this action occurs in either state, they are actually of opposite polarity and cannot occur as shown (refer to Figure 4-5).

4.2.2.6 SINGLE INSTRUCTION. When a single instruction is to be executed, the SINGLE INSTR switch is raised. When this is done, the ND812 will halt after the current instruction has been executed. With this switch in its operate state (up), depressing the CONT switch will execute the next instruction and each instruction thereafter, each time the CONT switch is depressed.

Start and stop of ND812 timing is controlled through the go control register reset logic (3A3 and Table 4-2) in a manner similar to that used for single stop operation. When the SINGLE INSTR switch is raised, signal SI* (3A2) is inverted to produce high-level signal RGO which is applied to AND logic together with pulse FFCLK to the clear input of the go control register. However, to get pulse FFCLK, the done latch must be set, and pulse PU6B must occur. Other conditions that must be satisfied to get this pulse are that the go control register must be in its run state and a clock pulse must be present. Because the done latch must be set and pulse PU6B must be present, the go control register cannot be initialized until the machine cycle has been completed. When interruption occurs, pulse PU7 is not permitted to recycle the pulser control register as described for the single-step operation. Hence, only single machine cycles are enabled.

Operation of timing and go control logic can be more readily understood by referring to Figure 4-6. With the SINGLE INSTR switch raised, signals SI* and RGO* (waveforms A and B) are low-level and high-level, respectively, and permitting pulse PU6B (waveform E) to initialize the go control logic causing a low-level GO signal when the done latch is set (waveform D) and when pulse CPPU (not shown) are coincident. When pulse CPPU occurs, pulse FFCLK (waveform F) is produced and the go control register is changed from its run state to its initialized state (waveform G). With the signal GO now in its low-level state, pulse PU7B, which is normally applied to the pulser control register (3A4) is inhibited so that pulse ST1 (waveforms H and I) does not occur. This pulse was previously described for the single step operation. When the CONT switch is depressed, normal operation resumes, but if the SINGLE INSTR switch remains reaised at this time, operation will again be halted after the next instruction has been executed.



* See paragraph 4.4.1.2

Figure 4-5. Single-Step Operation, Timing Diagram

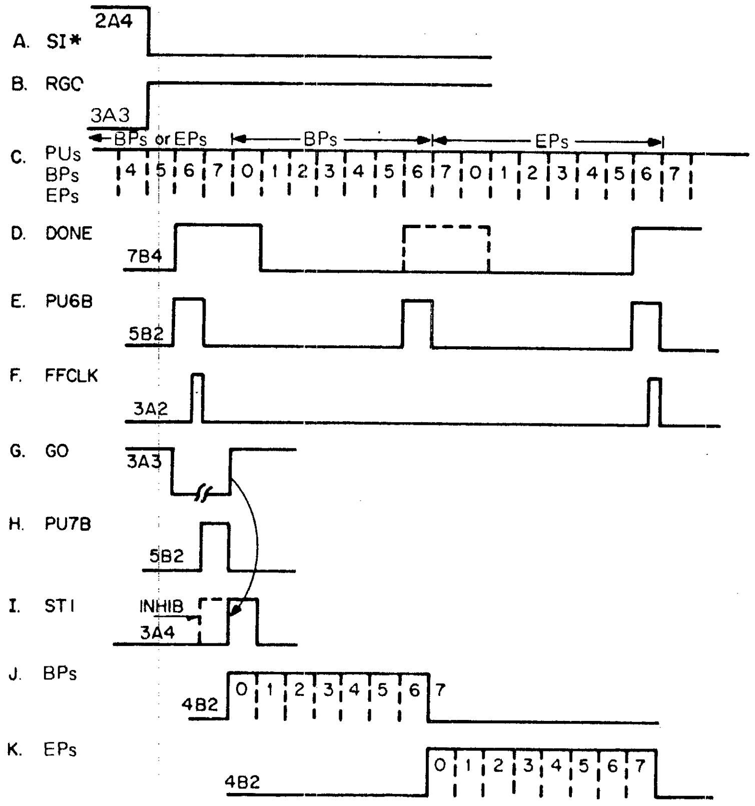


Figure 4-6. Single Instruction Operation, Timing Diagram

4.2.2.6.1 Done Latch. Termination of the current machine cycle is dependent on the state of the done latch. At the conclusion of a given instruction, the done latch is set at the end of a basic phase (dashed line, waveform D, Figure 4-6), or at the end of the execute phase. Instructions that set the done latch at the conclusion of the basic phase are: a halt instruction, an I/O instruction, a literal instruction, a jump instruction, and the group I and group II operate instructions. However, for multiply and divide operation, the basic phase is modified to provide sufficient time to complete the indicated operation. The done latch is set either at the end of the basic phase by pulse BP6 when one of the above listed instructions is processed, or by pulse EP6* when a two- or three-phase memory reference instruction is processed.

Table 4-5. Done Latch, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	BP6	SINGLE PHASE DONE 1. ↓TWHLT* + ↓I/O* + ↓ADL* + ↓SBL* + ↓ANL* + ↓OP2 + ↓JMPIND*→↑SDONE/7B4 2. (↑SDONE) (↑BP6)→↓SDONE*/7B4 = (↑DONE/7B4, ↓DONE*)	Single-cycle done.
B.	EP6	MULTI-PHASE DONE 1. ↓SDONE* + ↓EP6* = SET→DONE (↑DONE/7B4, ↓DONE*)	Two-cycle done.
C.	PB7 or EP7	RECYCLE, BASIC PHASE 1. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→↑RBPH→↓RBPH*/5B1 2. (↓RBPH*) (↑PU7) (↑OSCPUL) (↑STCLR*) = RESET→MAJOR-STATE REGISTER (↓EPH/5B2, ↑BPH)	Basic phase enabled.
D.	PU6	STOP, SINGLE INSTRUCTION 1. (↑DONE) (↑CPPU) (↑PU6B) (↑GO)→↓FFCLK*/3A2→ ↓FFCLK*→↑FFCLK/3A2 2. ↓RGPWR* + ↓SI* + ↓STOP*→↑RGO/3A2 3. (↑RGO) (↑FFCLK)→↓RGO*/3A3 4. (↓RGO*) (↑CNTRP)→INITIALIZE GO (↓GO/3A3, ↑GO*)	Stop occurs at period 6 when done latch is set.
E.	PU7	RESET LATCHES AND REGISTERS 1. (↑DONE) (↑PU7B)→↓STDN7*/3B3 2. (↑DONE) (↑PU0B)→↓MRI* = CLR→IR	All latches and registers set during machine cycle are cleared at period 7. (Table 4-2) Instruction register cleared at period BPO.

The done latch permits the basic phase to be reentered when the ordinary two-cycle memory reference instruction has its indirect bit set, when a two-word instruction is being processed, and when an XCT instruction is being processed. All these instructions cause the current machine cycle to be deferred to produce a multi-phase instruction. The enabled done latch also permits various registers and latches to be reset (Table 4-5). The done latch is cleared whenever pulse BP1 is generated.

4.2.2.7 REGISTER SELECT CONTROL. The status of any register whose output is applied to the MX multiplexer can be examined when the ND812 is in a stopped condition. Logic which permits these registers to be selected is enabled through the go control logic. These registers include the status, S, R, K, J, AR, PC, and the EXT inputs. The EXT inputs indicate any data on the EXT lines which are currently in transfer from an I/O device to the ND812. If the ND812 is not in a stopped condition the SELECTED REGISTER indications are meaningless. During troubleshooting and maintenance or software debug, the ability to examine these registers under single-step and single instruction operation gives the field service engineer a silent helper whose aid can be very helpful.

When the go control logic is in its initialized state after the stop switch has been depressed, register select logic Figure 4-7, (4) provides meaningful data to the MX multiplexer select logic via signals EMXS0*, EMXS1*, and EMXS2*. These signals provide 8 combinations, one for each position of the front panel SELECT REGISTER switch. In fact, when the register select logic (4) is enabled, signals EMXS0*, EMXS1*, and EMXS2* produce signals MXS0, MXS1 and MXS2 through the multiplexer select logic (5), thus, depending on the position of the SELECT REGISTER switch, one of 8 inputs to the MX multiplexer (6) is enabled.

Outputs from the MX multiplexer (signals MX00 through MX11) are applied to the adder (8). Under given conditions MX multiplexer data is summed with TS multiplexer data or a carry input, but not in this case, because neither the TS multiplexer outputs, or the carry input are enabled. Thus, the sum of MX multiplexer data, zero carry, and zero TS multiplexer data results simply in MX multiplexer output data. Adder output data is carried by signals B00-B11 and is applied to the output gates (9), and to other registers which are not enabled at this time. Output gates (9) are enabled by the output select logic (7) when the go control logic (1) is in its initialize state.

Utility gates (3) receive external input signals EXT00 through EXT11 and are enabled through external select logic (2) when the go control logic is in its initialize state.

Under conditions when the SELECT REGISTER switch (2A3) is in the STATUS position, +5 Vdc power is coupled through pullup resistors to three NAND gates. Also included as inputs to these NAND gates, is externally produced I/O signal ALODT*, which when enabled is low-level, so that normally this input, applied through inverting buffers, together with the resistive inputs, gives rise to high-level output signals EMXS0*, EMXS1*, and EMXS2*. For various positions of the SELECT REGISTER switch, one or more of these signals becomes low level as shown in Table 4-6. These combinations of low level signals are applied through inverting buffers (9A1, 9A2) to three NAND gates

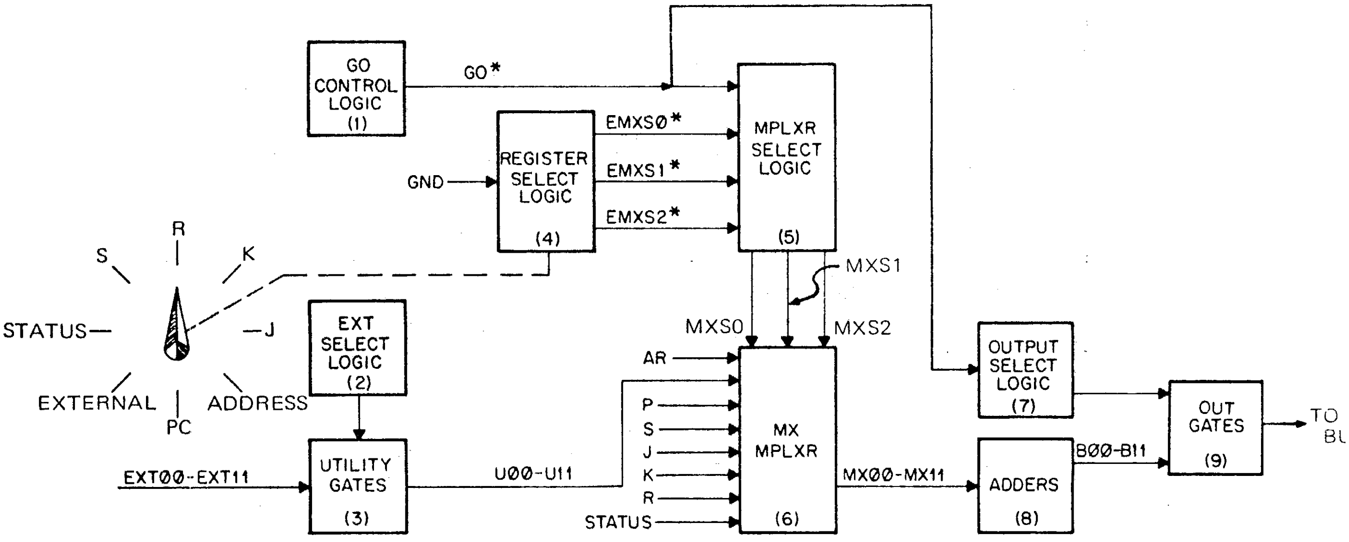


Figure 4-7. Register Select Control, Block Diagram

Table 4-6. Selectable Register Switch Outputs

Selected Register	EMXS2*	EMXS1*	EMXS0*
STATUS	0	0	0
S	1	0	0
R	0	0	1
K	0	1	0
J	0	1	1
AR	1	1	1
PC	1	0	1
EXT	1	1	0

which are enabled when the high-level GO* signal applied through an OR gate (9A1) is produced. This high-level signal is applied to the enabling NAND gates and to the MX multiplexer enabling logic. If the go control logic (3A3) is in its run state these enabling inputs are inhibited by the low-level GO* signal.

The EMXS signals produce corresponding, but oppositely-phased signals MXS0, MXS1, and MXS2 (9A2/9B2) which, in various combinations, select the MX multiplexer signals listed in Table 4-18 for output to the adders as signals MX00 through MX11 (sheets 14, 15 and 16). Adder outputs B00 through B11 are applied to output NAND gates (14B4, 15B4, and 16B4) which are enabled by output select logic (6B3) when high-level signal OUTST is produced. This signal is enabled through NAND gate C11 by a high-level GO* signal from the go-control logic when it is in its initialized state. The other enabling signals applied to this NAND gate are high-level when the go control logic is in its initialized state.

4.2.2.8 START CONTROL. When the START switch is depressed certain latches and registers are cleared. These latches and registers are listed in Table 4-7, the simplified logic is shown in Table 4-8. The latches and registers listed in A through N are reset only when the START switch is depressed, and under certain operating conditions. The latches and registers listed in P through Y of Table 4-7 are reset at the end of each instruction and when the START switch is depressed. They are also reset as the result of certain operating conditions.

When the START switch is depressed a low-level START* signal is developed by the start latch (2A3). This low-level signal is applied to the start-clear logic (sheet 3) to develop the low-level STCLR*, STPU1*, and STDN7* pulses. The STCLR* pulse provides the clearing signal for latches and registers listed as A through N in Table 4-2. The STDN7* pulse provides the clearing signal for latches and registers listed as P through Y. These latches and registers are also cleared at the end of each instruction when a high-level DONE signal and pulse PU7B are coincident. (Table 4-8 is the start control simplified logic.)

4.3 CLOCK GENERATOR AND TIMING LOGIC

Clock generator and timing logic includes the oscillator, clock generator, pulser, and register clocking and enable logic. These circuits provide synchronization when the processor is executing either a Memory Reference Instruction (MRI) or operate instruction. Certain modifications of the machine cycle, however, occur when an I/O instruction is being processed. When the instruction register decodes an I/O instruction, the normal machine cycle is interrupted and an input/output machine cycle is initiated. At the completion of an I/O machine cycle a normal machine cycle is resumed unless the instruction decoder again decodes an I/O instruction.

4.3.1 16MHZ OSCILLATOR AND CLOCK GENERATOR

The basic timing source for the ND812 processor is a free-running 16 MHz oscillator which supplies an input to the clock generator. The 16 MHz input is counted down to a 4 MHz output (CPPU) which is used as the clock pulse for the timer. The Clock Generator also encodes the 4 MHz signal into a 2 MHz (or 0.5 microsecond) clock pulse which is used for processing an I/O instruction.

Table 4-7. Latches and Registers Cleared by START Key

Table 4-1 Simplified Logic Reference	LATCH OR REGISTER	Diagram Location
A	Power Sense	19B4
B	GO Control	3A3
C	Clear Latches and Registers	3B2
D	Cycle Steal Register	4A2
D	Interrupt Permit	4A4
E	Power Interrupt Latch	3A2
F	Double Register	4A4
G	Instruction Register	17B1-17B3
H	Flag Register	10B4
I	Pulser	5A1
J	Major State Control	5B2
K	Indirect Latch	6A2
L	Overflow Register	11B4
M	Priority Register	11B3
N	Interrupt Latch	11B2
P	Enable Instruction-Cleared Latches	3B3
Q	Select J or K Latch	3B4
R	First/Last Memory	3B4
S	First/Last Locate	3B4
T	Execute FF Latch	6A3
U	Indirect Latch	6A2
U	Indirect FF Latch	6A3
V	Indirect Memory Latch	6A3
W	KOM Register	8B1
X	Wait Register	8B2
Y	Once Register	8B3

Table 4-8. Start Control, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	START	SET RUN STATE 1. ↓START*→↑START/3A1 2. ↓START* + ↓CONT*→↑STCNT→ ↓CLKGO*/3A3 3. (↓CLKGO*) (↑CNTRP) (↑RGO*) (↑EXPMR*) = SET→GO CONTROL (↑GO/3A3, ↓GO*) 4. ↓START→↑START/3A1 5. (↑START) (↑GO*)→↓STCLR*→ ↑STCLR→↓STDN7*/3B3	Run state set and latches and registers cleared at start and continue.

4.3.1.1 16 MHZ OSCILLATOR. The 16 MHz Oscillator (19B2) consists of two cross-coupled high-gain amplifiers. These amplifiers operate as a crystal-controlled astable multivibrator which provides a 16 MHz output. Two output signals are provided to the clock generator and timing circuits, 16MC and 16MCD, both of the same polarity, but the 16MCD signal is delayed from the 16MC signal by about 25 nanoseconds (Figure 4-8, waveforms A and B).

4.3.1.2 CLOCK GENERATOR. The clock generator (Table 4-9) is a divider that counts down the 16 MHz clock pulses by a factor of 2 twice, i.e., once to 8 MHz and once to 4 MHz (waveforms C and D, Figure 4-8).

4.3.1.2.1 8 MHz Countdown. The 8 MHz countdown logic (Table 4-10) consists of a half dual J/K integrated circuit (4A1) which is controlled through the power sense and go control logic (Table 4-3). Actually, enabling signal BLKMC* is dependent on the level of GO* and BDPWR signal becuase when the power sense latch is set (19A4 and Table 4-1) the BDPWR signal level is high and the control gate is primed; when bad power is detected, the clock is shut off after the go control logic is initialized.

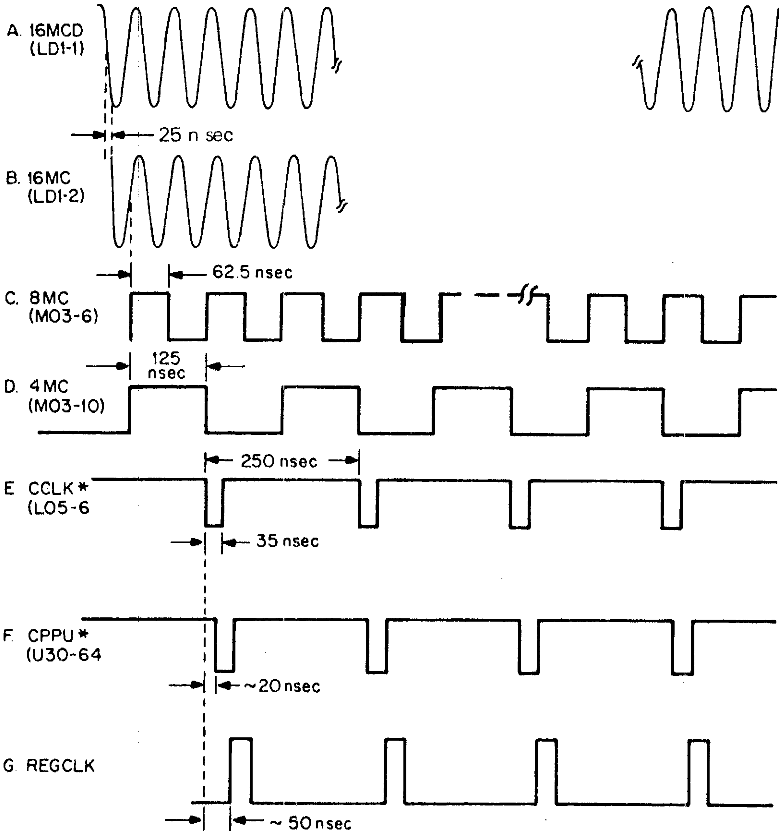


Figure 4-8. Basic Timing Pulses for the ND812

Table 4-9. Clock Generator, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	CLOCK	↑16MC/19B3 ↑16MCD/19B3	Free-running oscillator on.
B.	LOW PWR (ANY)	BLOCK 4MC 1. ↓BDPWR*→↑BDPWR/4A1 2. (↑BDPWR) (↑GO*)→↓BLKMC*/4A2	Increase clock rate on low power sense.
C.	PU6	SET 4MC 1. (↑OP4567) (↑PU6) (↑CNTR*) (↑ I4I5)→↓S4MC*/4A2 2. ↓S4MC*/4A2 = ENABLE RESET 3. (↓BLKMC*) (↑S4MC*) = CLR→4MC REGISTER (↓4MC/4A2, ↑4MC*)	Extends PU6 for shift and rotate instruction.
D.		ENABLE 4MC 1. ↓OP4567 + ↓PU6 + ↓CNTR* + ↓I4I5→↑S4MC*/4A2 2. ↓BDPWR + ↓GO*→↑BLKMC*/4A1 3. (↑S4MC*) (↑BLKMC*)→ENABLE→ TOGGLE	Enables normal clocking at end of shift and rotate instruction.
E.		16:8 MC DIVIDE (↑BLKMC*) (↑OSCPUL) (↑16MC) = 8MC (↑8MC/4A2, ↓8MC*)	Generates 8MC pulse.
F.		8:4 MC DIVIDE (↑8MC) (↑BLKMC*) (↓8MC*) (↑16MC) = 4MC (↑4MC/4A2, ↓4MC*)	Generates 4MC pulse.

4.3.1.2.2 4 MHz Countdown. The 4 MHz countdown logic is provided by the other half of the J/K integrated circuit (4A2). This countdown logic is enabled by the output of the 8 MHz countdown and its control logic. This control logic inhibits output pulses from the 4 MHz countdown by forcing the 4MC to stay in the set state until CNTR* (shift counter complete) goes low. A Group I operate instruction requires a shift or rotate of the contents of the J or K register. This is done to extend period BP6 of the basic phase to accommodate a longer basic phase to accomplish a maximum shift of 15 during the occurrence of the basic-phase time state when a Group I operate instruction has a 4, 5, 6, or 7 in the second octal character (bits 3 through 6 of the Group 2 instruction word).

Table 4-10. Clock Control, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	ANY	CLOCK CONTROL 1. (↑8MC) (↑4MCD*) (↑16MCD)→ ↓CCLK*/4A3 2. ↓CCLK*→↑CCLK/4A3 3. ↓I/O OSC + ↓SPDBL*→↑BLKCC*/4A3 4. (↑CCLK) (↑DBLE*) (↑BLKCC*)→ ↓CPPU*→↑CPPU→↓CPPUD*→ ↑REGCLK/4A4	Controls generator of clock pulses.
B.	PU6	CLR JK REGISTERS 1. ↓4MC*→↑4MCDD→↓4MCDD*→ ↑4MCD/4A3 2. (↑4MCD) (↑PU6)→↓CLRKJ*→ ↑CLRKJ/4A3	Clears J and K register.

Inhibiting the 4 MHz countdown logic when a shift or rotate Group I operate instruction has been called for, permits time period BP6 to be extended until the number of shifts or rotates called for in the instruction have been completed. The clock pulses used to count the shifts or rotates are derived from the unhibited 8 MHz divider. When the required number of shift or rotate operations have occurred, the 4 MHz countdown resumes normal operation.

4.3.2 PULSER

The pulser (Figure 4-9) consists of two 4-state shift registers and output gating (sheet 5). Pulser operation is initiated by a 0.25 microsecond CPPU* clock pulse from the clock generator, a master reset, and a pulser start signal. Clock pulses are fed into the CP inputs of the pulser shift register (first stage) and clocked through the eight-stage register at the 0.25 microsecond rate producing eight equal time periods (PU control pulses). However, provisions are made to change the state of the last four pulser shift register stages for certain operating conditions. For example, when an overflow occurs, when the reiteration loop is still active for MPY and DIV instructions, or when the operate instructions are coded for shift and rotate operations and the up-counter is preset to the number of desired shifts. This feature permits the processor to loop or hang-up in a time period until the BP6 shift or rotate operations are complete.

4.3.2.1 PU PULSES. The PU pulses control the generation of the basic phase (BP0, BP1, and BP3 through BP7) and execute phase (EP0, 1, 3, 4, 5, and 6) pulses used in the normal machine cycle. This arrangement makes it convenient to discontinue normal timing (when the current instruction cycle is complete) if an interrupt request is received. The PU pulses are also gated with the I/O OSC signal from the instruction decoder to generate the peripheral control pulses (PCP0, 1, 2 and 3). A major state control circuit together with the pulser logic prevents the execute and basic phases of a machine cycle from being active concurrently. This major state control logic is configured to ensure that pulse EP0 follows pulse BP7 if the decoded instruction is one which requires an execute phase.

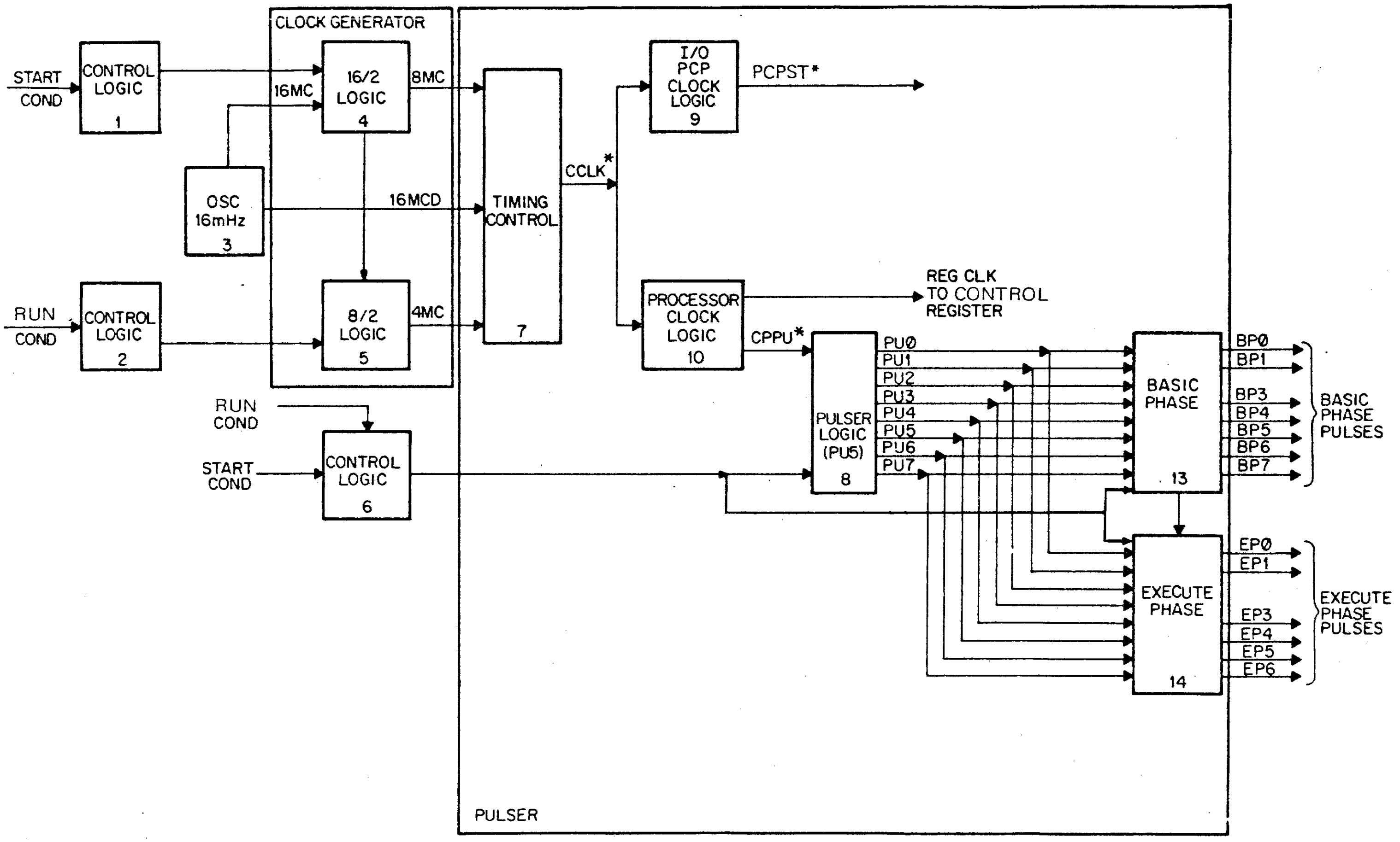


Figure 4-9. Clock Generator and Timing Logic, Block Diagram

4.3.2.2 PULSER CONTROL. When ths START switch is operated, pulser control logic is initialized. At first, assuming that the START switch has not been operated, application of power operates the power sense latch (Table 4-2 and 19A4) so that the initial low-level PWRDY* signal resets the pulser control register (3A4 and Table 4-2) and signal ST1 is low. The initialized low-level PU7B signal (3A4) produces a high level at the pulser control clear input and the low level PWRDY* signal produces a low-level at the register reset input. With these input conditions in effect, the pulser control register output, ST1 is a low-level signal which inhibits operation of the pulser (5AB1).

Pulser control is enabled when the go control logic (Table 4-2 and 3A3) is set to its run state and is inhibited when certain manual operations are performed. When the front panel START switch is depressed, the go logic is enabled and the GO signal goes from its previous low level to its high level state. However, no change has occurred except that the register is now primed and a start or continue can be accomplished. The toggle configuration is enabled when the PWRDY* signal goes high due to the enabling of the go logic (3A3, 19A4, and Table 4-11). Whenever the LOAD AR, LOAD MR, or the NEXT WORD front panel switches are operated a one-shot pulse generator (2A4) is enabled. This pulsed output is applied to the enabling gates of the pulser control when any of these switches are released simultaneously with the toggle input provided by the pulsed EXPMR* signal which is produced by the externally pulsed memory register logic (6B1 and 6B2). This circuit operates only when the ND812 is in its STOP state, or before the front panel START switch has been operated, due to the inhibiting GO* signal at the inputs of the externally pulsed memory register logic (6A1 and 6A2) and the toggle input of the pulser control (3A3).

4.3.2.3 PULSER LOGIC. The pulser consists of two ganged 4-bit shift registers (5AB1). Initially a master reset occurs when the power sense logic is set and when the START switch is operated. Operation of the shift register is dependent on the state of the ST1 signal and the incoming CPPU* clock pulses. When the master reset occurs, all outputs from the pulser are low-level, but the clock pulses have no effect because the ST1 signal is low-level. The J/K* inputs of the pulser shift registers permit D type entry when both are high-level at the same time. The output of a flip-flop follows the enabling input at the occurrence of the next clock pulse; thus, the shift register J/K* inputs are held low-level by signal ST1, and the first stage of the shift register which has been set to low level by the master reset remains low. When signal ST1 becomes high level, the D entry is enabled and the first clock pulse produces a high level PU0 signal which resets ST1 and disables D type entry. With each subsequent clock pulse the last stage previously set to its high-level output goes low, and the next stage goes high. Thus a series of high-level pulses (Figure 4-10, waveforms F through M) occur moving down the output of the shift register to the last stage.

Referring again to the pulser control logic (Table 4-11), it can be seen that the set and reset inputs receive pulses PU0* and PU7B. These pulses enable the pulser control to continuously recycle the pulser. If the recycling action did not occur, only light PU pulses could be generated. Recycling can be more easily understood by referring to the waveforms in Figure 4-10.

Table 4-11. Pulser Control, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	T'ON, START	CLR→PULSER REGISTERS	
		1. ↓STCLR* /5A1 = CLR→PULSER (↓PU0 - ↓PU7)	Refer to Table 4-1.
B.	START	START PULSER	
		1. ↓GO*→↑CKSTI*/3A4	Enables pulser.
		2. ↓PU7B + ↓GO + ↓DBLE*→↑SSTI*/3A4	
		3. (↓PWRDY*) (↑PU0*)→↓RSTI→ ↑RSTI*/3A4	
		4. (↑SSTI*) (↑RSTI*) = TOGGLE→ENABLE	
		5. (↑CKSTI*) (↑BLSTI*)→↑STI/3A4 = START→PULSER	
C.	PU0	CYCLE PULSER	
		1. ↓PU0*→↑RSTI→↓RSTI*/3A4	Pulser recycled at period PU0.
		2. ↓PU7B + ↓GO + ↓DBLE*→↑SSTI*/3A4	
		3. (↑SSTI*) (↓RSTI*)→↓STI/3A4 = CONTROL→OFF	
		4. (↑PU7B) (↑GO) (↑DBLE*)→↓SSTI*/3A4	
		5. (↑PU0*) (↑PWRDY*)→↓RSTI→↑RSTI*/3A4	
		6. (↑RSTI*) (↓SSTI*) = ↑STI/3A4 = CONTROL→ON	
D.	ALL	GENERATE PUs	
		1. (↑STI) (↑CPPU*) = ↑PU0/5A2, ↓PU0*, ↓STI	All clocked events depend on output of PU pulses
		2. (↑PU0) (↑CPPU*) = ↑PU1/5A2, ↓PU0	
		3. (↑PU1) (↑CPPU*) = ↑PU2/5A2, ↓PU1	
		4. (↑PU2) (↑CPPU) = ↑PU3/5A2, ↓PU2	
		5. (↑PU3) (↑CPPU*) = ↑PU4/5B2, ↓PU3	
		6. (↑PU4) (↑CPPU) = ↑PU5/5B2, ↓PU4	
		7. (↑PU5) (↑CPPU) = ↑PU6/5B2, ↓PU5	
		8. (↑PU6) (↑CPPU*) = ↑PU7/5B2, ↓PU6, ↑STI	
		9. (↑PU7) (↑CPPU*) ↑PU7/5B2	

When the pulser is enabled CPPU* clock pulses (A, Figure 4-10) are applied to the clock input of the two shift registers, however, until signal ST1 goes to its high-level state, these clock pulses have no effect (B, Figure 4-10). When ST1 is permitted to go positive, the next clock pulse applied to the shift register input will cause the first stage output to also go high and pulse PU0 (F, Figure 4-10) is produced as an output. However, because an inverted PU0 pulse is fed back to the pulser control, the positive-going PU0 and RST1* (C, Figure 4-10) pulse produces a low-level ST1 signal (shown when CPPU* pulse 0 is produced). This low-level ST1 signal resets the first stage of the shift register when the next CPPU* pulse is procuded (1, waveform A, Figure 4-10). Signal PU0 goes to its low-level state and remains there until the ST1 signal again goes to its high-level

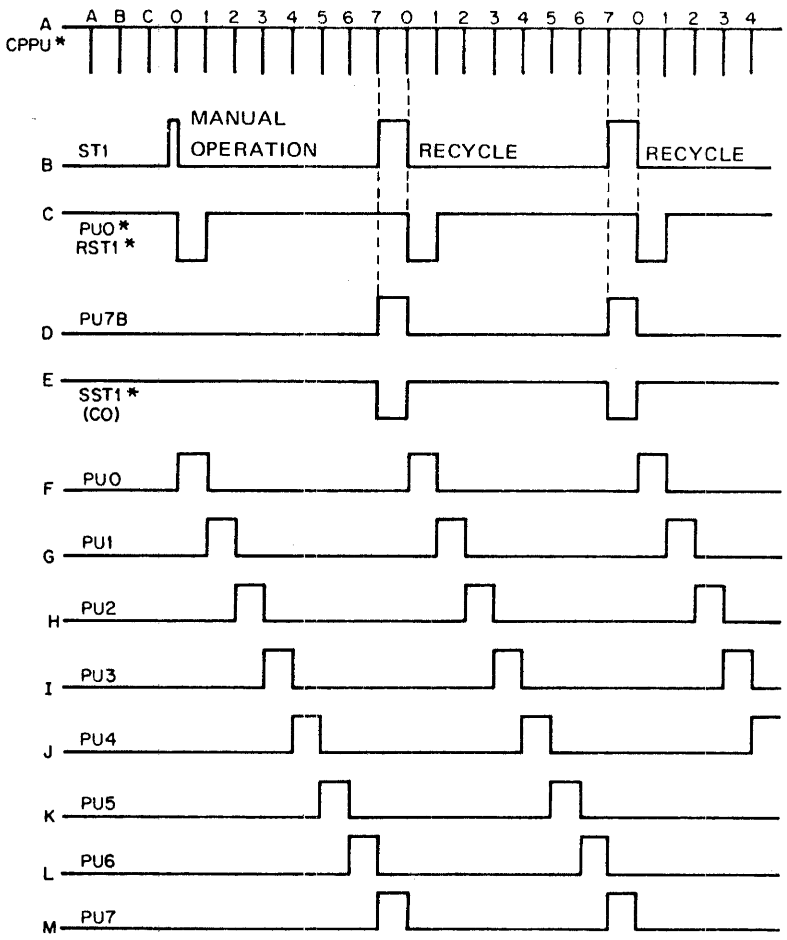


Figure 4-10. Pulser and Pulser Control, Timing Diagram

state. Subsequent input clock pulses perform the same action with respect to setting and resetting each stage of the shift register until pulse PU7 (waveform M) is produced. When pulse PU7 is produced, pulse PU7B (waveformD) is also produced and applied to the pulser control (3A4). When the next CPPU* clock pulse is produced the cycle is repeated.

Sheet 5 of the logic drawings shows the pulse and major-state control logic. In some cases, some of the PU, BP, and EP pulses are ORed to provide stretched timing functions. The stretched PU pulses that are generated are derived from pulses PU0*, PU1*, PU4*, PU5*. Pulses PU0 and PU1 are ORed to produce pulse PU001 (5A2), and pulses PU4* and PU5* are ORed to produce pulse PU45 (5A2). The stretched BP pulses that are generated are derived from pulses BP4*, BP5*, BP6* and BP7*. Pulses BP4*, BP5*, and BP6* are ORed to produce pulse BP456 (5A3) and pulses BP6* and BP7* are ORed to produce pulse BP67 (5B3). Also pulses BP5* and BP6* are ORed to produce pulse BP56 (11A2).

The stretched EP pulses are produced when pulses EP4 and EP5 are ORed to produce pulse EP45 (5A4).

4.3.3 MAJOR STATE CONTROL LOGIC

The machine cycle phase is controlled by the major state control logic (5B1 and 5B2) consisting of two dual J/K registers. One of these registers controls the generation of the major state phases and the other blocks generation of both basic and execute phases.

Operation of the major-state control logic depends on the following conditions.

a. That an interrupt request has not been initiated during the last machine cycle.

b. That no external device has requested a cycle-steal for a transfer from or to memory.

c. That the indirect bit (bit $4_2 = 0$) has not been set if the previous instruction was an MRI.

d. That the previous instruction was not an execute (7000_8) instruction.

e. That the previous instruction was not a two-word MRI.

f. The done latch is not set.

Four of the above instructions normally reset the basic phase because more than one basic phase is required.

The major state control is initialized through the start clear circuit when the front panel START switch is operated (5B2 and Table 4-12). The phase blocking register is initialized through the go-control logic (Table 4-8), the pulser, the interrupt, and the DMA control logic circuits. These circuits are described in the I/O Processor section of this manual. However, to initialize the phase blocking register an interrupt or cycle-steal request must not have been previously made, a go condition must exist, and pulse PU0 (Figure 4-9) must have been generated by the pulser.

When these conditions exist the major-state logic is initialized and either basic- or execute-phase pulses can be generated, and, because the STCLR* signal applied from the start-clear control (3B3) is normally high during the run condition, the major-state control register is normally held in its toggle enable configuration. Thus, when the J/K register inputs are both in their high-level state, the major state control will change states with the occurrence of each CPPU* clock pulse coincident with each PU7 pulse from the pulser.

The establishment of the basic and execute phases are dependent on the state of the done latch and various other previous conditions which may have been established

Table 4-12. Major State Control, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	START INTER-RUPT, CYCLE STEAL	ENABLE MAJOR STATES 1. (↓STCLR*) (↑OSCPUL) = SET→BPH (↓EPH, ↑BPH/5B2) 2. ↓INTCSF*→↑INTCS*/5B2 3. (↑INTCS*) (↑PU0) (↑GO)→↓RBLK*/5B2 4. ↓GO* + ↓PU0B→↑SBLK*/5B1 5. (↓RBLK*) (↑SBLK*)→↓BLKBEP/5B2	Start sets basic phase interrupt or cycle steal blocks both phases.
B.	PU7 NOT DONE	SET→EXECUTE PHASE 1. (↑DONE*) (↑TW*) (↑IND*) (↑XCT*)→↓RBPH→↑RBPH*/5B1 2. (↑OSCPULL) (↑STCLR*) = TOGGLE→ ENABLE 3. (↑RBPH*) (↑PU7) = SET→EPH (↑EPH/5B2, ↓BPH)	Execute phase set when done latch
C.	PU7 DONE	SET→BASIC PHASE 1. ↓DONE* + ↓TW* + ↓IND + ↓XCT*→↑RBPH/5B1→↓RBPH*/5B1 2. (↑OSCPULL) (↑STCLR) = TOGGLE→ ENABLE 3. (↓RBPH*) (↑PU7) = SET→BPH (↓EPH/5B2, ↑BPH)	Basic phase set when done latch is set at PU7.

during execution of an instruction. For example, when the execute phase of an instruction has been completed, pulse EP6* sets the done latch (7B4). When the done latch is set, it remains in its set state until pulse BP1 occurs so that during normal operation, the done latch set state overlaps two contiguous instructions except when a DMA or interrupt request occurs.

However, because operate instructions require no execute phase, the basic phase is re-entered at the conclusion of a previous basic phase because the done latch is set at the end of the basic phase rather than at the end of the execute phase. Also, this occurs for multi-basic phase instructions, particularly an indirect and a two-word instruction. The instructions that cause the basic phase to be re-entered after the conclusion of a basic phase are: a two word, a halt, an input/output, all the literal instructions, operates, and jump and indirects. All these instructions permit the done latch to be set by pulse BP6 (7A3) rather than EP6. Referring to Table 4-12, it can be seen that from B, the execute phase is set only when the done latch is not set and a two-word, execute, or indirect instruction is not in effect. Also, from C of the same table it can be seen that the basic phase is entered only when the done latch is set or a two-wrod, execute, or indirect instruction is in effect.

The major state control J/K* inputs assume their high-level state if a multi-basic phase instruction has not been sensed when pulse PU7 is present (Figure 4-11).

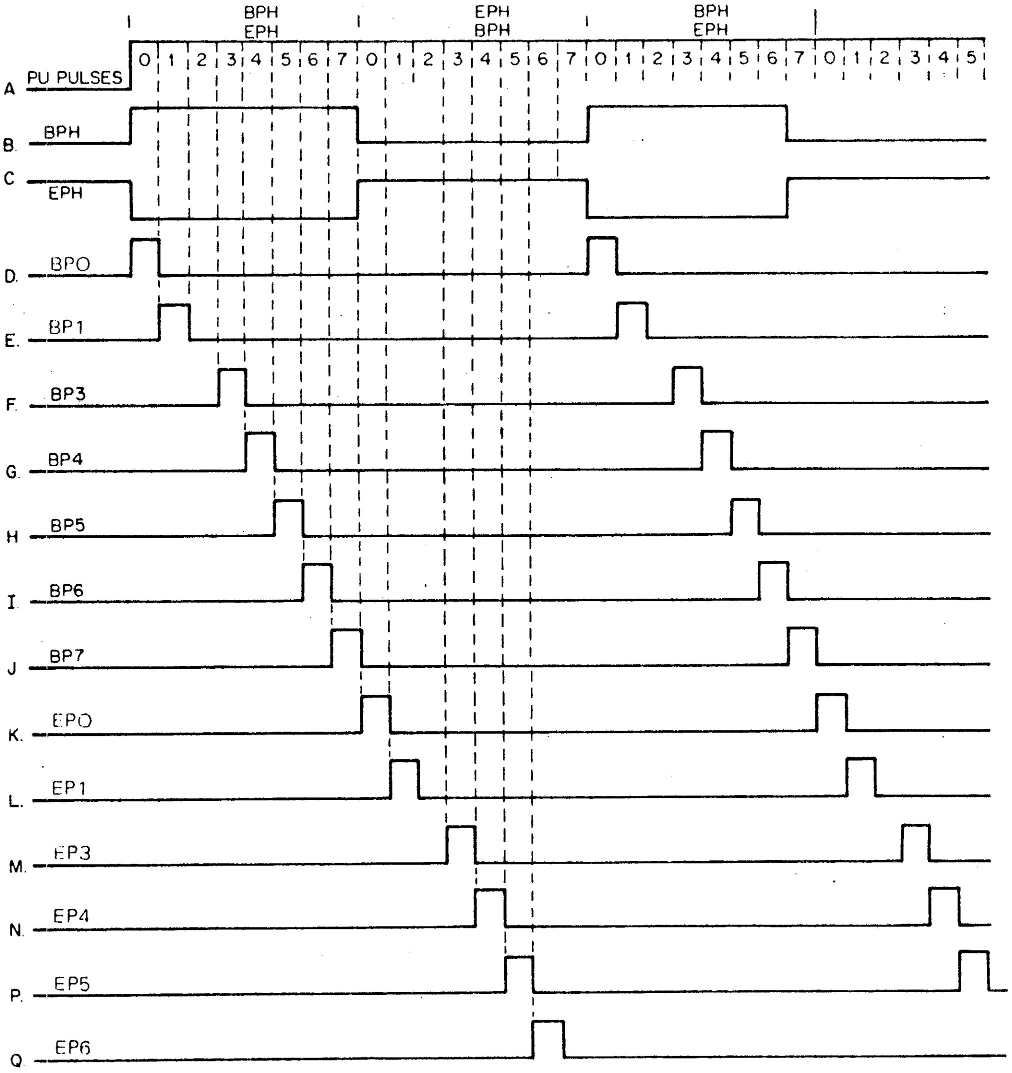


Figure 4-11. Basic and Execute Phase Pulses, Timing Diagram

When a cycle steal (DMA) or interrupt is sensed the major-state blocking register is primed to toggle (the clear input goes to its high-level state) except when pulse PU0 occurs so that at the beginning of each machine cycle the blocking register is reset to its non-blocking state. However, if a cycle steal or interrupt is sensed, pulse PU0 cannot reset the blocking register and it remains in its set configuration. Thus, during an interrupt or cycle steal, when pulse PU7 occurs simultaneously with clock pulse CPPU*, the blocking register is set to its block state (BLKBEP goes to its high-level state) and basic and execute phases are inhibited until the requesting peripheral device has been serviced. Because blocking occurs only in the presence of pulse PU7, the current phase is completed before the machine cycle is deferred, nor can the next phase take place until the cycle-steal request has been serviced and pulse PU0 occurs; this permits the blocking register to be set to its non-blocking state once more.

4.3.3.1 CONTROL REGISTER CLOCK/ENABLE LOGIC. The Register Clock/Enable Logic generates the clock pulses (CPK, CPJ, etc.) for the major registers of the ND812 processor. These signals are generated as a function of the decoded instruction, an I/O request, or control panel operation at discrete times during the execute (EP), basic (BP) or PU phase of the instruction cycle. The registers are 12-stage synchronous storage registers that can accept parallel data on each positive-going (low-to-high) transition of their input clock pulses. When the parallel enable signal is low, the input parallel data bits determine the next condition of the shift register when the clock pulse is generated.

4.3.4 CONTROL REGISTERS

The various registers in the ND812 are used to store data. They permit the processor to control the program, to skip instructions or data, permit the various arithmetic and logical operations to be carried out, and permit the program to access memory for reading or writing data. The control registers hold a 12-bit data word during various operations that take place during the execution of a given instruction. They are all of the same general type and are operated in a similar manner. However, the J and K accumulators, which are special registers used for arithmetic operation, can function as storage registers or as shift registers.

4.3.4.1 INSTRUCTION DECODERS. There are two instruction decoders, one detects the instruction operation codes and class of instruction and the other detects codes of operate-class instructions. Each instruction decoder consists of two Type 9301 one-of-ten decoders. The inputs of these decoders are obtained from specific bits of an instruction word. The instruction decoder that detects the instruction operation code and class of instruction is called the primary instruction decoder; the instruction decoder that detects the various operate class of instructions is called the operate decoder.

4.3.4.1.1 Primary Decoder. The primary instruction decoder (7A1) receives bits 0, 1, 2, and 3 of the word when an instruction is processed. The outputs from the primary instruction decoder are always available to the processor data control logic. Whenever an instruction is processed, only one output from one of the primary one-of-ten decoders is asserted for any four-bit combination of instruction register inputs. The primary decoders actually function as one-of-eight decoders with the A3 input of the selected decoder N20, or P20 active low. Inputs A0, A1, and A2 address an active low to any of eight outputs (0 - 7). Thus, each of the primary decoders is capable of detecting one of 8 (2^3) possible input combinations so that together they can decode any one of the 16 instructions ($2^3 + 2^3 = 16$). Table 4-13 lists the primary instruction decoder outputs for any one of the 16 instructions. The Op Code column gives the octal value of the operation code, the first digit of the operation code is the octal value of bits 0, 1, and 2 of any instruction, and can range between 0_8 (000_2) and 7_8 (111_2). The second digit of the operation code is the octal value of bit 3 of any instruction. If bit 3 is not set in the instruction word, the octal value of this digit is 0; if bit 3 is set, the octal value of this digit is 4. The second column lists

Table 4-13. Primary Decoder Outputs

OP CODE	BINARY CODE I00	I01	I02	I03	SIGNAL NAME	DECODER OUTPUT
00	0	0	0	0	00*	N20-(13)
04	0	0	0	1	01*	N20-(12)
10	0	0	1	0	OP1*	N20-(11)
14	0	0	1	1	OP2*	N20-(10)
20	0	1	0	0	LIT*	N20-(9)
24	0	1	0	1	SID*	N20-(3)
30	0	1	1	0	DSZ*	N20-(4)
34	0	1	1	1	ISZ*	N20-(5)
40	1	0	0	0	SBJ*	P20-(13)
44	1	0	0	1	ADJ*	P20-(12)
50	1	0	1	0	LDJ*	P20-(11)
54	1	0	1	1	STJ*	P20-(10)
60	1	1	0	0	JMP*	P20-(9)
64	1	1	0	1	JPS*	P20-(3)
70	1	1	1	0	XCT*	P20-(4)
74	1	1	1	1	I/O*	P20-(5)

the four-bit binary code required to obtain the various decoded instruction outputs listed in the Signal Name Column. Any output from the primary instruction decoder is asserted when the signal is low-level; all other primary decoder outputs are high level. The Decoder Output column lists the terminals which will be asserted for each of the input operation codes.

4.3.4.1.2 Operate Decoder. The operate decoder (11A1) provides asserted outputs whenever the primary instruction decoder detects a group 1 operate instruction. The Group 1 operate instructions enable various arithmetic and logical functions. These arithmetic and logical operations deal with data in the accumulator registers. They include data shifts and rotates, ANDing operations, subtractions and additions, and multiply and divide. The operate decoder receives the current bits 6 through 11 from the instruction register; however, its outputs are not asserted unless the primary instruction decoder has detected a Group 1 operate instruction. Whenever a Group 1 operate instruction is detected, only one of the 13 outputs available from the operate decoders is asserted. Its output depends on the input bit configuration.

In particular, if bits 4 or 5 or both are set, an arithmetic operation is called for. If bit 6 is set, a rotate or shift is called for. If bit 6 and bit 7 are set, a rotate is specified. If bits 4 and 5 are unset, the operation is a hardware multiply or divide, depending on whether bit 11 is set or unset. If the 11th bit is unset, the operation is a multiply; if the 11th bit is set, the operation is division.

Table 4-14 lists the operate decoder outputs for any of the Group 1 operate instructions. The various combinations of bit patterns permit the selection of 51 unique arithmetic operations. The Op Code column gives the octal value of the operation code

Table 4-14. Operate Decoder Outputs

				BINARY CODE			SIGNAL	DECODER
CODE	I06	I07	I08	I09	I10	I11	NAME	OUTPUT
0**	0	0	0	X	X	X	OP100*	M20-13
1**	0	0	1	X	X	X	OP101*	M20-12
2**	0	1	0	X	X	X	OP102*	M20-11
3**	0	1	1	X	X	X	OP103*	M20-10
4**	1	0	0	X	X	X	OP104*	M20-9
5**	1	0	1	X	X	X	OP105*	M20-3
6**	1	1	0	X	X	X	OP106*	M20-4
7**	1	1	1	X	X	X	OP107*	M20-5
0	X	X	X	0	0	0	HWM*	PO4-13
1	X	X	X	0	0	1	HWD*	PO4-12
2	X	X	X	0	1	0	RIST*	PO4-11
3	X	X	X	0	1	1	IOFF*	PO4-10
4	X	X	X	1	0	0	not used	PO4-9
5	X	X	X	1	0	1	not used	PO4-3
6	X	X	X	1	1	0	not used	PO4-4
7	X	X	X	1	1	1	I10I11*	PO4-5

**Arithmetic instructions
X don't care

for the various bit inputs to each of the decoders. The second columns list the three-bit binary codes required to obtain the asserted signals listed in the Decoder Output column. Any output from the instruction decoder is asserted when the signal is low level; all other operate decoder outputs are high-level. The Decoder Output Column lists the terminals which will be asserted for each input of the operation code bits 6, 7, and 8; and bits 9, 10, and 11.

4.3.4.2 INSTRUCTION REGISTER. The Instruction Register (17B1, 2, and 3) holds the current instruction until the execute phase has been completed. This register holds the 12 bits of the instruction word during the time an instruction is being processed. It receives its inputs from the bus and memory buffer multiplexer and provides its outputs to the instruction decoders, to the TS multiplexer, and to various logic elements in the processor. The primary instruction decoder receives bits 0, 1, 2, and 3, the operate decoder receives bits 6 through 11, and the TS multiplexer receives bits 6 through 11, and the processor logic receives all bits.

Three Fairchild Semiconductor Type 9300 4-bit registers (M19, N19, and P19, sheet 17) comprise this instruction register. These IC's are generally used as storage elements and not as shift registers, but when a two-word instruction is detected, an effective three-bit shift to the left is carried out by a single clock pulse which occurs during period BP3. Whenever register input PE is low-level, inputs PMR00 through PMR11 are admitted into the instruction register when a clock pulse (CP) is produced. For the instruction register, the enabling input signal is PEI* and the clock-pulse signal is CPI*. This register is cleared by a low-level MRI* signal which is produced whenever the done latch is set and pulse PU7B occurs. These signals are developed at the conclusion of either a memory reference, operate or input/output instruction. Refer to paragraph 4.4 for details. Data held in the instruction register is available until the conclusion of the machine cycle.

4.3.4.3 ADDRESS REGISTER. The address register (14B3, 15B3, and 16B3) holds the address of the location in memory currently being accessed. At the end of one instruction and before a new one is processed, the content of the address register is the last address accessed (for example, during an indirect or jump instruction). The address register holds 12 bits of address data (A00 through A11), which is derived from the adders via either the MX multiplexer, the TS multiplexer, or both (when a current address is summed with an effective address), for a jump (JMP) instruction or when the current address is incremented. Its outputs are provided to the MX multiplexer and tc the address decoders in the memory unit.

The address register also consists of three Fairchild Semiconductor Type 9300 4-bit shift registers (14B3-M06, 15B3-N06 and 16B3-P06). These IC's are not used as shift registers, but as storage registers. Whenever register input PE is low-level, inputs B00 through B11 are admitted from the adders into the address register when a low-level clock pulse (CP) is produced. Fcr the address register, the enabling input signal is PEA* and the clock-pulse signal is CPA*. The address register cannot be cleared because the steady-state high-level PULLU signal is present at the respective MR (reset) inputs; address register data is changed only when replaced by new data. Generally, the address register receives its inputs from the program counter through the MX multiplexer and adders at the beginning of an instruction; however, for memory reference instructions, this may be modified by an effective-address or replaced by an indirect pointer (second address).

When the ND812 is not in the run state 1 the content of the address register can be examined at the front panel SELECTED REGISTER indicators when the SELECT REGISTER switch is in the ADDRESS position. Refer to paragraph 4.2.2.7 for details of the selected register selection circuits. Details of address register operation are given in paragraph 4.4.

4.3.4.4 PROGRAM COUNTER. The program counter holds either the next location in memory from which an instruction will be read or the location of the current instruction being processed. At the beginning of an instruction it holds the next location in memory to be accessed; however, before the instruction is completed it is incremented to obtain the next address. Thus, the program counter holds only instruction addresses while the addres register can hold either instruction or data addresses. The program counter holds 12 bits of address data (P00 through P11) which is derived from the address via either the MX multiplexer, the TX multiplexer or both. Its outputs are provided to the MX multiplexer.

The program counter also consists of three Fiarchild Semiconductor Type 9300 4-bit shift registers as do the instruction and address registers (14B3-M05, 15B3-N05, and 16B3-P05). These IC's are also used as storage registers. Whenever register input PE is low level, inputs B00 through B11 are admitted from the adders into the program counter when a low-level clock pulse (CP) is produced. For the program counter, the enabling signal is PEP* and the clock-pulse if CPP*. The program counter cannot be cleared because the same steady-state high-level PULLU signal that disables the address register master reset input also disables the program counter master reset input. Thus, like the address register, it retains its information until it is replaced by new infomation. Generally, the program counter receives its first instruction location manually through the switch register (refer to paragraph 4.2.2.1). From then on, as a program is executed, the content of the program counter is automatically modified through the adders.

When the ND812 is not in the run state, the content of the program counter can be examined at the front panel SELECTED REGISTER indicators when the SELECT REGISTER switch is in the PC position. Refer to paragraph 4.2.2.7 for details of the program counter selection circuits. Details of program counter operation are given in paragraph 4.4.

4.3.4.5 MEMORY BUFFER REGISTER. The memory buffer register is the memory output link between the processor and memory. It receives parallel inputs from up to two 8K memory modules, including memory extension. Memory buffer register inputs are routed through connector R28 from the memory extension or from the basic self-contained memory modules (memory fields MF00 and MF01) through connector V29 and W29. Its outputs are supplies to the write amplifiers in the memory unit, so that any information read from a given location will be restored at the same location, and to the bus and memory buffer multiplexer. All words read from memory are strobed into the memory buffer register during period PU2 of any phase. The memory buffer register is cleared by low-level pulse RMB* which is derived through an inverter whenever pulse PU1B is generated.

The memory buffer register holds 24 bits of data in two 12-bit words. This 24-bit register system is used because of the 4K (MF00) + 4K (MF01) configuration of the basic 8K memory. The information obtained from the first 12-bit word is derived from one memory field and the data information obtained from the second 12-bit word is derived from the other memory field. When a memory extension is included as an option to the ND812, two more 4K fields can be added in an 8K stack making a total of 16K memory locations in four memory fields. The extended memory is also configured as two 4K memory fields.

The memory buffer register comprises six Fairchild Semiconductor Type 9314 4-bit latches. The first 4K memory field supplies outputs via lines MS00* through MS11* through connector W29 to the first field (MF00) memory buffer register (18AB1-M24, -N24, and -P24) and the second 4K memory field supplies outputs via data lines MS12* through MS23* through connector V29 to the second field (MF01) memory buffer register (18AB3-M25, -N25, and -P25). Memory extension inputs are configured similarly but extended memory data (MF10, and MF11) is coupled to the memory buffer register through connector R28. Memory buffer register outputs are applied as inputs to the bus and memory buffer multiplexer as signals MB00 through MB23. Word selection (MB00 through MB11 through MB23) is made at the multiplexer. The memory buffer register consists of six Fairchild Semiconductor Type 9314 4-bit latches configured in two groups of three (M24, N24, P24, M25, N25, and P25, sheet 18). With enable input E grounded, this terminal of these 4-bit latches is a static low-level signal which admits any low-level signal at inputs S_0, S_1, S_2, and S_3 when corresponding inputs D_0, D_1, D_2, and D_3 are high level. However, the D inputs are connected to static high-level signal PULLCK so that these latches are always enabled. When the master reset (MR) input goes low, these registers are cleared. Any set bits from memory are low level when asserted so that each memory buffer register remains cleared until the memory strobe (PU2) is generated during a basic phase.

4.3.4.6 MEMORY DATA REGISTER. The memory data register is the memory input link between the processor and memory. During a machine cycle, the memory data register holds any 12-bit word that must be written into memory during the current phase. It receives its inputs from the bus and memory buffer multiplexer. The bus and memory buffer multiplexer is an electronic switch that can provide as its output either memory buffer or adder information so that the memory data register can store either memory buffer or bus data from input lines PMR00 through PMR11. Memory data register outputs are also supplied from the memory data register to external equipment connected to the input/output bus through NAND gates that are enabled by I/O functions (17AB4). Memory data register outputs supplied to the self-contained basic 8K memory are routed through connectors W29 and V29. Information supplied through connector W 29 is for memory field MF00 and data supplied through connector V29 is for memory field MF01. Information supplied to the memory extension module is routed through connector R28 for memory fields MF10 and MF11. Data supplied to I/O devices is routed through connectors S26, S27, W30, Y30 or Z30.

The memory data register comprises three Fairchild Semiconductor Type 9300 4-bit shift registers (17A1-M21, 17A2-N21, and 17A3-P21); however, they are used as storage registers rather than shift registers. These registers are of the same type as the address register and program counter, enabled through the PE input and clock through the CP input. For the memory data register the enabling input is PEM* and the clock pulse input is CPM*. Like the address register and program counter, this register cannot be cleared because the steady-state high-level PULLCK signal is present at its respective reset inputs. Thus, like the address register and program counter, data in the memory data register is changed only when replaced by new information during a machine cycle.

The content of the memory data register is displayed by the front panel MEMORY REGISTER indicators. Refer to paragraph 4.5.4 for details of memory data register operation.

4.3.4.7 OTHER REGISTERS. The ND812 has other registers and latches most of which store single-valued bits of data. These registers include the up counter, which holds the value of a shift or rotate count, and the interrupt priority register which stores the current priority level under which the processor is controlled by the input/output processor. Also included is the overflow register and the flag register.

4.3.4.7.1 Overflow Register. The overflow register (11B4-A04) is a 1-bit JK register used to indicate an overflow during an arithmetic or a hardware multiply operation. The purpose of the overflow register is to indicate that the result of an arithmetic operation has exceeded the capacity of the accumulator. Each time an overflow occurs (as a result of an arithmetic operation), the overflow (OV) register is complemented. Thus, if overflow is possible as the result of an arithmetic operation, the status of the overflow register will indicate this condition. Thus, the programmer must monitor the condition of this register. The ND812 instruction set permits the overflow register to be set, cleared, complemented, and tested, but testing does not automatically clear it.

When an arithmetic operation requires two binary numbers to be summed, overflow occurs when the content of the affected register (the register in which the arithmetic operation occurs) is greater than 4095_{10} (7777_8). When overflow occurs the equivalent of the decimal quantity 4095_{10} must be added to the quantity left in the affected register to obtain the proper sum. This is not done by the hardware, but is an element of the bookkeeping requirements for the programmer. When an arithmetic operation requires two binary numbers to be subtracted, the twos complement of the subtrahend is added to the minuend. Thus, overflow is also possible when a subtraction is ordered by the instruction currently being processed, but the programmer must decide whether such an overflow condition has significance when determining the absolute value of the result. The theory of twos complement subtraction can be found in various elementary texts on computers.

The overflow register and logic receives timing inputs from the control unit, bits 6 and 7 from the IR, decoded instructions from the instruction decoder, and the carry output (COUT) from the most significant stage of the adder. These inputs are gated into the overflow register.

4.3.4.7.2 Up Counter. The up counter (11A3) is used when the instruction decoder detects shift or rotate or multiply and divide instructions. The up counter is a Fairchild Semiconductor Type 9316 four-stage binary counter which is preset to a value whose magnitude is the difference between the wanted count and 15. When the counter PE input signal SCNTR* is low level, signals I08*, I09*, I10* and I11* are admitted into the counter. These signals are the complements of the corresponding instruction register bits. Thus, if the up counter is to count out 12 pulses, the instruction register bits are set as follows: I08 = 0, I09 = 0, I10 = 1 and I11 = 1. However, because the complement of this value is loaded into the up counter, the bit configuration is that shown for 12 moves in Table 4-15. Figure 4-12 is the waveform diagram for operation of the counter. Taking the difference between 15 and 12 to be 3, the counter must be preloaded to the value 3. Thus, referring to Figure 4-12 when three CLKCNT pulses have occurred, the state of the counter is that shown in Table 4-12 for a count of 3, or, $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$, and $Q_3 = 0$. Counting the remaining pulses gives 4 through 15, or a total of 12.

The counter changes states with each low-to-high transition of the CLKCNT input pulses which are obtained through the operate logic or through the multiply and divide logic. There are two discrete countouts of importance. One of these is the terminal count which provides a high-level CNTR output signal when a full count of 15 has been reached; the other is a count of 12 which is detected by output AND gate D17. When this gate is enabled, a logical PS12* signal is produced that indicates the counter has received 12 pulses. This information is required for hardware multiply and divide operations. Details of up counter operation are described in paragraph 4.4, Processing Instructions.

4.3.4.7.3 Flag Register. The flag register (10B3-A20) is a one-bit register with associated gating used to store the status of any previous condition. It is programmer accessible

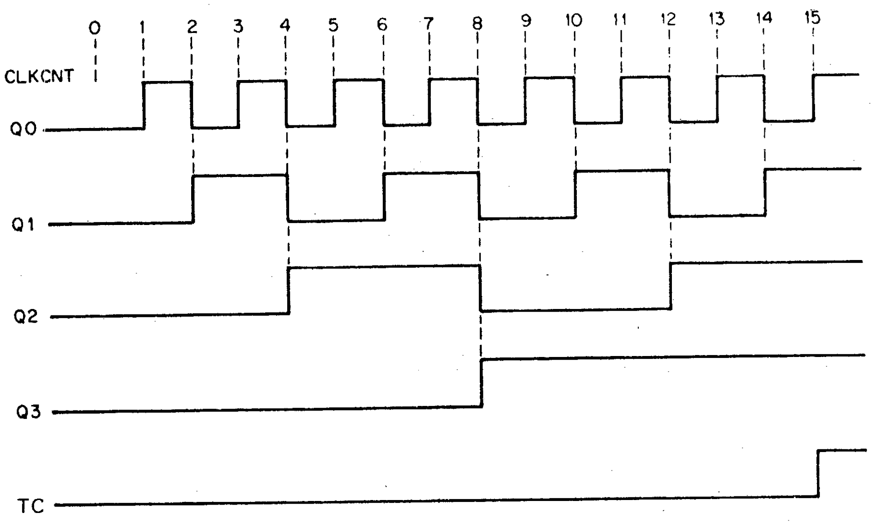


Figure 4-12. Up Counter, Timing Diagram

Table 4-15. Up Counter Preload Codes Vs. No. of Moves

Up Counter Preload Codes				
111*(Q_0)	110*(Q_1)	109*(Q_2)	108*(Q_3)	No. of Moves
1	1	1	1	0
1	1	1	0	1
1	1	0	1	2
1	1	0	0	3
1	0	1	1	4
1	0	1	0	5
1	0	0	1	6
1	0	0	0	7
0	1	1	1	8
0	1	1	0	9
0	1	0	1	10
0	1	0	0	11
0	0	1	1	12
0	0	1	0	13
0	0	0	1	14
0	0	0	0	15

through the group 2 operate instructions which permit the flag register to be set, reset, complemented, restored, and otherwise tested. In this manner, it can be used as a one-bit memory to store the presence of absence of some condition. Only the instructions which specifically address the flag register have any effect on its stage. The flag register output is also routed to the status input of the MX multiplexer. Details of flag register operation are described in paragraph 4.4.2.2.

4.3.4.8 ACCUMULATOR REGISTERS (J, K, R AND S). The accumulator registers are the arithmetic processing elements of the ND812 processor. All four of them are involved in additions, subtractions, multiplication and division. However, only the J and K registers are directly accessible to or from the memory. The R and S registers receive their inputs from the J and K registers, respectively. Data may be transferred between the J and K registers or between J, R, K, and S registers through the MX multiplexer; it may be shifted left in J and K registers, and it may be shifted right in R and S registers during hardware multiply. Arithmetic results can be called for either from the J or K registers. These registers provide the ND812 data path to the memory registers through the MX multiplexer so that the results of arithmetic and logical operations can find their way into specified memory locations. Data is transferred between the J and K registers and memory when certain memory reference instructions are processed. Arithmetic, logical, and multiply and divide operations are carried out by Group 1 operate instructions.

Each of these registers holds 12-bits of data. Arithmetic and logical operations cause data to appear in either the K or J register. The R and S registers receive their inputs from the J and K registers although their outputs are also routed to the MX multiplexer. When the operation is multiplication, the multiplicand is loaded into the K register and the multiplier is loaded into the J register. The product appears in the two subaccumulator registers, R and S with the most significant half of the product in the S register and the least significant half in the R register.

When division is the operation to be executed, the divisor must be loaded into the R register and the dividend must be loaded into the J and K registers. The most significant half of the dividend, a 23-bit word (the most significant bit must be zero) must be loaded into the K register and the least significant half into the J register. The division operation results in the quotient appearing in the J register and the remainder appearing in the K register.

4.3.4.8.1 Arithmetic Operations. Figure 4-13 is the block diagram of the ND812 arithmetic loop. In this configuration, the TS multiplexer (2), the MX multiplexer (3), and the adder (4) play an important part, not only in arithmetic operations, but in data exchanges, shifts and rotates, and in multiply and divide operations. For the purpose of programming, the R and S registers (7 and 8) are not directly accessible to or from memory; however, data contained in these registers may be transferred to the J or K registers (5 and 6) and then written into specified memory locations by a memory reference instruction. All the arithmetic instructions are group 1 operate instructions.

With the exception of the multiply instruction which leaves a 24-bit product in the R and S registers, the results of arithmetic operations are left in either the J or the K

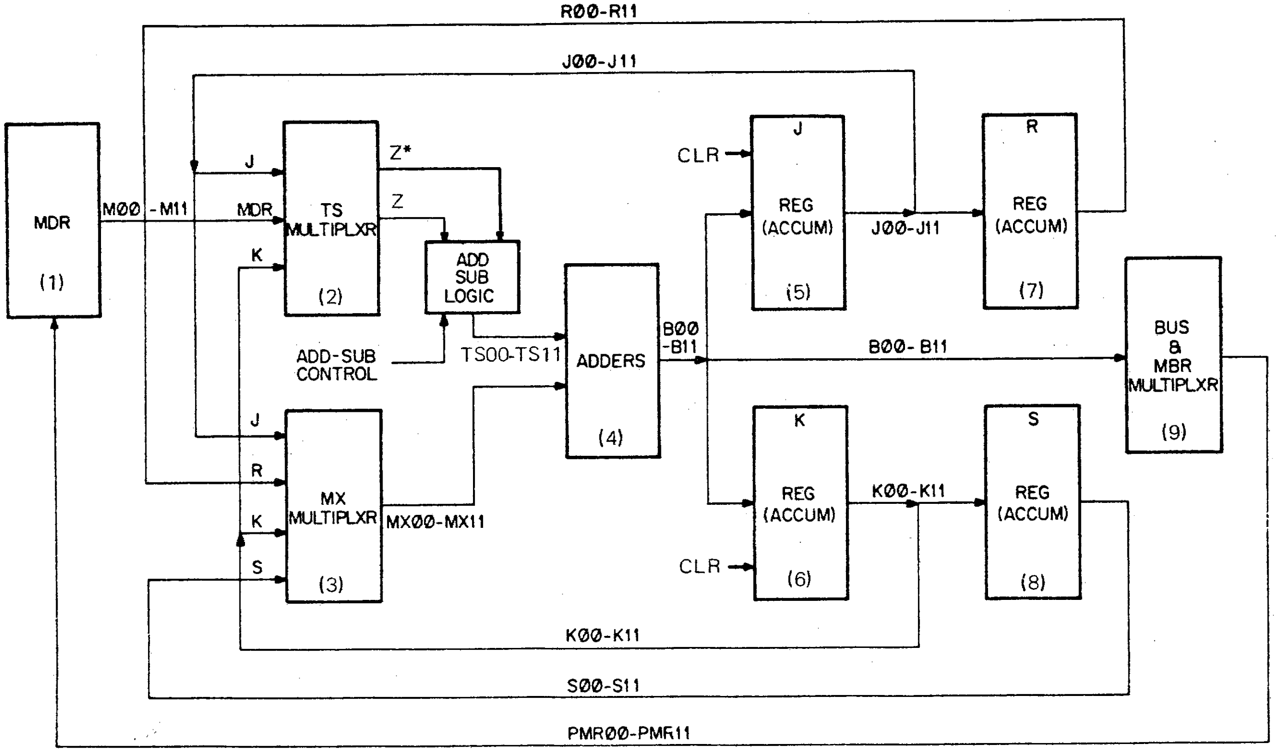


Figure 4-13. ND812 Arithmetic Data Loop, Block Diagram

registers. Upon inspection of Figure 4-13 the possible permutations of additions, subtractions, and data exchanges become apparent. For example, when the content of the J and K registers are summed and the result is to be left in the K register, one of the possibilities is to admit the output of the J register to the TS multiplexer and the output of the K register to the MX multiplexer. These data can then be summed by the adders and admitted to the K register.

Subtractions can be obtained in a similar manner; the complement of the content of the J register is taken from the TS multiplexer and incremented by a count of 1 to obtain the 2's complement of the J register data word (this logic is not shown but suggested in Figure 4-13), which is then summed with the content of the K register. Exchange and load operations permit other data transfers. It is interesting to note that although there is not a direct instruction that permits the content of the R and S registers to be exchanged, nevertheless, this is possible by first exchanging the data in J and K with the data in R and S (EXJRKS) and then exchanging the data between J and K (EXJK). Details of arithmetic operations are described in paragraph 4.4.2.1.

Although the bus and memory buffer multiplexer (9) and the memory data register (1) are shown, they are not used during addition, subtraction, or multiplication. They are, however, used during the divide operation. During divide operations, the memory data register is used to store a modified version of the divisor.

4.3.4.8.2 J Register. The J register holds 12 bits of arithmetic data (J00 through J11), or during the execution of certain memory reference instructions, a word transferred from or to memory, which is derived through the adders. The J register outputs are provided to the TS multiplexer, the MX multiplexer, and the R register (Figure 4-13).

The J register consists of three Fairchild Semiconductor Type 9300 4-bit shift registers (M16-14A3, N16-15A3, and P16-16A3), which are used as both storage and shift registers. Whenever register input PE is low level, inputs B00 through B11 are admitted from the adders into the J register when a low-level clock pulse CP is produced. For the J register, the enabling signal is PEJ* and the clock pulse is CPJ*. Whenever clock pulse CPJ* occurs and signal PEJ* is not low level, a single bit shift to the left occurs. The J register can be cleared when pulse MRJ* becomes low level. Thus, unlike many of the other registers in the ND812, the J register retains its data unless cleared by the low-level MRJ* pulse.

4.3.4.8.3 K Register. Like the J register, the K register holds 12 bits of arithmetic data (K00 through K11), or during the execution of certain memory reference instructions, an operand transferred from or to memory, which is also derived through the adders. The K register outputs are provided to the TS multiplexer, the MX multiplexer, and the S register (Figure 4-13).

The K register also consists of three Fairchild Semiconductor Type 9300 4-bit shift register (14A3, 15A3, and 16A3) which are used as both storage and shift registers. Whenever register input parallel enable is low level, inputs B00 through B11 are admitted from the adders into the K register when a low-level clock pulse is produced. For the K register, the enabling signal is PEK* and the clock pulse is CPK*. The K register can be cleared when pulse MRK* becomes low level. Thus, as with the J register, the K register retains its data unless cleared by the low-level MRK* pulse. Data in the K register can also be shifted one bit to the left whenever the clock pulse CPK* occurs when parallel enable signal PEK* is not low-level.

4.3.4.8.4 R Register. The R register holds 12 bits of data (R00 through R11), which are derived from the J register. The R register outputs are provided to the MX multiplexer (Figure 4-13). Data in the R register cannot be directly accessed by memory, but may be transferred through the J register.

The R register consists of three Fairchild Semiconductor Type 9300 4-bit shift registers (14A4, 15A4, and 16A4), which are used as both shift and storage registers. Whenever register input parallel enable is low level, inputs J00 through J11 are admitted into the R register when a low-level clock pulse is produced. For the R register, the enabling signal is PER* and the clock pulse is CPR*. The R register cannot be cleared because it has a static high-level PULLU signal at its reset input, MR*. Thus, like many other registers in the ND812, it retains its data until replaced by new data. Data in the R register cannot be shifted upon command; however, during a multiply operation, data in the R and S registers are shifted one bit to the right whenever clock pulse CPR* occurs and parallel enable pulse PER* is not low level. Details of the R and S rightward shift are described under Hardware Multiply (paragraph 4.4.2.1.1).

4.3.4.8.5 S Register. The S register is the K register subaccumulator and it functions in a manner similar to the R register in conjunction with the K register. The S register holds 12 bits of data (S00 through S11) which is derived from the K register. The S register outputs are provided to the MX multiplexer (Figure 4-13). The S register cannot be directly accessed by memory, but similar to the R register, it may accept data through the K register.

The S register consists of three Fairchild Semiconductor Type 9300 4-bit shift registers (14A4, 15A4, and 16A4), which are used as both shift registers and storage registers in a manner similar to that for the R register. Whenever register input parallel enable is low level, inputs K00 through K11 are admitted into the S register when a low-level clock pulse is produced. For the S register, the enabling signal is PES* and the clock pulse is CPS*. As with the R register, the S register cannot be cleared because it has a static high-level PULLU signal at its reset input, MR*. Thus similar to the R register, it retains its data until replaced by new data. Data in the S register cannot be shifted upon command; however, as with the R register, data in the S register is shifted one bit to the right during a multiply operation whenever clock pulse CPS* occurs and the parallel enable pulse PES* is not low level.

4.3.4.9 MULTIPLEXERS. The multiplexers are electronic switches that permit any one of a number of multiple inputs to be selected as their output. The multiplexers permit various data, held in the ND812 storage registers, to be manipulated for obtaining arithmetic operations, shifts, rotates, multiply, divide, data exchanges, complements, negations and logical operations to take place. There are three multiplexers in the ND812, the bus and memory buffer multiplexer, the TS multiplexer, and the MX multiplexer. The TS and MX multiplexers play a significant role in the ND812 arithmetic loop which has been described under Accumulator Registers (paragraph 4.3.4.8).

4.3.4.9.1 Bus and Memory Buffer Multiplexer. The purpose of the bus and memory buffer multiplexer (17A1,2,3 and 4) is to enable data stored in the memory buffer register or held by the adders to be returned to the memory data register. The bus and memory buffer multiplexer receives memory buffer outputs MB00 through MB23 and adder outputs B00 through B11. As previously described in paragraph 4.3.4.5, the memory buffer register provides memory data from memory fields MF00 through MF11 depending on the memory field selected.

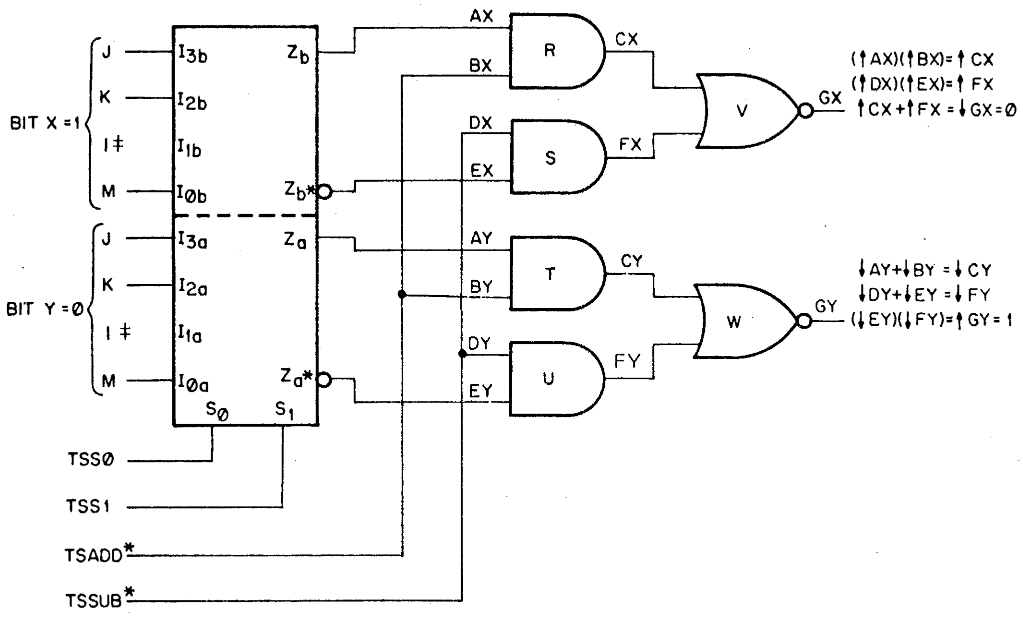
The bus and memory buffer multiplexer consists of six Fairchild Semiconductor Type 9309 Dual Four-Input Multiplexers having common input select logic (sheet 17). However, they are connected as Dual Three-Input Multiplexers because the adder inputs are connected to two input terminals. Depending on the configuration of the select logic bit pattern at terminals S_1 and S_0, only one of signals I_{0a} through I_{3a} and I_{0b} through I_{3b} of each multiplexer is selected from the bus and memory buffer multiplexer as signals PMR00 through PMR11. Table 4-16 lists the bit configuration for signal selection. The outputs from this multiplexer include both high-level and low-level assertion; however, only the high-level signal assertion is used. When signal RDB* is low-level adder outputs B00 through B11 are selected. These outputs appear at terminals Za and Zb of each of the six multiplexers integrated circuit modules. The bus and memory buffer multiplexer outputs are applied to both the memory data and the instruction registers (see Figure 1-2).

Table 4-16. Bus and Memory Buffer Multiplexer Outputs

ENABLING SIGNALS		
RDB*	ADDF1*	SELECTED INPUT SIGNAL
0	0	B00 through B11
0	1	B00 through B11
1	0	MB12 through MB23
1	1	MB00 through MB11

4.3.4.9.2 TS Multiplexer. The purpose of the TS multiplexer is two-fold: it is used to select outputs from one of four registers, and it is used in arithmetic operations to select a data word or its complement. The four registers providing data to the TS multiplexer are the J register, the K register, the instruction register, and the memory data register. Also used in conjunction with the TS multiplexer, and logically inseparable from it, is the add-subtract gate logic. This logic determines whether the register data word or its complement will be selected as the TS multiplexer output word. The output of the TS multiplexer is applied as one of the inputs to the adders. This adder input is summed with the output from the MX multiplexer which is also applied as an input to the adders. Thus, whenever a data-word or its complement is selected by the add-subtract logic, an addition or subtraction takes place at the adders with the result appearing as the adder output. During arithmetic operations, the adder is incremented when subtraction is ordered so that 2's complement subtraction occurs.

Figure 4-14 illustrates the add-subtract logic for one unit IC module of the TS multiplexer output. Assuming that the upper half of the integrated circuit module contains a bit of information, bit X, which is set, and the lower half contains a second bit of information, bit Y, which is unset, the following illustrates how either the data word or its complement is chosen for output from the TS multiplexer accordingly as an add or subtract is ordered from the add-subtract gate logic. Both the assertion and negation of each bit is available as outputs from the TS multiplexer and appear at outputs Z_b, Z_b*, Z_a, and Z_a*. When addition is ordered, signal TSADD* is asserted and low-level; TSSUB* is not asserted and is high-level.



I ‡ = IR BITS 0 THROUGH 5 NOT AVAILABLE TO THE TS MULTIPLEXER

Figure 4-14. Typical TS Multiplexer Add-Subtract Logic

Examining AND gate R for the effect of the low-level TSADD* signal and AND gate S for the effect of the high-level TSSUB* signal, it can be seen that the logic equations opposite OR gate W are satisfied for bit X and that bit X will appear as the data at the output of OR gate V. However, for bit Y, the conditions described by the logic equations opposite OR gate V are satisfied for bit Y so that the negation of bit Y, or binary 1, will appear at the output of OR gate W. Thus, when addition is ordered, the data word is selected as an output from the TS multiplexer because the add-subtract logic inverts its selected inputs.

When subtraction is ordered by the logic, signal TSSUB* is asserted and low level, and signal TSADD* is not asserted and high level. Examining the respective AND gates R, S, T, and U for the effect of these states, it can be seen that the conditions described by the logic equations opposite OR gates V and W, respectively, are obtained and that the complements of the bits X and Y are produced as outputs. Thus, an add command produces the data word, and a subtract command produces the complement of the data word, bit for bit. When neither an add nor subtract is ordered, the conditions described by the logic equations opposite OR gate V are obtained for all outputs from the add-subtract logic so that all outputs from the TS multiplexer will produce low-level outputs.

The TS multiplexer consists of six Fairchild Semiconductor Type 9309 Dual four-input multiplexers having common input select logic (sheet 13). Electrically, these integrated circuit modules permit any one of four input data words to be selected for application to the add-subtract logic, depending on the states of the select logic bit pattern at input terminals S_0 and S_1. Table 4-17 lists the bit configuration required for selection of the J-register (J00 through J11), the K-register (K00 through K11), the instruction register (bits I06 through I11 only), and the memory data register (M00 through M11) data words. It should be noted that in the absence of any selection command from the logic, signals TSS1 and TSS0 are low level so that the content of the memory data register is selected as the output data word. This condition occurs during the hardware divide operation.

Table 4-17. TS Multiplexer Selection Codes

Selection Bits TSS1	 TSS0	Signals	Source
0	0	M00-M11	Memory Data Register
0	1	I06-I11	Instruction Register
1	0	K00-K11	K Register
1	1	J00-J11	J Register

4.3.4.9.3 MX Multiplexer. The purpose of the MX multiplexer is to switch any one of eight input data words to the adders. The selected data word from the MX multiplexer is summed with the output from the TS multiplexer add-subtract logic (Figure 4-13) so that when a TS multiplexer data word is selected, the result of summation is addition; when a TS multiplexer data-word complement is selected the result of summation is subtraction; and when neither the data word nor its complement is selected, the result of summation is feed through, i.e., the selected MX multiplexer data word is summed with all zeros resulting is adder feedthrough of the MX multiplexer data word only.

Figure 4-13 shows only the registers in the arithmetic loop which provide inputs to the MX multiplexer; these registers include the four accumulators, J, K, R, and S. Other register inputs include the address register, the program counter, the utility gates, and the status register.

The MX multiplexer consists of 12 Fairchild Semiconductor Type 9312 (sheet 14, sheet 15, and sheet 16) Eight-Input Multiplexers having common input select logic. The MX multiplexer integrated circuit modules permit any one of eight input data words to be selected for application to the adders (Table 4-18) accordingly as the three-bit select signals S_0, S_1, and S_3 are high or low-level. Table 4-18 lists the input bit combinations of signals MXS0, MXS1, and MXS2 for selection of the various input data words. In order to admit input data to the MX multiplexer, parallel input enable signal MXEN* must be low level together with the required bit pattern. In the absence of any bit pattern signals, inputs MXS0, MXS1, and MXS2 are low level and the address register data word will be admitted whenever enable signal MXEN* is low level. Whenever signal MXEN* is disabled (high-level) the output from the MX multiplexer is zero (all binary 0's).

Table 4-18. MX Multiplexer Selection Codes

MXS2	MXS1	MXS0	Register
0	0	0	Address Register
0	0	1	Utility Gates
0	1	0	Program Counter
0	1	1	S. Register
1	0	0	J. Register
1	0	1	K. Register
1	1	0	R. Register
1	1	1	Status Register

4.3.4.10 ADDER. The adder is a parallel high-speed binary full adder consisting of six Type 9304 Fairchild Semiconductor integrated circuit modules (sheets 14, 15 and 16). Each of the six modules is capable of adding two bits; thus the six IC's provide a 12-bit capacity. Each stage of the adder is equipped with an A input (operand from MX Multiplexer), a B input (operand, from TS multiplexer), and a C input (carry from previous stage). Each stage of the adder also has three outputs. The sum(s) output is the sum of the A and B inputs (S=A+B). These outputs (B00 through B11) are placed on the sum bus for distribution to the accumulators (J or K), OUT, PR, or AR, (or IR and MR through the bus and memory buffer multiplexer). The S* output is the complement of the S output, and is routed to comparison logic to determine if all stages of the adder are in the zero state. The C_0 output is the output from the carry structure, which becomes the C input to the next higher order stage (except most significant bit). The C_0 output is approximately 8 nanoseconds delay from the sum outputs, and is the sum of the carry and/or the A and B inputs (i.e. C_0 = AB+BC+AC). The carry output (COUT) from the last stage is routed to the overflow register and logic.

4.3.4.11 UTILITY GATES. The utility gates are not actually registers, but 12 AND-OR invert gates that permit data words from two external sources or literal functions to be admitted into the MX multiplexer via input lines U00 through U11. External data word sources include the 12 front panel switch register switches and any peripheral data (EXT's) through which program-controlled word transfers are made. The literal functions permit literal data words to be ANDed with data from the J and K register with the result gated through the MX multiplexer. A literal is a direct representation of a symbolic data reference, thus, any representation of data which defines itself (a pure number) is a literal (refer to the ND812 Principles of Programming Text). Details of the utility gate operation are described in paragraph 4.4.9.

The utility gates consist of 12 Type 9008 Fairchild Semiconductor 4-input AND-OR Inverter Gates (sheet 14, sheet 15, and sheet 16).

4.3.4.12 STATUS REGISTER. The status register is not a register, but a collection of bits from various one-bit registers that are stored in the J register when the status word is called for by a programmed instruction. Table 4-18 describes the status register data format and the output bit configuration as it is collected in the MX multiplexer. Details of status register data collection are described in paragraph 4.4.2.2.

4.4 PROCESSING INSTRUCTIONS

There are 16 basic one-word instructions in the ND812 instruction repertoire. These 16 basic instructions are augmented through various bit permutations into more than several hundred one-word and two-word instructions, each obtaining a discrete operation of the processor. When the various possible input/output instructions are included in this number, the possibilities for discrete operations number in the thousands; these possibilities make the ND812 one of the most powerful mini-computers available today. This section describes how the various operate and memory reference instructions are processed by the ND812. These descriptions include a discussion at the block diagram level and at a detailed level. These diagrams provide the service engineer with a generalized understanding of how a given instruction is processed, the timing constraints, the registers affected, and the result of the operation. The detailed description is an analytical one, and is intended as an aid in trouble analysis. The fundamental operation tables describe the signals required for assertion of the required operations with the aid of a list of simplified logic expressions. These tables, together with the timing diagrams, offer detailed insight into the anatomy of instruction processing.

Operate and memory reference instructions serve complementary functions. The former are used as liaison between the central processor and arithmetic unit on the one hand and the central processor and the memory unit on the other hand. Memory reference instructions fetch data to the arithmetic unit which carries out some operation on the data. The resultant data is sent back to memory through the application of a separately programmed memory reference instruction. Operate instructions carry out all operations that require interaction between the various accumulators and the adder with its multiplexed inputs. In view of the above, the memory reference and operate instructions should be viewed as complementary instructions. Memory reference instructions supply and store the data and operate instructions manipulate the data according to the characteristics of the particular instruction.

The nearly one hundred instructions vary in their processing features considerably with respect to format, addressing, and execution time. Throughout this discussion, uniformity is sought; descriptions of common features between instructions have preference over repetitiously describing the same operation as it occurs for several instructions. This method of presentation gives a better insight into the underlying processing philosophy of the ND812. Some repetition, either serving to enhance familiarity with difficult operations, or for the sake of clarity, is retained, however.

Formats are illustrated for groups, or subgroups, of instructions along with their detailed descriptions. The various addressing schemes are discussed in detail in conjunction with memory reference instructions only, because operate instructions do not reference memory, and hence, addressing modes are not applicable.

In seeking uniformities, the basic phase is described first. The basic phase accounts for execution of the complete instruction for operate instructions, with the exception of shift/rotate operations, which require a phase extension in real time, but still are processed in one basic phase in terms of machine time. The memory reference instructions, in addition to one or more basic phases, also require an execute phase for completion. Because the basic phase is indispensible for all processing instructions, and because the operate instructions, which constitute the majority of instructions, require only one basic phase, the operate instructions are discussed, first. The following discussion treats only that part of the basic phase which is common to all processing instructions. The distinguishing features are given later as the detailed description for the instruction is provided.

The instruction descriptions generally include a block diagram which illustrates the data transfer paths invoked by the called instruction and a timing diagram which shows various time related functions of the hardware control logic together with two important tabulations called Event Summary and Fundamental Operations.

The Event Summary tabulation shows the relationship between the mnemonic symbol registered in the permanent symbol table of the ND812 assembler, the octal code which calls the various hardware routines (a condensed version of data listed in format tables), the hardware routines themselves, and a shorthand notation of the process which takes place during the operation.

The Fundamental Operation tabulation shows the relationship between the octal code, the hardware routine called by the code, and a simplified logic description of the operation in terms of the period in which it occurs.

These tables should be used in trouble analysis. The Event Summary identifies the data transfer paths, their time relationship in the machine cycle, and the individual hardware operations. These tables are useful in identifying the various multiplexer, registers, and accumulators used in the operation. On the other hand, the Fundamental Operations tabulations are useful in identifying the control logic which is used to obtain the hardware process.

4.4.1 COMMON BASIC PHASE

For the basic phase, eight discrete time periods occur. Each period has a duration of 0.25 microseconds so that each phase requires 2 microseconds. Not all periods are required for each instruction, and more than one event can occur during the same period. In some cases two or more adjacent time periods are ORed to form extended time periods.

4.4.1.1 BLOCK DIAGRAM DESCRIPTION. During the basic phase various latches are cleared (refer to Table 4-8), the content of the program counter is transferred to the address register to obtain the location of the instruction to be processed, and the instruction is loaded from memory into the instruction register. In Figure 4-15, the timing diagram, when pulse PU0 occurs, the various cleared latches provide inputs to the processor control logic that determine the event that will take place during each period of the basic phase. During the first period (PU0), the content of the program counter (2, Figure 4-16) is transferred over MX multiplexer data lines (MS00 through MS11) to the adder (10). Because at this time, TS multiplexer outputs have not been called for, signals TS00 through TS11 provide all zeros; thus the result of the adder summing operation is the program counter data which is transferred over bus B00 through B11 to the address register (11). Address register data is always available to memory control over data lines A00 through A11. The address data determines the location of the memory register, from which the operand must be fetched.

When pulse PU1B occurs, the memory buffer register (1) is cleared of the last word so that when the memory register is strobed for a read operation, meaningful data is available. If the memory buffer register (1) is not cleared, the current word would be ORed with the last fetched word.

During PU2, the data contained in the memory register, specified by the content of the address register, is read into the memory buffer register (1) over data lines MS00 through MS11. Data is permitted to enter the memory buffer register at any time. Pulse PU2 is used to read the addressed memory register into the memory buffer. Thus, this event occurs during every phase. Memory buffer data output is transferred over buses MB00 through MB11 to the bus and memory buffer multiplexer (3) which also receives inputs from the adder (10) over buses B00 through B11.

On the trailing edge of pulse PU3, memory buffer data (as opposed to adder data) is switched through the bus and memory buffer multiplexer (3) to the memory data register (6), and the instruction register (7). Pulse PU3 is used to read memory buffer register data into the memory data register (6), so that when the memory register is strobed for a write operation, the same data read from memory, will be written into memory at the location specified by the content of the address register. When the instruction is read into the instruction register (7) during period BP3, its data becomes immediately available to the data control logic and to the instruction decoders (9).

4.4.1.2 FUNDAMENTAL OPERATIONS. Most of the events pertaining to the individual operations of the instructions are carried out during PU4, PU5, and PU6. During period

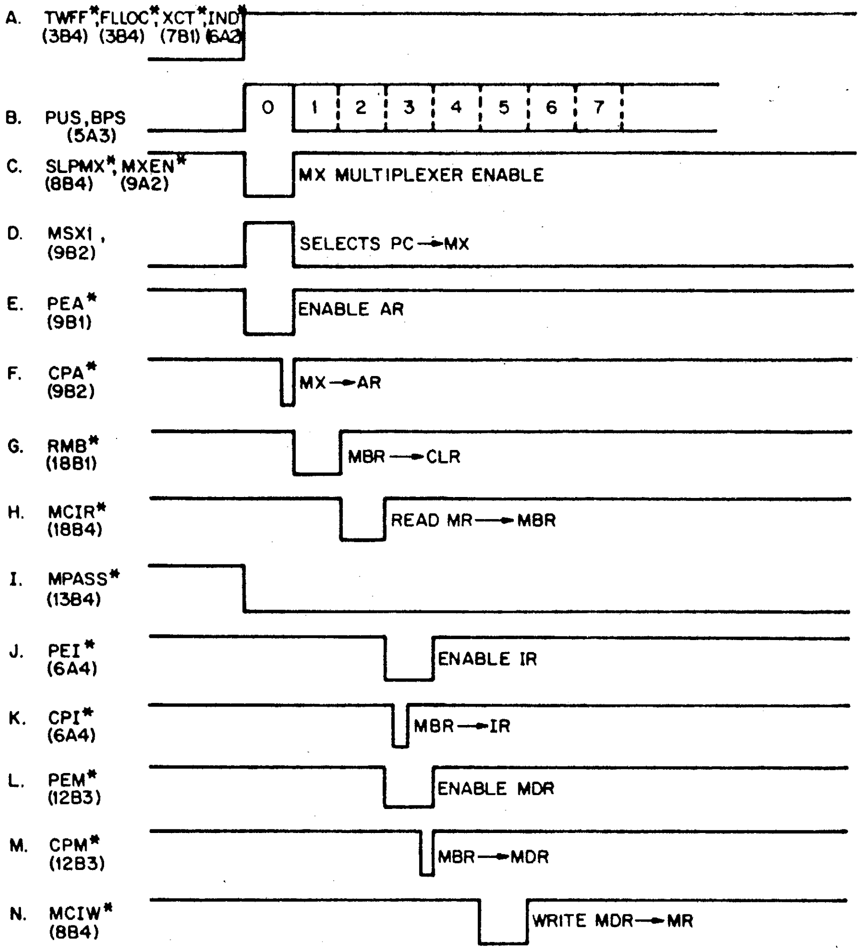


Figure 4-15. Common Basic Phase, Timing Diagram

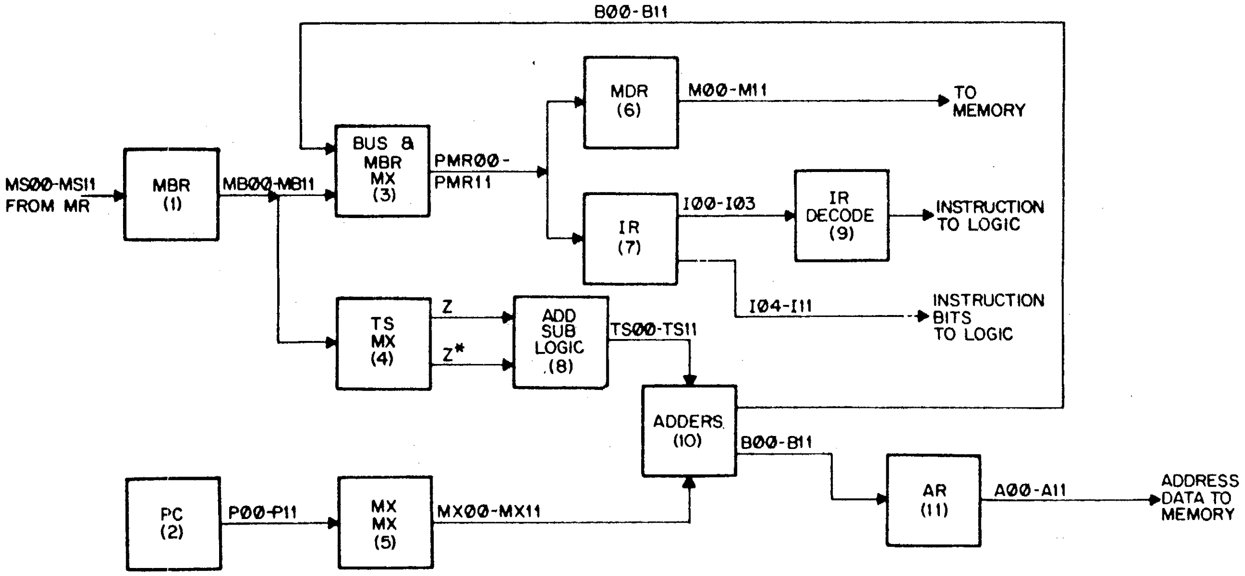


Figure 4-16. Common Basic Phase, Simplified Block Diagram

PU7 the major state control logic determines whether the next phase should be another basic phase or an execute phase.

These events conclude the operations that occur during the common basic phase. The above outlined operations are summarized in Table 4-19 and further detailed in terms of fundamental operations in Table 4-20.

Fundamental operation B is not carried out if there is an indication that this is not a common basic phase. To understand the significance of this provision, it must be remembered that the present instruction is decoded only during period PU3. Therefore, during period BP0, the presence of an execute instruction (XCT*), a two-word memory reference instruction (TWFF*), auto index instruction (FLLOC*), or an indirect fetch (IND*), tells the processing logic that the present phase is not a common basic phase, but a secondary basic phase. The result is that the content of the program counter is not selected as input to the MX multiplexer if any of the above signals are low level.

Fundamental operation D is also a conditional operation; it is not carried out if signal DONE is low level, meaning that processing is past the initial basic phase, and in a secondary basic phase. The secondary basic phase and execute phase are discussed in conjunction with memory reference instructions.

Fundamental operation E clears the done latch early in the basic phase. Later, at PU6, all instructions that require only one basic phase for processing (for example, all operate instructions) set the done latch, resulting in the next machine cycle being set as a basic phase when this condition is detected during fundamental operation K.

Fundamental operation I occurs if a two-word or indirect memory reference instruction is not in effect.

Fundamental operation K is concerned with setting the next phase. Another basic phase follows if the done latch is set, or the instruction decoder indicates the presence of a two-word, an execute, or an instruction using indirect addressing. An execute phase follows only if all above conditions are absent; this is the case, for example, when a memory reference instruction using direct addressing is being processed.

4.4.1.3 TIMING DIAGRAM DESCRIPTION. Figure 4-15 is the timing diagram for the signals generated during the basic phase when the memory reference instruction specifies direct addressing. At the outset, when pulse PU7 occurs during the execution of the previous instruction, the two-word first-last and indirect latches are cleared. Hence, signals TWFF*, FLLOC*, XCT*, and IND* are high-level. When pulse BP0 occurs, high-level output BP0 is ANDed with the above high level signals to develop the low-level SLPMX* signal which enables the MX multiplexer (9A2), and selects the program counter input (signals P00 through P11) for output from the MX multiplexer by producing high-level MXS1 signal (waveform D). This provides an MX selection code of 010 (Table 4-18). Pulse PU0B occurs simultaneously with pulse BP0, so that outputs from the MX multiplexer (signals MX00 through MX11) are applied through the adders (as signals B00 through B11) for entry into the address register. The address register, enabled by low latches, develops a low-level

Table 4-19. Common Basic Phase, Event Summary

Ref.	Period	Event
A.	BP6 or EP6	CLR→LATCHES
B.	BP0	PC→MX→ADDER
C.	PU0B	ADDER→AR
D.	PU0B	CLEAR→IR
E.	BP1	CLR→DONE
F.	PU1	CLR→MBR
G.	PU2	MR→MBR
H.	PU3	MBR→MDR
I.	BP3	MBR→IR
J.	PU5	MDR→MR
K.	BP6	SET→DONE (conditional)
L.	PU7	SET→PHASE

Table 4-20. Common Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU7	CLR→LATCHES 1. ↓EP6*/5A4 = SET→DONE (↑DONE/7B4, ↓DONE*) 2. (↑DONE/7B4) (↑PU7B/5B2)→↓STDN7*/3B3	The two-word, indirect and first-last latches are cleared to initialize basic cycle (Table 4-6).
B.	BP0	PC→MX→ADDER 1. (↑XCT*) (↑IND*)→↓X *I/8B2 2. (↓X *I) (↑BP0) (↑TWFF*) (↑FLLOC*)→↓PRAR*→↑SELP→↓SLPMX*/8B4 3. ↓SLPMX*→↑MXEN/9A2→↓MXEN*/9B2 = ENABLE→MX 4. ↓SLPMX*→↑MXS1 = PC→MX	The content of the program counter is input to the MX multiplexer.
C.	PU0B	ADDER→AR 1. ↑PU0B→↓PEA*→↑PEA/9B1 = ENABLE→AR 2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = ADDERS→AR	The content of the adder replaces the content of the address register.
D.	PU0B	CLR→IR 1. (↑PU0B) (↑DONE)→↓MRI*/6B4 = CLR→IR	The instruction register is cleared.
E.	BP1	CLR→DONE 1. (↓BP1/5A3) = CLR→DONE (↓DONE/7B4, ↑DONE*)	Done flipflop is cleared.
F.	PU1	CLR→MRB 1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	Memory buffer register is cleared.

Table 4-20. Common Basic Phase, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
G.	PU2	MR→MBR 1. (↑ADDF0*) (↑PU2)→↓MCIR*/8B4 = MR→MBR	Content of core memory register replaces content of memory buffer register.
H.	PU3	PU3, MBR→MDR 1. ↓PU3*→↑PEM/12B2 2. (↑PEM) (↑REGCLK)→↓CPM*/12A3 = MBR→MDR	
I.	BP3	MBR→IR 1. (↑TWFF*) (↑INDFF*)→↓MPASS*/13B4 2. ↓MPASS*→↑MPASS/13B4 3. ↓MPASS*→↑SPEI*/6A4 4. (↑SPEI*) (↑BP3)→↓PEI*/6A4 = ENABLE→IR 5. ↓PEI*→↑PEI/6A4 6. (↑PEI) (↑CCLK)→↓CPI*/6A4/17B1 = MBR→IR	Content of memory buffer register replaces content of instruction register.
J.	PU5	MDR→MR 1. (↑WTRL*) (↑PU5B)→↓PU5W*→↑PU5W/8B3 2. (↑PU5W) (↑ADDF0*)→↓MCIW*/8B4 = MDR→MR	Content of memory data register replaces content of memory register.
K.	BP6	SET→DONE (B) 1. ↓TWHLT* + ↓I/O* + ↓ADL* + ↓SBL* + ↓ANL* + ↓OP1* + ↓OP2*→ ↑SDONE/7B3 2. (↑SDONE) (↑BP6)→↓SDONE*/7B3 = SET→DONE (↑DONE/7B4, ↓DONE*)	Done flipflop is set.
L.	PU7	SET→PHASE 1. (↑DONE*) (↑TW*) (↑IND*) (↑XCT*)→ ↓RBPH→↑RBPH*/5B1 2. (↑RBPH*) (↓CPPU*) (↑PU7) = ↑EPH/5B2 3. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→ ↑RBPH→↓RBPH*/5B1 4. (↓RBPH*) (↓CPPU*) (↑PU7) = ↑BPH/5B2	Next phase is set to either a basic or an execute phase.

MPASS* signal (waveform I) which permits pulses PEI* (waveform J) and CPI* (waveform K) to be generated. Pulse PEI* enables the instruction register, and pulse CPI* loads it with data on lines PMR00 through PMR11.

When pulse PU5 occurs, the word previously read into the memory buffer register during period PU2 is written back into the same memory location when low-level pulse MCIW* is produced. No related events are produced when pulse PU6 occurs. However when pulse PU7 occurs, the execute phase is set, if the done latch is not set, and a two-word, indirect, or execute instruction is not being processed. The major state control goes from its basic phase output to its execute phase output when levels DONE*, TW*, IND*, and XCT* are high and ANDed to produce the high-level RBPH* pulse, which primes the major-state control register. Hence, when pulse PU7 occurs, the major-state control register is toggled to its execute state. Thus, during the common basic phase, the events can be summarized as follows.

a. The content of the program counter is loaded into the address register.

b. The content of the memory register, located at the address specified by the content of the address register, is read into the memory buffer register. (This is the current instruction.)

c. The content of the memory buffer register is loaded into both the memory location specified by the content of the address register, and into the instruction register.

d. During the last period, the next phase is set.

4.4.2 OPERATE INSTRUCTIONS

Operate instructions do, in general, manipulate data which is availabie either in one of the accumulators or in the status register; they have a single-word format and require only one basic phase. Operate instructions can be further classified into group 1 and group 2 instructions. Group 1 instructions include multiply and divide, various combining operations such as ANDing, arithmetic processes such as addition and subtraction, and shigt and rotate processes which shift data to the left one or more bits in the main accumulators. Generally, group 2 instructions manipulate data without employing an arithmetic process. These operations include skips, negations, complements and register clear and set operations.

The common basic phase, which is employed for all instructions, has been described. It has been pointed out that differences are noted as various types of processing for each instruction is discussed. Table 4-21 is a summary of fundamental events required to process the operate instructions. By comparing references for the operate instructions with those given for the common basic phase, it can be seen that up to and including fundamental event I, all operations are identical.

Table 4-21. Operate Instructions, Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase
A.	BP6 or EP6	CLR→LATCHES	A.
B.	BP0	PC→MX	B.
C.	PU0B	ADDER→AR	C.
D.	PU0B	CLR→IR	D.
E.	BP1	CLR→DONE	E.
F.	PU1	CLR→MBR	F.
G.	PU2	MR→MBR	G.
H.	PU3	MBR→MDR	H.
I.	BP3	MBR→IR	I.
J.	BP3	PC→MX	-
K.	BP4	ADDER→+1	-
L.	BP4	ADDER→PC	-
M.	PU5	MDR→MR	J.
N.	PU7	SET→PHASE	K.

Fundamental events K and L are not to be found during the common basic phase. They provide for incrementing the program counter in order to ready it for the next instruction (refer to Table 4-22). As mentioned earlier, for all other types of processing instructions, more than one basic phase is required during a machine cycle. For those instructions program counter incrementation is deferred to the next, or even later subperiods of the machine cycle.

4.4.2.1 GROUP 1 OPERATE INSTRUCTIONS. Group 1 operate instructions are characterized by bit pattern 0010 in the operate code field of the instruction, bit positions 0 through 3. These instructions operate on data contained in the upper and lower accumulators. The following descriptions illustrate the manner in which instructions in this group of operates are processed using block diagrams, timing diagrams, and various tabulations, including fundamental operations describing the events. For general orientation, paragraph 4.3.4.1 should be reviewed. This paragraph describes functions and operations of the primary instruction decoder and operate instruction decoder.

The operate instruction descriptions generally include a block diagram which illustrates the data transfer paths invoked by the called instruction and a timing diagram which shows various time-related functions of the hardware control logic together with two important tabulations called event summary and fundamental operation.

The event summary tabulation shows the relationship between the mnemonic symbol registered in the pemanent symbol table of the ND812 assembler, the octal code which calls the various hardware routines (a condensed version of data listed in Table 4-25), the hardware routines, themselves, and a shorthand notation of the process which takes place during the operation.

The fundamental operation tabulation shows the relationship between the octal code, the hardware routine called by the code, and a simplified logic description of the

Table 4-22. Operate Instructions, Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
B-I	BP0-BP3		Identical with common basic phase (Table 4-20) adder incremented by one.
J.	BP4	PC→MX→ADDER 1. (↑TWIO*) (↑BP4) (↑WTRL*) (↑INCPR*)→ ↓SLPMX*/8B4 2. ↓SLPMX*→↓MXEN*/9A1 = ENABLE→MX 3. ↓SLPMX*→↑MSX1/9B2 = PC→MX	The content of the program counter is switched through the MX multiplexer to the adder.
K.	BP4	+1→ADDER 1. (↑MREFI*) (↑A12*)→↓INCPR→ ↑INCPR*/9B3 2. (↑BP4) (↑INCPR*) (↑WTFL*) (↑HLTTW*)→ ↓CCIN2/12A2 3. ↓CCIN2→↑CIN/12A2/16A3 = +1→ADDER	The adder is incremented.
L.	BP4	ADDER→PC 1. ↓TWFF→↑TWIO*/9B3 2. (↑TWIO*) (↑BP4) (↑WTRL*) (↑INCPR*)→ ↓PEP*/9B3 3. ↓PEP*→↑PEP/9B1 4. (↑PEP) (↑REGCLK)→↓CPP*/9B2	Content of the adder is admitted to the program counter.
M.	BP5-BP7		Identical with common basic phase (Table 4-20).

operation in terms of the period in which it occurs.

These tables should be used in trouble analysis. The event summary identifies the data transfer paths, their time relationship in the machine cycle, and the individual hardware subroutines. These tables are useful in identifying the various multiplexers, registers, and accumulators used in the subroutine. On the other hand, the fundamental operation tabulations are useful in identifying the control logic which is used to obtain the hardware subroutine.

There are 53 Group 1 operate instructions in the repertoire of the ND812 computer. The instructions are divided into five subgroups (see Table 4-23) and are listed with their mnemonic and octal codes and with a shorthand description of their operation. Subgroups are formed by taking the common processing characteristics of instruction bit patterns into consideration. This approach follows the same organization that the operate instruction

Table 4-23. Group 1 Operate Instructions by Subgroup

Subgroup	Mnemonic	Octal Code	Operation
MPY-DIV	MPY ·	1000	JxK to R,S
	DIV	1001	J, K/R to J, REMAINDER IN K
LOGICAL AND	AND J	1100	LOGICAL AND J, K INTO J
	AND K	1200	LOGICAL AND J, K INTO K
	AND JK	1300	LOGICAL AND J, K INTO J, K
LOAD/EXCHANGE	LRFJ	1101	LOAD R FROM J
	LSFK	1201	LOAD S FROM K
	LFJR	1102	LOAD J FROM R
	LKFS	1202	LOAD K FROM S
	LRSFJK	1301	LOAD R, S FROM J,K
	LJKFRS	1302	LOAD J, K, FROM R, S
	LKFJ	1204	LOAD K FROM J
	EXJR	1103	EXCHANGE J AND R
	EXKS	1203	EXCHANGE K AND S
	EXJK	1374	EXCHANGE J, K
	EXJRKS	1303	EXCHANGE J, K AND R, S
ADD-SUBTRACT	AJK J	1120	J + K to J
	SJK J	1121	J-K to J
	ADR J	1122	R+J to J
	SBR J	1123	R-J to J
	ADS J	1124	S+J to J
	SBS J	1125	S-J to J
	NAJK J	1130	-(J+K) to J
	NSJK J	1131	K-J to J
	NADR J	1132	-(R+J) to J
	NSBR J	1133	J-R to J
	NADS J	1134	-(S+J) to J
	NSBS J	1135	J-S to J
	AJK K	1220	J+K to K
	SJK K	1221	J-K to K
	ADR K	1222	R+K to K
	SBR K	1223	R-K to K
	ADS K	1224	S+K to K
	SBS K	1225	S-K to K
	NAJK K	1230	-(J+K) to K
	NSJ K	1231	K-J to K
	NADR K	1232	-(R+K) to K
	NSBR K	1233	K-R to K
	NADS K	1234	-(S+K) to K
	NSBS K	1235	K-S to K
	AJK JK	1320	J+K to J,K
	SJK JK	1321	J-K to J,K
	NAJK JK	1330	-(J+K) to J,K
	NSJK JK	1331	K-J to J,K

Table 4-23. Group 1 Operate Instructions by Subgroup (Cont'd.)

		Octal	
Subgroup	Mnemonic	Code	Operation
SHIFT/ROTATE	SFTZ J	1140	SHIFT J LEFT N
	SFTZ K	1240	SHIFT K LEFT N
	SFTZ JK	1340	SHIFT J TO K LEFT N
	ROTD J	1160	ROTATE J LEFT N
	ROTD K	1260	ROTATE K LEFT N
	ROTD JK	1360	ROTATE J, K LEFT N
LOAD/READ	LJSW	1010	LOAD J FROM SWITCH REGISTER
	LJST	1011	LOAD J FROM STATUS REGISTER
	RFOV	1002	READ FLAG, OV FROM J

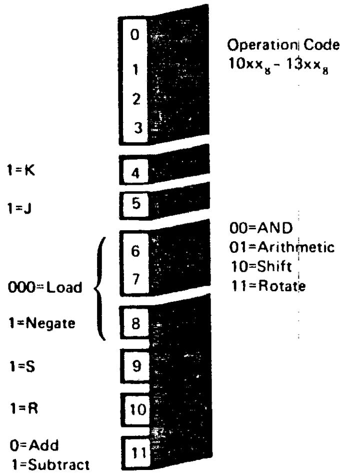
decode logic follows; it has the advantage of offering an easy survey of the simplified logic tables when direct reference to them is helpful in following the instruction execution flow. The subgroups are discussed one by one, with the exception of the multiply and divide instructions. The theory and details of operation for each of these is described separately. Multiply and divide instructions are each autonomous and belong to no group or subgroup of operation; hence, they are discussed as separate processes.

The format for the divide and multiply instructions, for the logical AND subgroup, and the add-subtract subgroup, is shown in Table 4-24. In addition to the format, Table 4-24 shows significant bits of the instructions in each subgroup aligned with the bit pattern for any of the instructions in the subgroup. Note that the various bit positions have different functional assignments; some specify accumulators, others detect the arithmetic operations, and others determine shift patterns. The operation code, however, invarient for all these operations, is 0010$_2$. Bits 4 and 5 of the instruction are used to call one or both of the main accumulators for the operation being processed. Bits 9 and 10 call one or two of the subaccumulators. Bits 6 and 7 control the arithmetic operation subgroup. Bit 8 controls negation, but is in reality, a subtract operation which either follows a previous addition or a previous subtraction. The previous subtraction, is controlled by bit 11. This operation is required because not all pairs of operands (contents of accumulators) can be ordered in the required sense for a subtraction because of transfer route limitations in the hardware.

4.4.2.1.1 Hardware Multiply (1000). Hardware multiply is carried out by the process of accumulation of a number of partial products accordingly as digits of the multiplier appear as a binary 1. The process is not unlike that carried out when multiplication is done with pencil and paper; for example, arbitrarily taking the decimal numbers 105 and 115 to be multiplied, there are several facts immediately apparent.

Example A	115	Multiplicand
	105	Multiplier
	575	1st partial product
	1150	2nd partial product
	12075	Resultant product

Table 4-24. Group 1 Operate Instruction, Bit Pattern Formats by Subgroup



This is the group 1 operate instruction format. Bits 4 and 5 select the K and J registers, respectively. Bits 6 and 7 determine the operation except for load. Bit 8 establishes either a load or arithmetic operation. In a logical AND, bits 8, 9, 10, and 11 have no significance; in an arithmetic operation bit 8 selects negate, bits 9 and 10; the S and R register, respectively, and bit 11 establishes an add or subtract. In a shift or rotate operation bits 8, 9, 10, and 11 establish the number of shifts or rotates.

Octal			Bit Pattern											
Code	Mnemonic	Subgroup	0	1	2	3	4	5	6	7	8	9	10	11
1000	MPY	MULTIPLY			1									
1001	DIV	DIVIDE			1									1
1100	AND J	LOGICAL			1			1						
1200	AND K	AND			1		1							
1300	AND JK				1		1	1						
1120	AJK J	ADD			1			1		1				
1121	SJK J	SUBTRACT			1			1		1				1
1122	ADR J				1			1		1			1	
1123	SBR J				1			1		1			1	1
1124	ADS J				1			1		1		1		
1125	SBS J				1			1		1		1		1
1130	NAJK J				1			1		1	1			
1131	NSJK J				1			1		1	1			1
1132	NADR J	ADD-SUBTRACT			1			1		1	1		1	
1133	NSBR J	(cont)			1			1		1	1		1	1
1134	NADS J				1			1		1	1	1		
1135	NSBS J				1			1		1	1	1		1
1220	AJK K				1		1			1				
1221	SJK K				1		1			1				1
1222	ADR K				1		1			1			1	
1223	SBR K				1		1			1			1	1
1224	ADS K				1		1			1		1		
1225	SBS K				1		1			1		1		1
1230	NAJK K				1		1			1	1			
1231	NSJK K				1		1			1	1			1
1232	NADR K				1		1			1	1		1	
1233	NSBR K				1		1			1	1		1	1
1234	NADS K				1		1			1	1	1		

Table 4-24. Group 1 Operate Instructions, Bit Pattern Format by Subgroup (Cont'd.)

Octal			Bit Pattern											
Code	Mnemonic	Subgroup	0	1	2	3	4	5	6	7	8	9	10	11
1235	NSBS K				1		1			1	1	1		1
1320	AJK JK				1		1	1		1				
1321	SJK JK				1		1	1		1				1
1330	NAJK JK				1		1	1		1	1			
1331	NSJK JK				1		1	1		1	1			1
1101	LRF J	LOAD- EXCHANGE			1			1						1
1102	LJFR				1			1					1	
1103	EXJR				1			1					1	1
1201	LSFK				1		1							1
1202	LKFS				1		1						1	
1203	EXKS				1		1						1	1
1204	LKFJ				1		1					1		
1301	LRSFJK				1		1	1						1
1302	LJKFRS				1		1	1					1	
1303	EXJRKS				1		1						1	1
1140	SFTZ J	SHIFT-ROTATE			1			1	1					
1160	ROTD J				1			1	1	1				
1240	SFTZ K				1		1		1					
1260	ROTD K				1		1		1	1				
1340	SFTZ JK				1		1	1	1					
1360	ROTD JK				1		1	1	1	1				
1374	EXJK				1		1	1	1	1	1	1		
1002	RFOV	LOAD-READ			1								1	
1010	LJSW				1						1			
1011	LJST				1						1			1

a. Partial products are shifted toward the most significant digit accordingly as a digit occurs in the multiplier.

b. It is the product successive summation of the multiplicand that produces the partial product; i.e., 115 x 5 is equal to 115 + 115 + 115 + 115 + 115 = 575.

c. A partial product exists for every digit of the multiplier.

While it is simple enough to implement this process, inputing of multiplier and multiplicand requires special manipulation of input data by the programmer or unnecessary logic would have to be added making the ND812 more costly.

Converting the decimal numbers 105 and 115 to their binary equivalents and carrying out the same multiplication process, we have;

Example B $115_{10} = 1110011_2$ and $105_{10} = 1101001_2$

$$\begin{array}{rrl}
115_{10} = & 1110011_2 & \text{Multiplicand} \\
105_{10} = & 1101001_2 & \text{Multiplier} \\
\hline
 & 1110011 & \text{1st partial product} \\
 & 111001100 & \text{1st, 2nd and 3rd shift} \\
\hline
 & 10000001011 & \text{2nd partial product} \\
 & 11100110 & \text{4th and 5th shift} \\
\hline
 & 1001001101011 & \text{3rd partial product} \\
 & 1110011 & \text{6th shift} \\
\hline
12075_{10} = & 10111100101011_2 & \text{4th partial product}
\end{array}$$

What is actually done through this process is that the order of significance of each multiplier digit is determined and the multiplicand is shifted to that order of significance before summing. This is the process used in the ND812 hardware multiply logic. Thus the multiplier is loaded into the K accumulator and the multiplicand is added to successively shifted partial products, accordingly, as the most significant digit of the multiplier is a binary 1.

In the foregoing example, the multiplier is loaded into the K accumulator and the multiplicand is loaded into the R accumulator (through the J accumulator). The multiplier is then successively shifted left and the most significant bit (K00) continually tested. As the leftward shift of the multiplier bits occur they are truncated (lopped off) because after testing they are no longer needed.

The hardware multiply flow diagram is shown in Figure 4-17. At the outset of a multiply operation the multiplier is loaded into the K accumulator and the multiplicand is loaded into the J accumulator through programmed load K and load J instructions. Next, the multiply instruction is ordered by the program.

The first event that occurs is that the content of the J accumulator (multiplicand) is transferred to the R register (A Figure 4-17) where it is retained throughout the multiply operation.

Next, the most significant bit of the K register is tested (C). If this bit is a 1, the K0M register is set. This memory element is required because when the first multiplier shift occurs, this information will be lost, but the data must be retained because it has important significance in determining the product. The K0M register holds this data until needed.

Because partial products are accumulated in the J accumulator, the J accumulator must first be cleared (B). However, the content of the J accumulator, the multiplicand, has been saved in the R accumulator.

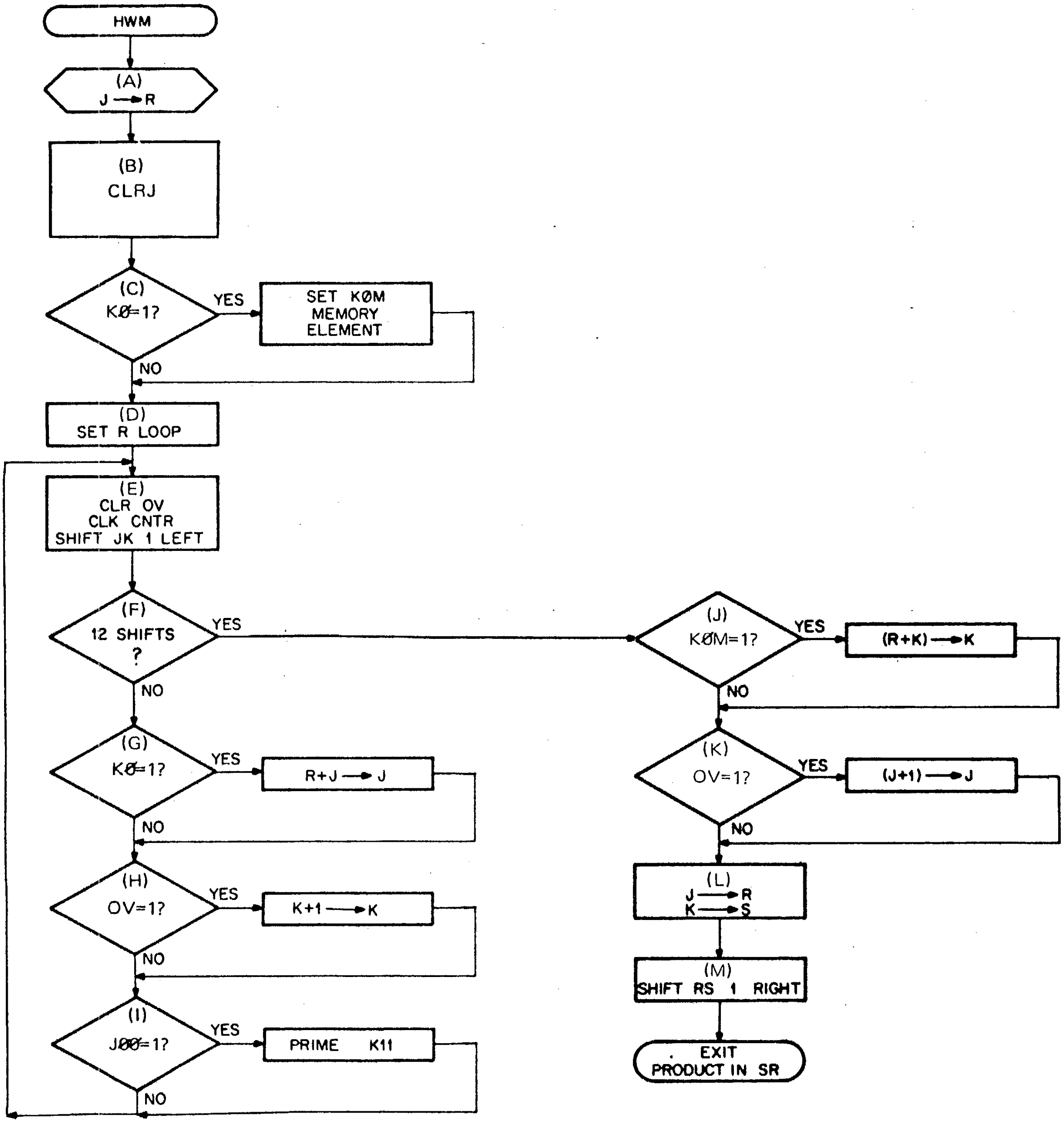


Figure 4-17. Multiply Instruction, Flow Diagram

NOTE

> Because they are able to store data, the J and K and R and S storage elements are certainly registers. However, because they acquire arithmetic results, they are more importantly accumulators.

Next the reiteration loop register (RLOOP) is set (D). The data stored in this register is used by the logic to obtain the required 12 shifts and other loop operations. The 12 shifts are obtained by recycling the pulser through periods BP4, BP5 and BP6 12 times. After RLOOP register is set, the pulser reenters period BP5 when period BP6 occurs, permitting the first reiteration cycle to occur.

At the beginning of each reiteration cycle the overflow register is cleared, the up counter is pulsed to record the number of cycles, and the content of both the J and K accumulators is shifted toward the most significant bits (J00 and K00), or leftward (E). As shifting progresses, the K00 bit is continually tested. In Figure 4-18 no data appears in the cleared J accumulator until the first 1 bit is detected (multiplier bit 5 appears in K00 of the K accumulator when shift 5 occurs).

At this time the content of the R accumulator (the saved multiplicand) is summed with the content of the J accumulator (Figure 4-17, F), which is still empty. Thus, at the conclusion of the 5th shift the J accumulator contains the first partial product (PP1) as the result of the first summation. In Figure 4-18, when the 6th shift occurs the 1st partial product is raised to the next order of significance and the multiplicand is again summed with this new partial product to produce the new second partial product (PP2). Subsequently, when the seventh shift occurs, a summation does not take place because a 1 bit has not been detected in the K00th position. Thus, a partial product shift occurs, but a summation does not. This process continues until 12 shifts (Figure 4-17, I) occur.

Because a 12-bit binary multiplier and multiplicand will result in a 24-bit product, the hardware multiply logic must provide for overflow of the J accumulator, which holds only 12 bits. Overflow can occur in two ways: (1) when the content of both the R and J accumulators are summed and a carry is produced, the overflow is entered into the K11th bit. (2) When the J00th bit becomes a 1, the K11th bit must receive a 1 (set) when the next shift occurs. The occurance of these conditions are continuously tested (Figure 4-17, G and H) by the multiply logic. If either event occurs, the K11th bit receives a 1. As can be seen in Figure 4-18, the K11th bit receives a 1 after the 10th and 12th shifts occur, resulting in an accumulated partial product in the J and K registers: K = 101, J = 111 001 010 110. Comparing this result with example B indicates that the content of the J register is too high by one order of binary significance and the content of the K accumulator is too high by one order of binary significance +1 in the K11th bit.

At the conclusion of the 12th reiteration (I) the content of the K0M memory and overflow registers are tested (Figure 4-17, J and K) and the content of the J and K accumulators modified accordingly. After testing the K0M and overflow registers, the content of

KR	0	1	2	3	4	5	6	7	8	9	10	11		JR	0	1	2	3	4	5	6	7	8	9	10	11
SHF1	0	0	0	0	1	1	0	1	0	0	1	0	←		0	0	0	0	0	0	0	0	0	0	0	0
SHF2	0	0	0	1	1	0	1	0	0	1	0	0	←		0	0	0	0	0	0	0	0	0	0	0	0
SHF3	0	0	1	1	0	1	0	0	1	0	0	0	←		0	0	0	0	0	0	0	0	0	0	0	0
SHF4	0	1	1	0	1	0	0	1	0	0	0	0	←		0	0	0	0	0	0	0	0	0	0	0	0
SHF5	1	1	0	1	0	0	1	0	0	0	0	0	←		0	0	0	0	0	0	0	0	0	0	0	0
	└→ + 1110011→JR													RR						1	1	1	0	0	1	1
														PP1 →	0	0	0	0	0	1	1	1	0	0	1	1
SHF6	1	0	1	0	0	1	0	0	0	0	0	0	←		0	0	0	0	1	1	1	0	0	1	1	0
	└→ + 1110011→JR													RR						1	1	1	0	0	1	1
														PP2 →				1	0	1	0	1	1	0	0	1
SHF7	0	1	0	0	1	0	0	0	0	0	0	0	←		0	0	1	0	1	0	1	1	0	0	1	0
SHF8	1	0	0	1	0	0	0	0	0	0	0	0	←		0	1	0	1	0	1	1	0	0	1	0	0
	└→ + 1110011→JR													RR						1	1	1	0	0	1	1
														PP3 →	0	1	0	1	1	1	0	1	0	1	1	1
SHF9	0	0	1	0	0	0	0	0	0	0	0	0	←		1	0	1	1	1	0	1	0	1	1	1	0
SHF10	0	1	0	0	0	0	0	0	0	0	0	1	←		0	1	1	1	0	1	0	1	1	1	0	0
SHF11	1	0	0	0	0	0	0	0	0	0	1	0			1	1	1	0	1	0	1	1	1	0	0	0
	└→ + 1110011→JR													RR						1	1	1	0	0	1	1
KR =	1	0	0	0	0	0	0	0	0					PP4 →	1	1	1	1	0	0	1	0	1	0	1	1
SHF12	0	0	0	0	0	0	0	0	0	1	0	1			1	1	1	0	0	1	0	1	0	1	1	0

Figure 4-18. Typical Hardware Multiply Configuration

the J accumulator is transferred to the R accumulator and the content of the K accumulator is transferred to the S accumulator.

To understand how the final product is obtained, the configuration of the accumulator interconnections must be understood. Figure 4-19 is the J to R accumulator interconnection. Data contained in the adders is admitted whenever the J accumulator PE (parallel enable) input becomes low-level and a clock pulse (CP) occurs. Thus any adder data is admitted to the J accumulator when a low-level PEJ* signal and a low level CPJ* signal occur. In the same manner, J accumulator data is admitted to the R accumulator when low-level signals PER* and CPR* occur. However, the relative significance of the internal R accumulator bits Q_0 through Q_3 is opposite that of the corresponding J accumulator bits. Both accumulators are configured so that when a data shift occurs the direction of the shift is toward the high order of significance; i.e., Q_0 is shifted to Q_1 which is shifted to Q_2 which, in turn, is shifted to Q_3.

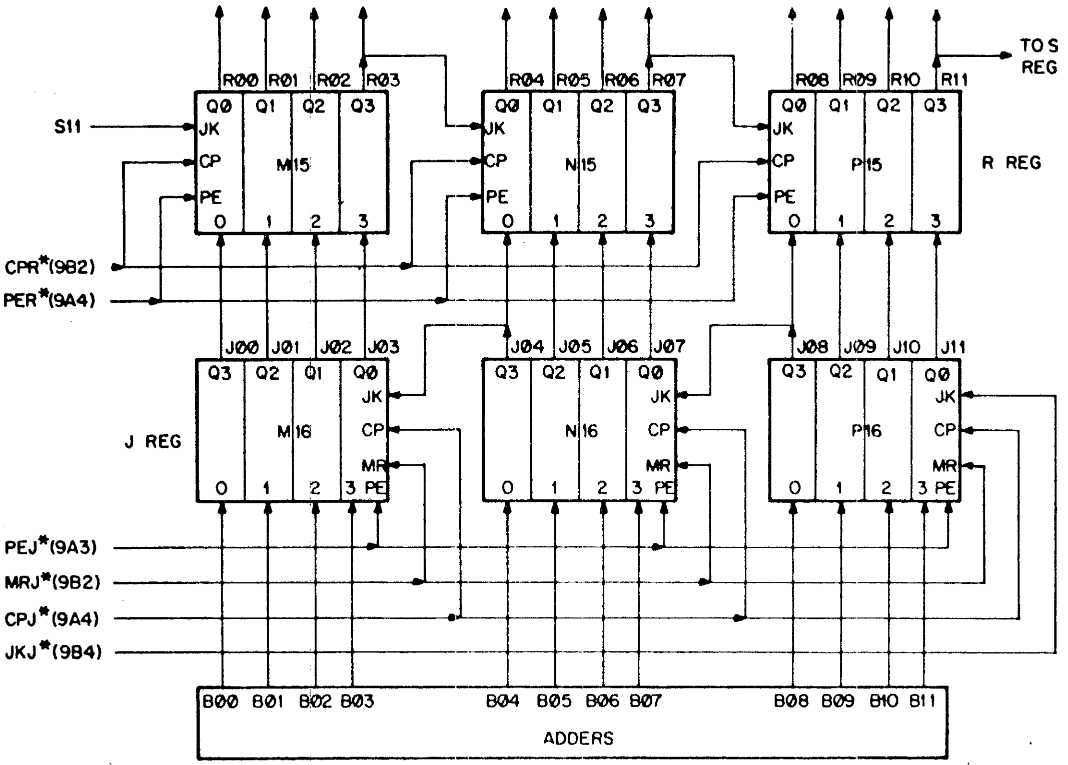


Figure 4-19. J to R Register Configuration, Block Diagram

Each element of each accumulator comprises three Fairchild Semiconductor Type 9300 4-bit shift registers. Integrated circuit modules M16 (14A3), N16 (14A3) and P16 (16A3) are connected as shown in Figure 4-19 to form the J accumulator and modules M15 (14A4), N15 (15A4), and P15 (16A4) are connected as shown to form the R accumulator. A similar arrangement is shown in Figure 4-20 for the K and S accumulators; modules M14 (14A3), N14 (15A3), and P14 (16A3) form the K accumulator and modules M13 (14A4), N13 (15A4) and P13 (16A4) form the S accumulator. Thus, data transferred from either the

J accumulator to the R accumulator or from the K accumulator to the S accumulator is parallel transferred bit-for-bit. However, when data in the J and K accumulators is shifted, shifting is leftward while for the R and S accumulators shifting is rightward.

To understand how this is done, it should be recalled from paragraph 4.3.4 that for the Type 9300 4-state shift register, when the J and K inputs are tied together D type data entry is obtained, and that further, shifting is toward the high-order bits. Because the JKJ and JKK signals are low level, successive clock pulses, CPJ* and CPK* produce zeros in bits 11 through 0 of both accumulators as each subsequent shift is produced by each clock pulse. Comparing the J to R accumulator configuration of Figure 4-19 with the K to S accumulator configuration of Figure 4-20, the JKK input signal applied to module P14 is either low or high as the J00th bit is high or low when either a hardware multiply or divide instruction (signal HWMD) is being processed. Thus, the requirement of event H of the multiply flow diagram (Figure 4-17) is satisfied.

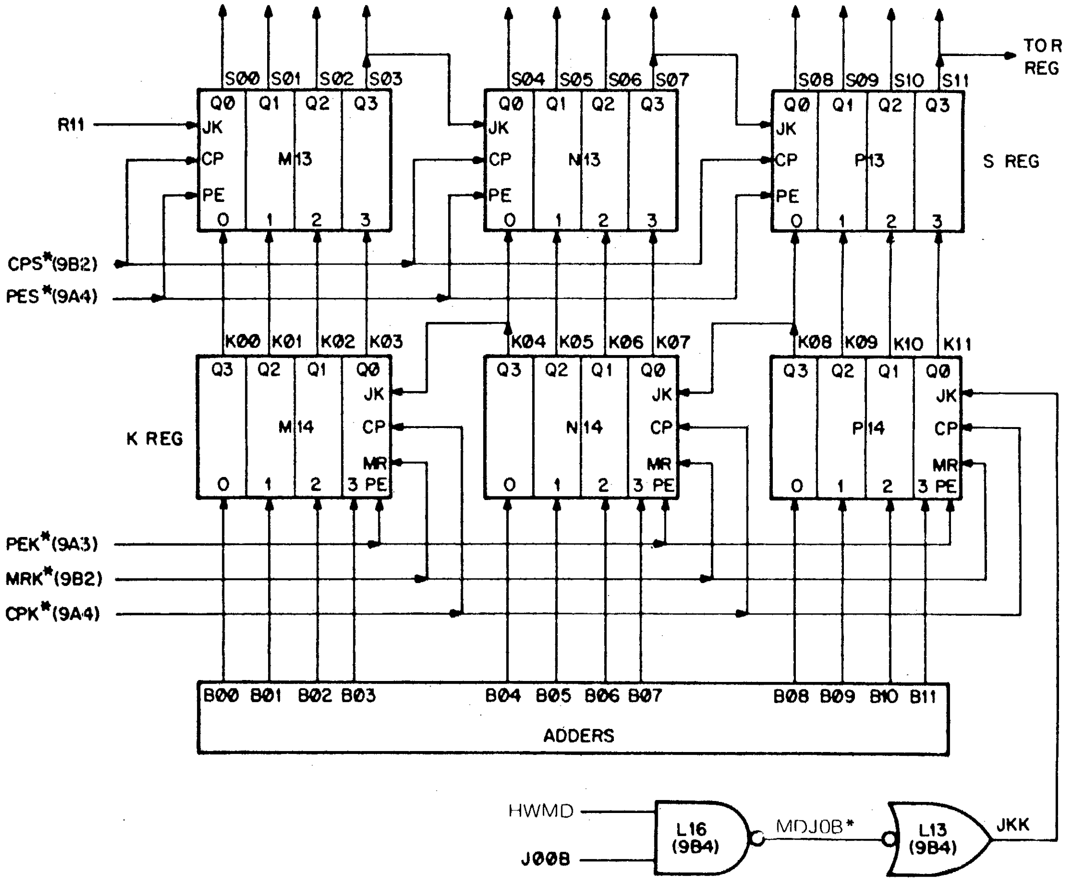
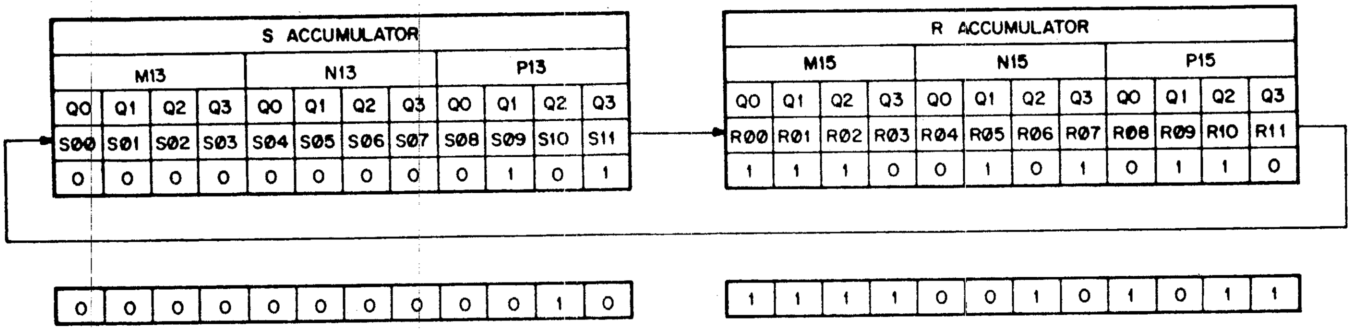


Figure 4-20. K To S Register Configuration, Block Diagram

Although data in both the R and S accumulators is shifted in a manner identical to that of the J and K registers the result is an effective shift to the right. Thus, accordingly as the S11 signal (Figure 4-19) is high level or low level, a high-level or low-level bit is shifted into bit R00 of the R accumulator; and accordingly as bit R11 is high level or low level (Figure 4-20), a high-level or low-level bit is shifted into bit S00 of the S accumulator. Figure 4-21 illustrates the R to S accumulator shift loop. These accumulators hold the partial product produced as the result of the 12th shift for the multiplication depicted in example C. When a shift occurs bits S00 through S10 will be shifted to the right 1 place; bit S11 will be shifted into bit R00, and bit R11 will be shifted into S00. Although the 12th partial product as accumulated in the R and S registers is incorrect, the final product is obtained when the requirement of event M of the flow diagram is carri ed out.



S ACCUMULATOR											
M13				N13				P13			
Q0	Q1	Q2	Q3	Q0	Q1	Q2	Q3	Q0	Q1	Q2	Q3
SØØ	SØ1	SØ2	SØ3	SØ4	SØ5	SØ6	SØ7	SØ8	SØ9	S10	S11
0	0	0	0	0	0	0	0	0	1	0	1

R ACCUMULATOR											
M15				N15				P15			
Q0	Q1	Q2	Q3	Q0	Q1	Q2	Q3	Q0	Q1	Q2	Q3
RØØ	RØ1	RØ2	RØ3	RØ4	RØ5	RØ6	RØ7	RØ8	RØ9	R10	R11
1	1	1	0	0	1	0	1	0	1	1	0

0	0	0	0	0	0	0	0	0	0	1	0

1	1	1	1	0	0	1	0	1	0	1	1

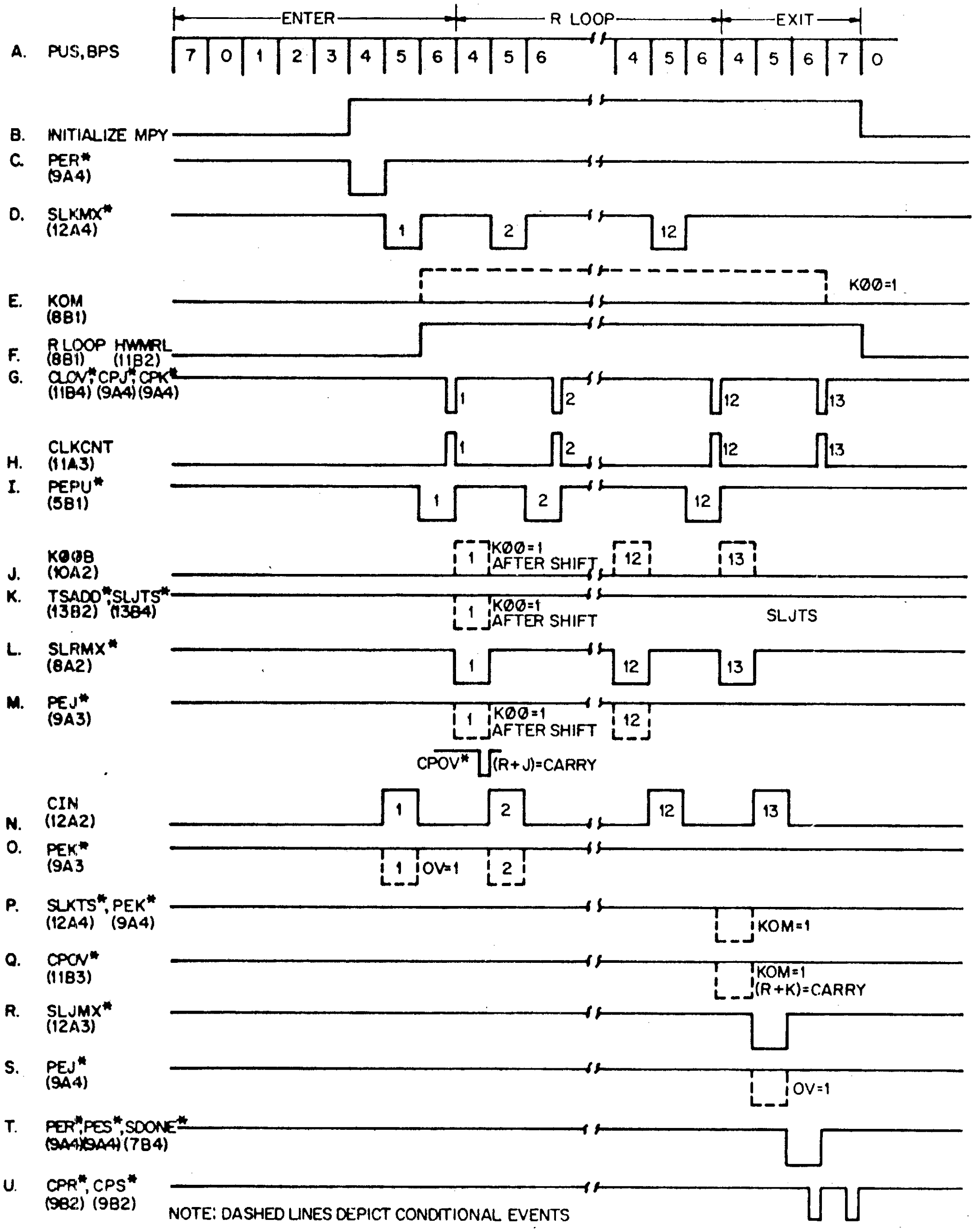
Figure 4-21. R-S Accumulator Shift Loop

Fundamental Operation - This paragraph describes the fundamental operations that are required to implement the hardware multiply logic. For data flow, refer to Figure 4-13, the arithmetic data loop block diagram, to Figure 4-22 for the multip.y operation timing diagram, to Table 4-25 for the event summary, and to Table 4-26 for the simplified logic. The events described in these diagrams and tables are keyed to the flow diagram, Figure 4-17.

The conditions required to initialize the multiply operation occur after the common basic phase events up to period BP3 occur. These are listed as event C in Tables 4-25 and 4-26 and suggested by wavefom B in Figure 4-22, the timing diagram. The events that occur are subdivided into three subperiods as shown in the timing diagram and Table 4-26; the subperiods are called enter, loop, and exit.

1. Enter. The enter peiod includes the common basic phase events, the initialize events, and certain operations such as register transfers, the memory element test and entry into the loop operation. These operations are described by A through H in Table 4-26 and A through F in Figure 4-22.

2. Loop. Although some operations labeled LOOP in Table 4-26 are not loop dependent (require the R LOOP signal for assertion of logic), they are listed as loop operations. For example, in Table 4-26, events H through J are not loop dependent because waveforms G and H show that they occur whenever time period BP5 occurs. All loop operations are dependent on the clock count logic. Hence, these operations require that signal PSI2* remain unasserted (high).



NOTE: DASHED LINES DEPICT CONDITIONAL EVENTS

Figure 4-22. Multiply Instruction, Timing Diagram

Table 4-25. Multiply Instruction, Event Summary

Ref.	Period	Event
A.	PU0	CLR→R LOOP
B.	BP0 - BP5	COMMON BASIC PHASE
C.	BP3	INITIALIZE MULTIPLY
D.	BP4	JR→RR
E.	BP5	KR→MX→ADDER
F.	BP5MD	TEST K00 BIT
G.	PU5	SET→R LOOP
H.	BP6MD	CLR→OV REG
I.	BP6MD	CLOCK PULSE→UP COUNTER
J.	BP6MD	SHIFT→JR, KR, 1-BIT LEFT
K.	BP6MD	RECYCLE PULSER→REPEAT BP4, BP5, BP6
L.	BP4, BP4MD	JR→TS→ADDER, IF K00 = 1
M.	BP4MD	RR→MX→ADDER
N.	BP4MD	ADDER→JR
O.	BP4MD	IF (K00 = 1) (R+J) = CARRY OUT, SET→OV
P.	BP5	KR→MX→ADDER
Q.	BP5MD	+1→ADDER = (K+1)
R.	BP5MD	ADDER→KR, IF OV = 1
S.	BP6MD	IF LOOP = 12, EXIT RECYCLE PULSER
T.	BP4MD	RR→MX→ADDER
U.	BP4MD	IF KOM = 1, JR→TS
V.	BP4MD	IF KOM = 1, ADDER→KR = (R + K)→K
W.	BP4MD	IF (R + K) = CARRY WHEN KOM = 1, SET→OV
X.	BP5	JR→MX→ADDER
Y.	BP5MD	+1→ADDER (J+1)
Z.	BP5MD	IF OV = 1, ADDER→JR
AA.	BP6MD	JR→RR
AB.	BP6MD	KR→SR
AC.	BP6	SET→DONE
AD.	BP7	SHIFT RR AND SR, 1-BIT RIGHT
AE.	PU0	ENTER COMMON BASIC PHASE

During loop operation, certain events occur as the result of previously established conditions which are established as a result of the hardware-controlled events. Hence, signal K00B is true only if, as the result of a leftward shift of J and K register is set. All conditional events are shown by dashed lines superimposed on the waveform base. Thus, event L in Tables 4-25 and 4-26 show that the admission of J-register data into the TS multiplexer occurs only if signal K00B is asserted (waveform K, Figure 4-22). In any case, R-register data is unconditionally admitted into the MX multiplexer (waveform L), but may, or may not be loaded into the J register and may or may not cause the overflow register to be set (waveform M). These events are listed in L through O in Tables 4-25 and 4-26. The next loop operation is shown by waveforms N and O and is listed as events P, Q, and R in Tables 4-25 and 4-26.

At the end of each loop operation (events I through R in Tables 4-25 and 4-26), the state of the loop counter is tested. This operation is listed as event S in Tables

Table 4-26. Multiply Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	PU0 ENTER	CLR→R LOOP REG 1. (↑PULLU) (↓PU0*) = CLR→R LOOP (↑RLOOP*/8B2) (↓RLOOP)	R LOOP register cleared.
B.	BP0 THROUGH BP5 ENTER		Common basic phase fundamental operations. See Table 4-20.
C.	BP3 ENTER	INITIALIZE MULTIPLY 1. (↓I00) (↓I01) (↑I02) (↓I03)→ ↓OP1*/7A1 2. (↓OP1*) (↓I06B) (↓I07B) (↓I08B)→ ↓OP100*→↑OP100/11A1 3. (↑I04*) (↑I05B*)→↓I4I5→↑I4I5*/9A1 4. (↑OP100) (↑I4I5*)→↓OP100X*/11B1 5. (↓OP100X*) (↓I09B) (↓I10B) (↓I11N)→ ↓HWM*→↑HWM/11A2 6. ↓HWM* +↑HWD*→↑HWMD→↓HWMD*/11B2 7. (↑HWMD) (↑J00B)→↓MDJ0B*→↑JKK/9B4	These events occur when memory data is loaded into instruction register.
D.	BP4 ENTER	JR→RR 1. (↑BP4MD) (↑HWM) (↑RLOOP*)→ ↓PER*/9A4 2. ↓PER*+↓HWM7*→↑PER/9B1 3. (↑PER) (↑REGCLK)→↓CPR*/9B2 = JR→RR	Multiplicand loaded into R register.
E.	BP5 ENTER	KR→MX→ADDER 1. ↓CNTR3 + ↓CNTR2 + ↓CNTR1* + ↓CNTR0*→↑PS12*→↓PS12/11A4 2. (↑HWM) (↑PS12*)→↓MT2*→↑SELK5/12A4 3. (↑SELK5) (↑BP5)→↓SLKMX*/12A4 4. ↓SLKMX*→↑MXEN→↓MXEN*/9B2 5. ↓SLKMX*→↑MXS0, ↑MXS2/9B2 = KR→MX→ADDER	Multiplier loaded into the adder through the MX multiplexer.
F.	BP5MD ENTER	TEST K00 BIT, IF K00 = 1, SET K0MREG 1. (↑K00B) (↑HWM) (↑RLOOP*)→ ↓STK0M*/8B1 2. ↓STK0M*→↑STK0M/8B1 3. (↑PULLU) (↑STDN7*) = TOGGLE→EN 4. (↑STK0M) (↑PULLU) (↓STK0M*) (↓BP5*) = SET→K0M REG (↑K0M/8B1)	The K0M register is used to remember that the K00TH bit is a 1 prior to the first shift.

Table 4-26. Multiply Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
G.	PU5 ENTER	SET→RLOOP REGISTER 1. ↓HWM*→↑STRL 2. (↑PULLU) (↑PU0*) = TOGGLE→EN 3. (↑STRL) (↑PULLU) (↓RLOOP) (↓BP5*) = SET→RLOOP (↑RLOOP/8B1, ↓RLOOP*)	The R-LOOP register is used to control the pulser recycle operation.
H.	BP6MD ENTER	CLR→OV REG 1. (↑HWM) (↑BP6MD)→↓CLOV*→ ↑ROV→↓ROV*/11B4 = CLR→OV (↓OV, ↑OV*/11B4)	Overflow register is used to detect a negative partial product.
I.	BP6MD LOOP	CLOCK PULSE→UP COUNTER 1. (↑HWM) (↑BP6MD)→↓HWM6*/11A2 2. ↓HWM6*→↑HW6D4/11A3 3. (↑HM6D4) (↑CPPU)→↓CLKCN*→ ↑CLKCNT/11A3 = COUNT→UP TO 12	The up-counter is used to count the number of shifts in the multiply instruction.
J.	BP6MD LOOP	SHIFT JR,KR 1. ↓HWM6*→↑PEJ, ↑PEK/9A4 2. (↑PEK) (↑REGCLK)→↓CPK*/9A4 = SHIFT→KR LEFT, 1 BIT 3. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = SHIFT→JR LEFT, 1 BIT	J and K registers shifted 1-bit left.
K.	BP6MD, LOOP (1-12)	RECYCLE PULSER→↓BP4MD, BP5MD, BP6MD 1. ↑HWD*→↓HWD 2. ↓HWD + ↓RLOOP*→↑HWD*RL*→ ↓HWDRL*/8A1 3. ↓HWDRL*→↑GET OUT*/5B1 4. (↑GET OUT*) (↑HWMD) (↑PS12*) (↑BP6MD)→↓PEPU*/5B1 = PRIME→ PULSER TO BP4 5. ↑CPPU→BP4/5A1	Pulser recycled to BP4 at BP6 when CPPU occurs.
L.	BP4, BP4MD LOOP (1-12)	JR→TS→ADDER, IF K00 = 1 1. (↑HWM) (↑RLOOP)→↓HWMRL*→ ↑HWMRL/11B2 2. ↓HWMRL*→↑TAD4/13B1 3. (↑TAD4) (↑BP4)→↓TSADD*/13B2 4. ↑K00→↓K00*→↑K00B/10A2 5. (↑HWMRL) (↑PS12*) (↑K00B) (↑BP4MD)→↓SLJTS*/12B3→ ↑TSS1, ↑TSS0/13B4 = JR→TS REG	Content of J register switched through TS multiplexer

Table 4-26. Multiply Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
M.	BP4MD LOOP (1-12)	RR→MX→ADDER 1. (↑HWMRL) (↑BP4MD)→↓SLRMX*/8A2 2. ↓SLRMX*→↑MXS1, ↑MXS2/9A2 3. ↓SLRMX*→↑MXEN→↓MXEN*/9A2 = RR→MX	Content of R register switched through to MX.
N.	BP4MD LOOP (1-12)	ADDER→JR = (J + R)→J, IF K00 = 1 1. (↑BP4MD) (↑HWMRL) (↑PS12*) (↑K00B)→↓PEJ*/9A3 = ENABLE→JR 2. ↓PEJ*→↑PEJ→↓CPJ*/9A4/14A4 = LOAD→JR	Content of R and J registers summed and loaded into JR.
O.	BP4MD LOOP (1-12)	IF (K00 = 1), (R + J) = COUT, SET OV 1. (↑K00B) (↑PS12*)→↓MOV*→↑MOV/12B4 2. (↑MOV) (↑COUT) (↑BP4MD) (↑HWMRL)→↓AROV*/12B4 3. (↑SOV*) (↑ROV*) = TOGGLE→OV 4. ↓AROV*→↑CPOV/11B3 5. (↑CPOV) (↑REGCLK)→↑CKOV*/11B4 = SET→OV	If bit K00 is set and the same J + R produces a carry, set overflow register.
P.	BP5 LOOP (1-12)	KR→MX→ADDER Same as E, this table.	K register switched through MX multiplexer.
Q.	BP5MD LOOP (1-12)	+1→ADDER = (K+1) 1. (↑HWM) (↑R LOOP) (↑BP5MD)→↓CIN3*→↑CIN/12A2 = +1→ADDER	Content of K register incremented by 1 count.
R.	BP5MD LOOP (1-12)	ADDER→KR IF OV = 1 1. (↑HWMRL) (↑PS12*) (↑OV)→↓MRLK*→↑PEK5/9B3 2. (↑PEK5) (↑BP5MD)→↓PEK*→↑PEK/9A4 = ENABLE→KR 3. (↑PEK) (↑REGCLK)→↓CPK*/9A4 = LOAD→KR	Content of adder loaded into KR (K + 1) if OV-register is set.
S.	BP6MD (EXIT)	EXIT→RECYCLE PULSER 1. ↓CNTR0→↑CNTR0*/11A3 2. ↓CNTR1→↑CNTR1*/11A3 3. (↑CNTR0*) (↑CNTR1*) (↑CNTR2) (↑CNTR3)→↓PS12*→↑PS12/11A4 4. ↑HWD* + ↓RLOOP*→↑HWD*RL*→↓HWDRL/8A1 5. ↓HWDRL→↑GETOUT*/5B1 6. ↓GETOUT* + ↓HWMD + ↓PS12* + ↓BP6MD→↑PEPU*/5B1 = INHIBIT PULSER RECYCLE AT BP4	After the loop-counter detects the 12th iteration, the pulser BP4 enabling signal is unprimed so that it cannot recycle again.

