REFERENCE MANUAL

SPC-12 AUTOMATION COMPUTER

GENERAL AUTOMATION, INC.

SPC-12 INSTRUCTIONS

Mnemonic		Instruction Name	Memory Cycles Required	Ref Man Page
MEMORY ADDRESSING				
LDB	m	Load B-Register	3	3-2
LDB	m,X	Load B-Register Indexed	3	3-2
STB	m	Store B-Register	3	3-3
STB	m,X	Store B-Register Indexed	3	3-3
ELB	m	Extended Load B-Register	4	3-3
JMP	m	Jump Unconditionally	2	3-3
SKIP				
SKS	n	Skip if Link Set	1	3-4
SKR	n	Skip if Link Reset	1	3-4
SKZ	n	Skip if Zero	1	3-4
SKN	n	Skip if Not Zero	1	3-4
SKP	n	Skip if Plus	1	3-5
SKM	n	Skip if Minus	1	3-5
SKT	n	Skip if I/O Test True	1	3-5
SKF	n	Skip if I/O Test False	1	3-5
ARITHMETIC/LOGICAL				
AZE	r	Zero Register	2	3-6
AND	r,B	Logical AND (B with r)	2	3-6
AND	r,v	Logical AND (v with r)	3	3-6
AXR	r,B	Logical Exclusive OR (B with r)	2	3-7
AXR	r,v	Logical Exclusive OR (v with r)	3	3-7
AOR	r,B	Logical OR (B with r)	2	3-7
AOR	r,v	Logical OR (v with r)	3	3-7
ALD	r,v	Load Register (v into r)	3	3-8
ASU	r,B	Subtract (B from r)	2	3-8
ASU	r,v	Subtract (v from r)	3	3-8
AAD	r,B	Add (B to r)	2	3-9
AAD	r,v	Add (v to r)	3	3-9
REGISTER TRANSFER				
RTR	s,d	Register Transfer	2	3-9
RIC	s,d	Register Transfer and Increment	2	3-10
RDC	s,d	Register Transfer and Decrement	2	3-10
RLK	s,d	Register Transfer and Add Link	2	3-10
SHIFT				
SHR	r	Shift Right	2	3-11
SHC	r	Shift Circular	2	3-11
SHL	r	Shift Circular Through Link	2	3-12
SHI	r	Shift Serial Data In	2	3-12
SHIO	r	Shift Serial Data In and Out	2	3-12
SHRO	r	Shift Right and Serial Data Out	2	3-13
SHCO	r	Shift Circular and Serial Data Out	2	3-13
SHLO	r	Shift Circular Through Link and Serial Data Out	2	3-13
CONTROL				
PLR		Pulse Link Reset	2	3-14
PLS		Pulse Link Set	2	3-14
PSA		Pulse Stall Alarm (optional)	2	3-15
TBB		Transfer BB- to B-Register	2	3-15
TBE		Transfer B- to E-Register	2	3-15
FOB		Function-Address Out from B-Register	2	3-15
DOB		Data Out from B-Register	2	3-16
DIB		Data Into B-Register	2	3-16
INE		Interrupt Enable	2 or 3	3-16
AUGMENTED MEMORY ADDRESSING				
GnL	r,t,I	Augmented Load	3 or 5	3-19
GnA	r,t,I	Augmented Add	3 or 5	3-19
GnS	r,t,I	Augmented Store	3 or 5	3-21
GnZ	r,t,I	Augmented Store and Zero	3 or 5	3-21
SERIAL INPUT/OUTPUT				
SHI	r	Shift Serial Data In	2	3-12
SHIO	r	Shift Serial Data In and Out	2	3-12
SHRO	r	Shift Right and Serial Data Out	2	3-13
SHCO	r	Shift Circular and Serial Data Out	2	3-13
SHLO	r	Shift Circular Through Link and Serial Data Out	2	3-13
PARALLEL INPUT/OUTPUT				
SKT	n	Skip if I/O Test True	1	3-5
SKF	n	Skip if I/O Test False	1	3-5
FOB		Function-Address Out from B Register	2	3-15
DOB		Data Out from B-Register	2	3-16
DIB		Data Into B-Register	2	3-16

88A00000A-B

January 1969
$5.00

REFERENCE MANUAL

SPC-12
AUTOMATION COMPUTER

GENERAL AUTOMATION, INC.
Automation Products Division
706 West Katella, Orange, California 92668 (714) 633-1091

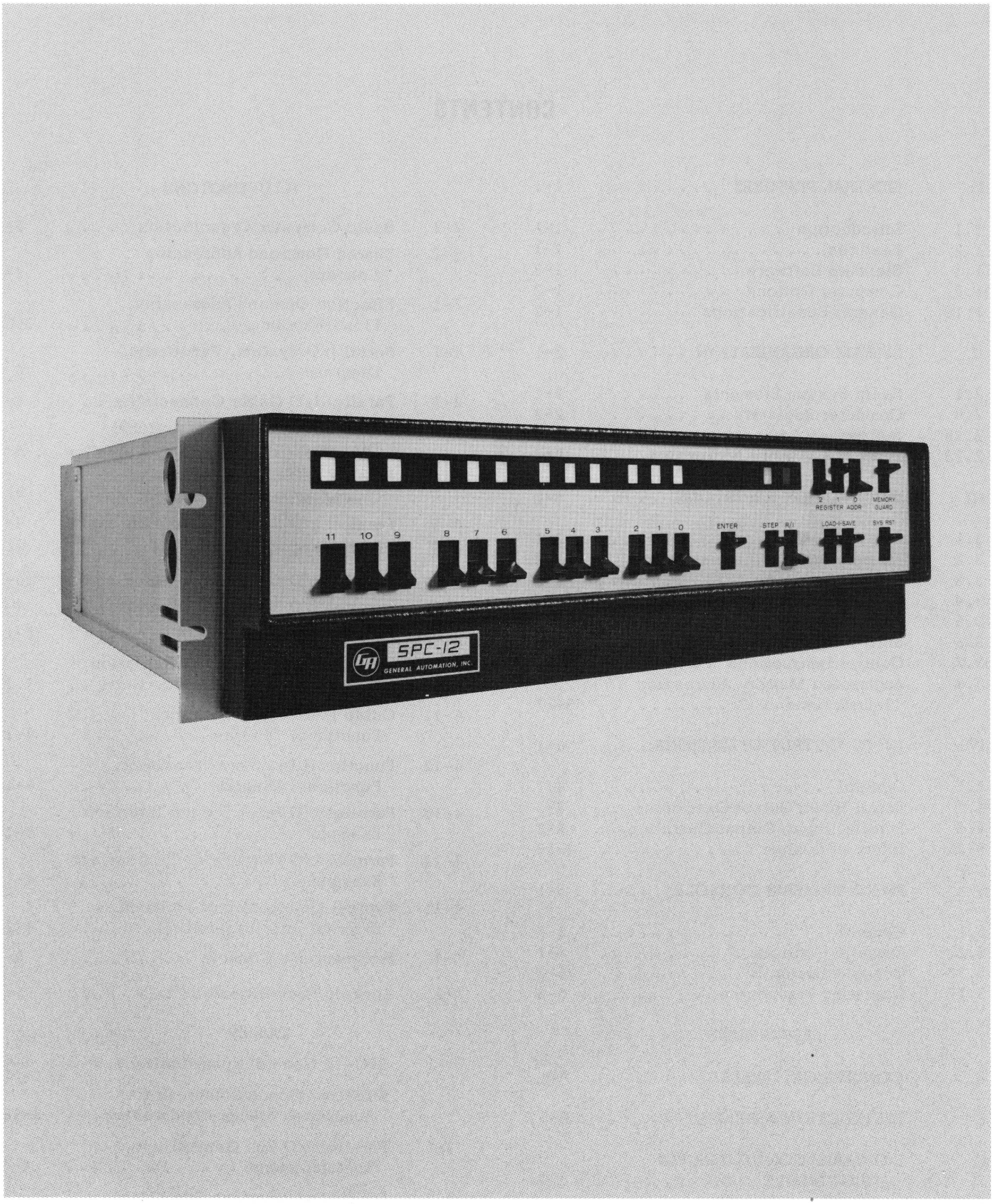
This publication supersedes 88A00000A-A dated October 1968

REVISION

SYMBOL	DESCRIPTION	APPROVED	DATE
A	Original Issue		Oct. 68
B	*Revised per ECO 1201. Pages affected are: Inside Front Cover, Title, ii, iii, 2-1, 2-6, 2-7, 3-1, 3-6 thru 3-17, 3-19, 3-21, 4-2 thru 4-4, 4-7 thru 4-9, 4-11 thru 4-13, 4-16, 4-21, 5-3, C-1 thru C-4, I-2 thru I-4, Programmer's Card, Inside Rear Cover		Jan. 69

*Revised information is indicated by a vertical bar in the margin of the page.

CONTENTS



SPC-12 Automation Computer

SECTION I
GENERAL FEATURES

1.1 INTRODUCTION

The SPC-12 Automation Computer provides a new concept in hardware design and software capability. The result is a computer that meets the most stringent requirements of real-time data collection, processing, and online control.

Careful planning and attention to user requirements have produced in the SPC-12 an automation computer of efficient and flexible design. It offers solutions to the following user needs:

- Space/Weight. Standing only 5-1/4 inches high, it can be installed in any standard 19-inch, rack-mount cabinet. It weighs only 30 pounds including 4096 words of core memory, power supplies and enclosure.
- Work per Byte. A new shared command addressing concept permits savings of up to 30 percent in the amount of memory required for program operation.
- No Addressing Restrictions. Direct addressing of up to 4096 words of core memory is possible.
- Reliability. Worst case design featuring wire-free construction, all integrated circuits, and an operating temperature range of 32° to 122°F permits unattended online operation 24 hours a day. Optional power failure detection and automatic restart features, together with a relative time clock, guarantee confidence for real-time, online control applications.
- Speed. The computer has a full memory cycle time of 2.16 μs.
- Low Cost. Engineering design with dedicated applications in mind permits low cost without sacrificing efficiency, capability, or flexibility.
- Maintainability. Mean time to repair (MTTR) of less than 5 minutes. Mean time between failure (MTBF) of over 10,000 hours.
- Powerful Instruction Repertoire. Fifty-one basic instructions including 6 memory addressing instructions, 8 skip instructions, 12 arithmetic/logical instructions, 4 register transfer instructions, 8 shift instructions, 9 control instructions, and 4 augmented memory addressing instructions. Addressing features include direct addressing, indirect addressing, literal addressing, preindexed and postindexed addressing, and automatic index incrementing.
- Operational Security. Total operating security is provided by a keylock switch that disables all console switches.
- Ruggedized Construction. Permits application of the computer in any industrial environment.

1.2 FEATURES

Basic features of the SPC-12 include:

- All silicon monolithic integrated logic circuits
- Eight-bit memory word length
- 4096-word memory, expandable to 16,384 words maximum
- 2.16 μs full memory cycle time
- Fully parallel operation
- Nine hardware registers
- One accumulator register
- Three accumulator/index registers
- Unique shared command addressing
- Relative time clock
- Priority interrupt system
- Independent serial and parallel input/output systems
- Programmer's console
- Full 12-bit organization

Optional features of the SPC-12 include:

- Power failure detection and automatic restart
- Stall alarm
- Read-only memory
- Additional 4096-word memory modules
- Memory expansion cabinet (required for core memory in excess of 8192 words)
- I/O peripheral controller expansion chassis (required with I/O peripheral controller options)
- Direct memory transfer I/O channel (permits direct communication between computer memory and external I/O devices)
- Eight additional interrupt lines with selective arm/disarm capability
- 24V battery power source

1.3 STANDARD SOFTWARE

Standard program systems provided with the computer comprise the Conversational Assembly System, the Basic Utility System, and a subroutine library.

1.4 CONVERSATIONAL ASSEMBLY SYSTEM

The Conversational Assembly System (CAS) is a symbolic assembly program that minimizes the time required for assembling a program. It permits the programmer to recover from errors online without having to restart the assembly process. With CAS, the programmer can insert corrections from the keyboard during assembly. The assembler also provides the programmer with a means of entering linkage, mapping, and common data. With CAS, instructions, data, memory addresses, and address modifiers can be coded and entered in symbolic notation.

The assembler provides 11 pseudo operation codes. A single pass of the input source program is sufficient to completely assemble the program.

1.5 BASIC UTILITY SYSTEM

The Basic Utility System (BUS) enables the programmer to trace through his program to correct errors, to enter and execute test cases, and to verify results. The system operates online with a teletypewriter. Data can be input, instructions can be changed, and small programs can be executed using the keyboard for input.

1.6 SUBROUTINE LIBRARY

The subroutine library includes a package of often used utility programs, I/O programs, mathematical subroutines, and a hardware maintenance and verification subsystem. The utility package contains routines for loading, punching, and listing programs as well as aids for debugging and updating programs. Input/output routines are available for such peripheral equipment as the Teletype Model 33 and Model 35 Send/Receive Sets, standard communication modem interfaces, discrete digital inputs and outputs, contact closure and sensing units, analog-to-digital and digital-to-analog conversion subsystems, and other standard peripheral devices.

Also provided is a library of mathematical subroutines including multiple precision fixed-point addition, subtraction, multiplication, and division routines.

1.7 COMPUTER OPTIONS

1.8 PERIPHERAL DEVICES

A complete line of peripheral devices is available for use with the computer. These include paper tape equipment, card equipment, printers, and mass storage devices.

1.9 SYSTEM INTERFACE UNITS

A complete line of system interface units are available as options for special system applications. These include analog-to-digital, digital-to-analog, and multiplexer units.

1.10 GENERAL SPECIFICATIONS

Table 1-1 lists the general specifications for the computer.

Table 1-1. SPC-12 General Specifications

Characteristic	Specification
Type	General purpose automation computer with magnetic core memory, parallel binary arithmetic capability, single-word addressing, serial and parallel input/output capability, and common shared memory for storage of inline instructions
Memory	Random access magnetic core, 2.16 μs full cycle time, 8-bit words, standard 4096 words minimum expandable in 4096 word modules to 16,384 words maximum
Arithmetic	Parallel, binary, fixed point, two's complement
Types of Addressing	Direct addressing, indirect addressing, literal addressing, preindexing and postindexing, automatic index incrementing
Instruction Types	Single word, double word, double word shared, double word immediate
Instruction Groups	Memory addressing, skip, arithmetic/logical, register transfer, shift, control, augumented memory addressing
Speed	Typical instruction execution times: Load/store from memory 6.48 μs Add/subtract registers 4.32 μs Add/subtract memory 6.48 μs Output from B-register 4.32 μs Input to B-register 4.32 μs
Input/Output	Serial input/output channel Parallel input/output channels Automatic priority interrupts Relative time clock Direct memory transfer (optional) Power failure detection/restart (optional)
Size	5-1/4 inches high, 17-3/5 inches wide, 23 inches deep including enclosure, power supply, and cooling
Weight	30 pounds, including enclosure, power supply, and cooling
Power	115 (±11.5)V ac, single phase, 60 (-13, +3) Hz
Environment	0° to 50°C (32° to 122°F), 10 to 90 percent relative humidity
Site Preparation	No special wiring, subflooring, air conditioning, or other installation preparation is required

SECTION II
SYSTEM ORGANIZATION

2.1 BASIC SYSTEM ELEMENTS

The SPC-12 is an extremely powerful general purpose, stored program, automation computer. By means of a unique bus structure, a high-speed memory, a versatile arithmetic and control unit, six programmable 12-bit registers, and a flexible input/output system the computer performs a large number of data handling and control operations with ease. A block diagram of the basic system elements is shown in figure 2-1.

2.2 TRANSFER BUSES

Three buses provide the main communication paths within the computer:

a. The addend bus (12-bits)

b. The augend bus (12-bits)

c. The data bus (12-bits)

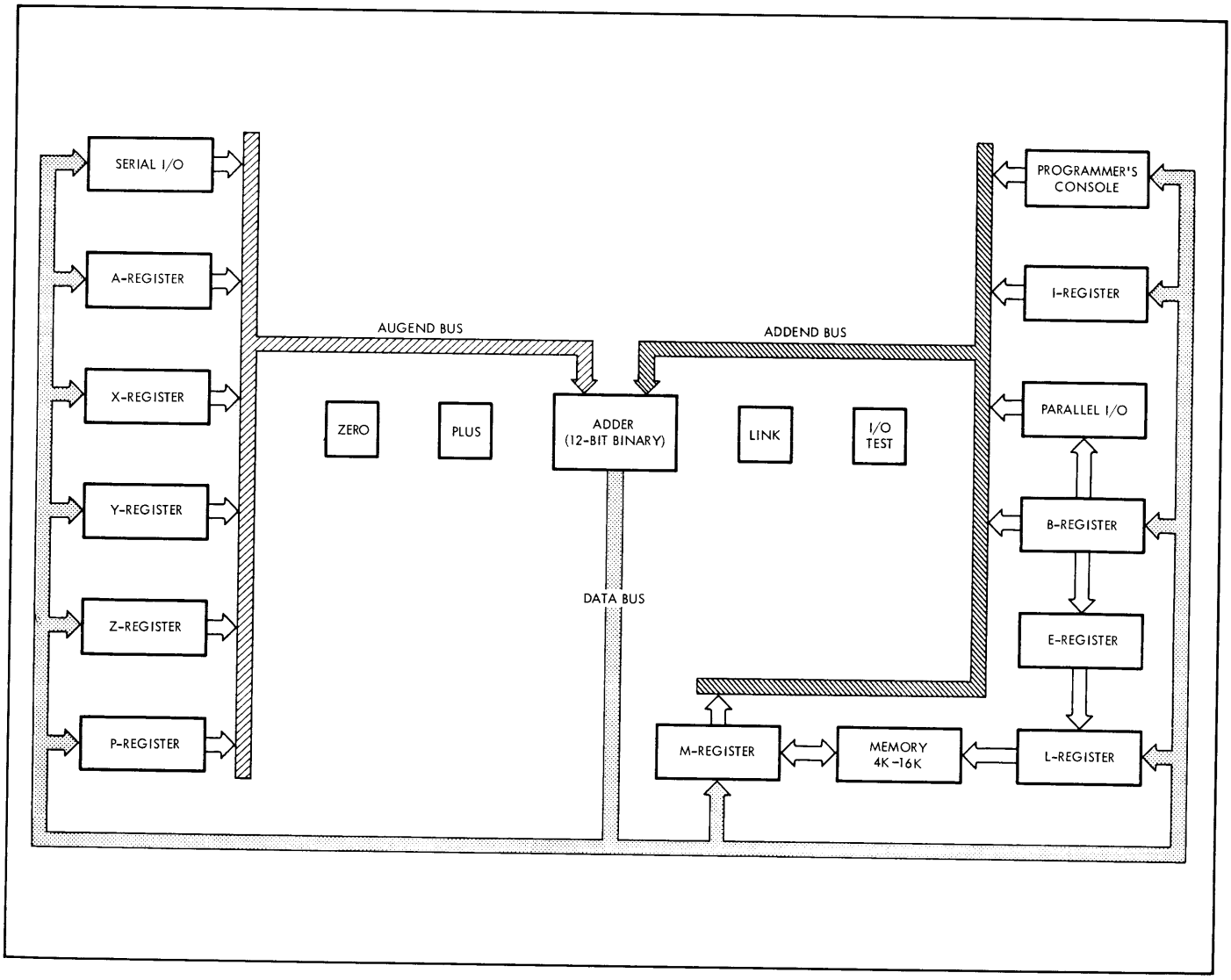


Figure 2-1. Basic Computer Organization

These buses allow a flow of information within the computer that is both efficient and uncomplicated. The addend and augend buses are the paths by which all data enters the adder. The data bus is the path by which data is transferred from the adder to other elements of the system.

2.3 MEMORY

The computer memory system includes a 4096-word, random access core memory, the memory location register (L-register), and the memory data register (M-register). Ferrite magnetic cores are used as the storage elements in the core memory, which has a memory cycle time of 2.16 μs. Each memory location provides storage for an 8-bit word. The basic core memory can be expanded from 4096 words to 16,384 words by adding optional memory modules in 4096-word increments. The basic computer enclosure accommodates up to 8192 words of memory. Expansion to 16,384 words is conveniently accomplished by adding an additional satellite enclosure.

2.4 ARITHMETIC/LOGIC

The arithmetic/logic section of the computer includes the adder and the Zero, Plus, and Link indicators. The adder is the central element of the computer. It is used in executing every instruction. Through the adder pass the results of all load, store, jump, data transfer, arithmetic, logical, shift, and input/output operations; and through the adder passes each instruction on its way from memory to the instruction register.

The three inputs to the adder are the addend bus, the augend bus, and the carry (increment) input to the least significant bit (bit 0). The output of the adder is the data bus. Results of various adder operations are recorded in the Zero, Plus, and Link indicators, which can be tested by skip instructions.

Two important concepts are basic to a thorough understanding of the operation of the computer:

a. When an instruction involving two operands is executed, one of the operands is always contained in one of the registers connected to the augend bus; the other, in one of the registers connected to the addend bus.

b. Information transferred to any of the registers must first pass through the adder.

2.5 CONTROL

The control section of the computer includes the programmer's console, the program sequencing register (P-register), and the instruction register (I-register). The programmer's console contains controls and displays for manually operating the computer. The P- and I-registers are used to control automatic program sequencing and execution.

2.6 OPERATIONAL REGISTERS

Six basic operational registers are available to the programmer:

a. The accumulator register (A-register)

b. Three accumulator/index registers (X-, Y-, and Z-registers)

c. The buffer register (B-register)

d. The P-register

The operational registers are described in detail in paragraph 2.9.

2.7 INPUT/OUTPUT

Two independent I/O systems are included as part of the basic computer: the serial I/O system and the parallel I/O system.

The serial I/O system interfaces with the augend and data buses of the computer, and is normally used with Teletype I/O devices. Only two basic instructions are required to transfer information between the serial I/O system and an I/O device.

The parallel I/O system interfaces with the addend bus and the B-register, and permits efficient handling of 8-, 12-, 16-, or 24-bit data transfers between the computer and I/O devices. The I/O Test indicator operates with the parallel I/O system to provide the facility for testing various status conditions of the I/O devices connected to the I/O bus. Only three basic instructions are required to perform parallel I/O operations.

A detailed description of the operation of the serial and parallel I/O systems is given in section IV.

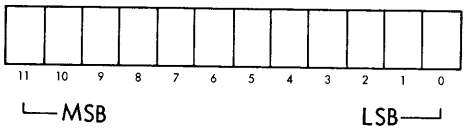
2.8 COMPUTER REGISTERS

The computer contains nine standard hardware registers, six of which can be manipulated under program control. In addition to these, the computer contains four single-bit registers (indicators) that can be altered or tested under program control.

2.9 PROGRAMMABLE REGISTERS

2.10 FULL REGISTERS

The six programmable registers are full 12-bit registers. The bit positions in each are numbered 0 through 11 as shown in the following diagram:



The programmable 12-bit registers are used as follows:

a. P-Register. Used as the program sequence register. It contains the address of the instruction currently being executed, and automatically advances to fetch the next instruction from memory. It can be altered during the execution of jump, skip, register transfer, and arithmetic instructions.

b. B-Register. Used as a buffer for memory accesses and parallel I/O operations. It holds one of the operands during the execution of most instructions that operate on two operands. It also holds the effective operand address during the execution of those instructions that are indexable and indirectly addressable.

c. A-Register. Holds the results of arithmetic and logical operations. It can be used as a 12-bit accumulator for address generation or as an 8-bit accumulator for arithmetic or information processing.

d. X-Register. Used as an index register to modify the operand address designated by an instruction or as an alternate accumulator.

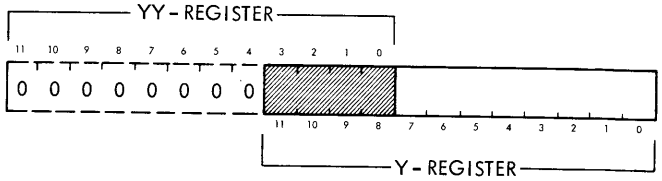
e. Y-Register. Used as an index register or alternate accumulator.

f. Z-Register. Used as an index register or alternate accumulator.

2.11 PARTIAL REGISTERS

Of the six programmable registers, portions of the Y-, Z-, and B-registers can be addressed and used as registers. These, plus the E-register, are designated as partial registers and are used as follows:

a. YY-Register. This register is actually the four most significant bits of the Y-register presented in a 12-bit form as shown in the following diagram:



YY_4 THROUGH YY_{11} ALWAYS ZERO

In the YY-register, only bits 0 through 3 are significant; bits 4 through 11 are always zeros. Although the eight most significant bits are always zeros, the programmer must consider it as a full 12-bit register whenever it is used in arithmetic, logical, transfer, or shift operations. Operations with the YY-register do not affect bits 0 through 7 of the Y-register. On the other hand, operations with the full 12-bit Y-register do affect the contents of the YY-register.

b. ZZ-Register. This register is similar to the YY-register except that it is the four most significant bits of the Z-register.

c. BB-Register. This register is similar to the YY-register except that it is the four most significant bits of the B-register.

d. E-Register. Used as the extended memory address register. It operates as an extension of the L-register to provide memory addressing capability to 16,384 words. (See paragraph 2.25 for a description of extended memory addressing.)

Standard number designations are assigned to the programmable registers as follows:

Octal Code	Register
0	A
1	X
2	Y
3	Z
4	P
5	B
6	YY
7	ZZ

These standard designations are used in the arithmetic/logical, register transfer, and shift instructions to specify the selected register. They are also used to address a specific register from the programmer's console.

2.12 SINGLE-BIT REGISTERS (INDICATORS)

The four single-bit registers are used to indicate the results of specific computer operations as follows:

a. Plus Indicator. Set to indicate PLUS if the sign of the result of the last adder operation was positive (bit 7 of the result was zero). It is set to indicate MINUS if the sign of the last adder operation was negative (bit 7 of the result was one). The Plus indicator can be tested for either state by the appropriate skip instruction.

b. Zero Indicator. Set to indicate ZERO if bits 0 through 7 of the result of the last adder operation were all zeros. It is set to indicate NOT ZERO if bits 0 through 7 of the result of the last adder operation were not all zeros. The Zero indicator can be tested for either state by the appropriate skip instruction.

c. Link Indicator. Set to 1 if there is a carry out of bit position 7 of the adder when any of the following instructions are performed:

1. Add
2. Subtract
3. Register Transfer and Increment
4. Register Transfer and Decrement
5. Register Transfer and Add Link

It is set to 0 if there is no carry when these instructions are performed. During the execution of any shift instruction the Link indicator is always set equal to the value of the bit shifted out of bit position 0 of the selected register. The Link indicator can be tested for either state by the appropriate skip instruction.

d. I/O Test Indicator. Records the status of the I/O test line (TEST-I) during an I/O test operation (paragraph 4.15). The I/O Test indicator can be tested for either the TRUE or FALSE state by the appropriate skip instruction.

2.13 NONPROGRAMMABLE REGISTERS

The computer contains three registers that are involved with and affected by the execution of each instruction but that cannot be directly manipulated by the programmer. These are designated as nonprogrammable registers and are used as follows:

a. L-Register. Used as 12-bit memory location (address) register. It specifies the location in memory from which or to which an instruction, an address, or data is to be read or stored.

b. M-Register. Used as an 8-bit memory data register to transfer information in and out of memory.

