

PRODUCT DESCRIPTION

PRELIMINARY

PARALLEL PROCESSING SYSTEM

(PPS-8)

MICROCOMPUTER

INTRODUCTION AND DESCRIPTION

PARALLEL PROCESSING SYSTEM (PPS-8)



Rockwell International

NOTICE

Information provided in this Product Description is for reference purposes only, and is subject to change without notice.

For specific detail information on these devices or for information on any of our other advanced microelectronic devices, please contact the nearest Rockwell International Microelectronic Device Division office.

TABLE OF CONTENTS

Section 1 – INTRODUCTION	1-1
Section 2 – DESCRIPTION	2-1
General	2-1
System Characteristics	2-1
Throughput	2-1
Instruction Repertoire	2-2
Instruction Memory (ROM)	2-2
Read/Write Memory (RAM)	2-2
Input/Output	2-2
Physical/Electrical	2-3
Support Equipment	2-3
Functional Description	2-4
Central Processor Unit (CPU)	2-4
Read Only Memory Device (ROM)	2-7
Random Access Memory Device (RAM)	2-8
Clock Generator	2-9
General Purpose Input/Output (GPI/O)	2-10
Parallel Data Controller (PDC)	2-11
Direct Memory Access Controller (DMAC)	2-21
Special Purpose Input/Output Devices	2-27
Section 3 – PRINCIPLES OF OPERATION	3-1
Basic System Operation	3-1
System Timing	3-1
Logic Levels	3-1
Multiplex System Data Transfer	3-1
Instruction Repertoire	3-3
Notes for Instruction Descriptions	3-13
Data Stack	3-16
Program Addressing	3-17
Branching	3-17
Subroutine Linkage	3-18
Operand Addressing	3-20
RAM Operands	3-20
ROM Operands	3-22
Data Pools	3-24
Command Pool	3-24
Literal Pool	3-26
Subroutine Entry Pool	3-27
Pool Utilization	3-27
Interrupts	3-28
Interrupt Processing	3-29
Input/Output	3-30



Rockwell International

Microelectronic Device Division

ROCKWELL INTERNATIONAL

c/o ROCKWELL INTERNATIONAL OVERSEAS CORPORATION

(U.S.A. – WEST COAST)

P. O. Box 3669
3310 Miraloma Avenue
Anaheim, California 92803

Phone: (714) 632-3698
TWX: (910) 591-1179, Telex: Via TWX

(U.S.A. – EAST COAST)

9 Kim Avenue
Smithtown, New York 11787

Phone: (516) 979-0183

(EUROPE)

D-6374 Steinbach/Taunus
Industriestrasse 8
Germany

Phone: (06171) 7755
TELEX: 410758

(FAR EAST)

Ichiban-Cho Central Building
22-1, Ichiban-Cho
Chiyoda-Ku, Tokyo, 102 Japan

Phone: 265 8808/8809
TELEX: J22198

Section 1. INTRODUCTION

The Rockwell 8-bit Parallel Processing System (PPS-8) is an advanced design, modular, microcomputer system which uses a unique 4-phase clock timing system for logic functions and control. Proven P-channel MOS technology is used to implement the PPS-8 system byte (8-bit) oriented architecture. Rockwell MOS technology and the unique design of the PPS-8 offers superior performance for less cost over a wide range of applications. Some of these applications include point-of-sale equipment, data entry terminals, and peripheral/process control. The wide range of applications and performance is achieved by high system throughput, efficient character-oriented instructions, flexible input/output capabilities, and system modularity/expandability.

The standard PPS-8 system is shown in Figure 1-1. The standard PPS-8 system consists of a Central Processor Unit (CPU) device, Random Access Memory(s) (RAM) device, Read Only Memory(s) (ROM) device, Multiphase Clock Generator, Direct Memory Access Controller (DMAC) device, Parallel Data Controller (PDC) device and Serial Data Controller (SDC) device.

The basic system can be supplemented with special purpose and custom input/output devices as required for specific applications. For example, a 1200 baud Telecommunications Data Interface device, a printer controller device, a bus-interface device, and a keyboard/display controller device are available.

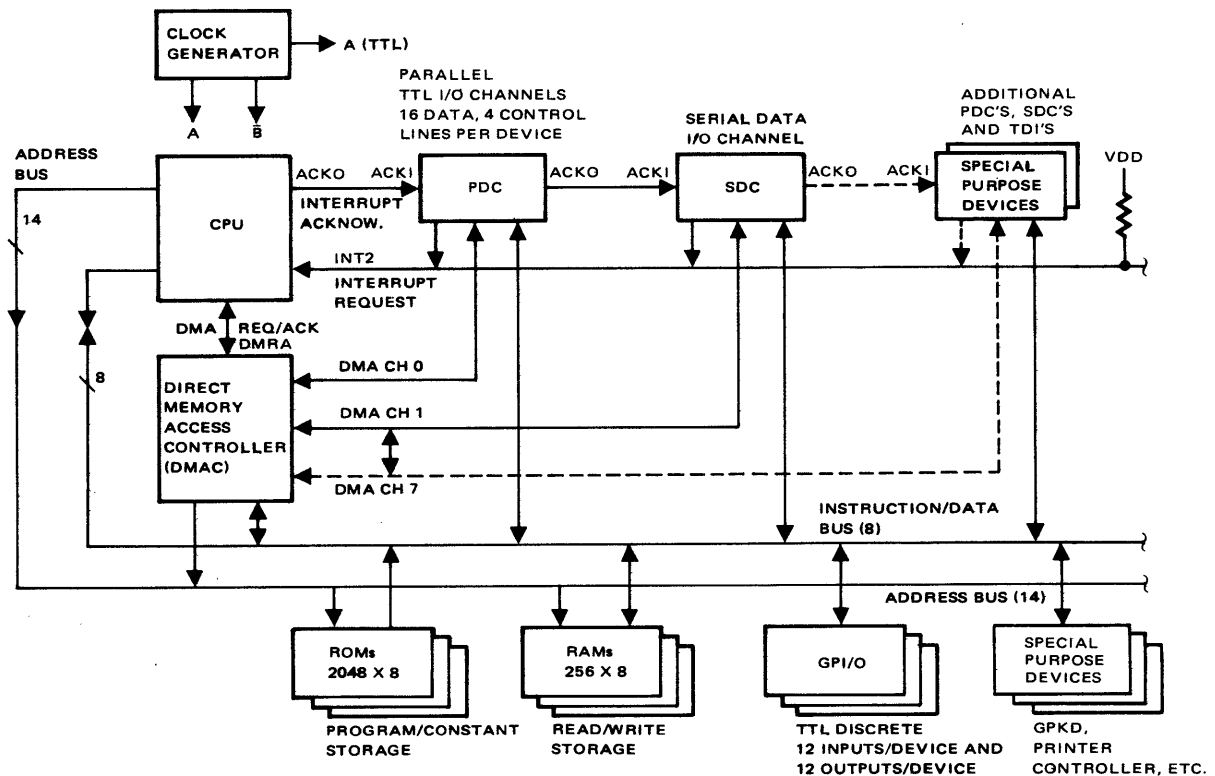


Figure 1-1. PPS-8 Microcomputer System

Section 2. DESCRIPTION

GENERAL

The various MOS devices used to make up the PPS-8 are all packaged in identical standard 42-lead plastic packages with the exception of the Clock Generator which is in a TO 100 package. The overall dimensions of the body of the plastic package, exclusive of the leads, are 0.640 inches in width, 1.060 inches in length, and 0.15 inches in thickness. This standardization in packaging provides optimum flexibility when designing and packaging a specific PPS-8 System.

A functional description of each of the below listed devices is provided in this section.

PPS-8 BASIC DEVICES

Part Number

Central Processor Unit (CPU)	10806
Read Only Memory (ROM)	A52--
Clock Generator	10706
General Purpose Input/Output (GPI/O)	10696
Parallel Data Controller (PDC)	10453
Direct Memory Access Controller (DMAC)	10817
Random Access Memory (RAM)	10809

SPECIAL PURPOSE DEVICES

Part Number

Printer Controller Circuit	10789
Telecommunication Data Interface (TDI)	10371
General Purpose Keyboard and Display (GPKD)	10788
Bus Interface (B/I)	10733
Serial Data Controller (SDC)	10930

SYSTEM CHARACTERISTICS

The following paragraphs provide a summarization of the characteristics and capabilities of the basic PPS-8 system.

THROUGHPUT

- Instruction Access - 2 μ s
- Operand Access - 2 μ s
- Complete Instruction Cycle - 4 μ s
- Decimal Addition - 12 μ s/digit
- Decimal Subtraction - 12 μ s/digit
- Block Data Moving - 12 μ s/byte
- Table Search - 12 μ s/byte
- Digit (4-bit) String Shifting - 8 μ s/digit

INSTRUCTION REPERTOIRE

- Over 90 Instructions
- Digit (4-bit) and Byte (8-bit) Manipulation
- Decimal and Binary Arithmetic
- Bit Setting/Resetting/Testing
- Single Byte Subroutine Call
- Data Stacking
- Common Data "Pools"

INSTRUCTION MEMORY (ROM)

- 2048 x 8 RCM devices
- Local Page Addressing to 128 Bytes
- Direct Addressing to 16,384 Bytes
- Bank Select Addressing to 32K Without External Circuitry
- Indirect, Auto-increment Addressing

READ/WRITE MEMORY (RAM)

- 256 x 8 RAM devices
- Direct Addressing to 16,384 Bytes
- Bank Select Addressing to 32K Without External Circuitry
- 3 RAM Addressing Registers
- Auto-Increment and Auto-Decrement Addressing

INPUT/OUTPUT

- Priority Interrupt System
 - Power-Fail Detect
 - Real Time Clock
 - I/O Device Service
- Direct Memory Access
 - Up to 8 Prioritized Channels
 - 250K Bytes/sec
- Program Controlled I/O
 - Byte (8-bit) or Digit (4-bit)
 - 75K Bytes/sec
- General Purpose TTL-Compatible Interface
 - Static Discretes
 - Bi-directional 8 or 16 Bit Parallel Bus(es)
 - Buffered Serial (standard communications format)
 - Full "Handshaking"
 - PPS-4 I/O Compatibility

- Special Purpose I/O Devices
 - Telecommunications Data Interface (TDI)
 - General Purpose Keyboard and Display (GPKD)
 - Printer Controller
 - Bus Interface (B/I)
 - Serial Data Controller (SDC)

PHYSICAL/ELECTRICAL

- Single -17 volt Power Supply
- Low Power 4-Phase Dynamic Logic
- 42-Pin Plastic Packages

SUPPORT EQUIPMENT

- Assembler on Tymshare, GE Information Services, and Rockwell TSO Operating System
- Interactive Simulator on Tymshare, GE Information Services, and Rockwell TSO Operating System
- Batch Assembler (Fortran)
- Batch Simulator (Fortran)
- Rockwell PPS-8 Assemulator (Assembler and Emulator)
- Pre-Packaged Evaluation Boards
 - Processor I (CPU, RAM(2), GPIO(2), Clock)
 - Processor II (CPU, RAM(2), PDC(2) DMAC, Clock)
 - RAM (RAM (8))
 - PROM
 - GPIO
 - TTL
 - ROM

FUNCTIONAL DESCRIPTION

CENTRAL PROCESSOR UNIT (CPU)

The PPS-8 Central Processor Unit, Part No. 10806, shown in Figures 2-1 and 2-2 is a complete 8-bit parallel processor implemented on a single MOS chip. The CPU uses four-phase dynamic logic for operation, and all power requirements are met from a single 17-volt power supply.

The CPU contains:

- (a) Logic necessary to receive and decode the instructions
- (b) 8-bit parallel adder-accumulator for arithmetic and logical operations
- (c) 14-bit P-Register for sequencing through the ROM program
- (d) 16-bit L-Register for subroutine linkage, RAM operand addressing, and ROM indirect addressing
- (e) Three 8-bit registers, (X, Y and Z) for RAM operand addressing
- (f) 5-bit stack pointer S for addressing a dedicated RAM area
- (g) Logic for processing a priority interrupt structure
- (h) Direct memory access (DMA) mode
- (i) Multiplexed receivers and drivers for interfacing with the 14-bit multiplexed address bus and the 8-bit bi-directional data/instruction bus.

The CPU, through time multiplexing, utilizes an 8-bit bi-directional bus to transfer instructions from ROM to CPU (and I/O) during ϕ_4 , and to transfer data between the CPU, RAMs and I/O devices during ϕ_2 .

Instruction Decode

The decode portion of the chip contains logic to decode the instructions, sense interrupt or DMA requests, and provide signals to control data transfer, arithmetic, logical, and indexing operations. Instructions are either one, two, or three bytes in length and require from one to three cycles for execution.

Accumulator Register and Arithmetic Logic Unit (ALU)

The adder is an 8-bit parallel binary adder with an internally connected carry flip-flop (C) for implementing extended precision arithmetic operations. In addition, the adder has built-in capability to facilitate packed BCD (decimal) arithmetic and manipulation of hexadecimal data. Circular shifting of the accumulator contents right and left with carry linkage is also provided. The adder, with the 8-bit Accumulator Register (A), and associated logic circuits forms the Arithmetic and Logical Unit (ALU) section of the CPU.

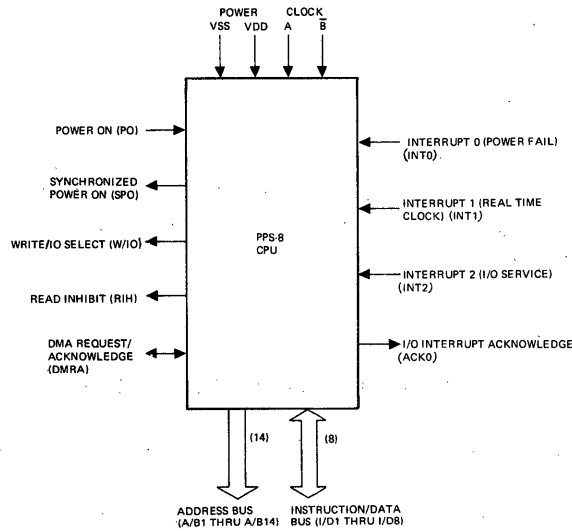


Figure 2-1. PPS-8 CPU Interface

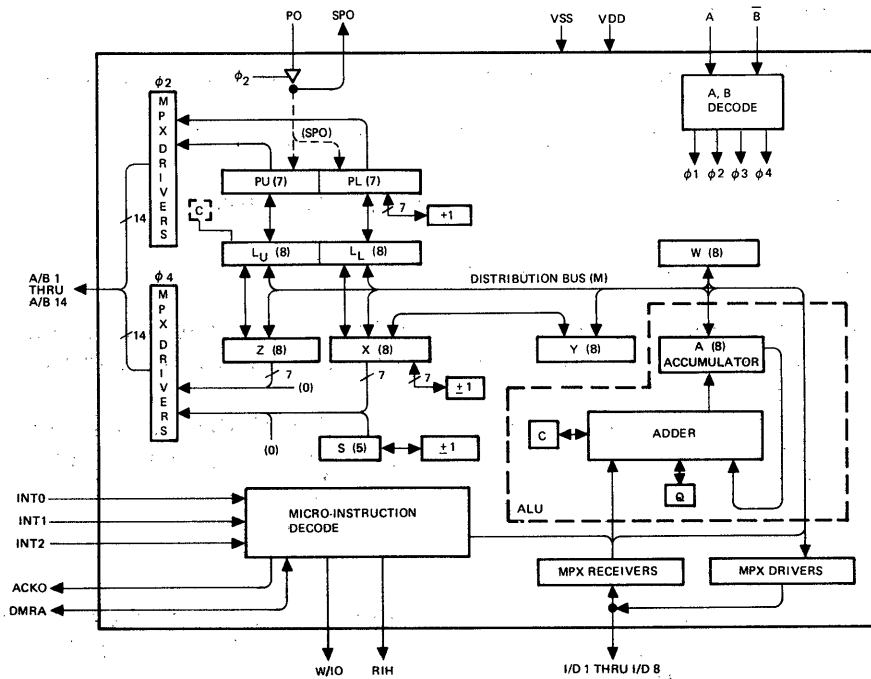


Figure 2-2. Central Processor Unit Block Diagram

In addition to its arithmetic functions, the A-Register is the primary working register in the CPU and is the central data interchange point for most data transfer operations.

P-Register (14-bits)

The P-Register contains the address of the instruction currently being executed, and automatically increments (least significant 7-bits) to fetch the next byte from instruction memory (ROM). It may be altered during the execution of Branch, Return or Skip instructions.

L-Register (16 Bits)

The L-Register is used to save the return address after a subroutine call or an interrupt. It is also used as an address register for indirect ROM operands. It can also be used as an alternate RAM address register or as a general purpose programming register.

Z-Register (8-bits)

This register holds the 7 most significant bits of the 14-bit RAM operand address or may be used as a general purpose programming register.

X-Register (8-bits)

The X-Register holds the 7 least significant bits of the 14-bit RAM operand address. The most significant bit (8th bit) is used as an upper RAM address control bit. If the upper address control bit is:

Logic 1 - the Z-Register contents are output for the most significant 7 bits of the RAM address.

Logic 0 - logic zero is output for the most significant 7 bits of the RAM address.

This register may be loaded, stored, auto-incremented, or auto-decremented under program control.

Y-Register (8-bits)

The Y-Register is used as an alternate lower RAM address register and as a "loop counter" or it may be used as a general purpose programming register.

S-Register (5-bit)

The 5-bit up-down counter register S is used as an address pointer to a 32 byte "stack" in RAM. This stack pointer is automatically incremented each time a byte is "pushed" into the stack and decremented each time a byte is "popped" from the stack.

W-Register (8-bit)

The W-register serves primarily as an internal buffer register. Additionally, it is used in conjunction with the LAL and PSHL instructions.

Power-On Reset (PO)

The Power-on input signal is used to initialize the CPU to a known starting address and state during a power-on sequence. The Power-on (PO) Signal is generated external to the CPU. The CPU receives this signal, initializes the internal logic states, and at the same time, generates a Synchronized Power-on Output (SPO) Signal which is used to initialize other circuits of the PPS-8.

READ ONLY MEMORY DEVICE (ROM)

The Read Only Memory (ROM), Part No. A52--, shown in Figure 2-3, is a 16,384 bit ROM organized in a 2048 x 8-bit configuration. It is designed for compatibility with the PPS-8 system, operating from a single 17-volt power supply and the A and \bar{B} system clocks. The access time is 1.8 μ s. It is intended to be used to store 8-bit instructions and constant program data within a PPS-8 system.

The 8-bit CPU, with its 14-bit parallel address outputs, can directly address up to 16,384 bytes. The ROM provides an additional address select input (AS5) that can be used in systems requiring more than 16,384 bytes of ROM storage. With this line, alternate banks of 16K ROM can be selected to provide up to 32K bytes.