Table 4-26. Multiply Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
T.	BP4MD (EXIT)	RR→MX→ADDER Same as M, this table.	Content of R-register switched through MX multiplexer.
U.	BP4MD (EXIT)	JR→TS, IF KOM = 1 1. (↑BP4MD) (↑KOM) (↑RLOOP) (↑PS12)→↓SLKTS*/12A4 2. ↓SLKTS*→↑TSS1/13A4 = KR→TS	Content of K-register loaded into TS multiplexer.
V.	BP4MD (EXIT)	ADDER→KR, IF KOM = 1: (R+K)→K 1. (↑BP4MD) (↑RLOOP) (↑PS12) (↑KOM)→↓PEK*/9A3 = ENABLE→KR 2. ↓PEK*→↑PEK/9A4 3. (↑PEK) (↑REGCLK)→↓CPK*/9A4 = ADDER→KR	Adder loaded into K-register if KOM = 1.
W.	BP4MD (EXIT)	OV→1, IF R+K = CARRY WHEN KOM = 1 1. (↑KOM) (↑PS12)→↓MOV*→↑MOV/12B4 2. (↑BP4MD) (↑COUT)(↑MOV) (↑HWMRL)→ ↓AROV*/12A4 3. ↓AROV*→↑CPOV/11B3 4. (↑CPOV) (↑REGCLK)→↑CKOV/11B4 = SET→OV	Overflow register set when R+K produces a carry out when the K0 memory register is set.
X.	BP5 (EXIT)	JR→MX→ADDER 1. (↑HWM) (↑PS12)→↓M12*/12A2 2. ↓MI2*→↑SELJ5, ↑MI2/12A3 3. (↑SELJ5) (↑BP5)→↓SLJMX*/12A3 4. ↓SLJMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 5. ↓SLJMX*→↑MXS2/9A2 = JR→MX	Content of J-register switched through MX multiplexer.
Y.	BP5MD (EXIT)	+1→ADDER = (J+1) Same as Q, this table.	Content of J-register incremented by 1 count.
Z.	BP5MD (EXIT)	ADDER→JR, IF OV = 1 1. (↑HWMRL) (↑PS12) (↑OV) (↑BP5MD)→ ↓PEJ*/9A3 = ENABLE→JR 2. ↓PEJ*→↑PEJ/9A4 3. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = ADDER→JR	Content of adder (J+1) loaded into J-register if overflow set.

Table 4-26. Multiply Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
AA.	BP6MD (EXIT)	JR→RR 1. (↑MI2) (↑BP6MD)→↓PER*/9A4 = ENABLE→RR 2. ↓PER*→↑PER/9A1 3. (↑PER) (↑REGCLK)→↓CPR*/9B2 = JR→RR	Content of J-register is loaded into R-register unconditionally.
AB.	BP6MD (EXIT)	KR→SR 1. (↑MI2) (↑BP6MD)→↓PES*/9A4 = ENABLE→SR 2. ↓PES*→↑PES/9A1 3. (↑PES) (↑REGCLK)→↓CPS*/9B2 = KR→SR	Content of K-register loaded into S-register unconditionally.
AC.	BP6 (EXIT)	SET→DONE, Establish basic phase 1. ↓OP1*→↑SDONE/7B4 2. (↑SDONE) (↑BP6)→↓SDONE*/7B4 = SET→DONE (↑DONE/7B4, ↓DONE*)	When done latch set basic phase enabled.
AD.	BP7	SHIFT R AND S RIGHT 1 BIT 1. (↑HWM) (↑BP7)→↓HWM7*/9B1 2. ↓HWM7*→↑PER, ↑PES/9A1 3. (↑PER) (↑REGCLK)→↓CPR*/9B2 = SHIFT→RR, 1-BIT RIGHT 4. (↑PES) (↑REGCLK)→↓CPS*/9B2 = SHIFT→SR, 1-BIT RIGHT	Content of R and S register shifted 1-bit right.
AE.	PU0	Enter common basic phase	

4-25 and 4-26. If the counter logic produces a true PSI2* signal the exit subperiod is entered; if this signal remains unasserted, the loop operation is reentered for another iteration and events I through R, Tables 4-25 and 4-26 reoccur.

3. Exit. The events that take place during the exit operation are listed as S through AD in Tables 4-25 and 4-26 and are shown by waveforms P through U in Figure 4-22. Initially, the state of the KOM register (waveform E) is tested. If this register is set, the R-register data (waveform L), which is switched through in the MX Multiplexer, and the K register data, which is admitted to the TS multiplexer (waveform P), is summed and admitted into the K register. If the result of this summation produces a carry, the overflow register is set (waveform P). The next operation that occurs is the admission of J-register data through the MX multiplexer (waveform R) and its incrementing (waveform N). If the overflow register is set as the result of the previous summation, this data is admitted into the J register (waveform S). The exit operation is concluded by admitting J-register data into the R register and K-register data into the S register (waveform T). These data are shifted one bit to the right (waveform U) as previously described.

4.4.2.1.2 Hardware Divide (1001). Hardware divide is carried out by the process of successive subtraction to the diviser from the most significant part of the dividend. As each subtraction occurs, the dividend is modified as the result of subtraction whenever subtraction yields a positive number. If the diviser is larger than the significant part of the dividend, subtraction will yield a negative number. Such a negative number is unacceptable and is disregarded by the ND812 hardware divide logic. An identical process takes place when division is done with pencil and paper.

Example A

```
              105 = quotient
   115      12075 = divident
           -115
             57   = 1st subtraction and carry
           -115
            -58   = 2nd subtraction (not acceptable)
             575  = 1st subtraction and carry and 2nd carry
            -575
               0  = 3rd subtraction
```

Because the first subtraction above yields a positive result, the 7 in the dividend was carried to the partial quotient and the divisor again subtracted. However, the second subtraction yields a negative number and this result is unacceptable. Because of this, the remainder obtained as the result of the first subtraction and the carry is used again with the next carry.

These comparisons are mentally made and no thought is given to the process. Normally, even the second subtraction would not have been made. Our familiarity with the decimal number system tells us that whenever the remainder is smaller than the divisor, a negative number will be produced as the result of the next subtraction. In the above example some other short cuts have been taken: for example, we know that if we subtract the divisor from the first and second digits of the dividend, the resultant subtraction yields a negative number. Therefore the division is begun where subtraction yields the first positive result. The mental short cuts are equivalent to three carries. Thus, the first carry in example A is, in effect, actually the fifth carry. When the problem is converted to its binary format, pencil and paper division yields:

Example B

```
                                 1101001  =  quotient
   divisor = 1110011      10111100101011  =  divident
                         -1110011
                          10010011        =  1st subtraction and 9th carry
                          -1110011
                             1110011      =  2nd subtraction, 10th and 11th carries
                            -1110011
                                 1110011  =  3rd subtraction, 12th, 13th and 14th car.
                                 1110011
                                       0  =  4th subtraction
```

In example B, if the divisor is lined up with the seven most significant bits of the dividend, the subsequent subtraction yields a negative binary number because the divisor is larger than that portion of the dividend having the same order of significance; therefore, to yield a positive remainder, the divisor is shifted to that position which yields the first positive result. Effectively, this rightward shift of the divisor is equivalent to a leftward shift of the dividend with each shift corresponding to a carry. In the ND812, the dividend is loaded into the J and K accumulators with the least significant half of the dividend filling the J accumulator. During a divide operation the content of the J and K accumulators is shifted leftward during successive subtractions. Upon inspection of example B, the following is apparent.

1. If subtraction of the divisor from the high order partial dividend produces a positive number, a binary 1 appears in the quotient and the result of subtraction is retained so that, effectively, it becomes the new partial dividend.

2. If subtraction results in a negative number, the partial dividend is unaltered (saved) and a binary 0 appears in the quotient.

3. Effectively, the dividend is shifted leftward one place each time a subtraction occurs and is equivalent to a carry into the partial dividend.

The conclusions enumerated above define the algorithm employed by the ND812 hardware divide logic; however, because the size of the quotient is limited by the capacity of the J accumulator and the size of the divident is limited by the capacity of both the J and K accumulators, the absolute magnitude of the divisor is limited. In the ND812 hardware, the absolute magnitude of the divisor is compared with the most significant part of the dividend. If the divisor is equal to or less than the most significant part of the dividend, the hardware divide logic rejects the argument because if a divide operation were permitted to be carried out the capacity of the quotient register (J accumulator) would be exceeded and the least significant part of the result would be lost.

Whenever the hardware divide logic rejects the divide argument, the ND812 overflow register is set. Hence, the status of the overflow register should be software tested by subsequent programming to determine if the divide operation was executed. In example B, the dividend must be loaded into the J and K accumulators and the divisor loaded into the R subaccumulator.

One other restriction is involved; the most significant digit of the K register must be a zero. If the dividend in example B were 24 instead of 23 bits, when the first shift occurred, the most significant binary 1 would be lost and an erroneous result obtained. After each shift occurs, the divisor is subtracted from the dividend until 12 shifts have occurred, one for each order of significance of the quotient.

The hardware divide flow diagram is shown in Figure 4-23. The first event that occurs (A) is that the overflow register is cleared. Next (B), the content of the K accumulator is subtracted from the content of the R subaccumulator. Subtraction in the ND812

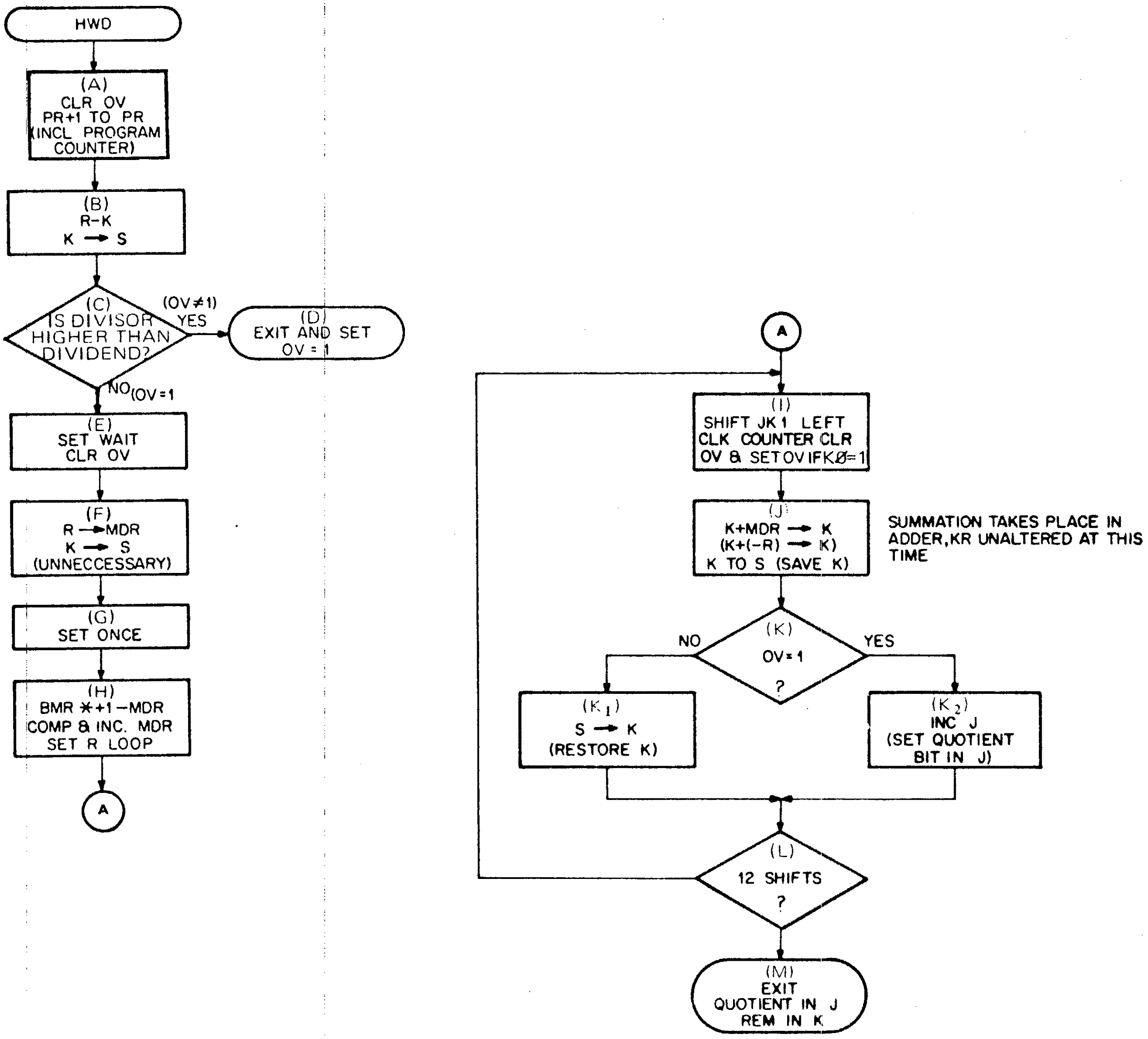


Figure 4-23. Divide Instruction, Flow Diagram

is usually carried out by 2's complement addition. However, in this case, subtraction is carried out by 1's complement addition. The reason for this is that if 2's complement addition is used when the content of both the K anr R registers is zero, 2's complement addition produces an overflow; 1's complement addition does not. Under these curcumstances the hardware would permit division by zero and that leads to trouble. The foregoing illustrates the problem; in 1's complement addition, subtraction of K from R for R greater than, equal to, and less than K gives:

Given $K = 0002_8$, $J = 7453_8$, $R = 0163_8$

(1) If $R > K$, $R = 0003_8$ (or greater) $R - K = (0003_8 + 7775_8 =$ (0V) 0000_8 (overflow set) and $R > K$.

(2) If $R = K$, $R = 0002_8$ $R - K = (0002_8 + 7775_8) = 07777_8$ (overflow not set)

(3) If $R < K$, $R = 0001_8$ (or less) $R - K = (0001_8 + 7775_8) = 07776_8$ (overflow not set and $R < K$)

(4) If $R = K = 0$, $R = 0000_8$ and $R - K = (0000_8 + 7777_8) = 07777_8$ (overflow not set)

Expressions 1 through 3 above show how an overflow results using the problems given in example B of this description when the content of the R accumulator is greater than, equal to, and less than the content of the K accumulator. In 2's complement subtraction the subtrahend would be one binary digit larger and expressions 1,2, and 4 (rather then (1) alone) would yield an overflow making division by zero possible.

Expression (2) is also rejected by the ND812 hardware divide logic because a quotient one bit greater than the capacity of the J accumulator would result whenever the content of the K and R accumulators are equal. If 2's complement subtraction is used, expression (2) yields an overflow, but subsequent division yields an erroneous resultant quotient in the J accumulator. The comparison of the content of the K and R accumulators is made at time (C) as shown in Figure 4-23. Overflow is indicative of a positive result, hence if the overflow register is not set as a result of this comparison, the divide instruction is exited by the logic and the overflow register is set. It is for this reason that the content of the overflow register should be tested by subsequent programming. If the overflow register is set, the hardware divide instruction is processed and event (E) is permitted to occur.

When event (E) occurs, the overflow register is cleared and a register called WAIT is set. The WAIT register is used to defer the first dividend shift and divisor subtraction until the content of the R accumulator (the divisor) can be converted to its 2's complement equivalent. For the divide operation the 2's complement equivalent of the divisor is stored in the memory data register (MDR). Thus, the first step in the hardware divide routine (F) is to get the content of the R accumulator into the MDR. Firgure 4-24 is a modification of Figure 4-13, the ND812 arithmetic data loop, showing the R accumulator-

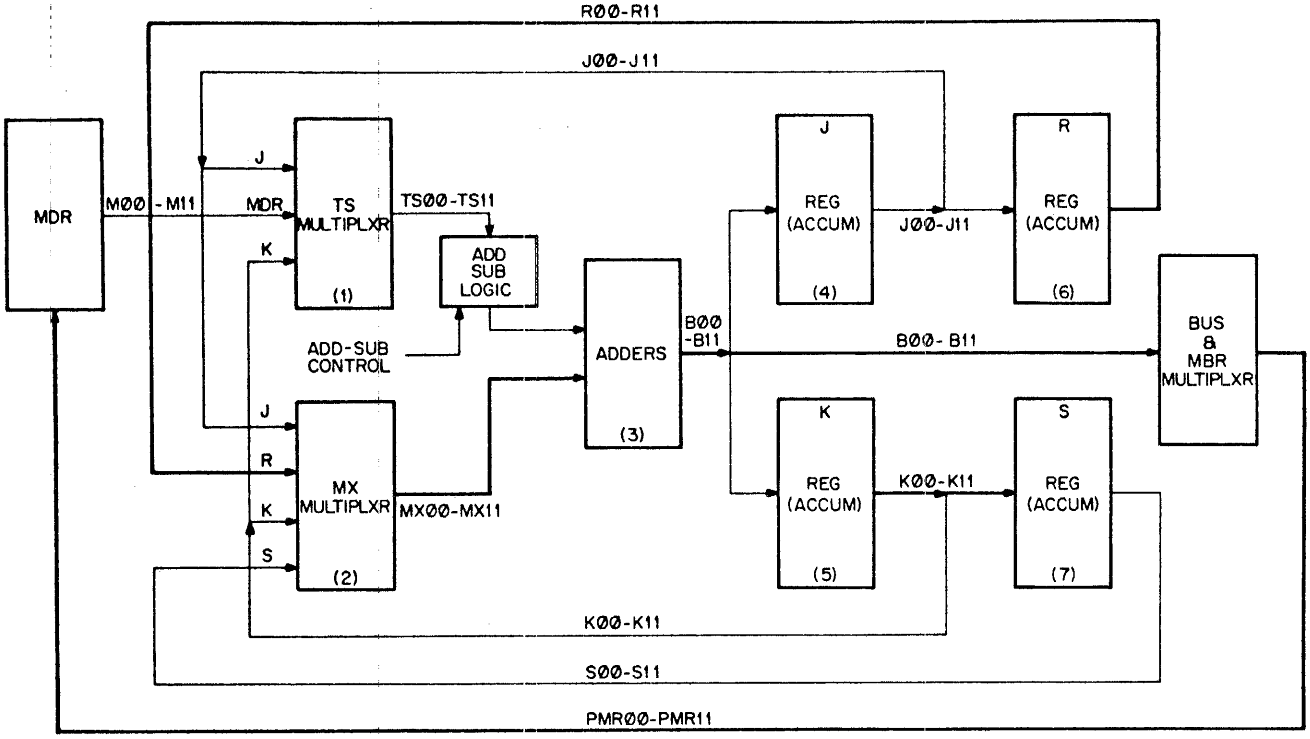


Figure 4-24. Divide Arithmetic Data Loop, Wait Path, Block Diagram

to-MDR data path during this part of the divide operation. The R register-to-MX multiplexer path is enabled and the MX multiplexer loaded with the content of the R accumulator. This data appears as an adder output because the add-subtract gates are disabled at this time and the TS multiplexer output is zero. The adder output is switched through the bus and memory buffer multiplexer to the MDR which is enabled during this event. At this time, the content of the K register is also transferred to the S register; however, this data transfer has no significance yet. When the WAIT register is set the ONCE register is also enabled. The ONCE register permits the content of the MDR to be complemented and incremented to get the 2's complement of the divisor. This event occurs only one time during the divide operation. With the ONCE register set (G) the 2's complement of the divisor is obtained (H).

How this is done can be understood with the aid of Figure 4-25. At the time the WAIT register is enabled the MDR receives the R accumulator data word. When the ONCE register is subsequently enabled, the data path shown in the figure is enabled. The output of the MDR is switched through the TS multiplexer and the 1's complement is selected from the add-subtract logic for application to the adders. Although the content of the R accumulator is held in the MX multiplexer, MX multiplexer output is all zeros because it is not enabled at this time. Simultaneously with the establishment of the ONCE data path, the adders are incremented by a count of 1 to obtain the 2's complement of the add-subtract output data word which currently is the 1's complement of the divisor. The adder output is switched through the bus and memory buffer multiplexer back to he MDR so that the previously loaded divisor is replaced by its 2's complement.

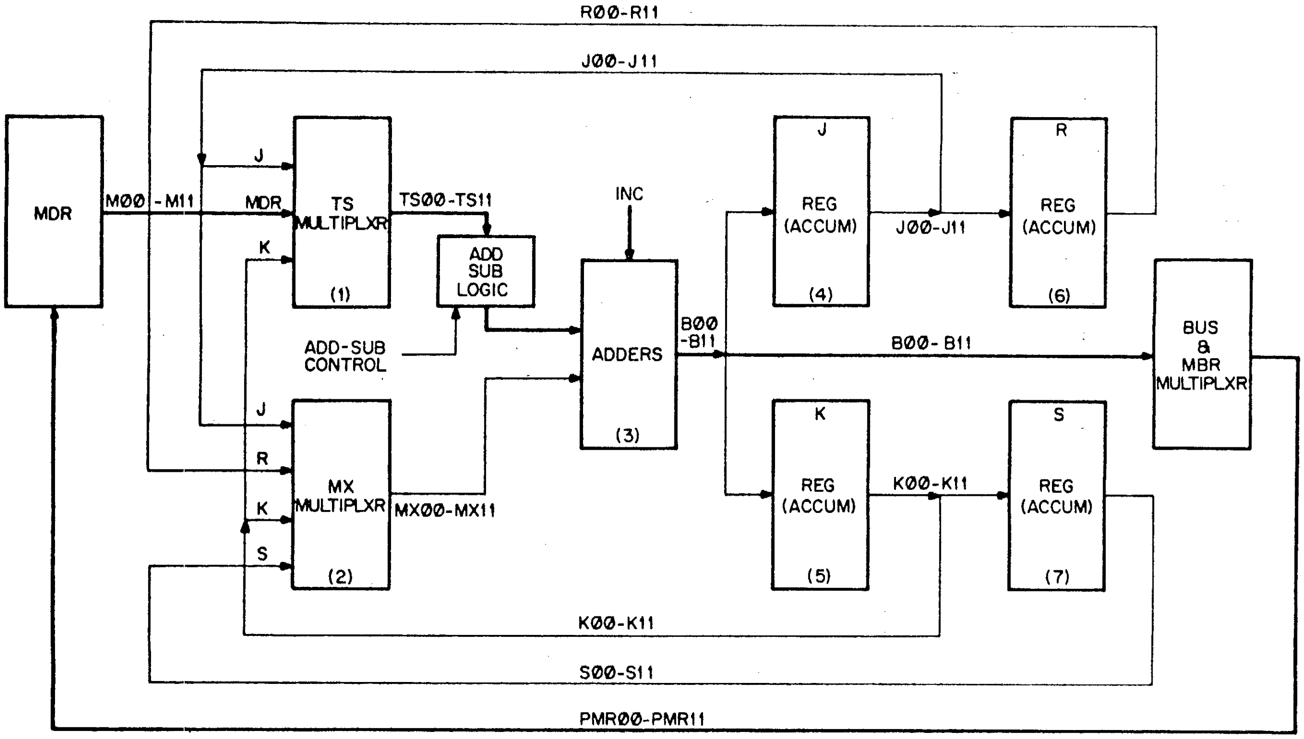


Figure 4-25. Divide Arithmetic Data Loop, Once Path, Block Diagram

As shown in the divide flow diagram (Figure 4-23), the RLOOP register is set simultaneously with the establishment of the ONCE data path. RLOOP is the reiteration loop register and it serves the same function in the divide logic that was served in the multiply logic; it is used to obtain 12 leftward shifts of the dividend by recycling the pulser through periods BP4, BP5, and BP6 twelve times. At the beginning of each reiteration cycle, the up counter is pulsed to record the number of shifts and then the content of the J and K accumulators is shifted one bit to the left. Simultaneously with the J and K shifts, the overflow register is cleared; although it is still cleared from the last clearing operation (E). This clearing operation has significance after every subsequent subtract operation.

Next, the 2's complement of the divisor is added to the content of the K register so that after each shift of the dividend occurs (I), a subtraction of the divisor from the dividend occurs (J). The result of each subtraction is stored in the K register and in the S register to save the K register data word. If the result of the subtraction produces a negative number the overflow register is not set. However, if the subtraction produces a positive number the overflow register is set. (Figure 4-26 shows the divide operation for 12 shifts of the dividend for examples A and B of this description.

This figure has two columns representing the K accumulator and the J accumulator. The dividend is shown loaded in the second row. The numbers at the head of each column indicate the orders of significance of the 24-bit dividend; the zero bit of the K accumulator

	K ACCUM													J ACCUM												
OV	0	1	2	3	4	5	6	7	8	9	10	11		12	13	14	15	16	17	18	19	20	21	22	23	
	0	0	0	0	0	0	0	0	0	0	1	0		1	1	1	1	0	0	1	0	1	0	1	1	DIVIDE
										1	0	1	SK$_1$	1	1	1	0	0	1	0	1	0	1	1	0	SJ$_1$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_1$													
0	1	1	1	1	1	0	0	1	0	0	1	0	(K-R)$_1$													
	0	0	0	0	0	0	0	0	1	0	1	1	SK$_2$	1	1	0	0	1	0	1	0	1	1	0	0	SJ$_2$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_2$													
0	1	1	1	1	1	0	0	1	1	0	0	0	(K-R)$_2$													
								1	0	1	1	1	SK$_3$	1	0	0	1	0	1	0	1	1	0	0	0	SJ$_3$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_3$													
0	1	1	1	1	1	0	1	0	0	1	0	0	(K-R)$_3$													
							1	0	1	1	1	1	SK$_4$	0	0	1	0	1	0	1	1	0	0	0	0	SJ$_4$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_4$													
0	1	1	1	1	1	0	1	1	1	1	0	0	(K-R)$_4$													
						1	0	1	1	1	1	0	SK$_5$	0	1	0	1	0	1	1	0	0	0	0	0	SJ$_5$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_5$													
0	1	1	1	1	1	1	1	0	1	0	1	1	(K-R)$_5$													
					1	0	1	1	1	1	0	0	SK$_6$	1	0	1	0	1	1	0	0	0	0	0	0	SJ$_6$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_6$													
1	0	0	0	0	0	1	0	0	1	0	0	1	(K-R)$_6$	1	0	1	0	1	1	0	0	0	0	0	1	JR→ +1
	0	0	0	0	1	0	0	1	0	0	1	1	SK$_7$	0	1	0	1	1	0	0	0	0	0	1	0	SJ$_7$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_7$													
1	0	0	0	0	0	0	1	0	0	0	0	0	(K-R)$_7$	0	1	0	1	1	0	0	0	0	0	1	1	JR→ +1
						1	0	0	0	0	0	0	SK$_8$	1	0	1	1	0	0	0	0	0	1	1	0	SJ$_8$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_8$													
0	1	1	1	1	1	1	0	0	1	1	0	1	(K-R)$_8$													
	0	0	0	0	1	0	0	0	0	0	0	1	SK$_9$	0	1	1	0	0	0	0	0	1	1	0	0	SJ$_9$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_9$													
1	0	0	0	0	0	0	0	0	1	1	1	0	(K-R)$_9$	0	1	1	0	0	0	0	0	1	1	0	1	JR→ +1
								1	1	1	0	0	SK$_{10}$	1	1	0	0	0	0	0	1	1	0	1	0	SJ$_{10}$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_{10}$													
0	1	1	1	1	1	0	1	0	1	0	0	1	(K-R)$_9$													
							1	1	1	0	0	1	SK$_{11}$	1	0	0	0	0	0	1	1	0	1	0	0	SJ$_{11}$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_{11}$													
0	1	1	1	1	1	1	0	0	0	1	1	0	(K-R)$_{11}$													
						1	1	1	0	0	1	1	SK$_{12}$	0	0	0	0	0	1	1	0	1	0	0	0	SJ$_{12}$
	1	1	1	1	1	0	0	0	1	1	0	1	MDR$_{12}$													
1	0	0	0	0	0	0	0	0	0	0	0	0	(K-R)$_{12}$	0	0	0	0	0	1	1	0	1	0	0	1	

Figure 4-26. Hardware Divide Operation for Examples A and B.

is the most significant digit of the 24-bit dividend. Although they may seem unnecessary, the low-order zeros in the 24-bit dividend must be loaded into the K accumulator to establish the position of the first positive result of subtraction. Thus, although the significant part of the dividend is equivalent to 2_8, 0002_8 must be loaded into the K accumulator and 7453_8, the least significant portion of the dividend, must be loaded into the J accumulator.

Between the two columns, the shift, the content of the MDR, and the result of the subtraction is given as follows; symbols SK_1 through SK_{12} indicate the required 12 shifts of the K register. Symbols MDR_1 through MDR_{12} indicate the number of times the content of the MDR (7615_8) is added (2's complement added) to the content of the K accumulator, and symbols $(K - R)_1$ through $(K - R)_{12}$ indicate the resultant binary number after addition. For any addition in which an overflow is not produced, the result is a negative number if the K0th bits is a binary 1. This occurs in all cases when the overflow register is not set.

As can be seen from the figure, when the overflow register is not set, the content of the K register, prior to the last subtraction, is used as the partial dividend. However, when the overflow register is set, the result is used as the dividend. Also, when an overflow occurs, the content of the J accumulator is incremented by a count of 1 to establish the value of the quotient. For the J accumulator column in Figure 4-26, the symbols SJ_1 through SJ_{12} indicate shifts of the content of the J accumulator. When an overflow occurs, the J accumulator is shown modified by the incrementing count prior to the next shift. The result of the divide operation yields a zero remainder in the K accumulator and the quotient in the J accumulator. If a remainder had resulted when the divide operation was completed it would have appeared in the K accumulator. In the figure, significant zeros of the K accumulator have been dropped to aid in understanding a somewhat complex example.

The data path employed during the addition (J, Figure 4-23) is shown in Figure 4-27. The content of the K accumulator is admitted into the MX multiplexer and is switched through the MX multiplexer to the adders. Simultaneously the content of the MDR (divisor 2's complement) is switched through the bus and memory buffer multiplexer where the data word is selected by the add-subtract logic and summed with the content of the MX multiplexer by the adders. This information is held in the adders until the overflow register is tested. Simultaneously, the content of the K accumulator is saved in the S subaccumulator.

When the overflow register is tested (K, Figure 4-23) if the result of the addition (J) produced a negative number the content of the S register is transferred back to the K register (K_1). However if the result of addition produced a positive number the content of the J accumulator is incremented (K_2) and the resultant dividend now held in the K accumulator is unaltered. When the next loop takes place either the new content of K is used, or the previous content (which is now held in the S subaccumulator) is used. These data paths are shown in Figures 4-28 and 4-29.

For the case where the overflow register is unset, the path shown in Figure 4-28 is enabled. The content of the S register, which now holds the result of the previous

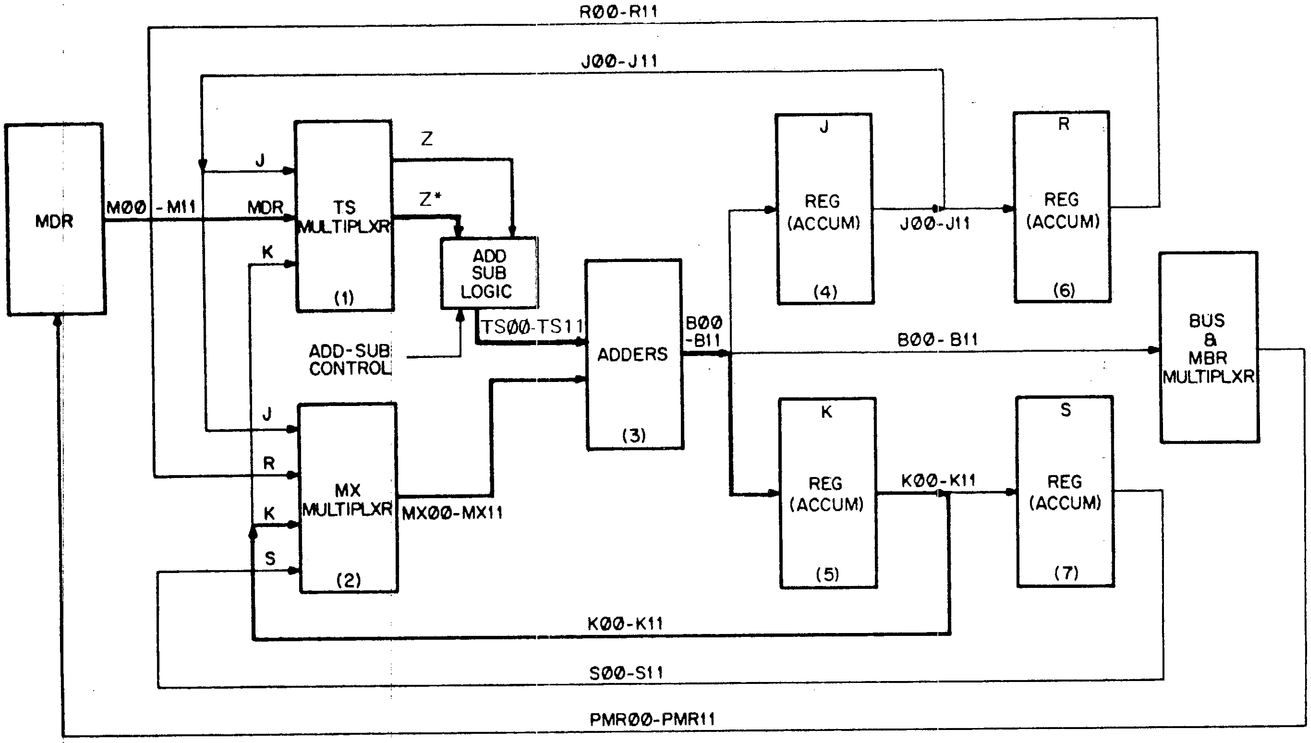


Figure 4-27. Divide Arithmetic Data Loop, Summation, Block Diagram

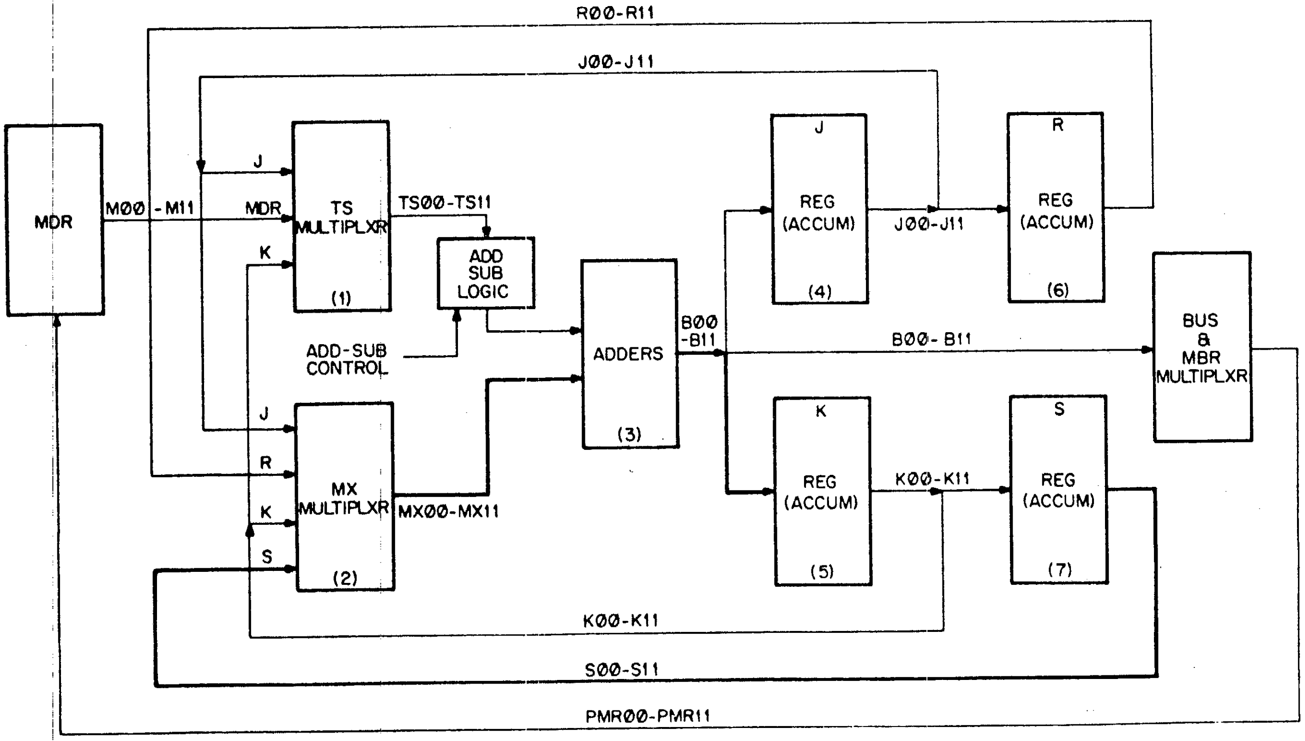


Figure 4-28. Divide Arithmetic Data Loop, S to K Register Data Path, Block Diagram

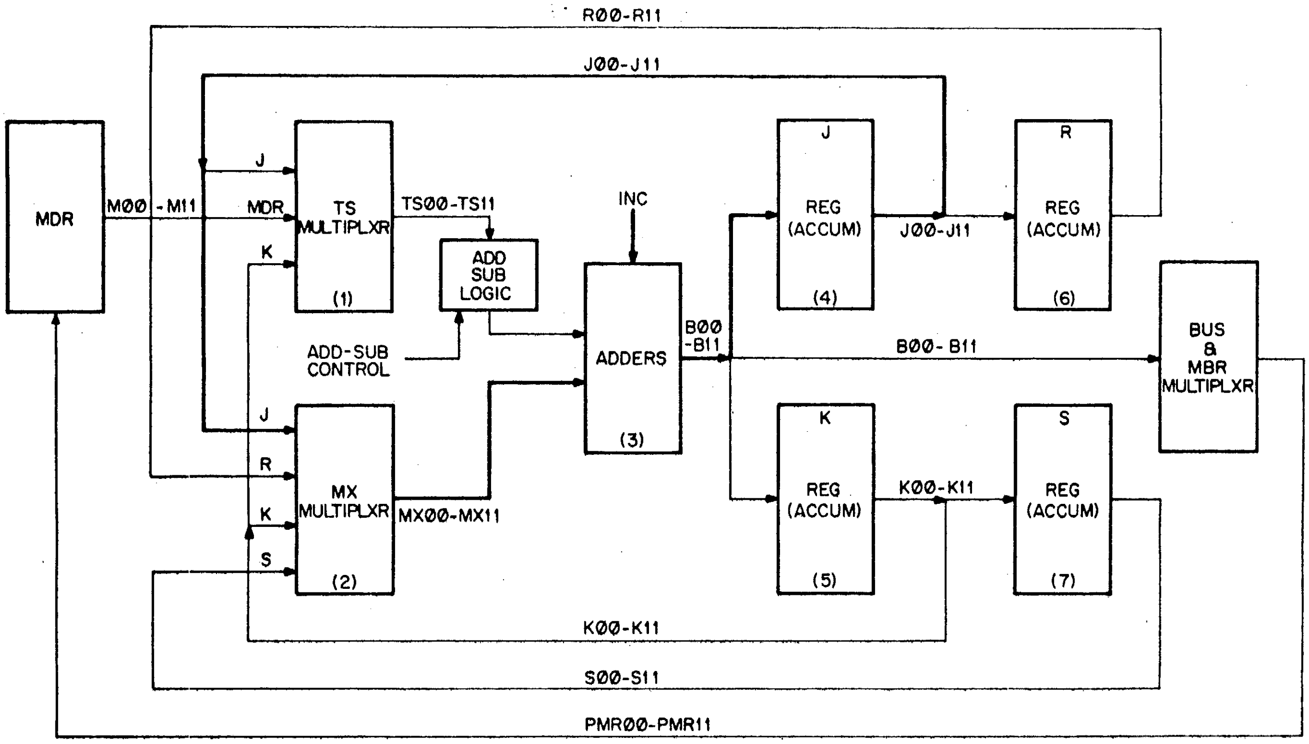


Figure 4-29. Divide Arithmetic Data Loop, J-Register Increment Data Path, Block Diagram

summation is admitted to the MX multiplexer and switched through the MX multiplexer to the adders. The output of the adders is admitted to the K register resulting in a transfer of the S register data into the K register.

For the case where the overflow register is set, the path shown in Figure 4-29 is enabled. The content of the J register is admitted to the MX multiplexer and switched through the MX multiplexer to the adders. Simultaneously the adders are incremented and their resultant output is admitted back into the J register. It is this operation that produces the quotient. After the 12th iteration, the divide operation is completed with the quotient in the J register and the remainder in the K register.

Fundamental Operation - This paragraph describes the fundamental operations that are required to implement the hardware divide logic. For data flow, refer to Figures 4-24, 4-25, and 4-26 through 4-29. Figure 4-30 is the divide timing diagram. Table 4-27 is the divide logic event summary and Table 4-28 describes the simplified logic.

As with the multiply operation, the conditions required to initialize the divide operation occur after the common basic phase events up to period BP3 occur. These events are listed in reference C of Tables 4-27 and 4-28 and suggested by waveform B in Figure 4-22, the timing diagram. The events that occur during a divide operation are also subdivided into subperiods as shown in the timing diagram and Table 4-28; the divide subperiods are called enter, wait, once, loop and exit.

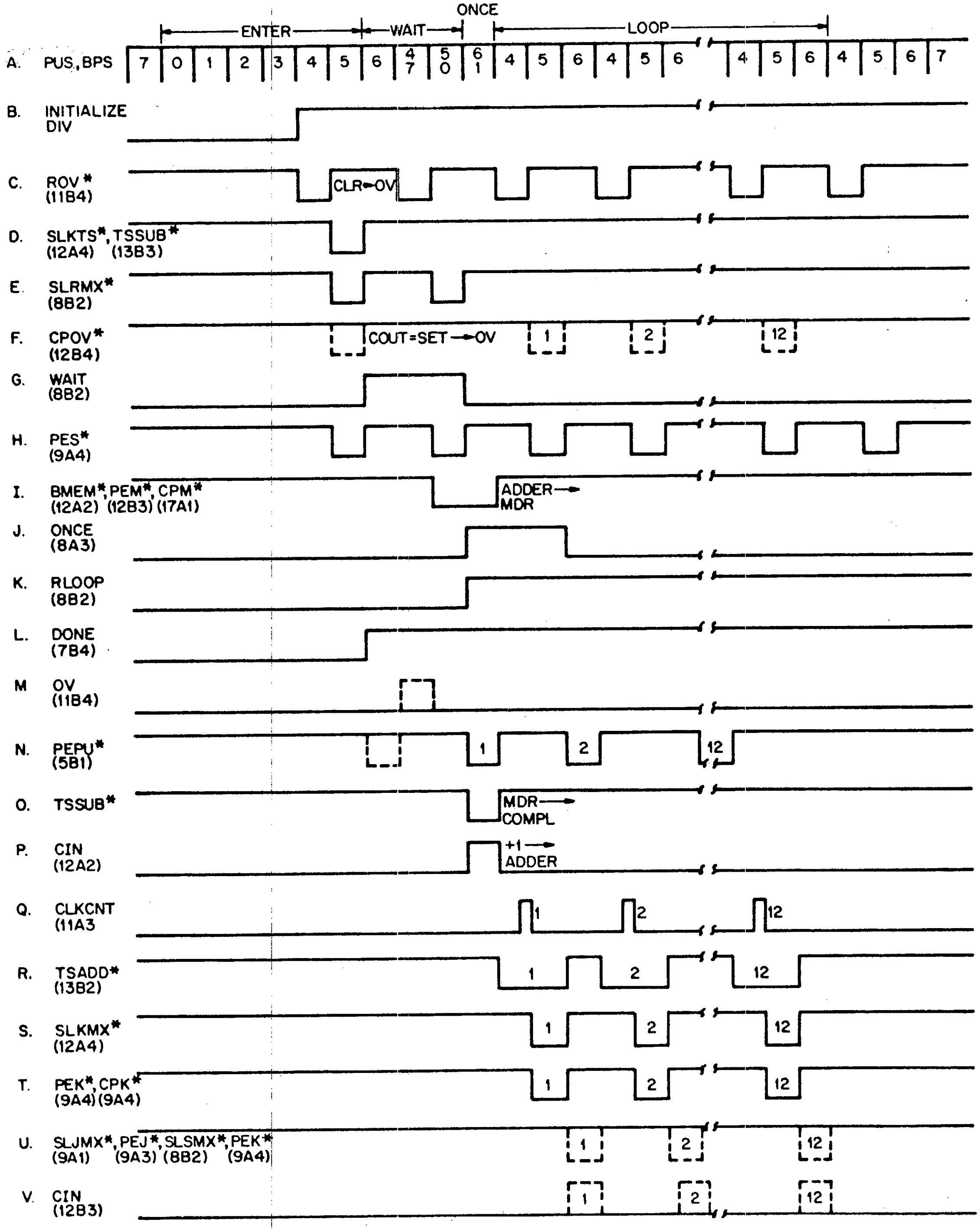


Figure 4-30. Divide Instruction, Timing Diagram

1. Enter. During the enter subperiod of the divide operation, the content of the K register is complemented to get a 1's complement of the most significant part of the dividend so that its magnitude can be compared with the magnitude of the divisor. These operations are described by A through H in Tables 4-27 and 4-28. Initially the overflow register is cleared (D, Tables 4-27 and 4-28) when pulse PU4 occurs (waveform C, Figure 4-30). This clearing operation occurs unconditionally during every period 4 of a divide operation. Next, the content of the K register (most significant part of the dividend), is admitted into the TS multiplexer and is complemented so that one input to the adder is effectively -K (E and F, Tables 4-27 and 4-28 and waveform D). The other input to the adder is the content of the R register (divisor) which is routed through the MX multiplexer (G, Tables 4-27 and 4-28 and waveform E). This event occurs only if register R LOOP (wavefom K) is not set.

If the resultant summation does not produce an overflow (positive result), the overflow register is not set (H, Tables 4-27 and 4-28 and wavefom F), and the hardware routine is exited.

2。 Wait. The purpose of the wait subperiod is to test the magnitude of the divisor to determine if it is greater than that of the dividend. Depending on whether a carry is produced by the adder (H, Tables 4-27 and 4-28 and waveform E), the divide routine may or may not continue. Thus, the WAIT subperiod may either cause the pulser to be recycled or the routine to be exited. In either case, the WAIT register is set and the content of the K register is saved in the S register (I and J, Tables 4-27 and 4-28 and waveforms G and H, respectively).

For the case when the overflow register is not set (negative result for R-K), the events described by operations K, L, and M in Tables 4-27 and 4-28 remain. If the overflow register is not set (waveform F), the pulser is not recycled and period PU7 is permitted to occur during the WAIT subperiod. Thus, the unset overflow register (waveform F) is set (waveform M). When the first BP6 period occurs, the DONE latch is set unconditionally so that when period BP7 occurs, the instruction is exited.

For the case when the overflow register is set (positive result for R-K), the events described by operation K, Table 4-28 is inhibited because signal OV* remains low level and low-level signal GET OUT* is not produced. Hence, signals GET OUT*, HWMD, BP6MD, and PS12* are high level and the pulser is primed to enter period BP4 when period BP6 occurs (waveform N). Thus, the hardware divide operation is permitted to continue.

At the conclusion of the WAIT subperiod, the content of the R register (O, Tables 4-27 and 4-28 and waveform E) is routed through the adder, the bus and memory buffer multiplexer (P, Tables 4-27 and 4-28 and waveform I), and to the memory data register. This is possible because in the absence of any asserting signals, the content of the adder is routed through the bus and memory buffer multiplexer. Because during this time the memory data register is enabled by low-level signal PEM*, the content of the adder is admitted to the memory data register. During this period, both the ONCE and R LOOP registers (Q and R, Tables 4-27 and 4-28) are set.

Table 4-27. Divide Instruction, Event Summary

Ref.	Period	Event
A.	PU7	CLR→WAIT, ONCE
	PU0	CLR→R LOOP
B.	BP0-BP5	COMMON BASIC PHASE
C.	BP3	INITIALIZE DIVIDE
D.	BP4MD	CLR→OV
E.	BP5	KR→TS
F.	BP5MD	TS→COMPL→ADDER (-K)
G.	BP5MD	RR→MX→ADDER
H.	BP5MD	SET→OV, IF R-K = CARRY OUT
I.	BP5MD	KR→SR
J.	BP5*	SET→WAIT
K.	BP6MD	RECYCLE PULSER→BP4, BP5, BP6, IF OV ≠ 1
L.	BP6	SET→DONE
M.	BP7	SET→OV, IF ↓ GET OUT*
N.	BP4	CLR→OV
O.	BP5MD	RR→MX→ADDER
P.	BP5MD	ADDER→BUS + MDR (RR→MDR)
Q.	BP5*	SET→ONCE
R.	BP5*	SET→R LOOP
S.	BP6MD	ADDER→BUS + MDR→MDR
T.	BP6MD	TS COMPL→ADDER (R = 2s compl)
U.	BP4MD	CLOCK PULSE→UP COUNTER
V.	BP4MD	CLR→OV
W.	BP4MD	SHIFT JR, KR LEFT 1 BIT
X.	PU45	-MDR→ADDER
Y.	BP5	KR→MX→ADDER
Z.	BP5MD	ADDER→KR
AA.	BP5MD	KR→SR
AB.	BP5MD	SET→OV, IF K-R = CARRY OUT
AC.	BP6MD	JR→MX→ADDER, IF OV = 1
AD.	BP6MD	+1→ADDER, IF OV = 1
AE.	BP6	ADDER→JR, IF OV = 1
AF.	BP6MD	SR→MX→ADDER, IF OV ≠ 1
AG.	BP6MD	ADDER→KR, IF OV ≠ 1
AH.	BP6MD	EXIT→INHIBIT PULSER
AI.	PU7	CLR→WAIT, ONCE
AJ.	PU0	CLR→R LOOP, ENTER COMMON BASIC PHASE

Table 4-28. Divide Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	PU7, PU0 ENTER	CLR→WAIT, ONCE, R LOOP registers 1. ↓STDN7*/3B3 = CLR→WAIT, ONCE (↓WAIT/8B2, ↑WAIT*) (↓ONCE/8B3, ↑ONCE*) 2. (↑PULLU) (↓PU0*) = CLR→R LOOP (↓R LOOP/8B2, ↑R LOOP*)	Wait, once, and R-LOOP registers cleared.

Table 4-28. Divide Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
B.	BP0 THROUGH BP5 Enter	Common basic phase fundamental operations, Table 4-20	See table 4-20.
C.	BP3 Enter	INITIALIZE DIVIDE 1. (↓I00) (↓I01) (↑I02) (↓I03)→ ↓OP1*/7A1 2. (↓OP1*) (↓I06B) (↓I07B) (↓I08B)→ ↓OP100*/11A1 3. (↑I04*) (↑I05B*)→↓I4I5→↑I4I5*/9A1 4. (↑OP100) (↑I4I5*)→↓OP1-00X*/11B1 5. (↓OP10X*) (↓I09B) (↓I10B) (↑I11N)→ ↓HWD*, ↑HWD/11B1 6. ↓HWD* + ↓HWM*→↑HWMD→ ↓HWMD*/11B2 7. (↑HWMD) (↑J00B)→↓MDJ0B*→ ↑JKK/9B4 = PRIME KR→ ↑K00 (after clock) 8. (↑HWD) (↑RLOOP*)→↓HWD*RL*→ ↑HWDRL/8A1	These events occur when memory data is loaded into instruction register.
D.	BP4MD Enter	CLR→OV 1. (↑HWD) (↑4MC*) (↑BP4MD)→↓CLOV*→ ↑ROV→↓ROV*/11B4 = CLR – OV	Overflow register is used to detect a negative result of subtraction.
E.	BP5 Enter	KR→TS REG 1. (↑HWDRL*) (↑WAIT*)→↓DR*LW*→ ↑OP1K/12A4 2. (↑OP1K) (↑BP5)→↓SLKTS*/12A4 = ENABLE TS 3. ↓SLKTS*→↑TSS1/13A4 = KR→TS	Content of K-register is switched through TS multiplexer.
F.	BP5MD Enter	TS COMPL→ADDER 1. ↓DR*LW*→↑SUB5/13B3 2. (↑SUB5) (↑BP5B')→↓TSSUB*/13B3 = COMPL→TS (-K→ADDER)	Ones complement of K-register data to adder.
G.	BP5MD Enter	RR→MX→ADDER 1. (↑HWD) (↑RLOOP*)→ ↓HWD*RL*→↑SELR23/8A1 2. (↑SELR23) (↑BP5MD)→↓SLRMX*/8B2 3. ↓SLRMX*→↑MXEN→↓MXEN*/9A3 = ENABLE→MX 4. ↓SLRMX*→↑MXS2, ↑MXS1/9A2 = RR→MX	Content of R-register switched through MX multiplexer and summed with -K.

Table 4-28. Divide Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
H.	BP5MD Enter	SET→OV, IF R-K POSITIVE 1. (↑BP5MD) (↑COUT) (↑HWD)→ ↓AROV*/12B4 2. ↓AROV*→↑CPOV/11B3 3. (↑CPOV) (↑REGCLK)→↓CKOV /11B4 = SET→OV (↑OV/12B4, ↓OV*)	The overflow register is set if summation produces a carry output from the adder.
I.	BP5MD Wait	KR→SR 1. ↓HWD*→↑OP1ES/9A4 2. (↑OP1ES) (↑BP5MD)→↓PES*/9A4 = ENABLE→SR 3. ↓PES*→↑PES/9B1 4. (↑PES) (↑REGCLK)→↓CPS*/9B2 = KR→SR	The content of the K-register is admitted into the S-register.
J.	BP5* Wait	SET→WAIT REG 1. (↑PULLU) (↑STDN7*) = TOGGLE→ ENABLE 2. (↑RLOOP) (↑HWD) (↓BP5*) = SET→WAIT (↑WAIT/8B2, ↓WAIT*)	The wait-register is set.
K.	BP6MD Wait	INHIBIT RECYCLE, IF OV ≠ 1 1. (↑OV*) (↑HWDRL)→↓GET OUT*/5B1 2. ↓GET OUT* + ↓HWMD + ↓BP6MD + ↓PS12*→↑PEPU*/5B1 = INHIBIT PULSER RECYCLE	The pulser is not recycled if the result of R-K does not yield a positive number.
L.	BP6 Wait	SET→DONE 1. ↓OP1*→↑SDONE/7B4 2. (↑SDONE) (↑BP6)→↓SDONE*/7B4 = SET DONE (↑DONE/7B4, ↓DONE*)	Done latch is set unconditionally at BP6.
M.	BP7 Wait	SET→OV, IF↓GET OUT* 1. (↑HWDRL) (↑BP7)→↓SOV*/11B3 = SET→OV (↑OV/11B4, ↓OV*)	The overflow register is set and instruction executed if pulser not recycled.
N.	BP4 Wait	CLR→OV Same as D, this table.	Overflow register is cleared.
O.	BP5MD Once	RR→MX→ADDER Same as G, this table.	

Table 4-28. Divide Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
P.	BP5MD Once	ADDER→BUS + MBR→MDR 1. (↑WAIT) (↑BP5MD)→↓WAIT5*/12B1 2. ↓WAIT5*→↑DRDB/12B1 3. (↑DRDB) (↑HWD)→↓BMEM*/12B2 4. ↓BMEM*→↑RDB→↓RDB*/12B3 = SELECT ADDER→BUS + MBR 5. ↓BMEM*→↑PEM→↓PEM*/12B3 = ENABLE→MDR 6. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = ADDER→MDR	The content of the R-register is admitted to the MDR.
Q.	BP5* Once	SET→ONCE REG 1. (↑PULLU) (↑STDN7)→TOGGLE ENABLE 2. (↑HWD) (↑WAIT) (↓BP5*) = SET→ ONCE (↑ONCE/8B3, ↓ONCE*)	Once register is used to complement R-register data.
R.	BP5* Once	SET→R LOOP REG 1. ↓WAIT→↑STRL/8B1 2. (↑PULLU) (↑PU0*) = TOGGLE→ ENABLE 3. (↑STRL) (↓BP5*) = SET→R LOOP (↑R LOOP, ↓RLOOP*/8B2) 4. ↓HWD + ↓R LOOP*→↑HWD*RL*→ ↓HWDRL/8A1	R-LOOP register set.
S.	BP6MD Once	ADDER→BUS + MBR→MDR 1. (↑ONCE) (↑BP6MD)→↓ONCE6*/12B1 2. ↓ONCE6*→↑DRDB/12B1 3. (↑DRDB) (↑HWD)→↓BMEM*/12B1 4. ↓BMEM*→↑RDB→↓RDB*/12B3 = SELECT ADDER→BUS + MBR 5. ↓BMEM*→↑PEM→↓PEM*/12B3 = ENABLE→MDR 6. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = ADDER→MDR	The content of the R-register is admitted to the MDR.
T.	BP6B Once	TS COMPL→ADDER 1. (↓ONCE*)→↑SUB6/13B3 2. (↑SUB6) (↑BP6B)→↓TSSUB*/13B3 = TS→COMPL (MDR→COMPL) 3. ↓HWD + ↓ONCE*→↑DCIN*/12A1 4. (↑DCIN*) (↑TSSUB) (↑OP2*)→ ↑CIN/12A2 = +1→ADDERS (2s COMPL, MDR)	The content of the MDR is admitted to the TS multiplexer. The TS MX is complemented and the adder incremented.

Table 4-28. Divide Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
U.	BP4MD Loop	CLOCK PULSE→UP COUNTER 1. (↑HWD) (↑RLOOP)→↓HWDRL*→ HWDRL/11B2 2. (↑HWDRL) (↑BP4MD)→↓HWD4* 3. ↓HWD4*→↑HM6D4/11A3 4. (↑HM6D4) (↑CPPU)→↓CLKCN*/11A3→ ↑CLKCNT = COUNT→UP TO 12	The up counter is used to count the number of shifts in the divide instruction.
V.	BP4MD Loop	CLR→OV Same as D, this table.	
W.	BP4MD Loop	SHIFT→JR, KR 1. ↓HWD4*→↑PEJ, ↑PEK/9A4 2. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = SHIFT→JR LEFT, 1 BIT 3. (↑PEK) (↑REGCLK)→↓CPK*/9A4 = SHIFT→KR LEFT, 1 BIT	J and K register shifted 1-bit left.
X.	PU45 Loop	-MDR→ADDER 1. ↓HWDRL*→↑TSEXT/13B1 2. (↑TSEXT) (↑PU45)→TSADD*/13B1 = MDR→ADDER	The content of the MDR (-R) is admitted to adders.
Y.	BP5 Loop	KR→MX→ADDER 1. ↓HWDRL*→↑SELK5/12A4 2. (↑SELK5) (↑BP5)→↓SLKMX*/12A4 3. ↓SLKMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 4. ↓SLKMX*→↑MXS0, ↑MXS2/9B2 = KR→MX→ ADDER	The content of the MDR and K-register is summed [K+(-R)].
Z.	BP5MD Loop	ADDER→KR 1. ↓HWDRL*→↑PEK5/9B2 2. (↑PEK5) (↑BP5MD)→↓PEK*/9A3 = ENABLE→KR 3. ↓PEK*→↑PEK/9A4 4. (↑PEK) (↑REGCLK)→↓CPK*/9A4 = ADDER→KR	Content of adder loaded into K-register.
AA.	BP5MD Loop	KR→SR Same as J, this table.	
AB.	BP5MD Loop	SET→OV, IF K-R = CARRY OUT Same as I, this table.	

Table 4-28. Divide Instruction, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
AC.	BP6MD Loop	JR→MX→ADDER, IF OV = 1 1. (↑OV) (↑HWD) (↑BP6MD)→ ↓SLJMX*/12A3 2. ↓SLJMX*→↑MXEN→↓MXEN*/9A2 = MX→ENABLE 3. ↓SLJMX*→↑MXS2/9A2 = JR→MX	If overflow register is set, admit content of J-register to MX multiplexer.
AD.	BP6MD Loop	+1→ADDER, IF OV = 1 1. (↑OV) (↑4MCDD*) (↑HWDRL) (↑BP6MD)→↓CIN3*/12B2 2. ↓CIN3*→↑CIN/12A2 = +1→ADDER	If overflow register is set, increment adder.
AE.	BP6 Loop	ADDER→JR, IF OV = 1 1. (↑HWDRL) (↑OV)→↓RLD*/9A2 2. ↓RLD*→↑PEJ6/9A3 3. (↑PEJ6) (↑BP6)→↓PEJ*/9A3 = ENABLE→JR 4. ↓PEJ*→↑PEJ/9A4 5. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = ADDER→JR	If overflow register is set, admit content of adder to J-register (J+1→J).
AF.	BP6MD Loop	SR→MX→ADDER, IF OV = 1 1. (↑DCIN) (↑OV*) (↑BP6MD) ↓SLSMX*/8B2 2. ↓SLSMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 3. ↓SLSMX*→↑MXS0/9B2, ↑MXS1/9B2 = SR→MX	If the overflow register is not set, the S-register, (saved F) is admitted to the MX multiplexer.
AG.	BP6MD Loop	ADDER→KR, IF OV ≠ 1 1. (↑HWDRL) (↑OV*) (↑ONCE*)→ ↓DRLO*→↑PEK6/9A3 2. (↑PEK6) (↑BP6)→↓ PEK*/9A3 = ENABLE→KR 3. ↓PEK*→↑PEK/9A4 4. (↑PEK) (↑REGCLK)→↓CPK*/ 9A4 = ADDER→KR	If the overflow register is not set, the content of the adder is admitted into the K-register.
AH:	BP6MD Exit	EXIT→INHIBIT PULSER 1. ↓CNTR0→↑CNTR0* /11A3 2. ↓CNTR1→↑CNTR1*/11A3 3. (↑CNTR0*) (↑CNTR1*) (↑CNTR2) (↑CNTR3)→↓PS12*→ ↑PS12/11A4 4. ↑HWD + ↓RLOOP*→↑HWD*RL*/8A1→ ↓HWDRL/8A1 5. ↓HWDRL→↑GET OUT*/5B1 6. ↑GET OUT* + ↓HWMD + ↓PS12*+ ↓BP6MD→↑ PEPU*/5B1 = INHIBIT PULSER RECYCLE AT BP4	After the up counter detects the 12th iteration, the pulser BP4 enabling signal is unprimed so that it cannot recycle again.

3. Once. The purpose of the ONCE subperiod is to produce the 2's complement of the divisor so that when it is summed with the most significant part of the dividend, the result is subtraction. This is accomplished by enabling the MDR-to-MDR loop through the TS multiplexer and incrementing the adder to yield the 2's complement of the divisor (-R). These operations occur during period 6 (S and T, Table 4-27 and 4-28 and waveforms I, O, and P, respectively). At the conclusion of period 6, period 4 is reentered and the divide loop operation commences.

4. Loop. The loop subperiod permits the actual divide arithmetic operations to occur. During each period 4, the overflow register is cleared, the up counter is pulsed, and the contents of the J and K registers (dividend) is shifted one bit to the left (U, V, and W, Tables 4-27 and 4-28 and waveforms C and Q, respectively). Also, during each loop, when period 5 occurs, the content of the memory data register and the K register (K-R) are summed (X and Y, Tables 4-27 and 4-28, waveforms R and S). The result of the summation is stored in the K and S registers (Z and AA, Tables 4-27 and 4-28 and waveforms T and H, respectively).

If the results of the loop summation is positive, a carry out is produced from the adder. Any resulting carry out signal sets the overflow register (AB Tables 4-27 and 4-28, and waveform F). If the result of the summation yields a positive number, the content of the J register is incremented and restored into the J register (AC, AD, and AE, Tables 4-27 and 4-28, and waveform U). If the result of the summation is negative, no carry out is produced from the adder and the overflow register is not set. In that event, the content of the S register (unsummed K register data) is stored in the K register for the next iteration (AF and AG, Tables 4-27 and 4-28. waveform U). After the twelfth iteration, the divide instruction is exited by inhibiting the pulser recycle mode (AH, Tables 4-27 and 4-28).

4.4.2.1.3 Logical AND Subgroup. For this subgroup, Figure 4-31 is the block diagram, Figure 4-32 is the composite timing diagram, Table 4-29 is the Event Summary which lists all fundamental operations for the subgroup, Table 4-30 is the instruction listing in terms of the Fundamental Operation.

The logical AND operation is characterized by the logical combination of two corresponding bits of two data words to be ANDed. For example, when bits J0 and K0 are binary 1's, the result of the operation is 1 only when both bits are also 1. Figure 4-31 illustrates how the bits of two words are ANDed through the arithmetic data loop. Hence, corresponding bits of data words stored in the J (5) and K (6) registers are ANDed (1) and the results appear at the input to the enabled utility gates (2). The utility gate outputs are admitted to the MX multiplexer (3) and adder (4) so that the content of the adder replaces the content of the J register (instruction 1100), the K register (instruction 1200), or both (instruction 1300).

OP100* - All processing paths for this instruction subgroup are initialized by signal which is output from the operate instruction decoder.

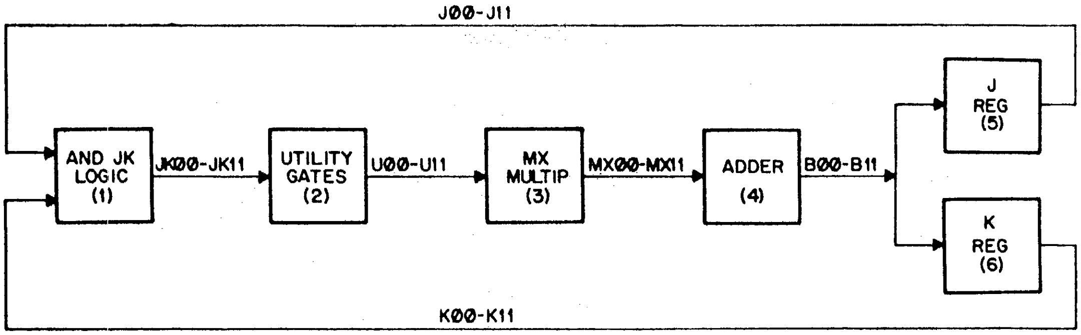


Figure 4-31. Logical AND Subgroup, Block Diagram

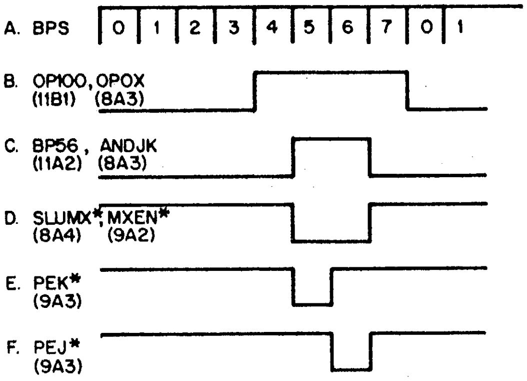


Figure 4-32. Logical AND Subgroup, Timing Diagram

Table 4-29. Logical AND Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
AND J	1100	A.	BP0-BP6	COMMON BASIC PHASE (Table 4-23)
		B.	BP56	UT→MX
		C.	BP3	AND J,K
			BP6	ADDER→J
AND K	1200	A.	BP0-BP6	COMMON BASIC PHASE (Table 4-23)
		B.	BP56	UT→MX
		D.	BP3	AND J,K
			BP5MD	ADDER→K
AND JK	1300	A.	BP0-BP6	COMMON BASIC PHASE (Table 4-23)
		B.	BP56	UT→MX
		C.	BP3	AND J,K
		D.	BP5MD	ADDER→K
			BP6	ADDER→J

Addressing and activating the MX and TS multiplexers is described in Section I. Transfer of data from the adder to the J and K registers is also discussed in Section I. It is important to observe, however, that all register clock signals, for example, CPK* and CPJ*, are gated by signal REGCLK which occurs during each timing period.

It should also be noted that the TS multiplexer is not shown on the block diagram. The reason for this is that the TS multiplexer does not play an active role in the execution of this instruction subgroup; it merely contributes zeros to the adder since neither an add nor a subtract operation is selected.

ANDJ, 1100 = Logical AND operation is performed on all twelve bit-pairs of the J and K registers. The result replaces the content of the J register.

ANDK, 1200 - Logical AND operation is perfomed on all twelve bit-pairs of the J and K registers. The result replaces the content of the J register.

ANDJK, 1300 - Logical AND operation is perfomed on all twelve bit-pairs of the J and K registers. The result replaces the contents of both the J and K registers.

Table 4-30. Logical AND Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1100) (1200) (1300)	BP56	UG→MX→ADDER 1. ↓OP100*→↑OP100/11A1 2. (↑OP100) (↑I09*) (↑I10*) (↑I11B*)→ ↓OP0X*/8A2 3. ↓OP0X*→↑OP0X/8A3 4. (↑OP0X) (↑BP56) (↑I415)→↓ANDJK*→ ↑ANDJK/8A3 5. ↓ANDJK*→↑SELU/8A3→↓SLUMX*/8A4 6. ↓SLUMX*→↑MXS0/9B2 7. ↓SLUMX*→↑MXEN→↓MXEN*/9A2	Content of utility gates enter MX multiplexer.
B. (1100) (1200) (1300)	BP3	AND J,K 1. ↓J00B + ↓K00B→↑JK00*/14B1 THRU ↓J11B + ↓K11B→↑JK11*/16A1 2. (↑JK00*) (↑ANDJK)→↑U00/14B2 THRU (↑JK11*) (↑ANDJK)→↓U11/16B2	Content of J and K are logically ANDed.
C. (1100) (1300)	BP3	ADDER→J 1. (↑OP100) (↑I05B) (↑I1110*)→ ↓OP16*/9A2 2. ↓OP16*→↑PEJ6/9A3 3. (↑PEJ6) (↑BP6)→↓PEJ*→↑PEJ 4. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4	Content of adder replaces content of J-register.
D. (1200) (1300)	BP5MD	ADDER→K 1. (↑OP100) (↑I1110*) (↑I04B)→ ↓OP1PK*/9A2 2. ↓OP1PK*→↑PEK5/9A3 3. (↑PEK5) (↑BP5MD)→↓PEK*/9A3 4. ↓PEK*→↑PEK/9A4 5. (↑PEK) (↑REGCLK)→CPK*/9A3	Content of adder replaces content of K-register.

4.4.2.1.4 Load/Exchange Subgroup. For this subgroup Figure 4-33 is the block diagram, Figure 4-34 is the timing diagram, Table 4-31 is the event summary of all fundamental operations, and Table 4-32 is the simplified logic in terms of fundamental operations.

Instructions of this subgroup perform data transactions between the four accumulator registers, J, K, R, and S. Depending on whether the operation specifies a load or an exchange operation, the transaction can result in new content only for one of the registers called for in the operation, or both will contain new values after the execution of the instruction. Principally, when the operation called for is a load, the content of one register replaces the content of the other register. For example, the instruction LRFJ (Tables 4-23 and 4-24) results in placing the content of the J register into the R register. The exchange operation is an extension of the load operation; it consists of two load operations. Instruction EXJKRS specifies a double exchange; two single exchanges are performed one after the other.

It should be noted that loading from main accumulators, J and K into their respective lower accumulators, R and S is done directly by generating the low-level enabling signals PER*, PES* and clock signals CPR* and CPS*. However, the upper accumulators have no direct access to the inputs of the main accumulators. In this case, the loading transaction is accomplished via the MX multiplexer and the adder, as required.

All processing paths for this instruction group derive from decoder signal OP100, which when combined with appropriate bits of the instruction, gives rise to the various control signals, OP1PER, OP1PES and others. These control signals generate the fundamental functions for this instruction subgroup, as can be seen in the listing of Table 4-31.

Table 4-32 lists all instructions for the load-exchange subgroup.

LSFK (1201) - The content of the K register (4, Figure 4-33) replaces the content of the S register (6). The content of the K register remains unchanged.

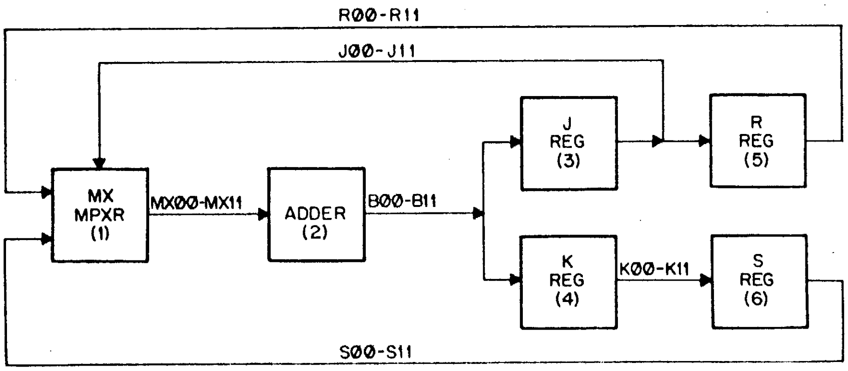


Figure 4-33. Load/Exchange Subgroup, Block Diagram

Table 4-31. Load/Exchange Subgroup, Event Summary

Mnemonic Code	Octal Code	Ref.	Period	Event
LRFJ	1101	A.	BP6MD	J→R
LSFK	1201	B.	BP5MD	K→S
LJFR	1102	C.	BP6MD	R→MX
		F.	BP6MD	ADDER→J
LKFS	1202	D.	BP5MD	S→MX
		G.	BP5MD	ADDER→K
LRSFJK	1301	A.	BP6MD	J→R
		B.	BP5MD	K→S
LJKFRS	1302	D.	BP5MD	S→MX
		G.	BP5MD	ADDER→K
		C.	BP6MD	R→MX
		F.	BP6MD	ADDER→J
LKFJ	1204	E.	BP5	J→MX
		G.	BP5MD	ADDER→K
EXJR	1103	B.	BP6MD	R→MX
		A.	BP6MD	J→R
		F.	BP6	ADDER→J
EXKS	1203	D.	BP5MD	S→MX
		B.	BP5MD	K→S
		G.	BP5MD	ADDER→K
EXJRKS	1303	D.	BP5MD	S→MX
		B.	BP5MD	K→S
		G.	BP5MD	ADDER→K
		C.	BP6MD	R→MX
		A.	BP6MD	J→R
		F.	BP6	ADDER→J

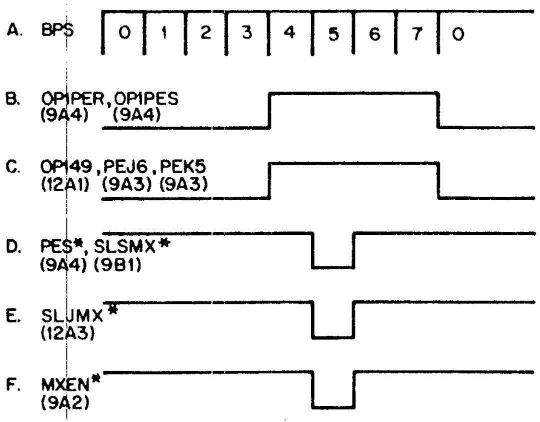


Figure 4-34. Load/Exchange Subgroup, Timing Diagram

Table 4-32. Load/Exchange Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1101) (1103) (1301) (1303)	BP6MD	J→R 1. (↑OP100) (↑I05B) (↑I11N)→ ↓OP1LDR→↑OP1PER/9A4 2. (↑OP1PER) (↑BP6MD)→↓PER*/9A4 = ENABLE→RR 3. ↓PER*→↑PER/9B1 4. (↑PER) (↑REGCLK)→↓CPR*/9B2 = LOAD→RR	The content of the J-register replaces the content of the R-register.
B. (1201) (1203) (1301) (1303)	BP5MD	K→S 1. (↑OP100) (↑I4SKP) (↑I11B)→ ↓OP1LRS*→↑OP1PES/9A4 2. (↑OP1PES) (↑BP5MD)→↓PES*/9A4 = ENABLE→SR 3. ↓PES*→↑PES/9B1 4. (↑PES) (↑REGCLK)→↓CPS*/9B2 = LOAD→SR	The content of the K-register replaces the content of the S-register.
C. (1302) (1303) (1102) (1103)	BP6MD	R→MX→ADDER 1. (↑OP100) (↑I05B) (↑I10B)→↓R00*/8B1 2. ↓R00*→↑SELR00*8B1 3. (↑SELR00) (↑BP6MD)→↓SLRMX*/8A1 4. ↓SLRMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 5. ↓SLRMX*→↑MSX1, ↑MXS2/9A1 = LOAD→MX (R)	The content of the R-register enters the MX multiplexer.
D. (1302) (1303) (1202) (1203)	BP5MD	S→MX→ADDER 1. (↑OP100) (↑I10B) (↑I04B)→↓S00*/8B1 2. ↓S00*→↑SELS/8B1 3. (↑SELS) (↑BP5MD)→↓SLSMX*/8B2 4. ↓SLSMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 5. ↓SLSMX*→↑MXS0, MXS1/9B1 = LOAD→MX (S)	The content of the S-register enters the MX multiplexer.
E. (1204)	BP5	J→MX→ADDER 1. (↑OP100) (↑I04B) (↑I09B)→ ↓OP149/12A2 2. ↓OP149→↑SELJS/12A3 3. (↑SELJS) (↑BP5)→↓SLJMX*/12A3 4. ↓SLJMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX (J) 5. ↓SLJMX*→↑MXS2/9A2	The content of the J-register enters the MX multiplexer.

Table 4-32. Load/Exchange Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
F. (1103) (1303)	BP6	ADDER→J 1. (↑OP100) (↑I05B) (↑I1110*)→ ↓OP16*/9A2 2. ↓OP16*→↑PEJ6/9A3 (↑PEJ6) (↑BP6)→↓PEJ*/9A3 = ENABLE→JR 3. ↓PEJ*→↑PEJ/9A4 4. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 LOAD→JR	The content of the adder replaces the content of the J register.
G. (1203) (1303)	BP5MD	ADDER→K 1. (↑OP100) (↑I1110*) (↑I04B)→ ↓OP1PK*/9A2 2. ↓OP1PK*→↑PEK5/9A3 3. (↑PEK5) (↑BP5MD)→↓PEK*/9A3 4. ↓PEK*→↑PEK/9A4/14A3 5. (↑PEK) (↑REGCLK)→↓CPK*/9A4	The content of the adder replaces the content of the K register.

LJFR (1102) - The content of the R register (5) is transferred via the MX multiplexer (1) and adder (2) to the J register (3) and replaces the latter's content. The content of the R register (5) remains unchanged.

LKFS (1202) - The content of the S register (6) is transferred via the MX multiplexer (1) and adder (2) to the K register (4) and replaces the latter's content. The content of the S register (6) remains unchanged.

LRSFJK (1301) - The contents of the J (3) and K (4) registers replace the content of the R (5) and S (6) registers, respectively. The contents of the J and K registers remain unchanged.

LJKFRS (1302) - The contents of the R (5) and S (6) registers are transferred via the MX multiplexer (1) and adder (2) to the J (3) and K (4) registers, respectively, and replace their contents. The contents of the R and S registers remain unchanged.

LKFJ (1204) - The contents of the J register (4) is transferred via the MX multiplexer (1) and adder (2) to the K register (3) and replaces the latter's content. The content of the J register remains unchanged.

EXJR (1103) - First, the content of the R register (5) is transferred via the MX multiplexer (1) to the adder (2). Next, the content of the J register (3) replaces the content of the R-register (5). Finally, the content of the adder (2) replaces the content of the J register (3).

LRFJ (1101) - The content of the J register (3) replaces the content of the R register (5). The content of the J register (3) remains unchanged.

EXKS (1203) - This instruction performs the same operation as EXJR on registers S (6) and K (4).

EXJKRS (1303) - This instruction combines the operations of EXJR and EXKS.

EXJK (1373) - This instruction is an exchange of the content of the J and K registers, but is not one of this subgroup because the contents of these two registers are exchanged in the shift and rotate subgroup.

4.4.2.1.5 Add/Subtract Subgroup. For this subgroup, Figure 4-35 is the block diagram, Figure 4-36 is the timing diagram, Table 4-33 is the instruction event summary for the fundamental operations and the instruction codes that invoke them, and Table 4-34 is the description of the 14 fundamental operations.

Instructions from this subgroup perform arithmetic operations on the contents of the four accumulators. For a given instruction only two of the registers can be party to the operation. The operation can be any of three types; addition, subtraction, or negation. Addition is carried out by combining two positive quantities through the MX and TX multiplexers. Because the MX multiplexer is capable of contributing only a positive quantity, subtraction and negation are performed through control of the TS multiplexer add-subtract logic.

The add-subtract operations can be considered as one-pass or two-pass operations, each occurring in its own time period, BP5 for the one-pass and BP6 for the two-pass. In a one-pass operation, the result of the summation, which takes place in the adder, is obtained by providing data paths that switch data through the TS and MX multiplexers only one time to obtain the desired result. Although other events occur during the second pass, or period BP6, they have no relevance as far as the operation is concerned. The irrelevant operations are listed in italics in Table 4-33. On the other hand, a two-pass operation produces the desired results providing a summation, performed in the adder during the first pass, and then taking the complement of the result through the add-subtract logic via the TS multiplexer. The two-pass operation has a relevance when a negative quantity cannot be produced by switching through the TS multiplexer and the add-subtract logic during period BP5. For example, the quantity -R cannot be produced through the TS multiplexer because the TS multiplexer does not accept -R (or -S) register data. The two-pass operation is also used in producing the negative of a sum such as -(A + B) or -(A - B).

In a one-pass operation, instruction SJK J (1121), for example, the contents of the J register is switched through the MX multiplexer, and the contents of the K register is switched through the TS multiplexer. Decoder control logic associated with this instruction prepared the add-subtract logic for a subtract operation through the TS multiplexer. When the adder output is formed, the J register is enabled and clocked, and the summation result replaces the contents of the J register. Although events L and N, Table 4-33, duplicate events D and H, they are meaningless becuase the operation has produced the required results during the first pass.

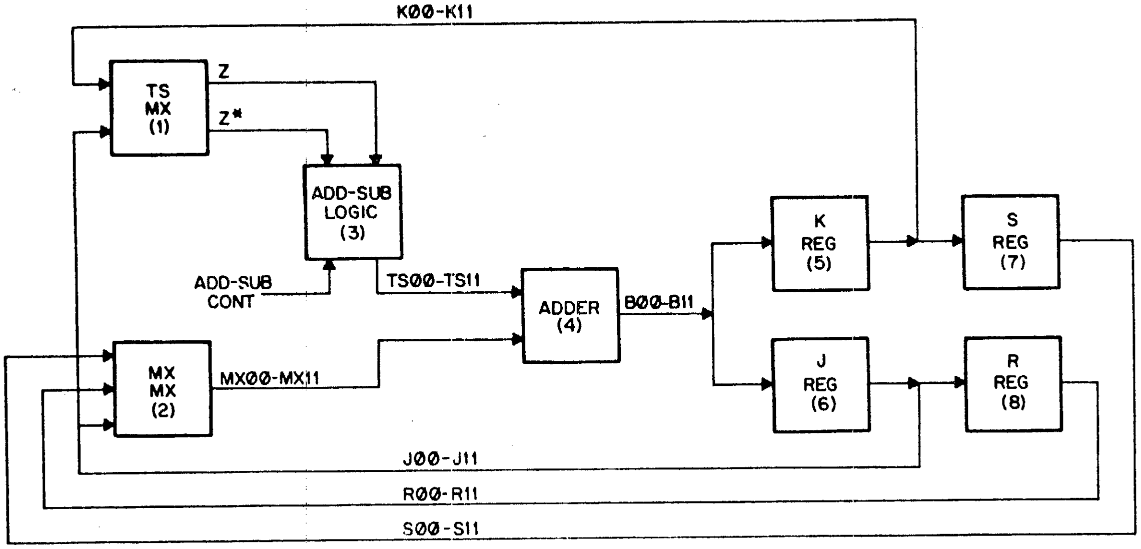


Figure 4-35. Add/Subtracr Subgroup, Block Diagram

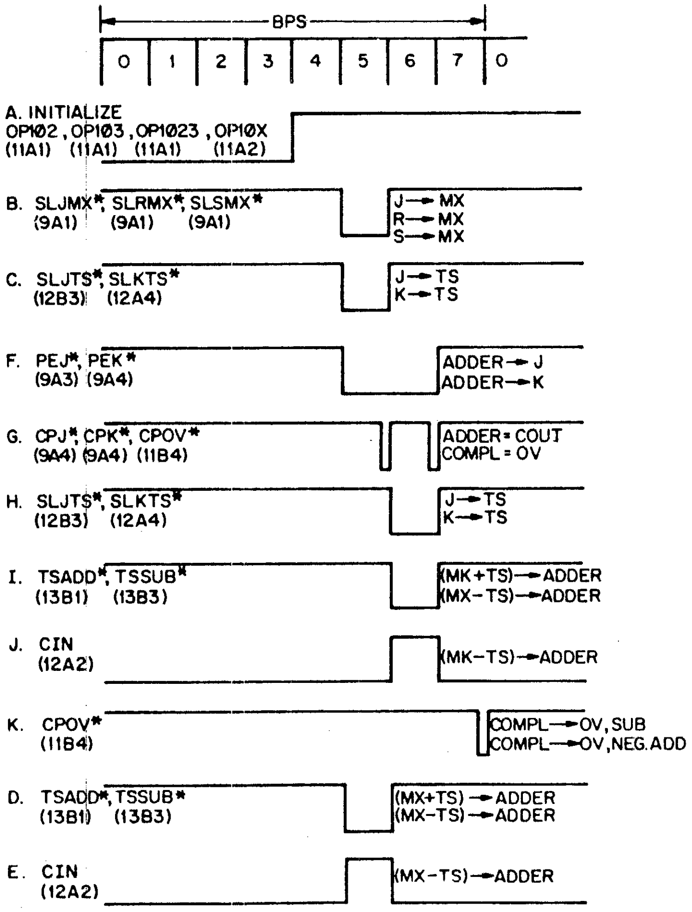


Figure 4-36. Add/Subtract Subgroup, Timing Diagram

Table 4-33. Add/Subtract Subgroup, Event Summary

	Octal				
Mnemonic	**Code**	**Ref.**	**Period**	**Pass**	**Event**
AJK J	1120	A.	BP5	1	J→MX→ADDER
(J+K)		E.	BP5	1	K→TS→±LOGIC
		H.	BP5B	1	(MX + TS)→ADDER
		I.	BP56	1,2	ADDER→J
		K.	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	*BP6B*	*2*	*J→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
SJK J	1121	A.	BP5	1	J→MX→ADDER
(J-K)		E.	BP5	1	K→TS→±LOGIC
		G.	BP5	1	(MX - TS)→ADDER
		I.	BP56	1,2	ADDER→J
		K.	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L.	*BP6B*	*2*	*J→TS→±LOGIC*
		N.	*BP6B*	*2*	*(MX + TS)→ADDER*
		P.	BP7	3	COMPL→OV, SUBTRACT
ADR J	1122	B.	BP5MD	1	R→MX→ADDER
(R+J)		D.	BP5	1	J→TS→± LOGIC
		H.	BP5B	1	(MX + TS)→ADDER
		I.	BP56	1,2	ADDER→J
		K.	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L.	*BP6B*	*2*	*J→TS→±LOGIC*
		N.	*BP6B*	*2*	*(MX + TS)→ADDER*
SBR J	1123	B.	BP5	1	R→MX→ADDER
(R-J)		D.	BP5	1	J→TS→± LOGIC
		G.	BP5	1	(MX - TS)→ADDER
		I.	BP56	1,2	ADDER→J
		K.	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L.	*BP6B*	*2*	*J→TS→±LOGIC*
		N.	*BP6B*	*2*	*(MX + TS)→LOGIC*
		P.	BP7	3	COMPL→OV, SUBTRACT
ADS J	1124	C.	BP5MD	1	S→MX→ADDER
(S+J)		D.	BP5	1	J→TS→ADDER
		H.	BP5B	1	(MX + TS)→ADDER
		I.	BP56	1,2	ADDER→J
		K.	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L.	*BP6B*	*2*	*J→TS→±LOGIC*
		N.	*BP6B*	*2*	*(MX + TS)→ADDER*
SBS J	1125	C	BP5MD	1	S→MX→ADDER
(S-J)		D	BP5	1	J→TS→±LOGIC
		G	BP5	1	(MX - TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	*BP6B*	*2*	*J→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
		P	BP7	3	COMPL→OV, SUBTRACT

Table 4-33. Add/Subtract Subgroup, Event Summary (Cont'd.)

Mnemonic	Octal Code	Ref.	Period	Pass	Event
NAJK J -(J+K)	1130	A	BP5	1	J→MX→ADDER
		E	BP5	1	K→TS→±LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→±LOGIC
		O	BP6B	2	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSJK J (K-J)	1131	A	BP5	1	J→MX→ADDER
		E	BP5	1	K→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER
NADR J -(R+J)	1132	B	BP5MD	1	R→MX→ADDER
		D	BP5	1	J→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→±LOGIC
		O	BP6B	2	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSBR J (J-R)	1133	B	BP5MD	1	R→MX→ADDER
		D	BP5	1	J→TS→±LOGIC
		G	BP56	1,2	(MX - TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER
NADS J -(S+J)	1134	C	BP5MD	1	S→MX→ADDER
		D	BP5	1	J→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSBS J (J-S)	1135	C	BP5MD	1	S→MX→ADDER
		D	BP5	1	J→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		I	BP56	1,2	ADDER→J
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER

Table 4-33. Add/Subtract Subgroup, Event Summary (Cont'd.)

Mnemonic	Octal Code	Ref.	Period	Pass	Event
AJK K	1220	A	BP5	1	J→MX→ADDER
(J+K)		F	BP5	1	K→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	*BP6B*	*2*	*K→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
SJK K	1221	A	BP5	1	J→MX→ADDER
(J-K)		F	BP5	1	K→TS ± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	*BP6B*	*2*	*K→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX+TS)→ADDER*
		P	BP7	3	COMPL→OV, SUBTRACT
ADR K	1222	B	BP5MD	1	R→MX→ADDER
(R+K)		F	BP5	1	K→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	*BP6B*	*2*	*K→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
SBR K	1223	B	BP5	1	R→MX→ADDER
(R-K)		F	BP5	1	K→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	*BP6B*	*2*	*K→TS→± LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
		P	BP7	3	COMPL→OV, SUBTRACT
ADS K	1224	C	BP5MD	1	S→MX→ADDER
(S+K)		F	BP5	1	K→TS→ADDER
		H	BP5B	1	(MX + TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	*BP6B*	*2*	*K→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
SBS K	1225	C	BP5MD	1	S→MX→ADDER
(S-K)		F	BP5	1	K→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	*BP6B*	*2*	*K→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
		P	BP7	3	COMPL→OV, SUBTRACT

Table 4-33. Add/Subtract Subgroup, Event Summary (Cont'd.)

Mnemonic	Octal Code	Ref.	Period	Pass	Event
NAJK K -(J+K)	1230	A	BP5	1	J→MX→ADDER
		F	BP5	1	K→TS→±LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	BP6B	2	K→TS→±LOGIC
		O	BP6B	2	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSJK K (K-J)	1231	A	BP5	1	J→MX→ADDER
		F	BP5	1	K→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	BP6B	2	K→TS→±LOGIC
		O	BP6B	2	(MX - TS)→ADDER
NADR K -(R+K)	1232	B	BP5MD	1	R→MX→ADDER
		F	BP5	1	K→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	BP6B	2	K→TS→±LOGIC
		O	BP6B	2	(MX + TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSBR K (K-R)	1233	B	BP5MD	1	R→MX→ADDER
		P	BP5	1	K→TS→±LOGIC
		G	BP56	1,2	(MX - TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	BP6B	2	K→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER
NADS K -(S+K)	1234	C	BP5MD	1	S→MX→ADDER
		F	BP5	1	K→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	BP6B	2	K→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSBS K (K-S)	1235	C	BP5MD	1	S→MX→ADDER
		F	BP5	1	K→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		M	BP6B	2	K→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER

Table 4-33. Add/Subtract Subgroup, Event Summary (Cont'd.)

	Octal				
Mnemonic	Code	Ref.	Period	Pass	Event
AJK JK	1320	A	BP5	1	J→MX→ADDER
(J+K)		E,F	BP5	1	K→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		I	BP56	1,2	ADDER→J
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	*BP6B*	*2*	*J→TS→± LOGIC*
		M	*BP6B*	*2*	*K→TS→± LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
SJK JK	1321	A	BP5	1	J→MX→ADDER
(J-K)		E,F	BP5	1	K→TS→± LOGIC
		G	BP5	1	(MX - TS)→ADDER
		I	BP56	1,2	ADDER→J
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	*BP6B*	*2*	*J→TS→±LOGIC*
		M	*BP6B*	*2*	*K→TS→±LOGIC*
		N	*BP6B*	*2*	*(MX + TS)→ADDER*
NAJK JK	1330	A	BP5	1	J→MX→ADDER
-(J+K)		E,F	BP5	1	K→TS→± LOGIC
		H	BP5B	1	(MX + TS)→ADDER
		I	BP56	1,2	ADDER→J
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2	J→TS→± LOGIC
		M	BP6B	2	K→TS→± LOGIC
		O	BP6B	2	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD
NSJK JK	1331	A	BP5	1	J→MX→ADDER
(K-J)		E.F	BP5	1	K→TS→± LOGIC
		H	BP5	1	(MX + TS)→ADDER
		I	BP56	1,2	ADDER→J
		J	BP56	1,2	ADDER→K
		K	BP56	1,2	COMPL→OV, IF ADDER=COUT
		L	BP6B	2,	J→TS→±LOGIC
		M	BP6B	2,	K→TS→±LOGIC
		O	BP6B	2,	(MX - TS)→ADDER
		Q	BP7	3	COMPL→OV, NEGATE ADD

Table 4-34. Add/Subtract Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP5	J→MX→ADDER	
(1120)		1. ↓OP102* + ↓OP103*→↑OP1023/11A1	Content of J-register is switched through MX multiplexer to adder.
(1121)		2. (↑OP1023) (↑I4I5)→↓OP10X*→↑OP10X/11A2	
(1130)		3. (↑OP10X) (↑I9I10*)→↓SLJ23*→↑SELJ5/12A3	
(1131)		4. (↑SELJ5) (↑BP5)→↓SLJMX*/12A4	
(1220)		5. ↓SLJMX*→↑MXS2/9A2 = JR→MX→ADDER	
(1221)		6. ↓SLJMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX	
(1230)			
(1231)			
(1320)			
(1321)			
(1330)			
(1331)			
B.	BP5MD	R→MX→ADDER	
(1122)		1. ↓OP102* + ↓OP103*→↑OP1023/11A1	Content of R-register is switched through MX multiplexer to adder.
(1123)		2. (↑OP1023) (↑I4I5)→↓OP10X*→↑OP10X/11A2	
(1132)		3. (↑OP10X) (↑I10B) (↑I09*)→↓R23*/8A1	
(1133)		4. ↓R23*→↑SELR23/8A1	
(1222)		5. (↑SELR23) (↑BP5MD)→↓SLRMX*/8A2	
(1223)		6. ↓SLRMX*→↑MXS1, ↑MXS2/9A1 = RR→MX→ADDER	
(1232)			
(1233)		7. ↓SLRMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX	
C.	BP5MD	S→MX→ADDER	
(1124)		1. ↓OP102* + ↓OP103*→↑OP1023/11A1	Content of S-register is switched through MX multiplexer to adder.
(1125)		2. (↑OP1023) (↑I4I5)→↓OP10X*→↑OP10X/11A2	
(1134)		3. (↑OP10X) (↑I10*) (↑I09B)→↓S23*/8B1	
(1135)		4. ↓S23*→SELS/8B1	
(1224)		5. (↑SELS) (↑BP5MD)→↓SLSMX*/8B2	
(1225)		6. ↓SLSMX*→↑MXS0, ↑MXS1/9B1 = SR→MX→ADDER	
(1234)			
(1235)		7. ↓SLSMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX	
D.	BP5B	J→TS→±LOGIC	
(1122)		1. ↓I09* + ↓I10*→↑X0X1*/11A4	Content of the J-register is switched through the TS multiplexer to the add-subtract logic.
(1123)		2. ↓OP102* + ↓OP103*→↑OP1023/11A1	
(1124)		3. (↑OP1023) (↑X0X1*) (↑I05X)→↓JARTH*/12B2	
(1125)		4. ↓JARTH*→↑JARTH/12B3	
(1132)		5. (↑JARTH) (↑BP5B)→↓SLJTS*/12B3	
(1133)		6. ↓SLJTS*→↑TSS0/13B4, ↑TSS1 = J→TS ± LOGIC	
(1134)			
(1135)			
E.	BP5	K→TS→±LOGIC	
(1120)		1. ↓OP102* + ↓OP103*→↑OP1023/11A1	Content of the K-register is switched through the TS multiplexer to the add-subtract logic.
(1121)		2. (↑OP1023) (↑I05B)→↓OP235*/9A3	
(1130)		3. ↓OP235*→↑OP235/12B3	
(1131)		4. (↑OP235) (↑I9I10*)→↓OP23K*→↑OP1K/12A4	
(1321)		5. (↑OP1K) (↑BP5)→↓SLKTS*→↑TSS1/13B4 = K→TS→±LOGIC	
(1331)			

Table 4-34. Add/Subtract Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
F. (1220) (1221) (1222) (1223) (1224) (1225) (1230) (1231) (1232) (1233) (1234) (1235) (1320) (1321) (1330) (1331)	BP5	K→TS→±LOGIC 1. ↓OP102* + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑I04B)→↓OP23I*/9A3 3. ↓OP23I*→↑OP1K/12A4 4. (↑OP1K) (↑BP5)→↓SLKTS*/12A4 5. ↓SLKTS*→↑TSS1/13B4 = K→TS→±LOGIC	
G. (1121) (1123) (1125) (1131) (1133) (1135) (1221) (1223) (1225) (1231) (1233) (1235) (1321) (1331)	BP5B	(MX-TS)→ADDER 1. ↓OP102* + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑I4I5)→↓OP10X*→↑OP10X/11A2 3. (↑OP10X) (↑I11N)→↓OP1SB*→↑SUB5/13B3 4. (↑SUB5) (↑BP5B)→↓TSSUB*/13B2 5. (↓TSSUB*) (↑DCIN*) (↑OP2*)→↓CIN2*→ ↑CIN/12A2 = MX - TS→ADDER	Difference of MX and TS multiplexer is the adder output.
H. (1120) (1122) (1124) (1130) (1132) (1134) (1220) (1222) (1224) (1230) (1232) (1234) (1320) (1330)	BP5B	(MX+TS)→ADDER 1. ↓OP102* + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑BP5B) (↑I4I5) (↑I11B*)→ ↓TSADD*/13B2 = MX + TS→ADDER	Sum of MX and TS multiplexer is the adder output.

Table 4-34. Add/Subtract Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
I. (1120) (1121) (1122) (1123) (1124) (1125) (1130) (1131) (1132) (1133) (1134) (1135) (1320) (1321) (1330) (1331)	BP56	ADDER→J 1. ↓OP102* + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑I05B)→↓OP235*/9A3 3. ↓OP235*→↑PEJ56/9A3 4. (↑PEJ56) (↑BP56)→↓PEJ* = ENABLE→JR 5. ↓PEJ*→↑PEJ/9A4 6. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = LOAD→JR	Content of adder replaces content of J-register.
J. (1220) (1221) (1222) (1223) (1224) (1225) (1230) (1231) (1232) (1233) (1234) (1235) (1320) (1321) (1330) (1331)	BP56	ADDER→K 1. ↓OP102* + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑I04B)→↓OP231*/9A3 3. ↓OP23I*→↑PEK5+6/9A3 4. (↓PEK5+6) (↑BP56)→↓PEK*/9A4 = ENABLE→KR 5. ↓PEK*→↑PEK/9A4 6. (↑PEK) (↑REGCLK)→↓CPK*/9A4 = LOAD→KR	Content of adder replaces content of K-register.
K. (1120) (1121) (1122) (1123) (1124) (1125) (1130) (1131) (1132) (1133) (1134) (1135) (1220) (1221)	BP56	COMPL→OV, IF ADDER = COUT 1. ↓BP5* + ↓BP6*→↑BP56/11A2 2. (↑BP56) (↑COUT) (↑OP1023)→↓COUTV*→↑CPOV/11B4 3. (↑CPOV) (↑REGCLK)→↓CKOV /11B4 4. ↓CKOV /11B4 = SET→OV (↑OV, ↓OV*/11B4)	Overflow register is complemented if a carry is produced by the adder.

Table 4-34. Add/Subtract Subgroup, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
(1222)			
(1223)			
(1224)			
(1225)			
(1230)			
(1231)			
(1232)			
(1233)			
(1234)			
(1235)			
(1320)			
(1321)			
(1330)			
(1331)			
L.			
(1120)	BP6B	J→TS→±LOGIC	
(1121)		1. ↓OP102* + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑I05B)→↓OP235*→↑OP235/12B3 3. (↑OP235) (↑BP6B)→↓SLJTS*/12B3 4. ↓SLJTS*→↑TSS0/13B4, ↑TSS1 = J→TS→±LOGIC	Content of the J-register is switched through the TS multiplexer to the add-subtract logic.
(1122)			
(1123)			
(1124)			
(1125)			
(1130)			
(1131)			
(1132)			
(1133)			
(1134)			
(1135)			
(1320)			
(1321)			
(1330)			
(1331)			
M.			
(1220)	BP6B	K→TS→±LOGIC	
(1221)		1. ↓OP102 + ↓OP103*→↑OP1023/11A1 2. (↑OP1023) (↑I04B)→↓OP23I*/9A3 3. ↓OP23I*→↑KTS6/12A4 4. (↑KTS6) (↑BP6B)→↓SLKTS*→↑TSS1/13B4 = K→TS→±LOGIC	Content of the K-register is switched through the TS multiplexer to the add-subtract logic.
(1222)			
(1223)			
(1224)			
(1225)			
(1230)			
(1231)			
(1232)			
(1233)			
(1234)			
(1235)			
(1320)			
(1330)			
(1321)			
(1331)			

Table 4-34. Add/Subtract Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
N. (1120) (1121) (1122) (1123) (1124) (1125) (1220) (1221) (1222) (1223) (1224) (1225) (1320) (1321)	BP6B	(MX+TS)→ADDER 1. (↑OP102) (↑I4I5)→↓OP245*→↑TAD6/13B1 2. (↑TAD6) (↑BP6B)→↓TSADD*/13B2 = MX+TS→ADDER	Sum of MX and TS multiplexer is the adder output.
O. (1130) (1131) (1132) (1133) (1134) (1135) (1230) (1231) (1232) (1233) (1234) (1235) (1330) (1331)	BP6B	(MX-TS)→ADDER 1. (↑OP103) (↑I4I5)→↓OP205*/13B2 2. ↓OP205*→↑SUB6/13B3 3. (↑SUB6) (↑BP6B)→↓TSSUB*→↑TSSUB/13B3 4. (↑TSSUB) (↑DCIN*) (↑OP2*)→↓CIN2*→↑CIN/12A2 = MX-TS→ADDER	Difference of MX and TS multiplexer is the adder output.
P. (1121) (1123) (1125) (1221) (1223) (1225)	BP7	COMPL→OV, SUBTRACT 1. ↓OP102*→↑OP102/11A1 2. (↑OP102) (↑I11B)→↓OP211*→↑OP1SBL/12B4 3. (↑OP1SBL) (↑BP7)→↓AROV*/12B4 4. ↓AROV*→↑CPOV/11B3 5. (↑CPOV) (↑REGCLK)→↓CKOV /11B4 6. ↓CKOV /11B4 = SET (↑OV→↓OV* + ↓OV→↑OV*/11B4) = COMPL→OV	Overflow register is complemented for add operation.
Q. (1130) (1132) (1134) (1230) (1232) (1234) (1330)	BP7	COMPL→OV, NEGATE ADD 1. ↓OP103*→↑OP103/11A1 2. (↑OP103) (↑11B*)→↓OP1311*→↑OP1SBL/12B4 3. (↑OP1SBL) (↑BP7)→↓AROV*/12B4 4. ↓AROV*→↑CPOV/11B3 5. (↑CPOV) (↑REGCLK)→↓CKOV /11B4 6. ↓CKOV /11B4 = SET (↑OV→↓OV* + ↓OV→↑OV*/11B4) = COMPL→OV	Overflow register is complemented for subtract operation.

Study of the requirements of instruction NSBR J (1133) in Table 4-33 shows that the difference between contents of (J-R) cannot be formed in a single pass because the quantity R has a negative sign. Figure 4-35, the block diagram reveals that the contents of the R register can only be sent to the MX multiplexer, which is not capable of treating a negative quantity. Therefore, this instruction must be carried out in a two-pass operation. The quantity R-J is produced during the first pass by switching the content of the J register through the TS multiplexer and subtracting it from the content of the R register, which is switched through the MX multiplexer. The resulting content of the adder, is loaded into the J register and then the content of the J register is switched through the T_s multiplexer during the second pass to produce a negation. During the second pass, the MX multiplexer is not enabled, and hence, can only contribute zeros to the adder. The result of this operation produces -(R - J), which is equivalent to J - R, as the instruction requires.

A two-pass operation is required for parenthetical expressions of the type -(A + B) and -(A - B) = (B - A). For these summations the TS multiplexer output is selected for addition during the first pass and for negation during the second pass. Whenever subtraction or negation is ordered, 2's complement addition between the two quantities occurs. The 2's complement or negative quantity is obtained by summing the complement of the data switched through the TS multiplexer with the data switched through the MX multiplexer and then incrementing the adder with a carry input (high-level CIN signal). For any of the add/subtract operations, overflow is possible. If addition produces an overflow, the operation yields an improper result and if subtraction does not produce an overflow, the operation also yields an improper result. For this reason, operation K tests for an overflow, and in the case of subtraction, and negation, operations P and Q (Table 4-34) cancel it if the operation yields a proper result.

All processing control for this instruction subgroup is derived from operate decoder signals OP102* and OP103*, which when combined with appropriate bits of the add/subtract subgroup instruction, gives rise to various other control signals. These control signals generate the fundamental operations listed in Table 4-34. Table 4-33, the event summary, lists all the operations specified in Table 4-34 which are used in the processing of the instruction. The instructions are described below.

AJK J (1120) - The contents of the J and K registers are added via the MX and TS multiplexers, respectively, during the first pass and the result, J + K replaces the contents of the J register. Operations L and N are irrelevant.

SJK J (1121) - The contents of the J and K registers are added after first complementing and incrementing the contents of the K register which is switched through the TS multiplexer resulting in a subtraction, J - K. The result replaces the content of the J register. Operations L and N are irrelevant.

ADR J (1122) - The contents of the R and J registers are added via the MX and TS multiplexers, respectively, and the result, R + J, replaces the contents of the J register. Operations L and N are irrelevant.

SBR J (1123) - The contents of the R and J registers are added after first complementing and incrementing the content of the J register which is switched through the TS multiplexer resulting in a subtraction, R - J. Operations L and N are irrelevant.

ADS J (1124) - The contents of the S and J registers are added via the MX and TS multiplexers, respectively, during the first pass and the results, S + J, replace the contents of the J register. Operations L and N are irrelevant.

SBS J (1125) - The contents of the S and J registers are added after first complementing and incrementing the content of the J register which is switched through the TS multiplexer resulting in a subtraction, S - J. Operations L and N are irrelevant.

NAJK J (1130) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction AJK J (1120). During period BP6B, the second pass takes place. The second-pass events L and O switch the content of the J register, J + *, through the TS multiplexer so that its output can be complemented and incremented. The result, -(J + K) replaces the content of the J register, which remains enabled. Pass 3 complements the overflow register.

NSJK J (1131) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction SJK J (1121). During period BP6B, the second pass takes place. The second-pass events L and O switch the contents of the J register J-K, through the TS multiplexer so that its output can be complemented and incremented. The result, K - J, replaces the content of the J register, which remains enabled. Pass 3 does not occur.

NADR J (1132) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction ADR J (1122). During period BP6B, the second-pass takes place. The second-pass events L and O switch the contents of the J register, R + J, through the TS multiplexer so that its output can be complemented and incremented. The results -(R + J), replace the contents of the J register, which remains enabled. Pass 3 complements the overflow register.

NSBR J (1133) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction SBR J (1123). During period BP6B, the second pass takes place. The second-pass events L and O switch the contents of the J register, R - J, through the TS multiplexer so that its output can be complemented and incremented. The results, J - R, replace the contents of the J register, which remains enabled. Pass 3 does not occur.

NADS J (1134) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction ADS J (1124). During period BP6B, the second-pass takes place. The second-pass events L and O switch the contents of the J register, S + J, through the TS multiplexer so that its output can be complemented and incremented. The results, -(S + J), replace the contents of the J register, which remains enabled. Pass 3 complements the overflow register.

NSBS J (1135) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction SBS J (1125). During period BP6B, the second-pass takes place. The second-pass events L and O switch the contents of the J register, S - J, through the TS multiplexer so that its output can be complemented and incremented. The results, J - S, replace the contents of the J register, which remains enabled. Pass 3 does not occur.

AJK K (1220) - The contents of the J and K registers are added via the MX and TS multiplexers, respectively, during the first pass and the result, J + K, replace the contents of the K register. Operations M and N are irrelevant.

SJK K (1221) - The contents of the J and K registers are added after first complementing and incrementing the contents of the K register which is switched through the TS multiplexer resulting in a subtraction, J - K. The result replaces the content of the K register. Operations M and N are irrelevant.

ADR K (1222) - The contents of the R and J registers are added via the MX and TS multiplexers, respectively and the result, R + K, replaces the contents of the K register. Operations M and N are irrelevant.

SBR K (1223) - The content of the R and K registers are added after first complementing and incrementing the content of the K register which is switched through the TS multiplexer resulting in a subtraction, R - K. Operations M and N are irrelevant.

ADS K (1224) - The contents of the S and K registers are added via the MX and TS multiplexers, respectively, during the first pass and the results, S + K, replace the contents of the J register. Operations M and N are irrelevant.

SBS K (1225) - The contents of the S and K registers are added after first complementing and incrementing the content of the K register which is switched through the TS multiplexer resulting in a subtraction, S - K. Operations M and N are irrelevant.

NAJK K (1230) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction AJK K (1220). During period BP6B, the second pass takes place. The second-pass events M and O switch the content of the K register, J + K, throug the TS multiplexer so that its output can be complemented and incremented. The result, -(J + K) replaces the content of the K register, which remains enabled. Pass 3 complements the overflow register.

NSJK K (1231) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction SJK K (1221). During period BP6B, the second pass takes place. The second-pass events, M and O, switch the contents of the K register J - K, through the TS multiplexer so that its output can be complemented and incremented. The result, K - J, replaces the content of the K register, which remains enabled. Pass 3 does not occur.

NADR K (1232) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction ADR K (1222). During period BP6B, the second-pass takes place. The second-pass events M and O switch the contents of the K register, R + K, through the TS multiplexer so that its output can be complemented and incremented. The results -(R + K), replace the contents of the K register, which remains enabled. Pass 3 complements the overflow register.

N$BR K (1233) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction SBR K (1223). During period BP6B, the second pass takes place. The second-pass events, M and O, switch the contents of the K register, R - K, through the TS multiplexer so that its output can be complemented and incremented. The results, K - R, replace the contents of the K register, which remains enabled. Pass 3 does not occur.

NADS K (1234) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction ADS K (1224). During period BP6B, the second-pass takes place. The second-pass events, M and O, switch the contents of the K register, S + K, through the TS multiplexer so that its output can be complemented and incremented. The results, -(S + K), replace the contents of the K register, which remains enabled. Pass 3 complements the overflow register.

N$BS K (1235) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction SBS K (1225). During period BP6B, the second pass takes place. The second-pass events, M and O, switch the contents of the K register, S - K, through the TS multiplexer so that its output can be complemented and incremented. The result, K - S, replaces the contents of the K register which remains enabled. Pass 3 does not occur.

AJK JK (1320) - The contents of the J and K registers are added via the MX and TS multiplexers, respectively, during the first pass and the result, J + K, replaces the contents of both the J and K registers. Operations L, M and N are irrelevant.

SJK JK (1321) - The contents of the J and K registers are added after first complementing and incrementing the contents of the K register which is switched through the TS multiplexer resulting in a subtraction, J - K. The result replaces the contents of both the J and K registers. Operations L, M, and N are irrelevant.

NAJK JK (1330) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first-pass events of instruction AJK JK (1320). During period BP6B, the second pass takes place. The second-pass events L, M, and O, switch the contents of both the J and K registers J + K, through the TS multiplexer so that its output can be complemented and incremented. The result, -(J + K), replaces the contents of both the J and K registers which remain enabled. Pass 3 complements the overflow register. That the contents of both J and K registers are switched through the TS multiplexer does not produce an erroneous result because both registers contain the sum J + K. Thus, although the data overlaps, there is no conflict in operations.

NSJK JK (1331) - This is a two-pass operation. During period BP5, the first pass takes place. The first-pass events are identical to the first pass events of instruction SJK JK (1321). During period BP6B, the second pass takes place. The second-pass events L, M, and O switch the contents of both the J and K registers, J - K, through the TS multiplexer so that its output can be complemented and incremented. The result, -(J - K) = (K - J), replaces the contents of both the J and K registers which remain enabled. Pass 3 complements the overflow register. This operation is similar to the SJK JK operation because the contents of both the J and K registers are switched through the TS multiplexer without producing conflicting results.

4.4.2.1.6 Shift/Rotate Subgroup. For this subgroup, Figures 4-37 and 4-38 are the block diagrams, Figure 4-39 is the timing diagram, Table 4-35 is the Event Summary, and Table 4-36 is the list of Fundamental Operations.

Instructions of this subgroup permit data in either the J or K registers, or both, to be shifted or rotated up to 15 times. Shifting is defined as a displacement of data from right to left. When a shift is ordered, successive zeros enter from the right (the J11th or K11th bit) and are displaced one bit at a time toward the left. Data contained in the J00th or K00th bit is shifted out of either the J or K register and is lost. It takes 12 shifts to propagate the zero of the J11th or K11th bit down to the J00th or K00th bit. The rotate instructions are extensions of the shift instructions with the exception that data in the J and K registers is not lost, but simply rotated from right to left. Data can be rotated in only the J or K registers or from the K register into the J register, and out of the J register into the K register. Thus, instead of being lost, data shifted out of the J00th or K00th bit is returned to the J11th or K11th bit, depending on the instruction.

The logic for this subgroup has two functions; it controls the application of the priming bit to the 11th bit of the affected register, and it controls the number of bit shifts that must take place. Figure 4-37 is the block diagram for the rotate and shift control and Figure 4-38 is the block diagram for the counter and clock control which determines the number of bit shifts that occur.

Rotate/Shift Control - The rotate/shift control (Figure 4-37) determines whether the J00th bit will be used to prime the J11th or the K11th bit. When rotate is ordered, operation code OP104 produces signal OP104. When a shift is ordered, operation code OP106 produces signal OP106. These operate code signals determine whether the operation is a shift or a rotate. Rotate operation codes are identified by a 6 as the 3rd octal digit. The heavy lines shown in Figure 4-37 show the bit paths for rotate instruction ROTD JK (1360). The diagram clearly shows that the J00th bit from the J register is used to prime the K11th bit of the K register and the K00th bit of the K register is used to prime the J11th bit of the J register. This operation results in data in the J register being rotated out of the J00th bit into the K11th bit, and data out of the K00th bit being shifted into the J11th bit. Hence, data in these registers is swapped, end-for-end. Some of the operations are not required. Thus, for example, whenever a shift occurs, none of the logic is exercised, and both the J11th and K11th bits are unprimed. Because the unprimed condition is a low-level signal, when the shifts occur, zeros are loaded into the J11th or K11th bit with each shift. Other bit transfer paths can be worked out by referring to the instruction on Figure 4-37, which exercises control.

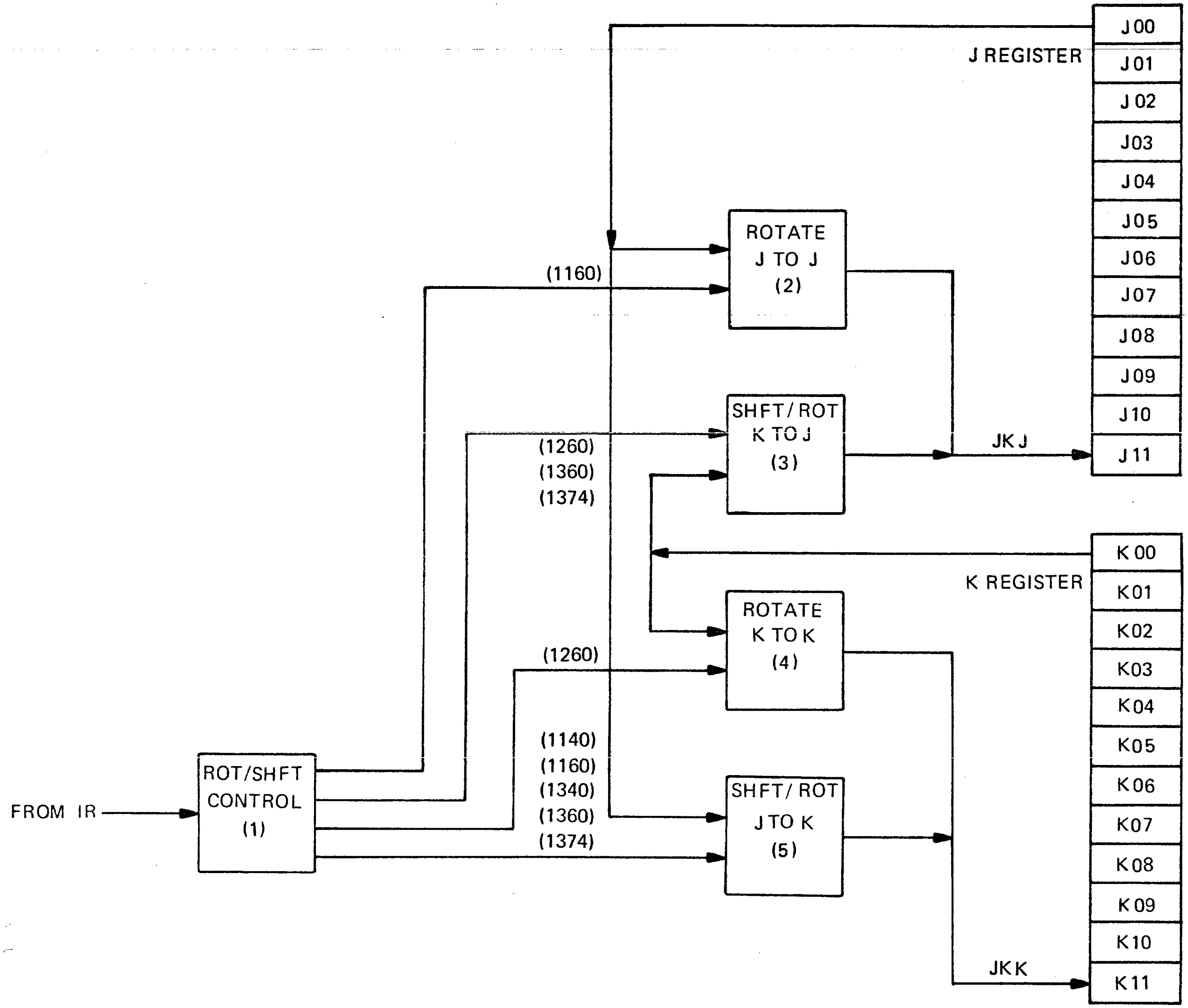


Figure 4-37. Shift/Rotate Subgroup Control, Block Diagram

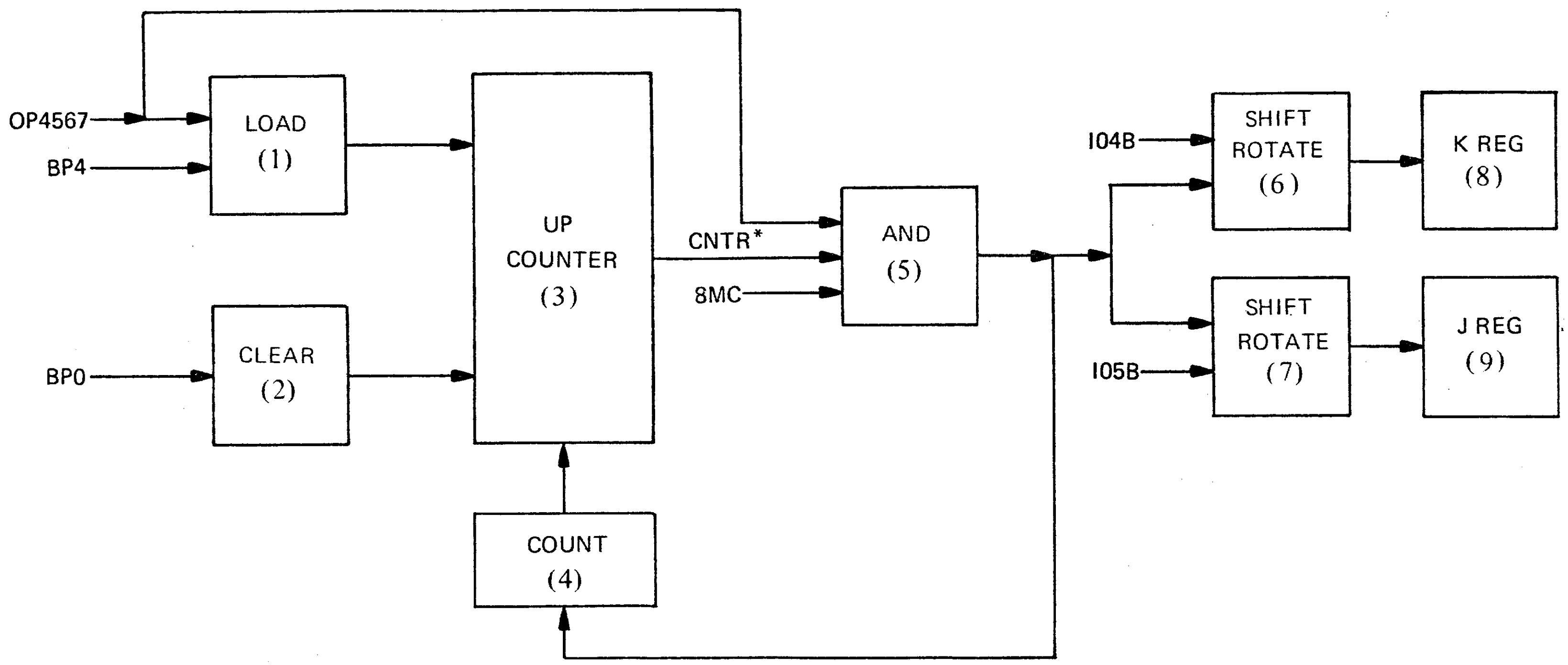


Figure 4-38. Shift/Rotate Subgroup Counter and Clock Control, Block Diagram

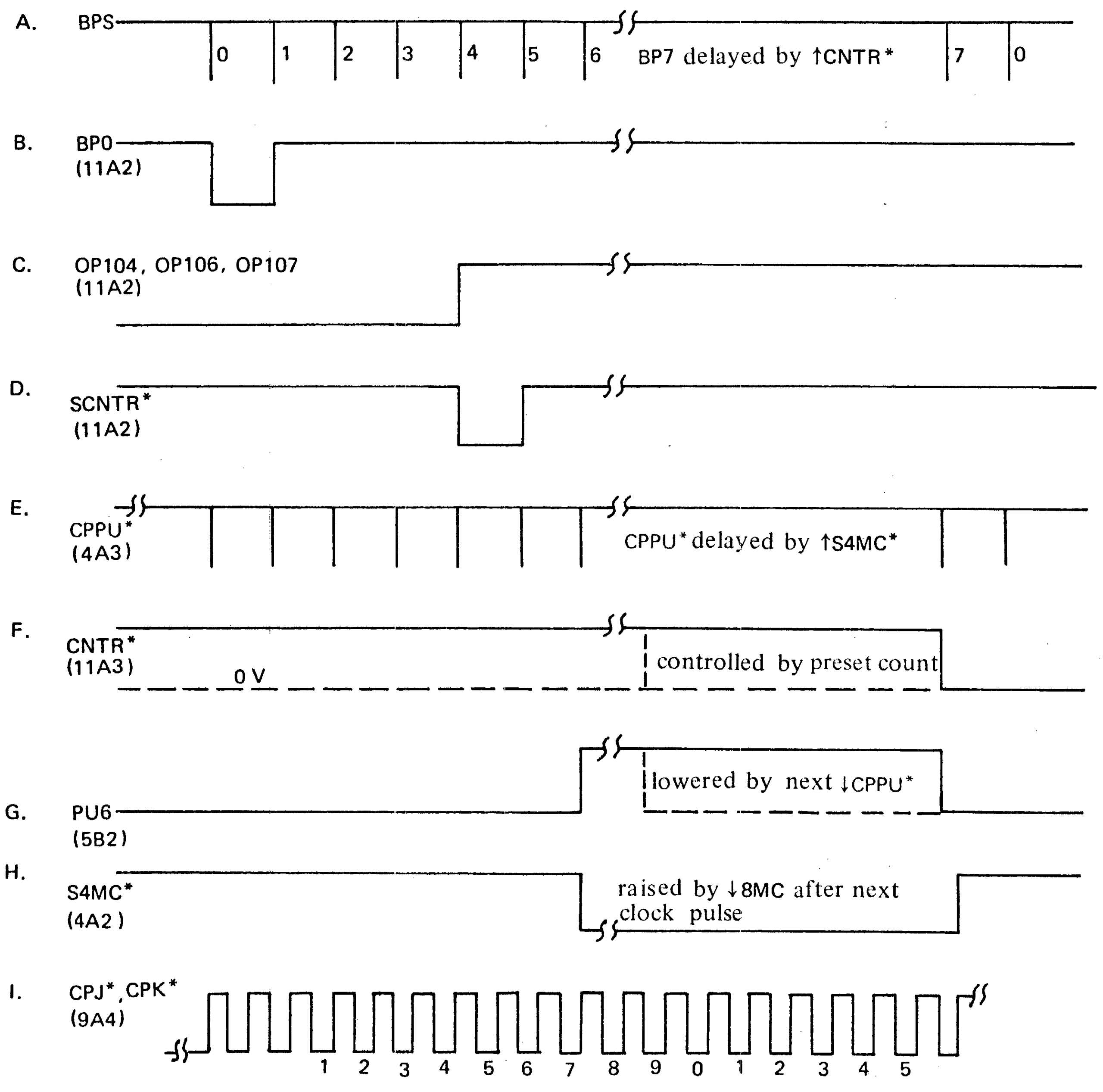


Figure 4-39. Shift/Rotate Subgroup, Timing Diagram

Table 4-35. Shift/Rotate Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
SFTZ J	1140	A	BP0	CLR→UP COUNTER
		B	BP3	INITIALIZE→SHIFT (OP104)
		D	BP3	INITIALIZE→COUNTS
		E	BP3	PRIME→K FOR SHIFT/ROTATE
		I	BP4	LOAD→COUNT
		J	BP56	CLOCK→J REGISTER TO SHIFT
		K	BP56	CLOCK→K REGISTER TO SHIFT
		L	BP56	COUNT→SHIFTS
		M	PU6	DEFER→4 MC CLOCK
ROTD J	1160	A	BP0	CLR→UP COUNTER
		C	BP3	INITIALIZE→ROTATE (OP106)
		D	BP3	INITIALIZE→COUNTS
		E	BP3	PRIME→K FOR ROTATE
		G	BP3	PRIME→J FOR ROTATE TO J
		I	BP4	LOAD→COUNT
		J	BP56	CLOCK→J REGISTER TO ROTATE
		K	BP56	CLOCK→K REGISTER TO ROTATE
		L	BP56	COUNT→ROTATES
		M	PU6	DEFER→4MC CLOCK
SFTZ K	1240	A	BP0	CLR→UP COUNTER
		B	BP3	INITIALIZE→SHIFT (OP104)
		D	BP3	INITIALIZE→COUNTS
		I	BP4	LOAD→COUNT
		J	BP56	CLOCK→J REGISTER TO SHIFT
		K	BP56	CLOCK→K REGISTER TO SHIFT
		L	BP56	COUNT→SHIFTS
		M	PU6	DEFER→4MC CLOCK
ROTD K	1260	A	BP0	CLR→UP COUNTER
		C	BP3	INITIALIZE→ROTATE (OP106)
		D	BP3	INITIALIZE→COUNTS
		F	BP3	PRIME→J FOR ROTATE
		H	BP3	PRIME→K FOR ROTATE TO K
		I	BP4	LOAD→COUNT
		J	BP56	CLOCK→J REGISTER TO ROTATE
		K	BP56	CLOCK→K REGISTER TO ROTATE
		L	BP56	COUNT→SHIFTS
		M	PU6	DEFER→4MC CLOCK

Table 4-35. Shift/Rotate Subgroup, Event Summary (Cont'd.)

Mnemonic	Octal Code	Ref.	Period	Event
SFTZ JK	1340	A	BP0	CLR→UP COUNTER
		B	BP3	INITIALIZE→SHIFT (OP104)
		D	BP3	INITIALIZE→COUNTS
		E	BP3	PRIME→K FOR SHIFT
		I	BP4	LOAD→COUNT
		J	BP56	CLOCK→J REGISTER TO SHIFT
		K	BP56	CLOCK→K REGISTER TO SHIFT
		L	BP56	CLOCK→SHIFTS
		M	PU6	DEFER→4MC CLOCK
ROTD JK, EXJK	1360, 1374	A	BP0	CLR→UP COUNTER
		C	BP3	INITIALIZE→ROTATE (OP106)
		D	BP3	INITIALIZE→COUNTS
		E	BP3	PRIME→K FOR ROTATE
		F	BP3	PRIME→J FOR ROTATE
		I	BP4	LOAD→COUNT
		J	BP56	CLOCK→J REGISTER TO ROTATE
		K	BP56	CLOCK→K REGISTER TO ROTATE
		L	BP56	COUNT→SHIFTS
		M	PU6	DEFER→4MC CLOCK

Table 4-36. Shift/Rotate Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	RPO	CLR→UP COUNTER	
(1140) (1160) (1240) (1260) (1340) (1360) (1374)		BP0*/11A2→CLR UP COUNTER (MR)	The master reset (MR) on the up counter is clocked to clear all counts.
B.	BP3	INITIALIZE→SHIFT (OP104)	
(1140) (1240) (1340)		↓OP104* + ↓OP105*→↑OP145→ ↓OP145*/11A2	Shifts are initialized by signal OP104 from the operate decoder.
C.	BP3	INITIALIZE→ROTATE (OP106)	
(1160) (1260) (1360) (1374)		↓OP106* + ↓OP107*→↑OP167→ ↓OP167*/11A2	Rotates are initialized by signal OP106 or OP107 from the operate decoder.

Table 4-36. Shift/Rotate Subgroup Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
D. (1140) (1160) (1240) (1260) (1340) (1360) (1374)	BP3	INITIALIZE→SHIFT/ROTATE COUNTS ↓OP145 + ↓OP167→↑OP4567/11A2	
E. (1140) (1160) (1340) (1360) (1374)	BP3	PRIME→K, FOR SHIFT/ROTATE (↑OP4567) (↑I05B) (↑J00)→↓ROTKJ*→ ↑JKK/9B4 = ↑K11 ON CLK	If the J00th bit is set, the K11th bit will become a 1 when a shift or rotate is executed.
F. (1260) (1360) (1374)	BP3	PRIME→J FOR ROTATE (↑OP167) (↑K00) (↑I04B)→↓ROTJK*→ ↑JKJ/9B4 = ↑J11 ON CLK	If the K00th bit is set, the J11th bit will become a 1 when a rotate is executed.
G. (1160)	BP3	PRIME→J FOR ROTATE TO J (↑OP167) (↑J00) (↑I04*)→↓ROTJ*→ ↑JKJ/9B4 = ↑J11 ON CLK	Data in J register is rotated !eft when a rotate is executed.
H. (1260)	BP3	PRIME→K FOR ROTATE TO K (↑OP167) (↑K00) (↑I05*)→↓ROTK*→ ↑JKK/9B4 = ↑K11 ON CLK	Data in K register is rotated left when a rotate is executed.
I. (1140) (1160) (1240) (1260) (1340) (1360) (1374)	BP4	LOAD→COUNT (↑OP4567) (↑BP4)→↓SCNTR*/11A2 = ENABLE→COUNTER (PE)	The number of shifts or rotates specified by bits 8, 9, 10, and 11 of the instruction is loaded into the up counter.
J. (1140) (1160) (1240) (1260) (1340) (1360)	BP56	CLOCK→J-REGISTER TO SHIFT/ROTATE 1. (↑OP4567) (↑CNTR) (↑8MC) (↑BP56)→ ↓SHROT*/11A3 2. ↓SHROT*→↑SHROT/11A3 3. (↑SHROT) (↑I05B)→↓CPJ*/9A4 = CLOCK→J	The J-register is shifted until CNTR goes low-level.

Table 4-36. Shift/Rotate Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
K. (1140) (1160) (1240) (1260) (1340) (1360) (1374)	BP56	CLOCK→K-REGISTER TO SHIFT/ROTATE 1. (↑OP4567) (↑CNTR) (↑8MC) (↑BP56)→ ↓SHROT*/11A3 2. ↓SHROT*→↑SHROT/11A3 3. (↑SHROT) (↑104B)→↓CPK*/9A4 = CLOCK→K	The K-register is shifted until CNTR goes low-level.
L. (1140) (1160) (1240) (1260) (1340) (1360) (1374)	BP56	COUNT→NUMBER OF SHIFTS/ROTATE 1. (↑OP4567) (↑CNTR) (↑8MC/2) (↑BP56)→ ↓SHROT*/11A3 2. ↓SHROT*→↑SHROT→↓SHRTD*→ ↑CLKCNT/11A3 = CLOCK→COUNTER	When the up counter reaches the total count CNTR goes low-level.
M. (1140) (1160) (1240) (1340) (1360) (1374)	PU6	DEFER 4 MC CLOCK 1. (↑OP4567) (↑PU6) (↑CNTR*) (↑1415)→ ↓$4MC*/4A2 2. ↓$4MC* = SET↑4MC/DIVIDER	The 4 MC divider is inhibited.

Counter and Clock Control - Figure 4-38 shows the hardware mechanization of the counter and clock control. Because time-dependent signals are also made use of in the block diagram, the timing diagram, Figure 4-39, should be consulted. When shift and rotate instructions are being processed, normal time periods (waveform A, Figure 4-39) occur up until pulse PU6 is produced (waveform G). When pulse PU6 is produced, and if the shift and rotate up counter (3, Figure 4-38) has not detected a total count (waveform F), the 4-megacycle counter in the 4-megacycle countdown logic (paragraph 4.3.3) is inhibited. Inhibiting the 4-MHz countdown logic (waveform H) when a shift or rotate occurs, permits time period BP6 to be extended until the number of shift or rotates called for in bits 8 through 11 of the instruction have been completed. The clock pulses (waveform I) used to count the shifts or rotates are derived from the uninhibited 8-megacycle divider. When the required number of shifts has occurred, the 4-megacycle countdown resumes its normal operation and permits the basic phase (waveforms A and E) to be completed.

Paragraph 4.3.4.7 and 4.3.4.8 should also be reviewed for the operation of the up counter and the main accumulators.

Table 4-15 lists the up-counter preload codes. At the onset of the execution of a shift or rotate instruction, the up counter (3, Figure 4-38) is cleared through the clear logic (2). Next, at time BP4 the up counter is preset to the shift-rotate count through the load logic (1). Signal OP4567 enables presetting of the up counter; this signal is also ANDed with an 8MC signal (5) which fixes the frequency of shifts at the rate of 0.125 microseconds per shifted bit and with signal CNTR* which stays at a high logic level as long as the count is below its preset value. The output of the AND-gate (5) goes to the clock input (4) of the up counter to increment the count. When the count reaches the preset value or the maximum count, signal CNTR* is asserted and becomes low level and disables the AND-gate (5) and stops the count. It also disables the shift-rotate operation which is carried out at an 8MC rate as long as all three inputs to the AND-gate stay high level. Bits I04B and I05B are the J or K register-select bits. The shift/rotate logic (6 and 7) provides the clock pulse for the registers (8 and 9 respectively) to trigger a shift.

For the particulars of manipulation for bits that have to be saved before they are shifted out, see Table 4-36, fundamental operations A through H. Fundamental operation G shows the priming for the J register. The logical value of signal JKJ replaces bit 11 of the J register. From the logic expression it can be seen that the value of JKJ will follow the value of J00 whenever the other two signals are at high logic levels; therefore the content of J00 will replace the content of J11 as soon as the bit displacement is carried out. Fundamental operation H is analogous to G but concerns the K register. Fundamental operation F provides for the replacement of bit 11 of the J register by bit 0 of the K register for a 24-bit register configuration and it also provides for the J register shift. Fundamental operation E provides for the J "carry" to the K register in the 24-bit register configuration and for the shift for the K register. Note that for the shift operation, the corresponding instruction register bits hold the expressions to zeros thus denying the actual assertion of the J00 or K00 bit. For the rotate operations, the instruction register bits are at high levels and enable the JKJ and JKK outputs to follow the J00 or K00 inputs. Fundamental operation M serves to provide extra time when the shift-rotate count exceeds 1, the count which just exhausted the available time in the machine-cycle.

For every additional shift/rotate count above 1, one-sixteenth of a machine cycle is required. The required time is gained by delaying the machine cycle in the time-slot BP6. The check signal if PU6. If by the time an assertion of PU6 occurs the final count on the up counter has not been reached, signal S4MC* is asserted at low logic level (Figure 4-39, waveform H) with the effect of stopping the 4MC clock signal generation, this results in freeing the PU pulse train. When the up counter reaches the final count, however, signal CNTR* is asserted at low logic level (waveform F) and frees the 4MC clock, which again continues the generation of the PU pulse train.

Table 4-24 lists the format for the shift rotate operations and shows a tabulation of the instruction code patterns. Bits 4 and 5 call for one or both of the main accumulators, bit 6 is always set, bit 7 augments the shifts operation into a rotate operation when set, and bits 8 through 11 specify the shift-rotate count.

All processing control for this instruction subgroup is derived from operate decoder signals OP104*, OP105* OP106*, and OP107*, which when combined with appropriate bits of the shift-rotate subgroup instructions, gives rise to various other control signals. These control signals generate the fundamental operations listed in Table 4-36. Table 4-35, the event summary, lists all the operations specified in Table 4-36 which are used in the processing of the instructions. The instructions are described below.

SFTZ J (1140) - The content of the J register is shifted left by the number of bits specified by the shift count in the instruction, bits 8 through 11. Bits shifted out of bit 00 if the J register are lost and zeros are shifted into bit J11, with the number of zeros corresponding to the number of shifts.

ROTD J (1160) - Contents of the J register are rotated left by the number of bits specified by the rotate count in the instruction, bits 8 through 11. Bits rotated out of the J00th bit are loaded into the J11th bit so that no data is lost. The rotation displacement corresponds to the number of shifts specified.

SFTZ K (1240) - This instruction is similar to SFTZ J (1140) except that the K register is affected. The content of the K register is shifted left by the specified number of counts. Bits shifted out are lost.

ROTD K (1260) - This instruction is similar to ROTD J (1160) except that the K register is affected. The contents of the K register are rotated left by the number of bits specified by the rotate instruction. No data is lost.

SFTZ JK (1340) - The contents of the J register are shifted into the K register. Bits shifted out leftward from the K00th bit are lost. Thus, the contents of the K register replace the contents of the J register with zeros entering the K register. The number of zeros shifted into the K register corresponds to the number of shifts specified by the instruction.

ROTD JK (1360) - The contents of both the K and J register are exchanged by the number of shifts specified by the instruction. The J00th bit is shifted into the K11th bit and the K00th bit is shifted into the J11th bit. No data is lost.

EXJK (1374) - This instruction is identical to ROTD JK (1360) with the exception that the number of rotate displacements is exactly 12. Hence, the content of the J register is rotated into the K register and the content of the K register is rotated into the J register.

4.4.2.1.7 Load/Read Subgroup. For this subgroup, Figures 4-40, 4-41 and 4-42 are the block diagrams, Figure 4-43 is the timing diagram, Table 4-37 is the Event Summary, and Table 4-38 is the list of Fundamental Operations.

This subgroup contains three instructions. Altogether there are only 7 fundamental operations. However, the individual instructions are described by individual block diagrams. Figure 4-40 illustrates the data transactions during the execution of the LSJW (1010) instruction. The contents of the switch register enters the utility gates. Then during period BP6B the utility gate data is switched through the MX multiplexer and enters the adder. In the final operation the content of the adder replaces the content of the J register. These events can also be traced from Table 4-37, the Event Summary and from Table 4-38, the list of Fundamental Operations. For a list of instruction code patterns see Table 4-24.

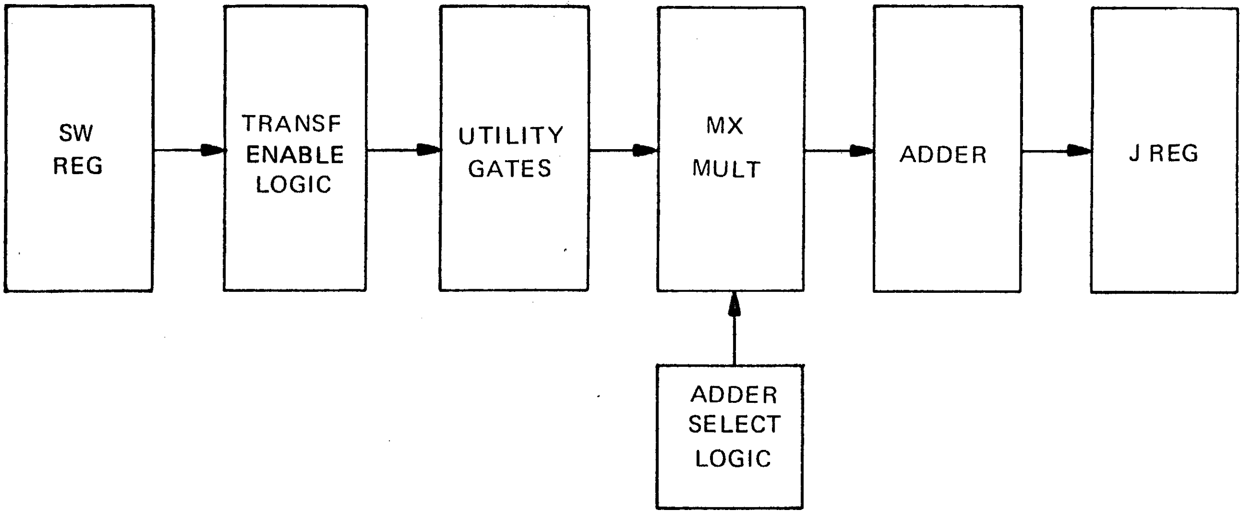


Figure 4-40. Load/Read Subgroup, LJSW Instruction, Block Diagram

Table 4-37. Load/Read Subgroup Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
LJSW	1010	A	BP56	SW→UT
		B	BP6B	UT→MX
		E	BP6	ADDER→J
LJST	1011	C	BP56	STATUS ADDRESS
		D	BP6B	STATUS→MX
		E	BP6	ADDER→J
RFOV	1002	F	BP7B	J1→OV
		G	BP7B	J0→FLAG

Figure 4-41 illustrates the basic flow of data when the content of the status register (paragraph 4.3.6) replaces the content of the J register when instruction LJST (1011) is executed. At the outset the address for the status register is formed. Referring to Table 4-19, the status register is selected when signals MXS0, MXS1, and MXS2 are all high level. However, as can be seen by referring to the logic for MX multiplexer selection (sheet 9AB1), there is no selection logic for the status register because no input is provided that enables all three MX multiplexer NOR gates simultaneously. Hence to obtain this condition, status register selection (D, Table 4-38) is obtained by enabling K- and S-register logic during period BP6. As seen from the logic expression for Fundamental Operation D in Table 4-38, the combination of S and K register addressing selects the status register as input to the MX multiplexer. The content of the status register, overflow and flag register combined, reaches the J register via the MX multiplexer and adder. Because no add or subtract selection is made. the data contribution from the TS multiplexer is all zeros.

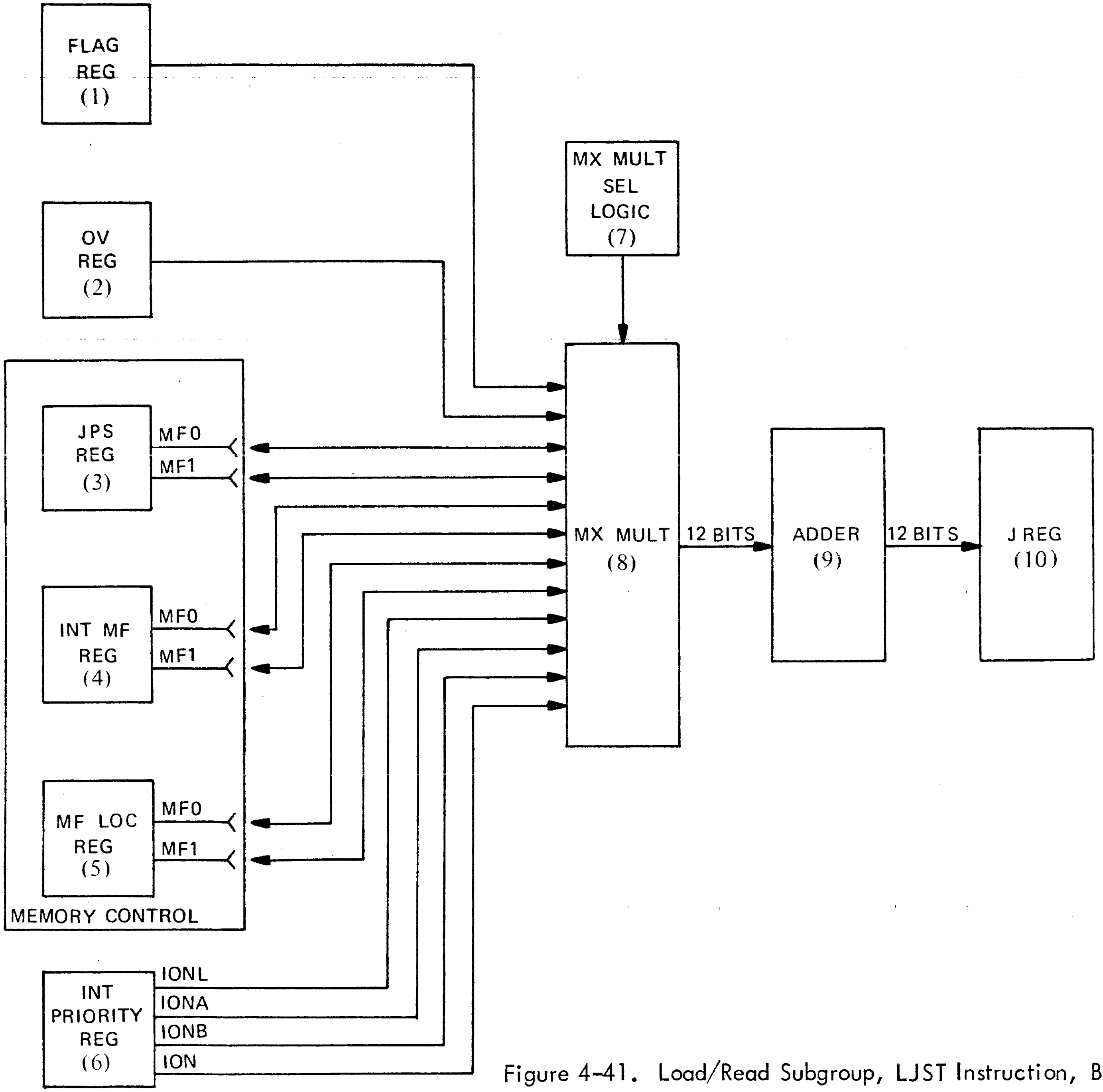


Figure 4-41. Load/Read Subgroup, LJST Instruction, Block Diagram

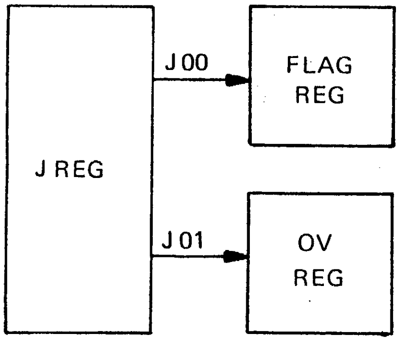


Figure 4-42. Load/Read Subgroup, RFOV Instruction, Block Diagram

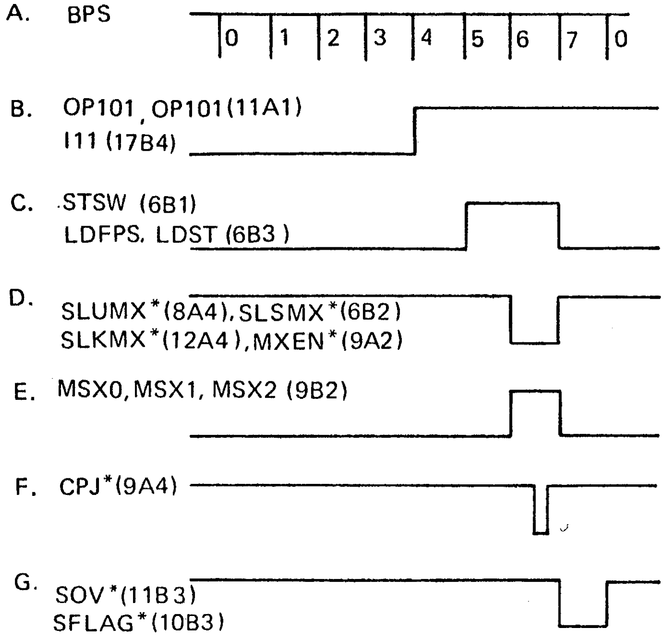


Figure 4-43. Load/Read Subgroup, Timing Diagram

Figure 4-41 is the block diagram for the LJST instruction. Data from the flag register (1), the overflow register (2), and the interrupt priority register (6) is supplied from the central processor to the input of the MX multiplexer (8). Data from the JPS register (3), the interrupt memory field register (4) and the memory-field location register (5) is supplied from the memory field control logic. When the LJST instruction is processed, the K and S register control provides a status register address to the MX multiplexer (8) through the MX multiplexer select logic (7). When the status register data is switched

Table 4-38. Load/Read Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1010)	BP56	SW→UG 1. (↓OP1*) (↑I08B)→↓OP101*→↑OP101/11A1 2. (↑OP101) (↑I4I5*) (↑BP56)→↓STSW*→ ↑STSW/6B1 3. (↑STSW) (↑I11*)→↓LDSW*→↑LDFPSW/6B3 4. (↑LDFPSW) (↑SW00-SW11)→↓U00-U11/14/15/16B2	Content of switch register enters utility gates.
B. (1010)	BP6B	UG→MX 1. (↑LDFPSW) (↑BP6B)→↓SELU6*→↑SELU/8A3 2. ↑SELU→↓SLUMX*/8A4 3. ↓SLUMX*→↑MSX0/9B1 = UT→MX 4. ↓SLUMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX	Content of utility gates enters MX multiplexer.
C. (1011)	BP56	SEL→STATUS ADDRESS 1. ↓OP101→↑OP101/11A1 2. (↑OP101) (↑I4I5*) (↑BP56)→↓STSW*→ ↑STSW/6B1 3. (↑STSW) (↑I11N)→↓LDST*→↑LDST/6B3	Addressing of status register is initialized.
D. (1011)	BP6	STATUS→MX 1. (↑LDST) (↑BP6)→↓SLSMX/8B2 2. ↓SLSMX→↑MSX0, ↑MSX1/9B2 = S→MX 3. ↓SLSMX→↑MXEN→↓MXEN*/9A2 = EN→MX 4. (↑LDST) (↑BP6)→↓SLKMX/12A4 5. ↓SLKMX→↑MSX0, ↑MSX2/9B2 = K→MX 6. ↓SLKMX→↑MXEN→↓MXEN*/9A2 = ENABLE→MX	Content of status register enters MX multiplexer.
E. (1010) (1011)	BP6	ADDER→J 1. ↓STSW*→↑PEJ6/9A3 2. (↑PEJ6) (↑BP6)→↓PEJ*→↑PEJ/9A4 3. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = LOAD→J-REG	Content of adder replaces content of J-register.
F. (1002)	BP7B	J1→OV 1. (↑OP100)(I4I5*)(↑I08B)→↓RIST*→↑RTST/11B1 2. (↑RTST) (↑BP7B)→↓PSST*→↑DSST/11B2 3. (↑DSST) (↑JO1)→↓SOV*/11B3 = SET→OV (↑OV, ↓OV*/11B4)	Content of bit J1 replaces content of overflow register.
G. (1002)	BP7B	J0→FLAG 1. (↑OP100)(↑I4I5*)(↑I08B)→↓RIST*→↑RTST/11B1 2. (↑RTST) (↑BP7B)→↓PSST*→↑DSST/11B2 3. (↑DSST) (↑J00B)→↓SFLAG*/10B3 = SET→FLAG (↑FLAG, ↓FLAG*)	Content of bit J0 replaces content of flag register.

through the MX multiplexer (8), data is supplied through the adders (9), to the J register.

Figure 4-42 is the block diagram for instruction RFOV (1002). This instruction is the reverse of the LJST (1011) previously described; information is transmitted between the same units, the status register and the J register, but the data path is reversed. The other difference is that no intermediaries are used, the J00 and J01 bits are gated by control signals directly to the status register. Note that if the flag and overflow registers have already been set, they remain set.

Table 4-24 is the instruction list for the Load/Read Subgroup.

LJSW (1010) - The content of the switch register, as determined by the positions of the front panel switches, replaces the content of the J register.

LJST (1011) - The contents of the status register replaces the previously cleared content of the J register.

RFOV (1002) - If bit J00 of the J register is a one, it replaces the content of the flag register; otherwise, it leaves it unchanged. If bit J01 of the J register is a one, it replaces the content of the overflow register; otherwise, it leaves it unchanged.

4.4.2.2 GROUP 2 OPERATE INSTRUCTIONS. Group 2 operate instructions are characterized by bit pattern 0011 in the operate code field of the instruction, bit positions 0 through 3. These instructions generally modify or test the content of the main accumulators and of the status register. The instructions of this group are microprogrammable; compatible bit patterns can be combined within one instruction to save core. If it is desired, for example, to determine the state of the J register, a 1 in bit 5 of the instruction addresses it. The same applies to bit 4 which addresses the K register. The conditional instructions all work the same way. If the condition tested is true, the program counter is incremented. Otherwise the instruction is effectively on idle.

There are 37 Group 2 operate instructions in the repertoire of the ND812 computer. The instructions are divided into six subgroups (see Table 4-39) and are listed with their mnemonic and octal codes and with a shorthand description of their operation. Subgroups are formed by taking the common processing characteristics of instruction bit patterns into consideration. This approach follows the same organization that the operate instruction decode logic follows; it has the advantage of offering an easy survey of the simplified logic tables when direct reference to them is helpful in following the instruction execution flow. The subgroups are discussed one by one.

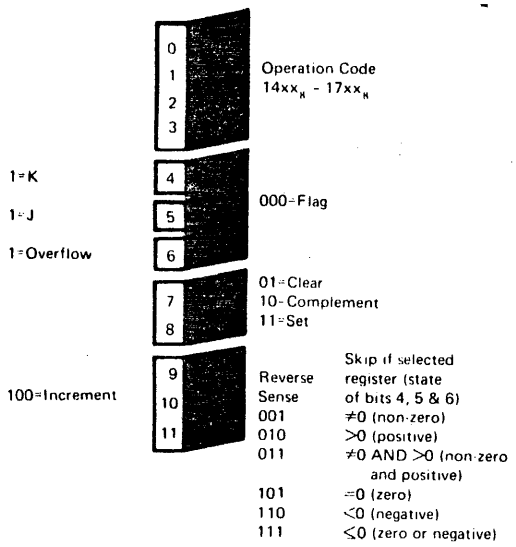
The bit pattern for Group 2 operate instructions is shown in Table 4-40. In addition to the bit patterns, this table contains all instructions in Group 2 with their bit patterns aligned within the instruction format. Note that the various bit positions have difference assignments; some address and others control. Bit 9 tests in conditional instructions and controls incrementation.

Table 4-39. Group 2 Operate Instructions By Subgroups

		Octal	
Subgroup	**Mnemonic**	**Code**	**Operation**
REGISTER MODIFY	CLR J	1510	CLEAR→J
	CLR K	1610	CLEAR→K
	CLR JK	1710	CLEAR→J,K
	CMP J	1520	COMPLEMENT→J
	CMP K	1620	COMPLEMENT→K
	CMP JK	1720	COMPLEMENT J,K
	SET J	1530	SET J TO IS
	SET K	1630	SET K TO IS
	SET JK	1730	SET J AND K TO IS
	INC J	1504	INCREMENT J
	INC K	1604	INCREMENT K
	NEG J	1524	NEGATE J
	NEG K	1624	NEGATE K
REGISTER SIGN SKIP	SIP J	1502	SKIP IF J POSITIVE
	SIP K	1602	SKIP IF K POSITIVE
	SIP JK	1702	SKIP IF J,K POSITIVE
	SIN J	1506	SKIP IF J NEGATIVE
	SIN K	1606	SKIP IF K NEGATIVE
	SIN JK	1706	SKIP IF J,K NEGATIVE
REGISTER ZERO SKIP	SIZ J	1505	SKIP IF J EQUALS ZERO
	SIZ K	1605	SKIP IF K EQUALS ZERO
	SIZ JK	1705	SKIP IF BOTH J,K EQUAL ZERO
	SNZ J	1501	SKIP IF J NOT EQUAL ZERO
	SNZ K	1601	SKIP IF K NOT EQUAL ZERO
	SNZ JK	1701	SKIP IF J,K NOT EQUAL ZERO
STATUS MODIFY	CLR	1410	CLEAR FLAG
	CLR O	1450	CLEAR OVERFLOW
	CMP	1420	COMPLEMENT FLAG
	CMP O	1460	COMPLEMENT OVERFLOW
	SET	1430	SET FLAG
	SET O	1470	SET OVERFLOW
STATUS SKIP	SIZ	1405	SKIP IF FLAG ZERO
	SIZ O	1445	SKIP IF OVERFLOW ZERO
	SNZ	1401	SKIP IF FLAG ONE
	SNZ O	1441	SKIP IF OVERFLOW ONE
CYCLE DELAY	IDLE	1400	ONE CYCLE DELAY

The format consists of four distinct fields. The operate field always contains bits 0011. The register field addresses one or more of the four registers. Note that a zero in bit 6 implies the flag register in the operation by the presence of other control signals that act on the flag register. The modify field allows for three operations; complementing, clearing and setting. For setting, bits 7 and 8 are at high logic. The increment-skip field is the most complex of the four fields. When bit 9 is the only bit set in the field, it stands for incrementing. Otherwise it assumes the value shown for the two sets of conditions

Table 4-40. Group 2 Operate Instructions, Bit Pattern Formats By Subgroup



This is the group 2 operate instruction format. Bits 4 and 5 select the J and K registers, respectively. However, if these bits are unset the overflow and flag registers are selected according to the state of bit 6. Bits 7 and 8 select the register operation and bits 9, 10, and 11 control reverse sensing.

Octal			Bit Pattern											
Code	**Mnemonic**	**Subgroup**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1504	INC J	REGISTER MODIFY			1	1		1				1		
1510	CLR J				1	1		1			1			
1520	CMP J				1	1		1		1				
1524	NEG J				1	1		1		1		1		
1530	SET J				1	1		1		1	1			
1604	INC K				1	1	1					1		
1610	CLR K				1	1	1				1			
1620	CMP K				1	1	1			1				
1624	NEG K				1	1	1			1		1		
1630	SET K				1	1	1			1	1			
1710	CLR JK				1	1	1	1			1			
1720	CMP JK				1	1	1	1		1				
1730	SET JK				1	1	1	1		1	1			
1502	SIP J	REGISTER SIGN SKIP			1	1		1					1	
1506	SIN J				1	1		1				1	1	
1602	SIP K				1	1	1						1	
1606	SIN K				1	1	1					1	1	
1702	SIP JK				1	1	1	1					1	
1706	SIN JK				1	1	1	1				1	1	
1501	SNZ J	REGISTER ZERO SKIP			1	1		1						1
1505	SIZ J				1	1		1				1		1
1601	SNZ K				1	1	1							1
1605	SIZ K				1	1	1					1		1
1701	SNZ JK				1	1	1	1						1
1705	SIZ JK				1	1	1	1				1		1
1410	CLR	STATUS MODIFY			1	1					1			
1420	CMP				1	1				1				
1430	SET				1	1				1	1			
1450	CLR O				1	1			1		1			
1460	CMP O				1	1			1	1				
1470	SET O				1	1			1	1	1			

Table 4-40. Group 2 Operate Instructions, Bit Pattern Formats By Subgroup (Cont'd.)

