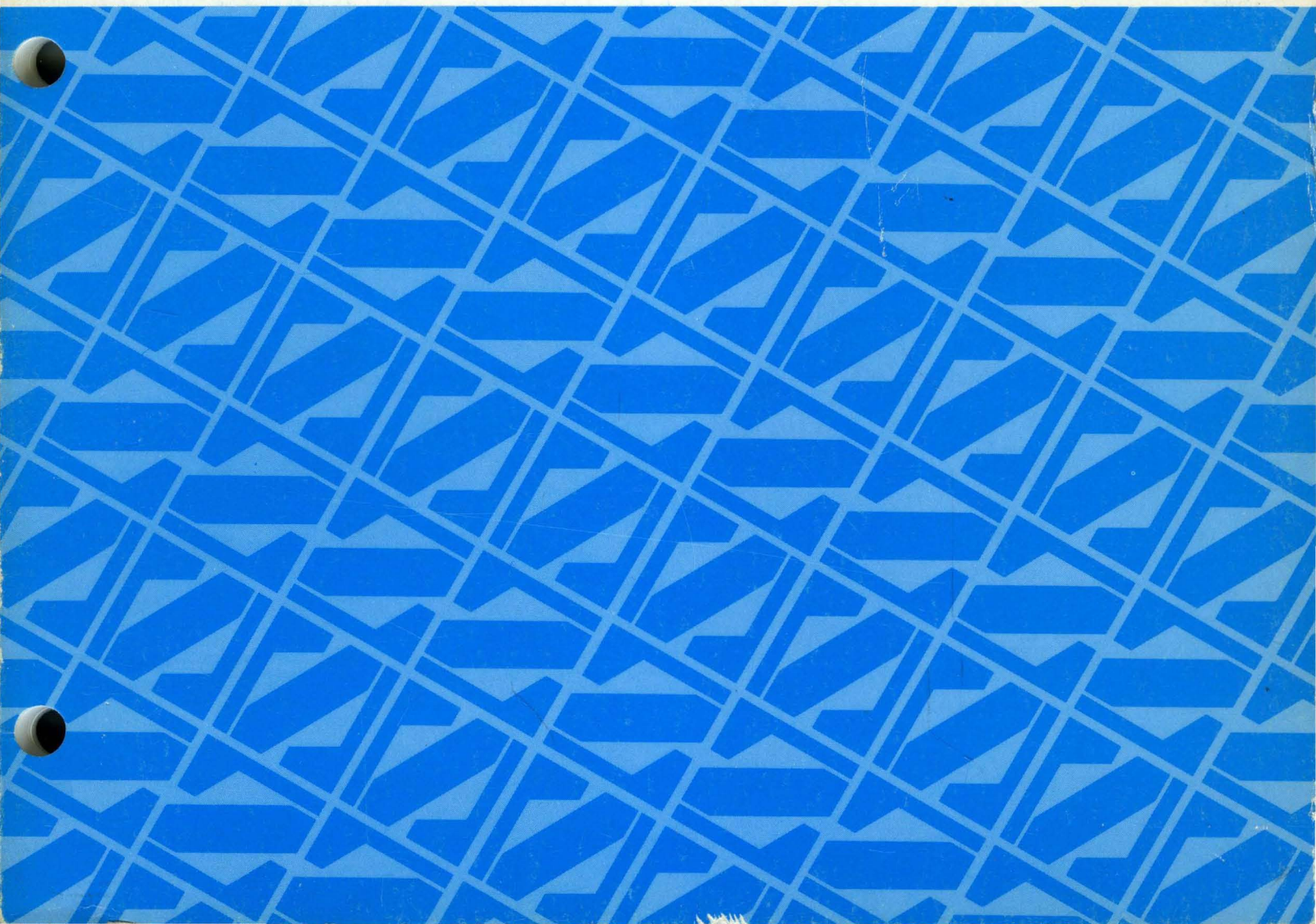
National Semiconductor
Order No. IMP-8C/932A
Pub. No. 4200032A

IMP-8C
Application Manual



Order Number IMP-8C/932A
Publication Number 4200032A

INTEGRATED MICROPROCESSOR-8C

IMP-8C
APPLICATION MANUAL

March 1974

2900 Semiconductor Drive
Santa Clara, California 95051

PREFACE

The IMP-8C Application Manual provides information pertaining to operational features, functional and logic curcuit descriptions, and input/output interfacing. The material provided is intended to be sufficiently informative to prepare the user to adapt the IMP-8C Microprocessor to his particular application.

The material in this manual is subject to change without notice. Circuit details and other data presented in the engineering documentation supplied with the IMP-8C Microprocessor take precedence over the information presented in this manual.

Copies of this publication and other National Semiconductor publications may be obtained from the sales offices listed on the back cover.

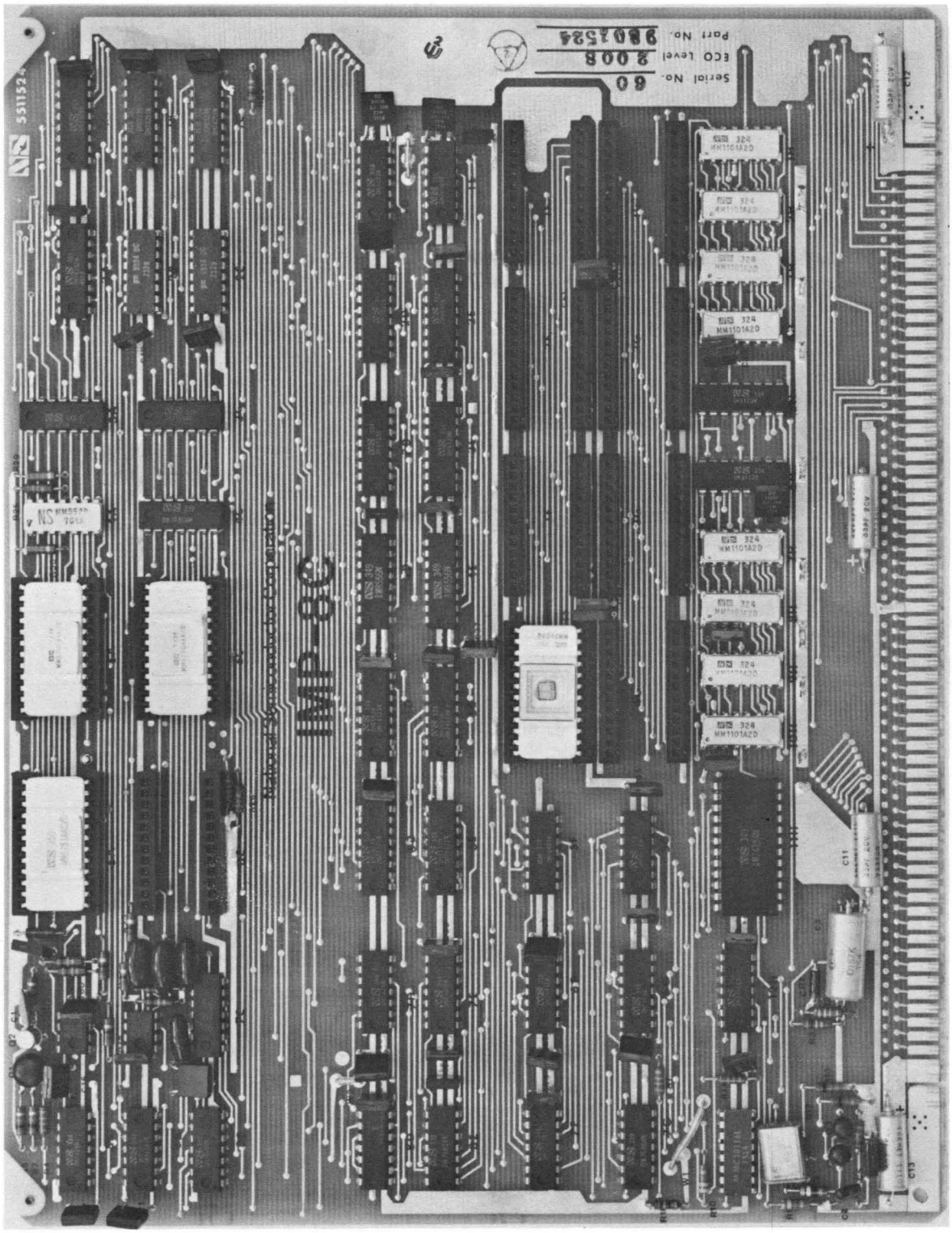
CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

TABLES



IMP-8C Microprocessor Card

Chapter 1

GENERAL INFORMATION

1.1 IMP-8C GENERAL DESCRIPTION

The IMP-8C is an 8-bit, byte-oriented, general-purpose, parallel processor using standard TTL and MOS/LSI components. The IMP-8C is packaged on an 8-1/2- by 11-inch printed wiring card, which is shown on the facing page.

The IMP-8C is configured around National Semiconductor MOS/LSI devices which consist of CROM (Control Read-Only Memory) and RALUs (Register and Arithmetic Logic Units) which form the CPU (Central Processing Unit).

The IMP-8C/200 is the first of a series of IMP-8C processors. Operation of the processors in the IMP-8C series is similar. The main difference between the processors in the IMP-8C series is the CROM-resident microprogram.

As shown in figure 1-1, the IMP-8C provides input and output interface to peripheral units and add-on memory. Data from the user's peripheral devices and add-on memory are received via the 8-bit Peripheral Data Lines and the 8-bit External Memory Data Lines, respectively. Data to both the peripheral units and the add-on memory is sent via the 8-bit Data Out Bus. A 16-bit Address Bus provides addressing for both the peripheral units and the add-on memory. Additionally, the IMP-8C provides timing and control signals for synchronization of the external devices. Control, addressing, and data signals between the IMP-8C and the external devices are brought out at the card-edge connector.

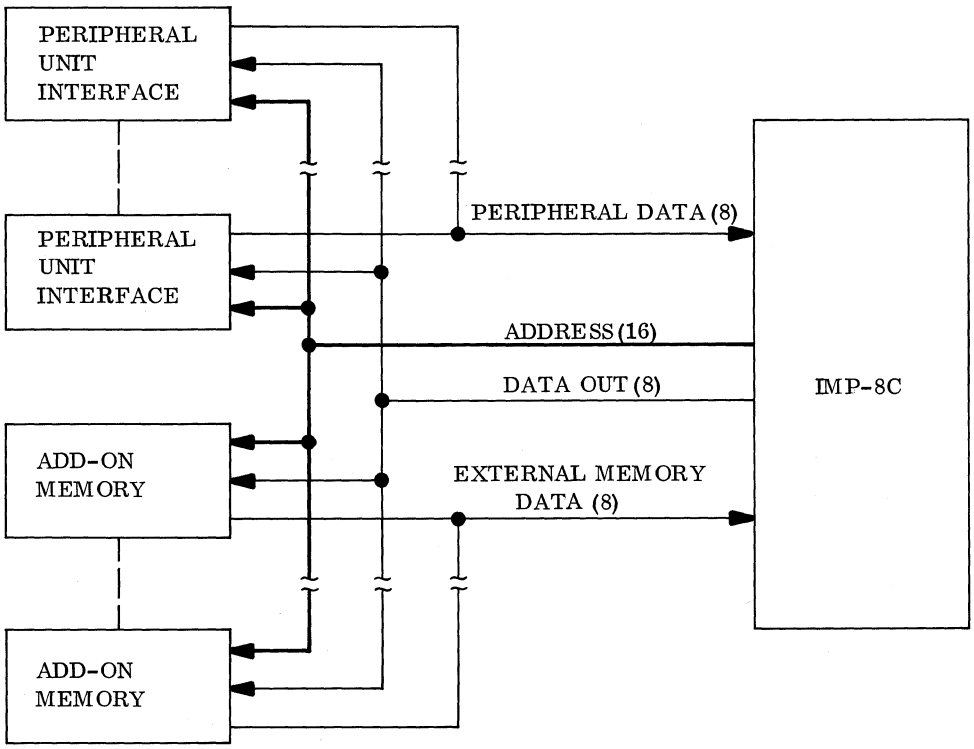


Figure 1-1. Overall System Block Diagram, IMP-8C

Chapter 2 contains a functional description of the IMP-8C. The instruction set is explained in chapter 3. Chapter 4 presents a circuit-level description. IMP-8C input/output and interrupt system operations are explained in chapters 5 and 6, respectively.

1.2 IMP-8C/200 OPERATIONAL FEATURES

Word Length	8 bits (1 byte)
Instruction Set	38 (implemented by CPU-resident microprogram)
Arithmetic	Parallel, binary, fixed-point
Memory	256 8-bit bytes of semiconductor read/write memory Sockets for 2048 8-bit bytes of semiconductor read-only memory
Addressing	Page-oriented, 256 bytes/page Expandable to 65,536 8-bit bytes • Direct, base or current page reference • Indexed, relative to Accumulator 3 • Indirect, base or current page reference
Cycle Time	1.4 microseconds
Input/Output and Control	• Two 8-bit data input ports • 8-bit Data Output Bus • 16-bit Address Bus • 4 general-purpose control flags • 1 general-purpose jump-condition input • 1 general interrupt input
Interface Levels	Standard TTL
Temperature	• Operating: 0° to 70° C • Storage: -20° to 100° C
Humidity	Maximum of 90% relative humidity without condensation

1.3 POWER REQUIREMENTS

The IMP-8C contains standard TTL integrated circuits and MOS/LSI components. These circuits receive +5 volts and -12 volts from regulated dc supplies. The On-Card Read/Write Memory, if used, requires a -9-volt supply.

Voltage and Current Requirements (Maximum at 25° C)	+5 volts ±5% at 2.5 Amperes -12 volts ±5% at 0.60 Ampere -9 volts ±5% at 0.30 Ampere

1.4 IMP-8C AND SUPPORT OPTIONS

Hardware, firmware, and software available with the IMP-8C are listed in table 1-1.

Table 1-1. IMP-8C and Support Options

Item	Order Number
IMP-8C/200	
8-1/2- by 11-inch printed wiring card including 256 bytes of semiconductor read/write memory	IMP-8C/200
• Applications Manual • Programming and Assembler Manual • Schematic Drawings • Assembly Drawing • List of Materials	
Hardware options:	
8K by 8 Read/Write Memory Storage Card	IMP-8P/008
Memory Timing and Interface Control Card	IMP-8P/008T
6-card Connector Panel:	
Six 144-pin connectors mounted with card guides	IMP-00H/880
3-card Connector Panel:	
Three 144-pin connectors (spaced for use with prototyping cards with wire-wrap sockets)	IMP-00H/881
Single Card Connector	IMP-00H/882
Extender Card	IMP-00H/890
Prototyping Card:	
64 wire-wrap socket positions; room for 64 16-pin or 58 16-pin, and 4 24-pin DIPs	IMP-00H/891
Prototyping Card:	
90 wire-wrap socket positions, any combination of 16- and 24-pin (24-pin uses two positions)	IMP-00H/892
Firmware options:	
ROM Programs (8 ROMs):	
IMP-8C/200 CPU Diagnostics	IMP-8F/501
Software[1] options:	
IMP-8 Cross Assembler	IMP-8S/900
For use on large-scale computer • Source Deck • Assembler Manual • Listing • Installation of IMP-16 Assembler on System 360/370 Manual • Tymshare Users Manual	IMP-16S/118Y

1 - Assembler is available on computer utility from Tymshare, Inc. Contact local Tymshare office. Purchase of assembler source deck includes all items listed. Individual items are available separately.

Chapter 2

IMP-8C FUNCTIONAL DESCRIPTION

2.1 FUNCTIONAL UNITS AND DATA FLOW

The IMP-8C comprises the following major functional units:

- Central Processing Unit (CPU)
- Clock Generators
- Input Multiplexer
- Data Buffer
- Data Multiplexer
- Control Flags
- Conditional Jump Multiplexer
- On-Card Memory
- Address Latches
- Page Counter

A simplified block diagram showing the data flow among the major functional units of the IMP-8C is given in figure 2-1.

The Clock Generator provides the MOS clock drivers and Central Processing Unit (CPU) timing signals. The system clocks are available at the edge connector for synchronization of peripheral units with the IMP-8C.

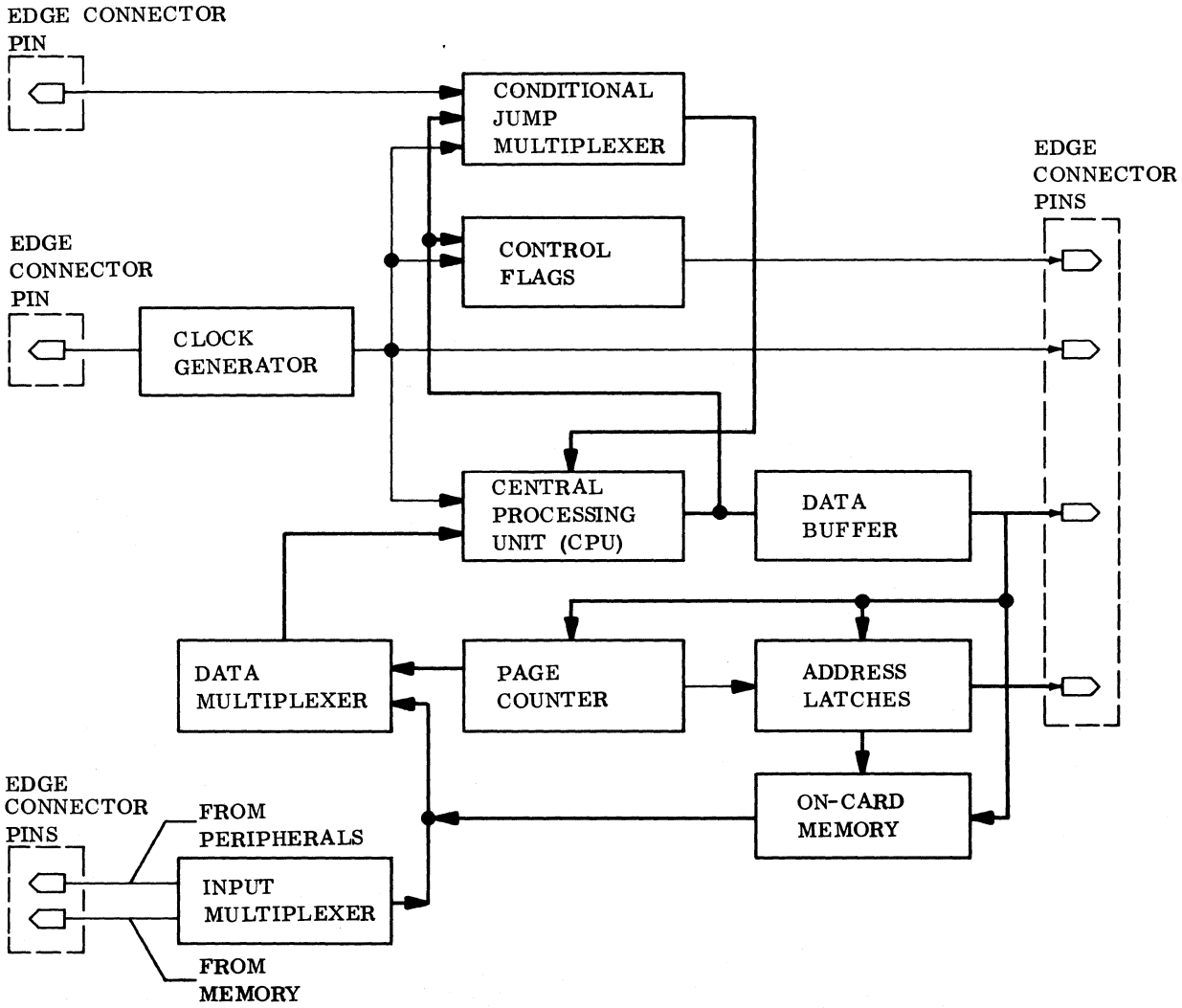


Figure 2-1. IMP-8C Simplified Block Diagram

The IMP-8C/200 CPU contains one CROM (Control Read-Only Memory) IMP-8A/520 and two RALUs IMP-00A/520 (Register and Arithmetic Logic Units). The 3-device kit is available as the IMP-8A/500. Each RALU handles 4 bits and the 8-bit byte is formed by the two RALUs. A 4-bit Control Bus is used by the CROM to communicate control information to the RALUs. The CROM controls the operations of the CPU. This control is effected by the microprogram stored in the read-only memory of the CROM. The actual processing of the data is accomplished in the two 4-bit RALUs under control of CROM-resident microprograms.

External to the MOS/LSI CPU, but within the IMP-8C, are the control flags and external interfacing circuits. These control flags are in addition to the status flags internal to the RALU and are discussed in chapter 3 under the control flag instructions.

Conditional branches are selected by the Conditional Jump Multiplexer. Branching and the branch conditions are discussed in chapter 3 under the Branch-On-Condition (BOC) instruction.

Data from the user peripheral devices and add-on memory are received at two 8-bit ports by the Input Multiplexer. The selected data byte is routed to the CPU via the Data Multiplexer. Data bytes from the On-Card Memory are also routed to the CPU via the Data Multiplexer.

The Data Buffer provides the transfer of output data from the CPU to the peripheral devices or add-on memory. The Data Buffer also transfers addressing information for storage by the Address Latches and the Page Counter or for use by the On-Card Memory. Addresses stored in the Address Latches are available for use by the user peripheral device interface or add-on memory. The address store in the Address Latches may also be used to address the On-Card Memory.

The basic IMP-8C On-Card Memory consists of 256 bytes of read/write memory and sockets for 2048 bytes of read-only memory. The memory may be expanded to 65,536 bytes by the addition of external add-on memory.

The following paragraphs provide a more detailed description of the functional operation and timing of the CPU and the IMP-8C. Chapter 4 provides a circuit-level description of the IMP-8C.

2.2 REGISTER AND ARITHMETIC LOGIC UNITS (RALU)

Two 4-bit RALUs constitute the arithmetic section of the CPU. As shown in figure 2-2, the arithmetic section includes the following major units:

- Input/Output Multiplexer
- Last-In/First-Out Stack (LIFOS)
- 8-bit Status Flag Register
- 8-bit Program Counter (PC1)
- Memory Data Register (MDR)
- Memory Address Register (MAR)
- 4 accumulators (AC0, AC1, AC2, AC3)
- Arithmetic and Logic Unit (ALU)
- Shifter
- 3 internal data buses (A, B, R)

All RALU operations are controlled by the CROM.

2.2.1 Internal Buses

Three buses are internal to the RALU. These buses are described below.

A-bus (Operand Bus). The contents of all RALU registers may be loaded onto the A-bus; data from the top of the LIFOS and from the RALU Status Flag Register (combined as an 8-bit byte) may be loaded as operands. Data loaded on the A-bus can be complemented, gated through the ALU and Shifter to the R-bus, and out of the RALU to the data bus through the Input/Output Multiplexer under control of the CROM.

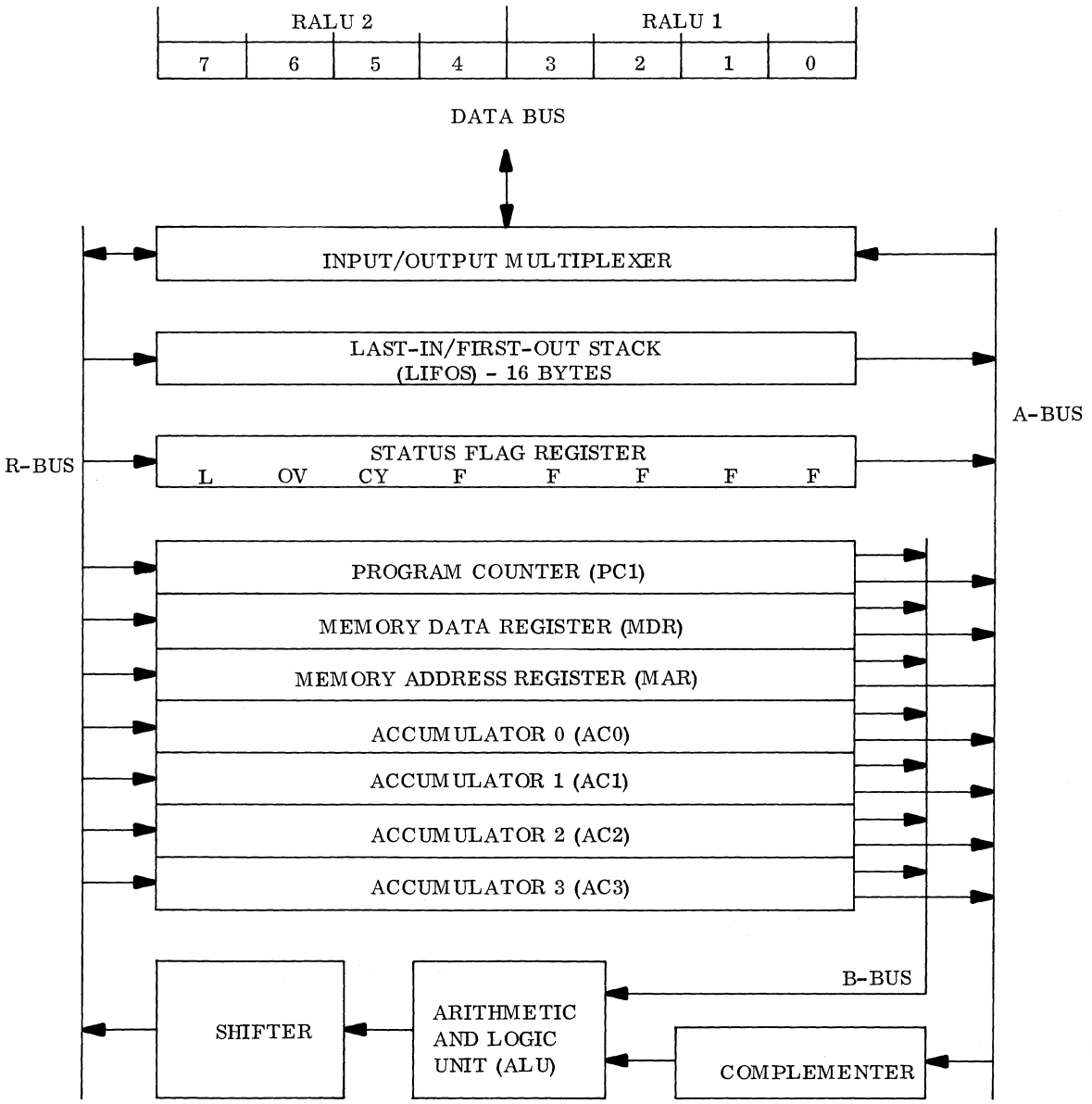


Figure 2-2. IMP-8C 8-bit Arithmetic Section

B-bus (Operand Bus). The contents of all RALU registers can be loaded onto the B-bus. The contents of the B-bus can be gated through the ALU.

R-bus (Result Bus). The R-bus transfers the results of ALU operations to any RALU register, the LIFOS, and the Input/Output Multiplexer.

2.2.2 Last-In/First-Out Stack (LIFOS)

Each RALU has a stack that operates on a last-in/first-out basis. The stack is 16 bytes deep and is accessible through the top location. An 8-bit data byte is entered into the stack via the R-bus and retrieved from the stack via the A-bus. As a byte is entered into the top location of the stack, the byte currently in the top location is pushed down one level, and the new byte occupies the top location. The contents of each lower level are replaced by the contents of the next higher location. The contents of the bottom location are lost. The reverse process occurs when a byte is retrieved from the stack; in this case, zeros are entered into the bottom location.

The stack is used primarily to save status during interrupts and to temporarily store subroutine return addresses. It may also be used to temporarily store data using the appropriate instructions described in chapter 3.

2.2.3 RALU Flags

There are eight RALU Status Flags in the Status Flag Register. These flags may be pushed onto the stack for temporary storage during interrupt processing. The flags may be transferred back into their respective flag flip-flops after completion of the interrupt service. Whenever the status flags are manipulated, the entire complement of flags is configured as an 8-bit register; the L(Link), OV (Overflow) and CY (Carry) Flags are the first, second, and third most significant bits, respectively; and the remaining 5 bits comprise the general-purpose flags.

2.2.4 Program Counter (PC1)

The Program Counter (PC1) holds the byte address of the next instruction to be executed. It is incremented by 1 immediately following the fetching of each byte during execution of the current instruction. When there is a branch to another address in the main memory, the branch address is set into the Program Counter. A skip instruction increments the Program Counter twice, causing one 2-byte instruction to be skipped. PC1 is used with the Page Counter (PC0) to provide the 16-bit address.

2.2.5 Memory Data Register (MDR) and Memory Address Register (MAR)

The Memory Data Register (MDR) holds data transferred between the main memory and the processor. When fetching data, the effective byte address is placed in the Memory Address Register (MAR), and the fetch instruction causes the data word to be transferred from the designated main-memory location to the MDR. Conversely, when storing data in the main memory, the data word is placed in the MDR; the effective address is placed in the MAR; and the store instruction causes the data word to be transferred to the designated memory location.

2.2.6 Accumulators (AC0, AC1, AC2, and AC3)

The accumulators are 8-bit registers that hold the operands during arithmetic and logical operations. Also, the result of an operation may be stored temporarily in one of the four accumulators. Data words may be read from memory to the accumulator or written into memory from the accumulator. The particular accumulator to take part in an operation is specified by the programmer in the instruction.

2.2.7 Arithmetic and Logic Unit (ALU), Shifter, and Complementer

The Arithmetic and Logic Unit (ALU), performs both arithmetic and logical operations: binary addition, AND, OR, and EXCLUSIVE-OR. Arithmetic and logical operations are effected by the microprogram that is part of the CROM. The IMP-8C performs arithmetic operations using the twos-complement. The contents of the A-bus may be selectively complemented under control of the CROM.

The output of the ALU is transferred to the R-bus through the Shifter and may then be stored in the stack or any of the RALU registers. The Shifter is capable of performing a single-position shift, to the left or right, during each basic machine cycle.

2.2.8 Input/Output Multiplexer

The Input/Output Multiplexer handles data, address, and instruction transfers between the RALUs and main memory or peripheral devices on a time-multiplexed basis. As shown in figure 2-2, output data (to data bus) must be on the A-bus, and input data passes from the Input/Output Multiplexer to the R-bus.

2.3 CONTROL AND READ-ONLY MEMORY (CROM)

Figure 2-3 is a simplified block diagram of the CROM. Data Buffers are provided between the CPU MOS/LSI devices and other parts of the equipment that have been implemented with TTL logic. These buffers are TRI-STATE ® logic elements that permit bus-connected inputs.

The primary control of the RALU devices is accomplished over the 4-bit Control Bus. The Control Bus is time-multiplexed to transfer four 4-bit control words of information per microcycle.

The CROM contains a microprogram that implements the standard instruction set. The microinstruction capacity of the CROM is 100 words (each word is 23 bits). During an instruction fetch, the first byte is transferred to the CROM; these 8 bits comprise the op code and other pertinent control fields of an instruction word. The instruction bits are decoded, and then the ROM Address Control in the CROM directs the control sequence to an entry point in the microprogram. The sequence continues until execution of the fetched instruction is completed. Then, the CROM goes through another fetch cycle to fetch the next instruction from memory. This process is continuously repeated.

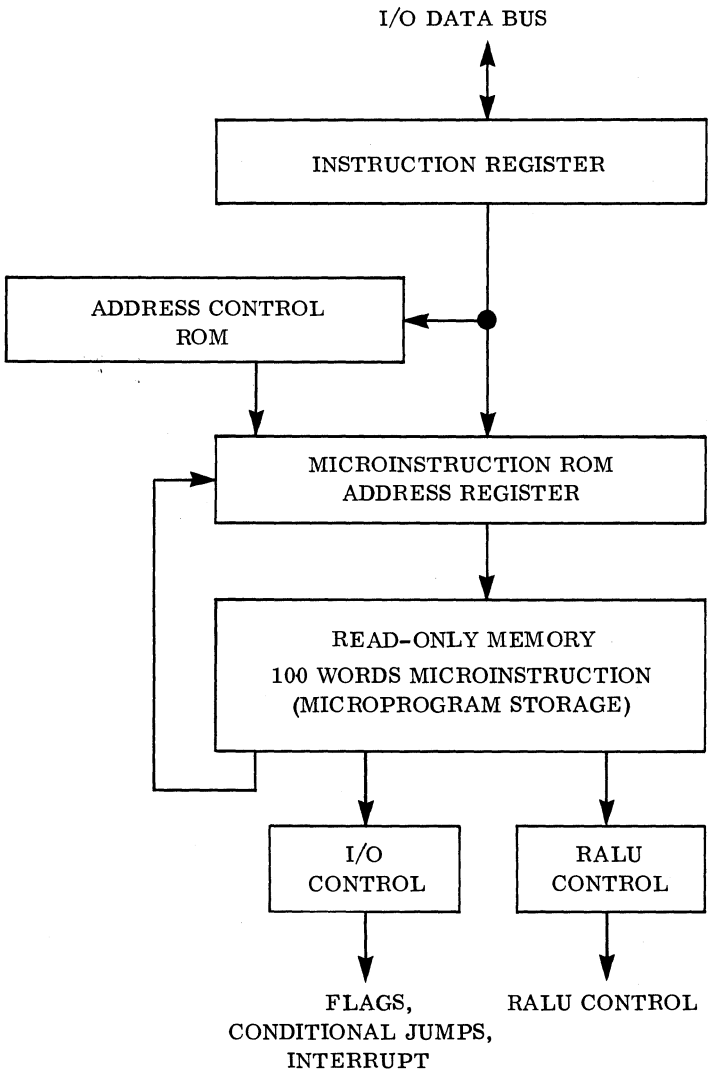


Figure 2-3. Control Read-Only Memory Simplified Block Diagram

2.4 IMP-8C TIMING

The basic machine cycle of the IMP-8C consists of the execution of a single microinstruction step. This cyclic time period is called a microcycle. The microcycle contains eight time intervals, T1 through T8. During a microcycle requiring access to memory, T4 is extended for three additional time intervals. Figure 2-4 shows the relationship of the basic timing signals in the IMP-8C during a microcycle.

The timing signals for the IMP-8C are generated by a crystal-controlled, 5.7143-MHz oscillator, which produces the Clock Signal (CLK). The Clock Signal drives an 8-bit shift register connected as a ring counter. The outputs of the 8 bits of the shift register are C1 through C8, which correspond to the eight time intervals of the basic microcycle.

During a microcycle containing a memory read or write operation, the clock to the shift register is inhibited at T4 for three clock intervals, therefore extending T4 to four time intervals (T4, T4+1, T4+2, and T4+3).

The Phase Clock Signals for the CROM (PH1, PH3, PH5, and PH7) are generated at the odd-numbered time intervals (T1, T3, T5, and T7, respectively). The Phase Clock Signals provide the timing for the CROM and the gating of control information from the CROM to the RALU via a 4-line-time-multiplexed Control Bus.

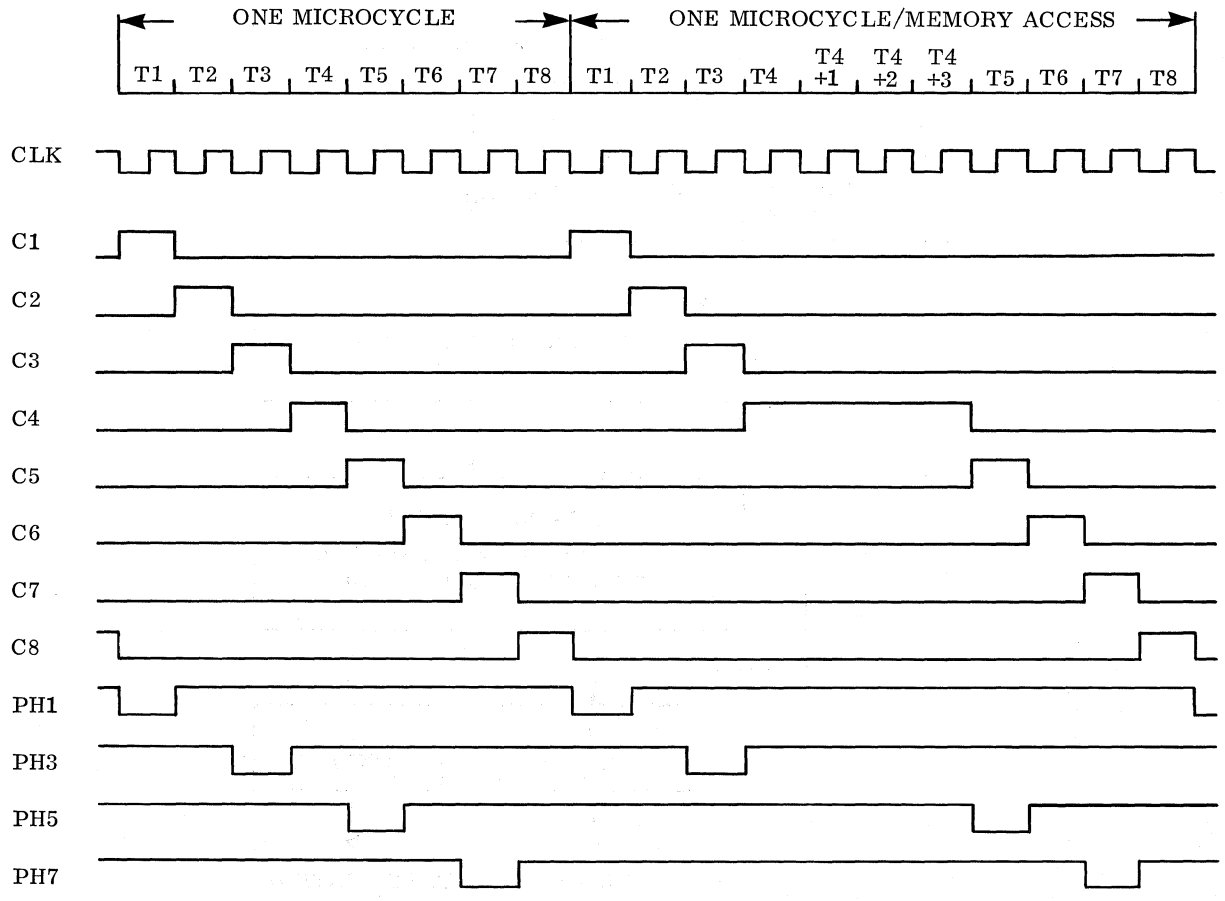


Figure 2-4. IMP-8C Timing

2.4.1 Control Information Timing

The Control Bus Signals (CB0 through CB3) are generated by the CROM to communicate with the RALU. The function of the 4 control bits, CB0 through CB3, is shown in figures 2-5 through 2-8 and described in the following paragraphs. A diagram of RALU control and timing is shown in figure 2-9.