The unique time-sharing design of the PPS address and data bus is such that information on the I/D bus is treated as 8 bits of microinstruction during clock phase 4 and RAM or I/O data during clock phase 2. The ROM is designed to accept addresses during phase 2 and supply instructions or data to the data bus during phase 4.

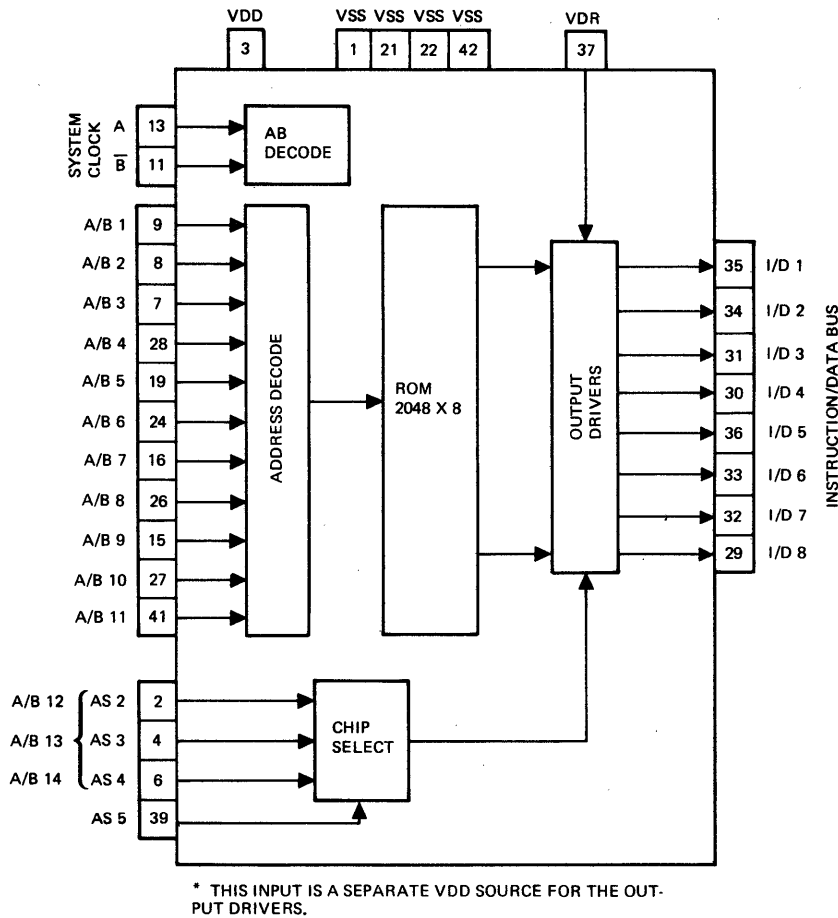


Figure 2-3. Read Only Memory (ROM) Block Diagram

RANDOM ACCESS MEMORY DEVICE (RAM)

The Random Access Memory (RAM), Part No. 10809, shown in Figure 2-4, is a 2048 bit RAM organized in 256 x 8-bit configuration. It is designed for compatibility with the PPS-8 system, operating from a single 17-volt power supply and the A and \bar{B} system clocks. It is a dynamic memory with automatic refresh logic and a 1.8 μs access time. It is intended to be used as the read/write data storage device for the PPS-8 system.

The CPU, with its 14-bit parallel address outputs can directly address up to 16,384 8-bit bytes of data. The RAM, with its eight parallel address lines and six-chip select inputs, provides for direct selection of up to 64 RAM devices. The one additional chip select input on the RAM chip can be used for memory expansion greater than 16,384 word locations. For large systems, the AS7 chip select input can be addressed by an output from an I/O circuit. For systems with RAM memory capacity of 16,384 words or smaller, the AS7 chip select input must be terminated at VSS.

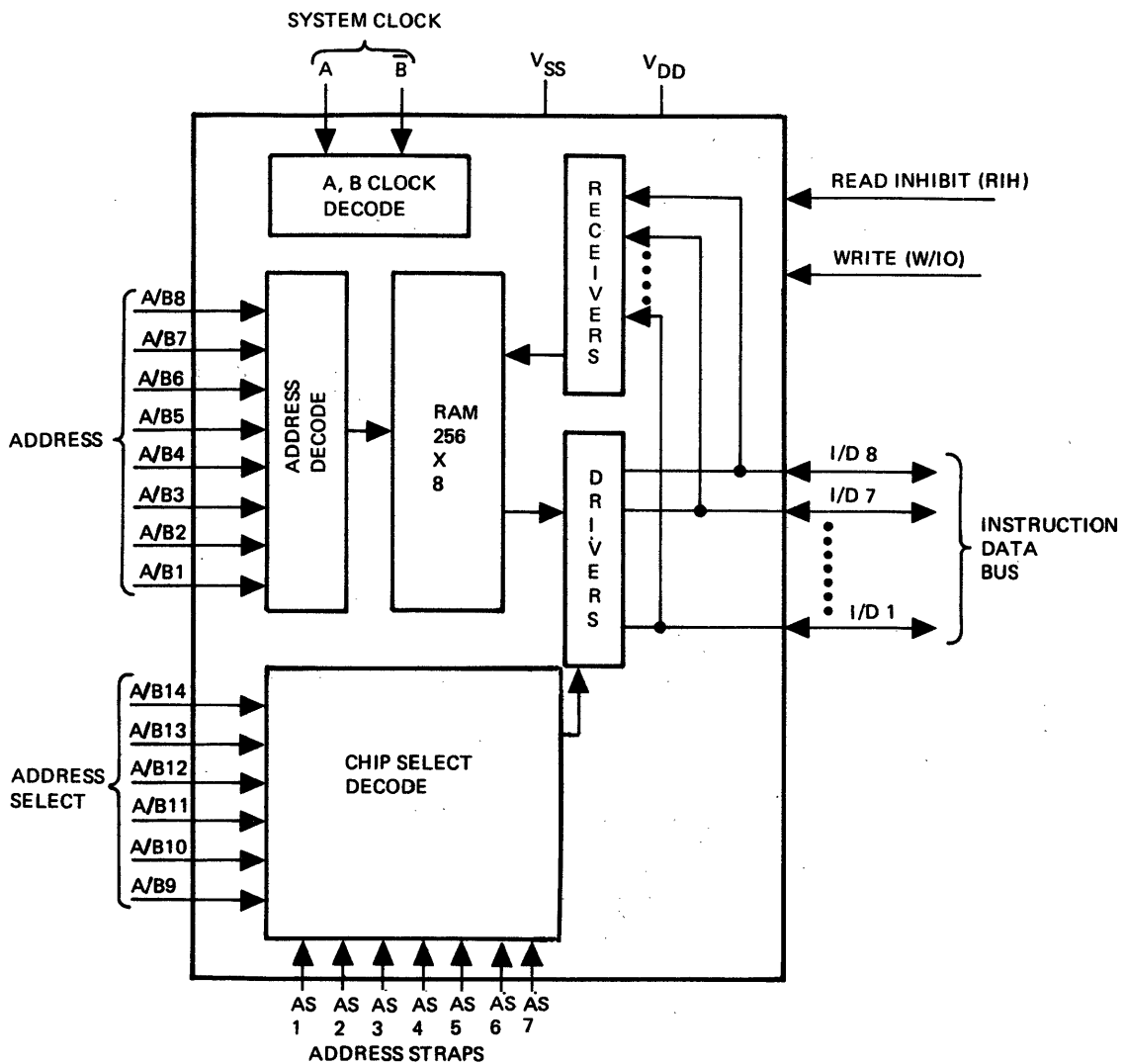


Figure 2-4. Random Access Memory Block Diagram

CLOCK GENERATOR

The Clock Generator circuit, Part No. 10706, shown in Figure 2-5, generates the "A" and "B" clock waveforms required by circuits in the PPS. The Clock Generator has an internal oscillator which is stabilized by connecting a quartz crystal to the appropriate inputs. The crystal is a 3.579545 MHz color TV crystal which is low in cost and readily available. The clock A output is a square wave and considered the primary clock. Output B is a pulse output occurring during each phase of clock A and has unique timing features required by the circuits within the PPS system. Clock A is also provided through an output which drives to ground such that TTL levels can be easily achieved for synchronizing equipment external to the PPS.

The input straps provide a countdown of the oscillator frequency equal to the number associated with the strap: i. e., S10 divides by 10, S12 input results in a countdown by 12, and S14 by 14. Thus, with a crystal frequency of approximately 3.58 MHz and input S14 terminated to V_{DD}, the clock A output frequency is 256 kHz, ($3.58 \text{ MHz} \div 14 = 256 \text{ kHz}$).

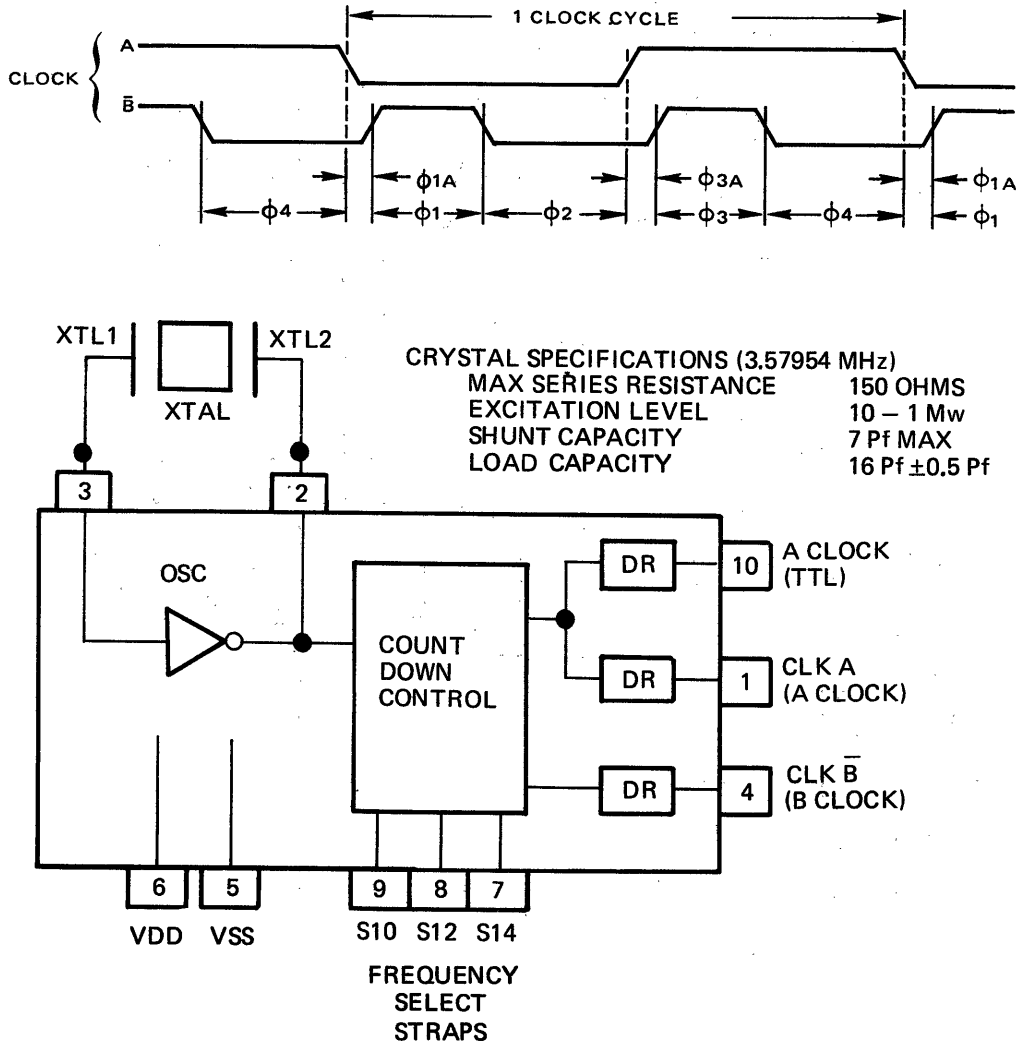


Figure 2-5. Clock Generator Block Diagram

Clocked Output (Figure 2-12)

The data lines are driven continuously from the data buffer. The CPU can load the data buffer at any time. A positive pulse is generated on CA1 each time the data buffer is loaded by the CPU or (Channel A only) via DMA. Control line CA2 is selectable as a CPU interrupt. Interrupt timing for CA2 is identical to that for Static Input (Figure 2-8).

Clocked I/O (Figure 2-13)

Data lines DA5 through DA8 are copied into the high order four bits of the data buffer on a TTL false-to-true transition of CA2 (same as for Clocked Input, Figure 2-11). Data lines DA1 through DA4 are driven continuously from the low order four bits of the data buffer. A positive pulse is generated on CA1 each time the data buffer is loaded by the CPU (same as for Clocked Output, Figure 2-12). The CPU can load or read the data buffer at any time. When the data buffer is loaded, only the low order four bits are modified. Control line CA2 is selectable as a CPU interrupt; interrupt timing for CA2 is identical to that shown for Clocked Input (Figure 2-11).

Handshake Input (Figure 2-14)

Control lines CA1 and CA2 are both initially at a TTL false level. The data lines are then copied into the data buffer on a TTL false-to-true transition of CA2. Control line CA1 is set to a TTL true level immediately after loading the input data into the data buffer. The CPU may read the data buffer at any time after CA2 has made the false-to-true transition. CA1 is set to a TTL false level when the data buffer is read by the CPU (or stored in RAM via DMA), provided that CA2 has returned to TTL false level. Control line CA2 is selectable as a CPU interrupt and/or (Channel A only) a DMA request. Interrupt timing for CA2 is shown in Figure 2-14.

Handshake Output (Figure 2-15)

The data lines are driven continuously from the data buffer. Control lines CA1 and CA2 are both initially at a TTL false level. Control line CA1 is then set to a TTL true level when the data buffer is loaded by the CPU or (Channel A only) via DMA. CA1 is set false on a TTL false-to-true transition of CA2. The data buffer may be reloaded any time after CA2 has returned false. Control line CA2 is selectable as a CPU interrupt and/or (Channel A only) a DMA request. Interrupt timing for CA2 is shown in Figure 2-15.

Direct Memory Access

Four of the modes (Clocked Input, Clocked Output, Handshake Input and Handshake Output) allow optional DMA operation on Channel A.

When the DMA option is selected for the Clocked Input mode or the Handshake Input mode, each DMA request is initiated 1.5 clock cycles after detection of the false-to-true transition of CA2 (see Figures 2-11 and 2-14).

When the DMA option is selected for the Clocked Output mode, the initial request is initiated immediately after Function Register A is loaded with the control data selecting the mode and option. The second and subsequent requests are initiated during $\phi 2$ preceding the TTL true period of CA1 (see Figure 2-12). These requests will continue to be generated with each output until the PDC is automatically informed of an end-of-block condition by the Direct Memory Access Controller.