Octal Code	Mnemonic	Subgroup	0	1	2	3	4	5	6	7	8	9	10	11
1401	SNZ	STATUS SKIP			1	1								1
1405	SIZ				1	1						1		1
1441	SNZ O				1	1			1					1
1445	SIZ O				1	1			1			1		1
1400	IDLE	CYCLE DELAY			1	1								

separated by the dotted line. In other words, in the strict sense, bits 10 and 11 define the condition for the skip. For example, instruction 1405, SIZ (skip if flag = zero) can be developed from the instruction format in a hypothetical case when the programmer does not have access to an instruction listing. Encoding from the left, it is seen that bits 2 and 3 are set; therefore, this is an operate 2 instruction. Next, observe that all bits up to bit 9 are zeros. At first thought it could be an increment instruction but then it is noticed that bit 11 is also set. Therefore we are dealing with a skip instruction. The set states of bit 9 and bit 11 test for the condition: = 0. Returning to the register field, it is noticed that all bits of the field are zero (bits 4,5 and 6). Therefore, the flag register is indicated.

Table 4-40 can serve as a basis for microprogramming; it can point out whether superimposed operations are possible and compatible. Note that only the 1's are shown for clarity.

The 36 Group 2 operate instructions in the repertoire of the ND812 computer are divided into six subgroups (see Table 4-39) and are listed with their mnemonic and octal codes and with a shorthand description of their operations. Subgroups were formed on the basis of similar processing characteristics of the instructions within the subgroup.

4.4.2.2.1 Register Modify Subgroup. For this subgroup Figure 4-44 is the block diagram, Figure 4-45 is the timing diagram, Table 4-41 is the Event Summary and Table 4-42 is the instruction listing in terms of fundamental operations.

All instructions of this subgroup are concerned with operations upon the contents of the J and K registers or both. The possible operations are the clear operation, which sets all bits of a register to zero; the complement operation, which sets all bits of a register to their complemented value; the set operation, which sets all bits of the register to 1; the increment operation, which adds one to the content of the register; this sum replaces the previous content of the register; and the negate operation which loads the register with the 2's complement of the previous content of the register. Operation details can be best described by considering Table 4-42, which contains the simplified logic expressions for the fundamental operations required for this subgroup. Clearing the J and K register is accomplished by combining decoder output signal OP2 with the I08B instruction code bit and signal CLRKJ. When all three signals are high logic level, either the J or the K register is cleared, depending on the value of signals I4SKP and I5BSKP, both of which are derivatives of instruction bits 4 and 5, respectively. Switching of the main accumulators through the TS multiplexer is also done by appropriate logic manipulations with decoder signal OP2

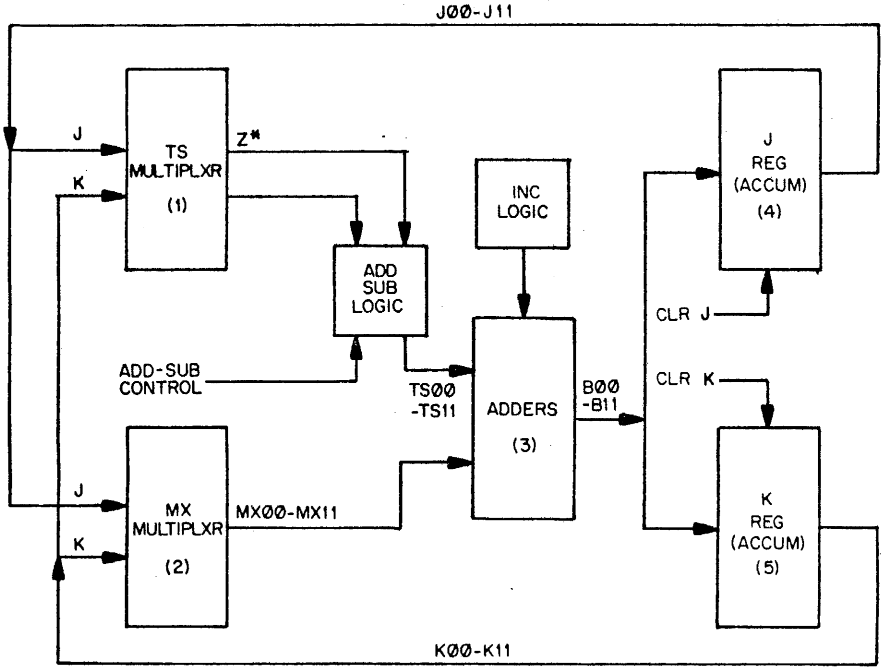


Figure 4-44. Register Modify Subgroup, Block Diagram

and the instruction-bit derivatives. Incrementation is carried out by utilizing fundamental operation E (Table 4-42), which supplies a carry to the adder. The incremented quantity is then returned via the TS multiplexer and adder to the register from where it was switched through the TS multiplexer at the outset of the operation. Fundamental operation F is the negation, and is equivalent to a 2's complement. As seen from the simplified logic expressions, a carry is propagated through the adder when the TS add-subtract logic is set to execute a subtract operation. Fundamental operation G, the 1's complement, is characterized by the suppression of the carry through the adder in agreement with 1's complement subtraction theory. The necessity of disallowing the carry is indicated by the instruction-bit pattern for the last three bits of the instruction which are the bit-codes for all 1's complement instructions. Fundamental operations H and I are concerned with generating the adder-to-register enabling and clocking signals. Note that for all instructions of this subgroup, the non-zero quantity supplied to the adder arrives via the TS multiplexer; the MX multiplexer contributes all zeros.

The set operation is a combination of fundamental operations A and G. First, all bits of the register content are set to zero; then 1's complementing the zeros yields the desired result, all 1's.

All processing control for this instruction subgroup is derived from operate decoder signal OP2*, which, when combined with appropriate bits of the register modify subgroup instructions, gives rise to various other control signals. These control signals generate the

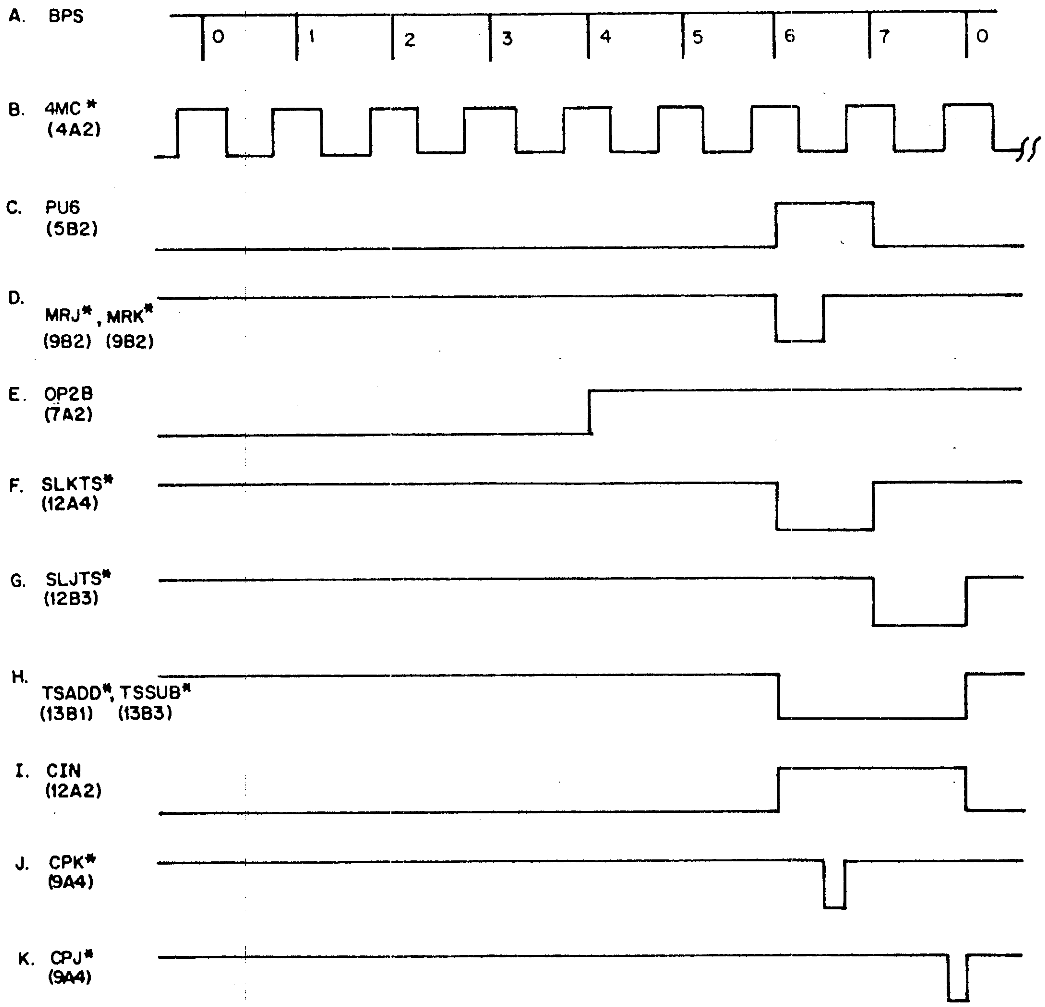


Figure 4-45. Register Modify Subgroup, Timing Diagram

fundamental operations listed in Table 4-42. Table 4-41, the Event Summary, lists all the operations specified in Table 4-42, which are used in the processing of the instructions. The instructions are described below.

INC J (1504) - The content of the J register is switched through the TS multiplexer to the add-subtract logic. The data word is switched through the add-subtract logic and is admitted into the adder. The adder is incremented and its contents loaded into the J register. The J register then holds its original value incremented by a count of 1.

CLR J (1510) - The content of the J register is cleared so that its original content is replaced by all zeros. Next the content of the J register is switched through the TS multiplexer and the add-subtract logic to the adder. The content of the adder is admitted into the J register so that its original value is replaced by all zeros. The two operations subsequent to the first clearing operation are not necessary, but they are important during other loop operations that occur when instructions of this subgroup are processed. Hence, the two operations that occur during BP7 are irrelevant during this instruction.

Table 4-41. Register Modify Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
INC J	1504	C	BP7	J→TS→±LOGIC
		E	BP67	TS + 1→ADDER
		H	BP7	ADDER→J
CLR J	1510	A	PU6	CLR→J (J=0000)
		C	BP7	J→TS→±LOGIC
		H	BP7	ADDER→J (J=0000)
CMP J	1520	C	BP7	J→TS→±LOGIC
		G	BP67	COMPL→TS→±LOGIC
		H	BP7	ADDER→J
NEG J	1524	C	BP7	J→TS→±LOGIC
		F	BP67	2s COMPL→TS→±LOGIC
		G	BP67	*1s COMPL→TS→±LOGIC*
		H	BP7	ADDER→J
SET J	1530	A	PU6	CLR→J (J=0000)
		C	BP7	J→TS→±LOGIC
		G	BP67	COMPL→TS→±LOGIC
		H	BP7	ADDER→J (J=7777)
INC K	1604	D	BP6B	K→TS→±LOGIC
		E	BP67	TS + 1→ADDER
		I	BP6	ADDER→K
CLR K	1610	B	PU6	CLR→K (K=0000)
		D	BP6B	K→TS→±LOGIC
		I	BP6	ADDER→K (K=0000)
CMP K	1620	D	BP6	K→TS→±LOGIC
		G	BP67	COMPL→TS→±LOGIC
		I	BP6	ADDER→K
NEG K	1624	D	BP6	K→TS→±LOGIC
		F	BP67	2s COMPL→TS→±LOGIC
		G	BP67	*1s COMPL→TS→±LOGIC*
		I	BP6	ADDER→K
SET K	1630	B	PU6	CLR→K (K=0000)
		D	BP6B	K→TS→±LOGIC
		G	BP67	COMPL→TS→±LOGIC
		I	BP6	ADDER→K (K=7777)
CLR JK	1710	A	PU6	CLR→J (J=0000)
		B	PU6	CLR→K (K=0000)
		C	BP7	J→TS→±LOGIC
		D	BP6B	K→TS→±LOGIC
		H	BP7	ADDER→J (J=0000)
		I	BP6	ADDER→K (K=0000)

Table 4-41. Register Modify Subgroup, Event Summary (Cont'd.)

Mnemonic	Octal Code	Ref.	Period	Event
CMP JK	1720	C	BP7	J →TS →±LOGIC
		D	BP6B	K →TS →±LOGIC
		G	BP67	1s COMPL →TS →±LOGIC
		H	BP7	ADDER →J
		I	BP6	ADDER →K
SET JK	1730	A	PU6	CLR →J (J=0000)
		B	PU6	CLR →K (K=0000)
		C	BP7	J →TS →±LOGIC
		D	BP6B	K →TS →±LOGIC
		G	BP67	1s COMPL →TS→±LOGIC
		H	BP7	ADDER →J (J=7777)
		I	BP6	ADDER →K (K=7777)

Table 4-42. Register Modify Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1510) (1530) (1710) (1730)	PU6	CLEAR →J 1. (↑4MCD) (↑PU6) →↓CLRKJ* →↑CLRKJ/4A3 2. (↑OP2) (↑I08B) (↑CLRKJ) →↓OP2CL* → ↑OP2CL/9B2 3. (↑OP2CL) (↑I5BSKP) →↓MRJ*/9B2	Clear contents of J-register to zero.
B. (1610) (1630) (1710) (1730)	PU6	CLEAR →K 1. (↑4MCD) (↑PU6) →↓CLRKJ* →↑CLRKJ/4A3 2. (↑OP2) (↑I08B) (↑CLRKJ) →↓OP2CL* → ↑OP2CL/9B2 3. (↑OP2CL) (I4SKP) →↓MRK*/9B2	Clear content of K-register to zero.
C. (1504) (1510) (1520) (1524) (1530) (1710) (1720) (1730)	BP7	J→TS→±LOGIC 1. ↓OP2* →↑OP2/7A3 2. (↑OP2) (↑I5BSKP) →↓OP2J* →↑OP2J/7A3 3. (↑OP2J) (↑BP7) →↓SLJTS*/12B3→ ↑TSSO, ↑TSS1/13B4	Contents of J-register enters TS multiplexer.
D. (1604) (1610) (1620) (1624) (1630) (1710) (1720) (1730)	BP6B	K→TS →±LOGIC 1. (↑OP2) (↑I4SKP)→↓OP2K*/7A3, ↓OP2K* →↑KTS6/12A4 2. (↑KTS6) (↑BP6B) →↓SLKTS* 12A4 3. ↓SLKTS* →↑TSS1/13B4	Contents of K-register enters TS multiplexer.

Table 4-42. Register Modify Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
E. (1504) (1604)	BP67	TS+1 →ADDER 1. ↓OP2* →↑OP2B/7A2 2. (↑OP2B) (↑I07*) (↑BP67)→↓TSADD*/13B1 → ↓INC* →↑INC/12B4 3. (↑INC) (↑BP67) →↓CIN2* →↑CIN/12A2	The contents of the TS multiplexer is incremented.
F. (1524) (1624)	BP67	2s COMPL→TS→±LOGIC 1. ↓OP2*→↑OP2B/7A2 2. (↑OP2B) (↑BPB7) (↑I07B) →↓TSSUB*/13B3 3. ↓TSSUB*→↑TSSUB/13B4 4. (↑OP2) (↑I09B) (↑I10*) (↑I11*)→ ↓INC*/12B3→↑INC/12B4 5. (↑INC) (↑BP67) →↓CIN2* →↑CIN/12A2	The TS multiplexer contributes the 2's complement to the adder.
G. (1520) (1524) (1530) (1620) (1624) (1630) (1720) (1730)	BP67	1s COMPL→TS→±LOGIC 1. (↑OP2B) (↑BPB7) (↑I07B)→↓TSSUB*/13B3 2. ↓TSSUB* →↑TSSUB/13B4 3. ↓OP2*→↑CIN2*/12A2 4. (↑CIN2*) (↑CIN1*) (↑CIN3*)→↓CIN/12B2	The TS multiplexer contributes the 1's complement to the adder.
H. (1504) (1510) (1520) (1524) (1530) (1710) (1720) (1730)	BP7	ADDER→J 1. ↓OP2*→↑OP2/7A3 2. (↑OP2) (↑I5BSKP)→↓OP2J* →↑OP2J/7A3 3. (↑OP2J) (BP7) →↓PEJ* →↑PEJ/9A4 4. (↑PEJ) (↑REGCLK) →↓CPJ*/9A4	Content of adder replaces content of J-register.
I. (1604) (1610) (1620) (1624) (1630) (1710) (1720) (1730)	BP6	ADDER →K 1. ↓OP2* →↑OP2/7A3 2. (↑OP2) (↑I4SKP) →↓OP2K*/7A3 3. ↓OP2K*→↑PEK6/9B3 4. (↑PEK6) (↑BP6)→↓PEK* →↑PEK/9A4 5. (↑PEK) (↑REGCLK) →↓CPK*/9A4	Content of adder replaces content of K-register.

CMP J (1520) - The content of the J register is switched through the TS multiplexer to the add-subtract logic, and from the add-subtract logic, to the adder. The complement of the data word is switched through the add-subtract logic so that the adder receives the 1's complement of the original content of the J register. The content of the adder is admitted into the J register. Thus, the J register contains the complement of its original data word.

NEG J (1524) - The content of the J register is switched through the TS multiplexer to the add-subtract logic. Next, the complement of the original J register data word is admitted into the adder and the adder is incremented by a count of 1 to obtain the 2's complement of the original J register data word. Next, the content of the adder is admitted into the J register so that it contains the 2's complement of its original value, or -J. Although the 1's complement operation is also performed when this instruction is processed, this operation has no relevance because the 1's complement data path is the same as the 2's complement data path, except that for 1's complementing, the adder is not incremented.

SET J (1530) - The content of the J register is cleared to all zeros. Next, the content of the J register is switched through the TS multiplexer to the add-subtract logic and through the add-subtract logicto the adder. The complement of the J-register data word is switched throug the add-subtract logic so that the adder holds all 1's (the complement of all zeros). Next the content of the adder is admitted into the J register so that the J register contains all 1's.

INC K (1604) - This operation is similar to instruction INCJ (1504) with the exception that the content of the K register is switched through the TS multiplexer, and the result replaces the content of the K register.

CLR K (1610) - This operation is similar to instruction CLR J (1510) with the exception that the content of the adder replaces the content of the K register.

CMP K (1620) - This operation is similar to instruction CMP J (1520) with the exception that the content of the K register is switched through the TS multiplexer, and the content of the adder replaces the content of the K register.

NEG K (1624) - This operation is similar to instruction NEG J (1524) with the exception that the content of the K register is switched through the TS multiplexer and the content of the adder is admitted into the K register.

SET K (1630) - This instruction is similar to instruction SET J (1530) with the exception that the K register is cleared and the content of the adder is admitted into the K register.

CLR JK (1710) - This operation combines both the CLR J and the CLR K instructions. Both the J and K registers are cleared during period PU6, and the content of the adder is admitted to the K register during period BP6 and into the J register during period BP7.

CMP JK (1720) - This operation combines both the CMP J and the CMP K instructions. The content of the K register is switched through the TS multiplexer during period BP6 and the content of the adder is admitted to the K register. During period BP7, the content of the J register is switched through the TS multiplexer and the content of the adder is admitted into the J register.

SET JK (1730) - This operation combines both the SET J and SET K instructions. The content of the K and J registers is cleared during period PU6 and the content of the adder is admitted into the K register during period BP6 and into the J register during period BP7.

4.4.2.2.2 Register Sign/Skip Subgroup. For this subgroup Figure 4-46 is the block diagram, Figure 4-47 is the timing diagram, Table 4-43 is the Event Summary, and Table 4-44 is the list of Fundamental Operations.

Instructions of this subgroup are structured as follows; when a given condition associated with the particular instruction is met, the porgram counter is advanced by a count of one. This incrementation of the program counter is additional to the normal advancing of the program counter during the basic phase. If the condition is not met, the program counter remains unchanged. All skip conditions are related in this subgroup to the assign bit of the J or K register or of both. The content of the reigster is considered positive only if the necessary and sufficient condition bit 0=0 (J00 or K00) is met. Otherwise, the content is negative.

For all instructions in the subgroup, fundamental operations G and H are common. The simplified logic expressions reflect the values of the variables for the case when the given condition is met. Fundamental operations A, B and C are carried out, with the given results, when the J or K registers, or both contain a positive value; fundamental operations D, E, and F correspond to the negative register values. Fundamental operation H is carried out only when the specified condition for the particular instruction is met. If so, a carry to the adder is propagated through the adder and added to the content of the program counter which is also selected as input to the adder.

Finally, the incremented value replaces the previous content of the program counter.

The control signals initiating the fundamental operations are attained through combinations of the instruction decoder signal OP2 with instruction code derivatives (I4SKP, I5BSKP) and by examination of the sign determining bits of the J or K registers. Table 4-43 contains the list of instructions in the register sign-skip subgroup. The instructions are described below.

SIP K (1502) - The content of the program counter is transferred to the MX multiplexer. Bit 0 of the J register is examined. If the bit is a zero, a carry is propagated through the adder. The result replaces the previous content of the program counter. Otherwise the content of the program counter is unchanged.

SIP K (1602) - This instruction is the same as the above except that the K register content is examined for the sign.

SIP JK (1702) - This instruction is the combination of the above two instructions; the program counter is incremented only if both register contents are positive.

SIN J (1506) - The content of the program counter is transferred to the MX multiplexer. Bit 0 of the J register is examined. If bit Ø is a one, a carry is propagated through the adder. The result replaces the previous content of the program counter. If the test condition is not met, the content of the program counter is returned unchanged.

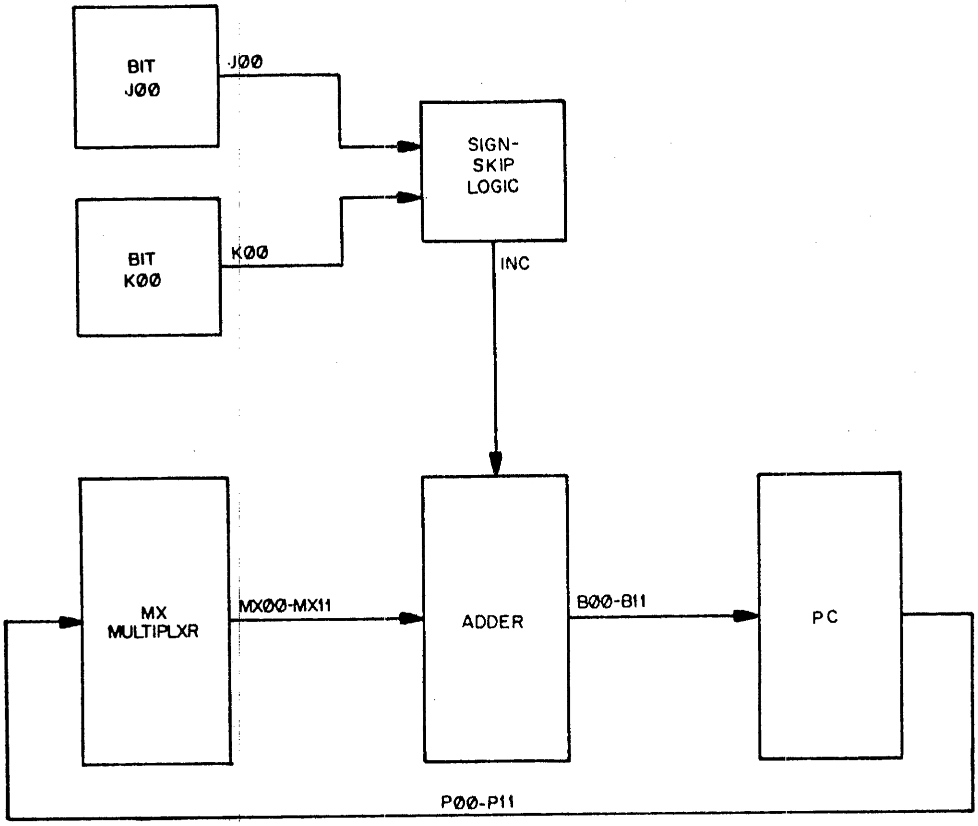


Figure 4-46. Register Sign/Skip Subgroup, Block Diagram

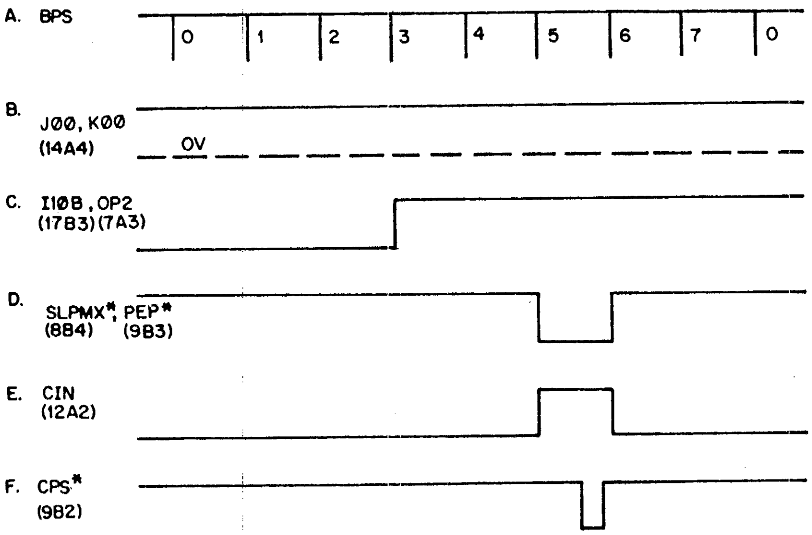


Figure 4-47. Register Sign/Skip Subgroup, Timing Diagram

Table 4-43. Register Sign/Skip Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
SIP J	1502	A	All	J→POS?
		C	BP3	REG→POS?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIP K	1602	B	All	K→POS?
		C	BP3	REG→POS?
		G	BP56	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIP JK	1702	A	All	J→POS?
		B	All	K→POS?
		C	BP3	REG→POS?
		G	BP56	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIN J	1506	D	BP3	J→NEG?
		F	BP3	REG→NEG?
		G	BP56	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIN K	1606	E	BP3	K→NEG?
		F	BP3	REG→NEG?
		G	BP56	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIN JK	1706	D	BP3	J→NEG?
		E	BP3	K→NEG?
		F	BP3	REG→NEG?
		G	BP56	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP56	ADDER→PC

Table 4-44. Register Sign/Skip Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1502) (1506) (1602) (1606) (1702) (1706)	All	J→POS? 1. ↓J00→↑J00*→↓J00B/10A2 2. ↓J00B, ↑SKJ0*→↑SKJPOS*/10A3	The content of the J-register is positive.

Table 4-44. Register Sign/Skip Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
B. (1502) (1506) (1602) (1606) (1702) (1706)	All	K→POS? 1. ↓K00→↑K00*→↓K00B/10A2 2. ↓K00B→↑SKK0*→↑SKPPOS*/10A3	The content of the K-register is positive.
C. (1502) (1602) (1702)	BP3	J+K REG→POS? 1. ↓I4SKP→↑SKKO/10A3 2. ↓I5BSKP→↑SKJO/10A3 3. ↓I04* + ↓I05* + ↓I06*→↑CKFL*/10A2 4. (↑CKFL*) (↑I10B) (↑I09*)→↓FLPOS*→ ↑FLPOS/10A3 5. (↑SKJO) (↑SKKO) (↑FLPOS)→ ↓SKPOS*/10A3 6. ↓SKPOS*→↑SKPOS/10A3 7. (↑SKPOS) (↑I11*)→↓OPSKP*→↑OP2SKP/10A4	The content of the tested register is positive.
D. (1502) (1506) (1602) (1606) (1702) (1706)	BP3	J→NEG? 1. ↑J00→↓J00*→↑J00B/10A2 2. (↑J00B) (↑I5BSKP)→↓SKJ0*→↑SKJ0/10B3	The content of the J-register is negative.
E. (1502) (1506) (1602) (1606) (1702) (1706)	BP3	K→NEG? 1. ↑K00→↓K00*→↑K00B/10A2 2. (↑K00B) (↑I4SKP)→↓SKK0*→↑SKK0/10B3	The content of the K-register is negative.
F. (1506) (1606) (1706)	BP3	J+K REG→NEG? 1. ↓I4SKP→↑SKK0/10B3 2. ↓I5BSKP→↑SKJ0/10B3 3. ↓I04* + ↓I05* + ↓I06*→↑CKFL*/10A2 4. (↑CKFL*) (↑I10B) (↑I09B)→↓SKNEG*→ ↑SKNEG/10B3 5. (↑SKJ0) (↑SKK0) (↑SKNEG)→↓OPSKP*→ ↑OP2SKP/10A4	The content of the tested register is negative.

Table 4-44. Register Sign/Skip Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
G. (1502) (1506) (1602) (1606) (1702) (1706)	BP5B	PC→MX→ADDER 1. (↑OP2) (↑BP5B)→↓SLPMX*/8B4 2. ↓SLPMX*→↑MXS1/9B2 3. ↓SLPMX*→↑MXEN→↓MXEN*/9A2	The contents of the program counter is input to the MX multiplexer.
H. (1502) (1506) (1602) (1606) (1702) (1706)	BP5	+1→ADDER 1. ↓OP2*→↑OP2B/7A2 2. (↑OP2SKP) (BP5) (↑OP2)→↓CIN3*→ ↑CIN/12A2 = +1→ADDER	The content of the adder is incremented.
I. (1502) (1506) (1602) (1606) (1702) (1706)	BP5B	ADDER→PC 1. (↑OP2) (↑BP5B)→↓BP205*→↑MPEP→ ↓PEP*/9B3 = ENABLE→PC 2. ↓PEP*→↑PEP/9B1 3. (↑PEP) (↑REGCLK)→↓CPP*/9B2 = LOAD→PC	The program counter is incremented.

SIN K (1606) - This instruction is analogous to instruction SIP K (1602) except that the register is examined for the negative condition.

SIN JK (1706) - This instruction is analogous to instruction SIP JK (1702) except that both the J and K registers are examined for the negative condition.

4.4.2.2.3 Register Zero/Skip Subgroup. For this subgroup Figure 4-48 is the block diagram, Figure 4-49 is the timing diagram, Table 4-45 is the Event Summary, Table 4-46 is the list of Fundamental Operations. Instructions of this subgroup are structured similarly to the register sign/skip subgroup. When a given condition associated with the particular instruction is met, the program counter is advanced by a count of one. This incrementation is additional to the normal advancing of the program counter during the basic phase. If the condition is not met the program counter remains unchanged. Three instructions within this subgroup result in a skip if the content of the given register is zero; the other three provide a skip when the content of the given register is non-zero. For all instructions in the subgroup fundamental operations G, H, and I are common. The simplified logic expressions reflect the values of the logic variables for the case when the given condition is met. Fundamental operations C, D, and F are carried out, with the shown results, when the J or K or both registers contain zero; fundamental operations A, B, and E correspond to non-zero register content(s). Fundamental operation H is carried out only when the specified condition for the particular instruction is met. If so, a carry to the adder is generated and

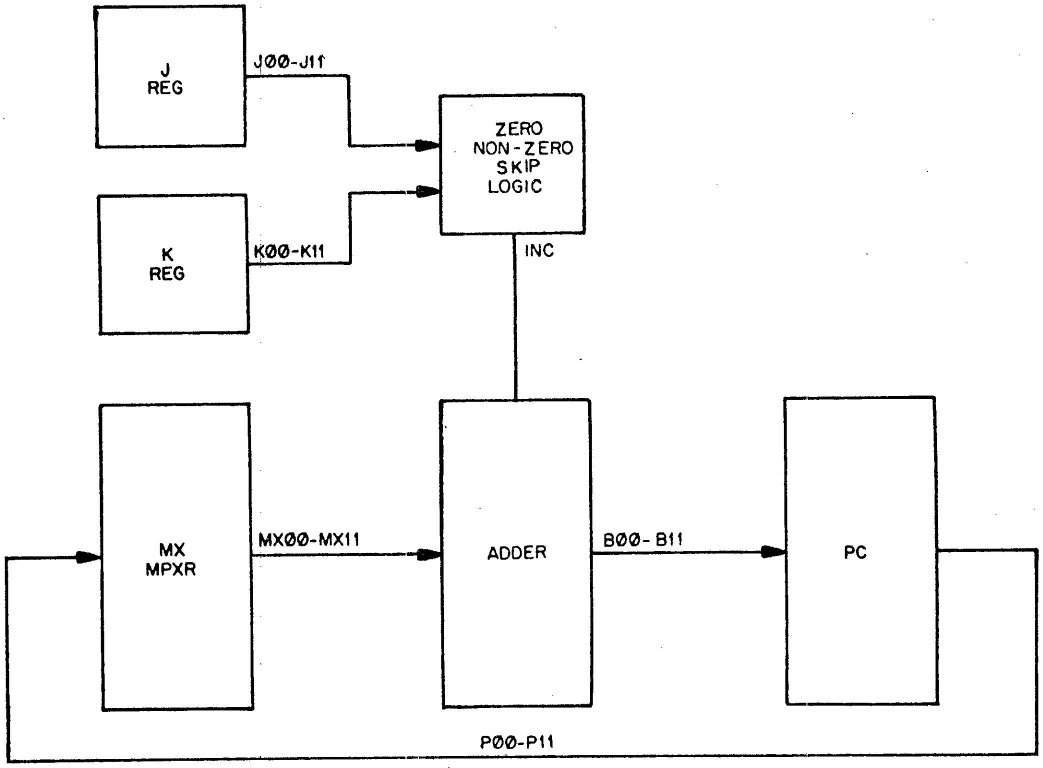


Figure 4-48. Register Zero/Skip Subgroup, Block Diagram

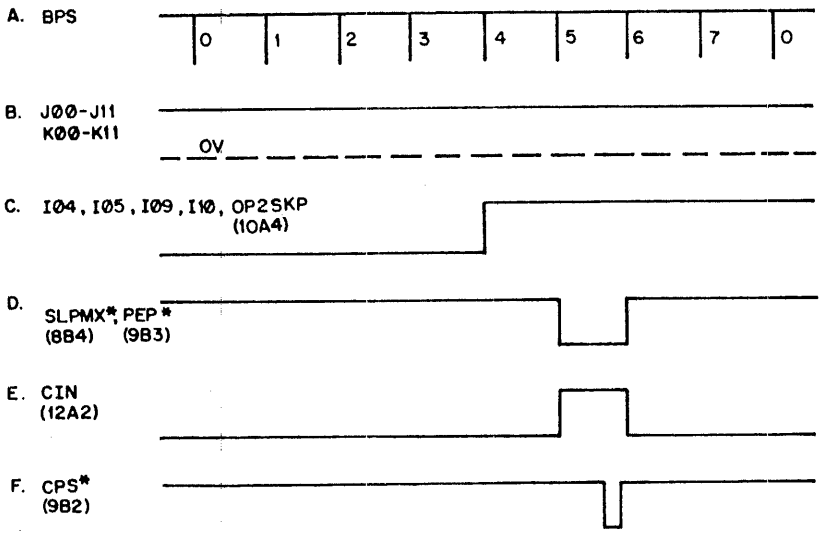


Figure 4-49. Register Zero/Skip Subgroup, Timing Diagram

Table 4-45. Register Zero/Skip Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
SNZ J	1501	A	All	J ≠ 0?
		E	BP3	REG ≠ 0?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SNZ K	1601	B	All	K ≠ 0?
		E	BP3	REG ≠ 0?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SNZ JK	1701	A	All	J ≠ 0?
		B	All	K ≠ 0?
		E	BP3	REG ≠ 0?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP56	ADDER→PC
SIZ J	1505	C	BP3	J = 0?
		F	BP5	REG = 0?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIZ K	1605	C	BP3	J = 0?
		D	BP3	K = 0?
		F	BP5	REG = 0?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC
SIZ JK	1705	C	BP3	J = 0?
		D	BP3	K = 0?
		F	BP5	REG = 0?
		G	BP5B	PC→MX→ADDER
		H	BP5	ADDER→+1
		I	BP5B	ADDER→PC

added to the content of the program counter which is also selected as input to the adder via the MX multiplexer. Finally, the incremented value replaces the previous content of the program counter.

The control signals initiating the fundamental operations are attained by combinations of the decoder signal OP2 with instruction code derivatives (I4SKP, I5BSKP) and by examination of all bits of the J or K registers. Table 4-45 contains the list of instructions in the register sign/skip subgroup.

Table 4-46. Register Zero/Skip Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1501) (1701)	All	J ≠ 0? 1. ↑J00 + . . . + ↑J07→↑JZ007*→ ↓JZ007/10A2 2. ↑J08 + . . . + ↑J11→↑JZ811*→ ↓JZ811/10B2 3. ↓JZ007 + ↓JZ811→↑JZ15*/10A2 4. ↑JZ15*→↓SKJZ/10A4	The content of the J-register is not zero.
B. (1601) (1701)	All	K ≠ 0? 1. ↑K00 + . . . + ↑K07→↑KZ007*→↓KZ007/10A2 2. ↑K08 + . . . + ↑K11→↑KZ811*→↓KZ811/10B2 3. ↓KZ007 + ↓KZ811→↑KZ14*/10A2 4. (↑KZ14*) (↑I4SKP)→↓SKKZ/10A4	The content of the K-register is not zero.
C. (1505) (1705)	BP3	J = 0? 1. (↓J00) . . . (↓J07)→↓JZ007*→ ↑JZ007/10A2 2. (↓J08) . . . (↓J11)→↓JZ811*/10B1→↑JZ811/10B2 3. (↑JZ007) (↑JZ811) (↑I5BSKP)→↓JZ15*/10A2 4. ↑JZ15*→↑SKJZ/10A4	The contents of the J-register is zero.
D. (1605) (1705)	BP3	K = 0? 1. (↓K00) . . . (↓K07)→↓KZ007*→↑KZ007/10A2 2. (↓K08) . . . (↓K11)→↓KZ811*→↑KZ811/10B2 3. (↑KZ007) (↑KZ811) (↑I4SKP)→↓KZ14*/10A2 4. ↓KZ14*→↑SKKZ/10A4	The content of the K-register is zero.
E. (1501) (1601) (1701)	BP3	REG ≠ 0? 1. ↓I4SKP→↑KZ14*/10A2 2. ↓I5BSKP→↑JZ15*/10A2 3. ↓I06B→↑OZ16*/10A2 4. ↓I04* + ↓I05* + ↓I06*→↑CKFL*/10A2 5. (↑CKFL*) (↑I09*) (↑I11B)→↓FLNZ*→ ↑FLNZ/10A3 6. (↑OZ16*) (↑KZ14*) (↑JZ15*) (↑FLNZ)→ ↓SKNZ*→↑SKNZ/10A3 7. (↑SKNZ) (↑I10*)→↓OPSKP*→↑OP2SKP/10A4	The content of the tested register is not zero.
F. (1505) (1605) (1705)	BP5	REG = 0? 1. ↓JZ15*→↑SKJZ/10A4 2. ↓KZ14*→↑SKKZ/10A4 3. ↓I04* + ↓I05* + ↓I06*→↑CKFL/10A2 4. (↑I09B) (↑I11B) (↑CKFL*)→↓SZ911*→ ↑SZ911/10A4 5. ↓I06B→↑SK0Z/10A4 6. (↑SKJZ) (↑SZ911) (↑SKKZ) (↑SK0Z)→ ↓SKZ*/10A4 7. ↓SKZ*→↑OP2SKP/10A4 8. (↓OP2SKP) (↑BP5) (↑OP2)→↓CIN3*→↑CIN/12A2	The content of the tested register is zero.

Table 4-46. Register Zero/Skip Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
G. (1501) (1505) (1601) (1605) (1701) (1705)	BP5B	PC→MX→ADDER 1. (↑OP2) (↑BP5B)→↓SLPMX*/8B4 2. ↓SLPMX*→↑MXSI/9B2 3. ↓SLPMX*→↑MXEN→↓MXEN*/9A2	The contents of the program counter is input to the MX multiplexer.
H. (1501) (1505) (1601) (1605) (1701) (1705)	BP5	ADDER→+1 1. (↓OP2*/7A2→↑OP2B/7A2 2. (↑OP2SKP/10A4) (↑BP5/5A3) (↑OP2/7A3)→ ↓CIN3*/12A2→↑CIN/12A2/16A3	The content of the adder is incremented.
I. (1501) (1505) (1601) (1605) (1701) (1705)	BP5B	ADDER→PC 1. (↑OP2/7A3) (↑BP5B/9B2)→↓BP205*→ ↑MPEP/9B3→↓PEP*/9B3 2. ↓PEP*/9B3→↑PEP/9B1 3. (↑PEP/9B1) (↑REGCLK/4A4)→ ↓CPP*/9B2/14B3/15B3/16B3	Content of adder replaces content of program counter.

SNZ J (1501) - The content of the program counter is transferred to the MX multiplexer. The content of the J register is examined. If it is zero, a carry is propagated through the adder. The result replaces the previous content of the program counter. If the condition is not met the content of the program counter is unchanged.

SNZ K (1601) - This instruction is the same as instruction SNZ J, except that the K register is examined for zero content.

SNZ JK (1701) - This instruction is the combination of instructions SNZ J and SNZ K; the counter is incremented only if both registers have zero value.

SIZ J (1505) - The content of the program counter is transferred to the MX multiplexer. The content of the J register is examined. If it is non-zero, a carry is sent to the adder. The result replaces the previous content of the program counter. If the condition is not met the content of the program counter is returned unchanged.

SIZ K (1605) - This instruction is analogous to instruction SNZ K (1601) except that the K register is examined for the non-zero condition.

SIZ JK (1705) - This instruction is analogous to instruction SNZ JK (1701) except that both the J and K registers are examined for the non-zero condition.

4.4.2.2.4 Status Modify Subgroup. For this subgroup Figure 4-50 is the block diagram, Figure 4-51 is the timing diagram, Table 4-47 is the Event Summary, and Table 4-48 is the list of Fundamental Operations.

Instructions in this subgroup are concerned with operations upon the contents of the overflow and flag registers. Possible operations are the clear, which sets the content of either register to zero; the complement, which sets either register to its complement value; and the set, which replaces the value of either register by a one. Operational details are described by considering Table 4-48, which contains the logic expressions for the fundamental operations required for this subgroup of instructions. Clearing the registers is accomplished by generating signal OP2 from the operate decoder. The high logic level OP2CL signal, when combined with instruction code-bit I06, sets the overflow register to zero. Complementing is accomplished in a similar fashion, but setting a register is done in two operations. First a clear, and then complementing results in yielding a one output for either register.

Table 4-47 contains the list of instructions for the status modify subgroup.

CLR (1410) - This instruction replaces the content of the flag register by a zero.

CLR O (1450) - This instruction replaces the content of the overflow register by a one.

CMP (1420) - This instruction replaces the content of the flag register by its complement value.

CMP O (1460) - This instruction replaces the content of the overflow register by its complement value.

SET (1430) - This instruction replaces the content of the flag register by a zero, then complements it with the result of the flag register containing a one.

SET O (1470) - This instruction replaces the content of the overflow register by a zero, then complements it with the result of the overflow register containing a one.

4.4.2.2.5 Status Skip Subgroup. For this subgroup Figure 4-52 is the block diagram, Figure 4-53 is the timing diagram, Table 4-49 is the Event Summary, and Table 4-50 is the list of Fundamental Operations. Instructions in this group are structured similarly to the register sign/skip and the register zero/skip subgroups with the exception that for these instructions the main accumulators are not affected, but the overflow and flag registers, or when combined, the status register content constitutes the object of the test. When the given condition associated with the particular instruction is met, the program counter is advanced by a count of one. This incrementation is additional to the normal advancing of the program counter as required by the basic phase. If the specified condition is not met the program counter remains unchanged. There are four instructions in this subgroup, the two registers are not tested in combination, only one or the other register is tested for its content. For all instructions of this subgroup fundamental operations E and G are required.

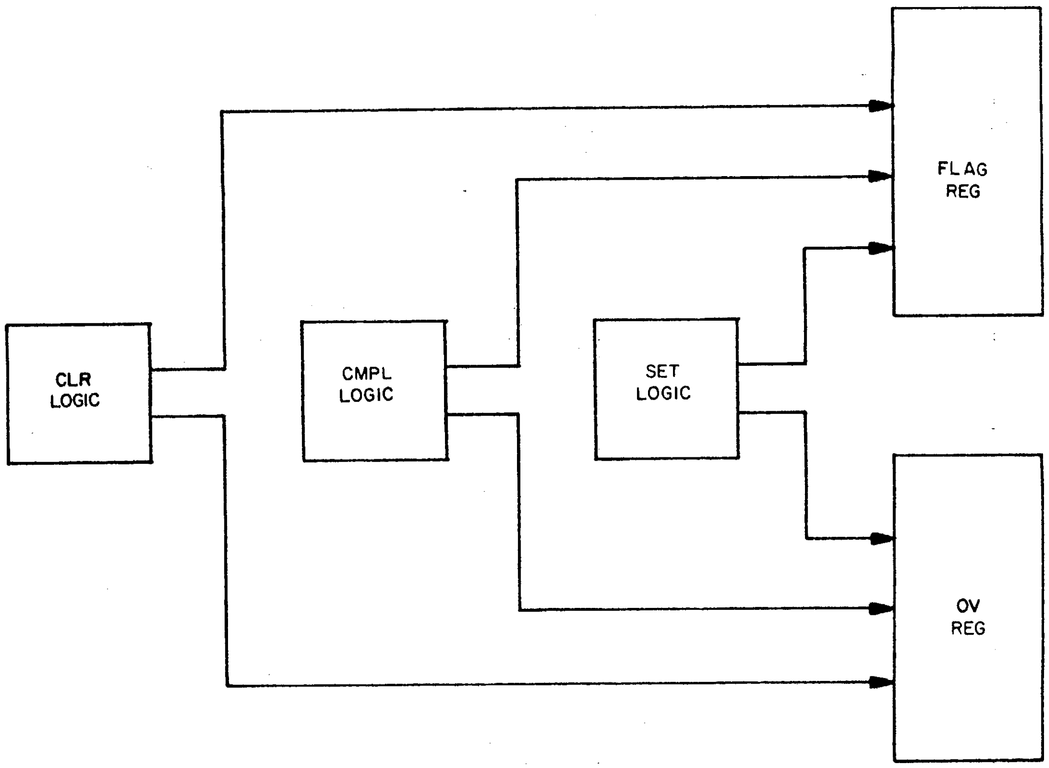


Figure 4-50. Status Modify Subgroup, Block Diagram

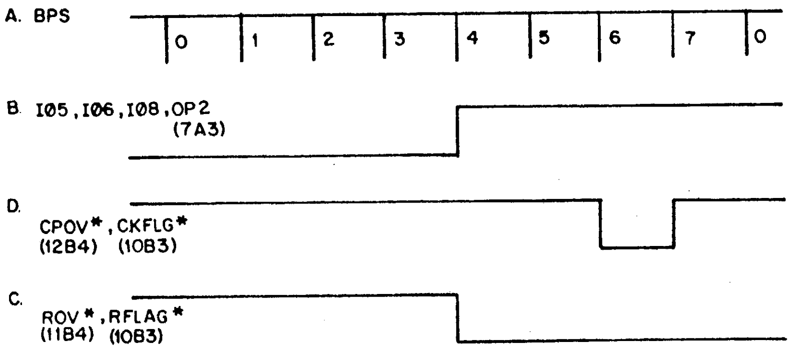


Figure 4-51. Status Modify Subgroup, Timing Diagram

Table 4-47. Status Modify Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
CLR	1410	B	BP3	CLEAR→FLAG
CLRO	1450	A	BP3	CLEAR→OV
CMP	1420	D	BP6	COMP→FLAG
CMPO	1460	C	BP6	COMP→OV
SET	1430	B	BP3	CLEAR→FLAG
		D	BP6	COMP→FLAG
SET O	1470	A	BP3	CLEAR→OV
		D	BP6	COMP→OV

Table 4-48. Status Modify Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1450) (1470)	BP3	CLEAR→OV 1. (↑OP2) (↑I08B) (↑CLRKJ)→↓OP2CL*→ ↑OP2CL/9B2 2. (↑OP2CL) (↑I06B)→↓CLOV*→↑ROV→ ↓ROV*/11B4 3. ↓ROV* = CLEAR→OV (↓OV, ↑OV*/11B4)	Sets content of overflow register to zero.
B. (1410) (1430)	BP3	CLEAR→FLAG 1. (↑OP2) (↑I08B) (↑CLRKJ)→↓OP2CL*→ ↑OP2CL/9B2 2. (↑I04*) (↑I05*) (↑I06*)→ ↓CKFL*/10A2 3. ↓CKFL*→↑CKFL/10B3 4. (↑OP2CL) (↑CKFL)→↓RFLAG*/10B3 5. ↓RFLAG* = CLEAR→FLAG (↓FLAG, ↑FLAG*/10B4)	Sets content of flag register to zero.
C. (1460) (1470)	BP6	CMPL→OV 1. (↑OP2) (↑BP6) (↑I06B) (↑I07B)→ ↓CPLOV*/11B3 2. ↓CPLOV*→↑CPOV/11B4 = TOGGLE ENABLE 3. (↑CPOV) (↑REGCLK)→↓CKOV/11B4 = COMPL→OV	Complements the content of the overflow register.
D. (1420) (1430)	BP6	CMPL→FLAG 1. (↑I04*) (↑I05) (↑I06)→↓CKFL*/10A2→ ↑CKFL/10B2 2. (↑CKFL) (↑BP6)→↓CKFLG*/10B3 3. (↑OP2) (↑PULSKP) (↑I07B) = TOGGLE ENABLE 4. ↓CKFLG*/10B3 = CMPL→FLAG	Complement the content of the flag register.

The simplified logic expressions reflect the values of the logic variables for the case when the given condition is met. Fundamental operation A is carried out when the overflow register contains zero, fundamental operation B is carried out when the overflow register contains one. Fundamental operations C and D are analogous to the above for the flag register. Fundamental operation F is carried out only if the specified condition for the given instruction is met. If so, a carry is propagated through the adder and added to the content of the program counter which is also switched to the adder through the MX multiplexer. Finally, the incremented value replaces the previous content of the program counter. The control signals initiating the fundamental operations are generated by combinations of operate decoder signal OP2 with instruction code bit I06. When the required condition is met, signal OP2SKP goes to a high logic level causing a carry to go to the adder. The instructions are described below.

SNZ (1401) - The content of the program counter is transferred to the MX multiplexer. The content of the flag register is examined. If it is a one, a carry is sent to the adder. The result replaces the previous content of the program counter. If the condition is not met the content of the program counter is returned unchanged.

SIZ (1405) - This instruction is the same as the instruction above except that skip is executed when the content of the flag register is one.

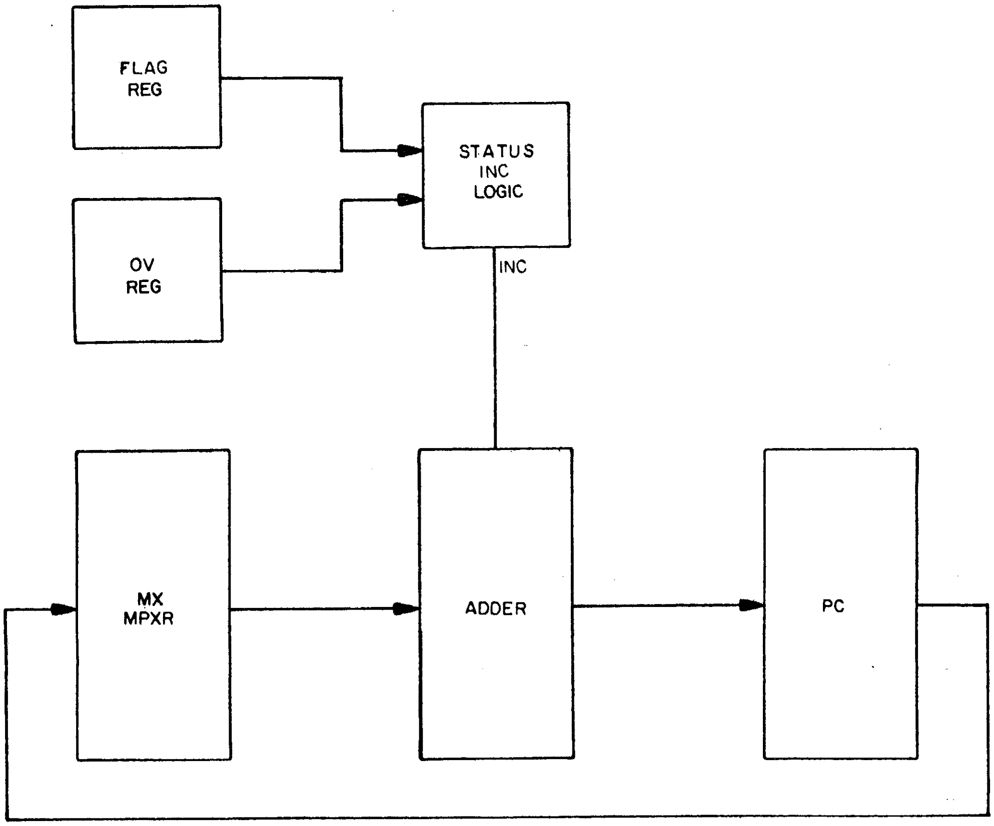


Figure 4-52. Status Skip Subgroup, Block Diagram

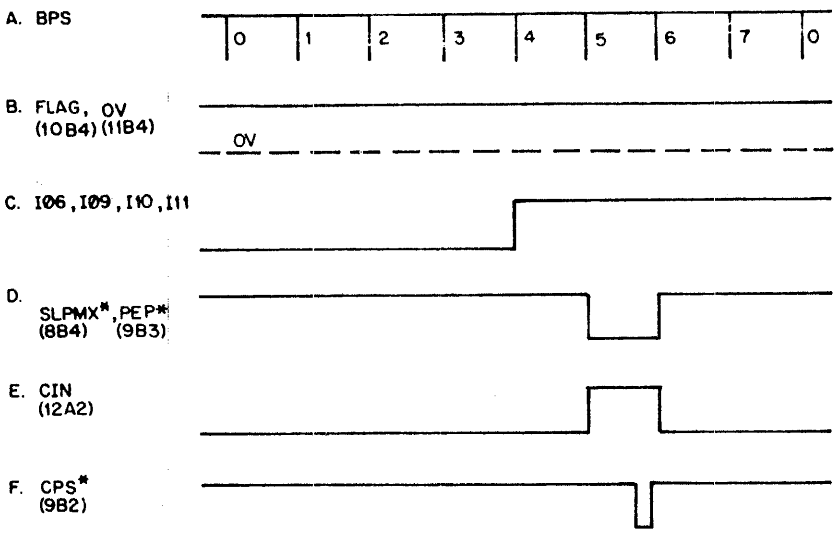


Figure 4-53. Status Skip Subgroup, Timing Diagram

Table 4-49. Status Skip Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
SNZ	1401	D	BP3	FLAG ≠ 0?
		E	BP56	PC→MX→ADDER
		F	BP5	ADDER→+1
		G	BP5B	ADDER→PC
SIZ	1405	C	BP3	FLAG = 0?
		E	BP5B	PC→MX→ADDER
		F	BP5	ADDER→+1
		G	BP5B	ADDER→PC
SNZ O	1441	B	BP3	OV ≠ 0?
		E	BP56	PC→MX→ADDER
		F	BP5	ADDER→+1
		G	BP5B	ADDER→PC
SIZ O	1445	A	BP3	OV = 0?
		E	BP56	PC→MX→ADDER
		F	BP5	ADDER→+1
		G	BP5B	ADDER→PC

Table 4-50. Status Skip Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (1445)	BP3	OV = 0? 1. (↑OV*) (↑I06B)→↓0Z16*→↑SK0Z/10A4 2. ↓I04* + ↓I05* + ↓I06*→↑CKFL*/10A2 3. (↑I09B) (↑I11B) (↑CKFL*)→↓SZ911*/10A4 4. ↓SZ911*→↑SZ911/10A4 5. ↓I4SKP→↑SKKZ/10A4 6. ↓I5BSKP→↑SKJZ/10A4 7. (↑SK0Z) (↑SZ911) (↑SKKZ) (↑SKJZ)→ ↓SKZ*/10A4 8. ↓SKZ*→↑OP2SKP/10A4	The overflow register contains a zero.
B. (1441)	BP3	OV = 1? 1. ↓OV*→↑OZ16*/10A2 2. ↓I4SKP→↑KZ14*/10A2 3. ↓I5BSKP→↑JZ15*/10A2 4. ↓I04* + ↓I05* + ↓I06*→↑CKFL*/10A2 5. (↓CKFL*) (↑I09*) (↑I11B)→↓FLNZ*→ ↑FLNZ/10A3 6. (↑OZ16*) (↑KZ14*) (↑JZ15*) (↑FLNZ)→ ↓SKNZ*/10A3 7. (↓SKNZ) (↑I10*)→↓OPSKP*→↑OP2SKP/10A4	The overflow register contains a one.
C. (1405)	BP3	FLAG = 0? 1. (↑FLAG*) (↑I09B)→↓SKFL*→↑SKFL/10B4 2. (↑I04*) (↑I0 5*) (↑I06*)→↓CKFL*→↑CKFL/10B3 3. (↑SKFL) (↑CKFL) (↑I11B)→↓SKFLP*/10B4 4. ↓SKFLP*→↑OP2SKP/10A4	Flag register contains zero.
D. (1401)	BP3	FLAG = 1? 1. (↑FLAG) (↑I09*)→↓SKFL*→↑SKFL/10B4 2. (↑I04*) (↑I05*) (↑I06*)→↓CKFL*→↑CKFL/10B3 3. (↑SKFL) (↑CKFL) (↑I11B)→↓SKFLP*/10B4 4. ↓SKFLP*→↑OP2SKP/10A4	Flag register contains one.
E. (1401) (1405) (1441) (1445)	BP5B	PC→MX→ADDER 1. (↑OP2) (↑BP5B)→↓SLPMX*8B4 2. ↑SLPMX*→↑MXS1/9B2 3. ↓SLPMX*→↑MXEN→↓MXEN*/9A2	The content of the program counter is input to MX multiplexer.
F. (1401) (1405) (1441) (1445)	BP5	ADDER→+1 1. ↓OP2*→↑OP2B/7A2 2. (↑OP2SKP/10A4) (↑BP5) (↑OP2/7A3)→ ↓CIN3*→↑CIN/12A2	The content of the adder is incremented.
G. (1401) (1405) (1441) (1445)	BP5B	ADDER→PC 1. (↑OP2) (↑BP5B)→↓BP205*→↑MPEP→ ↓PEP*/9B3 2. ↓PEP*→↑PEP/9B1 3. (↑PEP) (↑REGCLK)→↓CPP*/9B2	Content of adder replaces content of program counter.

SNZ O (1441) - This instruction is analogous to instruction SNZ (1401) except that the overflow register is examined for a one.

SIZ O (1445) - This instruction is analogous to instruction SIZ (1405) except that the overflow register is examined for a zero.

4.4.2.2.6 Cycle Delay Subgroup (1400). There is only one instruction in this subgroup. It has a basic phase, but besides the routine processing that every operate 2 instruction goes through, this instruction has no active role, it initiates no operation. Its sole purpose is to gain one basic phase.

4.4.3 SINGLE-WORD MEMORY REFERENCE INSTRUCTIONS

Among the basic instructions, there are 11 single-word memory reference instructions. In a most general sense, each of these instructions is processed in a somewhat similar manner, although particular details during basic and execute phases may vary. For example, in a one-word memory reference instruction, when the operand is fetched directly from the same memory field in which the instruction is located and further, from one of the memory reference locations which are relatively addressed directly by bits 6 through 11 of the instruction, the basic phase is used to load the instruction register between periods PU0 and PU3. However, when the operand is fetched indirectly, the machine cycle is modified to include two basic phases before the execute phase is entered, because the relative address does not name the location of the operand, but the address of the operand.

In the case of the direct fetch, the basic phase is used to fetch the instruction into the instruction register from its location in memory. Then, during the ensuing execute phase, periods EP0 through EP3 are used to modify the current address with the relative address to produce the effective address. The operand is then fetched from the location specified by the effective address. In the latter case, when an indirect fetch is instructed, the first basic phase cycle is used to fetch the instruction into the register and to set the indirect mode between periods PU0 and PU4. Then, the second basic phase is used to modify the current address with the relative address to fetch the indirect (effective) address into the memory data register. Next, during the ensuing execute phase, the indirect address is loaded into the address register and the operand fetched from that location. If an operand must be fetched from another memory field, a two-word memory reference instruction must be used because it only permits other memory fields to be named.

The memory reference instructions operate on data contained in memory. The following descriptions illustrate the processes that take place using block diagrams, timing diagrams, and various tabulations, including simplified logic describing the events, and the manner in which instructions of this group are processed. Paragraph 4.3.4 should be reviewed to understand how the instruction decoders operate.

The memory reference instruction descriptions generally include a block diagram which illustrates the data transfer paths invoked by the called instruction, and a timing diagram which shows various time-related functions of the hardware control logic, together with two important tabulations called Event Summary and Fundamental Operations.

The Event Summary tabulation shows the relationship between the mnemonic symbol registered in the permanent symbol table of the ND812 assembler, the octal code which calls the various hardware routines (a condensed version of data listed in Table 4-51), the hardware routines themselves, and a shorthand notation of the process which takes place during the operation.

The Fundamental Operation tabulation shows the relationship between the octal code, the hardware routine called by the code, and a simplified logic description of the operation in terms of the period in which it occurs.

These tables should be used in trouble analysis. The Event Summary identifies the data transfer paths, their time relationship in the machine cycle, and the individual hardware operations. These tables are useful in identifying the various multiplexers, registers, and accumulators used in the operation. On the other hand, the Fundamental Operations tabulations are useful in identifying the control logic which is used to obtain the hardware process.

Table 4-52 shows the significant bits of the memory reference instructions with the corresponding bit patterns for any of the instructions in the group. The relative address fields are not shown.

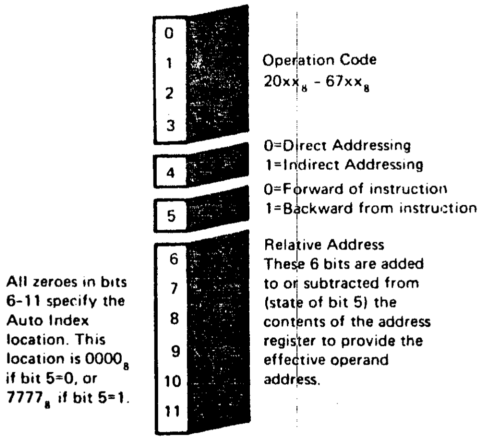
After following several instructions through their respective processing operations, these peculiarities of ND812 operation will become apparent. The descriptions in this section detail how each of the various memory reference instructions are processed during the execute phase. In these descriptions, the memory reference instructions are described in order of increasing complexity.

First the relative direct address memory reference instructions are discussed, followed by those instructions which use relative indirect addressing. The discussion is concluded by discussing the two-word memory reference instructions.

Table 4-51. Single-Word Memory Reference Instructions

Subgroup	Mnemonic	Octal Code	Operation
MRI	ANDF	2000	Logical AND J with memory
	SMJ	2400	Skip if J ≠ memory
	DSZ	3000	Decrement memory, skip MR = 0
	ISZ	3400	Increment memory, skip MR = 0
	SBJ	4000	Subtract memory from J
	ADJ	4400	Add memory to J
	LDJ	5000	Load memory into J
	STJ	5400	Store J into memory
	JMP	6000	Jump unconditionally
	JPS	6400	Jump to subroutine
	XCT	7000	Execute instruction n

Table 4-52. Single-Word Memory Reference Instruction Format and Bit Patterns


0
1
2
3
Operation Code
20xx₈ - 67xx₈
4
0=Direct Addressing
1=Indirect Addressing
5
0=Forward of instruction
1=Backward from instruction
6
7
8
9
10
11
Relative Address
These 6 bits are added to or subtracted from (state of bit 5) the contents of the address register to provide the effective operand address.
All zeroes in bits 6-11 specify the Auto Index location. This location is 0000₈ if bit 5=0, or 7777₈ if bit 5=1.

This is the single-word memory reference instruction format. Bit 4 selects direct or indirect addressing. Bit 5 determines the direction of displacement. Bits 6 through 11 are the relative address. When summed with the current address they form the effective address. If these bits are all zeroes they specify an auto index register accordingly as bit 5 is set or unset.

Octal Code	Mnemonic	OP Code 0	 1	 2	 3	Direct = 0 Indirect = 1 4‡	Add = 0 Subtract = 1 5
2000	ANDF		1				
2400	SMJ		1		1		
3000	DSZ		1	1			
3400	ISZ		1	1	1		
4000	SBJ	1					
4400	ADJ	1			1		
5000	LDJ	1		1			
5400	STJ	1		1	1		
6000	JMP	1	1				
6400	JPS	1	1		1		
7000	XCT	1	1	1			
7400	I/O	1	1	1	1		

‡For operation code 2XXX, if bit 4 is not set the operation is an AND; if bit 4 is set, the operation is a literal summation. No indirect reference is possible.

Before continuing, paragraph 4.4.1 should be reviewed to understand the common basic phase.

4.4.3.1 INSTRUCTION FORMAT. Table 4-51 is the list of single-word memory reference instructions. The basic memory reference instruction format consists of a 4-bit instruction code that permits any one of the 12 memory reference instructions (Table 4-52) to be named. The value of the first octal character ranges between 2 and 7. The 3rd binary bit in the second octal character may or may not be set depending on the instruction code. Hence, the 12 memory reference instruction operation codes are 20_8, 24_8, 30_8, 34_8, 40_8, 44_8, 50_8, 54_8, 60_8, 64_8, 70_8, and 74_8. The 4th and 5th binary bits of the 2nd octal character permit four permutations of the relative address defined by bits 6 through 11 (the 3rd and 4th octal characters).

4.4.3.2 ADDRESS PERMUTATIONS. These permutations together with the six effective address bits permit the current location ±63 locations (128 total) to be accessed to obtain the operand either directly or indirectly. When bit 4 of the instruction is not set, the operand is fetched directly from memory. When the instruction is processed, the current address is modified by the displacement value accordingly as the direction bit is set or unset. If the direction bit (bit 5) is not set, the relative address is added to the current address; if the direction bit is set, the relative address is subtracted from the current address. When bit 4 is set the operand is fetched indirectly from memory. In the indirect fetch, the current address is modified by the relative address in the same manner as for the direct fetch, however, the operand is not fetched from the named location, but from the address specified by the named location; hence, the address specified by the relative address and sign bit is not regarded as the location of the operand, but as the location of the address of the operand. Use of the indirect bit permits any location in the current memory field to be named. The permutations and octal equivalents of the one-word memory reference instruction are listed in Table 4-53.

Table 4-53. Single Word Memory Reference Instructions, Operational Permutations

Operation	Octal Code	Bit Status
Direct Add	X0ZZ	Y00
Direct Subtract	X1ZZ	Y01
Indirect Add	X2ZZ	Y10
Indirect Subtract	X3ZZ	Y11

X = Operation code for first octal character
Y = Operation code for second octal character (0_8, or 4_8)
Z = Relative address for 3rd and 4th octal characters

4.4.3.3 AUTOINDEXING. Two locations of the memory field in which the instruction resides are reserved for autoindexing, another special case of memory reference instructions. If the relative address field of the instruction specifies a zero displacement (00_8) and the indirect address bit is set, either the first location in the current memory field (0000_8), or the last location (7777_8) is referenced accordingly as direction bit 5 is unset or set. When this bit is unset, location 0000_8 (called first) is the autoindex register, and when set, location 7777_8 (called last) is the autoindex register.

When an autoindex instruction is processed the content of the specified memory location is incremented. Thus, the incremented operand is then used as the effective address from which the operand is fetched. When indirect bit 4 is unset, these locations are processed as any normal memory reference instruction.

4.4.3.4 DIRECT ADDRESSING. These instructions require, in addition to the common basic phase, one execute phase in order to complete an instruction. To summarize the events that take place during the common basic phase, it should be pointed out that the main functions of the basic phase are to access the memory location from which the instruction itself is to be brought and to provide for decoding of the instruction. Decoding results in generating the control signals which determine further processing. One of the primary functions is setting of the major state at the end of the execute phase.

During the preceding basic phase, the done latch is not set because there is no latch-setting input to the done-latch logic at the conclusion of the basic phase, when a single-word memory reference instruction is processed. This is verified by reference K, Table 4-20, operation No. 1. Thus, because the done latch remains unset, the instruction register is not cleared during period PU0 of the ensuing execute phase. This is verified by reference D, Table 4-20; the done latch must be set to clear the instruction register. Hence, data stored in the instruction register is available to processing logic during the current execute phase. At the end of the basic phase, during fundamental operation K (Table 4-20), operation 1 states, in effect, that the instruction in process is a relative direct-address single-word memory reference instruction, or that another instruction must be executed next. On the other hand, operation 3 of the fundamental operation K (Table 4-20), states that the instruction in process is completed, is a two-word memory reference instruction, or is an indirect single-word memory reference instruction. In the former case, the execute state (phase) is set and entered during the next time phase, while in the latter case, the basic state (phase) is reentered. Note that the common basic phase is completely invarient for all relative-direct single-word memory reference instructions; however, this is not true for an instruction which requires more than one basic phase. These anomolies are described under Indirect Memory Reference and Two-Word Memory Reference Instructions.

4.4.3.4.1 Execute Phase. For the direct execute phase Figure 4-54 is the block diagram, Figure 4-55 is the timing diagram, Table 4-54 is the Event Summary, and Table 4-55 is the list of Fundamental Operations.

Block Diagram Description - During the direct execute phase the current address is summed with the displacement address to become the effective address; this new address is stored in the address register and the program counter is incremented. An exception to this is the JMP instruction which inhibits advancing of the program counter at this time.

When pulse EP0 occurs, the content of the address register (9, Figure 4-54) is transferred over busses A00 through A11 to the MX multiplexer (5). During the same period the information contained by operand bits 6 through 11 (relative address) is transferred from the instruction register (2) over busses I06 through I11, to the TS multiplexer (4). Also, during this same period the output of the TS multiplexer (relative address) is summed with the current address switched through the MX multiplexer (5), to the adder (8). Adder outputs are available over busses B00 through B11. The result of this summation becomes the effective address. Pulse PU0B causes this new address to be transferred from the adders (8) back into the address register (9). This address data is available to memory control over data lines A00 through A11.

When pulse EP1 occurs the content of the program counter (10) is switched through MX multiplexer (5) via busses P00 through P11 to the adder (8). It should be noted that the program counter still contains the current address. However, the output of the MX multiplexer (5), which is applied to the adder (8) via busses MX00 through MX11, is incremented

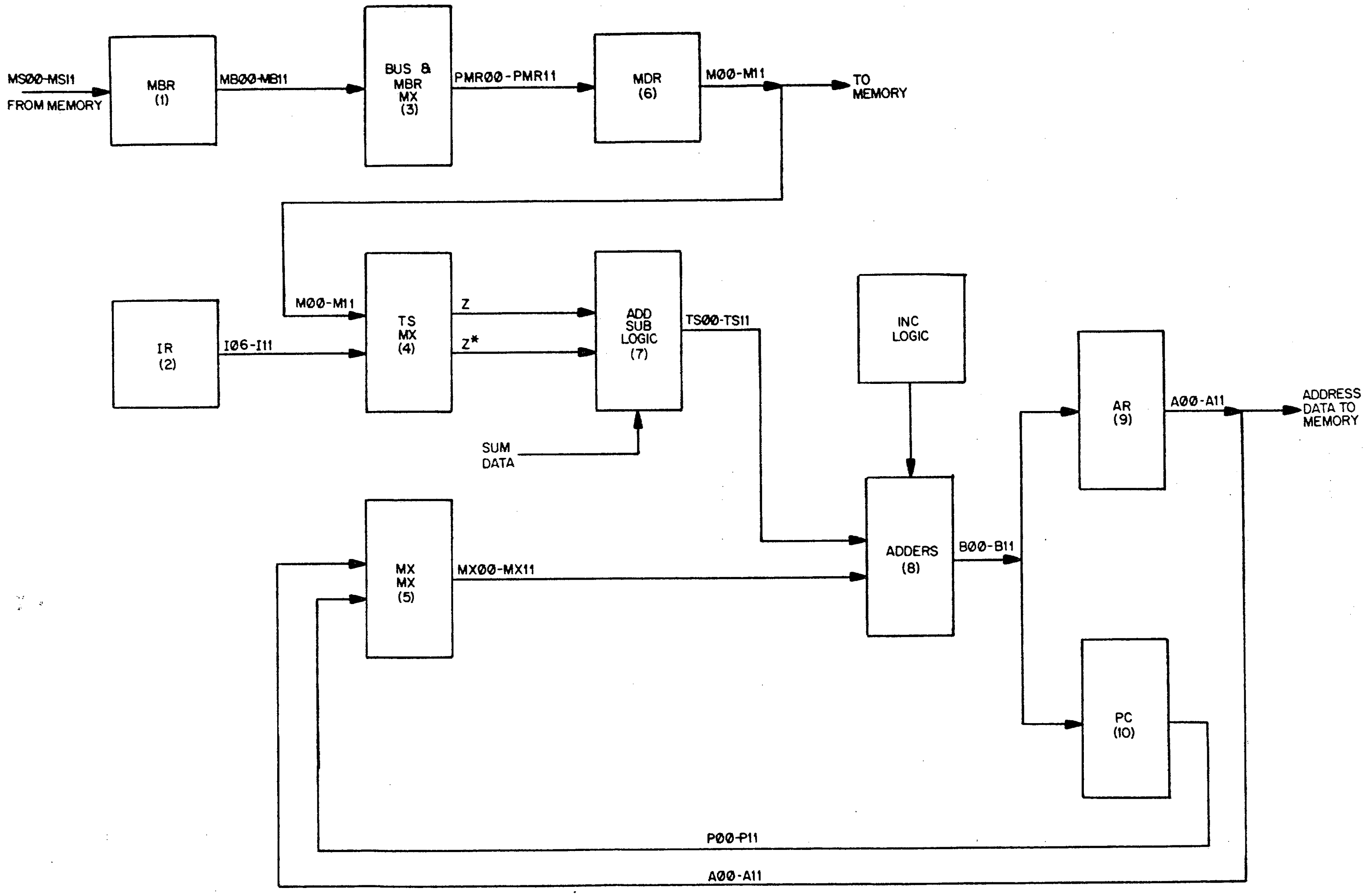


Figure 4-54. Single-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Block Diagram

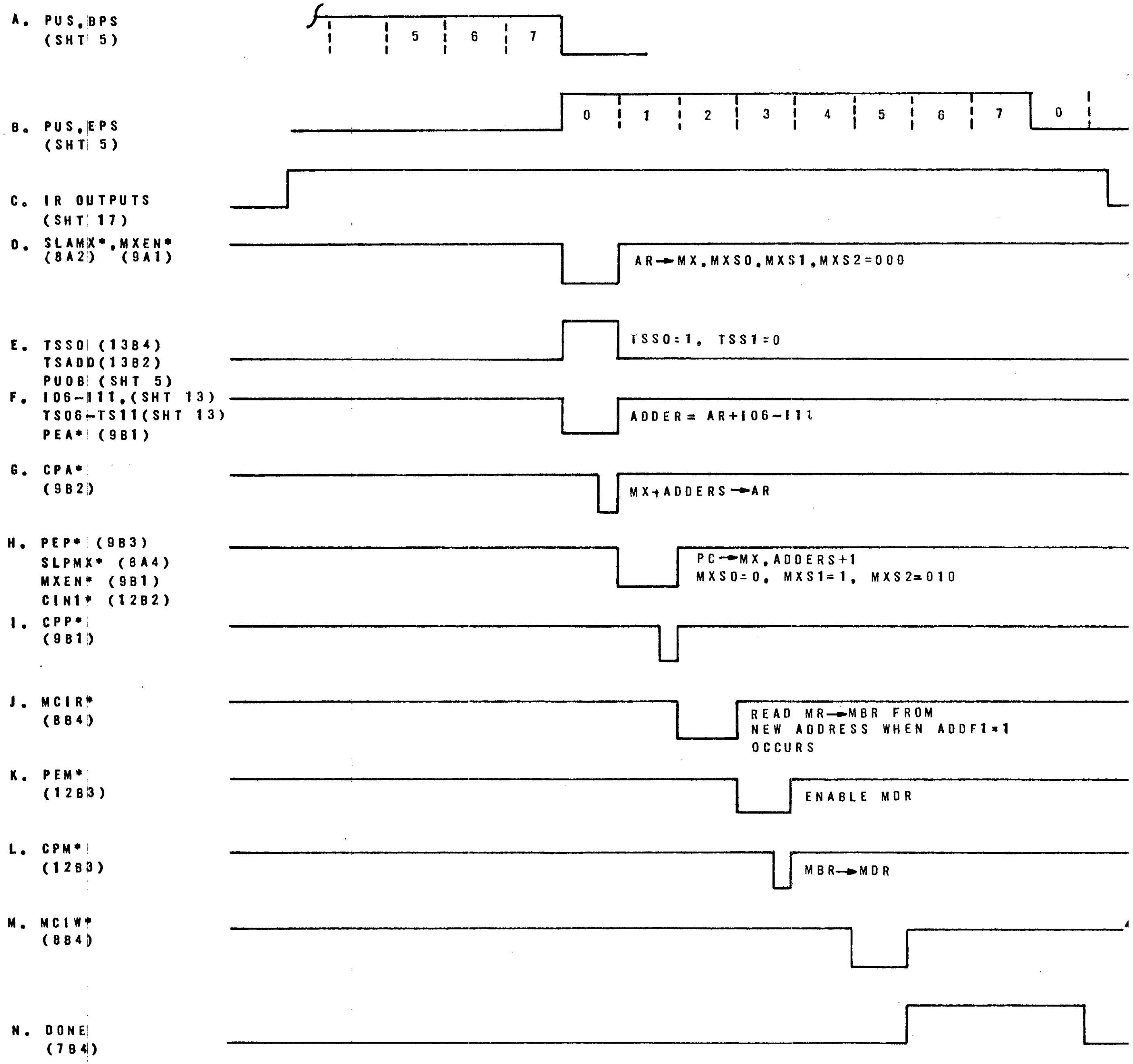


Figure 4-55. Single-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Timing Diagram

Table 4-54. Single-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Event Summary

Ref.	Period	Event
A	EP0	AR→MX→ADDER
B	EP0	REL→TS→ADDER
C	EP0	SELECT→(MX + TS)
D	EP0	SELECT→(MX - TS)
E	PU0B	ADDER→AR
F	PU1	CLR→MBR
G	EP1	PC→MX→ADDER
H	EP1	+1→ADDER
I	EP1	ADDER→PC
J	PU2	MR→MBR
K	PU3	MBR→MDR
L	PU5	MDR→MR
M	EP6	SET→DONE LATCH
N	PU7	SET→BASIC PHASE (only)

Table 4-55. Single-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	EP0	AR→MX→ADDER 1. (↑EP0) (↑FLLOC*) (↑MPASS)→ ↓SLAMX*/8A2→↑MXEN→↓MXEN*/9A2 = AR→MX→ADDER	The content of the address register is switched through the MX multiplexer to the adder.
B.	EP0	REL→TS→ADDER 1. (↑MPASS) (↑EP0)→↓SLITS*→↑TSS0/13B4 = (I06 - I11)→TS MPLEXER	The relative address is switched through the TS multiplexer to the adder to establish the effective address.
C.	EP0	(MX+TS)→ADDER 1. (↑MPASS) (↑I5B*)→↓ADDAR*→ ↑ADDEP0/13B1 2. (↑ADDEP0) (↑EP0) (↑FFLOC*)→ ↓TSADD*/13B2 = (IR + AR)→ADDER	Addition is selected for summation to establish the effective address.
D.	EP0	(MX - TS)→ADDER 1. (↑MPASS) (↑EP0) (↑FLLOC*) (↑I05X)→ ↓TSSUB*/13B3 2. ↓TSSUB*→↑TSSUB/13B4 3. ↓HWD + ↓ONCE→↓DCIN*/12A1 (see A and C, Table 4-28) 4. (↑DCIN*) (↑TSSUB) (↑OP2*)→↑CIN2*→ ↑CIN/12A2 = (AR-IR)→ADDERS	Subtraction is selected for summation to establish the effective address.

Table 4-55. Single-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
E.	PU0B	ADDER→AR 1. ↑PU0B→↓PEA*→↑PEA/9B1 2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = ADDER→AR	Content of adder replaces content of address register to become the pointer address for the direct fetch.
F.	PU1	CLR→MBR 1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	Memory buffer register cleared.
G.	EP1	PC→MX→ADDER 1. ↑EP1→↓SLPMX*/8B4 2. ↓SLPMX*→↑MXS1/9B2 = SELECT→PC 3. ↓SLPMX*→↑MXEN→↓MXEN*/9A2 = PC→MX→ADDER	Content of program counter switched through MX multiplexer to adder.
H.	EP1	+1→ADDER 1. (↑EP1) (↑JMP*)→↓CIN1*→↑CIN/12A2 = ADDERS→+1	Adder is incremented by one.
I.	EP1	ADDER→PC 1. ↑EP1→↓PEP*→↑PEP/9B1 2. (↑PEP) (↑REGCLK)→↓CPP*/9B2 = ADDERS→PC	Content of adder replaces content of program counter.
J.	PU2	MR→MBR 1. (↑ADDF0*) (↑PU2)→↓MCIR*/8B4 = MR→MBR (DIRECT OPERAND)	Content of memory register replaces content of memory buffer register.
K.	PU3	MBR→MDR 1. ↓PU3*→↑PEM/12B2 2. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = MBR→MDR	Content of memory buffer register replaces content of memory data register.
L.	PU5	MDR→MR 1. (↑WTRL*) (PU5B)→↓PU5W*→↑PU5W/8B3 2. (↑PU5W) (↑ADDF0*)→↓MCIW*/8B4 = MDR→MR	Content of memory data register replaces content of memory register.

Table 4-55. Single-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
M.	EP6	SET→DONE 1. ↓TWHLT* + ↓I/O* + ↓ADL* + ↓SBL* + ↓ANL* + ↓OP2* + ↓OP1*→↑SDONE/7B4 2. (↑SDONE) (↑BP6)→↓SDONE*/7B4 3. ↓SDONE*/7B4 = SET→DONE (↑DONE, ↓DONE*/7B4) 4. ↓EP6*/5A4 = SET→DONE (↑DONE, ↓DONE*/7B4)	Done flip-flop is set by EP6*.
N.	PU7	SET→BASIC PHASE 1. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→ ↑RBPH→↓RBPH*/5B1 2. (↓RBPH*) (↓CPPU*) (↑PU7) = SET→BASIC PHASE (↑BPH/5B2)	Next phase is set to a basic phase.

by a count of 1 and returned to the program counter via busses B00 through B11. The program counter (10) now contains the next instruction address, and the address register (9) contains the effective address of the operand, the current address plus or minus the relative address (CA ± RA = EA). During this time, pulse PU1B also clears the memory buffer register (1).

Pulses PU2 and PU3 permit effectively addressed memory register data to be transferred from the memory register to the memory buffer register (1) and from the memory buffer register (1) to the memory data register, through the bus and memory buffer multiplexer (3). However, the current address, formerly in the address register (9) has now been modified by the events that occurred during period EP1.

Timing Description - Instruction register data, which determines processor operations during the execute phase, is available after period BP3 of the basic phase. Instruction register data (signals I00 through I03) is applied to the instruction decoders (sheets 7 and 11) and to the TS multiplexer (signals I06 through I11).

When period EP0 occurs (waveform B, Figure 4-55), a number of simultaneous events occur (waveforms E through H) that permit bits 6 through 11 of the instruction word to be summed with the current address to obtain the effective address. Pulse EP0 is ANDed with the high-level FLLOC* signals from the field-locate latch (A, Table 4-55) and the high-level MPASS signal produced by the cleared two-word and indirect latches to produce the low-level SLAMX* signal (waveform D) which permits the address data inputs, on lines A00 through A11, to be switched through the MX multiplexer to the adder. Simultaneously, effective address bits I06 through I11 are switched through the TS multiplexer when high-level signal TSS0 (waveform F) is produced.

A third simultaneous event occurs when pulse PU0B admits outputs B00 through B11 from the adder to the address register when low-level signals PEA* and CPA* (waveforms G and H) are produced. Signal PEA* enables the address register and signal CPA* strobes adder output data into the address register. Thus, the current address is modified by the addition of the content of bits 6 through 11 of the instruction to obtain the effective address and the effective address is then loaded into the address register and applied to the memory address decoders.

Three simultaneous events also occur when pulse EP1 is generated, the content of the program counter is admitted on lines P00 through P11 into the MX multiplexer when signals SLPMX* and MXEN* are produced. These signals (waveform H) select program register outputs P00 through P11, and enable the MX multiplexer. Also, carry input, signal C1N1*, is propagated through the adders (except when the JMP instruction is processed) which receive the MX multiplexer outputs, when the program counter data is admitted into the MX multiplexer. And finally, the adder data, which is carried on busses B00 through B11, is reloaded into the program counter by signals PEP* and CPP*. Signal PEP* enables the program counter and signal CPP* (waveforms H and I) loads the program counter. Also during this time, pulse PU1B clears the memory buffer register when pulse RMB* is generated. The function of this pulse is described under basic cycle. The program counter now contains the address of the next instruction that will be fetched from memory.

Pulses PU2 and PU3 (waveforms J, K, and L) produce the same events as occurred during the basic phase when these pulses were generated. Their function and purpose is described under basic cycle.

When pulse PU5B occurs the content of the memory data register is strobed back into the memory location specified by the content of the address register (waveform M). Pulse EP6 sets the done latch (waveform N), which in turn clears the two-word, indirect, and first-last latches. The done latch is cleared again when pulse BP1 is generated when the next instruction is processed. Thus, during the execute phase, the events can be summarized as follows.

a. The content of the address register is switched through the MX multiplexer to the adders and summed with the content of the TS multiplexer, bits 6 through 11 of the instruction, and stored into the address register.

b. The content of the program counter is incremented by one count.

c. The content of the memory data register is written into the memory location specified by the content of the address register.

d. The major state control logic is set to the basic phase.

The content of bit 5 of the instruction word is tested during the execute phase when pulse EP0 is produced (C, Table 4-55). Thus, at the time low-level signal ADDAR* is produced, when high-level signals MPASS and I5B* are ANDed, this function is inhibited when bit 5 is set because signal I5B* becomes low-level. Since the low-level I5B* signal inhibits signal ADDAR*, the addition of bits I00 through I06 is not made, but subtraction (D, Table 4-55) is carried out instead.

Although the ADDAR* signal is inhibited by the set state of instruction bit 5, the TSSUB* signal is enabled by high-level signal 105X. Subtraction occurs in 2's complement notation. This subtraction results in the relative address, specified by bits 6 through 11 of the instruction word, being subtracted from the current address. All other events that occur during the basic and early execute phases remain the same.

4.4.3.5 INDIRECT ADDRESSING. There are two basic phases and one execute phase when an indirect memory reference instruction is processed. Tables 4-56, 4-57, and 4-58 are the Event Summary for operations that occur when an indirectly fetched operand is specified by the memory reference instruction being processed. The basic phases are subdivided into a primary basic phase and a secondary basic phase. The events that occur during the primary basic phase are identical to those that occur during the common basic phase, but with modified events. The modified events are produced when bit 4 of the instruction word is set, indicating indirect addressing; also, the execute phase is not set. As a consequence, the indirect latch is set so that when period PU7 occurs, a second basic phase will be enabled. During the secondary basic phase, most of the events described for the common execute phase occur. During the preceding basic phase the done latch is not set because there is no latch-setting input to the done-latch logic at the conclusion of a basic phase when a single-word memory reference instruction is processed. This is verified by reference K, Table 4-20, operation No. 1. Thus, because the done latch remains unset, the instruction register is not cleared during period PU0 of the ensuing basic phase. This is verified by reference D, Table 4-20; the done latch must be set to clear the instruction register. Hence, data stored in the instruction register is available to processing logic during the secondary basic phase. Setting of the indirect latch at period BP4 of the primary basic phase defers an execute phase in favor of a secondary basic phase. This is verified by reference K, Table 4-20, operation No. 3. One of the conditions that establishes a basic phase is a set indirect latch which produces a low-level IND* signal. Hence, at the conclusion of the primary basic phase, the secondary basic phase is entered. The indirect latch is cleared during the secondary basic phase so that at its conclusion, the execute phase can be set. Hence, the content of the instruction register remains unaltered because the events that normally fetch data into it are inhibited. As shown in Table 4-57 fundamental operations G, H, and I of the common execute phase are not carried out during the secondary basic phase.

4.4.3.5.1 Primary Basic Phase. Table 4-59 lists the Fundamental Operations and Figure 4-54 is the block diagram and Figure 4-56 is the timing diagram for the signals generated when the memory reference instruction names an indirect fetch through the set state of bit 4 in the instruction word. By comparing waveforms A through H of Figure 4-15 with those of Figure 4-54, it can be seen that for periods BP0 through BP3, the events that occur are identical. When pulse BP4 is generated, the similarities in processing temporarily disappear. Table 4-60 is the list of Fundamental Operations for the primary basic phase of a memory reference instruction indirect fetch. These logic descriptions begin with period BP4 of the primary basic phase. High-level signal MREFI is produced whenever a memory reference instruction is called for in a program. It can be seen that when bit 4 is set, high level signals MREFI, 104B, and INDFF (produced by the previously cleared INDFF latch), are ANDed to produce the set state of the IND and INDFF latches (Figure 4-56, waveforms I

Table 4-56. Single-Word Memory Reference Instructions, Indirect Addressing, Primary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	PU7	CLR→LATCHES	A
B	BP0	PC→MX→ADDER	B
C	PU0B	ADDER→AR	C
D	PU0B	CLR→IR	D
E	BP1	CLR→DONE	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	MBR→IR	I
J	BP4	SET→IND, INDFF	None
K	PU5	MDR→MR	J
L	PU7	SET BASIC PHASE	L

Table 4-57. Single-Word Memory Reference Instructions, Indirect Addressing, Secondary Basic Phase, Event Summary

Ref.	Period	Event	Common Execute Phase Ref. (Table 4-55)
A	BP0	AR→MX→ADDER	None
B	BP0	REL→TS→ADDER	B
C	BP0	SELECT→(MX + TS)	C
D	BP0	SELECT→(MX - TS)	D
E	PU0B	ADDER→AR	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	J
H	PU3	MBR→MDR	K
I	PU3	CLR→IND LATCH	None
J	PU5	MDR→MR	L
K	PU7	SET→EXECUTE PHASE (only)	N

Note: Operations G, H, I and M of the Common Execute Phase are not carried out; hence, no program counter modification results until the execute phase.

and J). The set state of the INDFF latch produces low-level signal INDFF*, which inhibits the production of high-level signal MPASS (reference I, operation No. 1, Table 4-20). Inhibiting high-level signal MPASS prevents the content of the memory buffer register from being fetched into the instruction register during the subsequent basic phase because the operand is already in the MDR.

When period PU7 occurs, the set state of the IND latch produces a low-level IND* signal which permits the major-state control register to again be reset to its basic-phase state as described by the logic of reference L, Table 4-59. Thus, the events that take place during the primary basic phase are:

Table 4-58. Single-Word Memory Reference Instructions, Indirect Addressing, Execute Phase, Event Summary

Ref.	Period	Event	Common Execute Phase Ref. (Table 4-55)
A	EP0	MDR→TS→ADDER	None
B	PU0	ADDER→AR	E
C	PU1	CLR→MBR	F
D	EP1	PC→MX→ADDER	G
E	EP1	+1→ADDER	H
F	EP1	ADDER→PC	I
G	PU2	MR→MBR	J
H	PU3	MBR→MDR	K
I	PU3	MDR→MR	L
J	EP6	SET→DONE LATCH	M
K	PU7	SET→BASIC PHASE (only)	N

a. The content of the program counter is loaded into the address register.

b. The content of the memory register located at the address specified by the content of the address register is read into the memory buffer register.

c. The content of the memory buffer register is loaded into both the memory data register and into the instruction register.

d. The indirect latch is set to prepare for a latter entry into the secondary basic phase.

4.4.3.5.2 Secondary Basic Phase. Table 4-60 is the list of Fundamental Operations for the secondary basic phase. The timing diagram is essentially the same as for the common execute phase (Figure 4-55), with the difference that functions of the secondary basic phase are based on the BP pulse train (Figure 4-56). As previously described for the primary basic phase, the high-level MPASS signal is inhibited by a low-level INDFF* signal, indicating indirect addressing. Because the high-level MPASS signal is the primary control signal for initiating the arithmetic, which leads to obtaining the effective address from the current address and the indirect pointer address for the execute phase, another signal must take over when the above address modification is called for during a basic phase. This signal, MLTAS* is generated, as shown by operation 3, reference B, Table 4-60.

Operation 1 of reference I, Table 4-60 shows an operation that is actually carried out during other cycles too but in those it finds the indirect latch already in the clear state. In the present case, however, by clearing the set state of the indirect latch, the subsequent execute phase is established. The other difference is that waveforms associated with Fundamental Operations G, H, and I of the common execute phase are not generated, because those operations are not carried out by the processing logic. These operations normally provide for the incrementation of the program counter, however, it is premature to carry out

Table 4-59. Single-Word Memory Reference Instructions, Indirect Addressing, Primary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-I	BP0-BP3	These operations are identical with operations A through I of the Common Basic Phase (Table 4-20)	Common Basic Phase
J.	BP4	SET→IND INDFF LATCHES 1. ↓SID* + ↓DSZ* + ↓ISZ*→↑ISDSS/7A2 2. ↑ISDSS→↓ISDSS*/7A2 3. ↓ISDSS* + ↓XCT* + ↓SBJ* + ↓ADJ* + ↓JPS* + ↓JMP* + ↓LDJ* + ↓STJ*→ ↑MREFI/7A3 4. ↑MREFI→↓MREFI*/7A3 5. (↑MREFI) (↑I04B) (↑INDFF*)→↓PREND/6A1 6. ↓PREND→↑PREND*/6A1 7. (↑PREND*) (↑BP4) (↑REGCLK)→↓SIND*/6A2 8. ↓SIND* = SET→IND (↑IND, ↓IND*/6A2) 9. ↓SIND*/6A2 = SET→INDFF (↑INDFF/6A3, ↓INDFF*/6A3)	The indirect fetch is initialized.
K.	PU5	MDR→MR 1. (↑WTRL*) (↑PU5B)→↓PU5W*→↑PU5W/8B3 2. (↑PU5W) (↑ADFF0*)→↓MCIW*/8B4 = MDR→MR	Content of memory data register replaces content of memory register.
L.	PU7	SET→BPH 1. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→ ↑RBPH/5B1 2. ↑RBPH→↓RBPH*/5B1 3. (↓RBPH*) (↓CPPU*) (↑PU7)→↑BPH/5B2, ↓EPH = SET→BPH	Next phase is set to a basic phase.

that operation during this subcycle; it is deferred until the execute phase. That operations G, H, and I (Table 4-55) are inhibited can be seen at once because they all require timing signal EP1 which cannot be generated during a basic phase.

When period PU7 occurs, the cleared state of the indirect latch, in conjunction with the coincidental cleared states of the done, two-word, and execute latches, produces a low-level RBPH signal (reference K, Table 4-60), which permits the major-state control register to be set to the execute state. The events that take place during the secondary basic phase are:

a. The content of the address register is switched through the MX multiplexer to the adder and summed with the content of the TS multiplexer, bits 6 though 11 of the instruction register, and stored into the address register. The address register now contains the relative address.

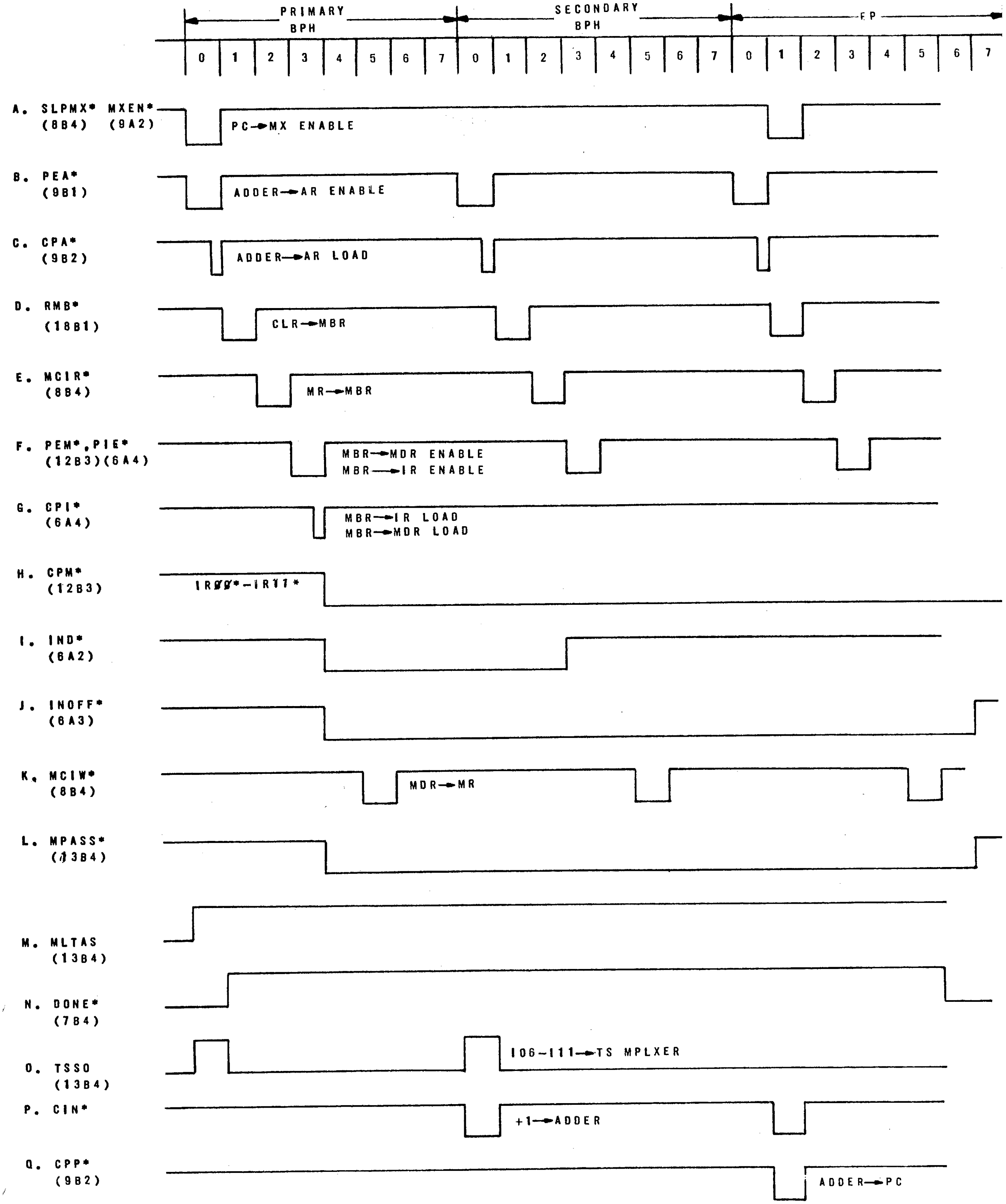


Figure 4-56. Single-Word Memory Reference Instructions, Indirect Addressing, Timing Diagram

Table 4-60. Single-Word Memory Reference Instructions, Indirect Addressing, Secondary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BPO	AR→MX→ADDER 1. ↓XCTFF + ↓IND* + ↓INDFF→ ↑XIFF*/6A3 2. ↓IND* + ↓TWFF* + ↓XCT*→↑XITFF/8A1 3. ↓TWFF + ↓IND→↑TIFF*/8A1 4. (↑XITFF) (↑XIFF*) (↑TIFF*)→↓SLAMU*/8A1 5. ↓SLAMU*→↑SLAMU/8A1 6. (↑SLAMU) (↑FLLOC*) (↑BPO)→ ↓SLAMX*/8A2 = ENABLE→MX MULTIPLEXER 7. ↓SLAMX*/8A2→↑MXEN/9A2→ ↓MXEN*/9A2 = AR→MX→ADDER	The content of the address register is switched through the MX multiplexer to the adder to obtain the current address.
B.	BPO	REL→TS→ADDER 1. ↓XCTFF + ↓IND* + ↓INDFF→↑XIFF*/6A3 2. (↑XIFF*) (↑TWFF*)→↓MLTAS*/6A4 3. ↓MLTAS*→↑MLTAS/13B4 4. (↑MLTAS) (↑BPO)→↓SLITS*→↑TSS0/13B4 = IR→TS→ADDER	Bits I06 through I11 are switched through the TS multiplexer to the add-subtract logic and the adder.
C.	BPO	SELECT→(MX+TS)→ADDER(Bit 5 = 0) 1. (↑MLTAS) (↑I5B*)→↓MULIS*→ ↑MSCTAD/13B1 2. (↑FLLOC*) (↑DONE*)→↓FLDN→ ↑FLDN*/13B2 3. (↑FLDN*) (↑MSCTAD) (↑BPO)→ ↓TSADD*/13B2 = (IR+AR)→ADDER	The relative address is added to the current address to establish the effective address.
D.	BPO	SELECT→(MX-TS)→ADDER (Bit 5 = 1) 1. (↑FLLOC*) (↑DONE*)→↓FLDN→ ↑FLDN*/13B2 2. (↑FLDN*) (↑MLTAS) (↑I05X) (↑BPO)→ ↓TSSUB*→↑TSSUB/13B3 3. ↓HWD + ↓ONCE*→↑DCIN*/12A1 (see A and C, Table 4-28) 4. (↑DCIN*) (↑TSSUB) (↑OP2*)→↓CIN2*/12A2→ ↑CIN/12A2 = +1→ADDER = (AR−IR)→ADDERS	The relative address is subtracted from the current address to establish the effective address.
E.	PU0B	ADDER→AR 1. ↑PU0B→↓PEA*/9B1→↑PEA/9B1 = ENABLE AR 2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = ADDERS→AR	The effective address is admitted into the AR to become the pointer address for indirect fetch.

Table 4-60. Single-Word Memory Reference Instructions, Indirect Addressing, Secondary Basic Phase, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
F.	PU1	CLR→MBR	
		1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	The memory buffer register is cleared.
G.	PU2	MR→MBR	
		1. (↑ADDF0*) (↑PU2)→↓MCIR*/8B4 = READ MR→MBR (Indirect Address→MBR)	The indirect address is loaded into the memory buffer register.
H.	PU3	MBR→MDR	
		1. ↓PU3*/5B2→↑PEM/12B2 = ENABLE→MDR 2. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = MBR→MDR	The content of the memory buffer register is loaded into the memory data register.
I.	PU3	CLR→IND LATCH	
		1. ↓CPM* = CLR→IND (↓IND/6A2, ↑IND*)	The indirect fetch is cleared.
J.	PU5	MDR→MR	
		1. (↑WTRL*) (↑PU5B)→↓PU5W*→ ↑PU5W/8B3 2. (↑PU5W) (↑ADDF0*)→↓MCIW/8B4 = MDR→MR	Content of memory data register replaces content of memory register.
K.	PU7	SET→EXECUTE PHASE	
		1. (↑DONE*) (↑TW*) (↑IND*) (↑XCT*)→ ↓RBPH→↑RBPH*/5B1 2. (↑RBPH*) (↑PU7) = SET→EPH/5B2	The execute phase is set.

b. The indirect latch is cleared.

c. The content of the memory data register is written into the memory location specified by the content of the address register.

d. The execute state is set for the next phase.

4.4.3.5.3 Indirect Execute Phase. The indirect execute phase is similar to the common execute phase (Tables 4-54 and 4-55). The pulser supplies PU and EP pulses; therefore, all operations generated for the common execute phase occur. The indirect execute phase can be considered a degenerate common execute phase because some functions normally carried out during the common execute phase, are performed during the secondary basic phase. Thus, upon examination of Table 4-57, fundamental operations A, B and E are not necessary because effective addressing took place during the previous secondary basic phase. In this case, the low-level MPASS signal, orders bypassing of the first two fundamental operations, and is used to select the summation operation for the TS multiplexer. All other steps throughout the execution of this execute phase, when an indirect memory reference instruction is processed, are identical to those of the common execute phase. At the conclusion of this execute phase, pulse PU7 resets the major-state control register to a basic phase again, readying the ND812 for the next instruction. Fundamental Operations for this phase are listed in Table 4-61. For timing of the indirect execute phase refer to Figure 4-56, keeping in mind the above described modifications.

During the indirect execute phase, the events that occur can be summarized as follows.

a. The content of the memory data register, the indirect address, is switched through TS multiplexer to the adder and into the address register.

b. The content of the program counter is incremented by one count.

c. The content of the memory data register is written into the memory location specified by the content of the address register.

d. The basic phase is set for the next instruction.

4.4.4 TWO-WORD MEMORY REFERENCE INSTRUCTIONS

There are fourteen two-word memory reference instructions. Two-word memory reference instructions comprise two consecutive 12-bit words. The first word contains the operation fields and the second word contains the absolute 12-bit address. Only absolute and indirect addressing is applicable for the two-word memory reference instructions. The absence of relative addressing does not imply a limitation; it is simply not necessary because any location in a 4K memory can be accessed by absolute addressing through the 12-bit address of the second word.

Comparison of the single and two-word memory reference instructions reveals other similarities and differences. Indirect addressing is possible and has the same meaning for either instruction; the initial address does not contain the operand but the location from where the operand is to be fetched. The displacement field and sign-field is not valid for the two-word memory instruction. The fourteen two-word memory reference instructions are listed in Table 4-62.

Table 4-63 lists the two-word memory reference instruction format and bit patterns. The most important difference between single- and two-word memory reference instructions is the capability of the latter to address memory fields other than a single 4K memory. This control is reflected by bits 9, 10 and 11 of the instruction format. If bit 9 is set, processor logic is notified that a memory field change is required for the instruction. In this case bits 10 and 11 are decoded. This 2-bit field-change address provides for selection of four possible

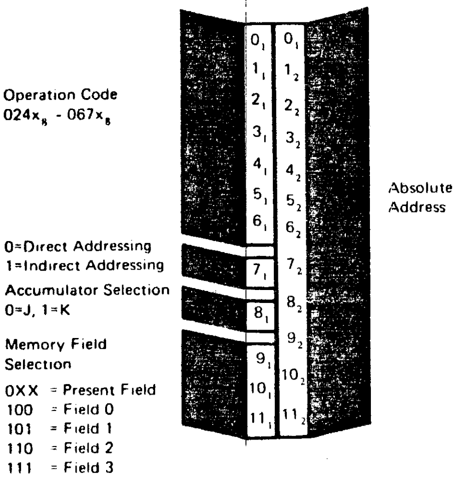
Table 4-61. Single-Word Memory Reference Instructions, Indirect Addressing, Execute Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	EP0	MDR→TS→ADDER 1. ↓TWFF* + ↓INDFF*→↑MPASS*→ ↓MPASS/13B4 2. ↓MPASS→↑ADDEP0/13B1 3. (↑ADDEP0) (↑FLLOC*) (↑EP0/5A4)→ TSADD*/13B2 = MDR→TS→ADDER	The indirect pointer address is switched through the TS multiplexer to the adder.
B.	PU0	ADDER→AR 1. ↑PU0B→↓PEA*→↑PEA/9B1 = ENABLE→AR 2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = ADDER→AR	The content of the adder is admitted into the address register.
C.	PU1	CLR→MBR 1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	The memory buffer register is cleared.
D.	EP1	PC→MX→ADDER 1. ↑EP1→↓SLPMX*/8B4 2. ↓SLPMX*→↑MXS1/9B2 = ENABLE→PC 3. ↓SLPMX*→↑MXEN→↓MXEN*/9A2 = PC→MX→ADDER	The content of the program counter is switched through the MX multiplexer to the adder.
E.	EP1	+1→ADDER 1. (↑EP1) (↑JMP*)→↓CIN1*→↑CIN/12A2 = +1→ADDER	The content of the adder is incremented by a count of one.
F.	EP1	ADDER→PC 1. ↑EP1→↓PEP*→↑PEP/9B1 2. (↑PEP) (↑REGCLK)→↓CPP*/9B2 = ADDERS→PC	The content of the adder is admitted to the PC.
G.	PU2	MR→MBR 1. (↑ADDF0*) (↑PU2)→↓MCIR*/8B4 = READ MR→MBR	The content of the memory register (the indirect operand) is admitted into the memory data register.
H.	PU3	MBR→MDR 1. ↓PU3*→↑PEM/12B2 2. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = MBR→MDR	The content of the memory buffer register is switched through the bus and memory buffer multiplexer to the memory data register.
I,J,K	PU5, EP6, PU7	Same as references L, M, and N, Table 4-55.	

Table 4-62. Two-Word Memory Reference Instructions

Subgroup	Mnemonic	Octal Code	Operation
Two-Word MRI	TWSMJ	0240	Skip if J ≠ memory
	TWSMK	0250	Skip if K ≠ memory
	TWDSZ	0300	Decrement memory and skip if = 0
	TWISZ	0340	Increment memory and skip if = 0
	TWSBJ	0400	Subtract memory from J
	TWSBK	0410	Subtract memory from K
	TWADJ	0440	Add memory to J
	TWADK	0450	Add memory to K
	TWLDJ	0500	Load memory into J
	TWLDK	0510	Load memory into K
	TWSTJ	0540	Store J in memory
	TWSTK	0550	Store K in memory
	TWJMP	0600	Jump unconditionally
	TWJPS	0640	Jump to subroutine

Table 4-63. Two-Word Memory Reference Instruction Format and Bit Patterns



This is the two-word memory reference instruction format. When bits 0, 1, and 2 are zero a two-word memory reference instruction is named. Bits 3 through 6 determine the operation. Bit 7 determines whether the next word is a direct or indirect address. Bit 8 determines whether the J or K register is affected. Bits 9, 10, and 11 determine the field in which the second-word address is located.

Octal		Bit Pattern											
Code	OPERATION	0	1	2	3	4	5	6	7	8	9	10	11
024X	TWSMJ					1		1					
025X	TWSMK					1		1		1			
030X	TWDSZ					1	1						
034X	TWISZ					1	1	1					
040X	TWSBJ				1								
041X	TWSBK				1					1			
044X	TWADJ				1			1					
045X	TWADK				1			1		1			
050X	TWLDJ				1		1						
051X	TWLDK				1		1			1			
054X	TWSTJ				1		1	1					
055X	TWSTK				1		1	1		1			
060X	TWJMP				1	1							
064X	TWJPS				1	1		1					

4K memory fields, including the current memory field. The current memory field is selected by setting bit 9 to a zero; under this condition, the content of bit 10 and 11 are irrelevant. In general, when a memory field change is ordered by the instruction, the program returns to the originating field after execution of the instruction. Exceptions to this rule are the jump (TWJMP) and the jump-to-subroutine (TWJPS) instruction. These two instructions result in a permanent field-change; their effect can be reversed only by another jump command. For all non-jump commands, the field selection change is only for the execute portion of that instruction.

4.4.4.1 FIELD CHANGE. Memory field change and return is effected by the memory field control logic, which is described in paragraph 4.5.4.2.

As can be seen from the descriptions of the single-word memory reference instructions, they make use of only one of the main accumulators, the J-register.

The two-word instructions make use of the K-register also. In order to specify one of the main accumulators uniquely, the two-word instruction format provides for this selection. If bit 8 is set, the K register is employed in the specified operation. Of course, this is only applicable if the accumulator takes an active part in the data transfer; the TWISZ and ISZ instructions do not make use of the upper accumulators.

From the point of view of processing, cycle duration, etc., the main difference between single- and two-word memory reference instruction is in the basic timing. Direct addressing for one-word MRI's requires one basic and one execute phase, but direct addressing for two-word MRI's requires an additional basic phase. The reason for this is that in the two-word format, the first basic cycle encounters only the first word of the instruction; thus the second word, containing the absolute address, has to be brought back from memory in order to locate the operand. Indirect addressing for single- and two-word MRI's carries the same timing relationship. One-word MRI's require two basic and one execute phase for the indirect addressing mode; two-word MRI's require three basic and one execute phase for the same operation.

Bits 0, 1 and 2 are always set to zero for a two-word instruction; they serve to indicate to the instruction decode logic that a two-word memory reference instruction is being processed. Bits 3 through 6 contain the instruction code. These codes are identical with the codes for the single word memory reference instructions. The only difference is in their location relative to the instruction format. However, because the two-word instructions are effectively shifted three positions to the left, the instruction field and the direct-indirect fields become coincident with the same fields of the single-word instruction. The effective three-bit shift operation makes it possible to process two-word instructions through the same logic circuits that are used to process single-word instructions. Bit 7 contains the indirect bit and bit 8 contains the J/K accumulator select bit. Bit 9 is the field-change select bit which, in conjunction with bits 10 and 11, establish the memory-field selection address. It should be remembered when specifying bit patterns for two-word memory reference instructions that the bit positions have no significance until the effective 3-bit shift occurs. Hence, the two-word and single-word formats are identical for processing.

4.4.4.2 DIRECT ADDRESSING. The directly addressed two-word memory reference instruction has two basic phases and one execute phase. Tables 4-64, 4-65 and 4-66 are the Event Summaries for the three phases, respectively. Table 4-67 is a list of Fundamental Operations for the primary basic phase, Table 4-68 is a list of Fundamental Operations for the secondary basic phase, and with only a single exception, Table 4-69 is a list of the Fundamental Operations for the two-word memory reference instruction execute phase. Refer to Figure 4-54 for the block diagram, and to Figure 4-57 for the timing diagram.

Table 4-64. Two-Word Memory Reference Instructions, Direct Addressing, Primary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	PU7	CLR→LATCHES	A
B	BP0	PC→MX→ADDER	B
C	PU0B	ADDER→AR	C
D	PU0B	CLR→IR	D
E	BP1	CLR→DONE	E
F	PU1B	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	MBR→IR	I
J	BP5	SET→TW LATCH	None
K	PU5	MDR→MR	J
L	PU7	SET→BASIC PHASE	L

Table 4-65. Two-Word Memory Reference Instructions, Direct Addressing, Secondary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	BP0	AR→MX→ADDER	None
B	BP0	+1→ADDER	None
C	BP0	ADDER→PC	None
D	PU0B	ADDER→AR	C
E	BP1	CLR→DONE	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	SHIFT→IR	None
J	PU5	MDR→MR	J
K	PU7	SET→EXECUTE PHASE	L

Table 4-66. Two-Word Memory Reference Instructions, Direct Addressing, Execute Phase Event Summary

Ref.	Period	Event	SING WD Execute Phase Ref. (Table 4-61)
A	EP0	MDR→TS→ADDER	A
B	PU0	ADDER→AR	B
C	PU1	CLR→MBR	C
D	EP1	PC→MX→ADDER	D
E	EP1	+1→ADDER	E
F	EP1	ADDER→PC	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	EP3	CLR→TW LATCH	None
J	PU5	MDR→MR	I
K	EP6	SET→DONE LATCH	J
L	PU7	SET→BASIC PHASE (only)	K

Table 4-67. Two-Word Memory Reference Instructions, Direct Addressing, Primary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-I	EP6- BP3	Identical with A through I of Table 4-20.	See Table 4-20.
J.	BP5	SET→TW LATCH 1. ↓I03B + ↓I04B + ↓I05B + ↓I06*→↑70*/3B1 2. ↓I03* + ↓I04B + ↓I05* + ↓I06*→↑20/3B1 3. ↓I03* + ↓I04*→↑01 + 00*/3B1 4. (↑70*) (↑20) (↑01 + 00*)→↓HLT/3B2 5. ↓HLT→↑HLT*/3B2 6. (↑TW) (↑HLT*)→↓HLTTW*→↑HLTTW/3B3 7. (↑HLTTW) (↑REGCLK) (BP5)→↓STWFF* = SET TWO-WORD LATCH (↑TWFF/3B4, ↓TWFF*)	The two-word latch is set.
K,L	PU5, PU7	Identical with J and L, respectively, Table 4-20.	See Table 4-20.

4.4.4.2.1 Primary Basic Phase. The primary basic phase is identical with the common basic phase (Table 4-20) with one exception. During period BP5 of the two-word memory reference instruction primary basic phase, the two-word latch is set; this operation is described as operation J in Table 4-67. Prior to the shift, bits I03, I04, I05, and I06 are tested. If they are all inactive, they are decoded as a two-word instruction and the two-word latch is set. At the end of the primary basic phase, during period BP6, the basic phase is entered again. The low-level signal TW* permits the basic phase to be entered again (Operation L3, Table 4-20). This signal is asserted whenever instruction register bits I00 through I03

Table 4-68. Two-Word Memory Reference Instructions, Direct Addressing, Secondary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP0	AR→MX→ADDER 1. ↓XCTFF + ↓IND* + ↓INDFF→↑XIFF*/6A3 2. ↓IND* + ↓TWFF* + ↓XCT*→↑XITFF/8A1 3. ↓TWFF + ↓IND→↑TIFF*/8A1 4. (↑XITFF) (↑XIFF*) (↑TIFF*) →↓SLAMU*/8A1 5. ↓SLAMU*→↑SLAMU/8A1 6. (↑SLAMU) (↑FLLOC*) (↑BP0)→ ↓SLAMX*/8A2 =ENABLE→MX MULTIPLEXER 7. ↓SLAMX*/8A2→↑MXEN/9A2→ ↓MXEN*/9A2 = AR→MX→ADDER	The content of the address register is switched through the MX multiplexer to the adder to obtain the current address.
B.	BP0	+1→ADDER 1. (↑TWFF) (↑IND*) (↑BP0)→↓CIN1*→ ↑CIN/12B2	The adder is incremented by a count of one.
C.	BP0	ADDER→PC 1. (↑XCTFF*) (↑INDFF*)→↓XIFF/6A3 2. ↓XIFF→↑X*I*FF/6A4 3. ↓STDN7*→↑INDM*/6A3 4. (↑INDM*) (↑TWFF) (↑X*I*FF) (↑BP0)→ ↓PEP*/9B3 5. ↓PEP*→↑PEP/9B1 = ENABLE→PC 6. (↑PEP) (↑REGCLK)→↓CPP*/9B2 = LOAD→PC	Content of the adder is loaded into the program counter.
D.	PU0B	ADDER→AR 1. ↑PU0B→↓PEA*→↑PEA/9B1 = ENABLE→AR 2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = LOAD→AR	Content of the adder is loaded into the address register.
E-H	BP1- PU3	Identical with operations E through H of Table 4-20.	See Table 4-20.
I.	BP3	SHIFT→IR 1. ↓IND* + ↓XCT→↑XCTND*/6A3 2. (↑XCTND*) (↑MPASS*)→↓SPEI*/6A3 3. ↓SPEI* + ↓BP3→↑PEI*/6A4 = DISABLE→IR 4. (↑BP3) (↑TWFF) (↑IND*)→↓SHIFT3*/6A4 5. ↓SHIFT3*→↑PEI/6A4 6. (↑PEI) (↑CCLK)→↓CPI*/6A4 = SHIFT→IR	The content of the instruction register is effectively shifted 3-bits to the left.
J,L	PU5, PU7	Same as operations J, and L of Table 4-20.	See Table 4-20.

Table 4-69. Two-Word Memory Reference Instructions, Direct Addressing, Execute Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-H	EP0- PU3	Identical with A through H of Table 4-62.	See Table 4-62.
I.	EP3	CLR→TW LATCH 1. ↓EP3*/5A4 = CLR→TWFF (↑TWFF*/3B4, ↓TWFF)	The two-word latch is cleared.
J,K, and L	PU5, EP6, PU7	Identical with I, J, and K of Table 4-62.	See Table 4-62.

or I04 are low level. These signals are low-level until the instruction register shift takes place; the shift does not occur until after the secondary basic phase takes place. Thus, during the primary basic phase, the processing events that take place can be summarized as follows.

a. The content of the program counter is loaded into the address register.

b. The two-word latch is set.

c. The content of the memory buffer register is loaded into the instruction register.

d. During the last period, another basic phase is set.

Note that other operations are also performed during the primary basic cycle but they have no active roles.

4.4.4.2.2 Secondary Basic Phase. The secondary basic phase of the two-word memory reference instruction retains most of its similarities with the common basic phase, except for four operations. First, in Fundamental Operation A, Table 4-65, the address register is selected as an input to the MX multiplexer. Next, the adder is incremented by a count of 1 to obtain the address of the second word of the two-word instruction group and the resulting data word is admitted into the address register and program counter. Another operation which differs from the common basic phase operations is that the instruction register is not cleared so that its data remains available for processing during this basic phase. Another difference in operation occurs when event I, Table 4-68 occurs. This operation effectively shifts the content of the instruction register three bits leftward. The content of the instruction register is the first word of the two-word memory reference instruction, which up until this operation occurs has remained unmodified. The three-bit shift is accomplished by providing only one shift pulse to the instruction register logic.

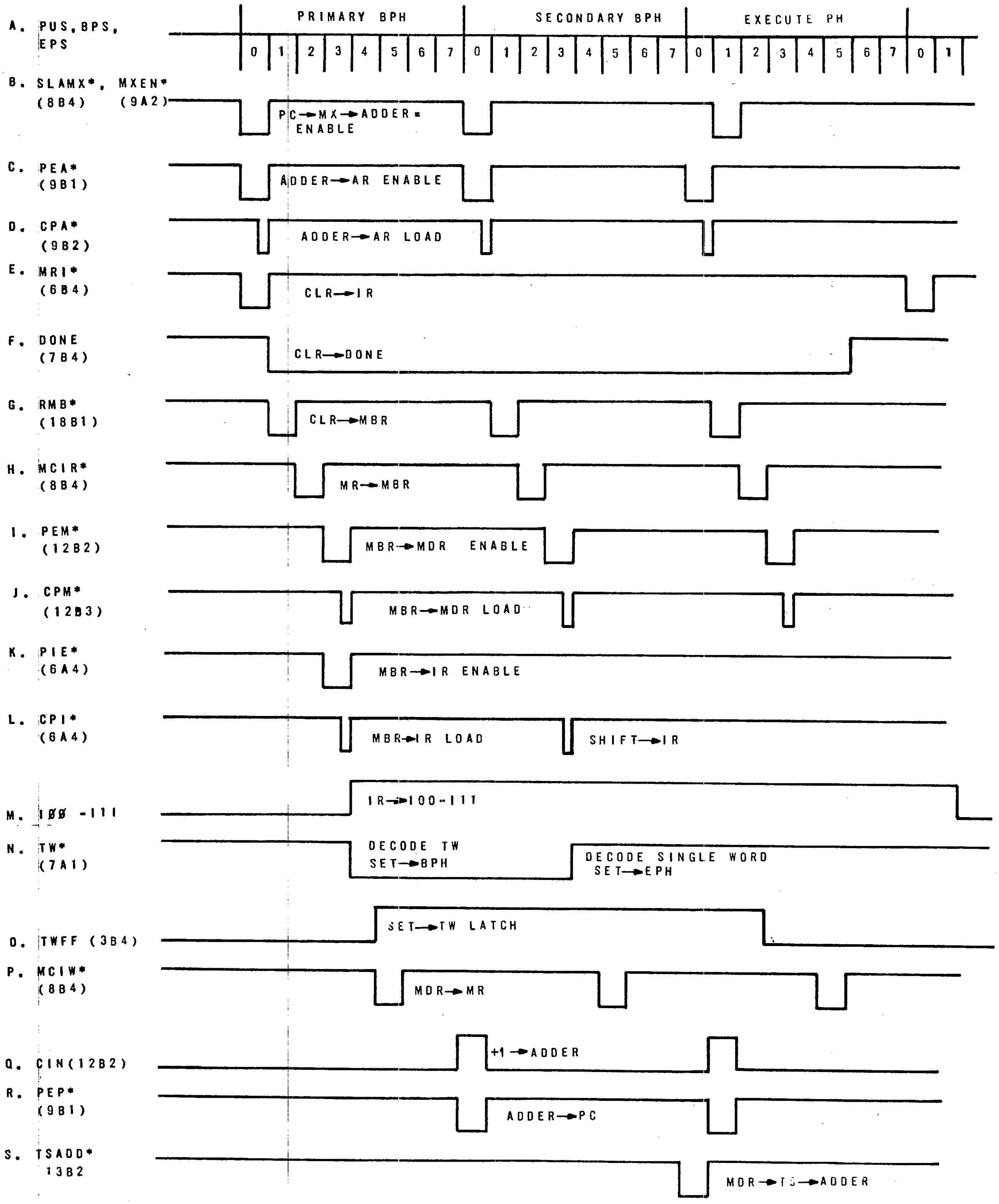
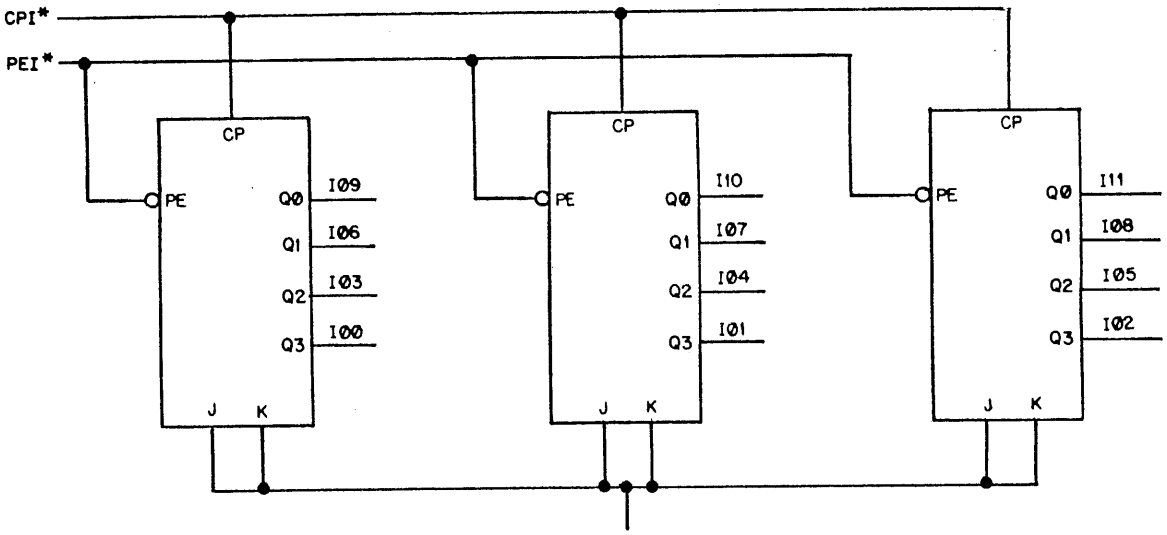


Figure 4-57. Two-Word Memory Reference Instructions, Direct Addressing, Timing Diagram

How this is done can be more clearly understood by referring to Figure 4-58. This figure is the simplified schematic diagram of the instruction register (the master reset and parallel enable inputs are not shown). The instruction register comprises three Fairchild Type 9300 4-bit shift registers configured as shown. Shifting, when done, causes a single leftward shift of data each time a clock pulse is received at the CP input and the parallel enable (PE) input is high level. Shifting is from the low-order numbers of each shift register to the high-order numbers, one bit for each clock pulse. Thus, when a single clock pulse is received, all Q_0 bits receive a low level (due to the ground applied at inputs J and K, respectively, of all the shift registers) and bits Q_1 through Q_3 receive corresponding high or low levels. The data contained in bit Q_3 of each shift register is shifted out and is lost. Thus, for the first shift register, when the shift occurs data in bit 100 is shifted out, bit 103 is shifted into bit 100, bit 106 is shifted into bit 103, bit 109 is shifted into bit 106, and a zero is shifted into bit 109. This same pattern occurs in the other two shift registers.

Although the instruction gets decoded at period BP3 of the secondary basic phase, processing is deferred to the ensuing execute phase. The concluding operation of the secondary basic phase is controlled by pulse PU7; the major state control logic orders an execute phase next. Thus, during the secondary basic phase the events that take place can be summarized as follows.



BIT PATTERN PRIOR TO SHIFT (0557_8)

TWSTK (MFØ3)	0	1	2	3	4	5	6	7	8	9	10	11
	0	0	0	1	0	1	1	0	1	1	1	1

TWSTK (MFØ3)	0	1	2	3	4	5	6	7	8	9	10	11
	1	0	1	1	0	1	1	1	1	0	0	0

BIT PATTERN AFTER SHIFT (5570_8)

Figure 4-58. Instruction Register, Simplified Schematic Diagram

a. The content of the program counter is incremented and stored into the address register and program counter.

b. The content of the first word of the instruction is shifted three positions and stored into the instruction register and instruction decoder.

c. During the last period, the execute phase is set.

4.4.4.2.3 Execute phase. The two-word memory reference instruction execute phase is similar to the one-word memory reference instruction execute phase. Initially, the content of the memory data register which now holds the address of the second word of the two word instruction group is switched through the TS multiplexer into the adder. This address is then admitted into the address register. Also, the content of the program counter is switched through the MX multiplexer to the adder and incremented to get the location of the next instruction in the program. This location is the next one that will be accessed during operation of the ND812. Next, the operand which is located at the address contained in the address register is admitted into the memory data register for further processing. This data is retained and operated upon during final processing.

During the execute phase of the two-word memory reference instruction the following events take place.

a. The content of the memory data register, the effective address, is switched through the TS multiplexer into the adder and into the address register.

b. The content of the program counter is incremented by one.

c. The two-word latch is cleared.

d. The content of the memory data register is written into the memory location specified by the content of the address register.

e. The basic phase is set by the major state control logic for the next instruction.

4.4.4.3 INDIRECT ADDRESSING. The indirectly addressed memory reference instructions combine the operations of both the two-word memory reference instructions with direct fetch, and the single-word memory reference instruction with indirect fetch. Tables 4-70 through 4-73 are the Event Summaries for the indirectly fetched two-word memory reference instruction. There is only one tabulation for Fundamental Operations because all other operations are described in previously presented tables. This table lists the Fundamental Operations for the tertiary basic phase. Figure 4-59 is the timing diagram for the two-word memory reference instruction indirect fetch. The indirectly fetched two-word memory reference instruction has three basic phases and one execute phase.

4.4.4.3.1 Primary Basic Phase. The primary basic phase events are identical to the primary basic phase events for the single-word memory reference instruction. The indirect latch is

Table 4-70. Two-Word Memory Reference Instructions, Indirect Addressing, Primary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	PU7	CLR→LATCHES	A
B	BP0	PC→MX→ADDER	B
C	PU0B	ADDER→AR	C
D	PU0B	CLR→IR	D
E	BP1	CLR→DONE	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	MBR→IR	I
J	BP5	SET→TW LATCH	None‡
K	PU5	MDR→MR	J
L	PU7	SET→BASIC PHASE	L

‡ Refer to Table 4-67, Operation J

Table 4-71. Two-Word Memory Reference Instructions, Indirect Addressing, Secondary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	BP0	AR→MX→ADDER	None ‡1
B	BP0	+1→ADDER	None ‡2
C	BP0	ADDER→PC	None ‡3
D	PU0B	ADDER→AR	C
E	BP1	CLR→DONE	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	SHIFT→IR	None ‡4
J	BP4	SET→IND, INDFF LATCHES	None ‡5
K	PU5	MDR→MR	J
L	PU7	SET→BASIC PHASE	L

‡1 Refer to Table 4-68, Operation A
‡2 Refer to Table 4-68, Operation B
‡3 Refer to Table 4-68, Operation C
‡4 Refer to Table 4-68, Operation I
‡5 Refer to Table 4-59, Operation J

not set during the primary basic phase. Although it may first appear that this is not true, it should be remembered that the two-word instruction does not admit data into the instruction decoder until after the instruction-register shift occurs. Referring to Table 4-59, Operation J, it can be seen that to obtain high-level signal MREFI, bits I00 through I03 must produce

Table 4-72. Two-Word Memory Reference Instructions, Indirect Addressing, Tertiary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	BP0	MDR→TS→ADDER	None
B	PU0	ADDER→AR	C
C	BP1	CLR→DONE	E
D	PU1	CLR→MBR	F
E	PU2	MR→MBR	G
F	PU3	MBR→MDR	H
G	PU3	CLR→IND LATCH	None ‡
H	PU5	MDR→MR	J
I	PU7	SET→EXECUTE PHASE	L

‡ Refer to Table 4-60, Operations H and I

Table 4-73. Two-Word Memory Reference Instructions, Indirect Addressing, Execute Phase, Event Summary

Ref.	Period	Event	SING-WD Ex. Phase Ref. (Table 4-61)
A	EP0	MDR→TS→ADDER	A
B	PU0	ADDER→AR	B
C	PU1	CLR→MBR	C
D	EP1	PC→MX→ADDER	D
E	EP1	+1→ADDER	E
F	EP1	ADDER→PC	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	EP3	CLR→TW LATCH	None ‡
J	PU5	MDR→MR	I
K	EP6	SET→DONE LATCH	J
L	PU7	SET→BASIC PHASE (only)	K

‡ Refer to Table 4-69, Operation I

one of the operate codes 24, 30, 34, 40, 44, 50, 60, 64 or 70. As can be seen from Table 4-13, these operate codes are only produced when one or more of bits I00 through I03 are asserted; this does not take place until the effective three-bit shift of the instruction register during the secondary basic phase. The events that take place can be more easily understood by referring to Table 4-74, a synopsis of the events that occur during an indirectly fetched two-word memory reference instruction.

The program listing is shown under the heading, Program. The operand list (right column) contains data only where the two-word instruction has relevance. The intent is that the data located at address 2055 be loaded into the J register; however, the second

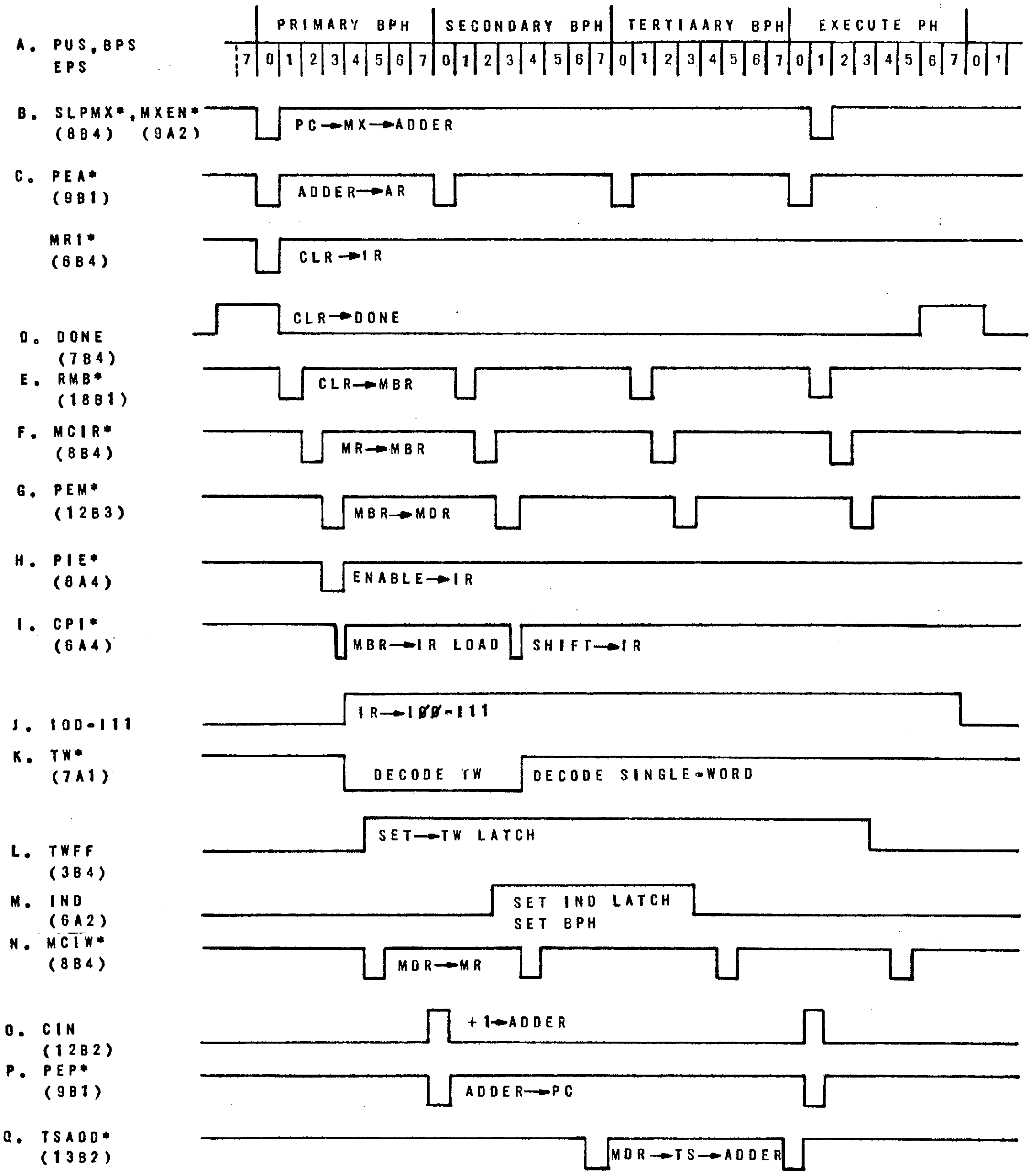


Figure 4-59. Two-Word Memory Reference Instructions, Indirect Addressing, Timing Diagram

Table 4-74. Two-Word Memory Reference Instructions, Indirect Addressing, Synopsis of Events

```
PROGRAM
   .
   .
   .
   .
  2043      TWLDJ
  2044       2052
  2045        .
  2046        .
  2047        .
  2050        .
  2051        .
  2052       2055
  2053        .
  2054        .
  2055       XXXX
   .          .
   .          .
   .          .
```

	Register Contents			
Phases	PC	AR	MDR	MR
1st BASIC PHASE	2043	2043	TWLDJ	TWLDJ
2nd BASIC PHASE	2044	2044	2052	2052
3rd BASIC PHASE	2044	2052	2055	2055
EXECUTE PHASE	2045	2055	XXXX	XXXX

word of the two-word instruction does not name the address of the data, but the address of its address. Hence, 2052 is not an operand, but the address of the address of the operand. When the indirectly fetched two-word instruction primary basic phase occurs, the program counter contains the address of the next instruction that is fetched from memory, or in this case, 2043. This address is admitted into the address register and the two-word instruction TWLDJ is loaded into the instruction register and memory data register from location 2043 in memory. Thus at the end of the primary basic phase, the program counter, the address register, the memory data register, and the memory register contain the data shown in the register contents columns of Table 4-74.

The low-level signal TW* permits the basic phase to be set again at period BP6. This signal is asserted whenever instruction register bits I00 through I03 or I04 are low level. This signal remains asserted until the effective three-bit instruction register shift occurs.

The events that take place during the primary basic phase can be summarized as follows.

a. The content of the program counter is loaded into the address register.

b. The content of the memory buffer register is loaded into the memory data register and into the instruction register.

c. The two-word latch is set.

d. A second basic phase is set.

4.4.4.3.2 Secondary Basic Phase. During the secondary basic phase, the content of the address register, 2043, is admitted into the adder and incremented to get the next address, 2044. The operand fetched from this address is another address, 2052. This address is admitted into the memory data register. The content of the instruction register is not modified because the done latch remains cleared during the secondary basic phase Table 4-20, Operations D and K. However, during period BP3, the effective three-bit instruction register shift occurs. This shift admits operate-code bits 100 through 103 into the instruction decoder and permits the indirect-latch logic to be primed by a high-level MREFI signal as shown by Operation J, Table 4-59. Hence, when period BP4 occurs, the indirect and indirect flip-clop latches are set. The set state of the indirect latch permits the basic phase to be entered a third time (Operation L3, Table 4-20). When the three-bit instruction register shift occurs, the low-level TW* signal becomes high-level again, but the asserted indirect latch permits entry into the basic phase again through low-level signal IND*. Thus, at the end of the secondary basic phase, the program counter, the address register, the memory data register, and the memory register contain the data shown in the register contents columns of Table 4-74. It should be noted that the operand listed for the content of the memory data register at the end of the second basic phase is the pointer address for the indirect fetch. The events that take place during the secondary basic phase can be summarized as follows.

a. The content of the address register is admitted into the adder and incremented.

b. The content of the adder is admitted into the address register and program counter.

c. The instruction register contents are shifted 3 bits leftward.

d. The indirect and indirect flip-flop latches are set.

e. The content of the memory buffer register is admitted into the memory data register.

f. Another basic phase is entered.

4.4.4.3.3 Tertiary Basic Phase. The tertiary (or third) basic phase is similar to the execute phase with the exception that the content of the program counter is not modified (Table 4-75). Also, during this phase, the indirect latch is cleared. Thus, because the done latch remains unset (signal DONE* remains high level), and signals TW*, IND* and XCT* are all high level as shown for Operation L1, Table 4-20, the execute phase is set. Thus, at the end of the tertiary basic phase, the content of the program counter is unmodified, the address register obtains the pointer address, and the memory data register and memory register obtain the

Table 4-75. Two-Word Memory Reference Instructions, Indirect Addressing, Tertiary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU0	MDR→TS→ADDER 1. (↑TWFF) (↑IND)→↓TIFF*/8A1 2. ↓TIFF* + ↓XIFF* + ↓MULIS*→ ↑MSCTAD/13B1 3. (↑FLLOC*) (↑DONE*)→↓FLDN→ ↑FLDN*/13B2 4. (↑MSCTAD) (↑FLDN*) (↑BP0)→ ↓TSADD*/13B2 = MDR→TS→ADDER	The pointer address, contained in the memory data register, is switched through the TS multiplexer to the adder.
B-F	PU0- PU3	Same as Operations C, and E through H, Table 4-20.	See Table 4-20.
G.	PU3	Same as Operations H and I, Table 4-60.	See Table 4-60.
H,I	PU5, PU7	Same as Operations J and L, respectively, Table 4-20.	See Table 4-20.

final address. The events that take place during the tertiary basic phase can be summarized as follows.

a. The content of the memory data register is switched through the TS multiplexer to the adder.

b. The content of the adder is admitted into the address register.

c. The content of the memory buffer register is read into the memory data register.

d. The indirect latch is cleared.

e. The execute phase is set.

4.4.4.3.4 Execute Phase. The two-word indirect fetch memory reference instruction execute phase is similar to the direct fetch two-word memory reference instruction execute phase. Initially, the content of the memory data register which now holds the address of the operand of the two word instruction group is switched through the TS multiplexer into the adder. This address is then admitted into the address register. Also, the content of the program counter is switched through the MX multiplexer to the adder and incremented to get the location of the next instruction in the program. This location is the next one that will be accessed during operation of the ND812. Next, the operand which is located at the address contained in the address register is admitted into the memory data register for further processing. This data is retained and operated upon during final processing.

During the execute phase of the indirectly fetched two-word memory reference instruction the following events take place.

a. The content of the memory data register, the effective address, is switched through the TS multiplexer into the adder and into the address register.

b. The content of the program counter is incremented by one.

c. The two-word latch is cleared.

d. The content of the memory data register is written into the memory location specified by the content of the address register.

e. The basic phase is set by the major state control logic for the next instruction.

4.4.4.4 ACCUMULATOR SELECTION. One of the basic differences between single-word and two-word memory reference instructions is that for a two-word memory reference instruction, either the J or K register may be selected for the operation as the result of final processing. For the single-word memory reference instruction, this is not the case. Whenever a a single-word memory reference instruction is processed, the J register is selected as the affected accumulator. This automatically takes place during period BP7 of the previous execute phase as shown by Operation A of Table 4-76, the list of Fundamental Operations for selection of accumulators during a two-word memory reference instruction. The other operations described in Table 4-76 include use of the J or K register as the effective accumulator for addition, subtraction, store, and load memory reference instructions. It is clear by inspecting Fundamental Operations A and B of Table 4-76, that unless a two-word memory reference instruction is in effect, only the J register is the effective accumulator.

4.4.5 EXECUTE INSTRUCTION

The execute instruction is a special case of the one-word memory reference instruction. It is not discussed in the one-word memory reference instruction groups because this instruction does not have an execute phase. This instruction has the same format as any single-word memory reference instruction (Table 4-53). The execute instruction enables any instruction which can be relatively addressed to be executed without changing the content of the program counter. Because indirect addressing for an execute instruction is also possible, any instruction in a program can actually be executed. The execute instruction is the only memory reference instruction for which there is no two-word memory reference instruction form. The instruction located at the effective or indirect address may be any legal ND812 instruction of an operate class, single-word, or two-word, and can include another execute instruction.

Figure 4-60 is the execute instruction timing diagram, Tables 4-77 and 4-78 are the primary basic and secondary basic phase event summaries, respectively. Tables 4-79 and 4-80 are the listings of Fundamental Operations for the primary and secondary basic phases, respectively.

Table 4-76. Two-Word Memory Reference Instructions, Accumulator Selection, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. All	PU7	SET→JCOM 1. (↑DONE) (↑PU7B) = SET→JCOM (↑JCOM/3A4, ↓KCOM)	The J-register is the effective accumulator.
B. (25) (41) (45) (51) (55)	BP1	SET→KCOM 1. (↑TW) (↑I08B) (↑BP1) = SET→KCOM (↓JCOM, ↑KCOM/3A4)	The K-register is the effective accumulator.
C. (55)	EP3	K→MX→ADDER 1. (↑STJ) (↑EP3) (↑KCOM)→↓SLKMX*/12A4 2. ↓SLKMX*→↑MXEN→↓MXEN*/9A1 3. ↓SLKMX*/9B1→↑MXS0, ↑MXS2/9A2	The content of the K-register is switched through the MX multiplexer.
D. (25)	EP45	K→MX→ADDER 1. (↑SID) (↑EP45) (↑KCOM)→↓SLKMX*/12A4 2. ↓SLKMX*/12A4→↑MXEN→↓MXEN*/9A1 3. ↓SLKMX*→↑MXS0, ↑MXS2/9A2	The content of the K-register is switched through the MX multiplexer.
E. (41) (45)	EP6	K→MX→ADDER 1. (↑ADJSBJ) (↑EP6) (↑KCOM)→↓SLKMX*/12A4 2. ↓SLKMX*/12A4→↑MXEN→↓MXEN*/9A1 3. ↓SLKMX*→↑MXS0, ↑MXS2/9A2	The content of the K-register is switched through the MX multiplexer.
F. (41) (45) (51)	EP6	ADDER→K→ADDER 1. (↑ADSBLD/7B2) (↑EP6/5A4) (↑KCOM/3A4)→ ↓PEK*/9A3 2. ↓PEK*/9A3→↑PEK/9A4 3. (↑PEK/9A4) (↑REGCLK/4A4)→↓CPK*/9A4	The content of the adder is admitted into the K-register.
G. (54)	EP3	J→MX→ADDER 1. (↑STJ) (↑JCOM)→↓STJCM*→↑STJCM/12A3 2. (↑STJCM) (↑EP3)→↓SLJMX*/12A3 3. ↓SLJMX*→↑MXEN→↓MXEN*/9A1 4. ↓SLJMX*→↑MXS2/9A1	The content of the J-register is switched through the MX multiplexer to the adder.
H. (24)	EP45	J→MX→ADDER 1. (↑SID) (↑JCOM) (↑EP45)→↓SLJMX*/12A3 2. ↓SLJMX*→↑MXEN→↓MXEN*/9A1 3. ↓SLJMX*→↑MXS2/9A1	The content of the J-register is switched through the MX multiplexer.

Table 4-76. Two-Word Memory Reference Instructions, Accumulator Selection, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
I. (40) (44)	EP6	J→MX→ADDER 1. (↑ADJSBJ) (↑JCOM) (↑EP6)→↓SLJMX*/12A3 2. ↓SLJMX*→↑MXEN→↓MXEN*/9A1 3. ↓SLJMX*→↑MXS2/9A1	The content of the J-register is switched through the MX multiplexer to the adder.
J. (40) (44) (50)	EP6	ADDER→J 1. (↑ADSBLD) (↑JCOM)→↓LDLOG*→↑LDJ LOG 2. (↑LDJ LOG) (↑EP6)→↓PEJ*→↑PEJ/9A4 3. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4	The content of the adder is admitted into the J-register.

Table 4-77. Execute Instruction, Direct Addressing, Primary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	PU7	CLR→LATCHES	A
B	BP0	PC→MX→ADDER	B
C	PU0B	ADDER→AR	C
D	PU0B	CLR→IR	D
E	BP1	CLR→DONE	E
F	PU1B	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	MBR→IR	I
J	PU5	MDR→MR	J
K	BP6	SET→XCT LATCH	None
L	PU7	SET→BASIC PHASE	L

Table 4-78. Execute Instruction, Direct Addressing, Secondary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	BP0	AR→MX→ADDER	None
B	BP0	REL ADDRESS→TS→±LOGIC	None
C	BP0	(MX + TS)→ADDER (BIT 5 = 0)	None
D	BP0	(MX - TS)→ADDER (BIT 5 = 1)	None
E	PU0B	ADDER→AR	C
F	BP1	CLR→DONE	E
G	PU1B	CLR→MBR	F
H	PU2	MR→MBR	G
I	PU3	MBR→MDR	H
J	BP3	MBR→IR	I
K	PU5	MDR→MR	J
L	PU7	SET→PHASE	L

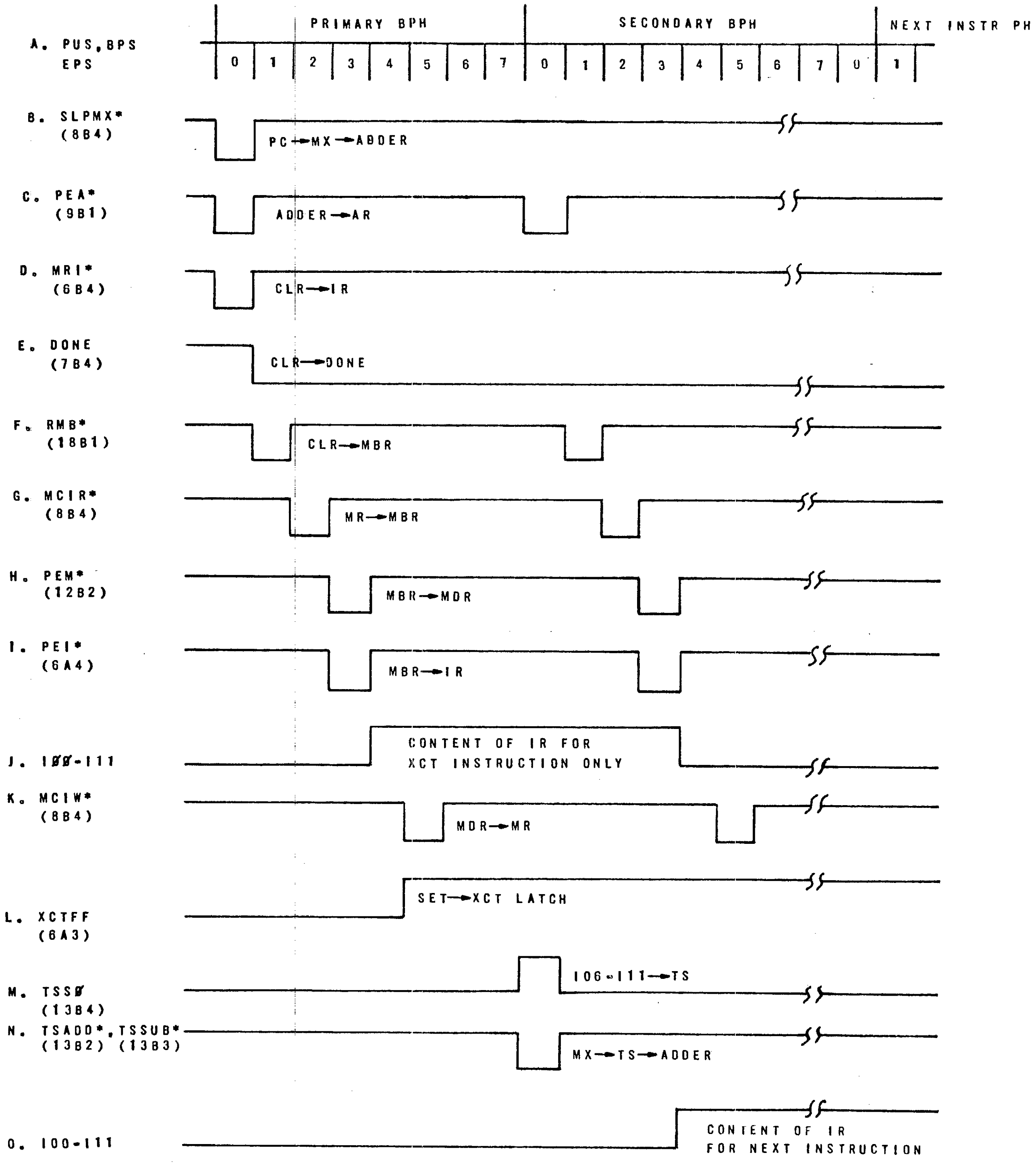


Figure 4-60. Execute Instruction, Direct Addressing, Timing Diagram

Table 4-79. Execute Instruction, Direct Addressing, Primary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-J	BP0- BP5	Identical to Operations A through J, Table 4-20.	See Table 4-20.
K.	BP6	SET→EXECUTE LATCH 1. (↑I00) (↑I01) (↑I02)→↓XCT*/7B1 2. ↓XCT*→↑XCT/7B1 3. (↑IND*) (↑XCT) (↑BP6)→↓SXCT*/6A2 = SET→EXECUTE LATCH (↑XCTFF/6A2, ↓XCTFF*)	The execute latch is set.
L.	PU7	SET→BASIC PHASE 1. ↓DONE* + ↓TW* + ↓IND*+↓XCT*→ ↑RBPH→↓RBPH*/5B1 2. (↓RBPH*) (↓CPPU*) (↑PU7) = SET→ BPH (↑BPH, ↓EPH/5B2)	A second basic phase is entered.

4.4.5.1 DIRECT ADDRESSING, PRIMARY BASIC PHASE. The primary basic phase for the directly fetched execute instruction is similar to the primary basic phase for any single-word memory reference instruction (Table 4-20) with one exception; during period BP6, the XCTFF latch is set. Setting of this latch permits the events of the secondary basic phase to be executed. Initially, the content of the program counter is admitted into the adder to obtain the operand for the current instruction, which in this case, is the execute instruction. This instruction is admitted into the instruction register and is decoded as an execute. As a result of this decoding operation, the XCTFF latch is set.

4.4.5.2 DIRECT ADDRESSING, SECONDARY BASIC PHASE. The secondary basic phase performs two functions. It determines the effective address and then becomes the primary basic phase for the instruction which must be executed. This is possible because the instruction register contents, which were loaded into the instruction register during the primary basic phase, are overwritten with the operand located at the effective address. Thus, during the first basic phase, the content of the program counter is loaded into the address register. Subsequently, the operand located at the effective address is loaded into the instruction register, but its location does not appear in the program counter. The secondary basic and subsequent phase operations can be understood by referring to Table 4-81, a synopsis of events for the execute instruction. The program listing is similar to the program listing previously presented in Table 4-74 with one exception, an execute instruction has been added at location 2035. The execute instruction is an order to execute the two-word load J (TWLDJ) instruction at location 2043.

Operation is as follows. When the program counter reaches 2035, the primary basic phase for the instruction is entered. At the beginning, the content of the program counter is loaded into the address register and instruction 7006 is loaded into the memory buffer register and into the instruction register and the memory data register. When the

Table 4-80. Execute Instruction, Direct Addressing, Secondary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BPO	AR→MX→ADDER 1. ↓XCT* + ↓IND* + ↓TWFF*→↑XITFF/8A1 2. ↓TWFF + ↓IND→↑TIFF*/8A1 3. (↑XITFF) (↑XIFF*) (↑TIFF*)→↓SLAMU*/8A1 4. ↓SLAMU*→↑SLAMU/8A1 5. (↑SLAMU) (↑BPO) (↑FLLOC*)→↓SLAMX*/8A2 6. ↓SLAMX*→↑MXEN→↓MXEN*/9A1 7. ↓SLAMX*→↓MXS0, ↓MXS1, ↓MXS2 = AR→MX→ADDER	The content of the address register is switched through the MX multiplexer to the adder.
B.	BPO	REL ADDRESS→TS→ADDER 1. ↓XCTFF + ↓IND* + ↓INDFF→↑XIFF*/6A3 2. (↑XIFF*) (↑TWFF*)→↓MLTAS*/6A4 3. ↓MLTAS*→↑MLTAS/13B4 4. (↑MLTAS) (↑BPO)→↓SLITS*→↑TSS0/13B4 = I06 - I11→TS→ADDER	Bits I06 - I11 are switched through the MX multiplexerto the adder as the relative (or displacement) address.
C.	BPO	(MX + TS)→ADDER (Bit 5 = 0) 1. (↑MLTAS) (↑I5B*)→↓MULIS*→↑MSCTAD/13B1 2. (↑FLLOC*) (↑DONE*)→↓FLDN→↑FLDN*/13B1 3. (↑MSCTAD) (↑FLDN*) (↑BPO)→↓TSADD*/13B2 = +TS→ADDER	If bit 5 is not set, bits I06 through I11 are added to the content of the address register.
D.	BPO	(MX - TS)→ADDER (Bit 5 = 1) 1. (↑MLTAS) (↑I05X) (↑FLDN*) (↑BPO)→↓TSSUB*/13B3 = -TS→ADDER	If bit 5 is set, bits I06 through I11 are subtracted from the content of the address register.
E.	PU0B	ADDER→AR 1. PU0B→↓PEA*→↑PEA/9B1 2. (↑PEA) (↑REGCLK)→↓CPA*/9B2 = ADDER→AR	The content of the adder is admitted into the address register.
F.	PU0B	CLR→DONE 1. ↓BP1*/5A3 = CLR→DONE (↓DONE/7B4, ↑DONE*)	The done latch is clearea.
G.	PU1B	CLR→MBR 1. ↑PU1B→↓RMB*/18B1 = CLR→MBR	The memory buffer register is cleared.
H.	PU2	MR→MBR 1. (↑ADDF0) (↑PU2)→↓MCIRW*/8B4 = MR→MBR	The content of the memory register is read into the memory buffer register.