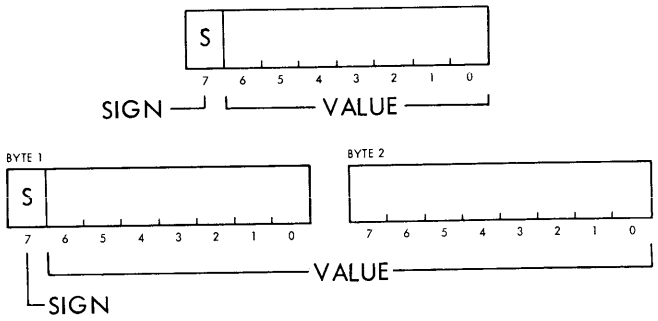
c. I-Register. Used as a 12-bit instruction register. It contains the instruction currently being executed.

2.14 WORD FORMATS

Three basic types of word formats are used in the computer: data words, address words, and instruction words.

2.15 DATA WORD FORMATS

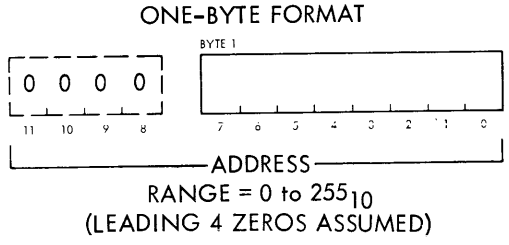
Data is stored in the computer in the form of eight-bit words, or bytes, as follows:



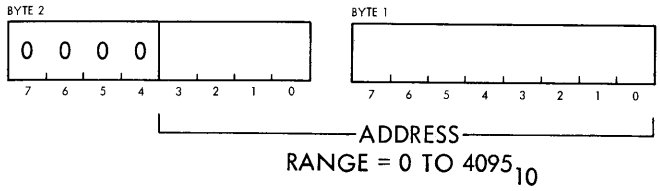
In the one-byte format, the data occupies bit positions 0 through 6 with the sign in bit position 7. When the data is contained in more than one byte, only the most significant byte contains the sign bit. All data in the computer is expressed in two's complement form; therefore, a sign bit of 0 represents a positive number, and a sign bit of 1 represents a negative number.

2.16 ADDRESS WORD FORMATS

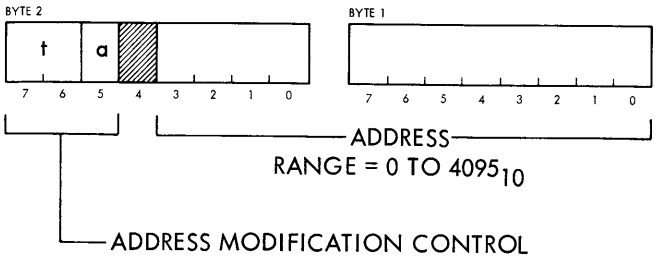
Addresses are always considered as unsigned 12-bit numbers with the following formats:



TWO-BYTE FORMAT



INDIRECT ADDRESS FORMAT

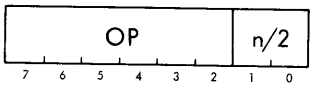


In the one-byte format, zeros are assumed for the four most significant bits, and the address value is contained in bits 0 through 7. In the two-byte format, zeros are contained in bits 4 through 7 of byte 2, and the address value is contained in bits 0 through 7 (least significant bits) of byte 1 and bits 0 through 3 (most significant bits) of byte 2. The indirect address format is used for indirect addressing (paragraph 2.22). This format is similar to the two-byte format except that bits 5 through 7 of byte 2 are used for address modification (post-indexing and autoincrementing.

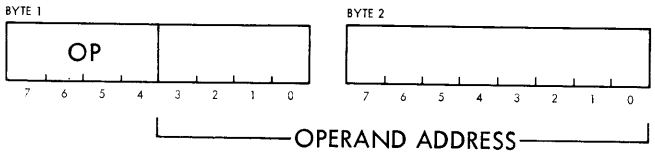
2.17 INSTRUCTION WORD FORMATS

Three different instruction formats are used in the computer as follows:

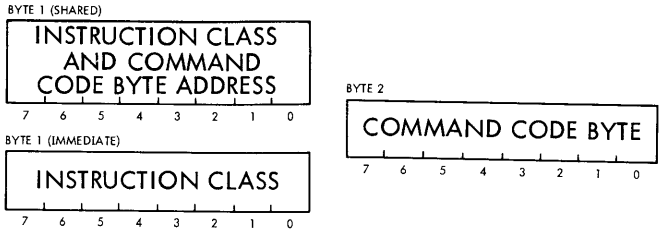
SKIP INSTRUCTIONS



MEMORY ADDRESSING INSTRUCTIONS



COMMAND CODE ADDRESSING INSTRUCTIONS



Skip instructions are nonaddressing instructions with a single-byte format. Bits 2 through 7 of the instruction define the particular transfer operation that is to be performed by the instruction. Skip instructions are described in paragraph 3.3.

Memory addressing instructions have the capability to directly address 4096 words of memory. These instructions use a two-byte format, with bits 0 through 7 of byte 2 and bits 0 through 3 of byte 1 designating the operand address, and bits 4 through 7 of byte 1 designating the particular operation to be performed. Memory addressing instructions are described in paragraph 3.2.

Command code addressing instructions include the arithmetic/logical, register transfer, shift, control, and augmented memory addressing instructions. These instructions use a two-byte format with the second byte of the instruction (designated the command code byte) being stored either in the memory location immediately following the first byte or in a shared memory pool (see paragraph 2.19). Byte 1 of the instruction identifies the instruction class and, if the second byte is stored in a shared memory pool, contains the location of the byte in the pool. Command code addressing instructions are described in paragraphs 3.4 through 3.9.

Before an instruction can be executed, it must be fetched from memory and assembled in the I-register. The methods of assembling the various instructions are shown in appendix B.

2.18 INSTRUCTION ADDRESSING

Two basic types of instruction addressing are used in the computer: command code addressing and operand addressing. An intimate knowledge of both types of addressing is required to understand how the various instructions are formed and executed.

2.19 COMMAND CODE ADDRESSING

Command code addressing is used with the following groups of instructions:

a. Arithmetic/logical

b. Register transfer

c. Shift

d. Control

e. Augmented memory addressing

Command code addressing permits the first byte of these two-byte instructions to specify the location from which the second byte (command code byte) is fetched. Normally, the command code byte is stored in the memory location immediately following the first byte. This is known as immediate command addressing. To minimize the number of storage locations required by a program, the SPC-12 employs a new technique known as shared command addressing. The shared command feature permits the command code byte to be stored in a common area of memory rather than being included as part of the inline coding of the program. With shared command addressing the command code byte can be used (shared) by more than one instruction. This sharing of command code bytes results in a considerable saving of the number of memory locations used with a given program.

Assigned to the command code addressing instructions are four blocks, or pools, of memory locations for storing the shared command code bytes. Each block contains 16 memory locations. The assignment of each block to a particular instruction group is fixed. The blocks cannot be reassigned to different instruction groups, nor can any shared command instruction group use any other portion of memory for storing shared command bytes. The addresses of the memory locations and the instruction groups to which they are assigned are as follows:

Octal Address	Instruction Group
0020 through 0037	Arithmetic/logical
0040 through 0057	Register transfer
0060 through 0077	Shift and control
0100 through 0117	Augmented memory addressing

The format of byte 1 of the command code addressing instructions specifies either immediate command addressing or shared command addressing. If immediate command addressing is specified, the second byte is taken from the next memory location, not from the shared memory pool. If shared command addressing is specified, the first byte specifies the relative location in the shared memory pool in which the second byte is stored. Figure 2-2 illustrates the concept of shared command addressing.

For a complete description of the command code addressing instructions, see paragraphs 3.4 through 3.9.

2.20 OPERAND ADDRESSING

Several variations of operand addressing are used in the computer. These variations include direct addressing, indirect addressing, indexed addressing, and literal addressing.

2.21 DIRECT ADDRESSING

Direct addressing is a feature of both the memory addressing instructions and the augmented memory addressing instructions. The six memory addressing instructions can directly address any of 4096 memory locations. The augmented memory addressing instructions can directly address four memory locations (0000_8, 0002_8, 0004_8, and 0006_8). These four locations can be used to store indirect address information or, with indexing, can be used as base addresses to point to indirect address information that can be stored anywhere in memory. For a complete description of augmented memory addressing, see paragraph 3.8.

2.22 INDIRECT ADDRESSING

Indirect addressing is a feature of the augmented memory addressing instructions. This feature, along with indexed addressing both before and after indirect addressing, gives the augmented memory addressing instructions an extremely powerful addressing capability. A detailed description of augmented memory addressing is given in paragraph 3.8.

2.23 INDEXED ADDRESSING

Indexed addressing is a feature of the Load B-Register and Store B-Register instructions of the

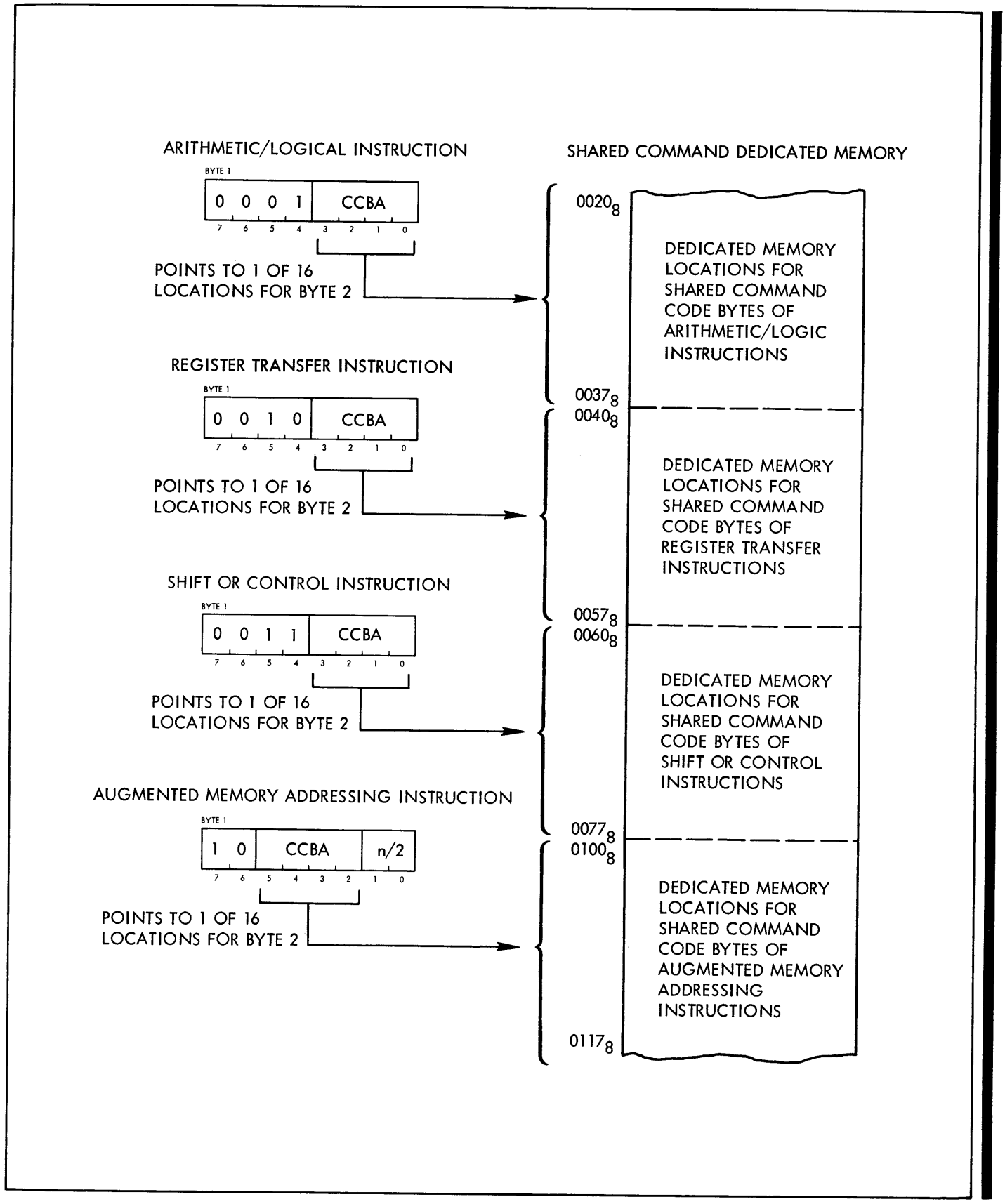


Figure 2-2. Shared Command Addressing Concept

memory addressing group, and of all instructions of the augmented memory addressing group. The Load B-Register and Store B-Register instructions may select index address modification by the X-register. The augmented memory addressing instructions may select index address modification by either the X-, Y-, or Z-register.

When indexing is specified, the contents of the selected index register are added to the operand address of the instruction to produce the effective operand address.

With the augmented memory addressing instructions, indexing may be selected to occur before indirect addressing (preindexing), after indirect addressing (postindexing), or both before and after indirect addressing. For a detailed description of augmented memory addressing, see paragraph 3.8.

2.24 LITERAL ADDRESSING

Literal addressing is a feature of the arithmetic/logical instructions. This feature provides the capability of using as the operand the literal value stored in the memory location immediately following the location of the instruction. A bit in the command code byte of the instruction specifies whether literal addressing is in effect. For a description of how literal addressing is used with the arithmetic/logical instructions, see paragraph 3.4.

2.25 EXTENDED MEMORY ADDRESSING

When more than 4096 words of memory are used in the computer, addressing of memory locations outside the basic 4096 locations is accomplished by using the program-controlled E-register. At any one time the operating memory of the computer consists of 4096 words made up of two 2048-word blocks. The first block is always memory locations 0000_8 through 3777_8; the second block can be of any one of the remaining 2048-word blocks. The E-register specifies which two blocks are being used as follows:

E-Register	Effective Memory
000	Block 0
001	Blocks 0 and 1
010	Blocks 0 and 2
011	Blocks 0 and 3
100	Blocks 0 and 4
101	Blocks 0 and 5
110	Blocks 0 and 6
111	Blocks 0 and 7

Whenever the computer is initialized (SYS RST switch on programmer's console pressed), the contents of the E-register are automatically set to 001. This permits the programmer to address the first 4096 words of memory and also permits programs written for only 4096 words of memory to be run without further concern. When the programmer wishes to work in an extended memory area, he must change the contents of the E-register accordingly.

The contents of the E-register are displayed on the programmer's console by register display lamps 9, 10, and 11 when either the YY- or ZZ-register (6_8 or 7_8) is selected for display.

SECTION III
INSTRUCTION REPERTOIRE

3.1 GENERAL

This section describes all SPC-12 instructions grouped into the following functional classes:

a. Memory Addressing

b. Skip

c. Arithmetic/Logical

d. Register Transfer

e. Shift

f. Control

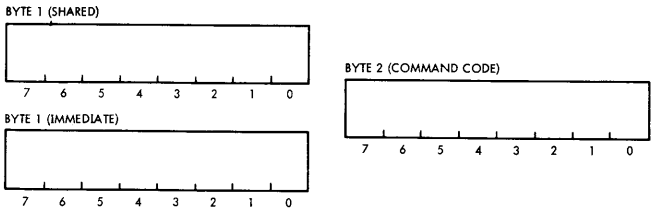
g. Augmented Memory Addressing

With each instruction description is a diagram showing the format of the instruction and its operation code. Bit positions or fields that are shaded in the diagram represent portions of the instruction that are ignored. Although these bits are not used at present, they should always be coded with zeros to preclude conflict with features that may be added in the future.

The instruction descriptions are given in the following format:

MNEMONIC① INSTRUCTION NAME② TIMING③

④



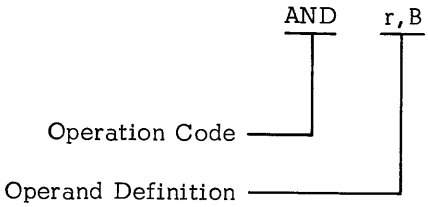
DESCRIPTION⑤

SYMBOLIC DESCRIPTION ⑥

INDICATORS⑦

QUALIFIED REGISTERS⑧

① The mnemonic for each instruction is shown using the syntax of the Conversational Assembly System (CAS). A typical mnemonic appears as follows:



During program assembly, CAS uses the operation code mnemonic and the operand definition to construct the machine language instruction as shown in the format diagram given with each instruction description.

② The instruction name is the descriptive title of the instruction.

③ Instruction timing is given both in the number of memory cycles required and in execution time in microseconds.

④ The instruction format diagram identifies the effective field locations within each byte. Both formats are shown for byte 1 of command code addressing instructions. The following abbreviations are used in the instruction format diagrams:

a	Autoincrement designator
CCBA	Command code byte address
d	Destination (register)
m	Memory location
n	Number (octal digit)
OP	Operation code
r	Register designator
s	Source (register)
t	Index (tag) register
v	Literal value

(5) The description of the instruction defines the operations performed by the computer as it executes the instruction.

(6) The symbolic description shows the instruction operation as a series of generalized symbolic statements. Symbolic terms used in the description are:

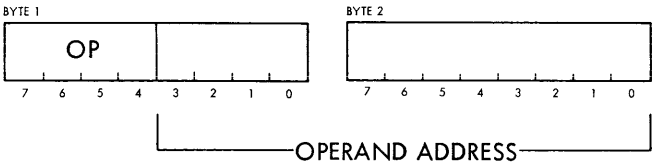
EOA	Effective operand address
( )	Contents of the register or memory location indicated
+	Addition
-	Subtraction
n	Logical product (AND)
u	Logical sum (OR)
(u)	Logical difference (exclusive OR)
⟶	Replaces

(7) The effect of the execution of the instruction upon the indicators (Zero, Plus, Link, I/O Test) is specified.

(8) All programmable registers that can be manipulated by the instruction are specified.

3.2 MEMORY ADDRESSING INSTRUCTIONS

Instructions of the memory addressing group provide the capability of directly addressing an 8- or 12-bit operand within 4096 words of operating memory. Memory addressing instructions have the following double-word format:



Bits 4 through 7 of byte 1 designate the operation to be performed; bits 0 through 3 of byte 1 contain the four most significant bits of the operand address, and bits 0 through 7 of byte 2 contain the eight least significant bits of the operand address. The operand is always exchanged between memory and the B-register.

The six memory addressing instructions are:

Load B-Register

Load B-Register Indexed

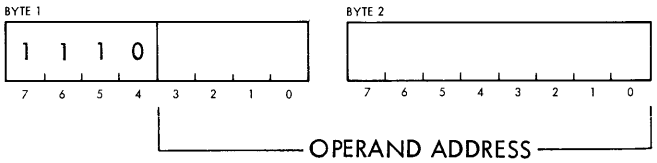
Store B-Register

Store B-Register Indexed

Extended Load B-Register

Jump Unconditionally

LDB m LOAD B-REGISTER **3 cy 6.48 μs**



The Load B-Register instruction replaces the contents of bits 0 through 7 of the B-register with the contents of the effective operand address (m). Zeros replace the contents of bits 8 through 11 of the B-register. The contents of the effective operand address remain unchanged.

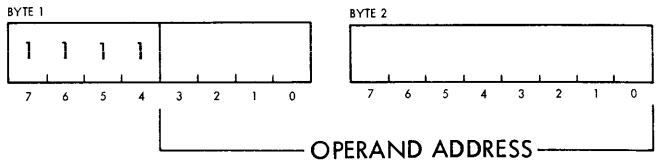
$$(EOA)_{0-7} \longrightarrow (B)_{0-7}$$

$$0's \longrightarrow (B)_{8-11}$$

Indicators: Zero and Plus record the adder result of the load operation. Link is not affected.

Qualified Registers: B

LDB m,X LOAD B-REGISTER INDEXED **3 cy 6.48 μs**

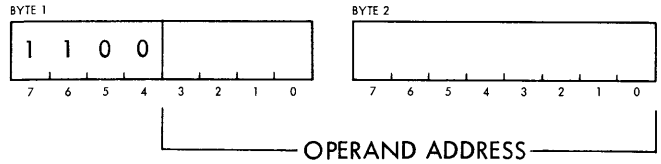


This instruction is the same as the Load B-Register instruction except that the contents of the X-Register are added to the operand address (m) to produce the effective operand address.

Indicators: Same as Load B-Register

Qualified Registers: B, X

STB m STORE B-REGISTER **3 cy 6.48 μs**



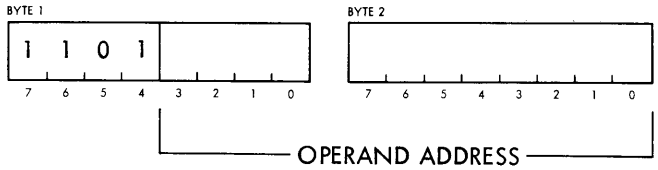
The Store B-Register instruction replaces the contents of the effective operand address (m) with the contents of bits 0 through 7 of the B-register. The contents of the B-register remain unchanged.

$$(B)_{0-7} \longrightarrow (EOA)_{0-7}$$

Indicators: Zero and Plus record the result of the store operation. Link is not affected.

Qualified Registers: B

STB m,X STORE B-REGISTER INDEXED **3 cy 6.48 μs**

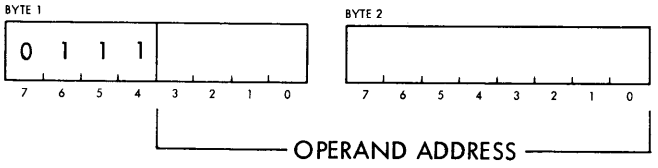


This instruction is the same as the Store B-Register instruction except that the contents of the X-Register are added to the operand address (m) to produce the effective operand address.

Indicators: Same as Store B-Register

Qualified Registers: B, X

ELB m EXTENDED LOAD B-REGISTER **4 cy 8.64 μs**



The Extended Load B-Register instruction replaces the contents of bits 0 through 7 of the B-register with the contents of the effective operand address (m) and replaces bits 8 through 11 of the B-register with the contents of bits 0 through 3 of the location immediately following the effective operand address. The contents of both memory locations remain unchanged.

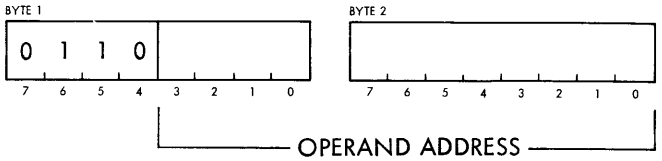
$$(EOA)_{0-7} \longrightarrow (B)_{0-7}$$

$$(EOA+1)_{0-3} \longrightarrow (B)_{8-11}$$

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: B

JMP m JUMP UNCONDITIONALLY **2 cy 4.32 μs**



The Jump Unconditionally instruction modifies program execution by causing the next instruction to be taken from the effective operand address (m) instead of from the next sequential memory location. The effective operand address replaces the contents of the P-register.

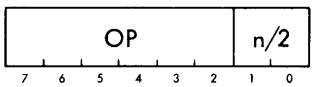
$$EOA_{0-11} \longrightarrow (P)_{0-11}$$

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: P

3.3 SKIP INSTRUCTIONS

Instructions of the skip group test the status of the Link, Zero, Plus, and I/O Test indicators (paragraph 2.12), and, based on the status of the indicators, allow the normal program sequence to continue or cause the next 0, 2, 4, or 6 bytes to be skipped before the program sequence continues. Skip instructions have the following single-word format:



Bits 2 through 7 designate the test to be performed; bits 0 and 1 designate the number of bytes to be skipped as follows:

0 0	Skip 0 bytes
0 1	Skip 2 bytes
1 0	Skip 4 bytes
1 1	Skip 6 bytes

Note that a skip of 0 bytes is in effect a No Operation instruction.

The eight skip instructions are:

Skip if Link Set

Skip if Link Reset

Skip if Zero

Skip if Not Zero

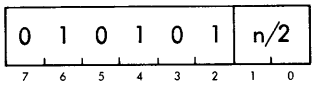
Skip if Plus

Skip if Minus

Skip if I/O Test True

Skip if I/O Test False

SKS n SKIP IF LINK SET **1 cy 2.16 μs**



The Skip if Link Set instruction tests the Link indicator for the SET state. If it is SET, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If Link is RESET, the program sequence continues from the next sequential location.

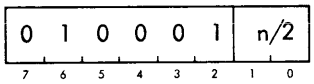
If SET: (P) + n + 1 ⟶ (P)

If RESET: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKR n SKIP IF LINK RESET **1 cy 2.16 μs**



The Skip if Link Reset instruction tests the Link indicator for the RESET state. If it is RESET, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If Link is SET, the program sequence continues from the next sequential location.

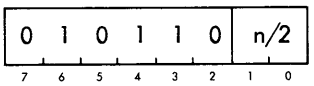
If RESET: (P) + n + 1 ⟶ (P)

If SET: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKZ n SKIP IF ZERO **1 cy 2.16 μs**



The Skip if Zero instruction tests the Zero indicator for the ZERO state. If it is ZERO, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If it is NOT ZERO, the program sequence continues from the next sequential location.

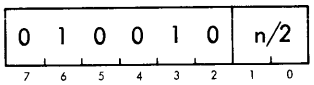
If ZERO: (P) + n + 1 ⟶ (P)

If NOT ZERO: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKN n SKIP IF NOT ZERO **1 cy 2.16 μs**



The Skip if Not Zero instruction tests the Zero indicator for the NOT ZERO state. If it is NOT ZERO, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If it is ZERO, the program sequence continues from the next sequential location.

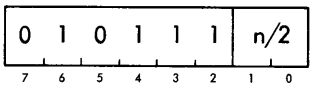
If NOT ZERO: (P) + n + 1 ⟶ (P)

If ZERO: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKP n SKIP IF PLUS **1 cy 2.16 μs**



The Skip if Plus instruction tests the Plus indicator for the PLUS state. If it is PLUS, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If it is MINUS, the program sequence continues from the next sequential location.

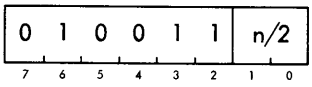
If PLUS: (P) + n + 1 ⟶ (P)

If MINUS: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKM n SKIP IF MINUS **1 cy 2.16 μs**



The Skip if Minus instruction tests the Plus indicator for the MINUS state. If it is MINUS, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If it is PLUS, the program sequence continues from the next sequential location.

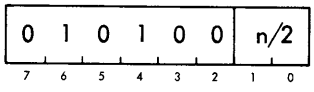
If MINUS: (P) + n + 1 ⟶ (P)

If PLUS: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKT n SKIP IF I/O TEST TRUE **1 cy 2.16 μs**



The Skip if I/O Test True instruction tests the I/O Test indicator for the TRUE state. If it is TRUE, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If I/O Test is FALSE, the program sequence continues from the next sequential location.