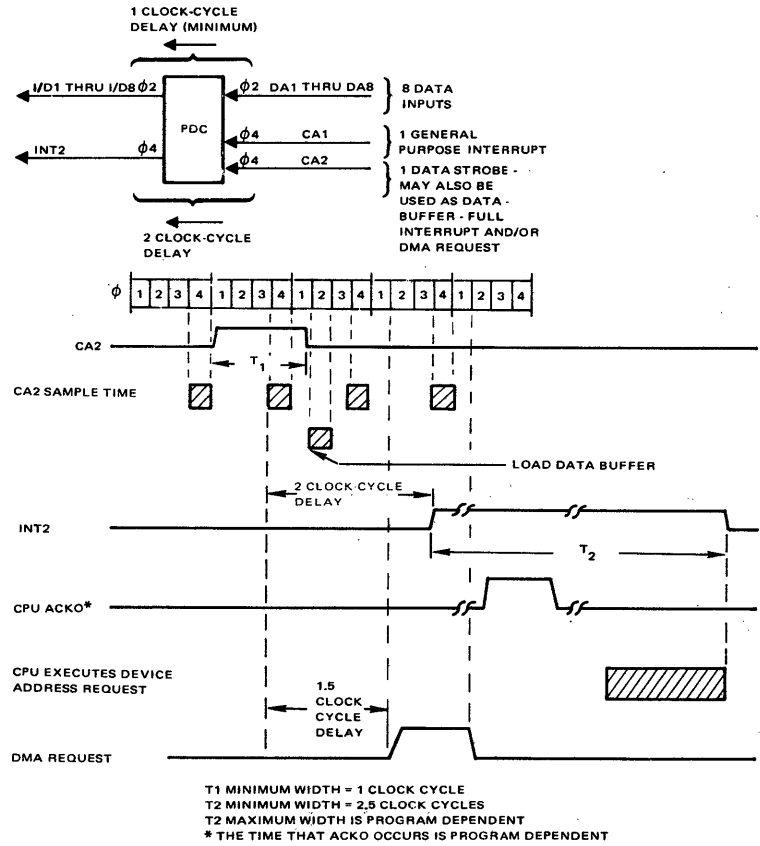


Figure 2-11. Clocked Input

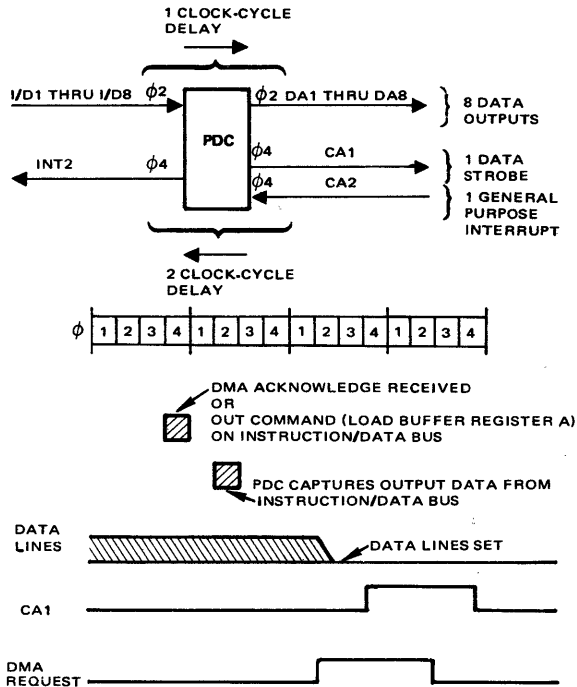


Figure 2-12. Clocked Output

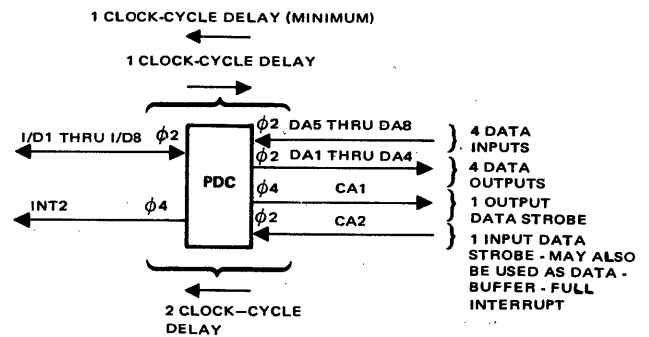


Figure 2-13. Clocked I/O

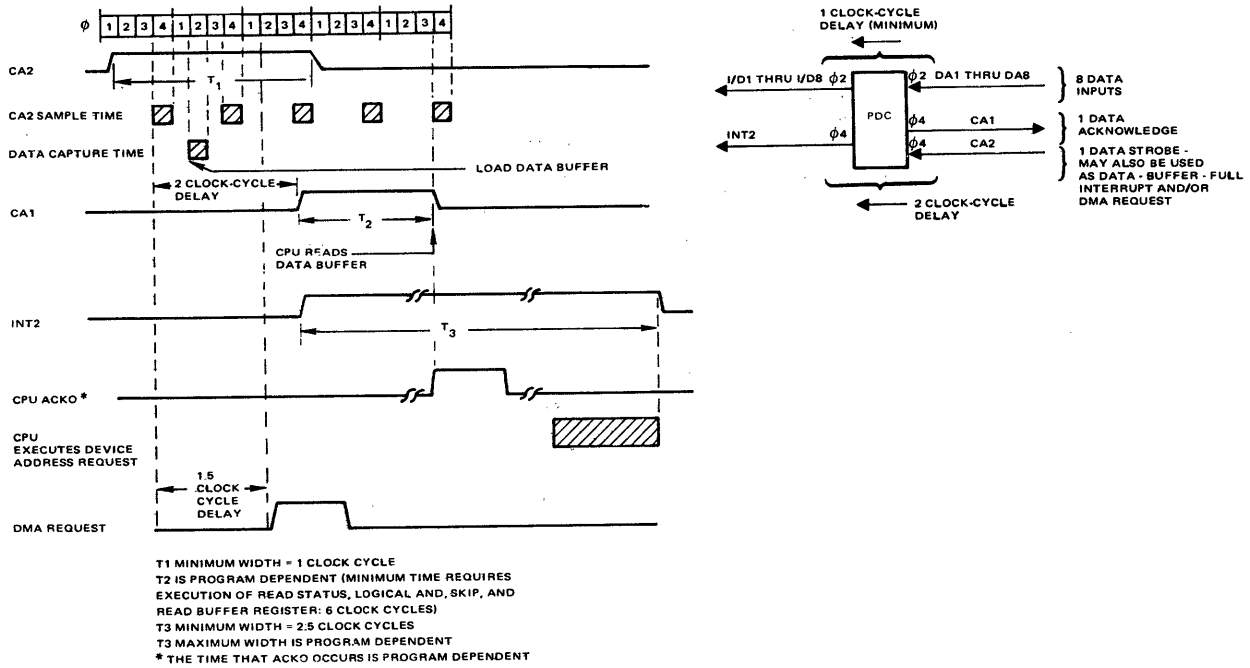


Figure 2-14. Handshake Input

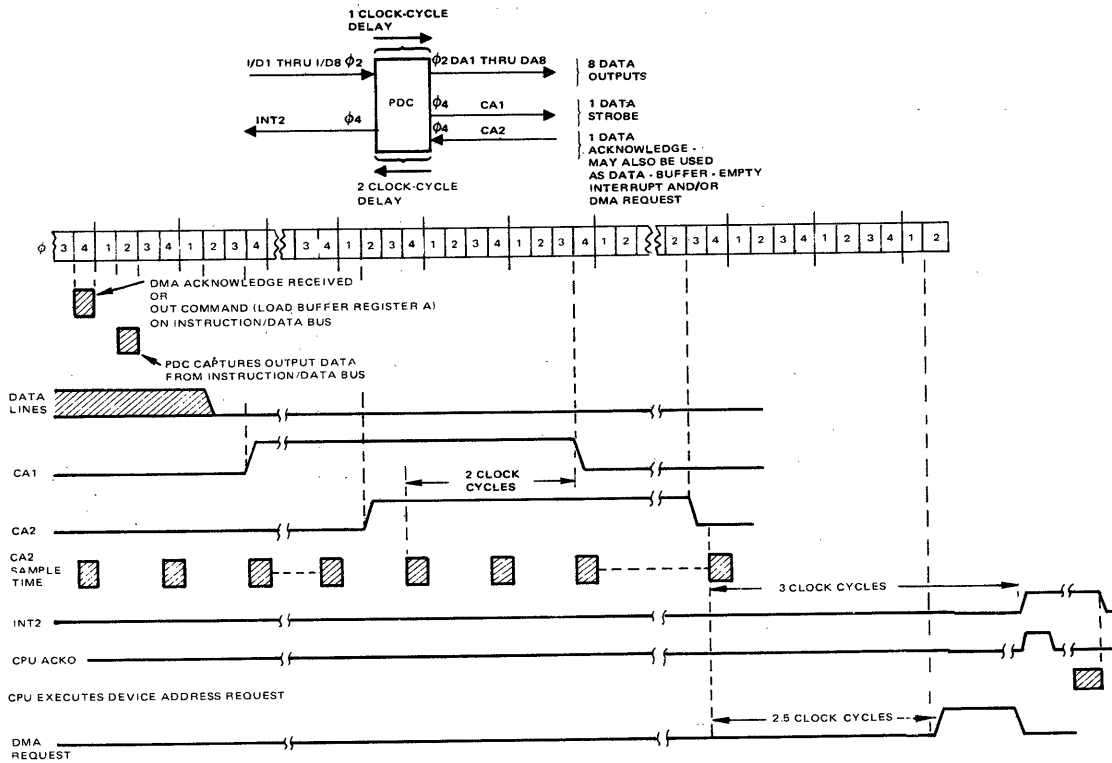
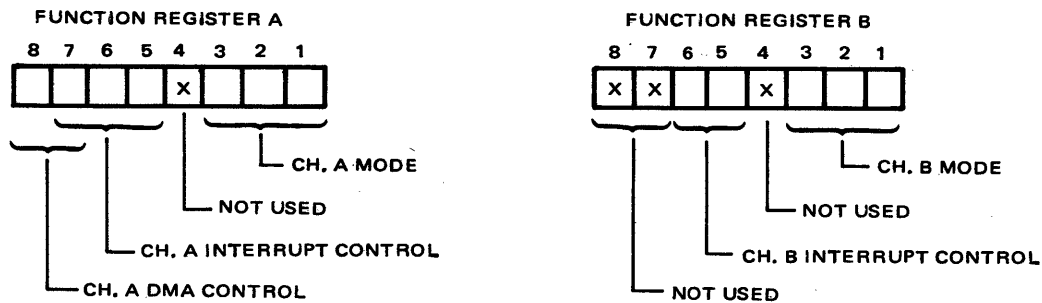


Figure 2-15. Handshake Output

When the DMA option is selected for the Handshake Output mode, the initial request is initiated immediately after Function Register A is loaded with the control data selecting the mode and option. The second and subsequent DMA requests are initiated following completion of the Handshake Output cycle, i.e., during the third $\phi 2$ following the TTL true-to-false transition of CA2 (see Figure 2-15).

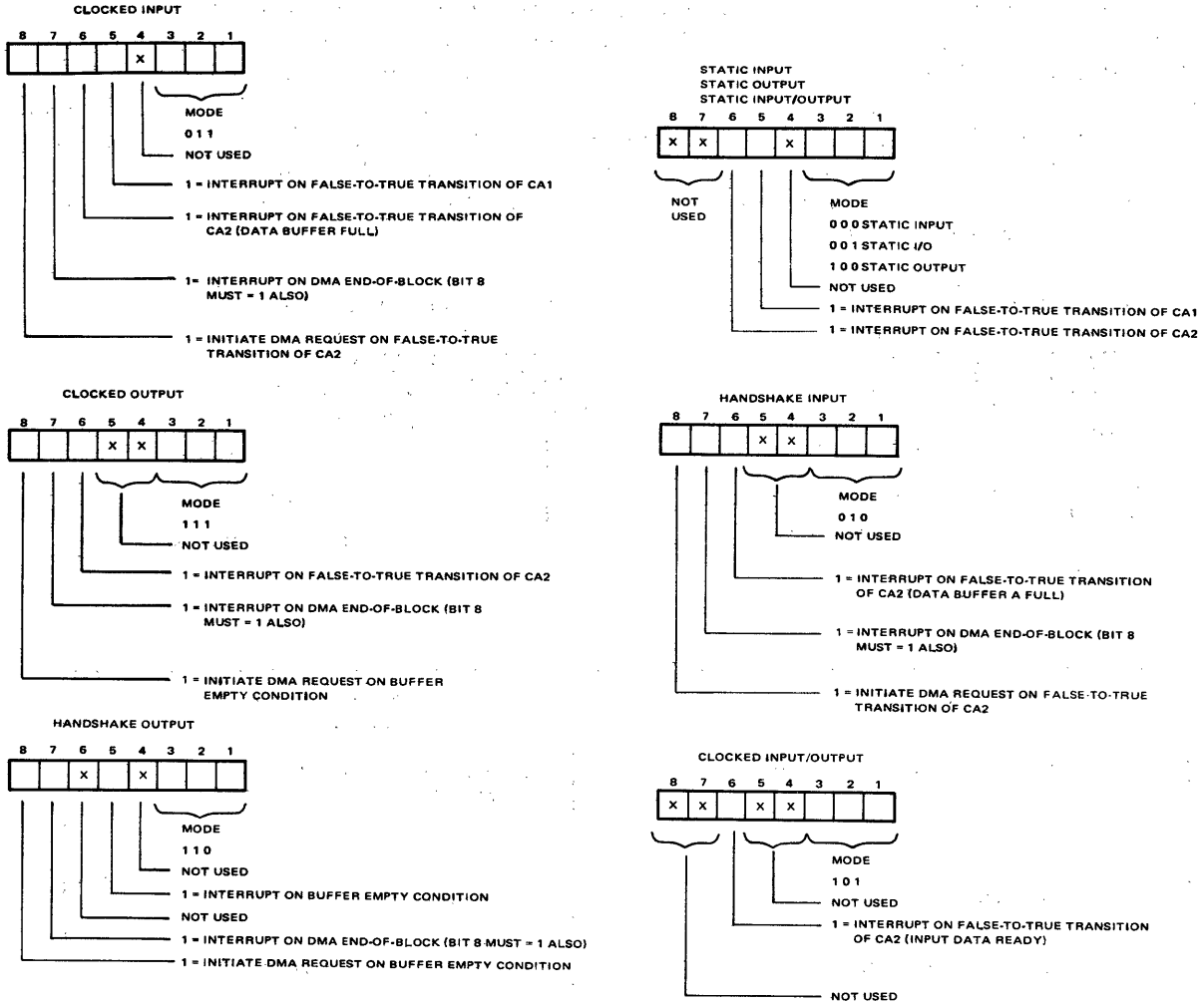
Function Registers

The functions (modes) of each channel are programmable and are controlled by the associated function register. Each register is loaded with a control word under CPU program control. Figure 2-16 shows the fields in each of the function registers. As indicated in Figure 2-16, each channel provides eight basic I/O modes; different modes can be programmed and operated simultaneously on the two channels. Figures 2-17 and 2-18 shows the bit meanings for Function Register A and Function Register B, respectively. Channels A and B have similar modes; however, only Channel A can initiate direct memory access requests.



BIT	FUNCTION REGISTER A	FUNCTION REGISTER B
1 - 3	ESTABLISHES 1 OF 8 BASIC I/O MODES	ESTABLISHES 1 OF 8 BASIC I/O MODES
4	NOT USED	NOT USED
5	CA1 INTERRUPT CONTROL	CB1 INTERRUPT CONTROL
6	CA2 INTERRUPT CONTROL	CB2 INTERRUPT CONTROL
7	DMA END-OF-BLOCK INTERRUPT CONTROL	NOT USED
8	DMA ENABLE	NOT USED

Figure 2-16. Function Register Fields



NOTE: THE BIT CONFIGURATIONS SHOWN ARE AS THEY APPEAR IN THE CPU ACCUMULATOR PRIOR TO TRANSFER TO THE PDC.

Figure 2-17. Function Register A Bit Interpretation

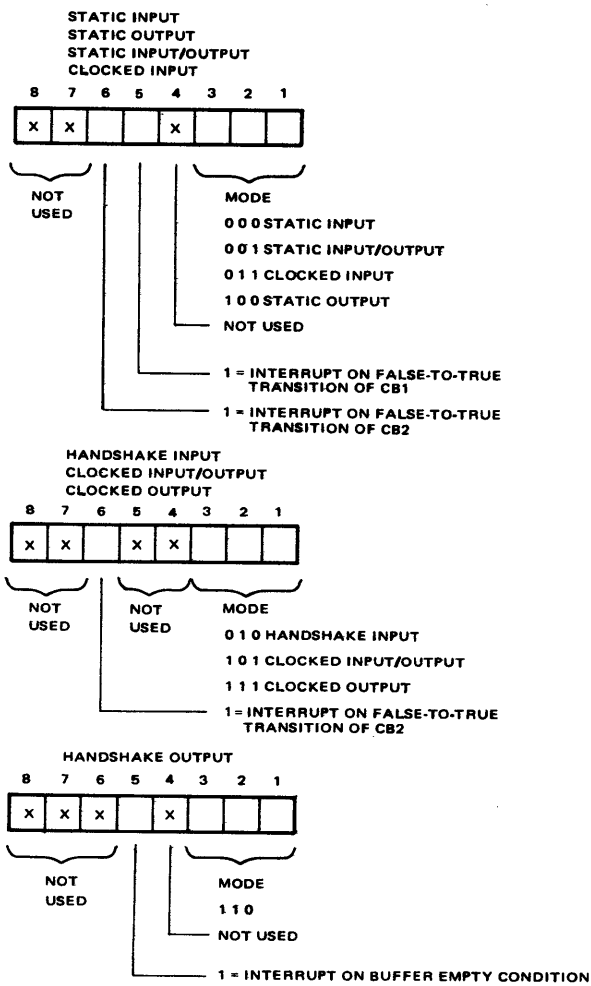


Figure 2-18. Function Register B Bit Interpretation

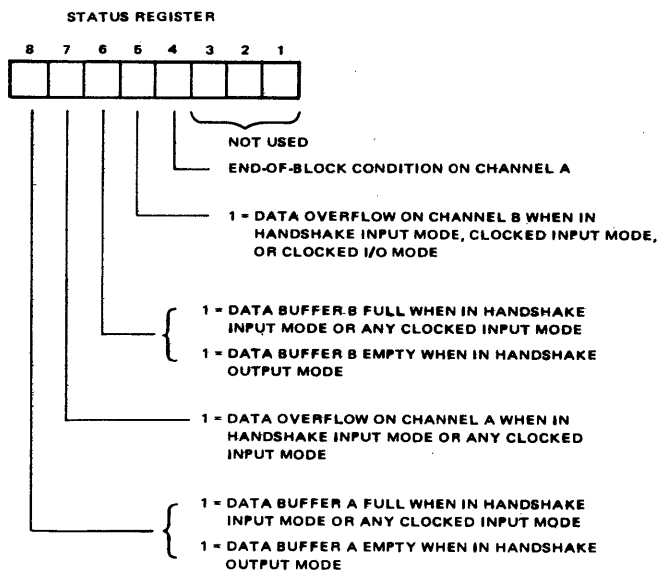


Figure 2-19. PDC Status Register Format

Status Register

The contents of the Status Register can be copied into the CPU accumulator under program control. Figure 2-19 shows the format of the information contained in the Status Register.

Interrupt Status Word

NOTE: Refer to Section 3 for a description of the interrupt system.

When the PPS-8 CPU executes a Read Interrupt Status command, the PDC will transmit an interrupt status word to the CPU accumulator provided that the following two conditions are both met.

- 1) The PDC is requesting interrupt service (INT2 true).
- 2) The PDC has received the ACKI acknowledge pulse from the CPU.

Note that in a PPS-8 system containing multiple PDC devices, the above conditions will be true for only one device at any given time.

Figure 2-20 shows the format and information contained in the interrupt status word. The exact meaning of this information depends upon the current PDC mode (see Table 2-1).

PDC Instruction Set

The PDC responds to eight commands from the PPS-8 CPU. Four of these are of the output (OUT) type and are described below.

- 1) LFRA – Load Function Register A

This command loads Function Register A with the contents of the CPU accumulator and resets bits 4, 7 and 8 of the status register.

- 2) LFRB – Load Function Register B

This command loads Function Register B with the contents of the CPU accumulator and resets bits 5 and 6 of the status register.

- 3) LBRA – Load Buffer Register A

This command loads Buffer Register A with the contents of the CPU accumulator and resets bit 8 of the status register.

- 4) LBRB – Load Buffer Register B

This command loads Buffer Register B with the contents of the CPU accumulator and resets bit 6 of the status register.

The remaining four commands are of the input (IN) type and are described below.

- 5) RBRA – Read Buffer Register A

The contents of Buffer Register A are placed in the CPU accumulator and bit 8 of the status register is reset.

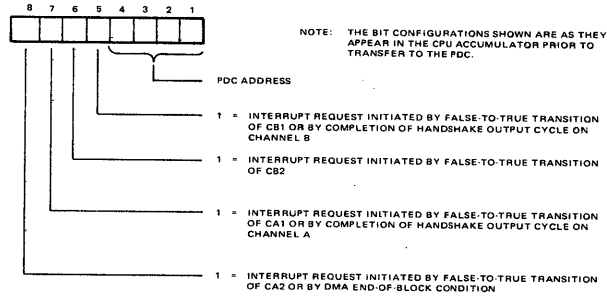


Figure 2-20. Interrupt Status Word

MODE	INTERPRETATION			
	CH. A INTERRUPT STATUS		CH. B INTERRUPT STATUS	
	Bit 8 = 1	Bit 7 = 1	Bit 6 = 1	Bit 5 = 1
Static Input Static Output Static I/O	CA2 Interrupt	CA1 Interrupt	CB2 Interrupt	CB1 Interrupt
Clocked Input Without DMA	CA2 Interrupt (Data Buffer Full)	CA1 Interrupt	CB2 Interrupt (Data Buffer Full)	CB1 Interrupt
Clocked Input with DMA	End-of-Block Interrupt	CA1 Interrupt		
Clocked Output Without DMA	CA2 Interrupt		CB2 Interrupt	
Clocked Output With DMA	End-of-Block Interrupt			
Clocked I/O	CA2 Interrupt (Input Data Ready)		CB2 Interrupt (Input Data Ready)	
Handshake Input Without DMA	CA2 Interrupt (Data Buffer Full)		CB2 Interrupt (Data Buffer Full)	
Handshake Input With DMA	End-of-Block Interrupt			
Handshake Output Without DMA		Channel A Handshake Cycle Complete Interrupt		Channel B Handshake Cycle Complete Interrupt
Handshake Output With DMA	End-of-Block Interrupt			

NOTE: Bit bit polarities shown are as they appear in the CPU accumulator after transfer from the PDC.