Table 4-80. Execute Instruction, Direct Addressing, Secondary Basic Phase, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
I.	PU3	MBR→MDR 1. ↓PU3*→↑PEM/12B2 2. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = MBR→MDR	The content of the memory buffer register is read into the memory data register.
J.	BP3	MBR→IR 1. (↑TWFF*) (↑INDFF*)→↓MPASS*/13B4 2. ↓MPASS*→↑MPASS/13B4 3. ↓MPASS*→↑SPEI* 4. (↑SPEI*) (↑BP3)→↓PEI*/6A4 5. ↓PEI*→↑PEI/6A4 6. (↑PEI) (↑ CCLK)→↓CPI*/6A4 = MBR→IR	The content of the memory buffer register is loaded into the instruction register.
K.	PU5	MDR→MR 1. (↑WTRL) (↑PU5B)→↓PU5W*→↑PU5W/8B3 2. (↑PU5W) (↑ADDF0*)→↓MCIW*/8B4 = MDR→MR	The content of the memory data register is read into the memory register.
L.	PU7	SET→PHASE 1. (↑DONE*) (↑TW*) (↑IND*) (↑XCT*)→ ↓RBPH→↑RBPH*/5B1 2. (↑RBPH*) (↓CPPU*) (↑PU7) = SET→ EXECUTE PHASE (↑EPH/5B2, ↓BPH) 3. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→ ↑RBPH→↓RBPH*/5B1 4. (↓RBPH*) (↓CPPU*) (↑PU7) = SET→ BASIC PHASE (↓EPH/5B2, ↑BPH)	The next phase of the executed operation is set.

execute instruction is decoded, the XCTFF latch is set and during the secondary basic phase, the content of the address register (current address) is switched through the MX multiplexer to the adder. Next, instruction register relative-address bits 100 through 106 are switched through the TS multiplexer and summed (in this case added) with the content of the address register to get effective address 2043, which is admitted into the address register replacing the previous current address. Next, operand 0500 (TWLDJ) at location 2043 is fetched into the memory buffer register, the instruction register, and the memory data register. The secondary basic phase of the execute instruction produces the same results as the primary basic phase of the TWLDJ instruction with one exception; the content of the program counter is not admitted into the MX multiplexer during period BP0. This can be verified by referring to Operation B1, Table 4-20; this operation is inhibited due to the low-level XCT* signal (waveform J, Figure 4-60) from the instruction decoder. The TWLDJ instruction is written over the XCT.+6 instruction because the two-word latch (TWFF) does not become set until

Table 4-81. Execute Instruction, Direct Addressing, Synopsis of Events

```
PROGRAM
  2035          XCT .+6     (7006)
    •              •
    •              •
    •              •
  2043          TWLDJ       (0500)
  2044           2052
  2045            •
  2046            •
  2047            •
  2050            •
  2051            •
  2052           2055
  2053            •
  2054            •
  2055           XXXX
    •              •
    •              •
    •              •
```

Phases	PC	AR	MDR	MR
		Register Contents		
(XCT) 1st BASIC PHASE	2035	2035	7006	7006
(XCT) 2nd BASIC PHASE‡	2035	2043	0500	0500
(TWLDJ) 2nd BASIC PHASE	2035	2044	2052	2052
(TWLDJ) 3rd BASIC PHASE	2035	2052	2055	2055
(TWLDJ) EXECUTE PHASE	2036	2055	XXXX	XXXX

‡ This is also the primary basic phase for the TWLDJ instruction

period BP5 occurs (Operation J, Table 4-70). When the TWLDJ instruction is written over the former XCT.+6 instruction, the XCT* signal becomes high-level again because now a two-word instruction is decoded. The subsequent processing operation is similar to the processing operation for the indirectly fetched two-word memory reference instruction as previously described in paragraph 4.4.4.3. During the secondary basic phase of the indirectly fetched two-word memory reference instruction, the content of the program counter is not modified due to the presence of low-level signal XCTFF*, which is produced by the set state of the XCTFF latch as verified by Operation C1, Table 4-68. Thus, with the exception of program counter modification, the execute instruction for an indirectly addressed two-word memory reference instruction is identical to the processing operation for the instruction itself, but with the addition of one phase. The program counter is incremented only during the execute phase of the executed instruction to obtain the address of the next instruction.

4.4.5.3 INDIRECT ADDRESSING. There are three basic phases when an indirect execute instruction is processed. Table 4-82, 4-83 and 4-84 are the Event Summaries for operations that occur when an indirectly fetched execute is specified. The basic phases are subdivided into a primary basic phase, a secondary basic phase, and a tertiary basic phase. The events that occur during the primary basic phase are identical to those that occur during the common

Table 4-82. Execute Instruction, Indirect Addressing, Primary Basic Phase, Event Summary

Ref.	Period	Event	Common Basic Phase Ref. (Table 4-20)
A	PU7	CLR→LATCHES	A
B	BP0	PC→MX→ADDER	B
C	PU0B	ADDER→AR	C
D	PU0B	CLR→IR	D
E	BP1	CLR→DONE	E
F	PU1B	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	PU3	MBR→IR	I
J	BP4	SET→IND, INDFF LATCHES	None‡
K	PU5	MDR→MR	J
L	PU7	SET→BASIC PHASE	L

‡ See Table 4-60.

Table 4-83. Execute Instruction, Indirect Addressing, Secondary Basic Phase, Event Summary

Ref.	Period	Event	Ind Fetch 2nd Phase Ref. (Table 4-60)
A	BP0	AR→MX→ADDER	A
B	BP0	REL→TS→±LOGIC	B
C	BP0	SELECT→(MX + TS)→ADDER	C
D	BP0	SELECT→(MX - TS)→ADDER	D
E	PU0B	ADDER→AR	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	PU3	CLR→IND LATCH	I
J	PU5	MDR→MR	J
K	PU7	SET→BASIC PHASE	None

basic phase, but with modified events. The modified events are produced when bit 4 of the execute instruction is set, indicating indirect addressing; also, the execute phase is not set. As a consequence, the indirect latch is set so that when period PU7 occurs, a second basic phase will be enabled. During the secondary basic phase, most of the events described for the common execute phase occur. During the preceding basic phase the done latch is not set because there is no latch-setting input to the done-latch logic at the conclusion of a basic phase when an execute instruction is processed. This is verified by reference K, Table 4-20, operation No. 1. Thus, because the done latch remains unset, the instruction register is not cleared during period PU0 of the ensuing basic phase. This is verified by reference D, Table 4-20: the done latch must be set to clear the instruction register. Hence, data stored in the instruction register is available to processing logic during the secondary basic phase. Setting of the indirect latch at period BP4 of the primary basic phase defers

Table 4-84. Execute Instruction, Indirect Addressing, Tertiary Basic Phase, Event Summary

Ref.	Period	Event	XCT Secondary Basic Phase Ref. (Table 4-80)
A	BP0	AR→MX→ADDER	A
B	BP0	REL ADDRESS→TS→ADDER	B
C	BP0	(MX + TS)→ADDER	C
D	BP0	(MX - TS)→ADDER	D
E	PU0	ADDER→AR	E
F	PU0B	CLR→DONE	F
G	PU1	CLR→MBR	G
H	BP1	CLR→INDFF, XCTFF	None
I	PU3	MR→MBR	H
J	PU3	MBR→MDR	I
K	PU3	MBR→IR	J
L	PU5	MDR→MR	K
M	PU7	SET→PHASE	L

an execute phase in favor of a secondary basic phase. This is verified by reference K, Table 4-20, operation No. 3. One of the conditions that establishes a basic phase is a set indirect latch which produces a low-level IND* signal. Hence, at the conclusion of the primary basic phase, the secondary basic phase is entered. The indirect latch is cleared during the secondary basic phase so that at its conclusion, the normal execute instruction events can occur. Hence, the content of the instruction register remains unaltered because the events that normally fetch data into it are inhibited, but only for this secondary basic phase. As shown in Table 4-83 fundamental operations G, H, and I of the common execute phase (Table 4-57) are not carried out during the secondary basic phase. With the exception of the INDFF and XCTFF latch clearing operation, Event H, Table 4-84, the tertiary basic phase is identical to the secondary basic phase for the directly fetched execute instruction; the tertiary phase also serves the same function as the secondary basic phase for the directly fetched execute instruction.

4.4.5.3.1 Primary Basic Phase. Tables 4-85, 4-86 and 4-87 list the Fundamental Operations and Figure 4-61 is the timing diagram for the signals generated when the execute instruction names an indirect fetch through the set state of bit 4 in the instruction word. By comparing waveforms A through I of Figure 4-61 with those of Figure 4-60, it can be seen that for periods BP0 through BP3, the events that occur are identical. When pulse BP4 is generated, the similarities in processing temporarily disappear. Table 4-85 is the list of Fundamental Operations for the primary basic phase of an execute instruction indirect fetch. These logic descriptions begin with period BP4 of the primary basic phase. High-level signal MREFI is produced whenever a memory reference instruction is called for in a program. It can be seen that when bit 4 is set, high level signals MREFI, I04B, and INDFF (produced by the previously cleared INDFF latch), are ANDed to produce the set state of the IND and INDFF latches (Figure 4-61, waveforms J and K). The set state of the INDFF latch produces low-level signal INDFF*, which inhibits the production of high-level signal MPASS (reference I, operation No. 1, Table 4-20). Inhibiting high-level signal MPASS prevents the content of the memory buffer register from being fetched into the instruction

Table 4-85. Execute Instruction, Indirect Addressing, Primary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Primary Basic Phase Ref. (Table 4-79)
A through I	BP0 - BP3	Same as Operations A through I, Table 4-20	See Table 4-20
J	BP4	Same as Operation J, Table 4-59	See Table 4-59
K and L	PU5, PU7	Same as Operations J and L, respectively, Table 4-79	See Table 4-79

Table 4-86. Execute Instruction, Indirect Addressing, Secondary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-J	BP0-PU5	Same as Operations A through J, Table 4-60.	See Table 4-60.
K.	PU7	SET→BASIC PHASE 1. ↓DONE* + ↓TW* + ↓XCT*+↓IND*→↑RBPH→ ↓RBPH*/5B1 2. (↓RBPH*) (↓CPPU*) (↑PU7) = SET→ BASIC PHASE (↑BPH/5B2, ↓EPH)	The basic phase is set.

Table 4-87. Execute Instruction, Indirect Addressing, Tertiary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-G	BP0-PU1	Same as Operations A through G, Table 4-80.	See Table 4-80.
H	BP1	CLR→INDFF, XCTFF 1. (↑XCTFF) (↑IND*) (↑INDFF) (↑BP1)→ ↓SINDM* 2. ↓SINDM* = CLR→INDFF (↓INDFF/6A3, ↑INDFF*) 3. ↓SINDM* = CLR→XCTFF (↓XCTFF/6A3, ↑XCTFF*)	The INDFF and XCTFF latches are cleared.
I-M	PU3-PU7	Same as Operations H through L, Table 4-80.	See Table 4-80.

register during the subsequent basic phase because the operand is already in the MDR.

When period PU7 occurs, the set state of the IND latch produces a low-level IND* signal which permits the major-state control register to again be reset to its basic-phase state as described by the logic of reference L, Table 4-79. Thus, the events that take place during the primary basic phase are:

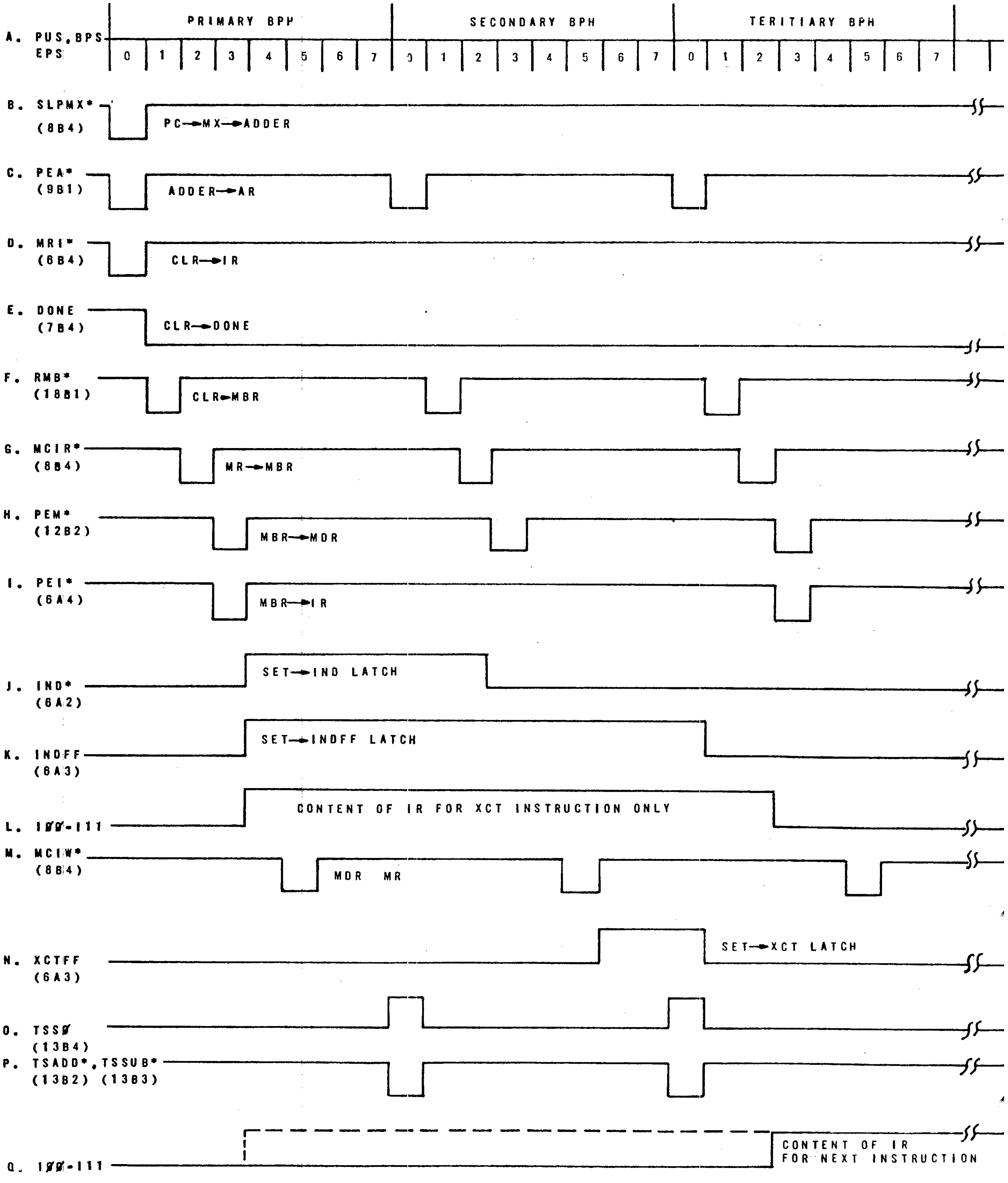


Figure 4-61. Execute Instruction, Indirect Addressing, Timing Diagram

a. The content of the program counter is loaded into the address register.

b. The content of the memory register located at the address specified by the content of the address register is read into the memory buffer register.

c. The content of the memory buffer register is loaded into both the memory data register and into the instruction register.

d. The indirect latch is set to prepare for a latter entry into the secondary basic phase.

4.4.5.3.2 Secondary Basic Phase. Table 4-86 is the list of Fundamental Operations for the secondary basic phase. The timing diagram is essentially the same as for the common execute phase (Figure 4-60), with the difference that functions of the secondary basic phase are based on the BP pulse train. As previously described for the primary basic phase, the high-level MPASS signal is inhibited by a low-level INDFF* signal, indicating indirect addressing. Because the high-level MPASS signal is the primary control signal for initiating the arithmetic, which leads to obtaining the effective address from the current address and the indirect pointer address for the ensuing tertiary phase, another signal must take over when the above address modification is called for during a basic phase. This signal, MLTAS* is generated, as shown by operation 3, reference 2, Table 4-60.

Operation 1 of reference I, Table 4-60 shows an operation that is actually carried out during other cycles too, but in those it finds the indirect latch already in the clear state. In the present case, however, by clearing the set state of the indirect latch, the subsequent execute phase is not established because of the presence of the low-level XCT signal from the instruction decoder. Hence, a third basic phase is entered. The other difference is that waveforms associated with Fundamental Operations G, H and I of the common execute phase are not generated, because those operations are not carried out by the processing logic. These operations normally provide for the incrementation of the program counter, however, it is premature to carry out that operation during this subcycle; it is deferred until the execute phase. That operations G, H and I (Table 4-55) are inhibited can be seen at once because they all require timing signal EP1 which cannot be generated during a basic phase.

When period PU7 occurs, the low-level state of signal XCT* inhibits transition of the major-state control register. Low-level signal XCT* produces a high-level RBPH signal (Operation K, Table 4-86), which permits the basic phase to be entered again. The events that take place during the secondary basic phase are:

a. The content of the address register is switched through the MX multiplexer to the adder and summed with the content of the TS multiplexer, bits 6 through 11 of the instruction register, and stored into the address register. The address register now contains the relative address.

b. The indirect latch is cleared.

c. The content of the memory data register is written into the memory location specified by the content of the address register.

d. The major-state control is set for the next phase.

4.4.5.3.3 Tertiary Basic Phase. The tertiary basic phase is identical to secondary basic phase of the directly fetched execute instruction and it serves identical functions. With the exception of the XCTFF and INDFF latch clearing operaions, all events are identical. This latch clearing operation at period BP1 permits the data contained in the instruction register to be overwritten by the newly fetched operand of the instruction to be executed. The events that take place during the tertiary basic phase are:

a. The content of the memory data register, the indirect address is switched through the TS multiplexer to adder and into the address register.

b. The content of the memory buffer register (indirect operand) is written into the instruction register and into the memory data register.

c. The content of the memory data register is written into the location specified by the address register.

d. The next phase for the executed instruction is entered.

Table 4-88 illustrates the growth of the compound memory reference instruction, during the basic and execute phases when different types of execute instructions call different types of one-or two-word memory reference instructions. The number of lines in the Basic Phase column corresponds to the total number of basic phases required for the execution of the compound instruction. For example, an execute instruction using relative direct addressing and calling for an indirect memory reference instruction requires 3 basic phases and one execute phase, or a total of 4 phases for processing. Note that the number of execute phases is either zero, as for operate instructions, or one as for all memory reference instructions.

Table 4-88. Multiphase Memory Reference Instructions

	PHASES	
Instruction	Basic (BPO)	Execute (EPO)
MEMREF	BP1. PR→AR	AR ±6 of IR→AR
MEMREF IND	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	MR→AR
TW	BP1. PR→AR	
	BP2. AR +1→AR & PR	MR→AR
TW IND	BP1. PR→AR	
	BP2. AR +1→AR & PR	
	BP3. MR→AR	MR→AR
XCT	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	AR ±6 of IR→AR

Table 4-88. Multiphase Memory Reference Instructions (Cont'd.)

Instruction	PHASES	
	Basic (BPO)	Execute (EPO)
XCT MEMREF IND	BP1. PR→AR	
	BP2. AR ±6→AR	
	BP3. AR ±6→AR	MR→AR
XCT IND MEMREF	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. MR→AR	AR ±6 of IR→AR
XCT IND MEMREF IND	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. MR→AR	
	BP4. AR ±6 of IR→AR	MR→AR
XCT IND XCT MEMREF	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. MR→AR	
	BP4. AR ±6 of IR→AR	AR ±6→AR
XCT IND XCT IND MEMREF	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. MR→AR	
	BP4. AR ±6 of IR→AR	
	BP5. MR→AR	AR ±6 of IR→AR
XCT TW	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. AR +1→AR	MR→AR
XCT TW IND	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. AR +1→AR	
	BP4. MR→AR	MR→AR
XCT IND TW	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. MR→AR	
	BP4. AR +1→AR	MR→AR
XCT IND TW IND	BP1. PR→AR	
	BP2. AR ±6 of IR→AR	
	BP3. MR→AR	
	BP4. AR +1→AR	
	BP5. MR→AR	MR→AR

4.4.6 MEMORY REFERENCE INSTRUCTION SUBGROUPS

Excluding the execute instruction which has been described in paragraph 4.4.5, there are 10 memory reference instructions. These instructions are listed in Table 4-51. Although the ANDF instruction is a memory reference instruction, it is processed as a literal; hence, the ANDF instruction is described in paragraph 4.4.9.

The following memory reference instruction subgroup headings reflect the one-word memory reference instructions with two-word memory reference instructions shown in parentheses. Two types of two-word memory reference instructions are noted in the headings. The first type is simply an extension of its one-word counterpart; the operation is carried out through the same main accumulator, the J register, when applicable. The second type of two-word memory instruction is developed by a modification of the effective register selection. All these operations concern the K register as the main accumulator.

The fundamental operations that parallel the descriptions of the individual instructions are developed for one-word instructions. Therefore, the logic expressions are written in terms of the J register as the main accumulator. For those two-word memory reference instructions which refer to the K register as the main accumulator, Table 4-76 lists the fundamental operations that replace the equivalent operations for the J register. For example, in Table 4-90, the instruction list for the memory skip subgroup, fundamental operation F is carried out for both the SMJ and TWSMJ instructions. However, for instruction TWSMK, this operation is replaced by Operation D of Table 4-76; it can be seen that both operations occur during period EP45.

4.4.6.1 MEMORY SKIP SUBGROUP. For this subgroup Figures 4-62 and 4-63 are the block diagrams. Instruction SMJ has a separate block diagram (Figure 4-62) but the other two instructions are illustrated by Figure 4-63. Figure 4-64 is the timing diagram for the SMJ instruction and Figure 4-65 is the timing diagram for the ISZ and DSZ instructions. Table 4-89 is the Event Summary for the subgroup and Table 4-90 is the instruction listing in terms of the Fundamental Operations.

Instructions of this subgroup are conditional. In each case the content of the memory register is switched through the TS multiplexer into the adder via the memory data register whose input is automatically selected when no other address selection is made. If the tested condition for the particular instruction is met the program counter is incremented. Descriptions of the individual instructions follow below.

SMJ 2400 (TWSMJ 0240, TWSMK 0250) - This instruction compares the content of the effective memory location with the content of the J register (2). If the result of the comparison indicates that they are equal the next location is accessed; otherwise, the next location is skipped. In either case, the content of the program counter (8) and the J register (2) is not changed.

The operand previously loaded into the memory data register (1) during period PU3 of the common execute phase (Operation K, Table 4-55), is switched through the TS multiplexer (3) from the memory data register (1), via busses M00 through M11, to the adder (6).

During period EP45, the content of the J register (2) is also switched through the MX multiplexer (4) via busses J00 through J11, to the adder (6) via busses MX00 through MX11.

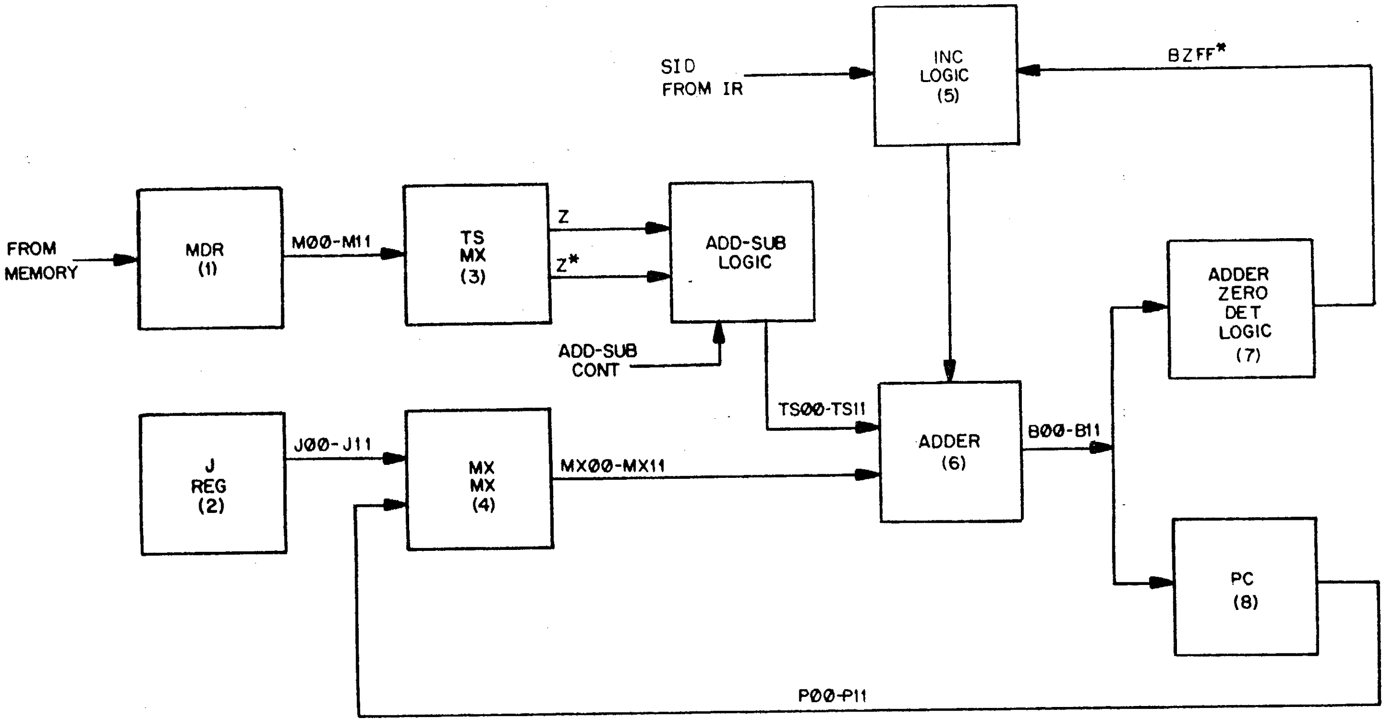


Figure 4-62. Memory Skip Subgroup, SMJ Instruction, Block Diagram

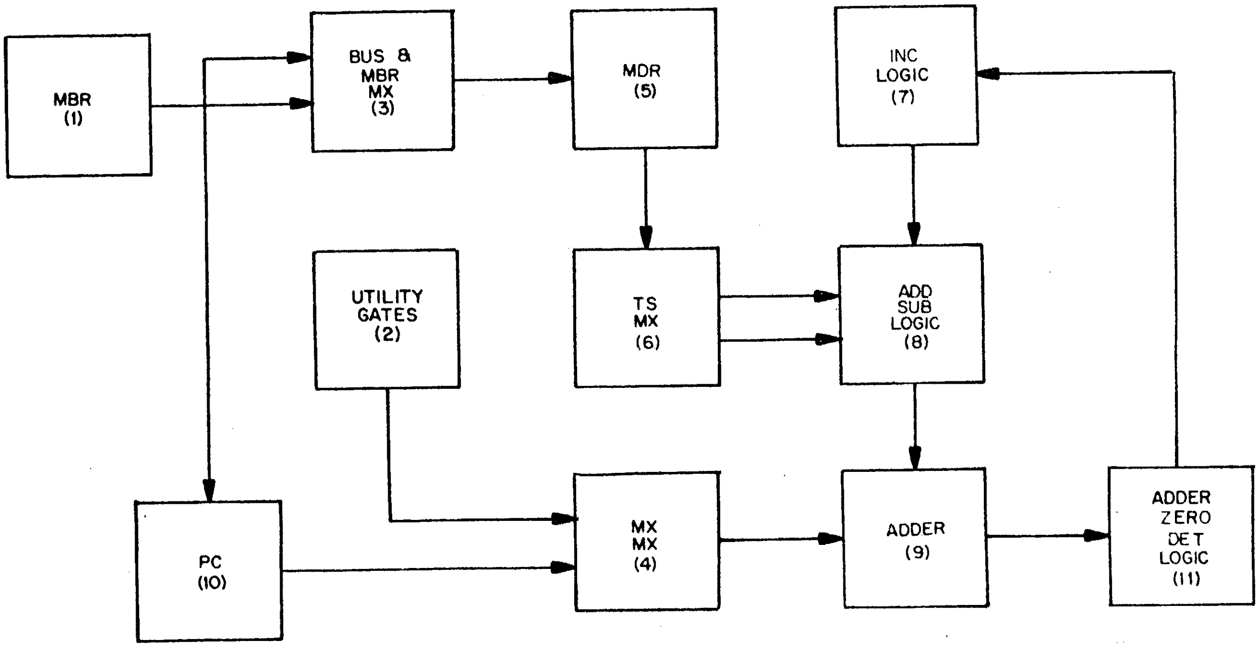
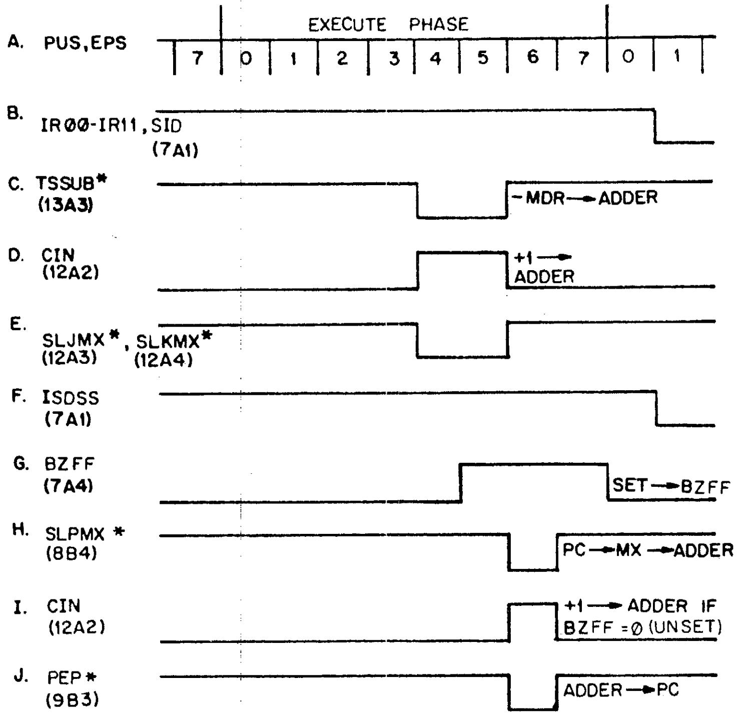


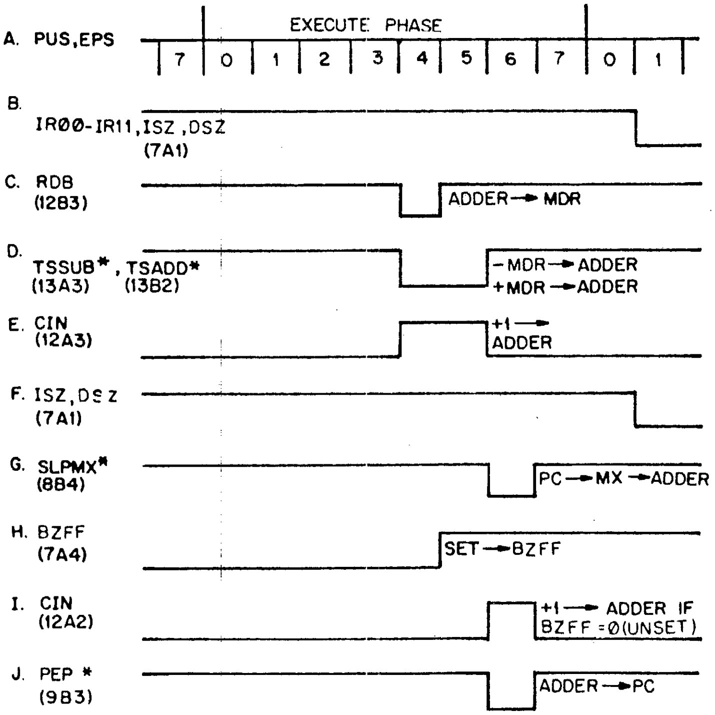
Figure 4-63. Memory Skip Subgroup, ISZ and DSZ Instructions, Block Diagram



NOTE

If BZFF unset, waveforms D and I will be combined.

Figure 4-64. Memory Skip Subgroup, SMJ Instruction, Timing Diagram



NOTE

If BZFF unset, waveforms E and I will be combined.

Figure 4-65. Memory Skip Subgroup, ISZ and DSZ Instructions, Timing Diagram

Table 4-89. Memory Skip Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
SMJ	0240	D	EP45	MDR - 1→ADDER
	0250	F	EP45	J→MX→ADDER
	2400	G	PU5B	TEST: B = 0, SET→BZFF
		H	EP6	PC→MX→ADDER
		I	EP6	IF BZFF = 0, +1→ADDER
		K	EP6	ADDERS→PC
DSZ	0300	A	EP4	ADDER→MDR
	3000	C	PU45	MDR→TS→ADDER
		E	EP4	-1→ADDER
		G	PU5B	TEST COMPARISON: B = 0, SET→BZFF
		H	EP6	PC→MX→ADDER
		J	EP6	IF BZFF = 1, +1→ADDER
		K	EP6	ADDERS→PC
ISZ	0340	A	EP4	ADDER→MDR
	3400	B	EP4	+1→ADDER
		C	PU45	MDR→TS→ADDER
		G	PU5B	TEST COMPARISON: B = 0, SET→BZFF
		H	EP6	PC→MX→ADDER
		J	EP6	IF BZFF = 1, +1→ADDER
		K	EP6	ADDERS→PC

The TS multiplexer (3) permits selection of the data word or the complement of the data word through the add-subtract logic. In this case, the complement of the data word is added to the content of the MX multiplexer (4), but the result is incremented by one through a carry input from the increment logic (5) to obtain a 2's complement value.

Instruction TWSMJ is simply the two-word extension of the SMJ instruction. Instruction TWSMK makes use of the K register; the logic expression is shown in Table 4-76.

Next, the resultant sum is transferred via adder busses B00 through B11, to the adder zero logic (7). If the adder zero detector logic detects an adder = 0 condition, the adder zero detector latch is set. When period EP6 occurs the content of the program counter (8) is transferred to the MX multiplexer (4) and adder (6).

The status of the adder zero detector latch is tested by the data control logic when period EP6 occurs. If the adder zero detector latch is in its set state (high-level BZFF signal) indicating that memory data is equal to J register data, the content of the adders (PL data) will not be altered because the content of the program counter (8) has already been incremented by one count during period EP1 (Tables 4-54 and 4-55). If the adder zero detector latch remains in the clear state, indicating that memory data is not equal to J register data, the adder (6) is incremented through the increment logic (7) again.

Table 4-90. Memory Skip Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (0300) (0340) (3000) (3400)	EP4	ADDER→MDR 1. (↓I00) (↑I01) (↑I02) (↓I03)→↓DSZ*/7A1 2. (↓I00) (↑I01) (↑I02) (↑I03)→↓ISZ*/7A1 3. ↓DSZ*→↑DSZ/7A1 4. ↓ISZ*→↑ISZ/7A1 5. ↓ISZ* + ↓DSZ*→↑ISZDSZ/7A2 6. (↑ISZDSZ) (↑EP4)→↓BMEM*/12B2 7. ↓BMEM*→↑PEM/12B3 8. (↑PEM) (↑REGCLK)→↓CPM*/12B3 9. ↓CPM* SETS→↑IND*/6A2 10. ↓BMEM*→↑RDB→↓RDB*/12B3 = ADDER→MDR	The content of the memory register replaces the content of the MDR.
B. (0340) (3400)	EP4	+1→ADDER 1. (↑ISZ) (↑EP4)→↓CIN3*→↑CIN/12A2	A carry is propagated through to the adder.
C. (0300) (0340) (3000) (3400)	PU45	MDR→TS→ADDERS 1. (↑ISZDSZ) (↑EPH)→↓IDEPH*/13B1 2. ↓IDEPH*→↑TSEXT/13B1 3. (↑TSEXT) (↑PU45)→↓TSADD*/13B1	Addition is selected for the output of the TS multiplexer.
D. (0240) (0250) (2400)	EP45	-MDR+1→ADDER 1. (↓I00) (↑I01) (↓I02) (↑I03)→↓SID*/7A1 2. ↓SID*→↑SID/7A1 3. ↓SID* + ↓DSZ* + ↓ISZ*→↑ISDSS/7A2 4. (EP45) (↑SID)→↓TSSUB*/13B3 5. ↓TSSUB*→↑TSSUB/13B4 6. (↑TSSUB) (↑DCIN*) (↑OP2*)→↓CIN2*/12A1 7. ↓CIN2*→↑CIN/12A2	2s complement subtraction is selected for the output of TS multiplexer.
E. (0300) (3000)	EP4	-1→ADDER 1. (↑DSZ) (↑EP4)→↓SLUMX*/8A4 2. ↓SLUMX*→↑MXEN→↓MXEN*/9A2 3. ↓SLUMX*→↑MXS0/9B1 4. ↑U00-U11/16A2→MX(7777_8)→ADDER	A negative carry is propagated through the adder.
F. (0240) (0250) (2400)	EP45	J→MX→ADDER 1. (↑EP45) (↑SID) (↑JCOM)→↓SLJMX*/12A3 2. ↓SLJMX*→↑MXS2/9A1 3. ↓SLJMX*→↑MXEN→↓MXEN*/9A2 = JR→MX→ADDERS	The content of the J-register is input to the MX multiplexer.

Table 4-90. Memory Skip Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
G.	PU5B	IF B = 0, SET→BZFF	
(0240)		1. ↓DSZ*/7A1→↑ISDSS/7A2	If the adder output is
(0250)		2. (↑ISDSS/7A2) (↑TW*/7A1) (↑IND*/6A2)→	all zeros, the BZFF
(0300)		↓SETBZ*/7A2	latch is set.
(0340)		3. ↓SETBZ*→↑BZLOG/7A3	
(2400)		4. (↑BZLOG/7A3) (↑BZERO/7A4) (↑REGCLK) (↑PU5B)→	
(3000)		↓SBZFF* = SET→BZFF (↑BZFF/7A4, ↓BZFF*)	
(3400)			
H.	EP6	PC→MX→ADDER	
(0240)		1. (↑ISDSS) (EP6)→SLPMX*/8B4	The content of the
(0250)		2. ↓SLPMX*→↑MXS1/9A1	program counter is
(0300)		3. ↓SLPMX*→↑MXEN→↓MXEN*/9A2 =	input to the MX
(0340)		PC→MX→ADDER	multiplexer.
(2400)			
(3000)			
(3400)			
I.	EP6	IF BZFF = 0, +1→ADDER	
(0240)		1. (↑SID) (↑BZFF*)→↓MRSKP*→↑JPSMRF/12A1	If the BZFF latch is
(0250)		2. (↑JPSMRF) (↑EP6)→↓CIN2*→↑CIN/12A1 =	unset, the adder is
(2400)		ADDERS→+1	incremented.
J.	EP6	IF BZFF = 1, +1→ADDER	
(0300)		1. (↑ISZDSZ) (↑BZFF)→↓MRSKP*/12A1	If the BZFF latch is
(0340)		2. ↓MRSKP*→↑JPSMRF/12A1	set, the adder is
(3000)		3. (↑JPSMRF) (↑EP6)→↓CIN2*→↑CIN/12A1	incremented.
(3400)			
K.	EP6	ADDER→PC	
(0240)		1. ↓ISDSS*→↑IDJS*/9B3	The content of the
(0250)		2. (↑IDJS*) (↑EP6)→↓PEP*→↑PEP/9B1	adder replaces the
(2400)		3. (↑PEP) (REGCLK)→↓CPP*/9B2 = ADDERS→PC	content of the
(3000)			program counter.
(3400)			

The resultant data in both cases is returned to the program counter (8) to obtain the location of the next instruction.

ISZ 3400 (TWISZ 0340) - This instruction admits the content of the effectively addressed memory location from the memory data register into the adder and increments it by a count of one. If the result of this incrementation is zero, the next instruction in the program is skipped as the result of a subsequent incrementation of the content of the program counter. If the incrementation of the adder does not result in zero, the program counter is not incremented and the next instruction in the program is processed.

Initially, the content of the memory data register (5, Figure 4-63), which contains the operand of the effectively addressed memory location, is switched from the memory data register (5) through the TS multiplexer (6) to the adder (9) through the add-subtract logic and incremented. In the absence of any other select signals, this is the normal switched state of the TS multiplexer. As a result of incrementation through the increment logic (7), if the resultant content of the adder (9) is zero, the detect zero logic (9) sets the BZFF latch. Also, simultaneously, the incremented memory data is switched from the adder (9) through the bus and memory buffer multiplexer (3) back into the memory data register (5), so that the new value of memory data can be written back into the same memory location from which it was fetched during period PU3.

When period EP6 occurs, the content of the program counter (10) is switched through the MX multiplexer (4) to the adder (9). The status of the detect zero latch is tested by the detect zero logic (11). If the detect zero latch is in its cleared state, indicating that the content of the adder was not zero as a result of the previous incrementation, the content of the adder is not altered because the content of the program counter (10) has already been incremented by a count of one during EP1 (Tables 4-54 and 4-55). If the detect zero latch is in its set state, indicating that the previously incremented memory data is now zero, the adder is incremented by one again causing the content of the program counter to be incremented by a total count of two during the processing operations. The result left in the adder replaces the content of the program counter. The TWISZ instruction is simply the two-word extension of the ISZ instruction.

DSZ 3000 (TWDSZ 0300) - This instruction is quite similar to the ISZ instruction in principle. Instead of adding one to the content of the effect memory location, a subtraction of one is required before the resulting quantity is tested for the zero condition. The test for zero and incrementation of the program counter are identical to the ISZ instruction. However, carrying out the subtraction of one is done in an indirect way. As can be seen from Table 4-89, Fundamental Operation E selects the utility gates as input to the MX multiplexer. Since no controlled inputs are admitted to the utility gates at this time, all utility gate inputs take on a 1 value; thus, the octal number 7777 is switched to the adder through the MX multiplexer. It can be verified with paper and pencil that any octal number x added to octal 7777 will produce X - 1. The fact that octal 7777 is equal to -1 can be intuitively seen by considering the definition of -1; minus one is the number which produces a zero when 1 is added to it.

4.4.6.2 MEMORY TO J SUBGROUP. For this subgroup Figure 4-66 is the block diagram, Figure 4-67 is the timing diagra, Table 4-91 is the Event Summary for the subgroup and Table 4-92 is the instruction listing of Fundamental Operations.

Instructions of this subgroup take the content of the effectively addressed memory location and the content of the J register, perform operations on them, and place the result in the J register.

LDJ 5000 (TWLDJ 0500, TWLDK 0510) - This instruction loads the J register (2) with the content of the effectively addressed memory. When selection is not specified for either multiplexer, the content of the memory data register (1) is the input to the TS multiplexer

(3) and zero is contributed by the MX multiplexer (5). Because addition is selected as an output from the TS multiplexer, the content of the address memory location is transferred via the adder (6) to the enabled J register (2). Because only one register takes part in the addition, overflow cannot occur during the execution of this instruction. For instruction TWLDK refer to Table 4-76, Fundamental Operations E.

ADJ 4400 (TWADJ 0440, TWADK 0450) - This instruction adds the content of the effectively addressed memory location to the content of the J register (2) and stores the result in the J register. This instruction is difference from the LDJ instruction only in two respects; first, the content of the J register is also selected, as input to the adders (6) via the MX multiplexer. Secondly, fundamental operation F is required because overflow can result from addition of the two registers. All other processing events are identical for the ADJ and LDJ instructions. For instruction TWADK refer to Table 4-76 Fundamental Operations D and E.

SBJ 4000 (TWSBJ 0400, TWSBK 0410) - This instruction subtracts the content of the effectively addressed memory location from the content of the J register (2) and stores the result in the J register. This instruction is different from the ADJ instruction only in one respect; the selected arithmetic operation is subtraction as a direct consequence of the instruction definition. All other events are identical for the SBJ and ADJ instructions, including the possibility of overflow. If overflow occurs, the overflow flip-flop is complemented and can be tested by programming the appropriate instruction. For instruction TWSBK refer to Table 4-76 Fundamental Operations E and F.

The remaining one-word memory reference instructions have more differences than similarities in their individual execute subphases. Therefore, they are individually described.

4.4.6.3 STJ 5400 (TWSTJ 0540, TWSTK 0550). This instruction replaces the content of the effectively addressed memory location with the content of the J register (2, Figure 4-66). The original content of the memory location is replaced but the content of the J register is retained. At the outset, the content of the J register (2) is switched through the MX multiplexer (5) to the adder. In the absence of TS multiplexer selection, its output is equivalent to all zeros so that no summation takes place in the adder. Next, the content of the adder is switched through the bus and memory buffer multiplexer (8) to the memory data register (1) where it is stored for the subsequent memory write operation at PU5. In the absence of any memory buffer register is switched through to the memory data register. However, because signal RDB* is low-level, the adder output is switched through. Then, at period PU5, the content of the memory data register (1) is written into memory. The TWSTJ and the TWSTK instructions are the two-word variations of the STJ instruction. For operations that affect the K register, refer to Table 4-76. Table 4-93 is the STJ instruction Event Summary and Table 4-94 is the list of Fundamental Operations.

4.4.6.4 JMP 6000 (TWJMP 0600). The jump instruction is the only memory reference instruction that requires only one phase to execute. Thus, this instruction requires only a single basic phase. The events described in Tables 4-19 and 4-20 through period BP5, are identical for the common basic phase and this instruction. The events that occur during

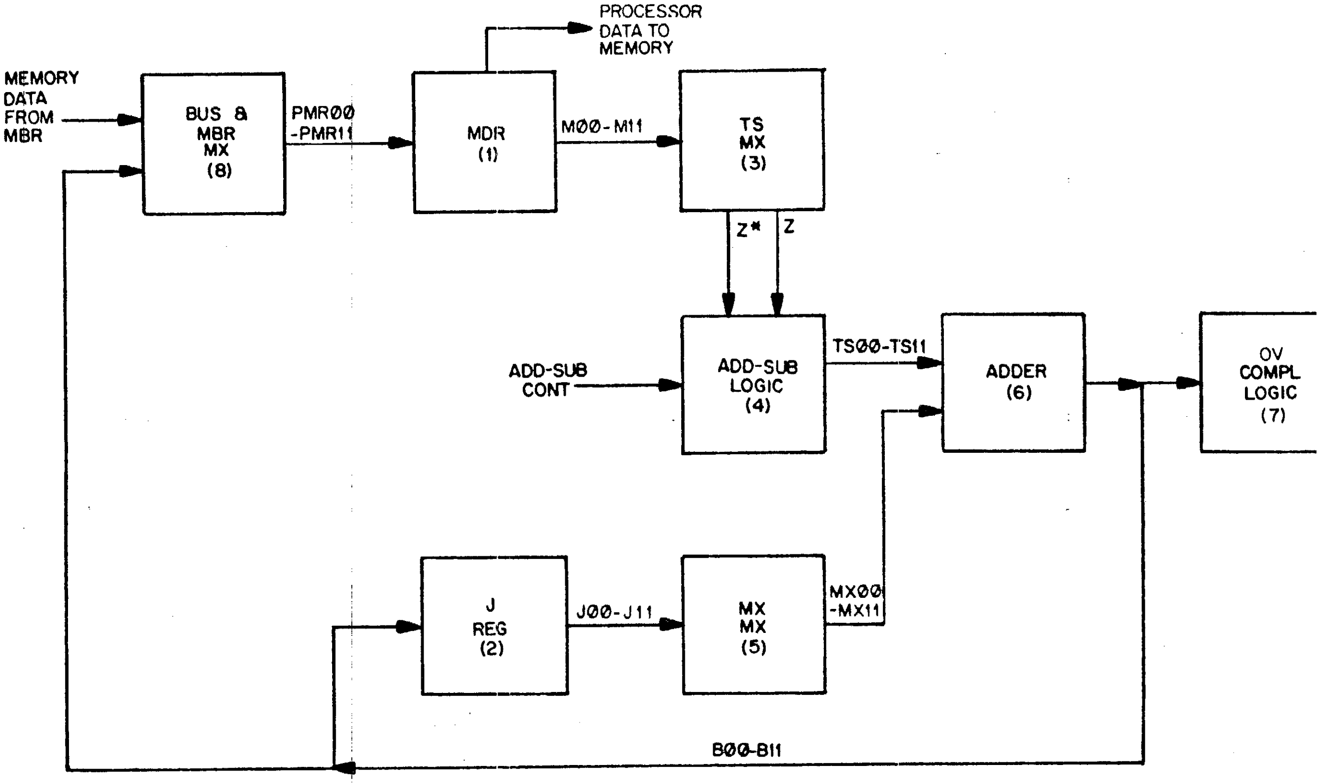


Figure 4-66. Memory to J Subgroup, Block Diagram

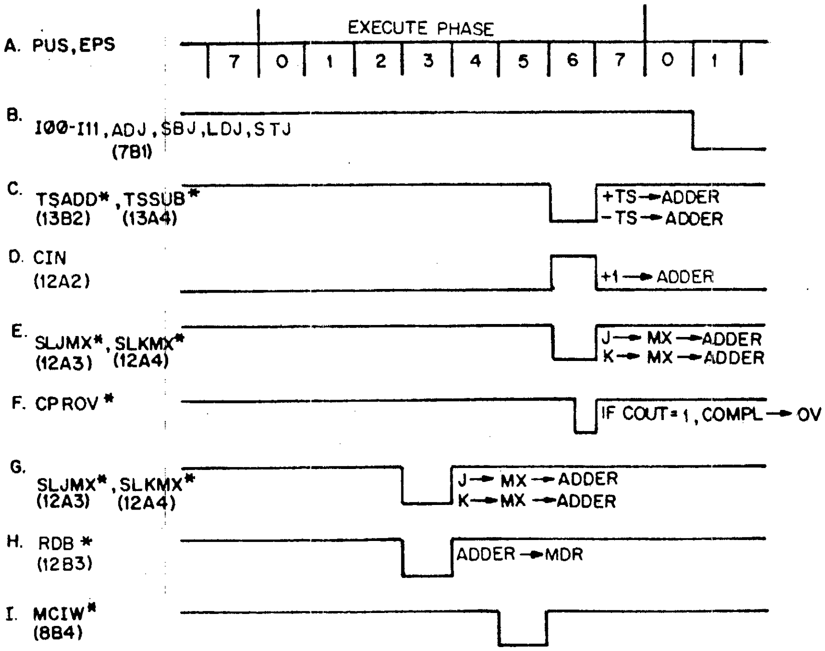


Figure 4-67. Memory to J Subgroup, Timing Diagram

Table 4-91. Memory to J Subgroup, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
LDJ	5000	A	PU3	INITIALIZE
		B	EP6	+TS→ADDER
		E	EP6	ADDER→J
ADJ	4400	A	PU3	INITIALIZE
		B	EP6	+TS→ADDER
		D	EP6	J→MX
		E	EP6	→ADDER→J
		F	EP6	IF OV→COMP
SBJ	4000	A	PU3	INITIALIZE
		C	EP6	TS SUB SELECT
		D	EP6	J→MX→ADDER
		E	EP6	ADDER→J
		F	EP6	IF OV→COMP

Table 4-92. Memory to J Subgroup, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (0400) (0410) (0440) (0450) (0500) (0510) (4000) (4400) (5000)	PU3	INITIALIZE 1. (↑I00*) (↓I01) (↓I02) (↓I03)→↓SBJ*/7B1 2. (↑I00*) (↓I01) (↓I02) (↑I03)→↓ADJ*/7B1 3. (↑I00*) (↓I01) (↑I02) (↓I03)→↓LDJ*/7B1 4. ↓ADJ* + ↓LDJ*→↑LDJADJ/7B2 5. ↓ADJ* + ↓SBJ*→↑ADJSBJ/7B2 6. ↓ADJ* + ↓LDJ* + ↓SBJ*→↑ADSBLD/7B2	Preselects combined operation groups.
B. (0440) (0450) (0500) (0510) (4400) (5000)	EP6	SELECT→+TS→ADDER 1. (↑LDJADJ) (↑EP6)→↓TSADD*/13B2	Select TS output for addition.
C. (0400) (0410) (4000)	EP6	SELECT-TS→ADDER 1. ↓SBJ*→↑SBJ/7A2 2. (↑SBJ) (↑EP6)→↓TSSUB*→↑TSSUB/13B4 3. (↓DCIN*/12A1) (↑TSSUB/13B4) (↑OP2*/7A2)→ ↓CIN2*/12A2→↑CIN/12A2	Selects TS multiplexer for subtraction.

Table 4-92. Memory to J Subgroup, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
D. (0400) (0410) (0440) (0540) (4000) (4400)	EP6	J→MX→ADDER 1. ↓ADJ* + ↓SBJ*→↑ADJSBJ/7B2 2. (↓ADJSBJ) (↑JCOM) (↑EP6)→↓SLJMX*/12A3 3. ↑SLJMX*→↑MXS2/9B1 4. ↓SLJMX*→↑MXEN→↓MXEN*/9A2	The content of the J-register is input to the MX multiplexer.
E. (0400) (0410) (0440) (0450) (0500) (0510) (4000) (4400) (5000)	EP6	ADDER→J 1. ↓LDJ* + ADJ* + SBJ*→↑ADSBLD/7B2 2. (↑ADSBLD) (↑JCOM)→↓LDLOG*→ LDJLOG/9A2 3. (↑LDJLOG) (↑EP6)→↓PEJ*→↑PEJ/9A4 4. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4	The content of the adder replaces the content of the J-register.
F. (0400) (0410) (0440) (0450) (4000) (4400)	EP6	IF COUT = 1, COMPL→OV 1. (↑ADJSBJ) (↑COUT) (↑EP6)→↓AROV*/12B4 2. ↓AROV*→↑CPOV/11B3 3. (↑CPOV) (↑REGCLK)→↓CKOV*/11B4 = COMPL→OV	If overflow, complement the overflow register.

Table 4-93. STJ Instruction, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
STJ	5400	A	EP3	J→ADDER
		B	EP3	ADDER→MDR
		C	PU5	MDR→MR

and after BP6 are described in Table 4-95, the Event Summary, and Table 4-96, the list of Fundamental Operations. Figure 4-68 is the jump instruction block diagram and Figure 4-69 is the timing diagram.

When this instruction is processed, the current content of the program counter (5) is replaced with the effective address. The effective address is obtained during period BP6 rather than the normal EP0 period (Tables 4-54 and 4-55). During this period, the content of bits 106 through 111 of the instruction register (1), the relative address, are summed with the current content of the address register (6) when the instruction register data (1) is switched through the TS multiplexer (3) to the adder (4). Simultaneously, the content of the address register (6) is switched through the MX multiplexer (4) to the adder.

Table 4-94. STJ Instruction, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A. (0540) (0550) (5400)	EP3	J→ADDER 1. (↑I00*) (↓I01) (↑I02) (↑I03)→↓STJ*/7B1 2. ↓STJ*→↑STJ/7B1 3. (↑STJ) (↑JCOM)→↓STJCM*→↑STJCM/12A3 4. (↑STJCM) (↑EP3)→↓SLJMX*/12A3 5. ↓SLJMX*→↑MXS2/9B2 6. ↓SLJMX*→↑MXEN→↓MXEN*/9A2	Content of J-register is input to adder.
B. (0540) (0550) (5400)	EP3	ADDER→MDR 1. ↑STJ→↑STJJPS/12B2 2. (↑STJJPS) (↑EP3)→↓BMEM*/12B3 3. ↓BMEM*→↑PEM/12B3 4. (↑PEM) (↑REGCLK)→↓CPM*/12B3 5. ↓BMEM*→↑RDB→↓RDB*/12B3	Content of adder replaces content of memory data register.
C. (0540) (0550) (5400)	PU5	MDR→MR 1. (↑WTRL*) (↑PU5B)→↓PU5W*→↑PU5W/8B3 2. (↑PU5W) (↑ADFF0*)→↓MCIW*/8B4	Content of the MDR replaces the content of the memory.

Table 4-95. JMP Instruction, Event Summary

Ref.	Period	Event
A	BP6B	REL ADR→TS
B	BP6B	SELECT→ +TS
C	BP6B	SELECT→ -TS
D	BP6B	AR→MX→ADDER
E	BP6B	ADDER→PC
F	BP6	SET→DONE
G	PU7	SET→BASIC PHASE

If bit 5 of the instruction is unset, these data are added (Operation B, Table 4-96); if bit 5 is set, however, these data are subtracted (Operation C, Table 4-96). The result replaces the content of the program counter (PC) and the basic phase is reset so that the current program counter content will become the address for the next instruction.

With the exception of admitting address data to the program counter, the operations which normally take place during a single-word jump instruction when the auto-index function is used are inhibited in favor of the auto-index operations. Refer to the description of the auto-index function for a discussion of these anomolies. The events that take place during the execution of a jump instruction can be summarized as follows.

a. The content of the program counter is admitted into the address register to get the address of the current jump instruction.

Table 4-96. JMP Instruction, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP6B	REL ADR→TS→ADDER 1. (↑I00) (↑I01) (↓I02) (↓I03)→↓JMP*/7B2 2. ↓JMP*→↑JMP/7B2 3. (↑TWFF*) (↑INDFF*)→↓MPASS*→ ↑MPASS/13B4 4. (↑JMP) (↑FLLOC*)→↓JMPFL*→ ↑JMP*FL/7B4 5. ↓JMPFL*→↑JMPDLS/7B4 6. (↑JMPDLS) (↑MPASS) (BP6B)→↓SLITS*→ ↑TSS0/13B4	Instruction register bits I06 through I11 are switched through the TS multiplexer to the adder.
B.	BP6B	SELECT→+TS(Bit 5 = 0) 1. ↓TWFF* + ↓INDFF*→↑MPASS*→ ↓MPASS/13B4 2. (↑MPASS) (↑I5B*) + ↓ADDAR*→ ↑ADDEP0/13B1 3. (↑JMPDLS) (↑ADDEP0)→↓TAD6*→ ↑TAD6/13B1 4. (↑TAD6) (↑BP6B)→↓TSADD*/13B2	During a normal jump, the relative address is admitted into the adder. During an auto-indexed jump, the content of the memory data register is admitted into the adder.
C.	BP6B	SELECT→-TS(Bit 5 = 1) 1. (↑JMPDLS) (↑MPASS) (↑I05X)→↓SUB6*→↑SUB6 (↑SUB6) (↑BP6B)→↓TSSUB*→↑TSSUB/13B4 2. (↑TSSUB) (↑DCIN*) (↑OP2*)→ ↓CIN2*→↑CIN/12A2	The 2s complement of the relative address is admitted into the adder.
D.	BP6B	AR→MX→ADDER 1. (↑JMP) (↑FLLOC*)→↓JMPFL*→ ↑JMP*FL/7B4 2. (↑JMP*FL) (↑BP6B) (↑MPASS)→↓CSELA/8A1 3. ↓CSELA→↓SLAMX*/8A1 4. ↓SLAMX*→↑MXEN→↓MXEN*/9A2	The content of the address register is admitted into the adder.
E.	BP6B	ADDER→PC 1. (↑JMP) (↑BP6B)→↓JMP6*→↑MPEP→ ↓PEP*/9B3 2. ↓PEP*→↑PEP/9B1 3. (↑PEP) (↑REGCLK)→↓CPP*/9B2	The content of the adder is admitted into the program counter.
F.	BP6	SET→DONE 1. (↑IND*) (↑TWJJS*) (↑JMP)→↓JMPIND*/7B3 2. ↓JMPIND*→↑SDONE 3. (↑SDONE) (↑BP6)→↓SDONE*/7B4 = SET→DONE (↑DONE/7B4, ↓DONE*)	The done latch is set.
G.	PU7	SET→BASIC PHASE 1. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→ ↑RBPH→↓RBPH*/5B1 2. (↓RBPH*) (↓CPPU*) (↑PU7) = SET→ BPH (↑BPH/5B2, ↓EPH)	The basic phase is set.

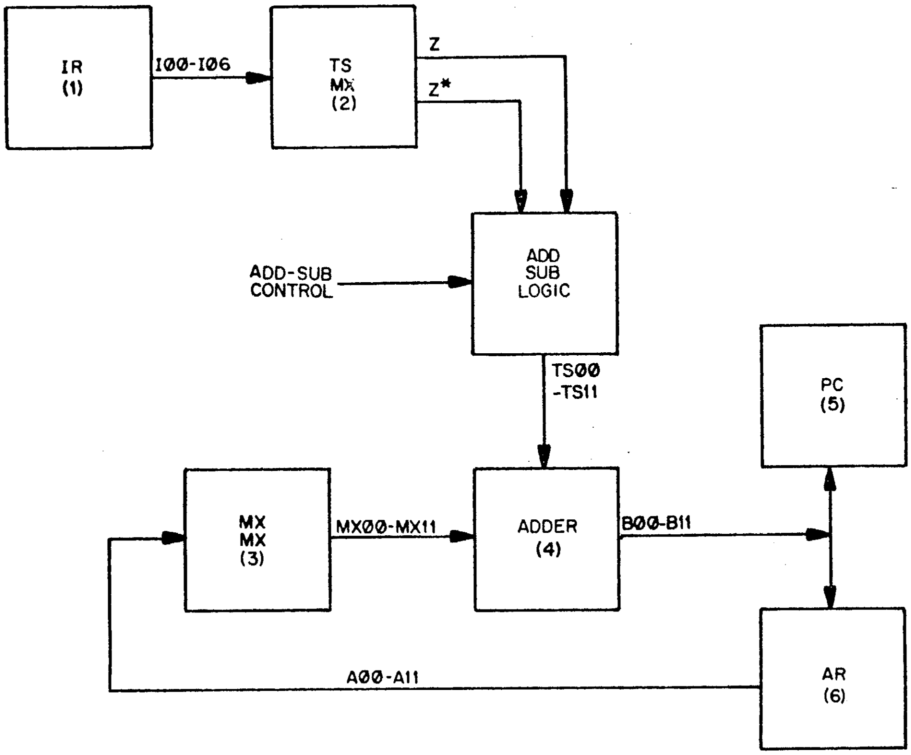


Figure 4-68. JMP Instruction, Block Diagram

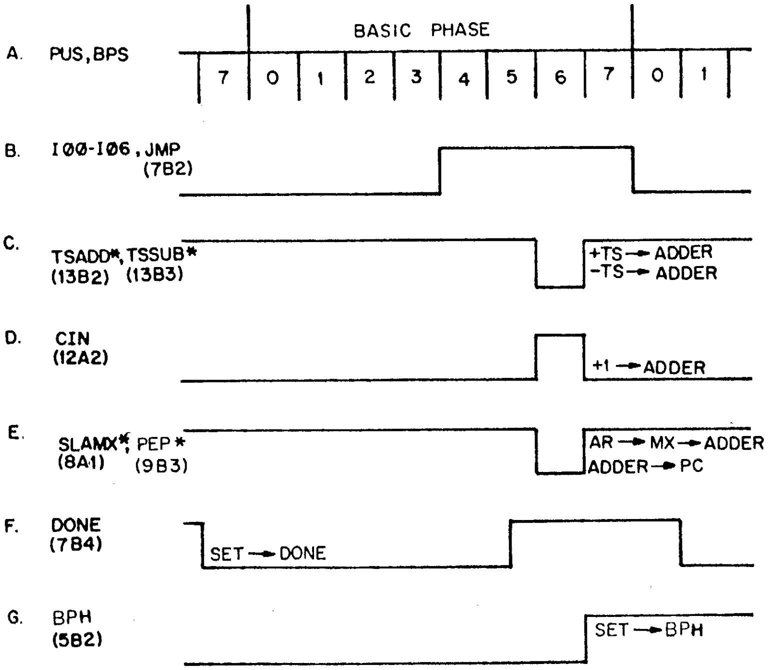


Figure 4-69. JMP Instruction, Timing Diagram

b. The content of the memory register is admitted into the memory buffer register.

c. The content of the memory buffer register is admitted into the memory data register and into the instruction register.

d. The content of the memory data register is written back into memory located at the address specified by the content of the address register.

e. The relative address, bits 106 through 111, is fetched from the instruction register, through the TS multiplexer, to the adder.

f. The current address is switched through the MX multiplexer to the adder and summed with the relative address. The result is admitted into the program counter to get the location of the next instruction to be processed.

g. The basic phase for the next instruction is set.

h. When the next instruction is processed, the address register gets the current content of the program counter as the address of the next location to be accessed.

4.4.6.5 JPS 6400 (TWJPS 0640). This instruction increments the content of the program counter to obtain the next address and stores this next address in the effective memory location. Next, the effective address is loaded into the program counter and the address register. When the next instruction is processed, the new content of the program counter becomes it address. This instruction has a basic and an execute phase. The baisc phase is identical to the common basic phase (Tables 4-19 and 4-20). However, the execute phase is modified so that the next address can be stored into the effectively addressed memory location. In addition to the normal common execute phase events that take place (Tables 4-54 and 4-55), the next address in the program is admitted into memory and then later is written into the specified memory location. Table 4-97 is the Event Summary and Table 4-98 is the list of Fundamental Operations for the execute phase modified to process this instruction. Figure 4-70 is the block diagram and Figure 4-71 is the timing diagram.

When this instruction is processed, relative address data, bits 106 through 111, are switched from the instruction register (3) through the TS multiplexer (4) to the adder (6), to produce the effective address. If bit 5 of the instruction is set, the relative address is added to the current address which is switched from the address register (7) through the MX multiplexer to the adder (6). If bit 5 of the instruction is set, the relative address is subtracted from the current address. After the effective address is obtained, it replaces the current address in the address register. When period EP3 occurs, the content of the program counter (8), which now contains the next address, is admitted into the adder (6), and from the adder, it is switched through the bus and memory buffer multiplexer (1) to the memory data register (2) to be loaded into the effectively addressed memory location when period BP5 occurs. Next, the content of the adder, which contains the effective address, is switched through the MX multiplexer (5) to the adder (6), is incremented by a count of one,

Table 4-97. JPS Instruction, Event Summary

Mnemonic	Octal Code	Ref.	Period	Event
JPS	6400	A	EP3	PC→MX→ADDER
		B	EP3	ADDERS→MDR
		C	EP6	AR→MX→ADDER
		D	EP6	ADDERS→+1
		E	EP6	ADDERS→PC

Table 4-98. JPS Instruction, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	EP3	PC→MX→ADDER 1. (↑I00*) (↑I01) (↓I02) (↑I03)→↓JPS*/7B2 2. ↓JPS*→↑JPS/7B2 3. (↑JPS) (↑EP3)→↓JPS3*→↑SELP→ ↓SLPMX*/8B4 4. ↓SLPMX→↑MXS1/9B2 5. ↓SLPMX→↑MXEN→↓MXEN*/9A2	The content of the program counter is input to the MX multiplexer.
B.	EP3	ADDER→MDR 1. ↓JPS*→↑STJJPS/12B2 2. (↑STJJPS) (↑EP3)→↓BMEM*/12A2 3. (↓BMEM*)→↑PEM/12B3 4. (↑PEM) (↑REGCLK/12B3)→↓CPM*/12B3 5. ↓BMEM*→↑RDB→↓RDB*/12B3	The content of the adder replaces the content of the memory data register.
C.	EP6	AR→MX→ADDER 1. (↑JPS) (↑EP6)→↓SLAMX*/8A2 2. ↓SLAMX*→↑MXEN→↓MXEN*/9B2	The content of the address register is switched through the MX multiplexer to the adder.
D.	EP6	+1→ADDER 1. ↓JPS*/7B2→↑JPSMRF/12A1 2. (↑JPSMRF/12A1) (↑EP6/5A4)→ ↓CIN2*/12A2→↑CIN/12A2	The content of the adder is incremented by a count of one.
E.	EP6	ADDER→PC 1. ↓JPS*→↑IDSJ*/9B3 2. (↑IDSJ*) (↑EP6)→↓PEP*/9B3 3. ↓PEP*→↑PEP/9B1 4. (↑PEP) (↑REGCLK)→↓CPP*/9A2	The content of the adder is admitted into the program counter.

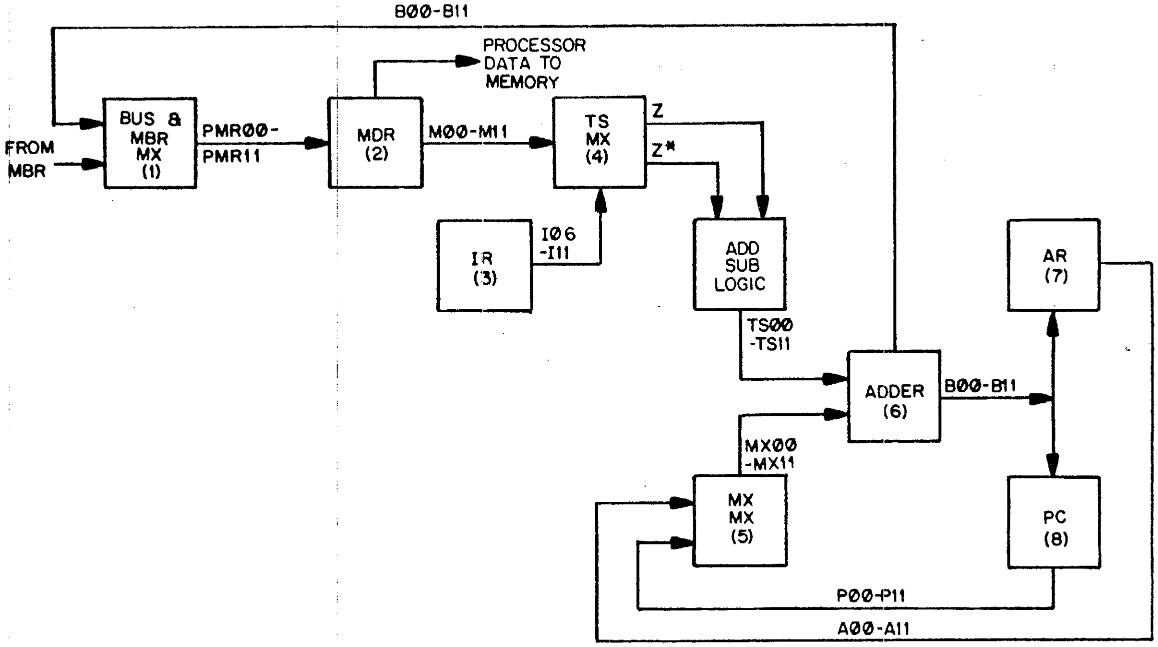


Figure 4-70. JPS Instruction, Block Diagram

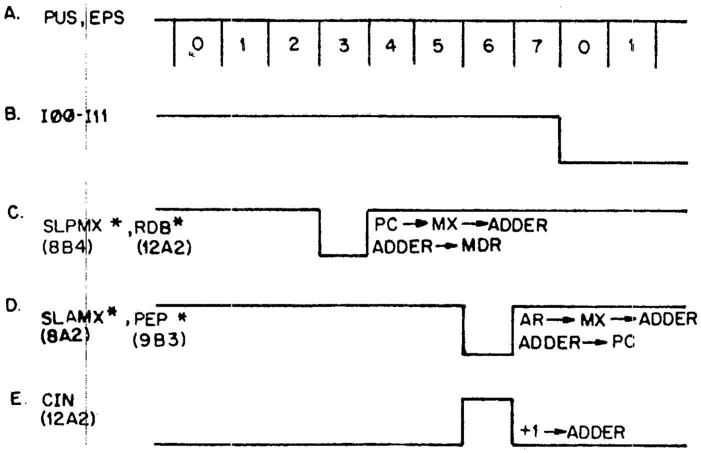


Figure 4-71. JPS Instruction, Timing Diagram

and then is admitted into the program counter (8) to obtain the new program location. The events that take place during the baisc phase of the jump-to-subroutine instruction are described in the discussion of the common basic phase. The events that take place during the execute phase of the jump-to-subroutine instruction can be summarized as follows.

a. The content of the address register, the current address, is summed with the relative address (bits 106 through 111 of the instruction register), and admitted into the address register to get the effective address.

b. The effective address is admitted into the address register.

c. The content of the program counter is incremented by a count of one.

d. The content of the memory register is read into the memory buffer register.

e. The content of the memory buffer register is switched into the memory data register through the bus and memory buffer multiplexer.

f. The content of the program counter (current address + 1) is admitted into the adder.

g. The content of the memory data register is replaced with the content of the adder so that the current memory location gets the incremented content of the program counter.

h. The content of the memory data register (PC+1) is written into the memory location specified by the effective address.

i. The content of the address register, the effective address, is incremented and admitted into the program counter to get the location of the next instruction to be processed.

4.4.7 AUTO INDEXING

In every memory field of the ND812, two locations are reserved for auto indexing. An auto-index register permits data stored in the addressed memory location to be incremented and then this incremented data becomes the address for the specified instruction in the operate field of the auto-index instruction. The two locations in each memory field reserved for the auto-index function are the first, and the last, or 0000_8 and 7777_8, but they may only be addressed as auto-index registers indirectly. If these locations are addressed directly, the instruction is processed as any other directly-addressed instruction. The auto-index instruction has a special format because it is possible to address only the first and last location of any given memory field through the one-word memory reference instruction; the relative address restrictions for forward and backward addressing do not apply for this instruction. Two-word auto-index instructions are not possible.

4.4.7.1 INSTRUCTION FORMAT. The auto-index instruction format requires that the relative address bits (bits I06 through I11) be set to zero. If bit 5 is set, the last location in the current memory field is referenced; if bit 5 is unset, the first location in the current memory field is referenced. Bit 4 must be set to obtain the auto-index function. When bit 4 is set, the content of the referenced auto-index location is incremented. Bit 4 of any memory reference instruction orders the fetch mode. When this bit is unset, the operand is fetched directly from memory; however, when this bit is set, the operand is fetched indirectly from memory. The incremented result is then loaded into the address register and restored to the referenced auto-index location. The content of the address register then becomes the effective address of the operand. The addressed operand must reside in the same memory field as the referenced auto-index location. The auto-index instruction has three phases, two basic and one execute. This is also true for the auto-index function for a jump (JMP) instruction. However, the jump instruction is a special case of auto-index function; it is separately described in paragraph 4.4.7.6.

Figure 4-72 is the block diagram, Figure 4-73 is the timing diagram. Tables 4-99, 4-100 and 4-101 are the Event Summaries and Tables 4-102 and 4-103 are the lists of Fundamental Operations.

4.4.7.2 BLOCK DIAGRAM DESCRIPTION. During the primary basic phase, the content of the program counter (9, Figure 4-72) is admitted into the address register and stored as the address of the current instruction. The instruction located at this current address, in this case an auto index, is loaded into the instruction register (3) and into the memory data register (2) through the bus and memory buffer multiplexer (1), from the memory buffer register (2) through the bus and memory buffer multiplexer (1), from the memory buffer register (not shown). During this phase, the IND, INDFF, and FLLOC latches are set. Setting of the IND and INDFF latches causes the ND812 to enter the indirect fetch mode. The setting of the FLLOC latch initiates the auto-index mode.

During the secondary basic phase, the indirect fetch, the content of the address register (current address) is not admitted to the MX multiplexer (6) for summation with instruction register (3) relative address bits I06 through I11. Rather, it only remains to identify whether the indirect address is located in the first or last memory location of the current memory field. This is done when bits I06 through I11 are switched from the instruction register (3), through the TS multiplexer (5) to the adder (7). At this time, if bit 5 of the current instruction is unset, no outputs are switched through the MX multiplexer (6), and the summation results in 0000_8 from the MX multiplexer (6) being added to 00_8 from the TS multiplexer (5). This summation results in the selection of the first location of the current memory field selected for the indirect address. If, on the other hand, bit 5 of the current auto-index instruction is set, the content of the unselected utility gates (4) is switched to the adder through the MX multiplexer (6). In the absence of any selection signals for the utility gates, the octal value 7777_8 is switched through the MX multiplexer (6) to the adder (7). When this data is added to the output from the TS multiplexer (only addition is possible), the last location of the current memory field is selected. This data is then admitted into the address register (8) to become the indirect pointer for the subsequent operation.

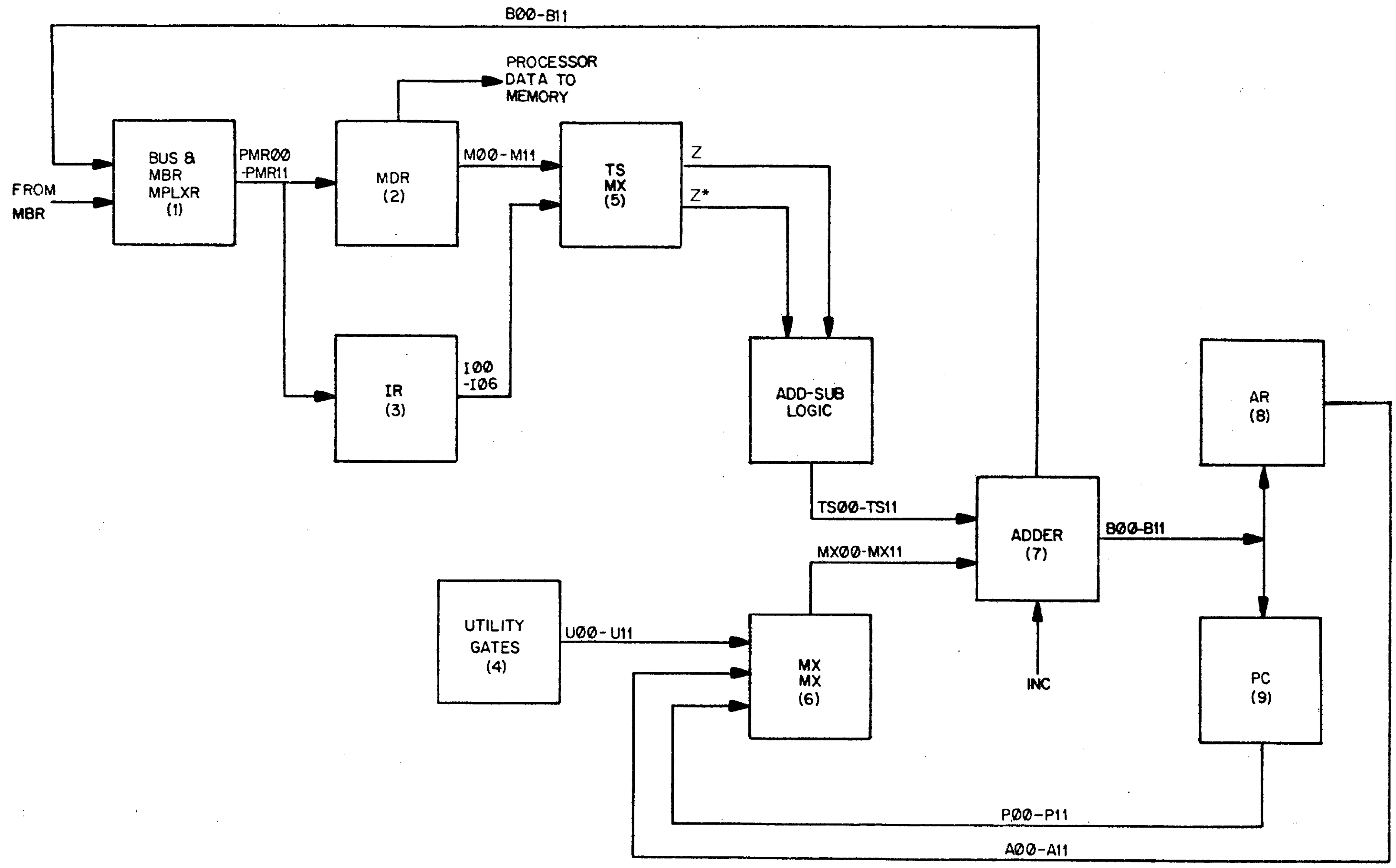


Figure 4-72. Auto-Index Function, Block Diagram

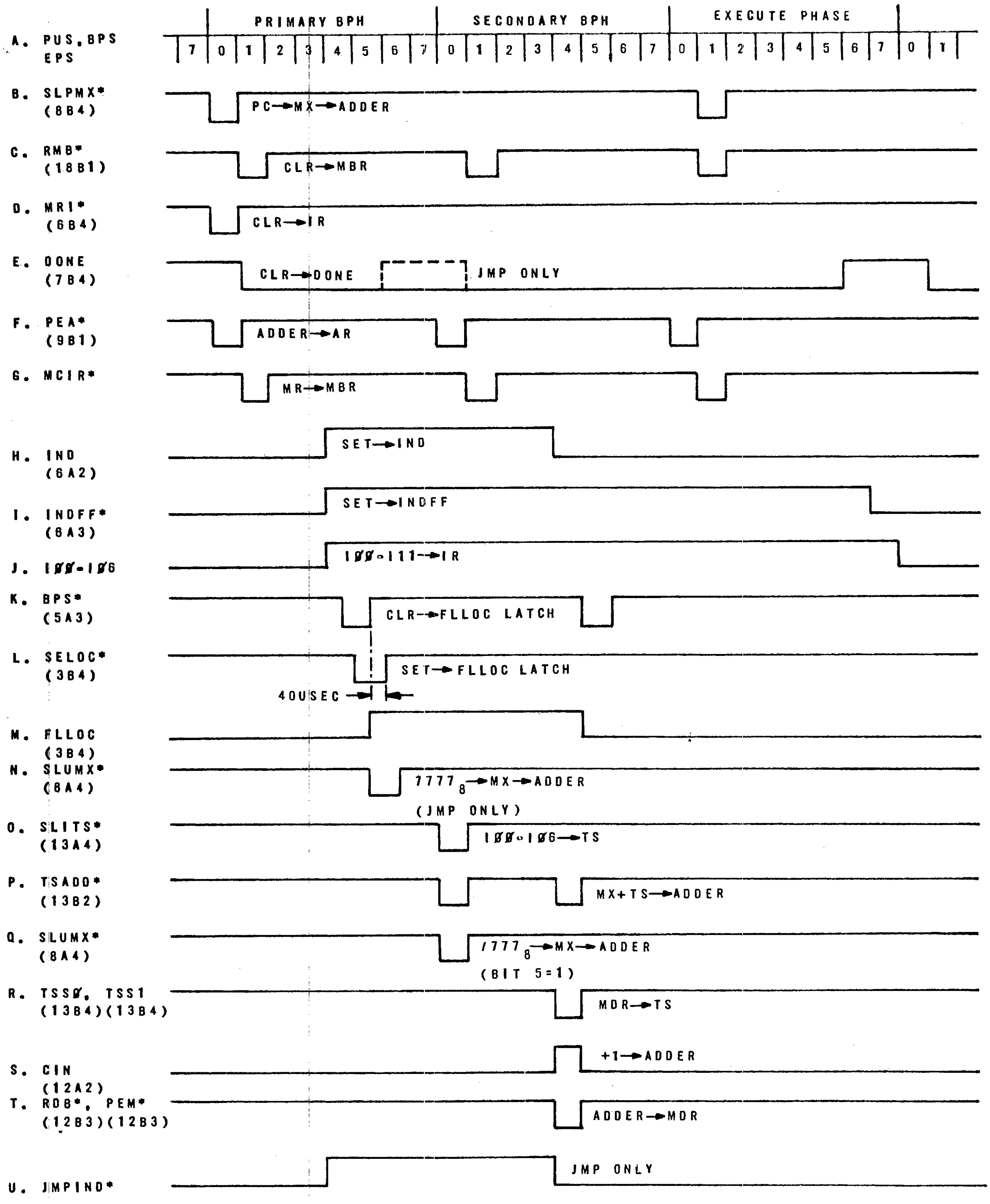


Figure 4-73. Auto-Index Function, Timing Diagram

Table 4-99. Auto-Index Function, Primary Basic Phase, Event Summary

Ref.	Period	Event	Indirect Basic Phase Ref. (Table 4-60)
A	PU7	CLR→LATCHES	A
B	BP0	PC→MX→ADDER	B
C	PU0B	ADDER→AR	C
D	PU0B	CLR→IR	D
E	BP1	CLR→DONE	E
F	PU1	CLR→MBR	F
G	PU2	MR→MBR	G
H	PU3	MBR→MDR	H
I	BP3	MBR→IR	I
J	BP3	INITIALIZE→AUTO INDEX	None
K	BP4	SET→IND, INDFF LATCHES	J
L	BP5*	CLR→FLLOC LATCH	None ‡1
M	BP5	SET→FLLOC LATCH	None
N	BP6B	7777_8→MX→ADDER	None ‡2
O	PU6B	ADDER→PC	None ‡3
P	PU7	SET→BASIC PHASE	None

‡1 Refer to timing diagram for operation of FLLOC latch

‡2 The address for the last location is transferred during BP6B for the JMP instruction only (Refer to Tables 4-95 and 4-96)

‡3 This operation occurs only during a jump (JMP) execute, refer to Table 4-97

Next, the content of the memory register, the auto-index address, located at the specified indirect address is admitted from the memory register, through the bus and memory buffer multiplexer (1) to the memory data register (2). Next, the address is incremented by a count of one and admitted into the address register (8) to obtain the actual address of the instruction; it is also switched from the adder (7) through the bus and memory buffer multiplexer (1) back into the memory data register (2), where it is subsequently written back into memory. At the end of the secondary basic phase, the FLLOC latch is cleared, and the execute phase is entered.

The execute phase (Table 4-101) of the auto-index function is identical to the execute phase for the single-word memory reference instruction; however, this phase is modified when an auto-index jump instruction is processed.

4.4.7.3 PRIMARY BASIC PHASE. The primary basic phase, Table 4-99, is similar to the single-word indirectly-fetched memory reference instruction primary basic phase. Because the indirect bit is set, during period BP4, the indirect latch is set. Additionally, the FLLOC latch is set at period BP5. Operations L and M, Table 4-99, appear to be in conflict because a period-5 derived pulse is used to both clear and set the FLLOC latch. However, Operation M does set the FLLOC latch. How this is done can be seen by referring to waveforms L and M of Figure 4-73, the timing diagram. When period PU5 occurs, both

Table 4-100. Auto-Index Function, Secondary Basic Phase, Event Summary

Ref.	Period	Event	Indirect 2nd Basic Phase Ref. (Table 4-60)
A	BP0	AR→MX→ADDER ‡1	A
B	BP0	REL→TS→ADDER	B
C	BP0	SELECT→(MX+TS)→ADDER	C
D	BP0	SELECT→(MX-TS)→ADDER ‡1	D
E	PU0B	7777_8→MX→ADDER (Bit 5=1)	None
F	PU0B	ADDER→AR	E
G	PU1	CLR→MBR	F
H	PU2	MR→MBR	G
I	PU3	MBR→MDR	H
J	PU3	CLR→IND LATCH	I
K	BP4	MDR→TS	None
L	BP4	SELECT→(MX+TS)→ADDER	None
M	BP4	+1→ADDER	None
N	BP4	ADDER→MDR	None
O	PU5	MDR→MR	J
P	BP5*	CLR→FLLOC LATCH	None
Q	BP6B	MDR→TS→ADDER ‡2	None
R	BP6B	ADDER→PC ‡3	None
S	PU7	SET→EXECUTE PHASE ‡4	K

‡1 These operations are inhibited. See Table 4-60 and Operation B, Table 4-20

‡2 This operation occurs only during a jump instruction, refer to Operation B, Tables 4-95 and 4-96

‡3 This operation occurs only during a jump instruction, refer to Operation E, Tables 4-95 and 4-96

‡4 When a jump is auto-indexed, the basic phase is set again. Refer to Operations F and G, Tables 4-95 and 4-96

Table 4-101. Auto-Index Function, Execute Phase, Event Summary

Ref.	Period	Event	Common Execute Phase Ref. (Table 4-55)
A	EP0	AR→MX→ADDER	A
B	EP0	REL→TS→ADDER	B
C	EP0	SELECT→(MX + TS)	C
D	EP0	SELECT→(MX - TS)	D
E	PU0B	ADDER→AR	E
F	PU1	CLR→MBR	F
G	EP1	PC→MX→ADDER	G
H	EP1	+1→ADDER	H
I	EP1	ADDER→PC	I
J	PU2	MR→MBR	J
K	PU3	MBR→MDR	K
L	PU5	MDR→MR	L
M	EP6	SET→DONE LATCH	M
N	PU7	SET→BASIC PHASE (only)	N

Table 4-102. Auto-Index Function, Primary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-I	BP0- BP3	Same as Operations A through I, Table 4-60.	See Table 4-60.
J.	BP3	INITIALIZE→AUTO-INDEX FUNCTION 1. ↓SID* + ↓DSZ* + ↓ISZ*→↑ISDSS/7A2 2. ↑ISDSS→↓ISDSS*/7A2 3. ↓ISDSS* + ↓XCT* + ↓SBJ* + ↓ADJ* + ↓JPS* + ↓JMP* + ↓LDJ* + ↓STJ*→ ↑MREFI→↓MREFI*/7A3	The auto-index function is initialized.
K.	BP4	SET→IND, INDFF LATCHES 1. (↑MREFI) (↑INDFF*) (↑I04B)→ ↓PREND/6A1 2. ↓PREND→↑PREND*/6A2 3. (↑PREND*) (↑BP4) (↑REGCLK)→ ↓SIND*/6A2 4. ↓SIND* = SET→IND LATCH (↑IND/6A2, ↓IND*) 5. ↓SIND* = SET→INDFF LATCH (↑INDFF/6A3, ↓INDFF*)	The indirect (IND) and indirect flip-flop latches are set to get a second basic phase.
L.	BP5*	CLR→FLLOC→LATCH 1. ↓BP5*/5A3 = CLR→FLLOC LATCH (↓FLLOC/3B4, ↑FLLOC*)	An initial clear of the FLLOC latch occurs.
M.	BP5	SET→FLLOC→LATCH 1. (↑I06*-↑I11*) (↑MREFI) (↑BP5)→ ↓MXX00*/3B3 2. ↓MXX00*→↑MXX00/3B3 3. (↑MXX00) (↑TWFF*) (↑FLMEM*)→ ↓SELOC*/3B4 4. ↓SELOC* = SET FLLOC LATCH (↑FLLOC/3B4, ↓FLLOC*)	At the end of period BP5, the FLLOC latch is set.
N.	BP6B	IF JMP, AND BIT 5 = 1, 7777_8→MX→ADDER 1. (↑FLLOC) (↑I05X) (↑JMP) (↑BP6B)→ ↓SLUMX*/8A4 2. ↓SLUMX*→↑MXEN→↓MXEN*/9A1 = ENABLE→MX 3. ↓SLUMX*→↑MXS0/9B1 = UG→MX→ADDER	If the operate code is a JMP instruction, the content of the utility gate (7777_8) is switched through the MX multiplexer to the adder.
O.	PU6B	Same as Operation E, Table 4-96, but for a jump (JMP) execute, only.	See Table 4-96.
P.	PU7	SET→BASIC PHASE 1. ↓DONE* + ↓TW* + ↓IND* + ↓XCT*→ ↑RBPH→↓RBPH*/5B1 2. (↓RBPH*) (↓CPPU) (↑PU7) = SET BPH (↑BPH/5B2, ↓EPH)	The basic phase is set.

Table 4-103. Auto-Index Function, Secondary Basic Phase, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP0	Operation A of the single-word memory reference instruction indirect fetch is inhibited. Refer to Table 4-60.	See Table 4-60.
B,C	BP0	Same as Operations B and C, Table 4-60.	See Table 4-60.
D.	BP0	Operation D of the single-word memory reference instruction indirect fetch is inhibited. Refer to Table 4-60.	See Table 4-60.
E.	PU0B	If Bit = 1, 7777_8→MX→ADDER 1. (↑FLLOC) (↑I05X) (↑PU0B)→↓SLUMX*/8A4 2. ↓SLUMX*→↑MXEN→↓MXEN*/9A1 = ENABLE→MX 3. ↓SLUMX*→↑MXS0/9B1 = UG→MX→ADDER	The content of the utility gate (7777_8) is switched through the MX multiplexer to the adder.
F-J	PU0B-PU3	Same as Operations E through I, respectively, Table 4-60.	See Table 4-60.
K.	BP4	In the absence of any select signal for the TS multiplexer, the content of the MDR is switched through the TS multiplexer to the add-subtract logic. Refer to Table 4-17	See Table 4-17.
L.	BP4	SELECT→(MX+TS)→ADDER 1. (↑FLLOC) (↑INDFF)→↓INDFL*/12B1 2. ↓INDFL*→↑INDFL/12B2 3. ↓INDFL*→↑TAD4/13B2 4. (↑TAD4) (↑BP4)→↓TSADD* = +TS→ADDER	The content of the TS multiplexer is added to the content of the MX multiplexer.
M.	BP4	+1→ADDER 1. (↑INDFL) (↑BP4)→↓CIN3*→↑CIN/12A2 = +1→ADDER	The content of the adder is incremented by a count of one.
N.	BP4	ADDER→MDR 1. (↑INDFL) (↑BP4)→↓BMEM*→↑RDB→↓RDB*/12B3 = ENABLE→BUS + MBR MX 2. ↓BMEM*→↑PEM→↓PEM*/12B3 = ENABLE→MDR 3. (↑PEM) (↑REGCLK)→↓CPM*/12B3 = LOAD→MDR	The content of the adder is switched through the bus and memory buffer multiplexer to the memory data register.
O.	PU5	Same as Operation J, Table 4-60.	See Table 4-60.
P.	BP5*	CLR→FLLOC LATCH, SET FLMEM LATCH 1. ↓BP5*→CLR FLLOC LATCH (↓FLLOC/3B4, ↑FLLOC*) 2. ↓BP5*→SET→FLMEM LATCH (↑FLMEN/3B4, ↓FLMEN*)	The FLLOC latch is cleared and the FLMEM latch is set.
Q.	PU7	Same as Operation K, Table 4-60.	See Table 4-60

pulses BP5* and BP5 are generated. However, the propagation delay associated with the generation of pulse BP5 in setting the FLLOC latch ensures that an approximate 40 microsecond delay occurs between the conclusion of the BP5* pulse and the latch-setting BP5 pulse, Operation M, Table 4-102.

When a jump (JMP) instruction is auto indexed, Operations N and O, Table 4-99, also occur; however, they do not occur when an auto-index function is processed for any other memory reference instruction. At the end of the primary basic phase, the basic phase is set again. The events that take place during the primary basic phase can be summarized as follows.

a. The content of the program counter is admitted into the address register to get the address of the current instruction.

b. The content of the memory register is admitted into the memory buffer register.

c. The content of the memory buffer register is admitted into the memory data register and into the instruction register.

d. The auto-index function is initialized.

e. The indirect fetch is initialized.

f. The content of the memory data register is written back into memory.

g. The basic phase is set again.

4.4.7.4 SECONDARY BASIC PHASE. The secondary basic phase is similar to the secondary basic phase of an indirectly fetched single-word memory reference instruction with the exception that three operations are inhibited. The first operation that is inhibited is the loading of the instruction register. This is not shown because the inhibiting operation has already been defined by Operation I of Table 4-20. For this operation, the set state of the INDFF latch produces a low-level INDFF* signal which inhibits the first expression of the operation. (Also refer to the description of the Single-Word Memory Reference Instruction, Indirect Addressing, paragraph 4.4.3.5.) The next operation that is inhibited is a subtraction of the relative and current addresses. This operation is not necessary because the auto-index function itself defines the indirect address. The third operation that is inhibited is the admission of the indirect address into the address register. This operation is inhibited because the auto-index function defines the indirect address through the unset instruction register relative address bits and the set state of the FLLOC latch. The events that take place during the secondary basic phase can be summarized as follows.

a. The content of the address register, the current address, is not summed with the relative address. In its place, the indirect address is established as either the first or last memory location.

b. The relative address, bits 106 through 111, is fetched from the instruction register through the TS multiplexer to the adder. This relative address is always 00_8.

c. The indirect address, either the first or last memory location of the current field, is admitted into the address register.

d. The content of the memory register, the pointer address, is admitted into the memory buffer register.

e. The content of the memory buffer register, the pointer address, is switched through the bus and memory buffer multiplexer to the memory data register.

f. The indirect, IND, latch is cleared.

g. The pointer address is fetched from the memory data register, through the TS multiplexer, into the adder.

h. The content of the adder, the pointer address, is incremented by a count of one to obtain the actual pointer address.

i. The pointer address is admitted into the address register to obtain the location of the operand to be processed during the execute phase.

j. The content of the memory data register, the pointer address, is written back into the first or last memory location of the current memory field.

k. The FLLOC latch (auto-index function) is cleared.

l. The execute phase is set.

4.4.7.5 EXECUTE PHASE. The execute phase of the auto-index function is similar to the execute instruction of the single-word memory instruction with the exception that no relative address is specified because instruction register bits 106 through 111 are all zeros. During the second basic phase, the address of the instruction to be executed was loaded into the address register (8) after first incrementing. During the execute phase, this address is switched through the MX multiplexer (6) to the adder (7). Next, the content of the relative address bits in the instruction register (3) is switched through the TS multiplexer (5) and summed with the content of the adder. However, because these bits are all zero, the resultant output from the adder, which is admitted into the address register (8), is the address of the instruction to be executed - no modification of the address has taken place. Next, subsequent execute-phase events are identical to those for the common execute phase as listed in Tables 4-54 and 4-55. However, the operate code bits of the auto-index function determine which of the memory reference instruction will be executed. The events that take place during the execute phase can be summarized as follows.

a. The content of the address register, the pointer address, is admitted into the adder and summed with relative address bits 106 through 111. These relative address bits have the value 00_8 so that the result of the summation leaves the pointer address in the adder.

b. The content of the adder, the pointer address, is admitted into the address register.

c. The content of the program counter, the current address, is admitted into the adder and incremented by a count of 1 to get the location of the next instruction.

d. The content of the memory register specified by the pointer address is loaded into the memory buffer register.

e. The content of the memory buffer register is loaded into the memory data register.

f. The operand is processed according to the operate code specified by instruction register bits 100 through 103.

g. The content of the memory data register is written back into memory at the location specified by the pointer address.

4.4.7.6 JMP AUTO-INDEX FUNCTION. When the auto-index function involves a jump (JMP) instruction, the same phase relationships are maintained; hence, when the JMP instruction is auto-indexed, only two basic phases are required, the execute phase does not take place. Table 4-104 lists the event summaries for a normal jump instruction basic phase and a jump auto-index function basic phase.

4.4.7.6.1 Primary Basic Phase. Referring to Table 4-104, it can be seen that until period BP3 occurs, the basic phase events are identical. However, for the auto-index jump, the auto-index function is invoked by setting the IND, the INDFF, and the FLLOC latches. For the normal basic phase operation, the relative address and current address are summed to get the effective address during period BP6. This operation does not occur when the jump instruction is auto-indexed; the operation is deferred until the next basic phase. Additionally, the indirect address (first or last location in the current memory field) is loaded into the program counter. At the conclusion of the primary basic phase, a second basic phase is set to obtain the indirect fetch. The purpose of the primary basic phase for the auto-index function is to get the indirect address into the program counter. The events that take place during the primary basic phase of the auto-indexed jump instruction can be summarized as follows.

a. The content of the program counter is admitted into the address register to get the address of the current auto-indexed jump instruction.

Table 4-104. JMP Instruction, Primary Basic Phase, Normal Addressing Vs. JMP Auto-Index Function

Normal		Auto-Index	
Period	Event	Period	Event
PU7	CLR-LATCHES	PU7	CLR→LATCHES
BP0	PC→MX→ADDER	BP0	PC→MX→ADDER
BP0	ADDER→AR	BP0	ADDER→AR
PU0B	CLR→IR	PU0B	CLR→IR
BP1	CLR→DONE	BP1	CLR→DONE
PU1	CLR→MBR	PU1	CLR→MBR
PU2	MR→MBR	PU2	MR→MBR
PU3	MBR→MDR	PU3	MBR→MDR
BP3	MBR→IR	BP3	MBR→IR
	———————	BP3	INITIALIZE→AUTO INDEX
	———————	BP4	SET→IND, INDFF LATCHES
	———————	BP5	CLR→FLLOC LATCH
	———————	BP5	SET→FLLOC LATCH
PU5	MDR→MR	PU5	MDR→MR
BP6B	REL ADR→TS	BP6B	REL ADR→TS ‡1
BP6B	SELECT→(MX+TS)→ADDER	BP6B	SELECT→(MX + TS)→ADDER ‡1
BP6B	SELECT→(MX-TS)→ADDER	BP6B	SELECT→(MX - TS)→ADDER ‡1
	———————————	BP6B	0000_8→MX→ADDER (bit 5 = 0)
	———————————	BP6B	7777_8→MX→ADDER (bit 5 = 1)
BP6B	ADDER→PC	PU6B	ADDER→PC ‡2
BP6	SET→DONE LATCH	BP6	SET→DONE LATCH ‡3
PU7	SET→BASIC PHASE	PU7	SET→BASIC PHASE

‡1. These operations inhibited by FLLOC latch signals.

‡2. This operation occurs only during a JMP auto-index function, refer to Operation E, Table 4-96.

‡3. This operation inhibited by ↓IND*, refer to Operation F. Table 4-96.

b. The content of the memory register located at the current address is read into the memory buffer register.

c. The content of the memory buffer register is read into the memory data register and into the instruction register.

d. The auto-index and indirect fetch functions are initialized.

e. The content of the memory data register is written back into the location specified by the current address.

f. The program counter gets the indirect address from the adder, which receives the first or last location value through the MX multiplexer.

4.4.7.6.2 Secondary Basic Phase. The events that take place during the secondary basic phase are identical with those that take place during the secondary basic phase for any other auto-indexed memory reference instruction with the exception of Operations Q and R, Table 4-100. These operations occur for any processed jump instruction; they do not normally occur when any other memory reference instruction is auto-indexed. The events that take place during the secondary basic phase can be summarized as follows.

a. The content of the address register, the current address, is not summed with the relative address. In its place, the indirect address is established as either the first or last memory location.

b. The relative address, bits I06 through I11, is fetched from the instruction register through the TS multiplexer to the adder. This relative address is always 00_8.

c. The indirect address, either the first or last memory location of the current memory field, is admitted into the address register.

d. The content of the memory register, the pointer address, is admitted into the memory buffer register.

e. The content of the memory buffer register, the pointer address, is switched through the bus and memory buffer multiplexer to the memory data register.

f. The indirect, IND, latch is cleared.

g. The pointer address is fetched from the memory data register, through the TS multiplexer, into the adder.

h. The content of the adder, the pointer address, is incremented by a count of one to obtain the effective address.

i. The effective address is admitted into the memory data register for subsequent writing into memory.

j. The content of the memory data register, the effective address, is written back into the first or last memory location of the current memory field.

k. The FLLOC latch (auto-index function) is cleared.

l. The content of the memory data register, the effective address, is written into the program counter.

m. The execute phase is set.

4.4.8 PROGRAMMED HALT

There are two ways to obtain a halt for the ND812. One of these is operation of the STOP switch on the front panel; the other is a programmed halt. The programmed halt occurs when the halt instruction occurs in a program when the ND812 is in its run state. There are several halt producing machine-language codes that may be used to obtain the halt condition. One of these is $07XX_8$, a form of the two-word execute instruction. Because there is no possibility for a two-word execute, this machine code is illegal. Other illegal codes are, $02XX_8$, the two-word ANDF form, $00XX_8$, the two-word form, and $01XX_8$, the two-word operate form. If any form of these illegal codes is programmed, the ND812 go-control logic will be set to its initialized state as described in paragraph 4.2.2.2.

The Fundamental Operations which produce the halt are listed below.

1. (↑I03B)(↑I04B)(↑I05B)(↑I06*)→↓70*/3B1
2. (↑I03*)(↑I04B)(↑I05*)(↑I06*)→↓20/3B1
3. (↑I03*)(↑I04*)→↓01 + 00*/3B1
4. ↓70* + ↓20 + (↓01 + 00*)→↑HLT/3B2
5. ↑HLT→↓HLT*/3B2
6. (↓I00)(↓I01)(↓I02)(↓I03)→↓00*/7A1
7. (↓I00)(↓I01)(↓I02)(↑I03)→↓01*/7A1
8. ↓00* + ↓01*→↑TW/7A1
9. (↑TW)(↑HLT)→↓TWHLT*/3B2
10. ↓TWHLT*→↑TWHLT/3A2
11. (↑TWHLT)(↑BP6B)→↓RGO*/3A2 = INITIALIZE→GO
12. (↓GO/3A2, ↑GO*)

4.4.9 LITERAL INSTRUCTIONS

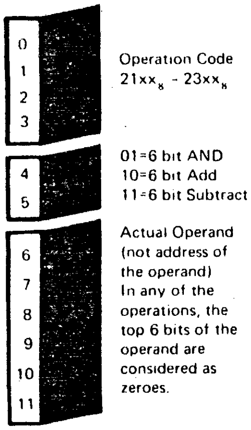
Literal instructions can be considered a special case of memory reference instructions because no indirect addressing is possible. The indirect address bit is used to produce the literal and masking functions if the operation code is 2_8 and bit 3 of the instruction word is not set. A literal is a direct representation of data which appears in the instruction calling the data. Literals conserve core because a symbolic tag and full-word storage is not required.

Actually the instructions in the literal group include two ANDing (ANDF, ANDL) and two literals. The ANDL instruction is processed as a literal because the instruction calling the logical operation also carries the data. The other AND instruction (ANDF), however, is processed as a memory reference instruction. There are only three literal instructions, which are listed in Table 4-105. The literal instruction format and bit patterns are listed in Table 4-106. All literal instructions have only one basic phase; however, the ANDF instruction is processed as a normal single-word memory reference instruction, and has both basic and execute phase.

Table 4-105. Literal Instructions

Subgroup	Mnemonic	Octal Code	Operation
LIT	ANDL	21XX	Logical AND last 6 bits of instruction (XX) with J_6 through J_{11}
	ADDL	22XX	Add last 6 bits of instruction (XX) to J
	SUBL	23XX	Subtract last 6 bits of instruction (XX) from J

Table 4-106. Literal Instruction Format and Bit Patterns


0
1
2
3
Operation Code
$21xx_8$ - $23xx_8$
4
5
01=6 bit AND
10=6 bit Add
11=6 bit Subtract
6
7
8
9
10
11
Actual Operand (not address of the operand) In any of the operations, the top 6 bits of the operand are considered as zeroes.

This is the literal instruction format. Bits 4 and 5 determine the logical or arithmetic operation. Bits 6 through 11 are not address data, but actual operands. It is this data which is operated upon in conjunction with any data word contained in the J register.

		Operation Code					
Octal Code	**Mnemonic**	**0**	**1**	**2**	**3**	**4**	**5**
2000	ANDF‡		1				
2100	ANDL		1				1
2200	ADDL		1			1	
2300	SUBL		1			1	1

‡ Memory reference instruction

The Event Summary for the ANDF memory reference instruction gives the entire listing for both the basic and execute phases; the execute phase events are indicated by boldface type. When the ANDF memory reference instruction is processed, the purpose of the basic phase is to initialize the literal function, which is used to obtain the ANDF operation when the ANDF memory reference instruction is specified. Actually, the initializing signals are the only functional commonality between the ANDF memory reference instructions and the literal instructions. Bits 4 and 5 of the instruction word are used to specify the logical operation, they cannot specify indirect addressing or backward relative addressing for the ANDF instruction. Although operations L, M and Q, Table 4-103 are possible during the basic phase, they have no relevance during this phase, because the result of the operation is not collected into the J register; actually, this event occurs during period EP6, when event AB takes place. Hence, although the AND function is invoked when the literal instruction is initialized, its results are not meaningful until the content of the utility gates is admitted through the MX multiplexer to the adder, and from the adder to the J register as indicated by Operations L, M, Q and AB, of Tables 4-107 and 4-108.

When the literal instructions are processed, no execute phase can occur because the done latch is set by the low-level ANL*, ADL* or SBL* signals. Hence, the ANDing events which had no relevance during the ANDF memory reference instruction, do perform required functions during the basic phase. The basic phase events described for the ANDF memory reference instruction in Table 4-107, also occur for the literals. Hence, the description of the literal instruction event summaries in Table 4-107 list only those operations which do not occur during the common basic phase, Table 4-20.

Table 4-107. Literal Instructions, Event Summary

Mnemonic Code	Octal Code	Ref. ‡1	Period ‡1	Event ‡1
ANDF‡2	2000	A	PU7	CLR→LATCHES
		B	BP0	PC→MX→ADDER
		C	PU0B	ADDER→AR
		D	PU0B	CLR→IR
		E	BP1	CLR→DONE
		F	PU1	CLR→MBR
		G	PU2	MR→MBR
		H	PU3	MBR→MDR
		I	BP3	MBR→IR
		J	BP3	INITIALIZE→LITERAL
		L	*‡3*	*AND→J_{00}-J_{11}, M_{00}-M_{11} ‡4*
		M	*‡3*	*AND→UG‡4*
		P	PU5	MDR→MR
		Q	*PU6*	*UG→MX→ADDER‡4*
		Z	PU7	SET→BASIC PHASE
		A	EP0	AR→MX→ADDER
		B	EP0	REL→TS→ADDER
		C	EP0	(MX+TS)→ADDER
		D	EP0	(MX-TS)→ADDER
		E	PU0B	ADDER→AR
		F	PU1	CLR→MBR
		G	EP1	PC→MX→ADDER
		H	EP1	+1→ADDER
		I	EP1	ADDER→PC
		J	PU2	MR→MBR
		K	PU3	MBR→MDR
		L	‡3	AND→J_{00}-J_{11}, M_{00}-M_{11} ‡4
		M	‡3	AND→UG ‡4
		N	PU5	MDR→MR
		Q	PU6	UG→MX→ADDER
		AB	EP6	ADDER→JR
		AC	EP6	SET→DONE LATCH
		AD	PU7	SET→BASIC PHASE (only)
ANDL	21XX	J	BP3	INITIALIZE→LITERAL ANDL
		K	BP3	AND $JR_{06\text{-}11}$, $MDR_{06\text{-}11}$
		M	BP3	AND→UG
		O	BP4	PC→MX→ADDER +1→PC
		Q	PU6	UG→MX→ADDER
		R	PU6	ADDER→JR
ADDL	22XX	J	BP3	INITIALIZE→LITERAL ADDL
		O	BP4	PC→MX→ADDER +1→PC
		S	BP6B	IR DATA→TS
		T	BP6B	+TS→ADDER
		W	BP6	JR→MX→ADDER
		X	BP6	IF JR+IR = CARRY, COMPL→OV

Table 4-107. Literal Instructions, Event Summary (Cont'd.)

Mnemonic Code	Octal Code	Ref. ‡1	Period ‡1	Event ‡1
SUBL	23XX	J	BP3	INITIALIZE→LITERAL SUBL
		O	BP4	PC→MX→ADDER +1→PC
		S	BP6B	IR DATA→TS
		U	BP6	-TS→ADDER
		V	BP6	+1→ADDER
		W	BP6	JR→MX→ADDER
		X	BP6	IF JR-IR = CARRY, COMPL→OV
		AA	BP7	COMPL→OV

‡1 Events listed in boldface type occur during the execute phase.

‡2 The ANDF instruction is a single-word memory reference. No indirect addressing is possible. Also any forward relative address may be specified.

‡3 These events are static; i.e., once established, they remain available until period BP1 of the ensuing basic phase.

‡4 Although they occur during the basic phase, these signals are irrelevant until the execute phase.

Table 4-108. Literal Instructions, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A-I	BP0- BP3	Same as Operations A through I, Table 4-20.	See Table 4-20.
J. (2000) (21XX) (22XX) (23XX)	BP3	INITIALIZE→LITERAL 1. (↓I00) (↑I01) (↓I02) (↓I03)→↓LIT*→ ↑LIT/7A2 2. (↑LIT) (↑I04*) (↑I05B)→↓ANL*/6A1 3. ↓LIT*→↑LITB/7A1 4. (↑I04*) (↑I05*) (↑LIT)→↓A12*/6A1 5. (↑I04B) (↑I05*) (↑LIT)→↓ADL*→ ↑ADLSBL/6B2 6. (↑I04B) (↑I05B) (↑LIT)→↓SBL*→ ↑ADLSBL/6A2	The literal operation is initialized.
K. (21XX)	BP3	AND $JR_{06\text{-}11}$, $MDR_{06\text{-}11}$ only 1. (↑LIT) (↑I05B) (↑I04*)→↓ANL*/6A1 2. (↑J00 - ↑J05) (↑M00B- ↑M05B) (↓ANL*/6A1)→↑JM00* - ↑JM05*/14AB1, 15B1 3. (↑J06 - ↑J11) (↑M06B - ↑M11B)→ ↓JM06 - ↓JM11/15A1, 16AB1→AND (J_{06} - J_{11}) (M_{06} - M_{11})	Only bits 6 through 11 of JR and MDR are anded.

Table 4-108. Literal Instructions, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
L. (2000)	BP3	AND JR_{00-11}, M_{00} - M_{11} 1. (↑J00 - ↑J05) (↑M00B - ↑M00B) (↑ANL*/6A1)→↓JM00* - ↓JM05*/14AB1, 15B1 2. (↑J06 - ↑J11) (↑M06B - ↑M11B)→ JM06* - JM11*/15A1, 16AB1→AND (J_{00} - J_{11}) (M_{00} - M_{11})	If, and only if corresponding J and MDR register bits are ones, does a 1 bit appear at the input of the utility gate.
M. (2000) (21XX)	BP3	AND→UG 1. ↑LIT + ↑JM00*- JM07*+ ↑PULLU→ ↑U00 - U07/14AB2, 15AB2 2. ↑LITB + ↓JM08*- JM11*+↑PULLU→ ↑U08 - U11/16AB2	The ANDed result appears at the MX multiplexer.
N. (2000)	BP4	INHIBIT PC→MX→ADDER +1→PC 1. ↓A12* + ↓MREFI*→↑INCPR→↓INCPR*/9B3 2. ↓INCPR* + ↑WTRL* + ↑TWIO* + ↑BP4→ ↑PEP*/9B3 = DISABLE→PC 3. ↓INCPR* + ↑WTRL* + ↑TWIO* + ↑BP4→↑SLPMX*/8B4 = INHIBIT PC→ MX→ADDER 4. ↑HLTTW* + ↑WTRL* + ↓INCPR* + ↑BP4→↑CIN2*→ ↓CIN/12A2 = INHIBIT +1→ADDER	The incrementing of the content of the program counter is inhibited during the common base phase.
O. (21XX) (22XX) (23XX)	BP4	PC→MX→ADDER +1→PC 1. (↑A12*) (↑MREFI*)→↓INCPR→↑INCPR*/9B3 2. (↑INCPR*) (↑WTRL*) (↑TWIO*) (↑BP4)→↓PEP*/9B3 = ENABLE→PC 3. (↑INCPR*) (↑WTRL*) (↑TWIO*) (↑BP4)→ ↓SLPMX*/8B4 4. ↓SLPMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 5. ↓SLPMX*→↑MXS1/9A2 = PC→MX→ ADDER 6. (↑HLTTW*) (↑WTRL*) (↑INCPR*) (↑BP4)→↓CIN2*→ ↑CIN/12A2 = +1→ADDER	During the common basic phase, the program counter is incremented.
P.	PU5	Same as Operation J, Table 4-20.	See Table 4-20.
Q. (2000) (21XX)	PU6	UG→MX→ADDER 1. (↑LITB) (↑I04*) (↑PU6B)→↓SLUMX*/8A4 2. ↓SLUMX*→↑MXEN→↓MXEN*/9A2 = ENABLE→MX 3. ↓SLUMX*/8A4→MXS0 = LOAD→UG	The content of the utility gates is admitted to the MX multiplexer.

Table 4-108. Literal Instructions, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
R. (21XX) (22XX) (23XX)	BP6	ADDER→JR 1. ↓I04* + ↓I05B*→↑I4I5/9A1 2. (↑I4I5) (↑LITB)→↓LITJ*/9A2 3. ↓LITJ*→↑PEJ6/9A3 4. (↑PEJ6) (↑BP6)→↓PEJ* ↑PEJ/9B3 = ENABLE→JR 5. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = LOAD→JR	The content of the adder is admitted into the J register at BP6 for a 6-bit AND.
S. (22XX)	PB6B	IR DATA→TS 1. ↑ADLSBL→ ↓AD*BL*→↑JMPDLS/7B4 2. (↑TWFF*) (↑INDFF*)→↓MPASS*/13B4 3. ↓MPASS*→↑MPASS/13B4 4. (↑MPASS) (↑JMPDLS) (↑BP6B)→↓SLITS*/13B4 5. ↓SLITS*→↑TSS0/13B4 = IR→TS	The content of instruction register bits 06 through 11 are admitted into the TS multiplexer.
T. (22XX)	BP6B	SELECT + TS→ADDER 1. (↑MPASS) (↑I5B*)→↓ADDAR*→ ↑ADDEP0/13B1 2. (↑ADDEP0) (↑JMPDLS)→↓TAD6*→ ↑TAD6/13B2 3. (↑TAD6) (↑BP6B)→↓TSADD*/13B2 = +IR→ADDER	The content of +TS is admitted into the adder.
U. (23XX)	BP6	SELECT - TS→ADDER 1. (↑MPASS) (↑JMPDLS) (↑I05X)→ ↓SUB6*→↑SUB6/13B3 2. (↑SUB6) (↑BP6B)→↓TSSUB*/13B3 = -IR→ADDER 3. ↓TSSUB*→↑TSSUB/13B4	The content of -TS is admitted into the adder.
V. (23XX)	BP6	IF TSSUB/13B4, +1→ADDER 1. ↓HWD + ↑ONCE*→↑DCIN*/12A1 2. (↑DCIN*) (↑TSSUB) (↑OP2*)→↓CIN2 *→ ↑CIN/12A2 = +1→ADDER	The adder is incremented if the complement of the data word is selected from the adder to get a 2s complement.
W. (22XX) (23XX)	BP6	JR→MX→ADDER 1. (↑ADLSBL) (↑BP6)→↓SLJMX*/12A3 2. ↓SLJMX*→↑MXEN→↓MXEN*/9B2 = ENABLE→MX 3. ↓SLJMX*→↑MXS2/9A2 = JR→MX→ ADDER	The content of the J register is admitted into the adder.

Table 4-108. Literal Instructions, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
X. (22XX) (23XX)	BP6	IF IR + JR = CARRY OUT, COMPL→OV 1. (↑COUT) (↑ADLSBL) (↑BP6)→↓ADSBLC/11B3 2. ↓ADSBLC→↑CPOV/11B3 3. (↑CPOV) (↑REGCLK)→↓CKOV/11B4 = TOGGLE→OV 4. (↑SOV*) (↑ROV*) = TOGGLE→ENABLE	If the summation of IR$_{06}$ - IR$_{11}$ and J$_{00}$ - J$_{11}$ produces a carry out of the adder, the overflow register is complemented.
Y,Z (21XX) (22XX) (23XX)	BP6	Same as Operations K and L, Table 4-20.	See Table 4-20.
AA. (23XX)	BP7	COMPL→OV 1. ↓SBL*→↑OP1SBL/12B4 2. (↑OP1SBL) (↑BP7)→↓AROV*/12B4 3. ↓AROV*→↑CPOV/11B3 4. (↑CPOV) (↑REGCLK)→↓CKOV/11B4 = TOGGLE→OV 5. (↑SOV*) (↑ROV*) = TOGGLE→ENABLE	The overflow register is complemented unconditionally if a literal subtraction occurs.
AB. (2000)	EP6	ADDER→JR 1. (↑I04*) (↑I05B*)→↓I4I5/9A1 2. ↓I4I5→↑I4I5*/9A1 3. (↑I4I5*) (↑LITB)→↓LDLOG*/9A2 4. (↑LDJLOG) (↑EP6)→↓PEJ*→↑PEJ/9A3 = ENABLE→JR 5. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4 = LOAD→JR	

4.4.9.1 BLOCK DIAGRAM DESCRIPTION. For the literal group of memory reference instructions, Figure 4-74 is the block diagram, Figure 4-75 is the timing diagram Table 4-107 is the Event Summary, which lists all fundamental events for the literal operations, and Table 4-108 is the instruction listing in terms of the fundamental operations.

ANDs - The two logical AND instructions are characterized by logically combining two corresponding bits of the two data words to be ANDed. In one case, the data to be ANDed is derived from the twelve-bit memory data register (1, Figure 4-74), through the bus and memory buffer multiplexer (not shown in the figure). In the other case the data is derived from the six least significant bits of the instruction word in the instruction register (2). The source for the second data word is the content of the J register (9). Figure 4-74 illustrates how the corresponding bits of two words are ANDed through the arithmetic data loop. Hence, corresponding bits of the two data words stored in the memory data register (1) and the J register (9) in one case, and in the instruction register (2) and the J register (9) in the second case, are ANDed (3) by the AND gates; the result appears at the input to the

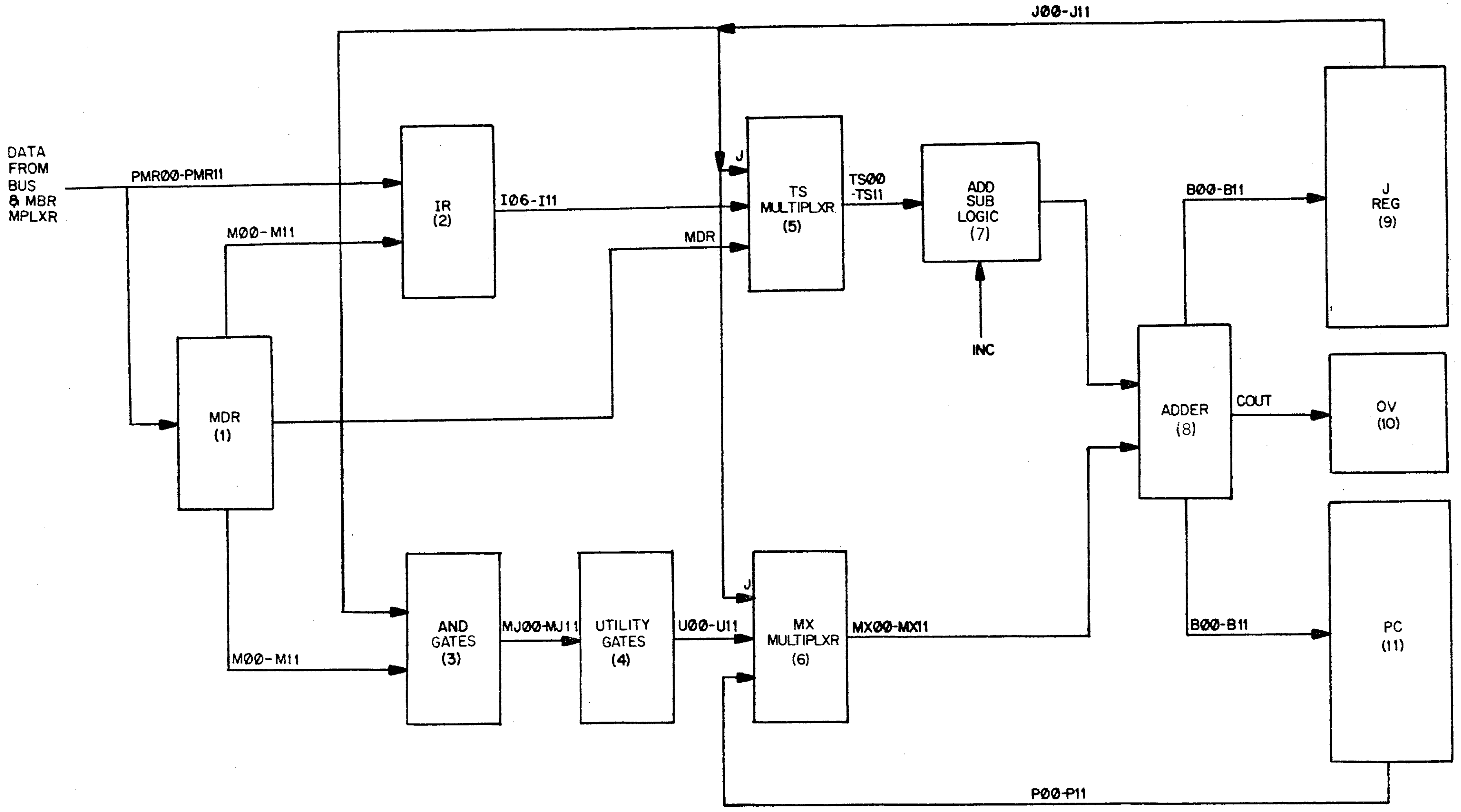


Figure 4-74. Literal Instructions, Block Diagram

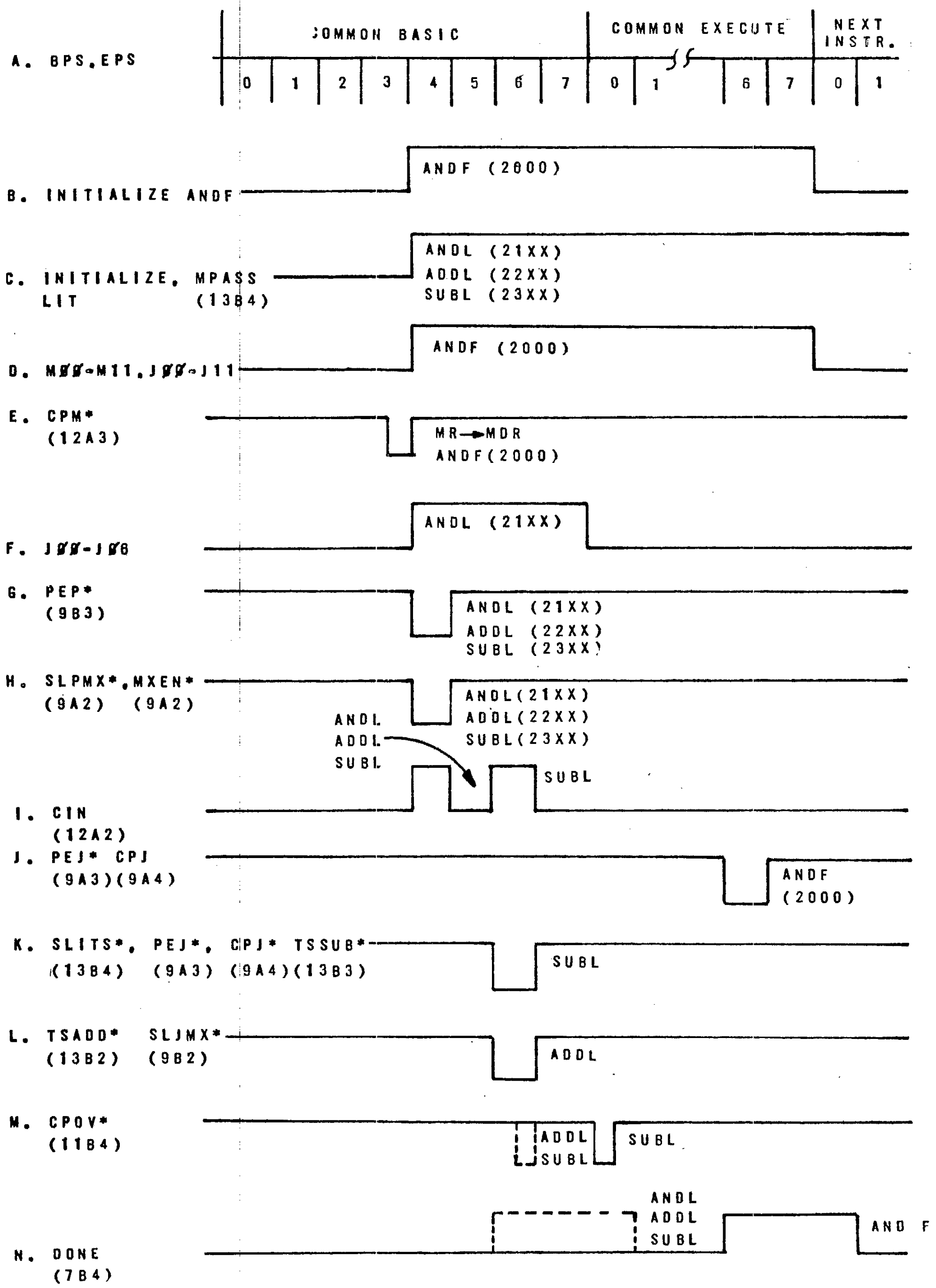


Figure 4-75. Literal Instructions, Timing Diagram

enabled utility gates. The utility gate outputs are switched through the MX multiplexer (6) and adder (8) so that contents of the adder replaces the content of the J register (instructions 2000 and 21XX).

For both AND operations, the program counter (11) must be incremented by the add-subtract logic (7) to provide the address for the next instruction which is processed when the current AND instruction execute phase is entered so that the operation requires two phases to complete the machine cycle. If, on the other hand, the AND operation takes a six-bit word from the lower 6 bits of the memory data register (2), the program counter content is incremented during period BP4 of the common basic phase. This AND operation permits the instruction to be processed during the common basic phase to complete the machine cycle.

ADDL (22XX) - For the addition literal instruction, the six least significant bits of the content of the instruction register (2, Figure 4-74) are switched through the TX multiplexer (5) and the add-subtract logic (7) to the input of the adder (8). The other input to the adder is derived from the content of the J register (9) and is switched through the MX multiplexer (6). If the adder produces a carry as the result of the summation of the instruction register and J register data, the overflow register (10) is complemented. A complemented overflow register indicates that the resultant sum is larger than the capacity of the J register. Responsibility for the state of the overflow register prior to and after the instruction is executed rests with the programmer. The resultant sum is stored in the J register (9).

For this operation, the program counter incrementing function takes place at period BP4. During the period, the content of the program counter (11) is admitted into the adder (8) through the MX multiplexer (6). Then the add-subtract logic (7) is incremented and the content of the adder (8) is readmitted to the program counter. At the conclusion of the machine cycle, the done latch is set so that the basic phase can be reestablished by the major state control logic when the next instruction is executed.

SUBL (23XX) - For the subtraction literal instruction, the six least significant bits of the content of the instruction register (2) are also switched through the TS multiplexer (5) and the add-subtract logic (7) to the input of the adder. However, it is the complement of the data word that is taken from the add-subtract logic (7) so that the adder contains the 1's complement of the instruction register data word. Next, the data-word 1's complement is incremented to obtain the 2's complement and the result is summed with the input from the MX multiplexer. This operation yields a 2's complement subtraction of the content of the instruction register from the content of the J register. The resultant sum is stored in the J-register. If the adder produces a carry as the result of the subtraction, the overflow register is complemented, and then complemented again at the conclusion of the instruction. As with addition, the responsibility for the state of the overflow register rests with the programmer. The program counter incrementing operation is identical to that for the ADDL operation.

4.5 MEMORY UNIT

This section discusses the organization and operation of the core memory options available with the ND812 computer. The following paragraphs will discuss in order; the memory options and their specifications (4.5.1), the theory of core memory operation (4.5.2), the hardware configurations (4.5.3), and the detailed principles of operation (4.5.4).

4.5.1 MEMORY OPTIONS

There are four core memory capacity options available with the ND812. The minimum capacity is 4K (4,096) twelve-bit words and this capacity can optionally be increased in 4K increments to a maximum of 16K (16,384) words. The cabinet of the processor itself can contain 4K or 8K of core. Addition of a memory extension cabinet provides for expansion to 12K or 16K word capacity.

4.5.2 BASIC THEORY OF CORE MEMORY OPERATION

Core memory is a magnetic information storage device. Each bit of information in the core memory is represented by the direction of magnetization of a single ferrite ring termed a "core". A typical core memory unit contains many thousands of these ferrite cores arranged in a three-dimensional array referred to as the memory stack. Electronic devices external to the memory stack are used to selectively change the states of magnetization of the cores, as well as sense their states at a given time. In this way, bit patterns are stored into and read out of the core array of the memory stack.

The ferrite cores have the property of being able to assume two states of magnetization and remain in either state indefinitely. When an electric current of sufficient intensity is passed through the hole in the center of the core, the core is magnetized as shown in Figure 4-76a. An equal current in the opposite direction results in the reverse magnetization as shown in Figure 4-76b. These two states are arbitrarily assigned logical values Ø and 1. The minimum current intensity sufficient to reverse the state of a core is known as the threshold current. Currents of intensity less than the threshold have no lasting effect on the magnetization of the core. Information is stored into the cores by momentary passage of superthreshold currents through them in the proper direction.

When a core changes its state of magnetization, a current is induced in any conductive loop through the core. The value represented by a core can be sensed or read by passing a superthreshold current through the core and detecting whether a current is produced in a sense wire that also passes through the core. This process is illustrated in Figure 4-77.

It is possible to build a core memory with only two wires through each core, but such an arrangement would require a separate drive and sensor circuit for each bit. Instead, a four wire circuit is used. The memory stack is organized as a stack of core mats, which are square planar arrays of core wired together. When a read or store function is performed, a single bit from each mat is read. Thus, a word consisting of as many bits as there are mats in the stack is the smallest unit of information that can be stored or read in a single operation.

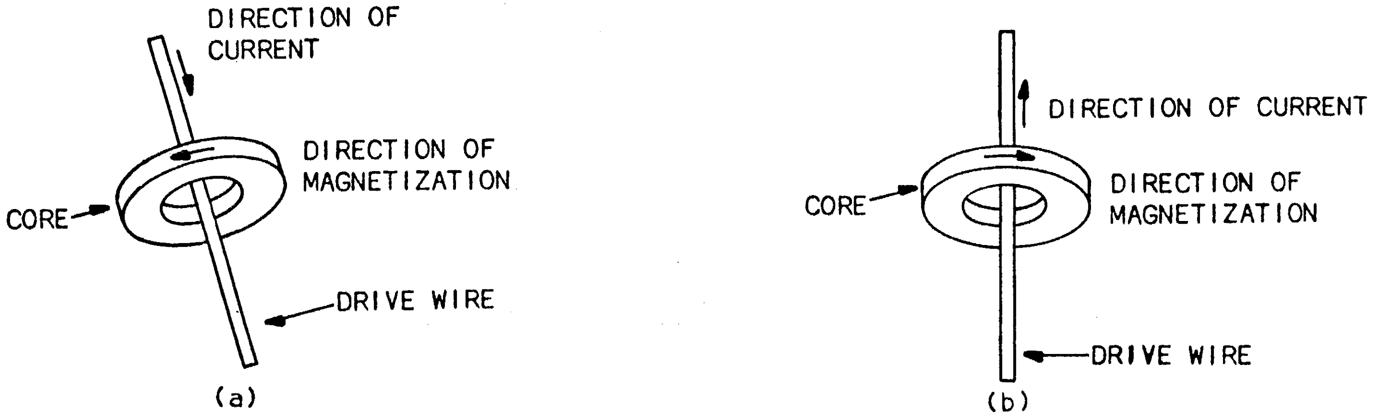


Figure 4-76. Magnetization of Cores

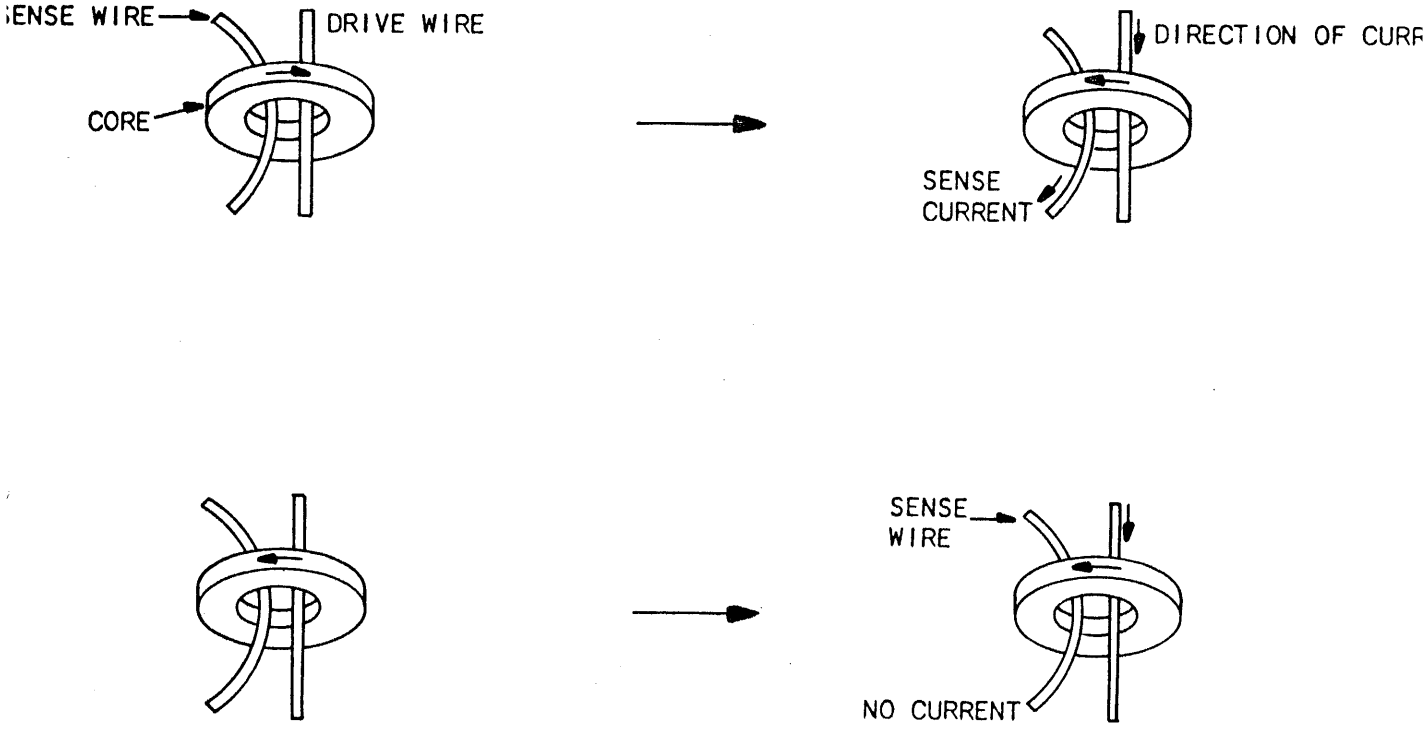


Figure 4-77. Core Sensing Process

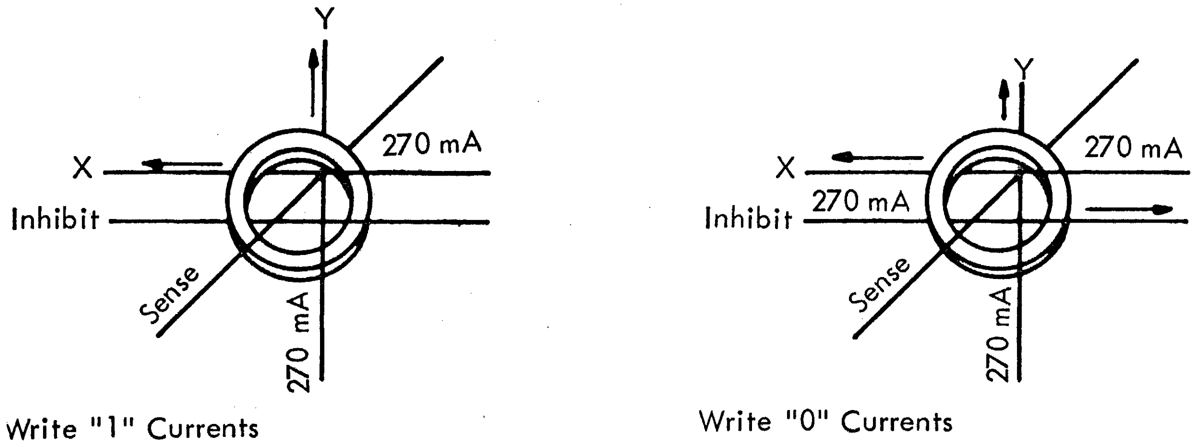
The single bit affected by a memory operation in each mat is selected by driving two wires (X- and Y-select wires) each with a subthreshold current greater than half the threshold current, and in the same direction through the core. Though both the X- and Y-select wires pass through more than one core, a given X-select wire and a given Y-select wire pass coincidentally through only a single core in each mat, (the selected core) and the effects of the X and Y currents add to produce a superthreshold magnetization current through that core. Thus, only the selected core of each mat receives stimulation sufficient to change its state. This type of operation is defined as coincident current.

Each mat contains a single sense wire and a single inhibit wire which are used respectively in read and store (write) operations. The sense wire carries signals indicating changes of magnetization of any core in the mat. The inhibit wire carries current equal to an X- or Y-select current in the opposite direction through the cores to prevent changes in magnetization of any of the cores of the mat. Thus, four wires pass through each ferrite core.

4.5.2.1 READ OPERATION. The read operation is performed by changing all the bits of the selected word to Ø and monitoring the sense wires of all the mats of the memory stack. If a selected core represents a logical Ø, no signal results on the sense wire. If the core represents a logical 1, the change in its magnetization results in a current pulse on the sense wire. The bit pattern of the selected word is then obtainable by sampling the voltage in the sense wires at the proper time. Since Ø's are written into the selected word, that word is erased from the memory stack by this process. Because of this the process is referred to as a destructive readout.

4.5.2.2 WRITE OPERATION. The write operation stores the bit pattern of a word into the memory stack by changing selected cores from Ø to 1. Since this operation does not change any 1's to Ø's, the cores to be written into must all be set to Ø prior to the write operation. This is done by performing a read operation on the selected word. The write operation itself consists of producing currents in the selected X and Y select wires in the opposite direction to the currents in the read operation. This would set all the bits to 1's were it not for the inhibit wires, which carry currents in the direction opposite to the select currents through those cores which are desired to be left as Ø's. The inhibit current cancels enough of the select current effect so that the core receives a subthreshold stimulus and does not change state. This process is illustrated in Figure 4-78.

4.5.2.3 MEMORY CYCLE. As mentioned in paragraph 4.5.2.1, the read operation results in the destruction of the representation of the selected word in the memory stack. It is not usually desireable to destroy a stored copy each time it is read, so each word read from the memory stack must be recopied into the stack. As mentioned in paragraph 4.5.2.2, the write operation must be preceded by a read operation. The memory therefore operates in strict cycles of a read operation followed by a write operation whether the desired function is a read or a write. The cycle time of a core memory is the time required to perform a read operation and a write operation in succession.



Write "1" Currents

Write "0" Currents

Figure 4-78. Write Operation

4.5.3 GENERAL DESCRIPTION

There are four core memory capacities available with the ND812 Computer; 4K (4,096), 8K (8,192), 12K (12,288), and 16K (16,384) 12-bit words. All of the options employ the four wire system described in paragraph 4.5.2 to implement the read and write operations and all have a cycle time of 2 microseconds.

Since each word contains twelve bits, a single word can represent 4,096 (2^{12} = 4,096) different binary numbers. Each word in the memory has a unique binary number as its address, with a 4K core memory, a single word can represent any address. With a larger memory, a single word is not capable of representing all possible addresses. For this reason, the memory is divided into blocks of 4K word capacity called fields. When addressing a word in a memory with more than one field, the particular field in which the desired word is located must be specified.

Specification of fields is done in two ways in the ND812. The last two bits of the first word of a two-word instruction are sometimes used to specify the desired memory field. If an instruction fails to specify a memory field, responsibility for determining which field to access is placed on a hardware device called the Memory Field Control logic.

Since 4K memories constitute only a single field, only those ND812 computers equipped with more than 4K of memory have the memory field control logic installed.

4.5.3.1 4K MEMORY. A block diagram of the 4K memory is shown in Figure 4-79. The memory control (1) handles timing of various functions of memory operation such as enabling data transfers and applying drive currents to the memory stack (12). The read/write driver

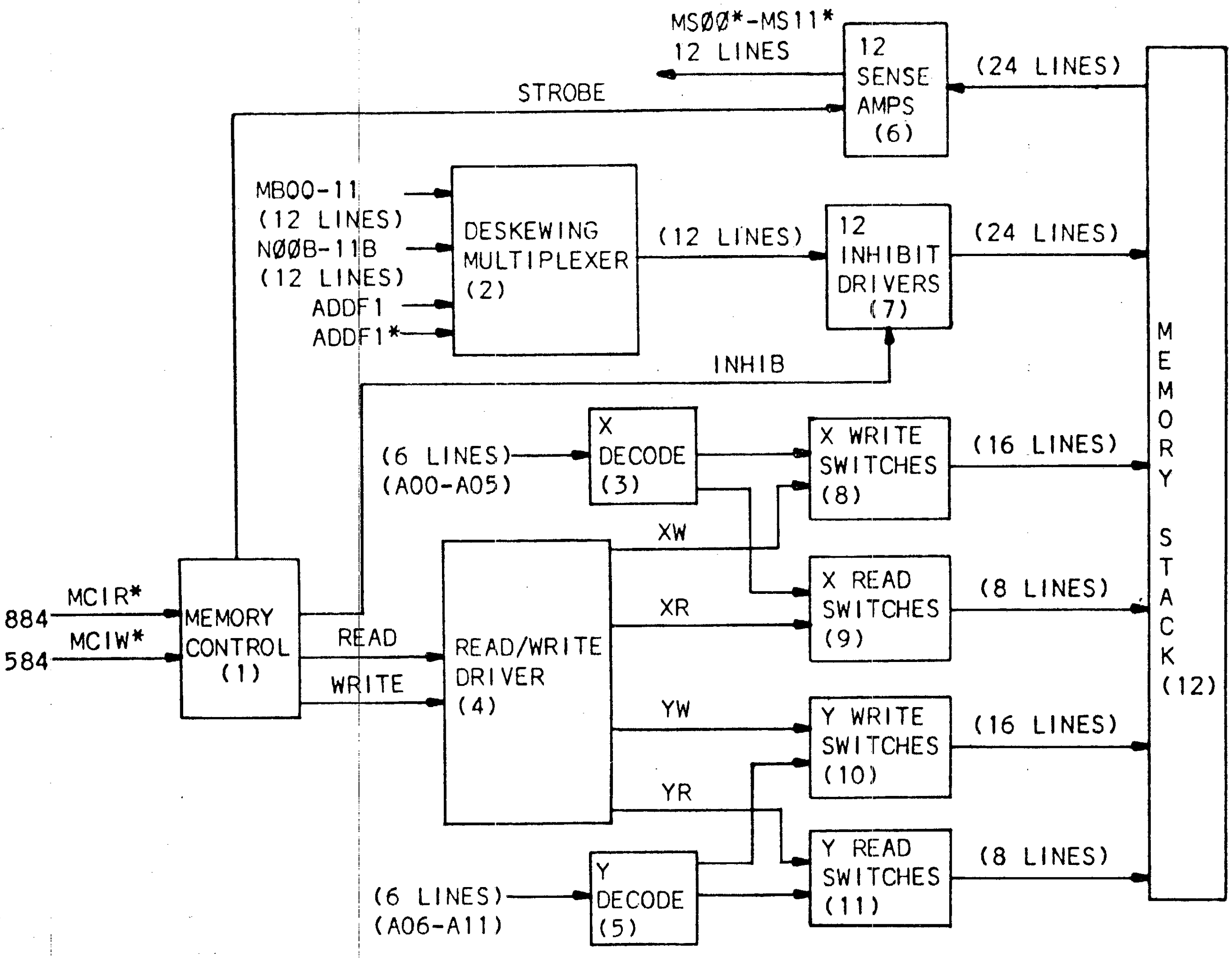


Figure 4-79. 4K Memory, Block Diagram

(4) produces the current pulses that drive the select wires of the memory stack. The X and Y read and write switches (8 through 11) route the current pulses from the read/write driver through the appropriate select wires of the memory stack. The X and Y decoders (3 and 5) decode the contents of address register of the central processor to control the X and Y switches, read and write, respectively, for purposes of addressing. The inhibit drivers (7) accept data from the memory data register in the central processor, and on the basis of this data, produce current pulses on the inhibit wires of the memory stack. The sense amplifiers (6) transform the current pulses from the sense wires of the memory stack into logic signals that are transferred to the memory buffer register of the central processor.

4.5.3.1.1 Memory Stack. The Memory Stack of the 4K memory is a 64x64x12 three dimensional array of cores organized as 12 64x64 square planar mats arranged as shown in Figure 4-80. Each read or write operation affects exactly one bit in each mat so that 12-bit words are read and written. There are 4,096 (64x64 = 4,096) 12-bit words in the Memory Stack.

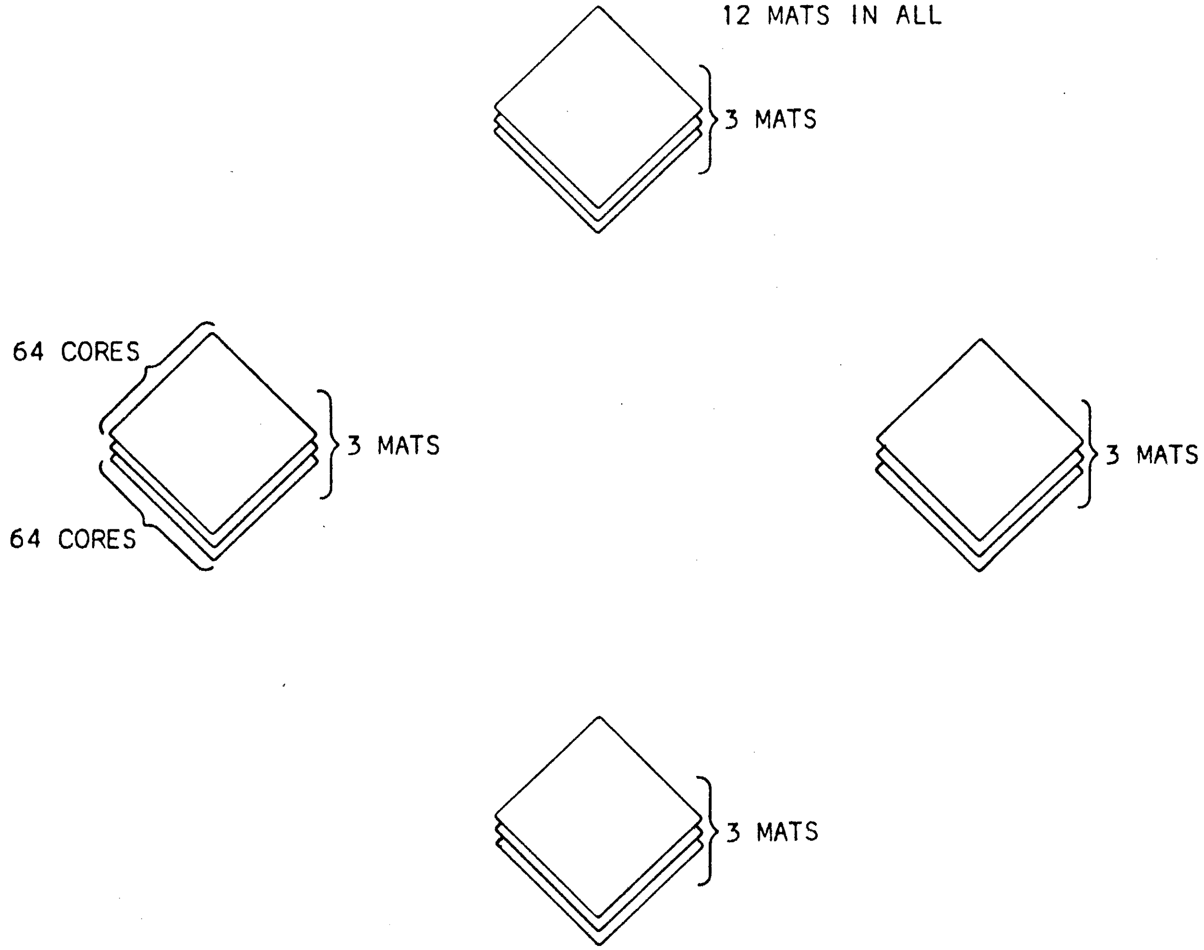


Figure 4-80. Core Mat Position

The mats that make up the memory stack are identical, each having 64 X-select wires, 64 Y-select wires, one sense wire and one inhibit wire. An illustration of the arrangement is shown in Figure 4-81. The illustration shows only a 4x4 array for reasons of simplicity and is only meant to show the most important features of the wiring arrangement. Notable is the fact that the inhibit wire must pass through all cores in the mat in the same direction, relative to read. The sense wire runs through half the cores in each direction, diagonally, for best noise cancelling. Each X-select wire passes directly through a single row of cores and each Y-select wire passes directly through a single column. Thus, in the actual mat, each select wire passes through 64 cores.

In order that as few select switching circuits as possible are required for the Memory Stack, the select wires are interconnected as shown in Figure 4-82. Each of the select terminals is connected by a single switch. By turning on one switch on each side of the mat (four switches) single X- and Y-select wires can be made to carry current and thereby affect a single core in each mat. The X- and Y-select wires pass through all 12 mats, one at a time. Thus only whole 12-bit words can be accessed.

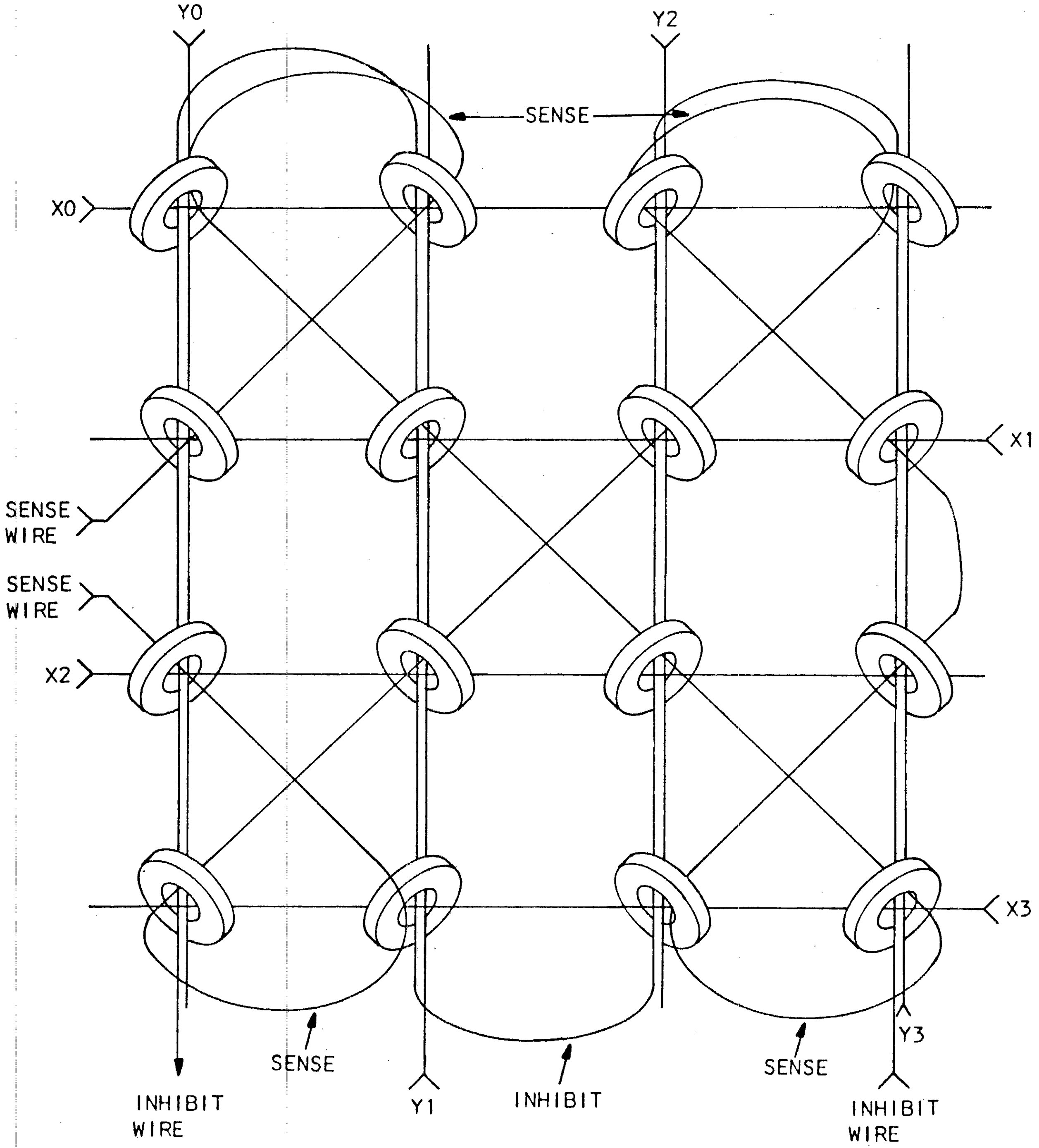


Figure 4-81. Typical Core Mat Layout

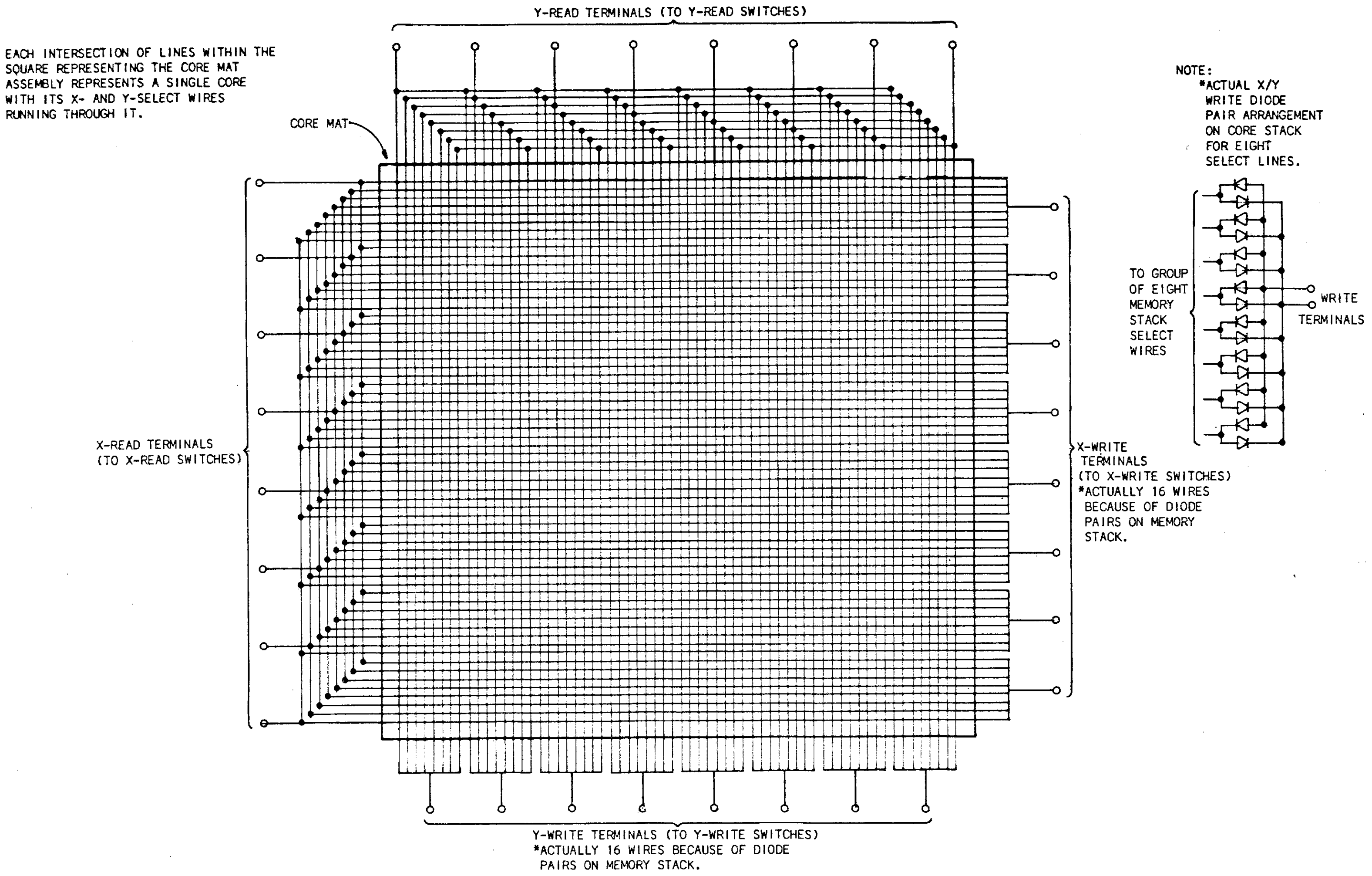

Y-READ TERMINALS (TO Y-READ SWITCHES)
EACH INTERSECTION OF LINES WITHIN THE SQUARE REPRESENTING THE CORE MAT ASSEMBLY REPRESENTS A SINGLE CORE WITH ITS X- AND Y-SELECT WIRES RUNNING THROUGH IT.
CORE MAT
NOTE:
*ACTUAL X/Y WRITE DIODE PAIR ARRANGEMENT ON CORE STACK FOR EIGHT SELECT LINES.
TO GROUP OF EIGHT MEMORY STACK SELECT WIRES
WRITE TERMINALS
X-READ TERMINALS (TO X-READ SWITCHES)
X-WRITE TERMINALS (TO X-WRITE SWITCHES)
*ACTUALLY 16 WIRES BECAUSE OF DIODE PAIRS ON MEMORY STACK.
Y-WRITE TERMINALS (TO Y-WRITE SWITCHES)
*ACTUALLY 16 WIRES BECAUSE OF DIODE PAIRS ON MEMORY STACK.