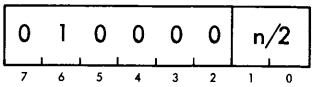
If TRUE: (P) + n + 1 ⟶ (P)

If FALSE: (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

SKF n SKIP IF I/O TEST FALSE **1 cy 2.16 μs**



The Skip if I/O Test False instruction tests the I/O Test indicator for the FALSE state. If it is FALSE, the program sequence continues after skipping the next 0, 2, 4, or 6 locations as designated by n. If I/O Test is TRUE, the program sequence continues from the next sequential location.

If FALSE: (P) + n + 1 ⟶ (P)

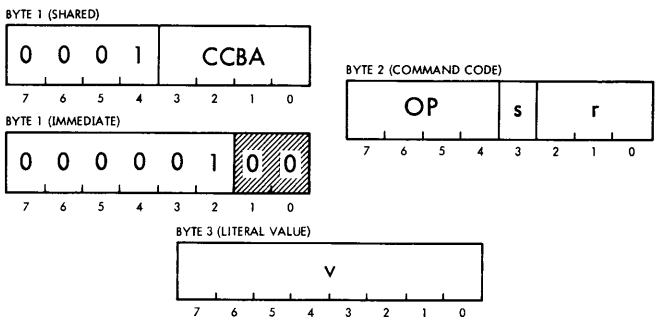
If TRUE (P) + 1 ⟶ (P)

Indicators: Link, Zero, and Plus are not affected.

Qualified Registers: None.

3.4 ARITHMETIC/LOGICAL INSTRUCTIONS

Instructions of the arithmetic/logical group are used to perform arithmetic or logical operations between a source operand and the contents of one of seven selected registers. The source operand can be the contents of the B-register or an eight-bit literal value. Arithmetic/logical instructions have the following format:



Byte 1 of the instruction identifies the instruction as belonging to the arithmetic/logical group and designates either immediate or shared command addressing. If immediate command addressing is designated, byte 2 of the instruction is taken from the location immediately following byte 1. If shared command addressing is designated, byte 2 is taken from 1 of 16 memory locations (0020_8 through 0037_8) shared by the arithmetic/logical instructions. The relative location of byte 2 in the shared memory pool is designated by bits 0 through 3 of byte 1.

Byte 2 of the instruction contains the operation code and operand definition. Bits 4 through 7 designate the operation to be performed. Bit 3 designates the source operand as follows:

a. If the source operand is the contents of the B-register, bit 3 is 0.

b. If the source operand is a literal value and shared command addressing is designated, bit 3 is a 1. For this condition the literal value is contained in byte 3 at the memory location immediately following byte 1 (P+1).

c. If the source operand is a literal value and immediate command addressing is designated, bit 3 is a 1. For this condition the literal value is contained in byte 3 at the memory location immediately following byte 2 (P+2).

Bits 0 through 2 designate the selected (destination) register.

The seven operations of the arithmetic/logical instructions are:

Zero Register

Logical AND

Logical Exclusive OR

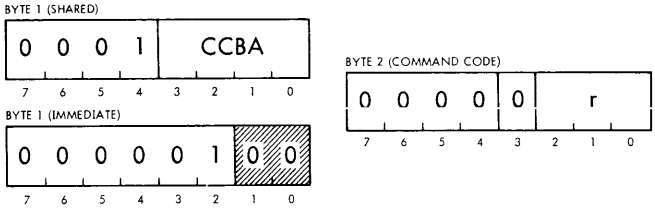
Logical OR

Load Register

Subtract

Add

AZE r ZERO REGISTER **2 cy 4.32 μs**



The Zero Register instruction replaces the contents of the selected register (r) with zeros.

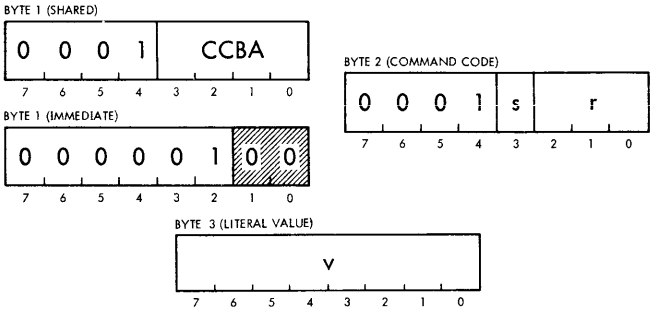
$$0's \longrightarrow (r)_{0-11}$$

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: A, X, Y, Z, P, YY, or ZZ

AND r,B LOGICAL AND (B with r) **2 cy 4.32 μs**

AND r,v LOGICAL AND (v with r) **3 cy 6.48 μs**



The Logical AND instruction forms the logical product of the source operand (B or v) and the contents of the selected register (r). The logical product replaces the contents of bits 0 through 7 of the selected register, and zeros replace the contents of bits 8 through 11.

If the source operand is the contents of the B-register, its contents remain unchanged.

If the source operand is the literal value in byte 3 at memory location P+1 or P+2, the contents of the memory location replace the contents of bits 0 through 7 of the B-register, zeros replace the contents of bits 8 through 11 of the B-register, and the contents of the memory location remain unchanged.

If source operand is B-register:

$$(B)_{0-7} \cap (r)_{0-7} \longrightarrow (r)_{0-7}$$

$$0's \longrightarrow (r)_{8-11}$$

If source operand is literal value:

$$((P) + 1 \text{ or } 2)_{0-7} \longrightarrow (B)_{0-7}$$

$$0's \longrightarrow (B)_{8-11}$$

$$(B)_{0-7} \cap (r)_{0-7} \longrightarrow (r)_{0-7}$$

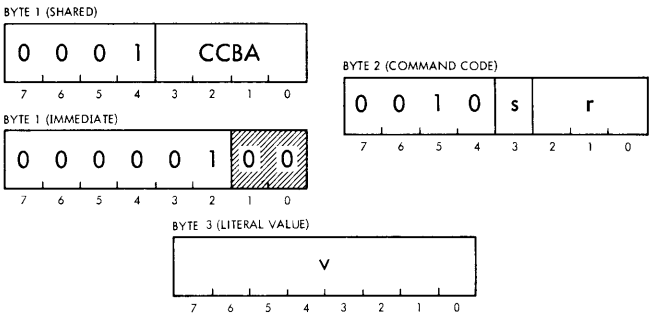
$$0's \longrightarrow (r)_{8-11}$$

Indicators: Zero and Plus record adder result of forming logical product. Link is not affected.

Qualified Registers: B (source); A, X, Y, Z, P, YY, or ZZ (destination)

AXR r,B **LOGICAL EXCLUSIVE OR (B with r)** **2 cy 4.32 μs**

AXR r,v **LOGICAL EXCLUSIVE OR (v with r)** **3 cy 6.48 μs**



The Logical Exclusive OR instruction forms the logical difference of the source operand (B or v) and the contents of the selected register (r). The logical difference replaces the contents of bits 0 through 7 of the selected register, and zeros replace the contents of bits 8 through 11.

If the source operand is the contents of the B-register, its contents remain unchanged.

If the source operand is the literal value in byte 3 at memory location P+1 or P+2, the contents of the memory location replace the contents of bits 0 through 7 of the B-register, zeros replace the contents of bits 8 through 11 of the B-register, and the contents of the memory location remain unchanged.

If source operand is B-register:

$$(B)_{0-7} \textcircled{\cup} (r)_{0-7} \longrightarrow (r)_{0-7}$$

$$0's \longrightarrow (r)_{8-11}$$

If source operand is literal value:

$$((P) + 1 \text{ or } 2)_{0-7} \longrightarrow (B)_{0-7}$$

$$0's \longrightarrow (B)_{8-11}$$

$$(B)_{0-7} \textcircled{\cup} (r)_{0-7} \longrightarrow (r)_{0-7}$$

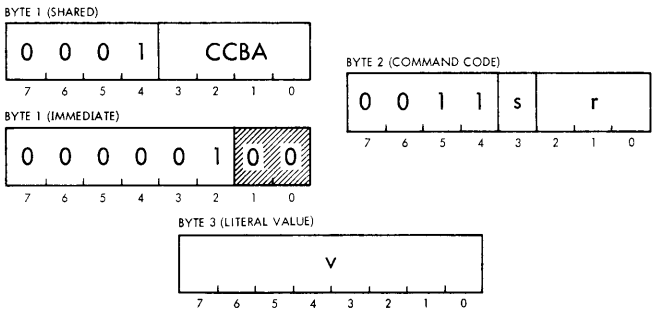
$$0's \longrightarrow (r)_{8-11}$$

Indicators: Zero and Plus record adder result of forming logical difference. Link is not affected.

Qualified Registers: B (source); A, X, Y, Z, P, YY, or ZZ (destination)

AOR r,B **LOGICAL OR (B with r)** **2 cy 4.32 μs**

AOR r,v **LOGICAL OR (v with r)** **3 cy 6.48 μs**



The Logical OR instruction forms the logical sum of the source operand (B or v) and the contents of the selected register (r). The logical sum replaces the contents of bits 0 through 7 of the selected register, and zeros replace the contents of bits 8 through 11.

If the source operand is the contents of the B-register, its contents remain unchanged.

If the source operand is the literal value in byte 3 at memory location P+1 or P+2, the contents of the memory location replace the contents of bits 0 through 7 of the B-register, zeros replace the contents of bits 8 through 11 of the B-register, and the contents of the memory location remain unchanged.

If source operand is B-register:

$$(B)_{0-7} \cup (r)_{0-7} \longrightarrow (r)_{0-7}$$

$$0's \longrightarrow (r)_{8-11}$$

If source operand is literal value:

$$((P) + 1 \text{ or } 2)_{0-7} \longrightarrow (B)_{0-7}$$

$$0's \longrightarrow (B)_{8-11}$$

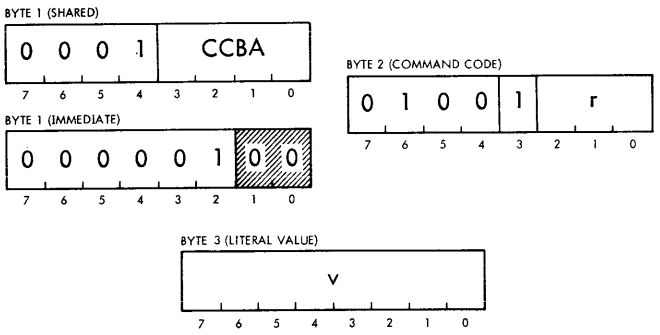
$$(B)_{0-7} \cup (r)_{0-7} \longrightarrow (r)_{0-7}$$

$$0's \longrightarrow (r)_{8-11}$$

Indicators: Zero and Plus record adder result of forming logical sum. Link is not affected.

Qualified Registers: B (source); A, X, Y, Z, P, YY, or ZZ (destination)

ALD r,v **LOAD REGISTER (v into r)** **3 cy 6.48 μs**



The Load Register instruction loads the selected register (r) with the literal value (v). The literal value in byte 3 at memory location P+1 or P+2 replaces the contents of bits 0 through 7 of the B-register, zeros replace the contents of bits 8 through 11 of the B-register, and the contents of the memory location remain unchanged.

$$((P) + 1 \text{ or } 2)_{0-7} \longrightarrow (B)_{0-7}$$

$$0's \longrightarrow (B)_{8-11}$$

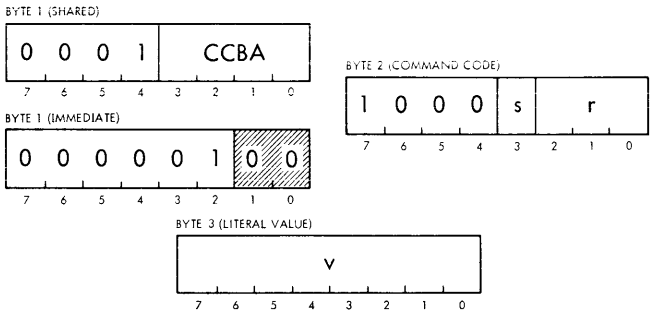
$$(B)_{0-11} \longrightarrow (r)_{0-11}$$

Indicators: Zero and Plus record adder result of load operation except when selected register is B-register; then Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: A, X, Y, Z, P, B, YY or ZZ

ASU r,B **SUBTRACT (B from r)** **2 cy 4.32 μs**

ASU r,v **SUBTRACT (v from r)** **3 cy 6.48 μs**



The Subtract instruction subtracts the source operand (B or v) from the contents of the selected register (r). The difference replaces the contents of the selected register.

If the source operand is the contents of the B-register, its contents remain unchanged.

If the source operand is the literal value in byte 3 at memory location P+1 or P+2, the contents of the memory location replace the contents of bits 0 through 7 of the B-register, zeros replace the contents of bits 8 through 11 of the B-register, and the contents of the memory location remain unchanged.

If source operand is B-register:

$$(r)_{0-11} - (B)_{0-11} \longrightarrow (r)_{0-11}$$

If source operand is literal value:

$$((P) + 1 \text{ or } 2)_{0-7} \longrightarrow (B)_{0-7}$$

$$0's \longrightarrow (B)_{8-11}$$

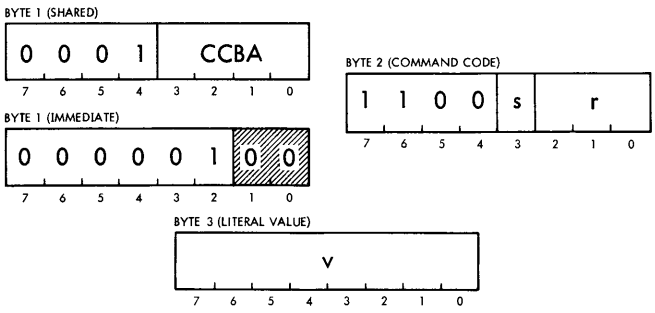
$$(r)_{0-11} - (B)_{0-11} \longrightarrow (r)_{0-11}$$

Indicators: Zero and Plus record adder result of subtraction. Link is set if a carry occurs from bit position 7 of the adder; if no carry occurs, Link is reset.

Qualified Registers: B (source); A, X, Y, Z, P, YY or ZZ (destination)

AAD r,B **ADD (B to r)** **2 cy 4.32 μs**

AAD r,v **ADD (v to r)** **3 cy 6.48 μs**



The Add instruction adds the source operand (B or v) to the contents of the selected register (r). The sum replaces the contents of the selected register.

If the source operand is the contents of the B-register, its contents remain unchanged.

If the source operand is the literal value in byte 3 at memory location P+1 or P+2, the contents of the memory location replace the contents of bits 0 through 7 of the B-register, zeros replace the contents of bits 8 through 11 of the B-register, and the contents of the memory location remain unchanged.

If source operand is B-register:

$$(B)_{0-11} + (r)_{0-11} \longrightarrow (r)_{0-11}$$

If source operand is literal value:

$$((P) + 1 \text{ or } 2)_{0-7} \longrightarrow (B)_{0-7}$$

$$0\text{'s} \longrightarrow (B)_{8-11}$$

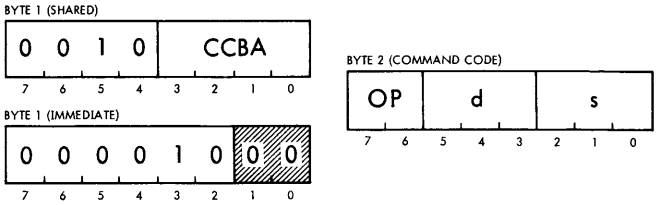
$$(B)_{0-11} + (r)_{0-11} \longrightarrow (r)_{0-11}$$

Indicators: Zero and Plus record adder result of addition. Link is set if a carry occurs from bit position 7 of the adder; if no carry occurs, Link is reset.

Qualified Registers: B (source); A, X, Y, Z, P, YY, or ZZ (destination)

3.5 REGISTER TRANSFER INSTRUCTIONS

Instructions of the register transfer group are used for register-to-register manipulation of any of eight registers (A, X, Y, Z, P, B, YY, and ZZ). Register transfer instructions have the following double-word format:



Byte 1 of the instruction designates either immediate or shared command addressing. If immediate command addressing is designated, byte 2 of the instruction is taken from the location immediately following byte 1. If shared command addressing is designated byte 2 is taken from 1 of 16 memory locations (40_8 through 57_8) shared by the register transfer instructions. The relative location of byte 2 in the shared memory pool is designated by bits 0 through 3 of byte 1.

Byte 2 of the instruction contains the operation code and operand definition. Bits 6 and 7 designate the operation to be performed. Bits 3 through 5 specify the destination register, and bits 0 through 2 specify the source register for the operation.

The four operations of the register transfer instructions are:

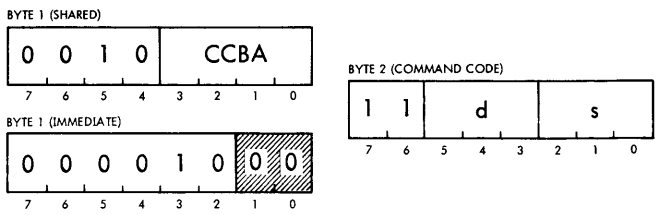
Register Transfer

Register Transfer and Increment

Register Transfer and Decrement

Register Transfer and Add Link

RTR s,d **REGISTER TRANSFER** **2 cy 4.32 μs**



The Register Transfer instruction replaces the contents of the destination register (d) with the

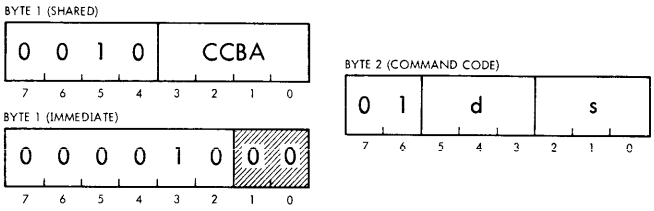
contents of the source register (s). The contents of the source register remain unchanged.

$$(s)_{0-11} \longrightarrow (d)_{0-11}$$

Indicators: Zero and Plus record the adder result of the transfer. Link is not affected.

Qualified Registers: A, X, Y, Z, P, B, YY or ZZ

RIC s,d REGISTER TRANSFER AND INCREMENT 2 cy 4.32 μs



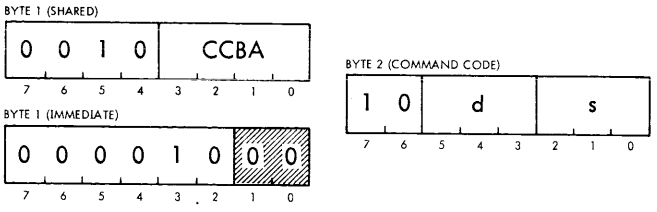
The Register Transfer and Increment instruction replaces the contents of the destination register (d) with the contents plus one of the source register (s). The contents of the source register remain unchanged unless it is also the destination register.

$$(s)_{0-11} + 1 \longrightarrow (d)_{0-11}$$

Indicators: Zero and Plus record the adder result of the transfer. Link is set if a carry occurs from bit position 7 of the adder; if no carry occurs, Link is reset. (NOTE: No carry can occur if the source register is YY or ZZ.)

Qualified Registers: A, X, Y, Z, P, B, YY, or ZZ

RDC s,d REGISTER TRANSFER AND DECREMENT 2 cy 4.32 μs



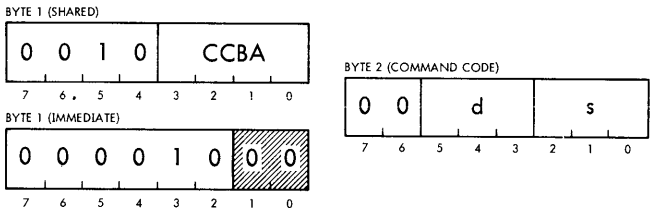
The Register Transfer and Decrement instruction replaces the contents of the destination register (d) with the contents minus one of the source register (s) unless the source register is the B-register. When the B-register is selected as the source register, all ones are transferred to the destination register. The contents of the source register remain unchanged unless it is also the destination register.

$$(s)_{0-11} - 1 \longrightarrow (d)_{0-11}$$

Indicators: Zero and Plus record the adder result of the transfer. Link is set if a carry occurs from bit position 7 of the adder; if no carry occurs, Link is reset.

Qualified Registers: A, X, Y, Z, P, B, YY, or ZZ

RLK s,d REGISTER TRANSFER AND ADD LINK 2 cy 4.32 μs



The Register Transfer and Add Link instruction replaces the contents of the destination register (d) with the contents of the source register (s) plus the contents of Link. The contents of the source register remain unchanged unless it is also the destination register.

$$(s)_{0-11} + (\text{Link}) \longrightarrow (d)_{0-11}$$

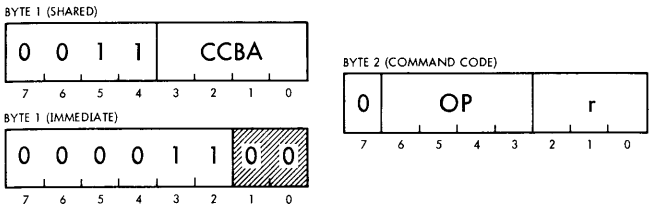
Indicators: Zero and Plus record the adder result of the transfer. Link is set if a carry occurs from bit position 7 of the adder; if no carry occurs, Link is reset. (NOTE: No carry can occur if the source register is YY or ZZ.)

Qualified Registers: A, X, Y, Z, P, B, YY, or ZZ

3.6 SHIFT INSTRUCTIONS

Instructions of the shift group are used to perform right shifts or circular shifts on the contents of a selected register, to shift data from the serial I/O system into a selected register, or to shift data

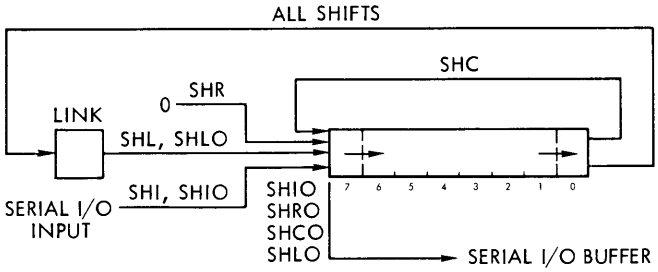
from a selected register to the serial I/O system. Shift instructions have the following double-word format:



Byte 1 of the instruction identifies the instruction as belonging to the shift group[1] and designates either immediate or shared command addressing. If immediate command addressing is designated, byte 2 of the instruction is taken from the location immediately following byte 1. If shared command addressing is designated, byte 2 is taken from 1 of 16 memory locations (60_8 through 77_8) shared by the shift and control instructions. The relative location of byte 2 in the shared memory pool is designated by bits 0 through 3 of byte 1.

Byte 2 of the instruction contains the operation code and operand definition. Bits 3 through 6 designate the operation to be performed. Bits 0 through 2 specify the register affected by the operation.

The following illustration shows the shift patterns for the various shift operations that can be performed by the shift instructions:



The eight operations of the shift instructions are:

Shift Right

Shift Circular

Shift Circular through Link

Shift Serial Data In

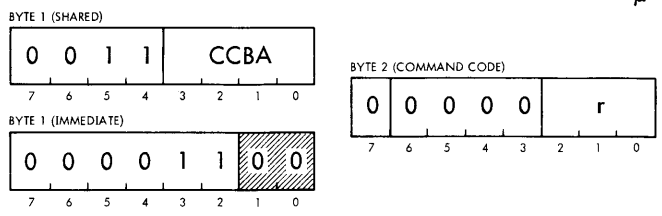
Shift Serial Data In and Out

Shift Right and Serial Data Out

Shift Circular and Serial Data Out

Shift Circular through Link and Serial Data Out

SHR r SHIFT RIGHT **2 cy 4.32 μs**



The Shift Right instruction shifts the contents of bits 0 through 7 of the selected register (r) one bit position to the right. Bit 0 of the selected register is shifted into Link, and zeros replace the contents of bit positions 7 through 11 of the selected register.

$$(r)_{n+1} \longrightarrow (r)_n \qquad n = 0\text{-}7$$

$$(r)_0 \longrightarrow (\text{Link})$$

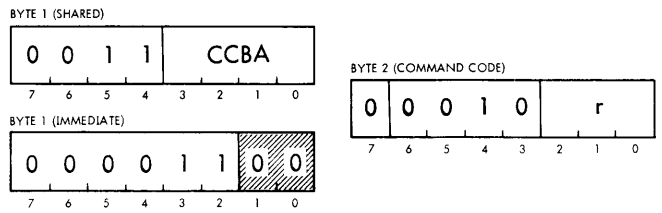
$$0 \longrightarrow (r)_7$$

$$0\text{'s} \longrightarrow (r)_{8\text{-}11}$$

Indicators: Zero records the adder result of the shift. Plus is always set to indicate PLUS. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, or Z

SHC r SHIFT CIRCULAR **2 cy 4.32 μs**



The Shift Circular instruction shifts the contents of bits 0 through 7 of the selected register (r) one bit

[1] Both the shift instructions and the control instructions have the same format for byte 1.

position to the right. The bit shifted out of bit position 0 is shifted into bit position 7, and zeros replace the contents of bit positions 8 through 11.

$$(r)_{n+1} \longrightarrow (r)_n \qquad n = 0\text{-}7$$

$$(r)_0 \longrightarrow (r)_7$$

$$(r)_0 \longrightarrow (\text{Link})$$

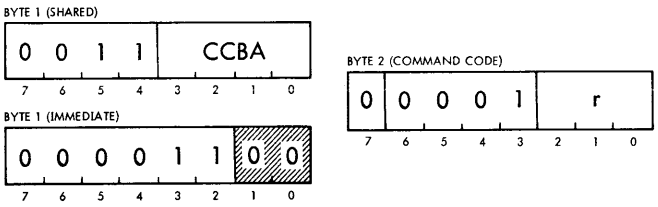
$$0\text{'s} \longrightarrow (r)_{8\text{-}11}$$

Indicators: Zero and Plus record the adder result after the shift. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, or Z.

SHL r SHIFT CIRCULAR THROUGH LINK

2 cy
4.32 μs



The Shift Circular through Link instruction shifts the contents of bits 0 through 7 of the selected register (r) one bit position to the right. The bit shifted out of bit position 0 is shifted into Link, and Link is shifted into bit position 7. Zeros replace the contents of bit positions 8 through 11.

$$(r)_{n+1} \longrightarrow (r)_n \qquad n = 0\text{-}7$$

$$(r)_0 \longrightarrow (\text{Link})$$

$$(\text{Link}) \longrightarrow (r)_7$$

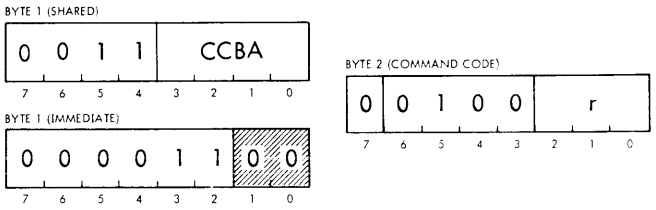
$$0\text{'s} \longrightarrow (r)_{8\text{-}11}$$

Indicators: Zero and Plus record the adder result after the shift. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, or Z

SHI r SHIFT SERIAL DATA IN

2 cy
4.32 μs



The Shift Serial Data In instruction samples the content of the serial I/O interface line and sets bit 7 of the selected register equal to the sampled value. The contents of bits 0 through 7 of the register are shifted one bit position to the right. The bit shifted out of bit position 0 is shifted into Link, and zeros replace the contents of bit positions 8 through 11. (See paragraph 4.2 for a description of the serial I/O system.)

$$(\text{Serial I/O Line}) \longrightarrow (r)_n$$

$$(r)_{n+1} \longrightarrow (r)_n \qquad n = 0\text{-}7$$

$$(r)_0 \longrightarrow (\text{Link})$$

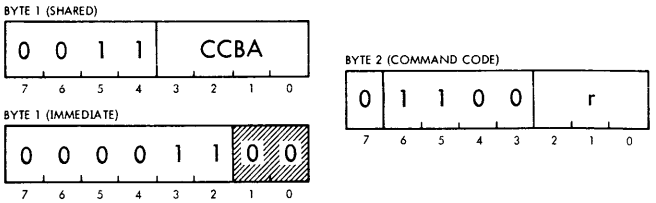
$$0\text{'s} \longrightarrow (r)_{8\text{-}11}$$

Indicators: Zero and Plus record the adder result after the shift. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, or Z.