Table 2-1. Interrupt Status Word Interpretation

6) RBRB – Read Buffer Register B

The contents of Buffer Register B are placed in the CPU accumulator and bit 6 of the status register is reset.

7) RSR – Read Status Register

The contents of the Status Register are placed in the CPU accumulator and bits 5 and 7 of the status register are reset.

8) RIS – Read Interrupt Status

If the appropriate conditions are met (see text: Interrupt Status Word), the contents of the Interrupt Status Word are placed in the CPU accumulator. (This command is unique in that it always contains a device address of 0.)

DIRECT MEMORY ACCESS CONTROLLER (DMAC)

The Direct Memory Access Controller (DMAC), Part No. 10817, allows PPS-8 I/O devices to access RAM on a cycle-steal basis without disturbing CPU program execution. The DMAC (Figure 2-21) provides control of the Address Bus (A/B1 through A/B14) and two memory control signals (RIH and W/IO) during direct memory access (DMA) operations. Control for eight separate DMA channels is provided by a single DMAC.

Eight DMA request/acknowledge lines (DMA0 through DMA7) provide bidirectional communication between the I/O devices and the DMAC. Bidirectional communication over a single line is accomplished through the use of time multiplexing and pulse coding techniques. Each DMA line (channel) has a fixed position in a priority structure used to resolve simultaneous requests. The channels are numbered in order of priority, with DMA0 having the highest priority. An additional DMA request/acknowledge line (DMRA) provides bidirectional communication between the DMAC and the CPU.

Eight 14-bit address registers and eight 8-bit record length registers are included in the DMAC; one address register and one record length register are associated with each of the eight DMA channels. Each register can be loaded under CPU program control. Two additional control bits are provided for each of the DMA lines. One of these control bits specifies whether the RAM is to read or to write when a DMA request occurs on the associated DMA line. The other control bit is used to select a special cycle mode; the purpose of this mode is described under Record Cycle Mode.

A holding register, associated with the record length registers, allows the CPU to sample the contents of any record length register. When any record length register is loaded by the CPU, the previous contents of that register are transferred to the holding register. The contents of the holding register may be read under CPU program control.

Figure 2-22 shows a typical DMAC application. For simplicity, only a single I/O device, a Parallel Data Controller (PDC), is shown connected to the DMAC. When the PDC requires DMA service, it transmits a pulsed DMA request over DMA0. The DMAC forwards the DMA request to the CPU over DMRA. The CPU acknowledges the DMA request at the completion of the current instruction, provided that the instruction is a non-I/O type. If the CPU is executing an I/O instruction, it will continue program execution until it has completed a non-I/O instruction, at which time it will acknowledge the DMA request. The CPU then enters a "wait" mode, during which it floats its Address Bus drivers, Data Bus drivers, RIH and W/IO. This wait mode is retained until completion of the DMA operations.

When the DMAC receives the CPU acknowledge signal, the following operations occur:

- 1) The DMAC drives an acknowledge signal over the highest priority line currently requesting service.
- 2) The DMAC drives the Address Bus with the contents of the appropriate address register.

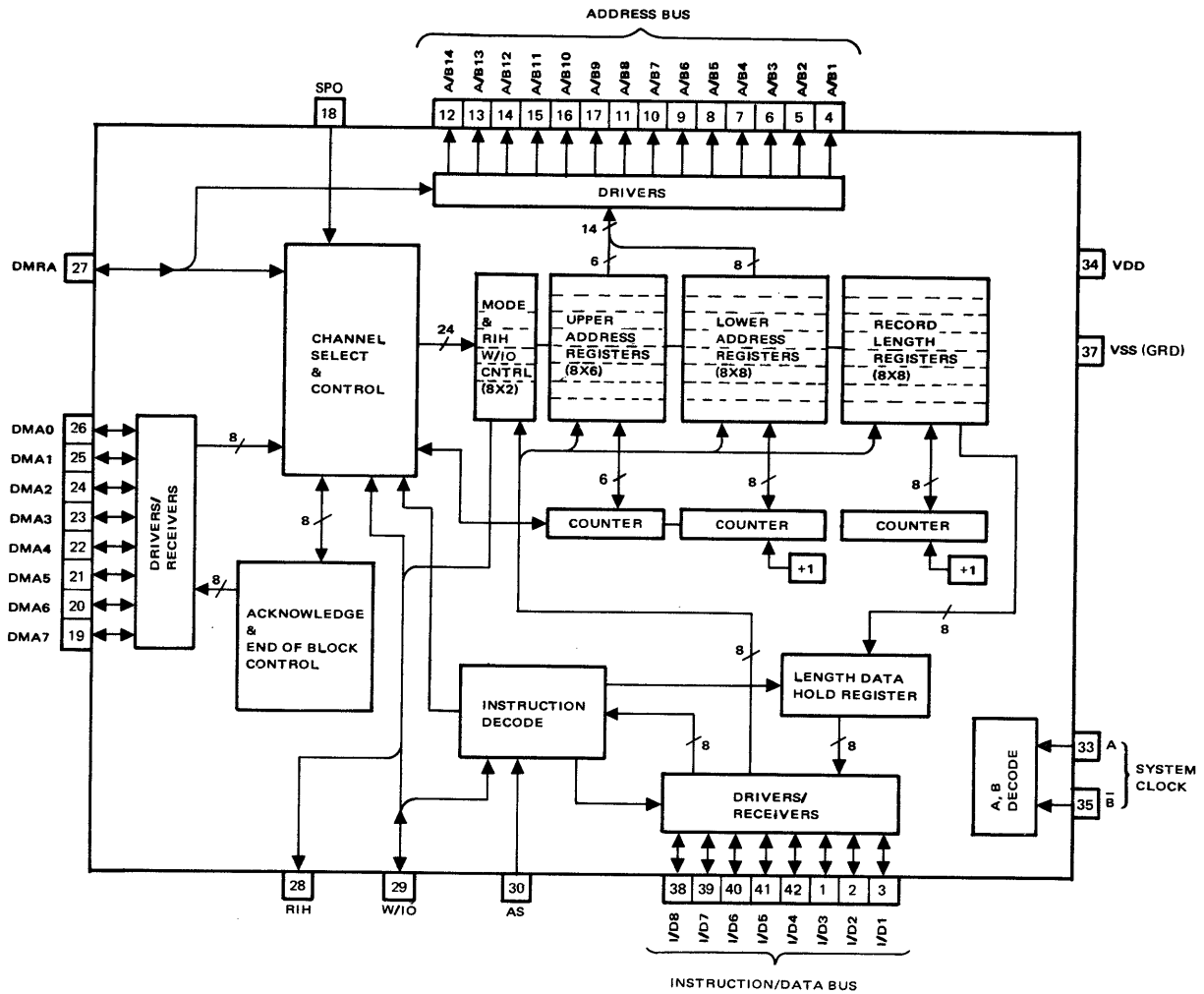


Figure 2-21. Direct Memory Access Controller Block Diagram

- 3) The DMAC drives RIH and W/IO false or true, depending upon whether the addressed RAM is to read or write.
- 4) The I/O device either drives data onto or captures data from the I/D Bus.

After each byte is transferred, the DMAC increments the appropriate address register and record length register. The DMA operations continue until one of the following events occurs.

- 1) The I/O device ceases to request DMA service.
- 2) The record length register makes a transition from 255 to zero.
- 3) The lower address register makes a transition from all ones to zeros.
- 4) CPU interrupt 0 is triggered.
- 5) The DMAC receives a request on a higher priority channel than that currently being serviced.

When the lower address register makes a transition from all ones to all zeros, a carry is propagated through the lower address register and into the upper address register. That is, when the lower address register goes from all ones to all zeros, the upper address register is incremented by one. Each time this operation occurs, the DMAC drops the acknowledge signal to the I/O device and the request signal to the CPU. One clock cycle later (assuming that the I/O device is still requesting DMA service), the DMAC again requests DMA service from the CPU.

When CPU interrupt 0 (INT0) is triggered, the DMAC is allowed to complete the byte transfer currently in progress; thereafter, the CPU ignores the state of DMRA and resumes program execution. When power is restored, SPO clears all DMA acknowledge signals and no further DMA operations occur until another DMA request is received.

When the DMAC receives a DMA request on a higher priority DMA channel than that currently being serviced, it completes the byte transfer currently in progress and then drops the acknowledge signal on the original (lower priority) channel. The DMAC then automatically switches to the higher priority channel and continues the DMA operation.

Record Cycle Mode

A control bit associated with each of the address registers is used to command a special Record Cycle Mode. This mode is available for any of the first seven channels. The control bit is loaded from the CPU: when the control bit is set (1), normal operation (as previously described) occurs; when the control bit is reset (0), the Record Cycle Mode is commanded. When an I/O device requests DMA service and the channel associated with the I/O device is in the Record Cycle Mode, the bytes are transferred as in normal operation until the lower address register (8 bits) makes the transition from 255 to zero. At this time, the contents of the Channel 7 record length register, address register, and control register (2 bits) are automatically transferred to the corresponding registers of the channel operating in the Record Cycle Mode. That is, the address register, record length register, and control register of the channel operating in the Record Cycle Mode are reset to the values stored in the corresponding Channel 7 registers. DMA operation then continues as previously described. Typical use of the Record Cycle Mode involves loading the Channel 8 registers with the same data that is initially loaded into the registers of the channel that is to operate in the Record Cycle Mode; thus, as long as the DMA request is present, the record (beginning at the address specified by the initial contents of the address register and ending at the next address boundary, modulo 256) will be repeatedly transferred between RAM and the I/O device. It should be noted that use of Channel 7 for DMA service is precluded when any channel is in the Record Cycle Mode.

DMA Instruction Set

The DMAC responds to four commands from the PPS-8 CPU. Three commands are of the OUT type and one is of the IN type. These commands are described in Table 2-2. (The basic IN/OUT command format is discussed in Section 3.) As indicated in Section 3, the second byte of the two-byte IN/OUT command contains a 3-bit command field and a 4-bit address field. The command code is interpreted and executed by the addressed I/O device. Normally, the address of an I/O device is

Table 2-2. DMAC Instructions

INSTRUCTION		DESCRIPTION
NAME	MNEMONIC	
Load Address Register, Lower	LARL	Transfers contents of CPU accumulator to lower DMAC address register N (where N is the number coded in the lower 3 bits of the instruction address field)
Load Address Register, Upper	LARU	Transfers contents of CPU accumulator (bits 1 through 6) to corresponding bits of the upper DMAC address register N (where N is the number coded in the lower 3 bits of the instruction address field). Bit 7 of CPU accumulator is transferred to Channel N mode control bit (1 = Normal Mode; 0 = Record Cycle Mode). Bit 8 of CPU accumulator is transferred to Read/Write control bit (0 = Read; 1 = Write).
Load Record Length Register	LRLR	Transfers contents of CPU accumulator to record length register N (where N is the number coded in the lower 3 bits of the instruction address field). Previous contents of record length register N are transferred to holding register.
Read Holding Register	RHR	Transfers contents of the record length holding register to CPU accumulator.

unique and, therefore, there is no possibility of an I/O device interpreting and executing an I/O command intended for some other I/O device. However, the DMAC has a single address strap; only the most significant bit of the 4-bit address field is used to address the DMAC (the lower three bits of the device address field in this case refer to the DMA channel number). Thus, if the address strap (AS) is connected to VSS (logic level zero), the DMAC will respond to any address in the range of zero to seven. Conversely, if AS is connected to VDD (logic level one), the DMAC will respond to any address in the range of 8 to 15. Therefore, care should be taken to ensure that any I/O device whose address falls within the selected DMAC address range does not share any common command codes with the DMAC. (At the time of this writing, the only I/O device that shares common command codes with the DMAC is the TDI. The TDI should not be assigned an address that falls within the selected DMAC address range.)

Timing

Figure 2-23 shows the DMA timing. As shown, there is a one-half clock cycle delay from the time that the DMAC receives a request from an I/O device until the time that the DMAC requests service from the CPU. The CPU response time is variable, depending upon the instruction being executed at the time that it receives the DMA request. The best-case condition exists when the current instruction is a non-I/O type whose execution will be completed during the clock cycle following that in which the request is received; in this case, there is a one-half clock cycle delay from the time that the CPU receives the request until the time that it sends an acknowledge pulse to the DMAC. The worst-case condition exists when the CPU is executing an I/O command (or a series of I/O commands). The CPU will continue program execution until it has executed a non-I/O command before it will acknowledge the DMA request. Completion of the I/O command may take one, or two clock cycles; completion of the following command may take one, two or three clock cycles. Thus, if the CPU is executing an I/O command at the time that it receives the DMA request, the maximum delay before it acknowledges the request will be four and one-half clock cycles (assuming that the command following the I/O command is a non-I/O type).

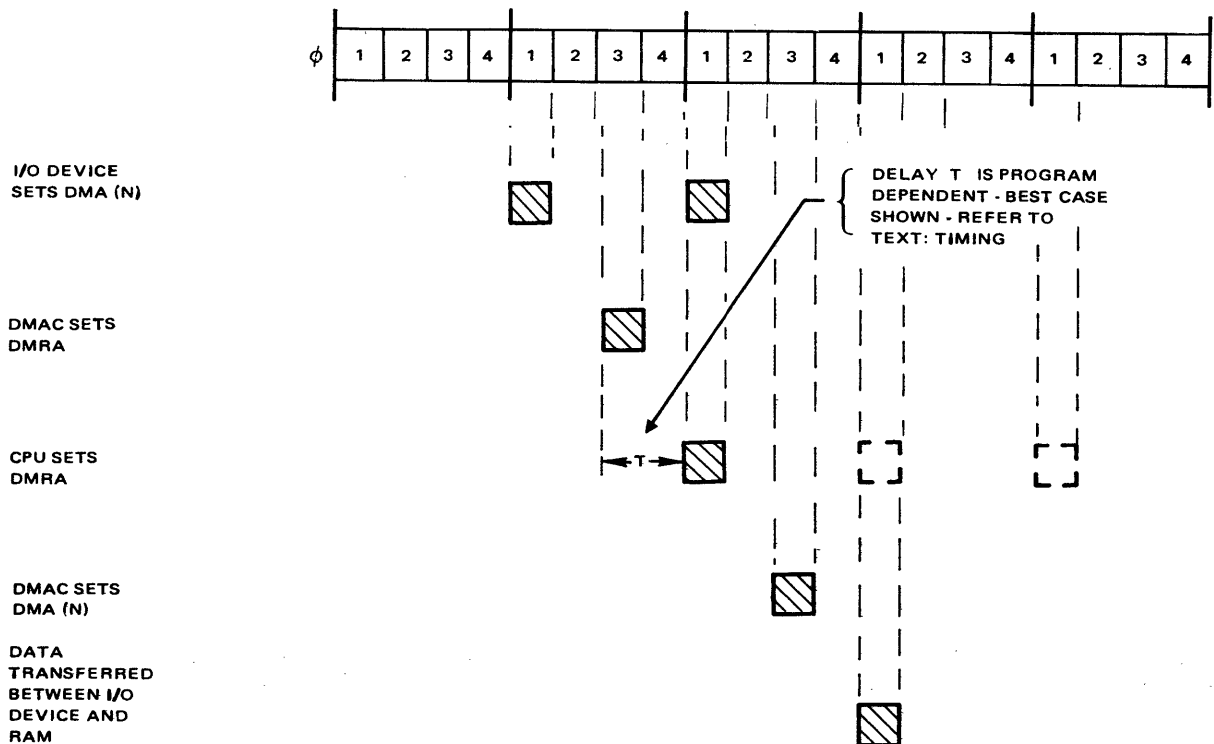


Figure 2-23. DMA Timing

There is a one-half clock cycle delay from the time that the DMAC receives the acknowledge pulse from the CPU until the time that the DMAC, in turn, transmits an acknowledge pulse on the highest priority DMA channel requesting service. Thus, the total delay from the time that an I/O device requests DMA service until it receives an acknowledge pulse is from one and one-half to five and one-half clock cycles (assuming that no higher priority devices are requesting DMA service and that the CPU is not executing a series of I/O commands).

SPECIAL PURPOSE INPUT/OUTPUT DEVICES

The high degree of flexibility found in the PPS-8 allows the use of a wide variety of special purpose input/output devices. Up to 16 of these special devices, in various combinations, may be directly connected to the PPS-8 at the same time, and through appropriate addressing the various devices can be utilized by the PPS-8 to control many different functions concurrently.

Each of the special purpose input/output devices is discussed in the following paragraphs. If additional detail information is required on any of these devices, refer to the applicable data sheet for the specific device.

Printer Controller Circuit

The Printer Controller Circuit device, Part No. 10789, is used for control of the SEIKO 101, 102, and 104 printers. The printer controller interfaces directly to the PPS system without additional components. Printer control is accomplished in a stand-alone manner. That is, after the print data has been loaded from the PPS system, the printer controller circuit controls the printer without additional attention from the PPS system. Features of the printer controller circuit are:

- Up to 21 columns of print data.
- Up to 8 discrete outputs if less than 21 columns of print data are required.
- Red/black color control.
- Automatic generation of from one-to-eight paper feeds.
- Busy signal generation for informing the PPS system as to when a new print line may be initiated.
- One discrete input.
- Address strap to allow two printer controller circuits in the same system.
- Motor on/off control.
- Motor "up-to-speed" detection.

Telecommunications Data Interface (TDI) Circuit

The Telecommunications Data Interface (TDI) circuit, Part No. 10371, together with external filters, gain circuitry, and an approved telephone network access arrangement, can provide either full-duplex or half-duplex, asynchronous data transmission. Transmission can be made over nonconditioned, voice-grade telephone lines. This device can be operated as a bit-serial universal modem or as an integral part of the Rockwell Parallel Processing System (PPS). Features of this device are listed below:

- Full-duplex operation
- Zero to 1200 bps transmission rate.
- Frequency modulation with controllable choice of two sets of modulation frequencies:
 - CCITT (mark = 1300 Hz; space = 2100 Hz)
 - Bell 202 (mark = 1200 Hz; space = 2200 Hz)

- TTL-compatible interface
- Controllable choice of two basic operating modes:
 - Serial Mode: Operates as universal modem with no internal data formatting, buffering, etc.
 - PPS Mode: Operates as part of PPS and as such has the following additional features
 - Data buffering
 - Automatic formatting: Unformatted, serial 8-bit words; character-formatted 8-bit words; variable-length words
 - Interrupt-controlled transfers
 - Controllable choice of even/odd parity
 - Strap-selectable choice of 15 possible addresses (1 to 15)

Bus Interface (B/I) Circuit

The Bus Interface (B/I) Circuit, Part No. 10738, is a multipurpose circuit designed to serve as an interface between the PPS address and data bus dynamic MOS level signals and static commercially available MOS or TTL peripheral memory devices. The B/I circuit can also be used to interface with core memories in those systems which must have a non-volatile memory. The B/I circuit is multipurpose in that it can be appropriately strapped at four input pins to act as an interface for 12 bits of the 14-bit address bus as well as the 8-bit instruction/data bus.