Phase 1 Control Bits

If the 3-bit Address Field is nonzero, the contents of the designated register are gated onto the A-bus. If the Address Field is zero and the Push/Pull Control Bit is '1', the contents of the byte on the top of the LIFOS are gated onto the A-bus. If the Address Field is zero and the Push/Pull Control Bit is '0', then a value of zero is gated onto the A-bus.

CB0	CB1	CB2	CB3
ADDRESS FIELD (REGISTER TO A-BUS)			STACK TO A-BUS

Figure 2-5. Phase 1 Control Bit Configuration

NOTE

Pushing data onto the stack is also contingent on the Push/Pull Control Bit, but is described under Phase 7 Control Bits, where the operation occurs.

Phase 3 Control Bits

If the 3-bit B-bus Address Field is nonzero, then the contents of the register designated are gated onto the B-bus. If the Address Field is zero, a value of zero is gated onto the B-bus. The Complement A-bus Bit causes the A-bus input to the ALU to be complemented.

CB0	CB1	CB2	CB3
ADDRESS FIELD (REGISTER TO B-BUS)			COMPLEMENT

Figure 2-6. Phase 3 Control Bit Configuration

Phase 5 Control Bits

The ALU Bits designate a function according to table 2-1. The Control Function Code Bits designate a function according to table 2-2. The no-op function applies only to the Control Function Code Field. The expression R← 0/SIGN means that either zeros or the sign of the less significant 4-bit byte of the word being transferred to the R-bus is propagated throughout the more significant 4-bit byte. (If the fourth Control Bit CB3 is '1' during phase 7, the sign is propagated; otherwise, zero is propagated.)

CB0	CB1	CB2	CB3
ALU FUNCTION CODE		CONTROL FUNCTION CODE	
(See table 2-1.)		(See table 2-2.)	

Figure 2-7. Phase 5 Control Bit Configuration

Table 2-1. ALU Function Code

Code	Function
00	AND
01	XOR
10	OR
11	ADD

Table 2-2. Control Function Code

Code	Function
00	No Op (no operation for control bits only)
01	R←0/SIGN (zero or sign propagation)
10	LSH (left shift)
11	RSH (right shift)

Phase 7 Control Bits

If the 3-bit Address Field is nonzero, the contents of the R-bus are gated into the register addressed by this field. If the R-bus Address is zero and the Push/Pull Control Bit was '1' during phase 1, data is pushed onto the LIFOS from the R-bus.

If the Control Function Code Bits transferred during phase 5 do not specify R←0/SIGN (see table 2-2), then CB3 specifies the source of the data gated onto the R-bus:

1. If CB3 is '0', the source is the output of the ALU.

2. If CB3 is '1', the data comes from an external source via the Input/Output Multiplexer. If R←0/SIGN is specified, then CB3 specifies whether zero or the sign is propagated throughout the more significant 4 bits.

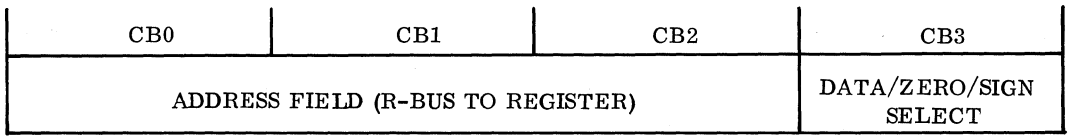


Figure 2-8. Phase 7 Control Bit Configuration

2.4.2 Data Transfer Timing

Data transfers between the RALU and the data bus may occur at T2, T4, and T7. At T2, the contents of the R-bus (result of preceding microcycle) are gated onto the data bus. Primarily, the data provide inputs to the Conditional Jump Multiplexer to permit testing of the result of the operation performed on the previous microcycle.

During T4, the data word presently on the A-bus is gated onto the data bus. The information that appears on the data bus at this time is either output data (to memory or an external device) or an address value to be loaded in the Address Register.

During T7, data to be transferred to the RALU appears on the data bus.. The data may then be gated onto the R-bus by the RALU Input/Output Multiplexer and, subsequently, may be stored in one of the working registers (AC0 through AC3).

2.4.3 Miscellaneous Timing Signals

The flag flip-flops may be set and reset at T2 and T6, respectively. A unique flag address is established during each machine cycle; at T2 the flag may be set, and at T6 the flag may be reset. It is thus possible for the microprogram to set, reset, or pulse a flag during a single machine cycle.

The Conditional Jump Multiplexer shares the same Address Lines as the flag flip-flops. If a conditional jump is being performed as determined by the microprogram, the condition is tested during T2.

Figure 2-9 shows the relationship of the control information, data transfer, and miscellaneous timing signals during a microcycle.

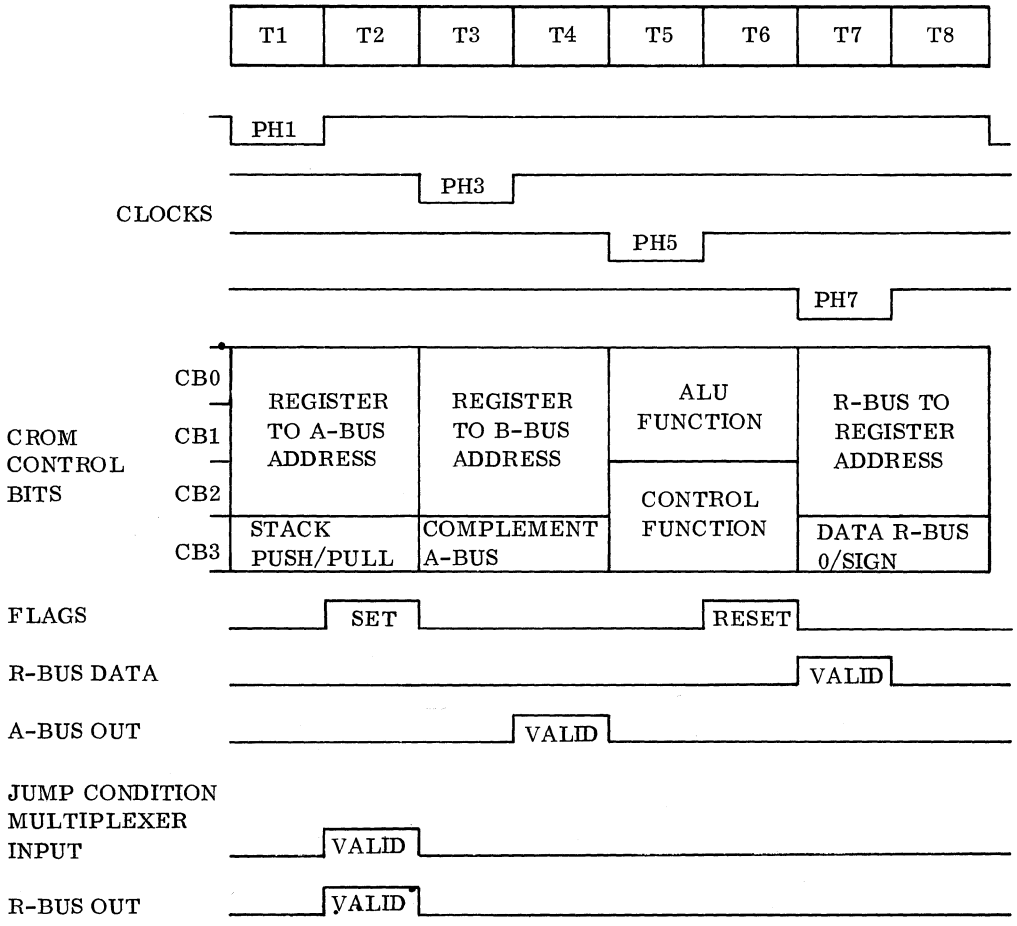


Figure 2-9. IMP-8C Timing Control

2.5 IMP-8C OPERATION

Control of the IMP-8C operations is accomplished by routines that constitute the microprogram stored in the CROM. The microprogram effects the implementation of the macroinstructions that comprise the IMP-8C instruction set. The actual data processing operations are performed in the RALU. The following paragraphs provide a functional description of the initialization, instruction execution, and memory access operations of the IMP-8C.

2.5.1 Initialization

When power is first applied or an External System Clear Signal is received by the IMP-8C, all RALU registers, flags, and the stack are cleared. The Program Counter is set to the starting location ($7FFE_{16}$) of the initialization sequence. In most applications, the initialization sequence consists of a macroprogram to initialize supervisory routines.

2.5.2 Instruction Execution

The CROM in the IMP-8C contains a microprogram that implements the standard instruction set. Each macroinstruction in a user's program is brought into the processor under control of the CROM microprogram instruction fetch routine. The macroinstruction is then decoded, and the ROM Address Control in the CROM directs the control sequence to an entry point in the microprogram execution routine for that macroinstruction. When execution is completed, the CROM goes through another instruction fetch cycle to bring in the next macroinstruction. The process is repeated continuously unless a HALT or an interrupt condition occurs.

A test for an interrupt condition is made at the beginning of each instruction fetch. If the interrupt line is high, the CROM transfers control to an interrupt sequence. For a general interrupt, the contents of the Program Counter are saved on the stack, and then the interrupt routine in memory location 0 is executed.

Figure 2-10 shows the sequence of operations for an instruction fetch. The instruction fetch begins during the last microcycle of the instruction prior to the fetch (IFETCH-1). During the last microcycle of an instruction, the page address of the next instruction to be executed is set in the eight high-order Address Latches. The first microcycle of the instruction fetch (IFETCH) loads the byte address of the instruction in the eight low-order Address Latches, and transfers the byte at the memory location specified to the CROM. During the next microcycle (IFETCH+1), the Program Counter, comprised of PC0 and PC1, is incremented. If the first byte read is a nonmemory reference, the operation specified by the byte is executed by the appropriate microprogram (EXEC).

If the first byte read defines an operation containing a memory reference, the byte address from the Program Counter is loaded into the Address Latches, and the second byte, containing the Displacement Field, is read (IFETCH+2). If the operation on the displacement byte is referenced to the base page, the eight high-order Address Latches are cleared during the next microcycle (IFETCH+3); otherwise, the Address Latches are unaffected during IFETCH+3. The operation code byte is examined during the subsequent microcycle (IFETCH+4) to determine if indexing is specified. If indexing is not indicated in the operation, the Displacement Field is loaded into the low-order eight bits of the Address Latches, and the operand is read into the RALU (IFETCH+5). The Program Counter is incremented during the next microcycle (IFETCH+6), and the appropriate microprogram is subsequently executed (EXEC).

An indexed instruction requires the calculation of a new page and byte address. The calculation of the byte address is accomplished by adding the contents of Accumulator 3 (AC3) to the Displacement Field in one microcycle (IFETCH+5). The page address is set into the Address Latches by transferring the contents of AC2 to the eight high-order Address Latches during the next microcycle (IFETCH+6). The new displacement containing the byte address is loaded into the low-order Address Latches, and the operand is read into the RALU (IFETCH+7). Before proceeding with the execution of the specified operation (EXEC), the Program Counter is incremented (IFETCH+8).

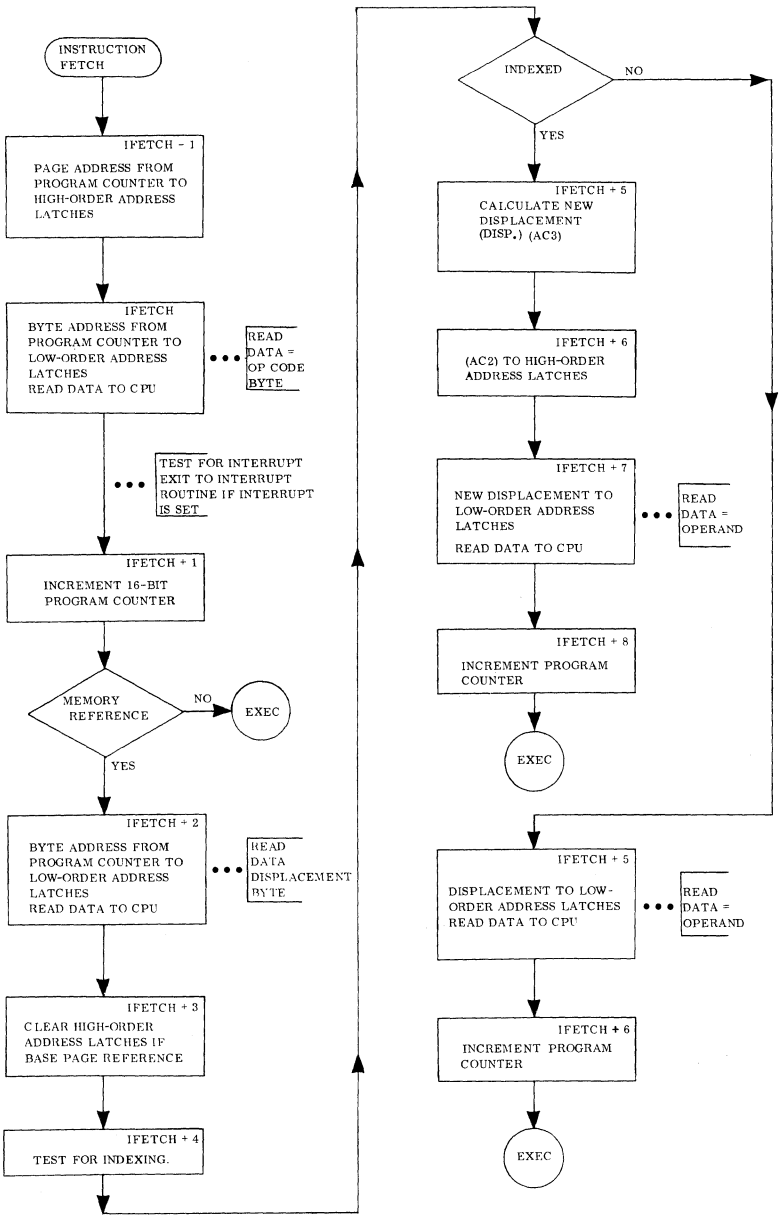


Figure 2-10. Instruction Execution Flowchart

2.5.3 Communications With Memory or Peripheral Devices

During input/output operations, the microprogram sends an address on the data bus at T4; this address is loaded into the Address Register under control of the Read Flag.

When executing a read operation, the processor sends out an address on the bus at T4 and expects data back at T7 of the same microcycle. A clock holding circuit, described in chapter 4, is included on the IMP-8C/200 card to extend the interval between T4 and T7 to accommodate memories whose access times are longer than the normal T4-to-T7 interval.

A write operation takes 2 microcycles, because both address and data have to be sent out. The address is sent out to the Address Register and latched during T4 of the first microcycle. The output data to memory arrives during T4 of the next microcycle and can be used directly to write. In the IMP-8C, T4 is stretched for three additional periods to accommodate both the read and write delays necessary to communicate with the IMP-8C memory.

Chapter 3

IMP-8C/200 INSTRUCTION SET

3.1 INTRODUCTION

Eight functional types of instructions comprise the IMP-8C/200 instruction set:

- Load and Store
- Arithmetic
- Logical
- Skip
- Shifts
- Transfer of Control
- Register
- Miscellaneous

The instructions for each functional type are described as a group. For each instruction, the name of the instruction, its mnemonic, byte format, its operation in the form of an equation, and a brief explanation of its operation are given. A tabulated summary of each type of instruction precedes the detailed descriptions.

Before describing the instructions, brief descriptions of the registers referred to in the instruction description are given.

The IMP-8C/200 instructions and their assembler language op code mnemonics are summarized in appendix A. For further programming details, see the IMP-8 programming/assembler manuals.

3.2 ARITHMETIC AND LOGIC UNITS REFERENCED IN IMP-8C/200 INSTRUCTIONS

The arithmetic and logic units referenced in the ensuing description of the IMP-8C/200 instructions are listed below and shown in block diagram form in figure 3-1.

- Last-In/First-Out Stack (LIFOS)
- Status Register
- Program Counter (Page Counter, PC0; Byte Counter, PC1)
- Accumulators 0, 1, 2, and 3 (AC0, AC1, AC2, AC3)

3.2.1 Last-In/First-Out Stack (LIFOS)

The IMP-8C/200 has a hardware stack to store or retrieve data bytes on a last-in/first-out basis. The stack is 16 bytes deep. As a data byte is entered into the stack, the contents of the top location and each other location are pushed down to the next lower level; if the stack is full, the last byte will be pushed down to a nonexistent location and, therefore, lost. Conversely, the contents of the top location are pulled from the stack during retrieval of a data byte; the top location and each lower location are replaced by the contents of the next lower location, and zeros are entered into the bottom location.

The stack is used primarily for saving status during interrupts and for saving subroutine return addresses. It may be used also for temporary storage of data using the PUSH, PULL, XCHRS, PUSHF, and PULLF instructions (described later in this chapter).

3.2.2 Status Register

There are eight RALU Status Flags. These flags may be pushed onto the stack (saved) or may be loaded from the stack (restored). During such operations, the flags are configured as an 8-bit word: the L (Link), OV (Overflow), and CY (Carry) Flags are the first, second, and third most significant bits, respectively, and the remaining five general-purpose flags comprise the remaining five less significant bits (figure 3-11).

The L Flag is primarily used in some shifting operations, and the CY and OV Flags are adjuncts for arithmetic operations. The specific uses of the flags are elaborated upon in the appropriate instruction descriptions.

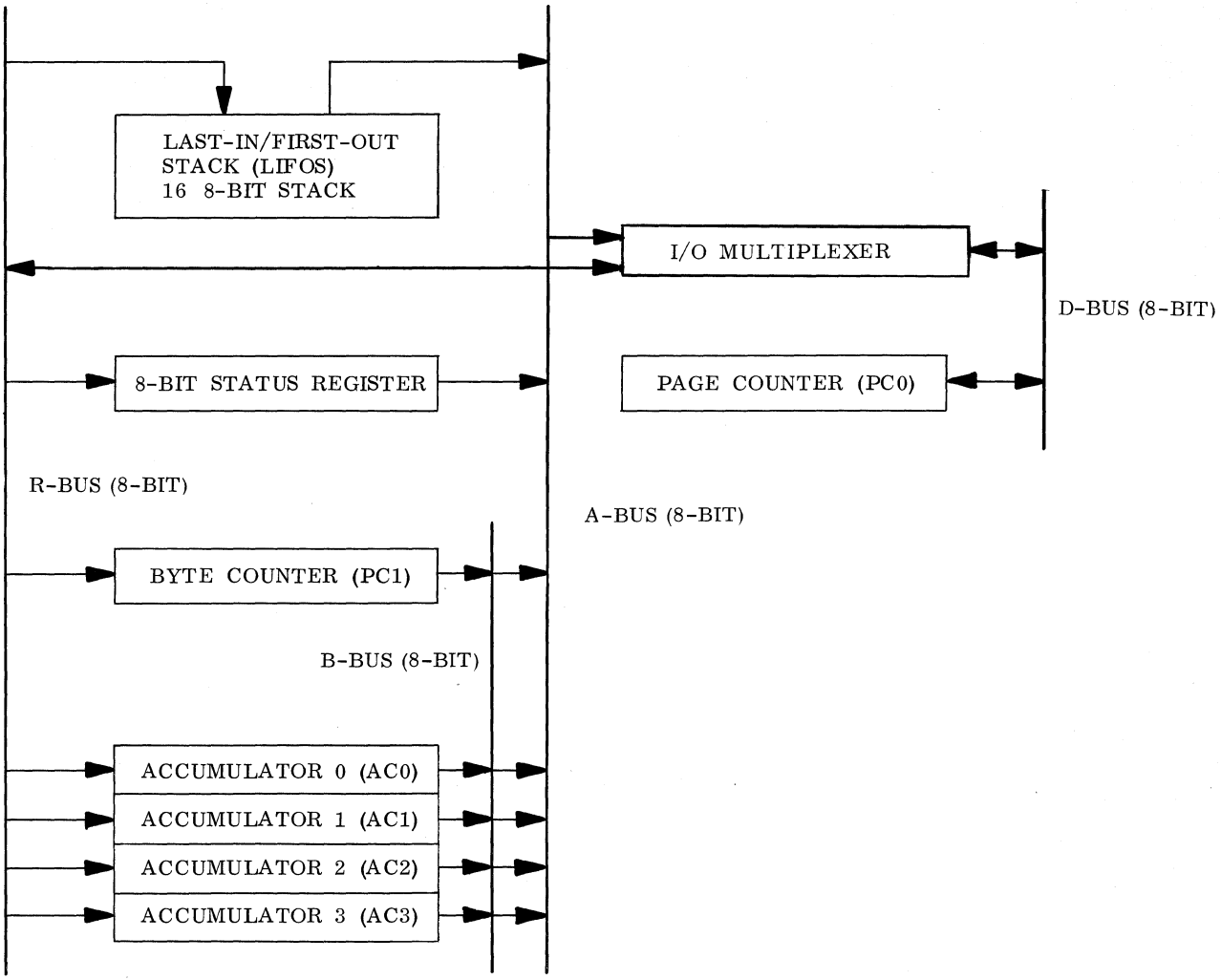


Figure 3-1. Arithmetic and Logic Units Referenced in IMP-8C/200 Instructions

3.2.3 Program Counter

The Page Counter (PC0) and the Byte Counter (PC1) comprise the Program Counter. The Program Counter contains the address of the next instruction to be executed. When there is a branch to another address in the main memory, the branch address is loaded into the Program Counter. A skip instruction increments the Byte Counter by 2, thus causing one 2-byte instruction or two single-byte instructions to be skipped.

3.2.4 Accumulators 0, 1, 2, and 3 (AC0, AC1, AC2, and AC3)

The accumulators are used as working registers for data manipulation. Data may be fetched from a location in memory to an accumulator, and may be stored from an accumulator to a location in memory. The particular accumulator to take part in an operation is specified by the programmer in the appropriate instruction.

3.3 DATA AND INSTRUCTIONS

3.3.1 Data Representations

Data are represented in the IMP-8C/200 in twos-complement integer notation. In this system, the negative of a number is formed by complementing each bit in the data byte and adding 1 to the complemented number. The sign is indicated by the most significant bit. In the 8-bit byte of the IMP-8C/200, when bit 7 is a '0', it denotes a positive number; when bit 7 is a '1', it denotes a negative number. Maximum number range for this system is $7F_{16}$ ($+127_{10}$) to 80_{16} (-128_{10}). The Carry Flag (CY) is set if there is a carry out of bit 7 as a result of an arithmetical operation. The Overflow Flag (OV) is set if the carry into bit 7 is different from the carry out of bit 7 as the result of an arithmetical operation.

3.3.2 Instructions

There are eight classes of IMP-8C/200 instructions. Each class of instruction and the associated instructions are summarized in a table preceding the descriptions of the instructions. Also, the applicable instruction byte formats are defined.

3.4 MEMORY ADDRESSING

The IMP-8C/200 instruction set provides for direct, indexed, and indirect memory addressing. Two types of direct and indirect addressing are available: base page addressing, where the address is on page 00 and the instruction is on any page; and current page addressing, where the address is on the same page as the instruction. Indexed addressing is specified by the XR Field of a 2-byte instruction. The word formats for direct, indexed, and indirect addressing are shown in figure 3-2.

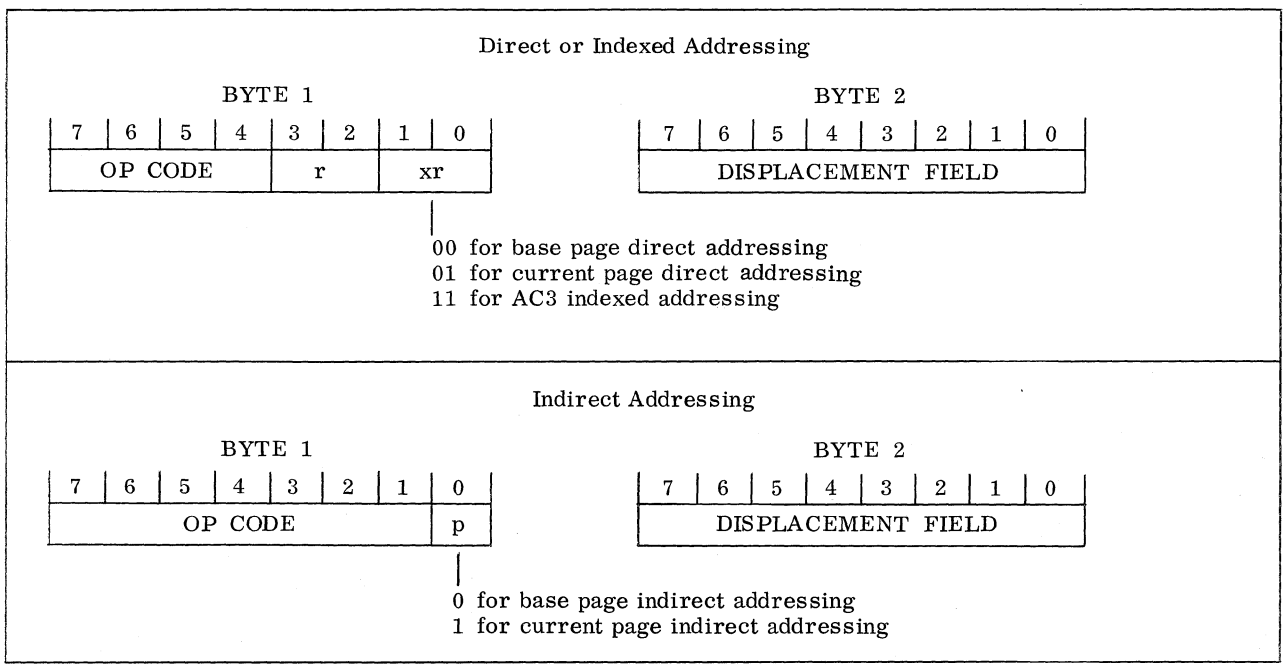


Figure 3-2. Instruction Bytes for Addressing Memory

3.4.1 Base Page Addressing

When the XR Field is 00, the instruction specifies base page addressing. Base page is directly accessible from any location in the address space of the memory. When base page addressing is specified, the effective byte address is defined by the contents of bits 0 through 7 of the second byte of the instruction and the page address is 0. As many as 256 bytes (page 0, byte 0 through page 0, byte 255) may be addressed in this manner.

3.4.2 Current Page Addressing

When the XR Field is 01, the instruction specifies current page addressing. The effective byte address is defined by the contents of bits 0 through 7 of the second byte of the instruction and the page address is defined by the current page address in the high order half of the Program Counter (PC0). As many as 256 bytes on the same page as the instruction may be addressed in this manner.

3.4.3 Indexed Addressing

When indexed addressing is specified, the contents of Accumulator 2 (AC2) define the page address and the contents of Accumulator 3 (AC3) is added to the displacement value to define the byte address. The addition of the Displacement Field to the accumulator will not alter the page count and any carry out of the accumulator will be lost. Indexed addressing provides an effective address to access any location in the 65,536 bytes of memory.

Table 3-1. Summary of Addressing Modes

XR Field	Addressing Mode	Effective Address
00	Base Page Direct Addressing	EA = Page 0, byte (disp)
01	Current Page Direct Addressing	EA = Page (PC), byte (disp)
11	Indexed Addressing Relative to AC3	EA = Page (AC2), byte (AC3 + disp)

3.4.4 Indirect Addressing

Indirect addressing is accomplished by using two bytes in the base or current page to provide the address of the operand. The 8-bit address provided by the instruction Displacement Field points to the first of two bytes; these two bytes contain in byte 1 the page of the desired memory address and in byte 2 the byte address (within the page addressed by byte 1) of the operand. The following instructions use indirect addressing:

- Load Indirect
- Store Indirect
- Jump Indirect
- Jump to Subroutine Indirect

3.5 NOTATIONS AND SYMBOLS USED IN IMP-8C/200 INSTRUCTION DESCRIPTIONS

Refer to table 3-2 for definitions of the notations and symbols used in the IMP-8C/200 instruction descriptions. The notations are given in alphabetical order followed by the symbols. Upper-case mnemonics refer to fields in the instruction word; lower-case mnemonics refer to the numerical value of the corresponding fields. In cases where both lower- and upper-case mnemonics are composed of the same letters, only the lower-case mnemonic is given in table 3-2. The use of lower-case notation designates variables.

Table 3-2. Notations Used in Instruction Descriptions

Notation	Meaning
ACr	Denotes a specific working register (AC0, AC1, AC2, or AC3), where r is the number of the accumulator referenced in the instruction.
AR	Denotes the Address Register used for addressing memory or peripheral devices.
cc	Denotes the 4-bit condition code value for conditional branch instructions.
CY	Indicates that the Carry Flag is set if there is a carry due to the instruction (either an addition or a subtraction).
disp	Stands for displacement value and it represents an operand in a nonmemory reference instruction or an Address Field in a memory reference instruction. The displacement value is expressed as an 8-bit unsigned number.
dr	Denotes the number of a destination working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
EAQ	Denotes the effective address specified by the instruction directly or by indexing. The contents of the effective address are used during execution of an instruction. See table 3-1.
fc	Denotes the number of the referenced flag (see paragraph 3.6.8, table 3-13, Control Flag Codes).
INEN	Denotes the Interrupt Enable Control Flag.
L	Denotes 1-bit Link (L) Flag.
OV	Indicates that the Overflow Flag is set if there is an overflow due to the instruction (either an addition or a subtraction).
p	Denotes base or current page location for byte address specified in an indirect addressing instruction.
PC	Denotes the Program Counter. During address formation, it is incremented by 1 to contain an address 1 greater than that of the instruction being executed. PC0 is page; PC1 is byte.
r	Denotes the number of a working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
SEL	Denotes the Select Control Flag. It is used to select the carry or overflow for output on the Carry and Overflow (CYOV) Line of the CPU, and to include the Link Bit (L) in shift operations.
sr	Denotes the number of a source working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
STK	Top two words of stack: STK0 is the top; STK1 is one down.
xr	Denotes memory addressing mode selected. See table 3-1.
( )	Denotes the contents of the item within the parentheses. (ACr) is read as "the contents of ACr." (EA) is read as "the contents of EA."
[ ]	Denotes "the result of."
~	Indicates the logical complement (ones complement) of the value on the right-hand side of
→	Means "replaces."
←	Means "is replaced by."

Table 3-2. Notations Used in Instruction Descriptions (Continued)

Notation	Meaning
@	Appearing in the Op Code Field of an instruction, denotes indirect addressing.
$\wedge$	Denotes an AND operation.
$\vee$	Denotes an OR operation.
∇	Denotes an EXCLUSIVE-OR operation.

3.6 INSTRUCTION DESCRIPTIONS

Each class and subclass of instruction is introduced by a table that lists and summarizes the instructions. The word format then is illustrated. Detailed descriptions are given, providing the following information:

- Name of instruction followed by operation code mnemonic in parentheses
- Operation code in word format diagram
- Operation in equation notation
- Description of operation in detail

3.6.1 Load and Store Instructions

There are four instructions in this group. These are summarized in table 3-3 and then individually described. The word format is shown in figure 3-3.

NOTE

For indirect operations, the symbol @ must follow the op code mnemonic designated in the operator field of the assembler instruction, and p denotes current or base page location of the displacement byte address.

Table 3-3. Load and Store Instructions

Instruction	Op Code	Operation	Assembler Format	
Load Any Register	1000	(ACr) ← (EA)	LD	ACr, disp (AC3)
Load Indirect	0010010	(AC0) ← ((EA))	LD@	disp
Store Any Register	1001	(EA) ← (ACr)	ST	ACr, disp (AC3)
Store Indirect	0011010	((EA)) ← (AC0)	ST@	disp

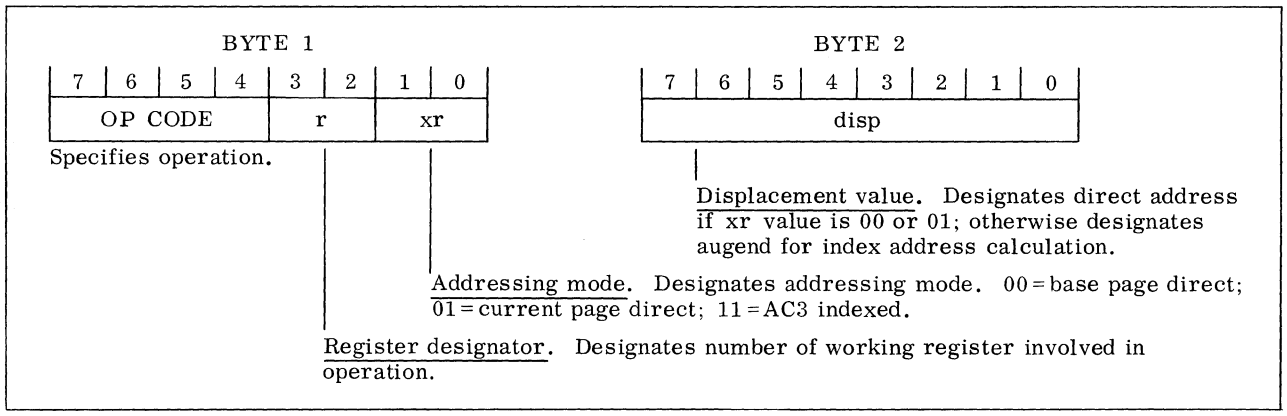


Figure 3-3. Load and Store Instruction Format

Load (LD)

Operation: (ACr) ← (EA)

Description: The contents of ACr are replaced by the contents of EA. The initial contents of ACr are lost; the contents of EA are unaltered.

Load Indirect (LD@)

Operation: (AC0) ← ((EA))

Description: The contents of AC0 are replaced indirectly by the contents of EA. The initial contents of AC0 are lost; the contents of EA and the location that designates EA are unaltered.

Store (ST)

Operation: (EA) ← (ACr)

Description: The contents of EA are replaced by the contents of ACr. The initial contents of EA are lost; the contents of ACr are unaltered.

Store Indirect (ST@)

Operation: ((EA)) ← (AC0)

Description: The contents of EA are replaced indirectly by AC0. The initial contents of EA are lost; the contents of AC0 and the location that designates EA are unaltered.

3.6.2 Arithmetic Instructions

There are two instructions in this group summarized in table 3-4 and then described individually. Either of these instructions may be carried out with any of the four general-purpose accumulators (AC0, 1, 2, or 3). The word format is shown in figure 3-4.

Table 3-4. Arithmetic Instructions

Instruction	Op Code	Operation	Assembler Format
ADD (ADD)	1010	(ACr) ← (ACr) + (EA), OV, CY	ADD r, disp(xr)
SUBTRACT (SUB)	111100	(AC0) ← (AC0) - (EA), OV, CY	SUB disp(xr)

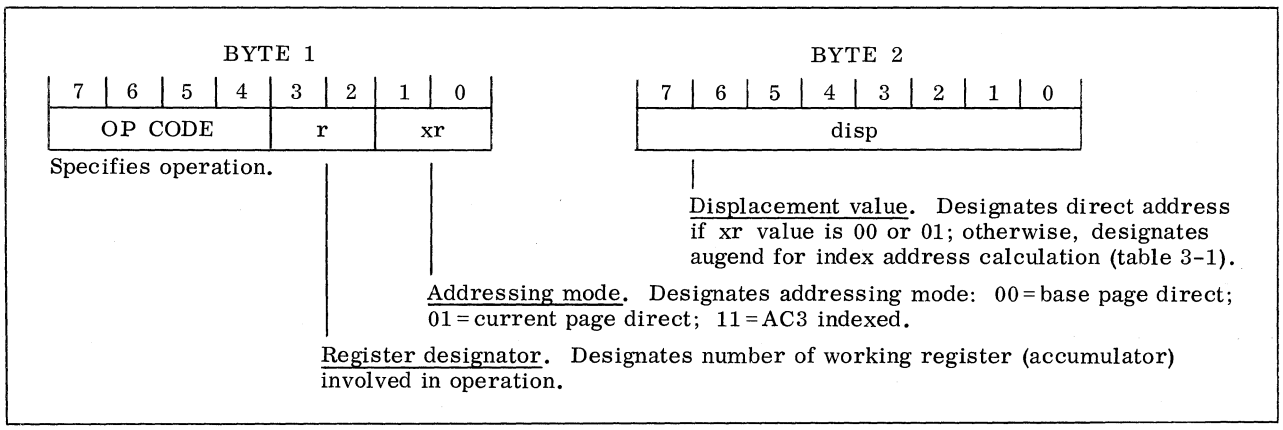


Figure 3-4. Arithmetic Instruction Format

Add (ADD)

BYTE 1

7	6	5	4	3	2	1	0
1	0	1	0	r		xr	

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (ACr) ← (ACr) + (EA), OV, CY

Description: The contents of ACr are added algebraically to the contents of the effective memory location EA. The sum is stored in ACr, and the contents of EA are unaltered. The preceding contents of ACr are lost. The Carry and Overflow Flags are set according to the result of the operation.

Subtract (SUB)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: AC0 ← (AC0) - (EA), OV, CY

Description: The contents of the effective memory address are subtracted from the contents of AC0, and the result is stored in AC0. The effective memory location is unaltered. The Carry and Overflow Flags are set according to the result of the operation.

3.6.3 Logical Instructions

There are two instructions in this group, summarized in table 3-5 and then described individually. Either of these instructions may be carried out with only Accumulator AC0. The word format is shown in figure 3-5.

Table 3-5. Logical Instructions

Instruction	Op Code	Operation	Assembler Format
AND	110000	(AC0) ← (AC0) ∧ (EA)	AND disp(xr)
OR	110100	(AC0) ← (AC0) ∨ (EA)	OR disp(xr)

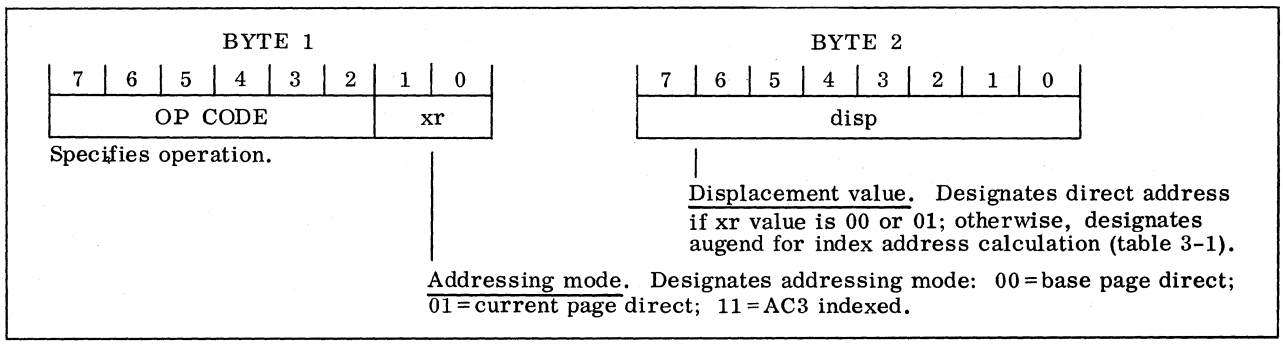


Figure 3-5. Logical Instruction Format

And (AND)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (AC0) ← (AC0) ∧ (EA)

Description: The contents of AC0 and the contents of the effective memory location EA are ANDed, and the result is stored in AC0. The initial contents of AC0 are lost, and the contents of EA are unaltered.