SHIO r SHIFT SERIAL DATA IN AND OUT

2 cy
4.32 μs



The Shift Serial Data In and Out instruction samples the content of the serial I/O interface line and places the sampled value into bit 7 of the selected register (r) and into the serial data output buffer. The contents of bits 0 through 7 of the selected register are shifted one bit position to the right.

The bit shifted out of bit position 0 is shifted into Link, and zeros replace the contents of bit positions 8 through 11. (See paragraph 4.2 for a description of the serial I/O system.)

(Serial I/O Line) ⟶ $(r)_7$ and (Serial data output buffer)

$(r)_{n+1}$ ⟶ $(r)_n$ n = 0-7

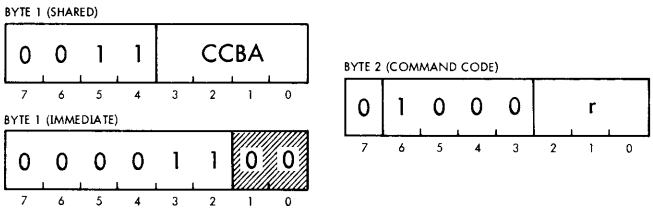
$(r)_0$ ⟶ (Link)

0's ⟶ $(r)_{8-11}$

Indicators: Zero and Plus record the adder result after the shift. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, Z.

SHRO r SHIFT RIGHT AND SERIAL DATA OUT 2 cy 4.32 μs



The Shift Right and Serial Data Out instruction shifts a zero into the serial data output buffer and into bit position 7 of the selected register (r). The contents of bits 0 through 7 of the register are shifted one bit position to the right. The bit shifted out of bit position 0 is shifted into Link, and zeros replace the contents of bit positions 8 through 11. (See paragraph 4.2 for a description of the serial I/O system.)

0's ⟶ (Serial data output buffer)

$(r)_{n+1}$ ⟶ $(r)_n$ n = 0-7

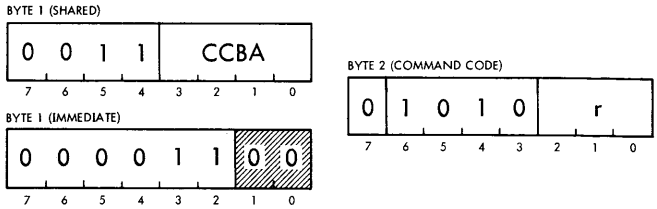
0's ⟶ $(r)_{7-11}$

$(r)_0$ ⟶ (Link)

Indicators: Zero records the adder result after the shift. Plus is set to indicate PLUS. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, or Z.

SHCO r SHIFT CIRCULAR AND SERIAL DATA OUT 2 cy 4.32 μs



The Shift Circular and Serial Data Out instruction shifts the contents of bits 0 through 7 of the selected register (r) one bit position to the right. The bit shifted out of bit position 0 is shifted into bit position 7 and into the serial data output buffer. Zeros replace the contents of bits 8 through 11.

$(r)_{n+1}$ ⟶ $(r)_n$ n = 0-7

$(r)_0$ ⟶ $(r)_7$ and (Serial Data Output Buffer)

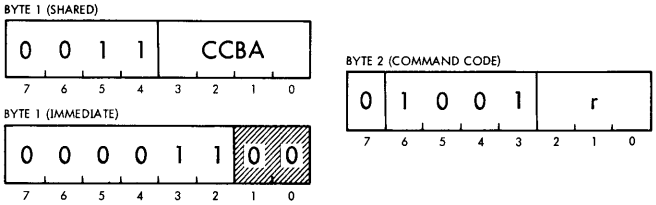
$(r)_0$ ⟶ (Link)

0's ⟶ $(r)_{8-11}$

Indicators: Zero and Plus record the adder result after the shift. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, or Z.

SHLO r SHIFT CIRCULAR THROUGH LINK AND SERIAL DATA OUT 2 cy 4.32 μs



The Shift Circular through Link and Serial Data Out instruction shifts the contents of bits 0 through 7 of the selected register (r) one bit position to the right. The bit shifted out of bit position 0 is shifted into Link, and Link is shifted into bit position 7 and into

the serial data output buffer. Zeros replace the contents of bits 8 through 11. (See paragraph 4.2 for a description of the serial I/O system.)

$$(r)_{n+1} \longrightarrow (r)_n \qquad n = 0\text{-}7$$

$$(r)_0 \longrightarrow (\text{Link})$$

$$(\text{Link}) \longrightarrow (r)_7 \text{ and (Serial data output buffer)}$$

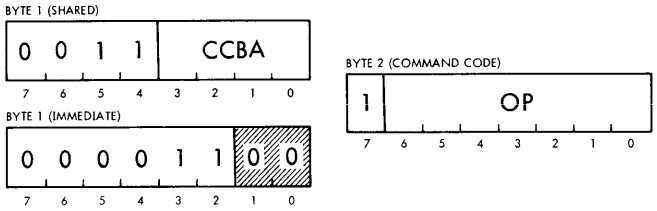
$$0\text{'s} \longrightarrow (r)_{8\text{-}11}$$

Indicators: Zero and Plus record the adder result after the shift. The state of bit 0 before the shift is recorded in Link after the shift.

Qualified Registers: A, X, Y, Z.

3.7 CONTROL INSTRUCTIONS

Instructions of the control group are used to perform internal control operations, to control operation of the parallel I/O system, and to interrogate the external and relative time clock interrupts. Control instructions have the following double-word format:



Byte 1 of the instruction identifies the instruction as belonging to the control group[1] and designates either immediate or shared command addressing. If immediate command addressing is designated, byte 2 of the instruction is taken from the location immediately following byte 1. If shared command addressing is designated, byte 2 is taken from 1 of 16 memory locations (60_8 through 77_8) shared by the control and shift instructions. The relative location of byte 2 in the shared memory pool is designated by bits 0 through 3 of byte 1.

Byte 2 of the instruction contains the operation code for the operation to be performed.

[1] Both the control instructions and the shift instructions have the same format for byte 1.

The nine operations of the control instructions are:

Pulse Link Reset

Pulse Link Set

Pulse Stall Alarm (optional)

Transfer BB- to B-Register

Transfer B- to E-Register

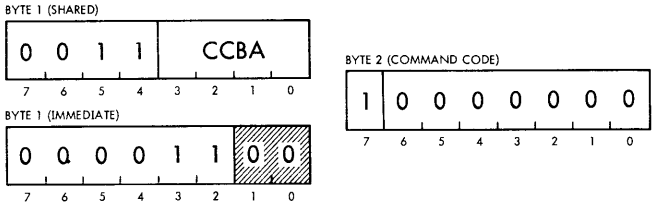
Function-Address Out from B-Register

Data Out From B-Register

Data Into B-Register

Interrupt Enable

PLR PULSE LINK RESET **2 cy 4.32 μs**



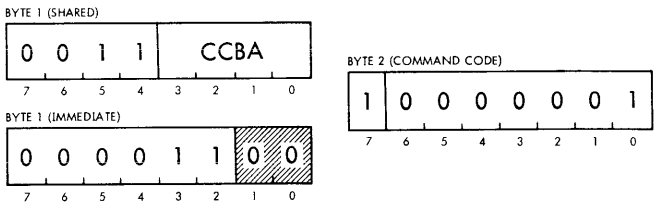
The Pulse Link Reset instruction resets the Link indicator.

$$0 \longrightarrow (\text{Link})$$

Indicators: Zero and Plus are not affected. Link is reset.

Qualified Registers: None.

PLS PULSE LINK SET **2 cy 4.32 μs**



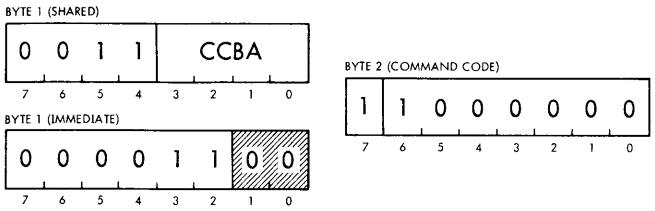
The Pulse Link Set instruction sets the Link indicator.

$$1 \longrightarrow (\text{Link})$$

Indicators: Zero and Plus are not affected. Link is set.

Qualified Registers: None.

PSA PULSE STALL ALARM (optional) 2 cy 4.32 μs



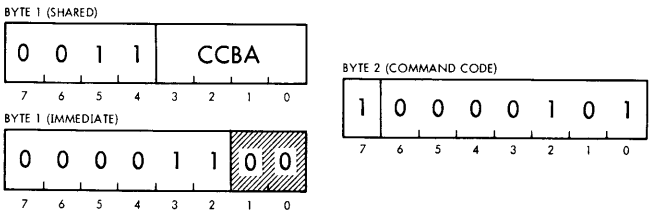
The Pulse Stall Alarm instruction is provided as part of the stall alarm option. The stall alarm warns of any abnormality in system operation, particularly in the event of sequential program interruptions due to component breakdowns or previously undetected program errors. It provides an orderly halt to instruction execution, and generates a signal through the safe line (SFEC-I) of the parallel I/O system which can be used for an audio-visual alarm or automatic switchover.

The PSA instruction resets the stall alarm timer each time the instruction is executed. Failure to execute the instruction within 250 (±100) ms after the previous execution activates the stall alarm. With the stall alarm activated the computer automatically switches from run mode to idle mode and the safe signal is removed from the safe line. The stall alarm timer does not begin timing until the instruction is executed for the first time. Thereafter, it must be continually reset to prevent the stall alarm from being activated.

Indicators: Zerc and Plus are not affected. Link is reset.

Qualified Registers: None

TBB TRANSFER BB- TO B-REGISTER 2 cy 4.32 μs



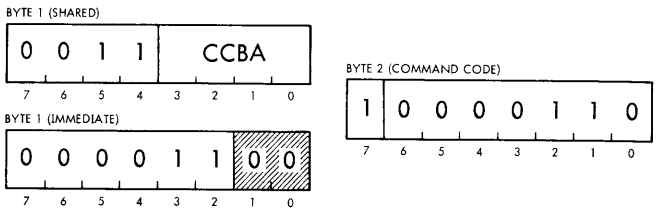
The Transfer BB- to B-Register instruction replaces the contents of bits 0 through 11 of the B-register with the contents of bits 0 through 11 of the BB-register.

$$(BB)_{0-11} \longrightarrow (B)_{0-11}$$

Indicators: Zero records the adder result of the transfer. Plus is set to indicate PLUS. Link is not affected.

Qualified Registers: BB (source) and B (destination)

TBE TRANSFER B- TO E-REGISTER 2 cy 4.32 μs



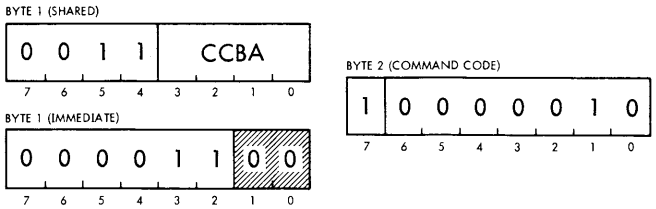
The Transfer B- to E-Register instruction replaces the contents of bits 0 through 2 of the E-register with the contents of bits 0 through 2 of the B-register. The contents of the B-register remain unchanged.

$$(B)_{0-2} \longrightarrow (E)_{0-2}$$

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: B (source) and E (destination)

FOB FUNCTION-ADDRESS OUT FROM B-REGISTER 2 cy 4.32 μs



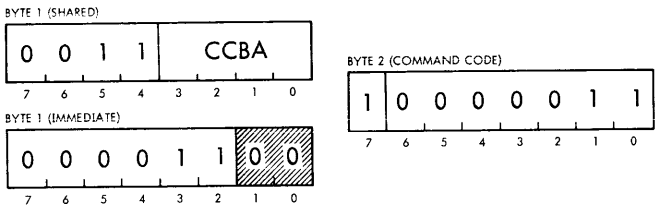
The Function-Address Out from B-Register instruction is used with a function-address word previously loaded into the B-register to send control information to, request status information from, or initiate data input or output operations with I/O devices connected to the parallel I/O system. Executing the instruction causes a 720-ns pulse to be sent to all devices on the function-address pulse line (FAP>-I). The function-address pulse identifies the information on the I/O data lines as a function-address word. (See paragraph 4.3 for a description of parallel I/O operations.)

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected. I/O Test records the status of the I/O test line.

Qualified Registers: B

DOB DATA OUT FROM B-REGISTER

2 cy
4.32 μs



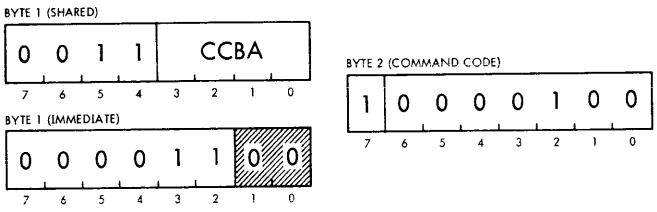
The Data Out from B-Register instruction causes data previously loaded into the B-register to be transferred to an I/O device connected to the parallel I/O system. This instruction must be preceded by a Function-Address Out from B-Register instruction with its corresponding function-address word specifying the data output mode. Executing the instruction causes a 720-ns pulse to be transmitted on the transfer-out pulse line (TØP>-I). The transfer-out pulse identifies the information on the I/O data lines as a data word. After the instruction has been executed the I/O device is automatically disconnected from the I/O bus. (See paragraph 4.3 for a description of I/O operations.)

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link and I/O Test are not affected.

Qualified Registers: B.

DIB DATA INTO B-REGISTER

2 cy
4.32 μs



The Data Into B-Register instruction causes data to be transferred into the B-register from an I/O device connected to the parallel I/O system. The data word on the parallel I/O data lines is logically ANDED with a mask word previously loaded into the B-register, and the logical product replaces the contents of the B-register.

This instruction must be preceded by a Function-Address Out from B-Register instruction with its corresponding function-address word specifying the data input mode. Executing the instruction causes a 720-ns pulse to be transmitted to the I/O device on the transfer-in pulse line (TIP>-I). The transfer-in pulse indicates that the data on the I/O data lines has been sampled and transferred into the B-register. After the instruction has been executed, the I/O device is automatically disconnected from the I/O bus. (See paragraph 4.3 for a description of I/O operations.)

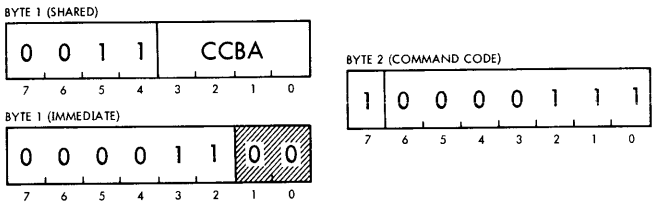
$$(B)_{0-11} \cap (\text{I/O data lines}) \longrightarrow (B)_{0-11}$$

Indicators: Zero and Plus record the adder result of the input operation. Link and I/O Test are not affected.

Qualified Registers: B

INE INTERRUPT ENABLE

2 or 3 cy
4.32 or 6.48 μs



The Interrupt Enable instruction interrogates for the presence of a priority interrupt request (from an external source or from the relative time clock). If no interrupt request is present, program execution continues in normal sequence. If an interrupt request is present, program control is transferred to a dedicated memory location (0010_8 for external interrupt requests; 0012_8 for relative time clock interrupt requests). Normally, jump instructions would be stored in these locations to transfer control to interrupt servicing routines. The relative time clock interrupt request is automatically cleared after the interrupt has been serviced. The external interrupt request is cleared after program acknowledgment.

If both an external interrupt request and a relative time clock interrupt request are present simultaneously, the relative time clock interrupt takes priority and is processed first.

No interrupt request present (2 cy):

$$(P)_{0-11} + 1 \longrightarrow (P)_{0-11}$$

External interrupt request present (3 cy):

$$(P)_{0-11} \longrightarrow (B)_{0-11}$$

$$0010_8 \longrightarrow (P)_{0-11}$$

Relative time clock interrupt request present (3 cy):

$$(P)_{0-11} \longrightarrow (B)_{0-11}$$

$$0012_8 \longrightarrow (P)_{0-11}$$

The recommended method for returning control from an interrupt servicing routine to the main program is a register-to-register transfer of the contents of the B-register to the P-register (RTR B,P).

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link and I/O Test are not affected.

Qualified Registers: None

3.8 AUGMENTED MEMORY ADDRESSING INSTRUCTIONS

Instructions of the augmented memory addressing group provide powerful memory operand addressing for the most basic operations: load, add, and store. The versatile addressing capability of these instructions permits eight variations of operand addressing to be performed:

a. Direct addressing

b. Indexed addressing

c. Indirect addressing

d. Indirect addressing with preindexing

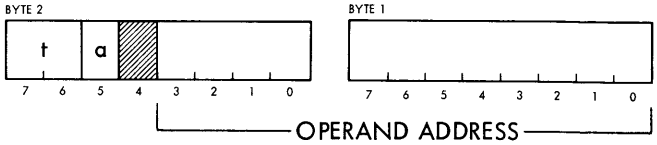
e. Indirect addressing with postindexing

f. Indirect addressing with preindexing and postindexing

g. Indirect addressing with postindexing and autoincrementing

h. Indirect addressing with preindexing, postindexing, and autoincrementing

The six indirect addressing variations of the augmented memory addressing instructions require, in addition to the instruction itself, a two-byte indirect address word with the following format:



Byte 2 of the indirect address word designates whether postindexing and autoincrementing are to be performed as follows:

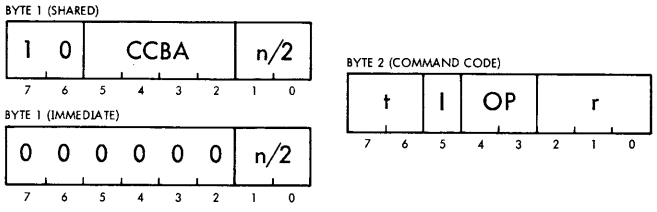
Bits	Description
6-7 (t)	Specify whether postindexing is to be performed and indicate index register to be used

7 6	Index Register
0 0	None
0 1	X-register
1 0	Y-register
1 1	Z-register

5 (a)	If autoincrementing is to be performed, bit 5 is a 1; if not, it is 0.

A description of how the effective operand address is formed for augmented memory addressing instructions is given in paragraph 3.9

Augmented memory addressing instructions have the following double-word format:



Byte 1 of the instruction identifies the instruction as belonging to the augmented memory addressing group and designates either immediate or shared command addressing. If immediate command addressing is designated, byte 2 of the instruction is taken from the location immediately following byte 1. If shared

command addressing is designated, byte 2 is taken from 1 of 16 memory locations (0100_8 through 0117_8) shared by the augmented memory addressing instructions. The relative location of byte 2 in the shared memory pool is designated by bits 2 through 5 of byte 1.

Byte 2 of the instruction designates the selected register and specifies the operation and type of operand addressing to be performed as follows:

Bits	Description
0-2 (r)	Designate selected register
3-4 (OP)	Specify operation to be performed

4 3	Operation
0 0	Load
0 1	Store and zero
1 0	Add
1 1	Store

5 (I)	Specifies whether indirect addressing is to be performed. If it is, bit 5 is a 1; it not it is 0.
6-7 (t)	Specify whether indexing is to be performed and indicate the index register to be used

7 6	Index Register
0 0	None
0 1	X-register
1 0	Y-register
1 1	Z-register

3.9 FORMING THE EFFECTIVE OPERAND ADDRESS

Table 3-1 shows the various ways of forming the effective operand address for the augmented memory addressing instructions. A flow chart of the computer operations for forming the effective operand address is shown in figure 3-1. An important point to note in figure 3-1 is that the effective operand address is assembled in the B-register; therefore, the effective operand address appears in the B-register following instruction execution unless it is the selected register.

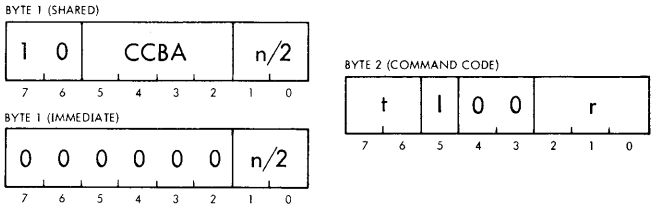
Table 3-1. Effective Operand Address for Augmented Memory Addressing

Addressing Type	Instruction Word Fields		Indirect Address Word Fields		Description
	t	I	t	a	
Direct	0	0			Effective operand address is equal to value of n ($n = 0_8, 2_8, 4_8$, or 6_8)
Indexed	1, 2, or 3	0			Effective operand address is equal to value of n plus contents of specified index register
Indirect	0	1	0	0	Location of indirect address is specified by n. Effective operand address is equal to value of operand address field of indirect address word
Indirect with preindexing	1, 2, or 3	1	0	0	Location of indirect address word is determined by adding contents of preindex register to value of n. Effective operand address is equal to value of operand address field of indirect address word
Indirect with postindexing	0	1	1, 2, or 3	0	Location of indirect address word is specified by n. Effective operand address is equal to value of operand address field of indirect address word plus contents of postindex register

Table 3-1. Effective Operand Address for Augmented Memory Addressing (Cont.)

Addressing Type	Instruction Word Fields		Indirect Address Word Fields		Description
	t	I	t	a	
Indirect with preindexing and post-indexing	1, 2, or 3	1	1, 2, or 3	0	Location of indirect address word is determined by adding contents of preindex register to value of n. Effective operand address is equal to value of operand address field of indirect address word plus contents of postindex register
Indirect with postindexing and auto-incrementing	0	1	1, 2, or 3	1	Location of indirect address word is specified by n. Effective operand address is equal to value of operand address field of indirect address word plus contents of postindex register after being incremented by 1. Post-index register contains value as incremented following instruction execution
Indirect with preindexing, postindexing, and auto-incrementing	1, 2, or 3	1	1, 2, or 3	1	Location of indirect address word is determined by adding contents of preindex register to value of n. Effective operand address is equal to value of operand address field of indirect address word plus contents of post-index register after being incremented by 1. Postindex register contains value as incremented following instruction execution

GnL r,t,I **AUGMENTED LOAD** **3 cy[1] 6.48 μs**



$(EOA)_{0-7} \longrightarrow (r)_{0-7}$

$0's \longrightarrow (r)_{8-11}$

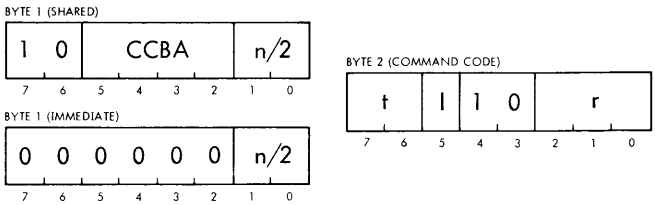
$EOA \longrightarrow (B)_{0-11}$ If B is not selected register

Indicators: Zero and Plus record adder result of load operation. Link is not affected.

Qualified Registers: A, X, Y, Z, P, B, YY, or ZZ

The Augmented Load instruction replaces the contents of bits 0 through 7 of the selected register (r) with the contents of the effective operand address. Zeros replace the contents of bits 8 through 11 of the selected register. The contents of the effective operand address remain unchanged. The effective operand address replaces the contents of the B-register if the selected register is not the B-register.

GnA r,t,I **AUGMENTED ADD** **3 cy[1] 6.48 μs**



The Augmented Add instruction algebraically adds the contents of the effective operand address to the contents of the selected register (r). The sum replaces the contents of the selected register. The effective

[1] 5 cy, 10.8 μs, for indirect addressing.

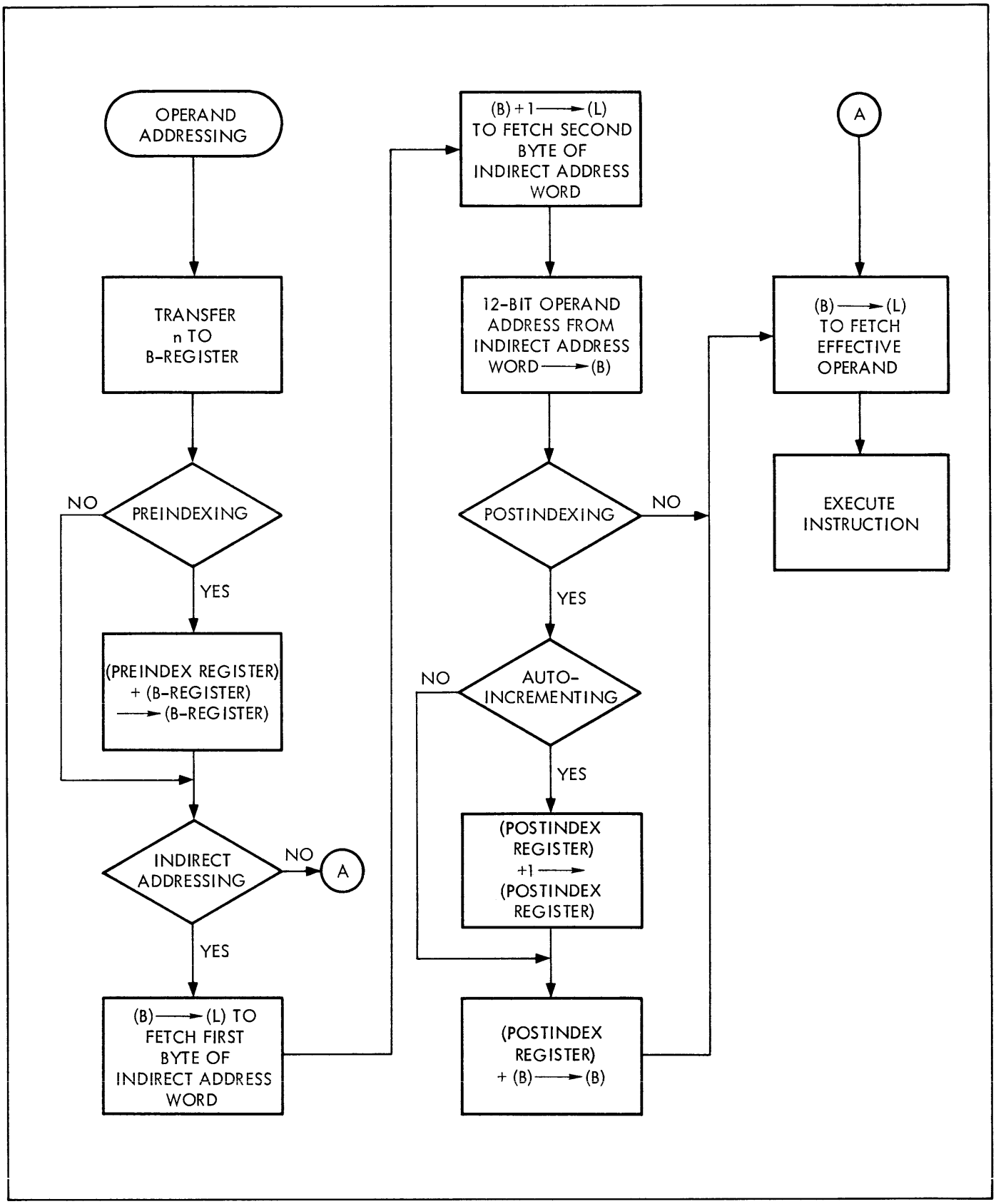


Figure 3-1. Effective Operand Addressing, Flow Diagram

operand address replaces the contents of the B-register. Contents of the effective operand address remain unchanged.

$$(EOA)_{0-7} + (r)_{0-11} \longrightarrow (r)_{0-11}$$

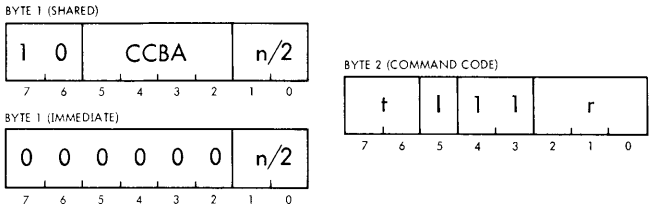
$$EOA \longrightarrow (B)_{0-11}$$

Indicators: Zero and Plus record adder result of addition. Link is set if carry occurs from bit position 7 of adder; if no carry occurs, Link is reset.