The B/I circuit provides the following features which involve timing, impedance, and logic level interfacing:

- Identical circuits for both instruction/data bus and address bus interface.
- Direct interface – no external components needed.
- MOS-compatible interface.
- TTL-compatible interface.
- Static outputs.

General Purpose Keyboard and Display (GPKD)

The General Purpose Keyboard and Display (GPKD) device, Part No. 10788, provides interfacing between the PPS-8 and a wide variety of keyboard and display devices. The GPKD connects directly with the PPS data bus and is initialized by the Synchronized Power-On (SPO) signal from the PPS-8 CPU.

The GPKD can strobe keyboards with up to 64 single-pole, single-throw momentary switches on an 8 x 8 matrix. A Nine-level, 8-bit key buffer allows up to nine key inputs to be stored until the PPS processes them. Inputs can be entered at an operating speed of 7.68 microseconds per character.

Up to 16 characters of display data can be controlled. The GPKD uses two hexadecimal outputs for data display, (eight outputs). A typical application of the data display is to use 4 bits, (one hexadecimal character) for numeric display, 2 bits for display of decimal point and comma, and the remaining 2 bits for display of up to 32 status indicators.

The following features are found in the GPKD:

KEYBOARD

- Nine-Key Buffering
- Two-Key Rollover
- 7.68 msec Key Debounce Delay
- Handle up to 64 keys on 8 x 8 matrix
- Chip Address Strap Encoded
- Initialized at Power On (SPO)

DISPLAY

- Display up to 16 characters
- 32 Digits of Display Possible
- Two 16-Digit Display Buffers
- Two Sets of Hexadecimal Code Outputs for Display
- TTL Compatible Interface
- Five Percent (5%) Duty Cycle

Serial Data Controller (SDC)

The Serial Data Controller (SDC), Part No. 10930, is a digital receiver-transmitter that interfaces the PPS-8 (or PPS-4) Instruction/Data Bus (eight parallel lines) to a serial communications channel. The SDC is capable of full-duplex or half-duplex operation at synchronous rates up to 250K-bits per second and asynchronous rates up to 18K-Baud. Input/output signals are directly compatible with TTL, MOS and CMOS. Significant features of the SDC are listed below.

- Full-Duplex or Half-Duplex Operation
- Synchronous or Asynchronous Operation
- Fully Programmable – All Control Stored in Internal Control Registers
- Interrupt Request on Receiver Buffer Full, Transmitter Buffer Empty, DMA End-of-Block, and Character Compare
- DMA Request on Receiver Buffer Full or Transmitter Buffer Empty
- Automatic Error Detection (Parity, Receiver Overrun, Framing and Carrier Detect Dropout)
- Receive and Transmit Simultaneously at Different Rates if Desired
- Programmable Transmission Codes:
 - Character Length of 5, 6, 7, or 8 Bits
 - Even Parity, Odd Parity or No Parity
 - One or Two Stop Bits (1 7/16 Stop Bits when 5-Bit Character and X16 Clock Asynchronous Operation are Selected)
- Double Buffered Data Input and Output
- Transmitter Clock Output (Output Clock at Transmitted Bit Rate)
- Three Discrete Inputs and Two Discrete Outputs
- Automatic Status Generation
- Programmable Character Search Mode Facilitates Party Line Operation
- Strap-Selectable 4-Bit or 8-Bit Operation for Compatibility with PPS-4 and PPS-8

Section 3. PRINCIPLES OF OPERATION

BASIC SYSTEM OPERATION

During each program counter bit time, the CPU will address the instruction memory, (ROM), read and decode the instruction, execute the instruction, increment (or load) the program counter in preparation for the next instruction. This bit time or a single cycle instruction execution time is 4 μ s at a 250 KHz frequency. The proprietary instruction architecture and multiphase clocking scheme of the 8-bit CPU result in unusually high data handling rates for a relatively slow external clocking system.

SYSTEM TIMING

The PPS circuits are controlled from a crystal controlled clock generator which provides two synchronized and phased clock signals. These signals, designated A and \bar{B} , are received by the CPU and they are logically divided into four phases, such that the internal signals are being manipulated at four times the frequency of the external A clock. For example, assume A clock is 250 KHz, logic signal flow within the CPU would occur at 1 MHz. The basic clock timing is shown in Figure 3-1.

The PPS parallel bus transfer lines are synchronized by the A and \bar{B} clock signals such that data transfer occurs only during $\emptyset 2$ and $\emptyset 4$ time as depicted by Figure 3-1.

LOGIC LEVELS

A negative logic notation is used in the PPS-8 system. That is, a logic one (1) is defined as the most negative voltage, while a logic zero (0) is defined as the most positive voltage.

MULTIPLEX SYSTEM DATA TRANSFER

In addition to the power and clock signals, there are 24 multiplexed lines interconnecting the CPU with ROM, RAM, and I/O devices. These lines are functionally grouped as follows:

- 14 Parallel Address Lines (A/B 1 through 14)
- 8 Parallel Bidirectional Data Lines (I/D 1 through 8)
- 1 Write Command/I/O Enable Line (W/I O)
- 1 Read Inhibit (RIH)

The fourteen address lines originate from the CPU and are time multiplexed within the CPU to provide direct addressing capability for up to 16,384 8-bit bytes of ROM and 16,384 bytes of RAM.

As are the address lines, the eight data bus lines are time shared lines between the CPU, ROM, RAM and I/Os.

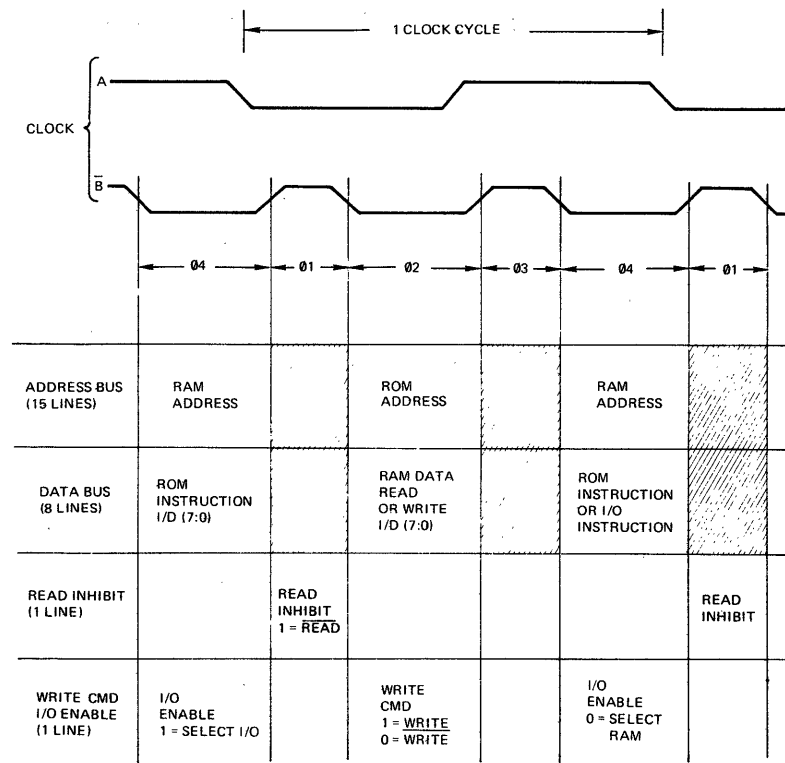


Figure 3-1. Parallel Processing System Bus Timing

During 02, a logical one (1) on the write enable line is interpreted by the RAMS as a write enable command and the data on the bus will be written into RAM using the preceding 04 address. During the preceding 01, the RAM would be commanded not to read via the Read Inhibit Command (logical 1). The RAM is a non-destructive read-out device and is always programmed to read unless instructed not to by the "read inhibit" command.

The same line providing the write command to RAM during 02 time serves as an I/O Enable signal during 04 time when communication is desired to (or from) an I/O device addressed on the data bus. If the I/O enable is on (logical 1) at 04, the "Read Inhibit" line will be used to disable the RAM during the 02 transmittal of information between the CPU and the addressed I/O device.

INSTRUCTION REPERTOIRE

The PPS-8 has an extensive instruction repertoire for performing a variety of arithmetic, logical, and data manipulation functions in a manner which is efficient, both in instruction memory required and speed of execution. Most instruction types require only a single 8-bit byte, but 2 and 3 byte formats are also included to provide extended addressing/functional capability.

The instructions are listed in Table 3-1. In the table, reference is made to notes. The applicable notes are listed following the table.

The symbology used in the instruction descriptions in Table 3-1 is defined below.

<u>Symbol</u>	<u>Symbolic Notation Item or Function</u>
A	Accumulator Register A(1:8)
X	RAM Address Register and Index, X(1:8)
Y	RAM Address Register Y(1:8)
Z	RAM Address Register Z(1:8)
L	Link Register L(1:16)
P	Program Counter Register P(1:14)
S	Stack Pointer Register
W	The W Register
C	Carry Flip-Flop
IC	Intermediate state of Carry Flip-Flop
Q	Intermediate Carry Flip-Flop
I	Instruction I(1:8)
I1	First byte of multiple byte instruction I1(1:8)
I2	Second byte of multiple byte instruction I2(1:8)
I3	Third byte of multiple byte instruction I3(1:8)
M	RAM memory contents
R(n)	Bit n of Register R
R(n:m)	Bit n through m of R, inclusively
W/IO	Write and I/O Enable Control Line
RIH	Read Inhibit Control Line
Byte	Eight-bit Data Field
Digit	Four-bit Data Field
Page	Block of 128 bytes
\longrightarrow	Replaces (or \longleftarrow)
\longleftrightarrow	Exchange
\overline{R}	1's complement of state R.
\wedge	Logical Product (AND)
\vee	Logical Sum (Inclusive OR)
∇	Logical (Exclusive OR)
-	Algebraic Subtract
+	Algebraic Add
>	Greater than
<	Less than
=	Equal to
I/D(8:1)	Instruction/Data Bus (lines 1 through 8)
(L)	ROM memory contents addressed by L
SP _u	Byte from upper address portion of Subroutine Entry Pool
SP _l	Byte from lower address portion of Subroutine Entry Pool

Table 3-1. List of Instructions

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
L	Load A	1	1	The current RAM operand is placed in the accumulator	$A \leftarrow M$	1
LN	Load A, Increment Address	1	1	Same as L. Additionally, the X register is incremented	$A \leftarrow M$ $X \leftarrow X+1$, skip if $X=0$	1,2,8
LD	Load A, Decrement Address	1	1	Same as L. Additionally, the X register is decremented	$A \leftarrow M$ $X \leftarrow X-1$, skip if $X=127$	1,2,8
LNXL	Load A, Increment Address, Exchange L	1	1	Same as LN. Additionally, the contents of the L register and the Z & X registers are exchanged	$A \leftarrow M$ $X \leftarrow X+1$, skip if $X=0$ $Z, X \leftrightarrow L$	1,2,8
LDXL	Load A, Decrement Address, Exchange L	1	1	Same as LD. Additionally, the contents of the L register and the Z & X registers are exchanged	$A \leftarrow M$ $X \leftarrow X-1$, skip if $X=127$ $Z, X \leftrightarrow L$	1,2,8
LNCX	Load A, Increment & Compare Address, Exchange L	1	1	Same as LNXL. Additionally, the next instruction is skipped if $X = Y$	$A \leftarrow M$ $X \leftarrow X+1$ Skip if $X=0$ or $X=Y$ $Z, X \leftrightarrow L$	1,2,8, 19
LDCX	Load A, Decrement & Compare Address, Exchange L	1	1	Same as LDXL. Additionally, the next instruction is skipped if $X = Y$	$A \leftarrow M$ $X \leftarrow X-1$ Skip if $X=127$ or $X=Y$ $Z, X \leftrightarrow L$	1,2,8, 19
LNXY	Load A, Increment Address, Exchange Y	1	1	Same as LN. Additionally, the contents of the X & Y registers are exchanged	$A \leftarrow M$ $X \leftarrow X+1$, skip if $X=0$ $X \leftrightarrow Y$	1,2,8
S	Store A	1	1	The contents of the accumulator are stored in the current RAM operand address	$M \leftarrow A$	1
SN	Store A, Increment Address	1	1	Same as S. Additionally, the X register is incremented	$M \leftarrow A$ $X \leftarrow X+1$, skip if $X=0$	1,2,8
SD	Store A, Decrement Address	1	1	Same as S. Additionally, the X register is decremented	$M \leftarrow A$ $X \leftarrow X-1$, skip if $X=127$	1,2,8
SNXL	Store A, Increment Address, Exchange L	1	1	Same as SN. Additionally, the contents of the L register and the Z & X registers are exchanged	$M \leftarrow A$ $X \leftarrow X+1$, skip if $X=0$ $Z, X \leftrightarrow L$	1,2,8
SDXL	Store A, Decrement Address, Exchange L	1	1	Same as SD. Additionally, the contents of the L register and the Z & X registers are exchanged	$M \leftarrow A$ $X \leftarrow X-1$, skip if $X=127$ $Z, X \leftrightarrow L$	1,2,8

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES	
				VERBAL	SYMBOLIC		
SNCX	Store A, Increment & Compare Address, Exchange L	1	1	Same as SNXL. Additionally, the next instruction is skipped if X = Y	$M \leftarrow A$ $X \leftarrow X+1$ Skip if X=0 or X=Y $Z, X \leftrightarrow L$	1,2,8, 19	
SDCX	Store A, Decrement & Compare Address, Exchange L	1	1	Same as SDXL. Additionally, the next instruction is skipped if X = Y	$M \leftarrow A, X \leftarrow X-1$ Skip if X=127 or X=Y $Z, X \leftrightarrow L$	1,2,8, 19	
SNXY	Store A, Increment Address, Exchange Y	1	1	Same as SN. Additionally, the contents of the X and Y registers are exchanged	$M \leftarrow A$ $X \leftarrow X+1$, skip if X=0 $X \leftrightarrow Y$	1,2,8	
X	Exchange	2	2	These instructions are identical to the corresponding store instructions except that the accumulator and the current RAM operand are exchanged	$A \leftrightarrow M$	1,6	
XN	Exchange, Increment Address	2	2		$A \leftrightarrow M$ $X \leftarrow X+1$, skip if X=0	1,2,6,8	
XD	Exchange, Decrement Address	2	2		$A \leftrightarrow M$ $X \leftarrow X-1$, skip if X=127	1,2,6,8	
XNXL	Exchange, Increment Address, Exchange L	2	2		$A \leftrightarrow M$ $X \leftarrow X+1$, skip if X=0 $Z, X \leftrightarrow L$	1,2,6,8	
XDXL	Exchange, Decrement Address, Exchange L	2	2		$A \leftrightarrow M$ $X \leftarrow X-1$, skip if X=127 $Z, X \leftrightarrow L$	1,2,6,8	
XNCX	Exchange, Increment & Compare Address, Exchange L	2	2		$A \leftrightarrow M$ $X \leftarrow X+1$ Skip if X=0 or X=Y $Z, X \leftrightarrow L$	1,2,6,8, 19	
XDCX	Exchange, Decrement & Compare Address, Exchange L	2	2		These instructions are identical to the corresponding store instructions except that the accumulator and the current RAM operand are exchanged	$A \leftrightarrow M$ $X \leftarrow X-1$ Skip if X=127 or X=Y $Z, X \leftrightarrow L$	1,2,6,8, 19

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
XNXY	Exchange, Increment Address, Exchange Y	2	2	These instructions are identical to the corresponding store instructions except that the accumulator and the current RAM operand are exchanged	$A \leftrightarrow M$ $X \leftarrow X+1$, skip if $X=0$ $X \leftrightarrow Y$	1,2,6,8
LX	Load X	1,2	2	The current RAM operand is placed in the X register	$X \leftarrow M$	4
LY	Load Y	1,2	2	The current RAM operand is placed in the Y register	$Y \leftarrow M$	4
LZ	Load Z	1,2	2	The current RAM operand is placed in the Z register	$Z \leftarrow M$	4
LAI	Load A Immediate	1-3	3	The specified literal operand is placed in the accumulator	$A \leftarrow I3$	3,4
LXI	Load X Immediate	1-3	3	The specified literal operand is placed in the X register	$X \leftarrow I3$	3,4
LYI	Load Y Immediate	1-3	3	The specified literal operand is placed in the Y register	$Y \leftarrow I3$	3,4
LZI	Load Z Immediate	1-3	3	The specified literal operand is placed in the Z register	$Z \leftarrow I3$	3,4
LAL	Load A through Link	1,2	3	The ROM operand addressed by the L register is placed in the accumulator	$W \leftarrow A$, $A \leftarrow (L)$ $L \leftarrow L+1$	4,5,14
LXL	Load X through Link	1,2	3	The ROM operand addressed by the L register is placed in the X register	$X \leftarrow (L)$ $L \leftarrow L+1$	4,5
LYL	Load Y through Link	1,2	3	The ROM operand addressed by the L register is placed in the Y register	$Y \leftarrow (L)$ $L \leftarrow L+1$	4,5
LZL	Load Z through Link	1,2	3	The ROM operand addressed by the L register is placed in the Z register	$Z \leftarrow (L)$ $L \leftarrow L+1$	4,5
LXA	Load X from A	1	1	The contents of the accumulator are placed in the X register	$X \leftarrow A$	16
LYA	Load Y from A	1	1	The contents of the accumulator are placed in the Y register	$Y \leftarrow A$	16
LZA	Load Z from A	1	1	The contents of the accumulator are placed in the Z register	$Z \leftarrow A$	16