Or (OR)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (AC0) ← (AC0) ∨ (EA)

Description: The contents of AC0 and the contents of the effective memory location EA are ORed inclusively, and the result is stored in AC0. The initial contents of AC0 are lost, and the contents of EA are unaltered.

3.6.4 Skip Instructions

Four instructions comprise the skip instructions, summarized in table 3-6. Word format is shown in figure 3-6.

Table 3-6. Skip Instructions

Instruction	Operation Code	Operation	Assembler Format
Memory Compare			
INCREMENT AND SKIP IF ZERO	111001	(EA) ← (EA) + 1; IF (EA) = 0, (PC) ← (PC) + 2	ISZ disp(xr)
DECREMENT AND SKIP IF ZERO	111101	(EA) ← (EA) - 1; IF (EA) = 0, (PC) ← (PC) + 2	DSZ disp(xr)
Register Compare			
SKIP IF NOT EQUAL	1011	IF (ACr) ≠ (EA), (PC) ← (PC) + 2	SKNE r, disp(xr)
Limited Register Compare			
SKIP IF AND IS ZERO	111000	IF [(AC0) ∧ (EA)] = 0, (PC) ← (PC) + 2	SKAZ disp(xr)

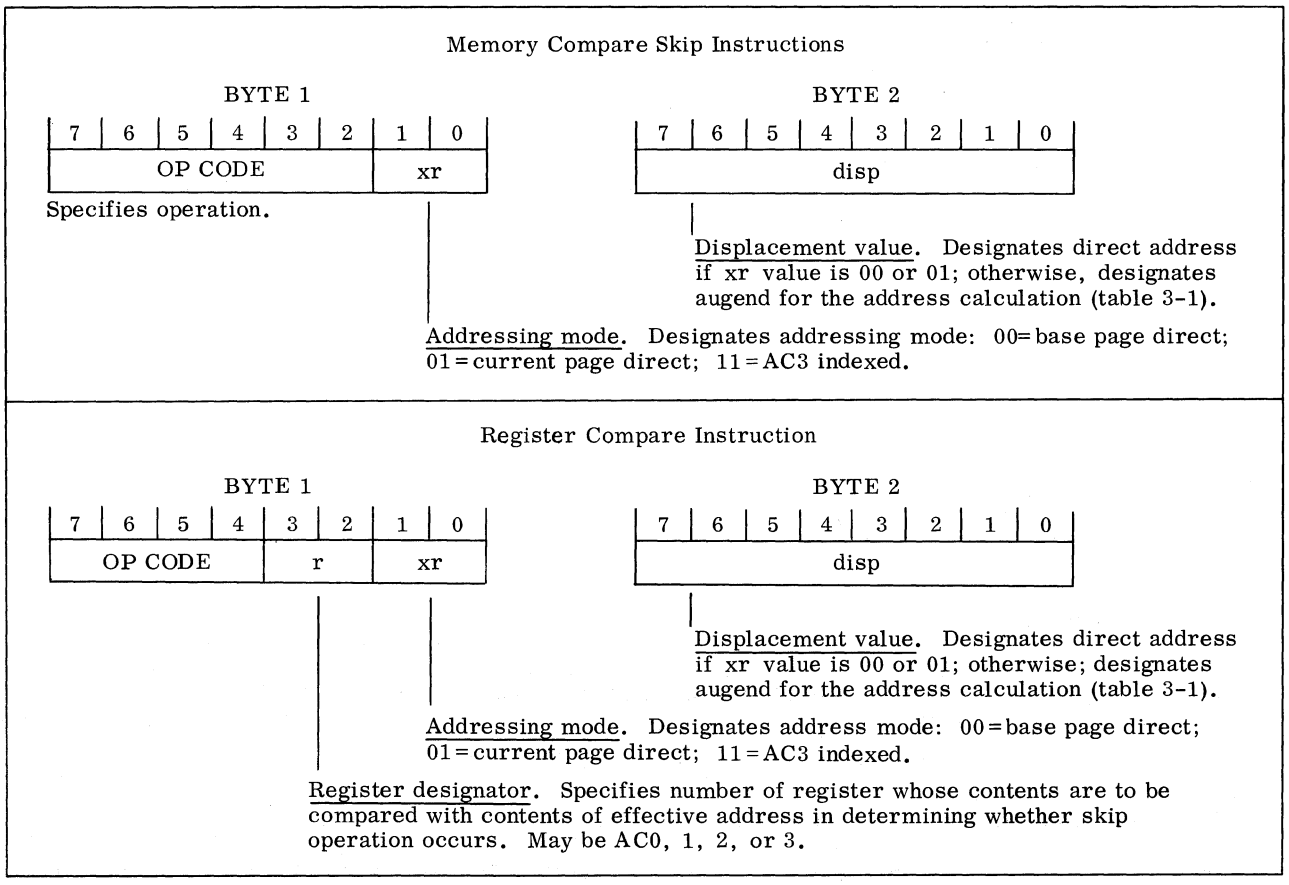


Figure 3-6. Skip Instruction Formats

Increment and Skip If Zero (ISZ)

Operation: (EA) ← (EA) + 1; if (EA) = 0, (PC) ← (PC) + 2

Description: The contents of EA are incremented by 1. The new contents of EA are tested to determine whether they equal zero. If the new contents of EA equal zero, the contents of PC are incremented by 2, thus skipping the memory location designated by the initial contents of and the memory location following PC.

Decrement and Skip If Zero (DSZ)

Operation: (EA) ← (EA) - 1; if (EA) = 0, (PC) ← (PC) + 2

Description: The contents of EA are decremented by 1. The new contents of EA are tested to determine whether they equal zero. If the new contents of EA equal zero, the contents of PC are incremented by 2, thus skipping the memory location designated by the initial contents of and the memory location following PC.

BYTE 1

7	6	5	4	3	2	1	0
1	0	1	1	r		xr	

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: If ACr ≠ (EA), (PC) ← (PC) + 2

Description: The contents of ACr (where ACr is AC0, 1, 2, or 3) and the contents of the effective memory location EA are compared. If the contents of ACr and the effective memory location EA are not equal, the contents of PC are incremented by two, thus skipping the next two byte instruction designated by the initial contents of PC. The initial contents of PC are lost. The contents of ACr and EA are unaltered.

Skip If AND Is Zero (SKAZ)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: If [(AC0) ∧ (EA)] = 0, (PC) ← (PC) + 2

Description: The contents of AC0 and the contents of the effective memory location EA are ANDed. If the result equals zero, the contents of PC are incremented by 2, thus skipping the next two byte instruction designated by the initial contents of PC. The initial contents of PC are lost. The contents of AC0 and EA are unaltered.

3.6.5 Transfer-of-control Instructions

There are seven instructions in this group, summarized in table 3-7. Three word formats are required and shown in figure 3-7.

Table 3-7. Transfer-of-control Instructions

Instruction	Operation Code	Operation	Assembler Format
Jumps			
JUMP	110001	(PC) ← EA	JMP disp(xr)
JUMP INDIRECT	0000010	(PC) ← (EA), (EA + 1)	JMP@ disp
JUMP TO SUBROUTINE	110101	(STK1) ← (PC1) (STK0) ← (PC0) (PC) ← EA	JSR disp(xr)
JUMP TO SUBROUTINE INDIRECT	0000110	(STK1) ← (PC1) (STK0) ← (PC0) (PC) ← (EA), (EA + 1)	JSR@ disp
Branch			
BRANCH-ON CONDITION	0001	IF CC IS TRUE (PC1) ← (PC1) + disp	BOC cc, disp (see NOTE after BOC description)
Returns			
RETURN FROM INTERRUPT	00000001	(PC) ← (STK); INT EN Flag set	RTI
RETURN FROM SUBROUTINE	00000010	(PC) ← (STK)	RTS

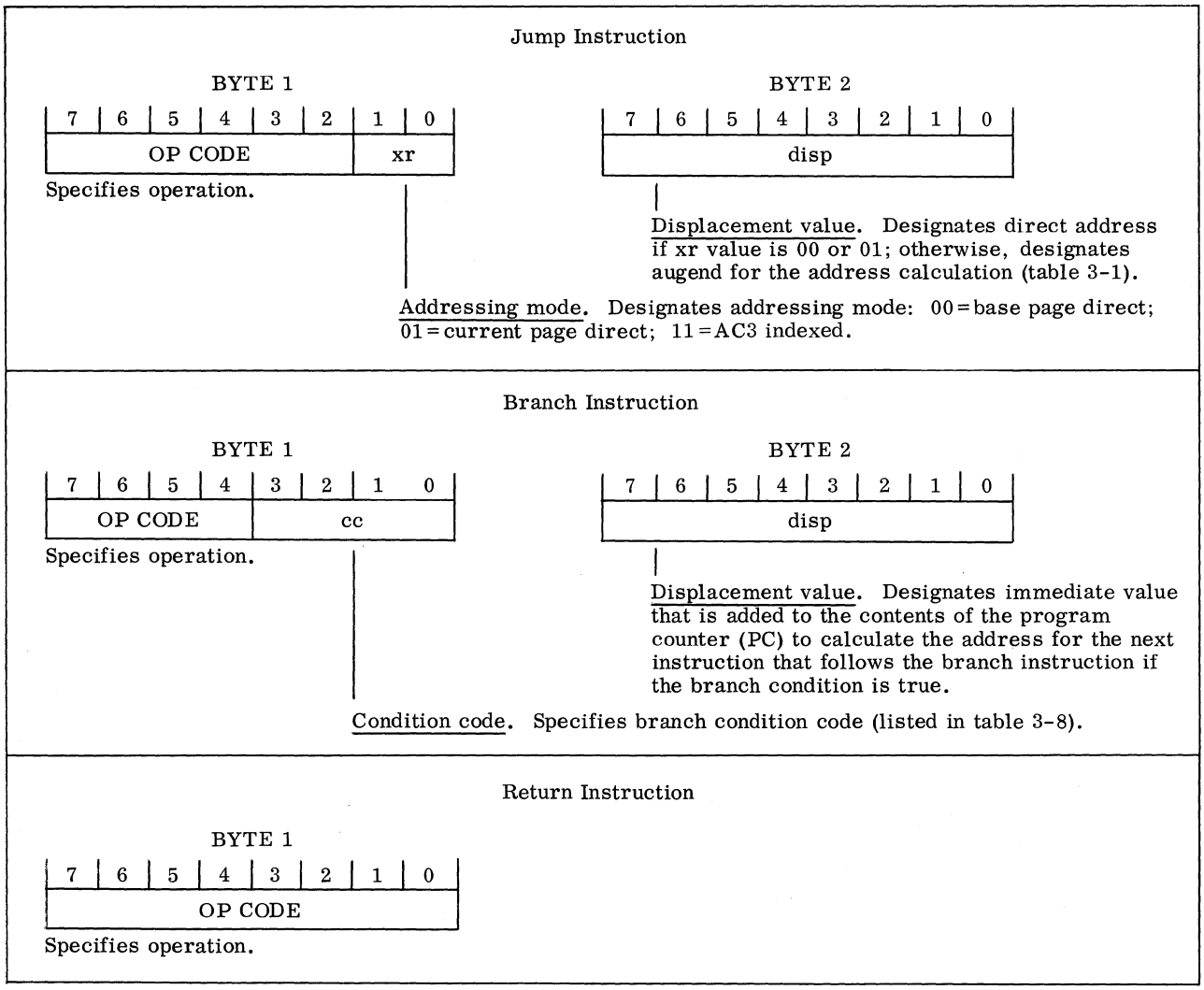


Figure 3-7. Transfer-of-control Instruction Formats

Jump (JMP)

Operation: (PC) ← EA

Description: The effective address EA replaces the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

Jump Indirect (JMP@)

NOTE

Current page addressing for JMP and JSR instructions is defined as the page of the next instruction.

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (PC0) ← (EA), (PC1) ← (EA+1)

Description: The contents of the effective two-byte address (EA) replaces the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

Jump to Subroutine (JSR)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (STK1) ← (PC1), (STK0) ← (PC0), (PC) ← EA

Description: The contents of PC are stored in the top of the stack. The effective address EA replaces the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

Jump to Subroutine Indirect (JSR@)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (STK1) ← (PC1), (STK0) ← (PC0), (PC0) ← (EA), (PC1) ← (EA+1)

Description: The contents of PC are stored in the top of the stack. The contents of the effective address (EA) replace the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

Branch-on Condition (BOC)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
disp							

Operation: (PC1) ← (PC1) + disp (PC0) unaltered

Description: There are 16 possible condition codes (cc). These are listed in table 3-8. If the condition for branching designated by cc is true, the value of disp is added to the contents of PC, and the sum is stored in PC. The initial contents of PC are lost. Program control is transferred to the location specified by the new contents of PC.

NOTE

PC is always incremented by 1 immediately following the fetching of an instruction, so the contents of PC during execution of an instruction is 1 greater than the address of that instruction. This must be considered during execution of the BOC instruction: for example, if the address of the BOC instruction is 100, then 101 is added to the value of disp.

Table 3-8. Branch-On-Condition Codes

(* indicates input required by microprogram)

CC	Input Line to CJ MUX	Condition Tested (Branch Occurs if Condition is True)
0	*INTR	Interrupt (not normally tested by programmer)
1	*NREQ0	(AC0) $\neq$ 0
2	*REQ0	(AC0) = 0
3	DATA(7) or LNK	Bit 7 of (AC0) a '1' (sign negative)
4	*DATA(1)	Bit 1 of (AC0) a '1'
5	CYOV	Carry or Overflow Flag; if SEL is set, overflow is tested; otherwise carry is tested.
6	STKFUL	Stack full
7	*START	Start Switch closed
8	*DATA(0)	Bit 0 of (AC0) a '1' (odd)
9	INEN	Interrupt Enable
10	DATA(2)	Bit 2 of (AC0) a '1'
11	DATA(3)	Bit 3 of (AC0) a '1'
12	DATA(4)	Bit 4 of (AC0) a '1'
13	DATA(5) or CY	Bit 5 of (AC0) a '1'
14	DATA(6) or OV	Bit 6 of (AC0) a '1'
15	USERJC	User Jump Condition

Return from Interrupt (RTI)

BYTE 1

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

Operation: Set INEN (Interrupt Enable Flag)
(PC) $\leftarrow$ (STK)

Description: The Interrupt Enable Flag (INEN) is set. The contents of PC are replaced by the contents of STK. Program control is transferred to the location specified by the new contents of PC. (RTI is used primarily to exit from an interrupt routine.)

Return from Subroutine (RTS)

BYTE 1

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

Operation: (PC) ← (STK)

Description: The contents of PC are replaced by the contents of STK. Program control is transferred to the location specified by the new contents of PC. (RTS is used primarily to return from subroutines entered by JSR.)

3.6.6 Shift Instructions

Four instructions comprise this group. All four instructions may be used with the Link (L) Bit by setting the SEL Flag. This is accomplished with a Set Flag (SFLG) instruction before executing the shift or rotate instruction. Examples of shifting with and without SEL set are given in diagram form for each instruction in the descriptions that follow. Note that the SEL Flag also affects the BOC instructions as indicated in table 3-8.

The shift instructions are summarized in table 3-9, and the word format is shown in figure 3-8.

All shift and rotate operations may be carried out with any of the four general-purpose accumulators, AC0, 1, 2, or 3.

Table 3-9. Shift Instructions

Instruction	Operation Code	Operation		Assembler Format
		SEL = 0	SEL = 1	
ROTATE LEFT	111110	$(ACr_0) \leftarrow (ACr_7)$, $(ACr_n) \leftarrow (ACr_{n-1})$	$(ACr_0) \leftarrow (L)$, $(L) \leftarrow (ACr_7)$, $(ACr_n) \leftarrow (ACr_{n-1})$	ROL r
ROTATE RIGHT	111111	$(ACr_7) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$	$(ACr_7) \leftarrow (L)$, $(L) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$	ROR r
SHIFT LEFT	111010	$(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$	$(L) \leftarrow (ACr_7)$, $(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$	SHL r
SHIFT RIGHT	111011	$(ACr_7) \leftarrow 0$, $(ACr_n) \leftarrow (ACr_{n+1})$	$(L) \leftarrow 0$, $(ACr_7) \leftarrow (L)$, $(ACr_n) \leftarrow (ACr_{n+1})$	SHR r

BYTE 1

7	6	5	4	3	2	1	0
OP CODE						r	

Specifies operation.

Register designator. Designates number of working register (AC0, 1, 2, or 3) involved in operation.

Figure 3-8. Shift Instruction Format

Rotate Left (ROL)

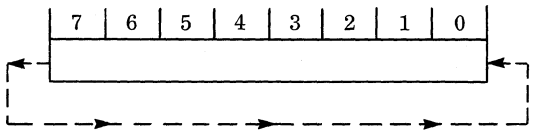
BYTE 1

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

SEL = 0

Operation: $(ACr_0) \leftarrow (ACr_7)$, $(ACr_n) \leftarrow (ACr_{n-1})$

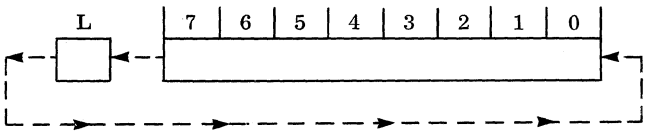
Description: The contents of ACr are shifted 1 bit to the left.



SEL = 1

Operation: $(ACr_0) \leftarrow (L)$, $(L) \leftarrow (ACr_7)$, $(ACr_n) \leftarrow (ACr_{n-1})$

Description: The contents of ACr are shifted around to the left one bit. (L) replaces (ACr_0), and (ACr_7) replaces (L).



Rotate Right (ROR)

BYTE 1

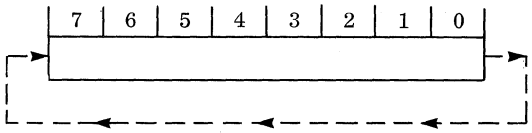
7	6	5	4	3	2	1	0
1	1	1	1	1	1	r	

Rotate Right (Continued)

SEL = 0

Operation: $(ACr_7) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$

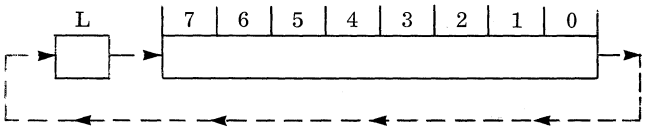
Description: The contents of ACr are shifted around to the right one bit. (ACr_0) replaces (ACr_7) for each shift.



SEL = 1

Operation: $(ACr_7) \leftarrow (L)$, $(L) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$

Description: The contents of ACr are shifted around to the right one bit. (L) replaces (ACr_7), and (ACr_0) replaces (L).



Shift Left (SHL)

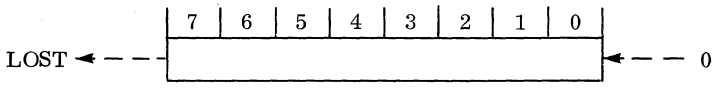
BYTE 1

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

SEL = 0

Operation: $(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$

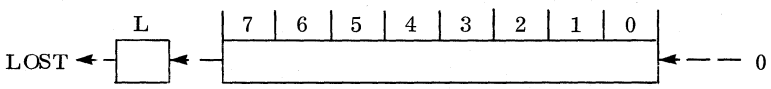
Description: The contents of ACr are shifted to the left one bit. (ACr_7) is lost, and zero replaces (ACr_0) for each shift.



SEL = 1

Operation: $(L) \leftarrow (ACr_7)$, $(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$

Description: The contents of ACr are shifted to the left one bit. (L) is lost. (ACr_7) replaces (L), and zero replaces (ACr_0).



Shift Right (SHR)

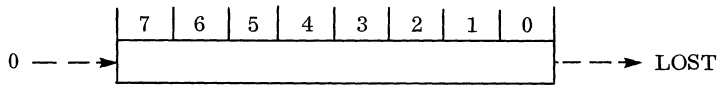
BYTE 1

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

SEL = 0

Operation: $(ACr_7) \leftarrow 0$, $(ACr_n) \leftarrow (ACr_{n+1})$

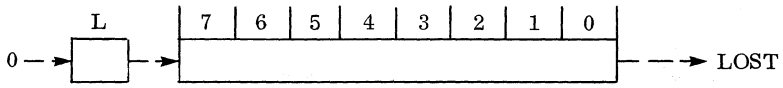
Description: The contents of ACr are shifted to the right one bit. (ACr_0) is lost, and zero replaces (ACr_7).



SEL = 1

Operation: $(L) \leftarrow 0$, $(ACr_7) \leftarrow (L)$, $(ACr_n) \leftarrow (ACr_{n+1})$

Description: The contents of ACr are shifted to the right one bit. (ACr_0) is lost, (L) replaces (ACr_7), and zero replaces (L).



3.6.7 Register Instructions

There are ten instructions in this group, summarized in table 3-10. Three word formats are required and are shown in figure 3-9.

Table 3-10. Register Instructions

Instruction	Operation Code	Operation	Assembler Format
Register and Stack			
PUSH ONTO STACK	110011	(STK0) ← (ACr)	PUSH r
PULL FROM STACK	110110	(ACr) ← (STK0)	PULL r
EXCHANGE REGISTER AND STACK	110010	(STK0) ← (ACr), (ACr) ← (STK0)	XCHRS r
Register and Immediate			
LOAD IMMEDIATE	001000	(ACr) ← disp	LI r, data
ADD IMMEDIATE, SKIP IF ZERO	001100	(ACr) ← (ACr) + disp, if (ACr) = 0, (PC) ← (PC) + 2	AISZ r, data
TWOs COMPLEMENT	110111	(ACr) ← ~ (ACr) + 1	COMP2 r

Table 3-10. Register Instructions (Continued)

Instruction	Operation Code	Operation	Assembler Format
Register to Register			
REGISTER ADD	0100	(ACdr) ← (ACsr) + (ACdr), OV, CY	RADD sr, dr
REGISTER EXCHANGE	0110	(ACsr) ← (ACdr), (ACdr) ← (ACsr)	RXCH sr, dr
REGISTER EXCLUSIVE-OR	0101	(ACdr) ← (ACsr) ∇ (ACdr)	RXOR sr, dr
REGISTER AND	0111	(ACdr) ← (ACsr) ∧ (ACdr)	RAND sr, dr

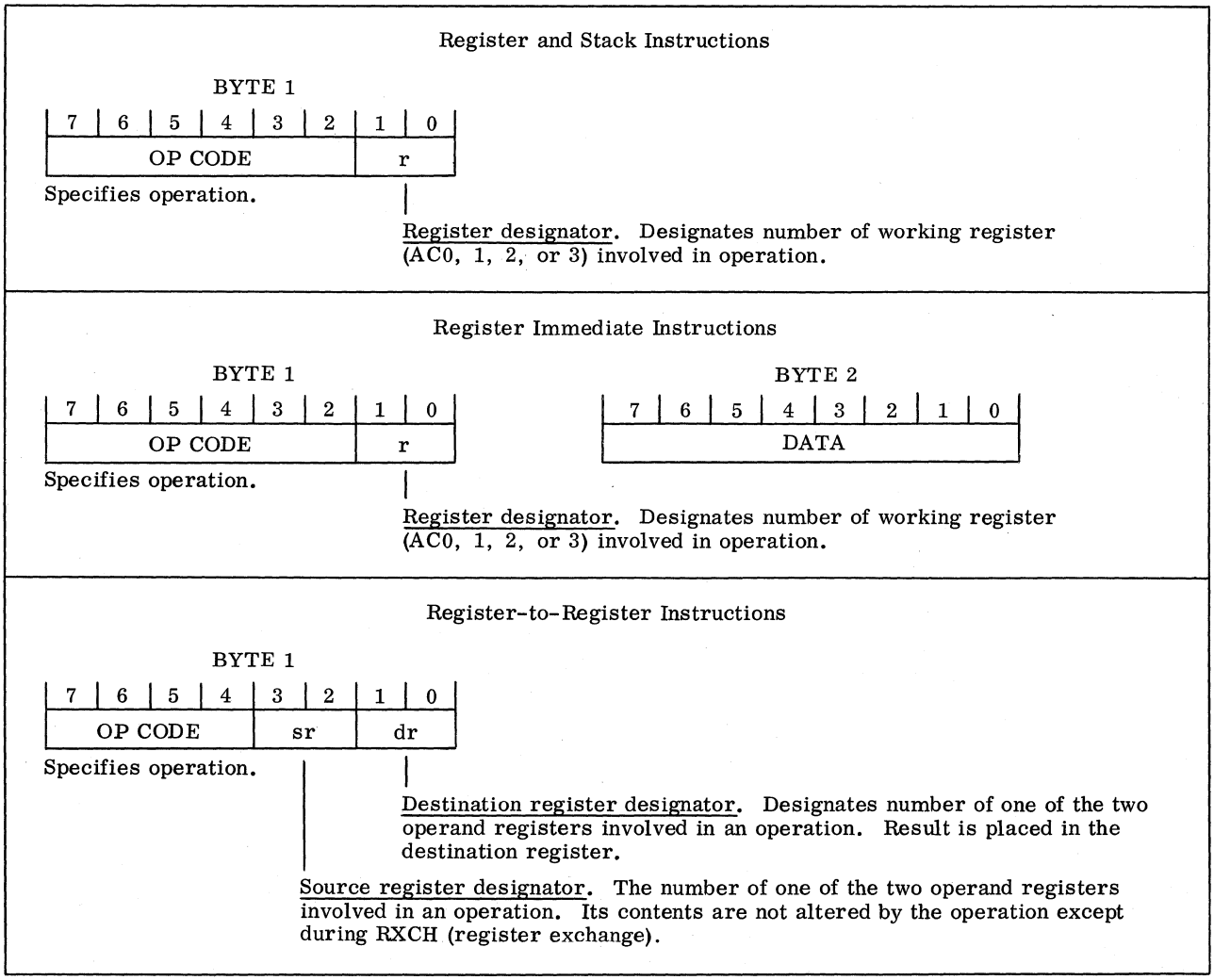


Figure 3-9. Register Instruction Formats

Push onto Stack (PUSH)

BYTE 1

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

Operation: (STK0) ← (ACr)

Description: The stack is pushed by the contents of the register (AC0, 1, 2, or 3) designated by r. Thus, the top of the stack then holds the contents of ACr, and the contents of all other levels in the stack are moved down one level. If the stack is full before the push occurs, the contents of the lowest level are lost. The initial contents of ACr are unaltered.

Pull from Stack (PULL)

BYTE 1

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

Operation: (ACr) ← (STK0)

Description: The stack is pulled. The contents from the top of the stack replace the contents of register number r (AC0, 1, 2, or 3). The initial contents of ACr are lost. The contents of each level of the stack moves up one level. Zeros enter the bottom of the stack.

Exchange Register and Stack (XCHRS)

BYTE 1

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

Operation: (STK0) ← (ACr), (ACr) ← (STK0)

Description: The contents of the top of the stack and the register designated by r (AC0, 1, 2, or 3) are exchanged.

Load Immediate (LI)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
data							

Operation: (ACr) ← data

Description: The value of data replaces the contents of ACr (AC0, 1, 2, or 3). The initial contents of ACr are lost. The immediate operand range is -128 to +127.

Add Immediate, Skip If Zero (AISZ)

BYTE 1

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

BYTE 2

7	6	5	4	3	2	1	0
data							

Operation: (ACr) ← (ACr) + data
If new (ACr) = 0, (PC) ← (PC) + 2

Description: The contents of register ACr are replaced by the sum of the contents of ACr and data. The initial contents of ACr are lost. If the new contents of ACr equal zero, the contents of PC are incremented by 1, thus skipping the next memory location. The immediate operand range is -128 to +127.

Twos Complement (COMP2)

BYTE 1

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

Operation: (ACr) ← ~ (ACr) + 1

Description: The contents of register ACr are twos-complemented. The result is then stored in ACr. The initial contents of ACr are lost.

Register Add (RADD)

BYTE 1

7	6	5	4	3	2	1	0
0	1	0	0	sr		dr	

Operation: (ACdr) ← (ACsr) + (ACdr), OV, CY

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the sum of the contents of ACdr and the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the contents of ACsr are unaltered. The Overflow and Carry Flags are set according to the result of the operation.

Register Exchange (RXCH)

BYTE 1

7	6	5	4	3	2	1	0
0	1	1	0	sr		dr	

Operation: (ACsr) ← (ACdr), (ACdr) ← (ACsr)

Description: The contents of ACsr (AC0, 1, 2, or 3) and ACdr (AC0, 1, 2, or 3) are exchanged.

Register Exclusive Or (RXOR)

BYTE 1

7	6	5	4	3	2	1	0
0	1	0	1	sr		dr	

Operation: (ACdr) ← (ACdr) ∇ (ACsr)

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the result of exclusively ORing the contents of ACdr with the contents of the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the initial contents of ACsr are unaltered.

Register And (RAND)

BYTE 1

7	6	5	4	3	2	1	0
0	1	1	1	sr		dr	

Operation: (ACdr) ← (ACdr) ∧ (ACsr)

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the result of ANDing the contents of ACdr with the contents of the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the initial contents of ACsr are unaltered.

3.6.8 Miscellaneous Instructions

Instructions that provide nonmemory control and status operations are listed in table 3-11. Word format is shown in figure 3-10.

Table 3-11. Input/Output, Halt, and Flag Instructions

Instruction	Operation Code	Operation	Assembler Format
Halt			
HALT	00000000	Processor halts.	HALT
Status Flags			
PUSH STATUS FLAGS ONTO STACK	00000111	(STK0) ← (STATUS FLAGS)	PUSHF
PULL STATUS FLAGS FROM STACK	00001110	(STATUS FLAGS) ← (STK0)	PULLF
Control Flags			
SET FLAG	00101	fc set	SFLG fc
PULSE FLAG	00111	fc pulsed	PFLG fc

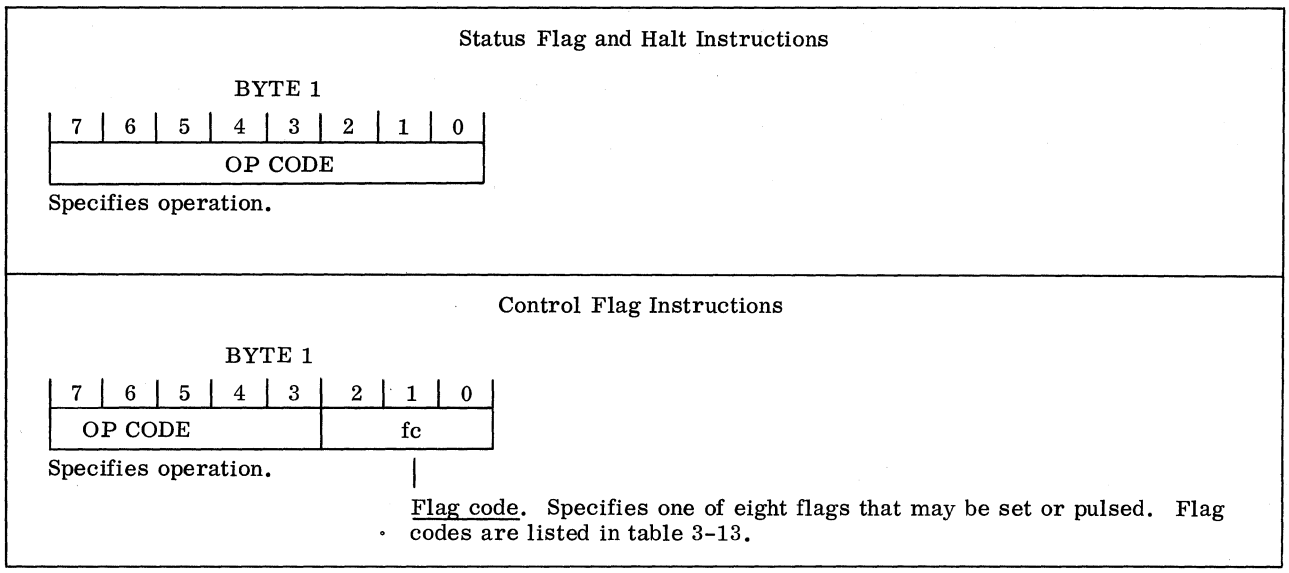


Figure 3-10. Input/Output, Halt, and Flag Instruction Formats

Halt (HALT)

BYTE 1

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

Description: The processor halts and remains halted until the START input makes a transition from logic 1 to 0. A switch may be wired to this input such that a logic 1 is applied momentarily.

Push Status Flags onto Stack (PUSHF)

BYTE 1

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

Operation: (STK0) ← (STATUS FLAGS)

Description: The contents of the top of the stack are replaced by the contents of the status flags. See figure 3-11 for the configuration of processor flags on the stack and table 3-12 for processor flag definitions. The previous contents of the top of the stack and lower levels are pushed down one level. The contents of the lowest level of the stack are lost.

Pull Status Flags from Stack (PULLF)

BYTE 1

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

Operation: (STATUS FLAGS) ← (STK0)

Description: The contents of the status flags are replaced by the contents of the top of the stack. See figure 3-11 for the configuration of processor flags on the stack and table 3-12 for processor-flag definitions. The previous contents of lower levels of the stack are pulled up by one level with zeros replacing the contents of the lowest level.

7	6	5	4	3	2	1	0	Bit Positions
FLAG								
L	OV	CY	GF	GF	GF	GF	GF	Flags

Figure 3-11. Configuration of Status Flags

Table 3-12. Status Flags

Bit Position	Flag Name	Mnemonic	Significance
7	Link	L	Used for double-byte shifts
6	Overflow	OV	Set if an arithmetic overflow occurs
5	Carry	CY	Set if a carry occurs (from most significant bit) during an arithmetic operation
4 through 0	General-purpose Flags	GF	Use specified by programmer

Set Flag (SFLG)

NOTE

(1) SFLG and PFLG refer to control flags external to the RALUs. These flags should not be confused with the status flags internal to the RALU, which are referenced by PUSHF and PULLF.

(2) Pulsing a control flag sets the flag at T2 and resets it at T6 during the same microcycle. The flag remains reset until again set or pulsed.

BYTE 1

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

Operation: fc set

Description: The control flag designated by the flag code fc is set. Flag codes are defined in table 3-13.

Pulse Flag (PFLG)

BYTE 1

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

Operation: fc pulsed

Description: The control flag designated by the flag code fc is pulsed. Flag codes are defined in table 3-13.

Table 3-13. Control Flag Codes

fc	Flag Output
0	(Used by Micro-Program)
1	USER 1 -- User Flag 1
2	USER 2 -- User Flag 2
3	USER 3 -- User Flag 3
4	USER 4 -- User Flag 4
5	(Used by Micro-Program)
6	SEL
7	INEN -- Interrupt Enable

Chapter 4

CIRCUIT DESCRIPTIONS

This chapter provides circuit level descriptions of the functional blocks that comprise the IMP-8C/200. Figure 4-10 is a functional block diagram of the IMP-8C/200 circuits and their interrelationship. A more detailed presentation of the individual circuits is contained in the IMP-8C/200 Microprocessor logic diagram, figure 4-11. The sheet number in the functional block of figure 4-10 corresponds to the sheet of the logic diagrams which contains the circuit for that specific functional block. As a convenience to the user of this manual, the functional block diagram and the four sheets of the logic diagram are located at the end of this chapter so that they may be readily referenced during the following circuit-level descriptions. Additionally, a parts listing and component layout of the IMP-8C/200 assembly are presented in table 4-3 and figure 4-12, respectively.

4.1 MASTER OSCILLATOR (Figure 4-11, Sheet 1)

The Master Oscillator provides the Master Clock Signal for the generation of the system clocks and synchronization of external devices. The Master Clock Signal (CLK) is generated by a 5.7143-megaHertz crystal-oscillator circuit made from a DM10116 triple line receiver connected as an amplifier and a schmitt trigger. The DM10116 generates a symmetrical 175-nanosecond master clock. Two transistors, Q3 and Q4, provide level shifting to convert the ECL levels of the DM10116 to TTL compatible logic signals. Additional buffering is furnished by a DM74H04 inverter whose output provides the Output Clock Signal (OCLK) to the card-edge connector and to a DM74H00 NAND gate. The DM74H00 NAND gate provides the clock to the System Clock Generator unless inhibited during initialization or a memory access microcycle by the signals INIT* or CHOLD*, respectively. The signals INIT* and CHOLD*, their purpose and their effect, are discussed under System Initialization and Clock Hold Logic descriptions. Signal names followed by an asterisk (*) denote active state is low.

4.2 SYSTEM CLOCK GENERATOR (Figure 4-11, Sheet 1)

The System Clock Generator provides the four phase clocks required for the CPU devices, one CROM and two RALUs, and other timing signals required by the IMP-8C/200 and the external devices. The output from the Master Oscillator is the clock for a DM8570 8-bit shift register connected as a ring counter. The DM8570 shift register generates eight Clock Signals, C1 through C8, each of which lasts for one time period. The odd interval clock signals are buffered by DM74H08 AND gates to drive the MH0026 clock drivers which provide the four phase clocks: PH1, PH3, PH4, and PH7. The MH0026 clock drivers are capable of driving a 1000-pfd load with 20-nanosecond rise and fall times. The Resistors (R17 through R24), in the output of the clock drivers, dampen oscillations in the clock lines due to the line inductance.

Figure 4-1 shows the timing pulses generated by the System Clock Generator during one microcycle.

During system initialization, the DM8570 shift register is cleared by the Initialize Signal, INIT*. The first clock pulse from the Master Oscillator sets C1. The next clock pulse resets C1 and sets C2. Subsequent clock pulses continue to reset and set the C2 through C8 Signals in a similar fashion. When C8 is reset, C1 is set, thereby starting the sequence for the next microcycle.

Additional timing pulses are generated from the basic timing signals by gating the appropriate shift register outputs and Master Clock Signals. The mnenomic used to designate these additional timing pulses denotes the corresponding time interval for which the signal is valid. For example; C34 refers to a timing pulse that is high during T3 and T4, Similarly, C4F is a buffered C4; C5B is a buffered clock formed by ANDing CLK and C5; and C81234 is high during T8 and T1 through T4 (not C5 NOR C6 NOR C7). These additional timing pulses are discussed under the functional block where they are used.