Figure 4-82. Select Wire Interconnection on Core Mat

Each mat has a single sense wire and a single inhibit and both ends of each wire form terminals, so each mat has four of these terminals and the entire Memory Stack has 48 (4x12) of these terminals. Thus, there are 24 sense terminals and 24 inhibit terminals.

Half the select terminals (on two sides of the square mat) as illustrated in Figure 4-82 are connected to pairs of diodes (shown in Figure 4-83) contained within the Memory Stack. These pairs of diodes (D2 and D4) prevent sneak current paths during read and write operations. Functionally, these diodes can be considered part of the switches (X and Y Write Switches). There are 32 select terminals in the Memory Stack (since the mats are connected together), so the select (i.e., read and write) switches are actually connected to the Memory Stack via 48 wires (8 plus 16 X and 8 plus 16 Y).

4.5.3.1.2 Memory Control. The Memory Control provides timing signals to the sense amplifier, inhibit drivers, and read/write driver (Figure 4-79) during read and write operations. A schematic diagram of the Memory control is contained in the upper portion of sheet 3 of the 6-sheet 4K memory drawings. The device consists of six IC monostable multivibrators (Fairchild Type 9601) with associated timing and coupling components and two DTL NAND gates (Fairchild Type 832). The left half (on schematic) of the circuit controls the write operation; the two halves operate independently of one another.

Two control signals (MCIR*/6M3A1 and MCIW*/6M3A3) enter the memory control from the central processor. A negative transition of the MCIR* signal triggers integrated circuits MTS10 (6M3A1) and MTS6 (6M3A2). MTS10, when triggered, produces a pulse that is passed through a 75 picofarad coupling capacitor to a NAND gate (MTS11/6M3A1) that produces the STROBE/6M3A2 signal which enables the sense amplifiers for the read operation. Circuit MTS6, when triggered, produces a positive pulse of approximately 600 nanoseconds duration that forms the READ/6M3A2 signal and an inverted pulse of the same duration that forms the READ*/6M3A2 signal. Both signals form inputs to the read/write driver.

A negative transition of the MCIW* signal triggers integrated circuit MTS9 (6M3A4) which produces the inverted pulse signal INHIB*/6M3A4 which is inverted by a NAND gate (MTS11/6M3A4) to produce the INHIB/6M3A4 signal that enables the inhibit drivers during a write operation. The leading edge of the INHIB* signal triggers integrated circuit MTS7 (6M3A4) which supplies a positive pulse of short duration to the input of integrated circuit MTS8 (6M3A4). MTS8 is triggered on the trailing edge of the pulse from MTS7 and produces a positive pulse of approximately 600 nanoseconds duration that becomes the WRITE/6M3A4 signal. Both these signals form inputs to the read/write driver.

There are four 10 kilohm potentiometers in the Memory Control for purposes of adjusting the durations of the pulse outputs of MTS10, MTS6, MTS9 and MTS8. Two of these, MTS6 and MTS8 are set for pulse durations of approximately 600 nanoseconds. MTS9 is set so that the trailing end of its output pulse does no precede the trailing end of the output pulse of MTS8. Circuit MTS7 produces a short duration pulse to delay triggering of MTS8 so that the inhibit current (controlled by INHIB) is established before the write current (controlled by WRITE and WRITE*). The controls that adjust the pulse durations are shown on the MTS board layout located in Section V.

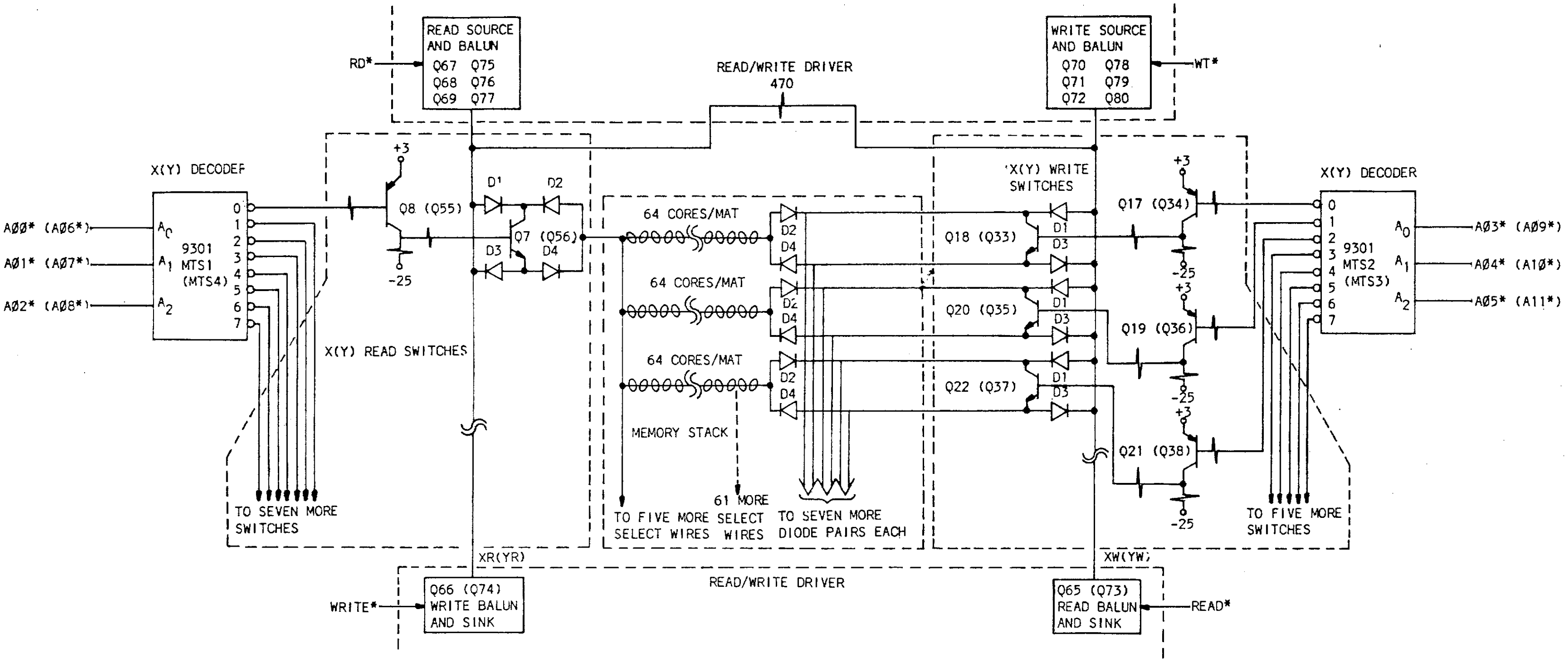


Figure 4-83. Core Selection Circuit, Block Diagram

A memory cycle is produced by a negative transition to initiate a read operation of MCIR* followed (at an appropriate distance in time) by a negative transition of the MCIW* to initiate a write operation.

4.5.3.1.3 Core Selection. The core selection mechanism consists of the X and Y decoders, the X and Y read and write switches and the read/write driver. The overall selection mechanism consists of two identical circuits, one to drive the X-select wires and one to drive the Y select wires. Figures 4-83 is a detailed diagram of one of these circuits. The unparenthesized signal and component names correspond to the X select circuit; the parenthesized names correspond to the Y select circuit.

Twelve bits from the address register (A00* - A11*) provide selection information. Six of these bits (A00* - A05*) determine X selection and the other six bits determine Y selection. Each Type 9301 decoder pulls one of its eight outputs low as determined by the three bit combination on the inputs.

The low-level output from each decoder causes the PNP transistor whose base is connected to that output to become saturated. This causes the NPN transistor that is cascaded with the saturated PNP transistor to become saturated as well.

When one of the NPN transistors of the X and Y read and write switches is saturated, the bridge circuit of which that transistor is an active element allows current to flow through it in either direction. Current coming from the read/write driver passes through the D1 diode, through the collector to emitter path of the saturated NPN transistor and out to the memory stack through the D4 diode. Current coming in the other direction (from the memory stack) passes through the D2 diode, through the collector to emitter path of the saturated NPN transistor, and into the read/write driver through the D3 diode.

Since the transistors of the X and Y read and write switches are saturated when selected, they require time to turn off when the addres is changed before the read operation of the memory cycle is begun. This turn off time is on the order of 50 to 100 nanoseconds.

A selected switch delivers current to twelve X or Y select wires, one on each mat. Each select wire passes through 64 cores of its mat. Only those select wires that are connected on both ends to activated switches are driven with current from the read/write driver. A given pair of X and Y select wires on a mat pass coincidently through a single core of that mat. A single unique core on each mat can have both X and Y select current passing through it for any given twelve bit input to the X and Y decoders. This is known as coincident current condition resulting in core turnover (unless an inhibit current is opposing this cordition). Thus, each address enables a unique set of twelve cores (one from each mat) to be driven, thus accessing a single word. (In the 8K stack there are 24 mats, so two words are accessed with each twelve bit address.)

For the read operation, current is driven by the read/write driver from the read source and balun, through the selected read switch, through the enabled select wire on each mat, through to the selected write switch, to the read balun and sink. The two read baluns are equal windings that share the same core. They and the 120-ohm terminating resistor in parallel with the memory stack help reduce capacitative and noise effects.

For the write operation, current is driven by the read/write driver from the write source and balun, through the selected write switch, through the enabled select wire on each mat, through to the selected read switch to the write balun and sink. The two write baluns are also matched windings on a common core. In the X select circuit, dual transformer T1 contains the four baluns. In the Y select circuit, it is transformer T3.

4.5.3.1.4 Read/Write Driver. The read/write driver generates the current pulses that drive the select lines during read and write operations. A schematic of the read/write driver is located on the lower portion of the 4K Memory sheet 3.

Signals RD*, READ*, WT*, and WRITE* (6M3B1) form the inputs to two nearly identical circuits. The left circuit supplies current pulses for the X-select wires as signals XR and XW (6M3A2). The right circuit supplies current pulses for the Y-select wires as signals YR and YW (6M3A3).

The left circuit operates as follows; for a read operation inverted pulses appear on the RD* and READ* signal connections. When RD* goes low, the emitter of transistor Q69 (6M3A1) goes low, turning Q69 on which causes transistors Q68 and Q67 (6M3A1) to conduct. These three transistors form a current source driving current into the upper winding of the left half of read balun, transformer MTST1 (6M3B1). Transformers T1 and T3 are the X and Y read/write balun source transformers. Most of the current flowing out of the current source follows this path; through the XR connection (a source sink bus) through the X-READ, a single X-select wire then returns via the XW connection. The amplitude of this current is adjustable by the 10 ohm potentiometer at drawing location 6M3A2. For a write operation, inverted pulses appear on the WT* and WRITE* signal connections. When WT* goes low, transistor Q73 (6M3A2) turns on, turning on transistors Q71 and Q70 (6M2A3). These three transistors form a current source driving current into the lower winding of the right half of the X write balun (transformer MTST1/6M3B2), through the XW connection (a source sink bus), one of eight X-select wires, through 64 cores, returns via the XR connection, through the upper winding of the right half of MTST1 and finally through transistor Q66 (6M3B2) which has been turned on due to the low WRITE* signal coupled through transformer T2. As in the read operation, most of the current does not flow through the 470 ohm resistor, but rather flows through the same path as for the read operation but in the opposite direction.

The right circuit operates in an identical fashion in conjunction with the Y read and write switches to drive the Y-select wires of the memory stack.

The RD* and WT* signals which control the current sources of the read/write driver are generated from the READ (6M3B3) and WRITE (6M3B3) signals respectively by two identical pulse generators (6M3B3). For a read operation, when the READ signal goes high, transistor Q84 is turned on and as a result, the RD* signal drops from +5 volts to some lower voltage (nominally -2 to -4 volts). This lower voltage is controlled by the voltage on the emitter of transistor Q84 which is controlled by a temperature compensation circuit. The WT* signal is generated by the WRITE signal in the same fashion as the pulse generator that includes transistor Q83.

The temperature compensation circuit employs a 15-kilohm (at 25°C) thermistor (6M3B2) to detect changes in temperature. This thermistor is incorporated into the bias circuit of transistor Q82 (6M3B2). As the temperature rises the resistance of the thermistor decreases, thereby increasing the base voltage at transistor Q1 (6M3B2), which increases the voltage at the emitter of Q81, and at the three parallel 4.7 kilohm resistors (6M3B3). The voltage at the emitters of Q83 and Q84 is thereby increased. This more positive voltage on the emitter of Q83 and Q84 reduces the amplitude of their corresponding collector swings, reducing the current source amplitudes at higher temperatures. The opposite effect occurs if the thermistor temperature decreases.

4.5.3.1.5 X and Y Decoders. Since there are 4,096 words in a memory field (see paragraph 4.5.3), to give each word a unique address requires that each address be a 12-bit binary number. Addresses from the address register of the central processor enter the X and Y decoders and are decoded by them to provide energizing signals to the proper Y read and write switches to access the addressed word.

The X decoder consists of two integrated circuits decoders (MTS1/6M2A3, MTS2/6M2A1, both Fairchild Type 9301) as does the Y-decoder (MTS3/6M2B1, MTS4/6M2B3). The X decoder accepts as inputs six of the twelve address bits (A00*, A01*, A02*/6M2A3, A03*, A04*, A05*/6M2A1), three bits going to each 9301 decoder. The Y decoder accepts the other six bits (A06*, A07*, A08*/6M2B3, A09, A10, A11/6M2B1). Each of the four integrated circuit decoders provides an energizing level (logical Ø) to a single switch, leaving the others nonenergized. When a given 12-bit address is applied to the X and Y decoders, one output of each of the address decoders provides an energizing signal to its associated selector switch; the four selector switches thus energized, access the word in core associated with the given address (2 for X, 2 for Y axis).

4.5.3.1.6 X and Y Read and Write Switches. The X and Y Read and Write switches are four groups of electronic switches that route current pulses from the read/write driver through the proper X- and Y-select wires during read and write operations.

All the switches are identical if the diodes contained in the memory stack are considered to be included in the X and Y Write switches. Each switch is designed to carry current in either direction when an energizing signal (logical Ø) from the X- or Y-decoder is present on its input. A schematic of an individual switch is shown in drawing location 6M2A4. When output Ø of MTS1 (6M2A3) is a logical 1 (+5 volts), transistor Q8 is cutoff causing transistor Q7 to be cutoff as well. The four diodes in the circuit form a bridge that does not conduct in either direction when Q7 is cutoff. When the output of MS1 is a zero (logical Ø) Q8 is turned on, turning Q7 on. When this happens, the diode bridge will conduct currents (via the collector-emitter circuit of Q7) in either direction between the connection with signal XR/6M2A3 and the memory stack (MTSAA/6M2A4).

X-Write Switches - The X Write Switches are the eight switch circuits at drawing location 6M2A2. During a write operation, current from the write current source connection XW/ 6M2A2, flows through the selected member of the X-read switches to the XR/6M2A3 connection.

X-Read Switches - The X Read Switches are the eight switch circuits at drawing location 6M2A4. During a read operation current flows from the current source to the connection XR/6M2A3, through the selected member of the X read switches, through a single X-select wire of the memory stack, through the selected member of the X Write Switches to the XW/6M2A1 connection.

Y-Write Switches - The Y Write Switches are the eight switch circuits at drawing location 6M2B2. The operation of these switches, in conjunction with the Y read switches and the Y-select wires, is completely analogous to the operation of the X-write switches.

Y-Read Switches - The Y Read Switches are the eight switch circuits at drawing location 6M2B4. The operation of these switches, in conjunction with the Y write switches and the Y-select wires, is completely analogous to the operation of the X read switches.

4.5.3.1.7 Sense Amplifiers. The Sense Amplifiers are a group of twelve integrated circuits (MIS4-MIS14, Type 154OG) whose schematic is shown at drawing locations 6M1A1 and 6M1B1. Each of the integrated circuits is connected to both ends of a single sense wire of the memory stack and detects bipolar voltage pulses generated in this wire during the read operation. A logical 1 (STROBE/6M1B2) signal enables the output of each integrated circuit. If a sufficiently large voltage pulse is present at the output of a sense wire while the STROBE signal is high (during the read (strobe) operation), the integrated circuit associated with that wire outputs a logical Ø pulse. Otherwise, the output of the integrated circuit stays at a logical 1. The output of the sense amplifiers are MS00* - MS11*/6M1A1, 6M1B1 which are the inputs to the memory buffer register of the central processor.

4.5.3.1.8 Inhibit Multiplexer. The inhibit multiplexer consists of six integrated circuits (MIS16 - MIS21/6M1A3, 6M1B3) that select, as input to the inhibit drivers the data in the memory data register (MB00 - MB11/6M1A3, 6M1B3) or the first twelve bits of the memory buffer register (MB00 - MB11/6M1A3, 6M1B3) in the central processor. Signals ADDF1/ 6M1B2 and ADDF1*/6M1B2 control selection between the two central processor registers, which is the source of the input data; a logical 1 level ADDF1* selects the memory data register. The 4K option does not really utilize the inhibit multiplexer because ADDF1* is tied to +5 volts so that the memory data register is always the source of input data. The data which is input to the inhibit multiplexer is inverted at its output.

4.5.3.1.9 Inhibit Drivers. The schematic of the inhibit drivers is located on 4K memory sheet no. 1 at the far right. They consist of twelve NAND gates (integrated circuits MIS1-MIS3/6M1A3, 6M1B3), six dual transformers (T1 - T6/6M1A4, 6M1B4) and six dual driver modules (containing transistors Q1 through Q12). These components comprise twelve identical circuits of which a detailed schematic is at drawing locations 6M1A3 and 6M1A4. A logical 1 pulse INHIB/6M1B2 signal enables the NAND gates. If the second input to a particulat gate (from the inhibit multiplexer) is a logical Ø level, the output of that gate remains a logical 1 level and that driver circuit remains inactive (does not inhibit). If the second input is a logical 1 level, the output of the gate falls to a logical Ø level and a current from the +5 volt supply then flows through the primary of the associated transformer. The current thereby induced in the transformer secondary turns on the transistor in the driver circuit. When turned on, the transistor shorts (through a 100-ohm 5-watt resistor) one end of the associated inhibit wire of the memory stack to an approximate -30 volt level (-V). The other end of the inhibit wire is connected to a +5 volt level so that when the transistor is on a current flows through its associated inhibit wire. For example, when the voltage at pin 3 of AND gate MIS3 (6M3A3) goes low, transistor Q12 (6M3A4) turns on; this results in a current flow into connection MI1 - DD (6M3A4) to -V, through the inhibit wire through 64 cores out of connection MI1 - 26, and through transistor Q12.

During a write operation, those inhibit drivers are activated that correspond to bit positions that contain logical Ø's.

4.5.3.2 8K MEMORY. The 8K Memory is identical to the 4K memory (paragraph 4.5.3.1) except for its augmented stack, sense amplifiers, inhibit drivers, and inhibit multiplexer. Operation of the 8K memory differs from that of the 4K memory only in matters of data transfer paths. A block diagram of the 8K memory is shown in Figure 4-84.

Since the 8K memory consists of more than one field, the memory field control logic electronics are required with this option to control field selection. These electronics are described in paragraph 4.5.3.5.

4.5.3.2.1 8K Memory Stack. The 8K Memory Stack is a 64x64x24 core array which stores 4,096 pairs of 12-bit words for a total capacity of 8, 192 words. Since only pairs of words are selectable with this arrangement, double words are either read or written in each memory operation and the central processor ignores half of each double word, using only the single word desired.

4.5.3.2.2 8K Sense Amplifiers. Since double-words are accessed in each read operation in the 8K memory, 24 sense amplifier circuits are required. Operation of the 8K sense amplifiers is identical to that of the 4K sense amplifiers. The schematic of the 8K sense amplifiers appears in drawing locations 10M1A1, 10M1B1, 10M2A1, and 10M2B1.

During a read operation the 24 outputs of the 8K sense amplifier (MS00*, MS05*/ 10M1A1, MS06* - MS11*/10M1B1, MS12* - MS17*/10M2A1, MS18* - MS23*/10M2B1) are used to load the 24-bit memory buffer register of the central processor. The memory field control logic controls a multiplexer in the central processor (Bus and MBR Multiplexer) which selects the first half (bits Ø - 11) or last half (bits 12-23) of the memory buffer register contents to be used by the central processor.

4.5.3.2.3 8K Inhibit Multiplexer. As explained in paragraph 4.5.2.1, the read operation is a data-destructive process. Since only double-words can be accessed in the 8K memory stack a means must be provided to preserve the unused portion of a double-word until it

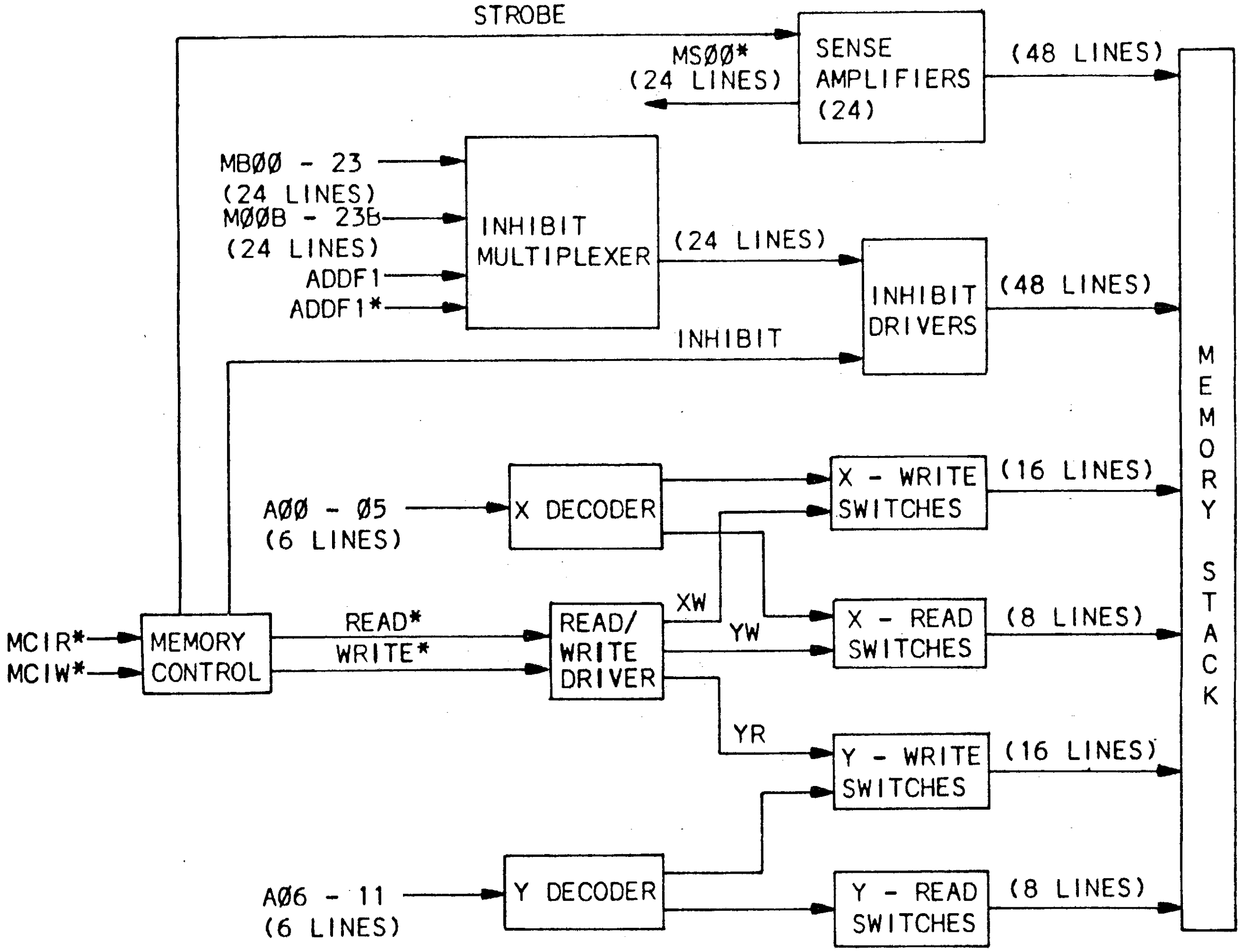


Figure 4-84. 8K Memory, Block Diagram

can be restored by the write operation of the memory cycle. (Paragraph 4.5.2.3.) The memory buffer register in the central processor stores the 24-bit double-word produced by each read operation until the succeeding read operation is performed (during the next memory cycle). The 8K inhibit multiplexer gates the unused half of the memory buffer register contents from the central processor into the proper 8K inhibit drivers during the write operation.

The schematic of the 8K inhibit multiplexer is shown on drawings 10M1A3, 10M1B3, 10M2A3, and 10M2B3. It consists of twelve integrated circuits (Fairchild Type 9005) working in parallel. The six integrated circuits on sheet 10M1 receive data inputs MB00 - MB11/10M1A3, 10M1B3 from the memory buffer register at the control processor and M00B - M11B/10M1A3, 10M1B3 from the memory data register of the central processor. The six integrated circuits on sheet 10M2 receive data inputs MB12 - MB23/10M2A3, 10M2B3 from the memory buffer register and M00B - M11B from the memory data register. Signals ADDF1/10M1B2. 10M2B2, which are generated by the memory field control logic, control the multiplexer. When signals ADDF1 = 1 and ADDF1* = Ø, signals MB00 - 11 and M00B - M11B are selected. Thus, the unused word is restored from the memory buffer register into its former location, while the contents of the memory data register finds its way into the former location of the used word.

4.5.3.2.4 8K Inhibit Drivers. Since double-words are accessed in each write operation in the 8K memory, 24 inhibit drivers are required. Operation of the 8K inhibit drivers is identical to that of the 4K inhibit drivers. The schematic of the 8K inhibit drivers is identical to that of the 4K inhibit drivers. The schematic of the 8K inhibit drivers appears on the far right of sheets 10M1 and 10M2. Input to the 8K inhibit drivers comes from the 24-bit output of the 8K inhibit multiplexer.

4.5.3.3 12K MEMORY. The 12K memory consists of an 8K memory (paragraph 4.5.3.2) internal to the computer plus a 4K memory extension which is identical to the 4K memory (paragraph 4.5.3.1). The memory field control logic controls access to the three memory fields. The 8K memory and the 4K memory extension share common data connections with the memory buffer register and the memory data register in the central processor. In the case of signals MS00* - MS11*, this interconnection results in a wired - OR* configuration. Common connections to control signals ADDF1 and ADDF1* are also shared. However, signals MCIR* and MCIW* service the 8K memory (see Figure 4-84) only, with MCIRX* and MCIWX* replacing these signals in the 4K memory extension. (See Figure 4-79.)

4.5.3.4 16K MEMORY. The 16K memory consists of an 8K memory (paragraph 4.5.3.2) internal to the computer, plus an 8K memory extension which is identical to the 8K memory. The memory field control logic controls access to all four memory fields. The 8K memory field and the 8K memory extension share common data connections with the memory buffer register and the memory data register in the central processor. In the case of signals MS00*-MS23*, this interconnection results in a wired-OR configuration. Common connections to control signals ADDF1 and ADDF1* are also shared. However, signals MCIR* and MCIW* service the 8K memory only (see Figure 4-84) with MCIRX* and MCIWX replacing these signals in the 8K memory extension.

4.5.3.5 MEMORY FIELD CONTROL LOGIC. The core memory of the ND812 Computer is divided into memory fields, each containing 4,096 uniquely addressable 12-bit word locations. Thus, the 4K, 8K, 12K, and 16K memory options contain one, two, three, and four fields, numbered Ø, 1, 2, and 3 respectively. Since a 12-bit word can represent only 4,096 unique addresses ($2^{12} = 4,096$), a single word of memory cannot fully specify the location of a particular storage word in a memory larger than the 4K option. In the three largest capacity memory options, 12-bit words are used to represent location within memory fields, with the desired memory field specified by two bits of information by various means.

The memory field control logic performs two functions; first, it controls the way in which memory fields are selected for any memory reference. Second, it has the capabitily to remember the memory field that was specified prior to the occurrence of an I/O interrupt, or before a JPS instruction is executed and can automatically return to that field upon execution of the proper JMP indirect instruction.

In performing the first of these functions, the memory field control logic controls signals ADDF0 and ADDF1 whose states provide the two bits of information required to specify one of the four possible fields.

A block diagram of the memory field control logic is shown in Figure 4-85. Blocks 2 through 13 are data handling elements. The control logic (1) decodes various instruction information and timing signals from the central processor to control the flow of information out of these data handling devices. The MFR input multiplexer (11), the memory field register (12), and the output multiplexer (13) perform the data handling functions for production of the memory field selection signals (ADDF0 and ADDF1). The other data handling elements are involved in the automatic return function. Descriptions of the hardware these blocks represent are presented in the next 13 paragraphs. Descriptions of their interactions under various circumstances are included in paragraph 4.5.4.2.

4.5.3.5.1 Input Multiplexer. The purpose of the input multiplexer is to route twelve bits of data into the memory field control logic either from signals A00* - A11*/10M5A1, 10M5B1, or signals OUT00*-OUT11*/10M5A1, 10M5B1. The input multiplexer consists of six Fairchild Type 9005 integrated circuits (MFC C14, C15, C16/10M5A1; D16, E16, F16/ 10M5B1). Control signals MFCI0*, MFST1, and MFST2 control the selection of input signals. When MFCI0* is high (making MFST1 and MFST2 low) signals A00* - A11* are inverted by the input multiplexer to form signals PLD00 - PLD11/10M5A1, 10M5B1, respectively. When MFCI0 is low (which is the case when a LDREG, LDJK, or RJIB instruction is being executed), MFST1 and MFST2 are high, and signals OUT00* - OUT11* are inverted to become signals PLD00 - PLD11, respectively. Signals PLD00 - PLD11 form data inputs to the JPS address register and interrupt address register, and signals PLD02 - PLD05 form data inputs to the status register loading gates.

4.5.3.5.2 JPS Add-ess REgister. The JPS address register is a 12-bit register that stores the output of the input multiplexer (PLD00 - PLD11/10M5A1, 10M5B1) when a JPS or LDREG instruction is executed. It consists of three Fairchild Type 9300 integrated circuits (MFC A16, B15, B16/10M5A2). Signals JPS00 - JPS11/10M5A2 are the data outputs of the JPS address register which form data inputs of the comparator multiplexer. Signals PEJPS*/ 10M5A1, CKJPS*/10M5A1, and CLJPS*/10M5A1 control loading and clearing functions of the register. A low PEJPS* signal enables data signals PLD00 - PLD11 to be loaded when an inverted pulse CKJPS* signal is received. A low CLJPS* signal clears the register (sets all bits to zero) regardless of the other inputs to the register.

4.5.3.5.3 INT Address Register. The INTaddress register is a 12-bit register that stores the output of the input multiplexer (PLD00 - PLD11/10M5A1, 10M5B1) when an interrupt occurs or a LDREG instruction is executed. It consists of three Fairchild Type 9300 integrated circuits (MFC A13, A14, A15/10M5B2). Signals INT00 - INT11/10M5B2 are the data outputs of the INT address register which form data inputs for the comparator multiplexer. Signals PEINT*/10M5B1, CKINT*/10M5B1, and CLINT*/10M5B1 control loading and clearing functions of the register. A low PEINT* signal enables data signals PLD00 - PLD11 to be loaded when an inverted pulse CKINT* signal is received. A low CLINT* signal clears the register (sets all bits to zero) regardless of the other inputs to the register.

4.5.3.5.4 Comparator Multiplexer. The comparator multiplexer routes twelve bits of data into the comparator and the EXT gates from either the JPS address register or the INT address register. It consists of six Fairchild Type 9005 integrated circuits (MFC E11, D11, D13,

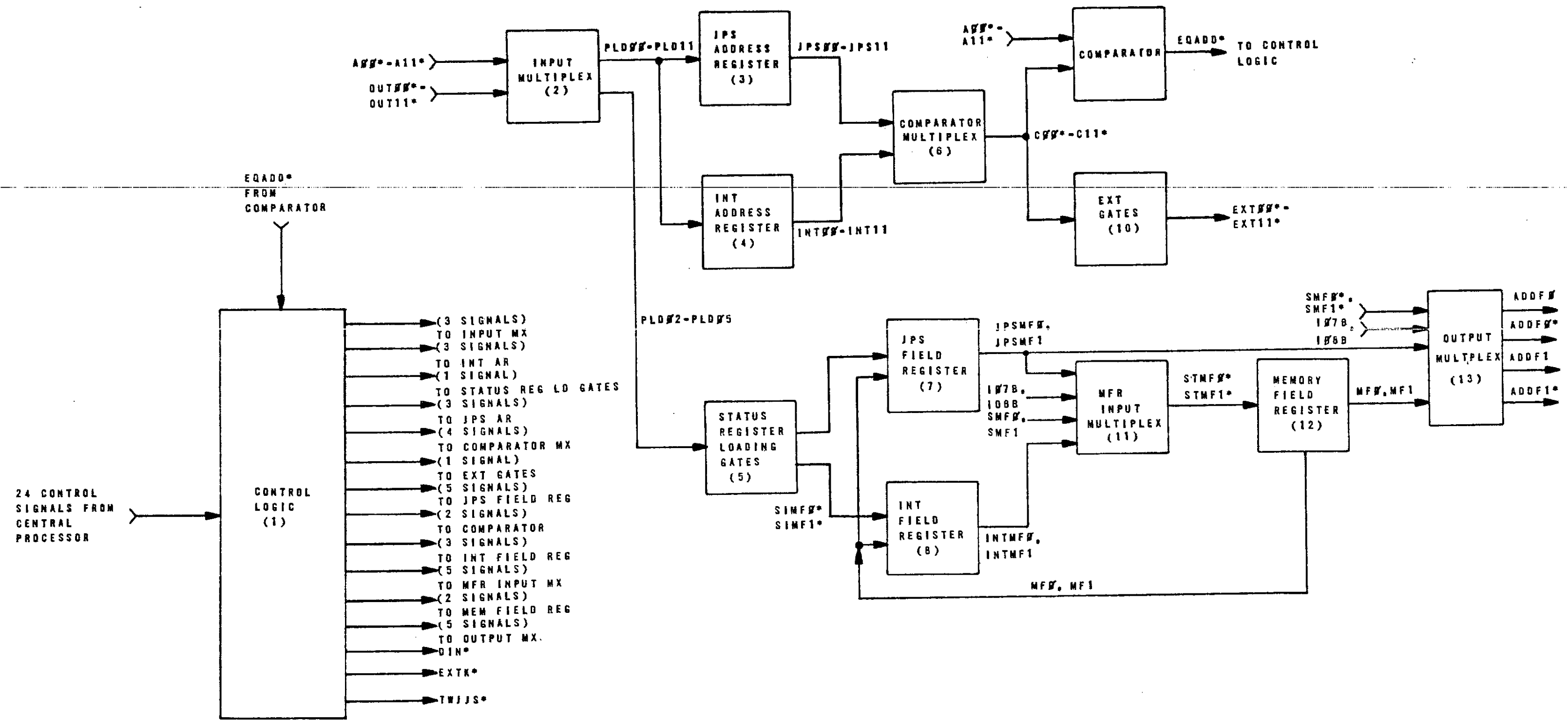


Figure 4-85. Memory Field Control Logic, Block Diagram

D12, C13, C12/10M5A2, 10M5B2). Control signals BP51/10M5B2, BP52/10M5B2, BP61/10M5A2, and BP62/10M5A2 control the selection of input data signals. When BP51 and BP52 are high, signals JPS00 - JPS11/10M5A2, 10M5B2 are inverted by the comparator multiplexer and to become data signals C00* - C11*/10M5A3, 10M5B3, respectively. When BP61 and BP62 are high, signals INT00 - INT11/10M5A2, 10M5B2 are inverted by the comparator multiplexer to become signals C00* - C11*, respectively. Signals C00* - C11* are data inputs of the comparator and the EXT gates.

4.5.3.5.5 Comparator. The comparator compares, on a bit-to-bit basis, data signals C00* - C11*/10M6A1, 10M6B1 with data signals A00* - A11*/10M6A1, 10M6B1; it gates the result of this comparison with control signals INDFF/10M6A1 and FLMEN*/10M6A1 to produce the single control signal EQADD*/10M6A1. The comparator consists of three Signetics Type 8200 integrated circuits (MFC B11, B12, B13/10M6A1) and one Fairchild Type 9007 integrated circuit (MFC B14/10M6A1). Signal EQADD* is low if and only if INDFF and FLMEM* are both high, and signals C00* - C11* match signals A00* - A11*, respectively.

4.5.3.5.6 EXT Gates. The EXT gates gate data signals C00* - C11*/10M5A3, 10M5B3 to signal connections EXT00* - EXT11*/10M5A3. They consist of twelve buffers (integrated circuits MFC E14, F14/10M5A3, 10M5B3) and twelve two-input NAND gates (integrated circuits MFC D15, E15, F15/10M5A3, 10M5B3). When control signal LDJK/10M5B3 is high, logic levels of C00* - C11* appear in signals EXT00* - EXT11*, respectively. When LDJK is low, signals EXT00* - EXT11* are isolated from signals C00* - C11* and remain high unless drawn low by circuitry external to the memory field control logic. Signals EXT00* - EXT11* are data outputs of the memory field control logic.

4.5.3.5.7 Status Register Loading Gates. The status register loading gates gate signals PLD02 PLD03/10M6B2, PLD04, PLD05/10M6B1 to form signals SJMF0*, SJMF1*/10M6B3, SIMF0*, SIMF1*/10M6B2, respectively. They consist of four two-input NAND gates (Fairchild Type 9002 integrated circuits; MFC F09/10M6B2, 10M6B3). When signal SFF/10M6B1 is high (which happens when a RJIB instruction is executed), signals PLD02 and PLD03 are inverted by the status register loading gates to form signals SJMF0* and SJMF1* that are inputs to the JPS field register, and signals PLD04 and PLD05 are inverted to form SIMF0* and SIMF1* that are inputs to the INT field register. When signal SFF is low, signals SJMF0*, SJMF1*, SIMF0*, and SIMF1* are all high.

4.5.3.5.8 JPS Field Register. The JPS field register (7) stores two memory field selection bits from the memory field register (12) when a JPS instruction is executed. It is composed of a single Fairchild Type 9022 dual J/K flip-flop integrated circuit (MFCA12/10M6B3). When signals ENJPS*/10M6B3, PWRDY*/10M6B2, SJMF0*/10M6B3, and SJMF1*/10M6B3 are all high, the JPS field register (7) is enabled, and an inverted pulse EP0* signal then results in signals MF0 and MF1 from the memory field register (12) being loaded into the JPS field register (7), whose output signals JPSMF0/10M6B3 and JPSMF1/10M6B3 indicate the logic values stored in it. The contents of the JPS field register can also be affected in two other ways. When the PWRDY* signals is low, both bits of the register are held at zero. When SJMF0* is low, bit Ø of the register (JPSMF0) is set high. When SJMF1* is low, bit 1 of the register (JPSMF1) is set high.

4.5.3.5.9 INT Field Register. The INT field register (8) stores two memory field selection bits from the memory field register (12) when an interrupt is initiated. It is composed of a single Fairchild Type 9022 dual J/K flip-flop integrated circuit (MFC A10/10M6B2). When signals LDREG*/10M6B1, PWRDY*/10M6B2, SIMF0/10M6B2, and SIMF1/10M6B2 are all high, the INT field register (8) is enabled and an inverted pulse CKINT*/10M6B2 signal then results in signals MF0 and MF1 from the memory field register (12) being loaded into the INT field register (8) whose output signals INTMF0/10M6B2 and INTMF1/10M6B2 indicate the logic values stored in it. The contents of the INT field register (8) can also be affected in two other ways. When the PWRDY* signal is low, both bits of the register are held at zero. When SIMF0* is low, bit Ø of the register (INTMF0) is set high. When SIMF1* is low, bit 1 of the register (INTMF1) is set high.

4.5.3.5.10 MFR Input Multiplexer. The MRF input multiplexer (11) routes two bits of data into the memory field register (12) from one of four pairs of data signals. It consists of two Fairchild Type 9008 integrated circuits (MFC B10/10SM6A2, D14/10M6B3). Control signals LDPR/10M6A2, 10M6B3; EQJPS/10M6A2, 10M6B3; EQINT/10M6A2, 10M6B3; EQFF*/10M6A2, 10M6B3; and TWJJS/10M6A2, 10M6B3 determine which pair of input data signals is selected to be inverted by the MRF input multiplexer (11) to become data output signals STMF0*/10M6A2 and STMF1*/10M6B3. When LDPR is high, SMF0/10M6A2 and SMF1/10M6B3 are selected. When EQJPS is high, JPSMF0/10M6A2 and JPSMF1/10M6B3 are selected. When EQINT is high, INTFM0/10M6A2 and INTMF1/10M6B2 are selected. When both EQFF* and TWJJS are high I07B/10M6B2 and I08B/10M6B3 are selected. Data signals STMF0* and STMF1* are inputs to the memory field register (12).

4.5.3.5.11 Memory Field Register. The memory field register (12) holds the two memory field selection bits that determine the current memory field. It is composed of a single Fairchild Type 9022 dual flip-flop integrated circuit (MFC A11/10M6A2). Contents of this register can be changed only when a positive transition of signal IC03/10M6B2 occurs to clock the levels of signals STMF0*/10M6A2 and STMF1*/10M6B3 into the flip-flops of MFC A11.

4.5.3.5.12 Output Multiplexer. The output multiplexer directly controls memory field specification signals ADDF0/10M7B4, ADDF1/10M7A4, and their complements. It consists of two Fairchild Type 9008 integrated circuits (MFC F13/10M7A4, MFC H13/10M7B4). Signals CSADD/10M7B3, EQMEM/10M7A3, TWMEM/10M7A3, TWMEM*/10M7A3, and CFF/10M7B3 control selection of the source of the pair of signals that will generate ADDF0 and ADDF1, when CSADD is high and SMF0/10M7A3 and SMF1/10M7A3 are selected. When EQMEM is high JPSMF0/10M7B3 and JPSMF1/10M7A3 are selected. When TWMEM is high, I07B/10M7B3, and I08B/10M7A3 are selected. When CFF and TWMEM* are both high, MF0/10M7B3 and MF1/10M7A3 are selected. When none of these conditions hold (as occurs during interrupts), both ADDF0 and ADDF1 are forced low, selecting memory field Ø.

4.5.3.5.13 Control Logic. The control logic (1) consists of all the gates and flip-flops of the memory field control logic (10M5, 10M6, 10M7) that do not handle data. It receives 24 control signals from the central processor and generates 33 signals to the data handling elements (2-13, Figure 4-85) of the memory field control logic. Table 4-109 lists the signals which enter the control logic from the central processor and explains what they indicate. Table 4-110 lists the signals from the control logic (1) to the data handling elements. In addition to these signals, the control logic receives one control signal EQADD*/ 106A1) generated by the comparator (9) and outputs three control signals (DIN*/10M6B3, EXTK*/10M6B4, TWJJS*/10M7A2) to the central processor. The low level EQADD* signal from the comparator indicates that the contents of the address register of the processor matches the contents of either the JPS address register (3) or the INT address register (4). The DIN* signal, when low or high, indicates that data is to be transferred to or from the processor, respectively. When EXTK* is low or high it indicates that the K or J register is to be involved in the data transfer, respectively. The TWJJS signal indicates that the current instruction is a two word JMP or JPS instruction with its field change bit set.

4.5.4 MEMORY UNIT DETAILED THEORY OF OPERATION

This paragraph is divided into two subparagraphs; paragraph 4.5.4.1, Memory Operation and paragraph 4.5.4.2, Memory Field Control Logic Operation. The first subparagraph gives a detailed explanation of how the memory stack and its associated electronics (drivers, sensors, etc.) interacts with the memory data register, the memory buffer register and the bus and MBR multiplexer of the central processor in performing read and write operations. The memory field control logic is treated as a black box in paragraph 4.5.4.1 and details of its operation are provided in paragraph 4.5.4.2.

4.5.4.1 MEMORY OPERATION. The memory provides to the processor the capability to to rapidly read and write 12-bit words with a large number of locations available, each with a unique address. To facilitate the flow of information between the processor and the memory electronics, two registers are used; the 24-bit memory buffer register and the 12-bit memory data register. A data path from the memory buffer register to the memory data register is provided by the bus and MBR multiplexer. The data flow among these devices is dependent upon which of the four possible memory fields is accessed.

The following four paragraphs each provide a detailed explanation of control signals and data flow involved in accessing a particular memory field. These four paragraphs will then be followed by a detailed explanation of control signals and data flow within the memory electronics.

4.5.4.1.1 Accessing Memory Field Ø. A memory cycle executed when signals ADDF0*/ 8B3 and ADDF1*/17A1 are both high accesses memory field Ø. When ADDF0* is high the MCIR*/8B4 and MCIW*/8B4 pulses are enabled to occur in conjunction with pulses PU2/8B3 and PU5W/8B3, respectively.

Fetch Cycle - The MCIR* pulse at PU2 initiates a read operation in the memory (Figure 4-86) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). The bus and MBR multiplexer (1) presents only the first twelve bits (MB00 - MB11 via PMR00 through PMR11) of this double-word to the memory data register (2) when ADDF1* is high. The 12-bit word from the memory data register (M00B-

Table 4-109. Control Logic Inputs From Processor

Signal Name/Location	Meaning
BP1/10M7B1 BP5*/10M5A4 BP6*/10M5A4 BP7B/10M6A3	Timing signals generated by processor clock indicating various stages of the basic phase.
CPPU*/10M7A1	Timing pulse generated by processor clock.
CSFF*/10M7B1	Indicates that present memory cycle is dedicated to a cycle steal.
EP0/10M7A2 EP0*/10M6B3 EP1/10M7A3	Timing signals generated by processor clock indicating various stages of the execute phase.
FLLOC*/10M6A1	Low level indicates that location 0000_8 or 7777_8 is the operand address of the current instruction.
HLT/10M5A4	High level indicates that bits 3-6 of the instruction register contain 1110, 0100, 0001, or 0000.
I06B/10M7A1 I10*/10M6B4 I11B*/10M6B3	Indicate contents of bits 6, 10, and 11 of instruction register.
INDFF/10M6A1	High level indicates that current instruction has its indirect bit set.
INTFF*/10M7B1	Low level indicates that an interrupt is being initiated.
I/O/10M5A4	High level indicates that current instruction is an input/output instruction.
JMP*/10M7A1	Low level indicates that current instruction is a jump instruction.
JPS*/10M7A1	Low level indicates that current instruction is a jump-to-subroutine instruction.
LDPR/10M7B1	High level indicates that the LOAD AR switch on the front panel is depressed.
PU0B/10M7A1	Timing signals generated by the processor clock indicating various stages of a machine cycle.
PWRDY*/10M7B2	Low level indicates that power has just come on and is not sufficient to begin operation.
TWFF*/10M7A1	Low level indicates that current instruction is a two word instruction.

Table 4-110. Control Logic Outputs To Data Handling Elements

Signal Name/Location	Indication or Purpose
BP51/10M5A4 BP52/10M5B4	High during BP5.
BP61/10M5B4 BP62/10M5B4	High during BP6.
CFF/10M7B3	High when current memory field (MF0, MF1) is accessed.
CKINT*/10M6A4	High-to-low-to-high transition clocks INT address register (4) when interrupt is being initiated.
CKJPS*/10M7A4	High-to-low-to-high transition clocks JPS address register (3) when JPS instruction with field change bit set is executed.
CLINT*/10M6A4	Low level clears INT address register (4) during automatic field return from interrupt routine.
CLJPS*/10M6A4	Low level clears JPS address register (3) during automatic field return from subroutine.
CSADD/10M7B3	High indicates that cycle steal is taking place.
ENJPS/10M7A3	High enables JPS field register (7) to be loaded when JPS instruction with field change bit set is executed.
EP0*/10M6B3	Low during EP0.
EQFF*/10M6A3	Low indicates that the contents of either the JPS or INT address register matched the contents of the address register.
EQINT/10M6A3	High indicates that the contents of the INT address register (3) matched the contents of the address register.
EQJPS/10M6A3	High indicates that the contents of the JPS address register (4) match the contents of the address register.
EQMEM/10M7A3	High when EQJPS is high and the current instruction is not a JMP instruction.
FLLMEM*/10M6A1	Low indicates that current operand address is 0000_8 or 7777_8.
IC03/10M6B2	Low-to-high transition clocks memory field register.
INDFF/10M6A1	High level indicates an indirect fetch.
LDJK/10M6B4	High indicates current instruction is an LDJK instruction.
LDPR/10M7B1	High indicates LOAD AR switch on front panel is depressed.

Table 4-110. Control Logic Outputs To Data Handling Elements (Cont'd.)

Signal Name/Location	Indication or Purpose
LDREG*/10M6B4	Low indicates that current instruction is an LDREG instruction.
MFCIO*/10M5B4	Low indicates that current instruction is an LDJK, LDREG, or RJIB instruction.
MFST1/10M5B4 MFST2/10M5B4	Complements of MFCIO*
PEINT*/10M6A4	Low level enables INT address register (4) to be parallel loaded.
PEJPS*/10M7A4	Low level enables JPS address register (3) to be parallel loaded.
PULL/10M6A2	Always high (pulls up unused inputs).
PWRDY*/10M7B2	Low indicates that power level is not yet sufficient to begin operation.
SFF/10M6B1	High during RJIB instruction to enable status register loading gates (5).
TWJJS/10M7A3	High indicates that current instruction is a two-word JMP or JPS instruction with its field change bit set.
TWMEM/10M7A3	High indicates that current instruction is a two-word memory reference instruction with its field change bit set.
TWMEM*/10M7A3	Complement of TWMEM.

M11B) and the second 12-bit word from the memory buffer register (3) (MB12-MB23) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIW* pulse (at PU5) initiates a write operation to restore the whole double word that was read at PU2.

Store Cycle - The MCIR* pulse at PU2 initiates a read operation in the memory (Figure 4-87) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). At PU3 (or in some cases PU4) the memory data register (2) is loaded with the data word to be stored from the bus or from the memory buffer register (MB00 - MB11) (B00 - B11) via the bus and MBR multiplexer (1). The twelve bits from the memory data register (M00B - M11B) and the second twelve bits from the memory buffer register (MB12 - MB23) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIW* pulse (at PU5) initiates a write operation to restore the twelve bits to memory field 1 (MB12 - MB23) and write the memory data register information into memory field Ø.

4.5.4.1.2 Accessing Memory Field 1. A memory cycle executed when signal ADDF0*/8B3 is high and signal ADDF1*/17A1 is low accesses memory field 1. When ADDF0* is high

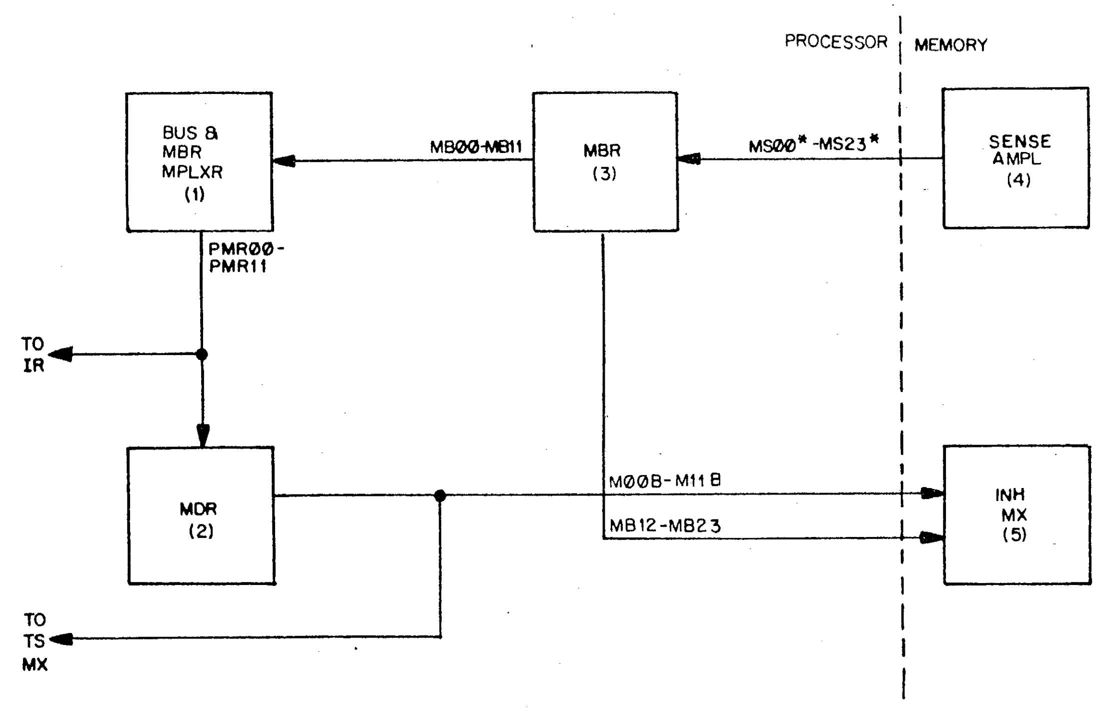


Figure 4-86. Data Flow For Fetch Cycle From Memory Field Ø Or 2

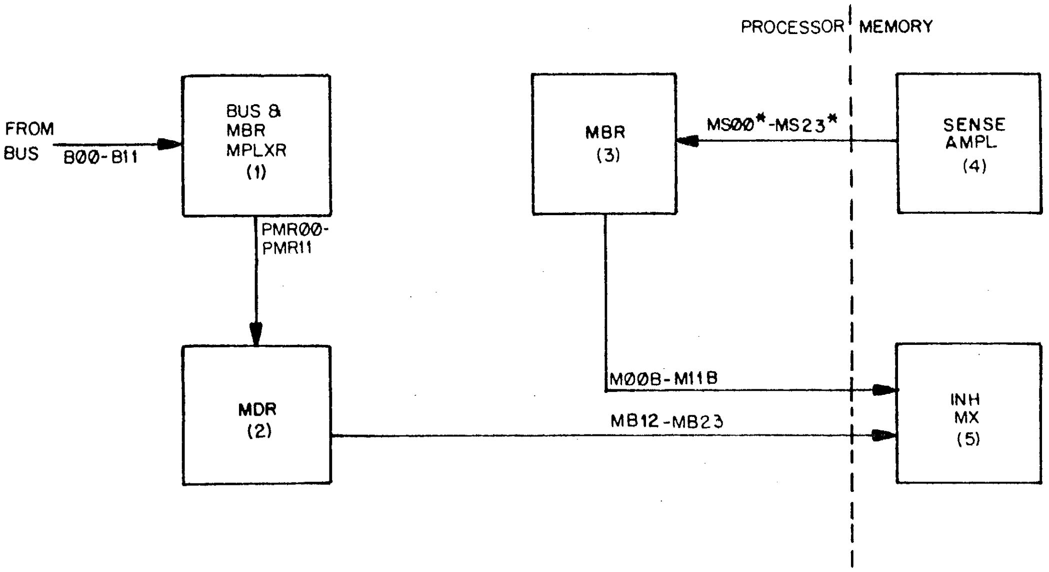


Figure 4-87. Data Flow For Store Cycle To Memory Field Ø Or 2

the MCIR*/8B4 and MCIW*/8B4 pulses are enabled to occur in conjunction with pulses PU2/8B3 and PU5W/8B3, respectively.

Fetch Cycle - The MCIR* pulse at PU2 initiates a read operation in the memory (Figure 4-88) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). The bus and MBR multiplexer (1) presents only the last twelve bits (MB12 - MB23 via PMR00 through PMR11) of this double-word to the memory data register (2) when ADDF1* is low. The 12-bit word from the memory data register (M00B - M11B), and the first 12-bit word from the memory buffer register (3) (MB00 - MB11) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIW* pulse (at PU5) initiates a write operation to restore the entire double-word that was read at PU2.

Store Cycle - The MCIR* pulse at PU2 initiates a read operation in the memory (Figure 4-89) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). At PU3 (or in some cases PU4) the memory data register (2) is loaded with the data word to be stored from the bus or from the memory buffer register (MB00 - MB11) (B00 - B11) via the bus and MBR multiplexer (1). The twelve bits from the memory data register (M00B 1 M11B), and the first twelve bits from the memory buffer register (MB00 - MB11) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIW* pulse (at PU5) initiates a write operation to restore the twelve bits to memory field Ø (MB00 - MB11) and write the memory data register information into memory field 1.

4.5.4.1.3 Accessing Memory Field 2. A memory cycle executed when signal ADDF0*/8B3 is low and ADDF1*/17A1 is high accesses memory field 2. When ADDF0 is high the MCIRX*/8B4 and MCIWX*/8B4 pulses are enabled to occur in conjunction with pulses PU1/8B3 and PU5W/8B3, respectively.

Fetch Cycle - The MCIRX* pulse at PU1 initiates a read operation in the memory extension (Figure 4-86) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). The bus and MBR multiplexer (1) presents only the first twelve bits (MB00 - MB11 via PMR00 through MPR11) of this double-word to the memory data register (2) when ADDF1* is high. The 12-bit word from the memory data register (M00B - M11B) and the second 12-bit word from the memory buffer register (3) (MB12 - MB23) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIWX* pulse (at PU5) initiates a write operation to restore the entire double word that was read at PU1.

Store Cycle - The MCIRX* pulse at PU1/8B3 initiates a read operation in the memory extension (Figure 4-87) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifier (4). At PU3 (or in some cases PU4) the memory data register (2) is loaded with the data word to be stored from the bus or from the memory buffer register (PMR00 - PMR11) via the bus and MBR multiplexer (1). The twelve bits from the memory data register (M00B - M11B) and the second twelve bits from the memory buffer register (MB12 - MB23) form the double word which the inhibit multiplexer (5) transfers to

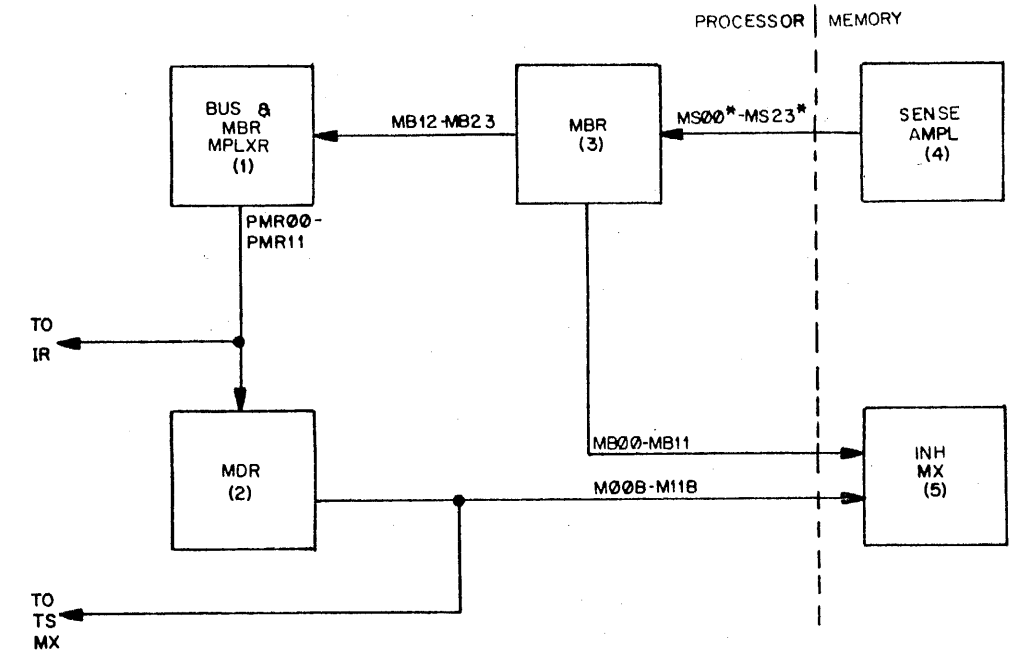


Figure 4-88. Data Flow For Fetch Cycle From Memory Field 1 Or 3

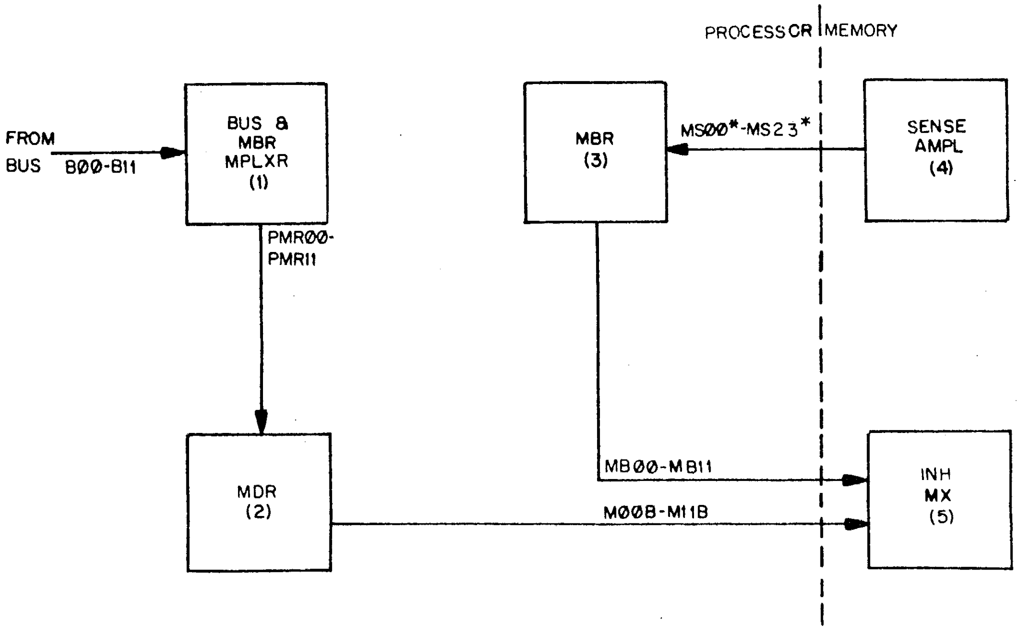


Figure 4-89. Data Flow For Store Cycle To Memory Field 1 Or 3

the memory stack when the MCIWX* pulse (at PU5) initiates a write operation to restore the twelve bits from the memory field 3 (MB12 - MB23) and write the memory data register information into memory field 2.

4.5.4.1.4 Accessing Memory Field 3. A memory cycle executed when signals ADDF0*/8B3 and ADDF1*/17A1 are both low accesses memory field 3. When ADDF0* is low the MCIRX*/8B4 and MCIWX*/8B4 pulses are enabled to occur in conjunction with pulses PU1/8B3 and PU5W/8B3, respectively.

Fetch Cycle - The MCIRX* pulse at PU1 initiates a read operation in the memory extension (Figure 4-88) that reads a 24-bit double-word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). The bus and MBR multiplexer (1) presents only the last twelve bits (MB12 - MB23) of this double-word to the memory data register (2) when ADDF1* is low. The 12-bit word from the memory data register (M00B - M11B) and the first 12-bit word from the memory buffer register (3) (MB00-MB11) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIWX* pulse (at PU5) initiates a write operation to restore the entire double word that was read at PU1.

Store Cycle - The MCIRX* pulse at PU1 initiates a read operation in the memory extension (Figure 4-89) that reads a 24-bit double word from the memory stack into the memory buffer register (3) via the sense amplifiers (4). At PU3 (or in some cases PU4) the memory data register (2) is loaded with the data word to be stored from the bus (PMR00 - PMR11) via the bus and MBR multiplexer (1). The twelve bits from the memory data register (M00B - M11B) and the first twelve bits from the memory buffer register (MB00 - MB11) form the double word which the inhibit multiplexer (5) transfers to the memory stack when the MCIWX* pulse (at PU5) initiates a write operation to rewrite the twelve bits from memory field 2 (MB00 - MB11) and restore the memory data register information into memory field 3.

4.5.4.1.5 Memory Electronics. The operation of the memory electronics (Figure 4-85) is controlled by four control signals from the central processor (MCIR*/8B4 MCIW*/8B4, MICRX*/8B4, and MCIWX*/8B4), a twelve bit address (A00 - A11) from the address register of the central processor, and signals ADDF1/10M7A4 and ADDF1*/10M7A4 from the memory field control logic. A timing diagram of memory electronics operation for control signals is shown in Figure 4-90. The sequence of operation is as follows.

a. At the trailing edge of PU0 the address register is loaded. Its output is decoded by the X- and Y-decoders to enable four of the X- and Y-read and write switches.

b. During PU1 the memory buffer register is cleared.

c. At the start of PU2, MCIR* (or MCIRX*) goes low causing monostable STROBE/10M4A2 and READ/10M4A2 to go high and READ*/10M4A2 to go low. The strobe monostable output remains high for approximately 200 nanoseconds and enables the outputs of the sense amplifiers on the trailing edge of this interval. Read remains high for about 600 nanoseconds and causes

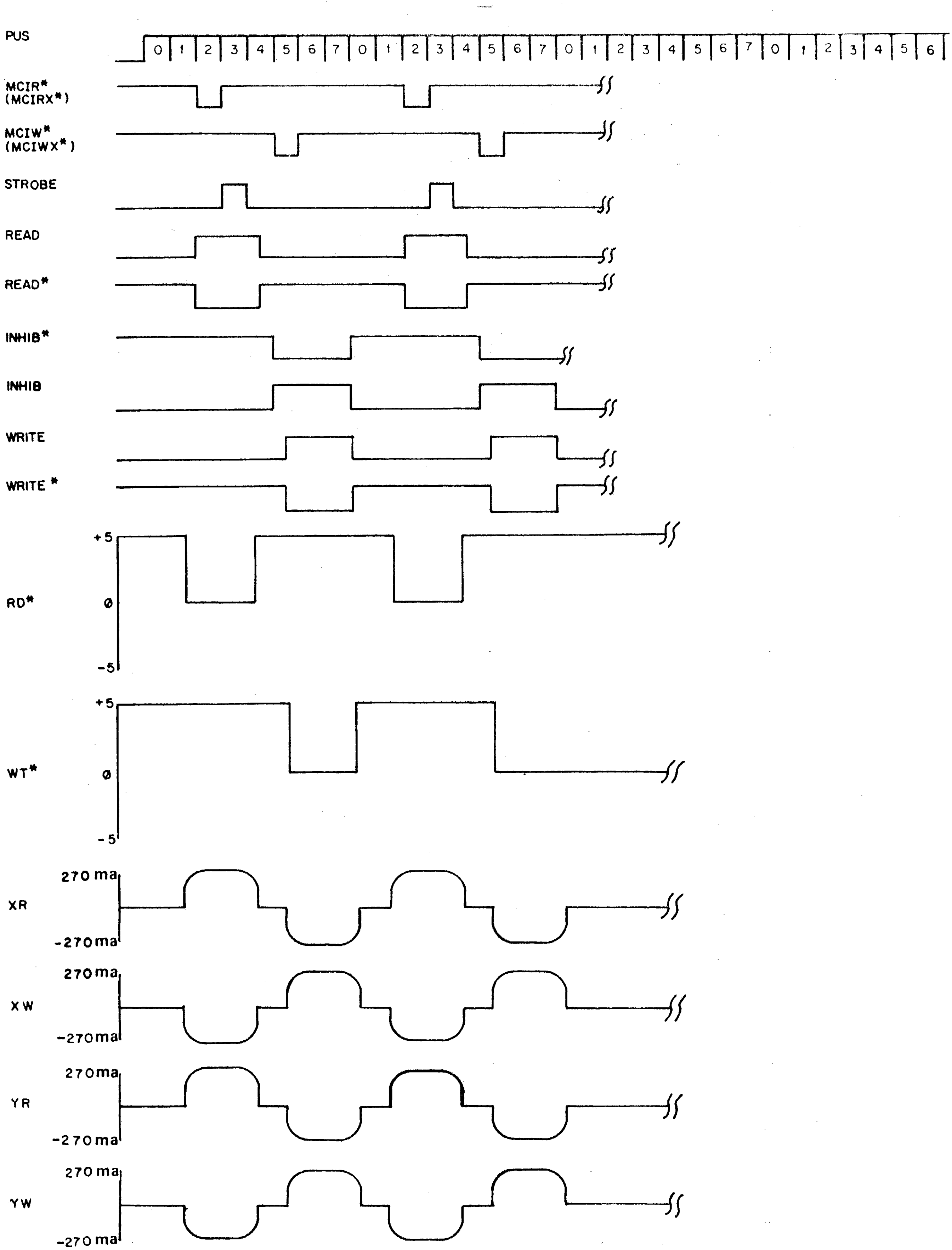


Figure 4-90. Memory Electronics, Timing Diagram

the non-logic level RD*/10M4B2 to go low for this interval. The 600 nanosecond READ* and RD* signals cause the read/write driver to generate currents approximately 270 milliamps in amplitude and 600 nanoseconds in duration from XR/10M4A2 to XW/10M5A2 and from YR/10M4A3 to YW/10M4A3. All this results in a 24-bit word from the memory being strobed into the memory buffer register.

d. During PU3, (sometimes during PU4) the memory data register is loaded.

e. At the start of PU5, MCIW* (or MCIWX*) goes low causing INHIB*/10M4A4 to go low for some interval greater than 600 nanoseconds. INHIB* is inverted to generate the INHIB/10M4A4 signal which enables the inhibit drivers. When INHIB* goes low, it triggers a positive pulse of short duration generated by IC MTS 7/10M4A4.

f. On the trailing edge of the pulse from MTS 7, the 600 nanosecond WRITE/10M4A4 and WRITE*/10M4A4 pulses are initiated. When WRITE is high it causes non-logic level WT*/10M4B3 to go low. The low WRITE* and WT* signals cause the read/write driver to generate currents approximately 270 milliamps in amplitude and 600 nanoseconds in duration from XW to XR and from YW to YR. This results in data from the memory buffer register and the memory data register being written into the memory stack. Signals ADDF1 and ADDF1* control the inhibit multiplexer and thereby determine the data flow during this stage as explained in the preceding paragraphs.

4.5.4.2 MEMORY FIELD CONTROL LOGIC, DETAILED DESCRIPTION. The memory field control logic performs the tasks of memory field selection and automatic memory field return. The first task involves controlling signals ADDF0, ADDF1, and their complements. The second task involves storage of memory field selection bits and addresses.

At any given time the contents of the memory field register specify what is referred to as the current memory field. In most cases instructions access the current memory field, though two-word instructions may access any field, ignoring the contents of the memory field register. Two-word JMP and JPS instructions and interrupts can change the current memory field. The memory field register can also be loaded from the front panel. During cycle steals the memory field is specified by the device which requested the cycle steal.

When the current memory field is changed by two-word JPS instruction or an interrupt, the old current memory field is stored in the JPS field register (for a JPS instruction) or in the INT field register (for an interrupt). Subsequent execution of the proper JMP indirect instruction then results in automatic restoration of the remembered memory field as the current memory field. The automatic field return mechanism is affected by three instructions specially intended to do so; the LDJK, the LDREG, and the RJIB instruction.

Each of the following twelve paragraphs discusses a different set of circumstances affecting memory field control logic operation. A slight modification of the symbology for simplified logic representation will be used here; to indicate data transfers, equal signs are used. For example, the statement

(↑DIN*/10M6B4) (↑EXTK*/10M6B4) OUT00-OUT11 = J REGISTER

is to be interpreted; When signals DIN* and EXTK* are both high, bits OUT00 - OUT11 are set equal respective to the twelve bits contained in the J register. The names on both sides of the equal sign do not necessarily refer to real signals and are meant only to represent data transfer or transmission.

4.5.4.2.1 Single Word Instruction. A single word memory reference instruction is fetched from the current memory field as determined by the contents of the memory field register. The operand (for a memory reference instruction) is also fetched from this field as is an indirect address if the indirect bit of the instruction is set. Table 4-111 is a simplified description of the signals that control the outputs (ADDF0, ADDF0*, ADDF1, ADDF1*) of the output multiplexer for the duration of execution of a single word instruction.

Table 4-111. Current Memory Field Selection, Fundamental Operations

Period	Simplified Logic	Event
(Asynchronous)	1. (↑CSADD*) (↑EQMEM*) (↑INTAD*)→ ↓CFF*/10M7B3 2. ↓CFF*→↑CFF/10M7B3 3. (↑CFF) (↑TWMEM*) (↓CSADD) (↓EQMEM) (↓TWMEM)→(ADDF0*/10M7B4 = MF0*/10M7B3) (ADDF0/10M7B4 = MF0/10M7B3) 4. (↑CFF*) (↑TWMEM*) (↓CSADD) (↓EQMEM) (↓TWMEM)→(ADDF1*/10M7A4 = MF0*/10M7A3) (ADDF1/10M7A4 = MF1/10M7A3)	The contents of the memory field register (MF0, MF1) are gated to ADDF1*, ADDF0* for current memory field selection.

4.5.4.2.2 Two-Word Instruction with Field Change Bit Unset. When the field change bit of a two word instruction is unset, all memory references involved in execution of the instruction access the current memory field as in the case of a single word instruction. A simplified diagram of the operation of the logic involved in memory field control in this case is shown in Table 4-111.

4.5.4.2.3 Two-Word Memory Reference Instruction with Field Change Bit Set. Memory field selection for a two word memory reference instruction with its field change bits set (except JPS and JMP) is performed as in the case where that bit is unset, until the execute phase; that is, during the basic phases, memory field selection is performed as described in Table 4-111. During the execute phase, however, the memory field from which the operand is fetched is selected by the field selection bits (bits 10 and 11) of the instruction.

Though the current memory field is selected for the two (three if the instruction specified indirect addressing) basic phases with no field change until the execute phase, certain changes occur in the state of the controlling logic during the basic phases that are important in effecting the field change. Figure 4-90 is the timing diagram that illustrates this for a direct fetch. During the first basic phase at BP3 the instruction register is loaded with the first word of the two-word instruction. The instruction is then recognized as a two word instruction and at BP5 (first basic phase) TWFF* goes low to indicate this. At BP3 of the second basic phase, the contents of the instruction register are shifted by three bit positions causing the field change bit and the two memory field bits of the instruction to appear in bit positions 6,7 and 8, respectively. Thus, signals I06B/10M7A1, I07B/10M7B3 and I08B/10M8A3 represent these bits. At EP0 signal TWMEM/10M7A2 goes high, changing the source of the memory field selection signals (ADDF0, ADDF1) from the memory field register (current memory field) to bits 7 and 8 of the instruction register. At PU1 after the execute phase, REF*/10M7B2 goes low causing TWMEM to go low, thus returning to the current memory field. Table 4-112 is a simplification of the logic in the memory field control logic for the execute phase of this process.

For an indirect instruction of this type, operation and timing are similar, but with an extra basic phase inserted between the second basic phase and the execute phase. During this third basic phase, the current memory field is, of course, selected.

Table 4-112. Two-Word Instruction Memory Field Change, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	EP0	I07→ADDF0, I08→ADDF1	
		1. (↑EP0) (↑RGCLK) (↑TWFF) (↑I06B)→ ↓STWMM*/10M7A2 2. ↓STWMM*→↑TWMEM/10M7A2, ↓TWMEM*/10M7A2 3. (↓CSADD) (↓EQMEM) (↑TWMEM) (↓TWMEM*)→ADDF0/10M7B4 = I07B/10M7B3, ADDF0*/10M7B4 = I08B*/10M7B3 4. (↓CSADD) (↓EQMEM) (↑TWMEM) (↓TWMEM*)→ADDF1/10M7A4 = I08B/10M7A3, ADDF1*/10M7A4 = I08B*/10M7A3	The field specification bits (I07, I08) are gated to become ADDF0 and ADDF1 to change memory fields.
B.	BP1	MFR→ADDF0*, ADDF1*	
		1. ↑BP1 + (↑PU1B) (↑INTCSF)→↓RFF*/10M7B2 2. ↓RFF*→↓TWMEM/10M7A2, ↑TWMEM*/10M7A2 3. (See A. 1-4)	The contents of the memory field register (MF0, MF1) are gated to ADDF0* and ADDF1* to restore current memory field selection.

4.5.4.2.4 Two-Word JMP Instruction with Field Change Bit Set. When a two word JMP instruction having its field change bit set is executed, the contents of the memory field register are changed to match the contents of bits 10 and 11 of the instruction. They are shifted left 3 places during the second basic phase, thereby changing the current memory field. The old contents of the memory field register are not saved by the hardware in this case, so that no automatic return to the old current memory field can be effected.

Table 4-113 contains the simplified logic for the memory field control for the two-word JMP instruction. For the two basic phases of instruction execution (three if the indirect bit is set) the current memory field is accessed as described in Table 4-111. At BP3 of the second basic phase (when the instruction register is shifted to facilitate decoding of the two-word instruction) the low TWJJS*/10M7A2 is generated. This signal forces the processor to enter the execute phase after completion of the basic phases. Then, at EP0, the contents of bits 7 and 8 of the instruction register (bits 10 and 11 of the two-word instruction), represented by signals 107B/10M6A1 and 108B/10M6B3, are loaded into the memory field register to specify a new current memory field.

4.5.4.2.5 Two-Word JPS Instruction with Field Change Bit Set. When a two-word JPS instruction with its field change bit is executed, the current memory field bits (the contents of the memory field register) are stored in the JPS field register and the memory field register is loaded with the memory field bits of the JPS instruction (from bits 10 and 11), thus changing the current memory field. The JPS address register is loaded with the address portion of the instruction (the second word of the two-word JPS instruction or the indirect address if the indirect bit of the instruction is set) when that address appears in the address register of the processor. These procedures set up the memory field control logic for an automatic field return to be signaled by a future JMP indirect instruction.

A simplified description for operation of the memory field control logic during execution of the two-word JPS instruction is shown in Table 4-114. A timing diagram for this is shown in Figure 4-91. This timing diagram assumes direct addressing, but the only difference in the case of indirect addressing is the addition of a third basic phase between the second basic phase and the execute phase. For the duration of the basic phases (two or

Table 4-113. Two-Word JMP Instruction Memory Field Change, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP3	GENERATE ↓TWJJS*	
		1. (↓JMP*) (↑JPS*)→↑IC02/10M7A2 2. ↓TWFF*→↑TWFF/10M7A1 3. (↑IC02) (↑106B) (↑TWFF)→↓TWJJS*/10M7A2	Decoding of the JMP instruction results in assertion of TWJJS*.
B.	EP0	107, 108→MFR	
		1. ↓TWJJS*→↑TWJJS/10M7A3 2. ↑TWJJS→STMF0 = 107B/10M6A2 3. ↑TWJJS→STMF1 = 108B/10M6A3 4. (↑TWJJS) (↑EP0)→↓IC03/10M6B2 5. (↓IC03) (↑PULL)→MF0 = STMF0, MF1 = STMF1	The field specification bits of the two word JMP instruction are loaded into the memory field register.

Table 4-114. Two-Word JPS Memory Field Control, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP3 (first basic phase)	See Table 4-111, A. 1-4)	Select current memory field.
B.	BP3 (second basic phase)	ENABLE DATA TRANSFERS 1. ↓TWFF*→↑TWFF/10M7A1 2. ↓JPS* + ↑JMP*→↑IC02/10M7A2 3. (↑IC02) (↑I06B) (↑TWFF)→↓TWJJS*/10M7A2 4. ↓TWJJS*→↑TWJJS/10M7A3 5. (↑TWJJS) (↑JMP*)→↓ENJPS*/10M7A3 6. ↓ENJPS*→↑ENJPS/10M7A3 7. (↑TWJJS) (↑EQFF*)→STMF0 = I07B 8. (↑TWJJS) (↑EQFF*)→STMF1 = I08B 9. ↑MFCI0*→↓MFST1/10M5A4, ↓MFST2/10M5A4 10. (↑MFCI0*) (↓MFST1) (↓MFST2)→ PLD00-PLD11 = A00-A11	Generate ↓TWJJS*, ↓ENJPS*; gate I07 to STMF0, I08 to STMF1, A00-A11 to PLD00-PLD11.
C.	EP0	MFR→JPSFR 1. (↑ENJPS) (↓EP0*)→JPSMF0 = MF0, JPSMF1 = MF1	The JPS field register is loaded with current memory field specification.
D.	EP0	I07, I08→MFR 1. (↑TWJJS) (↑EP0)→↓IC03/10M6B2 2. ↓IC03→MF0 = I07B, MF1 = I08B	The memory field specification bits of the two-word JPS instruction are loaded into the memory field register.
E.	EP0	AR→JPS AR 1. (↑ENJPS) (↑EP1)→↓PEJPS*/10M7A4 2. ↓PEJPS*→↑PEJPS/10M7A4 3. (↑PEJPS) (↑RGCLK)→↓CKJPS*/10M7A4 4. (↓PEJPS*) (↓CKJPS)→PLD00-PLD11→ JPS00-JPS11/10M5A2	The contents of the address register are loaded into the JPS address register.

three), memory access is to the current memory field. The simplified logic for this is shown in Table 4-111.

At BP3 of the second basic phase, the signals which enable the data transfers from the processor address register to the JPS address register, from the memory field register to the JPS field register, and from the instruction register to the memory field register are generated. The actual data transfers are not performed until the execute phase. These enabling signals, TWJJS/10M7A3, ENJPS*/10M7A3, ENJPS/10M7A3, MFCI0*/10M5A4,

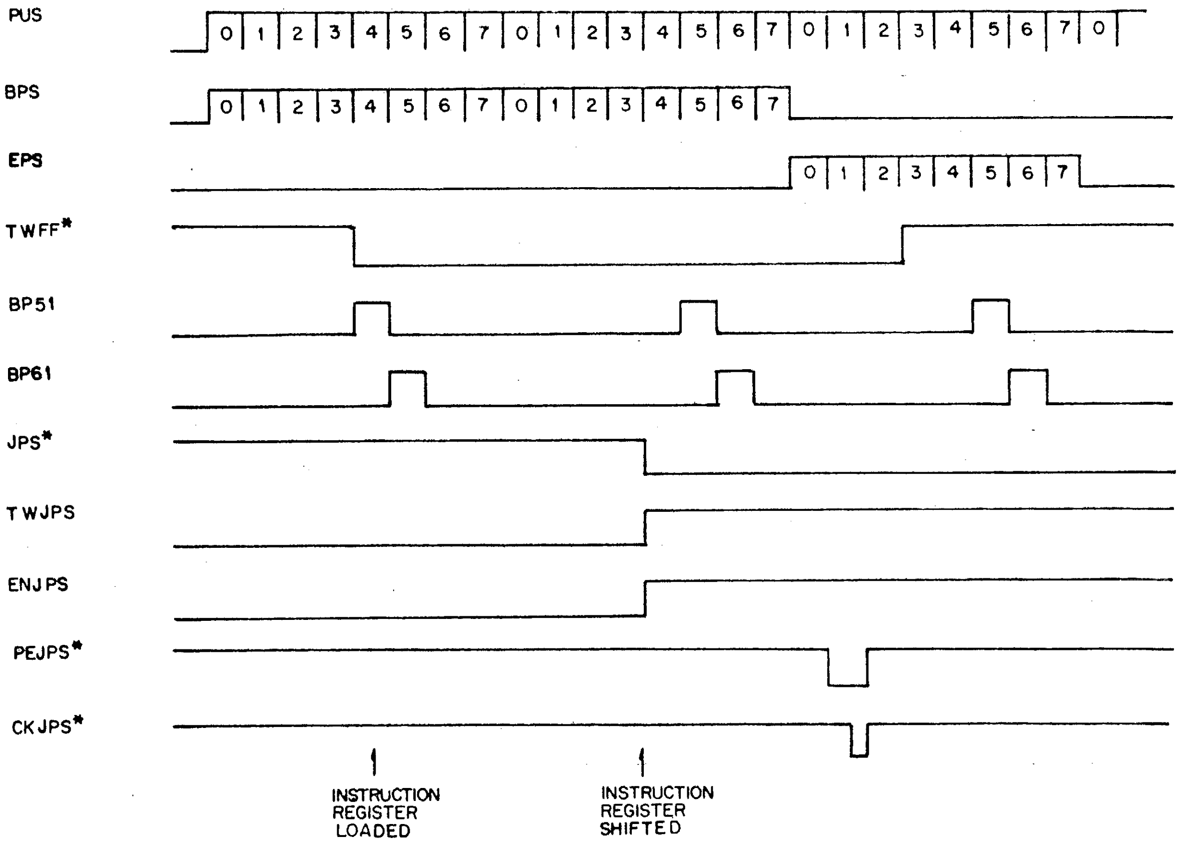


Figure 4-91. Two-Word JPS With Bit 9 Set, Direct Addressing, Timing Diagram

MFST1/10M5A4, and MFST2/10M5A4, are produced as a result of decoding the instruction, which resides in the instruction register, after the instruction register has been shifted by three bit positions.

At EP0, two data transfers take place simultaneously; the old current memory field is loaded into the JPS field register from the memory field register, the new current memory field is loaded into the memory field register from instruction register bits 7 and 8. At EP1 the contents of the address register, which contains the address portion of the two-word instruction at this time, is loaded into the JPS address register.

When this process has been completed, the processor begins execution of a subroutine. Throughout the execution of this subroutine, the following situation obtains: the memory location that was specified by the operand address of the JPS instruction contains the address of the instruction following the JPS instruction in the originating routine, and the JPS field register contains the memory field specification for locating these instructions. The JPS address register contains the operand address of the JPS instruction.

During period BP5 of execution of any instruction, the contents of the JPS address register are gated to the comparator, which compares these contents with the contents of the address register of the processor (Section A of Table 4-115). If these two addresses are equal and signals INDFF/6A1 and FLMEM*/6A1 are both high, a low EQADD*/6A1 signal is generated. The involvement of FLMEM* and INDFF makes assertion of the EQADD* signal an indication that the current instruction is a memory reference instruction going indirect through the location containing the return address to the originating routine, and that this address is neither 0000_8 nor 7777_8. (To prevent erroneous assertion of the EQADD* signal during BP5 when the processor is not executing a subroutine, the JPS address register is cleared when the return to the originating program is perfomed. Returns through location 0000_8 and 7777_8 are disallowed since referencing either of these locations results in a low FLMEM* signal.) If EQADD* is asserted during BP5, signal EQJPS is asserted (set high).

If the instruction being executed when EQJPS is asserted is not a JMP instruction, the memory field control logic selects the memory field specified by the JPS field register for the execute phase of the instruction (Sections B and C of Table 4-115). Thus, the first instruction to be executed after returning to the originating routine can be manipulated by the subroutine. Only this single location in the field or origin can be (that of the JPS return address) accessed in this manner since EQJPS will not be asserted unless an indirect instruction through the location containing the return address is executed.

If the instruction being executed when EQJPS is asserted is a JMP, the return to the program of origin is being executed and Section D and E (instead of B and C) of Table 4-115 describe the events. Signal CLJPS* is asserted to clear the JPS address register during BP7. Signal EQJPS gates the contents of the JPS field register to the memory field register. The memory field register is clocked at PU0 by signal IC03 to load the contents of the JPS field register, restoring the current memory field of the program of origin.

Subsequently, at BP1, signal RFF* is asserted and resets signal EQJPS low. This occurs whether a return has been executed or not. The simplified logic description of this function is shown in Section F of Table 4-115.

4.5.4.2.6 Interrupts. When an interrupt occurs, the current memory field bits (the contents of the memory field register) are stored in the INT field register and the contents of the memory field register are then set to Ø. These procedures set up the memory field control logic for an automatic field return to be signaled by a future instruction.

A simplified description for operation of the memory field control logic during initiation of an interrupt is shown in Table 4-116. A timing diagram is shown in Figure 4-92. Signal INTFF*/10M7B1 goes low at PU7 as the previous instruction is completed. Subsequently, at PU1, memory field Ø is selected by setting ADDF0/10M7B4 and ADDF1/10M7A4 to Ø. At PU1, the INT address register is loaded with the contents of the processor address register, the INT field register is loaded with the contents of the memory field register, and the memory field register is set to zero. At the succeeding PU1 period, the memory field register (now set to zero) is selected, specifying the current memory field as shown in Table 4-111.

Table 4-115. JPS Return, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP5	JPS AR = AR? 1. ↓BP5*→↑BP51/10M5A4, ↑BP52/10M5B4 2. ↑BP51→C00-C06 = JPS00-JPS06 3. ↑BP52→C07-C11 = JPS07-JPS11 4. (A00-A11 = C00-C11) (↑INDFF) (↑FLLOC*)→ ↓EQADD*/10M6A1 5. ↓EQADD*→↑EQADD/10M6A2 6. (↑BP52) (↑PULL) (↑RFF*) (↓CPPU*) (↑EQADD)→↑EQJPS/10M6A3, ↓EQJPS*/10M6A3 7. (↓EQJPS*) (↑EQINT*)→↑EQFF/10M6A3 8. ↑EQFF→↓EQFF*/10M6A3	EQFF* is set low to indicate that the address register contents equal the JPS address register contents.
B.	EP0	JPS PR→ADDF0, ADDF1 1. ↓CPPU*→↑REGCLK/10M7A1 2. (↑JMP*) (↑EP0) (↑EQJPS) (↑REGCLK)→ ↓SEQMM*/10M7A2 3. ↓SEQMM* + ↑EQMEM*→↑EQMEM/10M7A3 4. ↑EQMEM + ↑RFF*→↓EQMEM*/10M7A3 5. ↓EQMEM* + ↑CSADD* + ↑INTAD*→ ↑CFF*/10M7B3 6. ↑CFF*→↓CFF/10M7B3 7. (↓CSADD) (↑EQMEM) (↓TWMEM) (↓CFF)→ ADDF0 = JPSMF0, ADDF1 = JPSMF1/10M7A4, 10M7B4	The contents of the JPS field register are gated to signals ADDF0* and ADDF1* to select JPS field of origin.
C.	BP1	MFR→ADDF0, ADDF1 1. ↑BP1→↓RFF*/10M7B2 2. ↓RFF* + ↑EQMEM→↑EQMEM*/10M7A3 3. (↑EQMEM*) (↑SEQMM*)→↓EQMEM/10M7A3 4. ↑EQMEM* + ↑CSADD* + ↑INTAD*→ ↓CFF*/10M7B3 5. ↓CFF*→↑CFF/10M7B3 6. (↓CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↑CFF)→ADDF0 = MF0/10M7B4, ADDF1 = MF1/10M7B4	The contents of the memory field register are gated to ADDF0, ADDF1 to reselect current memory field.
D.	BP7	0000_8→JPS AR 1. ↓JMP*/10M7A1→↑JMP/10M7A2 2. (↑EQJPS/10M6A3) (↑BP7B/10M6A3) (↑JMP/10M6A3)→↓JPS7*/10M6A3 3. (↓JPS7*/10M6A3) (↑PWRDY*/10M6A3)→ ↑CLJPS/10M6A3 4. ↑CLJPS/10M6A3→↓CLJPS*/10M6A4 5. ↓CLJPS*/10M5A1→JPS00-JPS11 = 0000_8	The JPS address register is cleared.

Table 4-115. JPS Return, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
E.	PU0	JPS FR→MFR 1. ↑EQJPS→STMF0 = JPSMF0/10M6A2 2. ↑EQJPS→STMF1 = JPSMF1/10M6B3 3. (↑EQFF) (↑JMP) (↑PU0B)→↓IC03/10M6B3 4. (↓IC03) (↑PULL)→MF0 = STMF0, MF1-STMF1/10M6A2, 10M6A3	The contents of the JPS field register are loaded into the memory field register.
F.	BP1	↓EQJPS, ↑EQJPS* 1. ↑BP1→↓RFF*/10M7B2 2. ↓RFF*→↓EQJPS, ↑EQJPS*/10M6A3	Signal EQJPS is reset.

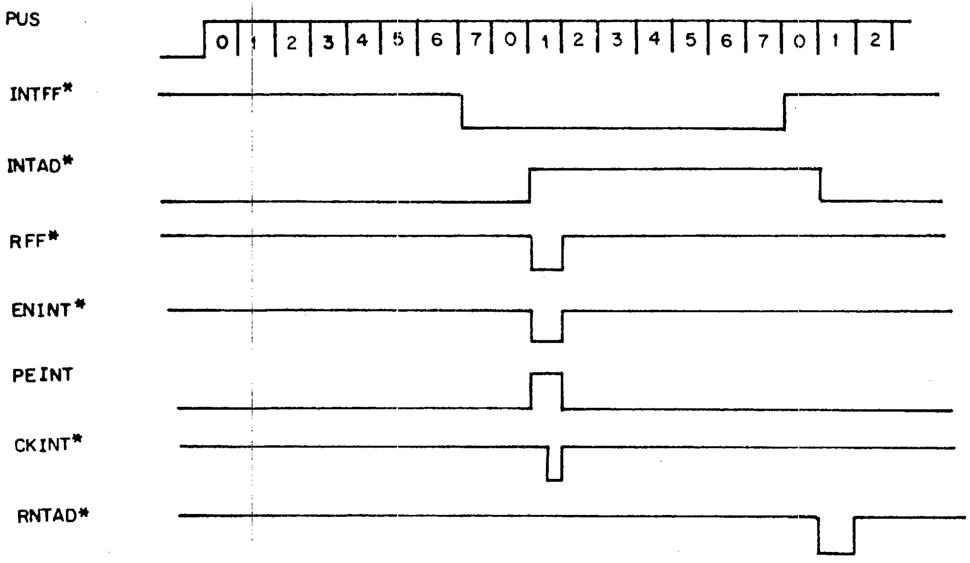


Figure 4-92. Interrupt Initiation, Timing Diagram

After this process is complete the processor begins execution of an interrupt routine. Throughout the execution of this routine, the following situation obtains; the memory location with the incremented trap address (or 00001$_8$) as its address, contains the address of the location of the next instruction to be executed in the interrupt program, and the INT field register contains the memory field specification for locating that instruction. The INT address register contains the incremented trap address.

During period BP6 of execution of any instruction the contents of the INT address register are gated to the comparator, which compares these contents with the contents of the address register of the processor (Section A, Table 4-117). If these two addresses are equal and signals INDFF/6A1 and FLMEM*/6A1 are high, a low EQADD*/6A1 signal is generated. The involvement of FLMEM* and INDFF makes assertion of the EQADD*

Table 4-116. Interrupt Initiation Memory Field Control, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU0	0→ADDF0, ADDF1	
		1. ↓INTFF*→↑INTFF/10M7B1	Memory field 0 is selected by unasserting ADDF0 and ADDF1.
		2. ↓CPPU*→↑REGCLK/10M7A1	
		3. (↑INTFF) (↑PU0B) (↑REGCLK)→ ↓I606/10M7B2	
		4. ↓I606→↑INTAD/10M7B2, ↓INTAD*/10M7B2	
		5. (↓INTAD*) (↑EQMEM) (↑CSADD*)→ ↑CFF*/10M7B3	
		6. ↑CFF*→↓CFF/10M7B3	
		7. (↓CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↓CFF)→↑ADDF0*/10M7B4	
		8. (↓CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↓CFF)→↑IC07/10M7A4	
		9. ↑ADDF0*→↓ADDF0/10M7B4	
		10. ↑IC07→↓ADDF1/10M7A4	
		11. ↓ADDF1→↑ADDF1*/10M7A4	
B.	PU1	AR→INT AR	
		1. (↑INTFF) (↑PU1B)→↓ENINT*/10M6A4	The contents of the address register are stored in the interrupt address register.
		2. ↓ENINT* + ↑REG6*→↑PEINT/10M6A4	
		3. ↑PEINT→↓PEINT*/10M6B4	
		4. (↑PEINT) (↑RGCLK)→↓CKINT*/10M6A4	
		5. ↑MFCI0*→↓MFST1/10M5A4, ↓MFST2/10M5A4	
		6. (↑MFCI0*) (↓MFST1) (↓MFST2)→ PLD00-PLD11 = A00-A11	
		7. (↓PEINT*) (↓CKINT*)→INT00-INT11 = PLD00-PLD11	
C.	PU1	MFR→INTFR	
		1. (See B, 1-4)	The contents of the memory field register are stored in the INT field register.
		2. (↑LDREG*) (↓CKINT*)→INTMF0 = MF0 INTMF1 = MF1/10M7B2	
D.	PU1	00_2→MFR	
		1. (↓LDPR) (↓EQJPS) (↓EQINT) (↓TWJJS)→↑STMF0*/10M6A2	The memory field register is cleared.
		2. (↓LDPR) (↓EQJPS) (↓EQINT) (↓TWJJS)→↑STMF1*/10M6B2	
		3. (↑LDREG*) (↑CKINT)→↓IC03/10M6B2	
		4. (↓IC03) (↑STMF0)→↓MF0/10M7A2	
		5. (↓IC03) (↑STMF1)→↓MF1/10M7A3	
E.	PU1	MFR→ADDF0, ADDF1	
		1. (↑INTFF*) (↑PU1B)→↓RNTAD*/10M7B2	The current memory field (0) is selected.
		2. ↓RNTAD*→↓INTAD/10M7B2, ↑INTAD*/10M7B3	
		3. (See A, 1-4)	

Table 4-117. Interrupt Return, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP6	INTAR = AR? 1. ↓BP6*→↑BP61/10M5B4, ↑BP62/10M5B4 2. ↑BP61→C00-C06 = INT00-INT06 3. ↑BP62→C07-C11 = INT07-INT11 4. (A00-A11 = C00-C11) (↑INDFF) (↑FLMEM*)→↓EQADD*/10M6A1 5. ↓EQADD*→↑EQADD/10M6A2 6. (↑BP62) (↑PULL) (↑RFF*) (↓CPPU) (↑EQADD)→↑EQINT/10M6A3, ↓EQINT*/10M6A3 7. (↓EQINT*) (↑EQJPS*)→ ↑EQFF/10M6A3 8. ↑EQFF→↓EQFF*/10M6A3	EQFF* is set low to indicate that the address register contents equal the INT address register contents.
B.	BP7	0000_8→INT AR 1. ↓JMP*→↑JMP/10M7A2 2. (↑EQINT) (↑BP7B) (↑JMP)→ ↓IC01/10M6A3 3. (↓IC01) (↑PWRDY*)→ ↑CLINT/10M6A3 4. ↑CLINT→↓CLINT*/10M6A4 5. ↓CLINT*→INT00-INT11 = 0000_8	The INT address register is cleared.
C.	PU0	INT FR→MFR 1. ↑EQINT→STMF0 = INTMF0/10M6A2 2. ↑EQINT→STMF1 = INTMF1/10M6B3 3. (↑EQFF) (↑JMP) (↑PU0B)→ ↓IC03/10M6B2 4. (↓IC03) (↑PULL)→MF0 = STMF0, MF1 = STMF1/10M6A2, 106A3	The contents of the INT field register are loaded into the memory field register.
D.	BP1	↓EQINT, ↑EQINT* 1. ↑BP1→↓RFF*/10M7B2 2. ↓RFF*→↓EQINT, ↑EQINT*/10M6A3	Signal EQJPS is reset.

signal an indication that the current instruction is a memory reference instruction going indirect through the location whose address is the incremented trap address, and that this address is neither 0000_8 nor 7777_8. (To prevent erroneous assertion of the EQADD* signal during BP6 when the processor is not servicing an interrupt, the INT address register is cleared when the return to the interrupted program is performed. Returns through locations 0000_8 and 7777_8 are disallowed since referencing either of these locations results in a low FLMEM* signal.) If EQADD* is asserted during BP6, signal EQINT is asserted (set high).

If the instruction being executed when EQINT is asserted is a JMP instruction, the return to the interrupted program is being executed and the events as shown in Sections B and C of Table 4-117 occur. Signal CLINT* is asserted to clear the INT address register during BP7. Signal EQINT gates the contents of the INT register to the memory field register. The memory field register is clocked at PU0 by signal IC03 to load the contents of the INT field register, restoring the current memory field of the interrupted program.

Subsequently, at BP1, signal RFF* is asserted and resets signal EQINT low. This occurs whether a return has been executed or not. The simplified logic description of this function is shown in Section D of Table 4-117.

4.5.4.2.7 Cycle Steal. When a cycle steal occurs, the memory field which is to be accessed during the cycle steal is specified by the device which requested the cycle steal. None of the registers of the memory field control logic are affected. The change of memory field is only temporary.

A simplified description of the memory field control logic operation for a cycle steal is shown in Table 4-118. At PU0, the low CSFF*/10M7B1 signal results in generation of the high CSADD/10M7B2 signal that causes selection of the memory field indicated by signals SMF0 10M7B3 and SMF1*/10M7A3 by the output multiplexer. Signals SMF0* and SMF1* are generated by the peripheral device which requested the cycle steal. After CSFF*/10M7B1 has gone high again, at PU1, CSADD/10M7B2 goes low causing the current memory field to be restored.

4.5.4.2.8 Front Panel Memory Field Control. When the LOAD AR switch on the front panel is depressed (assuming the front panel controls are enabled) the memory field register is loaded with the two bits represented by the state of the memory field switches on the front panel.

The simplified logic description for this process is shown in Table 4-119. Depressing the LOAD AR switch on the front panel results in a high LDPR/10M7B1 signal. At PU0 while the LDPR signal is high, the memory field register is loaded by the contents of data signals SMF0/10M6A1 and SMF1/10M6B3, which are generated by the setting of the memory field switches. Signals MF0/10M7B3 and MF1/10M7B3 are then transferred to signals ADDF0/10M7B4 and ADDF1/10M7A1 to perform field selection as shown in Table 4-111.

4.5.4.2.9 LDREG (7720) Instruction. The LDREG instruction loads the JPS address register and the INT address register from the J and K registers of the central processor, respectively. Execution of this instruction has no direct effect on memory field specification. Memory field selection for its execution is done as in any other single word instruction, the current memory field being selected as shown in Table 4-111.

The fundamental operations for LDREG instruction execution are shown in Table 4-120. At BP3, the instruction is loaded into the instruction register of the processor and decoded to produce signals I/O/10M5A4, TWFF*/10M5A4, and HLT/10M5A4. These three signals are further decoded along with bits 7, 10 and 11 of the instruction by the memory field control logic to recognize that the instruction is an LDREG instruction. This results in signals OUT00 - OUT11 being selected as inputs to the JPS address register and the INT address register. At BP5, the contents of the J register of the central processor are presented as signals OUT00 - OUT11 and are loaded into the JPS address register. At BP6,

Table 4-118. Cycle Steal Memory Field Selection, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU0	SMF0→ADDF0, SMF1→ADDF1	
		1. ↓CSFF*→↑CSFF/10M7B1	Externally specified memory field is selected.
		2. ↓CPPU*→↑REGCLK/10M7A1	
		3. (↑REGCLK) (↑CSFF) (↑PU0B)→ ↓SLSAD*/10M7B2	
		4. ↓SLSAD*→↑CSADD/10M7B3, ↓CSADD*/10M7B2	
		5. (↓CSADD*) (↑EQMEM*) (↑INTAD*)→↑CFF* ↓CFF/10M7B3	
		6. (↑CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↓CFF)→SMF0 = ADDF0	
		7. (↑CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↓CFF)→SMF1 = ADDF1	
B.	PU1	MFR→ADDF0, ADDF1	
		1. (↑CSFF*) (↑PU1B)→↓IC10/10M7B2	Current memory field is again selected.
		2. ↓IC10→↓CSADD/10M7B2, ↑CSADD*/10M7B2	
		3. (↑CSADD*) (↑EQMEM*) (↑INTAD*)→ (↓CFF*/10M7B3)	
		4. ↓CFF*→↑CFF/10M7B3	
		5. (↓CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↑CFF)→ADDF0 = MF0	
		6. (↓CSADD) (↓EQMEM) (↓TWMEM) (↑TWMEM*) (↑CFF)→ADDF1 = MF0	

Table 4-119. Front Panel Memory Field Control, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU0	SMF0, SMF1→MFR	
		1. ↑LDPR→STMF0 = SMF0	The memory field register is loaded by the front panel memory field switches.
		2. ↑LDPR→STMF1 = SMF1	
		3. (↑LDPR) (↑PU0B)→↓IC03/10M6B2	
		4. ↓IC03→MF0 = STMF0,MF1 = STMF1	
B.	(Asynch.)	MFR→ADDF0, ADDF1	
		1. ↑LDPR→↓RFF*/10M7B2	The contents of the memory field register are gated to ADDF0 and ADDF1 to select the current memory field.
		2. ↓RFF*→↓EQMEM/10M7A2, ↑EQMEM*/10M7A2	
		3. ↓RFF*→↓TWMEM/10M7A2, ↑TWMEM*/10M7A2	
		4. (See Table 4-111, A. 1-4)	

Table 4-120. LDREG Instruction, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP3	OUT00-OUT11→PLD00-PLD11	
		1. (↑I/O) (↑TWFF*) (↑I07B) (↑HLT)→ ↓MFCI0*/10M5A4	Signals OUT00-OUT11 are gated through the input multiplexer.
		2. ↓MFCI0*→↑MFST1/10M5A4, ↑MFST2/10M5A4	
		3. (↑MFST1) (↓MFCI0)→PLD00-PLD01 = OUT00-OUT01	
		4. (↑MFST2) (↓MFCI0)→PLD02-PLD11 = OUT02-OUT11	
B.	BP5	JR→JPS AR	
		1. (See A. 1-2)	The JPS address register is loaded with the contents of the J register.
		2. ↓BP5*→↑BP52/10M5B4	
		3. ↑BP6*→↓BP62/10M5B4	
		4. ↑I11B*→↓I11/10M6B4	
		5. (↓I11) (↑MFST2)→↑DIN*/10M6B4	
		6. (↑MFST2) (↓BP62)→↑EXTK*/10M6B4	
		7. (↑MFST2) (↓I11B*) (↑I10*)→ ↓LDREG*/10M6B4	
		8. ↓LDREG*→↑LDREG/10M6B4	
		9. (↑BP52) (↑LDREG)→↓PEJPS*/10M7A4	
		10. ↓PEJPS*→↑PEJPS/10M7A4	
		11. (↑PEJPS) (↑RGCLK)→↓CKJPS*/10M7A4	
		12. (↑DIN*) (↑EXTK*)→OUT00-OUT11 = J REGISTER	
		13. PLD00-PLD11 = OUT00-OUT11 (See A. 1-4)	
		14. (↓PEJPS*) (↓CKJPS)→JPS00-JPS11 = PLD00-PLD11	
C.	BP6	KR→INT AR	
		1. (See A. 1-2)	The INT address register is loaded with the contents of the K register.
		2. ↑BP5*→↓BP52/10M5B4	
		3. ↓BP6*→↑BP62/10M5B4	
		4. (See B. 4-5)	
		5. (↑MFST2) (↑BP62)→↓EXTK*/10M6B4	
		6. (See B. 7-8)	
		7. (↑BP62) (↑LDREG)→↓REG6*/10M6A4	
		8. (↓REG6*) (↑ENINT*)→↑PEINT/10M6A4	
		9. ↑PEINT→↓PEINT*/10M6A4	
		10. (↑PEINT) (↑RGCLK)→↓CKINT*/10M6A4	
		11. (↓DIN*) (↓EXTK*)→OUT00-OUT11 = K REGISTER*	
		12. PLD00-PLD11 = OUT00-OUT11 (See A. 1-4)	
		13. (↓PEINT*) (↓CKINT*)→INT00-INT11 = PLD00-PLD11	

the contents of the K register are presented as signals OUT00 - OUT11 and are loaded into the INT address register. The high DIN*/10M6B4 signal tells the processor that data is to be output from it. EXTK* determines which register (J or K) is output as signals OUT00 - OUT11.

4.5.4.2.10 LDJK (7721) Instruction. The LDJK instruction loads the contents of the JPS address register and the INT address register, respectively, into the J and K registers of the central processor. Execution of this instruction has no direct effect on memory field specification. Memory field selection for its execution is done as in any other single word instruction, the current memory being selected as shown in Table 4-111.

The fundamental operations for the LDJK instruction are shown in Table 4-121. At BP3 the instruction is loaded into the instruction register of the processor and decoded to produce signals I/O/10M5A4, TWFF*/10M5A4, and HLT/10M5A4. These three signals are further decoded, along with bits 7, 10 and 11 of the instruction, by the memory field control logic to recognize that the instruction is an LDJK instruction. This results in signals C00 - C11 being presented to the processor via EXT00 - EXT11. At BP5, the con-

Table 4-121. LDJK Instruction, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP3	C00-C11→EXT00-EXT11 1. (↑I/O) (↑TWFF*) (↑I07B) (↑HLT)→ ↓MFCI0*/10M5A4 2. ↓MFCI0→↑MFST2 3. ↓I11B*→↑I11/10M6B4 4. (↑MFST2) (↑I11)→↓LDJK*/10M6B4 5. ↓LDJK*→↑LDJK/10M6B4 6. ↑LDJK→EXT00-EXT11 = C00-C11	The output of the comparator multiplexer is gated to EXT00-EXT11.
B.	BP5	JPS AR→JR 1. ↓BP5*→↑BP51/10M5A4; ↑BP52/10M5B4 2. ↑BP6*→↓BP61/10M5B4, ↓BP62/10M5B4 3. (See A. 1-3) 4. (↓BP62) (↑MFST2)→↑EXTK*/10M6B4 5. (↑I11) (↑MFST2)→↓DIN*/10M6B4 6. (↑BP51) (↑BP52) (↓BP61) (↓BP62)→ C00-C11 = JPS00-JPS11 7. EXT00-EXT11 = C00-C11 (See A. 4-6) 8. (↓DIN*) (↑EXTK*)→J REGISTER = EXT00-EXT11	The J register is loaded with the contents of the JPS address register.
C.	BP6	INT AR→KR 1. ↑BP5*→↓BP51/10M5A4, ↓BP52/10M5B4 2. ↓BP6*→↑BP61/10M5B4, ↑BP62/10M5B4 3. (See A. 1-3) 4. (See B. 4-5) 5. (↓BP51) (↓BP52) (↑BP61) (↑BP62)→ C00-C11 = INT00-INT11 6. EXT00-EXT11 = C00-C11 7. (↓DIN*) (↓EXTK*)→K REGISTER = EXT00-EXT11	The K register is loaded with the contents of the INT address register.

tents of the JPS address register are presented as signals C00 - C11 and then transferred to the J register. At BP6, the contents of the INT register are presented as signals C00 - C11 and are transferred to the K register. The low DIN*/10M6B4 signal tells the processor that data is to be input to it. EXTK*/10M6B4 determines which register (J or K) is to receive the input from signals EXT00 - EXT11.

4.5.4.2.11 RJIB (7722) Instruction. The RJIB instruction loads the JPS field register and the INT field register from the J register of the central processor. Since the information is loaded into the flip-flops that comprise the field registers asynchronously via a SET input, only logical 1's can be loaded into these registers. Therefore, to effectively transfer the control of the J register bits into the JPS and INT registers, these registers must have been previously reset.

The simplified logic description for RJIB instruction execution is shown in Table 4-122. At BP3, the instruction is loaded into the instruction register of the processor and decoded to produce signals I/O/10M5A4, TWFF*/10M5A4, and HLT/10M5A4. These three signals are further decoded along with bits 7, 10 and 11 of the instruction, by the memory field control logic to recognize that the instruction is an RJIB instruction. This results in signals OUT00 - OUT11 being presented as signals PLD00 - PLD11. At BP5, signals PLD02 - PLD05 are gated by the status register loading gates into the SET inputs of the JPS and INT registers.

The high DIN*/10M6B4 signal tells the processor that data is to be output from it via OUT00 - OUT11. The high EXTK*/10M6B4 signal tells the processor that the output data is to come from the J register.

Table 4-122. RJIB Instruction, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP3	OUT00-OUT11→PLD00-PLD11	
		1. (↑I/O/10M5A4) (↑TWFF*/10M5A4) (↑I07B/10M5A4) (↑HLT/10M5A4)→ ↓MFCI0*/10M5A4	Signals OUT00-OUT11 are gated through the input multiplexer.
		2. ↓MFCI0*/10M5A4→↑MFST1/10M5A4, ↑MFST2/10M5A4	
		3. (↑MFST1/10M5A1) (↓MFCI0*/10M5A1)→ PLD00-PLD01 = OUT00-OUT01	
		4. (↑MFST2/10M5A1) (↓MFCI0*/10M5A1)→ PLD02-PLD11 = OUT02-OUT11	
B.	BP5	PLD02-PLD03→JPS FR, PLD04-PLD05→INTFR	
		1. (See A. 1-2)	Logical ones from the input multiplexer are loaded into the INT and JPS field register.
		2. ↑I11B*→↓I11/10M6B4	
		3. (↓I11) (↑MFST2)→↑DIN*/10M6B4	
		4. (↓BP62) (↑MFST2)→↑EXTK*/10M6B4	
		5. (↑I10) (BP52) (↑RGCLK) (↑MFST2)→ ↓SFF*/10M6B1	
		6. ↓SFF*→↑SFF/10M6B1	
		7. ↑SFF→SJMF0 = PLD02, SJMF1 = PLD03 SIMF0 = PLD04, SIMF1 = PLD05	
		8. JPSMF0 = SJMF0, JPSMF1 = SJMF1 INTMF0 = SIMF0, INTMF1 = SIMF1	

4.6 I/O INTERFACE

The input/output (I/O) hardware provides the capability for intercommunication between the computer and devices peripheral to the computer. It is this capability that enables the computer to exercise control over mechanical and electronic devices in a computer-controlled system. It allows the computer to control and exchange information with high capacity information storage devices (disc drives, tape drives, etc.), thereby enormously expanding the virtual memory capacity of the computer. Most importantly it enables the exchange of information between the computer and devices (keyboards, displays, printers, etc.) that provide efficient communication between humans and machines. It also enables different processors to exchange information and to exercise direct control over one another in multiprocessor systems.

As is the case with many computers, the input/output hardware of the ND812 that is included as part of the processor is designed to be simple and extremely flexible. The input/output hardware has very few provisions to account for the peculiarities of particular peripheral devices. Thus, a great deal of the responsibility for the control of I/O operations rests with hardware controllers that intervene between the processor and each peripheral device. Generally, each different peripheral device needs a controller designed specifically for it. In some cases a group of peripheral devices shares a single controller when connected with the computer.

The present section is intended as a guide both to servicing the I/O hardware and to designing the hardware controllers necessary for interfacing peripheral devices with the computer.

4.6.1 MODES OF I/O OPERATION

The principal task of the I/O system is to synchronize the operation of the computer with the operation of the peripheral devices connected to the computer. The data transfer rate of the computer is many times that of other peripheral devices. An efficient I/O system has provisions so that both the fastest and the slowest peripheral devices are interfaced with the computer in such a way that the computer is not excessively burdened and each peripheral device receives sufficiently prompt service.

The multiplicity of modes of I/O operation that can be implemented through the ND812 makes possible an efficient I/O system. For the faster devices, or those that handle large contiguous blocks of data (disk drives, for example), the direct memory access (DMA) mode is provided. For slower devices (teletypes, for example) the program controlled (PIO) mode is provided.

The principal instrument of the PIO mode is the interrupt mechanism by which peripheral devices in need of service by the computer can interrupt program execution to obtain service. Polling and trap addressing mechanisms direct the computer to the proper service routine. The polling mechanism is geared principally for devices that do not interrupt often. The trap addressing mechanism provides for devices that interrupt often because it requires less computation by the computer for each interrupt. Also, trap addressing makes it possible to break up a long polling routine into a number of smaller ones by having groups of devices sharing common trap addresses.

Several factors should be considered in determining the mode of I/O operation to be used for a given peripheral device. In general, the faster, more efficient modes require more elaborate hardware controllers to interface the peripheral device with the computer. The slower modes, though requiring less elaborate interfacing, take their toll in computational requirements--frequent execution of polling and service routines can use up a lot of time.

An additional mode of I/O operation is the autoload mode which is active only when the computer is halted (not executing instructions). Its function is to allow a peripheral information storage device to load the memory without the benefit of program control from the computer. The autoload mechanism operates in essentially the same manner as the front panel and operates at the data transfer speed of the peripheral information storage device involved.

4.6.2 I/O BUS

The I/O bus is the common carrier of signals between the computer and all hardware peripheral to the computer. The I/O bus can be characterized as an 86-wire cable, along whose length the peripheral interface controllers are connected in parallel. The actual construction of the I/O bus is, however, more detailed than this.

Though the peripheral interfaces are connected in parallel along the I/O bus, it is not usually constructed as a single contiguous cable. Instead, the hardware units are chained together in a series (daisy-chained) with cable lengths between them. Thus, each unit along the chain has two connectors with their corresponding pins tied together; one connector to accept the incoming cable and the other to pass the bus signals on to the next unit. The connections of the circuitry of the unit to the I/O bus are made between the two connectors. The connectors used for the I/O bus cables have 86 pins. Of these 86 pins, 76 carry the logic signals of the I/O bus. A table showing the signals corresponding to each pin is shown at drawing location A1 of sheet 22 of the schematic.

Figure 4-93 shows a simplified schematic of the ends of the I/O bus. A schematic of representative signal wires of the I/O bus with a wire grounded at both ends between them is illustrated; one carries signals from the computer to the peripheral hardware, the other carries signals from the peripheral hardware to the computer. Each line of the bus is as a wired-and connection, i.e., is held high (logical 1, approximately 3.2 volts) unless pulled low by a logic output from one of the DTL line drivers from the computer or a peripheral interface. It is this feature that allows multiple I/O interface circuits to transmit signals to the computer over common bus lines, though not simultaneously, of course.

When digital logic is interconnected by relatively long wires as is the case for the I/O bus, certain undesirable transmission line effects become appreciable. The time

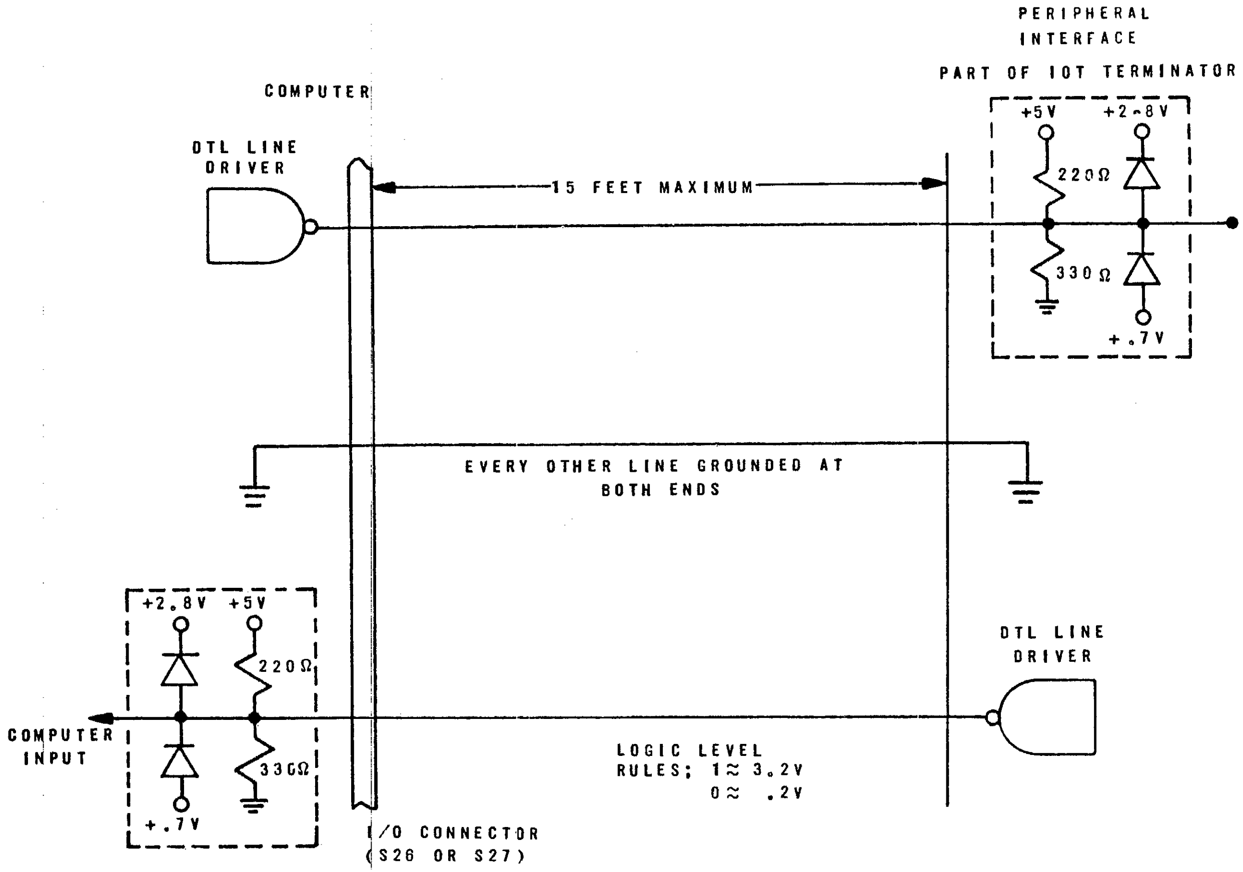


Figure 4-93. I/O Bus, Simplified Schematic Diagram

required to change a logic level across the length of a wire increases markedly with length, and is accounted for by slowing information transfer operations. Even with this provision, the total length of the I/O bus must be less than 15 feet. Long wires are susceptible to appreciable noise induced in them. Noise due to crosstalk between adjacent wires in the cable is reduced by alternating signal carrying wires with wires grounded at both ends.

Connection of the I/O bus with the computer is effected with one of two connectors--S26 or S27 (see Figure 4-94). The cable extending from the processor to the first peripheral device on the I/O bus may originate from either of these two connectors which have identical I/O signal connections. Though both connectors are identical, only one can be used for plugging in the cable of the I/O bus, the other is required to connect the IOR terminator board to the computer. The part number of the I/O cable supplied by Nuclear Data is 75-90XX where XX represents the length of the cable in feet.

The IOR terminator board (ND parg number 70-1813) provides termination for I/O bus lines carrying signals from peripheral devices to the computer. A schematic diagram of this board (Figure 4-95) is shown on sheet 24 of the system schematics.

The IOT terminator board (ND part number 70-1812) provides termination at the end of the I/O bus remote from the computer for lines carrying signals from the computer to the peripheral devices. The IOT board is plugged into the empty connector of the last device along the I/O bus. A schematic of the IOT board (Figure 4-95) is shown on sheet 25 of the system schematics.

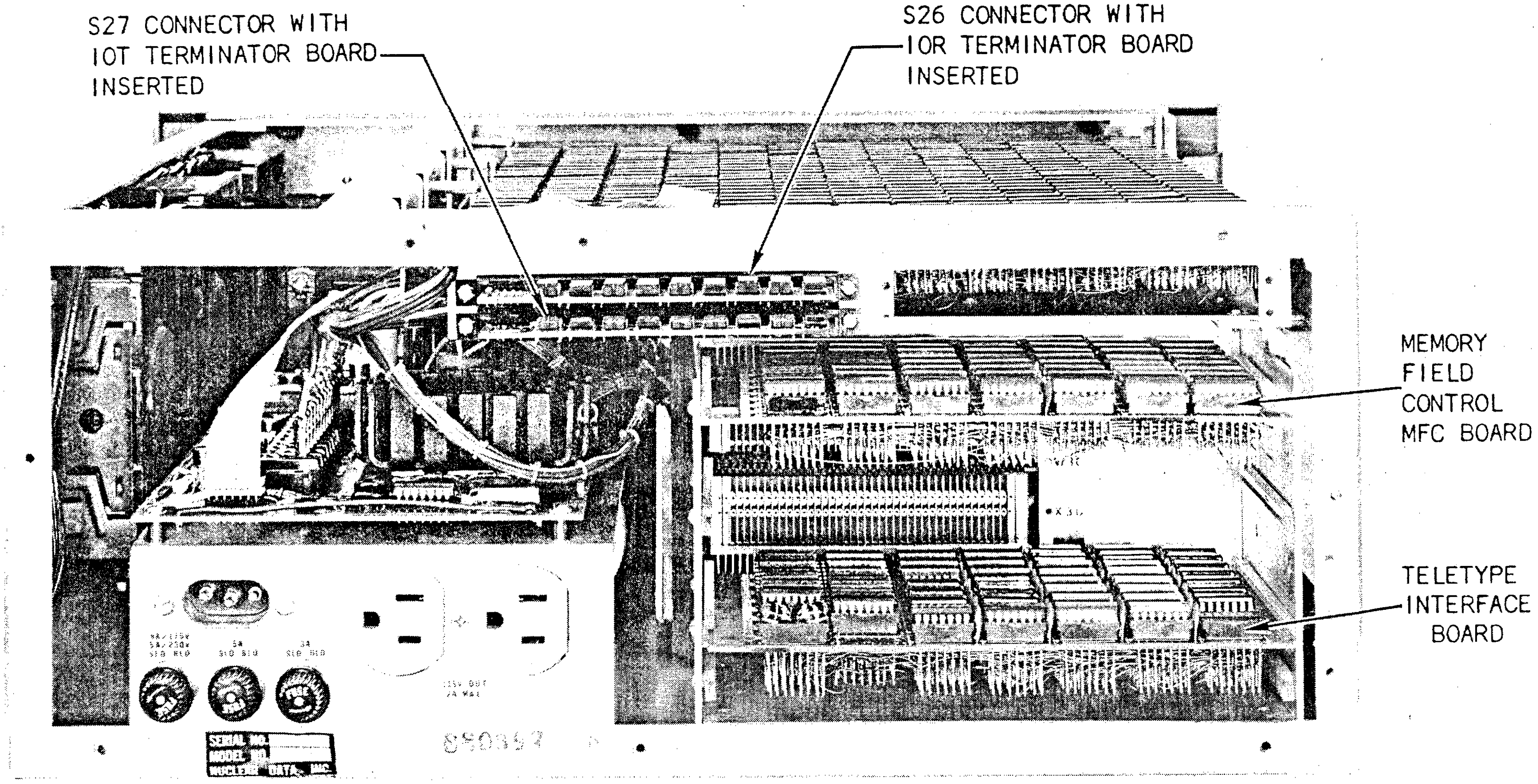
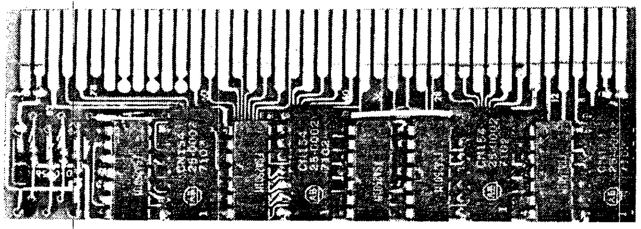
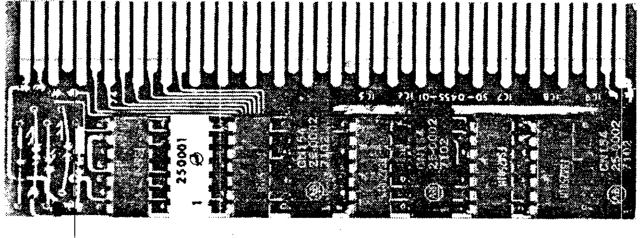


Figure 4-94. Rear View of ND812, Location of Connectors S26 and S27



IOT TERMINATOR BOARD
(ND 70-1813)



IOR TERMINATOR BOARD
(ND 70-1812)

Figure 4-95. IOR and IOT Terminator Boards

Tables 4-123 through 4-126 list all the signals of the I/O bus. Table 4-123 lists data signals from the computer to the bus, Table 4-124 lists data signals from the bus to the computer, Table 4-125 lists control signals from the computer to the bus, and Table 4-126 lists control signals from the bus to the computer.

Two additional transmission line problems are encountered by the terminators connected to the passive ends of the cable wires. The resistive part of these terminators reduces pulse reflections that arise from abrupt changes in impedance at the ends of the I/O bus. The resistors approximately match the cable impedance to eliminate the abrupt changes in impedance. The two diodes of each terminator are used to clamp the line level between approximately 0.2 volts and 3.2 volts. This is done to eliminate the voltages caused by inductive effects of long wires.

The problem of pulse reflections also requires branches off the I/O bus to be kept short (less than 1 foot) unless they are terminated.

4.6.3 I/O CONTROLLER

The basic ND812 contains a processor and memory consisting of two 4096 (4K) memory fields. The I/O controller consists of TTL and DTL integrated logic circuitry which permits the processor to interface with peripherals connected to the ND812 I/O bus. The

Table 4-123. Data Signals From Computer To Peripheral

Signal Name	S26 and S27 Pin Numbers	Function
OUT00*-OUT11*	8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30	Signals OUT00*-OUT11* are the data outputs for program-controlled I/O operation and for automatic testing of the computer. During a PIO data transfer, the twelve-bit contents of the J or K register are gated onto the processor bus (B00-B11). From there they are gated out onto signals OUT00*-OUT11* of the I/O bus. Signal OUT00* represents the most significant bit, OUT11* the least significant bit.
IOM00*-IOM11*	32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54	Signals IOM00*-IOM11* are the data signals for DMA output from the computer and for device addressing during PIO instruction execution. The contents of the memory data register are gated to IOM00*-IOM11* to generate these signals. Signal IOM00* represents the most significant bit, IOM11* the least significant bit.

high-capacity high-speed I/O capabilities of the ND812 controller permit it to operate a wide range of peripheral devices in addition to the standard ASR33 teletype keyboard/ printer and reader/punch. The peripheral options currently available with the ND812 processor include standard and high-density disk systems; magnetic-tape cassette systems, 9-track magnetic tape systems, high speed and low speed reader and punch (or both), teletype autoload, line printer, plotter, modem interface, and real-time clock.

Other peripherals such as magnetic drums, digital-to-analog and analog-to-digital converters, signal scanners and other programmable instrumentation hardware can be interfaced with the ND812 I/O bus. Up to 4,096 unique peripheral commands can be assigned to devices connected to the bus permitting designers the widest latitude in obtaining multiple functions in one unit of peripheral hardware. Input/Output connections for peripherals obtained as ND812 options require no special modification of computer hardware, and interfacing can be easily accomplished in the field. Table 4-127 is a list of currently available optional equipment that interfaces with the ND812 I/O bus.

Table 4-124. Data Signals From Peripherals To Computer

<table>
<tr><th>Signal Name</th><th>S26 and S27 Pin Numbers</th><th>Function</th></tr>
<tr><td>EMXS0*, EMXS1*, EMXS2*</td><td>70, 72, 74</td><td>These signals are register selection data. They are used during automatic testing of the processor. The three signals are decoded to gate a selected register through the MX multiplexer, adder, and onto the OUT* lines. Coding is as follows:
<table>
<tr><th>EMXS2*</th><th>EMXS1*</th><th>EMXS0*</th><th>Register</th></tr>
<tr><td>1</td><td>1</td><td>1</td><td>address register</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>utility gates</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>program counter</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>S register</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>J register</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>K register</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>R register</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>status register</td></tr>
</table>
In order for these signals to have an effect, the computer must be halted and signal ARMX* (see Table 4-126) must be pulled low to enable the MX multiplexer.</td></tr>
<tr><td>EXT00*-EXT11*</td><td>7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29</td><td>Signals EXT00*-EXT11* present twelve-bit data words from peripherals to the processor in PIO, DMA, and autoload data transfers. Trap addresses are transmitted to the processor via these signals during interrupt initiation. Signal</td></tr>
</table>

Table 4-124. Data Signals From Peripherals To Computer (Cont'd.)

Signal Name	S26 and S27 Pin Numbers	Function
		EXT00* represents the most significant bit, signal EXT11* the least significant bit.
SMF00*, SMF01*	57, 59	This pair of signals determines which memory field is referenced during autoloading and DMA mode data transfers. Coding is as follows:

SMF00*	SMF01*	Memory Field
1	1	0
1	0	1
0	1	2
0	0	3

These two signals are also affected by the front panel memory field switches when the computer is halted during an autoload transfer.

Table 4-125. Control Signals From Computer To Peripherals

Signal Name	S26 and S27 Pin Numbers	Function
EXGO*	6	Indicates when the computer is executing instructions; high when the computer is halted, low when the computer is running.
EXGO	60	Complement of EXGO*.
IONA*	78	When asserted (low) indicates that A level and highest level interrupts are enabled.
IONB*	80	When asserted (low) indicates B level and highest level interrupts are enabled.

Table 4-125. Control Signals From Computer To Peripherals (Cont'd.)

Signal Name	S26 and S27 Pin Numbers	Function
IONL*	82	When asserted (low) indicates that all interrupts are enabled.
XINTP*	35	When asserted (low) indicates that an interrupt has been permitted and is being initiated. Typically gates trap address onto EXT00*-EXT11*.
EXCSP*	39	When asserted (low) indicates that a cycle steal is being initiated. Typically, gates DMA address or to EXT00*-EXT11*.
CSMRZ*	45	When asserted (low) indicates that a current increment or decrement cycle steal has cleared the memory data register。
MRDY*	47	For DMA data transfer from the computer indicates that the data word is presented on signals IOM00*-IOM11*. Trailing edge indicates end of cycle steal cycle.
PCP0*	49	Peripheral control pulse to control timing of PIO mode operations. Can be used to cause a skip test. Duration: 0. 25 microseconds.
PCP1*	51	Peripheral control pulse to control timing of PIO mode operations. Can be used to cause a skip test or a data transfer. Duration: 0.5 microseconds.
PCP2*	53	Peripheral control pulse to control timing of PIO mode operations. Can be used to cause a skip test or a data transfer. Duration: 0.5 microseconds.

Table 4-125. Control Signals From Computer To Peripherals (Cont'd.)

Signal Name	S26 and S27 Pin Numbers	Function
PCP3*	55	Peripheral control pulse to control timing of PIO mode operations. Can be used for general control purposes only (i.e., not for skip test or data transfer). Duration: 0.5 microseconds.
IOCOM*	56	When asserted (low) signals execution of a PIO instruction and that IOM00*-IOM07* represents a device address. Asserted for the duration of all PCP time intervals.
XSTCL*	58	When asserted (low) signals initialization of processor registers for start-up of the computer. Occurs as a result of assertion of the START* signal.
PCPST*	62	Timing pulse associated with PCP pulses. Duration of this pulse is .25 microseconds starting at the beginning of each PCP time interval.

Table 4-126. Control Signals From Peripherals To Computer

Signal Name	S26 and S27 Pin Numbers	Function
DIN*	5	Indicates direction of data flow for PIO and DMA mode data transfers. High level indicates output from the computer, low level indicates input to the computer. During increment and decrement cycle steals, determines memory word is incremented or decremented. High level causes an increment operation, low level causes decrement operation.

Table 4-126. Control Signals From Peripherals To Computer (Cont'd.)

Signal Name	S26 and S27 Pin Numbers	Function
EXTK*	31	During PIO mode data transfers, selects the register affected. When EXTK* is high, the J register is affected. When EXTK* is low, the K register is affected.
EINTR*	33	When asserted (low) indicates that some peripheral device is requesting an interrupt.
CSR*	37	When asserted (low) indicates that a peripheral device operating in DMA mode is requesting a cycle steal.
ALTER*	41	When asserted (low) during a cycle steal, causes an increment or decrement (depending on DIN*) operation to be performed.
SDCS*	43	When asserted (low) during a cycle steal, causes another memory cycle to be stolen. The contents of the address register are incremented at the start of each additional cycle steal.
EXSKP*	61	If asserted (low) during pulses PCP0, PCP1, or PCP2, causes the program counter to be incremented, thereby skipping the next (single-word) instruction.
START*	63	When asserted (low), causes the computer registers to be initialized and causes the computer then to start executing instructions.
STOP*	65	When asserted (low) causes the computer to stop executing instructions after completion of the current instruction. Register status of the processor is unaffected.

Table 4-126. Control Signals From Peripherals To Computer (Cont'd.)

Signal Name	S26 and S27 Pin Numbers	Function
CONT*	67	When asserted (low) causes the computer to start executing instructions without changing the register status of the processor。
EEXAM*	69	When asserted (low) during testing (while the computer is halted) causes presentation of currently addressed memory location on OUT00*-OUT11*. Consecutive assertions of EEXAM* after the first assertion cause successive locations in memory to be presented on OUT00*-OUT11*.
XLDPR*	71	When asserted (low) during autoload, causes the twelve-bit word presented on signals EXT00*-EXT11* to be loaded into both the address register and the program counter. Also causes loading of the memory field register with the contents of signals SMF00* and SMF01*
XLDMR*	73	When asserted during autoloading, causes the twelve-bit word presented on signals EXT00*-EXT11* to be loaded into the addressed memory location。 Also causes the address register to be incremented。
SI*	75	During testing with the computer halted, each assertion (low) of this signal causes a single instruction to be executed.
ECHO*	79	Special signal for teletypes。 When asserted (low) causes input from the keyboard to be transferred through the computer to the printer of the teletype.

Table 4-126. Control Signals From Peripherals To Computer (Cont'd.)

Signal Name	S26 and S27 Pin Numbers	Function
ALODT*	64	When asserted (low) while computer is halted, causes the autoloading circuitry of the processor to be enabled. Also enables testing circuitry for automatic testing of the processor.
EXCLJ*	66	When asserted (low), asynchronously causes the J register to be cleared for initiation of checksum for autoloading.
TSTD*	68	When asserted (low) for the duration of an autoloading process, causes a checksum to be generated.
ARMX*	76	When asserted (low) during automatic testing of the processor when the processor is halted, causes the MX multiplexer to be enabled so that the registers of the processor can be examined externally.

Table 4-127. I/O Options Available For The ND812

PART NO.	DESCRIPTION
BULK STORAGE DEVICES	
STANDARD DENSITY CARTRIDGE DISK SYSTEM	
88-0441	Diablo Standard Density Cartridge Disk Interface with one each of the following:
84-0154	5V Disk Interface Power Supply
86-0216	Diablo Model 31 Standard Density Disk Drive Unit
88-0493	Diablo Disk Power Supply Unit

Table 4-127. I/O Options Available For The ND812 (Cont'd.)

86-0241	Diablo Connector Terminator
86-0223	Standard Density Disk Pack (12 Million Bits)
HIGH DENSITY CARTRIDGE DISK SYSTEM	
88-0472	Diablo High Density Cartridge Disk Interface with one each of the following:
84-0154	5V Disk Interface Power Supply
86-0227	Diablo Model 31 High Density Disk Drive Unit
88-0493	Diablo Disk Power Supply Unit
86-0241	Diablo Disk Terminator
86-0228	High Density Disk Pack (24 Million Bits)
FIXED-HEAD DISK SYSTEMS	
88-0440	EDP Fixed-Head Disk Interface with:
84-0154	5V Disk Interface Power Supply and one of the following;
86-0229	EDP Model 3008 Disk Memory System (0.4 Million Bits)
86-0230	EDP Model 3016 Disk Memory System (0.8 Million Bits)
86-0231	EDP Model 3032 Disk Memory System (1.6 Million Bits)
86-0210	EDP Model 3064 Disk Memory System (3.2 Million Bits)
86-0232	EDP Model 3120 Disk Memory System (6.0 Million Bits)

Table 4-127. I/O Options Available For The ND812 (Cont'd.)

MAGNETIC TAPE I/O DEVICES	
MAGNETIC TAPE CASSETTE SYSTEMS	
88-0423	Tape Cassettes System with,
84-0086	ND812 Interface and one of the following:
84-0093	Single Tape Cassette
84-0083	Dual Tape Cassette
84-0084	Triple Tape Cassette
7-TRACK COMPUTER COMPATIBLE MAGNETIC TAPE SYSTEM	
88-0456	7-Track Magnetic Tape Interline with one of the following combinations:
86-0208	PEC 6840-75 Transport, 7-Track, 45 IPS, 556/800 BPI and
86-0214	PEC FX4 9/7-YY/ZZ Magnetic Tape Formatter
86-0268	PEC 7540-72 Transport, 7-Track, 12.5 IPS, 200/556 BPI and
86-0269	PEC F847-6.95/2.5 Magnetic Tape Formatter
9-TRACK COMPUTER COMPATIBLE MAGNETIC TAPE SYSTEM	
88-0444	9-Track Magnetic Tape Interface with one of the following combinations:
86-0209	PEC 6850-9 Transport, 9-Track, 45 IPS, 800 BPI and
86-0214	PEC FX4 9/7-YY/ZZ Magnetic Tape Formatter
86-0255	PEC 7820-9 Transport, 9-Track, 12.5 IPS, 800 BPI and
86-0269	PEC F847-6.95/2.5 Magnetic Tape Formatter

Table 4-127. I/O Options Available For The ND812 (Cont'd.)

PAPER TAPE I/O DEVICES

HIGH SPEED READER OPTIONS

88-0482	High Speed Punch and Reader Interface with one of the following:
86-0188	Superior Electric Model TRP125-5 Photoelectric Tape Reader (125 Char/Sec)
86-0213	Dataterm Model HS300 Photoelectric Tape Reader (300 Char/Sec)
88-0492	Tally Punch and Reader Interface with:
88-0459	Tally Punch and Reader Drive and
86-0078	Tally Model 1504 Paper Tape Reader (60 Char/Sec)
88-0517	Remex Punch and Reader Interface with one of the following:
86-0266	Remex RPF1150B Fanfold Photoelectric Reader (200 Char/Sec)
86-0267	Remex RPS1150B Spooler Photoelectric Reader (200 Char/Sec)

HIGH SPEED PUNCH OPTIONS

88-0482	High Speed Punch and Reader Interface with one of the following:
88-0430	BRPE Punch Drive Unit with,
86-0062	Teletype Model BRPE11 High Speed Tape Punch (110 Char/Sec)
88-0492	Tally Punch and Reader Interface with:
88-0459	Tally Punch and Reader Drive and
86-0079	Tally Model 1505 Paper Tape Punch (60 Char/Sec)

Table 4-127. I/O Options Available For The ND812 (Cont'd.)

88-0517	Remex Punch and Reader Interface and one of the following;
86-0264	Remex RPF1075B Fanfold Mylar Punch (75 Char/Sec)
86-0265	Remex RPS1075B Spooler Mylar Punch (75 Char/Sec)
HIGH SPEED PUNCH AND READER OPTIONS	
88-0482	High Speed Punch and Reader Interface with one of the following combinations:
88-0430	BRPE Punch Drive Unit with,
86-0062	Teletype Model BRPE11 High Speed Tape Punch (110 Char/Sec) and,
86-0188	Superior Electric Model HS300 Photoelectric Tape Reader (125 Char/Sec)
88-0430	BRPE Punch Drive Unit with,
86-0062	Teletype Model BRPE11 High Speed Tape Punch (110 Char/Sec)
86-0213	Dataterm Model HS300 Photoelectric Tape Reader (300 Char/Sec)
88-0492	Tally Punch and Reader Interface with,
88-0459	Tally Punch and Reader Drive and one of the following combinations:
86-0079	Tally Model 1505 Paper Tape Punch (60 Char/Sec)
86-0078	Tally Model 1504 Paper Tape Reader (60 Char/Sec)
86-0101	Tally Model 1506 Paper Tape Punch and Reader Combination (60 Char/Sec)
88-0517	Remex Punch and Reader Interface with one of the following combinations:
86-0261	Remex RAF3075B Mylar Punch (75 Char/Sec) and Photoelectric Reader (300 Char/Sec) Fanfold Combination

Table 4-127. I/O Options Available For The ND812 (Cont'd.)

86-0263	Remex RAR3075B Mylar Punch (75 Char/Sec) and Photoelectric Reader (300 Char/Sec) Spooler Combination
HARD COPY I/O DEVICES	
LINE PRINTERS	
88-0483	Data Products Line Printer Interface with,
86-0212	Data Products Model 2410 Line Printer
88-0488	Centronics Line Printer Interface with,
86-0238	Centronics Model 101 Line Printer (165 Char/Sec)
PARALLEL PRINTER	
88-0487	Franklin Printer Interface with one of the following;
86-0048	Franklin Model 1220 Printer, 10 Column, 20 Lines/Sec
86-0050	Franklin Model 1230 Printer, 10 Column, 30 Lines/Sec
88-0051	Franklin Model 1240 Printer, 10 Column, 40 Lines/Sec
88-0486	Hewlett Packard Printer Interface with one of the following;
86-0115	Hewlett Packard Model 5050A Printer, 20 Lines/Sec
86-0225	Hewlett Packard Model 5055A Printer, 10 Lines/Sec
DIGITAL INCREMENTAL PLOTTER	
88-0485	Digital Incremental Plotter Interface with,
86-0226	Calcomp Digital Incremental Plotter

Table 4-127. I/O Options Available For The ND812 (Cont'd.)

MISCELLANEOUS ND812 OPTIONS

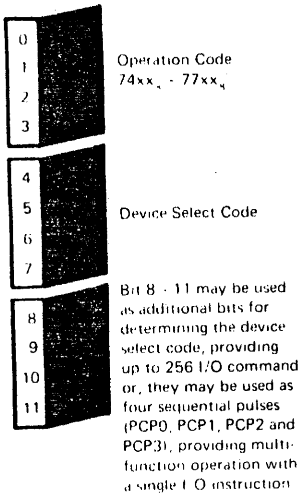
84-0098	Power Restart
88-0495	Real Time Clock
88-0446	I/O Card Cage Extension
84-0159	I/O Extension Card Module
84-0116	100 Foot Teletype Extension Cable
88-0481	Teletype Interface with Autoloading
88-0489	Mohawk Card Reader Interface
88-0470	Hewlett-Packard Card Reader Interface
88-0467	Data Phone Interface
88-0476	Computer-to-Computer Interface

4.6.3.1 PIO INSTRUCTIONS. Data transfers between a peripheral and the computer can take place in either direction; i.e., from the ND812 to the peripheral or from the peripheral to the ND812. In either case, the data transfer must be program controlled. Generally, each peripheral has several functions controllable through a set of instructions which address only the peripheral.

Table 4-128 is the I/O instruction word format in both the single-word and two-word forms. The single-word format permits sixteen discrete addresses and 16 discrete functions. In combination they permit 256 unique instructions. Various functions of the peripheral devices are obtained by assigning an address word and a function word from the address and function fields of either the single-word or two-word instructions. The functions are obtained when the ND812 provides as outputs, control pulses at different time intervals, depending on the state of bits 8 through 11 of the function field. These discrete pulses are called Peripheral Control Pulses. The two-word I/O instruction has a 7-bit address field permitting 256 possible unique addresses and, as with the single-word function field, a 4-bit field for generating peripheral control pulses; thus, in the two-word I/O format, 4,096 unique instructions are possible.

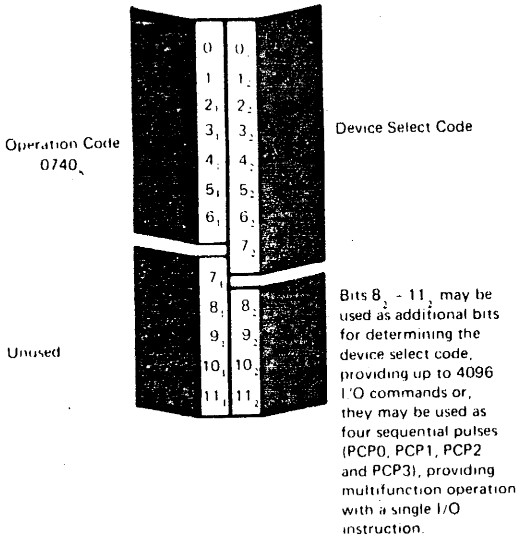
4.6.3.2 PRIORITY INTERRUPT INSTRUCTIONS. Because the I/O bus is a single common signal path between the ND812 processor and any peripheral connected to it, only one peripheral at a time may use it. Hence, in a situation where more than one peripheral may require service during a given moment of time, a queuing or waiting line must be established.

Table 4-128. I/O Instruction Formats



Bit 8 - 11 may be used as additional bits for determining the device select code, providing up to 256 I/O commands or, they may be used as four sequential pulses (PCP0, PCP1, PCP2 and PCP3), providing multifunction operation with a single I O instruction

This is the single-word input/output format. Its operation code begins at 74xx. Bits 4 through 7 permit discrete addressing and bits 8 through 11 permit discrete functions. In various combinations, they permit up to 256 unique instructions.



Bits 8_2 - 11_2 may be used as additional bits for determining the device select code, providing up to 4096 I/O commands or, they may be used as four sequential pulses (PCP0, PCP1, PCP2 and PCP3), providing multifunction operation with a single I/O instruction.

This is the two-word input/output format. Its operation code is 074X. Bits 7 through 11 of the first I/O word are unused. Bits 0 through 7 of the second word permit discrete addressing and bits 8 through 11 permit discrete functions. In various combinations, they permit up to 4,096 unique instructions.

This is done through the ND812 priority interrupt system which permits a programmer to specify the order of service in such a situation.

Table 4-129 lists the interrupt instruction subgroup together with the interrupt instruction bit pattern. Although these instructions appear in the ND812 operate group (operate code 1000_8), they do not perform arithmetic or logical operations on data in the J or K accumulators; hence, these instructions can be considered to be in a separate class. They deal only with the interrupt control logic of the ND812 I/O controller. When one

Table 4-129. Interrupt and Power Fail Instruction Subgroups, Bit Patterns

Octal			Bit Pattern											
Code	Mnemonic	Subgroup	0	1	2	3	4	5	6	7	8	9	10	11
1003	IOFF	INTERRUPT			1								1	1
1004	IONH				1							1		
1005	IONB				1							1		1
1006	IONA				1							1	1	
1007	IONN				1							1	1	1
1440	SKPL	POWERFAIL			1	1			1					
1500	PION				1	1		1						
1600	PIOF				1	1	1							

of these priority instructions is processed, the ND812 instruction decoder detects an operate instruction. However, because the content of the J or K registers is not affected (bits 4 or 5 of the instruction word not set), the data pattern established by bits 9, 10 and 11 of the instruction register is interpreted as priority interrupt data.

4.6.3.3 POWERFAIL INTERRUPT INSTRUCTIONS. Actually, there are two types of interrupt instructions, priority and powerfail. A powerfail interrupt instruction permits the ND812 to store the contents of its various registers in memory when the ND812 system loses a-c or d-c power. Powerfail interrupt instructions are also processed as operate instructions. When a power failure occurs, there is sufficient time to process a large number of instructions (about 1 millisecond) before faulty operation of the ND812 processor and memory occurs. Because of its magnetic properties, the memory is not affected by a power failure. Hence, a powerfail subroutine can be included in any program that will permit the content of any register in the ND812 to be saved.

4.6.3.4 TYPES OF PROGRAM CONTROLLED I/O INSTRUCTIONS. Generally, there are three types of I/O instructions for a given peripheral; data transfer, external skip, and function. Data transfer signals include data transfers in either direction between the ND812 and the peripheral devices connected to the I/O bus. External skips permit programmed interrogation of the peripheral to determine whether a data transfer or function has been completed. Function instructions control any of the numerous functions that a peripheral may perform. All transfers of data in the PIO mode are program-controlled. However, data transfers in the DMA mode are initialized through program-controlled instructions which specify the number of 12-bit data words to be transferred (called a block) and the address of the first word of the block of data to be transferred. Once the DMA transfer has begun, all the data in the entire block is transferred in successive single-word bytes until the total number of words on the block has been counted.

Program-controlled I/O (PIO) operation is effected by execution of any PIO instruction (Table 4-128). Execution of these instructions results in the presentation of device addresses and timing signals on the I/O bus. The device (or devices) affected, and the I/O operations that these signals result in, are determined by both the bit pattern

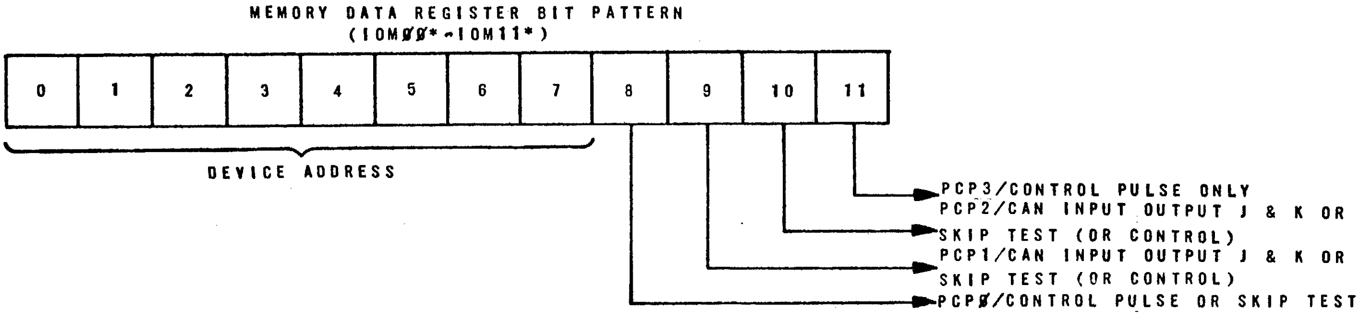

MEMORY DATA REGISTER BIT PATTERN
(IOMØØ*-IOM11*)
0
1
2
3
4
5
6
7
8
9
10
11
DEVICE ADDRESS
PCP3/CONTROL PULSE ONLY
PCP2/CAN INPUT OUTPUT J & K OR SKIP TEST (OR CONTROL)
PCP1/CAN INPUT OUTPUT J & K OR SKIP TEST (OR CONTROL)
PCPØ/CONTROL PULSE OR SKIP TEST

Figure 4-96. PIO Instruction Coding

of the PIO instruction (Figure 4-96) and the configurations of the hardware controllers connected to the I/O bus.

4.6.3.4.1 Data Transfer Instructions. The PIO and DMA logic are not separate electrical units in the ND812, but functionally, they perform all the operations required by the ND812 to communicate with any interconnected peripheral device. Logic circuitry that controls either program-controlled or DMA transfers between a peripheral and the ND812 processor is located on the main logic assembly of the processor (Figure 4-97). Hence, the I/O controller described in this Section includes all the logic required to implement communication between the ND812 and an interconnected peripheral.

The interrupt facility of the ND812 (4, Figure 4-97) forms an integral part of the PIO system. Interrupts enable peripheral devices to notify the ND812 processor of changes in their status. This ability makes it possible for the relatively slow peripheral devices to prepare for each I/O operation asynchronously with respect to programs running on the computer.

Most PIO operations are initiated by programs running on the computer. When an I/O operation is desired, a PIO instruction is executed to notify the appropriate peripheral device of this fact. Subsequently, the peripheral device generates an interrupt through the I/O control logic (2) to notify the ND812 that it should execute a service routine either to check the status of the peripheral device(s) involved, or to initiate an

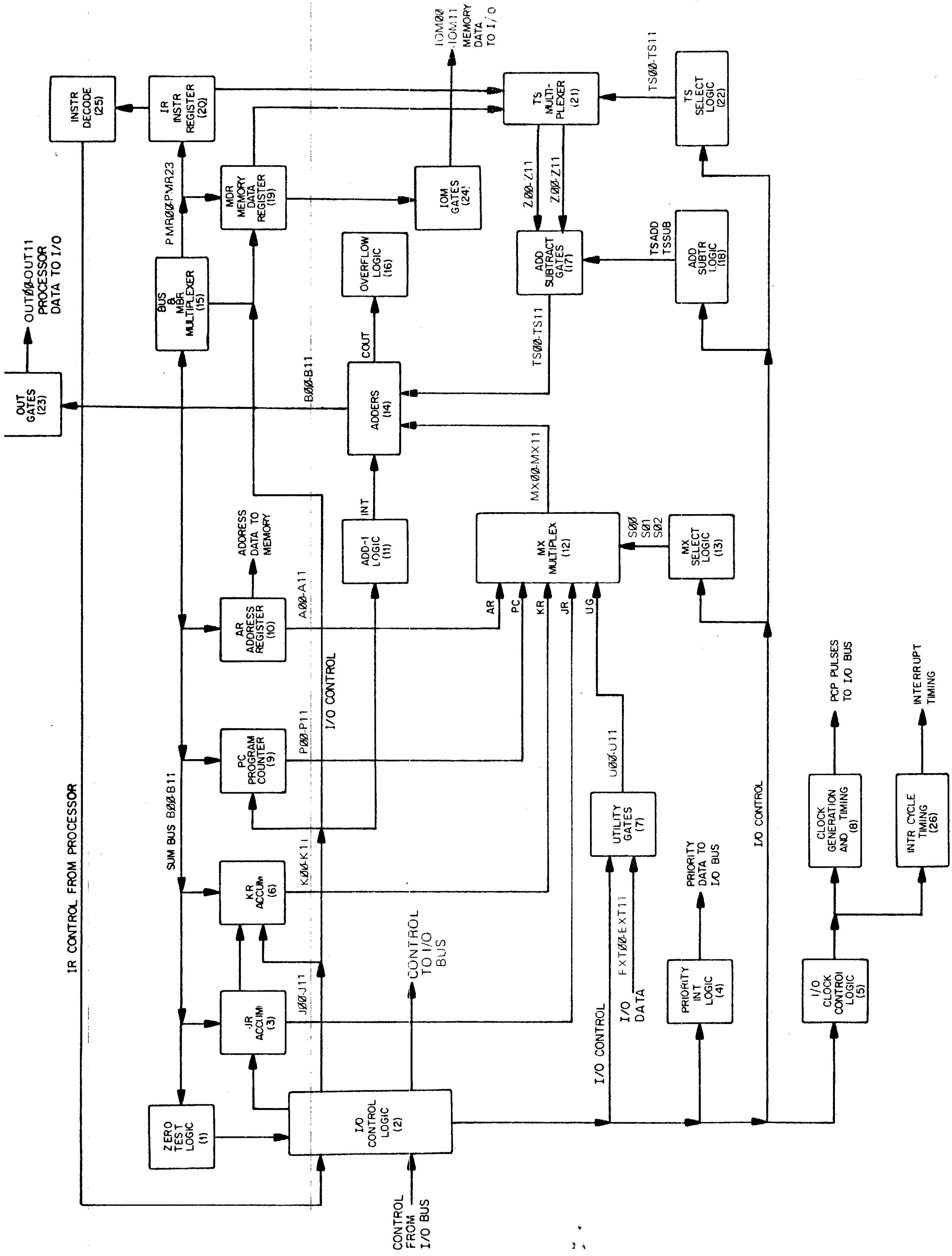


Figure 4-97 I/O Controller Block Diagram

I/O operation. Further interrupts may or may not be necessary during the performance of a particular I/O operation, depending on the priority level of the next device requesting service. The priority level logic (4) produces the I/O priority signals.

4.6.3.5 PIO INSTRUCTION CONTROL MECHANISM. When a PIO instruction is executed, three items of information are transmitted over the I/O bus. The IOCOM* signal is asserted to signify that an I/O instruction is being executed. Signals IOM00* - IOM11* present a twelve bit word through the IOM gates (24, Figure 4-97) that is used to address the appropriate device. One or more peripheral control pulses may be generated as signals PCP0* - PCP3* through the I/O clock control logic (5) and the clock generator and timing logic (8).

The device address and pattern of peripheral control pulses generated during execution of a PIO instruction is determined by the twelve-bit word contained in the memory data register (19) during the final basic phase of the I/O instruction execution. For a single-word PIO instruction, the instruction itself (including the operation code) is contained by the memory data register during the last (and only) basic phase of the machine cycle. For a two-word PIO instruction, the second word of the two-word instruction is contained in the memory data register during the last (second) basic phase of the machine cycle.

Figure 4-96 illustrates the way in which the bit pattern in the memory data register controls addressing and peripheral control pulse generation in a typical implementation. Though the entire twelve-bit word in the memory data register (19) is presented on the I/O bus (IOM00* - IOM11*) only the first eight bits (IOM00* - IOM07*) are used to represent device addresses. The last four bits determine which of the four peripheral control pulses (PCP0* - PCP3*) are generated for a particular PIO instruction. If the bit corresponding to a peripheral control pulse is set, that pulse is generated. Figure 4-96 also shows the possible operations generated by execution of each of the peripheral control pulses.

Figure 4-98 is a timing diagram showing the time relationships among the peripheral control pulses and other I/O signals. The final basic phase of any PIO instruction execution is extended fifty percent longer than a normal machine cycle. In particular, time intervals BP4 through BP7 are doubled in length from 0.25 microseconds to 0.5 microseconds each. From the start of period BP4 until the end of period BP7, signal IOCOM* is asserted (low) and the device address information is presented by bus signals IOM00* - IOM11*. Pulse PCP0* starts about midway through period BP4 and ends at the start of period BP5; it is only 0.25 microseconds long. Pulses PCP1* through PCP3* are concurrent with periods BP5 through BP7, respectively, and are each 0.5 microseconds long.