Qualified Registers: A, X, Y, Z, P, YY, or ZZ

GnS r,t,l AUGMENTED STORE

3 cy[1]
6.48 μs



The Augmented Store instruction replaces the contents of the effective operand address with the contents of bits 0 through 7 of the selected register (r). The contents of the selected register remain unchanged. The effective operand address replaces the contents of the B-register.

$$(r)_{0-7} \longrightarrow (EOA)_{0-7}$$

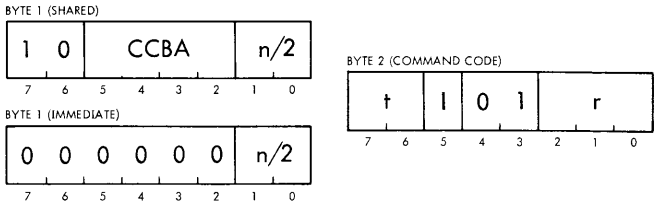
$$EOA \longrightarrow (B)_{0-11}$$

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: A, X, Y, Z, P, YY, or ZZ.

GnZ r,t,l AUGMENTED STORE AND ZERO

3 cy[1]
6.48 μs



The Augmented Store and Zero instruction replaces the contents of the effective operand address with the contents of bits 0 through 7 of the selected register (r), and replaces the contents of the selected register with zeros. If the selected register is the B-register, zeros replace both the contents of the effective operand address and the B-register. The effective operand address replaces the contents of the B-register if the selected register is not the B-register.

If selected register is not B:

$$(r)_{0-7} \longrightarrow (EOA)_{0-7}$$

$$0's \longrightarrow (r)_{0-11}$$

$$EOA \longrightarrow (B)_{0-11}$$

If selected register is B:

$$0's \longrightarrow (EOA)_{0-7}$$

$$0's \longrightarrow (B)_{0-11}$$

Indicators: Zero and Plus are set to indicate ZERO and PLUS. Link is not affected.

Qualified Registers: A, X, Y, Z, P, B, YY or ZZ

[1] 5 cy, 10.8 μs, for indirect addressing.

SECTION IV
INPUT/OUTPUT OPERATIONS

4.1 GENERAL

The versatile and flexible input/output capability of the SPC-12 permits it to communicate with external devices through either of two independent I/O systems: the serial I/O system or the parallel I/O system. Both systems are characterized by simplicity of operation: Only two instructions are required to perform serial I/O operations. Only three instructions are required to perform parallel I/O operations.

Both systems have the flexibility to satisfy the different requirements of general purpose and special control applications, yet their inherent simplicity enhances SPC-12 reliability, maintainability, and ease of use.

4.2 SERIAL INPUT/OUTPUT OPERATIONS

Serial I/O operations are carried out between the computer and external devices through the serial I/O bus (figure 4-1). The serial I/O bus is a three-wire, full duplex, telegraph grade line with a transfer rate of up to 150 bits per second. Operating power for the line (48V dc, 20 ma) is supplied by the external device, which can be located as far away as 2000 feet. The bus typically connects to a Teletype unit, and more than one unit can be connected if the addressing capability provided within the Teletype units is used.

To perform a serial input operation, either a Shift Serial Data In or a Shift Serial Data In and Out instruction is used. These instructions sample the serial data input channel and shift the sampled

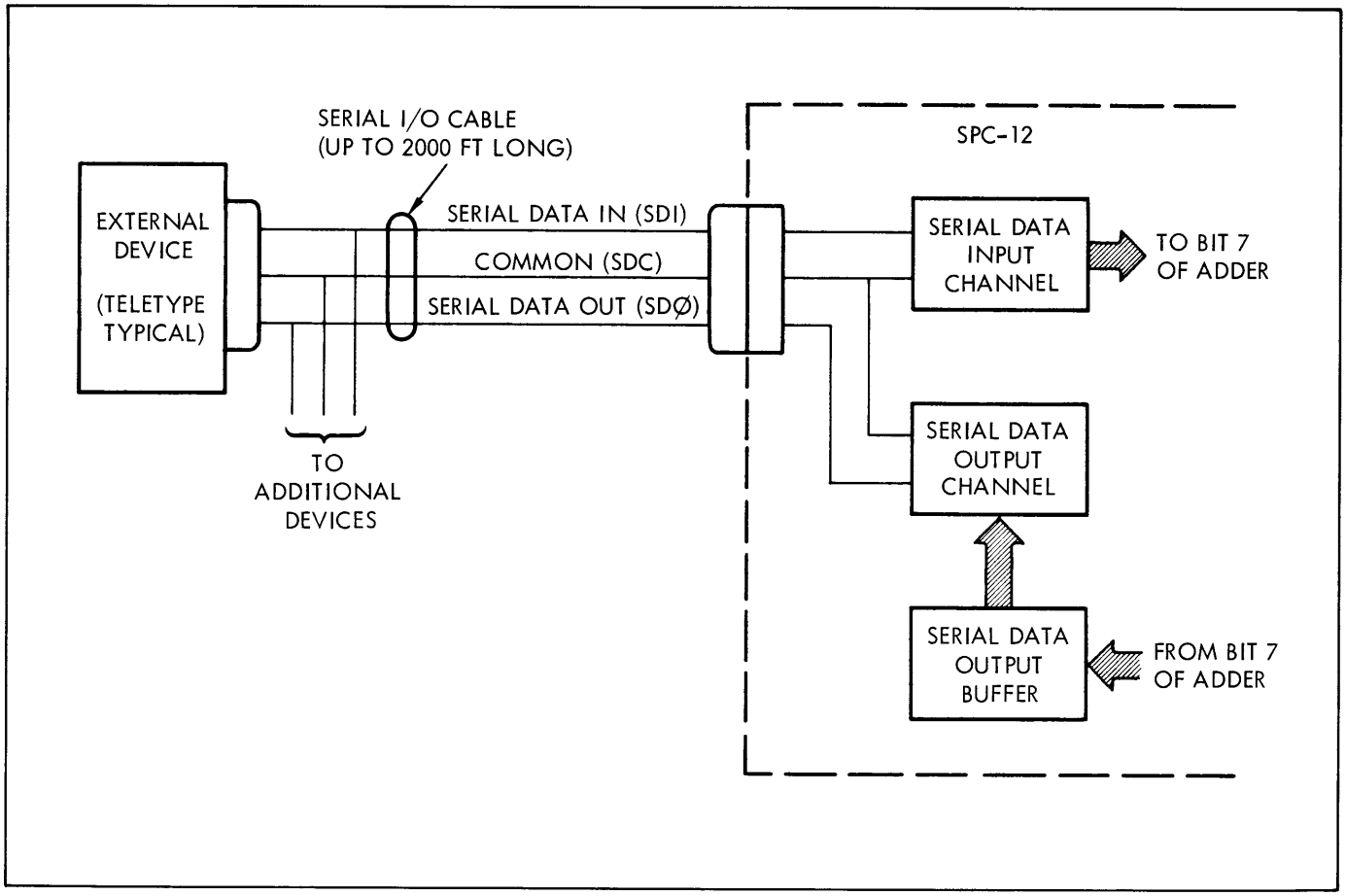


Figure 4-1. Serial I/O System, Functional Diagram

value into bit 7 of a selected register (A, X, Y, or Z). Since the adder is used to perform these instructions, the value of the sampled bit can be found by testing the Plus indicator after the input operation. When the Shift Serial Data In and Out instruction is used, sampled data is not only shifted into the selected register, but is also shifted into the serial data output buffer.

To perform a serial output operation either a Shift Serial Data In and Out, a Shift Right and Serial Data Out, a Shift Circular and Serial Data Out, or a Shift Circular through Link and Serial Data Out instruction is used. The SHRO instruction shifts zeros into the serial data output buffer. The SHCO instruction shifts bit 0 of a selected register (A, X, Y, or Z) into the serial data output buffer. The SHLO instruction shifts the bit stored in the Link indicator into the serial data output buffer. When any of these instructions are used, the value of the bit output can be found by testing the Plus indicator.

For a complete description of the serial I/O shift instructions, refer to the instruction descriptions in section III.

4.3 PARALLEL INPUT/OUTPUT OPERATIONS

Parallel I/O operations are carried out between the computer and external devices through the parallel I/O system completely independent of serial I/O operations. The parallel I/O system minimizes the need for external hardware. Three instructions provide maximum control of the I/O system without sacrificing speed or efficiency: One instruction (common to both input and output operations) addresses the I/O device and defines the function to be performed. One instruction transfers data from the I/O device to the computer. And one instruction transfers data from the computer to the I/O device. A complete description of the parallel I/O instructions is given in paragraph 3.7.

4.4 PARALLEL I/O CABLE

The parallel I/O cable carries data, address, and control signals between the computer and I/O devices connected to it. A simple I/O system can be structured by connecting I/O device controllers to the parallel I/O cable in partyline fashion as shown in figure 4-2. A more flexible I/O system can be built around standard processor interface units as described in paragraph 4.19. Whichever type system is used, the basic I/O operations described in the following paragraphs apply to both.

The I/O cable contains 12 bidirectional I/O data lines and 7 unidirectional control lines. The I/O data lines transmit control codes, addresses, and data between the computer and I/O device controllers. The control lines transmit timing signals to and from the computer to synchronize the information transfers over the data lines.

4.5 I/O DATA LINES

The I/O data lines are used in three different modes to perform three different functions:

a. To transfer address and control information to external devices in the control signal mode

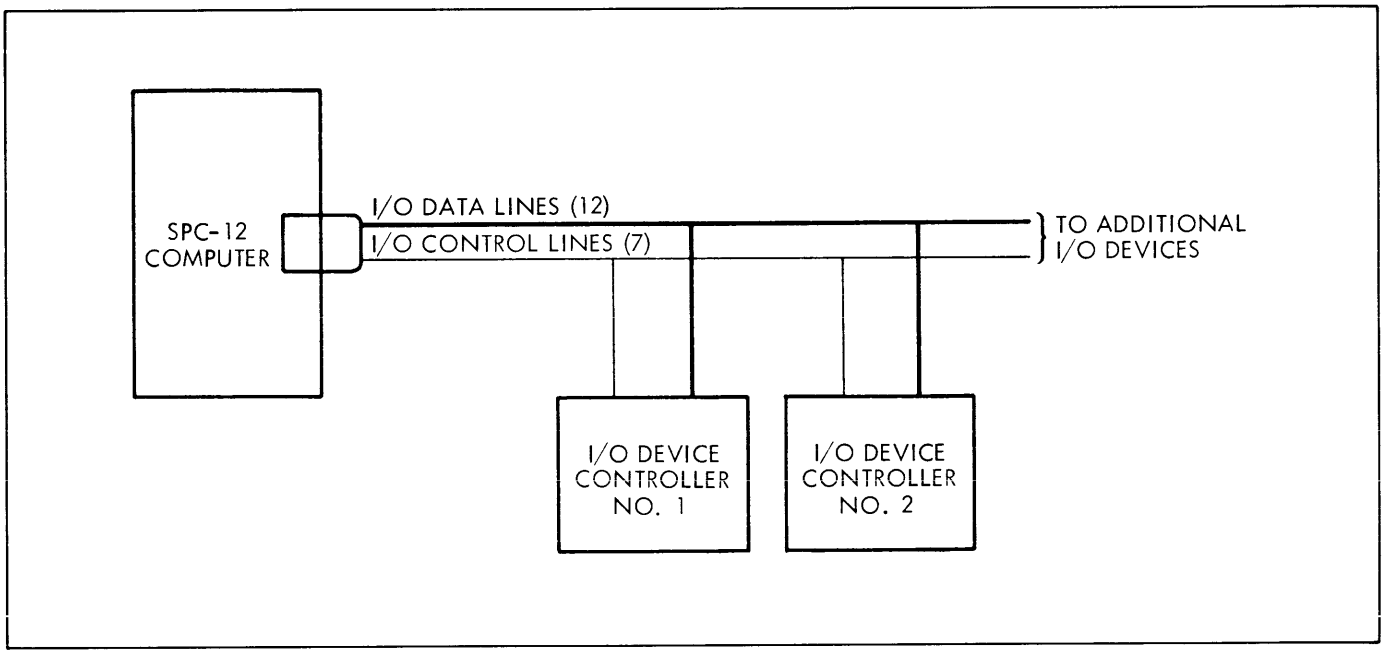


Figure 4-2. Parallel I/O Cable Connections

b. To transfer data from an external device to the computer in the data input mode

c. To transfer data from the computer to an external device in the data output mode

As shown in figure 4-3, input data from the I/O data lines is processed through the input receivers onto the addend bus. From the addend bus it is transferred through the adder into the B-register. Once in the B-register it can be manipulated or stored in memory. Data that is to be transferred to an external device must first be loaded into the B-register. From the B-register it is processed through the output drivers and onto the I/O data lines. The B-register, then, is used as a buffer register between the computer and external devices for all input and output data transfers.

4.6 FUNCTION-ADDRESS PULSE LINE

The function-address pulse line carries a signal (720-ns pulse) from the computer to the external device to indicate that the information on the I/O data lines is to be interpreted as a function-address word (paragraph 4.13). The function-address pulse is generated by the execution of the FOB instruction.

4.7 TRANSFER-IN PULSE LINE

The transfer-in pulse line carries a signal (720-ns pulse) from the computer to the I/O device to indicate that the input data on the I/O data lines has been sampled and transferred into the B-register. The trailing edge of the signal can be used by the I/O device to remove the data from the lines. The transfer-in pulse is generated by the execution of the DIB instruction.

4.8 TRANSFER-OUT PULSE LINE

The transfer-out pulse line carries a signal (720-ns pulse) from the computer to the I/O device to indicate that the information on the I/O data lines (contents of the B-register) is to be interpreted as an output data word for the I/O device. The transfer-out pulse is generated by the execution of the DOB instruction.

4.9 I/O TEST LINE

The I/O test line carries a signal from the I/O device to the computer in response to a request for status from the computer. The signal indicates whether the status of the condition specified by the

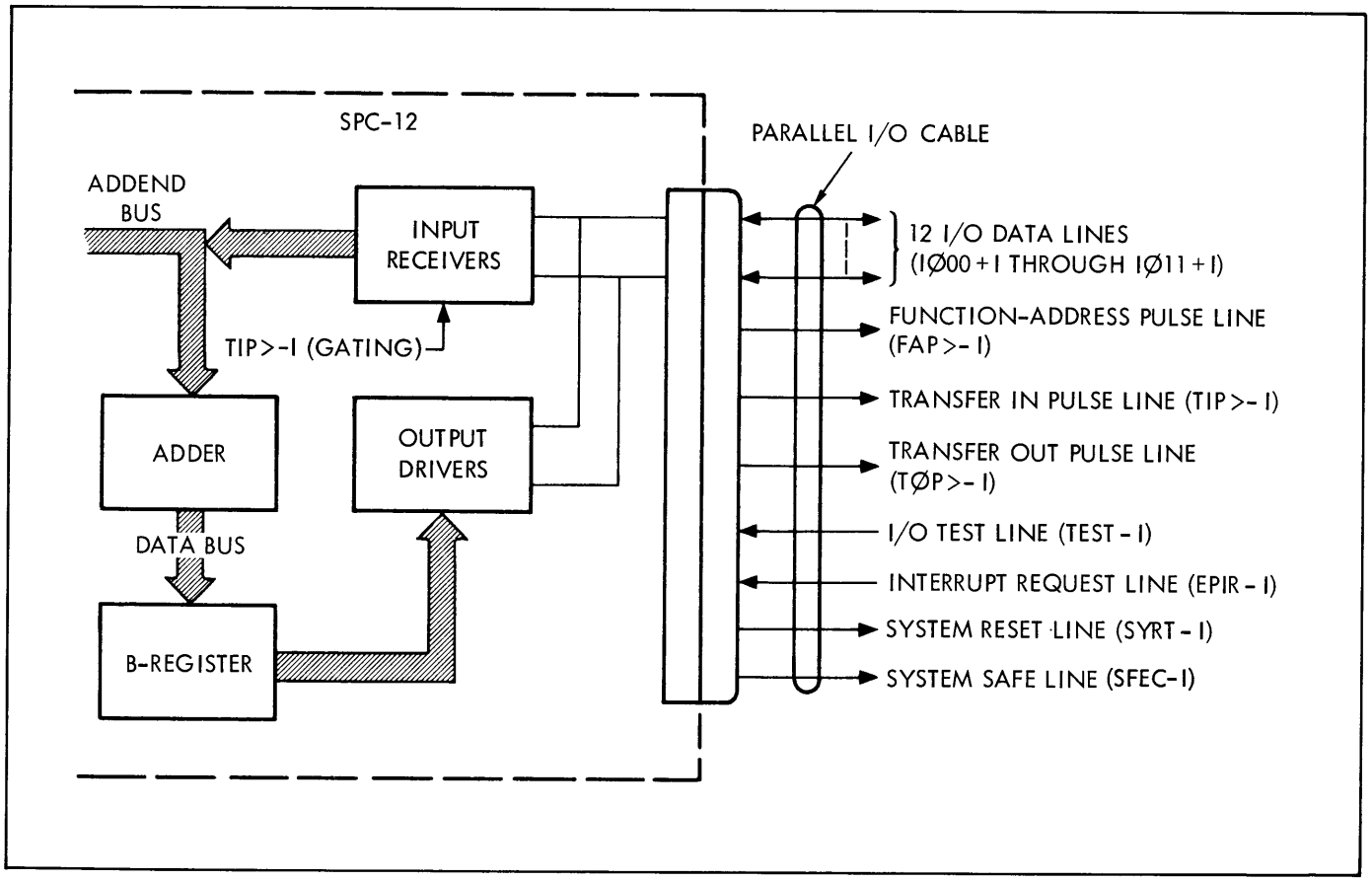


Figure 4-3. Parallel I/O System, Functional Diagram

computer is true or false. The status of this line is stored in the I/O Test indicator at each execution of the FOB instruction. A Skip if I/O Test True or Skip if I/O Test False instruction can be used to examine the I/O Test indicator.

4.10 INTERRUPT REQUEST LINE

The interrupt request line carries a signal from one or more I/O devices to request a program interrupt. The status of this line can be interrogated by an Interrupt Enable instruction to transfer program control to a servicing routine. The interrupt request signal is removed by the program servicing the I/O device.

4.11 SYSTEM RESET LINE

The system reset line carries a signal from the computer to all external devices to establish initial operation conditions. The signal can be generated manually from the programmer's console by pressing the SYS RST switch, or it can be generated automatically by the automatic restart feature during a restart sequence. When the system reset line is true, the I/O data lines contain logic zeros.

4.12 SYSTEM SAFE LINE

The system safe line carries a signal to indicate that the computer is operating in run mode. The signal is removed from the line if the computer is switched to idle mode either manually, as a result of a power failure, or by activation of the stall alarm.

Table 4-1 lists the specifications for the parallel I/O cable, and figure 4-4 shows the electrical characteristics for the I/O lines.

Table 4-1. Parallel I/O Bus General Specifications

Parameter	Specification
Signal Levels	
I/O data lines	Logical 1 = +5V, logical 0 = 0V
Control lines	Logical 1 = 0V logical 0 = +5V
Noise Immunity Features	+5V levels
	Drive current (for termination and load), 125 ma max at 0°C
	Twisted-pair cabling
	Wide-margin timing
Expansion (I/O bus device loading)	
I/O data lines	100 ma max at 0°C
Control lines	75 ma max at 0°C

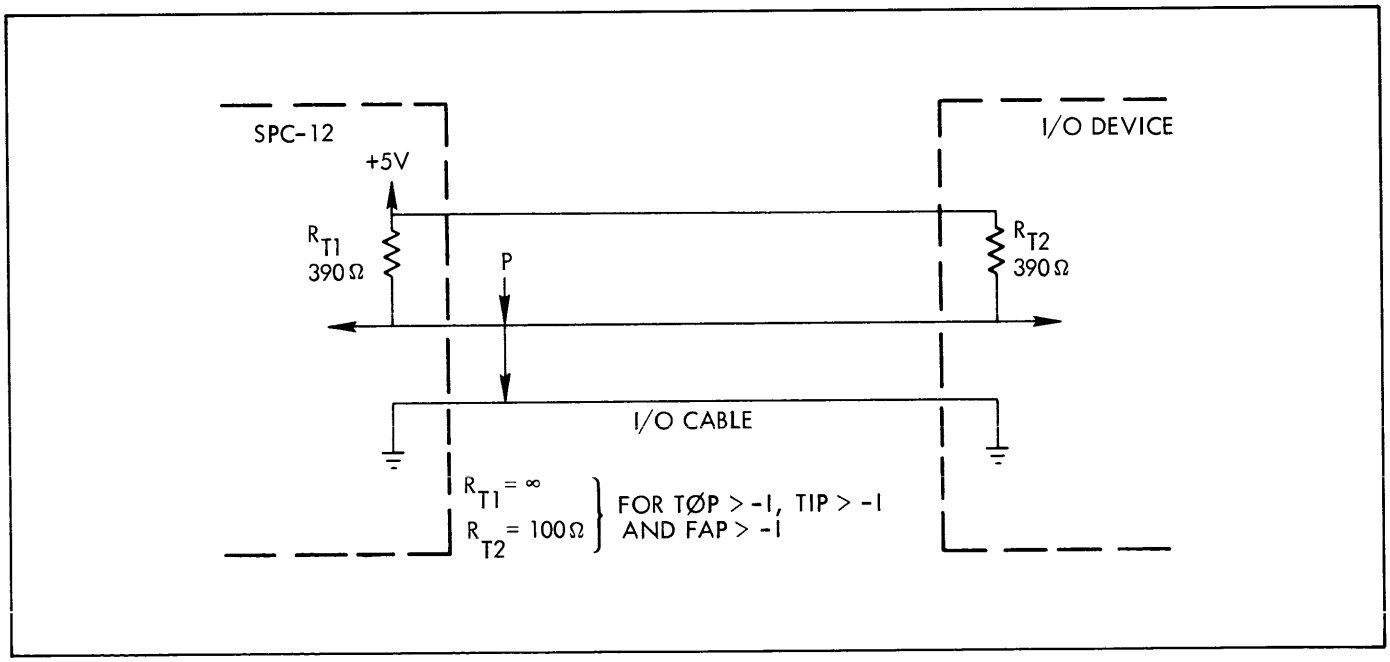


Figure 4-4. Parallel I/O Cable, Electrical Characteristics

4.13 I/O ADDRESSING AND BASIC TIMING

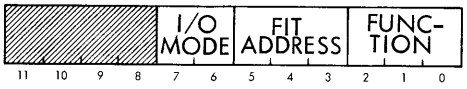
All I/O operations are either one or two phase; I/O test and control operations are single phase while data transfer operations are two phase. The two phases are designated the function-address phase and the data phase (figure 4-5). Each phase is terminated with a control pulse. During the function-address phase the computer transmits an eight-bit coded word, called the function-address word, to the I/O device or functional interface translator on the I/O data lines. (The functional interface translator is one of several standard interface units that can be used with the parallel I/O system; it is described in paragraph 4.19.) Accompanying the function-address word is a function-address pulse generated as a result of executing a FOB instruction. The function-address word specifies the function to be performed and addresses the device that is to perform the function.

During the data phase the actual transfer of data between the I/O device and the computer takes place. Data is transferred from the device to the computer when the computer generates a transfer data-in pulse as a result of executing a DIB instruction. Data is transferred from the computer to the device when the computer generates a transfer data-out pulse as a result of executing a DOB instruction.

4.14 FUNCTION-ADDRESS WORD

The function-address word, along with the control pulses generated by the FOB, DIB, and DOB instructions, is all that is required to perform an I/O operation. Although the 12 bits of the function-address word may be encoded in any way required for use by an I/O device, a standard format has been adopted for addressing and controlling the standard I/O devices available with the computer. (NOTE: To maintain maximum system efficiency and flexibility, no fixed address or function assignments have been made for standard I/O devices. Assignments are made for each system individually and are specified in the documentation supplied with each system.)

The following format is used for the function-address word:



Bits	Description
8-11	Not used.
6-7	I/O Mode. This field specifies one of four modes in which the computer and I/O device are to communicate.