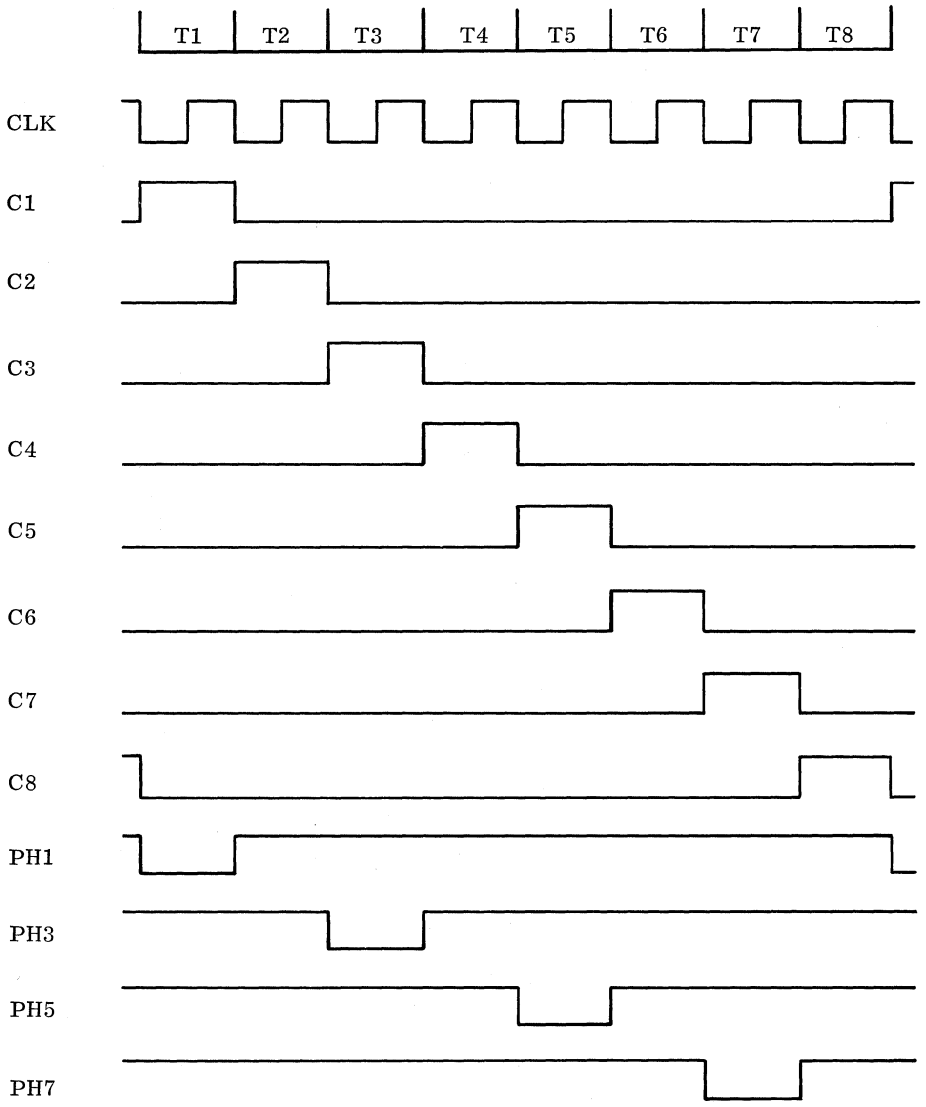


Figure 4-1. System Clock Generator Timing

4.3 MOS/LSI CPU LOGIC (Figure 4-11, Sheet 2)

The CPU consists of one CROM and two RALUs driven by the four phase clocks (PH1, PH3, PH5, and PH7). Control between the CROM and the RALUs is effected over the complemented Control Bit Lines, NCB(0) through NCB(3). The CROM controls the operations of the CPU by routines that constitute the microprogram stored in the CROM Read-Only Memory. The actual data processing operations of the CPU take place in the RALUs.

4.3.1 Control and Read-Only Memory (CROM)

The IMP-8A/520 Control and Read-Only Memory Unit (CROM) is a monolithic MOS/LSI circuit in a 24-pin dual in-line package. The CROM contains the control circuitry for program sequencing, subroutine execution, and translation of microinstructions into RALU commands. The CROM also contains the 100 by 23-bit word read-only memory containing the microinstructions. The CROM operates on +5 vdc and -12 vdc. Signals interfacing with the RALUs are MOS levels, while those that interface with other circuits of the IMP-8C/200 are TTL

levels. Details of signal functions and timing are presented in the following paragraphs. Positive true logic signals are used ('1' = more positive voltage = high, '0' = more negative voltage = low). Signal names beginning with N are complemented signals. Figure 4-2 shows the simplified block and connection diagrams for the CROM.

The command outputs to the RALU occur on pins 20, 18, 19, and 17 which correspond to command bits NCB(0), (1), (2), and (3). The command outputs are complemented MOS signals and are multiplexed over the four odd time intervals in each cycle (T_1, T_3, T_5, and T_7). Outputs are driven negative to logic '0' during the even time intervals. The command functions for each bit are indicated in the diagram. During T_1, the three least significant command bits specify the address of the register to be loaded onto the A-bus. The RALU registers are addressed by binary values of 1 - 7. A value of zero causes the A-bus to be set equal to zero. The fourth command bit is used to enable stack operations. If NCB(3) is at a logic '1' (most positive level), no stack operation occurs. If it is at a logic '0', stack operations are enabled, but only occur if the A- or R-bus address is zero. If the A-bus address is zero, the stack is pulled onto the A-bus. If the R-bus address is zero, the R-bus is pushed onto the stack. During T_3, the three least significant bits specify the address of the register to be loaded onto the B-bus. The most significant bit specifies that the A-bus is to be complemented when it is transferred to the ALU. During T_5, NCB(0) and NCB(1) specify the ALU operation to be performed, while NCB(2) and NCB(3) are used to specify control functions. During T_7, the three least significant bits specify the address of the register to be loaded from the R-bus. The most significant bit specifies that the R-bus is to be set equal to the output of the I/O Multiplexer rather than the Shifter.

Instructions to the CROM are transferred over the CROM Data Input Lines, DI(0) - DI(7), into bits 0 - 7 of the CROM Instruction Register. Data input occurs at T_7 when the Data Out Lines DO0 through DO3 are strobed to the CROM data input by DISTR* at T7 via DM8097 AND gates. DO4 through DO7 are not affected by DISTR*. As with all TTL inputs on the CROM, a 3k - 5k pull-up resistor is provided on the chip to ensure an adequate logic '1' level. The pull-up is provided by an MOS transistor which is turned on only during the data input interval (T_7). At other times, it is in the "off" or high impedance state. CROM Signal Lines DI(0) - DI(3) are also used to output a 4-bit address, JFA0 through JFA3, to the control flags and jump conditions. This address output becomes valid during T_1 and is stored in the Jump/Flag Address Latch.

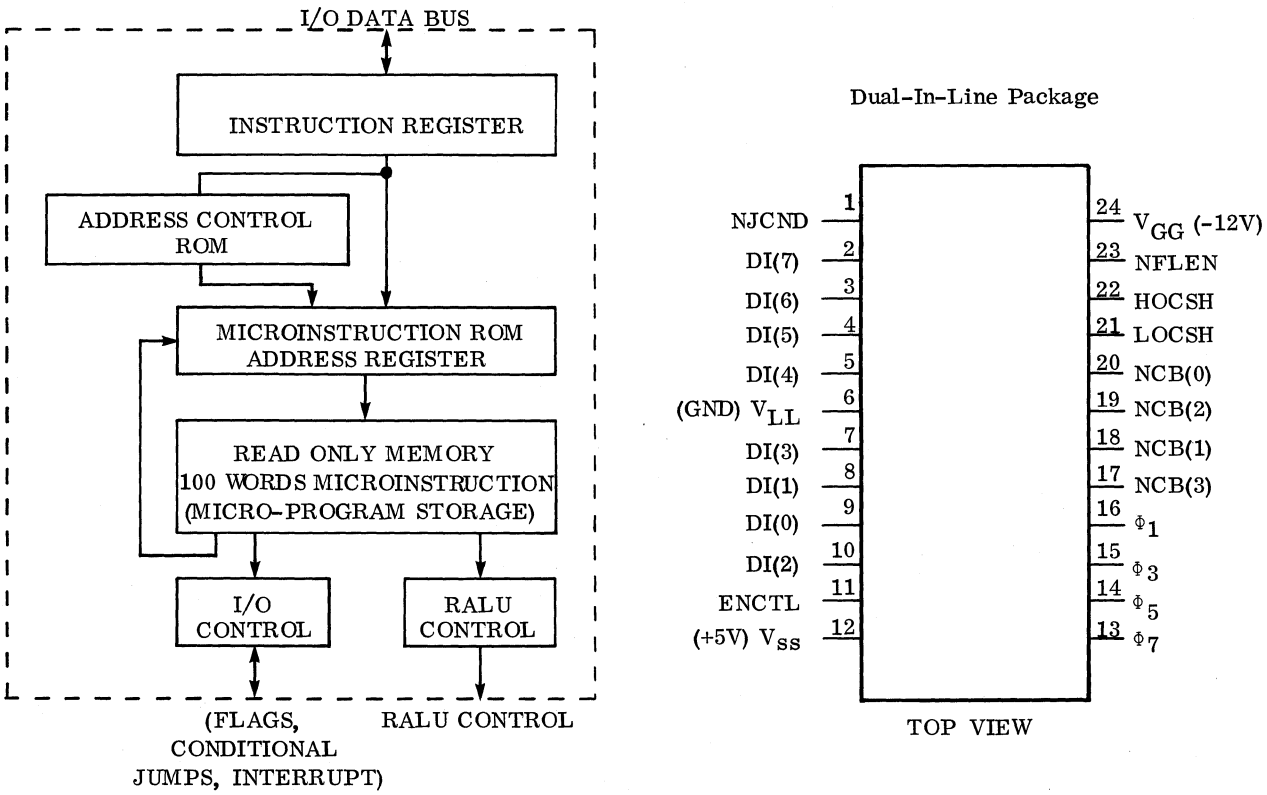


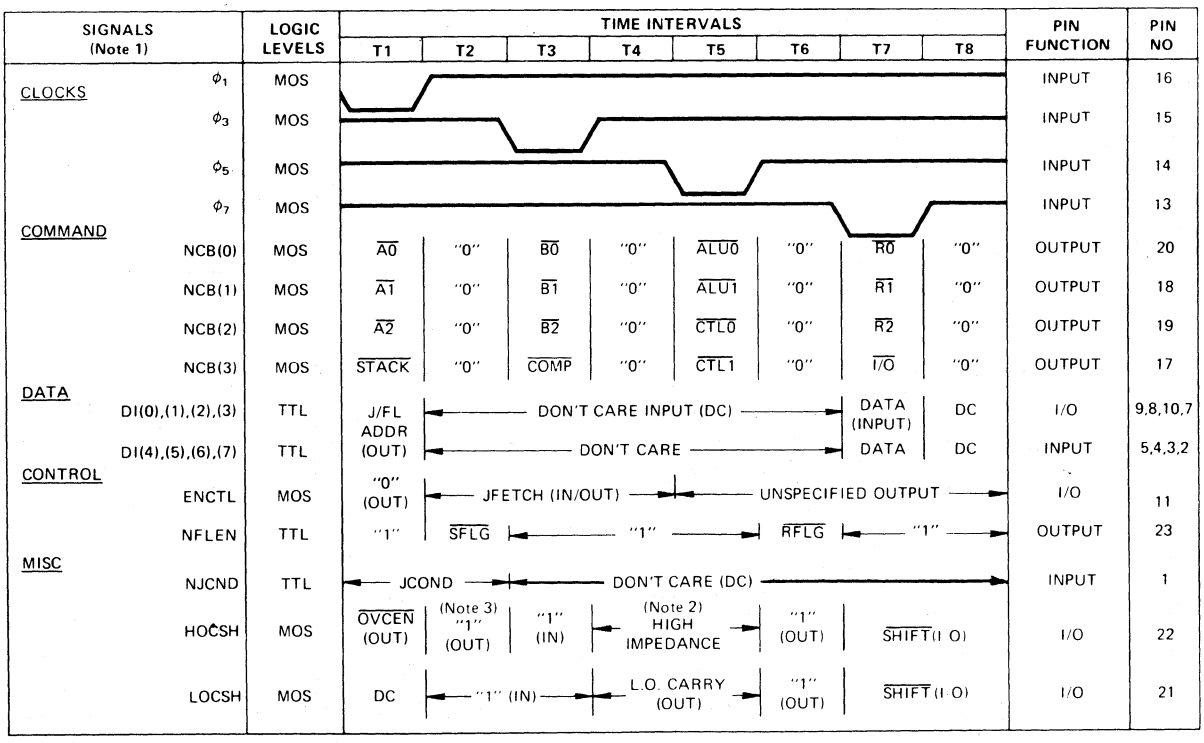
Figure 4-2. CROM Simplified Block and Connection Diagrams

The Enable Control (ENCTL) is a "wired-or" signal line required for operation of multiple CROMs. ENCTL provides a logic '1' output at T_2 - T_4 whenever a branch to the instruction fetch routine occurs, and responds to a logic '1' input at T_2 - T_4 by executing a branch to the instruction fetch routine and disabling the CROM if the instruction has not been implemented in that particular CROM. The Flag Enable (NFLEN) control output is used to set or reset flags addressed by DI(0) - DI(3) at T_1 of the current cycle. A logic '0' output at T_2 specifies setting of the addressed flag, while a logic '0' output at T_6 specifies resetting.

The Jump Condition Input Line (NJCND) is used to input conditional branch information at T_1 and T_2 as specified by the jump condition addressed by DI(0) - DI(3) at T_1. If the input is a logic '0' and a conditional branch is specified for the current cycle, a branch occurs.

The High and Low Order Carry/Shift Lines (HOCSH and LOCSH, respectively) are used to provide shift and carry information to the RALU chips. If a circular shift is specified, HOCSH and LOCSH are connected together on the CROM during T_7 and T_8 by a low on-resistance MOS transistor switch. For a circular left shift HOCSH serves as an input driven by the most significant RALU, and LOCSH serves as an output to the least significant RALU at the same voltage level as HOCSH. The direction of data transfer is reversed for circular right shift. In the case of open ended shifts, the CROM output (LOCSH for left shift and HOCSH for right shift) provides a logic '1' to the RALU (since the shift data is complemented, this provides a trailing '0' for the shift operation) and ignores the shift input data from the RALU. The carry input to the least significatnt RALU is provided by LOCSH at T_4 and T_5. The Overflow and Carry Flags on the RALU are enabled by the output of HOCSH at T_1. A logic '1' input is required for HOCSH at T_3. This line is precharged to a logic '1' at T_2. A logic '1' input is required for LOCSH at T_2 and T_3. The LOCSH input is preset to a logic '1' at T_1 when connected to CSH0 of a RALU.

Figure 4-3 shows the timing of CROM signals.



Note 1: A positive true logic convention is used for all signals except clocks. Signal names beginning with N are complemented signals.

Note 2: HOCSH at T4 and T5 is in the TRI-STATE high impedance output mode of CROM load drivers.

Note 3: "1" (OUT) means CROM is driving this node to the logic "1" level during the defined interval. For I/O lines the logic state is defined as "in" or "out." Input or output nodes are defined only as "1" or "0."

Figure 4-3. CROM Timing Diagram

4.3.2 Register and Arithmetic Logic Unit (RALU)

The IMP-00A/520 Register and Arithmetic Logic Unit (RALU) is a monolithic MOS/LSI circuit in a 24-pin dual in-line package. Each RALU provides a 4-bit slice of the 8-bit register and arithmetic portion of the CPU. The RALUs perform add, AND, OR, and EXCLUSIVE-OR operations on the data from the registers. The register data can be complemented by the RALU prior to performing the operation. A Shifter is provided for single-bit left and right shift and an I/O Data Multiplexer for communications with the Data Bus. Signals interfacing with the CROM are MOS levels, while those that interface with other circuits of the IMP-8C/200 are TTL levels. Figure 4-4 shows the simplified block and connection diagrams for the RALU. Details of the signal functions and timing are presented in the following paragraphs.

The command inputs to the RALU occur on pins 21, 19, 18, and 20 which correspond to Command Bits NCB(0), (1), (2), and (3), respectively. The command inputs are multiplexed over the four odd-time intervals in each RALU cycle (T_1, T_3, T_5, and T_7). The inputs are driven negative to logic '0' during the even time intervals. During T_1, the three least significant Command Bits specify the address of the RALU register to be loaded on the A-bus. If NCB(0), (1), and (2) are all logic '0', the A-bus is set equal to zero. The fourth Command Bit NCB(3) is used to enable stack operations. If NCB(3) is at a logic '1', no stack operations occur. If it is a logic '0', stack operations are enabled but only occur if the A- or R-bus address is zero. If the A-bus address is zero, the stack is pulled onto the A-bus. If the R-bus address is zero, the R-bus is pushed onto the stack. During T_3, the three least significant bits specify the address of the RALU register to be loaded on the B-bus. (Note that stack and flags cannot be accessed over the B-bus. An address of zero always gives zero data; NCB(0), (1), (2), and (3) are all 1's.) The most significant bit specifies that the A-bus is to be complemented when it is transferred to the ALU. During T_5, NCB(0) and NCB(1) specify the ALU operation to be performed; NCB(2) and NCB(3) are used to specify control functions. The R-bus control code (NCB(2) = '1'; NCB(3) = '0') is used in conjunction with the I/O Control Bit, NCB(3) at T_7, and the byte input (SININ at T_5) to set the value of the R-bus. During T_7, the three least significant bits specify the address of the RALU register to be loaded from the R-bus. The most significant bit specifies that the R-bus is to be set equal to the output of the I/O Multiplexer rather than the Shifter (unless R-bus control was specified at T_5).

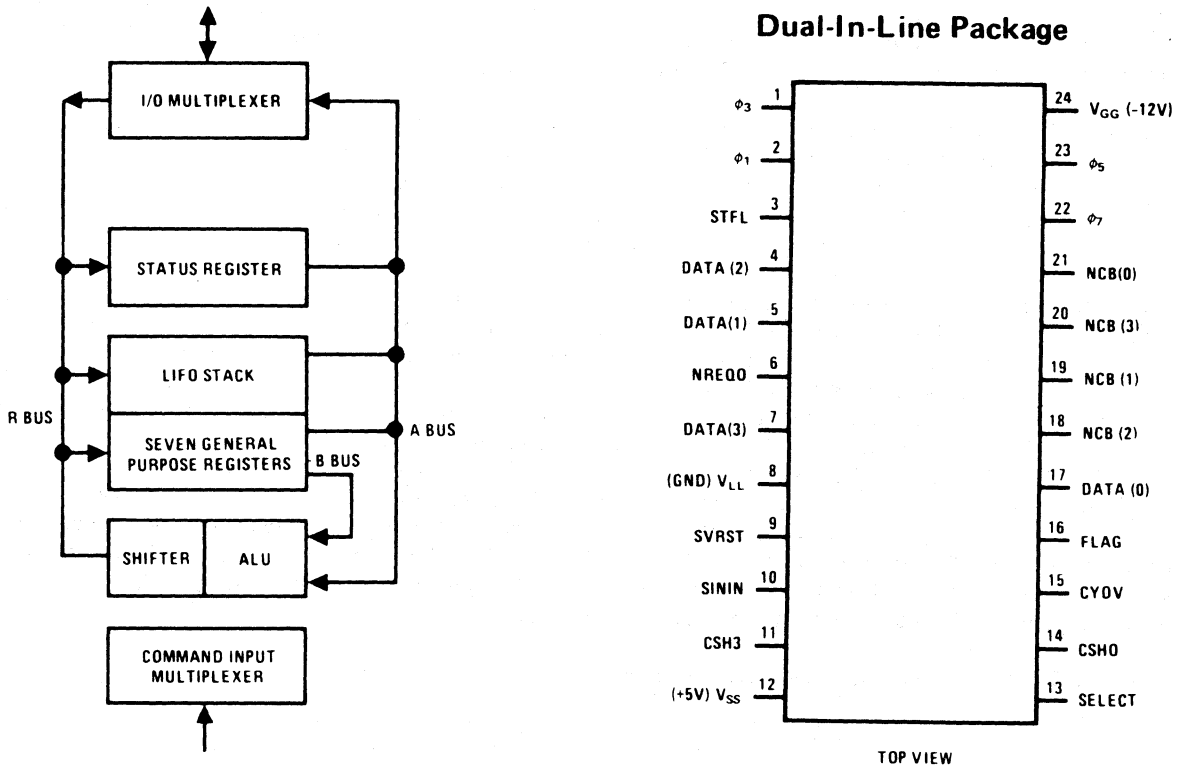


Figure 4-4. RALU Simplified Block and Connection Diagrams

The data transfers between the RALU and memory or peripheral devices occur on pins 17, 5, 4, and 7 which correspond to Data Bits DATA(0), (1), (2), and (3), respectively. During T_1 and T_2, the data lines are driven with the value of the R-bus which occurred at the end of the previous timing cycle. This output may be used by the CROM chip for conditional branch inputs. During T_3 and T_4, the data lines are driven with the value that was loaded onto the A-bus during the current timing cycle. This output is typically used for address and data output to system memory or peripheral devices. During T_5 and T_6, the data lines are driven to a logic '1'. During T_7, the data lines are used for input to the RALU from system memory or peripheral devices. During T_8, the data lines are again driven to a logic '1' by the RALU. As with all TTL inputs on the RALU, a 3K - 5K pull-up is provided on the chip to ensure an adequate logic '1' level. The pull-up is provided by an MOS transistor which is turned on only during the data input interval. At other times, it is in the "off" or high impedance state.

The RALU control lines provide a means of using the RALU Status Flags. The SELECT Line SEL is used as an input at T_5 and is unused at other times. If the SEL is a logic '1' at T_5, the Overflow Status Flag is selected as the output on the Carry or Overflow (CYOV) Line (pin 15) during the following cycle. If the RALU is in the most significant 4 bits of the processor (as specified by the byte input on the SININ Line), the Link Status Flag is included in any shift that occurs in the current cycle. The shift is a 5-bit shift with the link in the most significant bit position. If the select input is a logic '0' at T_5, the Carry Status Flag is selected at the CYOV output, and shift operations do not affect the link.

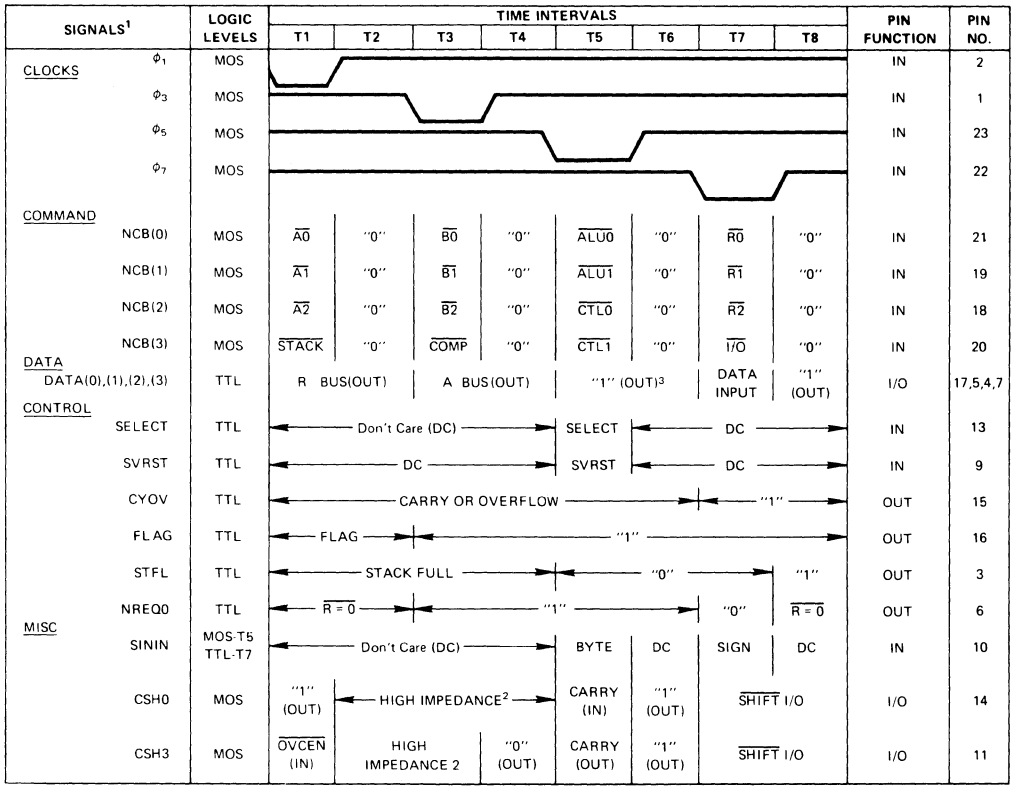
The Save/Restore (SVRST) Line is used as an input during T_5 and provides a means of modifying the status flags over the Data Bus. If SVRST is a logic '1' during T_5, the status flags are loaded onto the A-bus during T_1 of the following cycle, provided the A-bus address bits NCB(2), (1), and (0) at T_1 during that cycle are a logic '1'. If a pull stack operation is specified by NCB(3), (2), (1), and (0) at T_1, the SVRST input at T_7 inhibits it and instead the status flags are loaded on the A-bus. The SVRST Line also causes the status flags to be loaded from the R-bus at the end of the following cycle. (The status of SVRST during one cycle only affects conditions in the following cycle.) This occurs in parallel with the loading of any other register specified by the A-bus address.

The CYOV Line provides an output signal indicating the state of the Carry or Overflow Status Flag as determined by the select input. The flag output indicates the state of the general-purpose flag. The Stack-Full (STFX) output goes true when the bottom word of the stack is nonzero at the start of the preceding cycle. The Result Bus Equals Zero (NREQ0) output goes to logic '0' level when the R-bus contains all zeros. The CYOV, STFX, and NREQ0 outputs are tied to -12 vdc through R25, R26, and R27, respectively.

The SININ Line is used to input information as to whether the RALU is in the most or least significant 4 bits of a processor word and also the sign value which is propagated to the most significant 4 bits when the R-bus control function is specified. If SININ is a logic '1' at T_5, the RALU enables the functions of the most significant 4 bits. The functions enabled are inclusion of the link in shift operations and setting the R-bus to zero or sign as specified by the R-bus control code. If SININ at T_5 is a logic '0', the functions are not enabled.

The Carry Input (CSH0) and Carry Output (CSH3) Lines are used primarily for transfer of carry and shift information between RALUs or between RALU and CROM. If an add operation is specified, the value of carry input (CSH0) at T_5 is added to the A-bus inputs and B-bus inputs to the ALU. The resulting carry output from the most significant bit occurs from CSH3 at T_5. When a left shift is specified, the shift output from the most significant bit occurs on CSH3 at T_7 and T_8, while the shift input to the least significant bit must be provided on CSH0 during T_7 and T_8. The pins exchange roles for a right shift. During T_1, if the CSH3 input is a logic '0' , the Overflow and Carry Flags are enabled to be set to the result of an add operation. The Carry Flag is set equal to the value of the ripple carry out of the most significant bit of the ALU. The Overflow Flag is set if there is a twos-complement arithmetic overflow (sign of both operands the same and sign of the result different). For systems using multiple RALUs, the Overflow and Carry Flags of all but the most significant RALU are disabled by the logic '1' output of CSH0 generated by the adjacent RALU at T_1. These flags may therefore be used for general-purpose functions.

Figure 4-5 shows the timing of the RALU signals.



Note 1: A positive true logic convention is used for all signals (i.e., "1" = more positive voltage, "0" = more negative voltage). Signal names beginning with N are complemented signals.

Note 2: CSH0 and CSH3 high impedance state for intervals T_2 through T_4 is the TRI-STATE mode for output drivers.

Note 3: "1" (OUT) means RALU is driving this node to the "1" logic level during the defined interval. For bidirectional I/O lines the logic state is defined as "in" or "out."

Figure 4-5. RALU Timing Diagram

4.4 CONTROL FLAGS AND CONDITIONAL JUMP MULTIPLEXER LOGIC

External to the CPU portion of the IMP-8C/200 is the logic required to set and reset the control flags and to select 1 of 16 jump conditions. The Control Flags and Conditional Jump Multiplexer Logic consists of the following functional blocks on figure 4-11, sheet 2:

- Jump/Flag Address Latch
- Jump Multiplexer
- Stack-Full Control
- Control Flags

4.4.1 Jump/Flag Address Latch (Figure 4-11, Sheet 2)

The Jump/Flag Address Signals (JFA0 through JFA3) from the CROM share the same lines as the data input to the CROM from DO0 through DO3. During T1, the Jump/Flag Address Signals are available on the CROM bi-directional Data Input Lines. In order to retain the address for the remainder of the microcycle, they are latched by a DM9322 multiplexer connected as a latch by feeding the outputs of the DM9322 back to the second set of inputs. The contents of JFA0 through JFA3 are latched by C1 from the System Clock Generator. LJFA0 through LJFA3 are the latched address outputs to the Jump Multiplexer and control flags.

4.4.2 Jump Multiplexer (Figure 4-11, Sheet 2)

The Latched Jump/Flag Address Signals, LJFA0 through LJFA3, are used to select one of 16 jump conditions in two DM8121 8-to-1 multiplexers. Table 4-1 is a listing by jump condition address of the signal and condition selected.

A DM7475 4-bit latch is used to store the status at T8 of the asynchronously received Jump Condition Signals at the card-edge connector. The Interrupt Request (INTRQ), Start Switch (START), and User Jump Condition (USERJC) are latched into the DM7475 by the T8 clock (C8), thereby holding the status of the external jump condition stable through the next microcycle.

The outputs of the DM8121 multiplexers are tied directly to the Jump Condition (NJCOND) input of the CROM. The jump condition is tested by the CROM at T2 during condition jump operations.

4.4.3 Stack-full Condition (Figure 4-11, Sheet 2)

The status of the RALU stack during the current microcycle is latched at T4 in order to allow the Status Signal (STF) to be checked by subsequent conditional jump operations. The Stack-Full Signal (STFX) goes high when the bottom entry in the stack is nonzero data. The Stack-Full Lines of the two RALUs are tied together to detect a nonzero condition in either of the 4-bit slices. The Stack-Full Signal (STFX) is ANDed with C34 to provide the clock to a DM7474 latch. The data input of the DM7474 is pulled up to +5 vdc which causes the DM7474 latch to be set when STFX goes high at any time during T3 or T4. The DM7474 latch remains set until cleared by a Flag Clear Signal (FLGCLR*) or an External Clear Signal (ESTKCL*) through a DM74H08 AND gate.

Table 4-1. Jump Conditions

LJFA3	LJFA2	LJFA1	LJFA0	Selected Signal	Condition
0	0	0	0	INT	Jump if interrupt set
0	0	0	1	NREQ0	Jump if (AC0) not equal to 0
0	0	1	0	REQ0	Jump if (AC0) equal to 0
0	0	1	1	DO7	Jump if bit 7 of AC0 set
0	1	0	0	DO1	Jump if bit 1 of AC0 set
0	1	0	1	CYOV	Jump if Carry/Overflow Flag set
0	1	1	0	STF	Jump if Stack-Full Flag set
0	1	1	1	START	Jump if Start Switch closed
1	0	0	0	DO0	Jump if AC0, bit 0 set
1	0	0	1	INEN	Jump if Interrupt Enable set
1	0	1	0	DO2	Jump if AC0, bit 2 set
1	0	1	1	DO3	Jump if AC0, bit 3 set
1	1	0	0	DO4	Jump if AC0, bit 4 set
1	1	0	1	DO5	Jump if AC0, bit 5 set
1	1	1	0	DO6	Jump if AC0, bit 6 set
1	1	1	1	USERJC	Jump if user condition set

0 = Low
1 = High

4.4.4 Control Flags (Figure 4-11, Sheet 2)

The control flags are stored in two DM9334 8-bit addressable latches. Table 4-2 is a listing of the control flags by address.

The two DM9334 8-bit latches are addressed under control of the microprogram by the Latched Jump/Flag Address Signals (LJFA0 through JLFA3). LJFA3 is used to select one of two DM9334 devices. LJFA0, 1, and 2 specify the bit assignment within the 9334 latch selected. The complemented Flag Enable Signal (NFLEN) from the CROM controls the setting of the flags. NFLEN is inverted by a 74H04 inverter to JLFA3 (LJFA3*) to enable the other DM9334 latch so that only one of the two can be enabled at any given time. C81234 holds the set input to the DM9334 latches high during T8, T1, T2, T3, and T4; therefore, during T2, if FLEN is high, the addressed bit is set. During T6, if FLEN is high, the set input to the DM9334 latches is low and the addressed bit is reset. Each bit is set and reset individually except during system initialization when the Flag Clear Signal (FLGCLR*0) provides a master clear to both DM9334 latches.

Table 4-2. Control Flags

LJFA3	LJFA2	LJFA1	LJFA0	Flag	Condition
0	0	0	0	RDFF	Read Flip-flop
0	0	0	1	WRFF	Write Flip-flop
0	0	1	0	RPC	Read Program Counter
0	0	1	1	ADX1	High-order Address Mode Flag
0	1	0	0	LPC	Load Program Counter
0	1	0	1	SVRST	Save Restore Flag
0	1	1	0	UNUS2	External Flag
0	1	1	1	HALT	Halt Flip-flop
1	0	0	0	BASEPG	Base Page Address Flag
1	0	0	1	USER1	User Flag
1	0	1	0	USER2	User Flag
1	0	1	1	USER3	User Flag
1	1	0	0	USER4	User Flag
1	1	0	1	PCENBL	Program Counter Enable
1	1	1	0	SEL	Select Flag
1	1	1	1	INEN	Interrupt Enable Flip-flop

0 = Low
1 = High

4.5 ON-CARD MEMORY

The IMP-8C contains an On-Card Memory consisting of a 256-byte Read/Write Memory and sockets for a 2048-byte programmable Read-Only Memory. The On-Card Memory consists of the following functional elements:

- Read/Write Memory
- Read-Only Memory

- Address Latches
- Address Decoder
- Clock Hold Logic
- Memory/Input Timing Logic

4.5.1 Read/Write Memory (Figure 4-11, Sheet 4)

The On-Card Read/Write Memory consists of eight MM1101 Random Access Memory (RAM) devices. Each MM1101 RAM contains 256 bits of memory individually addressable via an 8-bit Address Bus. The eight MM1101 RAM devices each provide one bit of the 8-bit Read/Write Memory byte. The On-Card Read/Write Memory is selected when the Random Access Memory Select Signal (RAMS*) at the card-edge connector goes low. In the basic IMP-8C, the On-Card Read/Write Memory is assigned the memory addresses corresponding to bytes 0 through 255 of the base page (page 0); therefore, the RAMS* input is externally jumpered to the Page 0 Enable Signal (EN0*) from the Address Decoder at their corresponding card-edge connector pins.

When the Read/Write Memory is selected (RAMS* is low) and the Write Strobe Signal (WRSTR) is low, the memory data at the address specified by the 8-bit address on AD0 through AD7 is available on the Memory Data Input Lines (MD0 through MD7). A write operation loads the contents of the Data Out Bus (DO0 through DO7) at the location specified by the Address Lines AD0 through AD7 when RAMS* is low and the Write Strobe (WRSTR) is high. The Write Strobe Signal (WRSTR) is generated by the Memory/Input Timing Logic.

4.5.2 Read-Only Memory (Figure 4-11, Sheet 4)

The On-Card Read-Only Memory may consist of eight MM5203 electrically Programmable Read-Only Memory (PROM) devices or eight MM5213 mask programmable Read-Only Memory devices. Each PROM contains 2048 bits of memory connected to provide 256 individually addressable 8-bit bytes per MM5203 device. The PROMs are selected by the Read-Only Memory Select Signals (ROMS0* through ROMS7*) at the card-edge connector pins. In the basic system, ROMS0* through ROMS7* are externally jumpered to EN8* through EN15* from the Address Decoder to provide a unique page address for each of the PROM devices. The byte address within the PROM is specified by Address Lines AD0 through AD7.

A read operation occurs when one of the Read-Only Memory Select (ROMSn*) Signals goes low. The contents of the location specified by the Address Lines AD0 through AD7, on the MM5203 (MM5213) selected by the ROMSn* Signal are available on the Memory Data Input Lines (MDI0 through MDI7). A description of the data transfer and timing during a read operation is contained in the discussion of the Memory/Input Timing Logic.

4.5.3 Address Latches (Figure 4-11, Sheet 3)

The Address Register Latches form a 16-bit register which is divided into two 8-bit segments. Each 8-bit segement is, in turn, made up of two DM8551 quad latches. The input to both 8-bit segments is lines DO0 through DO7. The high-order 8 bits are latched by the signal ADL1, which is generated through a DM74H08 gate at C4 when either the high-order Address Mode Flag (ADX1) or the Program Counter Enable Signal (PCOE) is high.

The low-order 8 bits are latched by the signal ADL2, which is generated through a DM74H08 gate at C4 time when the Read Flip-flop Flag (RDFF) is set.

The upper 8 bits are cleared at C4 when the Base Page Flag Signal (BASEPG) is set and the Data Output Bus bits DO0 and DO1 are low.

4.5.4 Address Decoder (Figure 4-11, Sheet 3)

The Address Decoder consists of a DM74154 4-to-16 line decoder and external memory selection logic. The decoder receives Address Lines AD8 through AD11 and signals IMS1* and IMS2* (Internal Memory

Select). To enable the decoder, signal IMS1* must be high and IMS2* must be low. The basic IMP-8C/200 system would have IMS1* tied high, IMS2* tied to PS (Peripheral Select), and PS tied to address bit 15.

The External Memory Select Signal (EMEMS*0) is low when AD11 or AD12 or AD13 or AD14 is low and PS is low.

4.5.5 Clock Hold Logic (Figure 4-11, Sheet 1)

The Clock Hold Logic generates a Clock Inhibit Signal to extend T4 during a memory access microcycle. The Clock Hold Logic consists of a DM7474 latch, two DM74H103 flip-flops, and associated gating. At T4, the Read and Write Flags are tested; and, if either flag is set, the DM7474 latch is set. Clock Signal (C4F) is ANDed with the Read Flag (RDFF) or Write Flag (WRFF) to set the DM7474 latch. The outputs of the DM7474 latch provide the Clock Hold (CHOLD) Signal and its complement (CHOLD*), respectively. The CHOLD* Signal is ANDed with INIT* from the System Initialization to inhibit the Clock Signal (CLK) from the Master Oscillator to the System Clock Generator during a memory access microcycle or during initialization. The CHOLD Signal is used by the Memory/Input Timing Logic to generate the Write Timing Signal. Additionally, CHOLD enables a divide-by-3 counter consisting of two DM74H103 flip-flops. The divide-by-3 counter is incremented by the Master Oscillator Clock (CLK). The detected count of three resets the DM7474 latch at T4+3, thereby resetting CHOLD and setting CHOLD*, and allows the CLK Signal from the Master Oscillator to be gated to the System Clock Generator. Figure 4-6 shows the relation of the Clock Hold, Master Oscillator, and System Clock Generator Signals during a memory access microcycle.