A typical eight-bit device address decoder is illustrated in Figure 4-99. This particular decoder will recognize the bit pattern 01000100_2 (210_8) contained in bits 0 through 7 of the memory data register. When this bit pattern is represented by signals IOM00* - IOM07*, and signal IOCOM* is asserted, the ADDRESS signal is asserted to indicate that the proper address is being presented to other peripheral logic. Signal IOCOM*

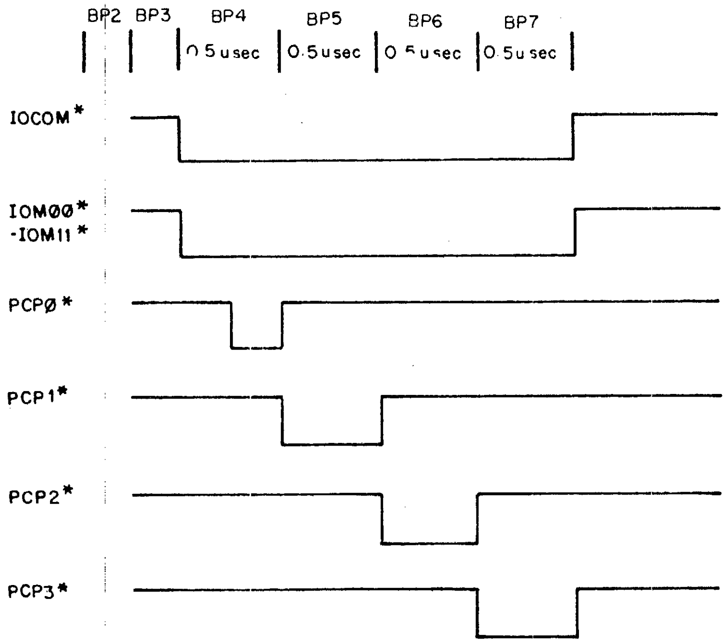
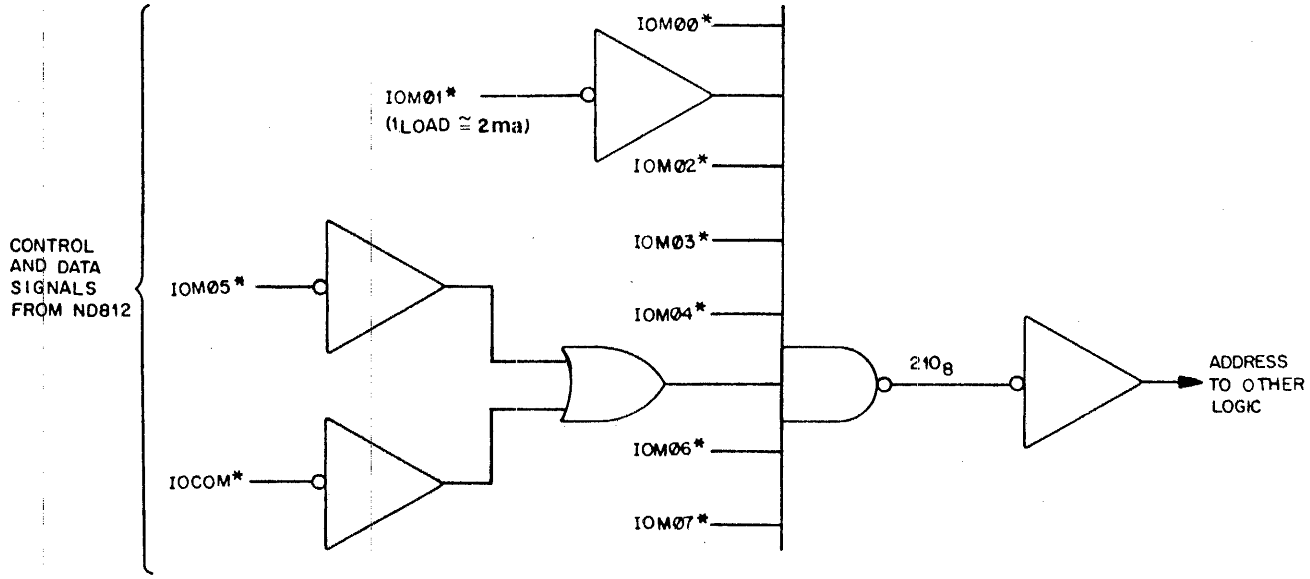


Figure 4-98. PCP Timing Diagram



ADDRESS 210 IS OBTAINED BY SETTING
BITS 1 AND 5 OF THE SECOND WORD OF
A TWO-WORD PIO INSTRUCTION

Figure 4-99. Typical Device Address Decoder

indicates to the decoder that a PIO instruction is being executed and that signals IOM00* - IOM07* represent a device address.

In implementing the address decoder (as well as the rest of a peripheral interface controller) three rules must be observed in loading the I/O bus.

1. Signals IOM00* - IOM11* can each sink approximately 30 milliamps.

2. Each interface should not load an output more than approximately 4 milliamps.

3. Each input should be capable of sinking approximately 20 milliamps.

4.6.3.6 POLLING MECHANISM. The polling mechanism is the means by which the ND812 determines the source of an interrupt request when the interrupting device does not provide a trap address that leads the processor directly to the proper service routine in a given program. This mechanism requires less interfacing hardware than a trap-addressing mechanism, but is less economical with respect to processor time.

Figure 4-100 shows a typical interrupt request and skip test interface circuit. When the peripheral device associated with this circuit is ready to request an interrupt, the interrupt flag (signal FLAG) is raised (set high). When the flag is up and the interface is permitted to generate interrupt requests (by an asserted IONA*, IONB*, or IONH* signal if this device is not of highest priority), signal EINTR* of the I/O bus is pulled low to request an interrupt from the processor.

After the interrupt has been permitted, the polling routine can be executed. During the polling routine a skip-test PIO instruction for this device is executed. Signal ADDRESS from a device address decoder goes high and two peripheral control pulses are generated. The first pulse (PCP0*, PCP1*, or PCP2*) causes a pulse of signal EXSKP*, which causes the processor to increment the program counter (9, Figure 4-97) thereby skipping the next instruction. When period BP7 occurs, the content of the program counter (9) is switched through the MX multiplexer (12) to the adder (14). Then, due to the presence of the EXSKP* signal at the I/O control logic (2), the add +1 logic (11) permits the adder (14) to be incremented. Thus, incremented data is returned to the program counter (9). The second pulse (PCP3*) causes the interrupt flag to be lowered (FLAG is reset). The instruction executed after the skip caused by EXSKP* leads the processor to the proper service routine. The requirement for the selection of PCP pulses is that the PCP pulse used for skip interrogation must be asserted before the PCP pulse used for the flag clear function.

In Figure 4-100, the skip interrogation signal is provided when peripheral control-pulse PCP0* is generated through the set state of bit 8 of the I/O instruction given as the typical example in Figure 4-96. Hence, the EXSKP* signal is produced when device address 210 is selected, the device flag is set, and peripheral control pulse PCP0* is present. The flag clearing signal, is obtained when this same device address is specified together with bit 11 of the I/O instruction; hence, the flag clear signal is generated whenever instruction 2101 is programmed.

4.6.3.7 TRAP ADDRESSING MECHANISM. The trap-addressing mechanism makes it possible for an interrupting peripheral device to lead the ND812 directly to the proper service routine or to a shortened polling routine, thereby reducing the amount of processor time required to service an interrupt. Implementation of this mechanism requires more interface circuitry than the polling mechanism.

Figure 4-101 shows a typical trap-address generator. When the ND812 recognizes the interrupt request generated by the interface containing the trap-address mechanism, the ND812 I/O control logic (2, Figure 4-97) pulls signal XINTP* low. Assertion of XINTP* in conjunction with the high FLAG signal (Figure 4-100) results in presentation of the trap-address on the signals EXT00* - EXT11* to the utility gates (7, Figure 4-97). Only those signals among EXT00* - EXT11* that correspond to logical 1's of the trap address are connected to line driver NAND gates and pulled low. Signal XINTP* (Figure 4-101) remains low (and the trap address is presented) for at least 0.5 microseconds ending at the start of period PU1 of the interrupt initiation cycle. After the interrupt initiation cycle, the processor begins execution of a service (or polling) routine starting two locations beyond the trap-address.

The interrupt is acknowledged by the ND812 when the current instruction is completed. The interrupt causes the I/O clock control logic (5, Figure 4-97) to inhibit normal clock generation and timing (8) and enable interrupt cycle timing (26). During interrupt cycle timing no basic- or execute-phase pulses are generated.

When period PU0 occurs, the trap address is gated from the EXT00* through EXT11* bus data lines through the utility gates (7, Figure 4-97), and through the MX multiplexer (12), to the adder (14). If there is no trap-address generator (Figure 4-100) in the peripheral, the unasserted EXT00* through EXT11* lines admit trap address 0000_8 into the adder (14, Figure 4-97); otherwise the specified address is admitted.

When the trap address is unspecified, a polling routine must be used to identify the source of the interrupt, because the interrupt can be generated by any peripheral whose priority has been asserted. However, if the trap address is specified, the service subroutine for the interrupting device can be entered directly. If more than one device has the same priority, however, they must have the same trap address, and a polling subroutine for that priority level must be used unless provisions are made to ensure that only one can raise its interrupt flag at a time. If this were allowed to occur, both would present their trap addresses simultaneously and the resulting trap address would be the result of ORing the two addresses together.

Next, the I/O control logic (2) causes the add +1 logic to increment the content of the adder (14). The reason for this incrementation is that an unspecified address names the first location in memory field 00. This same location is reserved for the autoindex feature, and therefore cannot be used. The incremented trap address is loaded into the address register (10). When period PU3 occurs, the content of the program counter (9, Figure 4-97) is switched through the MX multiplexer (12) to the adder (14), and from the adder, through the bus and memory buffer multiplexer (15) to the memory data register (19).

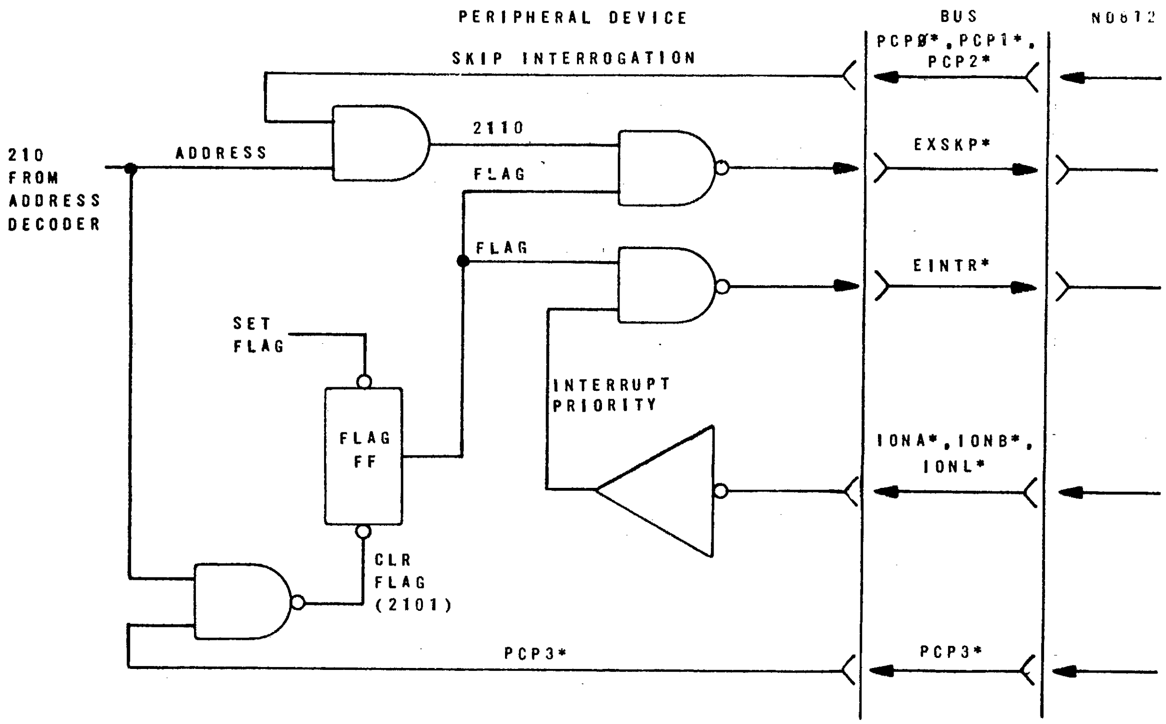


Figure 4-100. Typical Interface For Interrupt Request and Skip Test

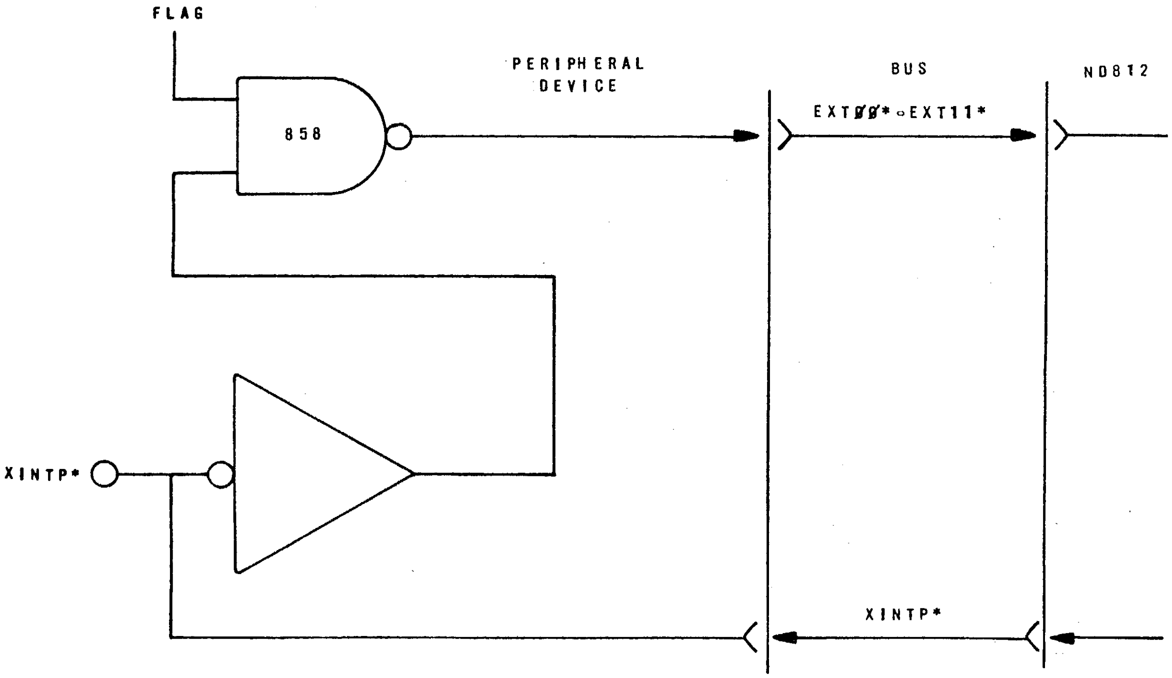


Figure 4-101. Trap Address Generator

This permits the content of the program counter (the next address in the program) to be written is the memory location specified by the trap address + 1 when period PU5 occurs.

When period PU6 occurs, the content of the address register (trap address + 1) is switched through the MX multiplexer (12) to the adder (14) and incremented again. This is done by the I/O control logic (2) through the add + 1 logic (11). This new address data (trap address +2) is then restored into the address register (10) and also admitted into the program counter (9). The new content of the program counter now specifies the address from which the next instruction is fetched, and the old content of the program counter is saved at the location specified by the trap address + 1 for the return to the interrupted program. This next instruction processed is the first instruction in the service routine for the peripheral requesting the interrupt.

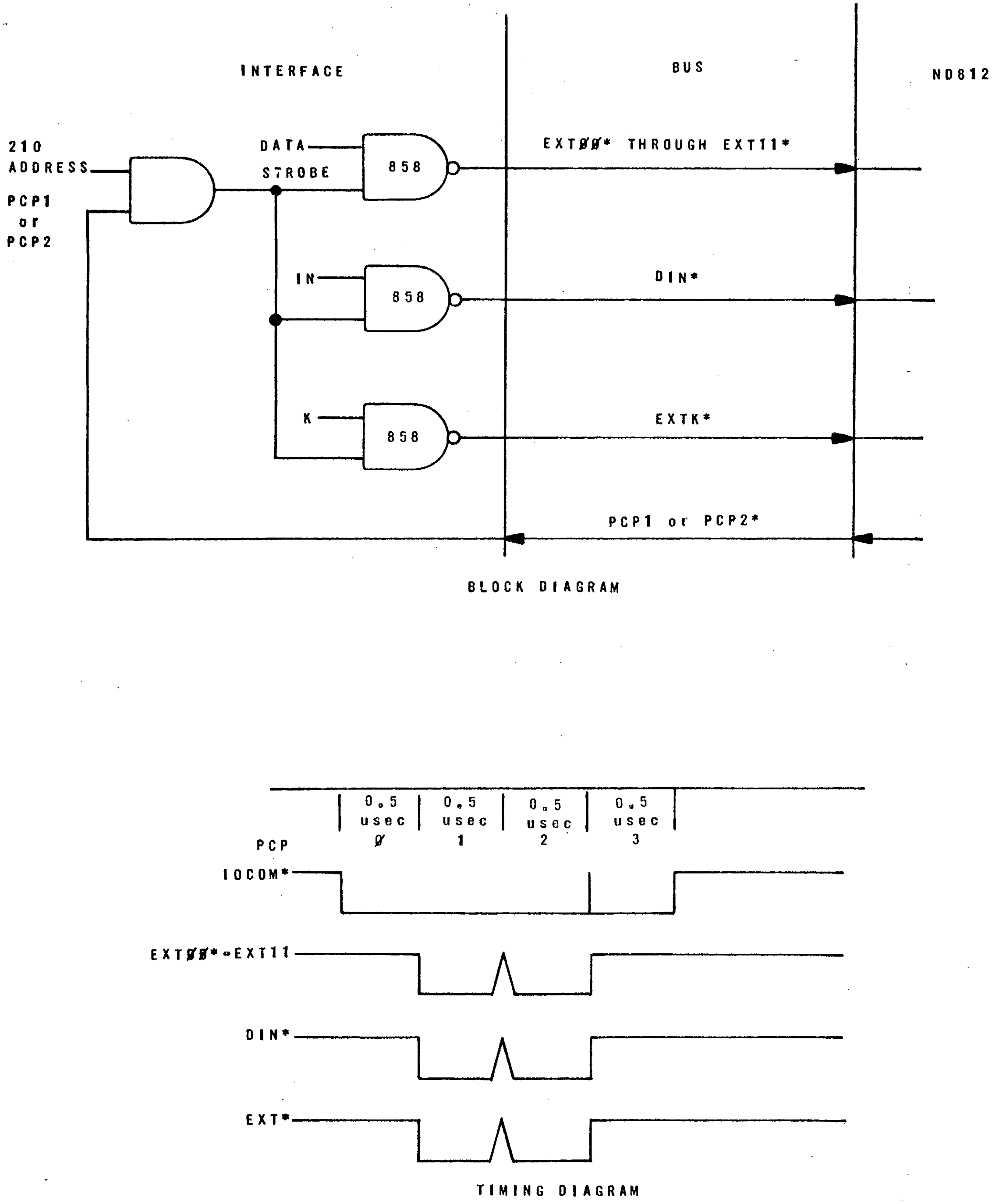
All interrupt service routines must end with an instruction to jump indirectly to the location specified by the content of the trap address + 1, the saved content of the program counter. In this manner, the program returns from the service subroutine to the next instruction which would have been processed, had the interrupt not occurred. Because of the priority system used in the ND812, it is possible to interrupt a service routine which is being processed as the result of a previous interrupt; the qualifying condition for the occurrence of interrupts within interrupted service routines (called nesting) is that the interrupting device have a higher priority and that the ND812 can discriminate its trap address. If interrupts are nested, the I/O control logic (2) merely permits new interrupt cycles.

At the conclusion of the interrupt cycle, a normal basic phase is entered and the new content of the program counter, the first address in the peripheral service subroutine, is admitted into the address register and the instruction located at that address (trap address +2) is processed.

4.6.3.8 PIO DATA TRANSFER MECHANISMS. In the PIO mode, all data transfers take place between the J and K registers (3 and 6, Figure 4-97) and peripheral devices. Up to two data transfers can be effected for each PIO instruction executed. The various combinations of data transfers possible for execution of a single PIO instruction are explained in detail later in paragraph 4.6.5.4 and are not discussed here.

Figure 4-102 is a typical interface circuit for transferring twelve-bit words from a peripheral device to the J or K register. These transfers are possible only when pulse PCP1* or PCP2* occurs. The PCP pulse is gated with the output of a device address decoder and generates a high level STROBE signal for the duration of the PCP pulse (0.5 microseconds). The STROBE pulse gates the twleve-bit word out on lines EXT00* - EXT11* of the I/O bus.

The STROBE pulse also enables the two control signals DIN* and EXTK*. The high IN signal causes DIN* to be pulled low to notify the ND812 I/O control logic (2, Figure 4-97) that data is being transferred to the J or K register (3 and 6, respectively). If signal K is high, signal EXTK* is pulled low to tell the I/O control logic (2) that the data is to be transferred to the K register (6). If signal K is low, signal EXTK* remains



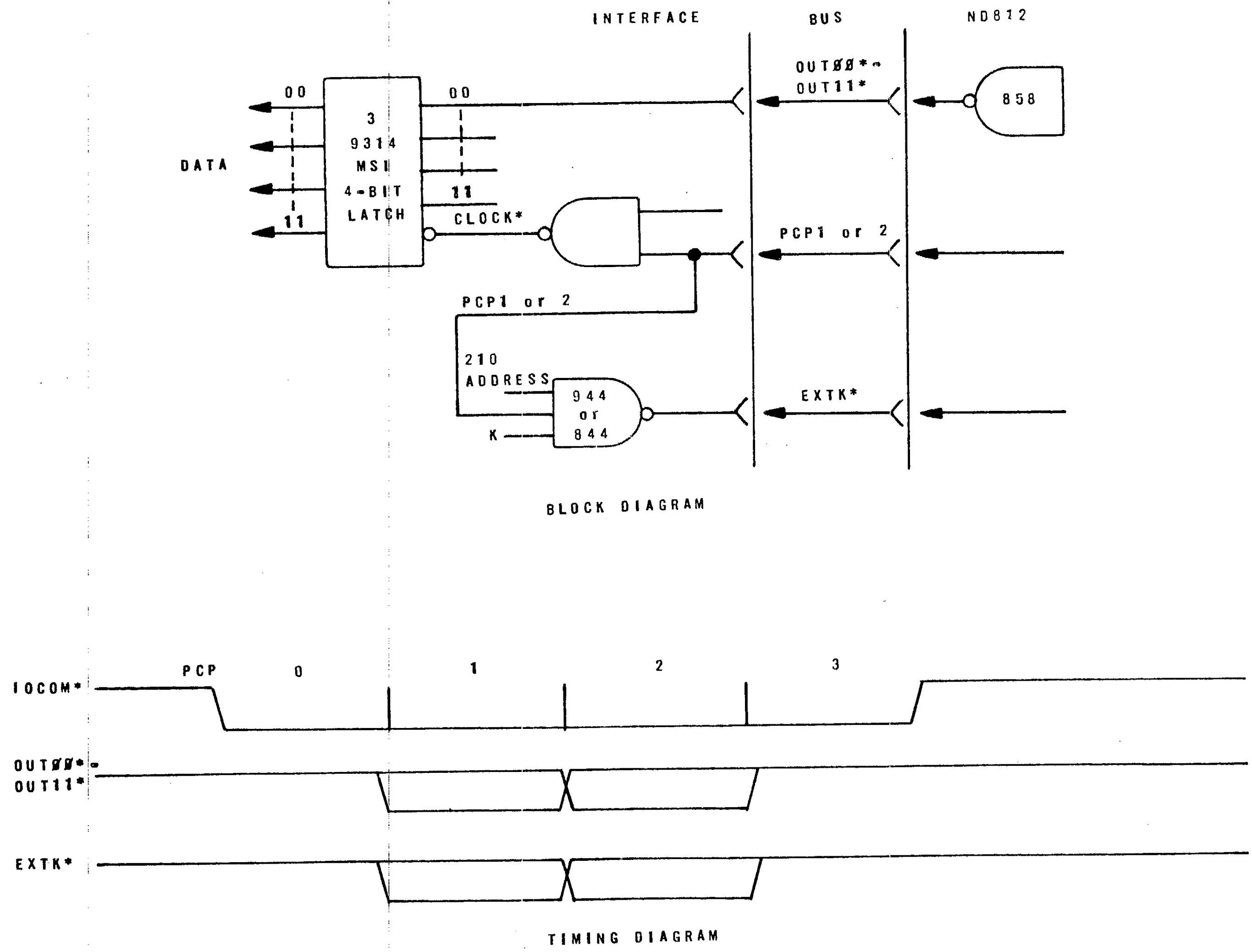
BLOCK DIAGRAM

TIMING DIAGRAM

Figure 4-102. Typical Interface For Input To J or K Accumulator

high and the data is automatically transferred to the J register.

Figure 4-103 shows a typical interface circuit for transferring twelve-bit words from the J or K register to a peripheral device. These transfers are possible only during a PCP1* or PCP2* pulse. The PCP pulse is gated from the clock generator and timing logic (8, Figure 4-97) with the output of a device address decoder (ADDRESS) to produce a low CLOCK* signal to three four-bit latches that form a buffer register. At the same time the PCP pulse and the high ADDRESS signal are gated with signal K. If K is high at this time, signal EXTK* is pulled low, causing the ND812 to present the contents of the K register on bus lines OUT00* - OUT11*. Otherwise, EXTK* remains high and the contents of the J register are presented instead. When the CLOCK* signal goes high again at the end of the PCP pulse, the twelve-bit word presented on signals OUT00* - OUT11* at that time (content of the J or K register) are preserved in the holding register.



BLOCK DIAGRAM

TIMING DIAGRAM

Figure 4-103. Typical Interface For Output From J or K Accumulator

4.6.3.8.1 Write Data Path. When the data transfer is from the ND812 to the storage element of the peripheral device, the data to be transferred is first loaded into either the J or K register (Figure 4-97, 3 and 6, respectively) and then switched through the MX multiplexer (12) to the adder (14), and through the out gates (23) onto the OUT8 signal lines of the I/O bus.

4.6.3.8.2 Read Data Path. When the data transfer is from the peripheral device to the ND812, the data to be transferred is admitted from the device storage element onto the EXT* lines of the I/O bus, through the utility gates (7), the MX multiplexer (12), and into the adder (14). From the adder, data from the peripheral is admitted into either the J register or the K register.

4.6.4 DIRECT MEMORY ACCESS OPERATION

The direct memory access (DMA) mode of operation allows a peripheral controller to manipulate directly the contents of memory, and to perform this manipulation asynchonously with respect to programs running on the computer. Use of this mode is intended for peripheral devices that transfer large blocks of data into or out of consecutive locations of memory. The DMA operations are carried out by generation of cycle steals which can occur between execution of successive instructions and interrupt program execution only for the duration of the cycle-steal memory cycle(s). Thus, after a program-controlled routine has initiated a DMA data transfer operation, no further program control over this operation need be effected.

When a DMA transfer of a block of data is initiated, the DMA interface controller must be given two items of information; the addres (including memory field) in memory of the first word of the block of memory involved, and the word length of the block. In addition, the block of storage space in the peripheral storage device involved must be specified.

The twelve-bit starting address is typically stored in a presetable counter in the DMA interface controller. This counter is loaded by an I/O instruction, and is incremented after each cycle steal transfer to generate the next address. The memory field bits are also stored.

The block length may be provided to the DMA controller in one of two different ways. The DMA controller may contain its own presetable counter to store the block length and decrement (or increment) its counter for each word transferred in order to be able to sense when the entire block has been transferred. Alternatively, the DMA controller may be provided with an address in the ND812 memory in which the word count (block size) is stored. In this case the DMA controller follows each DMA data-transfer cycle steal with a decrement (or increment) cycle-steal, during which the contents of the word count location are decremented (or incremented). The DMA controller is signalled when the word count reaches zero. With this method, the ND812 can easily monitor the progress of the DMA transfer by examining the contents of the word count location.

To provide for peripheral devices that need to transfer more than a single twelve-bit word at a time, multiple cycle steals can be generated. After a cycle steal is initiated, the DMA controller can steal further consecutive cycles in two ways; by holding the cycle steal request flag (CSR*) low or by pulling signal SDCS* low. As long as either of these two signals is held low, consecutive cycles are stolen. If the cycle steals are extended by signal SDCS*, the content of the address register in the ND812 is automatically incremented at the start of each new cycle.

4.6.4.1 DMA DATA TRANSFER MECHANISM. A typical DMA data transfer interface controller circuit is shown in Figure 4-104. When the DMA device is ready for a data transfer, signal CSRF is set high, causing the cycle steal request flag CSR* (waveform A, Figure 4-105) to the processor to be pulled low. When the processor has completed execution of its current instruction it initiates the cycle steal and pulls signal EXCSP* (waveform B) low to notify the DMA controller.

Signal EXCSP* is low for at least 0.5 microseconds, going high at the start of period PU1 of the current cycle-steal cycle. While asserted, signal EXCSP* (Figure 4-104) lowers the cycle-steal request flag, sets signal DMAON high, and gates the twelve-bit start address of the location in memory to be affected out onto signals EXT00* through EXT11*. This address is loaded into the address register (10, Figure 4-97) of the processor during period PU0, in a manner similar to that previously described for the Trap Addressing Mechanism.

If the data transfer is to the memory, when signal EXCSP* (waveform B, Figure 4-105) goes high at the beginning of period PU1, the address on bus lines EXT00* through EXT11* is replaced by a twelve-bit data word as follows: high level INP (Figure 4-104), EXCSP*, and DMAON signals gate the DATA signals out onto lines EXT00* through EXT11*. The high INP and DMAON signals combine to pull signal DIN* low to notify the ND812 that an input operation is being performed. Signal DMAON also gates the two memory field specification bits (FIELD BIT Ø and FIELD BIT 1) out to the processor on bus lines SMF0* and SMF1* to define the memory field that is the destination of the data word.

From the beginning of period PU5 until the end of period PU7, signal MRDY* (waveform F) is pulled low by the processor. Its falling edge triggers the pulse generator connected to the reset input of the DMAON flip-flop (Figure 4-104), which primes the input AND gate. When signal MRDY* goes high again, the resultant pulse resets DMAON low, ending presentation of the data word and the asserted DIN* signal.

If the data transfer is to be from the memory, when signal EXCSP* goes high at period PU1 (waveform B), address signals EXT00* through EXT11* are disabled (all set high), signal DMAON gates the memory field specification bits to the processor, and from the beginning of period PU5 until the end of period PU7, the data word fetched from memory is presented on bus lines IOM00* through IOM11*. A twelve-bit buffer register made up of three four-bit latches receives this data word. The trailing edge of signal MRDY* at the end of period PU7 triggers the pulse generator connected to the clock inputs of the

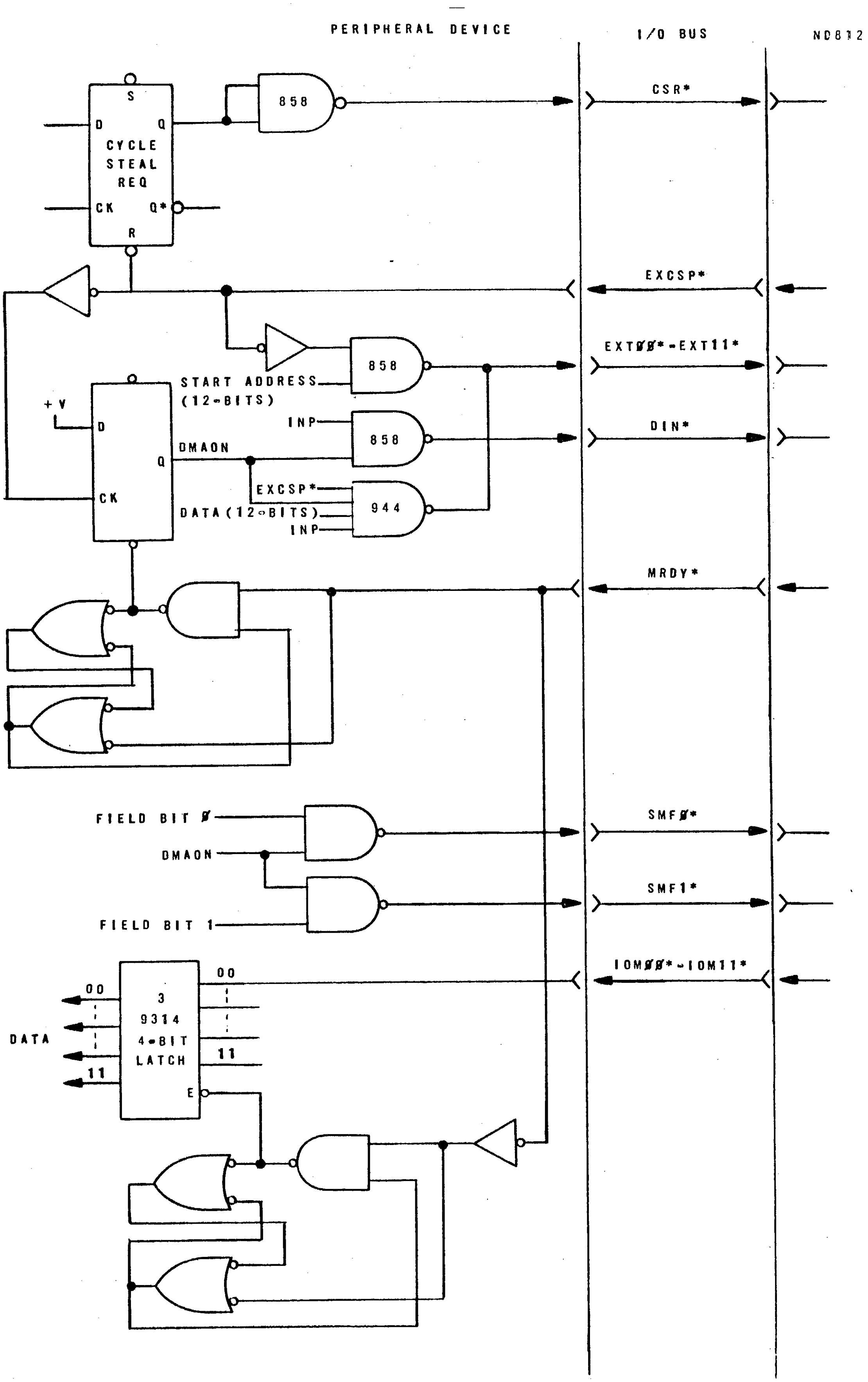


Figure 4-104. Typical DMA Data Transfer Interface

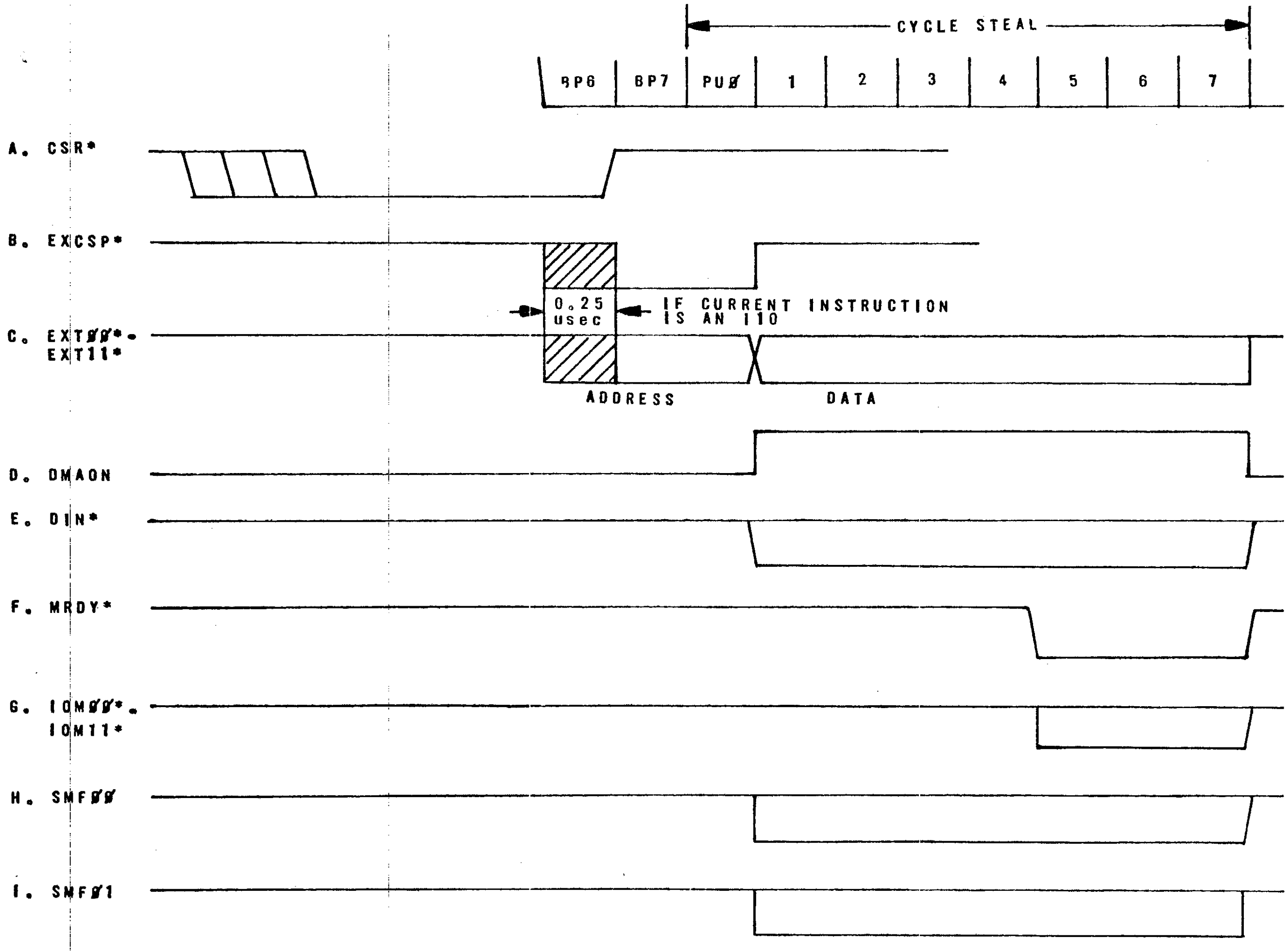


Figure 4-105. Typical DMA Timing Diagram

holding register. The resulting pulse causes the register to load the data word from bus lines IOM00* through IOM11*.

4.6.4.2 DMA INCREMENT/DECREMENT MECHANISM. A typical DMA increment-decrement interface circuit is shown in Figure 4-106. The operations of requesting the cycle steal, setting DMAON, and transferring the address are the same for this interface as for the DMA data transfer interface (see paragraph 4.6.4.1). The increment-decrement interface exercises its control over the processor through two signals on the I/O bus; signal ALTER*, when low, notifies the processor that the current cycle steal is an increment or decrement cycle steal. Signal DIN* determines whether an increment or decrement operation is performed.

The high MODIFY and DMAON signals combine to generate a low

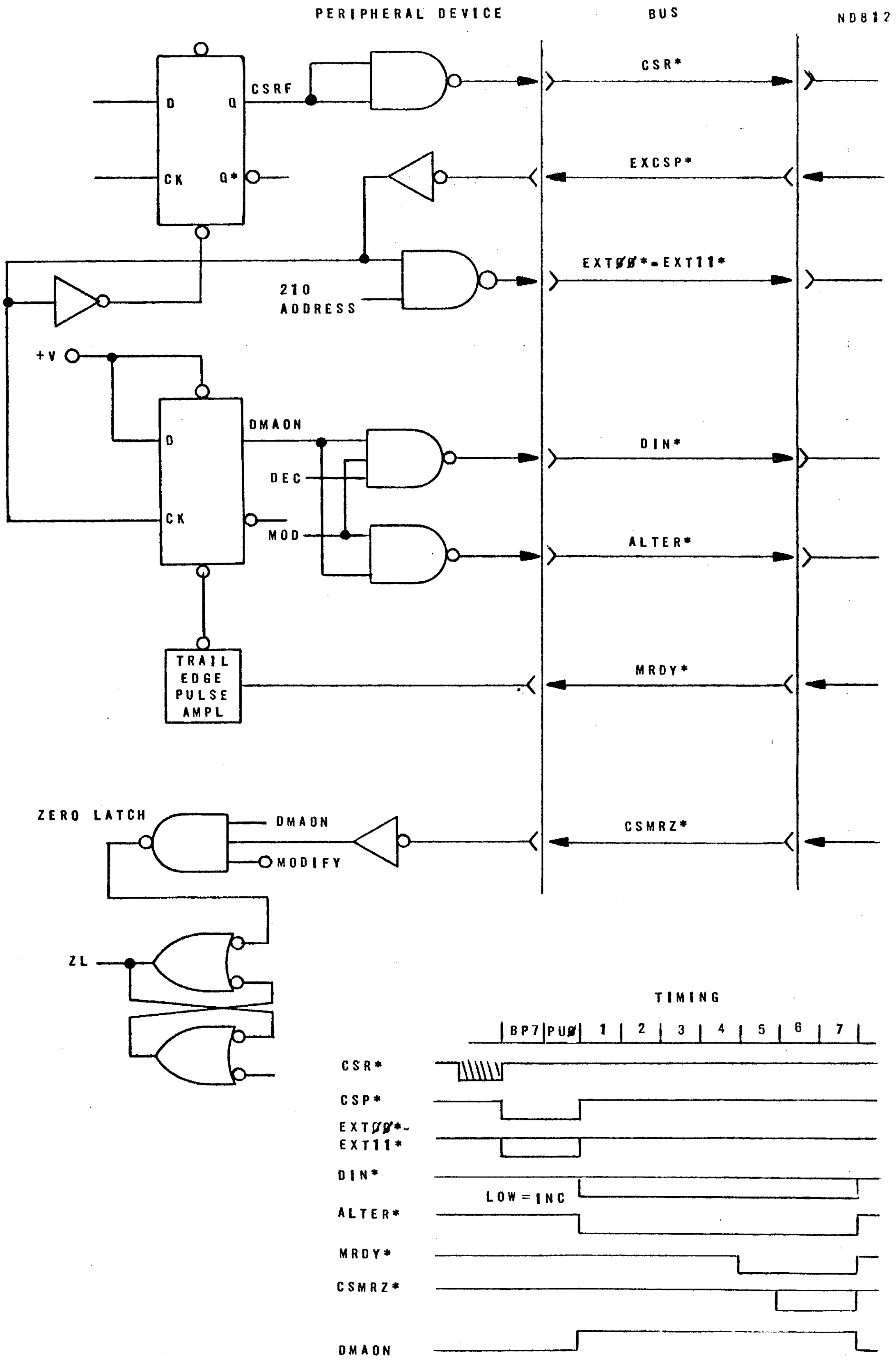


Figure 4-106. Typical DMA Increment/Decrement Interface

ALTER* signal. If signal DECREMENT* is high, signal DIN* is pulled low and an increment operation results. If signal DECREMENT* is low, signal DIN* remains high and a decrement operation is performed. The memory data register is incremented or decremented during period PU4. If the operation results in setting the contents of the memory data register to 0000_8, a low-signal CSMRZ* is generated by the processor to the DMA interface during periods PU6 and PU7. If CSMRZ goes low, the zero latch in the interface is set (signal ZL is set high). At the end of PU7, the trailing edge of the inverted MRDY* pulse causes DMAON to be reset. Thus, there are actually two alternatives of DMA processing, one requiring a single cycle-steal and the other, because of the increment/decrement requirement, two cycle-steal cycles for the transfer of a single twelve-bit data word either from or to memory. The first choice permits the highest transfer rate at the expense of address specifying and block-length counting hardware. The second choice permits use of ND812 registers for addressing and counting at the expense of speed.

4.6.5 DETAILED THEORY OF I/O OPERATION

This paragraph provides detailed descriptions of the I/O controller operation. These descriptions include modification of machine-cycle timing either by the initiation of an interrupt or a cycle-steal request from the peripheral. Also included are detailed descriptions of processing a program-controlled instruction, priority instructions, and increment/decrement cycles and multiple cycle steals.

The I/O controller descriptions do not contain block diagrams because data paths have already been described in paragraphs 4.6 through 4.6.4 and Figure 4-97. However, each description has a timing diagram which shows the timing relationships between the end of the current or interrupted phase, the interrupt or cycle-steal phase, and the beginning of the ensuing phase together with tabulations of the Fundamental Operations.

These tables should be used in trouble analysis because they are useful in identifying the various registers, multiplexers, and accumulators used in a given operation. They also identify the control logic which is used to obtain the hardware process.

4.6.5.1 INTERRUPT INITIATION. Interrupts enable a peripheral device requiring service by the computer to break into the normal execution of programs so that tha computer can turn its attention temporarily to providing that service. Interrupt initiation involves transferring control from a currently running program to a service routine, and making provision for returning to the interrupted program.

Interrupts can only occur if they have been previously enabled by execution of one of the interrupt enabling instructions (IONH, IONA, IONB, IONN). Any of these four instructions enables the computer to accept an interrupt request. The last three (IONA, IONB, IONN) enable various peripherals to generate interrupt requests as previously described in paragraph 4.6. through 4.6.4.

The interface designer can establish a hardwired priority level for any peripheral connected to the ND812 I/O bus. Actually there are four priority levels which the peripheral may use. These are: (1) highest, (2) special A, (3) special B, and (4) lowest. All these priority levels can be programmed.

The highest priority level (IONH) simply permits an external interrupting source to interrupt the program when the current instruction is completed. There is no actual bus wire named IONH which carries such a signal. Hence, when a device carries the highest interrupt priority, the interrupt request, which is issued from the device, is not priority-assignment dependent, so that in the case of Figure 4-100, for example, signal EINTR* is only dependent on the IONH interrupt instruction and request (FLAG), not the instruction, the request and the priority assignment.

The two special priority assignments, and the lowest, IONA, IONB, and IONN, respectively, are programmed (software assigned) and hardware controlled. When corresponding instructions are programmed, any one or all of these priority levels may be enabled. However, the device cannot interrupt the ND812 program until its priority level is enabled by the proper instruction. The simplified logic description for these events is shown in reference A of Table 4-130 and associated timing is shown by waveforms A through L, W, and X, of Figure 4-107.

The low XINTP* signal tells the interrupting peripheral device that the interrupt is permitted and gates the trap address (if there is one) out of the peripheral device as signals EXT00* through EXT11*. If the device supplies no trap address, signals EXT00* through EXT11* remain high and the trap address is 0000_8. At the end of period PU7, the data word represented by signals EXT00* through EXT11* is presented to the adder (14, Figure 4-97), and is gated through the utility gates through the MX multiplexer. The simplified logic description for these events is shown in reference B of Table 4-130 and associated timing is shown by waveforms A, E, L, M, and N of Figure 4-107.

Also, during period PU1, the trap address is incremented and loaded into the address register. The trap address is incremented by presenting 0000_8 from the add/subtract gates along with the trap address from the MX multiplexer to the adders and generating a high CIN signal which enters the carry input of the adder. When these signals are thus presented, the adder presents the incremented trap address as bus signals B00 through B11, and the address register is parallel enabled and clocked to store the incremented trap address. The simplified logic description for these events are listed in references C and D of Table 4-130 and associated timing is illustrated by waveforms A, E, O, P, Q, and R of Figure 4-107.

At period PU1, signal INTP is reset as listed in reference E of Table 4-130 and waveforms A, F, and S of Figure 4-107.

Whenever an interrupt occurs, all interrupts are automatically disabled and remain disabled until one of the four enabling instructions (IONH, IONA, IONB, IONN) is again executed. If this were not done, an interrupt could be initiated during an interrupt which might make it impossible to return to the previously interrupted program. One of the four enabling instructions should be included at the end of every interrupt service routine, just before the return to the interrupted program. Leaving it out would result in a completely

Table 4-130. Interrupt Initiation, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU7	SET ↑INTFF, ↓XINTP*, ↑INTCSF, ↓BLKE*, ↓BLKB* 1. ↓EINTR* + ↑INTR*→↑INTR/4B3 2. (↑DONE) (↑CSR*) (↑PU7B) (↑SPPU*→ ↓CKINT*/4B3→↑CLKINT/4B3 3. ↓CKINT* + ↑SPPU*→↑SPPU/4B3 4. (↑SPPU) (↑PU7B)→↓SPPU*/4B3 5. (↓SPPU*) (↑DONE) (↑CSR*) (↑PU7B)→ ↑CKINT*/4B3 6. ↑CKINT*→↓CLKINT/4B3 7. (↑CSFF*) (↓CLKINT)→↑INTCK*/4B3 8. (↑INTCK*) (↑ENINT) (↑INTR) (↑STPU1)→ ↑INTP/4B4, ↓INTP*/4B4 9. ↓INTP*→↑INTFF/4B4, ↓INTFF*/4B4 10. (↑INTFF) (↑INTP)→↓XINTP*/4A4 11. ↓INTFF* + ↑CSFF*→↑INTCSF/4B4 12. (↑INTCSF) (↑PU7) (↑CPPU*) (↑OSCPUL)→ ↑BLKBEP/5B2 13. ↑BLKBEP→↓BLKE*/5B3 14. ↑BLKBEP→↓BLKB*/5B3	Signal INTFF is set to indicate an interrupt initiation cycle; as a result, signals XINTP*, INTCSF, BLKE*, and BLKB* are also asserted. XINTP* is transmitted on the I/O bus. BLKE* and BLKB* inhibit the execute and basic phases for the duration of the cycle.
B.	PU0	EXT00-EXT11→MX00-MX11 1. ↓XINTP*→EXT00-EXT11 = trap address 2. ↓INTFF* + ↑CSFF*→↑INTCSP/8A3 3. (↑INTCSP) (↑PU0B)→↓LDXT0*/8A3 4. ↓LDXT0* + ↑IOSLU* + ↑LDXT3* + ↑LDXPR*→↑LDEXT/8A3 5. ↑LDEXT→U00-U11 = EXT00-EXT11/sheets 14, 15, 16 6. ↓LDXT0* + ↑IOSLU* + ↑SELU6* + ↑ANDJK*→↑SELU/8A3 7. ↑SELU→↓SLUMX*/8A4 8. ↓SLUMX*→↑MXEN/9A2 9. ↑MXEN→↓MXEN*/9A2 10. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2 11. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2 12. ↓SLUMX* + ↑SLKMX* + ↑EX0* + ↑SLSMX*→↑MXS0/9B2 13. (↓MXEN*) (↓MXS2) (↓MXS1) (↑MXS0)→MX00-MX11 = U00-U11/sheets 14, 15, 16	The trap address is gated from EXT00*-EXT11* through the MX multiplexer (MX00-MX11) to the adders.

Table 4-130. Interrupt Initiation, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
C.	PU0	B00-B11 = MX00-MX11 +1 1. ↓INTFF* + ↑CSCIN*→↑INTCIN/12A1 2. ↑INTCIN + ↑PU0B→↓CIN2*/12A2 3. ↓CIN2* + ↑CIN3* + ↑CIN1*→↑CIN/12A2 4. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13 5. (↑CIN) (TS00-TS11 = 0000_8) (MX00-MX11 = trap address)→B00-B11 = trap address +1/sheets 14, 15, 16	A carry-in is generated and the trap address is incremented by the adders and presented on the processor bus (B00-B11)
D.	PU0	B00-B11→AR 1. ↑PU0B→↓PEA*/9B1 2. ↓PEA*→↑PEA/9B1 3. (↑PEA) (↑REGCLK)→↓CPA*/9B2 4. (↓CPA*) (↓PEA*)→A00-A11 = B00-B11/sheets 14, 15, 16	The incremented trap address is loaded into the address register.
E.	PU1	RESET INTP, INTP* 1. ↓PU1* + ↑EXPMR* + ↑STCLR*→↑STPU1/3A3 2. ↑STPU1→↓STPU1*/3A3 3. ↓STPU1*→↓INTP/4B4, ↑INTP*/4B4	Signal INTP and its complement are reset.
F.	PU3	RESET ENINT 1. (↑INTFF) (↑PU3B)→↓INT3*/8B3 2. ↓INT3* + ↑IOF5* + ↑STCLR + ↑ENINT→↑ENINT*/11B2 3. (↑ENINT*) (↑CLKIN*)→↓ENINT/11B2	Signal ENINT is reset low to disable all interrupts.
G.	PU3	PC→MDR 1. (↑INTFF) (↑PU3B)→↓INT3*/8B3 2. ↓INT3* + ↑JPS3* + ↑PRAR*→↑SELP/8B3 3. ↑SELP→↓SLPMX*/8B4 4. ↓SLPMX→↑MXEN/9A2 5. ↑MXEN→↓MXEN*/9A2 6. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2 7. ↑EX1* + ↓SLPMX* + ↑SLRMX* + ↑SLSMX*→↑MXS1/9B2 8. ↑SLSMX* + ↑EX0* + ↑SLKMX* + ↑SLUMX*→↓MXS0/9B2 9. (↓MXEN*) (↓MXS0) (↑MXS1) (↓MXS2)→MX00-MX11 = P00-P11/sheets 14, 15, 16 10. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13	The contents of the program counter (P00-P11) are gated through the bus and MBR multiplexer (PMR00-PMR11) to the memory data register (M00-M11).

Table 4-130. Interrupt Initiation, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
		11. (↓CIN) (TS00-TS11 = 0000_8)→B00-B11 = MX00-MX11/sheets 14, 15, 16	
		12. ↓INTFF* + ↑CSRDB* + ↑LDMR*→ ↑RDB3/12B2	
		13. (↑RDB3) (↑PU3B)→↓BMEM*/12B2	
		14. ↓BMEM*→↑RDB/12B3	
		15. ↑RDB→↓RDB*/12B3	
		16. ↓RDB*→PMR00-PMR11 = B00-B11/sheet 17	
		17. ↓BMEM* + ↓PU3*→↑PEM/12B3	
		18. ↑PEM→↓PEM*/12B3	
		19. (↑PEM) (↑REGCLK)→↓CPM*/12B3	
		20. (↓PEM*) (↓CPM*)→M00-M11 = PMR00-PMR11/sheet 17	
H.	PU6	AR +1→AR	
		1. (↑INTFF) (↑PU6B)→↓SLAMX*/8A2	The contents of the address register (A00-A11) are incremented and replaced in the address register.
		2. ↓SLAMX→↑MXEN/9A2	
		3. ↑MXEN→↓MXEN*/9A2	
		4. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2	
		5. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2	
		6. ↑SLSMX* + ↑EX0* + ↑SLKMX* + ↑SLUMX*→↓MXS0/9B2	
		7. (↓MXEN*) (↓MXS0) (↓MXS1) (↓MXS2)→MX00-MX11 = A00-A11/sheets 14, 15, 16	
		8. (↑INTFF) (↑PU6B)→↓CIN1*/12B2	
		9. ↓CIN1* + ↑CIN3* + ↑CIN2*→ ↑CIN/12A2	
		10. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13	
		11. (↑CIN) (TS00-TS11 = 0000_8) (MX00-MX11 = trap address +1)→ B00-B11 = trap address +2/sheets 14, 15, 16	
I.	PU6	AR→PC	
		1. (↑INTFF) (↑PU6B)→↓INT6B/9B2	The contents of the address register (the trap address, now doubly incremented) is loaded into the program counter (P00-P11).
		2. ↓INT6B + ↑BP205* + ↑JMP6* + ↑EXM1B*→↑MPEP/9B3	
		3. ↑MPEP→↓PEP*/9B3	
		4. ↓PEP*→↑PEP/9B1	
		5. (↑PEP) (↑REGCLK)→↓CPP*/9B2	
		6. (↓PEP*) (↓CPP*)→P00-P11 = B00-B11/sheets 14, 15, 16	

Table 4-130. Interrupt Initiation, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
J.	PU7	RESET ↓INTFF, ↓INTCSF 1. (↑PU7B) (↑CPPU)→↓RDBLE*/4A4 2. ↓RDBLE* + ↑GO→↑RDBLE/4B2 3. (↑RDBLE) (↑INTP*)→↓RITFF*/4B4 4. ↓RITFF*→↓INTFF/4B4, ↑INTFF*/4B4 5. ↑INTFF* + ↑CSFF*→↓INTCSF/4B4	Signal INTFF is reset to mark the end of the interrupt initiation cycle; INTCSF is also reset.
K.	PU0	RESET ↑BLKE*, ↑BLKB* 1. ↓INTCSF→↑INTCS*/5B2 2. (↑INTCS*) (↑PU0) (↑GO)→ ↓RBLK*/5B2 3. ↓RBLK*→↓BLKBEP/5B2 4. ↓BLKBEP→↑BLKE*/5B3 5. ↓BLKBEP→↑BLKB*/5B3	Signals BLKE* and BLKB* are reset to unlock the basic and execute phases for the beginning of the first instruction of the interrupt routine.

disabled interrupt system. At period PU3, signal ENINT is brought low to disable all interrupts as shown in reference F of Table 4-130 and waveforms A, B, and F of Figure 4-107.

In order that the place in a program where an interrupt occurred be remembered, the contents of the program counter (9, Figure 4-97) are stored at the location specified by the incremented trap address. To facilitate this, during period PU3, the memory data register (19) is loaded with the contents of the program counter, from which these contents will later be stored into the memory. (See Memory, paragraph 4.5.) The contents of the program counter are gated through the MX multiplexer (12) to the adder (14) which pass them unchanged onto bus lines B00 through B11. The contents of the bus are then gated to the memory data register (19) through the bus and memory buffer multiplexer (15). The memory data register (19) is then parallel enabled and clocked to store the data word presented to it. The simplified logic description for these events is shown in references B and C of Table 4-130 and their timing is illustrated by waveforms of Figure 4-107.

The first instruction executed during an interrupt is located two locations past the trap address. At period PU6, the contents of the address register (which contains the incremented trap address) are again incremented to give the address of this location, and that address is stored in the program counter. The simplified logic description for these events is shown in references H and I of Table 4-130 and their timing is shown by waveforms of Figure 4-107.

At the trailing edge of pulse PU7, signals INTFF and INTCSF are reset to zero to end the interrupt initiation cycle. Then at period PU0, BLKE* and BLKB* are set high level so that the basic phase of the first instruction of the interrupt service routine can be entered. The simplified logic description of these events is shown in references J and K of Table 4-130 and their timing is illustrated by waveforms of Figure 4-107.

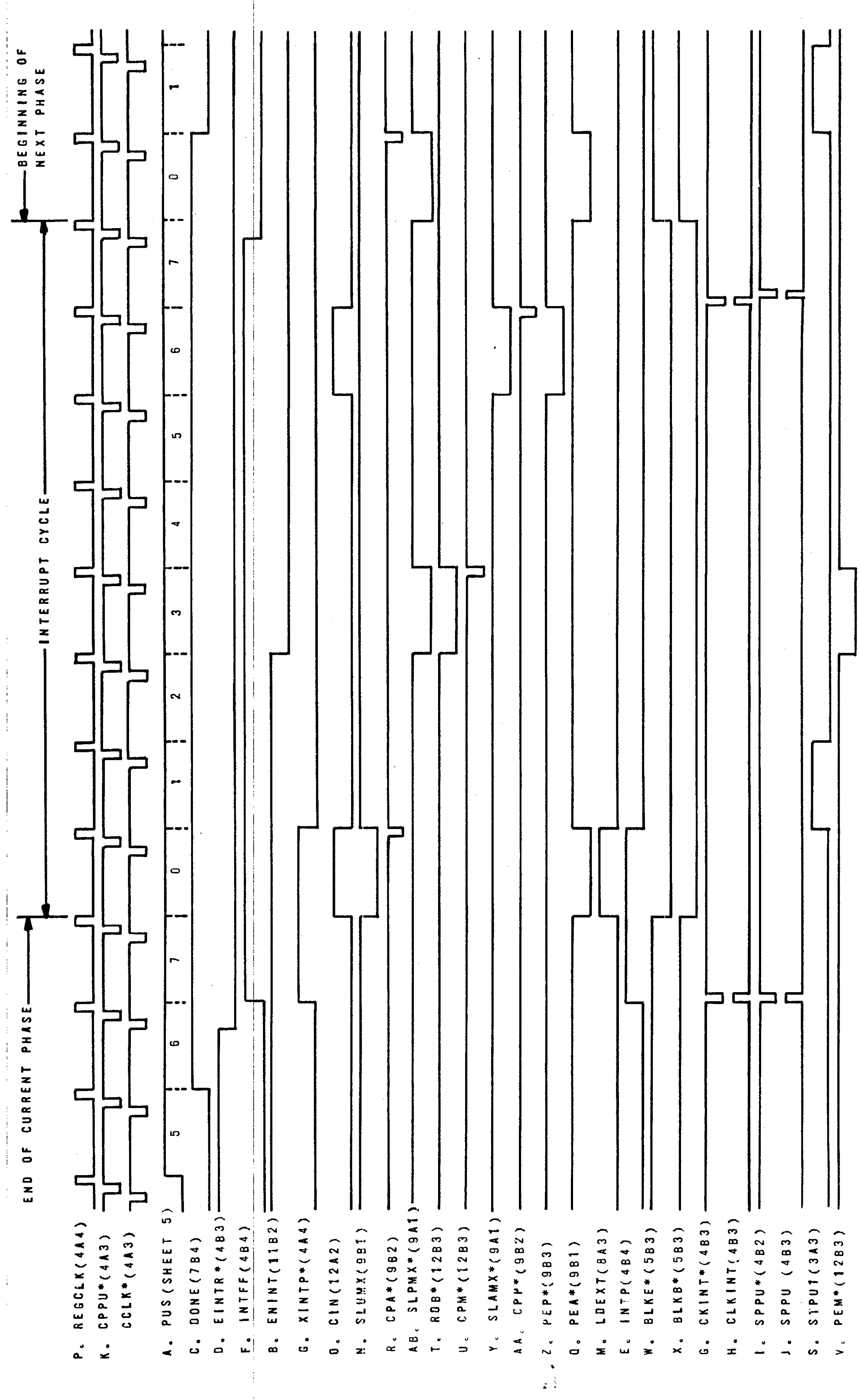


Figure 4-107. Interrupt Initiation, Timing Diagram

The routine entered during interrupt initiation will either be a service routine or a polling routine. If a trap address is used, the service routine can be directly entered. A polling routine starting at location 0002_8 or two locations beyond a trap address can be used to determine which one of the peripheral devices requested the interrupt. After this has been determined, the proper service routine can be entered. Return to the interrupted program is effected by an indirect JMP instruction through the location where the program counter contents were stored during initiation of the interrupt.

4.6.5.2 INTERRUPT CONTROL INSTRUCTIONS. The occurrence of normal interrupts (Powerfail interrupts are discussed in paragraph 4.6.5.3) is controlled by four signals; ENINT, IONA*, IONB*, IONL* (Table 4-131). Signal ENINT, when asserted (high) enables the processor to accept interrupts. Signals IONA*, IONB*, and IONL* are transmitted on the I/O bus and enable peripheral devices to request interrupts. These four signals are brought under program control by the five interrupt control instructions; IONH, IONA, IONB, `IONN, and IOFF. Instructions IONH, IONA, IONB, and IONN enable various interrupts to occur by asserting signal ENINT (also asserting signal ION, the status register bit representing signal ENINT) and combinations of the other three interrupt control signals. Instruction IOFF causes signal ENINT (and ION) to be unasserted.

Instruction IONH causes signals ENINT and ION to be asserted and signals IONA*, IONB*, and IONL* to be unasserted. At period BP7, an inverted pulse is generated as signal CLKIN* (reference A, Table 4-131). This CLKIN* pulse sets signal ENINT high, and clocks the interrupt priority latch causing it to load signals I09B, I10B, I11N and I10I11 (reference B, Table 4-131). This sets signals ION, IONA*, IONB*, and IONL* high.

Table 4-131. IONH Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	BP7	GENERATED CLKIN* PULSE 1. (↑I04*) (↑I05B*)→↓I415/9A1 2. (↓I00) (↓I01) (↑I02) (↓I03)→↓OP1*/7A1 3. ↓I415→↑I415*/9A1 4. (↓I06) (↓I07) (↓I08) (↓OP1*)→↑OP100/11A1 5. (↑OP100) (↑I415)→↓OP10X*/11B1 6. (↓OP10X*)→↑OP10X/11B1 7. (↑OP10X) (↑I09B) (↑BP7B) (↑CPPU)→ ↓CLKIN*/11B2	An inverted pulse is generated as signal CLKIN*.
B.	BP7	SET ↑ENINT, ↑ION; RESET ↑IONA*' ↑IONB*, ↑IONL* 1. ↓CLKIN*→↑ENINT/11B2 2. (↓CLKIN*) (I09-I11 = 100_2)→↑ION, ↓IONB, ↓IONA, ↓IONN/11B3 3. (↓IONB) (↑ENINT)→↓IONB*/11B4 4. (↓IONA) (↑ENINT)→↓IONA*/11B4 5. (↓IONN) (↑ENINT)→↓IONL*/11B4	The highest level interrupts are enabled.

Execution of instructions IONA, IONB, and IONN is identical to that of instruction IONH, except for the difference in signals I09B, I10B, I11N, and I10I11. For the IONA instruction, signals ION, IONB*, and IONL* are set high and IONA is set low (Table 4-132). For the IONB instruction, ION, IONA*, and IONL* are set high and IONA* is set low (Table 4-133). For the IONN instruction, signal ION is set high and signals IONA*, IONB*, and IONL* are all set low (Table 4-134). For the IONN operation (Table 4-134) and the IOFF operation (Table 4-135), fundamental operations are also dependent on the I09, I10, and I11 bit patterns.

Table 4-132. IONA Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	BP7	SET ↑ENINT, ↑ION, ↓IONA*; RESET ↑IONB*, ↑IONL*	
		1. ↓CLKIN*→↑ENINT/11B2	Enable highest level and A level interrupts.
		2. (↓CLKIN*) (I09-I11 = 101_2)→↑ION, ↓IONB, ↑IONA, ↓IONN/11B3	
		3. (↓IONB) (↑ENINT)→↑IONB*/11B4	
		4. (↑IONA) (↑ENINT)→↓IONA*/11B4	
		5. (↓IONN) (↑ENINT)→↑IONL*/11B4	

Table 4-133. IONB Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	BP7	SET ↑ENINT, ↓IONB*; RESET ↑IONA*, ↑IONL*	
		1. ↓CLKIN*→↑ENINT/11B2	Enable highest level and B level interrupts.
		2. (↓CLKIN) (I09-I11 = 110_2)→↑ION, ↑IONB, ↓IONA, ↓IONN/11B3	
		3. (↑IONB) (↑ENINT)→↓IONB*/11B4	
		4. (↓IONA) (↑ENINT)→↑IONA*/11B4	
		5. (↓IONN) (↑ENINT)→↑IONL*/11B4	

Table 4-134. IONN Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	BP7	SET ↑ENINT, ↓IONB*, ↓IONA*, ↓IONL*	
		1. (↓OP10X*) (I09-I11 = 111)→↓I1011*/11B2	All interrupt levels are enabled.
		2. ↓I1011*→↑I10I11/11B3	
		3. ↓CLKIN*→↑ENINT/11B2	
		4. (↓CLKIN*) (I09-I11 = 111) (↑I10I11)→↑ION, ↑IONB, ↑IONA, ↑IONN/11B3	
		5. (↑IONB) (↑ENINT)→↓IONB*/11B4	
		6. (↑IONA) (↑ENINT)→↓IONA*/11B4	
		7. (↑IONN) (↑ENINT)→↓IONL*/11B4	

Table 4-135. IOFF Instruction, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	BP5	RESET ↓ENINT, ↑IONB*, ↑IONA*, ↑IONL* 1. (↑I04*) (↑I05B*)→↓I415/9A1 2. (↓I00) (↓I01) (↑I02) (↓I03)→↓OP1*/7A1 3. ↓I415→↑I415*/9A1 4. (↓I06) (↓I07) (↓I08) (↓OP1*)→ ↑OP100/11A1 5. (↑OP100) (↑I415)→↓OP1-00X*/11B1 6. (↓OP1-00X*) (↓I09B) (↑I10B) (↑I11B)→ ↓IOFF*/11B1 7. ↓IOFF*→↑IOFF/11B1 8. (↑IOFF) (↑BP5)→↓IOF5*/11B2 9. ↓IOF5* + ↑ENINT + ↑INT3* + ↑STCLR*→ ↑ENINT*/11B2 10. (↑ENINT*) (↑CLKIN*)→↓ENINT/11B2 11. ↓ENINT→↑IONB*, ↑IONA*, ↑IONL*/11B4	All interrupt levels are disabled.

The IOFF instruction (Table 4-135) causes all the interrupt control signals to be unasserted. At period BP5, signal ENINT is reset low. The state of the priority interrupt latch is not affected, and signals IONA*, IONB*, and IONL* are all set low as a result of the low ENINT signal. Because signals ION, IONA, IONB, and IONN are not affected the interrupt bits of the status register are not changed.

4.6.5.3 POWERFAIL INTERRUPT SYSTEM. The powerfail interrupt system provides the computer with the capability to automatically store in core the contents of all processor registers when a power failure occurs. Since the contents of the core memory are not affected by a power failure, this allows programs that were running when a power failure occurred to be resumed, when power is restored, at the point where they were halted by the power failure.

The onset of a power failure is detected by circuitry associated with the system power supply (sheet 19). When the power level drops to a preset level (as determined by the bias circuit of transistor Q6/19A4), transistor Q6 turns on. This results in setting signal BDPWR*/19A4 low. If the powerfail interrupt system is enabled at that time, an interrupt request is generated. For about one millisecond after signal BDPWR* goes low, there is sufficient power to drive the logic circuitry of the computer. If within that time a suitable powerfail routine is executed, computation can be resumed where interrupted by the power failure. Otherwise, the processor status is lost.

The ability of the powerfail logic to generate interrupt requests is controlled by two instructions; PION and PIOF. Instruction PION enables the powerfail logic while instruction PIOF disables it.

When instruction PION is executed, signal PRINT Table 4-136 is set high and signal PRINT* is set low, enabling assertion of signal INTR* (interrupt request flag) by a

Table 4-136. Powerfail Interrupt System, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP3	SET ↑PRINT, ↓PRINT*	
		1. (↓I00) (↓I01) (↑I02) (↑I03)→↓OP2*/7A2	Powerfail interrupt system is enabled. (PION instruction)
		2. ↓OP2*→↑OP2/7A3	
		3. (↑OP2) (↑PULSKP) (↑I07*) (↑I08*) (↑I09*) (↑I10*) (↑I11*)→↓OP2PR*/3A1	
		4. ↓OP2PR*→↑OP2PR/3A2	
		5. (↑OP2PR) (↑I5BSKP)→↓SPRIN*/3A1	
		6. ↓SPRIN*→↑PRINT, ↓PRINT*/3A2	
B.	BP3	RESET ↓PRINT, ↑PRINT*	
		1. (As in A. 1-4)	Powerfail interrupt system is disabled. (PIOF instruction)
		2. (↑OP2PR) (↑I4SKP)→↓RPRIN*/3A1	
		3. ↓RPRIN*→↑PRINT*, ↓PRINT/3A2	
C.	PU6	RESET ↓GO, ↑GO*	
		1. (↑BDPWR) (↑PRINT*)→↓RGPWR*/3A2	The go flip-flop is reset to halt the processor.
		2. ↓RGPWR* + ↑SI* + ↑STOP*→↑RGO/3A2	
		3. (↑CPPU) (↑DONE) (↑PU6B) (↑GO)→ ↓FFCLK*/3A2	
		4. ↓FFCLK*→↑FFCLK/3A2	
		5. (↑RGO) (↑FFCLK)→↓RGO*/3A3	
		6. ↓RGO*→↓GO, ↑GO*/3A3	
D.	ANY	SET ↓BLKMC*; RESET ↓ENMI	
		1. (↑GO*) (↑BDPWR)→↓BLKMC*/4A2	The clock is stopped (BLKMC*) and the memory disabled (ENMI).
		2. ↓GO→↑RUN*/19A4	
		3. (↑RUN*) (↑BDPWR)→↓ENMI/19B4	
E.	BP5	PC +1→PC	
		1. (As in A. 1-4)	The contents of the program counter are incremented and loaded again into the program counter.
		2. (↑BDPWR) (↑OP2PR) (↑I06B)→ ↓SKBPW*/10B4	
		3. ↓SKBPW* + ↑SKZ* + ↑OPSKP* + ↑SKFLP*→↑OP2SKP/10A4	
		4. (↑OP2SKP) (↑BP5) (↑OP2)→ ↓CIN3*/12A2	
		5. ↓CIN3* + ↑CIN2* + ↑CIN1*→↑CIN/12A2	
		6. (↑OP2) (↑BP5B)→↓SLPMX*/8B4	
		7. ↑STEX* + ↑SLJMX* + ↑SLKMX* + ↓SLPMX* + ↑SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2	
		8. ↑MXEN→↓MXEN*/9A2	
		9. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2	
		10. ↑SLRMX* + ↑EX1* + ↓SLPMX* + ↑SLSMX*→↑MXS1/9B2	

Table 4-136. Powerfail Interrupt System, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
		11. ↑SLSMX* + ↑EX0* + ↑SLKMX* + ↑SLUMX*→↓MXS0/9B2	
		12. (↓MXEN) (↓MXS2) (↑MXS1) (↓MXS0)→ MX00-MX11 = P00-P11/sheets 14, 15, 16	
		13. (↑TSADD*) (↑TSSUB)→TS00-TS11 = 0000_8/sheet 13	
		14. (↑CIN) (TS00-TS11 = 0000_8) (MX00-MX11 = P00-P11)→B00-B11 = (P00-P11) +1/sheets 14, 15, 16	
		15. (↑OP2) (↑BP5B)→↓BP205*/9B3	
		16. ↓BP205* + ↑JMP6* + ↑INT6B + ↑EXM1B*→ ↑MPEP/9B3	
		17. ↑MPEP→↓PEP*/9B3	
		18. ↓PEP*→↑PEP/9B1	
		19. (↑PEP) (↑REGCLK)→↓CPP*	
		20. (↓PEP*) (↓CPP*)→P00-P11 = B00-B11	

high-level BDPWR signal (see reference A of Table 4-136). The contents of the instruction register (1500$_8$) are decoded to produce a low SPRIN* signal. This asserted SPRIN* signal asynchronously sets the set-reset latch that generates signals PRINT and PRINT*.

When instruction PIOF is executed, signal PRINT is reset low and signal PRINT* goes high, disabling assertion of signal INTR* (interrupt request flag), but enabling assertion of signal RGPWR* by a high-level BDPWR signal (see reference B of Table 4-136). The contents of the instruction register (1600$_8$) are decoded to produce a low RPRIN* signal. This asserted RPRIN* signal asynchronously resets the set-reset latch that generates signals PRINT and PRINT*.

If signal BDPWR becomes asserted while the powerfail logic is not enabled to generate an interrupt request, an orderly powerdown of the computer is performed immediately after completion of the current instruction execution. Assertion of signal RGPWR* causes the GO latch to be reset, thereby halting the processor as shown in reference C of Table 4-136.

The halted condition of the computer and the asserted BDPWR signal initiates the powerdown. Signal BLKMC* is asserted (reference D of Table 4-136) to block the clock output. Signal ENMI is brought low causing an electronic switch (transistor Q7 and Q8/19B4) to deprive the current sources of the core memory drivers of their power. Cutting off this power renders the core memory unalterable, thus protecting its contents from the unpredictable effects of the final loss of power.

Because none of the processor registers have their contents stored when a power failure occurs and the powerfail logic is disabled, the status of the processor is lost in this case. The memory contents are protected in this case, so that it can be safely relied upon that the contents of the memory are retained as they were just before the power failure occurred.

When a power failure occurs while the powerfail logic is enabled, the only thing that happens automatically is the generation of an interrupt request (signal INTR* is asserted). It is left to program control to halt the computer so that an orderly powerdown can be performed before the power level drops too low to operate the logic circuitry of the processor reliably. Five alternatives are possible.

1. The program can ignore the power failure indications and let the system crash in a disorderly fashion making the information stored in the memory unreliable.

2. The program can finish and come to a normal halt before the power completely fails, resulting in an orderly power down and loss of processor status information.

3. The program can execute a SKPL flag to check for a powerfail flag (BDPWR) and branch to a power failure handling routine to store the contents of all the processor registers and come to a halt. This would make it possible to continue later.

4. The program can allow interrupts, enabling the powerfail interrupt to occur, resulting in execution of a power failure handling routine as in case 3.

5. The program can execute a PIOF instruction, resulting in an immediate and orderly powerdown.

The interrupt request generated by the powerfail logic is not accompanied by a trap address. A SKPL instruction should be included in the polling routine starting at location 0002_8 of memory field 1. To check for the powerfail flag (BDPWR).

The SKPL instruction, when executed, increments the contents of the program counter if signal BDPWR is asserted at that time. The instruction is decoded and at period BF5 a carry in (CIN) signal to the adders is generated for signal BDPWR high (see Section E of Table 4-136). The contents of the program counter (P00 through P11) are gated through the MX multiplexer to the adders. The output from the TS multiplexer (TS00 through TS11) to the adders is set to 0000_8. The adders in response to these inputs present the incremented contents of the program counter on bus lines (B00 through B11). The program counter is then parallel enabled and loaded to store the incremented number.

4.6.5.4 PROGRAM-CONTROLLED PIO INSTRUCTIONS. Program-controlled I/O (PIO) instructions provide an interface between programs running on the processor and devices peripheral to the processor. This interface allows for the exchange of data and control information. A single PIO instruction can perform an arbitrary combination of four functions.

1. Generate control signals to produce certain functions in a peripheral device.

2. Input or output a twelve-bit word from the J register.

3. Input or output a twelve-bit word from the K register.

4. Cause a peripheral device to generate a skip signal EXSKP*, which in turn, causes the processor to increment the program counter, thereby skipping the next instruction.

The object of performance of these basic processes is discussed earlier in this section. This paragraph concerns itself with the way in which these processes are brought about.

There are two types of PIO instructions; single-word and two-word (Table 4-128). Both types produce the same types of control signals for control of the peripherals: an address presented on the IOM lines, and peripheral control pulses (PCP0* through PCP3*). A single-word PIO instruction has a four-bit address taken from bits 4 through 7 of the instruction. A two-word PIO instruction has an eight-bit address taken from bits 0 through 7 of the second word of the instruction. However, it should be understood that all twelve bits of a single-word PIO instruction or all twelve bits of the second word of a two-word PIO instruction are presented on the IOM lines of the I/O bus and may be used by peripheral devices as addressing information.

Regardless of the way in which device addressing is implemented, the last four bits (8-11) of a single-word instruction, or the second word of a two-word instruction, control the generation of peripheral control pulses. Set bits (logical 1's) in positions 8 through 11 result, respectively, in generation of negative pulses on lines PCP0* through PCP3* of the I/O Bus.

Pulses PCP0* through PCP3* are generated concurrently with pulses PU4 through PU7, which are extended to twice their normal duration during the final basic phase periods of PIO instruction execution. Table 4-137 is the description of Fundamental Operations for PCP generation. Figure 4-108 illustrates timing. Pulses PU4 through PU7 are extended by alternately blocking and passing the CCLK* pulse (waveform B) that generates the clock pulse (waveform C) for the pulser shift register. The blocking signals (BLKCC* and DBLE*) are controlled by the I/O timing control register. Pulse PCP0 is only half the duration of the other three PCP pulses because signal STDBL, (waveform H), which has a part in gating pulses PCP0* through PCP3* is not set high until the middle of the extended PU4 pulse.

Any combination of the four PCP pulses can be used as general control signals to a peripheral device controller. The other three basic PIO instruction functions are constrained to occur during certain time frames defined by the PCP pulses. An EXSKP* signal can only result in a program skip if it occurs simultaneously with pulses PCP0*, PCP1* or PCP2*.

A low-level EXSKP* signal during period BP4, BP5, or BP6 (pulses PCP0, PCP1, or PCP2) results in setting signal IOSKP high until the end of PIO instruction execution.

Table 4-137. PCP Generation, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP4	GENERATE ↓PCP0* 1. ↓CCLK*→↑CCLK/4A3 2. (↑I/OOSC) ((↑STDBL*)→↓BLKCC*/4A3 3. (↑CCLK) (↑DBLE*) (↓BLKCC*)→ ↑CPPU*/4A3 4. ↑CCLK*→↓CCLK/4A3 5. (↓CCLK) (↑DBLE*) (↓BLKCC)→ ↑CPPU*/4A3 6. (↑CCLK*) (↑I/OOSC) (↑BP4) (↑OSCPUL) (↑RDBLE*)→↑DBLE, ↓DBLE*/4A4 7. (↑PU4) (↑I/OOSC) (↑M08B) (↑STDBL)→ ↓PCP0*/5B4	The CPPU* pulse is blocked to extend BP4. If bit 8 of the MDR is set, PCP0* is pulled low for the last half of the extended BP4.
B.	BP5	GENERATE ↓PCP1* 1. ↓CCLK*→↑CCLK/4A3 2. (↑I/OOSC) (↓STDBL)→↑BLKCC*/4A3 3. (↑CCLK) (↑DBLE*) (↑BLKCC*)→ ↓CPPU*/4A3 4. ↑CCLK*→↓CCLK/4A3 5. (↓CCLK) (↑DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 6. (↑CPPU*) (↑PU4)→↑PU5/5B1 7. (↑PU5) (↑I/OOSC) (↑M09B) (↑STDBL)→↓PCP1* 8. (↑CCLK*) (↑I/OOSC) (↑STDBL) (↑RDBLE*)→↑DBLE, ↓DBLE*/4A4 9. (↑DBLE)→↓PCPST*/4A4 10. ↓CCLK*→↑CCLK/4A3 11. (↑CCLK) (↓DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 12. ↑CCLK*→↓CCLK/4A3 13. (↓CCLK) (↓DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 14. (↑CCLK*) (↑I/OOSC) (↑STDBL) (↑RDBLE*) (↑OSCPUL)→↓DBLE, ↑DBLE*/4A4 15. ↓DBLE→↑PCPST*/4A4	A CPPU* pulse generates BP5. Signal PCP1* is pulled low for the duration of BP5. The next CPPU* pulse is blocked to extend BP5 to double its usual length. Signal PCPST* is pulled low for the first half of BP5.
C.	BP6	GENERATE ↓PCP2* 1. ↓CCLK*→↑CCLK/4A3 2. (↑CCLK) (↑DBLE*) (↑BLKCC*)→ ↓CPPU*/4A3 3. ↑CCLK*→↓CCLK/4A3 4. (↓CCLK) (↑DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 5. (↑CPPU*) (↑PU5)→↑PU6/5B1 6. (↑PU6) (↑I/OOSC) (↑M10B) (↑STDBL)→ ↓PCP2*/5B4 7. (↑CCLK*) (↑I/OOSC) (↑STDBL) (↑RDBLE*)→↑DBLE, ↓DBLE*/4A4	A CPPU* pulse generates BP6. Signal PCP2* is pulled low for the duration of BP6. The CPPU* pulse is blocked to extend BP6 to double its usual length. Signal PCPST* is pulled low for the first half of BP5.

Table 4-137. PCP Generation, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
		8. ↑DBLE→↓PCPST*/4A4 9. ↓CCLK*→↑CCLK/4A3 10. (↑CCLK) (↓DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 11. ↑CCLK*→↓CCLK/4A3 12. (↓CCLK) (↓DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 13. (↑CCLK*) (↑I/OOSC) (↑STDBL) (↑RDBLE*) (↑OSCPUL)→↓DBLE, ↑DBLE*/4A4 14. ↓DBLE→↑PCPST*/4A4	
D.	BP7	GENERATE PCP3* 1. ↓CCLK*→↑CCLK/4A3 2. (↑CCLK) (↑DBLE*) (↑BLKCC*)→ ↓CPPU*/4A3 3. ↑CCLK*→↓CCLK/4A3 4. (↓CCLK) (↑DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 5. (↑CPPU*) (↑PU6)→↑PU7/5B1 6. (↑PU7) (↑I/OOSC) (↑M11B) (↑STDBL)→ ↓PCP3*/5B4 7. (↑CCLK*) (↑I/OOSC) (↑STDBL) (↑RDBLE*)→↑DBLE, ↓DBLE*/4A4 8. ↑DBLE→↓PCPST*/4A4 9. ↓CCLK*→↑CCLK/4A3 10. (↑CCLK) (↓DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 11. ↑CCLK*→↓CCLK/4A3 12. (↓CCLK) (↓DBLE*) (↑BLKCC*)→ ↑CPPU*/4A3 13. (↑CCLK*) (↑I/OOSC) (↑STDBL) (↑RDBLE*) (↑OSCPUL)→ ↓DBLE, ↑DBLE*/4A4 14. ↓DBLE→↑PCPST*/4A4	A CPPU* pulse generates BP7. Signal PCP3* is pulled low for the duration of BP7. The next CPPU* pulse is blocked to extend BP7 to twice its usual length. Signal PCPST* is pulled low for the first half of BP7.
E.	BP7	RESET ↓STDBL, ↑STDBL*, IR = 0000_8 1. ↓CCLK*→↑CCLK/4A3 2. (↑CCLK*) (↑DBLE*) (↑BLKCC*)→ ↓CPPU*/4A3 3. ↓CPPU*→↑CPPU/4A3 4. (↑CPPU) (↑PU7B)→↓RDBLE*/4A4 5. ↓RDBLE*→↓STDBL, ↑STDBL*/4A4 6. (↑PU7B) (↑I/OOSC) (↑CPPU)→ ↓CMRI, ↑EMRI/6B4 7. (↓CMRI) (↑EMRI)→↓MRI*/6B4 8. ↓MRI*→I00-I11 = 0000_8/sheet 17	Signal STDBL and its complement are reset and the instruction register cleared.

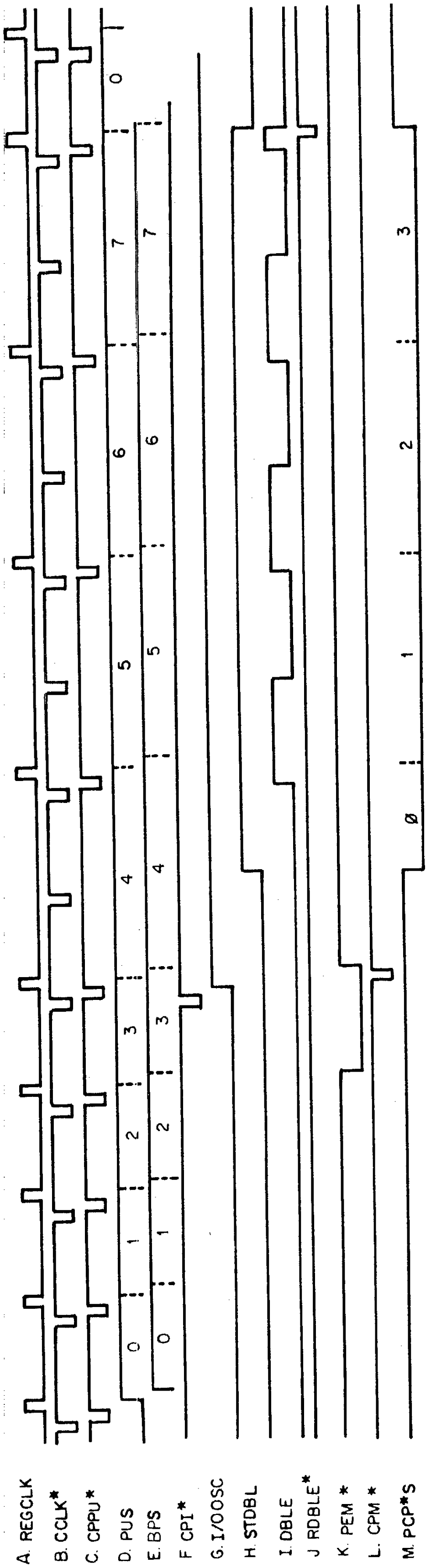


Figure 4-108. Program Controlled I/O Instructions, Timing Diagram

The fundamental operation for this event is shown in reference A of Table 4-138. When period BP7 occurs, if IOSKP has been set high, a carry-in signal (CIN) is generated and the program counter thereby incremented. The Fundamental Operation for this function is shown in reference B of Table 4-138.

A single twelve-bit word transfer between the J or K register and a peripheral device can be carried out during the occurrence of pulses PCP1* and PCP2*. Thus, a single PIO instruction can produce any one of the sixteen modes of data transfer listed in Table 4-139. Signals DIN* and EXTK*, generated by the peripheral device, control these modes. Two important features are not taken into account by this table. First, there is no provision

Table 4-138. Externally Generated Skip, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP4, BP5, or BP6	SET ↑IOSKP 1. ↓EXSKP*→↑EXSKP/12A1 2. (↑EXSKP) (↑BP456)→↓SIOSK*/12A1 3. ↓SIOSK* + ↑IOSKP*→↑IOSKP/12A1 4. ↑IOSKP + ↑I/O→↓IOSKP*/12A1 5. ↓IOSKP* + ↑SIOSK*→↑IOSKP/12A1	Signal IOSKP* is set high to generate a carry-in (CIN) later (BP7).
B.	BP7	PC +1→PC 1. (↑I/O) (↑BP7)→↓CSELP, ↑ESELP/8B3 2. (↓CSELP) (↑ESELP)→↓SLPMX*/8B4 3. ↓SLPMX* + ↑SLJMX* + ↑SLKMX* + ↑SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2 4. ↑MXEN→↓MXEN*/9A2 5. ↑SLJMX* + ↑EX2*/9A1 + ↑SLKMX* + ↑SLRMX→↓MXS2/9A2 6. ↑EX1* + ↓SLPMX* + ↑SLRMX* + ↑SLSMX*→↑MXS1/9B2 7. ↑SLSMX* + ↑EX0* + ↑SLKMX + ↑SLUMX*→↓MXS0/9B2 8. (↓MXEN*) (↓MXS0) (↑MXS1) (↓MXS2)→MX00-MX11 = P00-P11/sheets 14, 15, 16 9. (↑IOSKP) (↑BP7)→↓CIN3*/12A2 10. ↓CIN3* + ↑CIN2* + ↑CIN1*→ ↑CIN/12A2 11. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13 12. (↑CIN) (TS00-TS11 = 0000_8) (MX00-MX11 = P00-P11)→B00-B11 = (P00-P11) +1/sheets 14, 15, 16 13. (↑I/O) (↑BP7)→↓PEP*/9B3 14. ↓PEP*→↑PEP/9B1 15. (↑PEP) (↑REGCLK)→↓CPP* 16. (↓PEP*) (↓CPP*)→P00-P11 = B00-B11/sheets 14, 15, 16	Contents of program counter (P00-P11) are incremented and loaded again into the program counter.

Table 4-139. Possible PIO Data Transfers

	PCP1*		PCP2*		
	DIN*	EXTK*	DIN*	EXTK*	Data Transfer Enabled
1.	0	0	0	0	Input to K register twice
2.	0	0	0	1	Input to both J and K registers
3.	0	0	1	0	Input to K followed by output from K
4.	0	0	1	1	Input to K followed by output from J
5.	0	1	0	0	Input to J followed by input to K
6.	0	1	0	1	Input to J twice
7.	0	1	1	0	Input to J followed by output from K
8.	0	1	1	1	Input to J followed by output from J
9.	1	0	0	0	Output from K followed by input to K
10.	1	0	0	1	Output from K followed by input to J
11.	1	0	1	0	Output from K twice
12.	1	0	1	1	Output from K followed by output from J
13.	1	1	0	0	Output from J followed by input to K
14.	1	1	0	1	Output from J followed by input to J
15.	1	1	1	0	Output from J followed by output from K
16.	1	1	1	1	Output from J twice

for changing the content of either the J or K register in such a way as to allow output of two different words from the same register with a single instruction. Second, the processor cannot ignore an operation. If the DIN* signal is held low during the occurrence of either pulse PCP1* or PCP2*, either the J or K register (depending on signal EXTK*) will be loaded with the data word on the I/O bus (signals EXT00* through EXT11*) at that time. If the DIN* signal is high during PCP1* and PCP2*, the contents of the J or K register are presented on the I/O bus (signals OUT00 through OUT11) but may be ignored by the peripheral device addressed by that instruction. Thus, when no data transfers are desired and the contents of the J and K registers are important, signal DIN* must not be asserted (low) during the occurrence of pulses PCP1* or PCP2*.

A word is input to the K register from a peripheral during periods BP5 or BP6 (PCP1* or PCP2*) by asserting both DIN* and EXTK* signals. The data word is gated from EXT00* through EXT11* through the utility gates, MX multiplexer, and adders to the bus (signals B00 through B11) from which it is then loaded into the K register by parallel enabling and clocking the register. The Fundamental Operations for this process is shown in references A and B of Table 4-140.

A word is input to the J register from a peripheral during period BP5 or BP6 (PCP1* or PCP2*) by asserting the DIN* signal, leaving signal EXTK* unasserted. The data word is gated to the bus the same way as for an input to the K register. The J register is parallel enabled and clocked to load the word from the bus. The Fundamental Operation for this process is shown in references A and C of Table 4-140.

A word is output from the K register to a peripheral during period BP5 of BP6 (PCP1* or PCP2*) by asserting the EXTK* signal and leaving signal DIN* unasserted. The

Table 4-140. Input To K or J Register, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP5 or BP6	EXT00*-EXT11*→B00-B11 1. ↓DIN*→↑DIN/6B3 2. (↑I/O) (↑DIN) (↑BP56)→↓IOSLU*/8A3 3. ↑LDXT0* + ↓IOSLU* + ↑LDXT3* + ↑LDXPR→↑LDEXT/8A3 4. ↑ANDJK* + ↑SELU6* + ↓IOSLU* + ↑LDXT0*→↑SELU/8A3 5. ↑SELU→↓SLUMX*/8A4 6. ↑LDEXT→U00-U11 = EXT00-EXT11/sheets 14, 15, 16 7. ↑STEX* + ↑SLJMX* + ↑SLKMX* + ↑SLPMX* + ↓SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2 8. ↑MXEN→↓MXEN*/9A2 9. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2 10. ↑SLRMX* + ↑EX1 + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2 11. ↓SLUMX* + ↑SLKMX* + ↑EX0* + ↑SLSMX*→↑MXS0/9B2 12. (↓MXEN*) (↓MXS2) (↓MXS1) (↑MXS0)→MX00-MX11 = U00-U11/sheets 14, 15, 16 13. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13 14. (↓CIN) (TS00-TS11 = 0000_8) B00-B11 = MX00-MX11/sheets 14, 15, 16	The data word is gated from the I/O bus through the adders to the processor bus.
B.	BP5 or BP6	B00-B11→KR 1. ↓EXTK*→↑EXTK/9A2 2. (↑EXTK) (↑DIN) (↑I/O)→↓IOKIN*/9A2 3. ↓IOKIN* + ↑OP231*→↑PEK5 +6/9A3 4. (↑PEK5 +6) (↑BP56)→↓PEK*/9A3 5. ↓PEK* + ↑HWM6* + ↑HWM4*→↑PEK/9A4 6. (↑PEK) (↑REGCLK)→↓CPK*/9A4 7. (↓PEK*) (↓CPK*)→K00-K11 = B00-B11/sheets 14, 15, 16	The data word on the bus is loaded into the K register.
C.	BP5 or BP6	B00-B11→JR 1. (↑DIN) (↑I/O) (↑EXTK*)→↓IOJIN*/9A2 2. ↓IOJIN* + ↑OP235*→↑PEJ56 3. (↑PEJ56) (↑BP56)→↓PEJ* 4. ↓PEJ* + ↑HWM6* + ↑HWM4*→↑PEJ 5. (↑PEJ) (↑REGCLK)→↓CPJ* 6. (↓PEJ*) (↓CPJ*)→J00-J11 = B00-B11/sheets 14, 15, 16	The data word on the bus is loaded into the J register.

contents of the K register signals (K00 through K11) are gates through the MX multiplexer and the adders to the bus (B00 through B11). From the bus this data word is gated to signals OUT00* through OUT11* of the I/O bus. The Fundamental Operations for this process are shown in references A and C of Table 4-141.

A word is output from the J register to a peripheral during period BP5 or BP6 (PCP1* or PCP2*) by leaving both DIN* and EXTK* signals unasserted. The contents of the J register are gated through the MX multiplexer and the adders to the bus (signals B00 through B11). From the bus this data word is gated to bus lines OUT00 through OUT11 of the I/O bus. The Fundamental Operations for this process are shown in references B and C of Table 4-141.

4.6.5.5 AUTOLOADER. The autoloader hardware provides the capability to load the memory of the ND812 from a peripheral information storage device without program control while the computer is in a halted condition. Autoloading circuitry associated with the interface of a storage device works in conjunction with the autoloading circuitry of the processor to control the transfer of data from that storage device to the memory. Each peripheral device must have autoloader hardware uniquely designed for it in order to perform the autoloading function.

The operation of loading the memory with an autoloader is analogous to loading the memory using the front panel switches. The data paths for loading addresses and data are the same for both methods except that for autoloading the data enters via bus signals EXT00* through EXT11* instead of the switch register inputs. The autoloader also features the capability to automatically generate a checksum for error detection.

The autoloading facilities of the processor are activated by pulling signal ALODT* of the I/O bus low. Assertion of this signal causes the switch register inputs (SW00* through SW11*) to be disabled and enables the external inputs (signals EXT00* through EXT11*). The switch register is disabled by blocking assertion of signal LDPSW*. This signal normally generates the LDFPS signal that gates the switch register into the MX multiplexer whenever the LOAD AR or LOAD MR switch is depressed.

The checksum facility is activated by pulling signal TSTD* low. Whe TSTD* is asserted, each twelve-bit address and data word loaded into the computer is added to the J register. At the start of the autoloading operation, the J register must be cleared (set to zero) in order that the final sum be valid. Overflows from this running addition are ignored. The last twelve-bit word to be loaded into the computer by the autoloading process is equal to the 2's complement of the twelve-bit sum that should be in the J register if the data transferred was all correct. If the contents of the J register is zero after the last data transfer, it is unlikely that any of the data was transferred incorrectly and the data loaded is assumed valid.

If a checksum is to be generated, autoloading hardware clears the J register by asserting signal EXCLJ*. This occurs asynchronously as shown in reference A of Table 4-142.

An autoloader loads addresses by presenting them as signals EXT00* through

Table 4-141. Output K or J Register, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	BP5 or BP6	KR→MX00-MX11 1. (↑DIN*) (↑EXTK) (↑I/O)→↓IOKOT*/12A4 2. ↓IOKOT*→↑IOKOT/12A4 3. (↑IOKOK) (↑BP56)→↓SLKMX*/12A4 4. ↑STEX* + ↑SLJMX* + ↓SLKMX* + ↑SLPMX* + ↑SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2 5. ↑MXEN→↓MXEN*/9A2 6. ↑SLJMX* + ↑EX2* + ↑SLRMX* + ↓SLKMX*→↑MXS2/9A2 7. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2 8. ↑SLSMX* + ↑EX0* + ↓SLKMX* + ↑SLUMX*→↑MXS0/9B2 9. (↓MXEN) (↑MXS2) (↓MXS1) (↑MXS0)→ MX00-MX11 = K00-K11/sheets 14, 15, 16	The contents of the K register are gated through the MX multiplexer to the adders.
B.	BP5 or BP6	JR→MX00-MX11 1. (↑DIN*) (↑I/O) (↑EXTK*) (↑BP56)→ ↓CIOJ, ↑EIOJ/12A3 2. (↓CIOJ) (↑EIOJ)→↓SLJMX*/12A3 3. ↑STEX* + ↓SLJMX* + ↑SLKMX * + ↑SLPMX* + ↑SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2 4. ↑MXEN→↓MXEN*/9A2 5. ↓SLJMX* + ↑EX2* + ↑SLRMX* + ↑SLKMX*→↑MXS2/9A2 6. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2 7. ↑SLSMX* + ↑EX0* + ↑SLKMX* + ↑SLUMX*→↓MXS0/9B2 8. (↓MXEN*) (↑MXS2) (↓MXS1) (↓MXS0)→ MX00-MX11 = J00-J11/sheets 14, 15, 16	The contents of the J register are gated through the MX multiplexer to the adders.
C.	BP5 or BP6	MX00-MX11→OUT00-OUT11 1. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13 2. (↓CIN) (TS00-TS11 = 0000_8) B00-B11 = MX00-MX11/sheets 14, 15, 16 3. (↑BP56) (↑I/OOSC) (↑DIN*)→↓IOOUT*/6B3 4. ↑OUTST* + ↓IOOUT*→↑OUTST/6B3 5. ↑OUTST→OUT00-OUT11 = B00-B11/Sheets 14, 15, 16	The output of the MX multiplexer is gated out on I/O bus signals OUT00-OUT11.

EXT11* and asserting signal XLDPR*. The twelve-bit address is gated through the utility gates to the MX multiplexer (see reference B of Table 4-142). From there, the data path is the same as in loading an address from the front panel switches, the address being loaded into the program counter and address register. During period PU5, the address is added to the checksum (see reference C of Table 4-142). Signals EXT00* through EXT11* are still

Table 4-142. Autoloading, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	ASYNCH.	0000_8→JR	
		1. ↓EXCLJ*→↑EXCLJ/9B2	The J register is cleared by assertion EXCLJ*.
		2. (↑EXCLJ) (↑GO*)→↓MRJ*/9B2	
		3. ↓MRJ*→J00-J11 = 0000_8/sheets 14, 15, 16	
B.	ASYNCH.	EXT00-EXT11→U00-U11	
		1. ↓XLDPR*→↓LDPR* (See Front Operation Section)	The twelve-bit word on the external inputs is gated through the utility gates.
		2. ↓LDPR*→↑LDPR/6A2	
		3. (↑LDPR) (↑ALODT)→↓LDXPR*/8A3	
		4. ↓LDXPR* + ↑LDXT3* + ↑IOSLU* + ↑LDXT0*→↑LDEXT/8A3	
		5. ↑LDEXT→U00-U11 = EXT00-EXT11	
		6. Set P00-P11, A00-A11 = U00-U11 (See Front Panel Operation)	
C.	PU5	JR + (U00-U11)→JR	
		1. ↑LDPR→↓SELU6*/8A3	The input address is added to the checksum from the memory data register.
		2. ↓SELU6* + ↑ANDJK* + ↑IOSLU* + ↑LDXT0*→↑SELU/8A3	
		3. ↑SELU→↓SLUMX*/8A4	
		4. ↑STEX* + ↑SLJMX* + ↑SLKMX* + ↑SLPMX* + ↓SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2	
		5. ↑MXEN→↓MXEN*/9A2	
		6. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2	
		7. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2	
		8. ↓SLUMX* + ↑EX0* + ↑SLKMX* + ↑SLSMX*→↑MXS0/9B2	
		9. (↓MXEN*) (↓MXS2) (↓MXS1) (↑MXS0)→MX00-MX11 = U00-U11/sheets 14, 15, 16	
		10. ↓TSTD*→↑TSTD/12B2	
		11. (↑TSTD) (↑GO*) (↑ALODT)→↓ALJTS*/12B2	
		12. ↓ALJTS*→↑ALJTS/12B3	
		13. (↑PU5B) (↑ALJTS) (↑LDPR)→↓SLJTS*	
		14. ↓SLJTS* + ↑SLKTS*→↑TSS1/13A4	
		15. ↓SLJTS* + ↑SLITS*→↑TSS0/13B4	
		16. (↑TSS0) (↑TSS1)→Z00-Z11 = J00-J11/sheet 13	
		17. ↓ALJTS* + ↑CSALT* + ↑HWDRL* + ↑IDEPH*→↑TSEXT/13B1	
		18. (↑TSEXT) (↑PU45)→↓TSADD*/13B2	
		19. (↓TSADD) (↑TSSUB)→TS00-TS11 = Z00-Z11/sheet 13	
		20. (↓CIN) (MX00-MX11 = U00-U11) (TS00-TS11 = J00-J11)→B00-B11 = (J00-J11) + (U00-U11)/sheets 14, 15, 16	

Table 4-142. Autoloading, Fundamental Operations (Cont'd.)

Ref.	Period	Simplified Logic	Event
		21. (↑ALJTS) (↑PU5B)→↓CPEJ, ↑EPEJ/9A3	
		22. (↓CPEJ) (↑EPEJ)→↓PEJ*/9A3	
		23. ↓PEJ* + ↑HWM6* + ↑HWD4*→↑PEJ/9A4	
		24. (↑PEJ) (↑REGCLK)→↓CPJ*/9A4	
		25. (↓PEJ*) (↓CPJ*)→J00-J11 = B00-B11/sheets 14, 15, 16	
D.	PU3	EXT00-EXT11→U00-U11	
		1. ↓XLDMR*→↓LDMR* (See Front Panel Operation Section)	The data word presented on signals EXT00*-EXT11* are gated through the utility gates.
		2. ↓LDMR*→↑LDMR/6B2	
		3. (↑LDMR) (↑ALODT)→↓ALDMR*/8A2	
		4. ↓ALDMR* + ↑CSRDB*→↑XTMRCS/8A3	
		5. (↑XTMRCS) (↑PU3B)→↓LDXT3*	
		6. ↓LDXT3* + ↑LDXT0* + ↑IOSLU* + ↑LDXPR*→↑LDEXT/8A3	
		7. ↑LDEXT→U00-U11 = EXT00-EXT11	
		8. Set M00-M11 = U00-U11 (See Front Panel Operation Section)	
E.	PU5	JR + (U00-U11)→JR	
		1. ↓TSTD*→↑TSTD/12B2	
		2. (↑TSTD) (↑GO*) (↑ALODT)→↓ALJTS*/12B2	
		3. ↓ALJTS*→↑ALJTS/12B3	
		4. (↑ALJTS) (↑PU5B) (↑LDPR*)→↓CIOJ, ↑EIOJ/12A3	
		5. (↓CIOJ) (↑EIOJ)→↓SLJMX*/12A3	
		6. ↑STEX* + ↓SLJMX* + ↑SLKMX* + ↑SLPMX* + ↑SLUMX* + ↑SLAMX* + ↑SLSMX* + ↑SLRMX*→↑MXEN/9A2	
		7. ↑MXEN→↓MXEN*/9A2	
		8. ↓SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↑MXS2/9A2	
		9. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2	
		10. ↑SLSMX* + ↑EX0* + ↑SLKMX* + ↑SLUMX*→↓MXS0/9B2	
		11. (↓MXEN*) (↑MXS2) (↓MXS1) (↓MXS0)→MX00-MX11 = J00-J11/sheets 14, 15, 16	
		12. (↓TSS1) (↓TSS0)→Z00-Z11 = M00-M11/sheet 13	
		13. ↑CSALT* + ↓ALJTS* + ↑HWDRL* + ↑IDEPH*→↑TSEXT/13B1	
		14. (↑TSEXT) (↑PU45)→↓TSADD*/13B2	
		15. (↓TSADD*) (↑TSSUB*)→TS00-TS11 = Z00-Z11	
		16. (↓CIN) (TS00-TS11 = M00-M11) (MX00-MX11 = J00-J11)→B00-B11= (J00-J11) + (M00-M11)/sheets 14, 15, 16	
		17. (See D. 21-25)	

gated through the utility gates at that time and are presented by the MX multiplexer to the adder. The J register contents are gated through the TS multiplexer to the adders. The adders, in response, present the sum of the address and the J register contents on bus B00 through B11. The J register is parallel enabled and clocked to store this new sum.

When an address is loaded in the above manner, the memory field specification can be altered by the autoloading hardware in the same manner as it is controlled for front panel operation. Because the memory field selection switches on the front panel are not disabled during autoloading, the autoloader has full control over memory field selection only if these switches are set for selection of memory field 0.

An autoloader loads twelve-bit words into memory be presenting them as signals EXT00* through EXT11* and asserting signal XLDMR*. The first effect of asserting XLDMR* is the transfer of the contents of the program counter to the address register (at period PU0) followed by (at period PU1) incrementation of the program counter. Then during period PU3, the memory data register is loaded with the data word. Signals EXT00* through EXT11* are gated through the utility gates to the MX multiplexer (see reference D of Table 4-142). From there the loading process is the same as for front panel operation (see Front Panel Section).

During period PU5, the data word is added to the checksum (see reference E of Table 4-142). The contents of the J register are gated through the MX multiplexer to the adders. The contents of the memory data register, which contains the newly received data word, are gated through the TS multiplexer to the adders. The adder, in response, presents the sum of the data word and the J register contents on bus B00 through B11. The J register is parallel enabled and clocked to store this new sum.

4.6.6 DIRECT MEMORY ACCESS, CYCLE STEAL

The direct memory access (DMA) logic of the ND812 allows a peripheral device to manipulate the contents of the memory by stealing cycles without intervention by the processor and without interferring with the status of the processor. There are four kinds of cycle steals, two for data transfers (data in and data out) and two for housekeeping functions (increment and decrement). In addition, there are multiple cycle steals which can perform these operations on consecutive locations of memory during successive memory cycles.

Cycle steals are permitted by the processor whenever the computer is running and execution of an instruction has been completed. At that time a cycle steal request takes precedence over all other demands on the processor. Of the process registers, only the address and memory data registers are used during a cycle steal; the content of the other registers are unaffected. After the cycle steal is completed, normal processor functions are resumed.

4.6.6.1 SINGLE, DATA-IN CYCLE STEAL. The single data-in cycle steal, transfers a single twelve-bit word of data from a peripheral device to a location in memory specified by the peripheral device. It performs this function during the time of a single memory cycle.

To generate this type of cycle steal, the peripheral device pulls signals CSR* (Table 4-143) and DIN* low while leaving signals ALTER* and SDCS* high (Figure 4-109).

A peripheral device requests a cycle steal by pulling signal CSR* low. When the current instruction has completed execution and the DONE signal goes high at the trailing edge of pulse PU6, a low CSR* signal causes the cycle-steal latch to generate high CSP and CSFF signals. These two signals generate a low EXCSP* signal, and a high INTCSF signal. The high INTCSF signal results in the generation of low BLKE* and BLKB* signals that inhibit the generation of the BP and EP timing signals. The Fundamental Operations for these events is shown in reference A of Table 4-143 and their timing is illustrated by Figure 4-109.

The low EXCSP* signal tells the peripheral device that the cycle steal has been permitted, and the address of the location in memory to be accessed may be presented and transferred onto buses EXT00* through EXT11* from the peripheral device. At period PU0, the address represented by signals EXT00* through EXT11* is gated through the utility gates, the MX multiplexer, and the adders to bus signals B00 through B11. The address register is parallel enabled and clocked to load it with the address. The Fundamental Operations for these events is shown in Table 4-143 and their timing is shown by Figure 4-109.

At period PU1, signals CSP and EXCSP* are reset low and high, respectively. The high EXCSP* signal tells the peripheral device to stop presenting the start address and start presenting the data word to be transferred as signals EXT00* through EXT11*. Then at period PU3, the memory data register is loaded with this data word via the utility gates, MX multiplexer, adders, and bus and memory buffer multiplexer. At period PU5, the data word is stored in the memory. The Fundamental Operations for these events is shown in references C and D of Table 4-143, and their timing is illustrated by Figure 4-109.

Finally, at period PU7, the cycle steal ends and CSFF is reset low. Then, at the start of pulse PU0, signals BLKE* and BLKB* are reset high to restart generation of basic phase and execute phase signals. The Fundamental Operations for these events is shown in references E and F of Table 4-143 and their timing is illustrated by Figure 4-109.

4.6.6.2 SINGLE, DATA-OUT CYCLE STEAL. The single data-out cycle steal transfers a single twelve-bit word of data to a peripheral from a location in memory specified by the peripheral device. It performs this function in the time of a single memory cycle. To generate this type of cycle steal, the peripheral device pulls signal CSR* low while leaving signals DIN*, ALTER*, and SDCS* high (Figure 4-110).

A data-out cycle steal is initiated and the address to be accessed is loaded into the address register as in the data-in cycle steal. At period PU1, signals CSCIN and IOMST are set high. The high IOMST signal gates the contents of the memory data register out onto buses IOM00 through IOM11. The Fundamental Operations for these events are shown in reference A of Table 4-144 and their timing is illustrated by Figure 4-110.

Table 4-143. Single, Data-In Cycle Steal, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	PU6	SET ↑CSFF, ↓EXCSP*, ↓BLKE*, ↓BLKB* 1. ↓CSR*→↑CSR/4B1 2. (↑CPPU) (↑DONE) (↑PU6B) (↑GO)→ ↓FFCLK*/3A2 3. (↑HWMD*) (↑CSR) (↑STPU1*) (↑CNTRP) (↓FFCLK*)→↑CSP/4B2, ↓CSP*/4B2 4. ↓CSP*→↑CSFF/4B2, ↓CSFF*/4B2 5. (↑CSP) (↑CSFF)→↓EXCSP*/4B2 6. ↑INTFF* + ↓CSFF*→↑INTCSF/4B4 7. (↑INTCSF) (↑PU7) (↓CPPU*) (↑OSCPUL)→↑BLKBEP/5B2 8. ↑BLKBEP→↓BLKE*/5B3 9. ↑BLKBEP→↓BLKB*/5B3	Signal CSFF is set high to mark the initiation of the cycle steal cycle. Signal EXCSP* indicates this on the I/O bus. Signals BLKE*, BLKB* are asserted to block the execute and basic phases.
B.	PU0	EXT00-EXT11*→AR 1. ↓CSP* + ↑INTFF*→↑INTCSP/8A3 2. (↑INTCSP) (↑PU0B)→↓LDXT0*/8A3 3. ↓LDXT0* + ↑IOSLU* + ↑LDXT3* + ↑LDXPR*→↑LDEXT/8A3 4. ↑ANDJK* + ↑IOSLU* + ↑SELU6* + ↓LDXT0*→ ↑SELU/8A3 5. ↑SELU→↓SLUMX*/8A4 6. ↑LDEXT→U00-U11 = EXT00-EXT11/sheets 14, 15, 16 7. ↓SLUMX*→↑MXEN/9A2 8. ↑MXEN→↓MXEN*/9A2 9. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2 10. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2 11. ↓SLUMX* + ↑SLKMX* + ↑EX0* + ↑SLSMX*→↑MXS0/9B2 12. (↓MXEN*) (↓MXS2) (↓MXS1) (↑MXS0)→ MX00-MX11 = U00-U11/sheets 14, 15, 16 13. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13 14. (↓CIN) (TS00-TS11 = 0000_8)→ B00-B11 = MX00-MX11/sheets 14, 15, 16 15. ↑PU0B→↓PEA*/9B1 16. ↓PEA*→↑PEA/9B1 17. (↑PEA) (↑REGCLK)→↓CPA*/9B2 18. (↓CPA*) (↓PEA*)→A00-A11 = B00-B11/sheets 14, 15, 16	The address on signals EXT00*-EXT11* are gated to and loaded into the address register (A00-A11).

Table 4-143. Single, Data-In Cycle Steal, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
C.	PU1	RESET ↓CSP, ↑EXCSP* 1. ↓PU1* + ↑EXMPR* + ↑STCLR*→ ↑STPU1/3A3 2. ↑STPU1→↓STPU1*/3A3 3. ↓STPU1*→↓CSP/4B2, ↑CSP*/4B2 4. (↑CSFF) (↓CSP)→↑EXCSP*/4B2	Signals CSP and EXCSP* are reset.
D.	PU3	EXT00*-EXT11*→MDR 1. (↑DIN) (↑CSFF) (↑ALTER*)→ ↓CSRDB*/12B1 2. ↓CSRDB* + ↑INTFF* + ↑LDMR*→ ↑RDB3/12B2 3. (↑RDB3) (↑PU3B)→↓BMEM*/12B2 4. ↓BMEM*→↑RDB/12B3 5. ↑RDB→↓RDB*/12B3 6. ↓BMEM* + ↓PU3→↑PEM/12B3 7. ↑PEM→↓PEM*/12B3 8. (↑PEM) (↑REGCLK)→↓CPM*/12B3 9. ↓CSRDB* + ↑ALDMR*→↑XTMRCS/8A3 10. (↑XTMRCS) (↑PU3B)→↓LDXT3*/8A3 11. ↑LDXT0* + ↑IOSLU* + ↓LDXT3* + ↑LDXPR*→↑LDEXT/8A3 12. ↓CSRDB* + ↑LDMR*→↑SELU3/8A3 13. (↑SELU3) (↑PU3B)→↓SLUMX*/8A4 14. (See B. 6-14) 15. ↓RDB*→PMR00-PMR11 = B00-B11/sheet 17 16. (↓PEM*) (↓CPM*)→M00-M11 = PMR00-PMR11/sheet 17	The data on signals EXT00*-EXT11* is loaded into the memory data register (M00-M11) via the bus and MBR multiplexer.
E.	PU7	RESET ↓CSFF 1. (↑PU7B) (↑CPPU)→↓RDBLE*/4A4 2. ↓RDBLE* + ↑GO→↑RDBLE/4B1 3. (↑CSP*) (↑RDBLE) (↑SDCS*)→ ↓RCSFF*/4B2 4. ↓RCSFF*→↓CSFF/4B2, ↑CSFF*/4B2	The cycle steal flip-flop is reset to mark the end of the cycle steal cycle.
F.	PU0	RESET ↑BLKE*, ↑BLKB* 1. ↑INTFF* + ↑CSFF*→↓INTCSF/4B4 2. ↓INTCSF→↑INTCS*/5B2 3. (↑INTCS*) (↑PU0) (↑GO)→ ↓RBLK*/5B2 4. ↓RBLK*→↓BLKBEP/5B2 5. ↓BLKBEP→↑BLKE*/5B3 6. ↓BLKBEP→↑BLKB*/5B3	The basic and execute phase are unblocked to resume execution of program instructions.

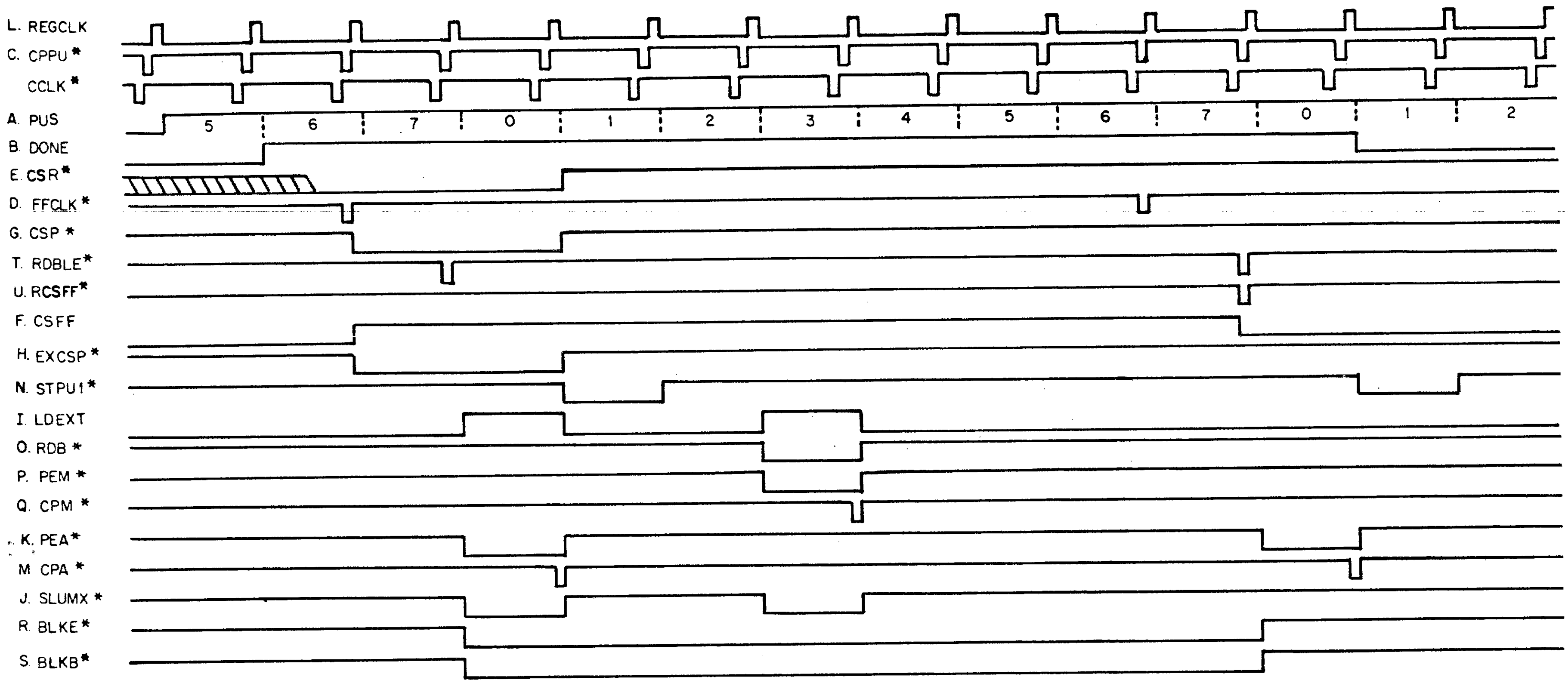


Figure 4-109. Single, Data-In Cycle Steal, Timing Diagram

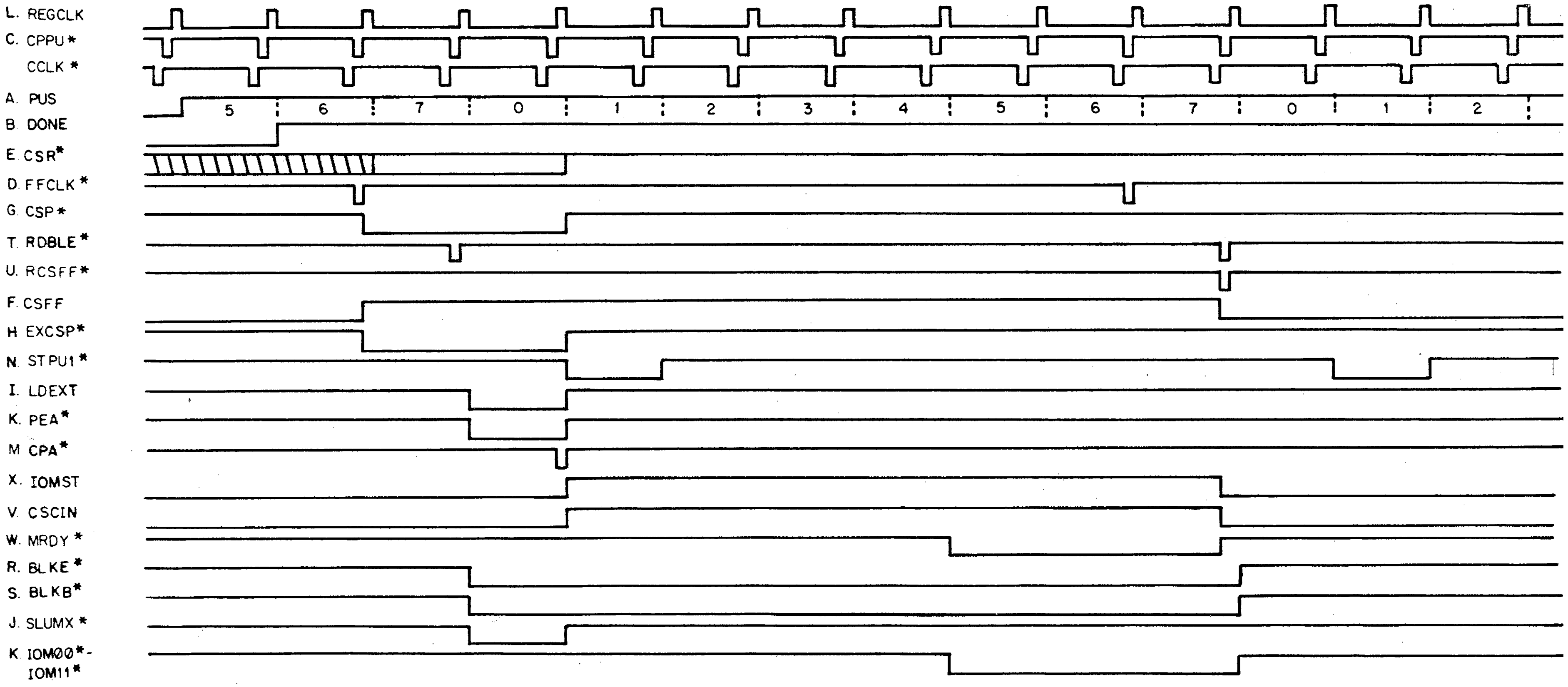


Figure 4-110. Single, Data-Out Cycle Steal, Timing Diagram

Table 4-144. Single, Data-Out Cycle Steal, Fundamental Operation

Ref.	Period	Simplified Logic	Event
		(See Table 4-143, Reference A-C, for cycle steal initiation and address loading)	
A.	PU1	MDR→IOM00*-IOM11* 1. (↑CSFF) (↑CSP*)→↓CSCIN*/12A1 2. ↓CSCIN*→↑CSCIN/7B3 3. (↑DIN*) (↑CSCIN) (↑ALTER*)→ ↓CSIOM*/6B3 4. ↓CSIOM* + ↑I/OCOM + ↑ALODT*→ ↑IOMST/6B3 5. ↑IOMST→IOM00-IOM11 = M00-M11/sheet 17	The contents of the memory data register (M00-M11) are gated out on I/O bus signals IOM00-IOM11.
B.	PU5, PU6, PU7	SET ↓MRDY* 1. (↑PU567) (↑CSCIN)→↓MRDY*/7B4	Signal MRDY* is asserted on the I/O bus.

The data word to be transferred is read out of memory at period PU2 and appears in the memory data register at the trailing edge of period PU3. During periods PU5, PU6, and PU7, a low MRDY* signal notifies the peripheral device that the data is ready on buses IOM00 through IOM11. The Fundamental Operations for these events is shown in reference B of Table 4-144 and their timing is shown by Figure 4-110. The data-out cycle steal is terminated in a manner similar to that for the data-in cycle steal.

4.6.6.3 SINGLE, INCREMENT CYCLE STEAL. The single increment cycle steal increments the contents of a location in memory specified by a peripheral device. It performs this function in the time of a single memory cycle. To generate this type of cycle steal, a peripheral device must pull signals CSR*, ALTER*, and DIN* low, leaving SDCS* high (Figure 4-111).

An increment cycle steal is initiated and the address of the location in memory to be accessed is loaded in a manner similar to the data-in cycle steal.

At period PU4, the contents of the memory data register are incremented. This is accomplished by gating the contents of the memory data register through the TS multiplexer to the adder and generating a carry-in (CIN) signal, resulting in presentation on the bus (signals B00 through B11) of the incremented contents of the memory data register. The data word on the bus is gated to the memory data register through the bus and memory buffer multiplexer. The memory data register is then parallel enabled and clocked to load the incremented word. The Fundamental Operation description for these events is shown in reference A of Table 4-145 and their timing is illustrated in Figure 4-111.

At period PU5, the contents of the memory data register are tested to see if they are equal to 0000_8. If they are, a low CSMRZ* signal is generated. This signal remains

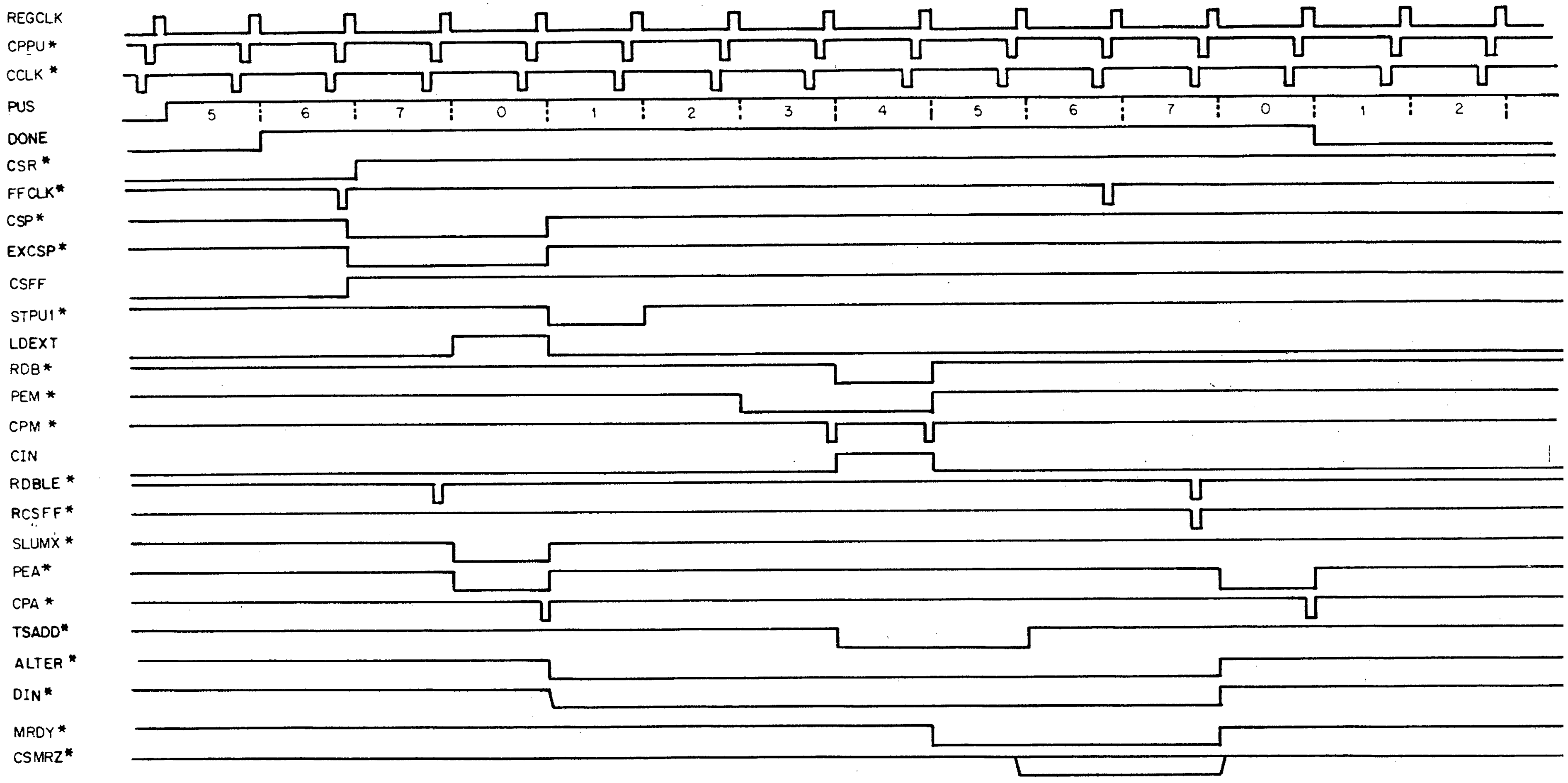


Figure 4-111. Single, Increment Cycle Steal, Timing Diagram

Table 4-145. Single, Increment Cycle Steal, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	PU4	MDR +1→MDR 1. (↓TSS0) (↓TSS1)→Z00-Z11 = M00-M11/sheet 13 2. ↓ALTER*→↑ALTER/6B3 3. (↑CSFF) (↑ALTER)→↓CSALT*/13B1 4. ↓CSALT*+ ↑ALJTS* + ↑HWDRL* + ↑IDEPH→↑TSEXT/13B1 5. (↑TSEXT) (↑PU45)→↓TSADD*/13B2 6. (↓TSADD*) (↑TSSUB*)→TS00-TS11 = Z00-Z11/sheet 13 7. ↑MXEN*→MX00-MX11 = 0000_8/sheets 14, 15, 16 8. (↑CSFF) (↑PU4B) (↑ALTER) (↑DIN)→ ↓CCIN1/12B1, ↑ECIN1/12B1/12B2 9. (↓CCIN1) (↑ECIN1)→↓CIN1*/12B2 10. ↓CIN1* + ↑CIN3* + ↑CIN2*→↑CIN/12B2 11. (↑CIN) (TS00-TS11 = Z00-Z11) (MX00-MX11 = 0000_8)→B00-B11 = (Z00-Z11) +1/sheets 14, 15, 16 12. (↑PU4B) (↑CSFF) (↑ALTER)→↓BMEM*/12B2 13. ↓BMEM*→↑RDB/12B3 14. ↑RDB→↓RDB*/12B3 15. ↓RDB*→PMR00-PMR11 = B00-B11/sheet 17 16. ↓BMEM* + ↑PU3*→↑PEM/12B3 17. ↑PEM→↓PEM*/12B3 18. (↓PEM) (↑REGCLK)→↓CPM*/12B3 19. (↓PEM*) (↓CPM*)→M00-M11 = PMR00-PMR11/sheet 17	The contents of the memory data register (M00-M11) are incremented and placed in the memory data register again.
B.	PU5	SET ↓CSMRZ* (ONLY IF MDR = 0000_8) 1. (↓TSS0) (↓TSS1)→Z00-Z11 = M00-M11 (=0000_8)/sheet 13 2. (↑CSFF) (↑ALTER)→↓CSALT*/13B1 3. ↓CSALT* + ↑ALJTS* + ↑HWDRL* + ↑IDEPH→↑TSEXT/13B1 4. (↑TSEXT) (↑PU45)→↓TSADD*/13B2 5. (↓TSADD*) (↑TSSUB)→TS00-TS11 = Z00-Z11 (=0000_8)/sheet 13 6. ↑MXEN*→MX00-MX11 = 0000_8/sheets 14, 15, 16 7. (↓CIN) (TS00-TS11 = Z00-Z11) (MX00-MX11 = 0000_8)→B00-B11 = Z00-Z11 (=0000_8)/sheets 14, 15, 16 8. B00-B11 = 0000_8→↓BZ007*/7A3, ↓BZ811*/7A3 9. ↓BZ007*→↑BZ007/7A4 10. ↓BZ811*→↑BZ811/7A4 11. (↑BZ007) (↑BZ811)→↓BZERO*/7A4 12. ↓BZERO*→↑BZERO/7A3	If the contents of the memory data register (M00-M11) equal zero, signal CSMRZ* is asserted.

Table 4-145. Single, Increment Cycle Steal, Fundamental Operation (Cont'd.)

Ref.	Period	Simplified Logic	Event
		13. ↓CSFF* + ↑SETBZ*→↑BZLOG/7A3	
		14. (↑BZERO) (↑BZLOG) (↑PU5B) (↑REGCLK)→↓SBZFF*/7A4	
		15. ↓SBZFF*→↑BZFF/7A4	
		16. (↑CSFF) (↑CSP*)→↓CSCIN*/12A1	
		17. ↓CSCIN*→↑CSCIN/7B3	
		18. (↑BZFF) (↑CSCIN)→↓CSMRZ*/7A4	
C.	PU0	RESET ↓BZFF	
		1. (↓PU0*) (↑SBZFF*)→↓BZFF/7A4	If the zero-bus flip-flop was set at PU5, it is now reset.

low until the CSFF latch is reset during period PU7. The Fundamental Operations for these events is shown in reference B of Table 4-145 and their timing is illustrated by signals of Figure 4-111.

Signal BZFF which is instrumental in generating the asserted CSMRZ* signal, is reset as explained in reference C of Table 4-145.

4.6.6.1 SINGLE, DECREMENT CYCLE STEAL. The single decrement cycle steal decrements the location in memory specified by a peripheral device. To generate this type of cycle steal, a peripheral device must pull signals CSR* and ALTER* low, leaving DIN* and SDCS* high (Figure 4-112). A decrement cycle steal is initiated, and the address of the location in memory to be accessed is loaded in a manner similar to the data-in cycle steal section.

At period PU4, the contents of the memory data register are decremented by adding 7777_8 (-1) to them. The contents of the memory data register are gated through the TS multiplexer and add/subtract gates to the adders. At the same time, 7777_8 is presented on buses MX00 through MX11 to the adders from the MX multiplexer. In response, the adders present the decremented contents of the memory data register on bus B00 through B11. This data word is gated through the bus and memory buffer multiplexer to the memory data register which is parallel enabled and loaded by a clock pulse with the decremented value. Fundamental Operations for these events are shown in Table 4-146.

As in the case of the increment cycle steal, the altered contents of the memory data register are checked to see if they are equal to 0000_8 in a manner similar to the increment cycle steal.

4.6.6.5 MULTIPLE CYCLE STEALS. When desired, a cycle steal can be prolonged through two or more consecutive cycles, performing a single cycle steal function (single-word data transfer, increment, or decrement) during each cycle. Multiple cycle steals can be produced in two different ways.

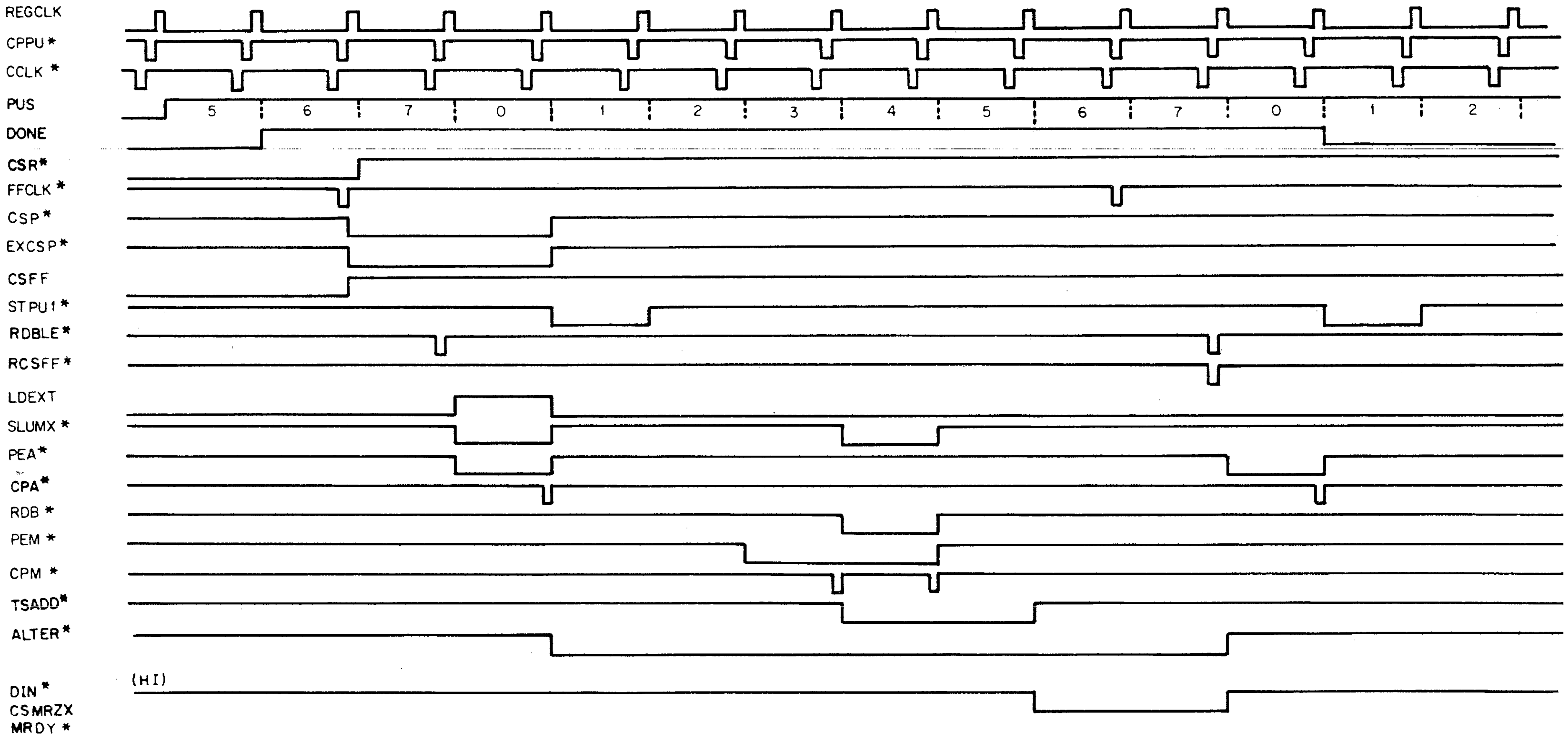


Figure 4-112. Single, Decrement Cycle Steal, Timing Diagram

Table 4-146. Single, Decrement Cycle Steal, Fundamental Operation

Ref.	Period	Simplified Logic	Event
A.	PU4	MDR -1→MDR 1. (↓TSS0) (↓TSS1)→Z00-Z11 = M00-M11/sheet 13 2. ↓ALTER*→↑ALTER/6B3 3. (↑CSFF) (↑ALTER) →↓CSALT*/13B1 4. ↓CSALT* + ↑ALJTS* + ↑HWDRL* + ↑IDEPH→↑TSEXT/13B1 5. (↑TSEXT) (↑PU45)→↓TSADD*/13B2 6. (↓TSADD*) (↑TSSUB*)→TS00-TS11 = Z00-Z11/sheet 13 7. (↑ALTER) (↑DIN*) (↑CSFF) (↑PU4B)→ ↓CSELU/8A4, ↑ESELU/8A4 8. (↓CSELU) (↑ESELU)→↓SLUMX*/8A4 9. ↓SLUMX*→↑MXEN/9A1 10. ↑MXEN→↓MXEN*/9A1 11. ↑SLJMX* + ↑EX 2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2 12. ↑SLRMX* + ↑SLPMX* + ↑EX1*+ ↑SLSMX*→↓MXS1/9B2 13. ↓SLUMX* + ↑SLKMX* + ↑EX0* + ↑SLSMX*→↑MXS0/9B2 14. (↓LDFPSW) (↓LDEXT) (↓ANDJK) (↓LIT)→U00-U11→7777_8/sheets 14, 15, 16 15. (↓MXEN*) (↓MXS2) (↓MXS1) (↑MXS0)→ MX00-MX11 = U00-U11 (=7777_8)/sheets 14, 15, 16 16. (↓CIN) (TS00-TS11 = Z00-Z11) (MX00-MX11 = 7777_8)→B00-B11 = (Z00-Z11) -1/sheets 14, 15, 16 17. (↑PU4B) (↑CSFF) (↑ALTER)→ ↓BMEM*/12B2 18. ↓BMEM*→↑RDB/12B3 19. ↑RDB→↓RDB*/12B3 20. ↓RDB*→PMR00-PMR11/sheet 17 21. ↓BMEM* + ↑PU3*→↑PEM/12B3 22. (↑PEM) (↑REGCLK)→↓CPM*/12B3 23. (↓PEM*) (↓CPM*)→M00-M11 = PMR00-PMR11/sheet 17	The contents of the memory data register (M00-M11) are decremented and stored again in the memory data register.

See Table 4-145, B for CSMRZ* Generation.

1. If signal CSR* (Figure 4-113) is pulled low during period PU6 of any cycle steal cycle, a subsequent cycle steal will be initiated.

2. If signal SDCS* is pulled low for period PU7 of a cycle steal cycle, signal CSFF is not reset and the cycle steal continues for another cycle.

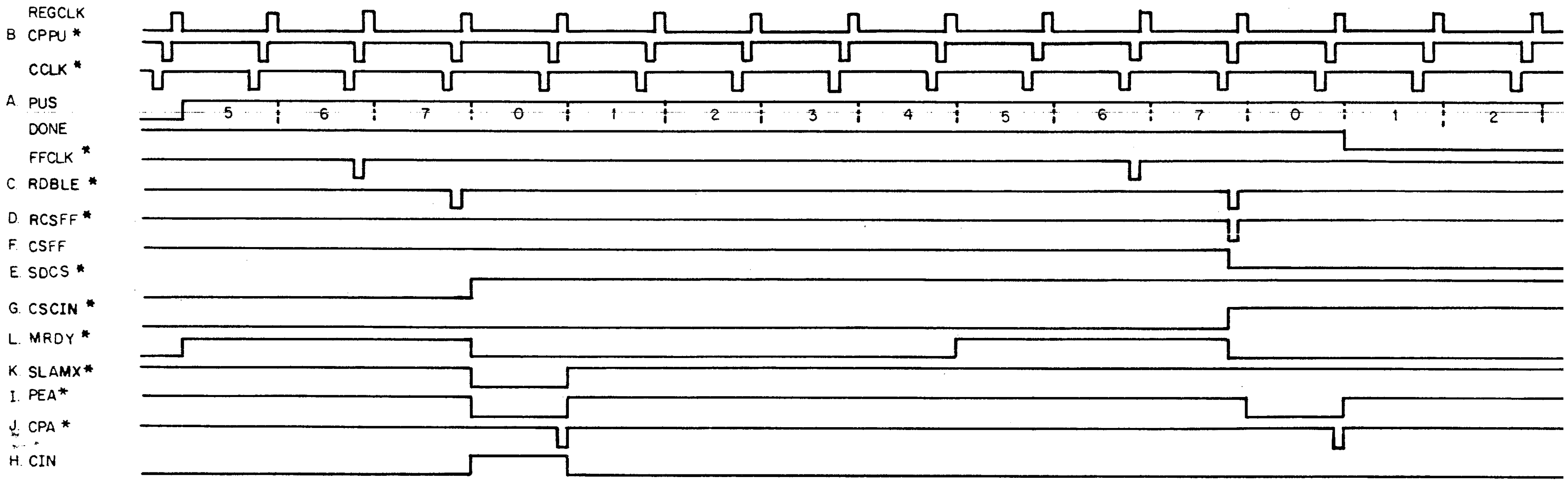


Figure 4-113. Multiple Cycle Steals, Timing Diagram

In addition to the difference in means of production of multiple cycle steals in these two cases, there is a difference in effect. If method 1 is employed a new address must be provided to the processor by the peripheral for each cycle. If method 2 is employed, after the initial cycle steal during which the address register is loaded, addresses for subsequent consecutive cycle steal cycles must be generated by incrementing the address register. Thus, in case 2 consecutive.

Since case 1 involves simply a concatenation of single cycle steals no detailed description of it is given here. For case 2, refer to Table 4-147 and Figure 4-113. The timing diagram (Figure 4-113) corresponds to the final cycle of a string of cycle steals generated as in case 2. The timing diagram for a concatenation of two cycle steals generated as in case 2 would show a repetition of the complete cycle as shown between the beginning of the initial period PU5 and the end of the subsequent period PU4.

The end of a cycle steal is marked by resetting signal CSFF at the end of period PU7. The sole function of signal SDCS is to prevent signal CSFF from being reset by blocking a negative pulse that normally would have occurred during signal RCSFF*. The Fundamental Operations description of this event is shown in reference A of Table 4-147 and timing is illustrated by signals A through E of Figure 4-113. Preventing signal CSFF from being reset during period PU7 commits the processor to another cycle-steal cycle.

At period PU0, with signal CSFF still asserted (high) and signal CSP remaining unasserted (low) the contents of the address register are incremented. The contents of the address register are gated to the adders along with 0000_8 from the add/subtract gates and a carry-in (CIN) signal is generated. In response, the adders present the complemented contents of the address register. The address register is then parallel enabled and clocked to load the data-word presented by the bus. The Fundamental Operations description for these events is shown in reference B of Table 4-147. Their timing is illustrated by Figure 4-113.

During periods PU5, PU6, and PU7, a low MRDY* signal is generated to signal the peripheral device that it may present a new data word on buses EXT00 through EXT11 or that a new data word is being presented on buses IOM00 through IOM11. See reference C of Table 4-147, Figure 4-113.

If neither signal CSR* nor SDCS* is pulled low at the appropriate time during periods PU6 and PU7, respectively, the cycle steal terminates in the same way any single-cycle steal terminates.

Table 4-147. Multiple Cycle Steals, Fundamental Operations

Ref.	Period	Simplified Logic	Event
A.	PU7	BLOCK RCSFF*	
		1. (↑PU7B) (↑CPPU)→↓RDBLE*/4A4 2. ↓RDBLE* + ↑GO→↑RDBLE/4B1 3. (↑RDBLE) (↑CSP*) (↓SDCS*)→ ↑RCSFF*/4B2	Signal RCSFF* is blocked to prevent the cycle steal flip-flop from being reset.
B.	PU0	AR +1→AR	
		1. (↑CSP*) (↑CSFF) (↑PU0B)→ ↓CSELA/8A1, ↑ESELA/8A1 2. (↓CSELA) (↑ESELA)→↓SLAMX*/8A2 3. ↓SLAMX→↑MXEN/9A2 4. ↑MXEN→↓MXEN*/9A2 5. ↑SLJMX* + ↑EX2* + ↑SLKMX* + ↑SLRMX*→↓MXS2/9A2 6. ↑SLRMX* + ↑EX1* + ↑SLPMX* + ↑SLSMX*→↓MXS1/9B2 7. ↑SLSMX* + ↑EX0* + ↑SLKMX* + ↑SLUMX*→↓MXS0/9B2 8. (↓MXEN*) (↓MXS0) (↓MXS1) (↓MXS2)→ MX00-MX11 = A00-A11 /sheets 14, 15, 16 9. (↑CSFF) (↑CSP*)→↓CSCIN*/12A1 10. ↓CSCIN* + ↑INTFF*→↑INTCIN/12A1 11. (↑INTCIN) (↑PU0B)→↓CIN2*/12A2 12. ↓CIN2* + ↑CIN1* + ↑CIN3*→↑CIN/12A2 13. (↑TSADD*) (↑TSSUB*)→TS00-TS11 = 0000_8/sheet 13 14. (↑CIN) (TS00-TS11 = 0000_8)→B00-B11 = (MX00-MX11) +1/sheets 14, 15, 16 15. ↑PU0B→↓PEA*/9B1 16. ↓PEA*→↑PEA/9B1 17. (↑PEA) (↑REGCLK)→↓CPA*/9B2 18. (↓CPA*) (↓PEA*)→A00-A11 = B00-B11/sheets 14, 15, 16	The contents of the address register are incremented and again stored in the address register.
C.	PU5	SET MRDY*	
		1. (↑PU567) (↑CSCIN)→↓MRDY*/7B4	Signal MRDY* is asserted on the I/O bus.

SECTION V
MAINTENANCE

5.1 GENERAL

This section contains instructions required to maintain ND812 Central Processor. The instuctions include preventive and corrective maintenance procedures. Preventive maintenance includes periodic inspection of the Central Processor. Corrective maintenance includes performance (diagnostic) tests/adjustments, troubleshooting procedures, and disassembly procedures.

5.1.1 MAINTENANCE PHILOSOPHY

The maintenance philosophy for the ND812 Central Processor is to locate a suspected malfunction utilizing the diagnostic tests listed in paragraph 5.3.1.1 and then troubleshoot and repair the ND812 Central Processor as necessary. Generally, the diagnostic routines exercise all the logic in the processor, I/O controller, and the memory control. When a malfunction is suspected, or the source of faulty operation of the ND812 is not apparent, the diagnostic routines should be run. These diagnostic routines should be run in the order given in the listing for each of the routines. If a malfunction occurs in the tested portion of the ND812 electronics, a programmed halt will occur. This programmed halt is indexed in each diagnostic routine listing so that reference to the content of the program counter when the ND812 is stopped will give the halt address. Generally, if other registers of the ND812 contain data as the result of processing to the stop instruction, the content of these registers is also given in the diagnostic listings.

If the programmed stop is recurrent, the source of the malfunction should be investigated. The diagnostics should not be regarded as absolute indicators of trouble in the ND812; they are simply tools that permit the field maintenance engineer to localize a suspect area. If, for example, a recurrent malfunction occurs, the instruction in question probably should be programmed in a jump-back loop and the logic examined for a defective integrated circuit component (see paragraph 5.3.1.3).

The information given in Section IV, Theory of Operation, describes the functions of each segment of the logic tested by the diagnostic routines. Thus, when a given instruction appears to fail recurrently, the description of the logic for that instruction should be referenced, together with the timing diagrams and the tables of Fundamental Operations.

5.1.2 TIMING DIAGRAMS

The timing diagrams in Section IV are not all inclusive; they do not show every signal generated by the logic when a given instruction is processed. However, they do provide insight as to those that should be checked first. If other signals must be checked refer to the Tables of Fundamental Operations for the signals in question. A diagram reference (Section VII) is specified in these tables following the slash. Turn to the indicated diagram to identify other signal sources from the logic which produce the reference signal. In this way, all signals for a given type of instruction can be checked. It should also be noted that the waveforms given in the timing diagrams are idealized and that seldom are output signals as sharp as they are drawn on the timing diagrams; however, the time relationships are accurate.

5.1.3 EVENT SUMMARIES

Each of the event summaries for each of the instructions described in Section IV indicate the process that takes place for the execution of each instruction. When an instruction process is under investigation, a short program should be input to obtain the data manipulations required. For example, if an ISZ (3400) instruction is under investigation, both the zero condition and the non-zero condition of the examined register should be looped to obtain continuous operation. Hence, to investigate the ISZ instruction, the following program could be loaded into the ND812.

Address	Operand	Symbology
0100	3403	ISZ.+3
0101	6101	JMP.-1
0102	6102	JMP.-2
0103	0000	(data)

Of course, this subprogram assumes that the jump instruction works and that memory control logic cycles data through the memory buffer and memory data registers. The above instructions can be single-stepped through the SINGLE STEP switch on the ND812 front panel. Referring to the Event Summary for the instruction in question will indicate the contents of the various registers for each phase of the instruction. Also, each instruction in the routine can be individually executed through the SINGLE INSTR switch on the ND812 front panel.

5.1.4 FUNDAMENTAL OPERATIONS

Each of the Tables of Fundamental Operations in Section IV list the functional process for each entry in the Event Summary tables. The Fundamental Operation tables list the logical functions required to process each event described by the Event Summary for the instruction in question. When signal checking through the logic, the event to be investigated should be identified from the Event Summary Table. Then, the reference for the operational logic which describes the operation should be identified from the Ref column of the Event Summary Table. This reference is a letter which also appears in the Ref column of the Fundamental Operations table for the instruction in question. Turn to the Table of Fundamental Operations for the logic in question and identify the reference. All the logic described within that reference should be checked.

5.1.5 LOCALIZING MALFUNCTIONS

Before the diagnostics can be run, the memory must first be operating. Hence, the Worst Case memory pattern test, Part No. ND41-8041 must be run and accepted first. If this program will not run, then check the power supply and perform a manual memory check. The memory must be able to cycle data before any malfunction in the processor or memory can be isolated. The power supply check procedure is given in paragraph 5.3.1.2 and the manual memory checkout procedure is given in paragraph 5.3.1.3.

5.1.6 EQUIPMENT REQUIRED FOR MAINTENANCE

Equipment required for test and trouble analysis of the ND812 is listed in Table 5-1. The test equipment specified is recommended; however, suitable substitutes may be used if they have specifications equal or better than that for the equipment listed.

Table 5-1. Test Equipment Required For Maintenance

NAME	MODEL OR PART NO.	MANUFACTURER
Oscilloscope	Type 454 Dual Trace (or equivalent)	Textronix
Current Probe	P 6021	Textronix
Multimeter	Type 630 or 310	Simpson
I/O Cable Extender for WWH Cards	75-9202	Nuclear Data
MIS/MTS Cable Extender for Memory Boards (MTS & MIS)	75-9301	Nuclear Data
IC Test Clip	AP 923700	A.P. Inc.

5.2 PREVENTIVE MAINTENANCE

Periodic inspection of the equipment is necessary to detect potential trouble before a malfunction occurs. Items such as condition of connectors and cables, proper operation of controls and indicators, and cleanliness of equipment, should be monitored and corrected as required prior to each use. Refer to visual inspection checklist Table 5-2.

Except for the checks listed in Table 5-2, there are no regular maintenance procedures for the ND812. When a malfunction occurs, the indication will be erratic processing of instructions in a given program. If erratic operation occurs, any system of which the ND812 is a part must first be isolated to ensure that a bussed peripheral device is not at fault. Generally, the easiest way to isolate a malfunctioning peripheral device is to disconnect the devices one at a time until all have been disconnected. Be sure that the system is shut down when each of the peripherals is disconnected. If the malfunction persists after all the peripherals have been isolated, the diagnostic routines should be run. However, if the teletype is not malfunctioning, it should be reconnected into the system. The function of each of the diagnostic routines is described in paragraph 5.3.1.1.

5.3 CORRECTIVE MAINTENANCE

Corrective maintenance includes performance tests, memory troubleshooting procedures, and disassembly procedures. When a malfunction occurs, these procedures provide an aid in testing and isolating the fault.

5.3.1 PERFORMANCE TESTS

Performance tests include Diagnostic tests, power supply checks/adjustments, and manual memory checkout/adjustments.

5.3.1.1 DIAGNOSTIC TESTS. The diagnostic routines supplied with the ND812, when performed, exercise the logic circuitry contained in the major functional operations. Table 5-3 lists these diagncstic tests. Unless the source of the error is obvious; for example, incorrect data as the result of a divide operation, the diagnostic test should be performed in the order presented in Table 5-3. Refer to ND812 Diagnostic Software Instruction Manual, IM41-8001 for specific procedures on loading and operation of diagnostic routines.

NOTE

> When loading programs using high speed paper tape reader or magnetic tape cassette, it is necessary to use Binary Loader ND41-0005. Refer to ND812 Utilities Manual for description and loading of this program.

5.3.1.2 POWER SUPPLY CHECKS/ADJUSTMENTS. Before the manual memory checks are run, both the -30 volt memory power supply and the integrated circuit +5 volt power supply must be operating. To be sure that they are, perform the following procedure.

a. Remove the 4 top-cover retaining screws and the top cover to gain access to the power supply compartment and the power supply adjustments (Figure 5-1).

NOTE

> Using the multimeter (Table 5-1), check the -30 volt and +5 volt power supplies. These voltages are outputs to terminat-strip AA (Figure 5-1) and are referenced on sheet 19 of the 28 sheet central processor logic diagrams in Section VII.

b. Connect positive lead of Multimeter to +5 Vdc terminal (Figure 5-1) and negative lead to ground. Multimeter shall indicate +5.0 Vdc. Adjust +5 Vdc potentiometer (Figure 5-1) as necessary to obtain indication of 5.0 Vdc on Multimeter.

c. Connect positive lead of multimeter to +5 Vdc terminal and negative to -25 Vdc terminal. The multimeter shall indicate +30.0 Vdc. Adjust -30 Vdc potentiometer (Figure 5-1) as necessary to obtain indication of +30.0 Vdc on multimeter.

d. Check that the power sense logic is operating (paragraph 4.2.1).

e. Depress the START switch and check that the RUN indicator lights. If the RUN indicator does not light, check the go-control logic (paragraph 4.2.2.2).

Table 5-2. Visual Inspection Checklist

ITEM	CHECK
Mechanical Connections	a. Check that all screws are tight and that all mechanical assemblies are secure.
	b. Check that all crimped lugs are secure and that all lugs are properly inserted in their mating connectors.
Wiring and Cables	a. Check all wiring and cables for breaks, cuts, frayed leads or missing lugs. Check wire wraps for broken, bent or missing pins.
	b. Check that no wire or cables are strained in their normal positions or have severe kinks. Check that cables do not chafe when panels are opened and closed.

Table 5-2. Visual Inspection Checklist (Cont'd.)

ITEM	CHECK
Fans	Check all fans for cleanliness and for normal air movement through cabinets.
IC's and Components	Check that all IC's are properly seated. Look for areas of discoloration on all exposed surfaces. Check all exposed capacitors for signs of discoloration or leakage, or corrosion. Check power supply capacitors for bulges.
Switches	Check for cracks, discoloration or other visual defects.