Octal Code	I/O Mode
0	Specifies control signal mode. Computer sends a control pulse to the I/O device
1	Specifies data output mode. Must be followed by a DOB instruction

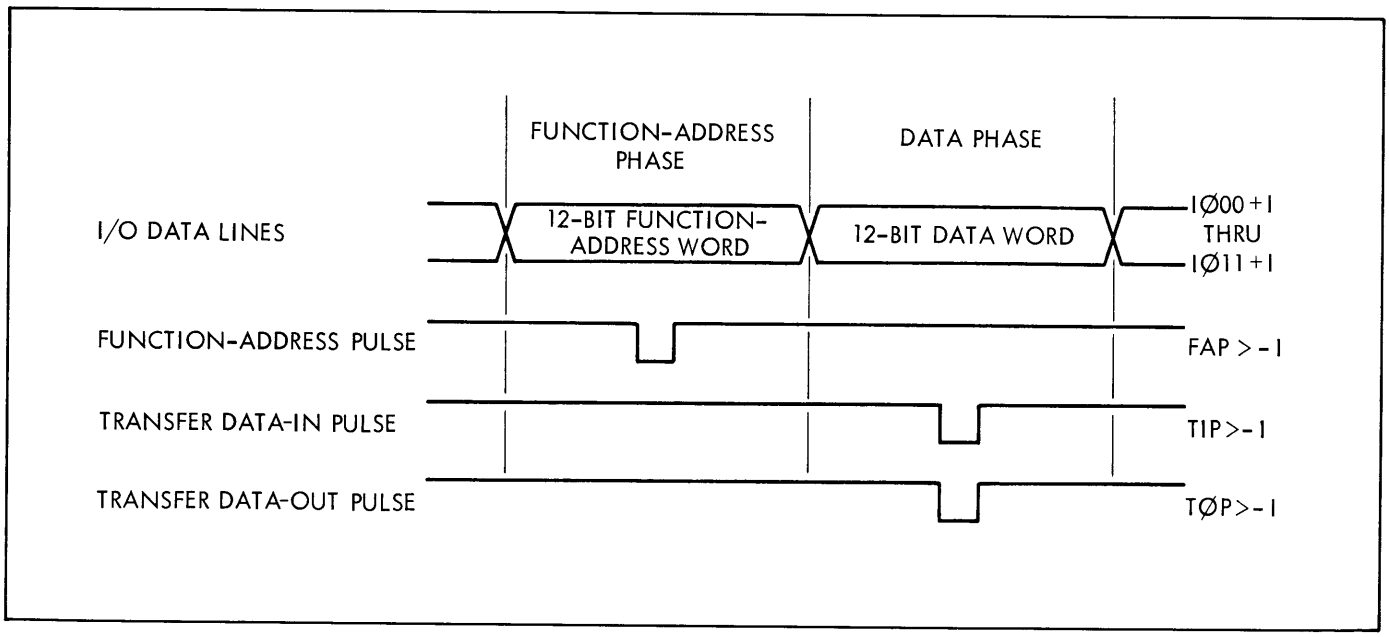


Figure 4-5. Parallel I/O System, Basic Timing

Bits	Description	
	Octal Code	I/O Mode
	2	Specifies data input mode. Must be followed by a DIB instruction
	3	Specifies test mode. Computer requests status condition from I/O device
3-5	Functional Interface Translator Address. This field specifies which one of eight I/O devices or functional interface translators is being addressed. (Note that I/O devices are addressed indirectly by addressing the functional interface translator to which the device is connected.	
0-2	Function. When I/O mode field designates control signal mode, this field specifies which one of eight control lines is to output a pulse.	
	When I/O mode field designates data output mode, this field, along with the address field, determines which I/O device is to receive the next data word transmitted by the computer.	
	When I/O mode field designates data input mode, this field, along with the address field, determines which I/O device is to send the next data word to the computer.	
	When I/O mode field designates I/O test mode, this field specifies which one of eight status lines is to be sampled by the computer.	

To start any I/O operation, a function-address word is first loaded into the B-register, and then a FOB instruction is executed. The function-address word is sent on the I/O data lines to all I/O devices or functional interface translators attached to the parallel I/O bus. The function-address pulse generated by the FOB instruction is also sent to all I/O devices or functional interface translators. The I/O device controller or functional interface translator addressed by the function-address word decodes the word and generates the signals required to perform the operation specified in the I/O mode and function fields. Paragraphs 4.15 through 4.18 describe the four basic I/O operations.

4.15 I/O TEST OPERATION

Figure 4-6 shows a typical program sequence and timing for testing the status of an I/O device. The sequence is begun by loading the B-register with a function-address word specifying the following:

a. I/O test mode

b. Address of the I/O device or functional interface translator to which the I/O device is connected

c. Function (status line) to be tested

Executing a FOB instruction then causes a function-address pulse to be generated. The function-address pulse is used to sample the selected status line and to store the sampled value in the I/O Test indicator of the computer. The status of the I/O Test indicator is tested by executing either a Skip if I/O Test True (SKT) or a Skip if I/O Test False (SKF) instruction.

4.16 I/O CONTROL OPERATION

Figure 4-7 shows a typical program sequence for sending a control pulse to an I/O device. The sequence is begun by loading the B-register with a function-address word specifying the following:

a. Control signal mode

b. Address of the I/O device or functional interface translator to which the I/O device is connected.

c. Function (control line) to be pulsed

Executing a FOB instruction then causes a function-address pulse to be generated by the computer. This pulse is sent to all I/O devices or functional interface translators, but only the one addressed by the function-address word directs the pulse to the specified control line of the I/O device.

4.17 DATA INPUT OPERATION

Figure 4-8 shows a typical program sequence and timing for transferring data from an I/O device to the computer. The sequence is begun by loading the B-register with a function-address word specifying the following:

a. Data input mode

b. Address of I/O device or functional interface translator to which the I/O device is connected

c. Function (I/O device source) from which data is to be transferred

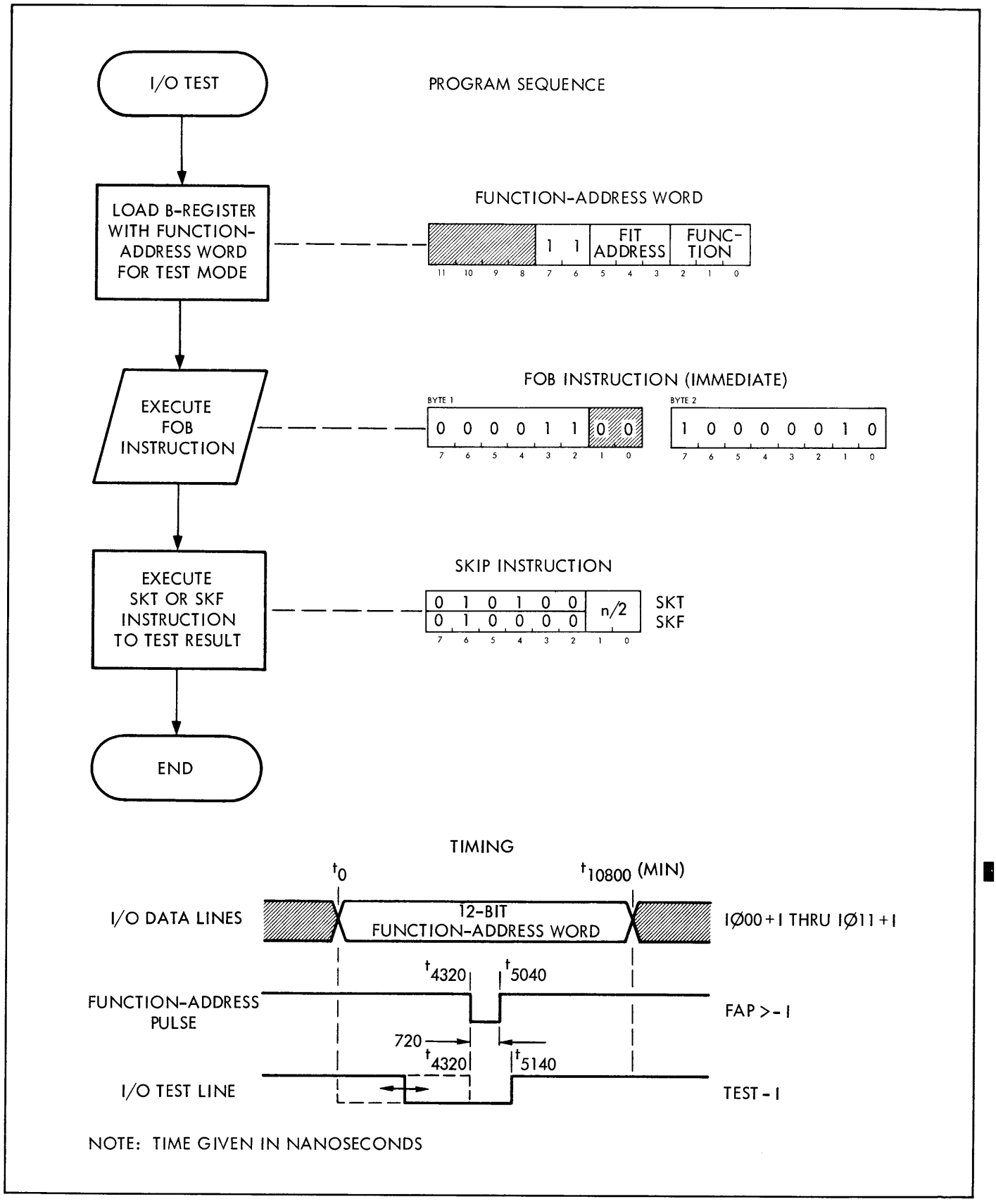


Figure 4-6. Parallel I/O Test Operation

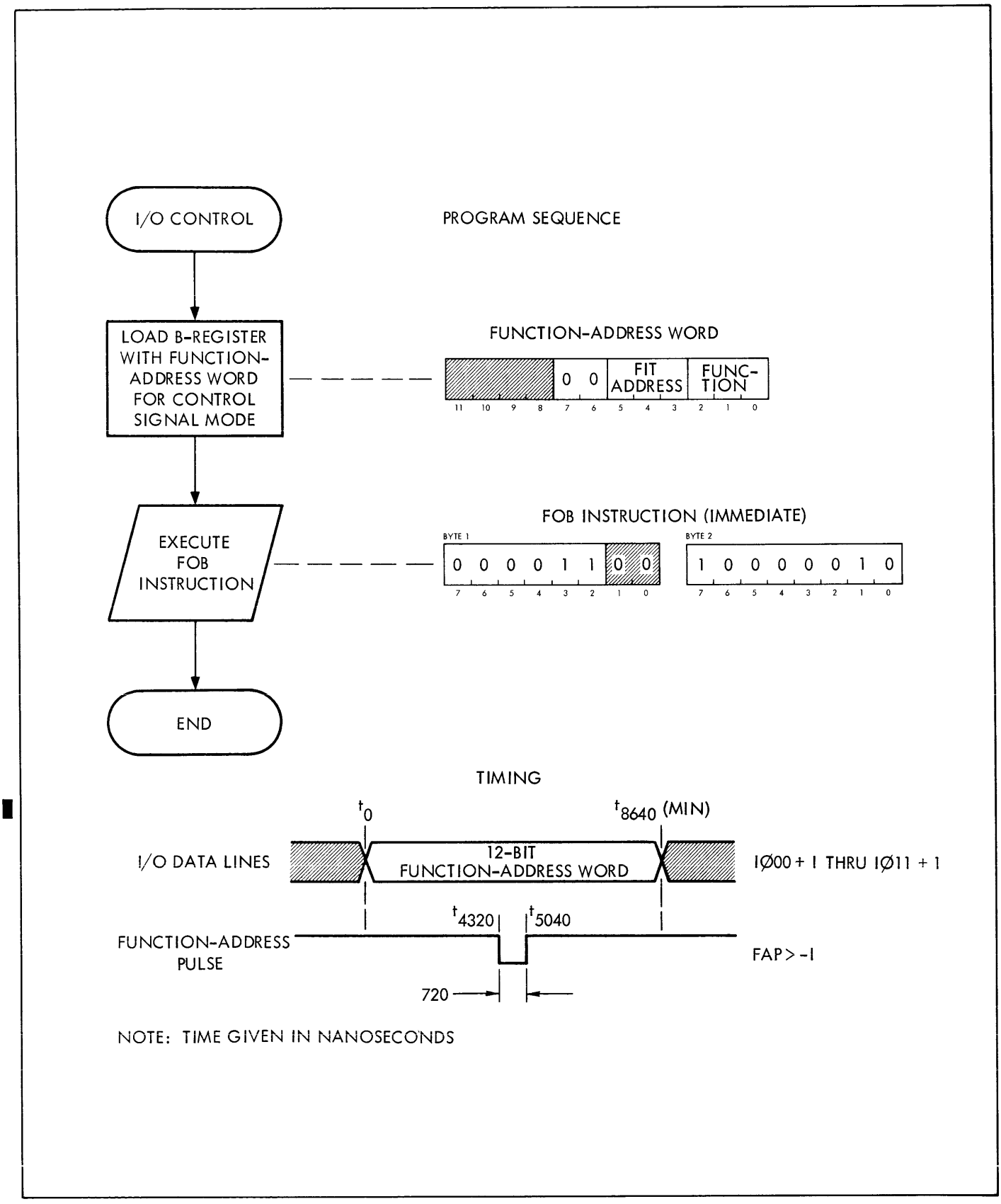


Figure 4-7. Parallel I/O Control Operation

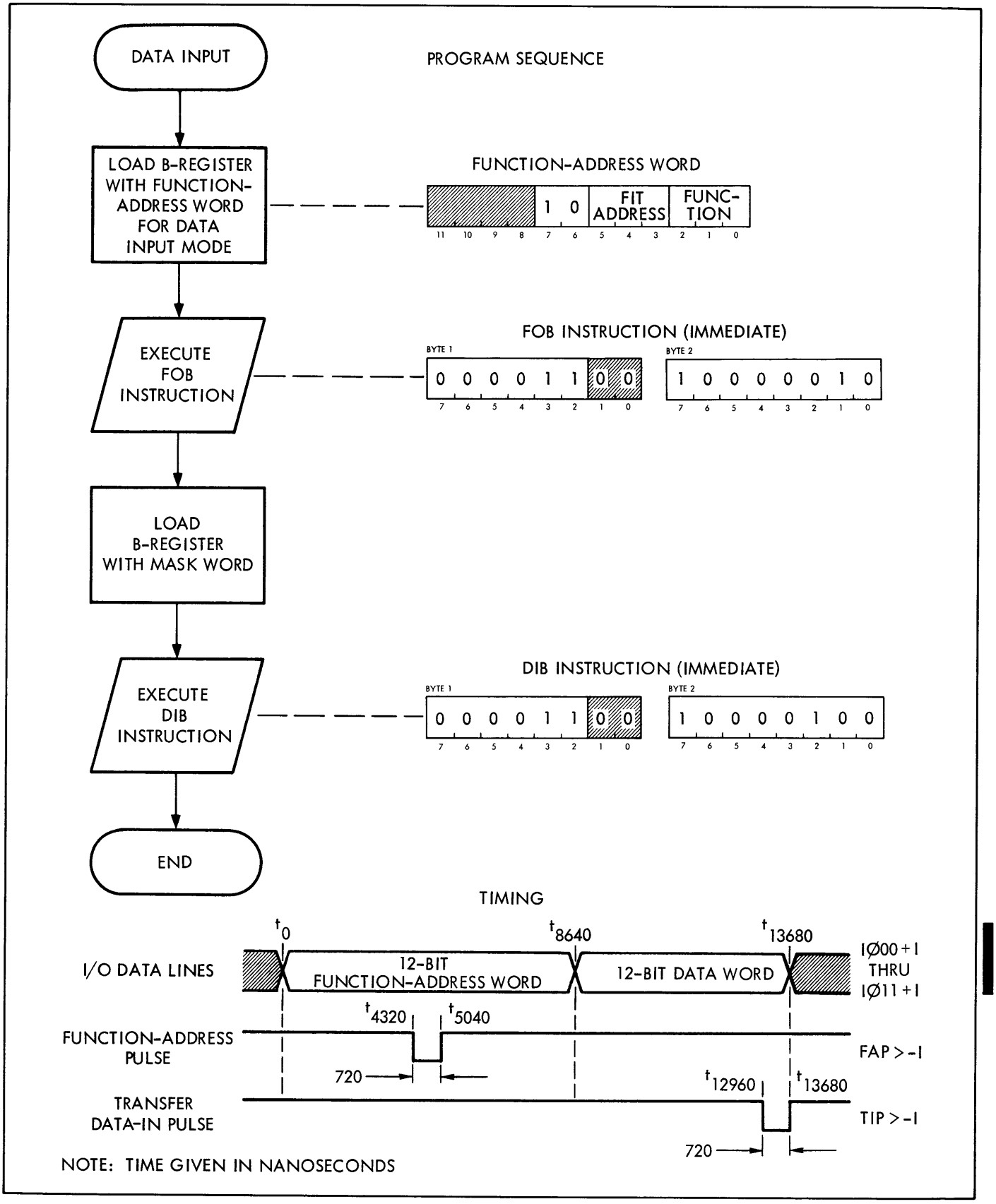


Figure 4-8. Parallel I/O Data Input Operation

Executing a FOB instruction then causes a function-address pulse to be generated by the computer. The addressed I/O device controller or functional interface translator decodes the function-address word, and, on receiving the function-address pulse, connects the I/O device for service. Before the data transfer can be made, the B-register must be loaded with a mask word.

To transfer the data from the I/O device to the computer, a transfer data-in pulse is generated by executing a DIB instruction. The pulse causes the logical product of the data on the I/O data lines and the mask word to replace the contents of the B-register. The trailing edge of the pulse disconnects the I/O device. (NOTE: The computer is not dependent on the successful transfer of data from the I/O device in order to complete the operation.)

4.18 DATA OUTPUT OPERATION

Figure 4-9 shows a typical program sequence and timing for transferring data from the computer to an I/O device. The sequence is begun by loading the B-register with a function-address word specifying the following:

a. Data output mode

b. Address of I/O device or functional interface translator to which the I/O device is connected

c. Function (I/O device destination) into which data is to be transferred

Executing a FOB instruction then causes a function-address pulse to be generated by the computer. The addressed I/O device controller or functional interface translator decodes the function-address word, and, on receiving the function-address pulse, connects the I/O device for service. The function-address pulse sets the device in a state ready to receive a data word from the computer on the I/O data lines. At this time a direct path exists between the B-register and the selected data register in the I/O device.

After the data word that is to be transferred is loaded into the B-register, a DOB instruction is executed to generate a transfer data-out pulse. The pulse gates the data from the I/O data lines into the I/O device. The trailing edge of the pulse disconnects the device. (NOTE: The I/O device need not generate any signal to acknowledge that data transmission has been completed.)

4.19 I/O INTERFACING

The parallel I/O system provides flexibility through the single set of data and control lines to interface readily with a wide variety of external devices. The 12 I/O data lines permit efficient handling of 8-, 12-, 16-, and 24-bit data transfers. The versatility of the I/O system permits extended use of data channels with unidirectional devices. One I/O channel can be separated into an input channel for data transfers to an input-only device, and an output channel for data transfers to an output-only device.

The computer need not be modified when new I/O devices or interfaces are added. Additional devices are connected simply by extending the I/O bus from one device to the next. Enough drive current is available to supply the maximum configuration of attached I/O devices.

Although I/O devices can be connected to the parallel I/O bus in a variety of ways, standard devices are normally connected through two processor system interface units: Cable Interface Translator PSIU-02 and Functional Interface Translator PSIU-01. These interface units permit the parallel I/O system to be used with maximum efficiency. Figure 4-10 shows a typical parallel I/O configuration using processor system interface units.

4.20 CABLE INTERFACE TRANSLATOR

The cable interface translator (figure 4-11) is a buffer between the parallel I/O bus and the functional interface translators. It contains cable drivers, receivers, and terminators. Four cable interface units are the maximum that ever need be connected to the parallel I/O bus, and each cable interface translator can interface with two functional interface translators.

4.21 FUNCTIONAL INTERFACE TRANSLATOR

The functional interface translator (figure 4-12) translates the information output by the computer in the form of the function-address word on the I/O data lines into operating signals directly usable by the I/O device controllers connected to it. Eight functional interface translators are the maximum that need be used. The number of I/O devices that can be connected to each functional interface translator is limited only by the number of functions that are to be controlled in the device.

Functional specifications for the functional interface translator are given in table 4-2.