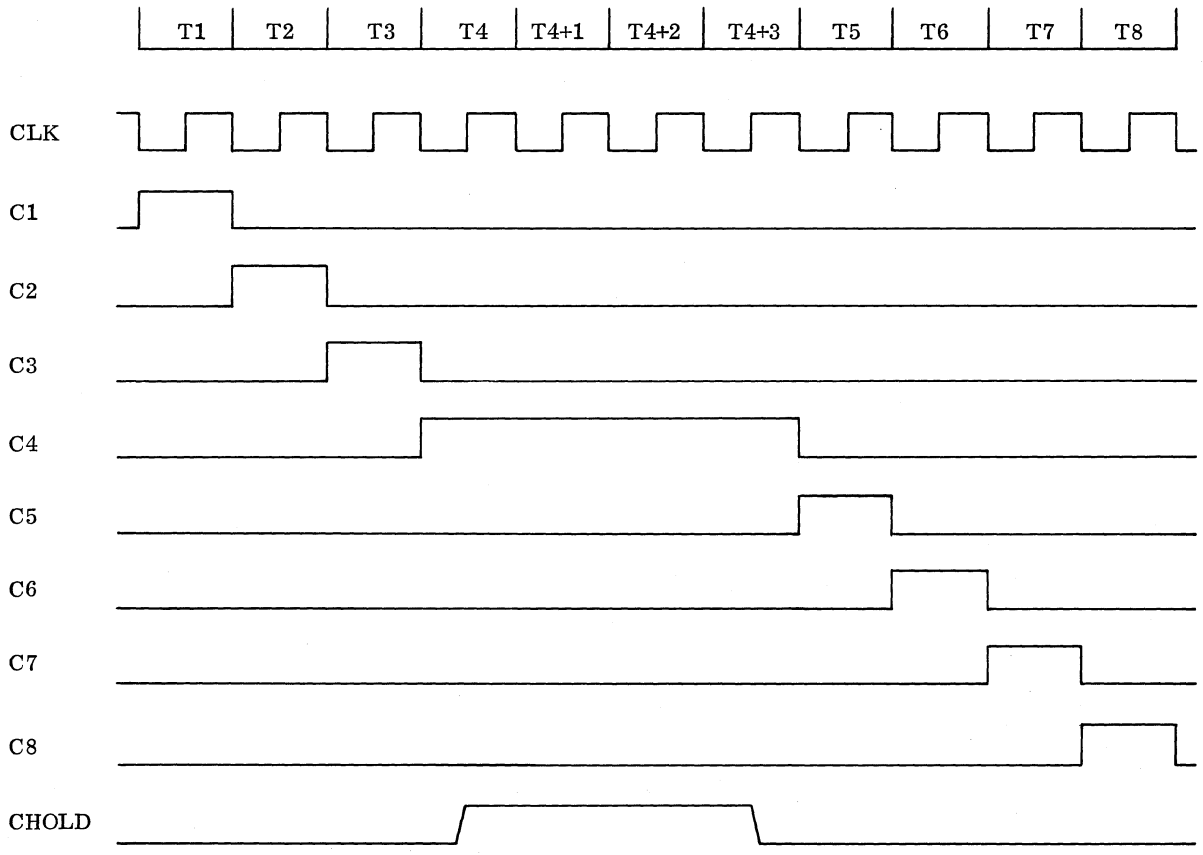


Figure 4-6. Clock Hold Timing

4.5.6 Memory/Input Timing Logic (Figure 4-11, Sheet 1)

The Memory/Input Logic provides the data input strobe to the Data Multiplexer and Control Signals, via the card-edge connector pins, to external input/output devices. The Data Input Strobe Signal (DISTR*) controls the transfer of the Memory Data Input (MDI0 through MD17) to the Data Input Lines (DI0 through DI7). The data on MDI0 through MDI7 is transferred to DI0 through DI7 by the Data Multiplexer when DISTR* goes low. The DISTR* Signal is produced by a DM7474 latch which is set by C7 and reset by C8 and clock (C8B). Data can be read in at T7 to the TALUs via the Data Input Lines (DI0 through DI7).

For a write operation, the data are placed on Data Out Lines DO0 through DO7 and are written into memory at T4 when the signal WRSTR goes high. This signal is created by ANDing CHOLD (Clock Hold) and WRFF (Write Flag). Therefore, data can be written into memory at any time during the expanded T4 period. Writing into memory requires 2 microcycles, as the address register can only be loaded during a read operation. Therefore, the memory location must be read first.

The External Memory Select Signal EMEMS* is high when either the Peripheral Select Signal PS is high or when Address Lines AD11 through AD14 are all high. In the basic system, PS is tied to Address Line AD15. EMEMS* normally is tied to IMS1*, to enable the gating of External Memory Data Lines EMD0 through EMD7 through the external Data Buffer onto the Memory Data In Lines.

External memory reading and writing is controlled by signals RDSTR and BWRSTR (Read and Write Strobe). RDSTR is generated by a DM7474 flip-flop set by Read Flag RDFF and cleared by timing signal C8B. BWRSTR is generated through an AND gate by Write Flag WRFF and Clock Hold Signal CHOLD. BWRSTR is identical to the internal Write Strobe WRSTR.

The Refresh Signal for dynamic memory, RFRSH, is generated from an external Refresh Request Signal, REFREQ. RFRSH follows REFREQ during time C4 except that RFRSH remains low if either the read or write flip-flops are set, producing signal RW.

The Cycle Request Signal, CR, is developed by a DM7474 flip-flop which is set when signals C4F, RW, and CLK are all high. It is reset when signal CHOLD (Clock Hold) goes low.

All signals necessary for operation of external memory are available on card-edge connector pins. Figure 4-7 shows the relation of the Memory/Input Timing Logic Signals during a read and write operation.

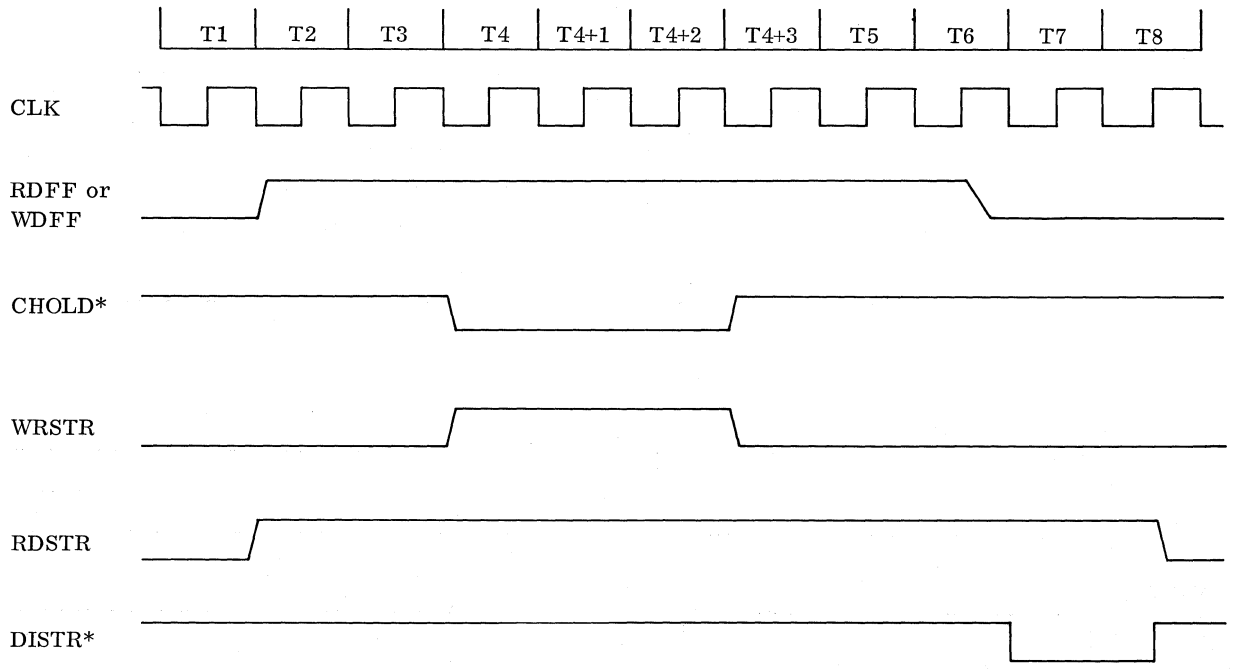


Figure 4-7. Memory/Input Timing

4.6 PROGRAM COUNTER AND PROGRAM COUNTER CONTROL

The Program Counter is an 8-bit presettable counter used to store and increment an 8-bit slice corresponding to the page address. The Program Counter Control provides output enable and incrementing control signals.

4.6.1 Program Counter (Figure 4-11, Sheet 3)

The Program Counter is an 8-bit presettable binary counter composed of two DM8556 4-bit binary counters that provide the page address of the instruction to be executed. During initialization, the Program Counter is cleared by INIT from the System Initialization Logic. After initialization, the Program Counter may be incremented by 1 or a new count may be loaded from the Data Out Bus as determined by the microprogram. The Program Counter is incremented by the Count Signal (CNT) from the Program Counter Control when the Program Count Enable Flag (PCENBL*) is set. The Program Counter is loaded from the Data Out Bus at T4 (C4F) when the Load Program Counter Flag (LPC) is set.

The contents of the Program Counter can be transferred to the Data Out Bus (CO0 through DO7) by the Program Count Enable Signal (PCOE*). The contents of the Program Counter provide an input to the Data Multiplexer for transfer to the Data Input Bus (CI0 through DI7) when the Read Program Counter Flag (RPC) is set.

4.6.2 Program Counter Control (Figure 4-11, Sheet 2)

The Program Counter Control provides the Page Counter Enable Signal (PCOE) and Increment Clock (CNT) to the Program Counter. CNT is high when C5B is high and the high-order carry shift is high. As both C4F and C5B cannot be high simultaneously and the Program Counter is loaded by C4F clock, the counter cannot be both loaded and incremented at the same time. The Program Counter increments by 1 when Program Count Enable Flag PCENBL is set and signal CNT is strobed. This occurs with a page boundary crossover condition, when the high-order carry bit HOCSH becomes high.

The Program Counter output appears on the Data Out Bus (CO0 through DO7) when Program Counter Enable Signal PCOE* is low. This signal is produced at T3 and T4 when CROM Enable Signal ENCTL is present.

4.7 SYSTEM INITIALIZATION (Figure 4-11, Sheet1)

Initialization is done in order to ensure the proper starting of the CPU logic and to ensure that the IMP-8C starts in a cleared condition.

When power is first applied, transistors Q1 and Q2 switch the supply voltage (SVGG) from +5v to -12v and the System Clear Signal (SYSCLR*) from +5v to 0v, ensuring the proper startup for the CPU logic chips and flag logic. The Initialization Signal (INIT) resets the Program Counter and the Clock Shift Register. The flags and the Stack-Full Flip-flop are cleared by FLGCLR* which is derived by a hardwire condition to SYSCLR*.

The system can be reinitialized by a low External Clear Signal (SYSCLR*). This causes SVGG to be switched from -12v to +5v, the output INIT* to go low, and FLGCLR* to go low. This takes approximately 5 msec and allows the CPU to stop before the clocks stop running in order to ensure that the gates of the RALU and CROM are discharged so that the devices are fully initialized. When SYSCLR* is again allowed to go high, initialization proceeds as when power was first applied.

4.8 INTERRUPT CONTROL (Figure 4-11, Sheet 2)

The interrupt handling portion of the IMP-8C/200 is enabled when the Interrupt Enable Flag INEN is set. INEN is one input to a two-input AND gate. The other input is either the Interrupt Signal INTR or the signal on point E1. In the basic system, E1 is jumped to the Stack-Full Flag to enable a stack-full condition to cause a program interrupt.

The INT Signal from the AND gate in the conditional jump multiplexer is applied to the CROM. When the CROM receives the interrupt condition, it causes an immediate program jump to location 0000 in memory, where an interrupt handling subroutine should start.

4.9 DATA INPUT LOGIC

The Data Input Logic consists of the Input Multiplexer and the Data Multiplexer. The Input Multiplexer selects the External Memory Bus (EMD0 through EMD7) or Peripheral Device Bus (PD0 through PD7) for input to the Memory Data Input Bus (MDI0 through MDI7). The Data Multiplexer selects the Program Counter or the Memory Data Input Bus to be transferred to the Data Input Bus (DI0 through DI7). Figure 4-8 shows the input timing relations for the Memory Data Input Bus and the Data Input Bus.

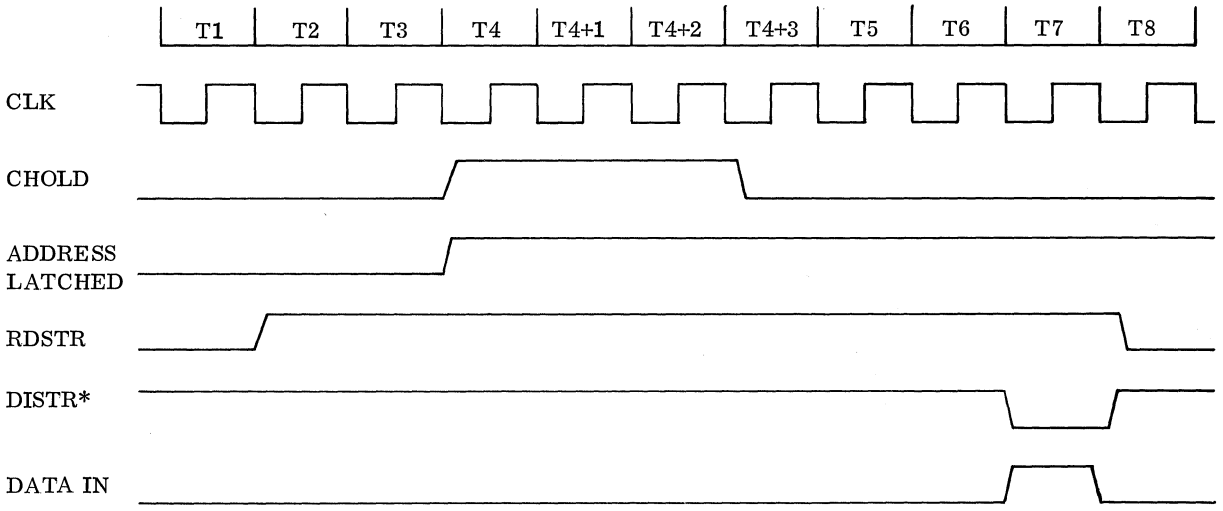


Figure 4-8. Read Data Timing

4.9.1 Input Multiplexer (Figure 4-11, Sheet 4)

The Input Multiplexer consists of two DM8123 4-bit multiplexers. When the Peripheral Select Signal (PS) is low, the External Memory Device Bus is selected as an input to the multiplexer. When the Peripheral Select Signal (PS) is high, the Peripheral Device Bus is selected as an input to the multiplexer. The selected input is transferred to the Memory Data Input Bus when the Internal Memory Select Signal (IMS1*) goes low or Peripheral Select (PS) goes high.

4.9.2 Data Multiplexer (Figure 4-11, Sheet 3)

The Data Multiplexer consists of two DM8123 4-bit multiplexers. The data from the Memory Data Input Bus (MDI0 through MDI7) or the Program Counter is transferred by the Data Multiplexer to the Data Input Bus (DI0 through DI7) at T7 by the Data Input Strobe Signal (DISTR*). If the Read Program Counter Flag (RPC) is set, the contents of the Program Counter are transferred to the Data In Bus. If the Read Program Counter Flag (RPC) is not set, the contents of the Memory Data Input Bus are transferred to the Data Input Bus at T7.

4.10 DATA BUFFER (Figure 4-11, Sheet 2)

The Data Buffer consists of two DM8097 hex buffers. The Data Buffer provides buffering of the data from the RALU and Data Input Bus (DI0 through DI7) to the Data Output Bus (DO0 through DO7). The Data Buffer output is inhibited during T3 and T4 when data is being transferred from the Program Counter to the Address Latches via the Data Output Bus. PCOE from the Program Counter Control is inverted and the complemented PCOE inhibits the transfer of data via the Data Buffer.

Figure 4-9 shows the relation of the data and timing signals on the Data Output Bus.

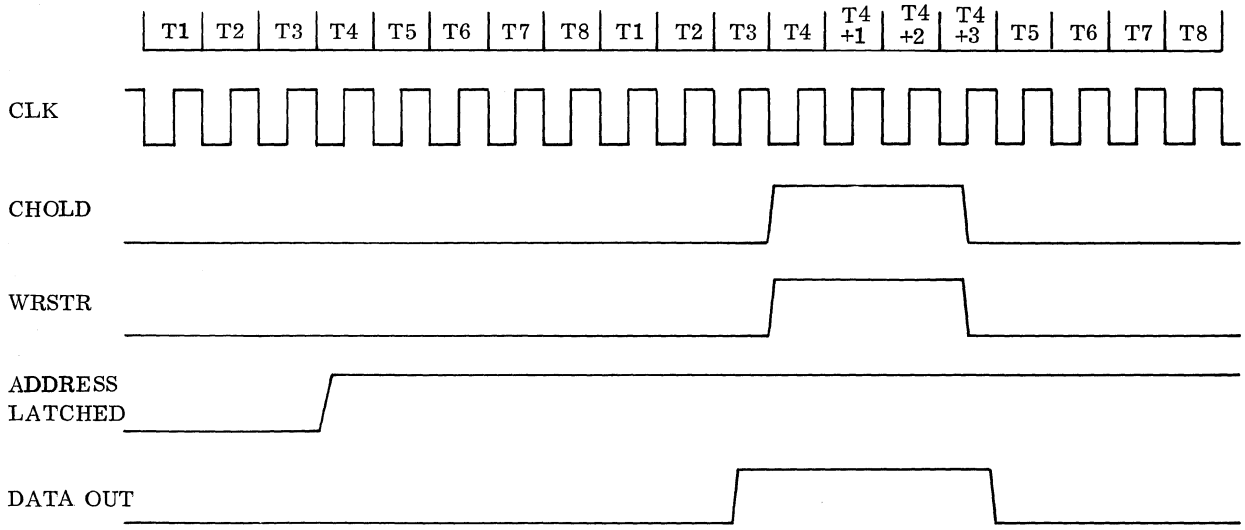


Figure 4-9. Write Data Timing

4.11 USER-AVAILABLE OPTIONS

This section covers the features available to the user which enable the use of the IMP-8C/200 with nearly any type of external device. For detailed descriptions of the operation of these features, refer to the appropriate section of this chapter.

4.11.1 Memory Options

The IMP-8C/200 has the capability of directly addressing 65,536 bytes of data in almost any type of memory. This capability is enabled by the 16-bit Address Register, whose outputs, Address Lines AD0 through AD15, are available on card-edge pins.

A great deal of added flexibility is given by use of the Address Decoder and its associated circuitry. The 16 page Enable Lines are activated by the 4 Address Lines AD8 through AD11, and so are active with addresses between 0 and 4096_{10}, producing sixteen 256-word pages.

The On-Card ROM and RAM chips are enabled by lines ROMS*0 through ROMS*7 and RAMS* on the card-edge connector pins and may be tied to the Enable Lines as desired. Generally, it is desirable for the read/write memory to reside in page 0, as it can be directly accessed from any point in memory. Likewise, it is also desirable to enable one page of nonvolatile storage of read-only memory in upper memory to hold the initialization routine. These are only suggested means of implementation, however. The user has complete freedom of memory allocation.

For slower memories, the Clock Hold Circuit can be externally controlled by removing the jumper which ties points E8 and E9 and installing a jumper between points E7 and E9. Point E7 is supplied by External Clear Signal ECL*, brought in on card Pin 129. This signal should be forced low when external memory read/write operations are finished.

Special circuits have been provided for National Semiconductor dynamic memories or any semiconductor memory requiring Refresh Signals and/or Cycle Signals. For details of these circuits, refer to chapter 5, Input/Output.

4.11.2 Timing Options

External Clock Signals for synchronization may be used in place of the Master Oscillator clock. In order to accomplish this, the jumper between E4 and E5 should be removed and replaced with a jumper between E6 and E5. The External Clock Signal is fed in on edge Pin 135. For details, refer to chapter 5, Input/Output.

The Internal Clock Signal from the IMP-8C is available on edge Pin 134. Clock Signals C2, C4, C6, C8 and C81234 are available also on Pins 127, 104, 131, 136, and 130, respectively.

4.11.3 CPU Options

The Stack-Full Signal from the RALU may be tied to the Interrupt Flip-flop by placing a jumper between points E3 and E1. Provision must be made to clear the Stack-Full Flip-flop immediately. If this is not done, interrupts constantly generated, effectively halting CPU operation. See section 4.8 for detailed description.

The flag flip-flops have four flags available for use as the user desires. These flags are output to Pins 10, 11, 39, and 19. One of the user flags may be used to clear the Stack-Full Flip-flop as described in the previous paragraph by tying it to the Stack-Full Flip-flop clear line on Pin 112. All of the user flags may be set or pulsed under program control. The Halt Flag is also brought to an edge pin for external use.

There is one jump condition available to the user for use with a Branch-On-Condition instruction. It is tied to Pin 13, and should be brought high when true. The Start Condition is also tied to a card edge pin for external implementation. It is available on Pin 15.

An extended instruction set can be enabled by implementing another CROM chip. The additional CROM is wired in parallel with the existing CROM, and control is passed back and forth with the user of signal ENCTL.

Initialization of the IMP-8C can be accomplished by driving signal SYSCLR* (available on pin 128) low, with either a switch or logic. External equipment may be initialized by employing signal INIT*, available on Pin 126. The CROM may be installed in either CROM socket.

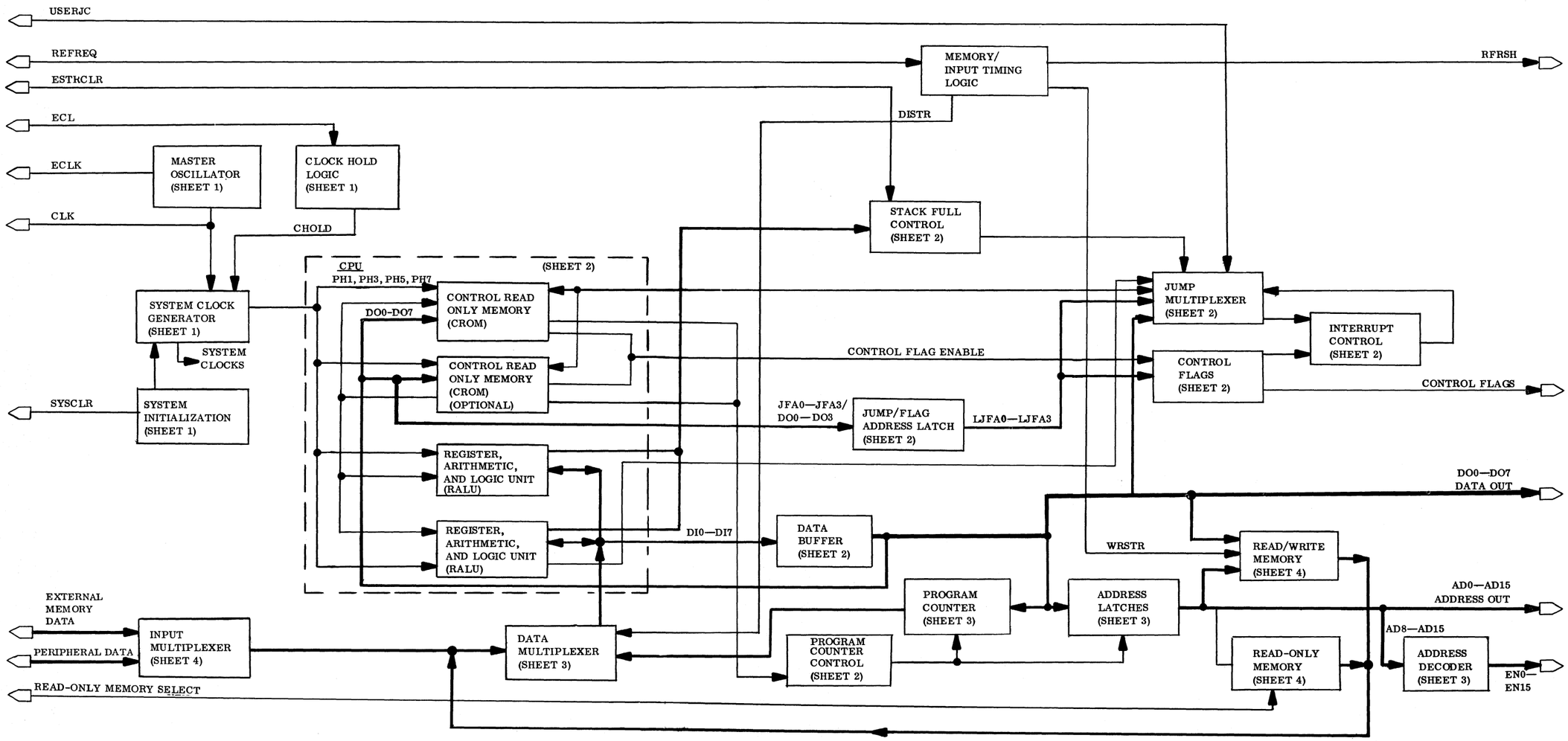


Figure 4-10. IMP-8C Functional Block Diagram

NOTE: INFORMATION ON THIS DIAGRAM IS SUBJECT TO CHANGE WITHOUT NOTICE.
ENGINEERING DIAGRAMS SUPPLIED WITH IMP-8C SHOULD BE REFERENCED.

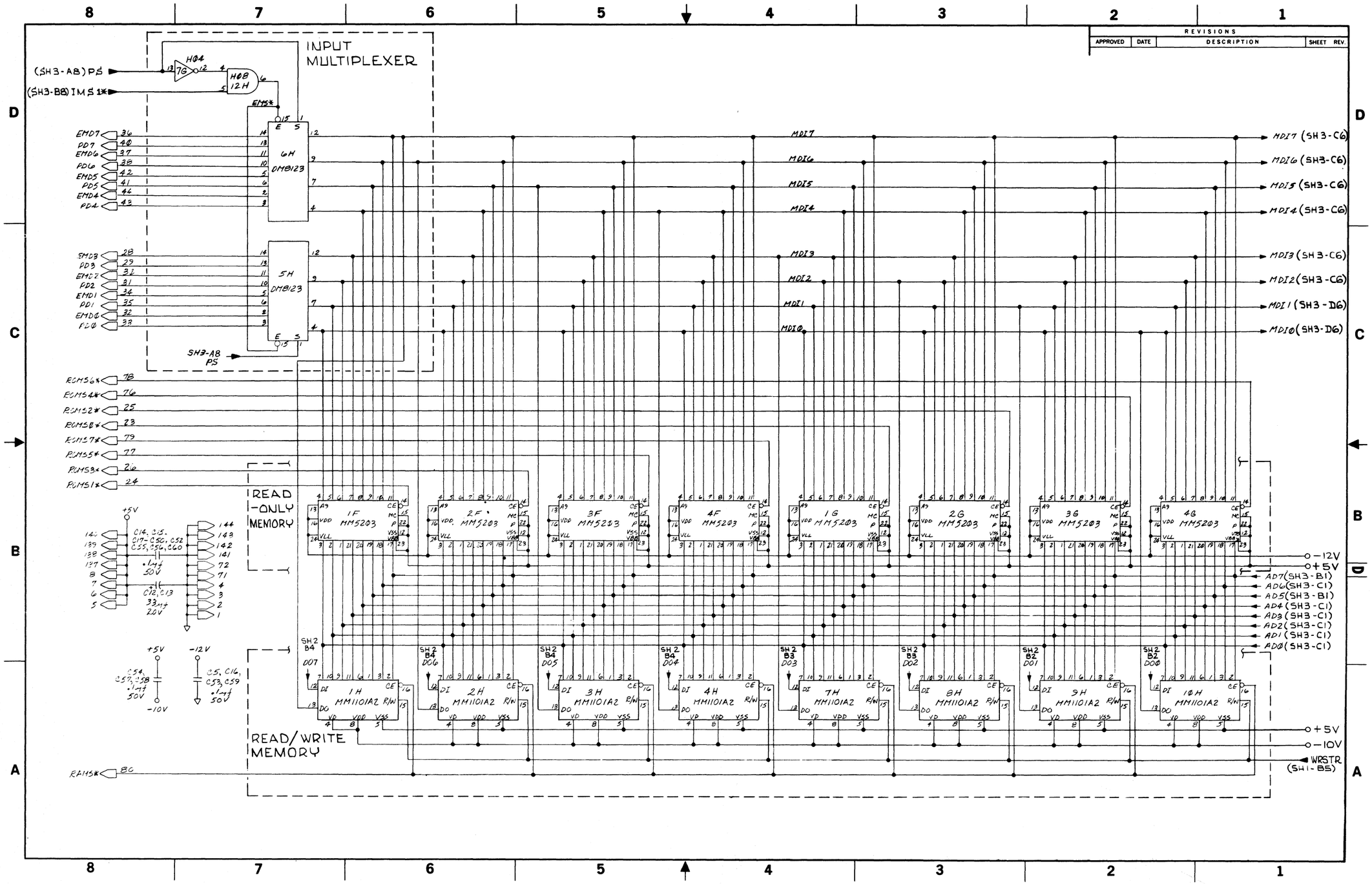
Figure 4-11. IMP-8C Microprocessor (Sheet 1 of 4)

NOTE: INFORMATION ON THIS DIAGRAM IS SUBJECT TO CHANGE WITHOUT NOTICE.
ENGINEERING DIAGRAMS SUPPLIED WITH IMP-8C SHOULD BE REFERENCED.

Figure 4-11. IMP-8C Microprocessor (Sheet 2 of 4)

NOTE: INFORMATION ON THIS DIAGRAM IS SUBJECT TO CHANGE WITHOUT NOTICE.
ENGINEERING DIAGRAMS SUPPLIED WITH IMP-8C SHOULD BE REFERENCED.

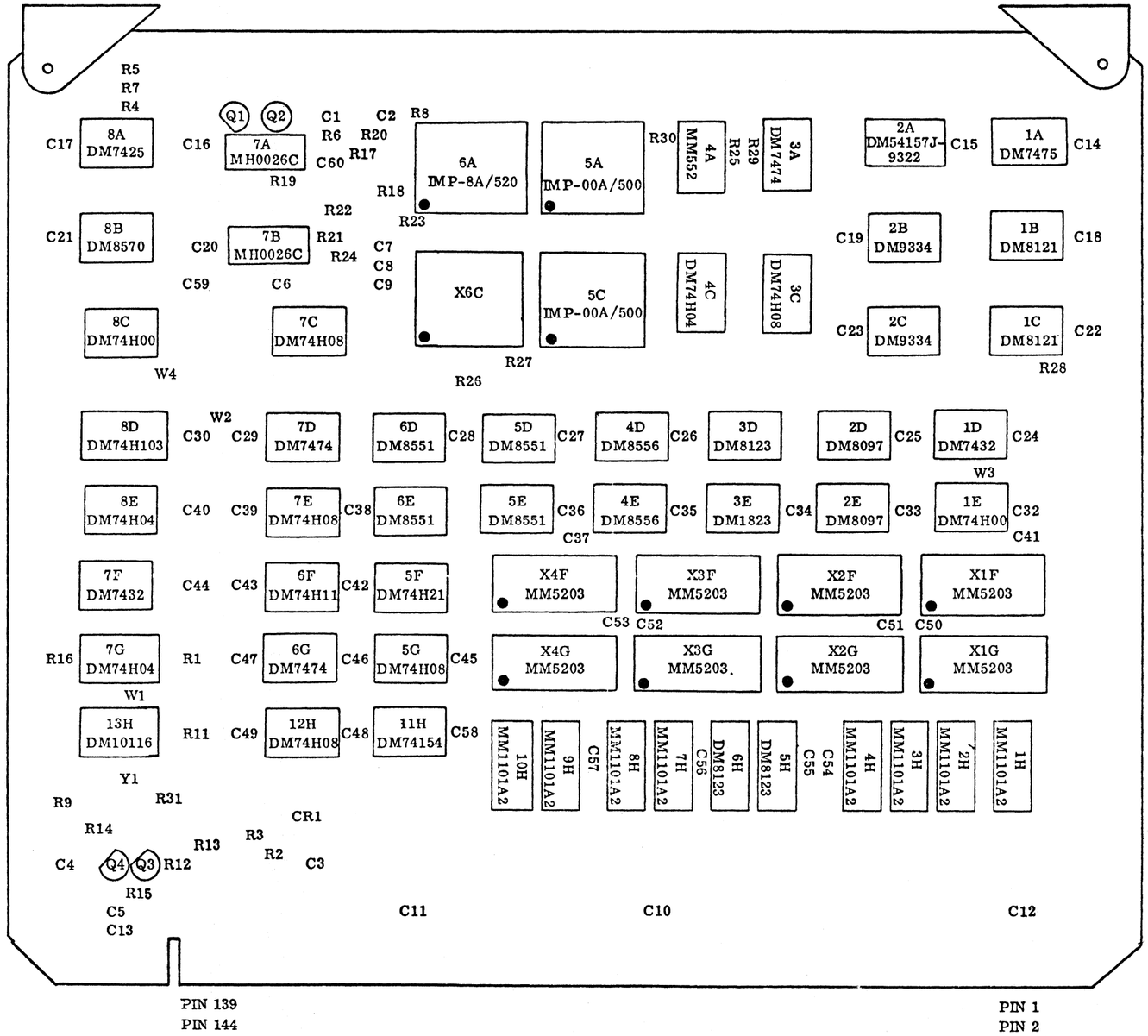
Figure 4-11. IMP-8C Microprocessor (Sheet 3 of 4)



REVISIONS			
APPROVED	DATE	DESCRIPTION	SHEET REV.

NOTE: **INFORMATION ON THIS DIAGRAM IS SUBJECT TO CHANGE WITHOUT NOTICE. ENGINEERING DIAGRAMS SUPPLIED WITH IMP-8C SHOULD BE REFERENCED.**

Figure 4-11 IMP-8C Microprocessor (Sheet 4 of 4)



NOTE: (1) For devices that are removable, a large dot indicates the location of pin 1.
(2) Odd-numbered pins are located on component side.
(3) Even-numbered pins are located on solder side.
(4) Prefix X denotes socket. Device designated is an option.

NOTE: INFORMATION ON THIS DIAGRAM IS SUBJECT TO CHANGE WITHOUT NOTICE. ENGINEERING DIAGRAMS SUPPLIED WITH IMP-8C SHOULD BE REFERENCED.

Figure 4-12. IMP-8C Component Layout

4-3. IMP-8C/200 List of Materials

Description	Reference Designation	Part Number	Quantity
INTEGRATED CIRCUITS			
TRI-STATE® Quad D-type Flip-flop	5D, 5E, 6D, 6E	DM8551	4
Quad 2 - input Multiplexer	3D, 3E, 6H, 6H	DM8123	4
TRI-STATE® 8-channel Digital Multiplexer	1B, 1C	DM8121	2
8-bit Serial-in, Parallel-out, Shift Register	8B	DM8570	1
Dual D-type Flip-flop	3A, 6G, 7D	DM7474	3
Dual 4-input AND Gate	5F	DM74H21	1
TRI-STATE® Hex Buffer	2D, 2E	DM8097	2
Hex Inverter	4C, 7G, 8E	DM74H04	3
Quad 2-input NAND Gate	1E, 8C	DM74H00	2
Quad 2-input AND Gate	3C, 5G, 7C, 7E, 12H	DM74H08	5
Triple 3-input AND Gate	6F	DM74H11	1
8-bit Addressable Latch	2B, 2C	DM9334	2
TRI-STATE® Programmable Binary Counter	4D, 4E	DM8556	2
4-line to 16-line Decoder/Demultiplexer	11H	DM74154	1
Quad Latch	1A	DM7475	1
Dual J-K Edge-triggered Flip-flop	8D	DM74H103	1
Quad 2-input OR gate	1D, 7F	DM7432	2
Triple Differential Line Receiver	13H	DM10116	1
Dual MOS Clock Driver	7A, 7B	MH0026C	2
Dual 4-input NOR Gate	8A	DM7425	1
Four MOS transistor package	4A	MM552	1
Quad 2-input Multiplexer	2A	DM54157J 9322	1
256-bit Static Random Access Memory	1H, 2H, 3H, 4H, 7H, 8H, 9H, 10H	MM1101A2	8
MOS/LSI Register, Arithmetic and Logic Unit (RALU)	6A	IMP-8A/520	1
MOS/LSI Control Read-only Memory (CROM)	5A, 5C	IMP-00A/500	2
2048-bit Electrically Programmable Read-only Memory (PROM)-option	1F-4F, 1G-4G	MM5203	8

Item	Description	Reference Designation	Part Number	Quantity
	TRANSISTORS			
27	Transistor	Q1	2N3638	1
28	Transistor	Q2	2N2222A	1
29	Transistor	Q3, Q4	2N4258A	2
	CAPACITORS			
30	Capacitor, 0.01 µf, 50V	C1		1
31	Capacitor, 0.1 µf, 50V	C2,C5,C14-C30,C32-C60		48
32	Capacitor, 33 µf, 20V	C10, C11, C12, C13		4
33	Capacitor, 150 µf, 20V	C3		1
34	Capacitor, 27 pf, 300V	C4		1
35	Capacitor, 750 pf, 300V	C6, C7, C8, C9		4
	DIODE			
36	Diode	CR1	1N645	1
	RESISTORS			
37	Resistor, 39 ohms, 5%, 1/4 W	R14		1
38	Resistor, 15 ohms 5%, 1/4 W	R18, R19, R22, R23		4
39	Resistor, 51 ohms, 5%, 1/4 W	R16		1
40	Resistor, 100 ohms, 5%, 1/4 W	R12, R13		2
41	Resistor, 220 ohms, 5%, 1/4 W	R4, R11, R15		3
42	Resistor, 330 ohms, 5%, 1/4 W	R2, R9, R10		3
43	Resistor, 220 ohms, 5%, 1/4 W	R1		1
44	Resistor, 1K, 5%, 1/4 W	R5, R6, R7		3
45	Resistor, 2.2K, 5%, 1/4 W	R29, R30, R31		3
46	Resistor, 5.1K, 5%, 1/4W	R22		1
47	Resistor, 5.6K, 5%, 1/4W	R25, R26, R27		3
48	Resistor, 10K, 5%, 1/4W	R3, R17, R20, R21, R24		5
49	Resistor, 2K, 5%, 1/4W	R28		1
	CRYSTAL			
50	Crystal, 5.7143 MHz	Y1		1

Chapter 5

INPUT/OUTPUT

5.1 GENERAL INFORMATION

This chapter discusses the IMP-8C input/output considerations, interface methods, and programs. Block diagrams are provided to assist in designing controllers suited to the user's individual requirements. Program flowcharts describe typical input and output routines.