Table 5-3. ND812 Diagnostic Tests

TEST	NUMBER
Worst Case Memory Pattern Test	ND41-8041
OPR-MRI Test	ND41-8001
XCT-TWI Test	ND41-8002
Memory Address Test	ND41-8004
Random ISZ-DSZ Test	ND41-8013
Random ADJ-SBJ Test	ND41-8014
Random LDJ-STJ Test	ND41-8015
Random JMP-JPS Test	ND41-8016
Hardware Multiply and Divide Test	
Effective for Serial Numbers 245 and below	ND41-8019
Effective for Serial Numbers 246 and above	ND41-8045
Multiple Field Random TWJPS-TWJPS@ and Interrupt Test	ND41-8026
High-Low Speed Reader Test	ND41-8005

Table 5-3. ND812 Diagnostic Tests (Cont'd.)

TEST	NUMBER
Low Speed Punch Test	ND41-8006
High Speed Punch Test	ND41-8007
High Speed Reader Test	ND41-8008

5.3.1.3 MANUAL MEMORY CHECKOUT/ADJUSTMENTS. The manual memory checkout ensures that the memory is cycling. If the memory does not cycle data into and from the processor, the ND812 will not run. Before the manual memory check is performed, be sure that the power supply is checked and operating properly (paragraph 5.3.1.2). To perform the manual memory checkout procedure, toggle the following program into the ND812 through the front panel switch register switches (i.e., set SWITCH REGISTER switches to 0000_8 and depress LOAD AR for address, the LOAD MR for instruction, etc.).

Address	Instruction	Symbolic
0000_8	0000_8	0
0001_8	3501_8	ISZ.-1
0002_8	6101_8	JMP.-1
0003_8	6102_8	JMP.-2

a. Remove the left side cover plate exposing the MTS, MIS and memory stack cards (Figure 5-2).

b. Set ND812 front panel power switch to POWER OFF position, and remove the MTS card from the unit. Leave the core stack connector (Figure 5-3) on the front of the board in place and remove the board from the internal male connector.

c. Interconnect the card extender cable, Part No. 75-9301 (Table 5-2), male connector to the ND812 internal female connector previously occupied by the MTS card. Connect card extender female connector to the MTS card internal male connector, taking care not to twist the cable (Figure 5-3). Set ND812 front panel power switch to POWER ON position.

d. Connect the current probe (Table 5-1) to one of the core stack bundles

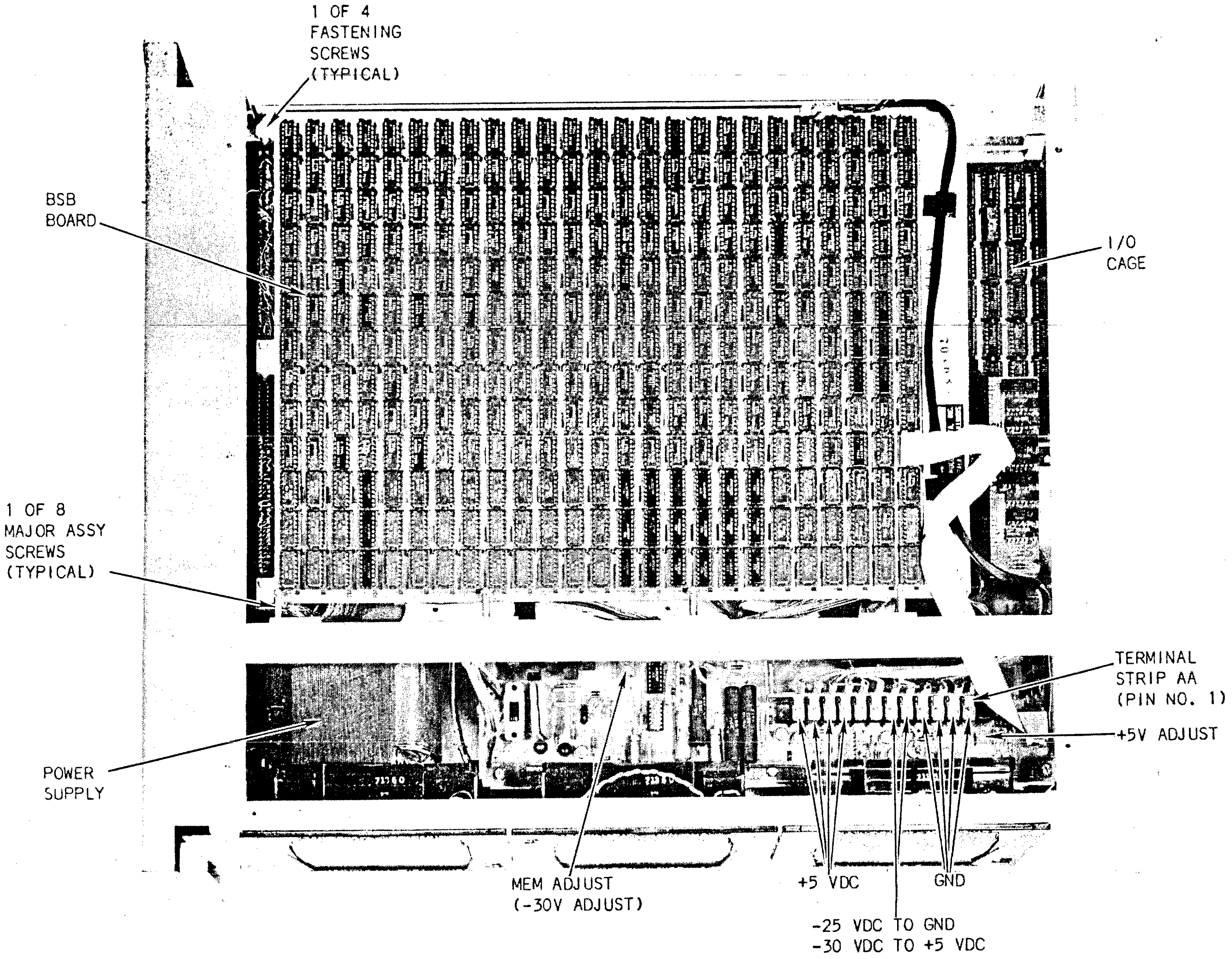


Figure 5-1. ND812, Top View, Top Cover Removed, Voltage Test Points and Adjustments

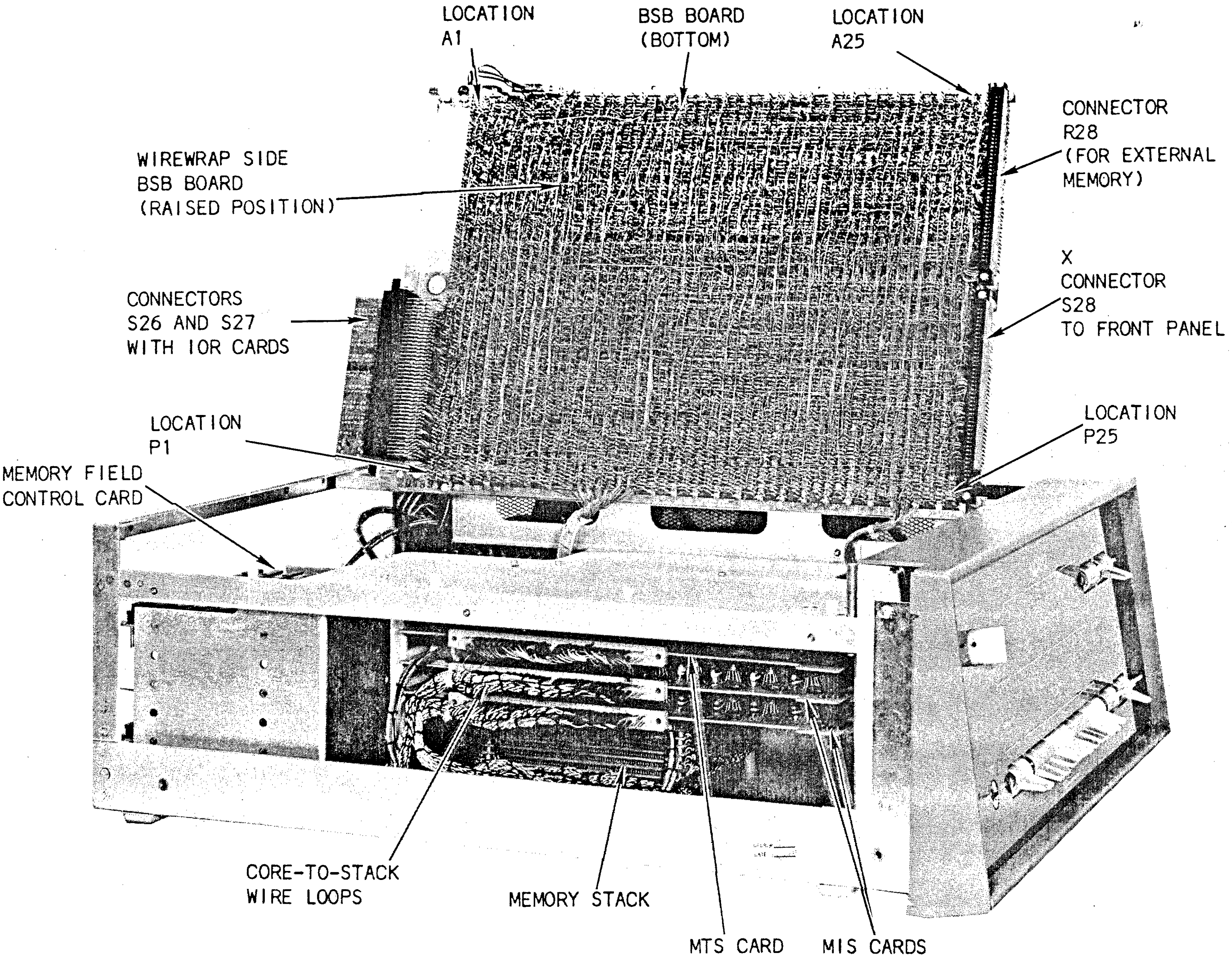


Figure 5-2. ND812, Memory Compartment Side View, Top Cover Removed, BSB Board Removed

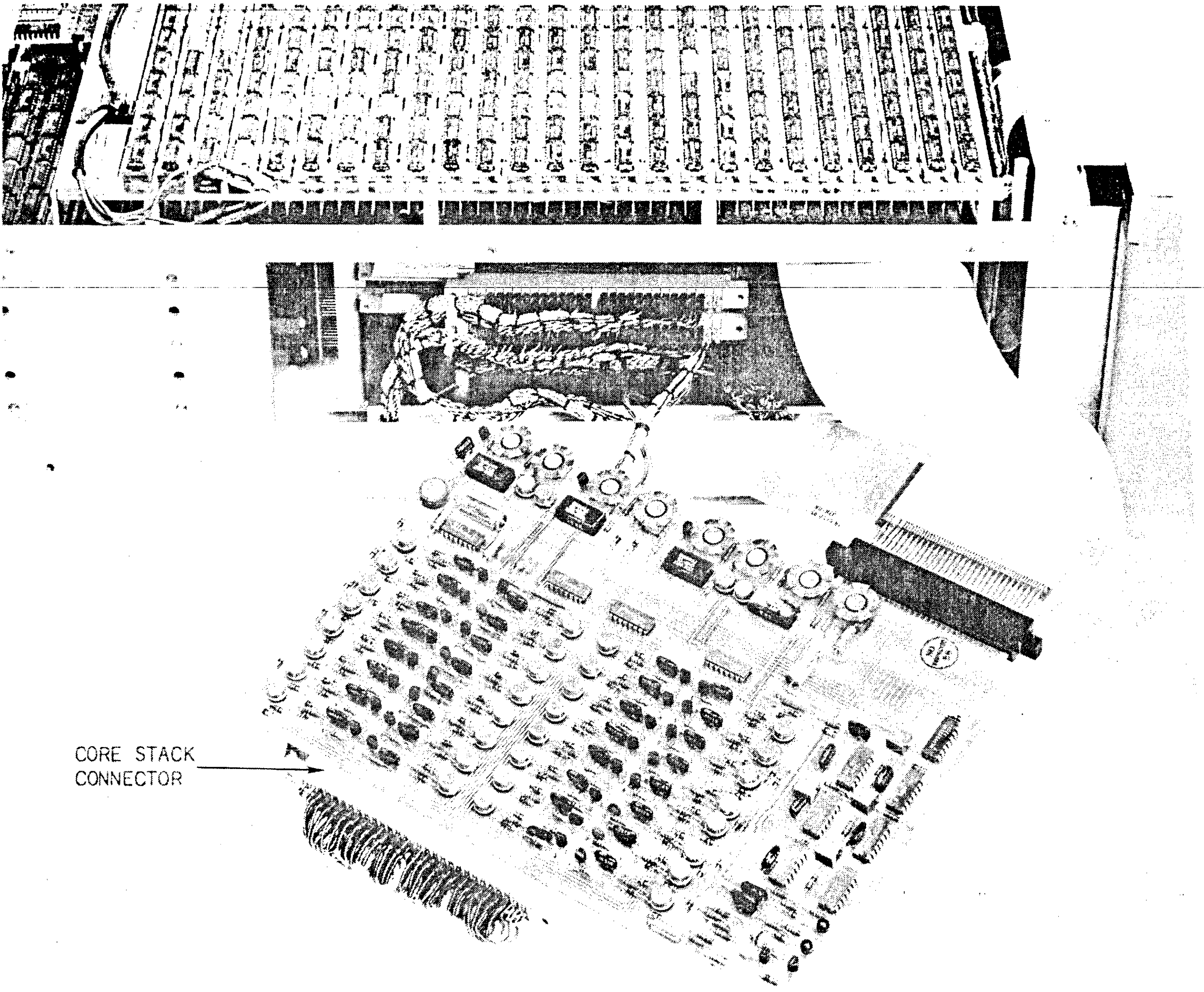


Figure 5-3. MTS Card on Cable Extender

(Figure 5-2). The MTS-to-core-stack connector cable harness has two bundles of 8 wires, each separated from the rest of the cable; one of these bundles is for X-axis currents and the other is for Y-axis currents. The end of the current probe should be clamped aroung the X-axis core-to-stack wire loop with the arrow on the probe pointing toward the connector.

e. Start the program at address 0001_8 and connect the other end of the current probe to one of the vertical inputs of a dual trace oscilloscope (Table 5-1). Set the current probe switch to the 2ma/mv position and the oscilloscope vertical amplifier to 50mv/cm. The actual indication at the display is the product of the current probe setting and the vertical amplitude setting at the input to the oscilloscope, 2ma X 50 = 100 ma/cm. Adjust the oscilloscope trigger sensitivity for a positive-going read current leading edge (Figure 5-4) with the trigger set for +AC. Set the horizontal sweep of the oscilloscope for 0.2 microsecond/cm. Center the display as shown for waveform C, Figure 5-5.

NOTE

If the processor runs the three-stop program, use it for the memory adjustment procedure. If the three-step program does not run, try different addresses and different memory fields. If the three-step program cannot be run, perform the Bit check procedure; otherwise, continue with the adjustment procedure.

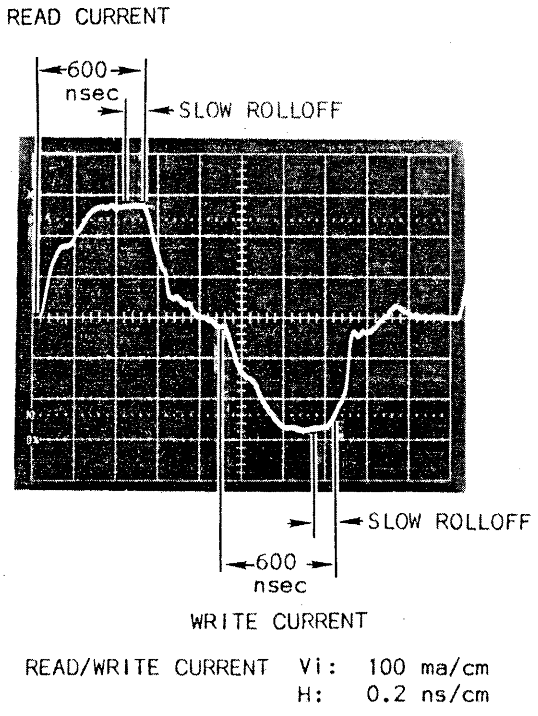
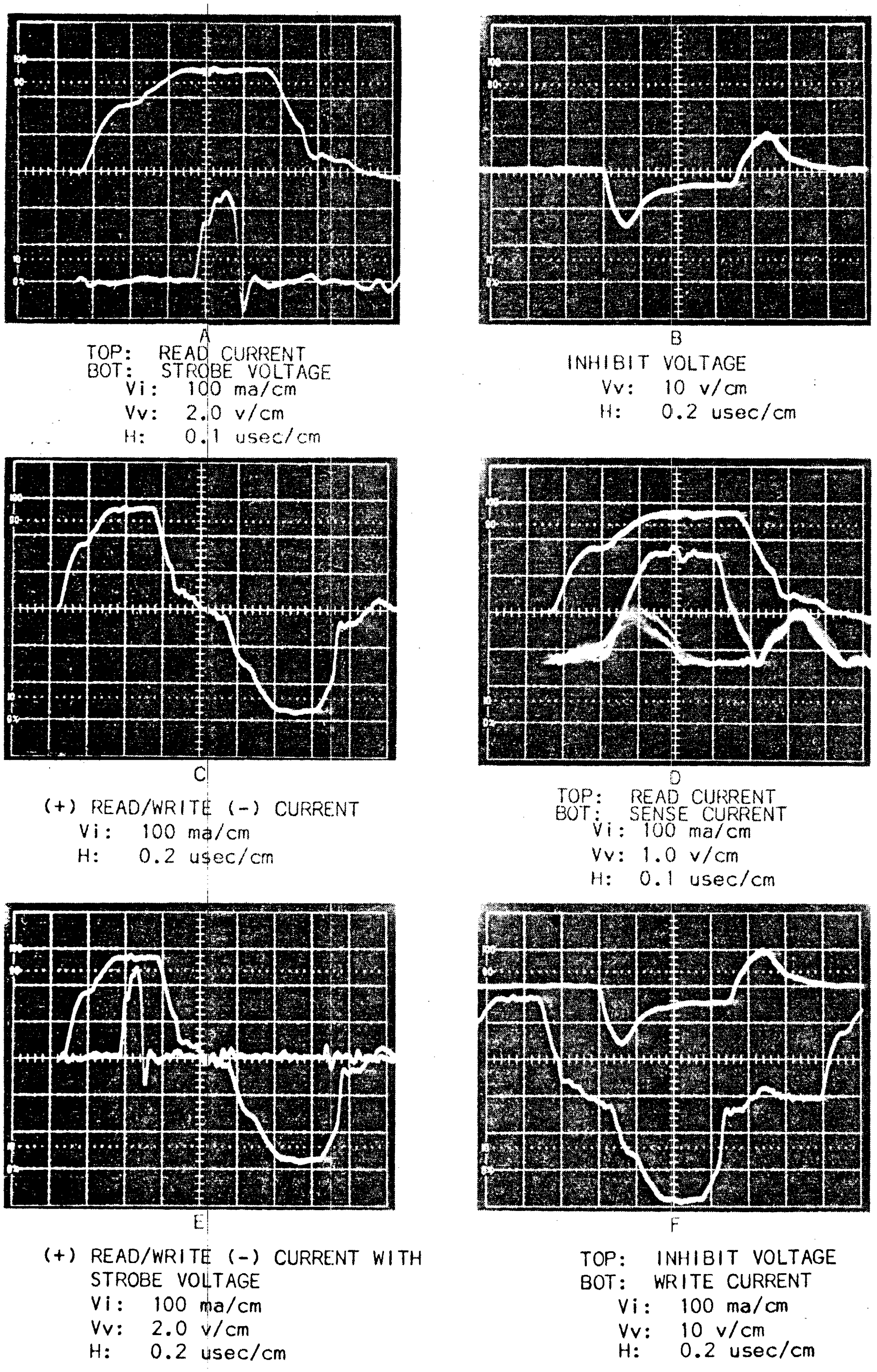


Figure 5-4. Typical Read/Write Current Waveform



A

TOP: READ CURRENT
BOT: STROBE VOLTAGE
Vi: 100 ma/cm
Vv: 2.0 v/cm
H: 0.1 usec/cm

B

INHIBIT VOLTAGE
Vv: 10 v/cm
H: 0.2 usec/cm

C

(+) READ/WRITE (-) CURRENT
Vi: 100 ma/cm
H: 0.2 usec/cm

D

TOP: READ CURRENT
BOT: SENSE CURRENT
Vi: 100 ma/cm
Vv: 1.0 v/cm
H: 0.1 usec/cm

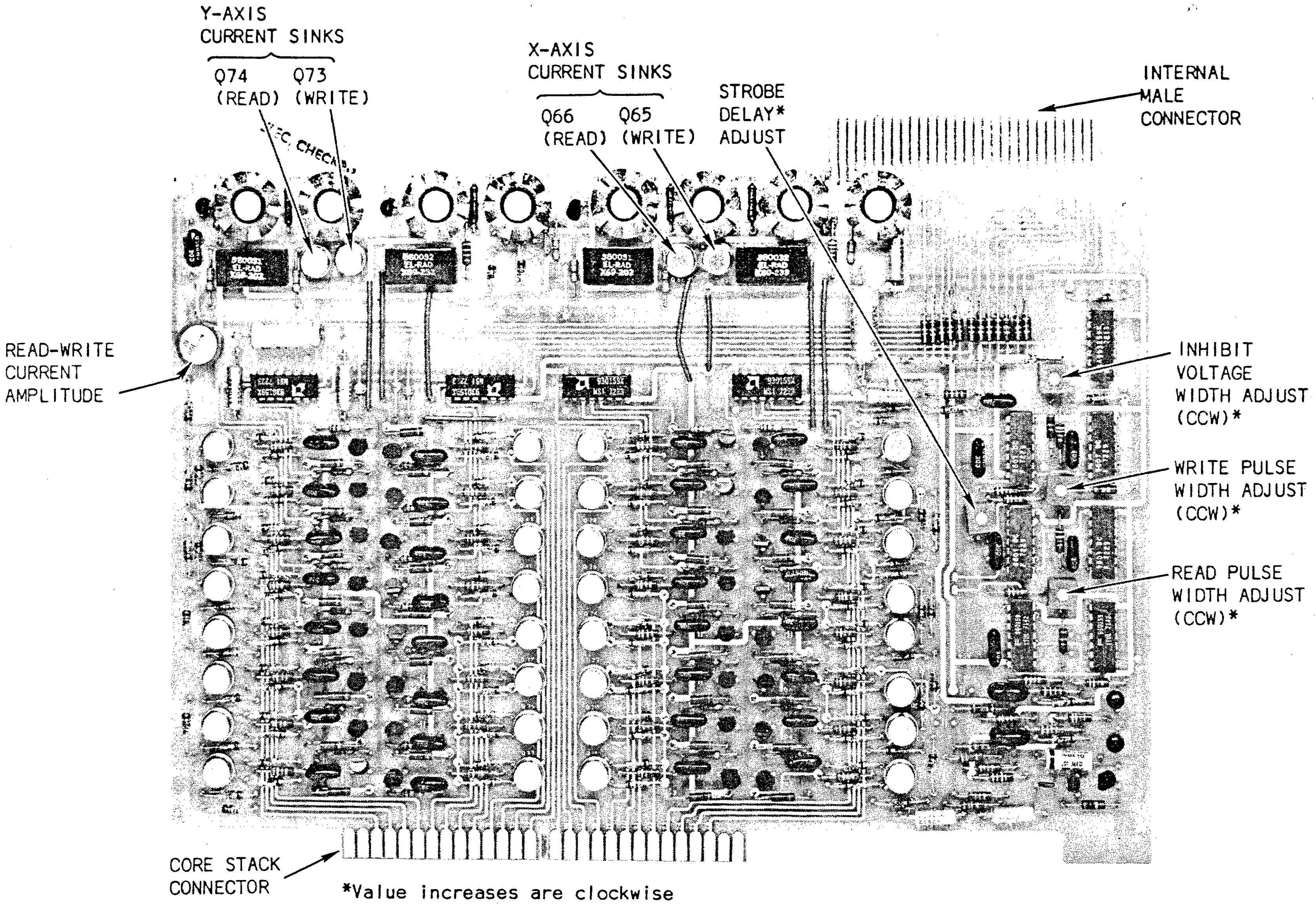
E

(+) READ/WRITE (-) CURRENT WITH
STROBE VOLTAGE
Vi: 100 ma/cm
Vv: 2.0 v/cm
H: 0.2 usec/cm

F

TOP: INHIBIT VOLTAGE
BOT: WRITE CURRENT
Vi: 100 ma/cm
Vv: 10 v/cm
H: 0.2 usec/cm

Figure 5-5. Memory Waveforms



*Value increases are clockwise direction unless noted by CCW (counterclockwise).

Figure 5-6. MTS Card Adjustments

f. With the three-step program in memory and running, adjust the oscilloscope to attain a display similar to the waveform shown in Figure 5-4 and centered as shown by waveform C, Figure 5-5; then adjust the read/write pulse-width potentiometers (Figure 5-6) as outlined in step g., below.

NOTE

> Know your oscilloscope calibration. The three-step program generates memory cycles at 2 microsecond intervals. Check the time base setting for 0.2 µsec/cm triggering on read current going positive; the second read-current pulse should be exactly 10 divisions out. Current probes are not all alike, but quite close. More likely oscilloscope voltage calibration can cause problems in obtaining uniform current adjustments, because current probes are all passive.

g. Ignoring the trailing edge ringing, the read/write waveforms are characterized by a slow rise (Figure 5-4), a flat top whose flatness varies somewhat with the bit pattern of data in memory, and a fall time which starts with a slow rolloff and then becomes a fast fall or rise to the baseline. The read/write pulse width is 600 nanoseconds, but is not supercritical. Adjust the read and write pulse width potentiometers (Figure 5-6) as necessary to obtain pulse width of 600 nsec as shown in Figure 5-4. Conventional waveform rise and fall time (10% amplitudes) does not apply for these adjustments. If there is no rolloff on the trailing edge of either the read or write waveform, the respective current sinks are defective and should be replaced with a Type 2N3725 transistor. Transistor Q74 controls the Y-axis read-current rolloff, Transistor Q73 controls the Y-axis write-current rolloff, Transistor Q66 controls the X-axis read/current rolloff, and Transistor Q65 controls the X-axis write-current rolloff. The rolloff must be present for both the read and write current pulses before memory can be successfully adjusted.

h. If the X-axis read/write currents are properly adjusted, and both the positive (read) and negative (write) waveforms have a slow rolloff, check the current waveforms for the Y-axis read/write currents and rolloff by changing the current probe from the X-axis wire bundle to the Y-axis wire bundle (refer to step d.). If the adjustments have already been made for the X-axis read/write currents, do not readjust for the Y-axis read/write currents. If the slow rolloff portion of the waveform is not present, change the appropriate current-sink transistor (refer to step g.).

i. Connect a second probe (voltage) to the second channel of the dual-trace oscilloscope and check the strobe-width and strobe-delay times as follows.

1) Set the vertical amplitude of the second channel to 2.0 volt/cm and the horizontal sweep of both channels to 0.1 μsec/cm.

2) Connect the voltage probe to the strobe test point and probe gnd to gnd test point (Figure 5-7), and check that the strobe pulse width is between 125 and 150 nanoseconds (waveform A, Figure 5-5). If the strobe pulse width does not fall within this prescribed range the SAT (selected at test) resistor in the differentiator (3A1, of the 6-sheet and 4A1 of the 10-sheet memory drawings) may be out of tolerance or the Fairchild 832 IC is defective. The resistance range should be between 2 and 10 kilohms.

3) Adjust the strobe delay control (Figure 5-6) so that the leading edge of the strobe pulse lies 280 nsec from leading edge of the read/current pulse (waveform A, Figure 5-5). Set ND812 front panel power switch to POWER OFF position and remove MIS card from unit. Leave core stack connector in place (Figure 5-3). Interconnect the card extender cable, part no. 75-9301 (Table 5-1) male connector to the ND812 internal female connector previously occupied by the MIS card. Connect the card extender female connector to the MIS card internal male connector taking care not to twist cable. Set ND812 front panel power switch to POWER ON position.

j. Remove the voltage probe from the strobe-voltage test point and place it at an active inhibit-voltage test point (Figure 5-7). Set the vertical amplitude for 10 volts/cm and the horizontal sweep for both channels to 0.2 μsec/cm. Check the inhibit voltage trace (waveform F, Figure 5-5) for the proper waveshape.

1) Adjust the inhibit-voltage width control (Figure 5-6) so that the inhibit voltage turns off at the same time that the write-current turns off.

2) The negative portion of the trace should be between 8 and 10 volts peak. The exponential decay of this trace should end at about the end of the write-current trace. The span of the negative-going inhibit voltage is the turn-on time. The positive-going portion of this trace is the turn-off time and should also be 8 to 10 volts peak before the exponential decay occurs.

NOTE

One or more of the test points called out for the inhibit will produce an active trace and some will not. For those test points which do not produce an active trace, a +5 Volt level will be present.

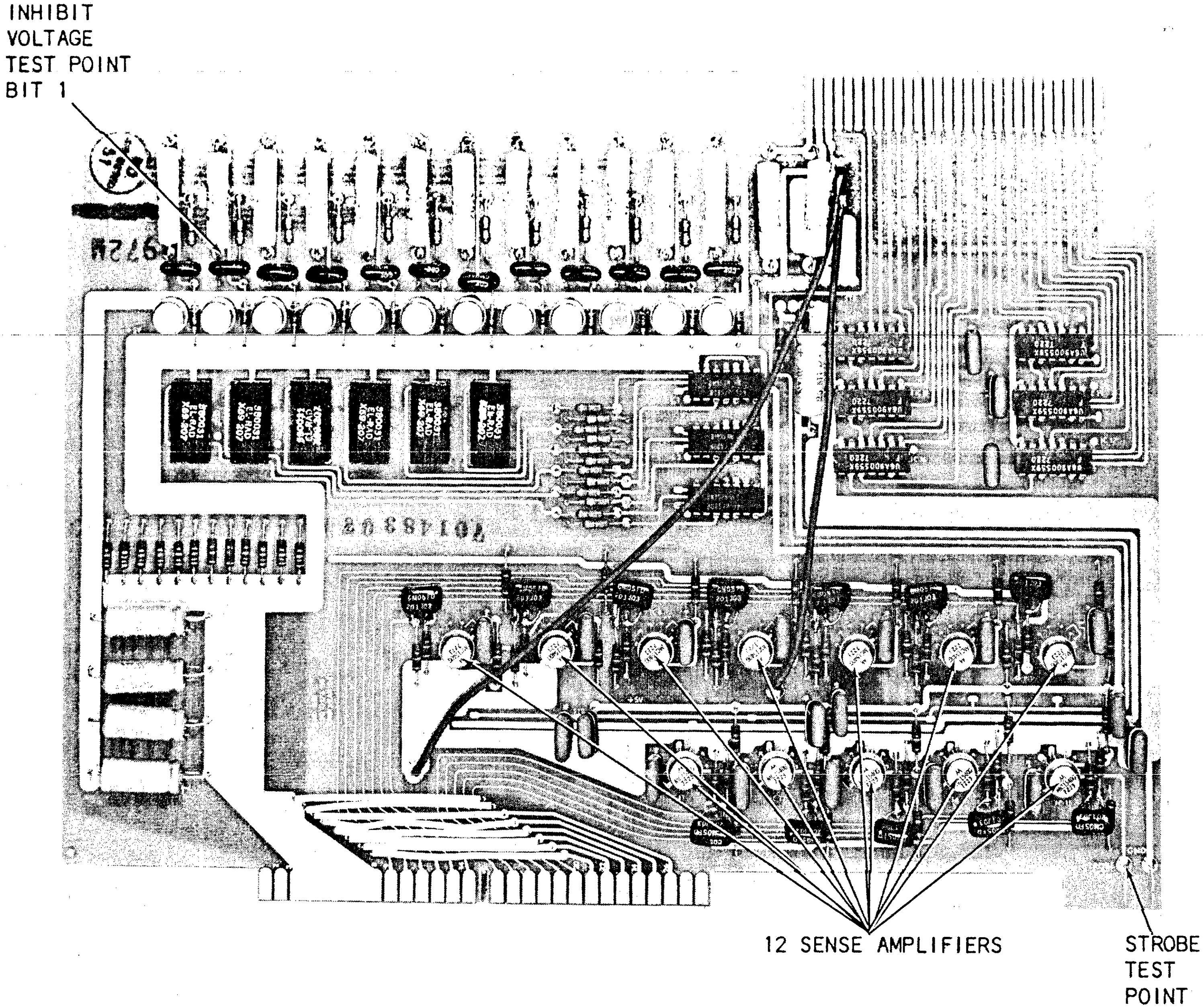


Figure 5-7. Location of Sense Amplifiers, MIS Card

k. Remove the voltage probe from the inhibit voltage test point. Reset the horizontal sweep of the oscilloscope for a sweep of 1 msec/cm. The read-write current trace should become a straight line. Adjust focus and brightness to accommodate the sharpest image. Center the image so that the center horizontal baseline of the display bisects the trace. Using a standard Farenheit thermometer, measure the ambient temperature in the vicinity of the MTS card. Adjust the read-write current amplitude for the temperature/amplitude setting defined by the curve in Figure 5-8 after there is no discernable change in temperature for at least 15 minutes. Reset the oscilloscope horizontal sweep to 0.2 μsec/cm and check that the read/write current waveform is similar to waveform C, Figure 5-5.

l. Remove test equipment and reinstall the MTS card into its slot in the memory compartment. Reinstall sides and covers and permit system temperature to stabilize again. Run the three-step program until the system temperature stabilizes again and then run the ND41-8041 Worst Case Memory Pattern Test Program for al teast 15 minutes.

5.3.2 MEMORY TROUBLESHOOTING

As with the processor, there are no fixed troubleshooting procedures. Familiarization with the memory principles of operation in Section IV, paragraph 4.5 of this manual is imperative. However, if the ND812 will not hold a program, faulty operation may be caused by the inability of memory to retain a bit, a faulty sense circuit, address decoder, or inhibit circuit. The bit check, applicable to each field independently may help isolate the problem, and may help locate one or more fields that are good, in which the three-step program may reside. Other troubleshooting hints are given in paragraphs 5.3.2.1 through 5.3.2.4.

5.3.2.1 BIT CHECK. To perform the bit check, proceed as follows.

a. Load address of 0000_8 of selected memory field (this address can be set for any location desired).

b. Load memory register with 0000_8.

c. Set ND812 SWITCH REGISTER switch 11 in the UP position and depress LOAD MR switch.

d. Set SWITCH REGISTER switch 1∅ in the UP position and depress LOAD MR.

e. Then set SWITCH REGISTER switches 09, etc., until all switches are up, depressing LOAD MR each time.

f. Set SWITCH REGISTER switch 11 in the DOWN position and depress LOAD MR.

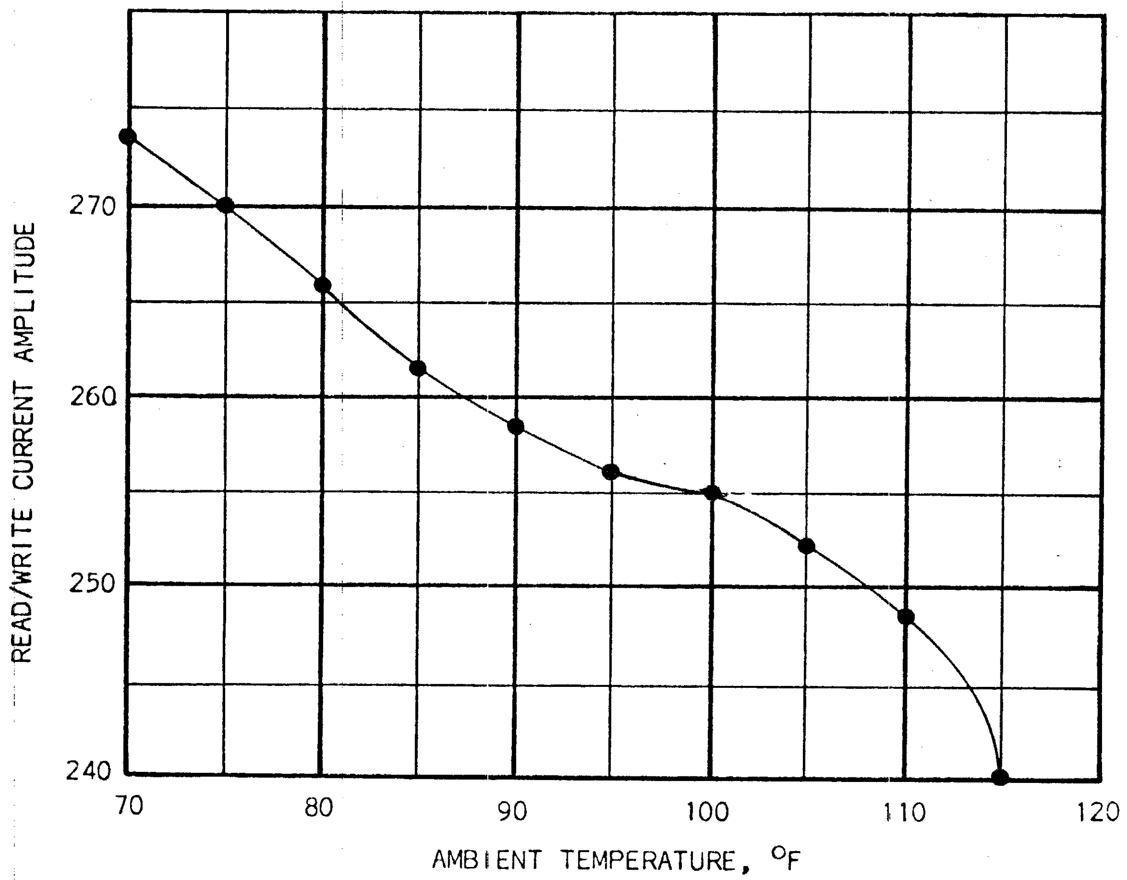


Figure 5-8. Read/Write Current Amplitude Vs. Temperature

g. Set SWITCH REGISTER switch 10 in the DOWN position and depress LOAD MR.

h. Then set SWITCH REGISTER switch 09, etc., until all switches are down, depressing LOAD MR each time.

i. Repeat step a., returning to original address.

j. Depress ND812 START switch and use NEXT WORD switch to examine all 24 locations loaded during steps a through h. Missing or extra bits will be obvious.

5.3.2.2 SENSE CIRCUITS. A weak sense circuit with more noise than nominal is often a clue to an inhibit circuit change in value. Raising memory power voltage (clockwise) tends to verify this, although each time power supply voltage is adjusted, read/write current must be corrected. A good check for proper sense/strobe conditions can be obtained using a voltage probe (1 volt/cm) at pin 1 or pin 10 of any (or a selected) sense amplifier. Positive, full-wave rectified and amplified sense pulses (wavefom D, Figure 5-5) can be observed, with a strobe pattern (initially positive, with some ringing) superimposed on the sense pulses. The pattern is caused by other sense amplifier outputs, as a result of strobe, and therefore appears perhaps 25nsec later than the tested strobe leading edge. Familiarity with waveshapes encountered here, particularly with the worse patterns program running can sometimes detect deficiencies. To check the sense amplifiers, proceed as follows.

a. Set ND812 front panel power switch to POWER OFF position. Remove the MIS card whose sense circuits are to be checked (Figure 5-2) from the memory compartment. Leave the core stack connector on the MIS card as shown for the MTS card in Figure 5-3.

b. Interconnect the card extender cable, part no. 75-9301 (Table 5-1), and connector by placing the male end in the internal female connector previously occupied by the MIS card. Connect the female connector of the cable to the MIS card internal male connector, taking care not to twist the cable. Set ND812 front panel power switch to POWER ON position.

c. Connect a voltage probe to one of the input channels of a dual-trace oscilloscope (Table 5-1) and connect it to pin 1 of pin 10 of any of the sense amplifiers (Figure 5-7). Check for the presence of a trace similar to that shown for waveform D (lower) of Figure 5-5.

NOTE 1

Pin 1 of all sense amplifiers is identified by a bracket ([ ) on printed circuit card. Count clockwise from this position looking down on the amplifier can.

NOTE 2

If sense voltage is checked with Worst Case Memory Pattern Program (ND41-8041) loaded and running in the ND812 Computer, observe the following; when the program resides in field Ø, field 1 is being exercised; when program resides in field 1, field Ø is being exercised, etc. Monitor ND812 front panel MEMORY FIELD lamps to determine where program resides. If MIS card for field 1 is being checked, check sense voltage when both MEMORY FIELD lamps are off.

5.3.2.3 WHEN NOTHING WORKS. Check line voltages and fuses and front panel key switch on position. Check temperature sensing a-c switch on power supply heat sink panel (Figure 7-2, Section VII). Also check power supply voltages on terminal strip AA for +5 Volt and -30V as specified in paragraph 5.3.1.2. If front panel indicators light but no manual operation from front panel switches can be performed, check;

a. That key switch is in power on position, not in control off position.

b. That signals BDPWR and PWRDY are active from power supply.

c. That the key switch is providing ground to control switches when in the PWR ON position.

d. If these checks are positive and no results are obtained, perform the front panel checkout procedure as outlined in paragraph 5.3.2.4.

5.3.2.4 FRONT PANEL CHECKOUT. To perform the front panel checkout, proceed as follows.

a. Check the +5V power supply when power is turned on.

b. Place the SELECT REGISTER switch to the ADDRESS position. Place all SWITCH REGISTER switches down and depress the LOAD AR switch. Check that all SELECTED REGISTER indicator lamps are out. Set the switch register switches to 0001_8 and press the LOAD AR switch. The SELECTED REGISTER indicator lamps should illuminate accordingly as the switch register is set. Use the following pattern to test the LOAD AR switch, SELECTED REGISTER lamps and switch register switches.

0001, 0002.........7775, 7777, 7776, 7774.....0002, 0000

c. Using the same bit pattern and the LOAD MR switch, check the MEMORY REGISTER indications.

d. The NEXT WORD switch is tested by interrogating the memory locations loaded in step c.

e. The MEMORY FIELD switches and lamps are tested by placing the switches in a binary sequence (00, 01, 10, 11) and depressing the LOAD AR switch. The lamps should light according to the bit pattern.

f. The START, STOP, CONT, SINGLE STEP, and SINGLE INST switches are tested as follows.

1) Load the following instructions into memory.

Address	Data	Mnemonic
0000	1400	IDLE
0001	0600	TW JMP
0002	0000	0

2) Load the starting address 0001 and depress START. The Run lamp should now be lit. Place the SINGLE INSTR switch up. The Run lamp should go out. When the CONT switch is depressed and released, the ADDR REG indication should should go back and forth between 0000 and 0002. Place the SINGLE STEP switch up. When the CONT switch is depressed and released, the ADDR REG should now indicate from 0000 to 0001 and 0002 to 0000. The PC indicator should be going from 0000 to 0001, not

stopping at 0002. Place both SINGLE INSTR and SINGLE STEP switches down and depress the CONT switch. The ND812 should now be running. Depress STOP, and the ND812 should halt.

g. The SELECTED REGISTER switch positions ADDRESS and PC have already been tested. The EXT position should illuminate all SELECTED REGISTER lamps when the ND812 is halted.

5.3.3 DISASSEMBLY

Generally, disassembly is straightforward. However, removal of interconnections is not always obvious. Usually reassembly is in the reverse order of disassembly. Follow the general procedures below for disassembly.

5.3.3.1 FRONT PANEL.

a. To remove the cover, loosen the three fasteners at top of front panel after first removing all covers.

b. Remove the power key from the front key lock and lift out front panel.

c. Remove black and white twisted power leads from key switch (Figure 5-9) and disconnect connector from S28.

5.3.3.2 BSB BOARD

a. Remove the four BSB hold down screws (Figure 5-10).

b. Carefully raise BSB board (Figures 5-2 and 5-11) making sure that memory extension cable R28 and front panel cable S28 are disconnected. Also be sure that interface cables connected to S26 or S27 are also removed.

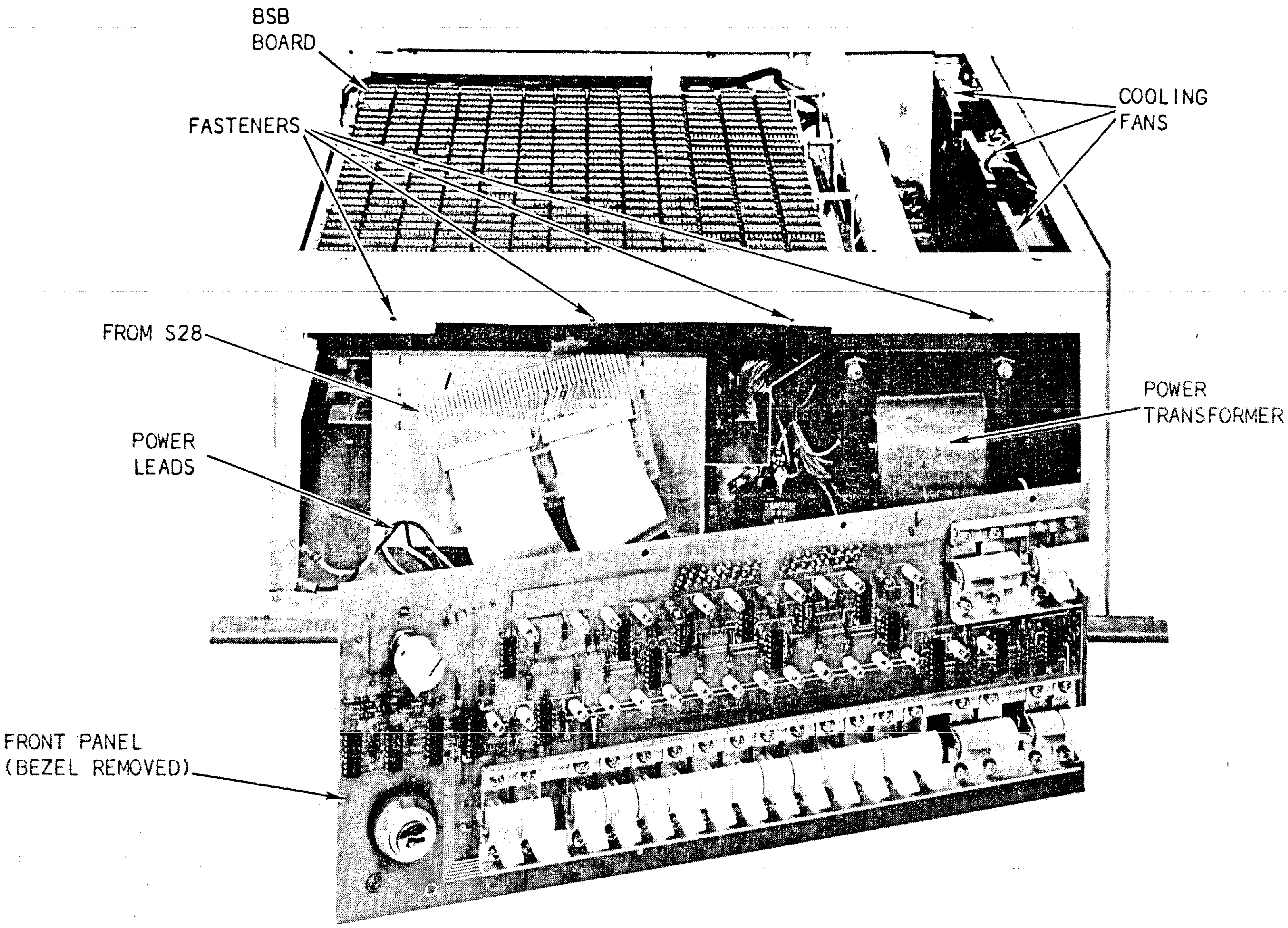


Figure 5-9. Front Panel, Partially Disassembled

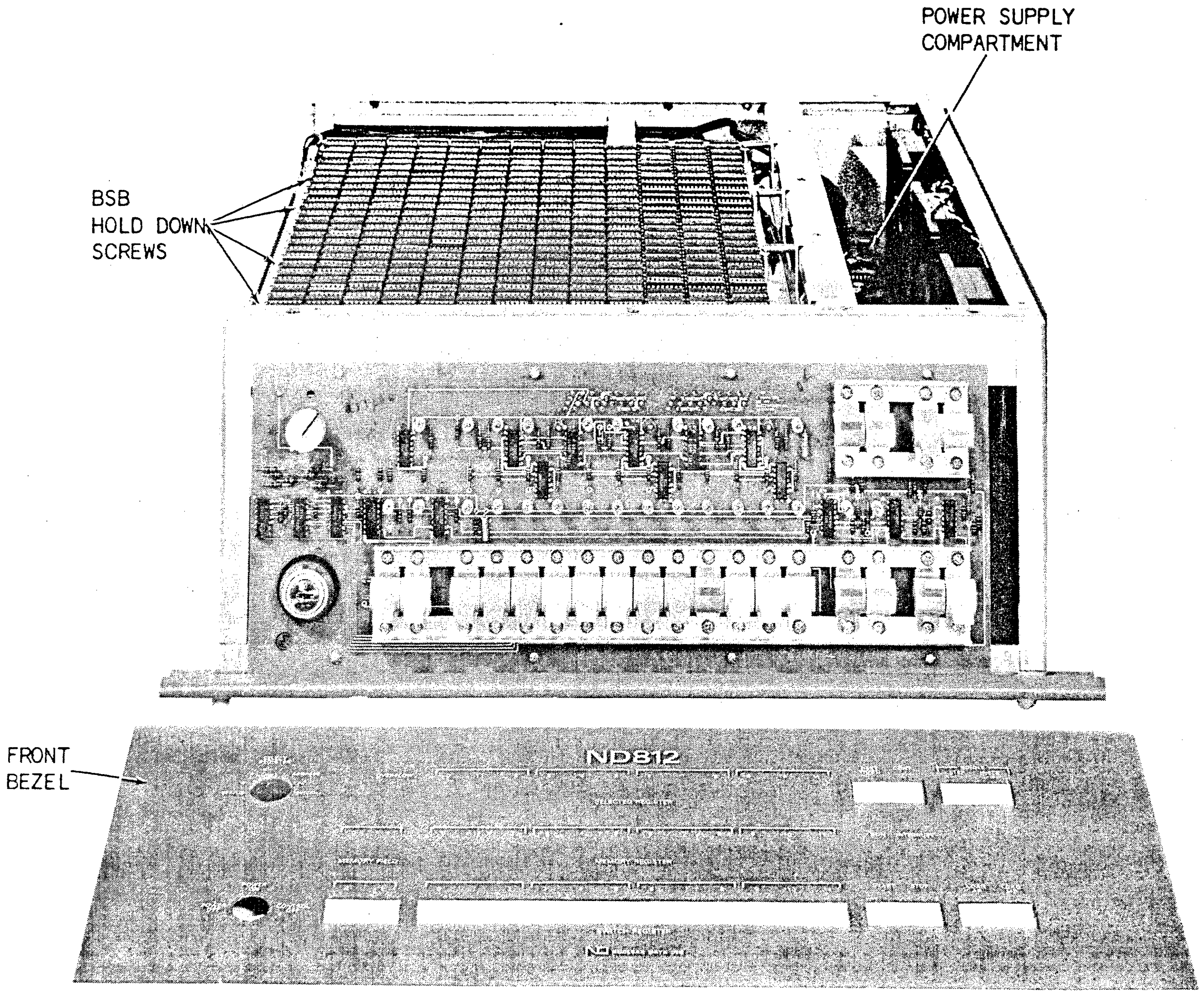


Figure 5-10. ND812, Top and Front Partially Disassembled

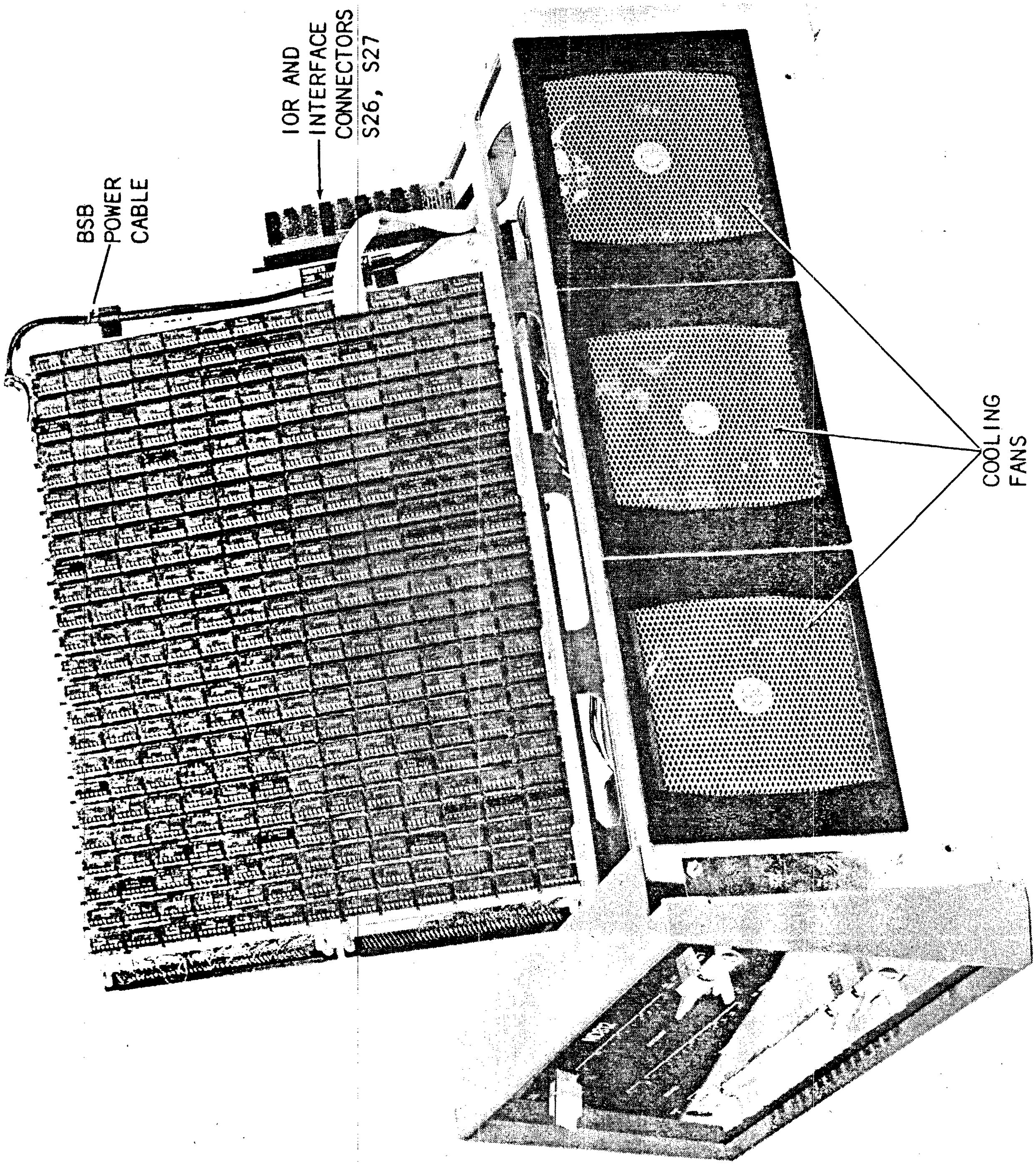


Figure 5-11. ND812 With Front Panel and BSB Board

SECTION VI
REPLACEABLE PARTS LIST

6.1 GENERAL

This section contains replaceable parts lists for the ND812 Central Processor, ND812 4K Memory and ND812 8K Memory. Table 6-1 provides a non-illustrated parts list for the ND812 Central Processor, Part Number 88-0397; Table 6-2 provides a non-illustrated parts list for the ND812 4K Memory, Part Number 84-0096; and Table 6-3 provides a non-illustrated parts list for the 8K Memory, Part Number 84-0097.

6.2 PARTS LOCATION

Refer to Figures 4-94, 5-1, 5-2, 5-9, 5-10 and 5-11 for physical location of major subassemblies and connectors as applicable. Refer to Figure 7-1 for information pertaining to physical location of integrated circuits as referenced on diagrams.

Table 6-1. Replaceable Parts List, ND812 Central Processor, Part Number 88-0397

ND PART NUMBER	DESCRIPTION	QUANTITY
14-1014	XTAL 16MHZ WRE	1
14-2027	LMP INCD 6V BIPN	29
21-0007	RES MG QW 5% 10 OHM	5
21-0023	RES MG QW 5% 47 OHM	2
21-0028	RES MG QW 5% 75 OHM	1
21-0031	RES MG QW 5% 100 OHM	14
21-0033	RES MG QW 5% 120 OHM	
21-0039	RES MG QW 5% 220 OHM	5
21-0041	RES MG QW 5% 270 OHM	
21-0043	RES MG QW 5% 330 OHM	2
21-0045	RES MG QW 5% 390 OHM	1

Table 6-1. Replaceable Parts List, ND812 Central Processor, Part Number 88-0397 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
21-0047	RES MG QW 5% 470 OHM	30
21-0051	RES MG QW 5% 680 OHM	4
21-0055	RES MG QW 5% 1K	6
21-0059	RES MG QW 5% 1.5K	1
21-0063	RES MG QW 5% 2.2K	
21-0071	RES MG QW 5% 4.7K	3
21-0073	RES MG QW 5% 5.6K	1
21-0079	RES MG QW 5% 10K	29
21-0083	RES MG QW 5% 15K	
21-0089	RES MG QW 5% 27K	2
21-0090	RES MG QW 5% 30K	
21-0091	RES MG QW 5% 33K	
21-5010	RES MF EW 1% 453 OHM	1
21-5016	RES MF EW 1% 1K	1
21-5019	RES MF EW 1% 1.37K	1
21-5027	RES MF EW 1% 6.19K	1
21-5102	RES MF EW 1% 1.82K	1
21-5104	RES MF EW 1% 2.32K	1
21-5132	RES MF QW 1% 27.4	
23-0022	RES WW 5W 5% 100 OHM	
23-0036	RES WW 10W 3% .25 OHM	3
23-0045	RES WW 5W 5% 180 OHM	
24-0093	RSVR HW 5% 500 OHM	2
24-0097	RSVR HW 5% 10K	
24-0111	RSVR HS 5% 10 OHM	
24-5003	THRM 50K	
25-0001	RS NTW 2. 2K 14 DIP	1
25-0002	RS NTW 220 330 16 DIP	7
26-0002	CAP SM 500WV 15PF	
26-0006	CAP SM 500WV 47PF	
26-0007	CAP SM 500WV 75PF	
26-0008	CAP SM 500WV 150PF	
26-0009	CAP SM 500WV 200PF	
26-0010	CAP SM 500WV 250PF	1
26-0012	CAP SM 500WV 500PF	1
26-0014	CAP SM 100WV 1000PF	

Table 6-1. Replaceable Parts List, ND812 Central Processor, Part Number 88-0397 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
26-0026	CAP SM 500WV 100PF	
26-0028	CAP SM 300WV 750PF	
26-1004	CAP MY 100WV .01MFD	1
26-1012	CAP MY 200WV .015MFD	
26-1033	CAP TBR 100WV .22MFD	
26-1037	CAP MY 250WV .1MFD	2
26-2024	CAP TM 22MFD 15V	5
26-4005	CAP CR .01MFD 150WV	
26-4018	CAP CR .1MFD 600WV	4
26-5012	CAP EY 25MFD 50WV AL	1
26-5016	CAP EY 100MFD 15WV	2
26-5031	CAP EY 500MFD 3WV	1
26-5040	CAP EY 63000MFD 15V	1
26-5041	CAP EY 11000MFD 75WV	1
26-7003	FLT LN/RECPT 10AMP	1
31-0006	DO ZR IN4736A	
31-0007	DO ZR IN4740A	1
31-0013	DO SL IN914B	16
31-0049	CO SL IN3208 STDMTD	2
31-0056	DO ZR IN4737A	1
31-0061	BRG RECT MDA952-2	1
31-0063	DO SL IN5059	4
31-0069	DO SL HI SPD	
31-0071	SCR MCR 2605 2	1
31-0080	DO NTW 16 DO 14 DIP	10
33-0033	TR SL NPN 2N3568	2
33-0045	TR SL NPN 2N4124	
33-0053	TR SL PNP 2N4126	
33-0065	TR SL NPN 2N3646	
33-0068	TR SL NPN 2N3055	2
33-0093	TR SL 2N4899 PNP	2
33-0098	TR SIL PNP 2N4890	5
33-0103	TR SL NPN 2N3725	
33-0104	TR SL PNP 2N2905A	
33-0106	TR SL PNP 2N4125	
35-0002	DTL-930 DIP 14 G	1

Table 6-1. Replaceable Parts List, ND812 Central Processor, Part Number 88-0397 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
35-0004	DTL932 DIP 14 DU BUF	
35-0006	DTL-944 DIP 14 G	2
35-0008	DTL-946 DIP 14 G	2
35-0009	DTL-948 DIP 14 FF	
35-0017	DTL-936 DIP 14 HXVT	4
35-0033	IC MC857P DIP 14 G	
35-0034	IC MC858P DIP 14 G	19
35-0039	TTL 9300 DIP 16 SFTR	27
35-0040	TTL 9304 DIP 16 FA	6
35-0041	TTL 9601 DIP 14 MONO	1
35-0053	IC SN7474N DIP 14 FF	1
35-0055	IC MC1540G TO5 10 SA	
35-0058	IC 9002 DIP 14 G	53
35-0059	IC 9003 DIP 14 G	38
35-0060	IC 9004 DIP 14 G	17
35-0061	IC 9005 DIP 14 G	14
35-0062	IC 9006 DIP 14 IE	20
35-0063	IC 9007 DIP 14 G	8
35-0064	IC 9008 DIP 14 G	36
35-0065	IC 9009 DIP 14 INBUF	3
35-0066	IC 9016 DIP 14 HXVT	47
35-0067	MSI 9309 DIP 16 MLPX	12
35-0068	MSI 9312 DIP 16 MLPX	12
35-0069	MSI 9301 DIP 16 DCDR	4
35-0071	IC 9001 DIP 14 JK	3
35-0072	IC 9022 DIP 16 DU JK	7
35-0081	IC 723 TO5 10L REG	2
35-0083	MSI 9316 DIP 16 CTR	1
35-0085	MSI 9314 DIP 16	6
35-0121	IC 9602 DIP 16 MONO	
35-0123	IC MSI 8200 DIP 14	
35-0140	IC SM74S04 DIP 14	1
35-0152	IC SM7438N DIP 14	1
37-0109	SWROT 1POL12POS NS1D	1
37-5016	SW SLD 115 230 PC	1
37-5017	SW SLD RK SPDT W DTN	8
37-5018	SW SLD RK SPDT T DTN	8
37-5019	SW SLD RK SPDT W SP	3

Table 6-1. Replaceable Parts List, ND812 Central Processor, Part Number 88-0397 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
37-5020	SW SLD RK SPDT T SP	3
37-6021	SWTGL SPDT 7101R	
37-8045	SWLMT	1
37-8053	SW KLK 2 POL 3 POS	1
38-0031	XTF DUL PLS DIP	
38-0032	XTF DUL BALUN DIP	
38-0035	IND B CRMG	6
38-0037	XTF PWR DU SEC	1
39-0003	GRILL FAN VNT	3
39-0004	CLIP MTG FAN	6
39-0005	FAN MFF VNT	3
39-2001	HT DSP TO-5 FAN	
39-2004	HT DSP TO-3 2X2-1/2	2
39-2005	HT DSP TO-3 3X3	2
39-3005	SLD DWR TRVL 18-7/8	1
39-4004	FUSE 3AMP S-B	1
39-4007	HLD FUSE P MTG	3
39-4011	FUSE 8AMP	1
39-4015	FUSE 5AMP S-B	1
39-4017	PLG CD ASSY FAN	1
39-4018	PWR CD 3 PRNG MACH	1
39-5011	BRR TERM 12 TERM	1
39-5012	FNN STRP 12 TERM RTAN	1
42-0001	WRP (BSB)	1
42-0006	WRP MFC-A	
50-0175	PCB BSB	1
50-0176	PCB MIS	
50-0177	FCB MTS	
50-0178	PCB P	1
50-0184	PCB WW4	
50-0436	PCB TCC	
50-0446	PCB MBD	1
50-0451	PCB MPC	1
50-0453	PCB CCC	1
50-0454	PCB PRS	

Table 6-1. Replaceable Parts List, ND812 Central Processor, Part Number 88-0397 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
50-0455	PCB IOR	1
50-0456	PCB IOT	1
54-0213	FP ND 812	1
63-0026	SWITCH BANKS	1
70-1423	F BOARD P812	1
70-1483	MIS-A	
70-1502	MTS-A	
70-1509	P BOARD 812	1
70-1685	BSB-A	1
70-1686	MFC-A	
70-1703	TCC-A	
70-1812	IOR-A	1
70-1813	IOT-A	1
70-1816	PRS-B	
70-1985	MBD-B	1
70-1986	MPC-A	1
71-0032	HARN PROCESSOR P.S.	1
71-0059	HRN. CCC TO F BD 812	1
71-0117	AC PWR CBL	1
71-0118	FP KEYLOCK SW HARN	1
72-0096	KNB SML 25SHAFT 123	1
72-0740	ND 812 PWR SUP	1
72-0744	F.P. 812	1
72-0773	ND 812 ENCL F BD	1
72-0774	ND 812	1
72-1011	WIRE WRAP ND 812	1
72-1012	RP ND 812 PROCESSOR	1
75-9505	CBL 14PIN PT-PT 12IN	1
84-0096	4 K MEMORY	
84-0097	8 K MEMORY	
84-0098	POWER RESTART	

Table 6-2. Replaceable Parts List, ND812 4K Memory, Part Number 84-0096

ND PART NUMBER	DESCRIPTION	QUANTITY
21-0007	RES MG QW 5% 10 OHM	13
21-0031	RES MG QW 5% 100 OHM	27
21-0033	RES MG QW 5% 120 OHM	12
21-0041	RES MG QW 5% 270 OHM	30
21-0043	RES MG QW 5% 330 OHM	4
21-0047	RES MG QW 5% 470 OHM	6
21-0051	RES MG QW 5% 680 OHM	32
21-0055	RES MG QW 5% 1K	72
21-0063	RES MG QW 5% 2.2K	1
21-0071	RES MG QW 5% 4.7K	8
21-0079	RES MG QW 5% 10K	13
21-0083	RES MG QW 5% 15K	2
21-0091	RES MG QW 5% 33K	1
21-5132	RES MF QW 1% 27.4	8
23-0022	RES WW 5W 5% 100 OHM	12
23-0045	RES WW 5W 5% 180 OHM	2
24-0097	RSVR HW 5% 10K	4
24-0111	RSVR HS 5% 10 OHM	1
24-5003	THRM 50K	2
26-0002	CAP SM 500WV 15PF	1
26-0006	CAP SM 500WV 47PF	2
26-0007	CAP SM 500WV 75PF	33
26-0008	CAP SM 500WV 150PF	3
26-0009	CAP SM 500WV 200PF	12
26-0026	CAP SM 500WV 100PF	12
26-0028	CAP SM 300WV 750PF	2
26-1004	CAP MY 100WV .01MFD	1
26-1037	CAP MY 250WV .1MFD	21
26-2024	CAP TM 22MFD 15V	7
26-5012	CAP EY 25MFD 50WV AL	7
31-0006	DO ZR IN4736A	1
31-0063	DO SL IN5059	3
31-0069	DO SL HI SPD	115
33-0045	TR SL NPN 2N4124	2
33-0053	TR SL PNP 2N4126	4

Table 6-2. Replaceable Parts List, ND812 4K Memory, Part Number 84-0096 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
33-0065	TR SL NPN 2N3646	2
33-0103	TR SL NPN 2N3725	48
33-0104	TR SL PNP 2N2905A	8
33-0106	TR SL PNP 2N4125	32
35-0004	DTL932 DIP 14 DU BUF	1
35-0033	IC MC857P DIP 14 G	3
35-0041	TTL 9601 DIP 14 MONO	6
35-0055	IC MC1540G TO5 10 SA	12
35-0061	IC 9005 DIP 14 G	6
35-0069	MSI 9301 DIP 16 DCDR	4
38-0031	XTF DUL PLS DIP	8
38-0032	XTF DUL BALUN DIP	2
39-2001	HT DSP TO-5 FAN	8
50-0176	PCB MIS	1
50-0177	PCB MTS	1
50-0436	PCB TCC	1
70-1483	MIS-A	1
70-1502	MTS-A	1
70-1703	TCC-A	1
73-0037	4KX12 MEM.STK.DRW.	1

Table 6-3. Replaceable Parts List, ND812 8K Memory, Part Number 88-0097

ND PART NUMBER	DESCRIPTION	QUANTITY
21-0007	RES MG QW 5% 10 OHM	25
21-0031	RES MG QW 5% 100 OHM	48
21-0033	RES MG QW 5% 120 OHM	24
21-0041	RES MG QW 5% 270 OHM	54

Table 6-3. Replaceable Parts List, ND812 8K Memory, Part Number 88-0097 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
21-0043	RES MG QW 5% 330 OHM	4
21-0047	RES MG QW 5% 470 OHM	6
21-0051	RES MG QW 5% 680 OHM	32
21-0055	RES MG QW 5% 1K	73
21-0063	RES MG QW 5% 2.2K	1
21-0071	RES MG QW 5% 4.7K	8
21-0079	RES MG QW 5% 10K	25
21-0083	RES MG QW 5% 15K	2
21-0091	RES MG QW 5% 33K	1
21-5132	RES MF QW 1% 27.4	8
23-0022	RES WW 5W 5% 100 OHM	24
23-0045	RES WW 5W 5% 180 OHM	4
24-0097	RSVR HW 5% 10K	4
24-0111	RSVR HS 5% 10 OHM	1
24-5003	THRM 50K	2
26-0002	CAP SM 500MV 15PF	1
26-0006	CAP SM 500WV 47PF	2
26-0007	CAP SM 500WV 75PF	33
26-0008	CAP SM 500WV 150PF	3
26-0009	CAP SM 500WV 200PF	24
26-0026	CAP SM 500WV 100PF	24
26-0028	CAP SM 300WV 750PF	2
26-1004	CAP MY 100WV .01MFD	1
26-1037	CAP MY 250WV .1MFD	42
26-2024	CAP TM 22MFD 15V	8
26-4005	CAP CR .01MFD 150WV	8
26-5012	CAP EY 25MFD 50WV AL	12
31-0006	DO ZR IN4736A	2
31-0063	DO SL IN5059	3
31-0069	DO SL HI SPD	127
33-0045	TR SL NPN 2N4124	2
33-0053	TR SL PNP 2N4126	4
33-0065	TR SL NPN 2N3646	2
33-0103	TR SL NPN 2N3725	60

Table 6-3. Replaceable Parts List, ND812 8K Memory, Part Number 88-0097 (Cont'd.)

ND PART NUMBER	DESCRIPTION	QUANTITY
33-0104	TR SL PNP 2N2905A	8
33-0106	TR SL PNP 2N4125	32
35-0004	DTL932 DIP 14 DU BUF	1
35-0033	IC MC857P DIP 14 G	6
35-0034	IC MC858P DIP 14 G	4
35-0039	TTL 9300 DIP 16 SFTR	6
35-0041	TTL 9601 DIP 14 MONO	6
35-0055	IC MC1540G TO5 10 SA	24
35-0058	IC 9002 DIP 14 G	6
35-0059	IC 9003 DIP 14 G	4
35-0060	IC 9004 DIP 14 G	1
35-0061	IC 9005 DIP 14 G	24
35-0063	IC 9007 DIP 14 G	1
35-0064	IC 9008 DIP 14 G	6
35-0065	IC 9009 DIP 14 INBUF	2
35-0066	IC 9016 DIP 14 HXVT	7
35-0069	MSI 9301 DIP 16 DCDR	4
35-0072	IC 9022 DIP 16 DU JK	4
35-0123	IC MSI 8200 DIP 14	3
38-0031	XTF DUL PLS DIP	14
38-0032	XTF DUL BALUN DIP	2
39-2001	HT DSP TO-5 FAN	8
42-0006	WRP MFC-A	1
50-0176	PCB MIS	2
50-0177	PCB MTS	1
50-0184	PCB WW4	1
70-1483	MIS-A	2
70-1502	MTS-A	1
70-1686	MFC-A	1
73-0039	4K X 24/30 MIL56CNN	1

SECTION VII
DIAGRAMS

7.1 GENERAL

This section contains schematic and logic diagrams to be used for troubleshooting the ND812 Central Processor, ND812 4K Memory, and ND812 8K Memory. Table 7-1 provides an index for diagrams contained in this section. Figure 7-1 provides general notes which apply to logic diagrams in this section; and illustrates how to identify connectors, connector pins, integrated circuit types, printed circuit board name, and location of integrated circuits on printed circuit board. Figures 7-2, 7-3, and 7-4 provide logic/schematic diagrams for the ND812 Central Processor, 4K Memory, and 8K Memory respectively. Figures 7-5 through 7-12 provide a Logic Diagram/Functional Description for Medium Scale Integrated Circuits (MSI's) used in the ND812 Central Processor and Memory.

Table 7-1. Diagram Index

Table 7-1. Diagram Index (Cont'd.)

Table 7-1. Diagram Index (Cont'd.)

NOTES:

1 - ALL DIODES ARE G964 OR EQUIVALENT, EXCEPT AS NOTED.

2 - ALL RESISTORS ARE 1/4W, ±5%, EXCEPT AS NOTED.

3 - ALL CAPACITORS ARE pf, EXCEPT AS NOTED.

4 - I.C. VOLTAGES, EXCEPT AS NOTED:
14 PIN DIP, PIN (7) GND: PIN (14) +5V
16 PIN DIP, PIN (8) GND: PIN (16) +5V
24 PIN DIP, PIN (12) GND: PIN (24) +5V

5 - THE FOLLOWING SYMBOLS/NOTATIONS ARE USED ON THE DIAGRAM AND/OR PRINTED CIRCUIT BOARD ASSEMBLY.

IC - INTEGRATED CIRCUIT	SAT - SELECT AT TEST	⇥ - GERMANIUM DIODE
Q - TRANSISTOR	(P1) - PRECISION RESISTORS 100PPM 1/8W, ±1% METAL FILM	▶| - SILICON DIODE
( ) - IC PIN DESIGNATION		▶| - ZENER DIODE
≫ - CONNECTOR DESIGNATION	⏚○ - DC COMMON	▶| - TUNNEL DIODE
NC - NO CONNECTION	[FB] - FERRITE BEAD	▷| - SELENIUM DIODE

ADC* — SIGNAL NAME
[4A2] — SIGNAL SOURCE DRAWING LOCATION, OR
(4A2) — SIGNAL LOAD DRAWING LOCATION
ZONE LOCATION
SHEET NUMBER

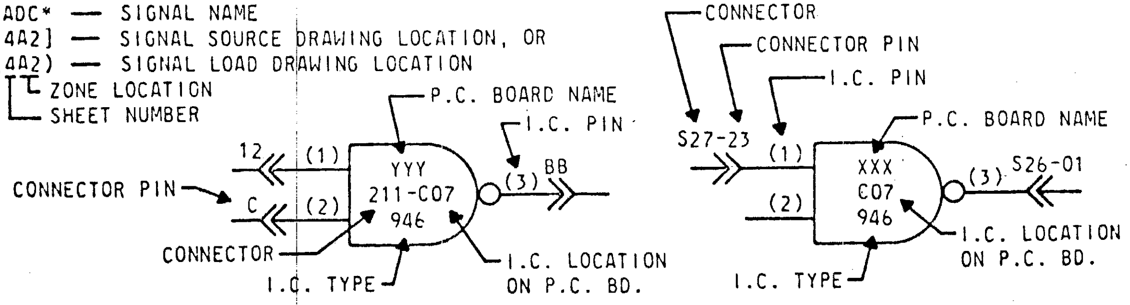


Figure 7-1. Logic Diagram General Notes

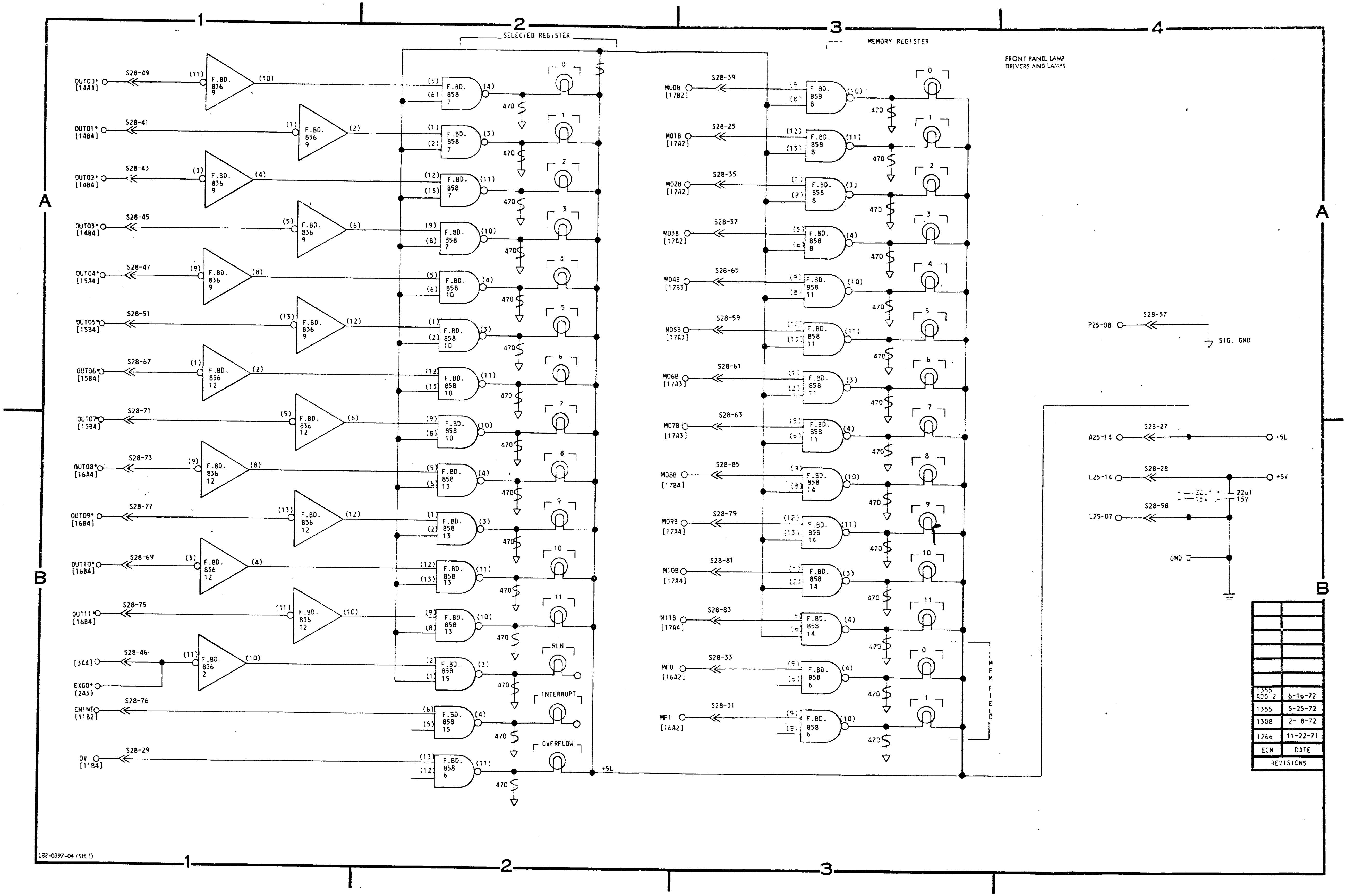


Figure 7-2. ND312, Front Panel Logic Diagram (Sheet 1 of 28)

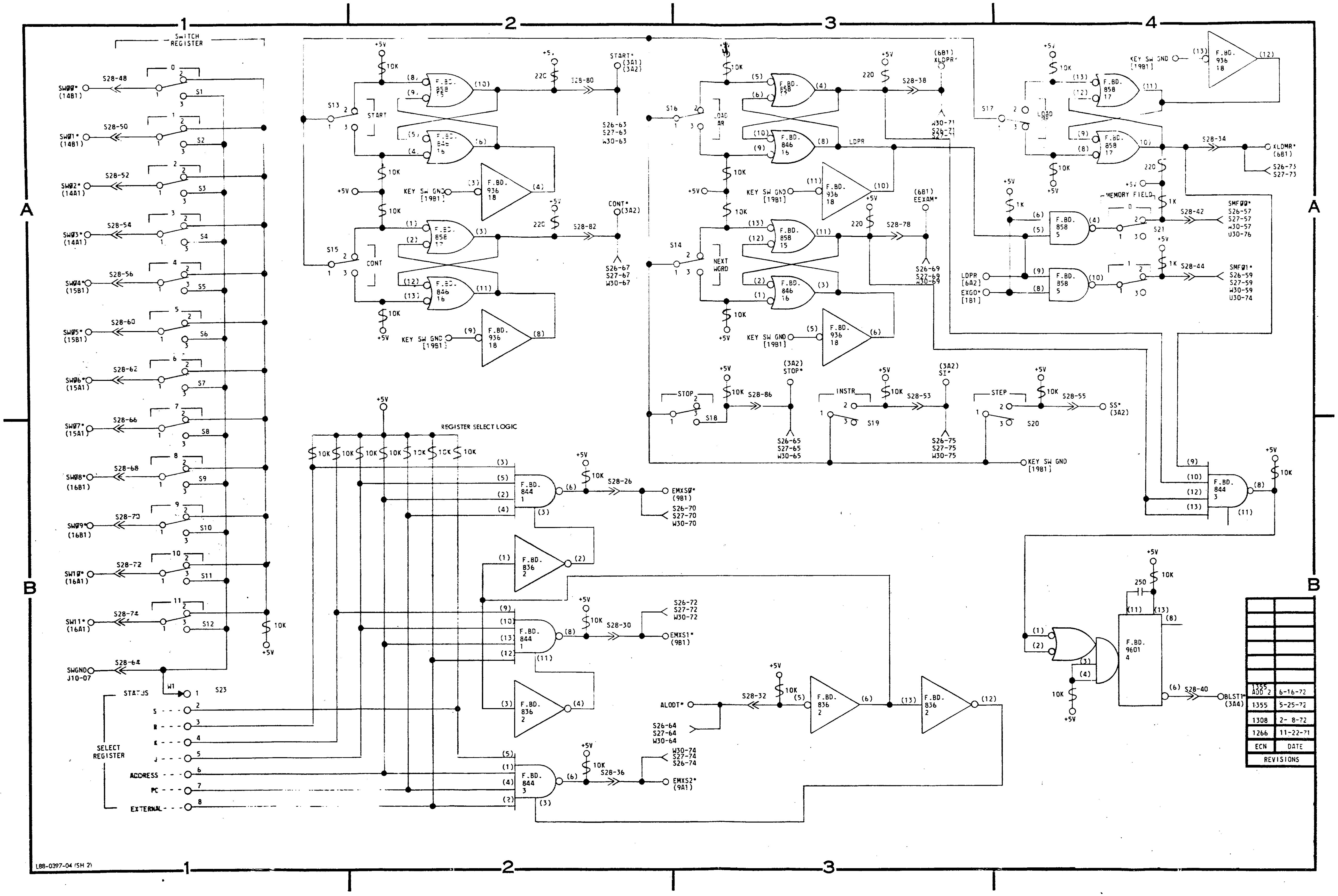
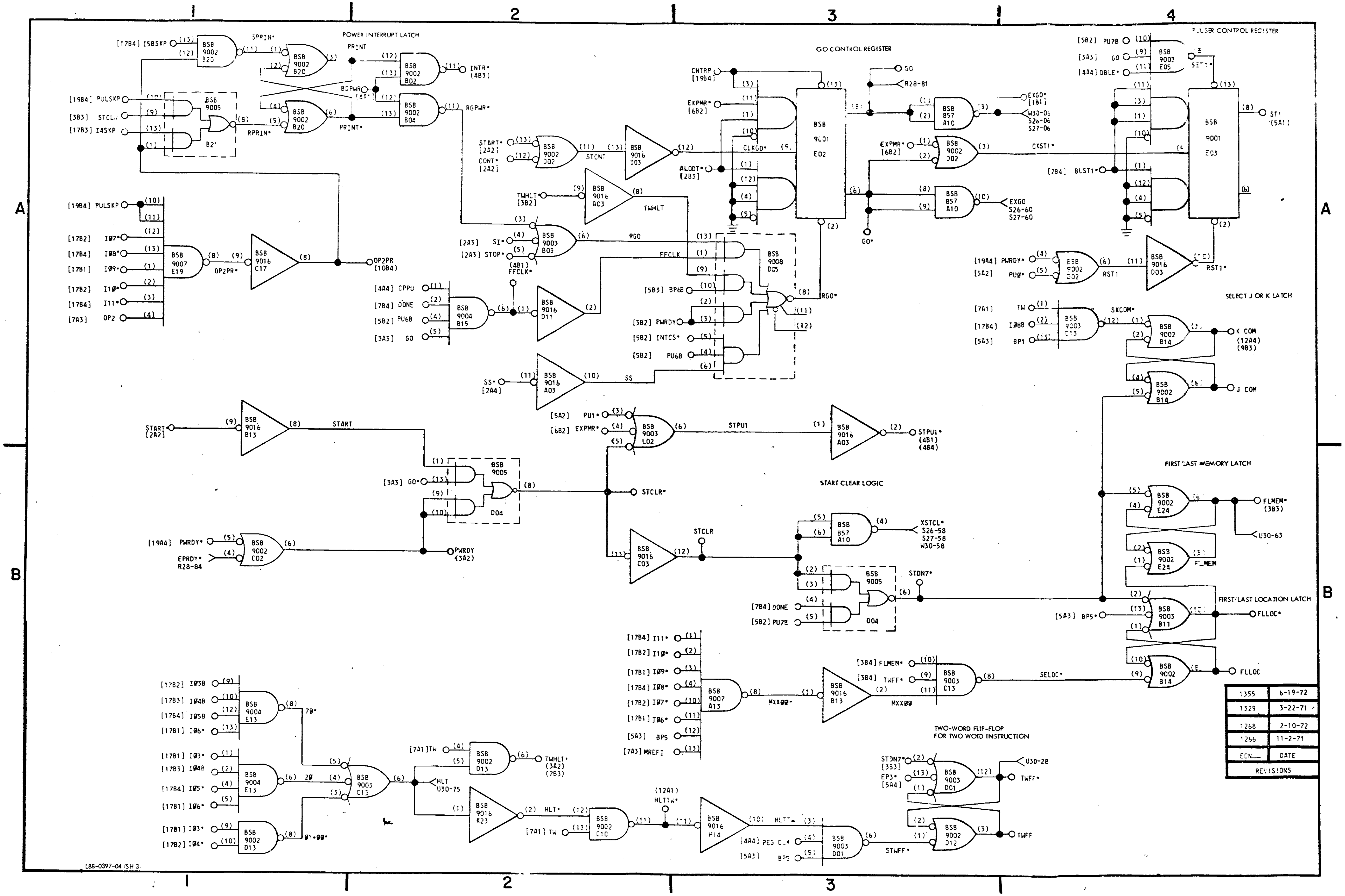


Figure 7-2. ND812, Front Panel Logic Diagram (Sheet 2 of 28)



1355	6-19-72
1329	3-22-71
1268	2-10-72
1266	11-2-71
ECN	DATE
REVISIONS	

Figure 7-2. ND812, Start Clear, First/ Last Memory Latch, and Two-Word, Logic Diagram (Sheet 3 of 28)

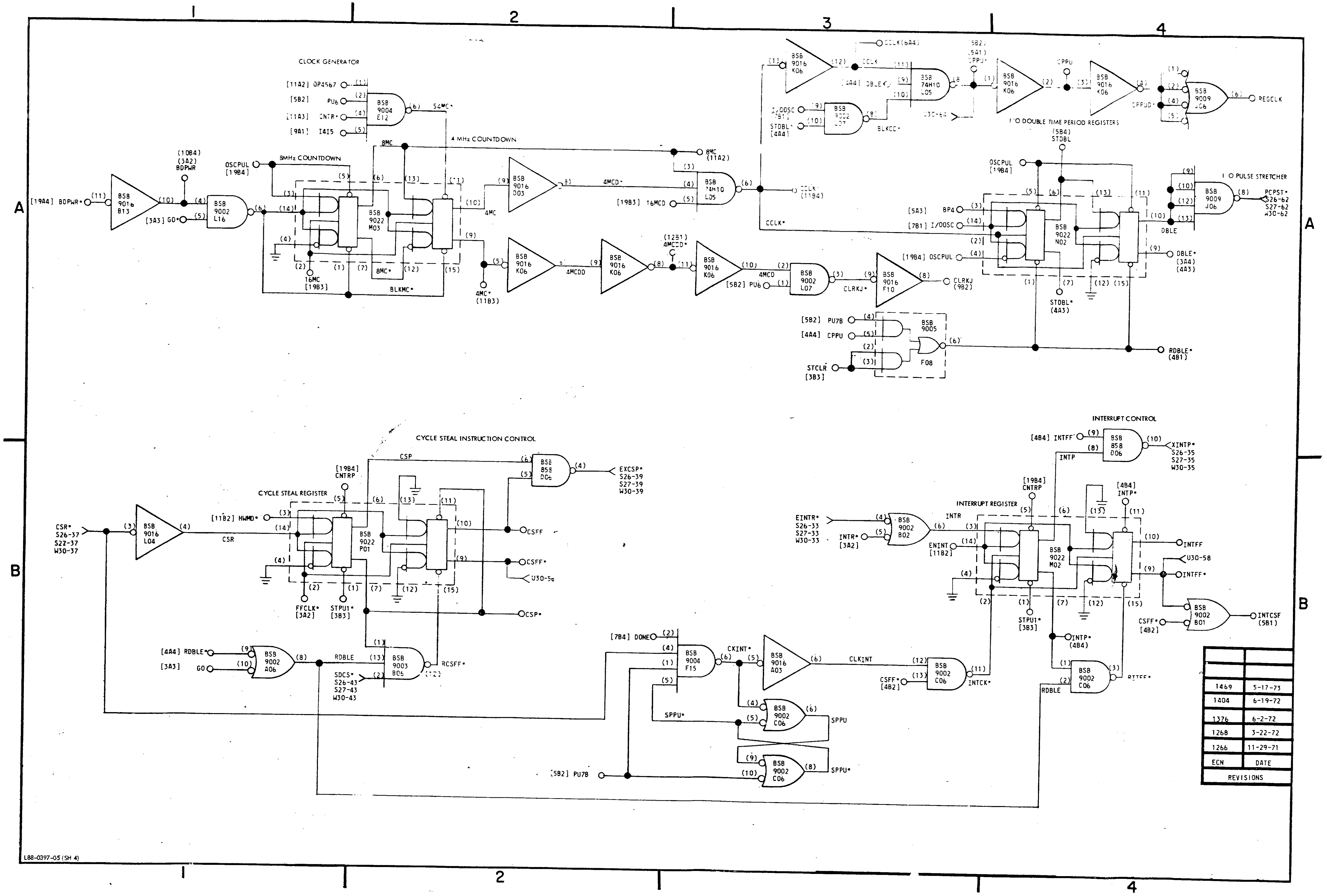


Figure 7-2. ND812 Clock, DMA, and Interrupt Control, Logic Diagram (Sheet 4 of 28)

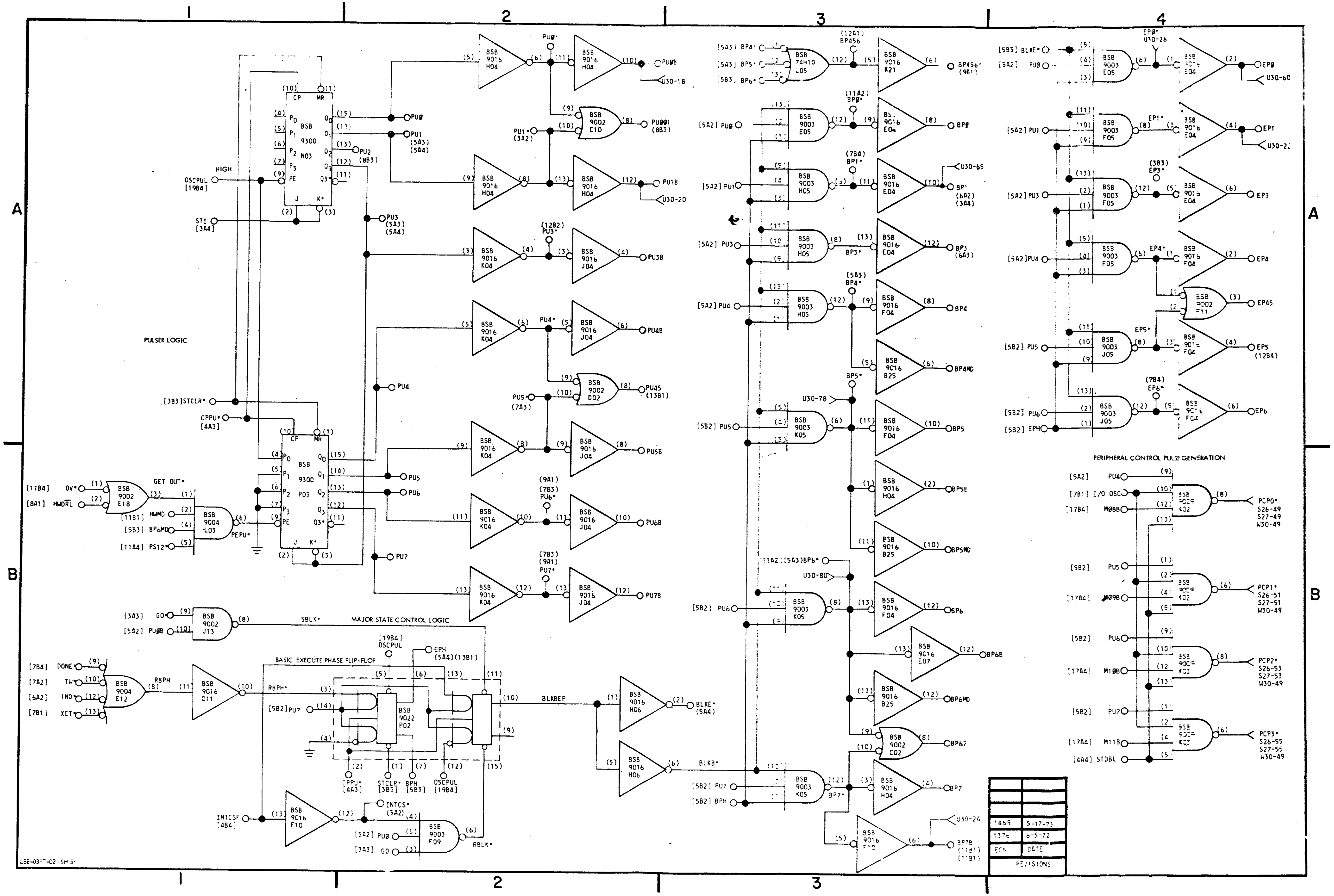


Figure 7-2. ND812, Pulser, Logic Diagram (Sheet 5 of 28)

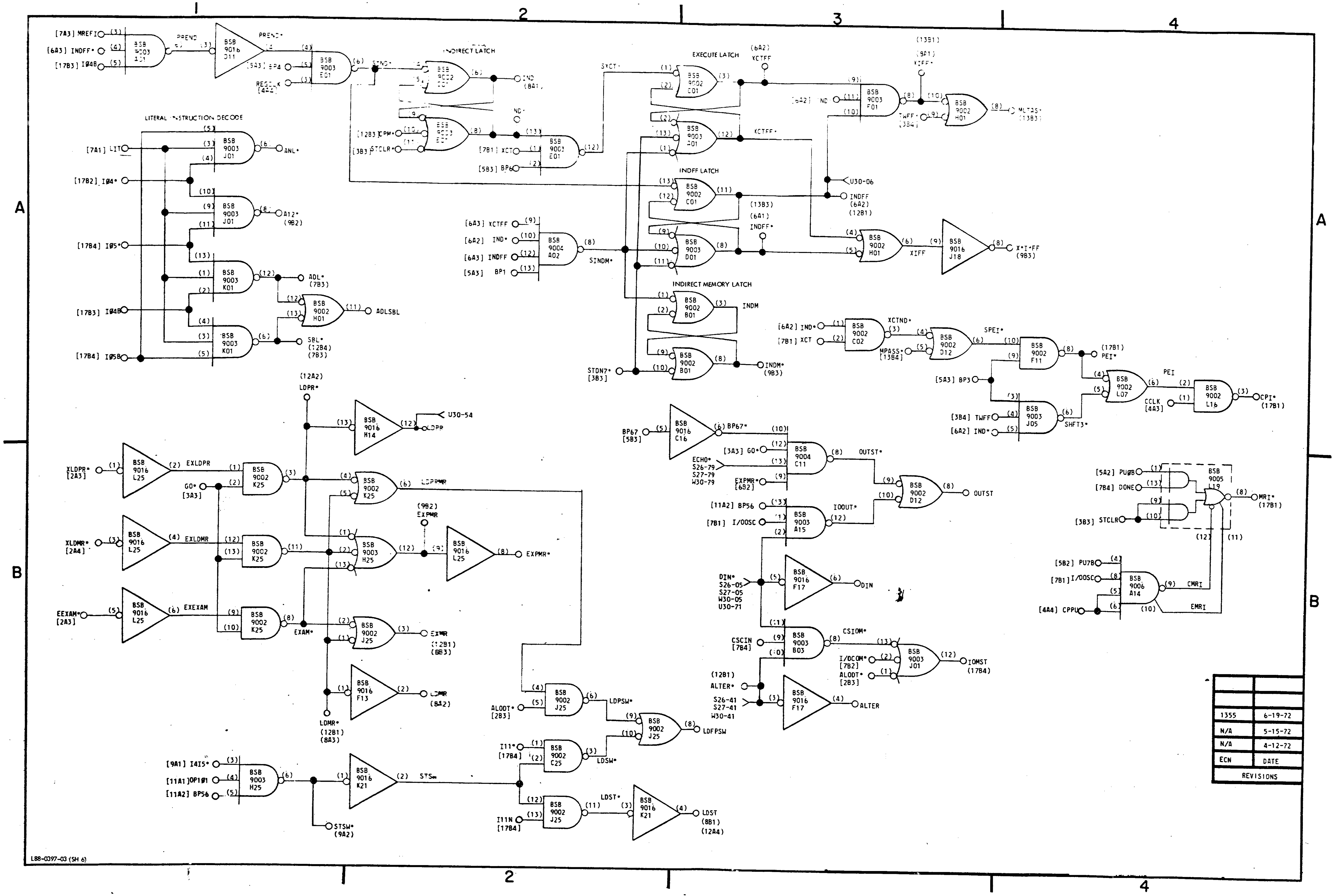


Figure 7-2. ND812, Literal Instruction Decode and Indirect Latch, Logic Diagram (Sheet 6 of 28)

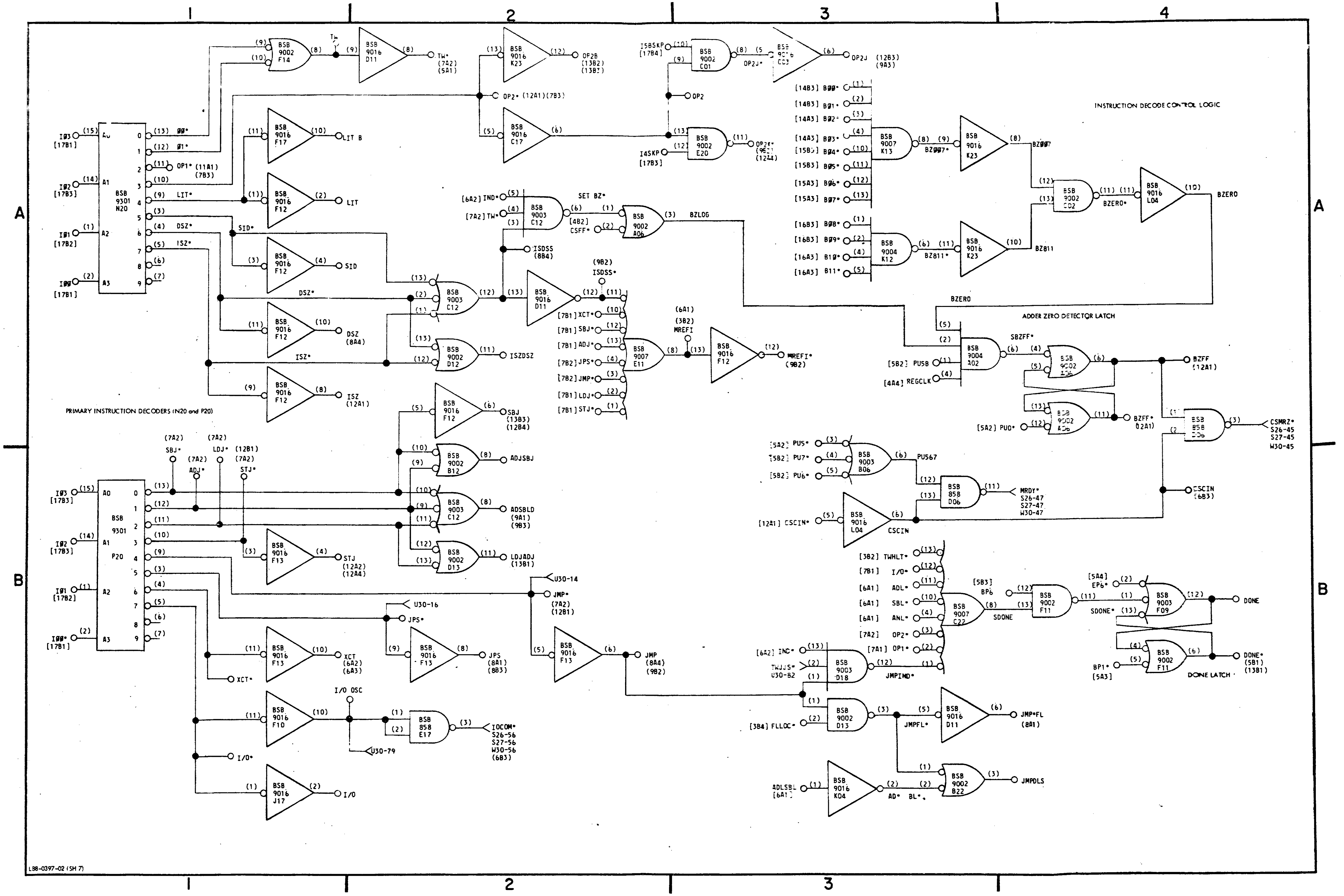


Figure 7-2. ND812, Instruction Decode and Adder Zero Detect, Logic Diagram (Sheet 7 of 28)

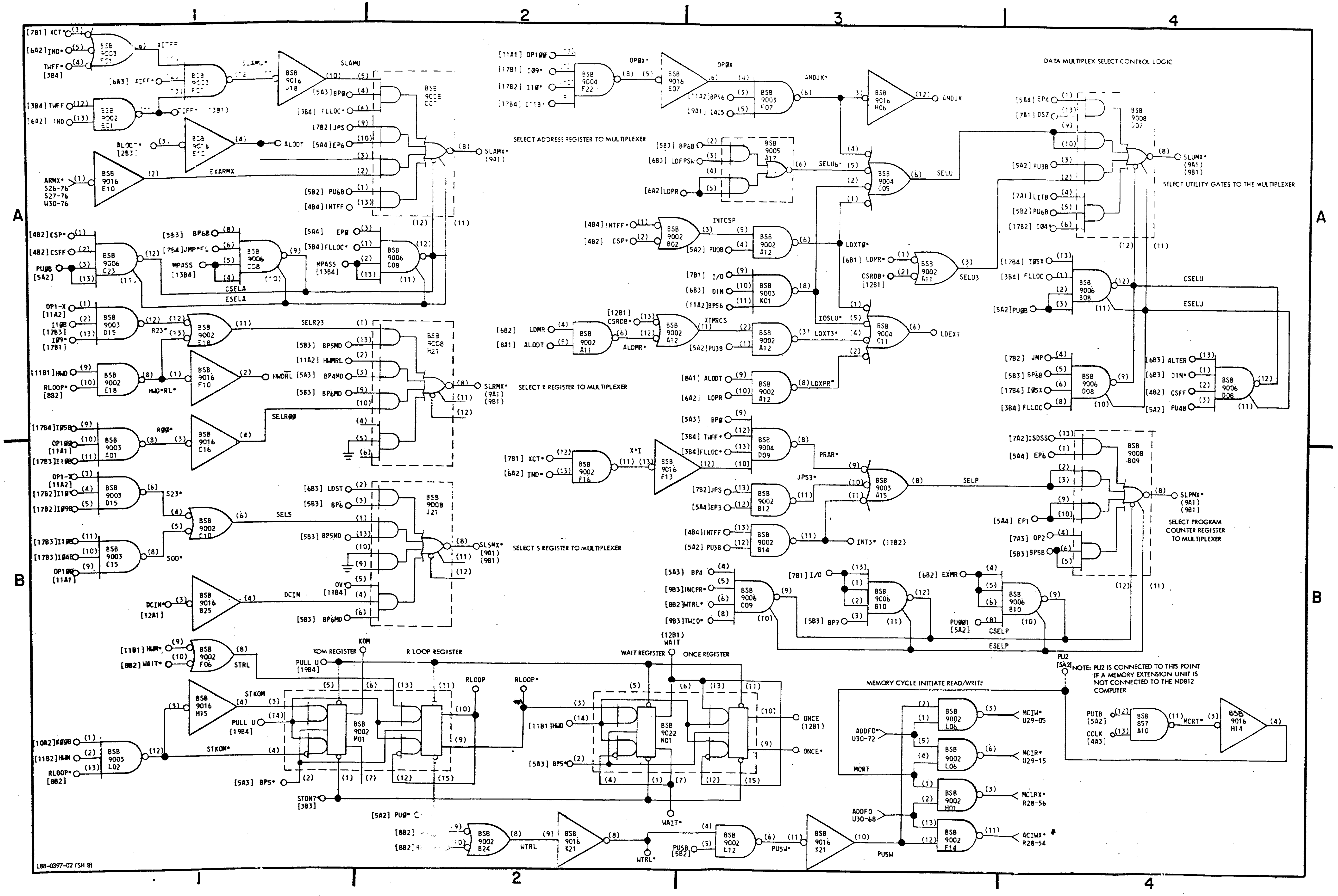

1
2
3
4
A
B
DATA MULTIPLEX SELECT CONTROL LOGIC
SELECT ADDRESS REGISTER TO MULTIPLEXER
SELECT UTILITY GATES TO THE MULTIPLEXER
SELECT R REGISTER TO MULTIPLEXER
SELECT S REGISTER TO MULTIPLEXER
SELECT PROGRAM COUNTER REGISTER TO MULTIPLEXER
KOM REGISTER
R LOOP REGISTER
WAIT REGISTER
ONCE REGISTER
MEMORY CYCLE INITIATE READ/WRITE
NOTE: PU2 IS CONNECTED TO THIS POINT IF A MEMORY EXTENSION UNIT IS NOT CONNECTED TO THE ND812 COMPUTER
SLAMX*
SLRMX*
SLSMX*
SLUMX*
SLPMX*
SELU
SELP
CSELU
ESELU
CSELP
ESELP
LDEXT
ANDJK
INT3* (11B2)
ONCE (12B1)
ONCE*
MCIW* U29-05
MCIR* U29-15
MCLRX* R28-56
ACIWX* R28-54
MCRT*
STKOM
STRL
RLOOP
RLOOP*
WAIT
WAIT*
WTRL
WTRL*
PUSW
L88-0397-02 (SH 8)

Figure 7-2. ND812, Multiplex Select and R Loop, Logic Diagram (Sheet 8 of 28)

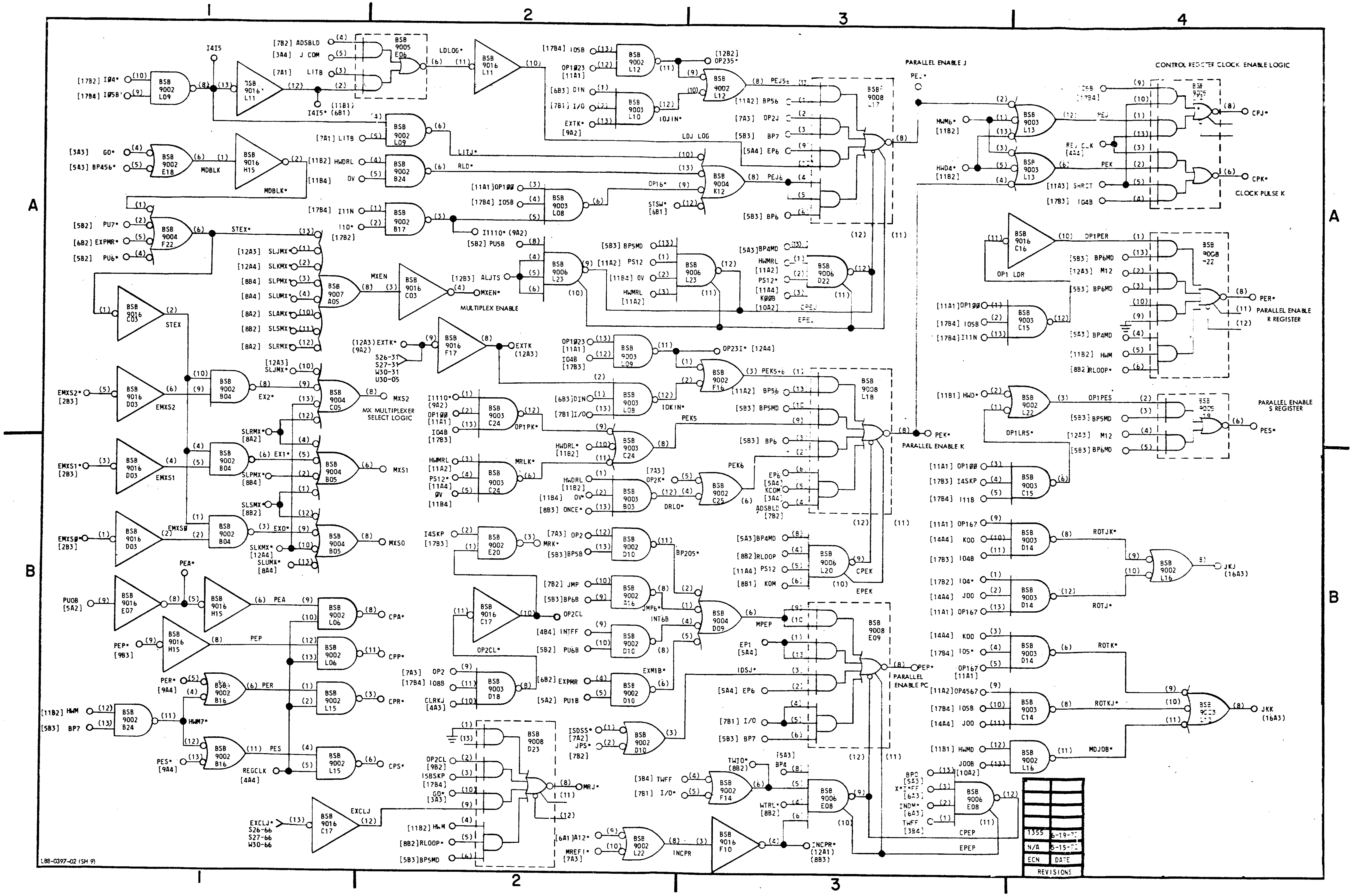


Figure 7-2. ND812, Parallel Enable, Logic Diagram (Sheet 9 of 28)

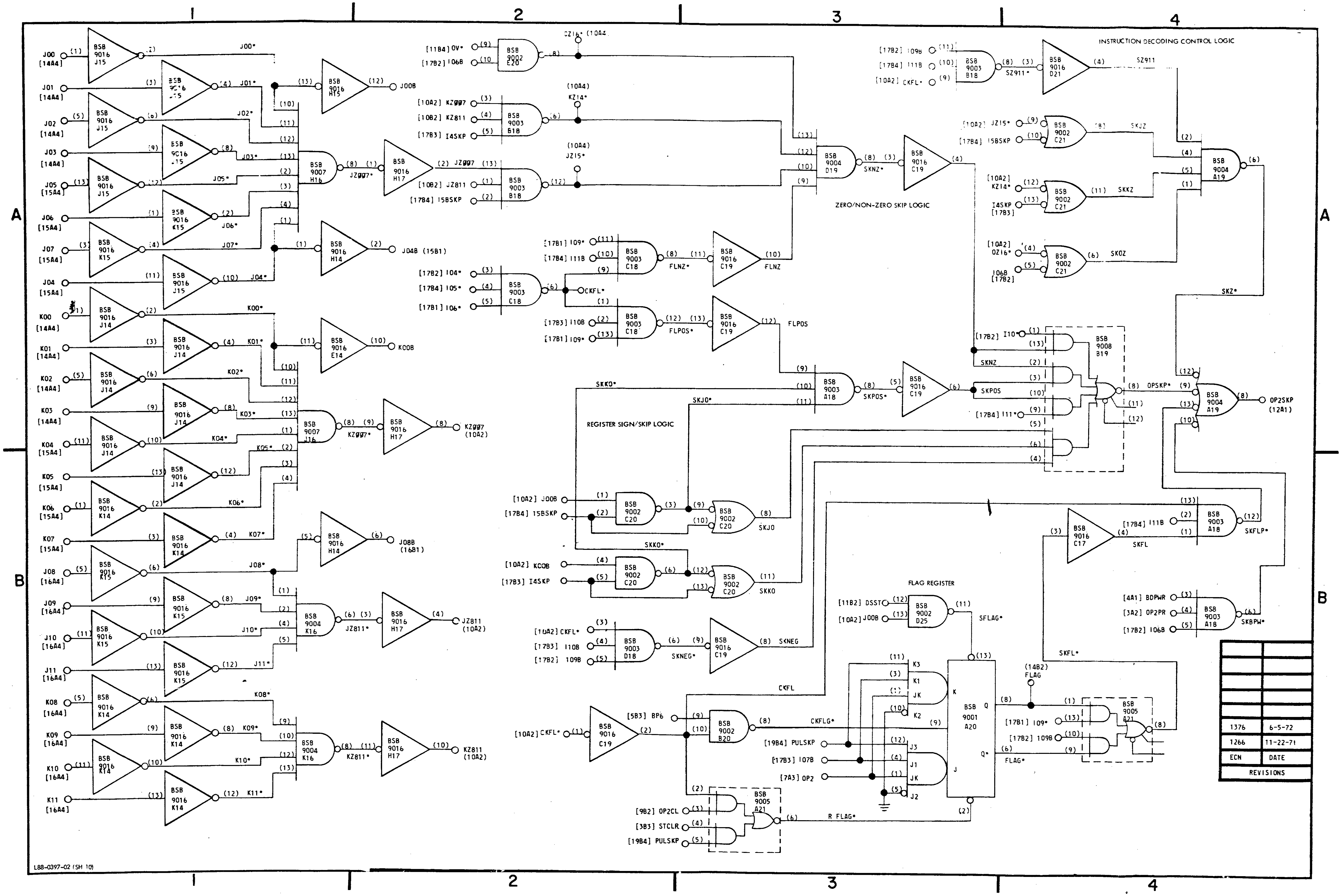


Figure 7-2. ND812, Zero/Non-Zero Skip and Flag Register, Logic Diagram (Sheet 10 of 28)

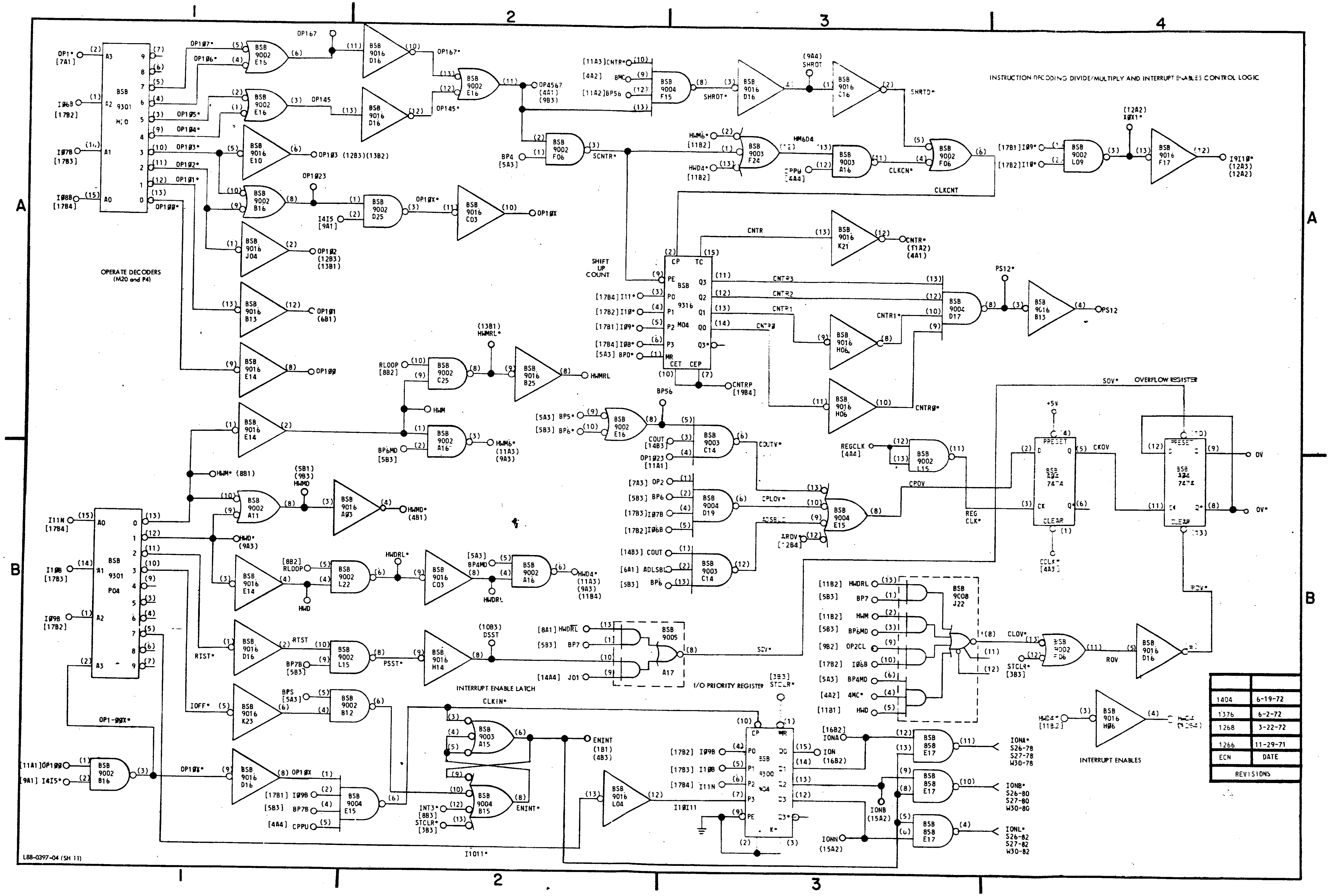


Figure 7-2. ND812, Operate Decoders, Shift Up Counter, and Overflow, Logic Diagram (Sheet 11 of 28)

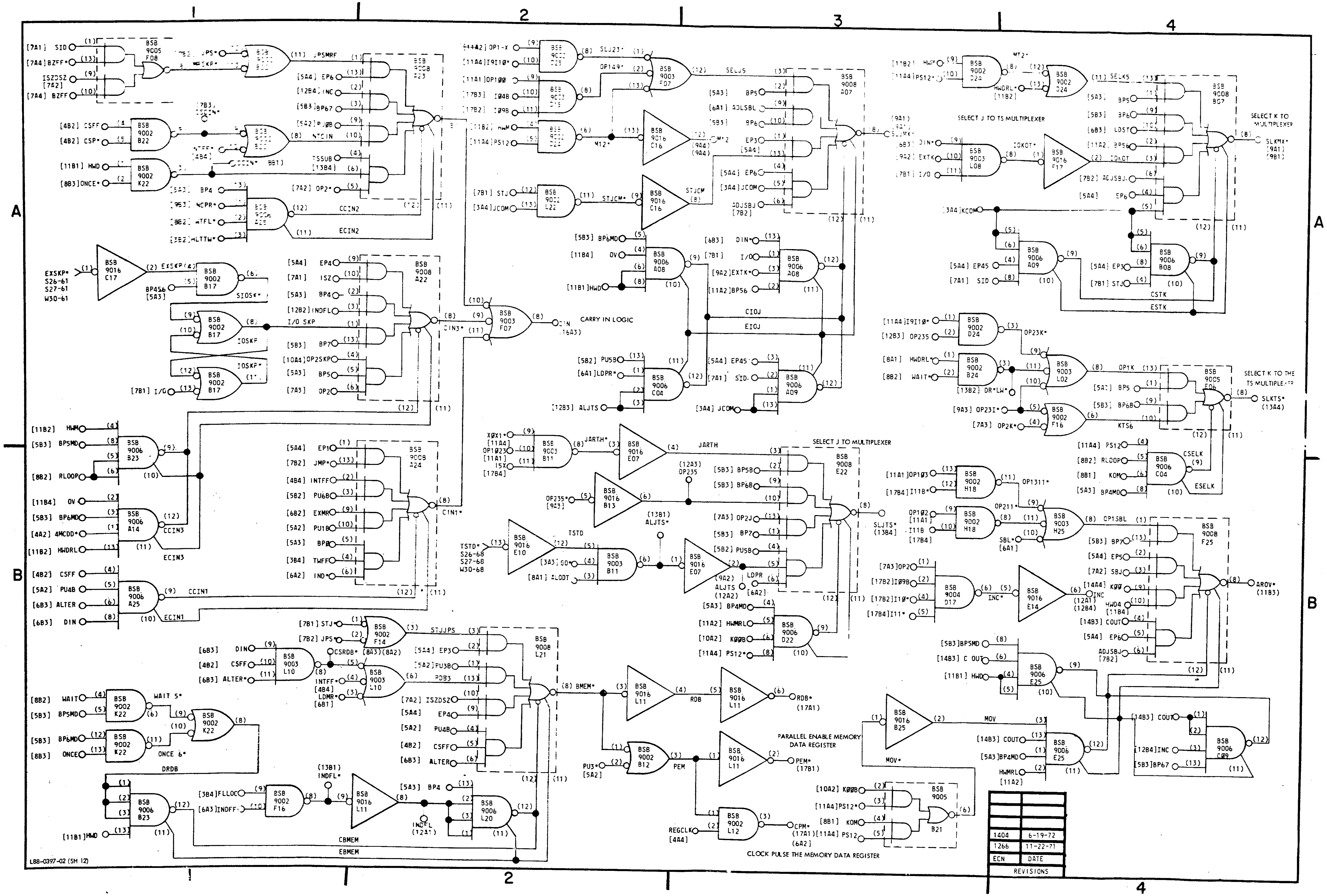


Figure 7-2. ND812, Carry In, Select J and K, Logic Diagram (Sheet 12 of 28)

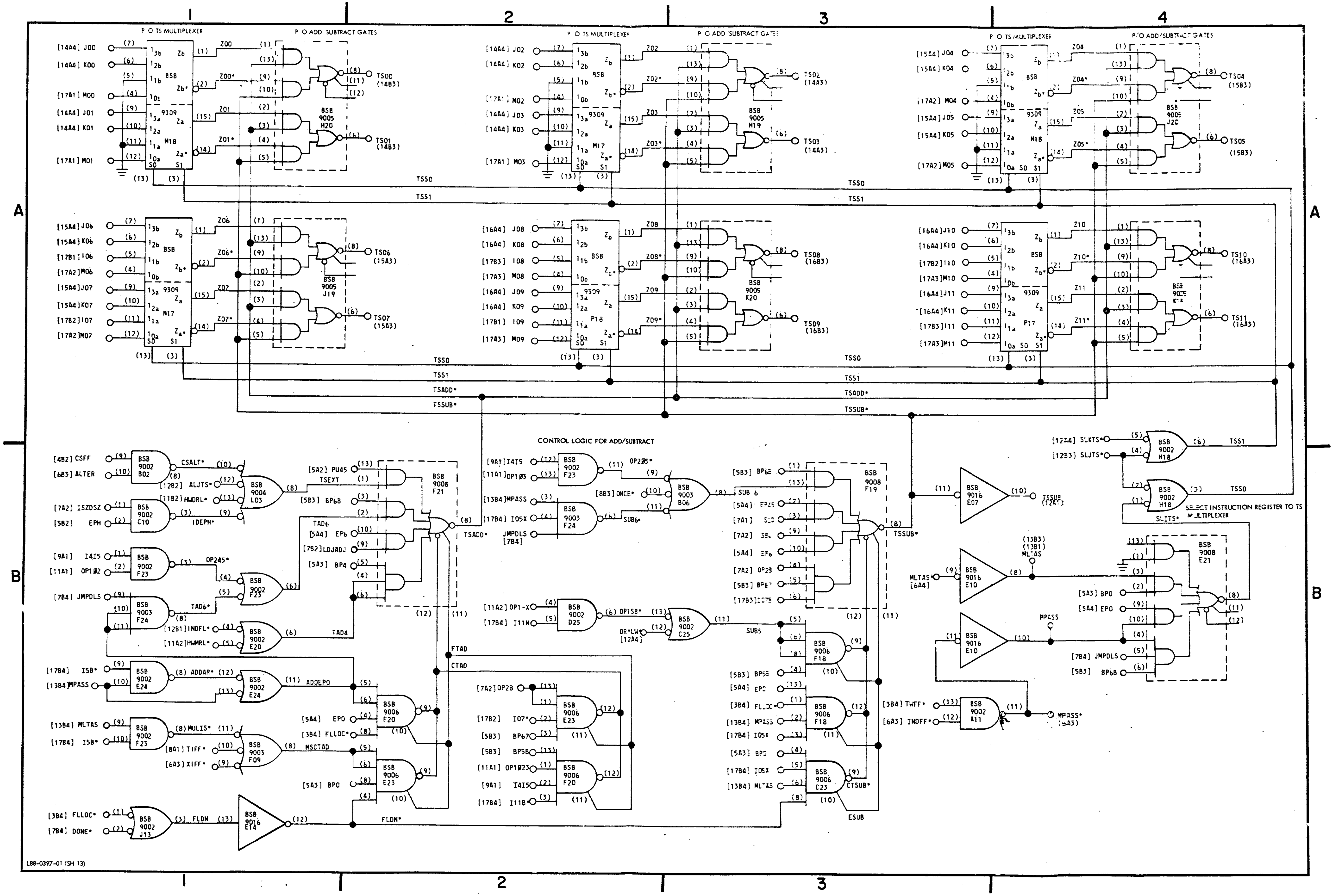


Figure 7-2. ND812, Add and Subtract Control, Logic Diagram (Sheet 13 of 28)

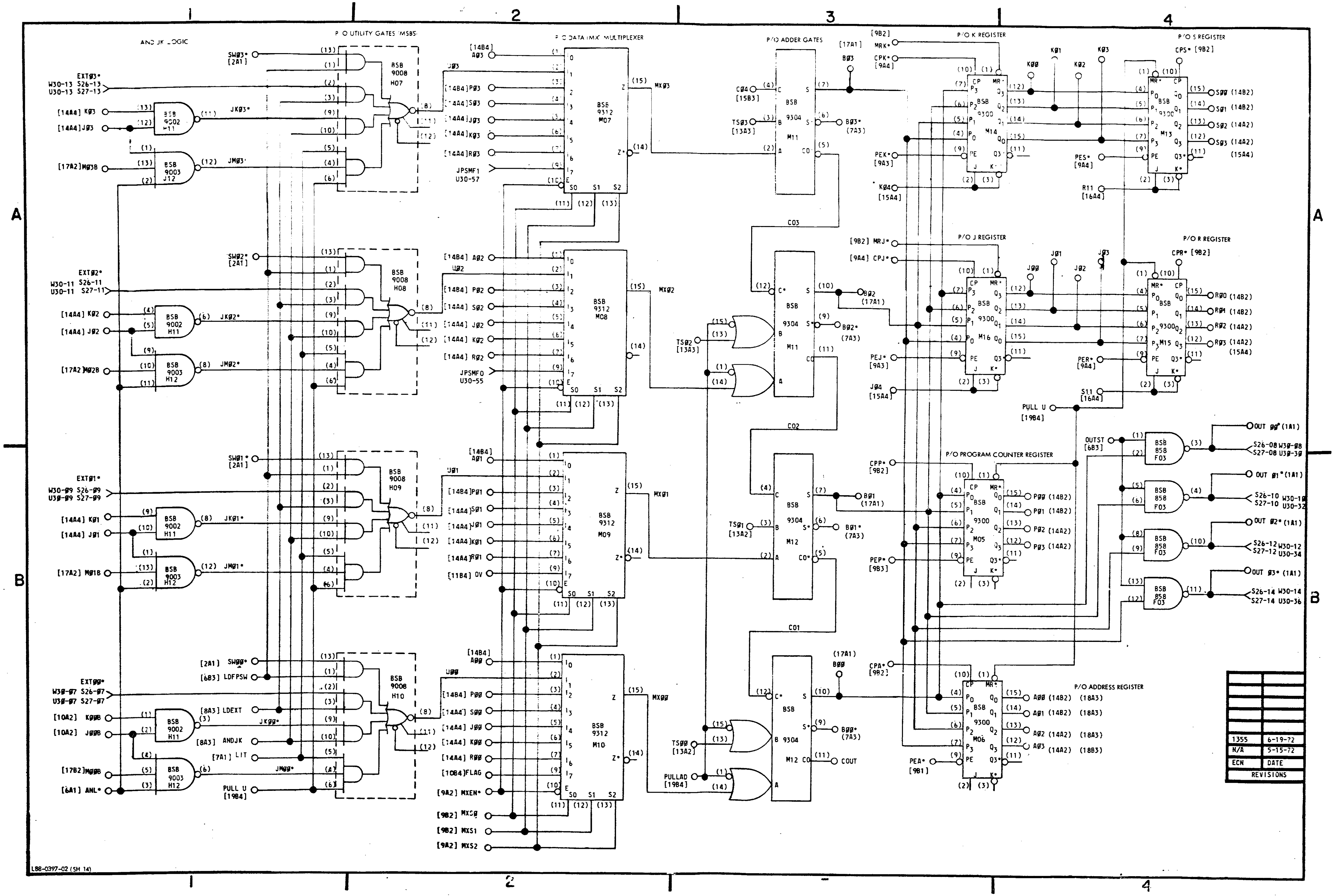


Figure 7-2. ND812, AND JK, Data Multiplexer, and Adder Gates; and J, K, R, S, AR, and PC Register, Logic Diagram (Sheet 14 of 28)

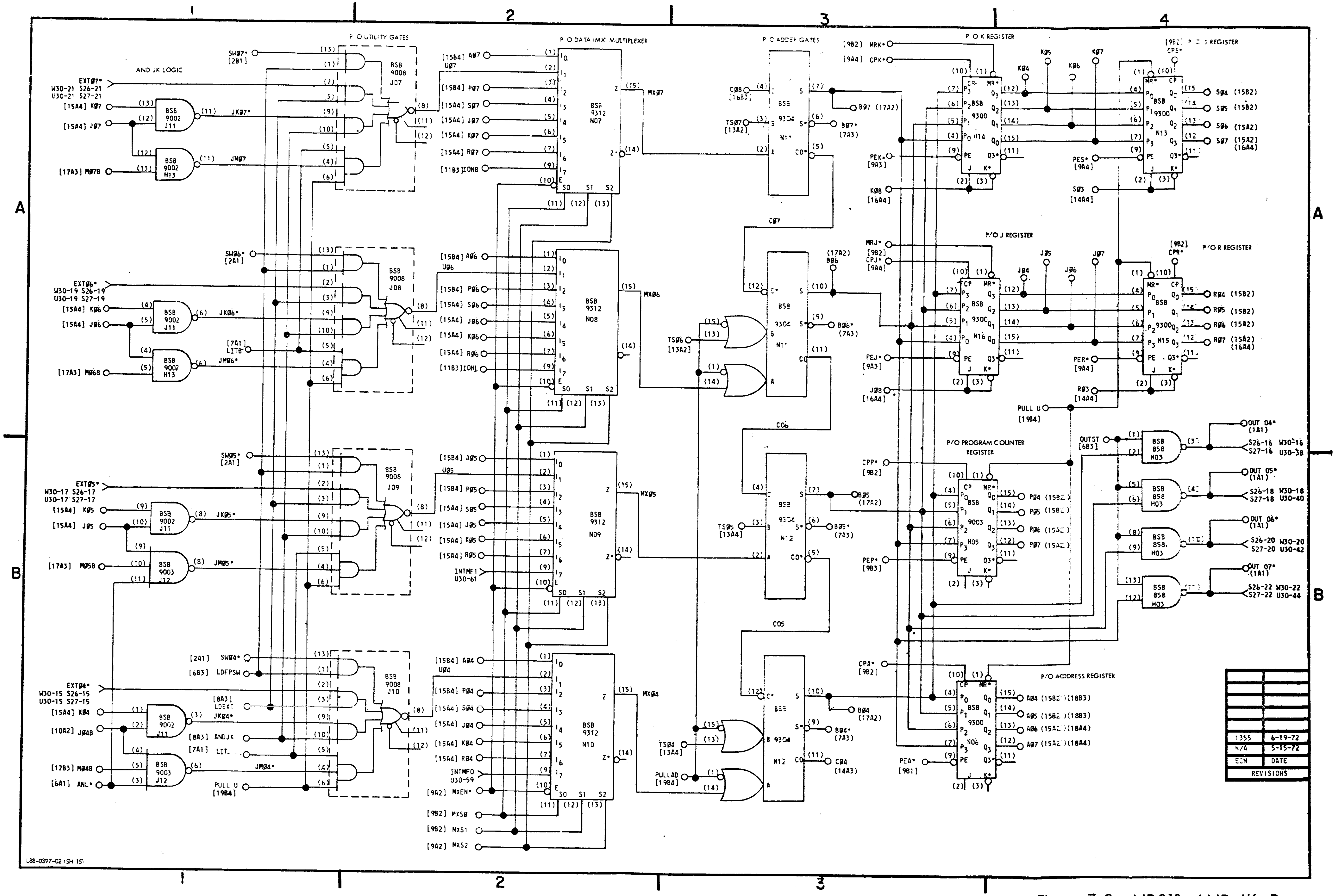


Figure 7-2. ND812, AND JK, Data Multiplexer, and Adder Gates; and J, K, R, S, AR, and PC Registers, Logic Diagram (Sheet 15 of 28)

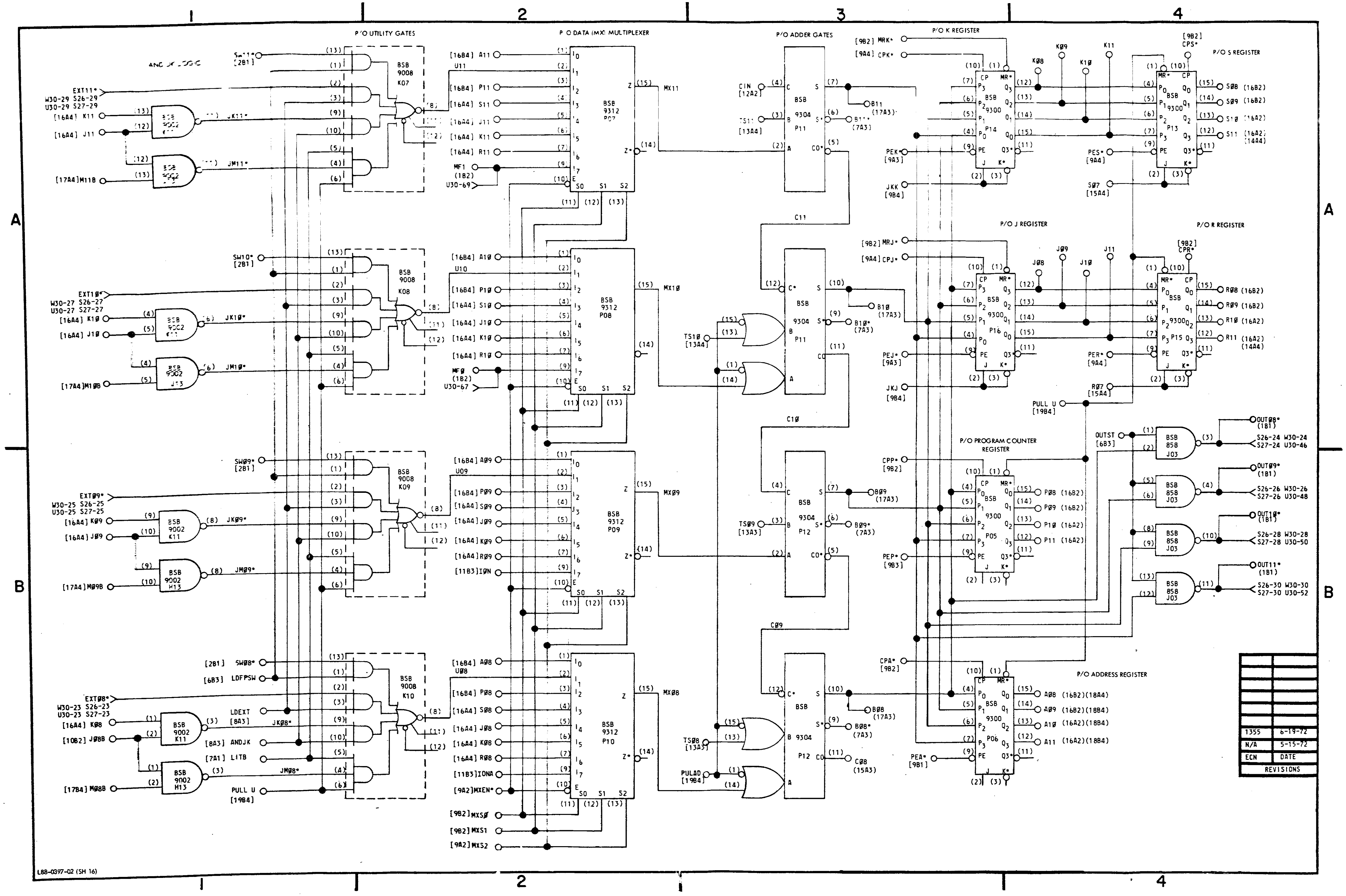


Figure 7-2. ND812, AND JK, Data Multiplexer, and Adder Gates; and J, K, R, S, AR, and PC Registers, Logic Diagram (Sheet 16 of 28)

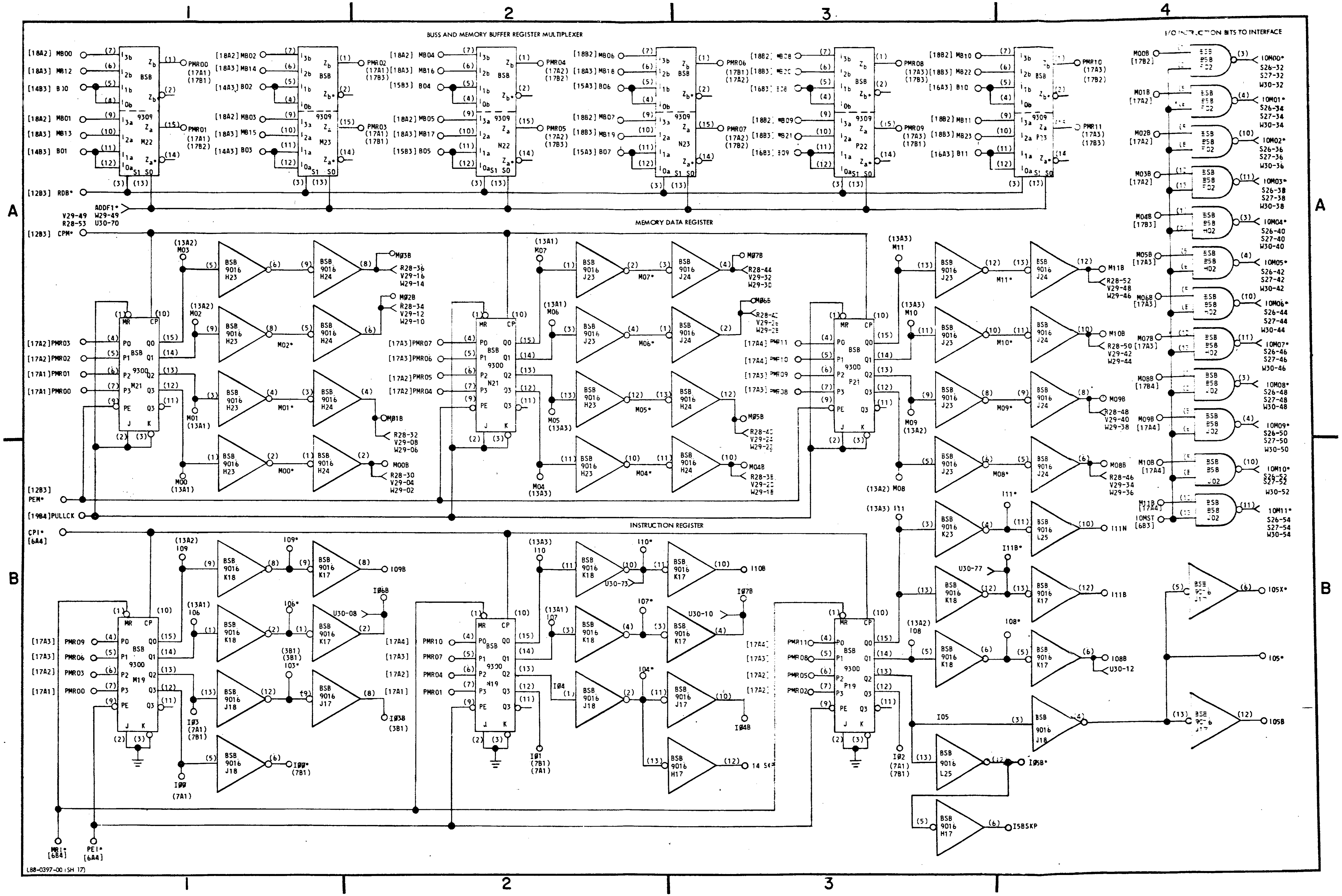


Figure 7-2. ND812, Memory Data Register and Instruction Register, Logic Diagram (Sheet 17 of 28)

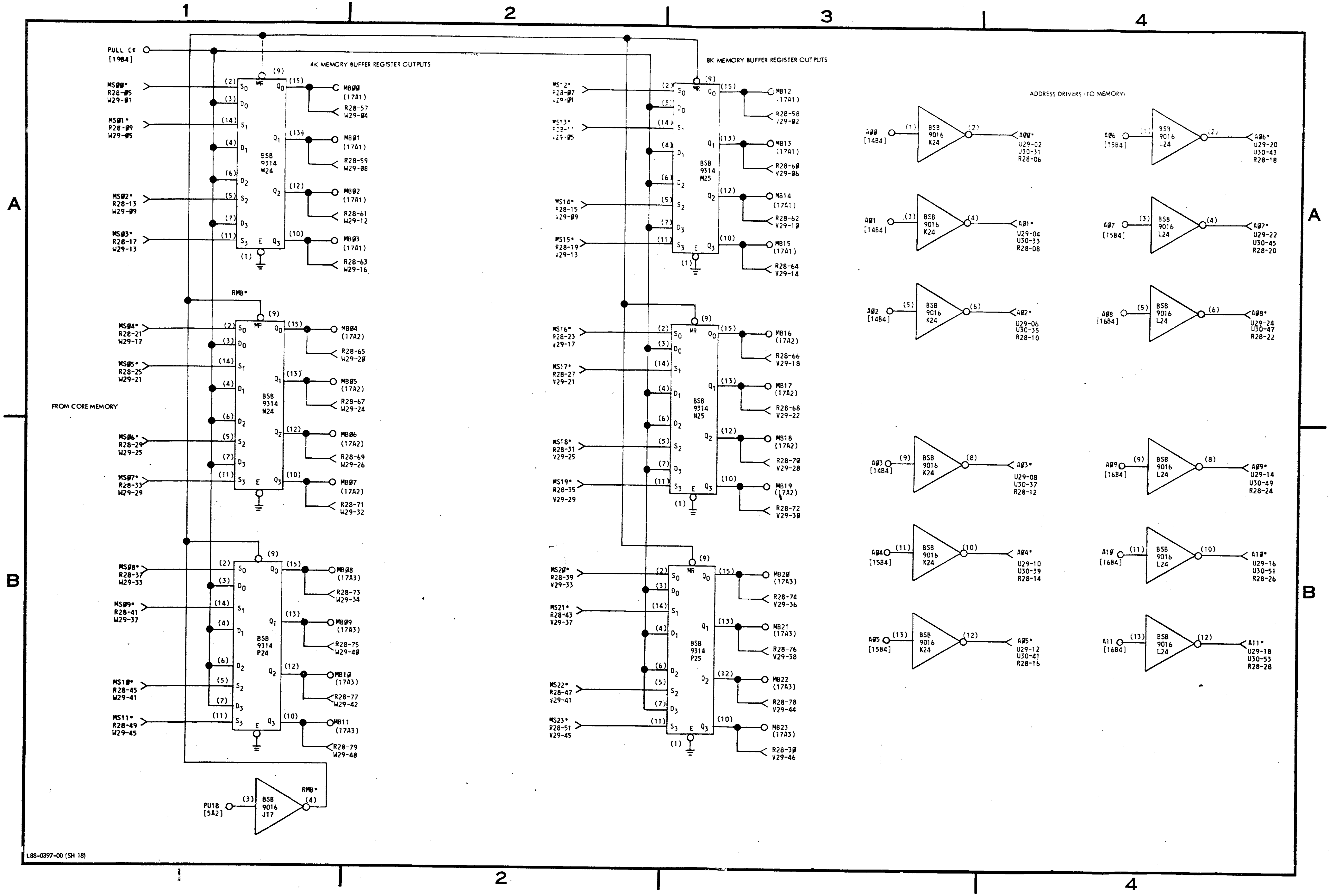

1
2
3
4
A
B
PULL CK [1984]
4K MEMORY BUFFER REGISTER OUTPUTS
8K MEMORY BUFFER REGISTER OUTPUTS
ADDRESS DRIVERS (TO MEMORY)
FROM CORE MEMORY
RMB*
MSØØ* R28-Ø5 W29-Ø1
MSØ1* R28-Ø9 W29-Ø5
MSØ2* R28-13 W29-Ø9
MSØ3* R28-17 W29-13
BSB 9314 M24
MBØØ (17A1) R28-57 W29-Ø4
MBØ1 (17A1) R28-59 W29-Ø8
MBØ2 (17A1) R28-61 W29-12
MBØ3 (17A1) R28-63 W29-16
MSØ4* R28-21 W29-17
MSØ5* R28-25 W29-21
MSØ6* R28-29 W29-25
MSØ7* R28-33 W29-29
BSB 9314 N24
MBØ4 (17A2) R28-65 W29-2Ø
MBØ5 (17A2) R28-67 W29-24
MBØ6 (17A2) R28-69 W29-26
MBØ7 (17A2) R28-71 W29-32
MSØ8* R28-37 W29-33
MSØ9* R28-41 W29-37
MS1Ø* R28-45 W29-41
MS11* R28-49 W29-45
BSB 9314 P24
MBØ8 (17A3) R28-73 W29-34
MBØ9 (17A3) R28-75 W29-4Ø
MB1Ø (17A3) R28-77 W29-42
MB11 (17A3) R28-79 W29-48
PU1B [5A2]
BSB 9016 J17
RMB*
MS12* R28-Ø7 V29-Ø1
MS13* R28-11 V29-Ø5
MS14* R28-15 V29-Ø9
MS15* R28-19 V29-13
BSB 9314 M25
MB12 (17A1) R28-58 V29-Ø2
MB13 (17A1) R28-6Ø V29-Ø6
MB14 (17A1) R28-62 V29-1Ø
MB15 (17A1) R28-64 V29-14
MS16* R28-23 V29-17
MS17* R28-27 V29-21
MS18* R28-31 V29-25
MS19* R28-35 V29-29
BSB 9314 N25
MB16 (17A2) R28-66 V29-18
MB17 (17A2) R28-68 V29-22
MB18 (17A2) R28-7Ø V29-28
MB19 (17A2) R28-72 V29-3Ø
MS2Ø* R28-39 V29-33
MS21* R28-43 V29-37
MS22* R28-47 V29-41
MS23* R28-51 V29-45
BSB 9314 P25
MB2Ø (17A3) R28-74 V29-36
MB21 (17A3) R28-76 V29-38
MB22 (17A3) R28-78 V29-44
MB23 (17A3) R28-3Ø V29-46
S0 D0 S1 D1 D2 S2 D3 S3 E MR Q0 Q1 Q2 Q3
AØØ [14B4] BSB 9016 K24 AØØ* U29-02 U30-31 R28-06
AØ1 [14B4] BSB 9016 K24 AØ1* U29-04 U30-33 R28-08
AØ2 [14B4] BSB 9016 K24 AØ2* U29-06 U30-35 R28-10
AØ3 [14B4] BSB 9016 K24 AØ3* U29-08 U30-37 R28-12
AØ4 [15B4] BSB 9016 K24 AØ4* U29-10 U30-39 R28-14
AØ5 [15B4] BSB 9016 K24 AØ5* U29-12 U30-41 R28-16
AØ6 [15B4] BSB 9016 L24 AØ6* U29-20 U30-43 R28-18
AØ7 [15B4] BSB 9016 L24 AØ7* U29-22 U30-45 R28-20
AØ8 [16B4] BSB 9016 L24 AØ8* U29-24 U30-47 R28-22
AØ9 [16B4] BSB 9016 L24 AØ9* U29-14 U30-49 R28-24
A1Ø [16B4] BSB 9016 L24 A1Ø* U29-16 U30-51 R28-26
A11 [16B4] BSB 9016 L24 A11* U29-18 U30-53 R28-28
L88-0397-00 (SH 18)

Figure 7-2. ND812 Memory Buffer Register Output and Address Drivers, Logic Diagram (Sheet 18 of 28)

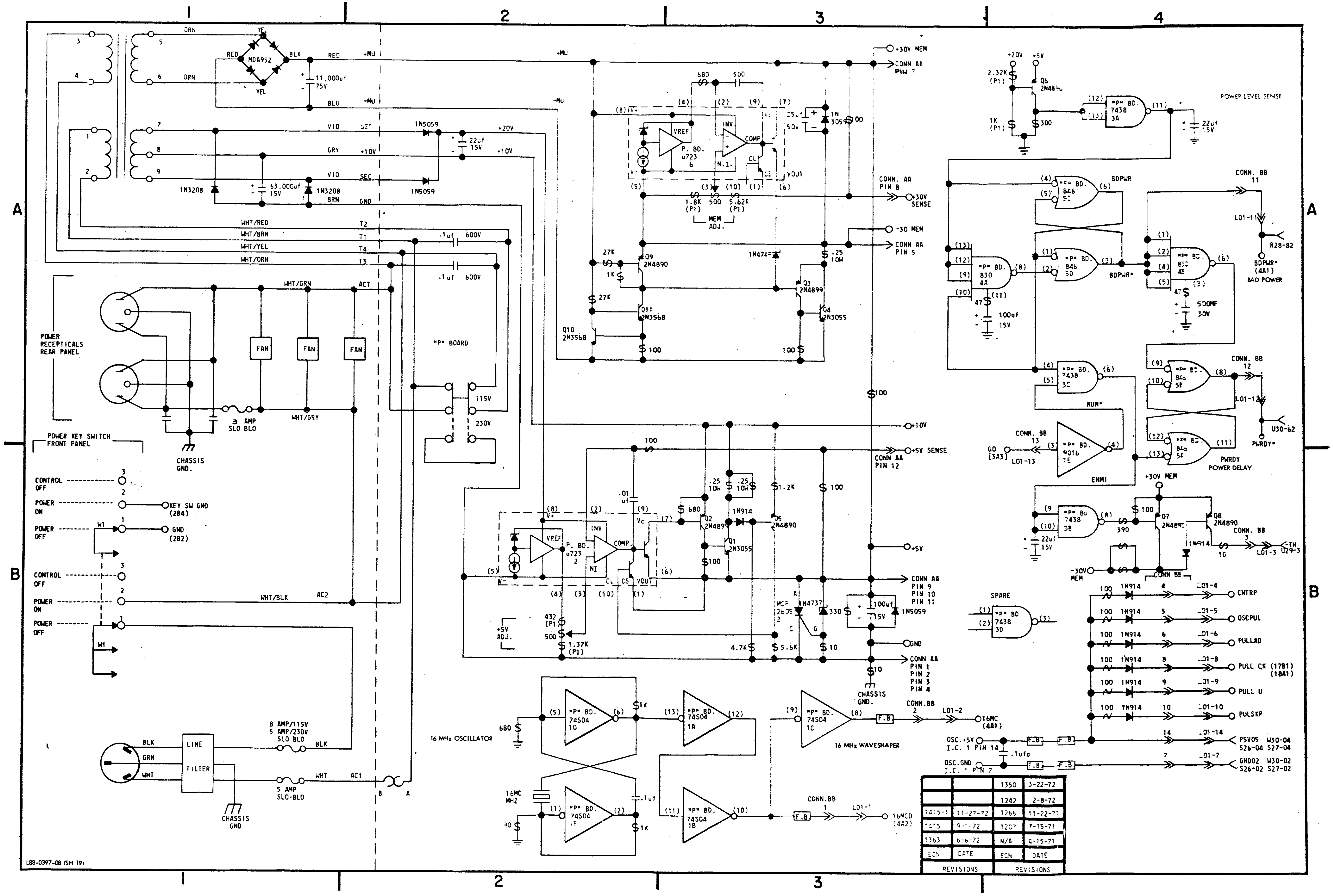


Figure 7-2. ND812, Power Supply and 16MHz Oscillator, Logic Diagram (Sheet 19 of 28)

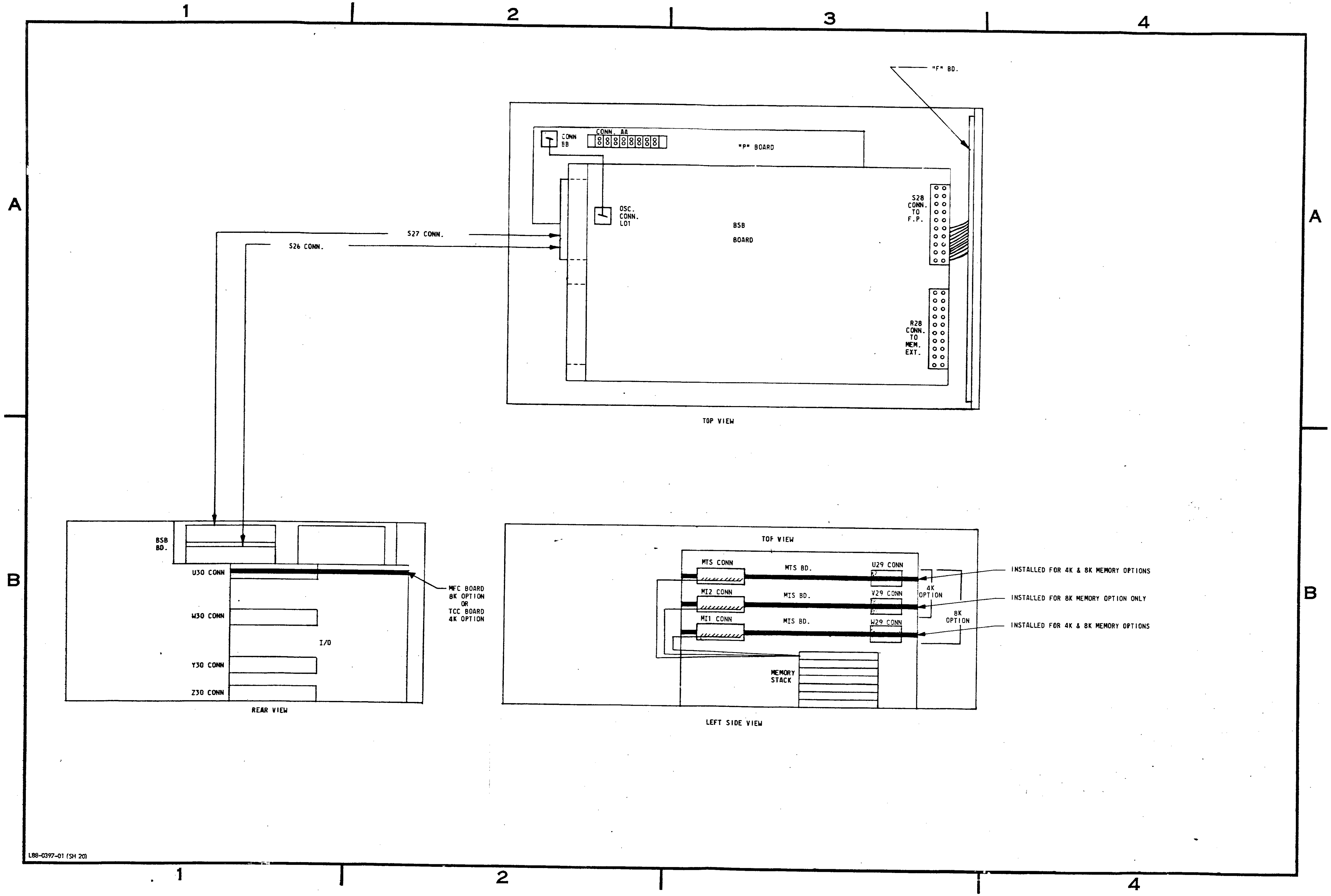


Figure 7-2. ND812, Connector Location Diagram (Sheet 20 of 28)

MEMORY EXTENSION
R28

PIN	SIGNAL	PIN	SIGNAL
1	XGND1 TO M22-Ø8	2	XGND1 TO M22-Ø8
3		4	
5	MSØØ* [18A1]	6	AØØ* [18A4]
7	MS12* [18A2]	8	AØ1* [18A4]
9	MSØ1* [18A1]	10	AØ2* [18A4]
11	MS13* [18A2]	12	AØ3* [18B4]
13	MSØ2* [18A1]	14	AØ4* [18B4]
15	MS14* [18A2]	16	AØ5* [18B4]
17	MSØ3* [18A1]	18	AØ6* [18A4]
19	MS15* [18A2]	20	AØ7* [18A4]
21	MSØ4* [18A1]	22	AØ8* [18A4]
23	MS16* [18A2]	24	AØ9* [18B4]
25	MSØ5* [18A1]	26	A1Ø* [18B4]
27	MS17* [18A2]	28	A11* [18B4]
29	MSØ6* [18B1]	30	MØØB [17B2]
31	MS18* [18B2]	32	MØ1B [17A2]
33	MSØ7* [18B1]	34	MØ2B [17A2]
35	MS19* [18B2]	36	MØ3B [17A2]
37	MSØ8* [18B1]	38	MØ4B [17B3]
39	MS20* [18B2]	40	MØ5B [17A3]
41	MSØ9* [18B1]	42	MØ6B [17A3]
43	MS21* [18B2]	44	MØ7B [17A3]
45	MS1Ø* [18B1]	46	MØ8B [17B4]
47	MS22* [18B2]	48	MØ9B [17A4]
49	MS11* [18B1]	50	M10B [17A4]
51	MS23* [18B2]	52	M11B [17A4]
53	ADDF1* (17A1)	54	ACIWX* [8B4]
55	ADDF1	56	MCLRX* [8B4]
57	MBØØ [18A2]	58	MB12 [18A3]
59	MBØ1 [18A2]	60	MB13 [18A3]
61	MBØ2 [18A2]	62	MB14 [18A3]
63	MBØ3 [18A2]	64	MB15 [18A3]
65	MBØ4 [18A2]	66	MB16 [18A3]
67	MBØ5 [18A2]	68	MB17 [18A3]
69	MBØ6 [18B2]	70	MB18 [18A3]
71	MBØ7 [18B2]	72	MB19 [18B3]
73	MBØ8 [18B2]	74	MB2Ø [18B3]
75	MBØ9 [18B2]	76	MB21 [18B3]
77	MB1Ø [18B2]	78	MB22 [18B3]
79	MB11 [18B2]	80	MB23 [18B3]
81	GO [3A3]	82	BD PWR* [19A4]
83		84	EPRDY* (3B1)
85	XGND2 TO M25-8	86	XGND2 TO M25-8