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
LLA	Load L from A	1	1	The contents of the accumulator are placed in the upper 8 bits of the L register	$L(16:9) \leftarrow A$	16
XY	Exchange Y	1	1	The contents of the X and Y register are exchanged	$X \leftrightarrow Y$	
XL	Exchange L	1	1	The contents of the L register and the Z & X registers are exchanged	$L \leftrightarrow Z, X$	
XAX	Exchange A and X	1	1	The contents of the X register and the accumulator are exchanged	$A \leftrightarrow X$	
XAY	Exchange A and Y	1	1	The contents of the Y register and the accumulator are exchanged	$A \leftrightarrow Y$	
XAZ	Exchange A and Z	1	1	The contents of the Z register and the accumulator are exchanged	$A \leftrightarrow Z$	
XAL	Exchange A and L	1	1	The contents of the upper half of the L register and the accumulator are exchanged	$A \leftrightarrow L(16:9)$	
INCX	Increment X	1	1	The X register is incremented by one	$X \leftarrow X+1$ Skip if X=0	2
DECX	Decrement X	1	1	The X register is decremented by one	$X \leftarrow X-1$ Skip if X=127	2
INXY	Increment X, Exchange Y	1	1	The X register is incremented and the contents of the X and Y registers are exchanged	$X \leftarrow X+1$, skip if X=0 $X \leftrightarrow Y$	2
DEXY	Decrement X, Exchange Y	1	1	The X register is decremented and the contents of the X and Y registers are exchanged	$X \leftarrow X-1$, skip if X=127 $X \leftrightarrow Y$	2
INCY	Increment Y	2	2	The Y register is incremented by one	$Y \leftarrow Y+1$ Skip if Y=0	2,7,8
DECY	Decrement Y	2	2	The Y register is decremented by one	$Y \leftarrow Y-1$ Skip if Y=127	2,7,8
PSHA	Push A	1,2	2	The contents of the accumulator are pushed into the stack	$A \rightarrow (S)$ $S \leftarrow S+1$	4,13
PSHX	Push X	1,2	2	The contents of the X register pushed into the stack	$X \rightarrow (S)$ $S \leftarrow S+1$	4,13
PSHY	Push Y	1,2	2	The contents of the Y register are pushed into the stack	$Y \rightarrow (S)$ $S \leftarrow S+1$	4,13
PSHZ	Push Z	1,2	2	The contents of the Z register are pushed into the stack	$Z \rightarrow (S)$ $S \leftarrow S+1$	4,13

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
PSHL	Push L	1	3	The contents of the L register are pushed into the stack and replaced by the contents of the A and W registers	$L \rightarrow (S+1, S)$ $A, W \rightarrow L$ $S \leftarrow S+2$	13,14
POPA	Pop A	1,2	2	The uppermost byte is popped from the stack and placed in the accumulator	$S \leftarrow S-1$ $A \leftarrow (S)$ Skip if $S=31$	4,13
POPX	Pop X	1,2	2	The uppermost byte is popped from the stack and placed in the X register	$S \leftarrow S-1$ $X \leftarrow (S)$ Skip if $S=31$	4,13
POPY	Pop Y	1,2	2	The uppermost byte is popped from the stack and placed in the Y register	$S \leftarrow S-1$ $Y \leftarrow (S)$ Skip if $S=31$	4,13
POPZ	Pop Z	1,2	2	The uppermost byte is popped from the stack and placed in the Z register	$S \leftarrow S-1$ $Z \leftarrow (S)$ Skip if $S=31$	4,13
POPL	Pop L	1	3	The uppermost 2 bytes are popped from the stack and placed in the L register	$S \leftarrow S-2$ $L \leftarrow (S+1, S)$	13
A	Add	1	1	The sum of the accumulator and the current RAM operand are placed in the accumulator	$C, A \leftarrow A+M$ $Q \leftarrow IC$	10,1
AC	Add with Carry	1	1	Same as A except the carry flip-flop, C, is used as a carry-in	$C, A \leftarrow A+M+C$ $Q \leftarrow IC$	10,1
ASK	Add, Skip on Carry	1	1	Same as A. Additionally, the next instruction is skipped if a carry-out is generated	$C, A \leftarrow A+M$ $Q \leftarrow IC$ Skip if $C=1$	10,1
ACSK	Add with Carry, Skip on Carry	1	1	Same as AC. Additionally, the next instruction is skipped if a carryout is generated	$C, A \leftarrow A+M+C$ Skip if $C=1$	10,1
AISK	Add Immediate, Skip on Carry	1-3	3	The sum of the accumulator and the specified literal operand is placed in the accumulator	$A \leftarrow A+I3$ $Q \leftarrow IC$ Skip if carry-out	3,4,10
INCA	Increment A	1	1	The accumulator is incremented by one	$A \leftarrow A+1$ $Q \leftarrow IC$	10
DC	Decimal Correct (1)	1	1	The hexadecimal value 66 is added to the accumulator	$A \leftarrow A+66_{16}$ $Q \leftarrow IC$	10,16
DCC	Decimal Correct (2)	1	1	The accumulator is modified based on the states of the C&Q flip-flops	C, Q 0,0 $A \leftarrow A+(9A)_{16}$ 0,1 $A \leftarrow A+(A0)_{16}$ 1,0 $A \leftarrow A+(FA)_{16}$ 1,1 No change	10,16

Table 3-1. List of Instructions (Continued)

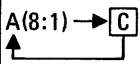
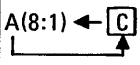
MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
AN	Logical AND	1	1	The logical product of the accumulator and the current RAM operand is placed in the accumulator	$A \leftarrow A \wedge M$	1
ANI	Logical AND Immediate	1-3	3	The logical product of the accumulator and the specified literal operand is placed in the accumulator	$A \leftarrow A \wedge I3$	3,4
OR	Logical OR	1	1	The logical sum of the accumulator and the current RAM operand is placed in the accumulator	$A \leftarrow A \vee M$	1
EOR	Logical Exclusive OR	1	1	The logical exclusive or (addition without carry) of the accumulator and the current RAM operand is placed in the accumulator	$A \leftarrow A \vee M$	1
COM	Complement	1	1	The one's complement of the accumulator is placed in the accumulator	$A \leftarrow \bar{A}$	16
SC	Set Carry	1	1	The carry flip-flop, C, is set (1)	$C \leftarrow 1$	10
RC	Reset Carry	1	1	The carry flip-flop, C, is reset (0)	$C \leftarrow 0$	10
RAR	Rotate A Right	1	1	The accumulator and C flip-flop are circular shifted one bit to the right		16
RAL	Rotate A Left	1	1	The accumulator and C flip-flop are circular shifted one bit to the left		16
MDR	Move Digit Right	1	1	The accumulator is shifted right 4 bits and the least significant 4 bits of the current RAM operand are placed in the vacated accumulator positions	$A(8:5) \rightarrow A(4:1)$ $M(4:1) \rightarrow A(8:5)$	16,18
MDL	Move Digit Left	1	1	The accumulator is shifted left 4 bits and the most significant 4 bits of the current RAM operand are placed in the vacated accumulator positions	$A(8:5) \leftarrow A(4:1)$ $A(4:1) \leftarrow M(8:5)$	16,18
SB	Set Bit (n)	1,2	2	The specified bit of the current RAM operand is set (1)	$M \leftarrow M \vee 2^{(n-1)}$	4
RB	Reset Bit (n)	1,2	2	The specified bit of the current RAM operand is reset (0)	$M \leftarrow M \wedge \overline{2^{(n-1)}}$	4
B	Branch	1,2	1,2	The specified address is placed in the P-register	$P(7:1) \leftarrow I1(7:1)$ If $I1(8)=1$, $P(14:8) \leftarrow I2(7:1)$	12,16

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
BDI	Branch, Disable Interrupts	2	2	Same as B. Additionally, the interrupts are disabled	$P(7:1) \leftarrow 11(7:1)$ $P(14:8) \leftarrow 12(7:1)$ Disable Interrupts	12,16
BL	Branch and Link	1,2	3	The specified address is placed in the P-register. The previous contents of the P-register (incremented) are saved in the L register together with the state of the C flip-flop. The previous contents of the L register are pushed into the stack	$L \rightarrow (S+1,S)$ $S \leftarrow S+2$ $P \rightarrow L(15:9,7:1)$ $C \rightarrow L(16)$ If $I(6)=1$ $P(14:8) \leftarrow SP_U(7:1)$ $P(7:1) \leftarrow SP_1(7:1)$ If $I(6)=0$ $P(12:8) \leftarrow 11(5:1)$ $P(7:1) \leftarrow 12(7:1)$ $P(13) \leftarrow 12(8)$ $P(14) \leftarrow 0$	15,16
RT	Return	1	3	The P-register and C flip-flop are loaded from the L-register. The uppermost 2 bytes are popped from the stack and placed in the L-register	$P \leftarrow L(15:9,7:1)$ $C \leftarrow L(16)$ $S \leftarrow S-2$ $L \leftarrow (S+1,S)$	11
RSK	Return & Skip	1	3	Same as RT except that the next instruction (i.e., the instruction at the "return" location) is skipped	$P \leftarrow L(15:9,7:1)$ $C \leftarrow L(16)$ $S \leftarrow S-2, L \leftarrow (S+1,S)$ Skip next instruction	11
RTI	Return, Enable Interrupts	1	3	Same as RT. Additionally, the interrupts are enabled	$P \leftarrow L(15:9,7:1)$ $C \leftarrow L(16)$ $S \leftarrow S-2, L \leftarrow (S+1,S)$ Enable interrupts	
NOP	No Operation	1	1	No function is performed. The branch condition tag is used		
SKC	Skip if Carry	1	1	The next instruction is skipped if the carry flip-flop, C, is set	Skip if $C=1$	8
SKNC	Skip if No Carry	1	1	The next instruction is skipped if the carry flip-flop, C, is reset	Skip if $C=0$	8
SKZ	Skip if Zero	1	1	The next instruction is skipped if the accumulator equals zero	Skip if $A=0$	8
SKNZ	Skip if Non-Zero	1	1	The next instruction is skipped if the accumulator does not equal zero	Skip if $A \neq 0$	8
SKP	Skip if Positive	1	1	The next instruction is skipped if the most significant bit of the accumulator is zero	Skip if $A(8)=0$	8

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
SKN	Skip if Negative	1	1	The next instruction is skipped if the most significant bit of the accumulator is one	Skip if $A(8)=1$	8
SKE	Skip if Equal	1	1	The next instruction is skipped if the accumulator and the current RAM operand are equal	Skip if $A=M$	8
BBT	Branch if Bit (n) True	2,3	2,3	A program branch is executed if the specified bit of the current RAM operand is true (1)	If $M \wedge 2^{(n-1)}=1$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16
BBF	Branch if Bit (n) False	2,3	2,3	A program branch is executed if the specified bit of the current RAM operand is false (0)	If $M \wedge 2^{(n-1)}=0$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16
BC	Branch if Carry	2,3	2,3	A program branch is executed if the carry flip-flop, C, is set (1)	If $C=1$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8
BNC	Branch if No Carry	2,3	2,3	A program branch is executed if the carry flip-flop, C, is reset (0)	If $C=0$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8
BZ	Branch if Zero	2,3	2,3	A program branch is executed if the accumulator equals zero	If $A=0$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8
BNZ	Branch if Non-Zero	2,3	2,3	A program branch is executed if the accumulator does not equal zero	If $A \neq 0$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8
BP	Branch if Positive	2,3	2,3	A program branch is executed if the most significant bit of the accumulator is zero	If $A(8)=0$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8
BN	Branch if Negative	2,3	2,3	A program branch is executed if the most significant bit of the accumulator is one	If $A(8)=1$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8
BNE	Branch if Not Equal	2,3	2,3	A program branch is executed if the accumulator is not equal to the current RAM operand	If $A \neq M$, then $P(7:1) \leftarrow I2(7:1)$ & if $I2(8)=1$, $P(14:8) \leftarrow I3(7:1)$	9,12,16, 8

Table 3-1. List of Instructions (Continued)

MNEMONIC	NAME	BYTES	CYCLES	DESCRIPTION		NOTES
				VERBAL	SYMBOLIC	
IO4	Digit I/O (C, D)	2	2	Command C is transmitted to I/O device D. Bits 8-5 of the accumulator are transmitted to the device and bits 1-4 are received from the device	I2 → I/D(8:1) A(8:5) → I/D(8:5) A(4:1) ← I/D(4:1)	16,17
IN	Input (C, D)	2	2	Command C is transmitted to I/O device D. The accumulator is loaded with a data byte transmitted by the device. If D is omitted, a zero (all-call) device address is transmitted	I2 → I/D(8:1) A ← I/D(8:1)	16,17
OUT	Output (C, D)	2	2	Same as IN except the accumulator contents are transmitted to the device	I2 → I/D(8:1) A → I/D(8:1)	16,17
RIS	Read Interrupt Status	2	2	The accumulator is loaded with the interrupt status word from the highest priority I/O device currently requesting service.	I2 → I/D(8:1) A ← I/D(8:1)	16,17

NOTES FOR INSTRUCTION DESCRIPTIONS LISTED IN TABLE 3-1

- 1) The address (14-bits) for RAM operands is specified by the Z and X registers. The lower 7 bits of X specify the byte address (least significant 7 bits). If bit 8 of the X register is a one, the high order 7 bits of RAM address (the page address) are specified by the least significant 7 bits of the Z register. Otherwise, the page address is set to zero.
- 2) All instructions which increment or decrement the X or Y registers will automatically skip the next sequential instruction in the event of an overflow, $X/Y(7:1) = 0$, or underflow, $X/Y(7:1) = 127$, respectively.
- 3) Immediate or "literal" operands are stored in ROM (instruction memory) as the third byte of the instruction. However, a single literal can be shared by multiple instructions by placing it in the Literal Pool. See discussion of Data Pools.
- 4) These instructions can utilize the Command Pool to "share" the second instruction byte. See discussion of Data Pools.
- 5) All instructions which load registers using the ROM address in the L-register also automatically increment the L register by 1.
- 6) The instructions which exchange a RAM operand with the accumulator are all two byte instructions. The second byte is actually the corresponding one byte "store" instruction. The first byte essentially "conditions" the store operation for an exchange. For this reason, the second byte of these instructions can be executed individually (by branching to it) if advantageous to the programmer.
- 7) The instructions, INCY and DECY are each made up of two individual instructions (XY and INXY or DEXY).
- 8) When a "skip" is executed, the next sequential instruction is ignored (not just the next byte). One cycle is required for each byte which is skipped. The only deviation from this rule occurs when the skip is followed by one of the composite instructions: "memory exchange" (Note 6), conditional branches (Note 9) or increment/decrement Y (Note 7). In these cases only one byte is skipped.
- 9) Conditional branch instructions require two or three bytes. The first byte is actually the corresponding "skip" instruction and the second (and third, if required) byte is a branch instruction. If this type instruction were preceded by another conditional SKIP instruction, only the first byte of the conditional branch would be skipped.
- 10) The DC and DCC instructions are provided to facilitate decimal arithmetic mechanizations in which two decimal digits are packed in one 8-bit byte.

The C flip-flop is used to save the "carry-out" from the 8-bit adder to facilitate software mechanizations for extended precision arithmetic. In addition, an intermediate state of the carry propagation logic; i. e., the carry

from the fourth bit position, is saved in the Q flip-flop for use by the DCC instruction. Instruction sequences for decimal addition and subtraction are illustrated below.

<u>Addition</u>			<u>Subtraction</u>		
LZI	PAGE	} Load Addresses of Operands	{	LZI	PAGE
LXI	OP1			LXI	OP1
LYI	OP2			LYI	OP2
RC		Initialize Carry/Borrow		SC	
LOOP	LNXY	} Add/Subtract two digits with carry/ borrow from previous result.	{	LOOP	LNXY
	DC				COM
	AC				AC
	DCC				DCC
	SNXY				SNXY
	B LOOP				B LOOP

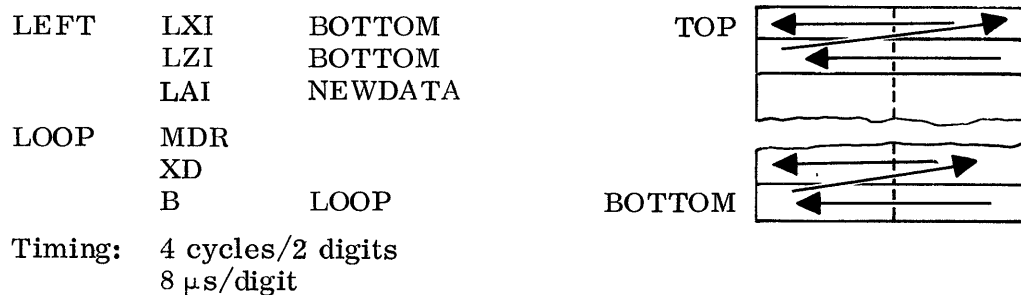
- 11) If the next instruction to be executed or skipped following a RT or RSK instruction is a branch, B or BDI, the branch tag (bit 7) must be set in the RT or RSK instruction. This is accomplished by coding RT\$ or RSK\$.
- 12) Branch instructions are provided in two formats. The one byte format modifies only the low order 7 bits of the P register, allowing a branch within the current 128 byte page. The two byte format modifies the entire 14 bits of the P register and allows a branch anywhere in memory (16K). The two formats are distinguished by bit 8 of the first byte (0=long format, 1=short format). If bit 8 of the second byte of the two byte format is zero (the BDI instruction), the interrupts are disabled. Note that the PPS-8 Assembler selects the optimum format automatically for all branch-type instruction. See discussion of Program Addressing.
- 13) See discussion of Data Stack.
- 14) The LAL and PSHL instructions provide a unique capability to reduce overhead in subroutines which require two full addresses* as part of their calling sequence. An efficient software mechanization for this type of subroutine is to use the Z, X registers for one address and the L register for the other, leaving the Y register for a block size parameter, for instance. The following example illustrates the use of the LAL and PSHL instructions in such a subroutine.

Calling Sequence	}	BL	SUBR	
		DW*	N	Block size
		DWA*	SOURCE	Two byte source address
		DWA*	DEST	Two byte destination address

*DW and DWA are assembler pseudo-op codes. DW is a one-byte data word. DWA is a two-byte data word address, assembled with the least-significant seven bits in the first byte and the most-significant seven bits in the second byte.

SUBR	LYL	Block size → Y
	LXL	Source address → Z, X
	LZL	
	LAL	Destination address → A, W
	LAL	
	PSHL	A, W → L, Return address → stack
	.	
	.	
	.	
	PØPL	Return address → L
	RT	Exit from subroutine

- 15) Branch and Link instructions (subroutine calls) are provided in two formats. The one byte format uses shared addresses from the Subroutine Entry Pool (see discussion of Data Pools) and allows a branch anywhere in memory (16K). The two byte format does not use the Subroutine Entry Pool. It allows a branch within the lower half (8K) of memory. Note that the PPS-8 Assembler selects the optimum format automatically for all branch-type instructions. See discussion of Program Addressing.
- 16) These instructions may not be followed immediately by a Branch (B) instruction. However, the PPS-8 Assembler will automatically insert a one byte "NOP" with a branch tag in this case.
- 17) All input/output instructions require two bytes. The second byte is gated to the I/O devices with the W/IO signal. See discussion of Input/Output.
- 18) The MDR and MDL instructions can be used to efficiently shift a string of BCD (decimal) digits as would be required for normalizing a number in packed decimal format, for instance. An example is shown below.



- 19) Comparison of X and Y registers are for least-significant seven bits only.

DATA STACK

The first 32 bytes of RAM (addresses 0-31) are organized as a pushdown stack to facilitate processing of external interrupts and subroutine "nesting." The stack operates on a last-in, first-out (LIFO) basis and instructions are provided to "push" (store) the contents of any register into the stack or "pop" (load) registers from the stack. In addition, return addresses are automatically saved in the stack when normal or interrupt subroutines are entered, and restored from the stack when subroutines are exited.*

The 5-bit S register is used as a stack pointer. It is incremented after each byte is pushed into the stack and decremented before each byte is popped from the stack. The pointer thus specifies the last byte popped from (or the next byte to be pushed into) the stack. If the stack pointer is popped to a value of 31 with a PØPX, PØPY, PØPZ or PØPA instructions, the next instruction will be skipped. This feature is provided so that the value of the stack pointer can be determined under program control for use in a power failure recovery routine.