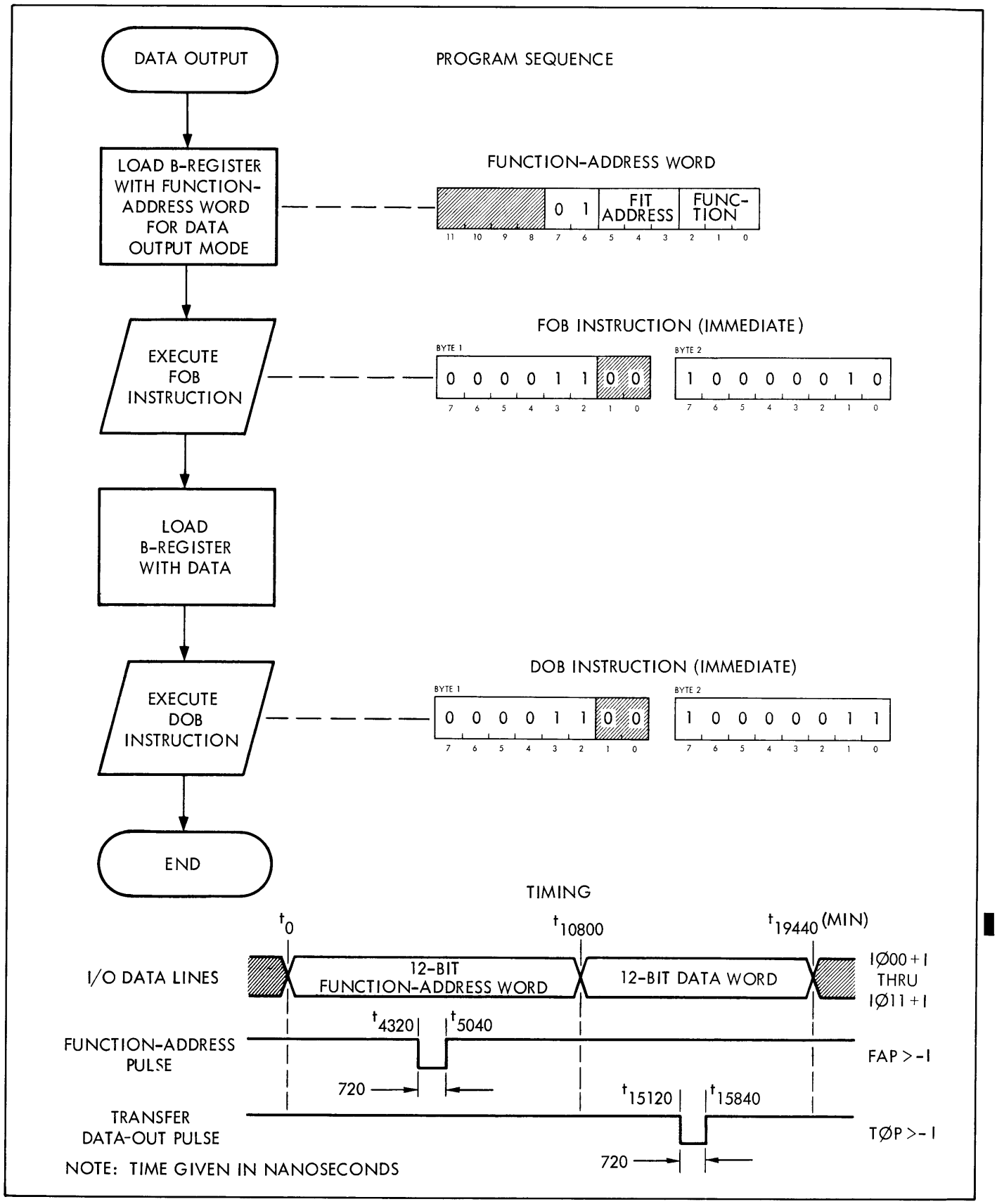


Figure 4-9. Parallel I/O Data Output Operation

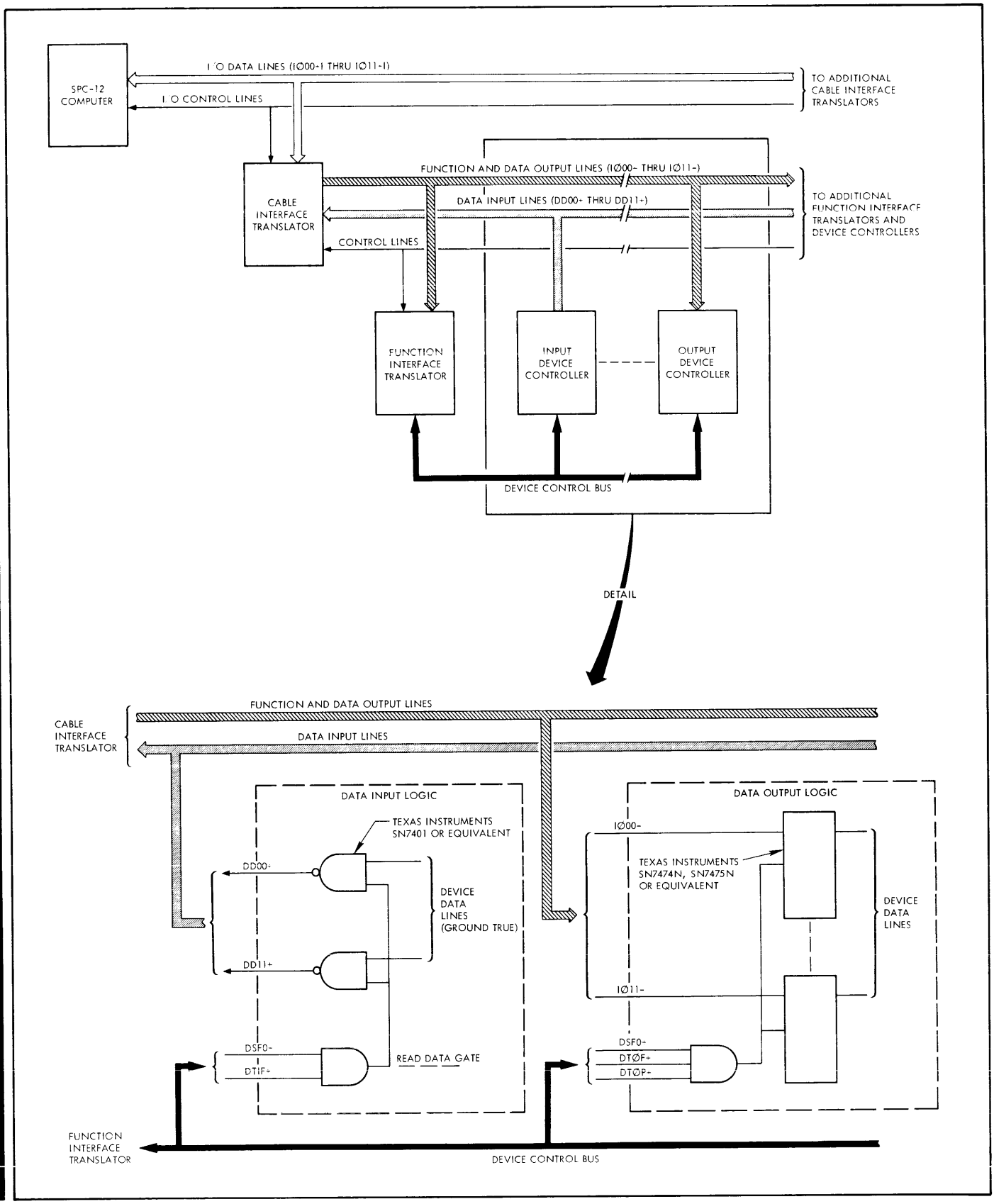


Figure 4-10. Typical Parallel I/O Configuration Using Processor Interface Units

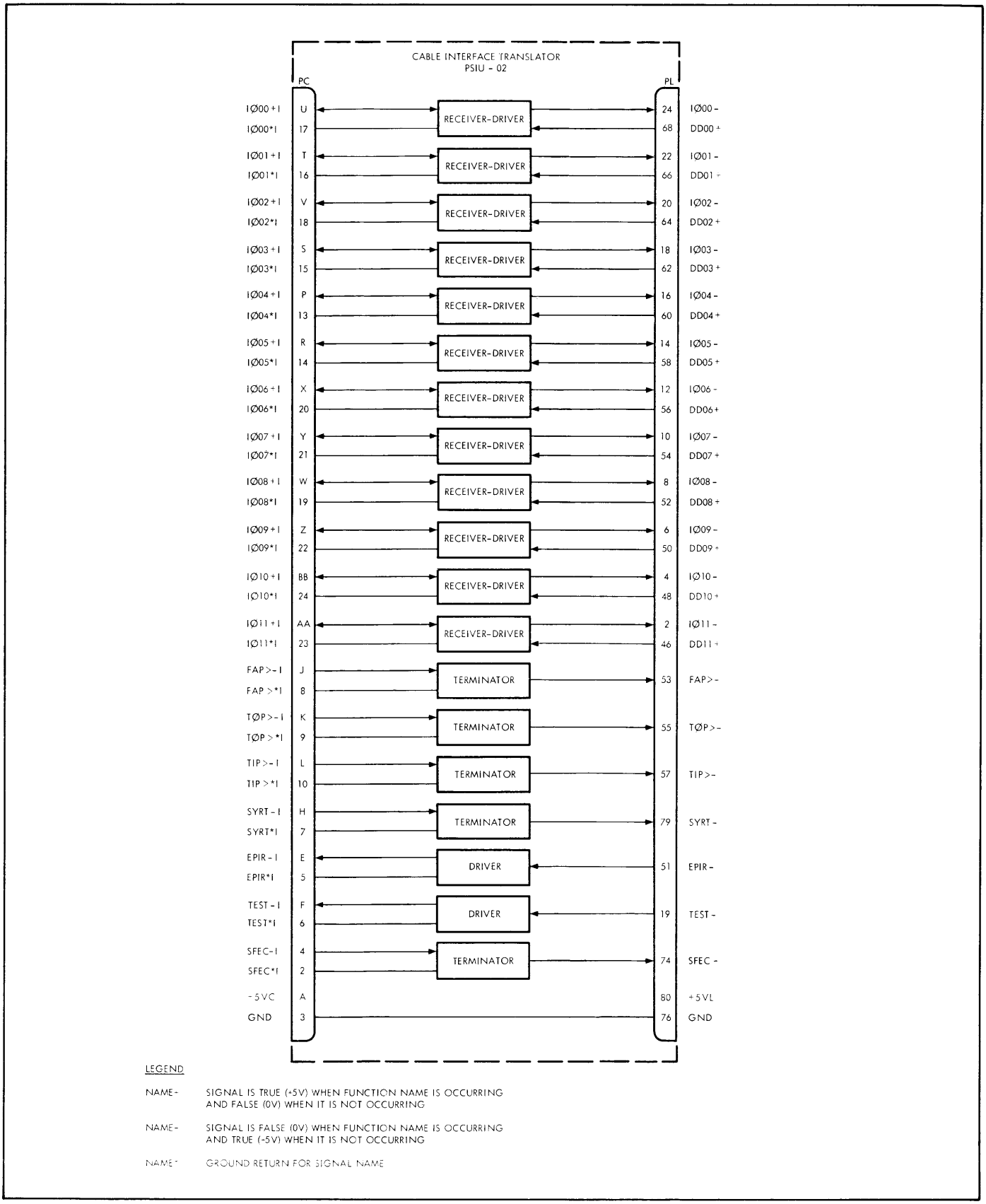


Figure 4-11. Cable Interface Translator, Functional Diagram

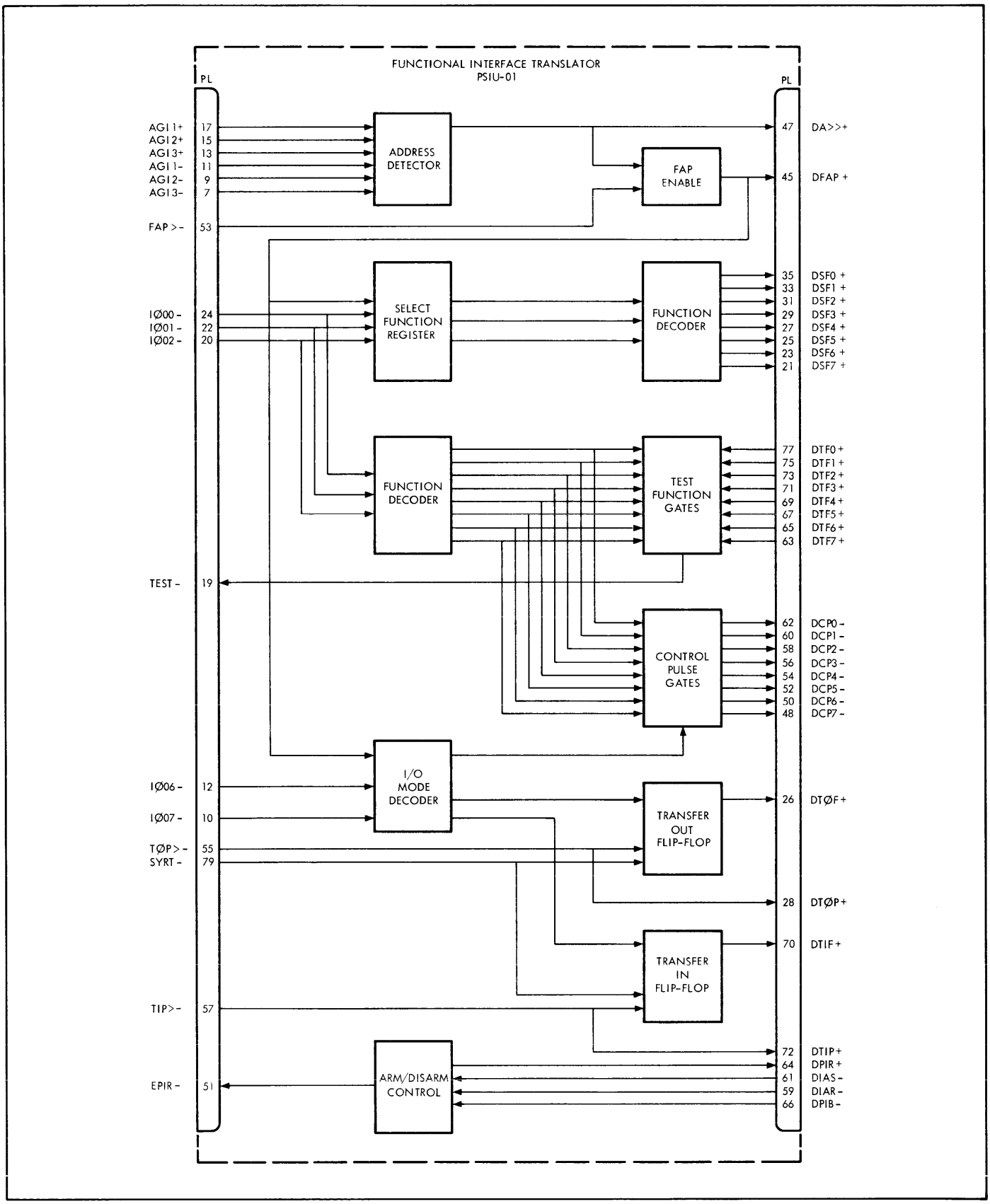


Figure 4-12. Functional Interface Translator, Functional Diagram

Table 4-2. Functional Interface Translator, Functional Specifications

I/O Line	Description
IØ00- thru IØ11-	Function inputs. Specify one of eight test, control, data input, or data output operations to be performed. (See paragraph 4.14.)
AGI1+ thru AGI3+, and AGI1- thru AGI3-	Address inputs. Designate which one of eight functional interface translators is being addressed. These lines connect to lines IØ03- through IØ05- of the cable interface translator. The way in which they are connected determines the address of the functional interface translator as follows:

Required Connection

FIT Address	AGI1+	AGI2+	AGI3+	AGI1-	AGI2-	AGI3-
0	Gnd	Gnd	Gnd	IØ03-	IØ04-	IØ05-
1	IØ03-	Gnd	Gnd	Open	IØ04-	IØ05-
2	Gnd	IØ04-	Gnd	IØ03-	Open	IØ05-
3	IØ03-	IØ04-	Gnd	Open	Open	IØ05-
4	Gnd	Gnd	IØ05-	IØ03-	IØ04-	Open
5	IØ03-	Gnd	IØ05-	Open	IØ04-	Open
6	Gnd	IØ04-	IØ05-	IØ03-	Open	Open
7	IØ03-	IØ04-	IØ05-	Open	Open	Open

IØ06- and IØ07-	I/O mode inputs. Specify whether functional interface translator is to operate in control signal, data output, data input, or test mode. (See paragraph 4.14.)
FAP>-	Function-address pulse input
TØP>-	Transfer data-out pulse input
TIP>-	Transfer data-in pulse input
SYRT-	System reset input
EPIR-	External priority interrupt request output
TEST-	I/O test output
	Device Lines (All outputs occur when functional interface translator is addressed)
DA≫+	Device address output. Positive level when lines IØ03- through IØ05- contain address of functional interface translator
DFAP+	Device function-address pulse output. 720-ns positive pulse when functional interface translator is addressed and function-address pulse occurs

Table 4-2. Functional Interface Translator, Functional Specifications (Cont.)

I/O Line	Description
DSF0+ thru DSF7+	Device select function output. Positive level occurs at leading edge of function-address pulse in conjunction with decoded function inputs. This level is stored and can be used with transfer data-in pulse or transfer data-out pulse to gate information into or out of I/O device, or to select different input or output operations
DTF0+ thru DTF7+	Device test function input. Positive level on one of these lines in conjunction with decoded function inputs indicates that I/O device condition being tested is true. For example, these lines can be used to determine if selected device is ready to send or receive data
DCP0- thru DCP7-	Device control pulse output. A 720-ns negative pulse occurs in conjunction with decoded function input, I/O mode input, and function-address pulse. Typical uses are: a. Read one card from card reader b. Print one line on line printer c. Read one record from magnetic tape unit
DTØF+	Device transfer-out function. Positive level occurs at trailing edge of function-address pulse in conjunction with decoded I/O mode input. This level can be used with device select function output and transfer data-out pulse to transfer data from the computer to the I/O device
DTØP+	Device transfer-out pulse. Positive pulse occurs concurrent with transfer data-out pulse. Trailing edge of pulse resets device transfer out function
DTIF+	Device transfer-in function. Positive level occurs at trailing edge of function-address pulse in conjunction with decoded I/O mode input. This level can be used with device select function output and transfer data-in pulse to transfer data from the I/O device to the computer
DTIP+	Device transfer-in pulse. Positive pulse occurs concurrent with transfer data-in pulse. Trailing edge of pulse resets device transfer in function
DIAS-	Device interrupt arm set. Negative pulse on this line sets arm flip-flop. This input is normally connected to a device control pulse output
DIAR-	Device interrupt arm reset. Negative pulse on this line resets arm flip-flop. This input is normally connected to a device control pulse output
DPIB-	Device priority interrupt bus. Negative level on this line initiates an interrupt request if arm flip-flop is set. Inputs are normally connected to function interrupt request flip-flops in I/O device, or to interrupt expander bus
DPIR+	Device priority interrupt request. Positive level on this line indicates that I/O device is requesting an interrupt. This output is normally connected to a device test function input when two or more functional interface translators are used; otherwise, it is left unconnected

4.22 I/O INTERFACE EXAMPLES

Figures 4-13 and 4-14 are typical examples showing how I/O devices can be connected to the parallel I/O bus through the cable interface translator and the functional interface translator. Figure 4-13 shows typical connections for an input device such as a card reader. Figure 4-14 shows typical connections for an output device such as a line printer.

4.23 INPUT DEVICE INTERFACE (See figure 4-13)

With the input device connected as shown in the diagram the interface lines are used as follows:

a. Action control line. This line is pulsed by executing a FOB instruction with a function-address word specifying the control signal mode and the appropriate control fine (function).

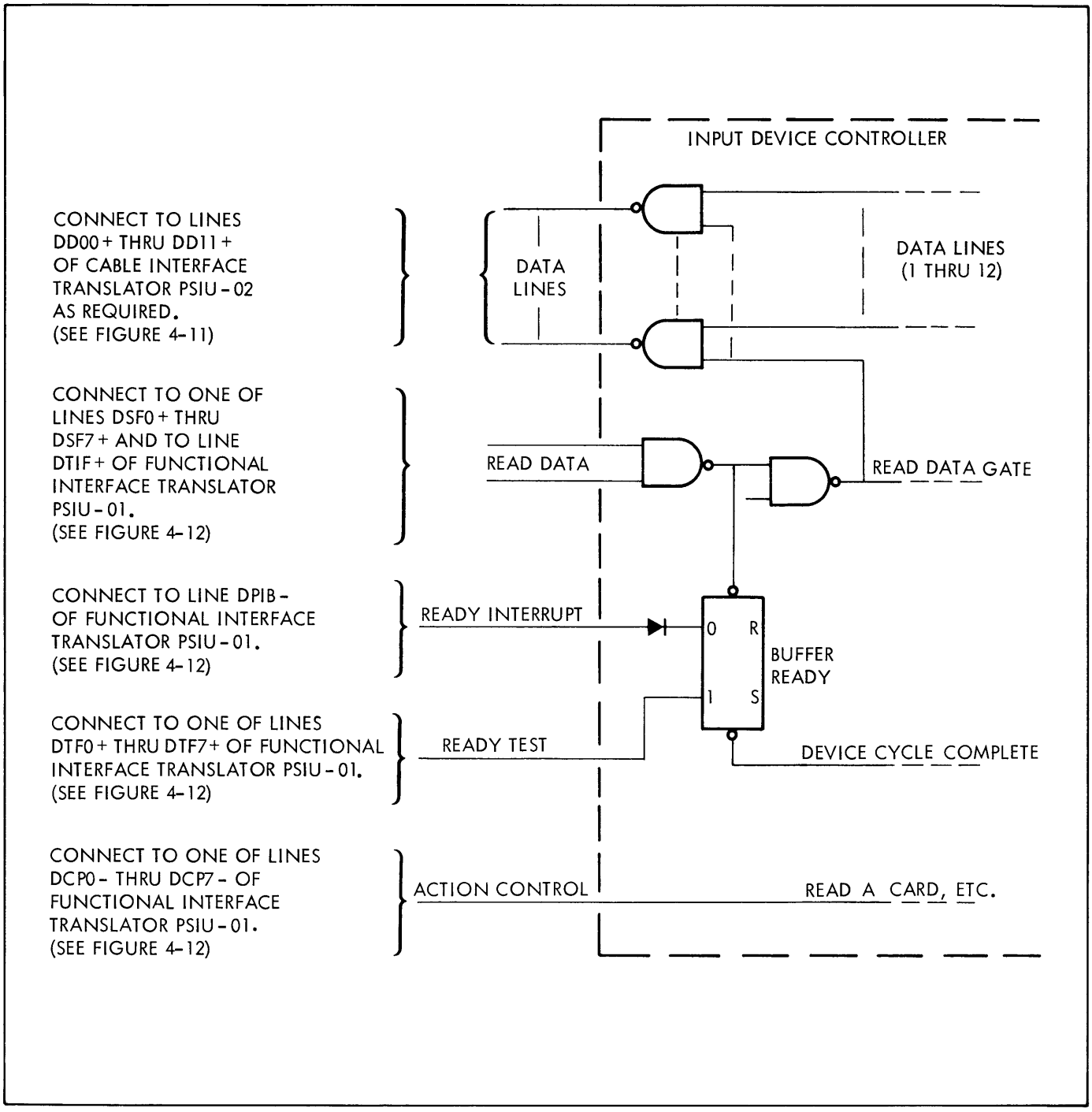


Figure 4-13. Parallel I/O Input Device Interface Example

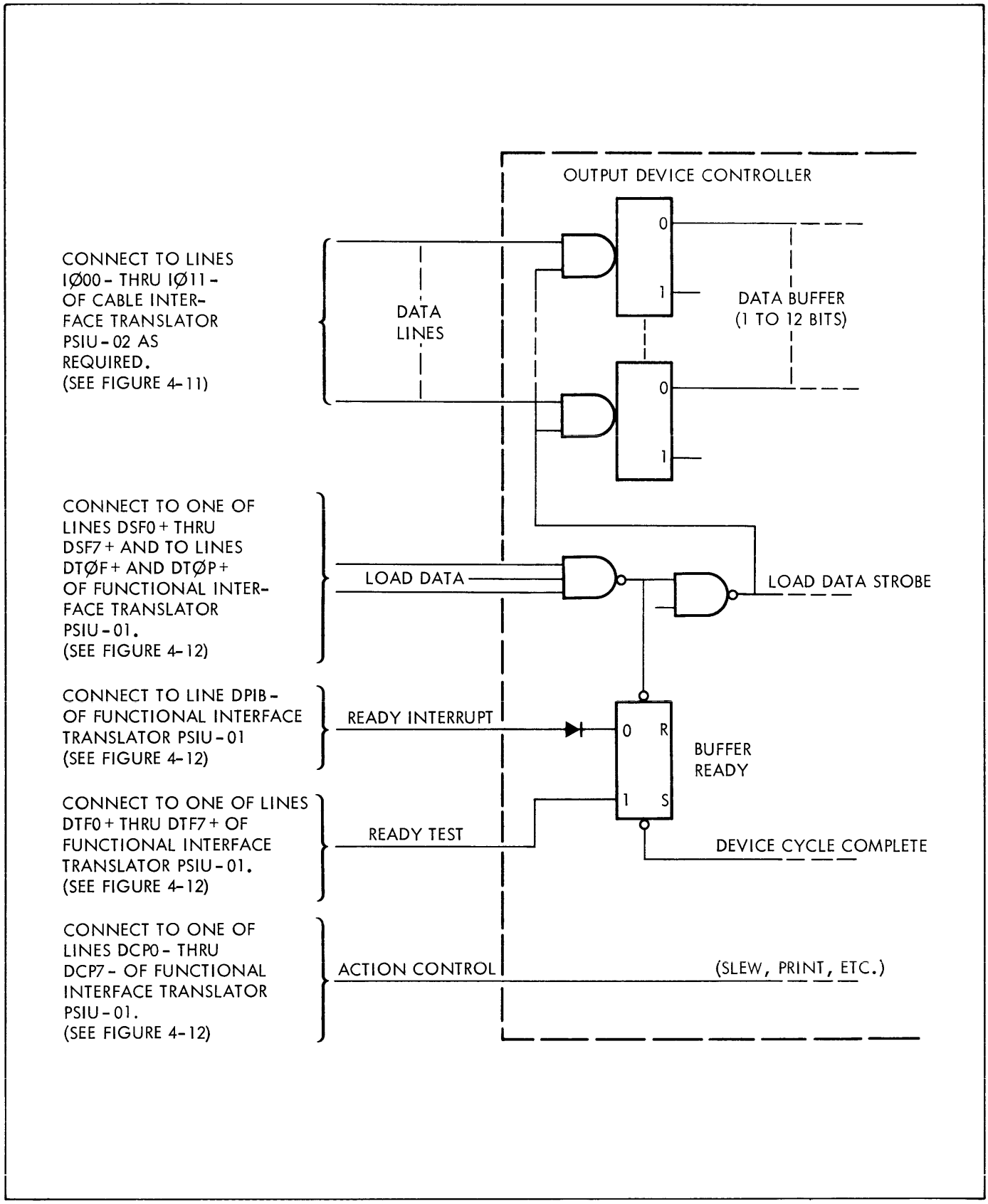


Figure 4-14. Parallel I/O Output Device Interface Example

b. Ready interrupt line. This line connects to the interrupt request line of the computer. It is interrogated by executing an INE instruction.

c. Ready test line. This line is sampled by executing a FOB instruction with a function-address word specifying the I/O test mode and the appropriate test line (function). The sampled value is stored in the I/O Test indicator.

d. Read data lines. These lines are activated by executing a FOB instruction with a function-address word specifying the data input mode and the appropriate function line, followed by a DIB instruction.

e. Data lines. Data is gated onto these lines and into the B-register of the computer when the read data lines are activated.

4.24 OUTPUT DEVICE INTERFACE (See figure 4-14)

With the output device connected as shown in the diagram the interface lines are used as follows:

a. Action control line. This line is pulsed by executing a FOB instruction with a function-address word specifying the control signal mode and the appropriate control (function) line.

b. Ready interrupt line. This line connects to the interrupt request line of the computer. It is interrogated by executing an INE instruction.

c. Ready test line. This line is sampled by executing a FOB instruction with a function-address word specifying the I/O test mode and the appropriate test (function) line. The sampled value is stored in the I/O Test indicator.

d. Load data lines. These lines are activated by executing a FOB instruction with a function-address word specifying the data output mode and the appropriate function line, followed by a DOB instruction.

e. Data lines. Data is gated from these lines into the data buffer of the I/O device when the load data lines are activated.

4.25 INTERRUPT SYSTEM

The basic interrupt system includes an external priority interrupt, a relative time clock interrupt, and an automatic restart interrupt. Each interrupt is assigned a dedicated memory location to which control is passed when the interrupt becomes active. The external priority and relative time clock interrupts operate on a request-response basis with the Interrupt Enable (INE) instruction. This instruction interrogates for the presence of an interrupt request and, if one is present, permits the interrupt to become active. If both requests are present simultaneously, the relative time clock interrupt takes priority.

The automatic restart interrupt operates independently of the external priority and relative time clock interrupts. It is activated by the automatic restart feature of the computer during a power-on sequence. It is primarily used to transfer program control to a restart routine when power is reapplied after having been removed.

4.26 EXTERNAL PRIORITY INTERRUPT

The external priority interrupt permits an I/O device connected to the parallel I/O system to request an interruption of the program being executed. The request is transmitted from the device to the computer on the interrupt request line (EPIR-I) of the parallel I/O cable (see figure 4-3). Although a request is present, the interrupt cannot become active until an INE instruction is executed. (See paragraph 3.7 for a description of the INE instruction.) Executing the INE instruction when an external priority interrupt request is present (and no relative time clock interrupt request is present) causes the following to occur:

a. The contents of the P-register are transferred into the B-register to save the return linkage back to the main program.

b. Address 0010_8 is placed into the P-register, and the computer executes the instruction stored in that location.

Normally, a Jump instruction would be stored in location 0010_8 to transfer program control to an interrupt servicing routine. As the I/O device has been serviced, it removes the interrupt request from the line.

When cable interface translators and functional interface translators are used in the parallel I/O system, a multilevel interrupt structure can be formed by connecting the system as shown in figure 4-15. This structure has the following features:

a. Each group of device interrupt requests can be armed or disarmed by setting or resetting the device interrupt arm flip-flop of the functional interface translator to which the group is connected. Arming or disarming is accomplished by executing a FOB instruction with a function-address word specifying the control signal mode and the appropriate control (function) line.

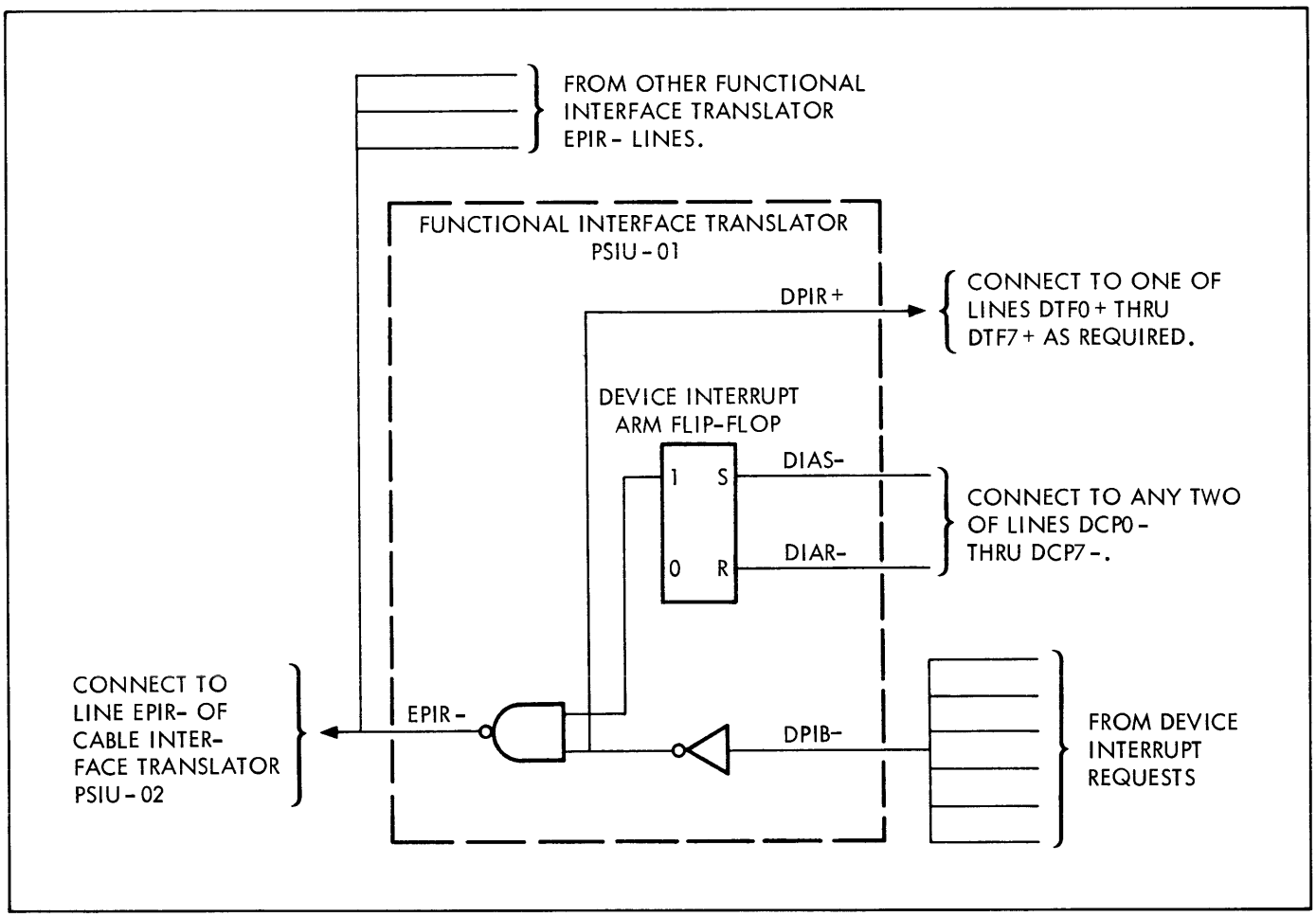


Figure 4-15. Typical Connection of Multiple External Priority Interrupts

b. The functional interface translator that is requesting an interrupt can be determined in the servicing routine by sampling the I/O test lines to which the interrupt request line is connected. (See paragraph 4.15.) If more than one functional interface translator is requesting an interrupt, priority can be established by the sequence in which the test lines are sampled.

c. The I/O device that is requesting an interrupt can be determined in the servicing routine by sampling the I/O test line to which the device is connected. If more than one device is requesting an interrupt, priority can be established by the sequence in which the test lines are sampled.