A number of user control flags are available to control various input/output operations. In the following discussion, emphasis is placed on program-controlled operations to illustrate the simplicity of the interface requirements. The Teletype interface using control flags (described in this chapter) is an example of a relatively simple peripheral interface that is flexible enough for all operations but requires a large degree of program control. This is acceptable for most applications where the processor is not doing other functions during Teletype operations.

The IMP-8C addresses peripheral devices and memory in an identical manner. Each peripheral device controller must have a unique address assigned to the controller that is not duplicated by another device controller or memory location.

5.2 INPUT/OUTPUT BUS STRUCTURE

Figure 5-1 is a simplified block diagram of the I/O bus structure of the IMP-8C microprocessor. Control, timing, and status information is exchanged between the IMP-8C and the peripheral devices or add-on memory via individual lines. Table 5-1 is a listing of the control, timing, and status signals available at card-edge connector.

Data signals from peripheral devices and add-on memory are received by the IMP-8C Input Multiplexer via two 8-bit data busses; the Peripheral Data Bus (PD0 through PD7) and the External Memory Data Bus (EMD0 through EMD7), respectively. Data signals to peripheral devices and add-on memory are sent by the IMP-8C Data Buffer via the 8-bit Data Out Bus (DO0 through DO7). A 16-bit address contained in the IMP-8C Address Latches is available to the peripheral device controllers and add-on memory via the Address Out Bus (AD0 through AD15).

5.3 TIMING CONSIDERATIONS

The timing for Read and Write Data and Control Signals as provided by the hardware during a read and a write microcycle are shown in figure 5-2. A write operation is always preceeded by a read operation in order to set the device address on the Address Lines.

During a read microcycle, the Read Flag (RDFF) is set by the microprogram at T2. The Read Flag sets the Read Strobe (RDSTR) to the external device. The buffered T4 Clock (C4F) sets the Clock Hold Signal (CHOLD), thereby extending T4 for an additional 3 clock times. The Read Flag (RDFF) is reset at T6; however, the Read Strobe (RDSTR) remains set until T8. At T7, the Data Input Strobe (DISTR*) is set to allow the data on the Data Input Bus selected to be entered to the RALU.

During a write operation, the Address Latches are set by a previous read operation. The Write Flag is set by the microprogram at T2. The master clocks are inhibited at T4 by the Clock Hold Signal (CHOLD) to extend T4 for an additional 3 cycles.

During T4 the Write Strobe (WRSTR) is set, and at T4 +3 the strobe is reset. During T4 through T4 +3, the data from the RALUs is sent out via the Data Out Bus. At T6, the Write Flag (WRFF) is reset by the microprogram.

Table 5-1. External Control, Timing, and Status Signals

Signal	Function	Connector Pin
BWRSTR	Buffered Write Strobe	116
C2	System Clock Corresponding to T2	127
C4	System Clock Corresponding to T4	104
C6	System Clock Corresponding to T6	131
C8	System Clock Corresponding to T8	136
C81234	System Clock Corresponding to T8, T1, T2, T3, and T4	130
C8B*	Buffered Master Clock during last half of T8	132
ECL*	External Clock Hold Clear Signal	129
ECLK	External Clock Signal	135
EN0* - EN15*	Decoded Address Enable Signals	⚠1
EMEMS*	External Memory Select Signal	133
ESTKCL*	External Stack Clear Signal	112
HALT	Halt Flag	9
IMS1, 2	Internal Memory Select Signal	123, 120
INTRQ	External Interrupt Request Signal	17
OCLK	Output Master Clock Signal	134
PCOE	Page Counter Output Enable Signal	75
PS	Peripheral Select	81
RAMS*	Random Access Memory Select Signal	80
RDSTR	Read Strobe Signal	122
REFREQ	Memory Refresh Request Signal	124
RFRSH	Memory Refresh Signal	121
ROMS0* - ROMS7*	Read-Only Memory Select Signal	⚠1
SYSCLR*	External System Clear Signal	128
UNUS2	Unused Microprogrammable Flag	11
USER1-4	User Flags 1-4	⚠1
USERJC	User Jump Condition Signal	13

⚠1 See appendix B for listing of signals, pin numbers, and loading of all card-edge connector signals.

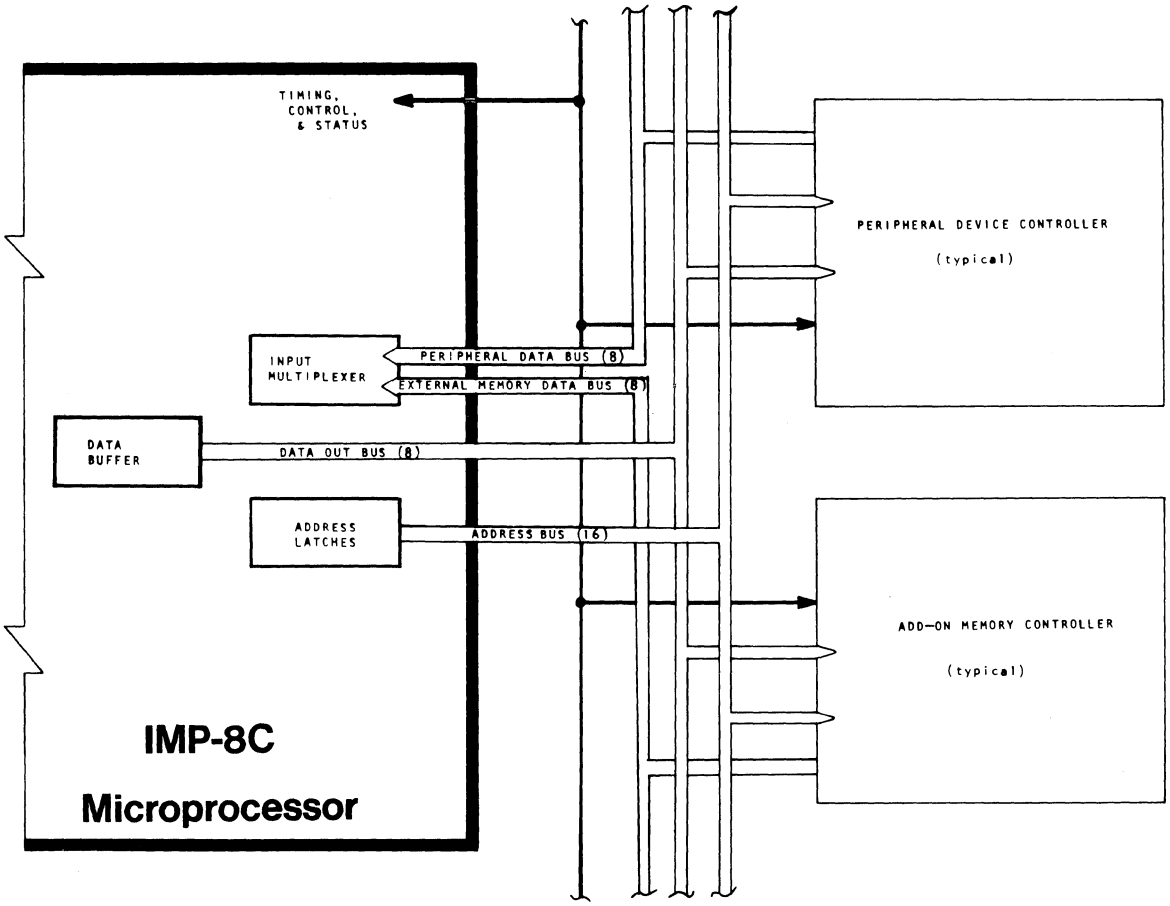


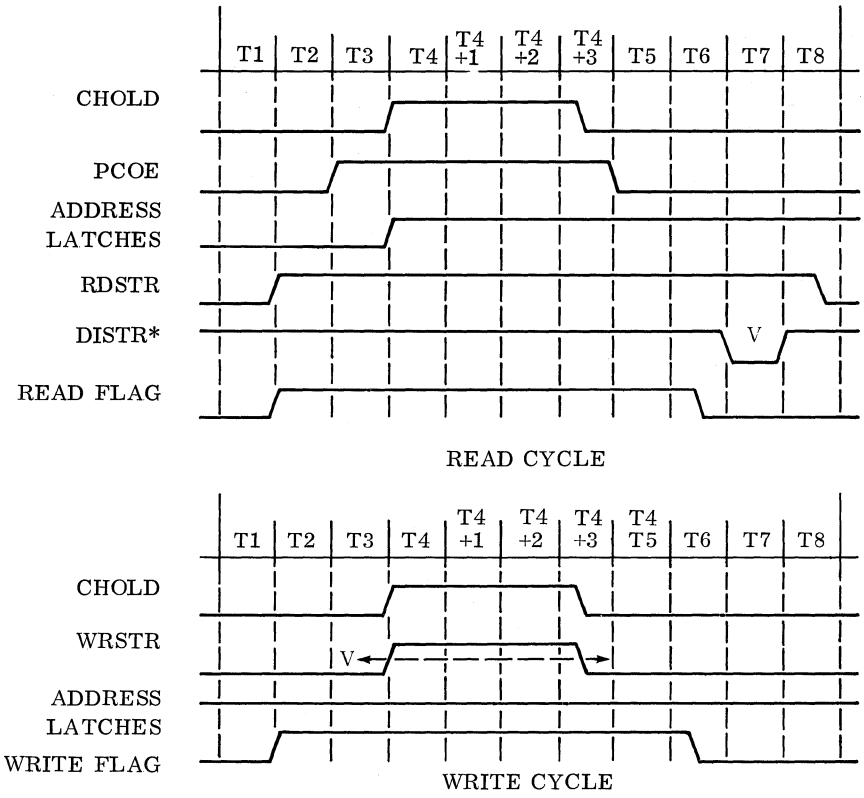
Figure 5-1. IMP-8C Input/Output Bus Structure

5.4 PROGRAMMING CONSIDERATIONS

Data transfers between the 4 RALU Accumulator Registers (Accumulator 0 through 3) and external devices are accomplished by using the same memory reference instructions that transfer data to or from memory. The user assigns addresses to the external devices that are not allocated to memory.

5.4.1 Data Transfer Instructions

The Load (LD) instruction may be used to input data, and the Store (ST) instruction may be used to output data. Other memory reference instructions may be used for more complex input/output operations. Additionally, the user flags may be used to output serial or single bit data by setting or pulsing the flags by Set Flag (SFLG) and Pulse Flag (PFLG) instructions, respectively; and the User Jump Condition may be used to input serial or single bit data by using the Branch-On-Condition (BOC) instruction. Table 5-2 is a listing of memory reference instructions that may be used for input/output operations.



NOTE: V = Data valid; data must not change during this time.

Figure 5-2. IMP-8C Read/Write Timing

Table 5-2. Memory Reference Instructions

Instruction	Use
Load (LD)	INPUT
Store (ST)	OUTPUT
Load Indirect (LD@)	INPUT
Store Indirect (ST@)	OUTPUT
Add (ADD)	INPUT
Skip if Not Equal (SKNE)	INPUT
Logical And (AND)	INPUT
Logical Or (OR)	INPUT
Skip if Logical And is Zero (SKAZ)	INPUT
Subtract (SUB)	INPUT
Increment and Skip if Zero (ISZ)	OUTPUT
Decrement and Skip if Zero (DSZ)	OUTPUT

5.4.2 IMP-8C Instruction Execution Time

The actual instruction execution time depends upon the specific system implementation. The execution of an instruction is determined by the number of microcycles required to execute the instruction, the number of read cycles, and the number of write cycles. Typical values for a microcycle are 1.4 microseconds plus an additional 0.56 microsecond for a read or write cycle. Total execution time (T) for an instruction can, therefore, be defined by the number of microprogram cycles (N), the number of write cycles (W), and the number of read cycles (R) required to accomplish the instruction; where: T (in microseconds) = 1.4N + 0.56W + 0.56R or T = 1.4 (N + 0.4R + 0.4W). Table 5-3 is a listing of the IMP-8C instructions, the total number of microprogram cycles, the number of read cycles, and the number of write cycles. A range of values for N is shown since indexing, skips, base page jumps, and an unsuccessful branch-on condition require additional microprogram execution cycles.

Table 5-3. IMP-8C Instruction Execution Time

Instruction	Total Microprogram Cycles (N)	Read Cycles (R)	Write Cycles (W)
COMP2, PFLG, PULL, PUSH, RADD, RAND, ROL, ROR, SFLG, SHL, SHR	3	1	-
PULLF, PUSHF, RTS	4	1	-
RTI, RXCH, XCHRS	5	1	-
LI	5	2	-
AISZ	6 (+3 if skip)	2	-
BOC	7 (6 if branch)	2	-
JMP	10 (+1 if base page)	2	-
JSR	14 (+1 if base page)	2	-
JMP@	12	4	-
JSR@	18	4	-
LD, ADD, AND, OR, SUB	8 (+2 if indexed)	3	-
ST	10 (+2 if indexed)	3	1
SKNE, SKAZ	9 (+2 if indexed) (+3 if skip)	3	-
LD@	12	5	-
ST@	13	5	1
ISZ, DSZ	11 (+2 if indexed) (+3 if skip)	3	1
HALT	Indefinite	1	-

5.5 SERIAL TELETYPE INTERFACE USING FLAGS

A simple program-controlled Teletype interface can be implemented using a hex inverter integrated circuit and a few discrete components. This Teletype interface provides communication between the IMP-8C and a serial Teletype using the User Jump Condition and two user control flags. This approach eliminates the need for device address decoding and the associated hardware. Figure 5-3 is a schemetic diagram of a flag-controlled serial Teletype interface for use with the IMP-8C. The IMP-8C User Jump Condition (USERJC) is used to input the data, and a user control flag (USER1 in figure 5-3) is used to output the data. Additionally, another user control flag (USER2 in figure 5-3) is used as a Read/Enable Control Signal.

5.5.1 Serial Teletype Flag-controlled Transmit Character Routine

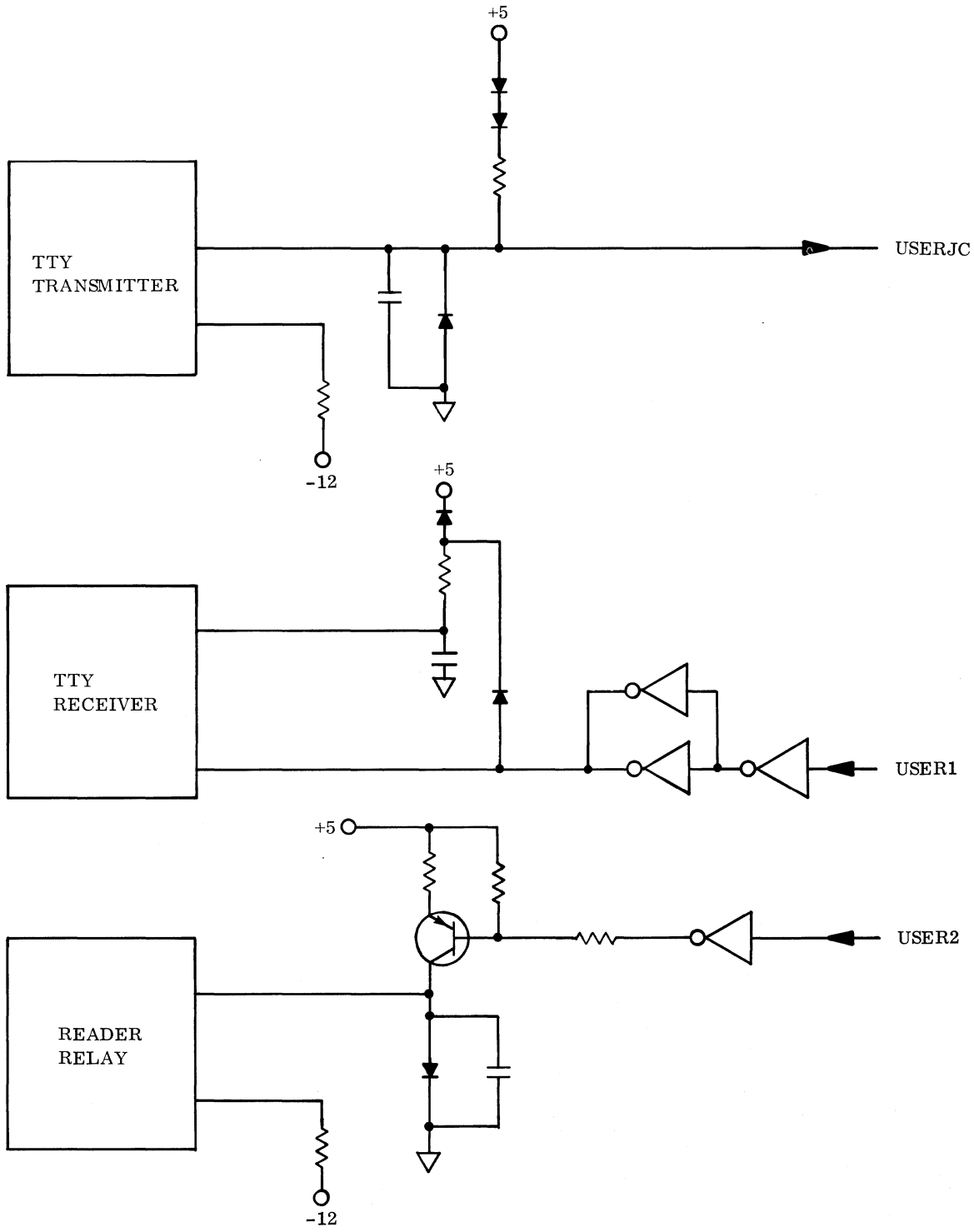
Figure 5-4 shows a program flowchart for a serial Teletype, single-character, transmit routine. The routine sends one character (right justified in AC0) to the Teletype. Since this program is written as a subroutine, the contents of the accumulators are saved in temporary storage before transmitting the character in the routine. The stack provides convenient temporary storage for the contents of the accumulators. In order to provide the required delay between the transmission of the individual bits, a delay subroutine is used in the transmit routine.

5.5.2 Serial Teletype Receive Character Routine

Figure 5-5 shows a program flowchart for a serial Teletype, single-character, receive routine using the User Jump Condition. The routine takes one character from the Teletype and loads it into Accumulator 0 (AC0) right justified. Since this program is written as a subroutine, the contents of the accumulator prior to entering the character transfer portion of the subroutine are stored in the stack. The delay between the reception of individual bits in character stream is the same as the delay during transmit and, therefore, may use the same subroutine.

5.6 SERIAL TELETYPE INTERFACE USING DEVICE ADDRESSES

A program-controlled Teletype interface using a peripheral device address requires more hardware than a Teletype interface using flags, but has the advantage of allowing use of the IMP-8C Teletype routines directly, without tieing up any of the user flags. This teletype interface provides full-duplex, bit-serial communication between the IMP-8C and the Teletype. The formatting and timing of the bit stream must be controlled by the IMP-8C Program. All data communications between the IMP-8C and the interface is over the data busses. Figure 5-6 is a schematic diagram of serial Teletype interface with device address decoding for use with the IMP-8C.



Note: Component values are determined by the user's individual interface requirements or component selection.

Figure 5-3. Serial Teletype Interface Using Flags

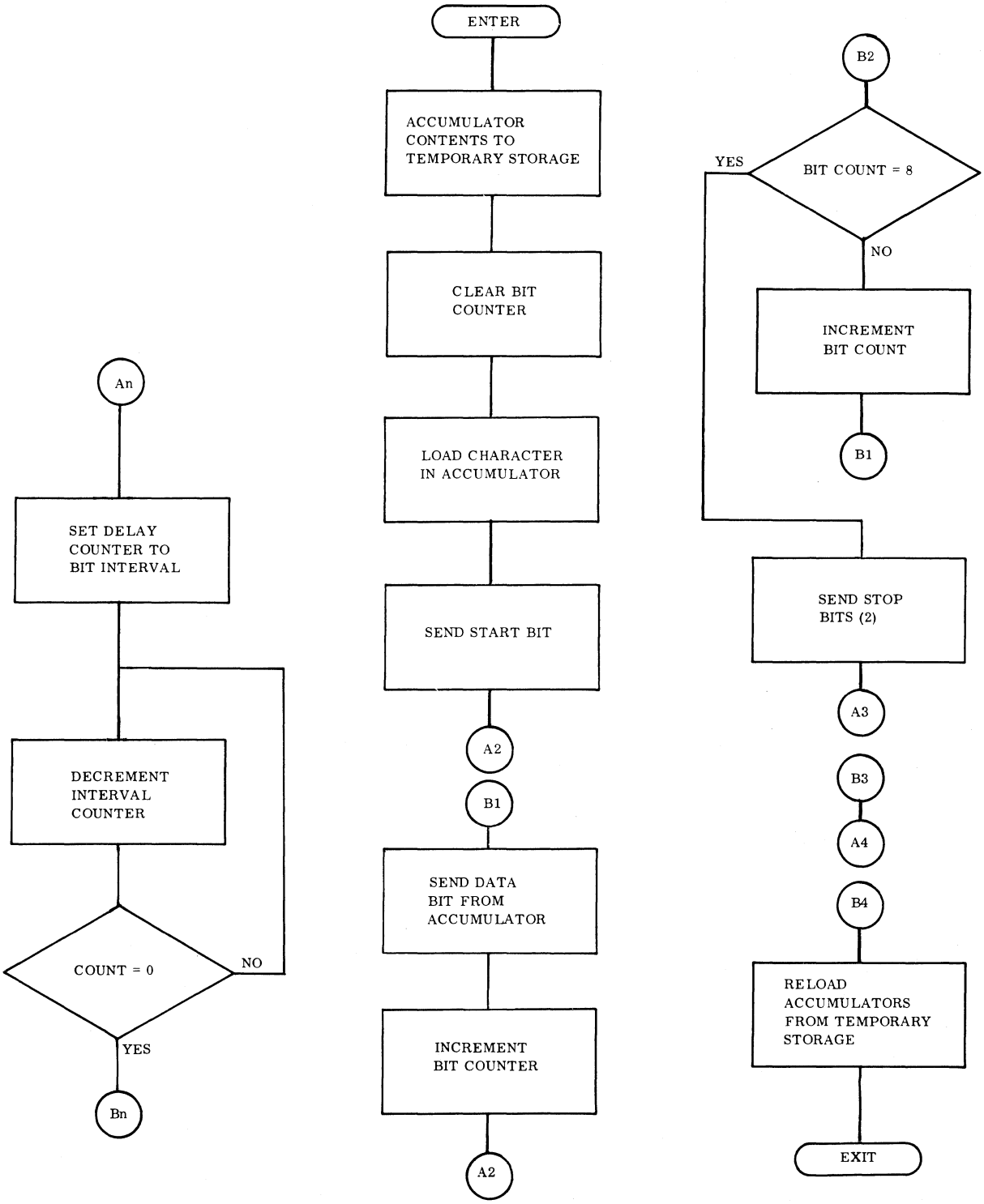


Figure 5-4. Teletype Transmit Character Routine Using Flags

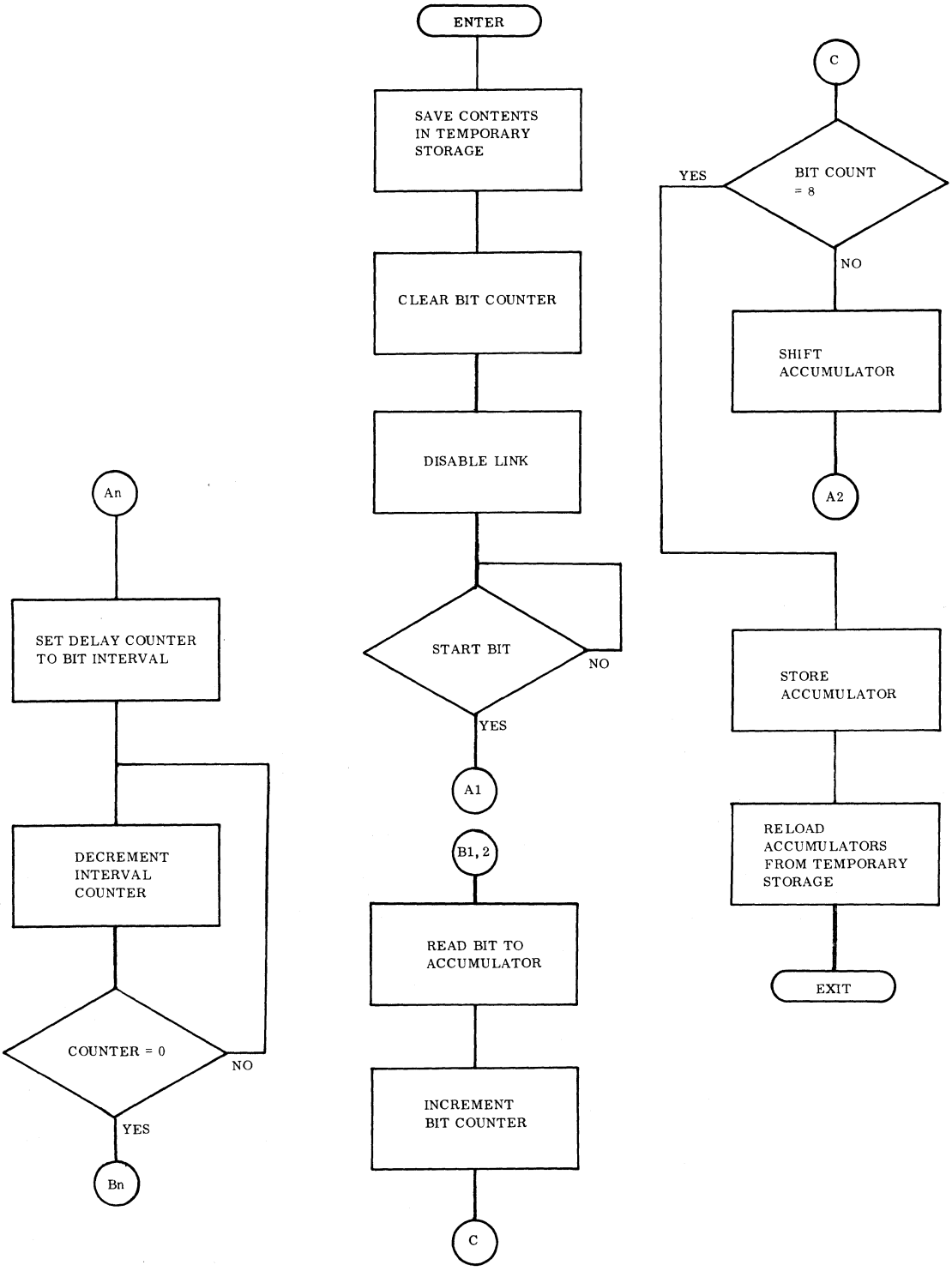
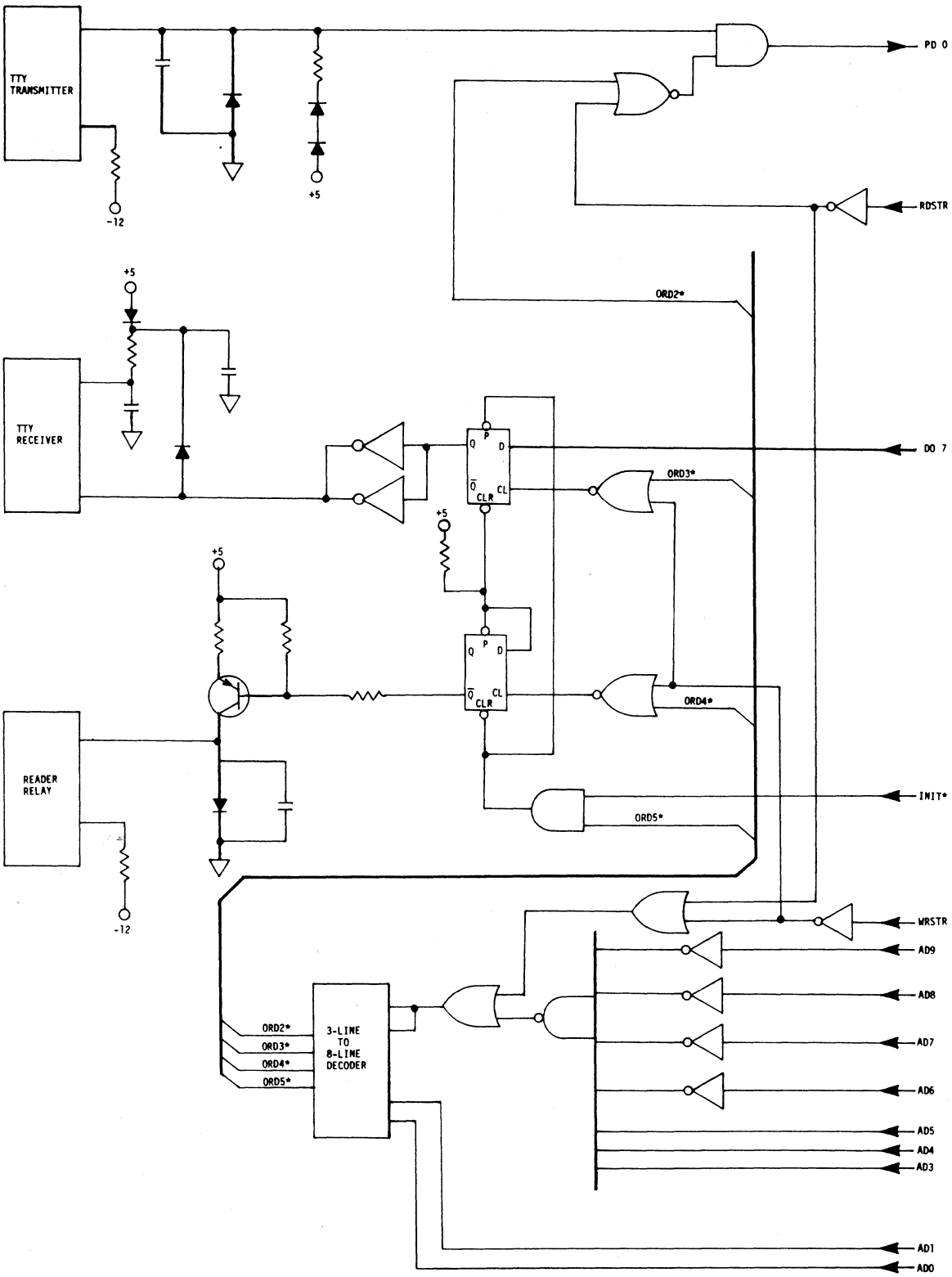


Figure 5-5. Teletype Receive Character Routine Using User Jump Condition



NOTE: Component values are determined by the user's individual interface requirements or component selection.

Figure 5-6. Serial Teletype Interface Using Device Address Schematic Diagram

As shown in figure 5-6, the data received from the Teletype transmitter may be read on the Peripheral Data Bus by a Load (LD) instruction. The bits comprising the character are input serially, least significant bit first; therefore, only one bit (bit 0) of the input byte contains the data. The remainder of the bits are undefined and should be marked by the program. (The undefined bits may also be used for status information.)

The data to the Teletype receiver may be sent on the Data Out Bus by a Store (ST) instruction. The bits comprising the character are output serially, least significant bit first. Since only one bit of the output byte contains data, the remainder of the bits are undefined and may be used for control information.

The Paper Tape Reader Control may be used for Teletypes with a Paper Tape Reader Circuit. This optional feature allows program control of the paper tape. This feature is especially desirable where data processing is concurrent with the read operation and the reader may be stopped during the data processing.

Four order codes decoded from the device address are acknowledged by the Teletype interface. The four typical order codes and their effect are listed in table 5-4.

5.6.1 Serial Teletype Device Address-controlled Transmit Character Routine

Figure 5-7 shows a program flowchart for a serial Teletype, single-character, transmit routine. The routine sends one character (right justified in AC0) to the Teletype. Since this program is written as a subroutine, the contents of the accumulators are saved in temporary storage before transmitting the character in the routine.

5.6.2 Serial Teletype, Device Address-controlled Receive Character Routine

Figure 5-8 shows a program flowchart for a serial Teletype, single-character, receive routine using device address control. The routine takes 8 bits of serial data from the Teletype transmitter interface and packs them in AC0. Since the program is written to be used as a subroutine, the contents of the accumulators are saved in temporary storage before receiving the character in the routine.

5.7 CARD READER INTERFACE

A simple program-controlled Card Reader Interface is described in the following paragraphs. The Documentation 600 (or 300) Card Reader has been used in the example; timing for this card reader is shown in figure 5-9. The IMP-8C processor initiates the operation by sending out a "pick" command to fetch a card. After the card has been picked, the Card Reader sends out 80 index marks that signify the start of each column of data. The IMP-8C program detects the presence of these marks and reads and stores each column of data into a 160-byte buffer.

Table 5-4. Serial Teletype/Address Control Typical Order Codes

Code	Effect
2	Bit 0 of the Peripheral Data Bus is set equal to the output of the Teletype. The IMP-8C transfers the contents to AC0. ('1' = Mark; '0' = Space)
3	Teletype receiver is set to the value of bit 7 of the Data Out Bus. ('1' = Mark; '0' = Space)
4	Paper Tape Reader is turned on.
5	Interrupt Request and Interrupt Enable Flags are turned off. Paper Tape Reader is turned off. Teletype receiver is set to idle (marking) state.