* The current "level" of subroutine linkage is saved in the L (Link) register and the previous L register contents are pushed into the stack.

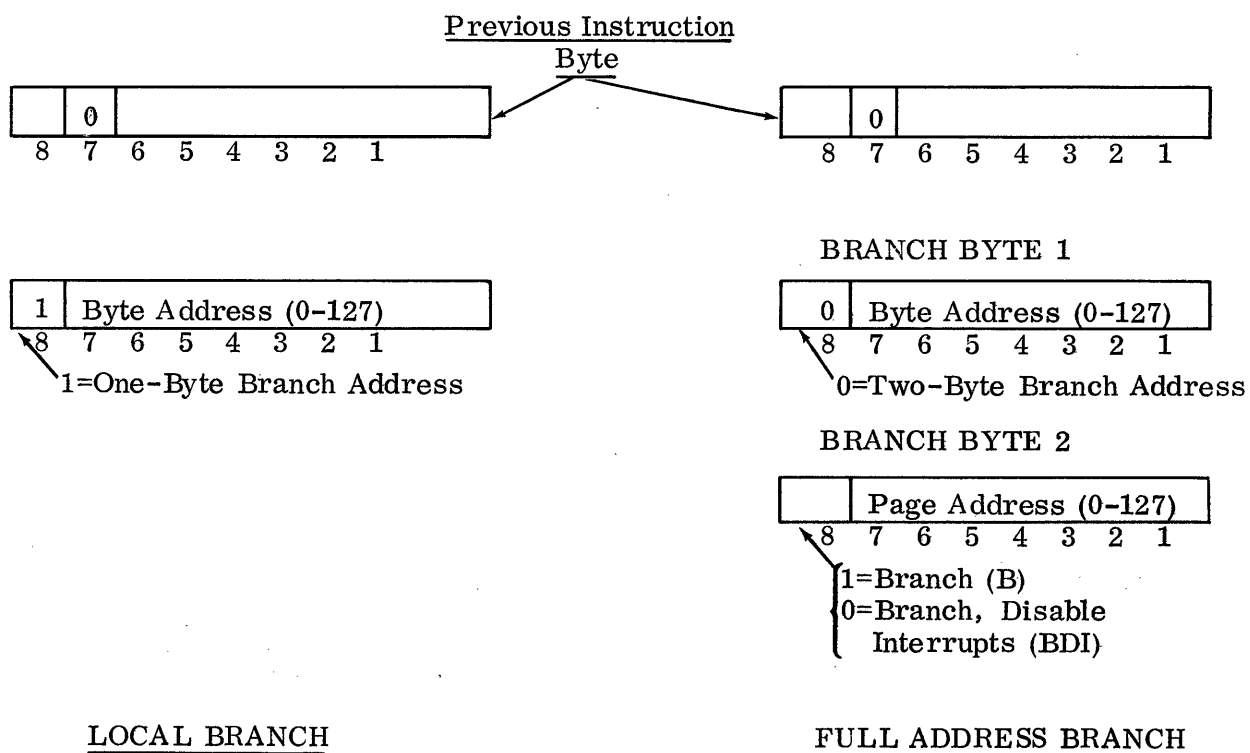
PROGRAM ADDRESSING

Programs for the PPS-8 system are stored in 16K bit (2048 byte) ROM devices.* Up to 16K bytes of ROM are directly addressable but the memory is functionally organized into 128-word instruction pages. The address of the current instruction byte is determined by the P-register (program counter), the least significant 7 bits of which are incremented automatically with each cycle. The instruction sequence can be modified under program control by the execution of branch type instructions.

BRANCHING

Two formats are provided with the unconditional Branch (B) instruction; the first requires only one instruction byte and specifies a "local" branch to an address within the current 128-byte instruction page. The second format requires two bytes and allows a direct branch to any of 16,384 ROM locations.

The ability to branch to a relatively large (128 byte) segment with a single 8-bit instruction is made possible by a unique design technique. Each instruction designates through the use of bit 7, whether the next instruction is to be interpreted as a Branch instruction. Likewise, one bit (bit 8) of the first byte of the Branch instruction designates whether a one or two byte Branch is required.



*It is also possible to store programs in RAM devices in some cases.

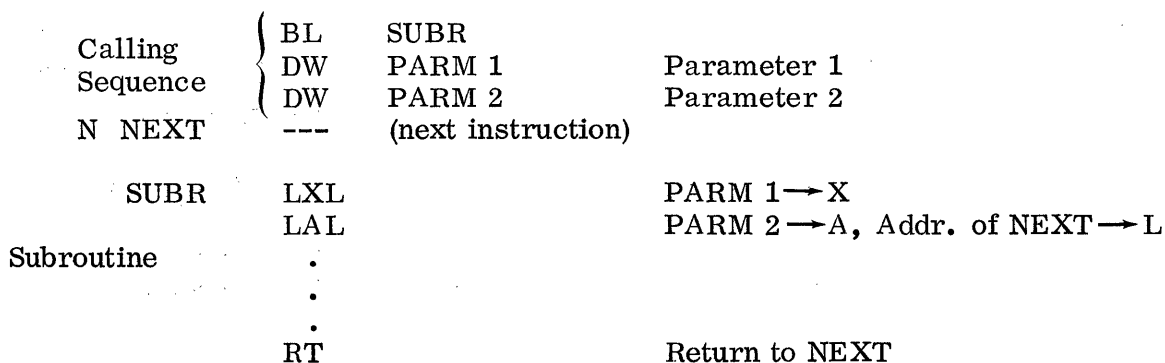
The necessity to insert the Bit 7 "branch tag" into instructions preceding a Branch is masked, i. e., made invisible to the programmer, by the PPS-8 Assembler. The Assembler will insert these tags automatically so the PPS-8 Programmer need not be aware of their existence. Additionally, the Assembler will automatically select and generate the proper type of Branch required by determining whether the destination address is within the current instruction page.

Note that the CPU logic automatically protects this "branch tag" status from inadvertent modification from a program interrupt.

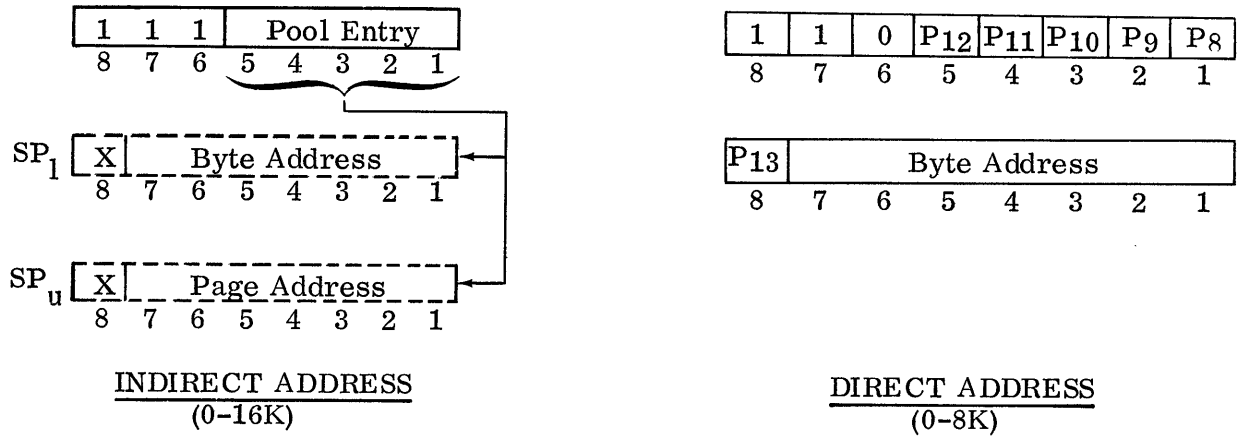
Conditional branches are provided in both the "skip-on-condition" and "branch-on-condition" formats. When the skip condition is satisfied, the PPS-8 CPU automatically skips the next full instruction (rather than simply the next instruction byte) regardless of whether it requires one, two, or three bytes. The branch-on-condition instructions are constructed as a skip-on-condition instruction (one byte) followed by an unconditional branch instruction (one or two bytes). As before, the Assembler will automatically choose the proper format for the branch instruction.

SUBROUTINE LINKAGE

PPS-8 instructions are provided to efficiently enter and exit subroutines and to pass arguments from the main program (ROM) to the subroutines. When a subroutine is called via a Branch and Link (BL) instruction, the return address, i. e., the incremented P-register, is automatically saved in the Link (L) register and previous contents of the L-register are pushed into the stack, thus, automatically accommodating up to 16 levels of nested subroutines. The Return (RT) instruction restores the P-register from the L-register and pops the previous L-register value from the stack. Fixed parameters to be passed to a subroutine are stored immediately following the BL instruction. The parameters can then be readily accessed by the subroutine using the "load thru link" instructions. A typical subroutine and calling sequence is illustrated below.



One and two byte formats are provided for the Branch and Link (BL) instruction. These formats are shown below.



The one-byte format uses shared addresses from the Subroutine Entry Pool which is described in a later section. Note again, that the PPS-8 Assembler will automatically select the required, i. e., the most efficient format for each BL instruction coded in a program.

OPERAND ADDRESSING

The PPS-8 features a unique design with respect to addressing of program data (operands). This design takes advantage of the organizational characteristic of program data to achieve high processor throughput together with efficient ROM utilization.

A high percentage of the data manipulated by a microcomputer program is (or could be) organized as sequential lists, tables, strings, etc. (the sequence of characters from an I/O device, the digits of a decimal number, the data file for a display, etc.). Since processing data lists or blocks of data implies repetitive program "loops," the efficiency (or lack of efficiency) of this processing in terms of execution speed is multiplied relative to other, less repetitive portions of the program. Therefore, the PPS-8 architecture optimizes this type of processing in terms of execution speed. Non-repetitive portions of a PPS-8 program which require proportionately less of the total processor throughput can be optimized for ROM utilization (with some penalty in execution speed) by taking advantage of the one-byte subroutine call (BL) and the data pools.

RAM OPERANDS

All dynamically variable data for PPS-8 programs are stored in 2048 bit (256 byte) RAM devices. Up to 16K bytes of RAM are directly addressable.

The byte address (lower 7 address bits) of RAM operands is specified by the X-register. The page address (upper 7 address bits) of RAM operands is specified by the Z-register or is set equal to page zero depending on bit 8 of the byte address register (1=Z-register, 0=page 0).

Though a single register pair (Z, X) is used for all RAM operand addresses, the PPS-8 instruction repertoire provides considerable power and efficiency for operand addressing. Using the ability to exchange the Y-register and the L-register with the Z, X registers, two full (14 bit) addresses and one byte (7 bit) address are directly available—selected in conjunction with normal data transfer (load/store) operations. A single, one-byte instruction can perform all of the following functions in one cycle:

- 1) Transfer an 8-bit operand from the accumulator to the RAM or vice versa.
- 2) Increment or decrement the operand address.
- 3) Test for a page boundary and skip if one is encountered.
- 4) Test for completion of a program loop (byte addresses equal) and skip if completed.
- 5) Select another byte address to be used for a subsequent RAM operand, i. e., exchange X with Y or Z, X and L.

This organization is particularly efficient when dealing with blocks of data; i. e., program loops. This efficiency is demonstrated in the following examples:

EXAMPLE 1: Move a Block of Data

a) One data block must be in page 0, aligned on page boundary

```

LZI    SOURCE
LXI    SOURCE
LYI    DEST
LOOP LNXY
      SNXY
      B      LOOP } 3 cycles/byte
                          (up to 128 bytes)

```

b) Data blocks can be anywhere in memory

```

LZI    DEST
LXI    DEST
XL
LZI    SOURCE
LXI    SOURCE
LYI    END
LOOP LNXL
      SNCX
      B      LOOP } 3 cycles/byte

```

EXAMPLE 2: Clear or Initialize a Block of Data

a) Block must be in page 0, aligned on page boundary

```

LXI    BLOCK
LAI    0 (or initialization constant)
LOOP SN
      B      LOOP } 2 cycles/byte

```

b) Data block can be anywhere in memory

```

LZI    BLOCK
LXI    BLOCK
LYI    COUNT
LAI    0 (or initialization constant)
LOOP SNXY
      DEXY
      B      LOOP } 3 cycles/byte

```

EXAMPLE 3: Decimal Addition

a) Augend in page 0 (aligned); Addend anywhere

```

LZI    ADDEND
LXI    ADDEND
LYI    AUGEND
RC
LOOP LNXY
      DC
      AC
      DCC
      SNXY
      B      LOOP } 6 cycles/2 digits

```

b) Augend and Addend anywhere, not aligned on page boundary

```

LZI    AUGEND
LXI    AUGEND
XL
LZI    ADDEND
LXI    ADDEND
LYI    END
RC
LOOP LNXL
      DC
      AC
      DCC
      SNCX
      B      LOOP } 6 cycles/2 digits

```

ROM OPERANDS

The PPS-8 provides the capability to store constant data in ROM devices together with the PPS-8 instructions. This type of data is important for two reasons:

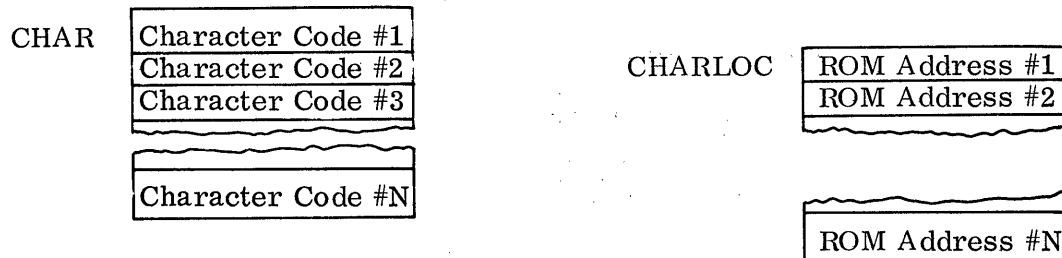
- 1) ROM storage is significantly less expensive than RAM storage, and
- 2) All data which is to be permanently retained (preserved during power-off) must be stored in ROM devices.

Therefore, considerable flexibility is provided for storing and accessing program data (operands) from ROM. In particular, the PPS-8 is designed to accommodate three types of ROM operands:

- 1) Immediate (literal) Operands. This type of operand is stored either in-line with the instruction sequence or in the Literal Pool. These operands are accessed by "immediate" type instructions (LAI, LXI, AISK, ANI, etc.).
- 2) Subroutine Parameters. This type of ROM operand is stored in-line with the instruction sequence immediately following a subroutine call (BL instruction) and represents data to be "passed" to a subroutine. These operands are accessed by "load thru link" type instructions (LAL, LXL, LYL, etc.). See Program Addressing, Subroutine Linkage.
- 3) Constant Tables. This type of operand need not be stored in-line with the instruction sequence and can be placed wherever convenient for the programmer. Tables of constants, such as character codes or data look-up tables, can be stored in this manner and accessed with the "load thru link" type instructions (LAL, LXL, LYL, etc.). Note that a table of branch addresses can be constructed in ROM and accessed using the subroutine return instruction (RT). The following example illustrates this capability.

Problem: One of N possible control characters has been received from some input device. Select the appropriate processing routine corresponding to the particular character.

Sample Program: Two tables are stored in ROM, aligned with respect to page boundaries.



First, the character is "looked up" in the character code table, CHAR, which would contain all functionally legitimate characters. Assume the character to be processed is initially stored in location TEMP in RAM page 0.

LZI	CHAR	}	Load address of character table into L.
LXI	CHAR		
XL			
LXI	TEMP		Address of character → X
LOOP LAL		}	Get character from CHAR } 5 cycles/ Compare characters } character
BNE	LOOP		

The index of the character (its position in CHAR) is now in the L-register.* This value is used as an index to select the address of the processing routine from CHARLOC. The Return, RT, instruction is then used to branch to the routine.

LAI	CHARLOC, U	Page address of CHARLOC into L
LLA		
LXL		Address from CHARLOC → X
LZI	PROCESS	Page address → Z
XL		Routine Address → L
RT		Branch to processing routine

* The value in L is actually the character's index plus one. Therefore, the CHARLOC table is displaced by one relative to the CHAR table.

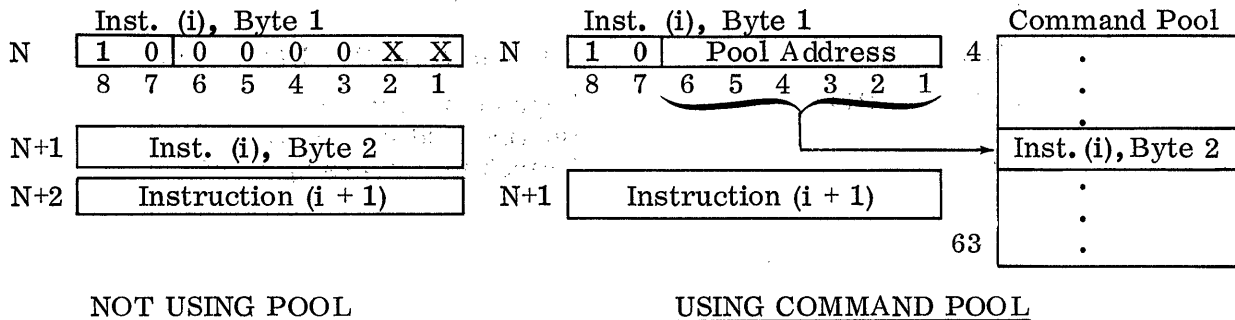
DATA POOLS

The PPS-8 has a unique feature designed to minimize instruction storage for multi-byte instructions. This feature allows certain data to be "shared" by multiple instructions through access to common data pools stored in a dedicated portion of ROM. Three separate pools are provided: the Command Pool, the Literal Pool, and the Subroutine Entry Pool. Figure 3-2 shows the location of these pools in ROM.

COMMAND POOL

The Command Pool occupies the first 64 bytes of page 0 (decimal locations 0-63). The first four bytes of this pool are dedicated to power-on initialization; the remaining 60 bytes are available for assignment by the PPS-8 Programmer.

Most two and three-byte instruction types can utilize the Command Pool for storage of the second instruction byte. The format of these instructions is pictured below.



The two-byte instructions of this type are LX, LY, LZ, LXL, LYL, LZL, LAL, PSX, PSY, PSZ, PSH, POPX, POPY, POPZ, POPA, SB, and RB. All of these instruction types could be included in the pool by using 31 bytes (the "bit" designator in the SB and RB instructions is part of the second instruction byte so these instructions require 16 Command Pool bytes to cover all possibilities). In this case, these instructions would require only one byte/instruction no matter how many of them were used in a given program.