MULTI FIELD CONTROL
U30

PIN	SIGNAL	PIN	SIGNAL
1	GNDØ1 TO S26-Ø1	2	GNDØ2 TO S26-Ø2
3	P5VØ3 TO S26-Ø3	4	P5VØ4 TO S26-Ø4
5	EXTK* (9A2)	6	INDFF (,A3)
7	EXTØØ* (14B1)	8	1Ø6B [17B2]
9	EXTØ1* (14B1)	10	1[illegible]B [17B3]
11	EXTØ2* (14A1)	12	1Ø7B [17B4]
13	EXTØ3* (14A1)	14	JMP* [7B2]
15	EXTØ4* (15B1)	16	JPS* [7B2]
17	EXTØ5* (15B1)	18	PUØB [5A2]
19	EXTØ6* (15A1)	20	PU1B [5A2]
21	EXTØ7* (15A1)	22	EP1 [5A4]
23	EXTØ8* (16B1)	24	BP7B [5B4]
25	EXTØ9* (16B1)	26	EPØ* [5A4]
27	EXT1Ø* (16A1)	28	TWFF* [3B4]
29	EXT11* (16A1)	30	OUTØØ* [14B4]
31	AØØ* [18A4]	32	OUTØ1* [14B4]
33	AØ1* [18A4]	34	OUTØ2* [14B4]
35	AØ2* [18A4]	36	OUTØ3* [14B4]
37	AØ3* [18B4]	38	OUTØ4* [15B4]
39	AØ4* [18B4]	40	OUTØ5* [15B4]
41	AØ5* [18B4]	42	OUTØ6* [15B4]
43	AØ6* [18A4]	44	OUTØ7* [15B4]
45	AØ7* [18A4]	46	OUTØ8* [16B4]
47	AØ8* [18A4]	48	OUTØ9* [16B4]
49	AØ9* [18B4]	50	OUT1Ø* [16B4]
51	A1Ø* [18B4]	52	OUT11* [16B4]
53	A11* [18B4]	54	LDPR [6A2]
55	JPSMFØ (14A2)	56	CSFF* [4B2]
57	JPSMF1 (14A2)	58	INTFF* [4B4]
59	INTMFØ (15B2)	60	EPØ [5A4]
61	INTMF1 (15B2)	62	PWRDY* [19A4]
63	FLMEM* [3B4]	64	CPPU* [4A3]
65	BP1 [5A3]	66	ADDF1
67	MFØ (16A2)	68	ADDFØ (8B3)
69	MF1 (16A2)	70	ADDF1* (17A1)
71	DIN* (6B3)	72	ADDFØ* (8B3)
73	I1Ø* [17B2]	74	SMF1* (2A4)
75	HLT [3B2]	76	SMFØ* (2A4)
77	I11B* [17B3]	78	BP5* [5A3]
79	IØOSC [7B1]	80	BP6* [5B3]
81		82	TWJJS* (7B3)
83	P5V83 TO S26-83	84	P5V84 TO S26-84
85	GND85 TO S26-85	86	GND86 TO S26-86

MEMORY MTS
U29

PIN	SIGNAL	PIN	SIGNAL
1	TH+	2	AØØ* [18A4]
3	TH- [19B4]	4	AØ1* [18A4]
5	MCIW* [8B4]	6	AØ2* [18A4]
7	WGATE2	8	AØ3* [18B4]
9	INHIB	10	AØ4* [18B4]
11	INHIB*	12	AØ5* [18B4]
13	WID	14	AØ9* [18B4]
15	MCIR* [8B4]	16	A10* [18B4]
17	STROBE	18	A11* [18B4]
19	R/R	20	AØ6* [18A4]
21	RGATE1	22	AØ7* [18A4]
23	WRITE	24	AØ8* [18A4]
25	READ*	26	
27		28	
29		30	
31		32	
33		34	
35		36	
37		38	
39		40	
41		42	
43		44	
45		46	
47		48	
49		50	
51		52	
53		54	
55	GND TO M24-Ø8	56	GND
57	+5V	58	+5V
59	-V	60	-V

MEMORY MIS (2ND 4K)
V29

PIN	SIGNAL	PIN	SIGNAL
1	MS12* [18A2]	2	MB12 [18A3]
3		4	MØØB [17B2]
5	MS13* [18A2]	6	MB13 [18A3]
7		8	MØ1B [17A2]
9	MS14* [18A2]	10	MB14 [18A3]
11		12	MØ2B [17A2]
13	MS15* [18A2]	14	MB15 [18A3]
15		16	MØ3B [17A2]
17	MS16* [18A2]	18	MB16 [18A3]
19		20	MØ4B [17B3]
21	MS17* [18A2]	22	MB17 [18A3]
23		24	MØ5B [17A3]
25	MS18* [18B2]	26	MØ6B [17A3]
27		28	MB18 [18A3]
29	MS19* [18B2]	30	MB19 [18B3]
31		32	MØ7B [17A3]
33	MS20* [18B2]	34	MØ8B [17B4]
35		36	MB20 [18B3]
37	MS21* [18B2]	38	MB21 [18B3]
39		40	MØ9B [17A4]
41	MS22* [18B2]	42	M1ØB [17A4]
43		44	MB22 [18B3]
45	MS23* [18B2]	46	MB23 [18B3]
47		48	M11B [17A4]
49	ADDF1* (17A1)	50	
51	ADDF1	52	
53	INHIB	54	STROBE
55	GND TO M21-Ø8	56	GND TO M25-Ø8
57	+5V	58	+5V
59	-V	60	-V

MEMORY MIS (1ST 4K)
W29

PIN	SIGNAL	PIN	SIGNAL
1	MSØØ* [18A1]	2	MØØB [illegible]
3		4	MBØØ [illegible]
5	MSØ1* [18A1]	6	MØ1B [illegible]
7		8	MBØ1 [illegible]
9	MSØ2* [18A1]	10	MØ2B [illegible]
11		12	MBØ2 [illegible]
13	MSØ3* [18A1]	14	MØ3B [illegible]
15		16	MBØ3 [illegible]
17	MSØ4* [18A1]	18	MØ4B [illegible]
19		20	MBØ4 [illegible]
21	MSØ5* [18A1]	22	MØ5B [illegible]
23		24	MBØ5 [illegible]
25	MSØ6* [18B1]	26	MBØ6 [illegible]
27		28	MØ6B [illegible]
29	MSØ7* [18B1]	30	MØ7B [illegible]
31		32	MBØ7 [illegible]
33	MSØ8* [18B1]	34	MBØ8 [illegible]
35		36	MØ8B [illegible]
37	MSØ9* [18B1]	38	MØ9B [illegible]
39		40	MBØ9 [illegible]
41	MS1Ø* [18B1]	42	MB1Ø [illegible]
43		44	M1ØB [illegible]
45	MS11* [18B1]	46	M11B [illegible]
47		48	MB11 [illegible]
49	ADDF1* (17A1)	50	
51	ADDF1	52	
53	INHIB	54	STROBE
55	GND TO M[illegible]-Ø8	56	GND TO M[illegible]-Ø8
57	+5V	58	+5V
59	-V	60	-V

L88-0397-02 (SH 21)

ECN	DATE
1329	3-22-72
1266	11-29-71

REVISIONS

Figure 7-2. ND812, Memory Connector Signal Table (Sheet 21 of 28)

I/O CABLE CONNECTORS S26 & S27				I/O P.C. BD. CONNECTORS W30, Y30 AND Z30				FRONT PANEL S28			
PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	GNDØ1 TO J01-7	2	GNDØ2 [19B4]	1	GNDØ1 TO J01-7	2	GNDØ2 [19B4]	1		2	
3	P5VØ3 TO J01-14	4	P5VØ4 [19B4]	3	P5VØ3 TO J01-14	4	P5VØ4 [19B4]	3		4	
5	DIN* (6B3)	6	EXGO* [3A4]	5	DIN* (6B3)	6	EXGO* [3A4]	5		6	
7	EXTØØ* (14B1)	8	OUTØØ* [14B4]	7	EXTØØ* (14B1)	8	OUTØØ* [14B4]	7		8	
9	EXTØ1* (14B1)	10	OUTØ1* [14B4]	9	EXTØ1* (14B1)	10	OUTØ1* [14B4]	9		10	
11	EXTØ2* (14A1)	12	OUTØ2* [14B4]	11	EXTØ2* (14A1)	12	OUTØ2* [14B4]	11		12	
13	EXTØ3* (14A1)	14	OUTØ3* [14B4]	13	EXTØ3* (14A1)	14	OUTØ3* [14B4]	13		14	
15	EXTØ4* (15B1)	16	OUTØ4* [15B4]	15	EXTØ4* (15B1)	16	OUTØ4* [15B4]	15		16	
17	EXTØ5* (15B1)	18	OUTØ5* [15B4]	17	EXTØ5* (15B1)	18	OUTØ5* [15B4]	17		18	
19	EXTØ6* (15A1)	20	OUTØ6* [15B4]	19	EXTØ6* (15A1)	20	OUTØ6* [15B4]	19		20	
21	EXTØ7* (15A1)	22	OUTØ7* [15B4]	21	EXTØ7* (15A1)	22	OUTØ7* [15B4]	21		22	
23	EXTØ8* (16B1)	24	OUTØ8* [16B4]	23	EXTØ8* (16B1)	24	OUTØ8* [16B4]	23		24	
25	EXTØ9* (16B1)	26	OUTØ9* [16B4]	25	EXTØ9* (16B1)	26	OUTØ9* [16B4]	25	MØ1B (1A2)	26	EMXS0* [2B3]
27	EXT1Ø* (16A1)	28	OUT1Ø* [16B4]	27	EXT1Ø* (16A1)	28	OUT1Ø* [16B4]	27	+5L (2B4)	28	+5V (2B4)
29	EXT11* (16A1)	30	OUT11* [16B4]	29	EXT11* (16A1)	30	OUT11* [16B4]	29	CY (1B1)	30	EMXS1* [2B3]
31	EXTK* (9A2)	32	IOMØØ* [17A4]	31	EXTK* (9A2)	32	IOMØØ* [17A4]	31	MF1 (1B2)	32	ALODT* (2B3)
33	EINTR* (4B3)	34	IOMØ1* [17A4]	33	EINTR* (4B3)	34	IOMØ1* [17A4]	33	MFØ (1B2)	34	XLDMR* [2A4]
35	XINTP* [4B4]	36	IOMØ2* [17A4]	35	XINTP* [4B4]	36	IOMØ2* [17A4]	35	MØ2B (1A2)	36	EMXS2* [2B3]
37	CSR* (4B1)	38	IOMØ3* [17A4]	37	CSR* (4B1)	38	IOMØ3* [17A4]	37	MØ3B (1A2)	38	XLDPR* [2A3]
39	EXCSP* [4B2]	40	IOMØ4* [17A4]	39	EXCSP* [4B2]	40	IOMØ4* [17A4]	39	MØØB (1A2)	40	BLST1* [2B4]
41	ALTER* (6B3)	42	IOMØ5* [17A4]	41	ALTER* (6B3)	42	IOMØ5* [17A4]	41	OUTØ1* (1A1)	42	SMFØØ* [2A4]
43	SDCS* (4B2)	44	IOMØ6* [17A4]	43	SDCS* (4B2)	44	IOMØ6* [17A4]	43	OUTØ2* (1A1)	44	SMFØ1* [2A4]
45	CSMRZ* [7A4]	46	IOMØ7* [17A4]	45	CSMRZ* [7A4]	46	IOMØ7* [17A4]	45	OUTØ3* (1A1)	46	EXGO* (1B1)
47	MRDY* [7B4]	48	IOMØ8* [17A4]	47	MRDY* [7B4]	48	IOMØ8* [17A4]	47	OUTØ4* (1A1)	48	SWØØ* [2A1]
49	PCPØ* [5B4]	50	IOMØ9* [17A4]	49	PCPØ* [5B4]	50	IOMØ9* [17A4]	49	OUTØØ* (1A1)	50	SWØ1* [2A1]
51	PCP1* [5B4]	52	IOM1Ø* [17B4]	51	PCP1* [5B4]	52	IOM1Ø* [17B4]	51	OUTØ5* (1A1)	52	SWØ2* [2A1]
53	PCP2* [5B4]	54	IOM11* [17B4]	53	PCP2* [5B4]	54	IOM11* [17B4]	53	SI* [2A3]	54	SWØ3* [2A1]
55	PCP3* [5B4]	56	IOCOM* [7B2]	55	PCP3* [5B4]	56	IOCOM* [7B2]	55	SS* [2A4]	56	SWØ4* [2A1]
57	SMFØØ* [2A4]	58	XSTCL* [3B3]	57	SMFØØ* [2A4]	58	XSTCL* [3B3]	57	SIG GND (2A4)	58	GND (2B4)
59	SMFØ1* [2A4]	60	EXGO [3A4]	59	SMFØ1* [2A4]	60	M25V	59	MØ5B (1A2)	60	SWØ5* [2A1]
61	EXSKP* (12A1)	62	PCPST* [4A4]	61	EXSKP* (12A1)	62	PCPST* [4A4]	61	MØ6B (1A2)	62	SWØ6* [2A1]
63	START* [2A2]	64	ALODT* (2B3)	63	START* [2A2]	64	ALODT* (2B3)	63	MØ7B (1B2)	64	SWGND (2B1)
65	STOP* [2A3]	66	EXCLJ* (9B1)	65	STOP* [2A3]	66	EXCLJ* (9B1)	65	MØ4B (1A2)	66	SWØ7* [2B1]
67	CONT* (2A2)	68	TSTD* (12B2)	67	CONT* (2A2)	68	TSTD* (12B2)	67	OUTØ6* (1A1)	68	SWØ8* [2B1]
69	EEXAM* (2A3)	70	EMXSØ* [2B3]	69	EEXAM* (2A3)	70	EMXSØ* [2B3]	69	OUT1Ø* (1B1)	70	SWØ9* [2B1]
71	XLDPR* [2A3]	72	EMXS1* [2B3]	71	XLDPR* [2A3]	72	EMXS1* [2B3]	71	OUTØ7* (1A1)	72	SW1Ø* [2B1]
73	XLDMR* [2A4]	74	EMXS2* [2B3]	73	XLDMR* [2A4]	74	EMXS2* [2B3]	73	OUTØ8* (1B1)	74	SW11* [2B1]
75	SI* [2A3]	76	ARMX* (8A1)	75	SI* [2A3]	76	ARMX* (8A1)	75	OUT11* (1B1)	76	ENINT (1B1)
77	CSPO*	78	IONA* [11B4]	77	CSPO*	78	IONA* [11B4]	77	OUTØ9* (1B1)	78	EEXAM* [2A3]
79	ECHO* (6B3)	80	IONB* [11B4]	79	ECHO* (6B3)	80	IONB* [11B4]	79	MØ9B (1B2)	80	START* [2A2]
81	INTPO*	82	IONL* [11B4]	81	INTPO*	82	IONL* [11B4]	81	M1ØB (1B2)	82	CONT* [2A2]
83	P5V83 TO P01-16	84	P5V84 TO M01-16	83	P5V83 TO P01-16	84	P5V84 TO M01-16	83	M11B (1B2)	84	
85	GND85 TO P01-8	86	GND86 TO M01-8	85	GND85 TO P01-8	86	GND86 TO M01-8	85	MØ8B (1B2)	86	STOP* [2A3]

P BOARD

POWER SUPPLY

AA CONNECTOR	
PIN	SIGNAL
1	GND [19B3]
2	GND [19B3]
3	GND [19B3]
4	GND [19B3]
5	-30 MEM [19A3]
6	
7	+30V MEM [19A3]
8	+30V SENSE [19A3]
9	+5V [19B3]
10	+5V [19B3]
11	+5V [19B3]
12	+5V SENSE [19B3]

OSCILLATION CONNECTOR

BB CONNECTOR	
PIN	SIGNAL
1	16MCD [19B3]
2	16MC [19B3]
3	-TH [19B4]
4	CNTRP [19B4]
5	OSCPUL [19B4]
6	PULLAD [19B4]
7	GNDØ2 [19B4]
8	PULLCK [19B4]
9	PULLU [19B4]
10	PULSKP [19B4]
11	BDPWR* [19A4]
12	PWRDY* [19A4]
13	GO [19A4]
14	P5VØ5 [19B4]

BSB

OSCILLATOR

LO1 CONNECTOR	
PIN	SIGNAL
1	16MCD [19B3]
2	16MC [19B3]
3	-TH [19B4]
4	CNTRP [19B4]
5	OSCPUL [19B4]
6	PULLAD [19B4]
7	GNDØ2 [19B4]
8	PULLCK [19B4]
9	PULLU [19B4]
10	PULSKP [19B4]
11	BDPWR* [19A4]
12	PWRDY* [19A4]
13	GO [19A4]
14	P5VØ5 [19B4]

1355	6-16-72
ECN	DATE
REVISIONS	

L88-0397-01 (SH 22)

Figure 7-2. ND812, Power Supply and Oscillator Signal Table (Sheet 22 of 28)

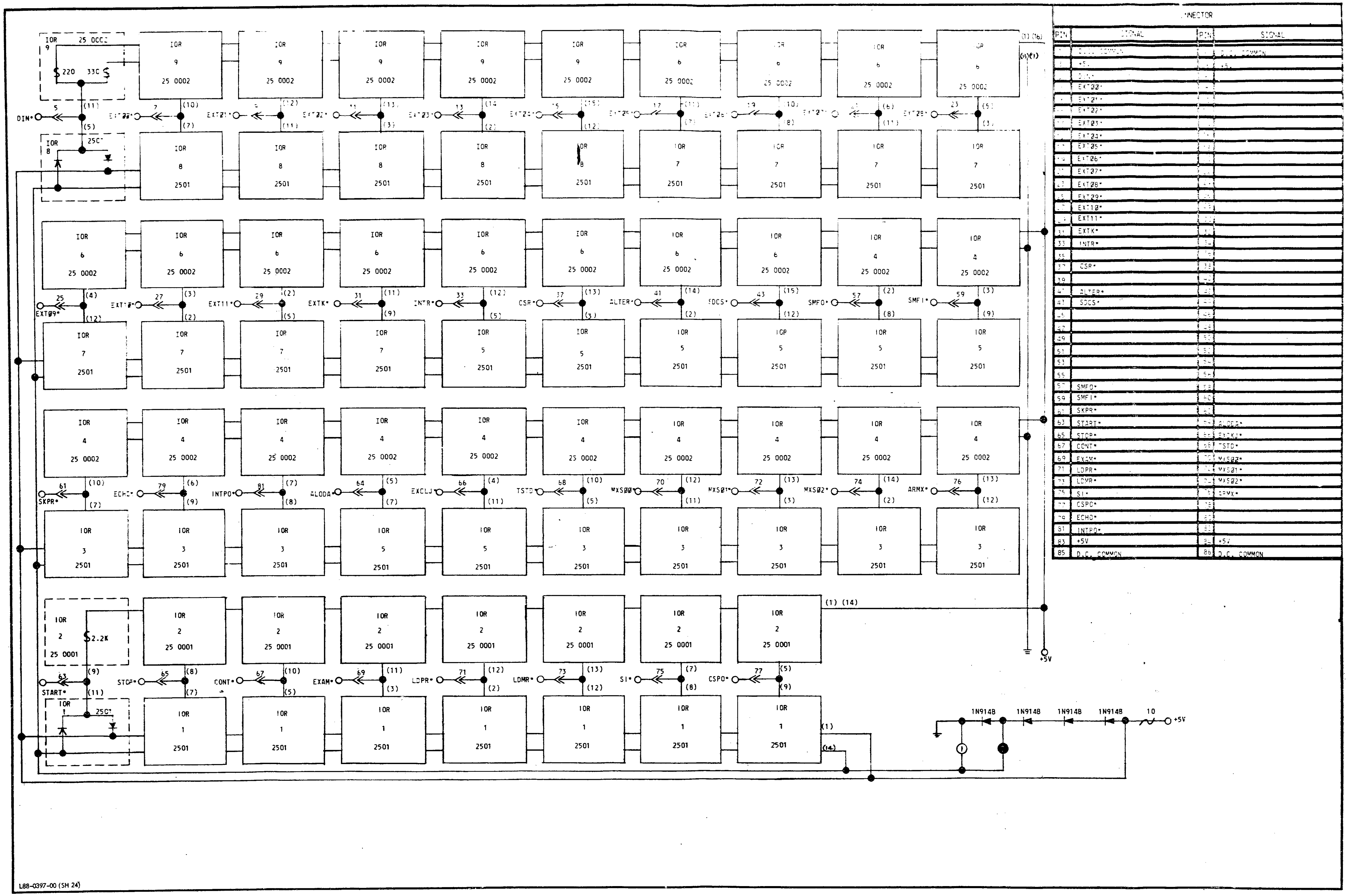
ADJSBJ	[7B2] (12A3) (12A4) (12B4)
ADLSBL	[6A1] (7B3) (11B3) (12A3)
ALOOT	[8A1] (8A2) (8A3) (12B2)
ALOOT*	[2B3] (3A3) (6B2) (6B3) (8A1)
ALTER	[6B3] (8A4) (12B1) (12B2) (13B1)
ANDJK	[8A3] (14B1) (15B1) (16B1)
ANL*	[6A1] (7B3) (14B1) (15B1)
BPØ	[5A3] (8A1) (8A3) (9B3) (12B1) (13B1) (13B3) (13B4)
BP4	[5A3] (4A3) (6A1) (8B3) (9B3) (11A2) (12A1) (12A1) (12A2) (13B1)
BP4MD	[5A3] (8A1) (9A3) (9A4) (9B3) (11B2) (11B3) (12B3) (12B4) (12B4)
BP5	[5A3] (3B2) (3B3) (12A1) (12A3) (12A4) (12A4) (11B1)
BP5*	[5A3] (5A3) (8B1) (8B2) (11A2) (3B4)
BP56	[11A2] (6B1) (6B3) (8A3) (8A3) (9A3) (9A3) (11A2) (12A3) (12A4)
BPØB	[5B3] (8B4) (9B2) (12B3) (13B3) (13B2)
BP5MD	[5B3] (8A1) (9A2) (9A3) (9A4) (9B2) (12B1) (12B1) (12B4) (8B1)
BP6	[5B3] (6A2) (7B4) (8B1) (9B3) (9A3) (10B2) (11B3) (11B3) (12A3) (12A4)
BP67	[5B3] (6B2) (12A1) (13B2) (13B3) (12B4)
BP6B	[5B3] (3A3) (8A1) (8A3) (8A4) (9B2) (12A4) (12B3) (13B2) (13B3) (13B4)
BP6MD	[5B3] (5B1) (8A1) (8B1) (9A4) (9A4) (9A4) (11B2) (11B3) (12A2) (12B1) (12B1)
BP7	[5B3] (8B3) (9A3) (9B3) (9B1) (11B2) (11B3) (12A1) (12B3) (12B4)
CKFL*	[10A2] (10B2) (10B2) (10A3)
CNTRP	[19B4] (3A3) (4B2) (4B4) (11A3)
COUT	[14B3] (11B2) (11B2) (12B4) (12B4) (12B4)
CPA*	[9B2] (14B3) (15B3) (16B3)
CPJ*	[9A4] (14A3) (15A3) (16A3)
CPK*	[9A4] (14A3) (15A3) (16A3)
CPP*	[9B2] (14B3) (15B3) (16B3)
CPPU	[4A4] (3A2) (4A3) (6B4) (11A3) (11B1)
CPR*	[9B2] (14A4) (15A4) (16A4)
CPS*	[9B2] (14A4) (15A4) (16A4)
CSFF	[4B2] (8A1) (8A4) (12B1) (12B1) (12A1) (12B2) (13B1)
CSFF*	[4B2] (4B3) (4B4) (7A2)
CSP*	[4B2] (8A1) (8A2) (12A1)
DIN	[6B3] (8A3) (9A2) (9A2) (12B1) (12B1)
DIN*	[6B3] (8A4) (12A3) (12A3)
DONE	[7B4] (3A2) (3B3) (4B2) (6B4)
EPØ	[5A4] (8A1) (13B1) (13B3) (13B4)
EP1	[5A4] (8B4) (9B3) (12A1)
EP3	[5A4] (8B3) (12A3) (12A4) (12B2)
EP4	[5A4] (8A4) (12A1) (12B2)
EP45	[5A4] (12A3) (12A4) (13B3)
EP6	[5A4] (8A1) (8B4) (9A3) (9B3) (9B3) (12A1) (12A3) (12A4) (12B4) (13B1) (13B3)
EXPMR*	[6B2] (3A3) (3A3) (3B2) (6B3) (9A1)
FFLOC	[3B4] (8A4) (8A4) (12B1)
FFLOC*	[3B4] (7B3) (8A1) (8A1) (8B3) (13B1) (13B1) (13B3)
GO	[3A3] (3A2) (3A4) (4B1) (5B2) (19B4)
GO*	[3A3] (3B2) (4A1) (5B1) (6B1) (6B3) (9A1) (9B2) (12B2)
HWD	[11B1] (8A1) (8B2) (11B3) (12A1) (12A2) (12B1) (12B4)
HWDRL	[11B2] (9A1) (9B2) (11B3) (12B1)
HWDRL*	[11B2] (9B2) (12A4) (13B1)
HWD $\overline{RL}$	[8A1] (5B1) (11B2) (12A3)
HWM	[11B2] (8B1) (94A) (9B1) (9B2) (11B3) (12A1) (12A2) (12A3)
HWMRL	[11A2] (8A1) (9A2) (9A3) (9B2) (12B3) (12B4)
IØ4*	[17B2] (3B1) (6A1) (8A4) (9A1) (9B3) (10A2)
IØ4B	[17B3] (3B1) (3B1) (6A1) (6A1) (9A2) (8B1) (9B2) (9A4) (9B3) (12A2)
I4I5	[9A1] (4A1) (8A3) (11A1) (13B1) (13B2) (13B2)
I4SKP	[17B3] (3A1) (7A2) (9B2) (9B3) (10A2) (10A4) (10B2)
IØ5*	[17B4] (3B1) (6A1) (9B3) (10A2)
IØ5B	[17B4] (3B1) (6A1) (8A1) (9A2) (9A2) (9A3) (9A4) (9B3)
IØ5B*	[17B4] (9A1) (13B1) (13B1)
I5BSKP	[17B4] (3A1) (7A2) (9B2) (10A2) (10A4) (10B2)
I5X	[17B4] (8A4) (8A4) (12B2) (13B2) (13B3) (13B3)
IØ6*	[17B1] (3B1) (3B1) (3B2) (10A2)
IØ6B	[17B2] (10A2) (11B3) (10A4) (10B4) (11A1) (11B3)
IØ7*	[17B2] (3A1) (3B2) (13B2)
IØ7B	[17B3] (10B3) (11A1) (11B3) (13B3)
IØ8*	[17B4] (3A1) (3B2) (11A2)
IØ8B	[17B4] (3A4) (9B2) (11A1)
IØ9*	[17B1] (3A1) (3B2) (8A1) (8A2) (10A2) (10A2) (10B4) (11A2) (11A4)
IØ9B	[17B2] (8B1) (10A3) (10B2) (10B4) (11B1) (11B1) (11B3) (12A2) (12B3)
I1Ø*	[17B2] (3A1) (3B2) (8A2) (8B1) (9A1) (10A4) (11A2) (11A4) (12B3)
I1ØB	[17B3] (8A1) (8B1) (8B1) (10A2) (10B2) (11B3) (11B1)
I11*	[17B4] (3A1) (3B2) (6B2) (10A4) (11A2) (12B3)
I11B	[17B4] (9B3) (10A2) (10A3) (10B4) (12B3)
I11B*	[17B4] (8A2) (12B3) (13B2)
I11N	[17B4] (6B2) (9A1) (9A3) (11B1) (11B3) (13B2)
IND*	[6A2] (5B1) (6A2) (6A3) (6A3) (6A4) (7A2) (7B3) (8A1) (8B2) (12B1)
INTFF	[4B4] (4B4) (8A1) (8B3) (9B2) (12B1)
INTFF*	[4B4] (8A2) (12A1) (12B1)
I/Ø	[7B1] (8A3) (8B3) (9A2) (9A2) (9B3) (12A1) (12A3) (12A3)
I/Ø*	[7B1] (7B3) (9B3)
I/ØOSC	[7B1] (4A3) (4A4) (5B4) (6B3) (6B4)
ISZDSZ	[7A2] (12A1) (12B2) (13B1)
JØØ	[14A4] (9B3) (9B3) (10A1) (13A1) (14B2)
JØØB	[10A2] (9B3) (10B2) (10B3) (14B1)
JØ1	[14A4] (10A1) (11B2) (13A1) (14B1) (14B2)
JØ2	[14A4] (10A1) (13A2) (14A1) (14A2)
JØ3	[14A4] (10A1) (13A2) (14A2) (14A1)
JØ4	[15A4] (10A1) (13A3) (14A3) (15B2)
JØ5	[15A4] (10A1) (15B1) (15B2) (13A3)
JØ6	[15A4] (10A1) (13A1) (15A1) (15A2)
JØ7	[15A4] (10A1) (13A1) (15A1) (15A2)
JØ8	[16A4] (10B1) (13A2) (15A3) (16B2)
JØ9	[16A4] (10B1) (13A2) ([illegible]B1) (16B2)
J1Ø	[16A4] (10B1) (13A3) (16A1) (16A2)
J11	[16A4] (10B1) (13A3) (16A1) (16A2)
JCOM	[3A4] (9A1) (12A2) (12A3) (12A3)
JMPDLS	[7B4] (13B1) (13B2) (13B4)
JPS*	[7B2] (7A2) (9B2) (12A1) (12B1)
KØØ	[14A4] (10A1) (9B3) (9B3) (13A1) (12B4)
KØØB	[10A2] (8B1) (9A3) (10B2) (12B3) (12B3) (14B1)
KØØM	[8B1] (9B3) (12B3) (12B4)
KØ1	[14A4] (10A1) (13A1) (14B1) (14B2)
KØ2	[14A4] (10A1) (13A2) (14A1) (14A2)
KØ3	[14A4] (10A1) (14A1) (14A2) (13A2)
KØ4	[15A4] (10A1) (13A3) (14A3) (15B1) (15B2)
KØ5	[15A4] (10B1) (13A3) (15B1) (15B2)
KØ6	[15A4] (10B1) (13A1) (15A1) (15A2)
KØ7	[15A4] (10B1) (13A1) (15A1) (15A2)
KØ8	[16A4] (10B1) (13A2) (15A3) (16B1) (16B2)
KØ9	[16A4] (10B1) (13A2) (16B1) (16B2)
K1Ø	[16A4] (10B1) (13A3) (16A1) (16A2)
K11	[16A4] (10B1) (13A3) (16A1) (16A2)
LDEXT	[8A3] (14B1) (15B1) (16B1)
LDFPSW	[6B3] (8A3) (14B1) (15B1) (16B1)
LDPR	[6A2] (8A2) (8A3) (12B3) (2A3)
LIT	[7A1] (6A1) (14B1) (15B1)
LITB	[7A1] (8A4) (9A1) (9A1) (15A1) (16B1)
MØØB	[17B2] (1A2) (14A1) (17A4)
MØ1B	[17A2] (1A2) (14B1) (17A4)
MØ2B	[17A1] (1A2) (14A1) (17A4)
MØ3B	[17A2] (1A2) (14A1) (17A4)
MØ4B	[17B3] (1A2) (15B1) (17A4)
MØ5B	[17A3] (1A2) (15B1) (17A4)
MØ6B	[17A3] (1A2) (15A1) (17A4)
MØ7B	[17A3] (1B2) (15A1) (17A4)
MØ8B	[17B4] (1B2) (5B4) (16B1) (17A4)
MØ9B	[17A4] (1B2) (5B4) (16B1) (17A4)
M1ØB	[17A4] (1B1) (5B4) (16A1) (17B4)
M11B	[17A4] (1B2) (5B4) (16A1) (17B4)
MPASS	[13B1] (8A1) (8A1) (13B1) (13B2) (13B3)
MRJ*	[9B2] (14A3) (15A3) (16A3)
MRK*	[9B2] (14A3) (15A3) (16A3)
MXEN*	[9A2] (14B2) (15B2) (16B2)
MXSØ	[9B2] (14B2) (15B2) (16B2)
MXS1	[9B2] (14B2) (15B2) (16B2)
MXS2	[9B2] (14B2) (15B2) (16B2)
ONCE*	[8B3] (9B2) (12A1) (13B2)
OP1	[11A2] (8A1) (8B1) (12A2) (13B2)
OP2	(7A3) (3A1) (8B4) (9B2) (9B2) (10B3) (11B3) (12A1) (12B3)
OP2CL	[9B2] (9B2) (10B2) (11B3)
OP1ØØ	[11A1] (8A2) (8B1) (8B1) (9A2) (9A2) (9A3) (9B3) (11B1) (12A2)
OP1Ø23	[11A1] (9A2) (9A2) (11A2) (12B2)
OP167	[11A1] (9B3) (9B3) (9B3)
OSCPUL	[19B4] (4A1) (4A3) (4A4) (5A1) (5B2) (5B2)
OUTST	[6B3] (14B4) (15B4) (16B4)
OV	[11B4] (1B1) (9A1) (9A2) (9B2) (12A2) (12B1) (14B2)
OV*	[11B4] (5B1) (8B1) (9B2) (10A2)
PEA*	[9B1] (14B3) (15B3) (16B3)
PEJ*	[9A3] (14A3) (15A3) (16A3)
PEK*	[9A3] (14A3) (15A3) (16A3)
PEP*	[9B3] (9B1) (14B3) (15B3) (16B3)
PER*	[9A4] (9B1) (14A4) (15A4) (16A4)
PES*	[9A4] (9A1) (14A4) (15A4) (16A4)
PS12	[11A4] (9A2) (9B3) (12A2) (12B3) (12B4)
PS12*	[11A4] (5B1) (9A3) (9B2) (12A3) (12B3) (12B3)
PUØ	[5A2] (5A3) (5A4) (5B2)
PUØ*	[5A2] (3A4) (7A4) (8B2)
PUØB	[5A2] (5B1) (6B4) (8A1) (8A3) (8A4) (9B1) (12A1)
PU1B	[5A2] (9B2) (12B1) (18B1)
PU3B	[5A2] (8A3) (8A4) (8B3) (12B2)
PU4	[5A2] (5A3) (5A4) (5B4)
PU4B	[5A2] (8A4) (12B1) (12B2)
PU5	[5B2] (5A3) (5A4) (5B4)
PU5B	[5B2] (7A3) (8B3) (9A2) (12A2) (12B3)
PU6	[5B2] (4A1) (4A3) (5A4) (5B3) (5B4)
PU6B	[5B2] (3A2) (3A3) (8A1) (8A4) (9B2) (12B1)
PU7	[5B2] (5B1) (5B3) (5B4)
PU7B	[5B2] (3A4) (3B3) (4A3) (4B2) (6B4)
PULLAD	[19B4] (14B3) (15B3) (16B3)
PULSKP	[19B4] (3A1) (3A1) (10B2) (10B3)
PULLU	[19B4] (8B1) (14A4) (14B1) (15A4) (15B1) (16A4) (16B1) (8B1)
PWRDY*	[19A4] (3A4) (3B1)
REGCLK	[4A4] (3B3) (6A1) (12B3) (7A3) (9A4) (9B1) (11B3)
RLOOP	[8B2] (9B3) (11A2) (11B1) (12B1) (12B4)
RLOOP*	[8B2] (8A1) (8B1) (8B2) (9A4) (9B2)
SIC	[7A1] (12A1) (12A3) (12A4) (13B3)
STCLR	[3B3] (3A1) (4A3) (6B4) (10B2)
STCLR*	[3B3] (5A1) (5B2) (6A2) (11B3) (11B4) (11B2)
STDN7*	[3B3] (3B3) (6A2) (8B1)
TW	[7A1] (3A4) (3B2) (3B2)
TWFF	[3B4] (6A4) (8A1) (9B2) (9B3) (12B1)
TWFF*	[3B4] (3B3) (6A3) (8A1) (8B3) (13B3)
WAIT*	[8B2] (8B1) (8B2) (12A3)
WTRL*	[8B2] (8B3) (9B3) (12A1)
XCT*	[7B1] (5B1) (7A2) (8A1) (8B2)
KEY SW GND	[19B1] (2A2) (2A2) (2A3) (2A3) (2A4) (2B4)

L88-0397-04 (SH 23)

ECN	DATE
1355	6-16-72
1376	6-2-72
1329	3-22-72
1266	11-29-71

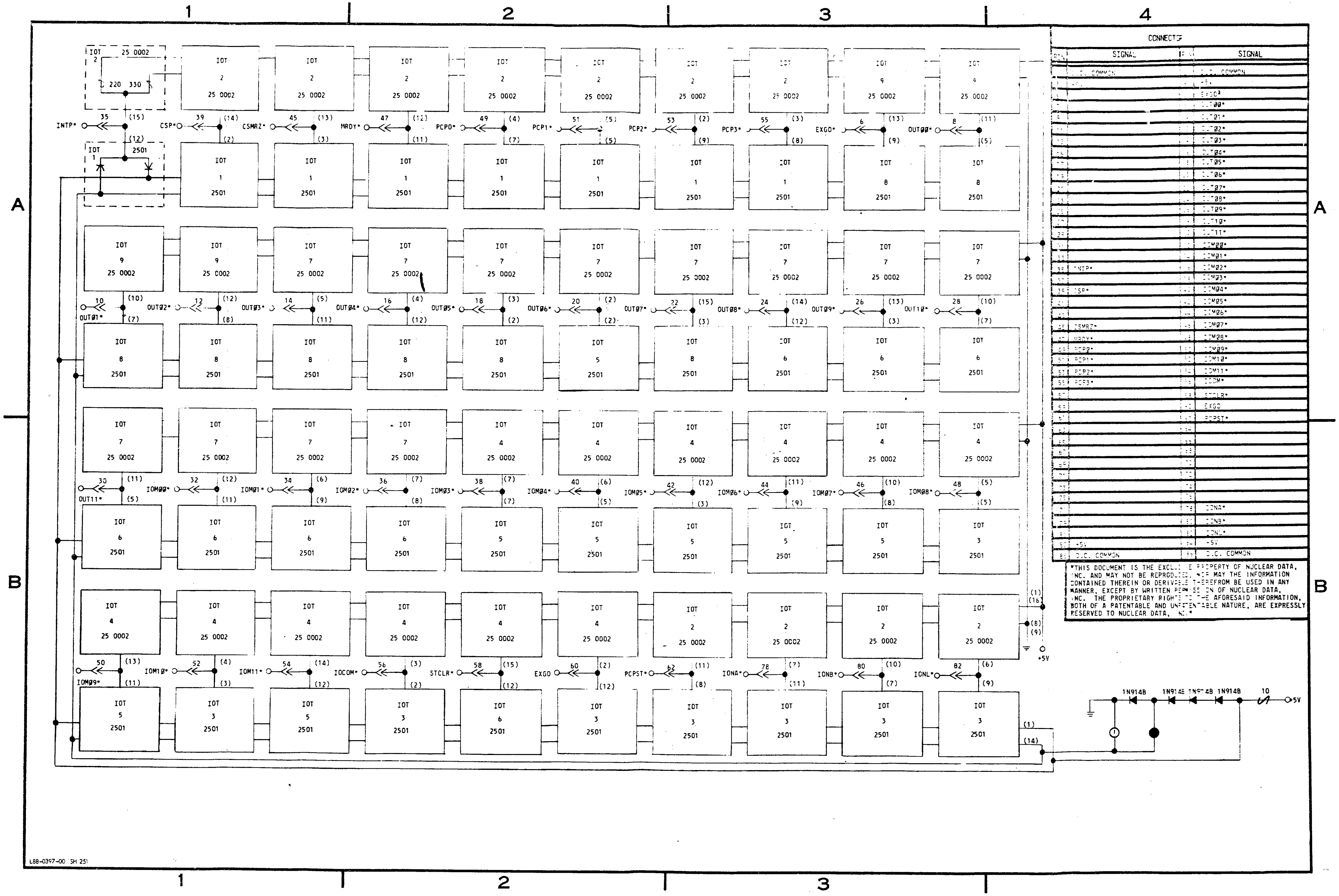
REVISIONS

Figure 7-2. ND312 Long Bus Signal Table (Sheet 23 of 28)



CONNECTOR			
PIN	SIGNAL	PIN	SIGNAL
1	D.C. COMMON	2	D.C. COMMON
3	+5V	4	+5V
5	DIN*	6	
7	EXT00*	8	
9	EXT01*	10	
11	EXT02*	12	
13	EXT03*	14	
15	EXT04*	16	
17	EXT05*	18	
19	EXT06*	20	
21	EXT07*	22	
23	EXT08*	24	
25	EXT09*	26	
27	EXT10*	28	
29	EXT11*	30	
31	EXTK*	32	
33	INTR*	34	
35		36	
37	CSR*	38	
39		40	
41	ALTER*	42	
43	SDCS*	44	
45		46	
47		48	
49		50	
51		52	
53		54	
55		56	
57	SMFO*	58	
59	SMFI*	60	
61	SKPR*	62	
63	START*	64	ALODA*
65	STOP*	66	EXCLJ*
67	CONT*	68	TSTD*
69	EXAM*	70	MXS00*
71	LDPR*	72	MXS01*
73	LDMR*	74	MXS02*
75	SI*	76	ARMX*
77	CSPO*	78	
79	ECHO*	80	
81	INTPO*	82	
83	+5V	84	+5V
85	D.C. COMMON	86	D.C. COMMON

Figure 7-2. ND812, ND-IOR Board, Logic Diagram (Sheet 24 of 28)

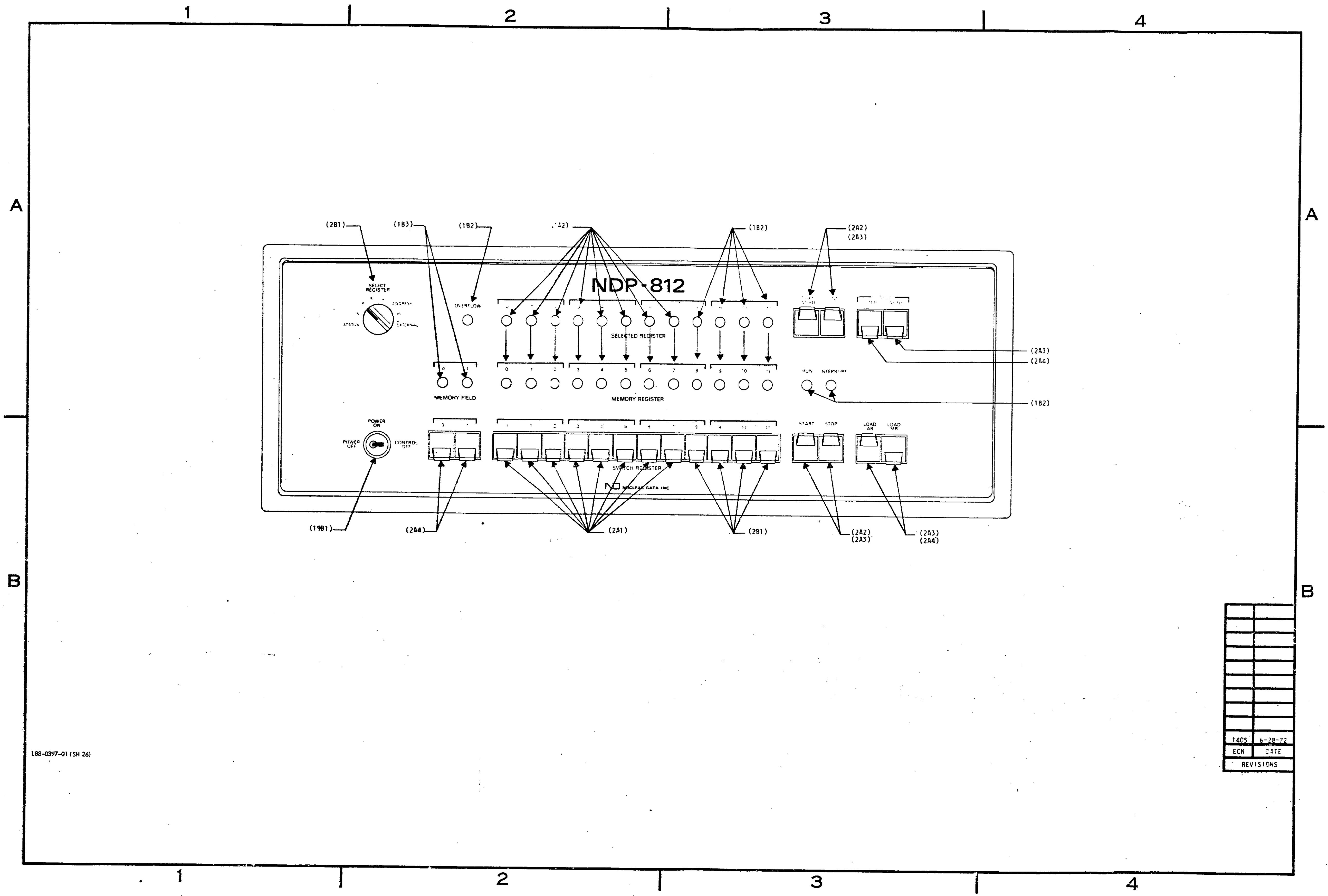


CONNECTOR

PIN	SIGNAL	PIN	SIGNAL
1	D.C. COMMON	2	D.C. COMMON
3	+5V	4	+5V
5		6	EXGO*
7		8	OUTØØ*
9		10	OUTØ1*
11		12	OUTØ2*
13		14	OUTØ3*
15		16	OUTØ4*
17		18	OUTØ5*
19		20	OUTØ6*
21		22	OUTØ7*
23		24	OUTØ8*
25		26	OUTØ9*
27		28	OUT1Ø*
29		30	OUT11*
31		32	IOMØØ*
33		34	IOMØ1*
35	INTP*	36	IOMØ2*
37		38	IOMØ3*
39	CSP*	40	IOMØ4*
41		42	IOMØ5*
43		44	IOMØ6*
45	CSMRZ*	46	IOMØ7*
47	MRDY*	48	IOMØ8*
49	PCPØ*	50	IOMØ9*
51	PCP1*	52	IOM1Ø*
53	PCP2*	54	IOM11*
55	PCP3*	56	IOCOM*
57		58	STCLR*
59		60	EXGO
61		62	PCPST*
63		64	
65		66	
67		68	
69		70	
71		72	
73		74	
75		76	
77		78	IONA*
79		80	IONB*
81		82	IONL*
83	+5V	84	-5V
85	D.C. COMMON	86	D.C. COMMON

"THIS DOCUMENT IS THE EXCLUSIVE PROPERTY OF NUCLEAR DATA, INC. AND MAY NOT BE REPRODUCED, NOR MAY THE INFORMATION CONTAINED THEREIN OR DERIVABLE THEREFROM BE USED IN ANY MANNER, EXCEPT BY WRITTEN PERMISSION OF NUCLEAR DATA, INC. THE PROPRIETARY RIGHTS TO THE AFORESAID INFORMATION, BOTH OF A PATENTABLE AND UNPATENTABLE NATURE, ARE EXPRESSLY RESERVED TO NUCLEAR DATA, INC."

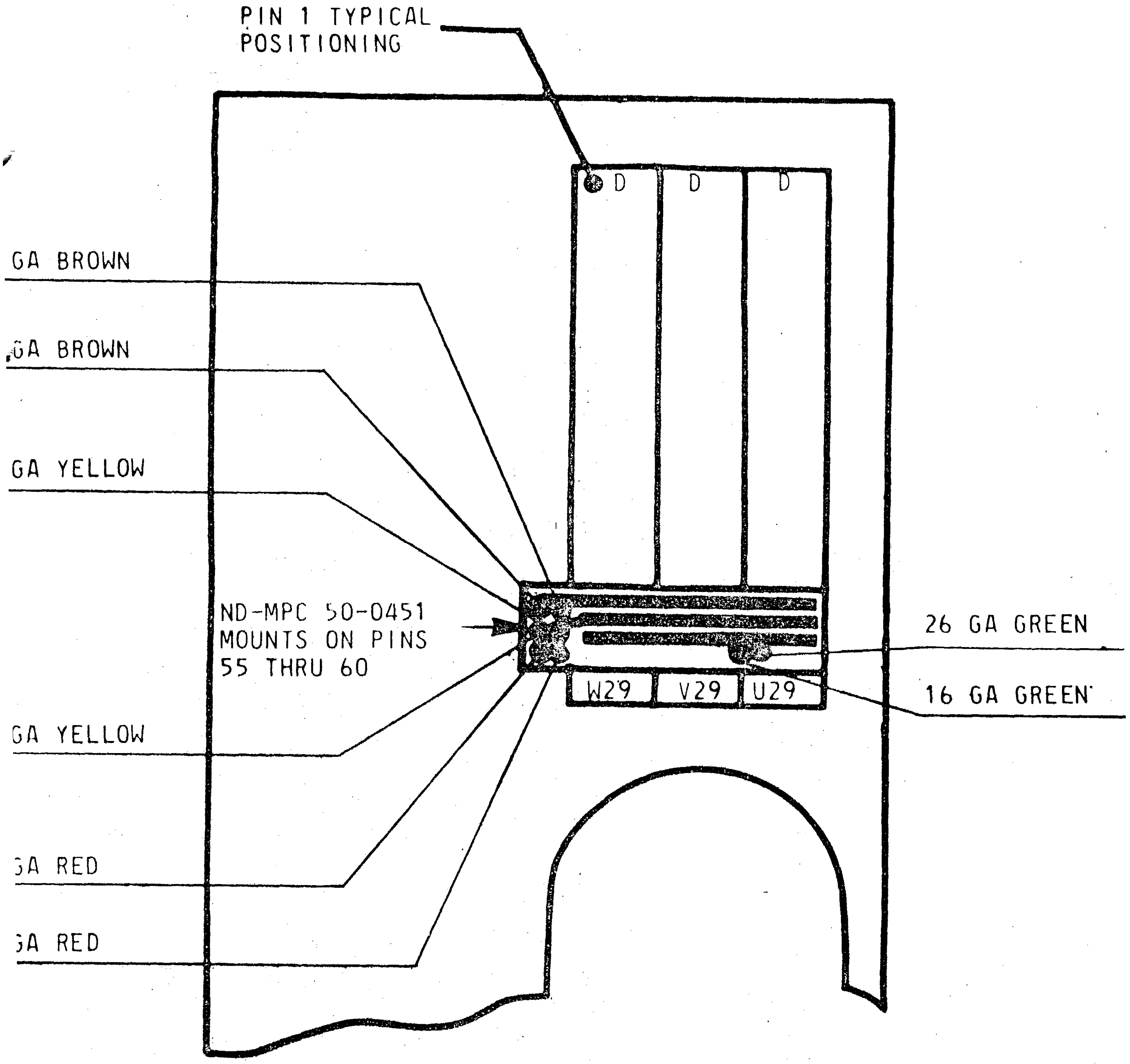
Figure 7-2. ND812, ND-IOT Board, Logic Diagram (Sheet 25 of 28)



1405	6-28-72
ECN	DATE
REVISIONS	

Figure 7-2. ND812, Front Panel Rendering and Diagram Location References (Sheet 26 of 28)

' DENOTES DOUBLE CONNECTOR



L88-0397-01 (SH 27)

Figure 7-2. ND812, Bussing Diagram (Sheet 27 Of 28)

)TES:

- "D" DENOTES DOUBLE CONNECTOR.

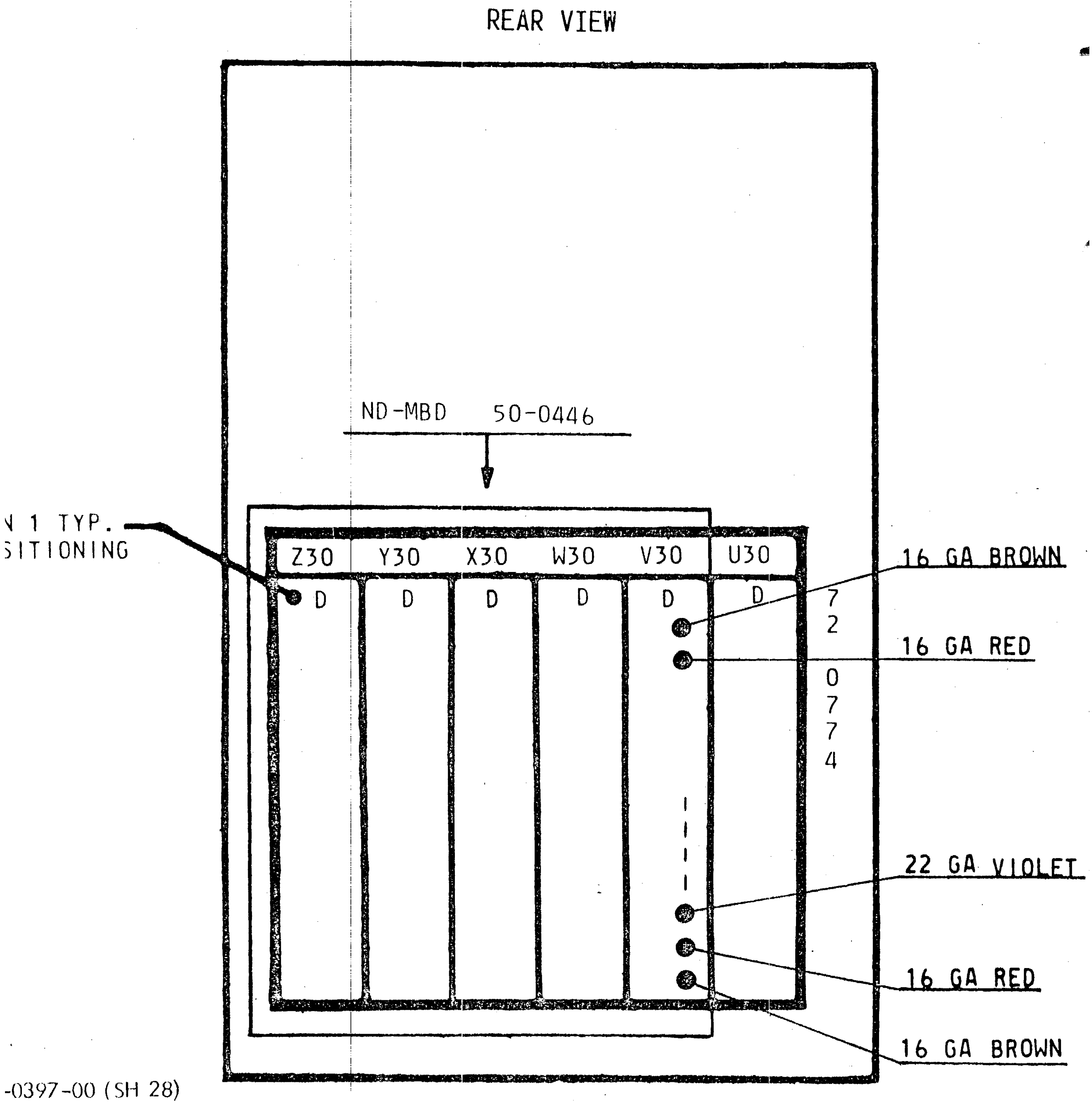
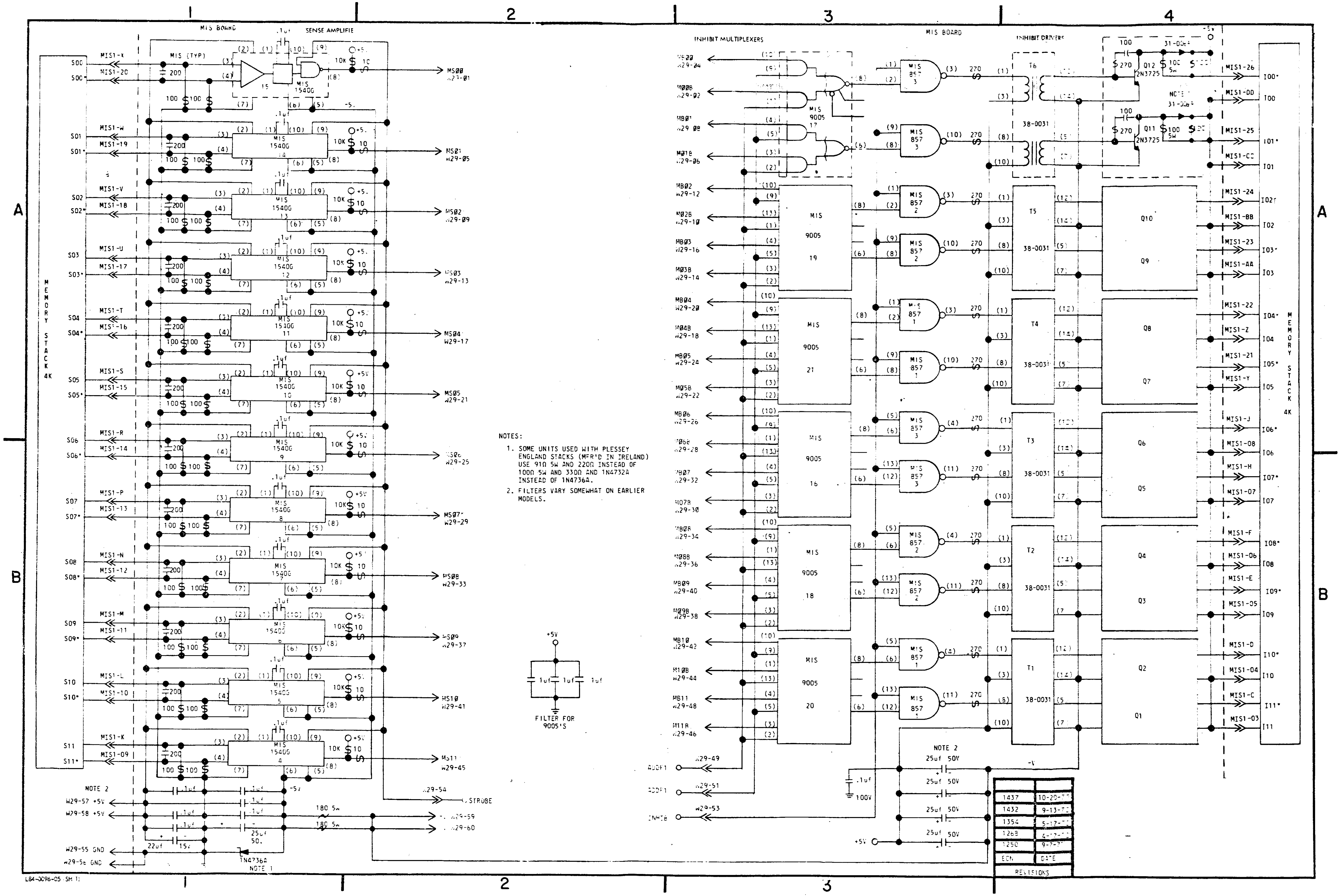


Figure 7-2. ND812, Bussing Diagram (Sheet 28 of 28)



NOTES:

1. SOME UNITS USED WITH PLESSEY ENGLAND STACKS (MFR'D IN IRELAND) USE 91Ω 5W AND 220Ω INSTEAD OF 100Ω 5W AND 330Ω AND 1N4732A INSTEAD OF 1N4736A.
2. FILTERS VARY SOMEWHAT ON EARLIER MODELS.

1437	10-20-77
1432	9-13-77
1354	5-17-77
1268	4-17-77
1250	9-7-77
ECN	DATE
REVISIONS	

Figure 7-3. ND812 4K Memory, ND-MTS1 Board, Sense Amps and Inhibit Drivers, Logic Diagram (Sheet 1 of 6)

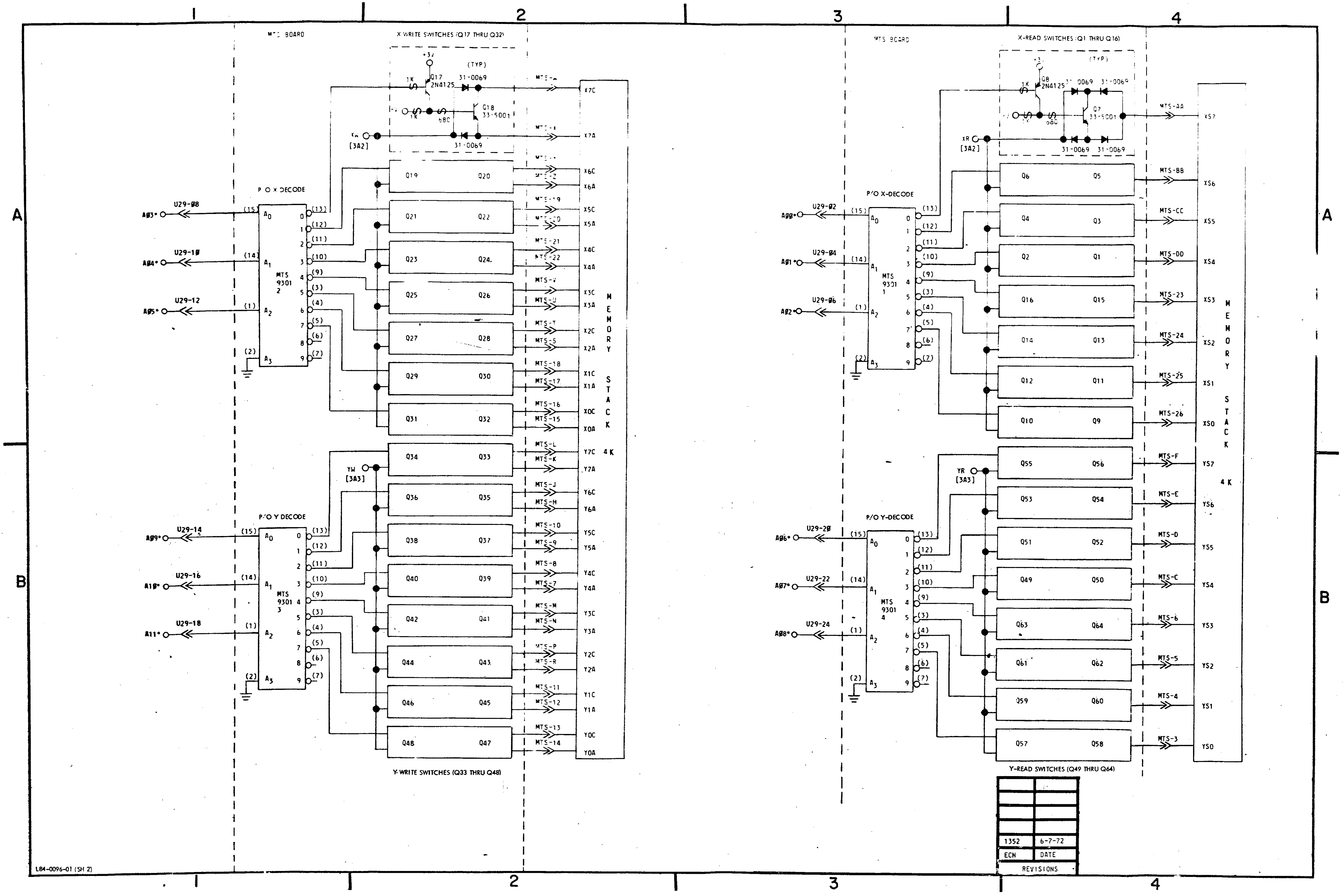


Figure 7-3. ND812 4K Memory, ND-MTS Board, X/Y Read and X/Y Write, Logic Diagram (Sheet 2 of 6)

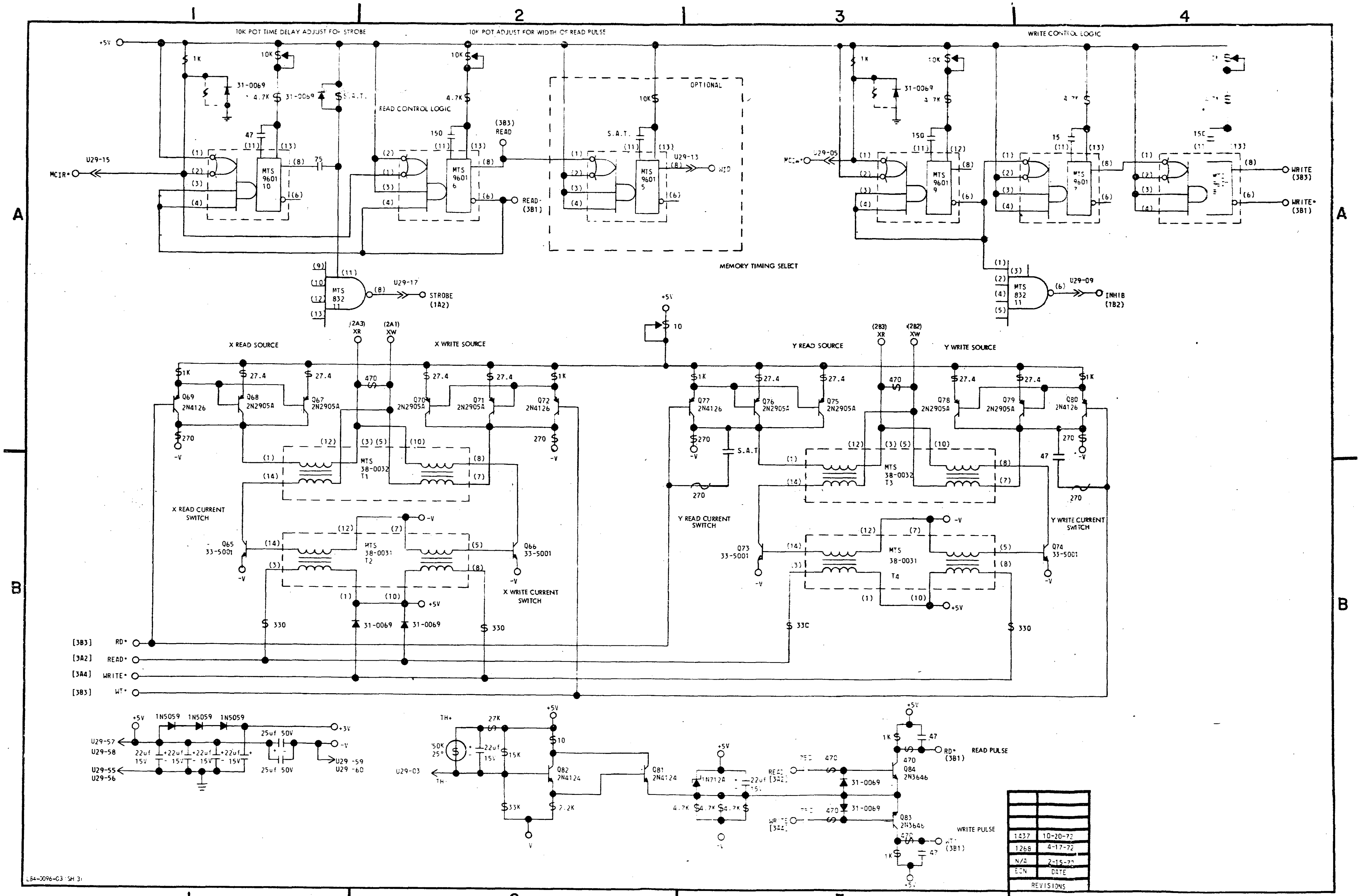


Figure 7-3. ND812 4K Memory, ND-MTS Board, Read/Write Control, Logic Diagram (Sheet 3 of 6)

MTS CONNECTOR J29

PIN	SIGNAL	PIN	SIGNAL
1		2	AØØ* (2A3)
3	TH- (3B2)	4	AØ1* (2A3)
5	MCI-W* (3A3)	6	AØ2* (2A3)
7		8	AØ3* (2A1)
9	INHIB [3A4](1B3	10	AØ4* (2A1)
11		12	AØ5* (2A1)
13	WID (3A3)	14	AØ9* (2B1)
15	MCIR* (3A1)	16	A1Ø* (2B1)
17	STROBE [3A2](1B2	18	A11* (2B1)
19		20	AØ6* (2B3)
21		22	AØ7* (2B3)
23		24	AØ8* (2B3)
25		26	
27		28	
29		30	
31		32	
33		34	
35		36	
37		38	
39		40	
41		42	
43		44	
45		46	
47		48	
49		50	
51		52	
53		54	
55	D.C. COM. (3B1)	56	D.C. COM. (3B1)
57	+5 (3B1)	58	+5 (3B1)
59	-V (3B1)	60	-V (3B1)

MTS CONNECTOR J29

PIN	SIGNAL	PIN	SIGNAL
1	MSØØ* [1A2]	2	MØØB (1A3)
3		4	MBØØ (1A3)
5	MSØ1* [1A2]	6	MØ1B (1A3)
7		8	MBØ1 (1A3)
9	MSØ2* [1A2]	10	MØ2B (1A3)
11		12	MBØ2 (1A3)
13	MSØ3* [1A2]	14	MØ3B (1A3)
15		16	MBØ3 (1A3)
17	MSØ4* [1A2]	18	MØ4B (1A3)
19		20	MBØ4 (1A3)
21	MSØ5* [1A2]	22	MØ5B (1A3)
23		24	MBØ5 (1A3)
25	MSØ6* [1B2]	26	MBØ6 (1A3)
27		28	MØ6B (1A3)
29	MSØ7* [1B2]	30	MØ7B (1B3)
31		32	MBØ7 (1B3)
33	MSØ8* [1B2]	34	MBØ8 (1B3)
35		36	MØ8B (1B3)
37	MSØ9* [1B2]	38	MØ9B (1B3)
39		40	MBØ9 (1B3)
41	MS1Ø* [1B2]	42	MB1Ø (1B3)
43		44	M1ØB (1B3)
45	MS11* [1B2]	46	M11B (1B3)
47		48	MB11 (1B3)
49	ADDF1* (1B3)	50	
51	ADDF1 (1B3)	52	
53	INHIB (1B3)[3A4]	54	STROBE (1B2)[3A2]
55	GND (1B1)	56	GND (1B1)
57	+5 (1B1)	58	+5 (1B1)
59	-V (1B2)	60	-V (1B2)

MTS CONNECTOR TO STACK

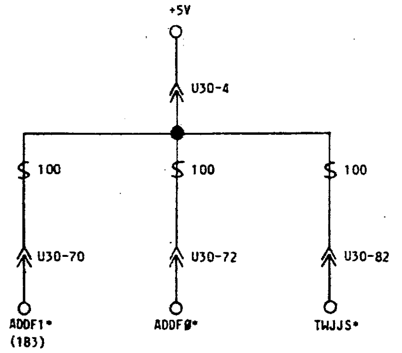
PIN	SIGNAL	PIN	SIGNAL
1		A	
2		B	
3	YSØ [2B4]	C	YS4 [2B4]
4	YS1 [2B4]	D	YS5 [2B4]
5	YS2 [2B4]	E	YS6 [2B4]
6	YS3 [2B4]	F	YS7 [2B4]
7	Y4A [2B2]	H	Y6A [2B2]
8	Y4C [2B2]	J	Y6C [2B2]
9	Y5A [2B2]	K	Y7A [2B2]
10	Y5C [2B2]	L	Y7C [2B2]
11	Y1C [2B2]	M	Y3C [2B2]
12	Y1A [2B2]	N	Y3A [2B2]
13	YØC [2B2]	P	Y2C [2B2]
14	YØA [2B2]	R	Y2A [2B2]
15	XØA [2A2]	S	X2A [2A2]
16	XØC [2A2]	T	X2C [2A2]
17	X1A [2A2]	U	X3A [2A2]
18	X1C [2A2]	V	X3C [2A2]
19	X5C [2A2]	W	X7C [2A2]
20	X5A [2A2]	X	X7A [2A2]
21	X4C [2A2]	Y	X6C [2A2]
22	X4A [2A2]	Z	X6A [2A2]
23	XS3 [2A4]	AA	XS7 [2A4]
24	XS2 [2A4]	BB	XS6 [2A4]
25	XS1 [2A4]	CC	XS5 [2A4]
26	XSØ [2A4]	DD	XS4 [2A4]
27		EE	
28		FF	

MIS 1 CONNECTOR TO STACK

PIN	SIGNAL	PIN	SIGNAL
1		A	
2		B	
3	I11 [1B4]	C	I11* [1B4]
4	I1Ø [1B4]	D	I1Ø* [1B4]
5	IØ9 [1B4]	E	IØ9* [1B4]
6	IØ8 [1B4]	F	IØ8* [1B4]
7	IØ7 [1B4]	H	IØ7* [1B4]
8	IØ6 [1A4]	J	IØ6* [1A4]
9	S11* [1B1]	K	S11 [1B1]
10	S1Ø* [1B1]	L	S1Ø [1B1]
11	SØ9* [1B1]	M	SØ9 [1B1]
12	SØ8* [1B1]	N	SØ8 [1B1]
13	SØ7* [1B1]	P	SØ7 [1B1]
14	SØ6* [1B1]	R	SØ6 [1B1]
15	SØ5* [1A1]	S	SØ5 [1A1]
16	SØ4* [1A1]	T	SØ4 [1A1]
17	SØ3* [1A1]	U	SØ3 [1A1]
18	SØ2* [1A1]	V	SØ2 [1A1]
19	SØ1* [1A1]	W	SØ1 [1A1]
20	SØØ* [1A1]	X	SØØ [1A1]
21	IØ5* [1A4]	Y	IØ5 [1A4]
22	IØ4* [1A4]	Z	IØ4 [1A4]
23	IØ3* [1A4]	AA	IØ3 [1A4]
24	IØ2* [1A4]	BB	IØ2 [1A4]
25	IØ1* [1A4]	CC	IØ1 [1A4]
26	IØØ* [1A4]	DD	IØØ [1A4]
27		EE	
28		FF	

TCC CABLE CONNECTOR U30

PIN	SIGNAL	PIN	SIGNAL
1	GND	2	GND
3	+5V	4	+5V (4B3)
67	GND	66	GND
69	GND	68	GND
83	+5V	70	ADDF1* (4B3)
85	GND	72	ADDFØ* (4B3)
		82	TWJJS* (4B3)
		84	+5V
		86	GND

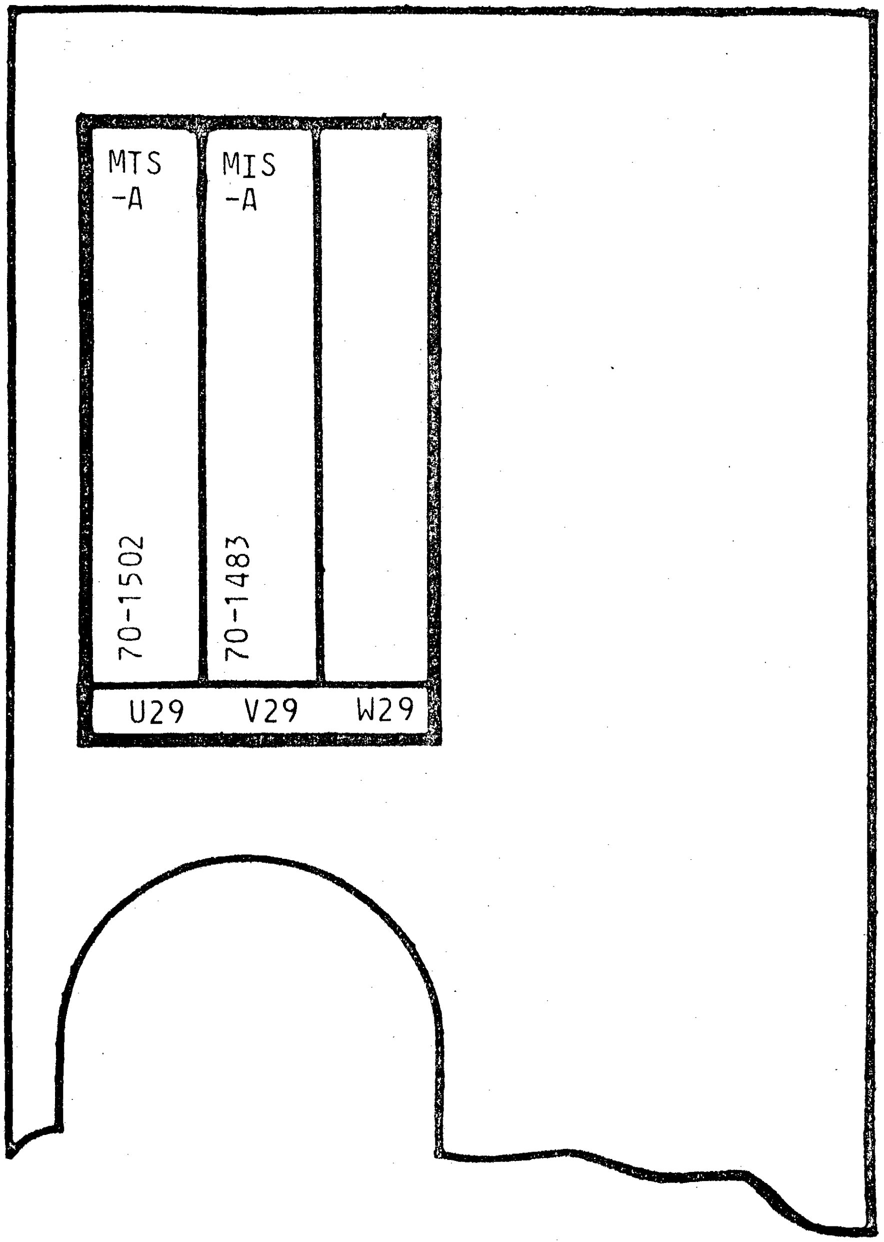


1268	4-17-72
ECN	DATE
REVISIONS	

L84-0096-01 (SH 4)

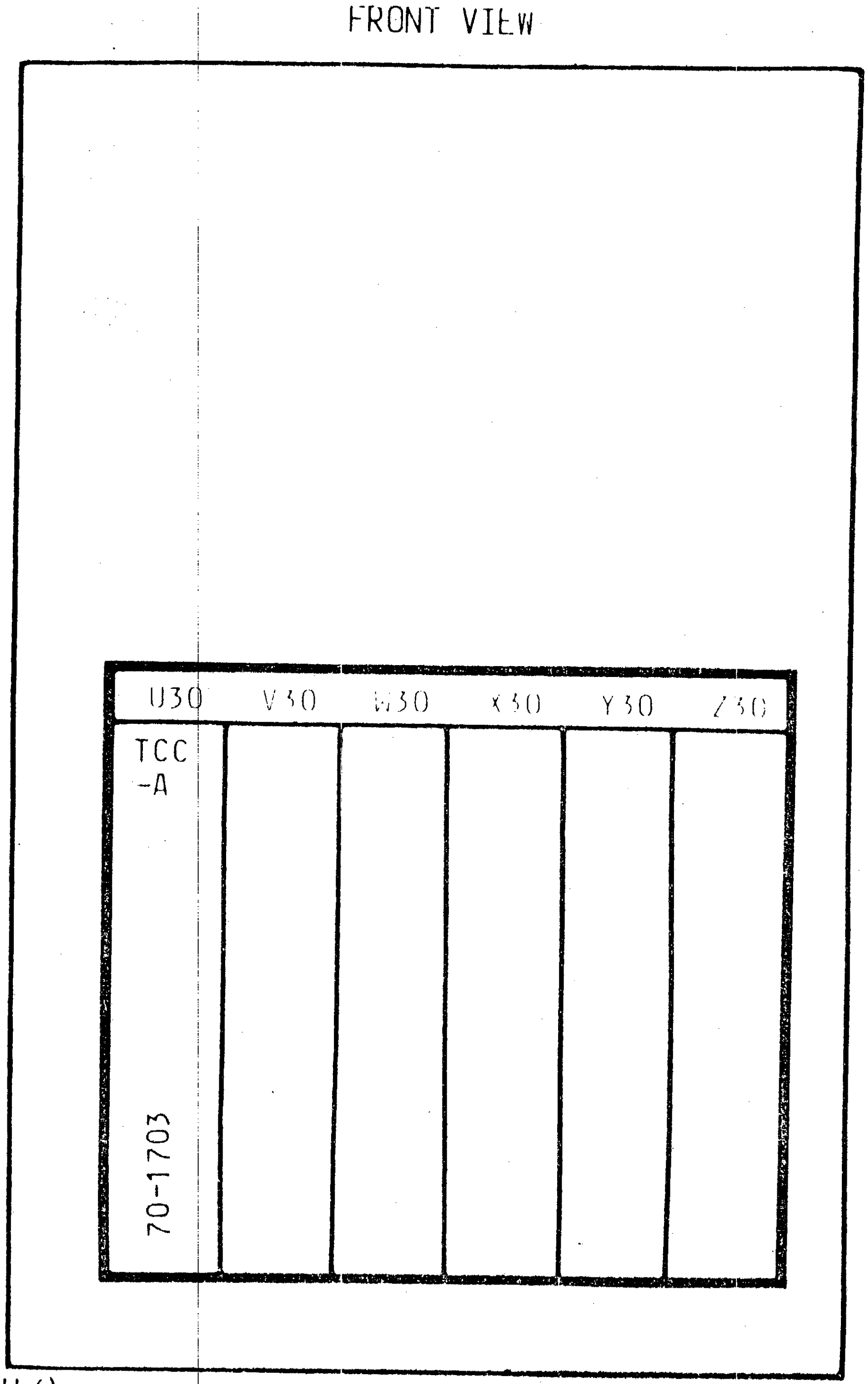
Figure 7-3. ND812 4K Memory, Connector Pin/Signal Designation/Location (Sheet 4 of 6)

FRONT VIEW



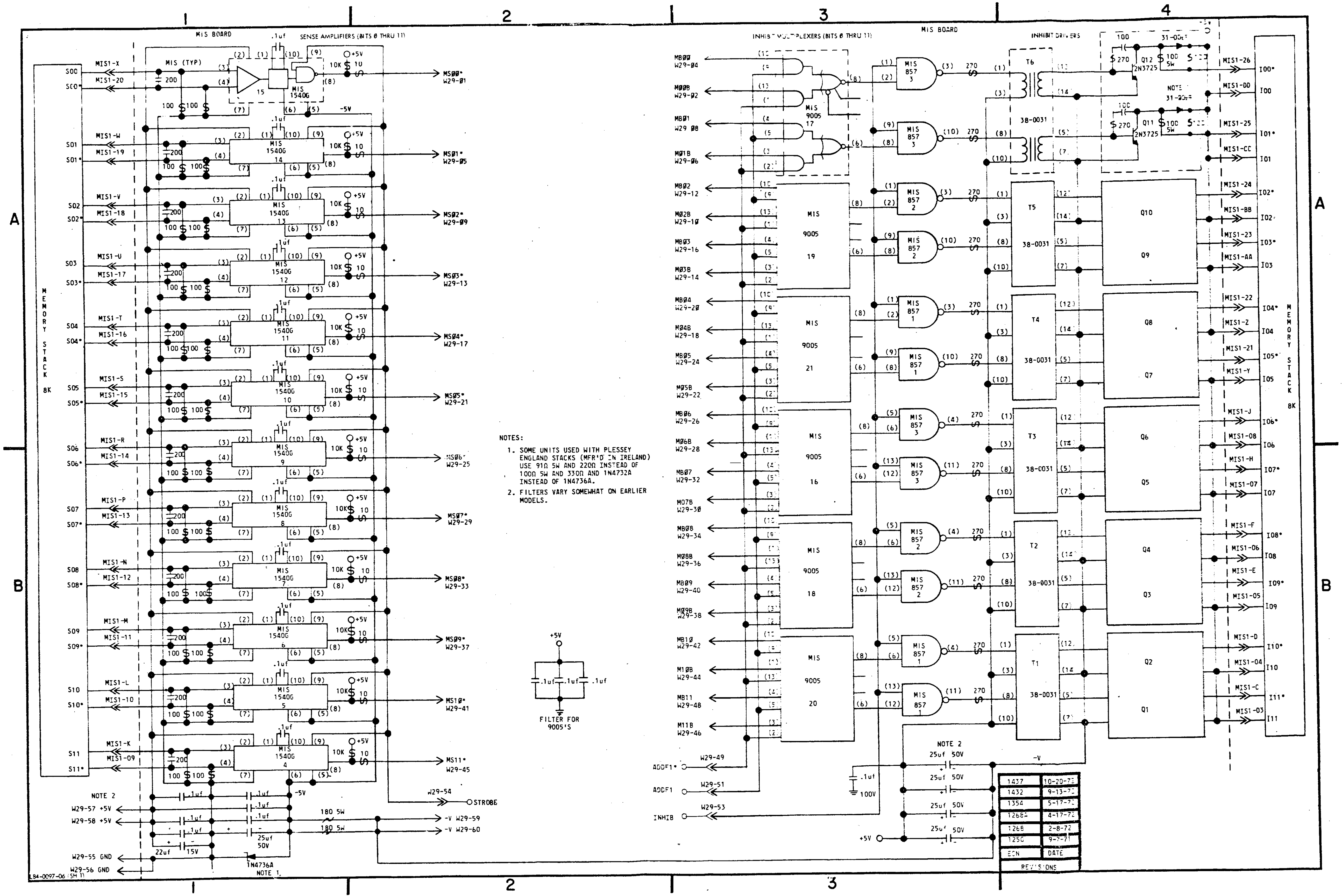
L84-0096-00 (SH 5)

Figure 7-3. ND812 4K Memory, Loading Diagram (Sheet 5 of 6)



L84-0096-00 (SH 6)

Figure 7-3. ND812 4K Memory, Loading Diagram (Sheet 6 of 6)

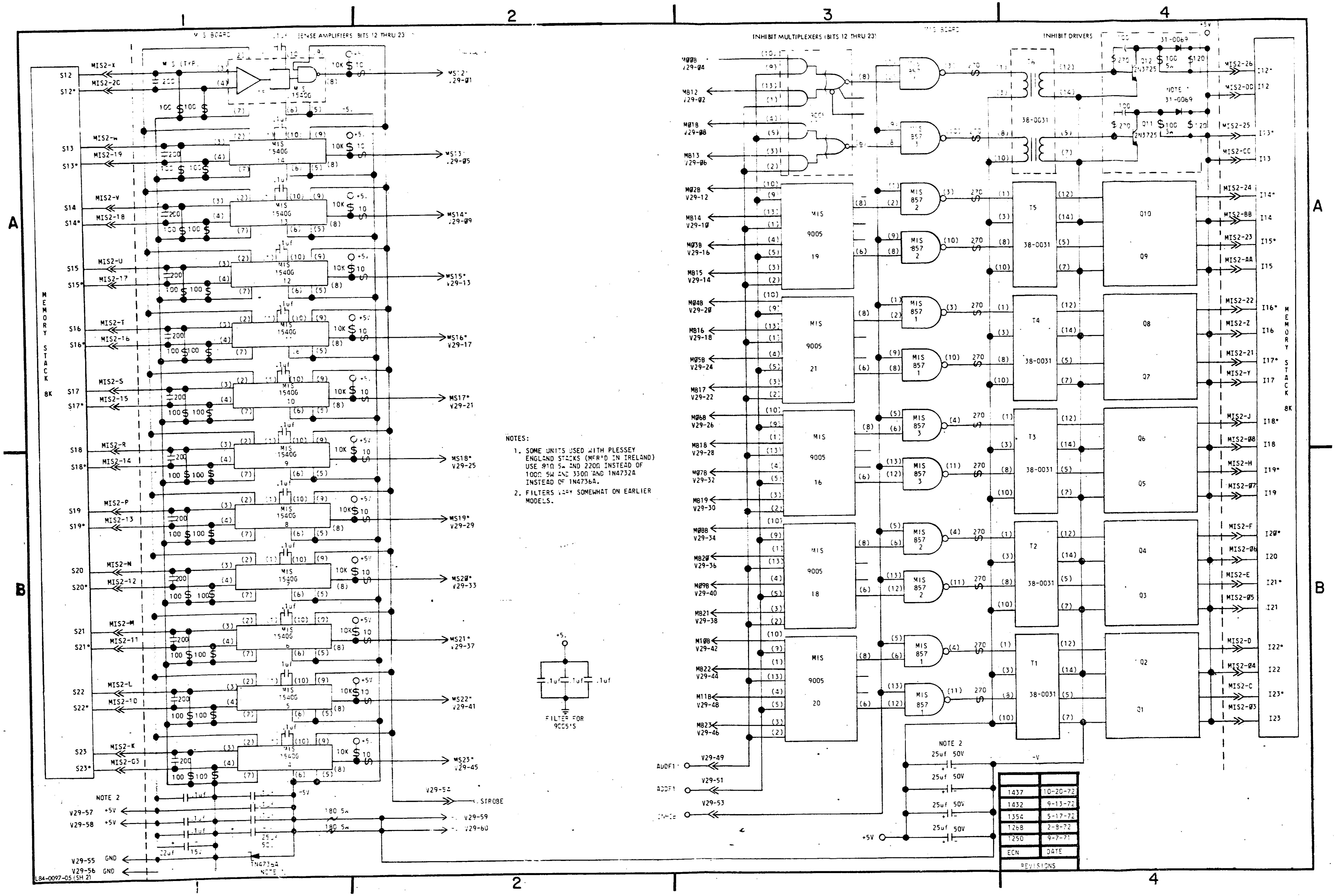


NOTES:

1. SOME UNITS USED WITH PLESSEY ENGLAND STACKS (MFR'D IN IRELAND) USE 91Ω 5W AND 220Ω INSTEAD OF 100Ω 5W AND 330Ω AND 1N4732A INSTEAD OF 1N4736A.
2. FILTERS VARY SOMEWHAT ON EARLIER MODELS.

1437	10-20-72
1432	9-13-72
1354	5-17-72
1268A	4-17-72
1268	2-8-72
1250	9-7-71
ECN	DATE
REVISIONS	

Figure 7-4. ND812 8K Memory, ND-MTS1 Board, Sense Amps and Inhibit Drivers, Logic Diagram (Sheet 1 of 10)



NOTES:

1. SOME UNITS USED WITH PLESSEY ENGLAND STACKS (MFR'D IN IRELAND) USE 91Ω 5W AND 220Ω INSTEAD OF 100Ω 5W AND 330Ω AND 1N4732A INSTEAD OF 1N4736A.
2. FILTERS VARY SOMEWHAT ON EARLIER MODELS.

1437	10-20-72
1432	9-13-72
1354	5-17-72
1268	2-8-72
1250	9-7-71
ECN	DATE
REVISIONS	

Figure 7-4. ND812 8K Memory, ND-MTS2 Board, Sense Amps and Inhibit Drivers, Logic Diagram (Sheet 2 of 10)

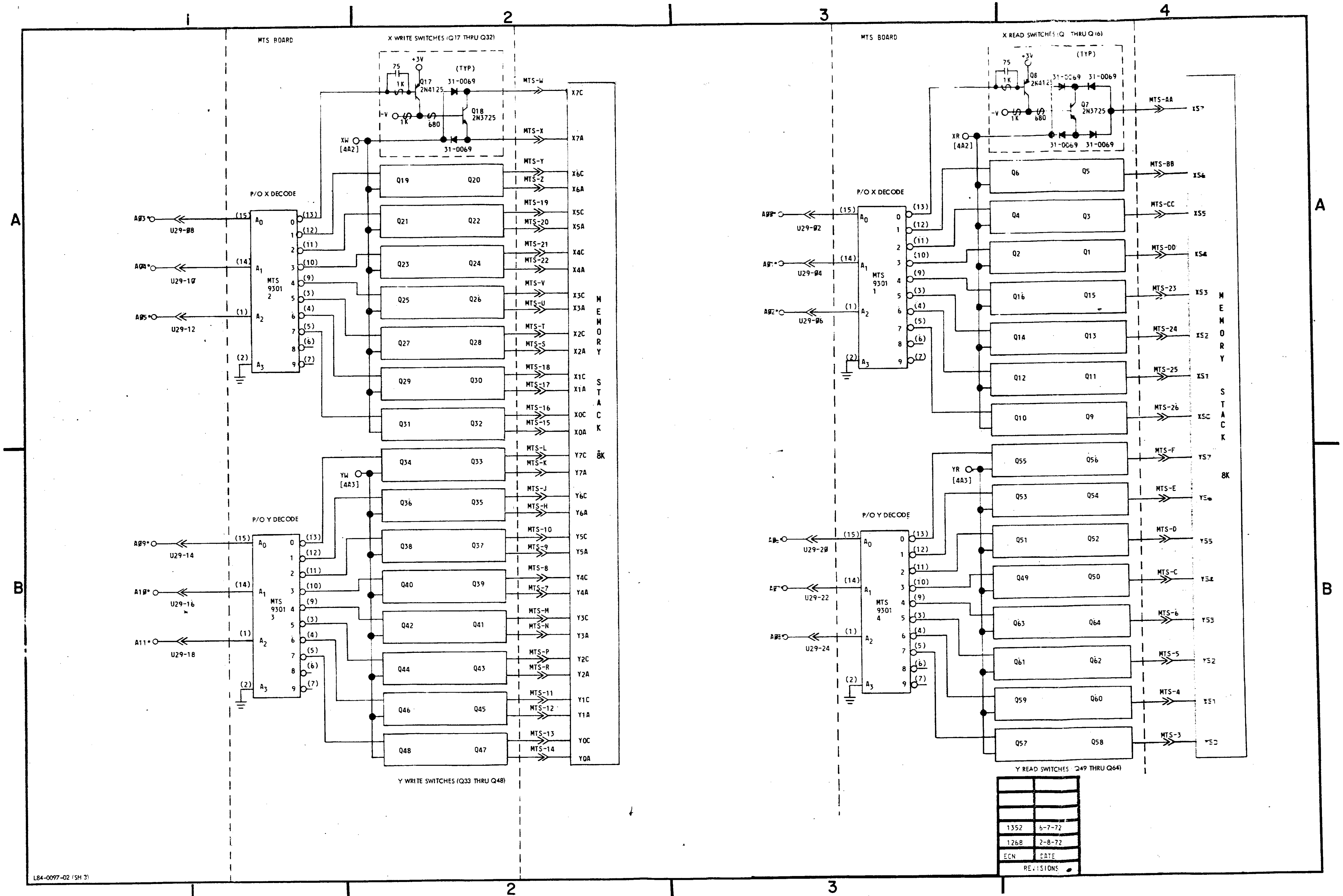


Figure 7-4. ND812 8K Memory, ND-MTS Board, X/Y Read and X/Y Write, Logic Diagram (Sheet 3 of 10)

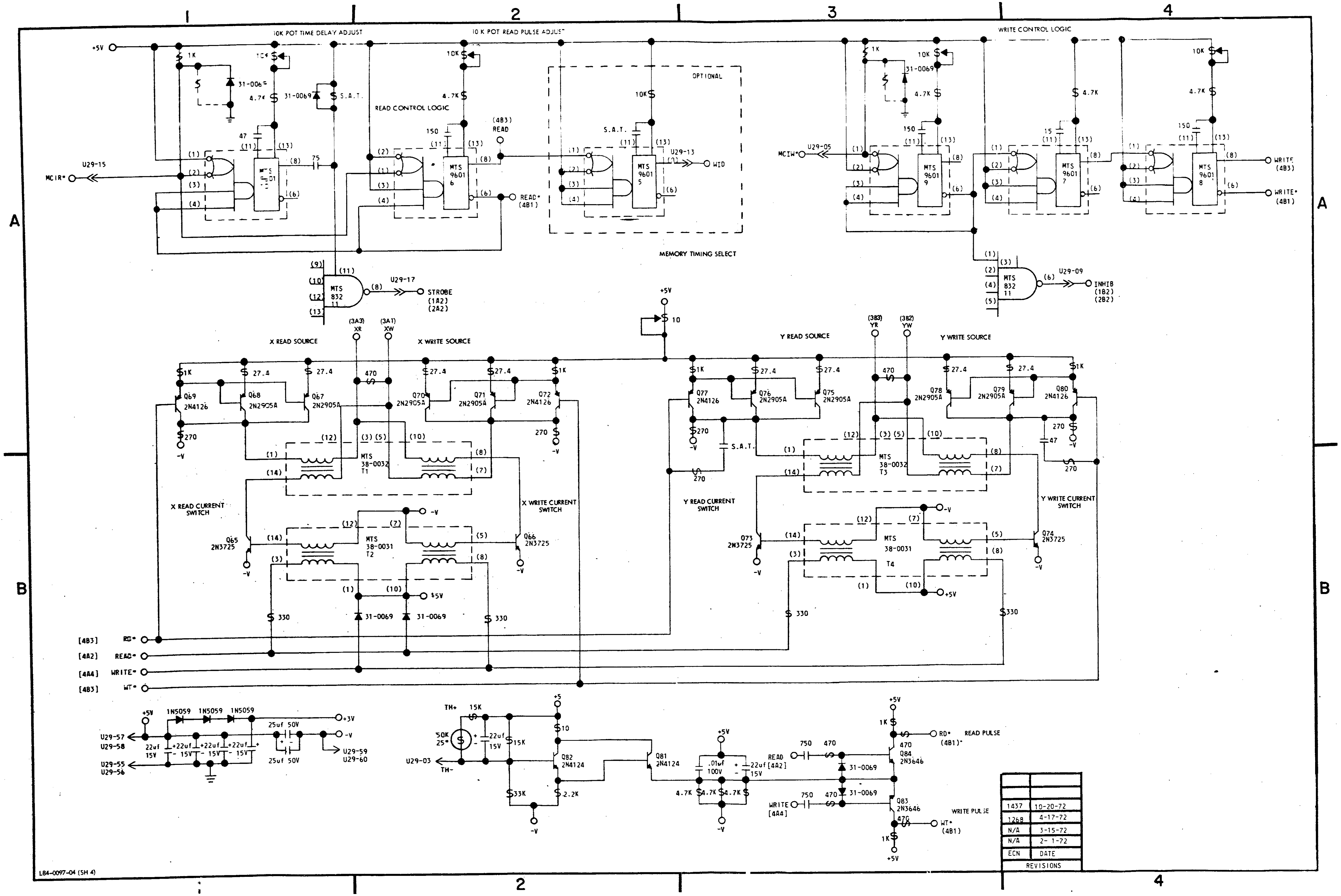


Figure 7-4. ND812 8K Memory, ND-MTS Board, Read/Write Control, Logic Diagram (Sheet 4 of 10)

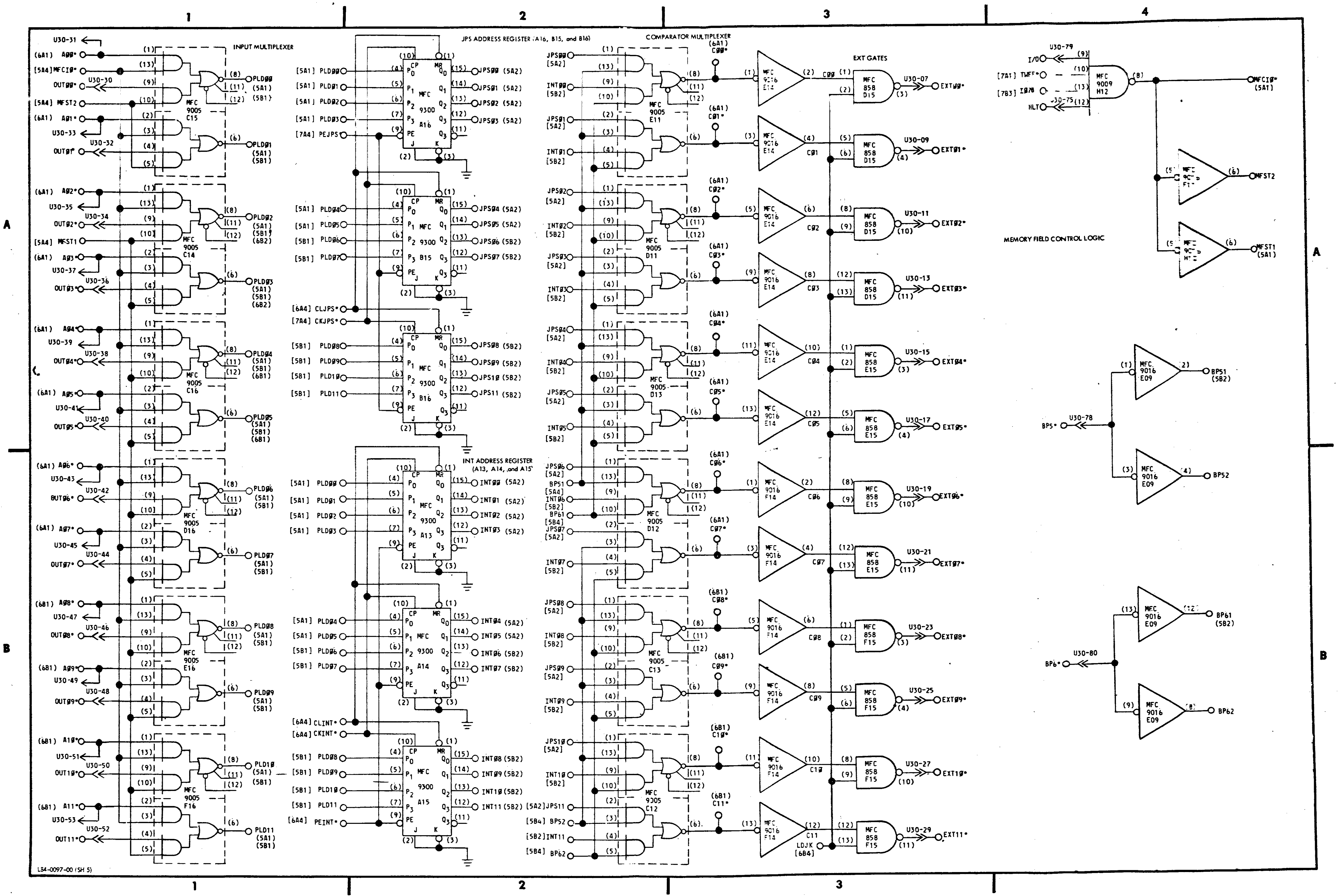


Figure 7-4. ND812 8K Memory, ND-MFC Board, Memory Field Control, Logic Diagram (Sheet 5 of 10)

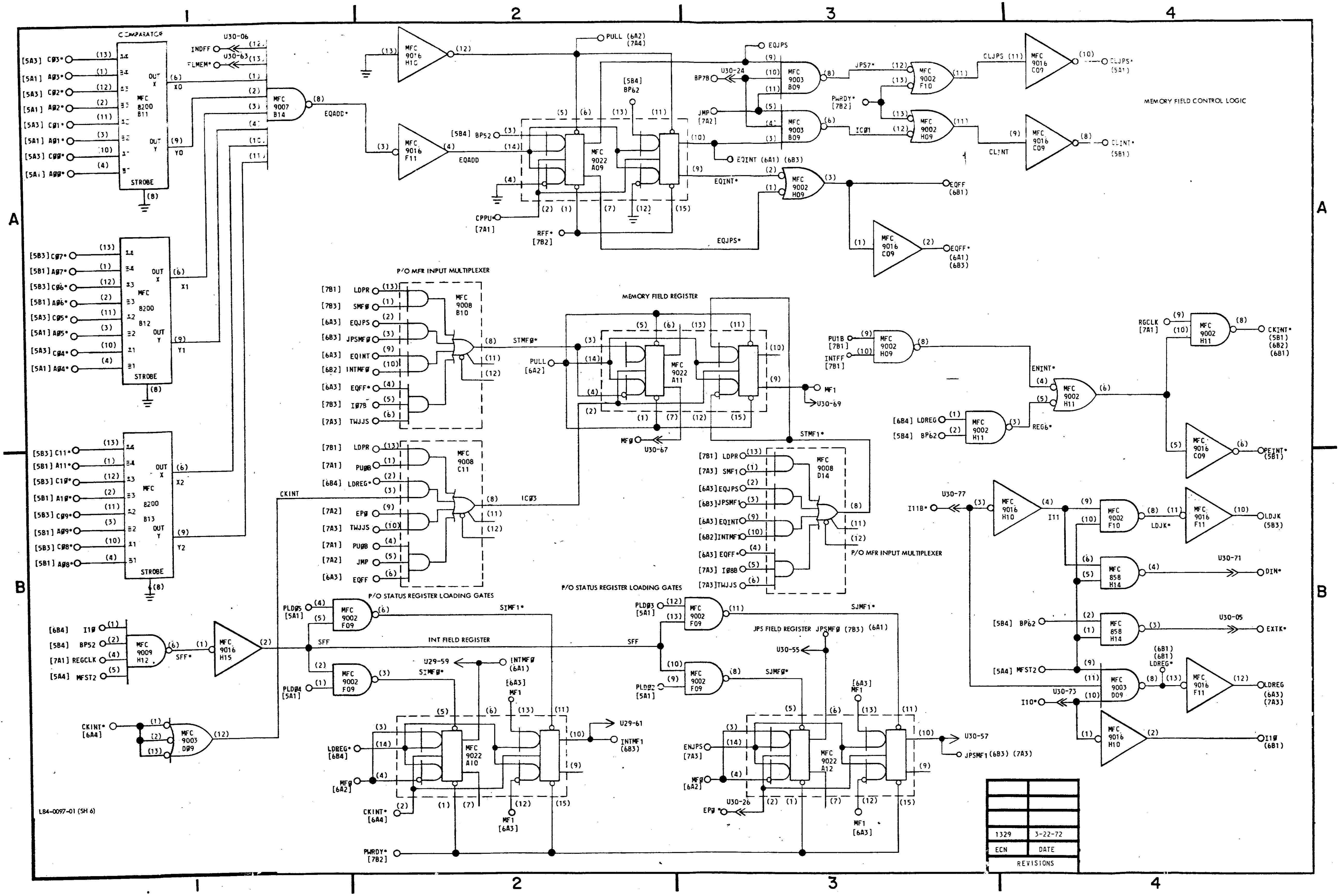


Figure 7-4. ND812 8K Memory, ND-MFC Board, Logic Diagram (Sheet 6 of 10)

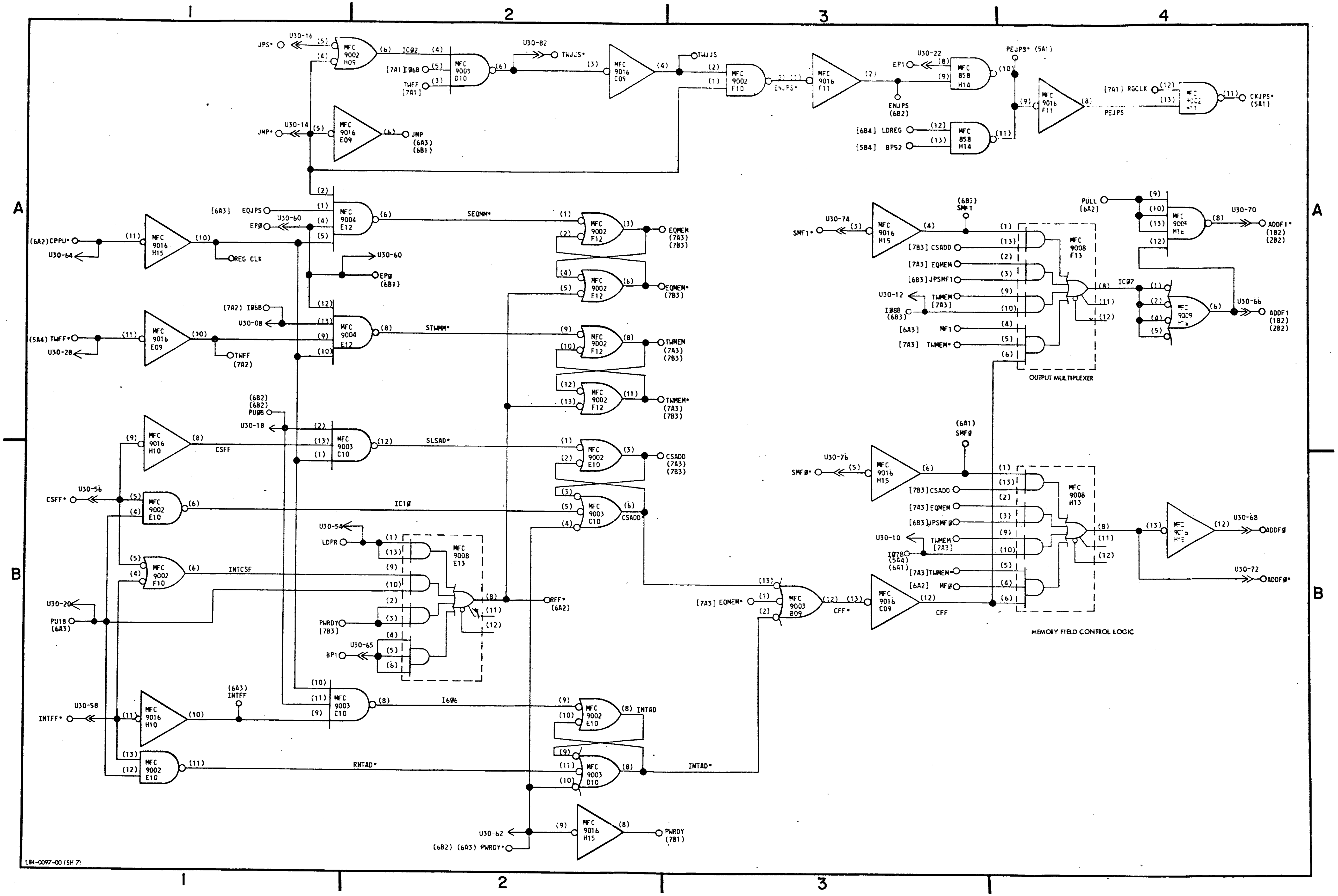


Figure 7-4. ND812 8K Memory, ND-MFC Board, Memory Field Control, Logic Diagram (Sheet 7 of 10)

MTS CONNECTOR J29

PIN	SIGNAL	PIN	SIGNAL
1		2	AØØ* (3A3)
3	TH- (4B2)	4	AØ1* (3A3)
5	MCIW* (4A3)	6	AØ2* (3A3)
7		8	AØ3* (3A1)
9	INHIB [4A4](2B2)	10	AØ4* (3A1)
11		12	AØ5* (3A1)
13	WID (4A3)	14	AØ9* (3B1)
15	MCIR* (4A1)	16	A1Ø* (3B1)
17	STROBE[4A2](1A2)(2A2	18	A11* (3B1)
19		20	AØ6* (3B3)
21		22	AØ7* (3B3)
23		24	AØ8* (3B3)
25		26	
27		28	
29		30	
31		32	
33		34	
35		36	
37		38	
39		40	
41		42	
43		44	
45		46	
47		48	
49		50	
51		52	
53		54	
55	GND (4B1)	56	GND (4B1)
57	+5V (4B1)	58	+5V (4B1)
59	-V (4B1)	60	-V (4B1)

MIS 2ND 4K J29

PIN	SIGNAL	PIN	SIGNAL
1	MS12* [2A2]	2	MB12 (2A3)
3		4	MØØB (2A3)
5	MS13* [2A2]	6	MB13 (2A3)
7		8	MØ1B (2A3)
9	MS14* [2A2]	10	MB14 (2A3)
11		12	MØ2B (2A3)
13	MS15* [2A2]	14	MB15 (2A3)
15		16	MØ3B (2A3)
17	MS16* [2A2]	18	MB16 (2A3)
19		20	MØ4B (2A3)
21	MS17* [2A2]	22	MB17 (2A3)
23		24	MØ5B (2A3)
25	MS18* [2B2]	26	MØ6B (2A3)
27		28	MB18 (2A3)
29	MS19* [2B2]	30	MB19 (2B3)
31		32	MØ7B (2B3)
33	MS20* [2B2]	34	MØ8B (2B3)
35		36	MB20 (2B3)
37	MS21* [2B2]	38	MB21 (2B3)
39		40	MØ9B (2B3)
41	MS22* [2B2]	42	M1ØB (2B3)
43		44	MB22 (2B3)
45	MS23* [2B2]	46	MB23 (2B3)
47		48	M11B (2B3)
49	ADDF1* (2B2)[7A4]	50	
51	ADDF1 (2B2)(7B4)	52	
53	INHIB (2B2)(4A4)	54	STROBE(1A2)(2A2)[4A2]
55	GND (2B1)	56	GND (2B1)
57	+5V (2B1)	58	+5V (2B1)
59	-V (2B2)	60	-V (2B2)

MIS 1ST 4K J29

PIN	SIGNAL	PIN	SIGNAL
1	MSØØ* [1A2]	2	MØØB (1A3)
3		4	MBØØ (1A3)
5	MSØ1* [1A2]	6	MØ1B (1A3)
7		8	MBØ1 (1A3)
9	MSØ2* [1A2]	10	MØ2B (1A3)
11		12	MBØ2 (1A3)
13	MSØ3* [1A2]	14	MØ3B (1A3)
15		16	MBØ3 (1A3)
17	MSØ4* [1A2]	18	MØ4B (1A3)
19		20	MBØ4 (1A3)
21	MSØ5* [1A2]	22	MØ5B (1A3)
23		24	MBØ5 (1A3)
25	MSØ6* [1B2]	26	MBØ6 (1A3)
27		28	MØ6B (1A3)
29	MSØ7* [1B2]	30	MØ7B (1B3)
31		32	MBØ7 (1B3)
33	MSØ8* [1B2]	34	MBØ8 (1B3)
35		36	MØ8B (1B3)
37	MSØ9* [1B2]	38	MØ9B (1B3)
39		40	MBØ9 (1B3)
41	MS1Ø* [1B2]	42	MB1Ø (1B3)
43		44	M1ØB (1B3)
45	MS11* [1B2]	46	M11B (1B3)
47		48	MB11 (1B3)
49	ADDF1* (1B2)	50	
51	ADDF1 (1B2)	52	
53	INHIBIT (1B2)	54	STROBE(1A2)(2A2)[4A2]
55	GND (1B1)	56	GND (1B1)
57	+5 (1B1)	58	+5 (1B1)
59	-V (1B2)	60	-V (1B2)

MFC CONNECTOR U30

1	GNDØ1	2	GNDØ2
3	P5VØ3	4	P5VØ4
5	EXTK* [6B4]	6	INOFF (6A1)
7	EXTØØ* [5A3]	8	IØ6B (7A1)
9	EXTØ1* [5A3]	10	IØ7B (7B3)
11	EXTØ2* [5A3]	12	IØ8B (7A3)
13	EXTØ3* [5A3]	14	JMP* (7A1)
15	EXTØ4* [5A3]	16	JPS* (7A1)
17	EXTØ5* [5A3]	18	PUØB (7A1)
19	EXTØ6* [5B3]	20	PU1B (7B1)
21	EXTØ7* [5B3]	22	EP1 (7A3)
23	EXTØ8* [5B3]	24	BP7B (6A3)
25	EXTØ9* [5B3]	26	EPØ* (6B3)
27	EXT1Ø* [5B3]	28	TWFF* (7A1)
29	EXT11* [5B3]	30	OUTØØ* (5A1)
31	AØØ* (5A1)	32	OUTØ1* (5A1)
33	AØ1* (5A1)	34	OUTØ2* (5A1)
35	AØ2* (5A1)	36	OUTØ3* (5A1)
37	AØ3* (5A1)	38	OUTØ4* (5A1)
39	AØ4* (5A1)	40	OUTØ5* (5A1)
41	AØ5* (5A1)	42	OUTØ6* (5B1)
43	AØ6* (5B1)	44	OUTØ7* (5B1)
45	AØ7* (5B1)	46	OUTØ8* (5B1)
47	AØ8* (5B1)	48	OUTØ9* (5B1)
49	AØ9* (5B1)	50	OUT1Ø* (5B1)
51	A1Ø* (5B1)	52	OUT11* (5B1)
53	A11* (5B1)	54	LDPR (7B1)
55	JPSMFØ [6B3]	56	CSFF* (7B1)
57	JPSMF1 [6B3]	58	INTFF* (7B1)
59	INTMFØ [6B2]	60	EPØ (7A2)
61	INTMF1 [6B2]	62	PWRDY* (7B2)
63	FLMEM* (6A1)	64	CPPU* (7A1)
65	BP1 (7B1)	66	ADDF1 [7A4]
67	MFØ [6A2]	68	ADDFØ [7B4](2B2)
69	MF1 [6A3]	70	ADDF1* [7A4](2B2)
71	DIN* [6B4]	72	ADDFØ* [7B4]
73	I1Ø* (6B4)	74	SMF1* [7A3]
75	HLT (5A4)	76	SMFØ* [7B3]
77	I11B* (6B3)	78	BP5* (5A4)
79	I/O (5A4)	80	BP6* (5B4)
81		82	TWJJS* [7A2]
83	P5V83	84	P5V84
85	GND85	86	GND86

MTS CONNECTOR TO STACK

PIN	SIGNAL	PIN	SIGNAL
1		A	
2		B	
3	YSØ [3B4]	C	YS4 [3B4]
4	YS1 [3B4]	D	YS5 [3B4]
5	YS2 [3B4]	E	YS6 [3B4]
6	YS3 [3B4]	F	YS7 [3B4]
7	Y4A [3B2]	H	Y6A [3B2]
8	Y4C [3B2]	J	Y6C [3B2]
9	Y5A [3B2]	K	Y7A [3B2]
10	Y5C [3B2]	L	Y7C [3B2]
11	Y1C [3B2]	M	Y3C [3B2]
12	Y1A [3B2]	N	Y3A [3B2]
13	YØC [3B2]	P	Y2C [3B2]
14	YØA [3B2]	R	Y2A [3B2]
15	XØA [3A2]	S	X2A [3A2]
16	XØC [3A2]	T	X2C [3A2]
17	X1A [3A2]	U	X3A [3A2]
18	X1C [3A2]	V	X3C [3A2]
19	X5C [3A2]	W	X7C [3A2]
20	X5A [3A2]	X	X7A [3A2]
21	X4C [3A2]	Y	X6C [3A2]
22	X4A [3A2]	Z	X6A [3A2]
23	XS3 [3A4]	AA	XS7 [3A4]
24	XS2 [3A4]	BB	XS6 [3A4]
25	XS1 [3A4]	CC	XS5 [3A4]
26	XSØ [3A4]	DD	XS4 [3A4]
27		EE	
28		FF	

MTS2 CONNECTOR TO STACK

PIN	SIGNAL	PIN	SIGNAL
1		A	
2		B	
3	I23 [2B4]	C	I23* [2B4]
4	I22 [2B4]	D	I22* [2B4]
5	I21 [2B4]	E	I21* [2B4]
6	I2Ø [2B4]	F	I2Ø* [2B4]
7	I19 [2B4]	H	I19* [2B4]
8	I18 [2A4]	J	I18* [2A4]
9	S23* [2B1]	K	S23 [2B1]
10	S22* [2B1]	L	S22 [2B1]
11	S21* [2B1]	M	S21 [2B1]
12	S2Ø* [2B1]	N	S2Ø [2B1]
13	S19* [2B1]	P	S19 [2B1]
14	S18* [2B1]	R	S18 [2B1]
15	S17* [2A1]	S	S17 [2A1]
16	S16* [2A1]	T	S16 [2A1]
17	S15* [2A1]	U	S15 [2A1]
18	S14* [2A1]	V	S14 [2A1]
19	S13* [2A1]	W	S13 [2A1]
20	S12* [2A1]	X	S12 [2A1]
21	I17* [2A4]	Y	I17 [2A4]
22	I16* [2A4]	Z	I16 [2A4]
23	I15* [2A4]	AA	I15 [2A4]
24	I14* [2A4]	BB	I14 [2A4]
25	I13* [2A4]	CC	I13 [2A4]
26	I12* [2A4]	DD	I12 [2A4]
27		EE	
28		FF	

MIS 1 CONNECTOR TO STACK

PIN	SIGNAL	PIN	SIGNAL
1		A	
2		B	
3	I11 [1B4]	C	I11* [1B4]
4	I1Ø [1B4]	D	I1Ø* [1B4]
5	IØ9 [1B4]	E	IØ9* [1B4]
6	IØ8 [1B4]	F	IØ8* [1B4]
7	IØ7 [1B4]	H	IØ7* [1B4]
8	IØ6 [1A4]	J	IØ6* [1A4]
9	S11* [1B1]	K	S11 [1B1]
10	S1Ø* [1B1]	L	S1Ø [1B1]
11	SØ9* [1B1]	M	SØ9 [1B1]
12	SØ8* [1B1]	N	SØ8 [1B1]
13	SØ7* [1B1]	P	SØ7 [1B1]
14	SØ6* [1B1]	R	SØ6 [1B1]
15	SØ5* [1A1]	S	SØ5 [1A1]
16	SØ4* [1A1]	T	SØ4 [1A1]
17	SØ3* [1A1]	U	SØ3 [1A1]
18	SØ2* [1A1]	V	SØ2 [1A1]
19	SØ1* [1A1]	W	SØ1 [1A1]
20	SØØ* [1A1]	X	SØØ [1A1]
21	IØ5* [1A4]	Y	IØ5 [1A4]
22	IØ4* [1A4]	Z	IØ4 [1A4]
23	IØ3* [1A4]	AA	IØ3 [1A4]
24	IØ2* [1A4]	BB	IØ2 [1A4]
25	IØ1* [1A4]	CC	IØ1 [1A4]
26	IØØ* [1A4]	DD	IØØ [1A4]
27		EE	
28		FF	

BP52	[5B4] (5B2) (6A1) (6B1) (7A3)
BP62	[5B4] (5B2) (6A2) (6B3) (6B4)
EQJPS	[6A3] (6A1) (6B3) (7A1)
LDPR	[7B1] (6A1) (6B1) (6B3)
MFST2	[5A4] (5A1) (6B1) (6B4)
MFØ	[6A2] (6B2) (6B3) (7B3)
MF1	[6A3] (6B2) (6B3) (7A3)
RGCLK	[7A1] (6A4) (6B1) (7A4)
TWJJS	[7A3] (6A2) (6B2) (6B3)

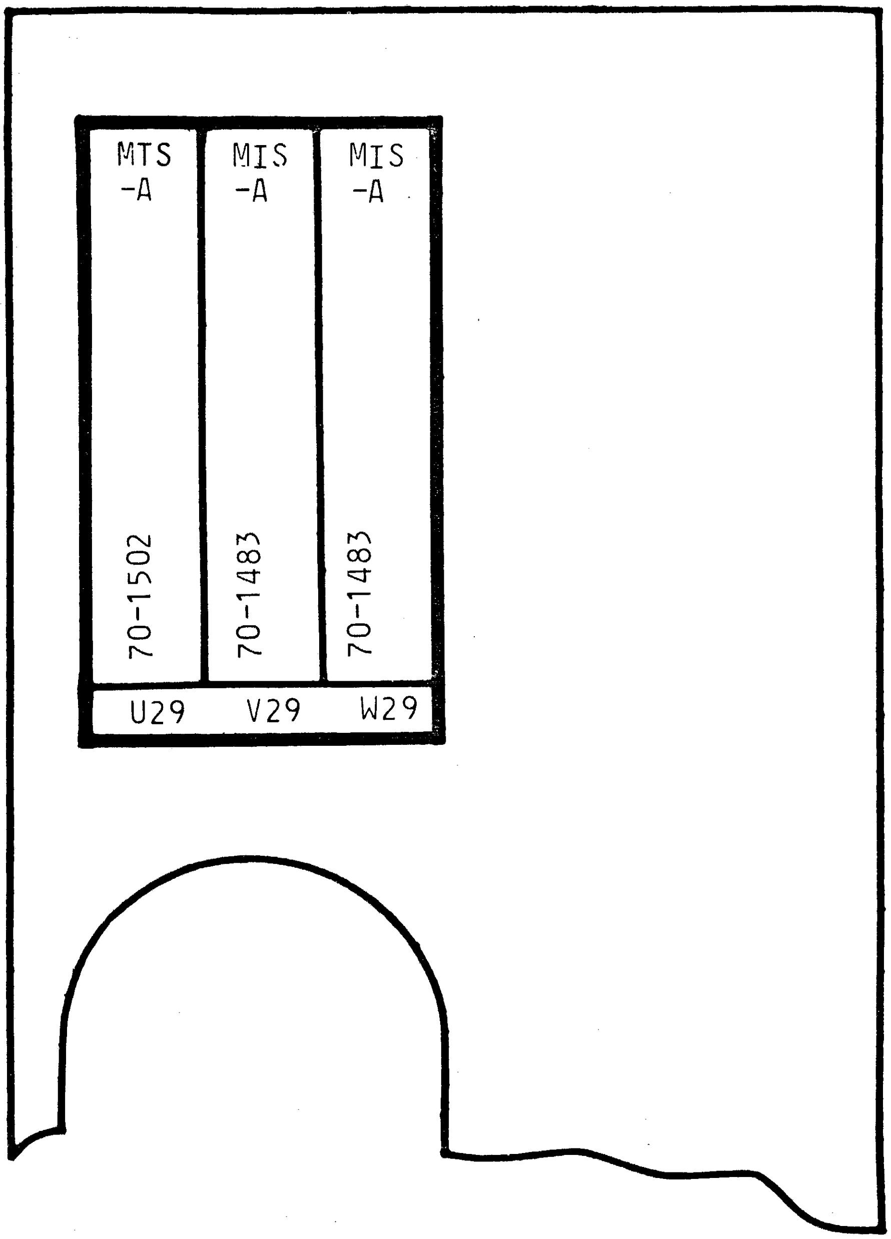
ECN	DATE
1268	4-17-72
1329	3-22-72

REVISIONS

LB4-0097-02 (SH 8)

Figure 7-4. ND812 8K Memory, Connector Pin/Signal Designation/Location (Sheet 8 of 10)

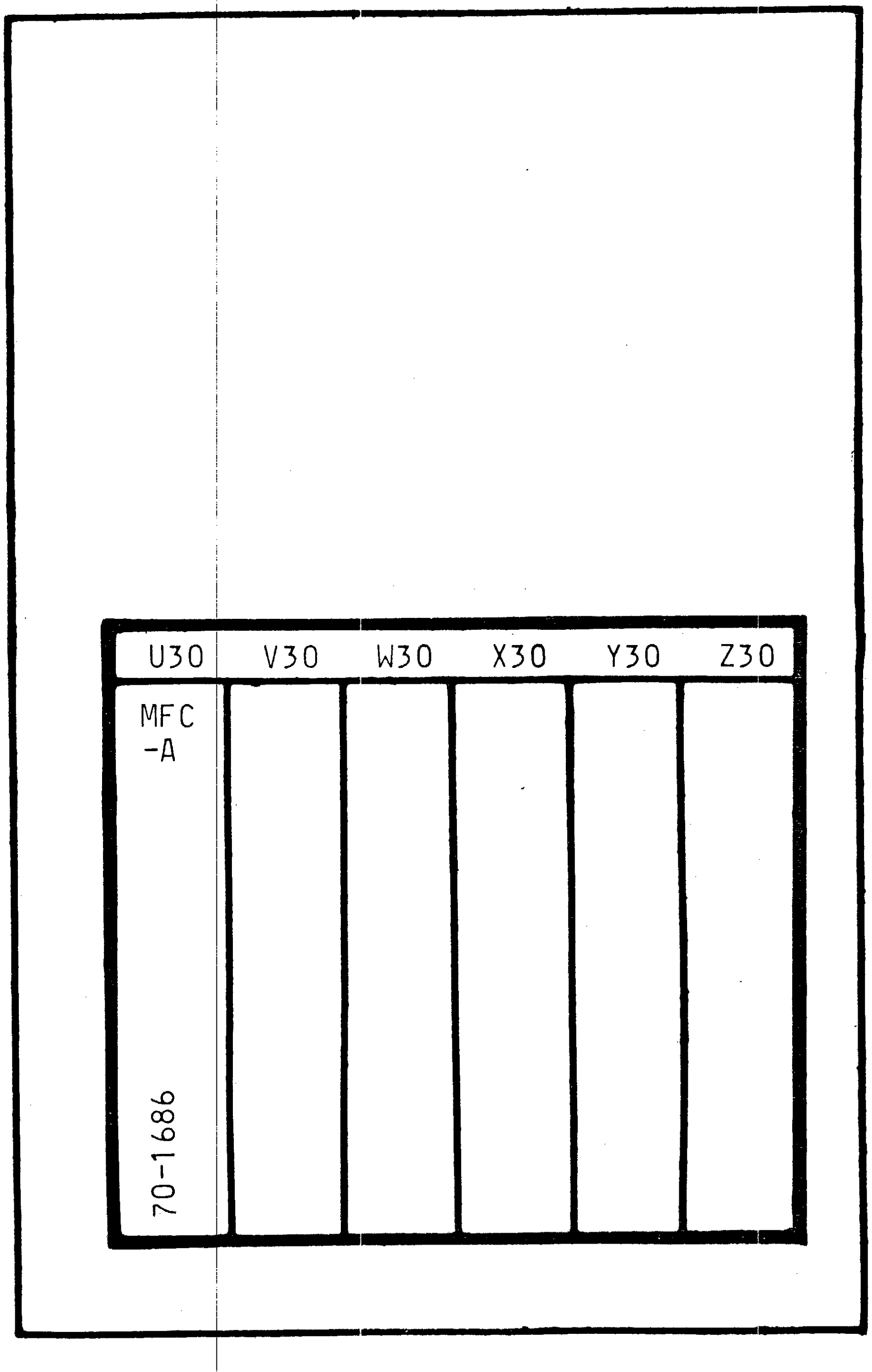
FRONT VIEW



L84-0097-00 (SH 9)

Figure 7-4. ND812 8K Memory, Loading Diagram (Sheet 9 of 10)

FRONT VIEW



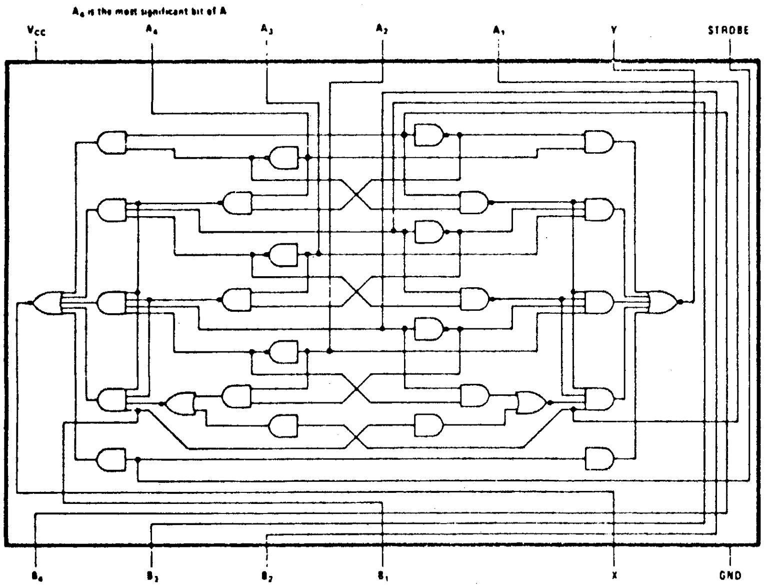
U30	V30	W30	X30	Y30	Z30
MFC -A 70-1686					

0097-00 (SH 10)

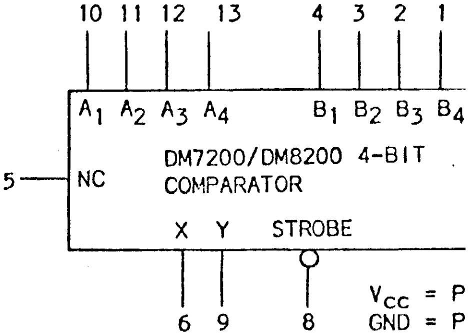
Figure 7-4. ND812 8K Memory, Loading Diagram (Sheet 10 of 10)

DM7200/DM8200 FOUR-BIT COMPARATOR

LOGIC DIAGRAM



LOGIC SYMBOL



FUNCTIONAL DESCRIPTION

The DM7200/DM8200 MSI is a monolithic TTL (Transistor-Transistor Logic) circuit which compares the numerical values of two four-bit binary numbers. Providing STROBE input is logic Ø, an X or Y output is produced as shown below in the truth table if "A is greater than B", "A is less than B", or if "A is equal to B". A logic 1 level at STROBE input forces both X and Y outputs to logic Ø regardless of inputs at A and B. Also, numerical comparisons of words longer than four bits may be made by using additional DM7200/DM8200 devices.

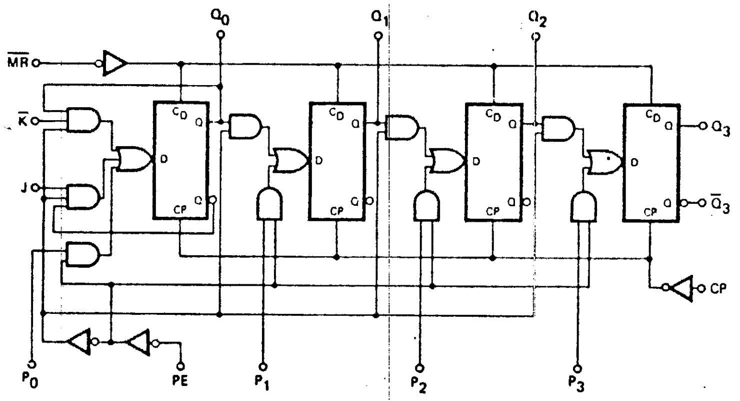
TRUTH TABLE

Input				Output	
Number $A_4A_3A_2A_1$		Number $B_4B_3B_2B_1$	Strobe	X	Y
A	>	B	0	1	0
A	<	B	0	0	1
A	=	B	0	1	1
A	⋛	B	1	0	0

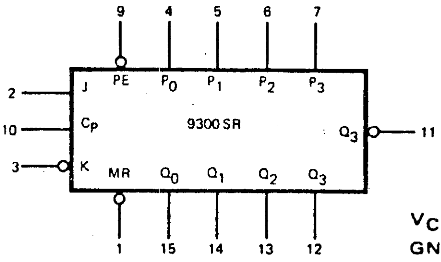
Figure 7-5. 8200 MSI Logic Diagram/Functional Description

9300 FOUR BIT SHIFT REGISTER

LOGIC DIAGRAM



LOGIC SYMBOL



V_{CC} = PIN 16
GND = PIN 8

FUNCTIONAL DESCRIPTION

The 9300 MSI is a synchronous Four-bit shift register which performs functions such as serial or parallel shifting; counting, storage, or serial code conversion. Synchronous shift occurs after a LOW to HIGH transition of clock pulse (CP) input. An overriding asynchronous Master Reset (MR) LOW input allows setting of the four outputs Q_0, Q_1, Q_2, & Q_3 to LOW regardless of their previous states and independent of the state of the other inputs. The parallel enable (PE) input selects the operating mode. With the parallel enable (PE) input HIGH, the P_0, P_1, P_2, and P_3 inputs are inhibited and the device acts as a serial shift register with data entered via the J and K inputs. Also, tying J and K together allows D-type entry. With the parallel enable (PE) input LOW, serial operation is inhibited, and P_0, P_1, P_2, and P_3 inputs are enabled. The inputs are shifted in parallel to the corresponding output synchronous with a LOW to HIGH transition of the clock pulse (CP) input.

Serial Shifting

With parallel enable (PE) input HIGH, data is entered via the J and K inputs to the first bit position Q_0 synchronous with clock pulse (CP) LOW to high transition. Data is shifted on consecutive clock pulses to Q_1, Q_2, Q_3, and then out of register to next stage via Q_3 (Refer to truth table I). With the J and K inputs tied together, D-type entry is obtained (Refer to truth table II).

Parallel Shifting

With the parallel enable (PE) input LOW, the register operates as four synchronous clocked D-type flip-flops (Refer to table III). The four D flip-flop inputs P_0, P_1, P_2, and P_3 are shifted to the corresponding Q_0, Q_1, Q_2, Q_3 outputs synchronous with a LOW to HIGH transition of the clock pulse (CP) input. A complementary output $\overline{Q_3}$ of bit four is also provided.

TABLE I
Serial Shift (Parallel Enable-HIGH) K Input-Active LOW

J	$\overline{K}$	$Q_0(t_{n+1})$
L	L	L
L	H	$Q_0(t_n)$ no change
H	L	$Q_0(t_n)$ toggle
H	H	H

TABLE II
Serial Shift (Parallel Enable PE-LOW)

J&$\overline{K}$ Connected	$Q_0(t_{n+1})$
L	L
H	H

TABLE III
Parallel Shift (Parallel Enable-LOW)

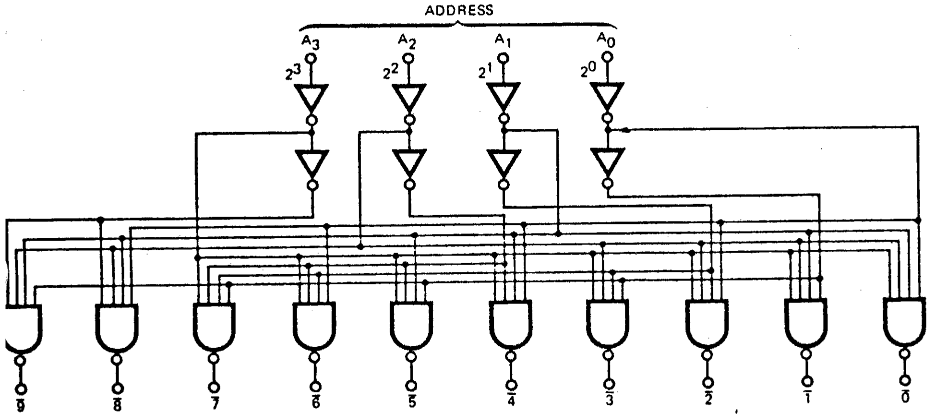
D-Input (P_0, P_1, P_2 or P_3)	Output Q (t_{n+1}) (Q_0, Q_1, Q_2 or Q_3)
L	L
H	H

t_{n+1} indicates state after next clock pulse

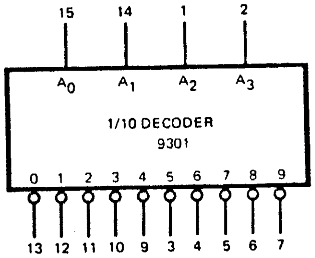
Figure 7-6. 9300 MSI Logic Diagram/Functional Description

9301 DEMULTIPLEXER/ONE-OF-TEN DECODER

LOGIC DIAGRAM



LOGIC SYMBOL



V_{CC} = P
Gnd = F

FUNCTIONAL DESCRIPTION

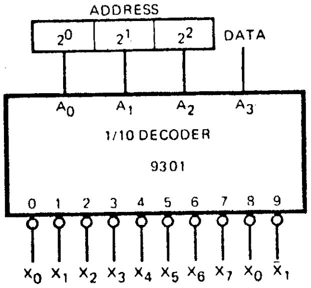
The 9301 MSI is a demultiplexer/one-of-ten decoder which accepts four active HIGH BCD weighted inputs and provides one-of-ten virtually exclusive active LOW outputs (refer to Table I). When a binary weighted code greater than nine is applied to A_0, A_1, A_2, and A_3, the outputs $\bar{0}$ through $\bar{9}$ will reside HIGH. The inputs A_0, A_1, A_2, and A_3 are weighted 2^0, 2^1, 2^2, and 2^3 respectively. The 9301 MSI can also function as a one-of-eight decoder by using the A_3 input as an active low enable. Eight channel demultiplexing results when data is applied to the A_3 input and the desired output is addressed by A_0, A_1, and A_2 (refer to Table II).

TABLE I
ONE-OF-TEN DECODE

INPUTS				OUTPUTS									
A_0	A_1	A_2	A_3	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{8}$	$\bar{9}$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
H	L	L	L	H	L	H	H	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H	H	H
H	H	L	L	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	H	H	H	L	H	H	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
H	H	H	L	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
L	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = HIGH Logic Level
L = LOW Logic Level

TABLE II
DIGITAL DEMULTIPLEXER



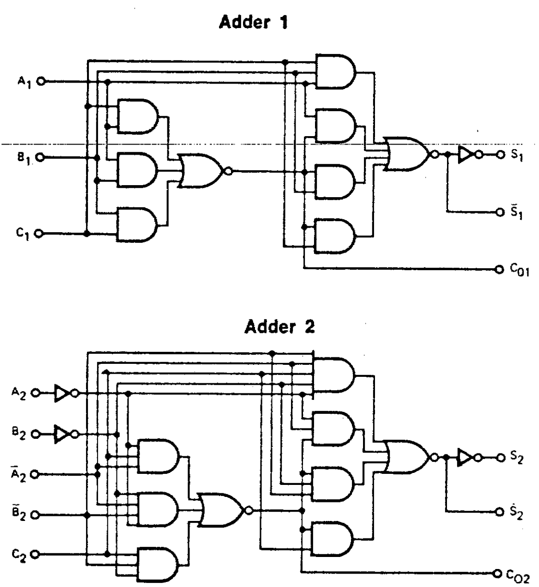
ADDRESS			OUTPUT LINE
A_0	A_1	A_2	
0	0	0	0
1	0	0	1
0	1	0	2
1	1	0	3
0	0	1	4
1	0	1	5
0	1	1	6
1	1	1	7

Data may be routed from a source to any of eight (0-7) outputs by addressing that output. The seven non-addressed outputs remain high.

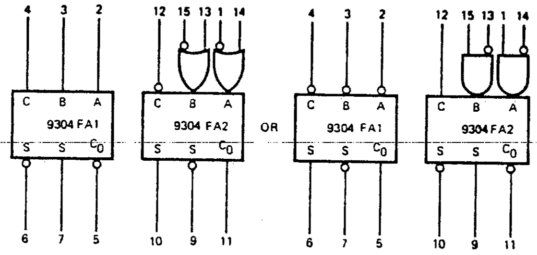
Figure 7-7. 9301 MSI Logic Diagram/Functional Description

9304 DUAL FULL ADDER

LOGIC DIAGRAM



LOGIC SYMBOLS



V_{CC} = PIN 16
GND = PIN 8

TRUTH TABLES

Adder 1

Inputs			Outputs		
C_1	B_1	A_1	$\overline{C}_{O1}$	$\overline{S}_1$	S_1
L	L	L	H	H	L
L	L	H	H	L	H
L	H	L	H	L	H
L	H	H	L	H	L
H	L	L	H	L	H
H	L	H	L	H	L
H	H	L	L	H	L
H	H	H	L	L	H

Adder 2

Inputs					Outputs		
$\overline{C}_2$	B_2	A_2	$\overline{B}_2$	$\overline{A}_2$	C_{O2}	S_2	$\overline{S}_2$
L	L	L	L	L	H	H	L
L	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H
L	L	L	H	H	L	H	L
L	L	H	L	L	H	H	L
L	L	H	L	H	H	H	L
L	L	H	H	L	H	L	H
L	L	H	H	H	H	L	H
L	H	L	L	L	H	H	L
L	H	L	L	H	H	L	H
L	H	L	H	L	H	H	L
L	H	L	H	H	H	L	H
L	H	H	L	L	H	H	L
L	H	H	L	H	H	H	L
L	H	H	H	L	H	H	L
L	H	H	H	H	H	H	L
H	L	L	L	L	H	L	H
H	L	L	L	H	L	H	L
H	L	L	H	L	L	H	L
H	L	L	H	H	L	L	H
H	L	H	L	L	H	L	H
H	L	H	L	H	H	L	H
H	L	H	H	L	L	H	L
H	L	H	H	H	L	H	L
H	H	L	L	L	H	L	H
H	H	L	L	H	L	H	L
H	H	L	H	L	H	L	H
H	H	L	H	H	L	H	L
H	H	H	L	L	H	L	H
H	H	H	L	H	H	L	H
H	H	H	H	L	H	L	H
H	H	H	H	H	H	L	H

H = HIGH Voltage Level
L = LOW Voltage Level

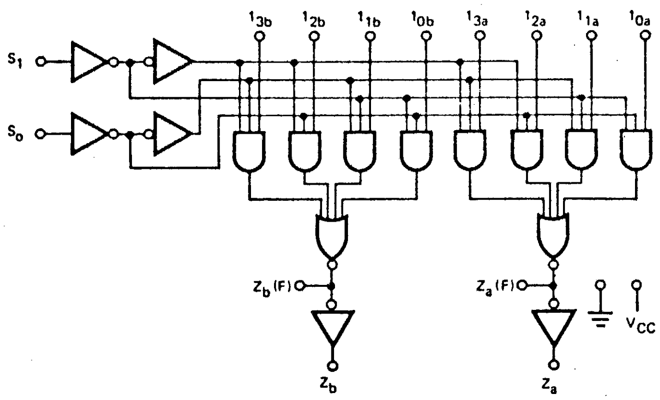
FUNCTIONAL DESCRIPTION

The 9304 MSI consists of two separate binary full adders which perform functions such as multiple bit parallel add/serial carry addition, code conversion, parity generation and checking, and majority gating. Each adder has a sum and its complement and carry as outputs. Adder 2 above has provisions for either active HIGH or active LOW A and B operand inputs. The adders produce a LOW carry and both LOW and HIGH sum with active HIGH inputs. A HIGH carry and HIGH and LOW sum are produced when active LOW inputs are used. This feature allows two representations of the logic symbols as shown above. Table I provides a truth table for adder 1, and table II provides a truth table for adder 2.

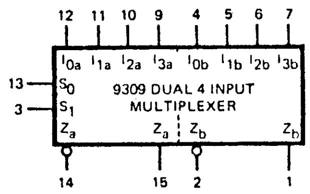
Figure 7-8. 9304 MSI Logic Diagram/Functional Description

9309 DUAL FOUR-INPUT MULTIPLEXER

LOGIC DIAGRAM



LOGIC SYMBOL



12 11 10 9 4 5 6 7
I0a I1a I2a I3a I0b I1b I2b I3b
13
S0
3
S1
9309 DUAL 4 INPUT MULTIPLEXER
Za
Za
Zb
Zb
14 15 2 1

Vcc = Pin 16
Gnd. = Pin 8

FUNCTIONAL DESCRIPTION

The 9309MSI is a dual four-input multiplexer with common input select logic (S_0 and S_1) which allows two bits of input data (from two 4-bit sources) to be switched in parallel to the appropriate outputs ($Z_a/\overline{Z_a}$ and $Z_b/\overline{Z_b}$). Both TRUE and FALSE (F) outputs are provided. Select lines S_0 and S_1 are weighted 2^0 and 2^1 respectively, which allows selection of any input 0_a, 1_a, 2_a, or 3_a; and 0_b, 1_b, 2_b, or 3_b to be switched to the Z_a output and Z_b output respectively. The following truth table illustrates both four-bit multiplexer outputs for each selected data input.

TRUTH TABLE

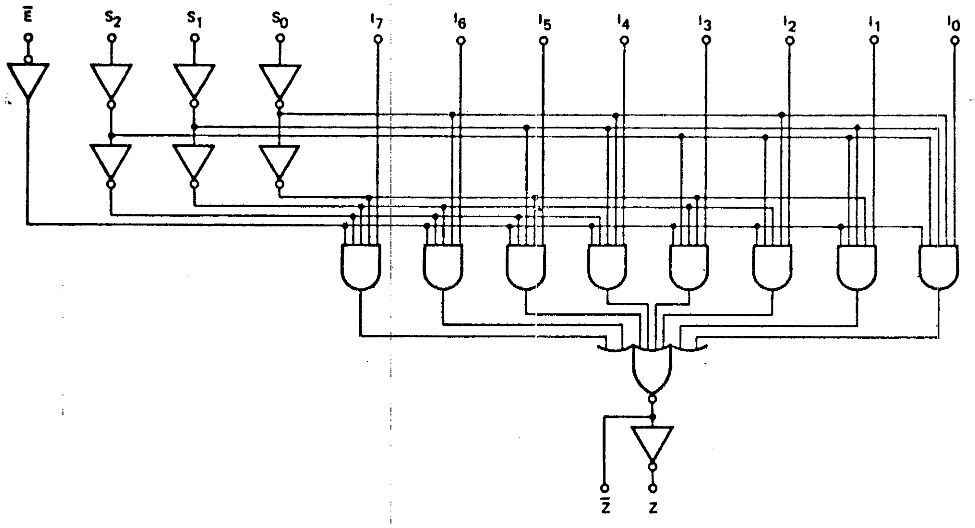
Select Inputs		Data Inputs				Outputs	
S_0	S_1	I_{0a}	I_{1a}	I_{2a}	I_{3a}	Z_a	$\overline{Z}_a$(F)
L	L	L	X	X	X	L	H
L	L	H	X	X	X	H	L
H	L	X	L	X	X	L	H
H	L	X	H	X	X	H	L
L	H	X	X	L	X	L	H
L	H	X	X	H	X	H	L
H	H	X	X	X	L	L	H
H	H	X	X	X	H	H	L
S_0	S_1	I_{0b}	I_{1b}	I_{2b}	I_{3b}	Z_b	$\overline{Z}_b$(F)
L	L	L	X	X	X	L	H
L	L	H	X	X	X	H	L
H	L	X	L	X	X	L	H
H	L	X	H	X	X	H	L
L	H	X	X	L	X	L	H
L	H	X	X	H	X	H	L
H	H	X	X	X	L	L	H
H	H	X	X	X	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

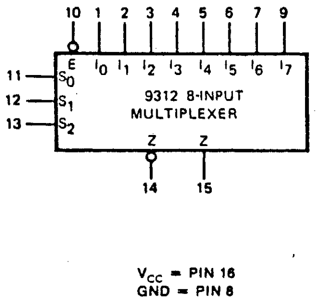
Figure 7-9. 9309 MSI Logic Diagram/Functional Description

9312 EIGHT-INPUT MULTIPLEXER

LOGIC DIAGRAM



LOGIC SYMBOL



FUNCTIONAL DESCRIPTION

The 9312 MSI is an eight-bit multiplexer which can select one bit of data from up to eight sources. Both TRUE and FALSE outputs are provided. The enable output (E) must be low in order to select a particular input (I_0 through I_7) to the output (Z). When the enable input (E) is set high, output Z is low, and $\overline{Z}$ is high regardless of all input conditions.

Select lines S_0, S_1, and S_2 are weighted 2^0, 2^1, and 2^2 respectively, which allows selection of any input (I_0 through I_7) to be switched to the Z output when the enable (E) input is low. The following truth table illustrates the multiplexer output state for each selected data input.

TRUTH TABLE

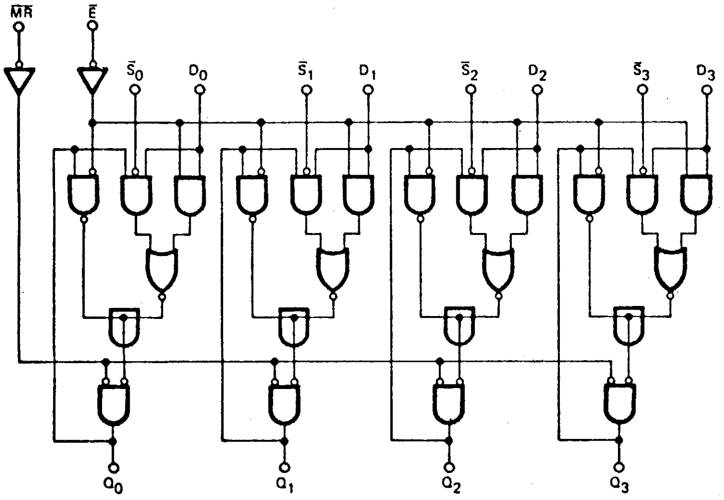
Inputs												Outputs	
$\overline{E}$	S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	$\overline{Z}$(F)	Z
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	L	X	X	X	X	H	X	X	X	L	H
L	H	L	H	X	X	X	X	X	L	X	X	H	L
L	H	L	H	X	X	X	X	X	H	X	X	L	H
L	H	H	L	X	X	X	X	X	X	L	X	H	L
L	H	H	L	X	X	X	X	X	X	H	X	L	H
L	H	H	H	X	X	X	X	X	X	X	L	H	L
L	H	H	H	X	X	X	X	X	X	X	H	L	H

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't care

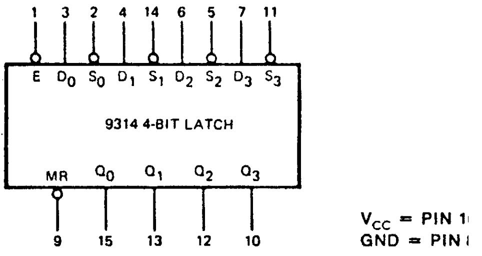
Figure 7-10. 9312 MSI Logic Diagram/Functional Description

9314 FOUR-BIT LATCH

LOGIC DIAGRAM



LOGIC SYMBOL



FUNCTIONAL DESCRIPTION

The 9314 MSI is a four-bit latch with a common active LOW enable and overriding active LOW master reset. The 9314 MSI provides both D-type Latch operation or SET/RESET Latch operation. When the common enable ($\overline{E}$) is held HIGH, data present in the latches is stored at the Q_0 through Q_3 outputs, and the state of the latch is not affected by the $\overline{S}$ and D inputs. When the master reset ($\overline{MR}$) goes LOW, all latch outputs (Q_0 through Q_3) are forced LOW regardless of all other input conditions.

D-Type Latch Operation - Each of the four latches function as individual D-type Latches when the appropriate inputs (S_0 through S_3) are held LOW. When the common enable ($\overline{E}$) goes LOW, the latch outputs (S_0 through S_3) follow the D inputs (D_0 through D_3). The data at the Q output is stored when the common enable ($\overline{E}$) goes HIGH.

SET/RESET Latch Operation - When the common enable ($\overline{E}$) goes low, any individual latch is reset by a LOW on the appropriate D input, and is set by a LOW on the appropriate $\overline{S}$ input if the D input is HIGH. When both $\overline{S}$ and D inputs are LOW, the D input will dominate and the latch will be reset. When the common enable ($\overline{E}$) goes HIGH, the latch will remain in the last state prior to the LOW to HIGH transition of ($\overline{E}$).

The following truth table illustrates individual latch operations in the D mode and the R/S Mode.

TRUTH TABLE

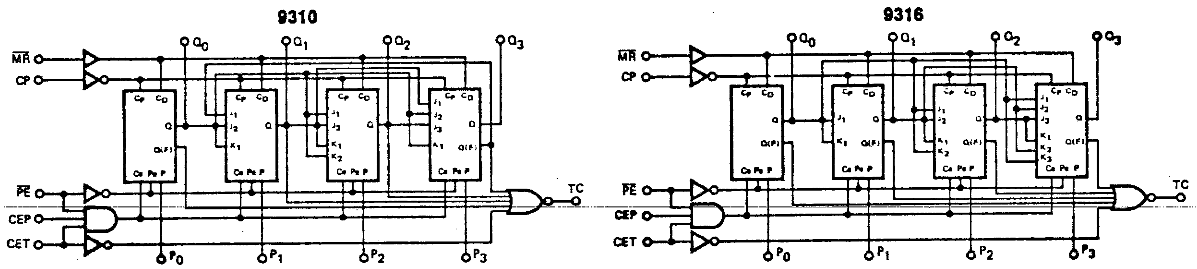
$\overline{MR}$	$\overline{E}$	D	$\overline{S}$	$Q_{(tn+1)}$	Operation
L	X	X	X	L	Reset
H	L	L	L	L	D Mode
H	L	H	L	H	
H	H	X	X	$Q_{(tn)}$	
H	L	L	L	L	R/S Mode
H	L	H	L	H	
H	L	L	H	L	
H	L	H	H	$Q_{(tn)}$	
H	H	X	X	$Q_{(tn)}$	

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

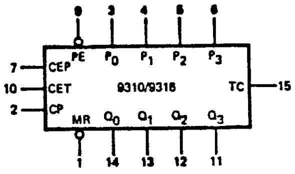
Figure 7-11. 9314 MSI Logic Diagram/Functional Description

9310/9316 BCD DECADE COUNTER/FOUR-BIT BINARY COUNTER

LOGIC DIAGRAM



LOGIC SYMBOL



V_{CC} = PIN 16
GND = PIN 8

FUNCTIONAL DESCRIPTION

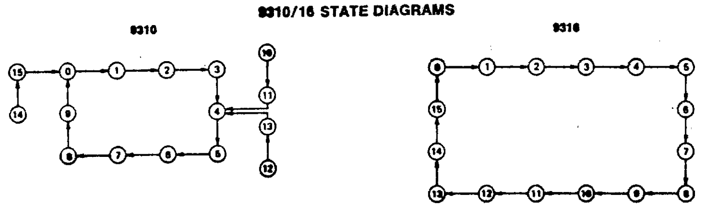
The 9310 and 9316 MSI counters consist of four master-slave JK flip-flops which are driven synchronously by a buffered clock pulse (CP) input. The 9310 is configured as a BCD decade counter and the 9316 is configured as a binary counter. During the LOW to HIGH transition of clock pulse (CP), the master flip-flop stage is inhibited from further change. Following master flip-flop lockout, data is transferred from the master to slave flip-flop outputs (Q_0, Q_1, Q_2, and Q_3). During the period that clock pulse (CP) remains HIGH, the master flip-flop is inhibited from data entry and the master slave data transfer path remains established. During the high to low transition of the clock pulse (CP), the slave is inhibited from further change, followed by enabling of the masters for acceptance of data at the parallel inputs (P_0, P_1, P_2, and P_3). Three control inputs, parallel enable ($\overline{PE}$), count enable parallel (CEP) and count enable trickle (CET), select the mode of operation as shown in table I. When conditions for counting are satisfied, the rising edge of the clock pulse will change the counters to the next state of the count sequence (refer to figure 1). The count mode is enabled when the CEP, CET, and $\overline{PE}$ inputs are HIGH. The clock pulse (CP) input must be HIGH during the HIGH to LOW transition of CEP and CET and during the LOW to HIGH transition of $\overline{PE}$ for correct logic operation.

The 9310 and 9316 counters can be synchronously preset by the four parallel inputs (P_0, P_1, P_2, and P_3) when the parallel enable input ($\overline{PE}$) is LOW. When the parallel enable ($\overline{PE}$) and clock pulse (CP) are LOW, each master of the flip-flops is connected to the appropriate parallel input (P_0, P_1, P_2, and P_3), and the slaves (Q_0, Q_1, Q_2, and Q_3) remain in their previous state. When the clock pulse (CP) goes HIGH, the masters are inhibited from data entry, and the master slave data transfer path remains established. The parallel enable input overrides both count enable inputs, presetting the counter when LOW.

The terminal count (TC) output is active HIGH when the counter is at terminal count (state 9 for the 9310, and state 15 for the 9316), and count enable trickle (CET) is HIGH (refer to table II). The asynchronous master reset ($\overline{MR}$ LOW) overrides all other inputs resetting the outputs (Q_0, Q_1, Q_2, and Q_3) LOW.

FIGURE I

9310/16 STATE DIAGRAMS



The state diagrams show the count sequence after the counters are preset to any one of the sixteen possible states.

TABLE I

9310 AND 9316
MODE SELECTION

$\overline{PE}$	CEP	CET	MODE
L	L	L	Preset
L	L	H	Preset
L	H	L	Preset
L	H	H	Preset
H	L	L	No Change
H	L	H	No Change
H	H	L	No Change
H	H	H	Count

($\overline{MR}$ = HIGH)

TABLE II

TERMINAL COUNT GENERATION

CET	9310 $(Q_0 \bullet \overline{Q}_1 \bullet \overline{Q}_2 \bullet Q_3)$	9316 $(Q_0 \bullet Q_1 \bullet Q_2 \bullet Q_3)$	TC
L	L	L	L
L	H	H	L
H	L	L	L
H	H	H	H

$TC = CET \bullet Q_0 \bullet \overline{Q}_1 \bullet \overline{Q}_2 \bullet Q_3$ (9310)
$TC = CET \bullet Q_0 \bullet Q_1 \bullet Q_2 \bullet Q_3$ (9316)

POSITIVE LOGIC = H = HIGH Voltage Level
L = LOW Voltage Level

Figure 7-12. 9316 MSI Logic Diagram/Functional Description