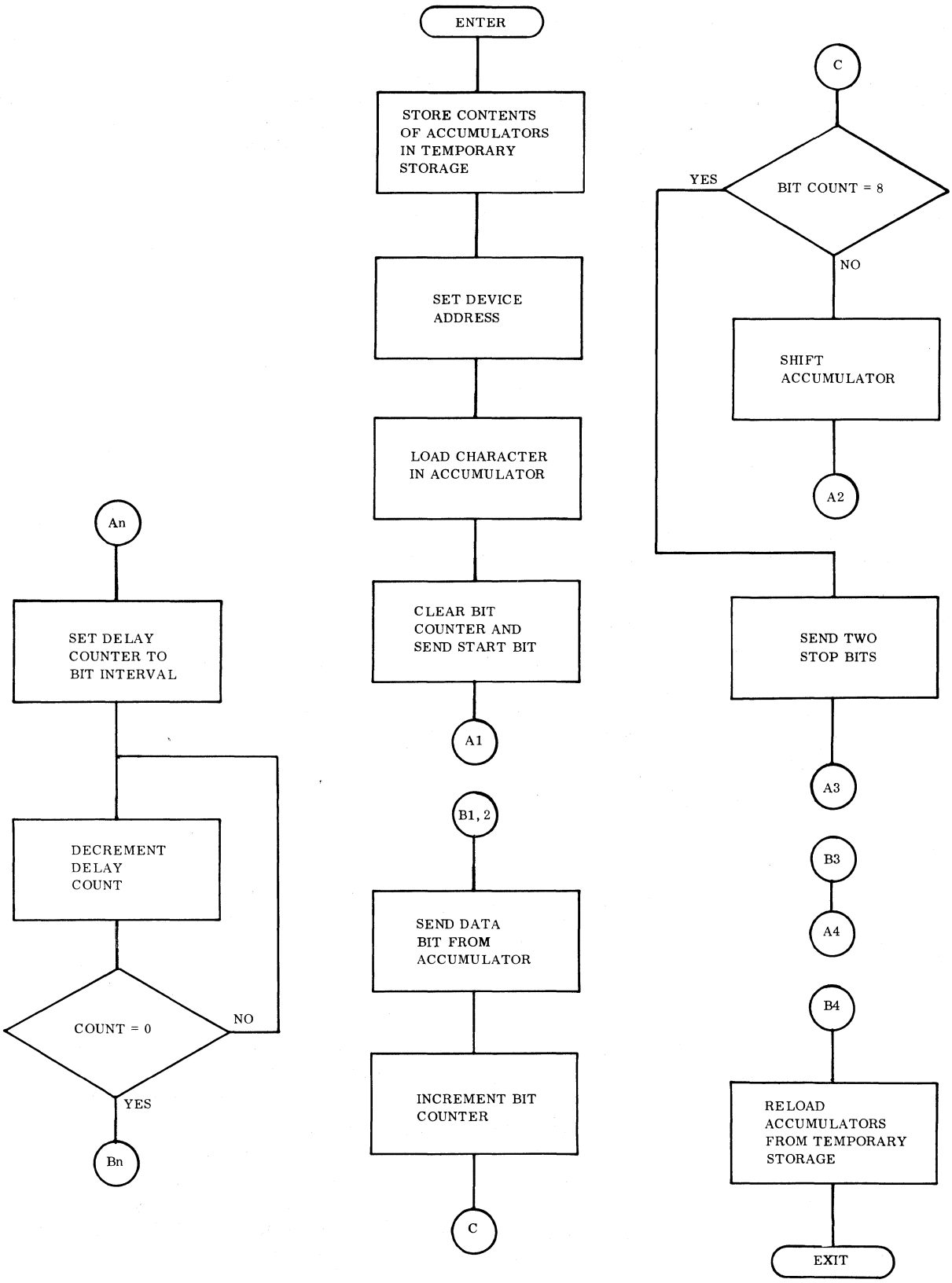


Figure 5-7. Device Address-controlled Transmit Character Routine

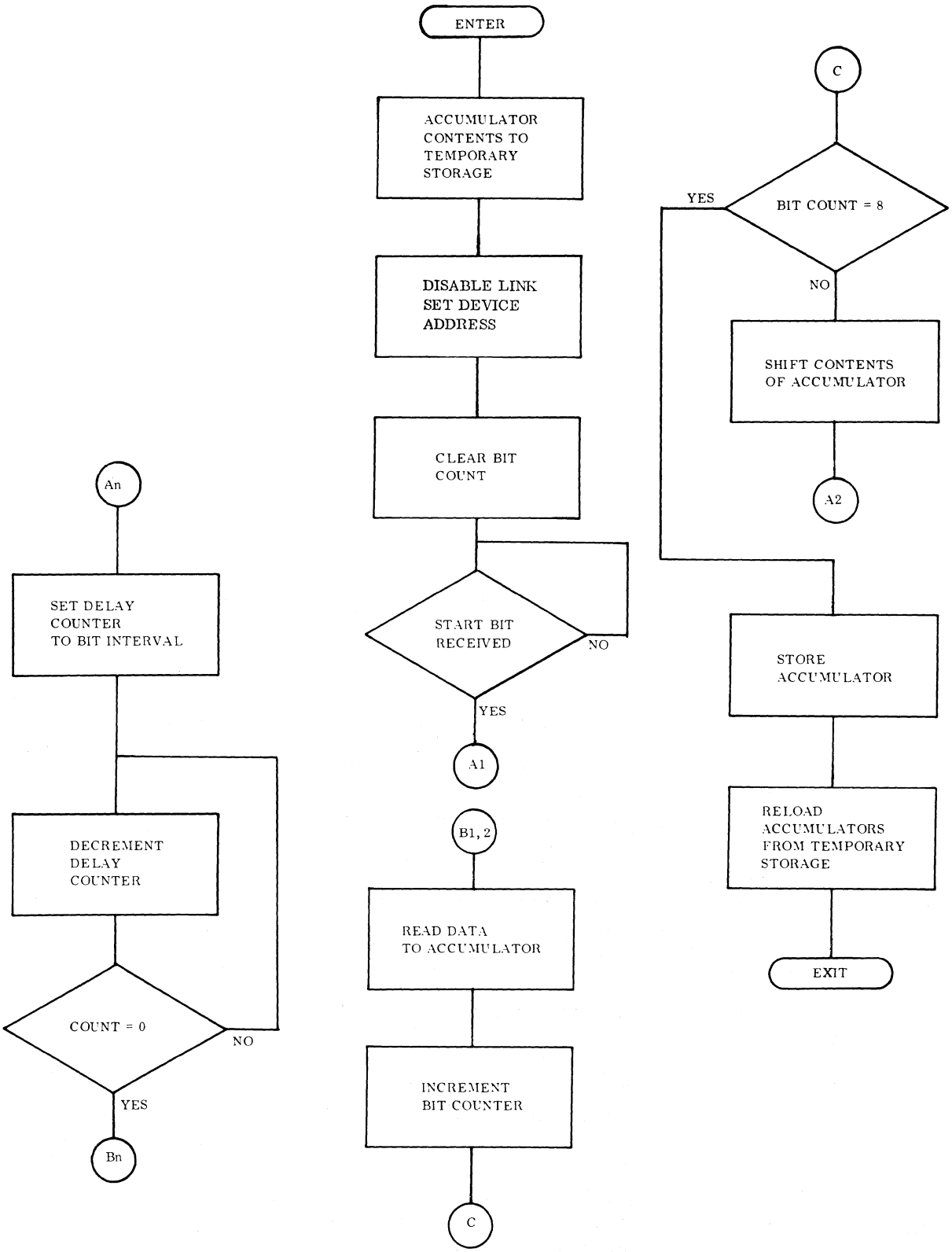


Figure 5-8. Serial Teletype Receive Character Routine Using Device Address

TIMING CHART	MODEL	A (msec)	B (msec)	C (μsec)	D (μsec)	E (msec)
	M200	24	6.25	1314	2014	8.05
	M300	24	2.60	435	870	102.66
	M600	24	2.60	435	870	3.48

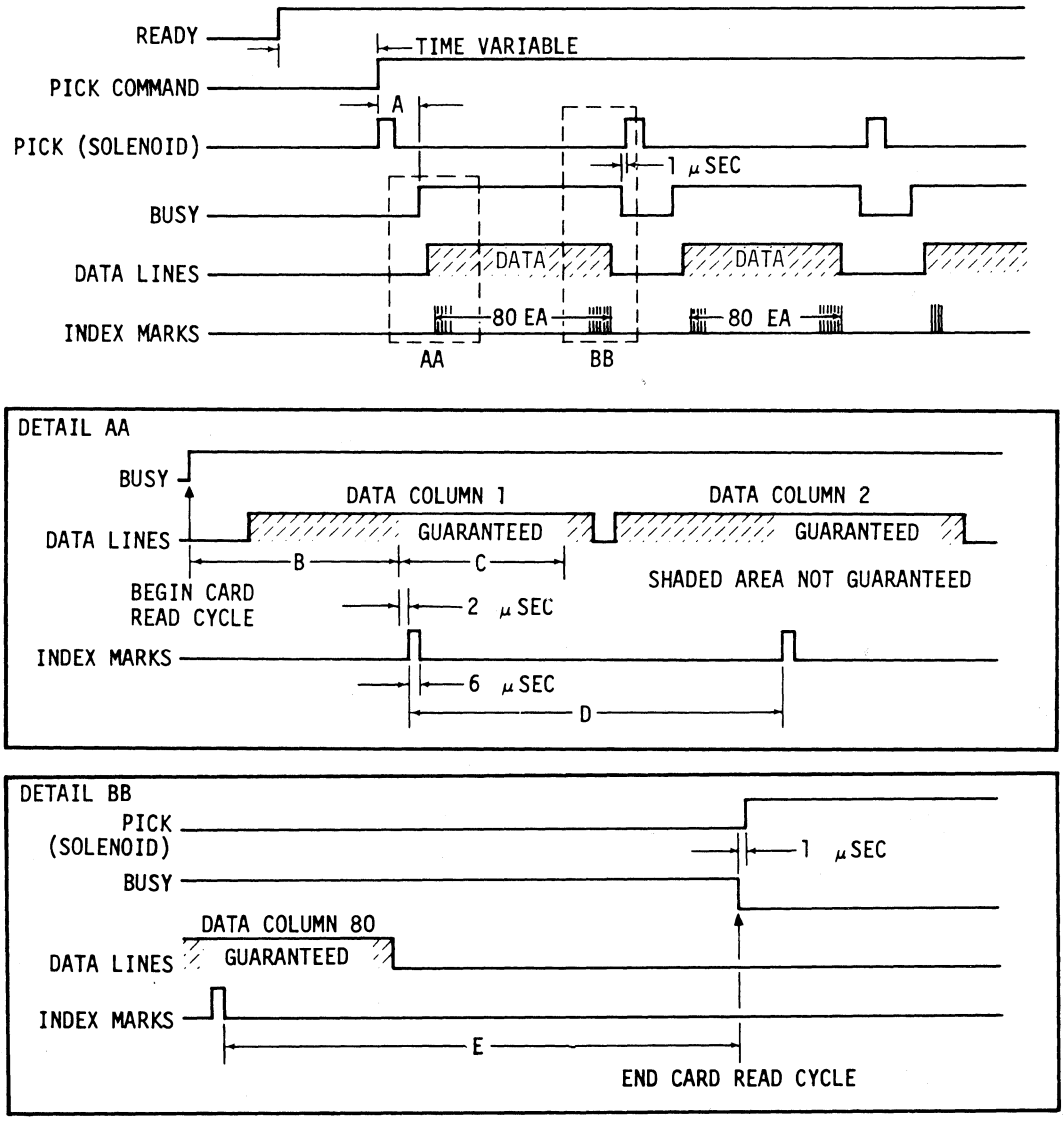


Figure 5-9. Standard Interface Timing for Documentation M-Card Readers

5.7.1 Card Reader Program-controlled Interface

Figure 5-10 shows an example of a program-controlled Card Reader Interface that can be implemented using five IC devices. Data input to the IMP-8C is accomplished by a Load (LD) instruction. Control operations are accomplished by Set Flag (SFLG) and Pulse Flag (PFLG) instructions.

5.7.2 Read Card Routine

Figure 5-11 shows a program flowchart for a read card routine using control flags and device address. The routine reads one 80-column card and stores the data in buffer storage. Since this program is written to be used as a subroutine, the contents of the accumulators are stored prior to reading the card.

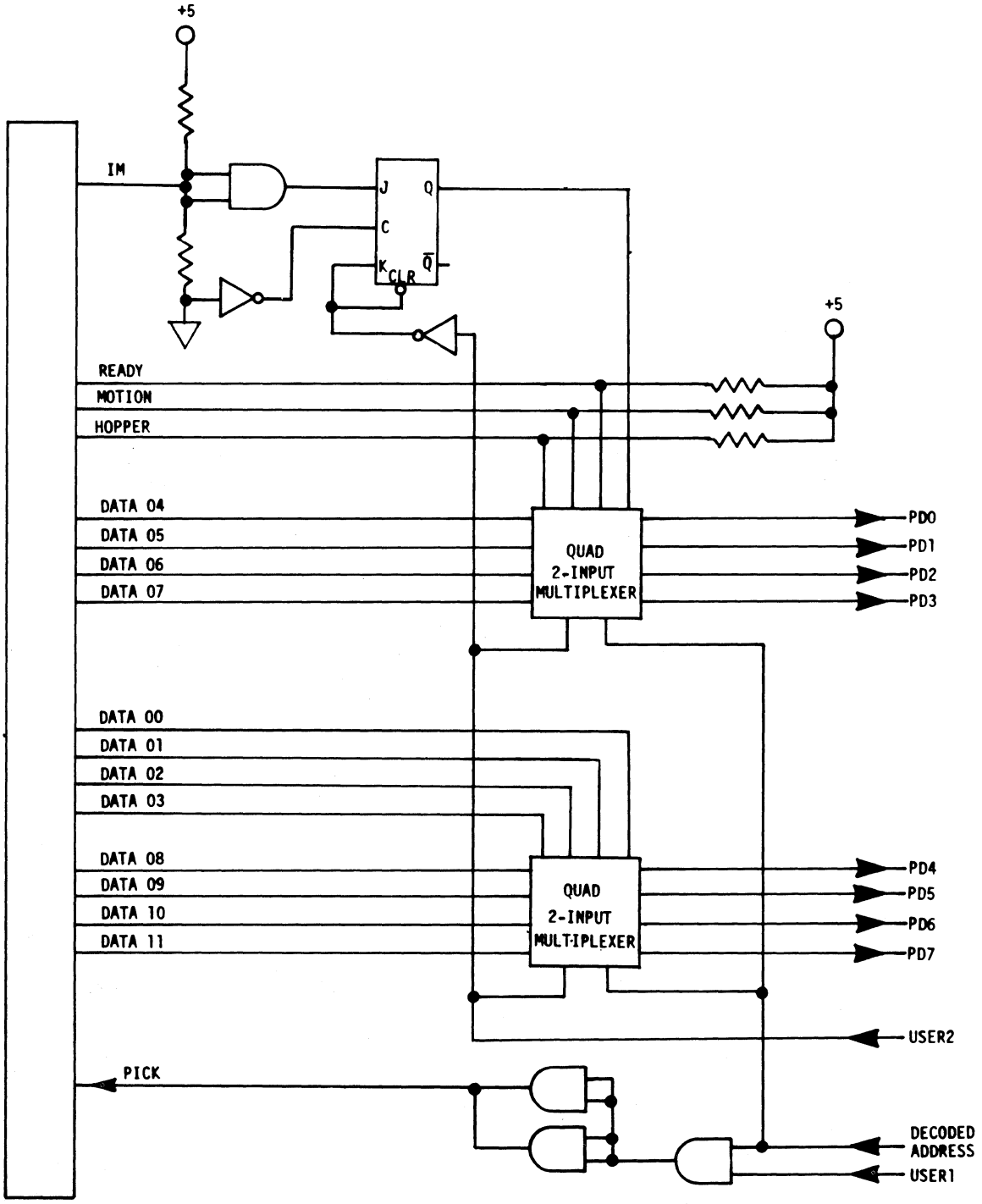


Figure 5-10. Card Reader Interface

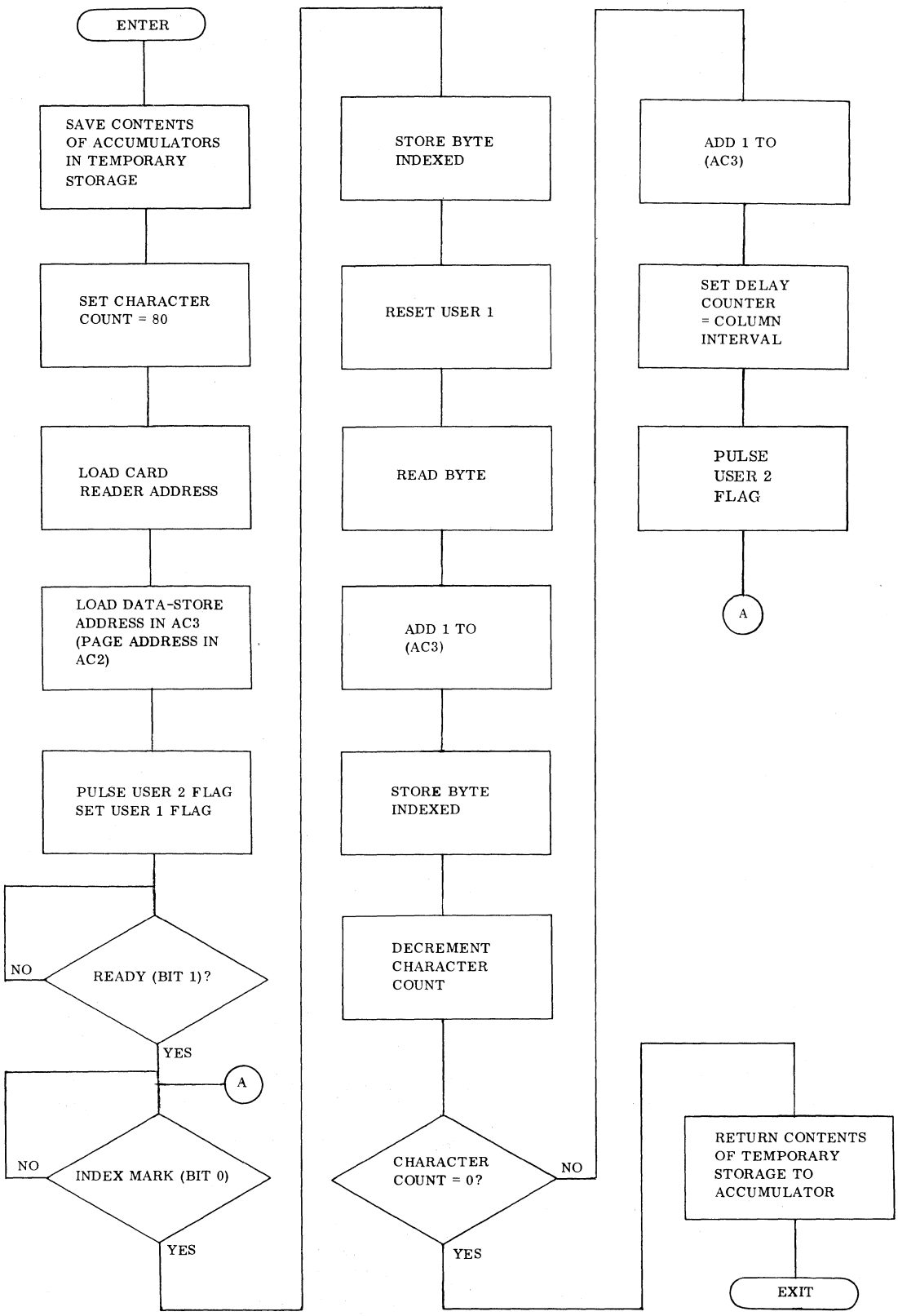


Figure 5-11. Read Card Routine

5.8 EXTERNAL MEMORY INTERFACE

The IMP-8C On-Card Memory can be augmented by external add-on memory to provide a maximum of 65,536 individually addressable bytes of memory. The add-on memory can be addressed by the contents of the IMP-8C Address Latches via the 16-bit Address Out Bus. The data bytes from the add-on memory are read via the 8-bit External Memory Data Bus to the IMP-8C Input Multiplexer. The data bytes to the add-on memory are sent via the 8-bit Data Out Bus from the IMP-8C Data Buffer. Provisions for the exchange of timing, status, and control signals between the IMP-8C and add-on memory are provided at the card-edge connector. Table 5-5 is a listing of the timing, control, and status signals used with the add-on memory that are available at the card-edge connector, their function, and pin number.

5.8.1 Timing

External memory read and write timing is similar to the on-card memory timing. As shown in figure 5-12, the External Memory Select (EMEMS*) Signal is generated when the address is set in the Address Latches, if AD11 through AD14 are set or the Peripheral Select (PS) Signal is set. In the basic IMP-8C system, AD15 is tied to PS, and EMEMS* is tied to the Internal Memory Select (IMS1*) Signal to enable the selection of the External Memory Data Bus (lines EMD0 through EMD7) by the IMP-8C Input Multiplexer. The Read Strobe (RDSTR) is generated by the Read Flag (RDFF) and is set at T2 of a read cycle. The Read Strobe is reset at T8 of the read cycle by C8B. Since the Read Strobe occurs before the actual data input time, it may be used to set the external memory data into buffer latches and thereby ensure that the data is stable during input to the CPU. The Buffered Write Strobe (BWRSTR) is generated by ANDing the Write Flag (WRFF) and the Clock Hold (CHOLD) Signal and is identical on the On-Card Memory Write Strobe (WRSTR).

Table 5-5. Add-on Memory Control, Timing, and Status Signals

Signal	Function	Pin
BWSTR	Buffered Write Strobe Signal	116
C2	System Clock Corresponding to T2	127
C4	System Clock Corresponding to T4	104
C6	System Clock Corresponding to T6	131
C8	System Clock Corresponding to T8	136
C81234	System Clock Corresponding to T8, T1-T4	130
C8B*	Buffered Master Clock Occurring During Last Half of T8	132
CR	Cycle Request Signal	125
ECL*	External Clock Hold Clear Signal	129
ECLK	External Clock	135
EN0-15	Decoded Addressing Signals, Enable 0-15	⚠1
EMEMS*	External Memory Select Signal	133
IMS1*, 2*	Internal Memory Select Signals	123, 120
OCLK	Output Clock Signal	134
RAMS*	Random Access Memory Select Signal	80
RDSTR	Read Strobe Signal	122
REFREQ	Memory Refresh Request Signal	124
RFRSH	Memory Refresh Initiate Signal	121

⚠1 See appendix B for a listing of the individual pin numbers.

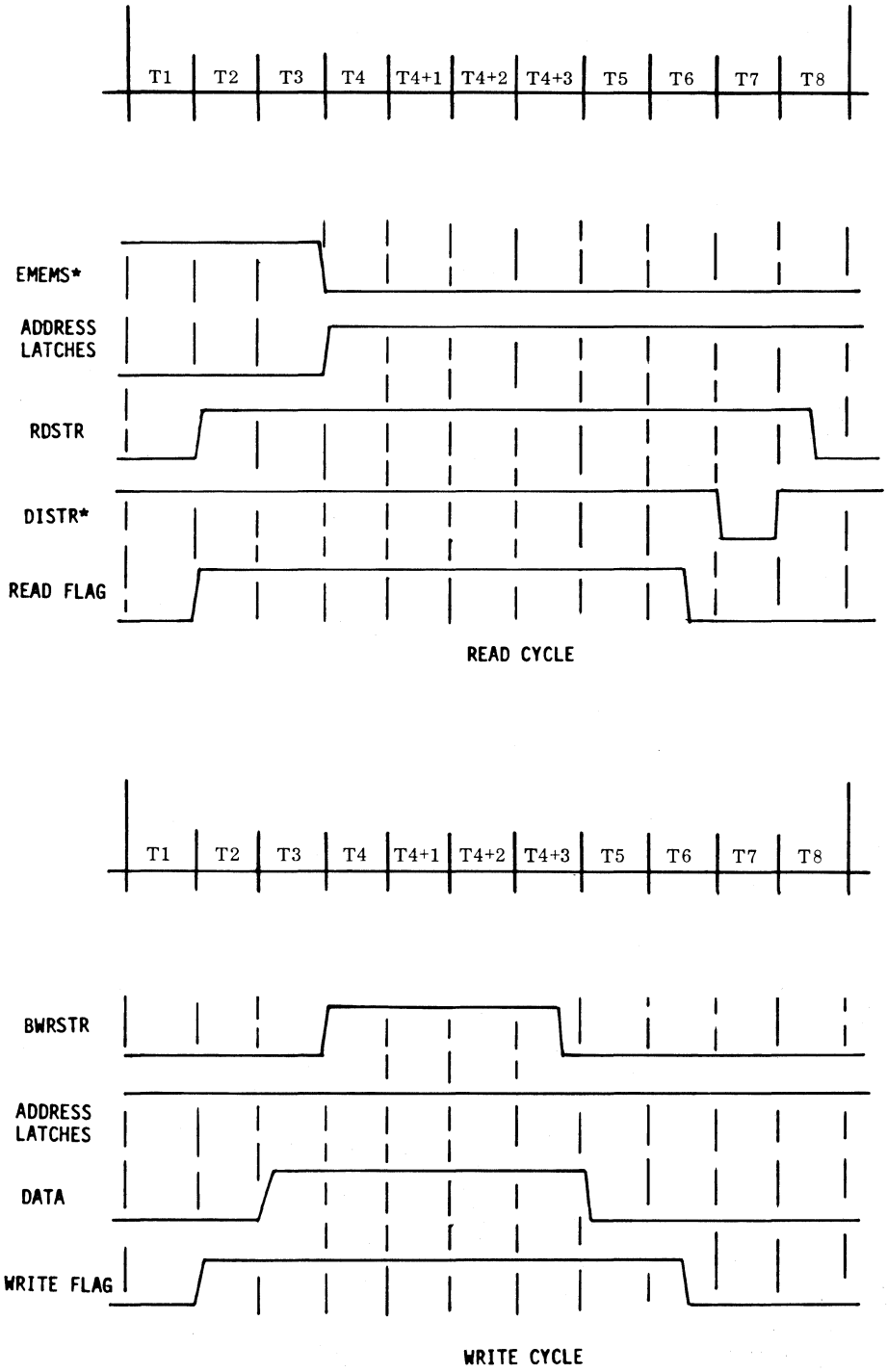


Figure 5-12. External Memory Timing Signals

5.8.2 Dynamic Memory Interface

Provisions are included in the IMP-8C circuits to enable external memory interfacing with dynamic memory modules. A dynamic memory is a volatile storage system requiring a refresh cycle. The IMP-8C contains the logic to acknowledge Memory Refresh Request Signals and to generate the required Refresh and Read/Write Initiate Signals.

When the IMP-8C receives a request for memory refresh (REFREQ) from the dynamic memory module, the IMP-8C sends a Memory Refresh (RFRSH) Signal at T4 if the IMP-8C is not busy with a read or write operation. If the IMP-8C is busy with a read or write operation, a Cycle Request (CR) Signal is sent to the memory at T4. The Cycle Request Signal is reset at T4+3 when the Clock Hold (CHOLD) Signal goes low.

5.8.3 8K by 8 Read/Write Memory Storage

The IMP-8P/008 8K by 8 Read/Write Memory Storage Card and associated Memory Timing and Control Card provide 8,192 bytes of memory, timing logic, and control logic to interface the storage card to TTL logic levels. Table 5-6 is a listing of the IMP-8P/008 8K by 8 Read/Write Memory Storage Card characteristics.

Table 5-6. IMP-8P/008 8K by 8 Read/Write Memory Storage Card Characteristics

Item	Characteristic
Storage Capacity	8,192 bytes (8 or 9 bits per byte)
Cycle Time	950 nanoseconds
Access Time	550 nanoseconds
Operating Modes	Write, write with byte control (2 zones), read, and refresh
Operating Temperature	0°C to 50°C
Power Required	+5 vdc, ± 5% -12 vdc, ± 5%
Physical Description	8-1/2-inch by 11-inch card with 50-position, 100-pin edge connector
Input - Addresses	15 lines, (A0-A15) ⚠1
Data	Up to 9 lines (DI)
Controls	5 lines: Cycle Request (CR) C0: mode control (GROUND) C1: mode control (BWRSTR) External Refresh (Ext RFSH) Memory Select (MS*)
Output - Data	Up to 9 lines
Controls	2 lines: Refresh Request (RFSH REQ*) Refresh in Progress (RFSH PROG*)

⚠1 Corresponding to AD0-AD15 Address Line on the IMP-8C

All cycles except Refresh are initiated by raising CR (Cycle Request) high. The type of cycle initiated is determined by Control Lines C0 and C1 as shown in the truth table.

TRUTH TABLE

C1 (BWRSTR)	C0 (GND)	
0	0	Read
1	0	Write

Read - Read cycles are initiated by Cycle Request (CR); this causes the memory to read data previously stored at the memory location currently addressed by AD0-AD15.

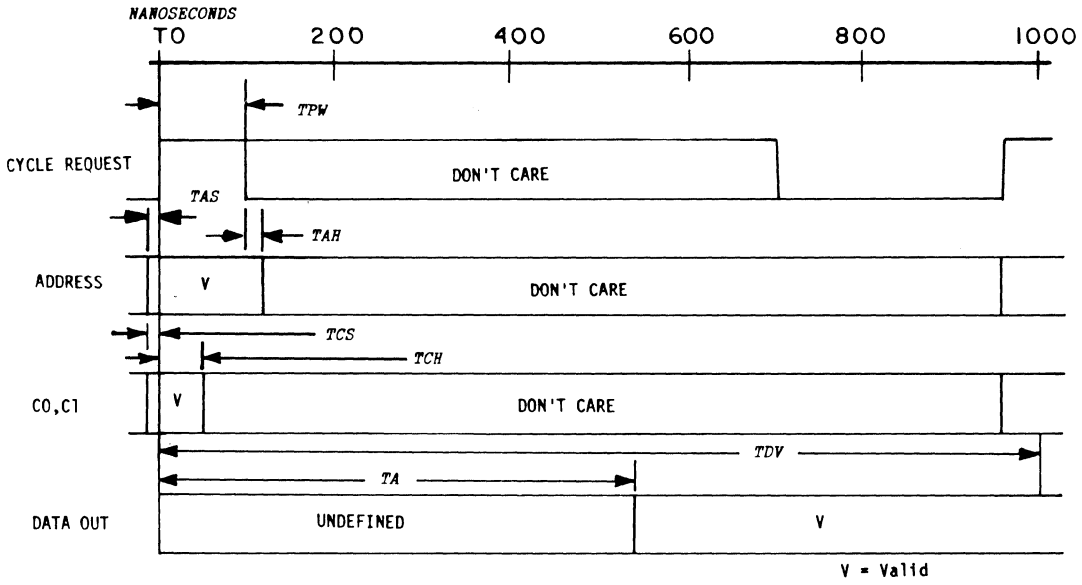
Write - Write cycles are also initiated by CR. During write cycles, the byte present on the input lines at the beginning of the cycle is written into the address that was present at the beginning of the cycle.

Refresh - The need for a refresh cycle is determined by the memory system, which times out at an interval of 50 microseconds. At this time, the Refresh Request Line is activated for 10 microseconds. Normal operation continues until either an external refresh is received or the 10-microsecond interval expires. The standard instruction set provides that the 10-microsecond interval never expires. While the refresh cycle is in progress, all inputs to the system are ignored. At the end of the refresh cycle, all inputs are activated and normal operation resumes.

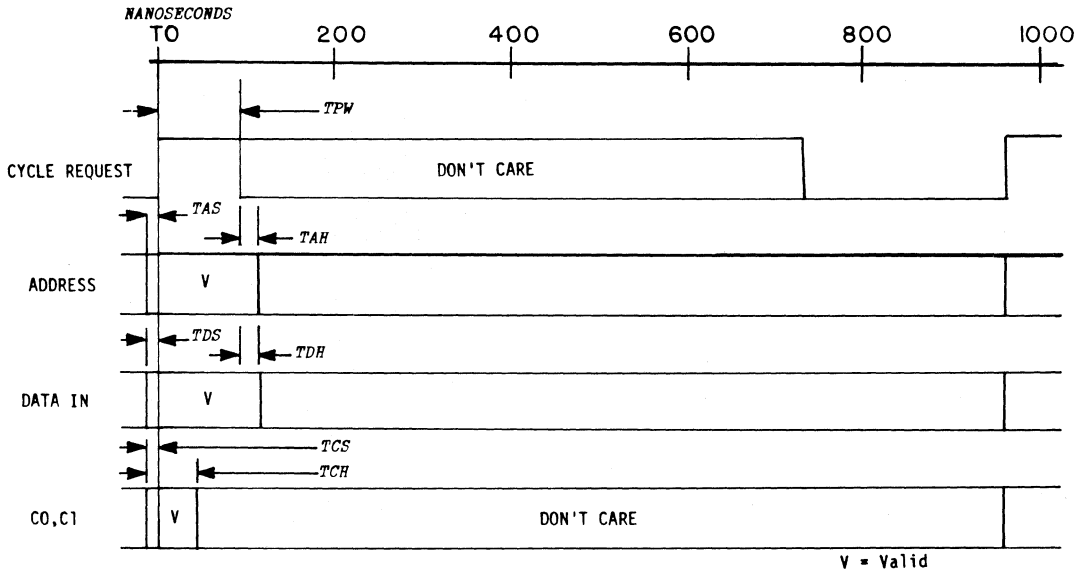
Table 5-7 is a listing of the timing characteristics of the IMP-8P/008 Memory Storage Signals, and figure 5-13 shows the timing of IMP-8P/008 Memory Storage Signals.

Table 5-7. IMP-8P/008 8K by 8 Memory Storage Signal Timing Characteristics

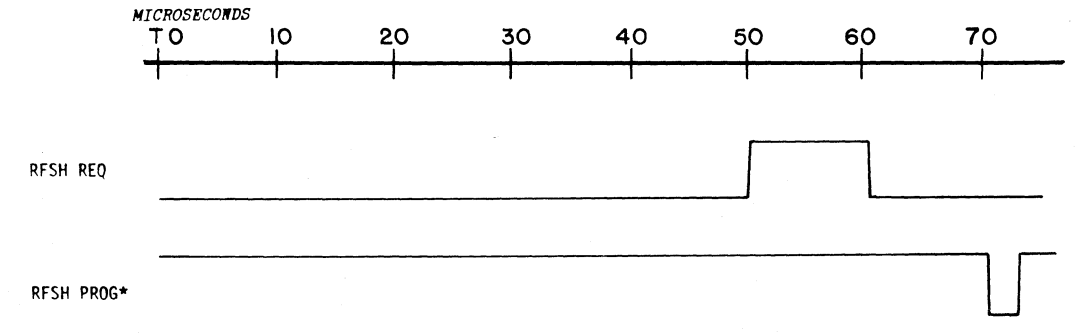
Signal	Timing	
	Minimum	Maximum
Cycle Request Pulse Width (tpw)	100 ns	700 ns
Set-up Time (tAs)	0 ns	
Hold Time (tAh)	20 ns	
Control Set-up Time (tcs)	0 ns	
Control Hold Time (tch)		50 ns
Memory Cycle Active (tMBL)		125 ns
Memory Cycle Inactive (tMBH)		950 ns
DATA Active (tODAL)		425 ns
DATA Inactive (tODAH)		625 ns
Access Time (tA)		550 ns
Data Valid Time (tDV)	0.95 μs	1.5 μs
Input Data Set-up Time (tDS)	0 ns	
Input Data Hold Time (tDN)	20 ns	



READ CYCLE



WRITE CYCLE



REFRESH CYCLE

Figure 5-13. IMP-8P/008 8K by 8 Memory Timing

Chapter 6

INTERRUPTS

6.1 GENERAL INFORMATION

In the IMP-8C, there are two processor Interrupt Request Lines: one for peripheral devices, and one for stack overflow interrupts. All peripheral devices capable of interrupting are wired to the main Interrupt Request Line, INTRQ. If any device generates an interrupt request, the line goes high. If the 16-level Last-In/First-Out Stack (LIFOS) has a nonzero word in the sixteenth position the Stack-Full Line (STFX) is activated. In either case, if the master Interrupt Enable Flag (INEN) is set, the processor is interrupted after it completes execution of the current instruction. The SFLG and PFLG instructions may be used to enable or disable the INEN Flag.

If the stack-full interrupt is not used, STF to interrupt control must be grounded for proper interrupt response. See chapter 4, paragraphs 4.8 and 4.11.3.

6.2 INTERRUPT RESPONSE

The processor responds to an interrupt request when fetching the next instruction. Six microinstructions are executed in order to transfer program control to the interrupt service routine, according to the following 3 steps.

1. The Interrupt Line is tested during the first cycle of the instruction fetch cycle. If the line is active, steps 2 and 3 are executed.

2. The contents of the Program Counter (PC0 and PC1) are pushed onto the stack; the Interrupt Enable (INEN) Flag is disabled (cleared) to prevent further interrupts.

3. PC0 and PC1 are cleared to 0000 and the instruction at that location is fetched, thus initiating the interrupt service routine.

The interval between the time of the Interrupt Request (INTRQ true) and the fetching of the first location of the interrupt routine is the interrupt latency or overhead time.

Latency time = EC + 7.4T

Where:

EC = time to complete execution of instruction at time of interrupt (varies according to the instruction).

T = time for one IMP-8C microcycle

The processor detects the interrupt when fetching the next instruction and then goes through the 3 steps described above before executing the instruction at location 0.

As an example, let T = 1.4 microseconds; then a register exchange instruction (five execution cycles) interrupted at cycle four has a latency time of (5-4) + 7.40 1.4 = 11.76 microseconds. It should be noted that in calculating EC, the number of read or write cycles remaining adds 0.4 microcycle each.

6.3 INTERRUPT GENERATION

A peripheral controller requiring interrupt service by the processor sets its Interrupt Request Flag, thus causing a true signal on the Interrupt Request Line. The Interrupt Request Line (INTRQ) is common to all peripheral controllers and only requires that one peripheral device be requesting service in order to set the line to the true state. INTREQ is connected to INTRQ through a pin on the card-edge connector.

The main program, when ready, sets the master Interrupt Enable Flag (INEN) via the SFLG 7 instruction. This indicates that the processor may perform an interrupt service. If INEN is enabled and either STFX or INTRQ is true, then the Interrupt Line (INT) forces a processor interrupt after execution of the current instruction is completed.

6.4 INTERRUPT SERVICE CONSIDERATIONS

The interrupt service routine is responsible for:

1. Saving and restoring the accumulators (AC0, AC1, AC2, AC3), LIFOS, and RALU Status Flags.
2. Checking for Stack-Full Line =1 and distinguishing between stack-full and a stack-overflow interrupt. The latter condition requires special servicing.
3. Determining the highest priority peripheral device requesting service and branching to the appropriate coding.
4. Servicing the selected interrupt request, resetting the device interrupt, and returning control to the main program.

6.4.1 Saving and Restoring Registers, Stack, Flags

It is recommended that the four accumulators and the stack and the status flags be saved in main memory at the beginning of the interrupt service routine. If this is not done, and the routine uses the accumulators or flags, temporary results and status are lost. Saving the stack prevents stack-full conditions that could occur if:

1. A higher priority interrupt (higher level in a multi-level system) comes up before the present interrupt servicing is complete.
2. The interrupt service code calls subroutines. (NOTE: With case 1 or 2, return addresses are pushed onto the stack.)
3. The interrupt service code saves data on the stack (upon completion of interrupt service the saved information should be restored).

6.4.2 Stack-Full Interrupt

If the LIFOS has a nonzero word in the last (sixteenth) position, the Stack-Full Line (STFX) is activated. At T3 and T4, Clock (C34) the Stack-Full Control Signal (STF) is set. If INEN is set, a Stack-Full interrupt is initiated. This interrupt causes the PC to be pushed onto the stack, thus losing the last two stack words.

The systems programmer must take some precautions to prevent stack-overflow interrupts and to guarantee that data on the stack is not lost.

1. Two protection words of all ones should be initially pushed onto the stack to prevent loss of data on a stack-full interrupt.

2. Words of all zeros must not be pushed onto the stack. Otherwise a stack-full condition might go undetected.

3. The stack should be saved in main memory before entering a loop where subroutines are called. It should be restored after exiting from the loop.

4. When the Stack-Full Line is set, the interrupt service code should distinguish between an external device interrupt causing the stack-full and a microprogram-forced interrupt. In the former case, control is transferred back to the service code. In the latter, the protection words of all ones are lost, and stack data is in danger of being lost. Control may be transferred to a recovery routine after a diagnostic error message is printed, or the program may halt.

In most applications, use of the stack is reserved for subroutine return addresses, interrupt return addresses, and for saving the RALU Flags while servicing interrupts. If the above precautions are taken, stack overflow is unlikely and should be treated as a systems programming error.

6.4.3 Identifying Peripheral Device Interrupts

When an interrupt occurs, the interrupt service routine directs the processor to determine the devices requiring service and to select one peripheral device. This may be accomplished as follows:

1. An interrupt select status order is sent out to all peripheral controllers. Upon receipt of the interrupt select status order, the peripheral controller places its interrupt request status on its assigned bus line.

2. Each peripheral device is assigned one of the 8 peripheral data bus lines to report its interrupt status. Each peripheral device responds simultaneously with other peripheral devices, indicating whether or not it requires interrupt servicing. A binary 1 indicates a service request. Typical interrupt assignments are shown in table 6-1

3. The interrupt service routine resolves interrupt priority and selects the peripheral device for interrupt servicing. This is done by sequentially checking each bus line (polling) for an active request and branching when the first such line is found. Devices are assigned a fixed priority on the bus.

Assignments are in decreasing order of priority. Thus, the real time clock has the highest priority. Priority is resolved at the time of interrupt recognition. For one device to interrupt another a multi-level interrupt structure is required.

Table 6-1. Typical Interrupt Select Status Bit Assignments

Bit	Assigned Peripheral
0	Real Time Clock
1	Disc
2	Display
3	Card Reader
4	Parallel Teletype or Printer
5	Pushbuttons
6	Thumbwheel Switches
7	Unassigned

6.4.4 Peripheral Device Interrupt Service

After a peripheral device obtains interrupt access:

1. The status of that device must be sensed.
2. The device conditions that caused the interrupt are determined.
3. The device interrupt must be serviced.
4. The device interrupt request must be reset.
5. Control must be transferred back to the main program.

The implementation of these requirements is illustrated in the following example: when the processor is interrupted, the contents of the program counter are pushed onto the stack, and a jump or jump indirect instruction at location 0 is executed; this transfers control to the interrupt service routine. After saving the accumulators, stack and flags and testing for stack full, the interrupt status of all external peripheral devices is read. This is done by establishing a select interrupt status order. The highest priority device interrupting is determined via a branch-chain identification sequence according to the priority established by table 6-1.

As shown in figure 6-1, the pushbutton is used to signal the processor that the status of the unit specified by the thumbwheel switches is to be displayed on the lamp. The processor is monitoring the status of all units in real time and stores the resulting information in RAM. When the operator wishes to inspect a given unit, he positions the thumbwheels to the address of the unit and presses the pushbutton.