The three-byte instructions of this type are LAI, LXI, LYI, LZI, AISK, and ANI. The third byte of these instructions is an immediate (literal) operand which can be stored either in the Literal Pool or in-line with the Instruction sequence. Bits 1-6 of the second instruction byte specify where the literal operand is stored. Therefore, using six bytes of the Command Pool, the six instruction types can be included assuming an in-line literal operand. This would effectively reduce these to two-byte instructions. However, if both the Command Pool and Literal Pool are to be used for the same three-byte instruction (reducing it to one unique byte), a separate Command Pool byte must be used for each instruction/literal operand combination. The following section discusses this further.

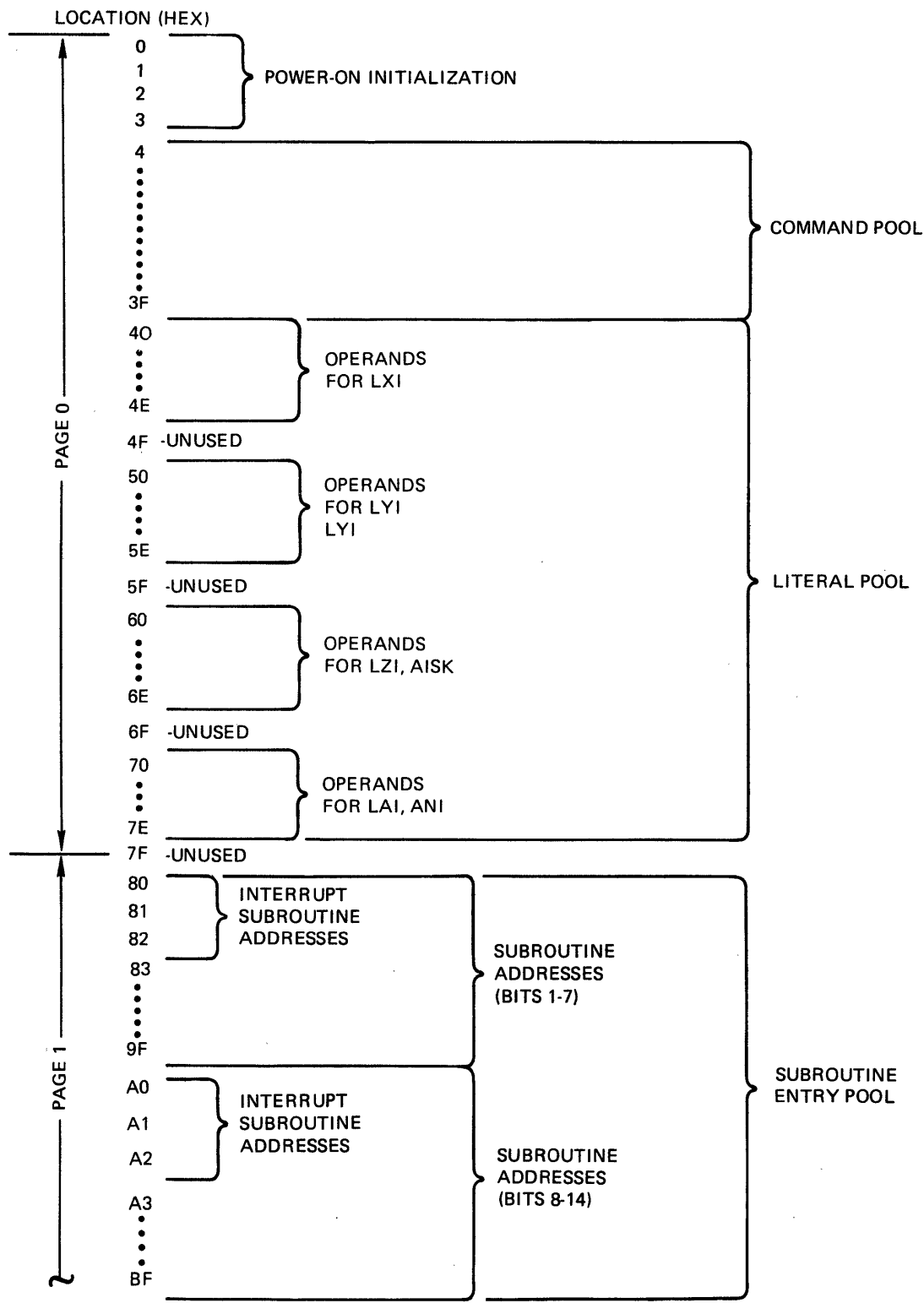


Figure 3-2. Data Pools

LITERAL POOL

The Literal Pool occupies 64 words of ROM page 0, decimal locations 64-127. However, due to addressing restrictions, four of these locations (79, 95, 111, and 127) cannot be used by the PPS-8 Programmer.

Six instruction types, LAI, LXI, LYI, LZI, AISK, and ANI, can use the Literal Pool for storage of the third instruction byte; i. e., the immediate (literal) operand. The Literal Pool is subdivided into four 15-byte segments. Each segment corresponds to a particular instruction type(s). See Figure 3-2. Note that these instruction types can also use the Command Pool for storage of the second instruction byte. The format of these instructions is shown in Figure 3-3.

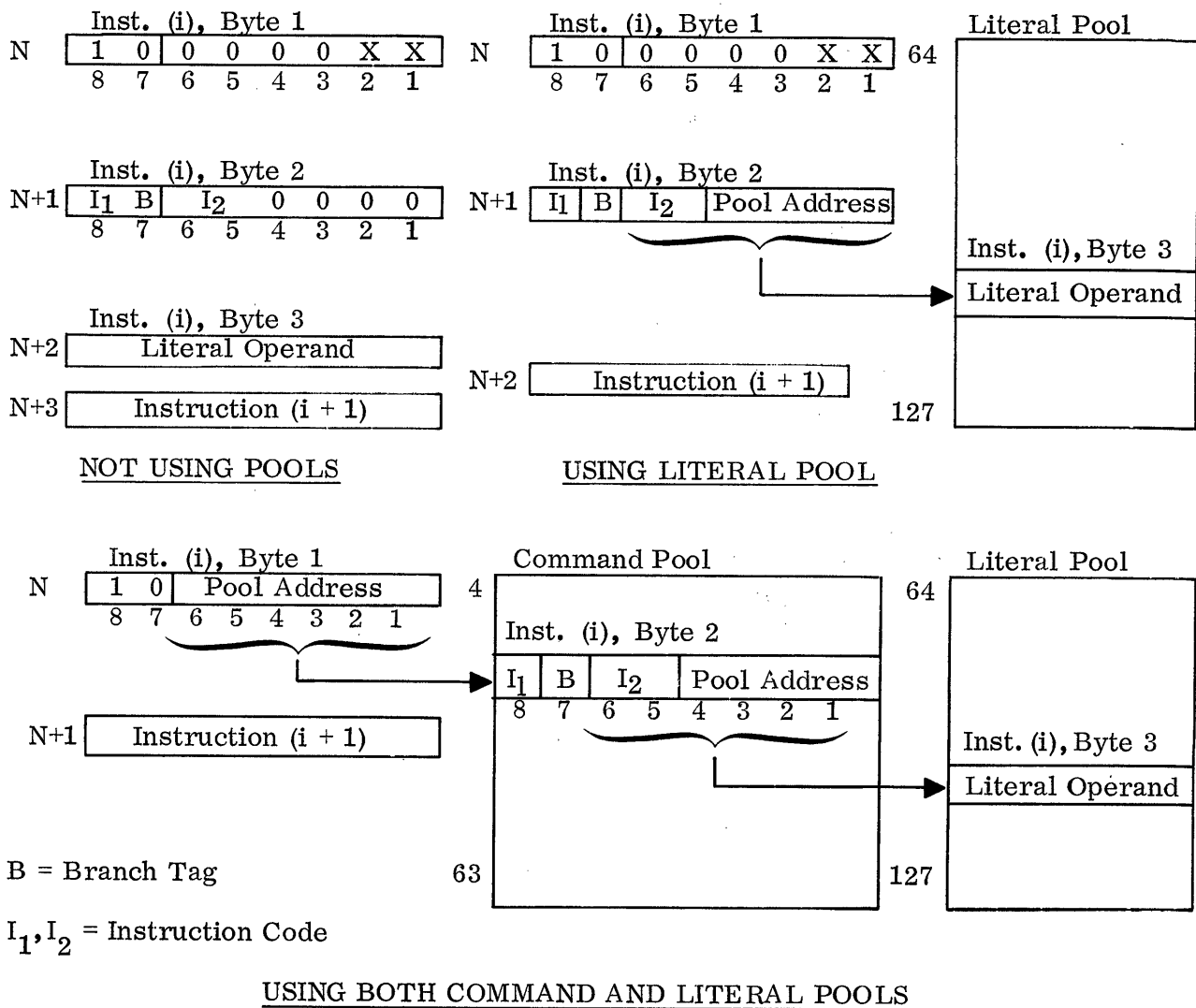
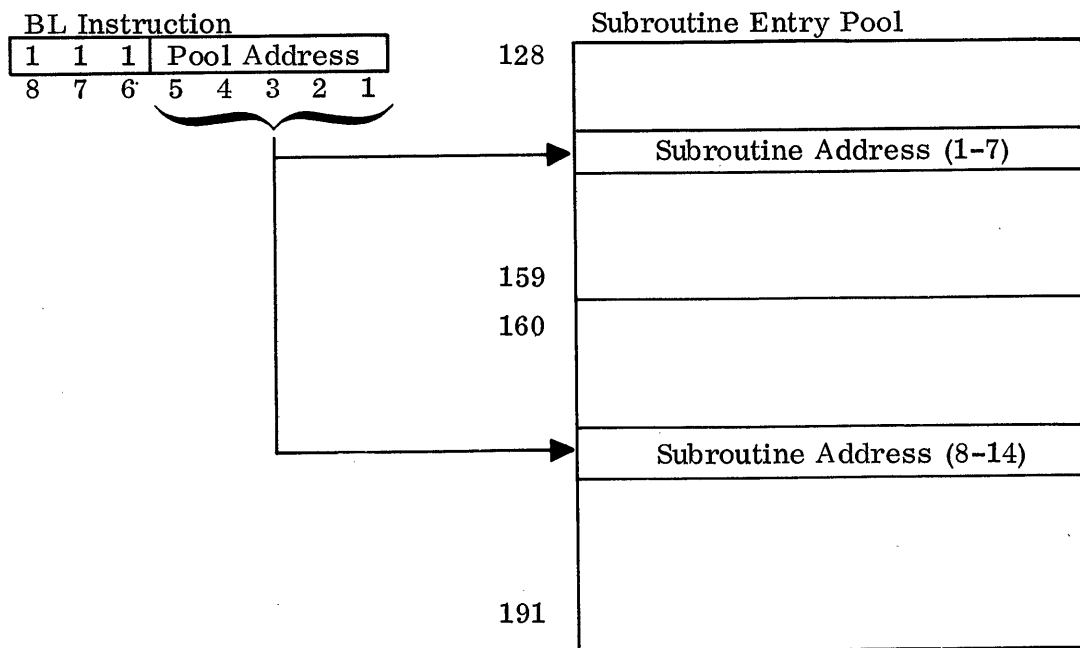


Figure 3-3. Command and Literal Pool Access

SUBROUTINE ENTRY POOL

The Subroutine Entry Pool occupies the first 64 bytes of page 1, decimal locations 128-191. As shown in Figure 3-2, this pool is divided into two parts. The first 32 bytes specify the lower seven bits of 32 subroutine entry addresses; the second 32 bytes specify the corresponding upper seven address bits.

This pool provides for storage of up to 32 subroutine entry addresses, the first three of which are reserved for the three interrupt processing subroutines. This pool is accessed by the one byte format of the Branch and Link (BL) instruction and effectively provides full-address (16K) subroutine calls with a single 8-bit instruction. The format of this instruction is shown below.



POOL UTILIZATION

The PPS-8 Assembler provides extensive capability to enable the programmer to make effective use of the data pool feature of the PPS-8.

Assignment of data to the pools is done by the programmer using the POOL pseudo-operation. A "standard" pool assignment is available if desired. The Assembler provides statistics on the frequency of usage of the instructions which can access the pools so that the programmer may make the most efficient assignments.

Access to the data in the pools is completely automatic. During the assembly process, the PPS-8 Assembler searches the shared data pools for each instruction for which pool access is possible. If the appropriate data is available in the pool, the Assembler automatically uses the shared data format for the particular instruction.

Thus, effective use can be made of this powerful feature with little additional effort on the part of the programmer.

INTERRUPTS

The PPS-8 system provides a multi-level priority interrupt structure. The CPU has three interrupt input signals (INT0, INT1, INT2) which have a fixed priority relationship with respect to resolving simultaneous requests. Interrupt 0 has highest priority and Interrupt 2 has lowest priority. Interrupts 0 and 1 are designed for dedicated special purpose functions, nominally power failure detection and a real-time clock. Interrupt 2 is designed to operate as a single request input for a multi-source, priority system constructed "externally" in the input/output devices.

Interrupt 0. This is the highest priority interrupt and would normally be dedicated to a function requiring rapid response, such as power failure detection. This level cannot be disabled under program control and will be honored at the completion of the current instruction. * The program interrupt is triggered by a false-to-true transition of the INT0 input signal.

Interrupt 1. This interrupt has priority over Interrupt 2 if they occur simultaneously, and is honored at the completion of the current instruction provided that:
1) Interrupt 0 is not pending, and 2) the interrupts are enabled. * This interrupt is triggered by a false-to-true transition of the INT1 input signal. This interrupt is intended as a real-time clock.

Interrupt 2. This is the primary PPS-8 interrupt and is used for servicing all I/O device interrupt requests. It is honored at the completion of the current instruction provided that 1) Interrupt 0 or 1 is not pending and 2) the interrupts are enabled. * The interrupt is triggered by a true state on the INT2 input signal. An acknowledge signal (ACKO) is provided and is designed to allow a multi-level priority interrupt structure to be constructed external to the CPU. Logic for this interrupt structure is included in the I/O devices that require access to the CPU interrupt, e.g., the PDC and SDC. The system inter-connection for the interrupt structure is shown in Figure 1-1 and is described below.

- 1) Each device; interrupt source, to be included in the interrupt structure has an Interrupt Request (INT2) output signal. The INT2 output signals from all the devices are "OR-tied" to form the INT2 input to the CPU.
- 2) In addition, each device has an Acknowledge Input (ACKI) and an Acknowledge Output (ACKO). These signals are interconnected to form a priority "chain." The ACKO signal from one device is connected to the ACKI signal of the next lower priority device. The ACKI signal of the highest priority device is connected to the Interrupt Acknowledge (ACKO) output of the CPU.
- 3) When a device wants interrupt service, it sets its INT2 output true. Any number of devices can simultaneously request interrupt service.

*Interrupts are not honored under the following conditions: 1) just prior to a branch (B) instruction, 2) immediately following a skip-type instruction or add instruction, and 3) (INT1 and INT2 only) during a direct memory access (DMA) operation.

- 4) When an interrupt request is honored, the CPU transmits a 1-cycle pulse on the ACKO output. This pulse is propagated, 1-cycle device, down the priority chain until it reaches the first device; i. e., the highest priority device, currently requesting service. This device does not propagate the acknowledge signal any further, thus breaking the priority chain.
- 5) After a time delay sufficient to allow the acknowledge to be propagated completely through the priority chain, the CPU program executes a Read Interrupt Status (RIS) command (a specific command code in the "all-call" format). The requesting device which has received the acknowledge pulse transmits its device address (1-15) and interrupt status information over the I/D bus to the CPU. This address is decoded under CPU software control and used to identify the proper interrupt processing routine.
- 6) Receipt of the Device Address Request is also used by the I/O device to reset its INT2 output (however, the INT2 input to the CPU may still be true due to other requests.)

INTERRUPT PROCESSING

Processing a PPS-8 interrupt involves the following steps:

- 1) The CPU "honors" the interrupt request by disabling the interrupt system and executing a Branch and Link (BL) instruction. The BL instruction is "created" by the CPU logic and requires no additional data other than the interrupt request signal. The operand; i. e., subroutine entry address, for the BL instruction is taken from one of the first three locations in the Subroutine Entry Pool depending on which interrupt is being processed.
- 2) If Interrupt 2 is being honored, the Interrupt Acknowledge (ACKO) pulse is automatically transmitted.
- 3) The interrupt processing subroutine would normally first save the contents of any CPU registers necessary in processing the interrupt by pushing them into the data stack. Note that the acknowledge pulse for Interrupt 2 would be propagated down the priority chain while the CPU is performing this function.
- 4) If Interrupt 2 is being processed, the interrupt processing subroutine would execute a Read Interrupt Status (RIS) instruction to determine which I/O device caused the interrupt.
- 5) At the completion of the interrupt processing routine, the CPU registers would be restored by popping their original contents from the stack.
- 6) Execution of the interrupted program can be resumed by executing an RTI instruction which also re-enables the interrupt system.

INPUT/OUTPUT

The PPS-8 system is designed to interface with the external environment via both general purpose and special purpose input/output devices. The I/O devices communicate with the CPU or RAM over the system data bus, I/D1 through 8.

Two basic types of input/output commands are available in the PPS-8 CPU. The I/O instructions are all two byte, two cycle instructions. The second instruction byte is interpreted by the I/O devices. Presence of this byte on the I/D bus is indicated by the W/IO signal set by the CPU. The specific function of a particular I/O command is dependent on the type of I/O device being addressed. The two I/O command types are described below.

DIGIT INPUT/OUTPUT (IO4)

0	1	0	0	1	1	1	1
8	7	6	5	4	3	2	1

 Byte 1

Device Address				Command			
8	7	6	5	4	3	2	1

 Byte 2

This command is provided for compatibility between the PPS-8 system and the PPS-4 input/output devices. One of 16 I/O devices can be addressed and up to 16 unique commands can be selected. A bidirectional transfer of four bit digits is performed. Bits 5-8 of the PPS-8 accumulator are driven over the I/D bus to the I/O device and bits 1-4 of the accumulator are loaded with data driven from the I/O device.

BYTE INPUT/OUTPUT (IN, OUT)

0	1	0	0	1	1	1	0
8	7	6	5	4	3	2	1

 Byte 1

Device Address				I/O	Command		
8	7	6	5	4	3	2	1

 Byte 2

This is the primary I/O command in the PPS-8 system. It provides for addressing one of 16 devices, specifying one of eight command options, and transferring one byte of data between the accumulator and the I/O device. Bit 4 of the second instruction byte determines whether the CPU is to transmit or receive the data.

A device address of 0 is used as an "all-call" command to which all PPS-8 I/O devices react. (One of the command options in this format is the Read Interrupt Status command mentioned previously in the discussion of interrupt processing.)