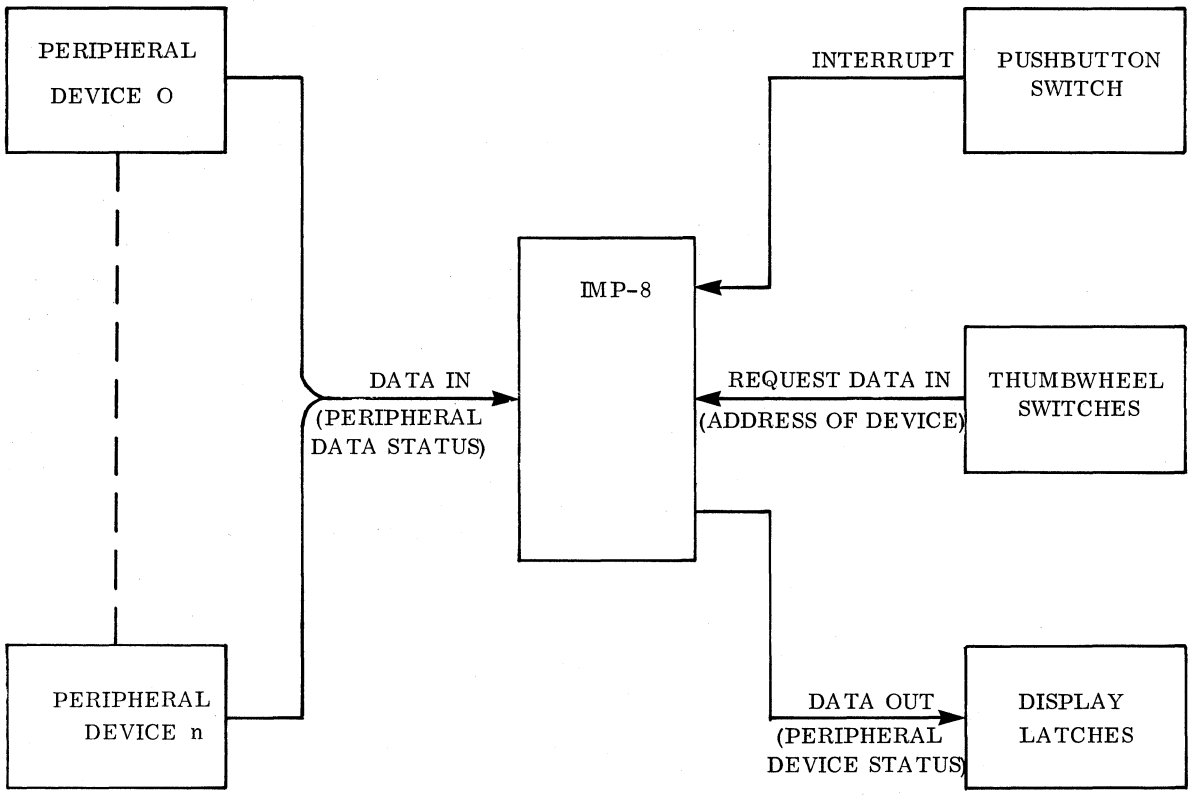


Figure 6-1. Typical Peripheral Device Interrupt Service

The interrupt service coding converts the thumbwheel input from BCD to binary and uses the result to index the status data table. The data selected is then output to a lamp driver. Up to 16 status points may be displayed. After servicing the request the device interrupt indicator is reset; saved information is restored; and control is transferred back to the main program via the Return from Interrupt (RTI) instruction.

The RTI instruction causes PC0 and PC1 to be loaded with the top two words of the stack. It also sets the INEN Flag.

6.5 MULTILEVEL INTERRUPTS

It is possible to use the IMP-8C for multileveled interrupt service by use of the control flags. These flags may be used as Interrupt Enable Flags for up to 4 individual levels; they can be modified through use of the SFLG or PFLG instructions. A typical arrangement that makes use of the available INTRQ input is shown in figure 6-2.

The processor responds to inputs on the Interrupt Request Lines that are labeled INTREQ1 and INTREQ2 in figure 6-2. Several devices may be wire-ORed to each of these Interrupt Request Lines. If any device generates an interrupt request, the line will go high and interrupt the processor if that level of interrupt is enabled. There is a separate Interrupt Enable Flag for each of the interrupt levels and a master Interrupt Enable (labeled INEN) for all levels plus the stack-overflow interrupt. The INEN Flag is one of the IMP-8C control flags. It is modified by use of the Set Flag (SFLG) and the Pulse Flag (PFLG) instructions. The availability of several different interrupt levels provides a convenient means of controlling interrupts for devices of different priority in a system with a number of peripherals. When only one Interrupt Request Line is used, the interrupt service program must issue an order to each device of lower priority than the one being serviced to reset the lower priority Interrupt Enable Flags on the peripheral controllers. This could be very time consuming in a system with many peripherals. In a system with several interrupt levels, all devices of like priority are tied to the same Interrupt Request Line. For all devices on an individual level that have a common Interrupt Enable Flag at the processor, the Interrupt Enable Flag can disable all devices on that level simultaneously. When an interrupt occurs, lower priority levels are therefore disabled in a minimum amount of time.

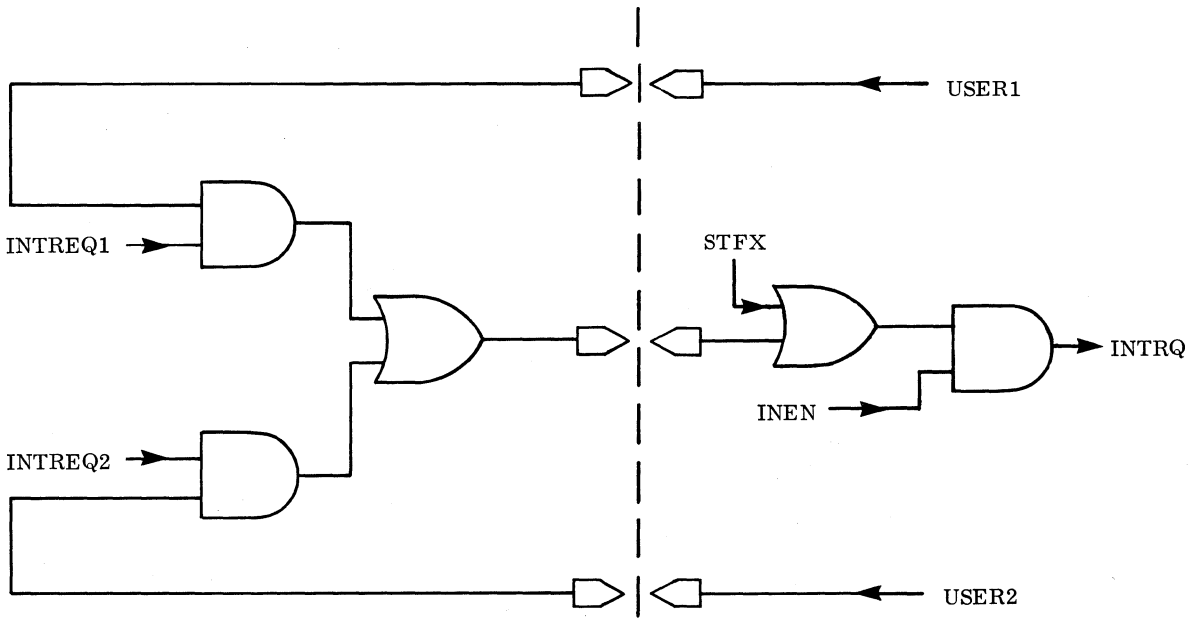


Figure 6-2. Multilevel Interrupt Interface

Appendix A

SIGNAL MNEMONICS

AD0-15	Memory and Peripheral Device Address Lines
ADX1	High-Order Address Latches Flag
BASEPG	Base Page Clear Enable Flag
BPCL	Base Page Clear Signal to High-Order Address Latches
BWRSTR	Buffered Write Strobe Signal
C1-C8	System Clocks Corresponding to Microcycle T1 to T8
C34	System Clock Corresponding to Microcycle T3 and T4
C81234	System Clock Corresponding to Microcycle T8, T1 - T4
C4F	Buffered C4 Clock
C5B	Buffered Master Clock during last half of microcycle T5
C8B	Buffered Master Clock during last half of microcycle T8
CHOLD	Master Clock Inhibit to System
CLK	Master Clock
CNT	Program Counter Increment Clock
CR	Cycle Request Signal
CYOV	Carry or Overflow Signal
DI0-7	Data In Lines
DISTR	Data Input Strobe Signal
DO0-7	Data Out Lines
ECL	External Clock Hold Clear Signal
ECLK	External Clock
EN0-15	Decoded Address Lines; enable 0-15
EMD0-7	External Memory Device Data Input Lines
EMEMS	External Memory Select Signal
EMS	External Memory Selected to Memory Data Input Bus
ENCTL	Enable Control Signal
ESTKCL	External Stack Clear Signal
FLEN	Flag Enable Signal
HALT	Halt Flag
HOCSH	High-Order Carry Shift Signal
IMS1, 2	Internal Memory Select Signal
INEN	Interrupt Enable Signal
INIT	Initiate Signal (system clear)
INT	Interrupt Signal
INTR	Interrupt Request Flag
INTRQ	External Interrupt Request Signal
JFA0-3	Jump/Flag Address Lines
LJFA0-3	Latched Jump/Flag Address Lines
LOCSH	Low Order Carry/Shift Signal
LPC	Load Program Counter Flag
LSTART	Latched Start Signal
LUSERJC	Latched User Jump Condition Signal

MDI0-7	Memory Data Input Lines
NCB0-3	Complemented Control Bit Signals
NFLEN	Complemented Flag Enable Signal
OCLK	Output Clock Signal
PCENBL	Program Counter Enable Signal
PCOE	Page Counter Output Enable Signal
PD0-7	Peripheral Device Data Input Lines
PH1,3,5,7	Phase Clocks corresponding to microcycle T1,3,5,7
PS	Peripheral Select Signal
RAMS	Random Access Memory Select Signal
RDFF	Read Data Flag
RDSTR	Read Strobe Signal
REFREQ	Memory Refresh Request Signal
REQ0	R-bus equal 0 signal
RFRSH	Memory Refresh Initiate Signal
ROMS0-7	Read only Memory Select Lines
RPC	Read Program Counter Signal
SEL	Select Flag
SININ	Sign-in Signal
START	External Start Signal
STF	Latched Stack Full Signal
STFX	Stack Full Signal
SVGG	Switched VGG
SVRST	Save/Restore Flag
SYSCLR	External System Clear Signal
TENCTL	TTL-level Enable Control Signal
UNUS2	Unused Microprogrammable Flag
USER1-4	User Flags 1-4
USERJC	User Jump Condition
VGG	-12 Volts dc
VLL	Ground
VSS	+5 Volts dc
WRFF	Write Flag
WRSTR	Write Strobe

Note: An asterisk (*) after a signal mnemonic denotes that the active state for that signal is a low or '0' condition.

Appendix B

PIN LIST

Pin Number	Signal	Loading (Milliamperes)
1-4	GROUND	
5-8	+5V (Vcc)	
9	HALT	9.6
10	USER1	9.6
11	UNUS2	9.6
12	DO7	12.8
13	USERJC	-3.2
14	DO6	12.8
15	START	-3.2
16	DO3	12.8
17	INTRQ	-3.2
18	DO2	12.8
19	USER4	9.6
20	DO5	12.8
21	DO1	10.8
22	DO0	10.8
23	ROMS0*	-1 μamp
24	ROMS1*	-1 μamp
25	ROMS2*	-1 μamp
26	ROMS3*	-1 μamp
27	USER2	9.6
28	EMD3	-1.6
29	PD3	-1.6
30	EMD2	-1.6
31	PD2	-1.6
32	EMD0	-1.6
33	PD0	-1.6
34	EMD1	-1.6
35	PD1	-1.6
36	EMD7	-1.6
37	EMD6	-1.6
38	PD6	-1.6
39	USER3	9.6
40	PD7	-1.6
41	PD5	-1.6
42	EMD5	-1.6
43	PD4	-1.6
44	EMD4	-1.6
45	DO4	12.8
46-50	Blank (No signal)	
51-52	-9V (Vdd)	
53-70	Blank (No signal)	
71-72	GROUND	
73-74	Blank (No signal)	
75	PCOE	15.2
76	ROMS4*	-1 μamp
77	ROMS5*	-1 μamp
78	ROMS6*	-1 μamp

Pin Number	Signal	Loading (Milliamperes)
79	ROMS7*	-1 μamp
80	RAMS*	-3 μamp
81	PS	-6.8
82	AD7	15.9
83	AD6	15.9
84	AD5	15.9
85	AD4	15.9
86	AD3	15.9
87	AD0	15.9
88	AD1	15.9
89	AD2	15.9
90	EN12*	16.0
91	EN11*	16.0
92	AD9	14.4
93	EN10*	16.0
94	AD14	16.0
95	EN9*	16.0
96	EN14*	16.0
97	EN8*	16.0
98	AD15	16.0
99-100	-12V (Vgg)	
101	EN7*	16.0
102	AD13	14.4
103	EN6*	16.0
104	C4	1.6
105	EN5*	16.0
106	AD10	14.4
107	EN4*	16.0
108	AD11	12.8
109	EN3*	16.0
110	AD12	14.4
111	EN2*	16.0
112	ESTKCL*	-2.0
113	EN1*	16.0
114	TENCTL	8.0
115	EN0*	16.0
116	BWRSTR	20.0
117	AD8	14.4
118	EN13*	16.0
119	EN15*	16.0
120	EMS2*	-1.6
121	RFRSH	20.0
122	RDSTR	16.0
123	IMS1*	-4.0
124	REFREQ	-2.0
125	CR	16.0
126	INIT*	11.6
127	C2	6.4

Pin Number	Signal	Loading (Milliamperes)
128	SYSCLR*	-3.0
129	ECL*	-3.2
130	C81234	8.0
131	C6	6.4
132	C8B*	13.6

Pin Number	Signal	Loading (Milliamperes)
133	EMEMS*	16.0
134	OCLK	10.8
135	ECLK	-2.0
136	C8	6.0
137-140	+5V (Vcc)	
141-144	GROUND	

Appendix C

IMP-8C BASIC INSTRUCTION SET AND EXECUTION MICROCYCLES

Instruction	Mnemonic	Execution Microcycles
GENERAL REGISTER-MEMORY REFERENCE (2-byte instructions)		
Load	LD	8, 10 if indexed
Store	ST	10, 12 if indexed
Add	ADD	8, 10 if indexed
Skip if Not Equal	SKNE	9, 11 if indexed (+3 if skip)
ACCUMULATOR 0-MEMORY REFERENCE (2-byte instructions)		
AND	AND	8, 10 if indexed
OR	OR	8, 10 if indexed
Skip if AND is Zero	SKAZ	9, 11 if indexed (+3 if skip)
Subtract	SUB	8, 10 if indexed
ACCUMULATOR 0-INDIRECT MEMORY REFERENCE (2-byte instructions)		
Load Indirect	LD@	12
Store Indirect	ST@	13
IMMEDIATE (2-byte instructions)		
Load Immediate	LI	5
Add Immediate, Skip if Zero	AISZ	6 (+3 if skip)
JUMP (2-byte instructions)		
Jump	JMP	10, 11 if base page
Jump to Subroutine	JSR	14, 15 if base page
Jump Indirect	JMP@	12
Jump to Subroutine Indirect	JSR@	18
RETURNS (1-byte instructions)		
Return from Interrupt	RTI	5
Return from Subroutine	RTS	4
MEMORY MODIFY (2-byte instructions)		
Increment, Skip if Zero	ISZ	11, 13 if indexed (+3 if skip)
Decrement, Skip if Zero	DSZ	11, 13 if indexed (+3 if skip)
BRANCH-ON CONDITION (2-byte instructions)		
Branch-On Condition	BOC	7, 6 if branch occurs

Instruction	Mnemonic	Execution Microcycles
HALT (1-byte instruction)		
Halt	HALT	Indefinite
ACCUMULATOR TO ACCUMULATOR (1-byte instructions)		
Register-to-Register Add	RADD	3
Register-to-Register EXCLUSIVE OR	RXOR	3
Register-to-Register Exchange	RXCH	5
Register-to-Register AND	RAND	3
SINGLE ACCUMULATOR (1-byte instructions)		
Push Register onto Stack	PUSH	3
Pull Register from Stack	PULL	3
Exchange Register and Stack	XCHRS	5
Twos Complement	COMP2	3
Shift Left	SHL	3
Shift Right	SHR	3
Rotate Left	ROL	3
Rotate Right	ROR	3
Register Exchange	RXCH	5
FLAG (1-byte instructions)		
Set Flag	SFLG	3
Pulse Flag	PFLG	3
Push Flags onto Stack	PUSHF	4
Pull Flags from Stack	PULLF	4

Appendix D

INSTRUCTION SUMMARY

Abbreviations used in this summary are as follows:

(1) In general instruction format:

OP	Instruction mnemonics
Disp	Displacement or data
ACr	Accumulator referenced by the instruction
(AC3)	Accumulator specifying indexed addressing
[ ]	Square brackets enclose optional field

(2) Under instruction description:

EA	Effective address
(EA)	Contents of memory byte addressed directly by EA
((EA))	Contents of memory byte addressed indirectly by EA
(ACr)	Contents of any accumulator
(AC0)	Contents of Accumulator AC0
(PC)	Contents of two byte program counter
(PC0)	Contents of page byte of program counter
(PC1)	Contents of displacement (address within page) byte of Program Counter.
STK	Stack. STK(0) to STK(F) refer to individual bytes of the stack.
sr	Source accumulator
dr	Destination accumulator
OV	Overflow status bit set on carry out of bit 7 but not into bit 7.
CY	Carry status bit set on carry out of bit 7.
←	Value to left of arrow replaced by value to right of arrow.
(sr)	Contents of source accumulator
(dr)	Contents of destination accumulator
LSB	Least significant bit (bit 0)
MSB	Most significant bit (bit 7)
L	Link status bit
SEL	Select Flag
CC	Either a label which has been assigned a value between 0 and F or a value in that range.
(ST)	Contents of status byte.
FLAG	Either a label which has been assigned a value between 0 and 7 or a value in that range.

D.1 GENERAL ACCUMULATOR - MEMORY REFERENCE INSTRUCTIONS

Addressing: Direct, base page, current page, or indexed.

Format:

LABEL: OP ACr, Disp (AC3)

OP	Description
LD	(ACr) ← (EA) Load ACr from memory.
ST	(EA) ← (ACr) Store ACr in memory.
ADD	(ACr) ← (ACr) + (EA) OV, CY Add memory to register.
SKNE	If (ACr) ≠ (EA), (PC) ← (PC) +2 Skip the next two byte instructions if ACr does not equal memory.

D.2 AC0 - MEMORY REFERENCE, INDIRECT INSTRUCTIONS

Addressing: Indirect, base page, or current page.

Format:

LABEL: OP Disp

OP	Description
LD@	(AC0) ← ((EA)) Load AC0 from memory indirect.
ST@	((EA)) ← (AC0) Store AC0 in memory indirect.

D.3 AC0 - MEMORY REFERENCE INSTRUCTIONS

Addressing: Direct, base page, current page, or indexed.

Format:

LABEL: OP Disp [(AC3)]

OP	Description
AND	(AC0) ← (AC0) AND (EA) AND AC0 with memory.
OR	(AC0) ← (AC0) OR (EA) OR AC0 with memory.
SKAZ	If (AC0) AND (EA) = 0, (PC) ← (PC) + 2 Skip the next two byte instructions if AND of AC0 and memory are zero.
SUB	(AC0) ← (AC0) - (EA) OV, CY Two-complement and add the contents of memory to the contents of AC0.

D.4 ACCUMULATOR-TO-ACCUMULATOR INSTRUCTIONS

Format:

LABEL: OP sr, dr

OP	Description
RADD	(dr) ← (sr) + (dr) OV, CY Accumulator-to-accumulator ADD
RXOR	(dr) ← (sr) XOR (dr) Accumulator-to-accumulator EXCLUSIVE-OR
RXCH	(dr) ← (sr) , (sr) ← (dr) Exchange accumulators
RAND	(dr) ← (sr) AND (dr) Accumulator-to-accumulator AND

D.5 IMMEDIATE INSTRUCTIONS

Format:

LABEL: OP ACr, Disp

OP	Description
LI	(ACr) ← Disp Load accumulator immediate.
AISZ	(ACr) ← (ACr) + Disp, if (ACr) = 0, (PC) ← (PC) + 2 Add to accumulator immediate. Skip next two byte instructions if result is zero.

D.6 SINGLE ACCUMULATOR INSTRUCTIONS

Format:

LABEL: OP ACr

OP	Description
XCHRS	(ACr) ← (STK(0)) (STK(0))←(ACr) Exchange accumulator and stack.
PUSH	(STK) ← (ACr) Push accumulator onto stack.
PULL	(ACr) ← (STK) Pull stack into accumulator.
COMP2	(ACr) ← - (ACr) Twos-complement accumulator.
SHL	Shift (ACr) left one bit. Load LSB with zero. If SEL is set, (L) ← (MSB).
SHR	Shift (ACr) right one bit. If SEL is reset, load MSB with zero. If SEL is set, (MSB)←(L) and load L with zero.
ROL	Rotate (ACr) left one bit. If SEL is set, rotate through L; that is, (LSB) ← (L) and (L) ← (MSB).
ROR	Rotate (ACr) right one bit. If SEL is set, rotate through L; that is, (MSB)←(L) and (L)←(LSB).

D.7 JUMP AND RETURN INSTRUCTIONS

(A) Jump Direct (JMP, JSR)

Addressing: Direct or indexed, base or current page.

Format:

LABEL: OP Disp (AC3)

(B) Jump Indirect (JMP@, JSR@)

Addressing: Indirect, base or current page.

Format:

LABEL: OP Disp

(C) Return (RTI, RTS)

Addressing: Indirect via stack STK(0), STK(1)

Format:

LABEL: OP

OP	Description
JMP	(PC) ← EA Jump direct.
JSR	(STK(1))← (PC1) (STK(0))← (PC0) (PC) ← EA Jump to subroutine direct and push return address onto stack.
JMP@	(PC) ← (EA) Jump indirect.
JSR@	(STK(1)) ← (PC1) (STK(0)) ← (PC0) (PC) ← (EA) Jump to subroutine indirect and push return address onto stack.
RTI	(PC0) ← (STK(0)) (PC1) ← (STK(1)) set INEN. Return from interrupt and enable interrupts.
RTS	(PC0) ← (STK(0)) (PC1) ← (STK(1)) Return from subroutine.

D.8 MEMORY MODIFY - SKIP TEST INSTRUCTIONS

Addressing: Direct or indexed, base or current page.

Format:

LABEL: OP Disp (AC3)

OP	Description
ISZ	(EA)←(EA) + 1 if (EA) = 0, (PC)←(PC) + 2 Increment memory and skip if result is zero.
DSZ	(EA)←(EA) - 1 if (EA) = 0, (PC)←(PC) + 2 Decrement memory and skip if result is zero.

D. CONTROL AND STATUS FLAG INSTRUCTIONS

(A) Branch-On-Condition (BOC)

Addressing: Program Counter relative, current page direct.

Format:

LABEL: OP CC, Disp

(B) Control Flag (SFLG, PFLG)

Format:

LABEL: OP FLAG

(C) Status Flag (PUSHF, PULLF)

Format:

LABEL: OP

OP	Description
BOC	If jump condition (CC) is true, branch direct (see table D-1).
SFLG	Set control flag specified by FLAG (see table D-2).
PFLG	Pulse flag specified by FLAG (see table D-2)
PUSHF	(STK(0)) ← (ST) Push status byte onto stack.
PULLF	(ST) ← (STK(0)) Pull stack onto status byte.

D.10 HALT INSTRUCTION

Format:

LABEL: HALT

OP	Description
HALT	Halt until START Switch is pressed and released.

Table D-1. IMP-8 Assignment of Inputs to Conditional Jump Multiplexer
(* indicates input required by microprogram)

CC	Input line to CJ MUX	Condition tested (branch occurs if condition is true).
0	* INTR	Interrupt (not normally tested by programmer)
1	* NREQ0	(AC0) $\neq$ 0
2	* REQ0	(AC0) = 0
3	DATA(7) or LNK	Bit 7 of (AC0) a '1' (sign negative)
4	* DATA(1)	Bit 1 of (AC0) a '1'
5	CYOV	Carry or Overflow Flag; if SEL is set, overflow is tested; otherwise carry is tested.
6	STKFUL	Stack Full
7	* START	Start Switch closed
8	* DATA(0)	Bit 0 of (AC0) a '1' (odd)
9	INEN	Interrupt Enable
10	DATA(2)	Bit 2 of (AC0) a '1'
11	DATA(3)	Bit 3 of (AC0) a '1'
12	DATA(4)	Bit 4 of (AC0) a '1'
13	DATA(5) or CY	Bit 5 of (AC0) a '1'
14	DATA(6) or OV	Bit 6 of (AC0) a '1'
15	USERJC	User Jump Condition

Table D-2. IMP-8 Flag Flip-flop Assignments

FC	Flag Output	FC	Flag Output
0	(used by Microprogram)	5	(used by Microprogram)
1	USER 1 -- User Flag 1	6	SEL
2	USER 2 -- User Flag 2	7	INEN -- Interrupt Enable
3	USER 3 -- User Flag 3		
4	USER 4 -- User Flag 4		

Appendix E

INSTRUCTION MACHINE CODE SUMMARY

This appendix provides the instruction byte hexadecimal code for all implemented IMP-8 instructions. Instructions are grouped by format as given in appendix D.

NOTE

AC2 indexing is recognized by the IMP-8C hardware but is not accepted by the IMP-8 Assembler.

The following abbreviations are used:

a base page direct addressing
b current page direct addressing
c (AC2) indexed addressing
d (AC3) indexed addressing
A0, A1, A2, A3 ACr = AC0, AC1, AC2 or AC3
S0, S1, S2 sr = AC0, AC1, AC2 or AC3
D0, D1, D2, D3 dr = AC0, AC1, AC2 or AC3

OP	A0				A1				A2				A3			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
ADD	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
LD	80	81	82	83	84	85	86	87	88	90	8A	8B	8C	8D	8E	8F
SKNE	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
ST	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F

OP	a	b	c	d
AND	C0	C1	C2	C3
DSZ	F4	F5	F6	F7
ISZ	E4	E5	D6	D7
JMP	C4	C5	C6	C7
JMP	04	05	--	--
JSR	D4	D5	D6	D7
JSR	0C	0D	--	--
LD	24	25	--	--
OR	D0	D1	D2	D3
SKAZ	E0	E1	E2	E3
ST	34	35	--	--
SUB	F0	F1	F2	F3

OP	A0	A1	A2	A3
ASIZ	30	31	32	33
COMP2	DC	DD	DE	DF
LI	20	21	22	23
PULL	D8	D9	DA	DB
PUSH	CC	CD	CE	CF
ROL	F8	F9	FA	FB
ROR	FC	FD	FE	FF
SHL	E8	E9	EA	EB
SHR	EC	ED	EE	EF
XCHRS	C8	C9	CA	CB

sr→	A0				A1				A2				A3			
dr→	A0	A1	A2	A3	A0	A1	A2	A3	A0	A1	A2	A3	A0	A1	A2	A3
OP↓																
RADD	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
RAND	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
RXCH	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
RXOR	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

OP	
BOC	10 to 1F, where second digit is condition indicator.
HALT	00
PFLG	38 to 3F, where second digit is (Flag + 8)
PULLF	0E
PUSHF	07
RTI	01
RTS	02
SFLG	28 to 2F, where second digit is (Flag +8)

Appendix F

INSTRUCTION EXECUTION TIMES

	Read (R)	Write (W)	Total (N)
RADD, RXOR; RAND PUSH, PULL, COMP2 SHL, SHR, ROL, ROR SFLG, PFLG	1	-	3
PULLF, PUSHF, RTS	1	-	4
RXCH, XCHRS, RTI	1	-	5
LI	2	-	5
AISZ	2	-	6, 9 if skip
BOC	2	-	7, 6 if branch occurs
JMP	2	-	10, 11 if base page
JSR	2	-	14, 15 if base page
JMP@	4	-	12
JSR@	4	-	18
LD, ADD, AND, OR, SUB	3	-	8, 10 if indexed
ST	3	1	10, 12 if indexed
SKNE, SKAZ	3	-	9, 11 if indexed (+3 if skip)
LD@	5	-	12
ST@	5	1	13
ISZ, DSZ	3	1	11, 13 if indexed (+3 if skip)
HALT	1	-	

The table defines the execution time in terms of the total number of microprogram cycles (N), the number of main memory write cycles (W), and the number of main memory read cycles (R) required. For the standard IMP-8C, the values are: 1.4 microseconds for microprogram cycle time, 0.56 microsecond of delay per main memory read cycle, and a similar delay for each main memory write cycle. Total execution time may be calculated using the formula $T = 1.4N + 0.56\,W + 0.56R$, where N, W, and R are obtained from the table.

A range of values for N is shown in the table. This results from differences in the number of microprogram cycles executed depending upon: (1) the mode of indexing for memory reference instructions, and (2) whether or not a conditional skip test is successful.

Appendix G

CHARACTER SETS

Table G-1 contains the 7-bit hexadecimal code for each character in the ANSI character set. Table G-2 contains the legend for nonprintable characters.

Table G-1. ANSI Character Set in Hexadecimal Representation

Character	7-bit Hexadecimal Number	Character	7-bit Hexadecimal Number	Character	7-bit Hexadecimal Number
NUL	00	+	2B	V	56
SOH	01	,	2C	W	57
STX	02	-	2D	X	58
ETX	03	.	2E	Y	59
EOT	04	/	2F	Z	5A
ENQ	05	0	30	[	5B
ACK	06	1	31	\	5C
BEL	07	2	32	]	5D
BS	08	3	33	↑	5E
HT	09	4	34	←	5F
LF	0A	5	35	`	60
VT	0B	6	36	a	61
FF	0C	7	37	b	62
CR	0D	8	38	c	63
SO	0E	9	39	d	64
SI	0F	:	3A	e	65
DLE	10	;	3B	f	66
DC1	11	<	3C	g	67
DC2	12	=	3D	h	68
DC3	13	>	3E	i	69
DC4	14	?	3F	j	6A
NAK	15	@	40	k	6B
SYN	16	A	41	l	6C
ETB	17	B	42	m	6D
CAN	18	C	43	n	6E
EM	19	D	44	o	6F
SUB	1A	E	45	p	70
ESC	1B	F	46	q	71
FS	1C	G	47	r	72
GS	1D	H	48	s	73
RS	1E	I	49	t	74
US	1F	J	4A	u	75
SP	20	K	4B	v	76
!	21	L	4C	w	77
"	22	M	4D	x	78
#	23	N	4E	y	79
$	24	O	4F	z	7A
%	25	P	50		7B
&	26	Q	51		7C
'	27	R	52	ALT	7D
(	28	S	53	ESC	7E
)	29	T	54	DEL, RUBOUT	7F
*	2A	U	55		

Table G-2. Legend for Nonprintable Characters

Character	Definition
NUL	NULL
SOH	START OF HEADING; ALSO START OF MESSAGE
STX	START OF TEXT; ALSO EOA, END OF ADDRESS
ETX	END OF TEXT; ALSO EOM, END OF MESSAGE
EOT	END OF TRANSMISSION (END)
ENQ	ENQUIRY (ENQRY); ALSO WRU
ACK	ACKNOWLEDGE. ALSO RU
BEL	RINGS THE BELL
BS	BACKSPACE
HT	HORIZONTAL TAB
LF	LINE FEED OR LINE SPACE (NEW LINE): ADVANCES PAPER TO NEXT LINE BEGINNING OF LINE
VT	VERTICAL TAB (VTAB)
FF	FORM FEED TO TOP OF NEXT PAGE (PAGE)
CR	CARRIAGE RETURN TO
SO	SHIFT OUT
SI	SHIFT IN
DLE	DATA LINK ESCAPE
DC1	DEVICE CONTROL 1
DC2	DEVICE CONTROL 2
DC3	DEVICE CONTROL 3
DC4	DEVICE CONTROL 4
NAK	NEGATIVE ACKNOWLEDGE
SYN	SYNCHRONOUS IDLE (SYNC)
ETB	END OF TRANSMISSION BLOCK
CAN	CANCEL (CANCL)
EM	END OF MEDIUM
SUB	SUBSTITUTE
ESC	ESCAPE. PREFIX
FS	FILE SEPARATOR
GS	GROUP SEPARATOR
RS	RECORD SEPARATOR
US	UNIT SEPARATOR
SP	SPACE

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
(408) 732-5000
TWX: 910-339-9240

National Semiconductor Electronics SDNBHD
Batu Berendam
Free Trade Zone
Malacca, Malaysia
Telephone: 5171
Telex: NSELECT 519 MALACCA (c/o Kuala Lumpur)

National Semiconductor GmbH
D 808 Fuerstenfeldbruck
Industriestrasse 10
West Germany
Telephone: (08141) 1371
Telex: 27649

National Semiconductor (UK) Ltd.
Larkfield Industrial Estates
Greenock, Scotland
Telephone: (0475) 33251
Telex: 778 632

NS Electronics (PTE) Ltd.
No. 1100 Lower Delta Rd.
Singapore 3
Telephone: 630011
Telex: 21402

REGIONAL AND DISTRICT SALES OFFICES

ALABAMA
DIXIE DISTRICT OFFICE
3322 Memorial Parkway, S.W. #67
Huntsville, Alabama 35802
(205) 881-0622
TWX: 810-726-2207

ARIZONA
*ROCKY MOUNTAIN REGIONAL OFFICE
7349 Sixth Avenue
Scottsdale, Arizona 85251
(602) 945-8473
TWX: 910-950-1195

CALIFORNIA
*NORTH-WEST REGIONAL OFFICE
2680 Bayshore Frontage Road, Suite 112
Mountain View, California 94043
(415) 961-4740
TWX: 910-379-6432

NATIONAL SEMICONDUCTOR
*DISTRICT SALES OFFICE
Valley Freeway Center Building
15300 Ventura Boulevard, Suite 305
Sherman Oaks, California 91403
(213) 783-8272
TWX: 910-495-1773

NATIONAL SEMICONDUCTOR
SOUTH-WEST REGIONAL OFFICE
17452 Irvine Boulevard, Suite M
Tustin, California 92680
(714) 832-8113
TWX: 910-595-1523

CONNECTICUT
AREA OFFICE
Commerce Park
Danbury, Connecticut 06810
(203) 744-2350

*DISTRICT SALES OFFICE
25 Sylvan Road South
Westport, Connecticut 06880
(203) 226-6833

FLORIDA
*AREA SALES OFFICE
2721 South Bayshore Drive, Suite 121
Miami, Florida 33133
(305) 446-8309
TWX: 810-848-9725

CARIBBEAN REGIONAL SALES OFFICE
P.O. Box 6335
Clearwater, Florida 33518
(813) 441-3504

ILLINOIS
NATIONAL SEMICONDUCTOR
WEST-CENTRAL REGIONAL OFFICE
800 E. Northwest Highway, Suite 203
Mt. Prospect, Illinois 60056
(312) 394-8040
TWX: 910-689-3346

INDIANA
NATIONAL SEMICONDUCTOR
NORTH-CENTRAL REGIONAL OFFICE
P.O. Box 40073
Indianapolis, Indiana 46240
(317) 255-5822

KANSAS
DISTRICT SALES OFFICE
13201 West 82nd Street
Lenexa, Kansas 66215
(816) 358-8102

MARYLAND
CAPITAL REGIONAL SALES OFFICE
300 Hospital Drive, No. 232
Glen Burnie, Maryland 21061
(301) 760-5220
TWX: 710-861-0519

MASSACHUSETTS
*NORTH-EAST REGIONAL OFFICE
No. 3 New England, Exec. Office Park
Burlington, Massachusetts 01803
(617) 273-1350
TWX: 710-332-0166

MICHIGAN
*DISTRICT SALES OFFICE
23629 Liberty Street
Farmington, Michigan 48024
(313) 477-0400

MINNESOTA
DISTRICT SALES OFFICE
8053 Bloomington Freeway, Suite 101
Minneapolis, Minnesota 55420
(612) 888-3060
Telex: 290766

NEW JERSEY/NEW YORK CITY
MID-ATLANTIC REGIONAL OFFICE
301 Sylvan Avenue
Englewood Cliffs, New Jersey 07632
(201) 871-4410
TWX: 710-991-9734

NEW YORK (UPSTATE)
CAN-AM REGIONAL SALES OFFICE
104 Pickard Drive
Syracuse, New York 13211
(315) 455-5858

OHIO/PENNSYLVANIA/
W. VIRGINIA/KENTUCKY
EAST-CENTRAL REGIONAL OFFICE
Financial South Building
5335 Far Hills, Suite 214
Dayton, Ohio 45429
(513) 434-0097

TEXAS
*SOUTH-CENTRAL REGIONAL OFFICE
5925 Forest Lane, Suite 205
Dallas, Texas 75230
(214) 233-6801
TWX: 910-860-5091

WASHINGTON
DISTRICT OFFICE
300 120th Avenue N.E.
Building 2, Suite 205
Bellevue, Washington 98005
(206) 454-4600

INTERNATIONAL SALES OFFICES

AUSTRALIA
NS ELECTRONICS PTY, LTD.
Cnr. Stud Road & Mountain Highway
Bayswater, Victoria 3153
Australia
Telephone: 729-6333
Telex: 32096

CANADA
*NATIONAL SEMICONDUCTOR CORP.
1111 Finch Avenue West
Downsview, Ontario, Canada
(416) 635-9880
TWX: 610-492-1334

DENMARK
NATIONAL SEMICONDUCTOR
SCANDINAVIA
Vordingborggade 22
2100 Copenhagen
Denmark
Telephone: (01) 92-OBRO-5610
Telex: DK 6827 MAGNA

ENGLAND
NATIONAL SEMICONDUCTOR (UK) LTD.
The Precinct
Broxbourne, Hertfordshire
England
Telephone: Hoddesdon 69571
Telex: 267-204

FRANCE
NATIONAL SEMICONDUCTOR
FRANCE S.A.R.L.
28, Rue de la Redoute
92260-Fontenay-Aux-Roses
Telephone: 660-81-40
TWX: NSF 25956F

HONG KONG
*NATIONAL SEMICONDUCTOR
HONG KONG LTD.
9 Lai Yip Street
Kwun Tung, Kowloon
Hong Kong
Telephone: 3-458888
Telex: HX3866

JAPAN
*NATIONAL SEMICONDUCTOR JAPAN
Nakazawa Building
1-19 Yotsuya, Shinjuku-Ku
Tokyo, Japan 160
Telephone: 03-359-4571
Telex: J 28592

SWEDEN
NATIONAL SEMICONDUCTOR SWEDEN
Sikvagen 17
13500 Tyreso
Stockholm
Sweden
Telephone: (08) 712-04-80

WEST GERMANY
*NATIONAL SEMICONDUCTOR GMBH
8000 Munchen 81
Cosimstrasse 4
Telephone: (0811) 915-027

***Microprocessor System Specialist Available**

D10M44