The external priority interrupt request line can be expanded, in groups of eight, to more rapidly select a single requestor by using the optional PSIU-03 Priority Interrupt Expander. The expander unit contains eight priority interrupt lines with address generation, line receivers, and arm/disarm capability for each interrupt level.

4.27 RELATIVE TIME CLOCK INTERRUPT

The relative time clock interrupt produces an interrupt request every 3 ms (1, 2, 4, or 5 ms optional). This interrupt request is handled exactly the same as an external priority interrupt: The interrupt cannot become active until an INE instruction is executed. (See paragraph 3.7 for a description of the INE instruction.) Executing the INE instruction when a relative time clock interrupt request is present causes the following to occur:

a. The contents of the P-register are transferred into the B-register to save the return linkage back to the main program.

b. Address 0012_8 is placed into the P-register, and the computer executes the instruction stored in that location.

Normally, a Jump instruction would be stored in location 0012_8 to transfer program control to an interrupt servicing routine. The interrupt request is automatically reset when the interrupt is serviced.

4.28 AUTOMATIC RESTART INTERRUPT

The automatic restart feature of the computer generates an interrupt each time a power-on sequence is initiated. This interrupt permits a restart routine to be performed after power is removed and then reapplied to the computer. The automatic restart interrupt operates as follows:

a. When power returns to operable limits, the power failure detection feature signals the automatic restart feature.

b. The automatic restart feature initiates a system reset signal and then executes the instruction stored in memory location 0014_8 if the computer is in run mode.

Normally, a jump instruction would be stored in location 0014_8 to transfer program control to a restart routine. (NOTE: The automatic restart sequence can be overridden by placing the R/I switch on the programmer's console into the idle mode position before the restart sequence is initiated.)

4.29 STALL ALARM OPTION

The stall alarm option protects the system against abnormal operation and provides a signal to warn of the abnormality. It is particularly valuable in detecting improper program sequencing caused by component breakdowns or previously undetected program errors. Once activated, the stall alarm option brings instruction execution to an orderly halt and generates a signal through the system safe line (SFEC-I) of the parallel I/O system. This signal can be used to control an audio-visual alarm or automatic switchover.

The Pulse Stall Alarm (PSA) instruction is provided with the stall alarm option to both arm the alarm and to reset it during normal operation. The instruction starts the stall alarm timer when it is executed for the first time and resets the timer each time it is executed thereafter. Failure to execute the instruction within 250 (±100) ms after the previous execution activates the stall alarm. When the alarm becomes activated, the computer automatically switches from run mode to idle mode, and the safe signal is removed from the safe line.

The procedure for clearing the stall alarm after it has been activated is the same as that for applying power and initializing as given in paragraphs 5.18 and 5.19. An important point to remember in clearing the stall alarm when the automatic restart feature is not present is that the R/I switch must be returned to the idle position before the SYS RST switch is pressed.

SECTION V
PROGRAMMER'S CONSOLE

5.1 GENERAL

The programmer's console (figure 5-1) provides the facility to control and monitor all primary operations of the computer. Two system safety features are incorporated in the console design to prevent inadvertent operation of console controls when the system is online and operating. A key-operated console lock switch, at the rear of the computer, must be unlocked for console controls to be operable. Console controls are also disabled when the computer is placed in the run mode.

5.2 CONSOLE CONTROLS

5.3 DATA SWITCHES (0 THROUGH 11)

The 12 data switches are used with the ENTER switch to enter data into one of eight registers selected by the REGISTER ADDR switches. The data switches are two-position toggle switches with the upper position representing a binary 0 and the lower position representing a binary 1. The switches are arranged in groups of three for ease of working with octal notation and are numbered to correspond with the bit positions of the selected register. Data is entered into the selected register by first setting the data switches to the desired configuration and then momentarily pressing the ENTER switch. The data switches are operable only when the computer is in idle mode and the console lock switch is set to UNLOCK.

5.4 ENTER SWITCH

The ENTER switch is used with the 12 data switches to enter data into one of eight registers selected by the REGISTER ADDR switches (see paragraph 5.3). The switch is spring-loaded to return to the upper position. It is operable only when the computer is in idle mode and the console lock switch is set to UNLOCK.

5.5 STEP SWITCH

The STEP switch is used to step through a stored program one instruction at a time. It is spring-loaded to return to the upper position. Pressing the switch causes the computer to execute the instruction currently in the I-register and to advance the P-register. The switch is operable only when the computer is in idle mode and the console lock switch is set to UNLOCK.

5.6 R/I (RUN-IDLE) SWITCH

The R/I switch controls the mode of operation in which the computer operates. It is a two-position toggle switch with the upper position designating the run mode and the lower position, the idle mode. Setting the switch to the upper position and pressing the STEP switch causes the computer to execute its stored program automatically. With the switch in this position all other controls on the console are inoperative. Setting the switch to the lower position

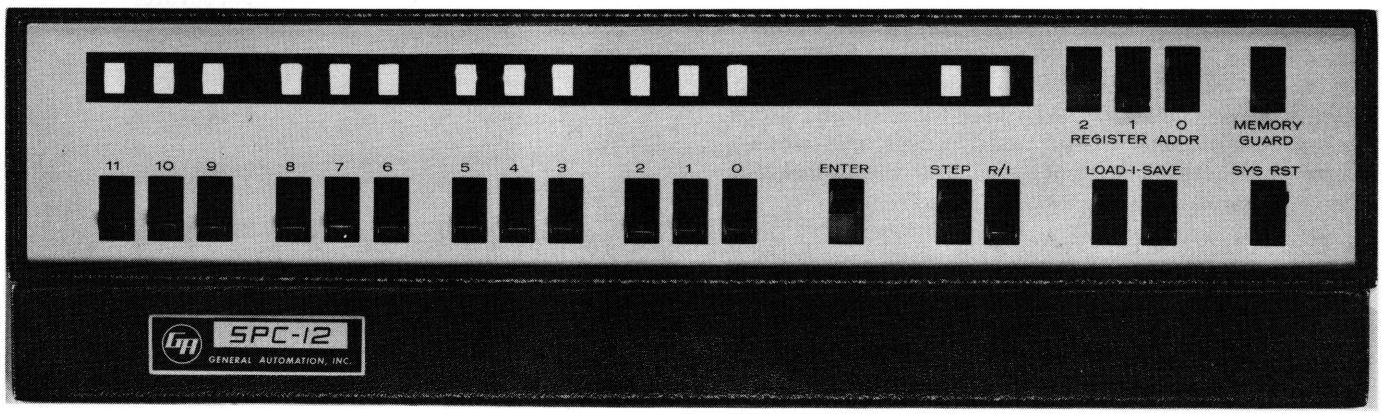


Figure 5-1. Programmer's Console

causes the computer to stop program execution immediately after the first byte of the next instruction in the program has been fetched from memory and placed in the I-register. The computer can then be manually controlled using the console switches. The R/I switch is operable only when the console lock switch is set to UNLOCK.

5.7 LOAD-I SWITCH

The LOAD-I switch is used to transfer the contents of the register selected by the REGISTER ADDR switches to the I-register. The switch is spring-loaded to return to the upper position. Pressing the switch causes the transfer of register contents. The switch is operable only when the computer is in idle mode, the console lock switch is set to UNLOCK and the SAVE-I switch is in the upper (off) position.

5.8 SAVE-I SWITCH

The SAVE-I switch is used to prevent the contents of the I-register from being lost during an instruction execution performed in the idle mode. It is a two-position toggle switch with the upper position off and the lower position on. Setting the switch to lower (on) position makes the LOAD-I switch inoperative. The SAVE-I switch is operable only when the computer is in idle mode and the console lock switch is set to UNLOCK.

The SAVE-I switch is used primarily to display the contents of sequential locations in memory from the console and to store information into sequential locations in memory from the console. To gain access to a block of words in memory for either loading or storing requires that the proper instruction (Load B-Register Indexed or Store B-Register Indexed) be placed in the I-register. Since the index register is incremented during instruction execution before the load or store operation is performed, the algebraic sum of the contents of the address field of the instruction and the contents of the index register must initially be one less than the desired effective operand address. (NOTE: Automatic incrementing of the index register during execution of the Load B-Register Indexed and Store B-Register Indexed instructions occurs only when these instructions are executed from the programmer's console with the SAVE-I switch on.)

The procedures for displaying the contents of sequential locations in memory and for storing information into sequential locations in memory from the programmer's console are described in paragraphs 5.25 and 5.24, respectively.

When the SAVE-I switch is off, the contents of the I-register can be altered from the programmer's console either by stepping through a program stored in memory one instruction at a time or by transferring the contents of any of the addressable registers into the I-register using the LOAD-I switch.

5.9 SYS RST (SYSTEM RESET) SWITCH

The SYS RST switch is used to reset the computer and all external I/O devices to an initialized condition. The switch is spring-loaded to return to the upper position. Pressing the switch resets the system. (NOTE: Register contents are not predictable after the switch is pressed.) The switch is operable only when the computer is in idle mode and the console lock switch is set to UNLOCK.

5.10 REGISTER ADDR (REGISTER ADDRESS) SWITCHES

The three REGISTER ADDR switches (0, 1, 2) are used to select one of eight programmable registers within the computer. The contents of the selected register are displayed by the 12 display lamps above the data switches. The contents of a selected register can be altered using the data switches and the ENTER switch, or the contents can be transferred to the I-register using the LOAD-I switch.

The REGISTER ADDR switches are two-position toggle switches with the upper position representing a binary 0 and the lower position representing a binary 1. The binary settings of the switches for addressing the programmable registers (which are the same as the register designations used in the instruction encoding) are as follows:

Switch Setting 2	1	0	Register Selected
0	0	0	A-register
0	0	1	X-register
0	1	0	Y-register
0	1	1	Z-register
1	0	0	P-register
1	0	1	B-register
1	1	0	YY-register
1	1	1	ZZ-register

When the YY- or ZZ-register is selected for display, register display lamps 9, 10, and 11 display the contents of the E-register instead of the three most significant bits of the YY- or ZZ-register.

The REGISTER ADDR switches are operable only when the computer is operating in idle mode and the console lock switch is set to UNLOCK.

5.11 MEMORY GUARD SWITCH

The MEMORY GUARD switch is used to protect the contents of memory. It is a two-position toggle switch with the upper position off and the lower position on. Setting the switch to the lower position (on) disables memory drive currents so that no location in memory can be read or written into. Setting the switch to the upper position (off) permits access to all memory locations for both reading and storing. The switch is operable only when the computer is operating in idle mode and the console lock switch is set to UNLOCK. The switch should be set to the lower (on) position when the computer is in idle mode. It should always be set to the upper (off) position if the computer is equipped with the power failure detection and automatic restart option.

5.12 CONSOLE LOCK SWITCH

The console lock switch is used to enable or disable all controls on the programmer's console. It is a two-position, key-operated switch located at the rear of the computer (figure 5-2). When the switch is set to LOCK, all controls on the console are inoperative. When it is set to UNLOCK, the console controls are operative if the computer is operating in idle mode. The key can be removed from the switch when it is in either the LOCK or UNLOCK position.

5.13 CONSOLE DISPLAYS

5.14 REGISTER DISPLAY (0 THROUGH 11)

The 12 register display lamps display the contents of the data bus. The display lamps are located directly above the data switches, and like the data switches, are arranged in groups of three for ease of working with octal notation. Each of the lamps is lit when the corresponding bit position of the data bus contains a 1-bit.

When the computer is operating in idle mode, the display lamps display the contents of the register selected by the REGISTER ADDR switches (since, in idle mode, the selected register is connected to the data bus). When either the YY- or ZZ-register (6_8 or 7_8) is selected, register display lamps 9, 10, and 11 display the contents of the E-register instead of the most significant bits of the YY- or ZZ-register.

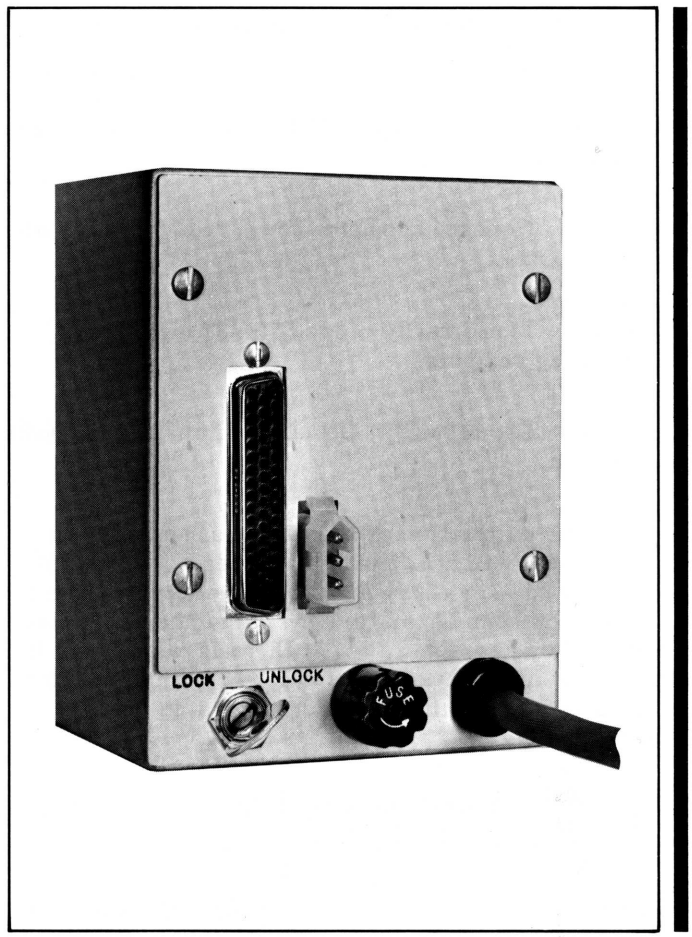


Figure 5-2. Console Lock Switch

5.15 IDLE MODE DISPLAY

The idle mode display lamp, located directly above the STEP switch, is lit when the computer is operating in idle mode.

5.16 RUN MODE DISPLAY

The run mode display lamp, located directly above the R/I switch, is lit when the computer is operating in run mode.

5.17 OPERATING PROCEDURES

5.18 APPLYING POWER AND INITIALIZING (WITHOUT AUTOMATIC RESTART)

To apply power and initialize the system, proceed as follows:

1. Verify that computer power cable is not connected to ac power outlet.

2. Set console lock switch (at rear of computer) to UNLOCK.

3. Set R/I switch to lower (idle mode) position.

4. Set MEMORY GUARD switch to lower (on) position.

5. Connect computer power cable to 110V ac power outlet.

6. Momentarily press SYS RST switch to initialize computer.

7. Set MEMORY GUARD switch to upper (off) position.

With the completion of step 7 the computer is in idle mode with all console switches operable. If a program is already stored in memory, program execution can be started by using the procedure given in paragraph 5.26. If there is no program in memory, information can be entered into the computer by using the procedure given in paragraph 5.24.

5.19 APPLYING POWER AND INITIALIZING (WITH AUTOMATIC RESTART)

When power is applied to a computer containing the power failure detection and automatic restart feature, the computer automatically begins program execution with the instruction stored in memory location 0014_8. This location would normally contain a jump instruction to transfer program control to a power startup routine. If the startup routine is not present or if there is some doubt that it is present, the procedure given in paragraph 5.18 should be used to apply power. To apply power with the startup routine present, proceed as follows:

1. Verify that computer power cable is not connected to ac power outlet.

2. Set console lock switch (at rear of computer) to LOCK.

3. Connect computer power cable to 110V ac power outlet. Computer automatically begins program execution with instruction stored in memory location 0014_8.

With the completion of step 3 the computer is locked in run mode, and all console switches are inoperative.

5.20 REMOVING POWER (WITHOUT POWER FAILURE DETECTION)

To remove power from the system, proceed as follows:

1. Set console lock switch to UNLOCK.

2. Set R/I switch to lower (idle mode) position.

3. Set MEMORY GUARD switch to lower (on position.

4. Disconnect computer power cable from ac power outlet.

5.21 REMOVING POWER (WITH POWER FAILURE DETECTION)

To remove power from a system equipped with the power failure detection option, simply disconnect the computer power cable from the ac power outlet.

5.22 ALTERING REGISTER CONTENTS

To alter the contents of any of the eight programmable registers from the programmer's console, proceed as follows:

1. Set console lock switch to UNLOCK

2. Set R/I switch to lower (idle mode) position.

3. Set the REGISTER ADDR switches to the address of the register to be altered.

4. Set the 12 data switches to the desired configuration (upper position represents 0, lower position represents 1).

5. Momentarily press ENTER switch.

6. Verify that contents of register have been altered by observing register display lamps.

5.23 ALTERING I-REGISTER CONTENTS

To alter the contents of the I-register from the programmer's console, proceed as follows:

1. Set console lock switch to UNLOCK

2. Set R/I switch to lower (idle mode) position.

3. Set SAVE-I switch to upper (off) position.

4. Enter the instruction that is to be transferred to the I-register into one of the eight programmable registers by using the procedure given in paragraph 5.22, steps 3 through 6.

5. Press the LOAD-I switch to transfer the contents of the selected register into the I-register.

5.24 ALTERING MEMORY CONTENTS

To alter the contents of memory locations, proceed as follows:

1. Set console lock switch to UNLOCK.

2. Set R/I switch to lower (idle mode) position.

3. Set MEMORY GUARD switch to upper (off) position.

4. Set SAVE-I switch to upper (off) position.

5. Set the contents of the B-register to 6400_8 (Store B-Register Indexed instruction) using the procedure given in paragraph 5.22, steps 3 through 6.

6. Press the LOAD-I switch to transfer the instruction from the B-register to the I-register.

7. Set SAVE-I switch to lower (on) position.

8. Set the contents of the X-register to one less than the address of the memory location to be altered by using the procedure given in paragraph 5.22, steps 3 through 6.

9. Set the REGISTER ADDR switches to select the B-register (101_2).

10. Set data switches 0 through 7 to the bit pattern of the information to be stored in memory.

11. Press the ENTER switch.

12. Momentarily press the STEP switch. The new information is now stored in the memory location whose address is currently in the X-register.

13. To continue storing information in consecutive memory locations, repeat steps 9 through 12.

5.25 DISPLAYING MEMORY CONTENTS

To display the contents of memory locations from the programmer's console, proceed as follows:

1. Set console lock switch to UNLOCK.

2. Set R/I switch to lower (idle mode) position.

3. Set MEMORY GUARD switch to upper (off) position.

4. Set SAVE-I switch to upper (off) position.

5. Set the contents of the B-register to 7400_8 (Load B-Register Indexed instruction) using the procedure given in paragraph 5.22, steps 3 through 6.

6. Press the LOAD-I switch to transfer the instruction from the B-register to the I-register.

7. Set SAVE-I switch to lower (on) position.

8. Set the contents of the X-register to one less than the address of the memory location to be displayed by using the procedure given in paragraph 5.22, steps 3 through 6.

9. Set the REGISTER ADDR switches to select the B-register (101_2).

10. Momentarily press the STEP switch. Register display lamps 0 through 7 now display the contents of the memory location whose address is currently contained in the X-register.

11. To continue displaying the contents of consecutive memory locations, repeat step 10.

5.26 INITIATING PROGRAM EXECUTION

To begin executing a program that is already stored in memory, proceed as follows:

1. Set console lock switch to UNLOCK.

2. Set R/I switch to lower (idle mode) position.

3. Set MEMORY GUARD switch to upper (off) position.

4. Set SAVE-I switch to upper (off) position.

5. Set the contents of any of the six 12-bit registers (A, X, Y, Z, P, or B) to 2100_8 (No Operation instruction) using the procedure given in paragraph 5.22, steps 3 through 6.

6. Press the LOAD-I switch to transfer the instruction from the selected register to the I-register.

7. Set the contents of the P-register to one less than the address of the memory location where program execution is to begin by using the procedure given in paragraph 5.22, steps 3 through 6.

8. Set R/I switch to upper (run mode) position.

9. Momentarily press the STEP switch. The computer now begins executing the program with the instruction stored in the starting memory location.

10. Set console lock switch to LOCK and remove key.

With the completion of step 10 the computer is locked in run mode with all console switches inoperative.

APPENDIX A
CONVERSION TABLES

This appendix contains the following reference tables:

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625

DECIMAL/BINARY POSITION TABLE

Largest Decimal Integer	Decimal Digits Req'd*	Number of Binary Digits	Largest Decimal Fraction
1		1	.5
3		2	.75
7		3	.875
15	1	4	.937 5
31		5	.968 75
63		6	.984 375
127	2	7	.992 187 5
255		8	.996 093 75
511		9	.998 046 875
1 023	3	10	.999 023 437 5
2 047		11	.999 511 718 75
4 095		12	.999 755 859 375
8 191		13	.999 877 929 687 5
16 383	4	14	.999 938 964 843 75
32 767		15	.999 969 482 421 875
65 535		16	.999 984 741 210 937 5
131 071	5	17	.999 992 370 605 468 75
262 143		18	.999 996 185 302 734 375
524 287		19	.999 998 092 651 367 187 5
1 048 575	6	20	.999 999 046 325 683 593 75
2 097 151		21	.999 999 523 162 841 796 875
4 194 303		22	.999 999 761 581 420 898 437 5
8 388 607		23	.999 999 880 790 710 449 218 75
16 777 215	7	24	.999 999 940 395 355 244 609 375
33 554 431		25	.999 999 970 197 677 612 304 687 5
67 108 863		26	.999 999 985 098 838 806 152 343 75
134 217 727	8	27	.999 999 992 549 419 403 076 171 875
268 435 455		28	.999 999 996 274 709 701 538 085 937 5
536 870 911		29	.999 999 998 137 354 850 769 042 968 75
1 073 741 823	9	30	.999 999 999 068 677 425 384 521 484 375
2 147 483 647		31	.999 999 999 534 338 712 692 260 742 187 5
4 294 967 295		32	.999 999 999 767 169 356 346 130 371 093 75
8 589 934 591		33	.999 999 999 883 584 678 173 065 185 546 875
17 179 869 183	10	34	.999 999 999 941 792 339 086 532 592 773 437 5
34 359 738 367		35	.999 999 999 970 896 169 543 266 296 386 718 75
68 719 476 735		36	.999 999 999 985 448 034 771 633 148 193 359 375
137 438 953 471	11	37	.999 999 999 992 724 042 385 816 574 096 679 687 5
274 877 906 943		38	.999 999 999 996 362 021 192 908 287 048 339 843 75
549 755 813 887		39	.999 999 999 998 181 010 596 454 143 524 169 921 875
1 099 511 627 775	12	40	.999 999 999 999 090 505 298 227 071 762 084 960 937 5
2 199 023 255 551		41	.999 999 999 999 545 252 649 113 535 881 042 480 468 75
4 398 046 511 103		42	.999 999 999 999 772 626 324 556 767 940 521 240 234 375
8 796 093 022 207		43	.999 999 999 999 886 313 162 278 383 970 260 620 117 187 5
17 592 186 044 415	13	44	.999 999 999 999 943 156 581 139 191 985 130 310 058 593 75
35 184 372 088 831		45	.999 999 999 999 971 578 290 569 595 992 565 155 029 296 875
70 368 744 177 663		46	.999 999 999 999 985 789 145 284 797 996 282 577 514 648 437 5
140 737 488 355 327	14	47	.999 999 999 999 992 894 572 642 398 998 141 288 757 324 218 75
281 474 976 710 655		48	.999 999 999 999 996 447 286 321 199 499 070 644 378 662 109 375
562 949 953 421 311		49	.999 999 999 999 998 223 643 160 599 749 535 322 189 331 054 687 5
1 125 899 906 842 623	15	50	.999 999 999 999 999 111 821 580 299 874 767 661 094 665 527 343 75
2 251 799 813 685 247		51	.999 999 999 999 999 555 910 790 149 937 383 830 547 332 763 671 875
4 503 599 627 370 495		52	.999 999 999 999 999 777 955 395 074 968 691 915 273 666 381 835 937 5
9 007 199 254 740 991		53	.999 999 999 999 999 888 977 697 537 484 345 957 636 833 190 917 968 75
18 014 398 509 481 983	16	54	.999 999 999 999 999 944 488 848 768 742 172 978 818 416 595 458 984 375
36 028 797 018 963 967		55	.999 999 999 999 999 972 244 424 384 371 086 489 409 208 297 729 492 187 5
72 057 594 037 927 935		56	.999 999 999 999 999 986 122 212 192 185 543 244 704 604 148 864 746 093 75
144 115 188 075 855 871	17	57	.999 999 999 999 999 993 061 106 096 092 771 622 352 302 074 432 373 046 875
288 230 376 151 711 743		58	.999 999 999 999 999 996 530 553 048 046 385 811 176 151 037 216 186 523 437 5
576 460 752 303 423 487		59	.999 999 999 999 999 998 265 276 524 023 192 905 588 075 518 608 093 261 718 75
1 152 921 504 606 846 975	18	60	.999 999 999 999 999 999 132 638 262 011 596 452 794 037 759 304 046 630 859 375

*Larger numbers within a digit group should be checked for exact number of decimal digits required.

Examples of use:

1. *Q.* What is the largest decimal value that can be expressed by 36 binary digits?
   A. 68,719,476,735.
2. *Q.* How many decimal digits will be required to express a 22-bit number?
   A. 7 decimal digits.

CONSTANTS

π	=	3.14159 26535 89793 23846 26433 83279 50
$\sqrt{3}$	=	1.732 050 807 569
$\sqrt{10}$	=	3.162 277 660 1683
e	=	2.71828 18284 59045 23536
ln 2	=	0.69314 71805 599453
ln 10	=	2.30258 50929 94045 68402
$\log_{10} 2$	=	0.30102 99956 63981
$\log_{10} e$	=	0.43429 44819 03251 82765
$\log_{10} \log_{10} e$	=	9.63778 43113 00537 - 10
$\log_{10} \pi$	=	0.49714 98726 94133 85435
1 degree	=	0.01745 32925 11943 radians
1 radian	=	57.29577 95131 degrees
$\log_{10}(5)$	=	0.69897 00043 36019
7!	=	5040
8!	=	40320
9!	=	362,880
10!	=	3,628,800
11!	=	39,916,800
12!	=	479,001,600
13!	=	6,227,020,800
14!	=	87,178,291,200
15!	=	1,307,674,368,000
16!	=	20,922,789,888,000
$\frac{\pi}{180}$	=	0.01745 32925 19943 29576 92369 07684 9
$\left(\frac{\pi}{2}\right)^2$	=	2.4674 01100 27233 96
$\left(\frac{\pi}{2}\right)^3$	=	3.8757 84585 03747 74
$\left(\frac{\pi}{2}\right)^4$	=	6.0880 68189 62515 20
$\left(\frac{\pi}{2}\right)^5$	=	9.5631 15149 54004 49
$\left(\frac{\pi}{2}\right)^6$	=	15.0217 06149 61413 07
$\left(\frac{\pi}{2}\right)^7$	=	23.5960 40842 00618 62
$\left(\frac{\pi}{2}\right)^8$	=	37.0645 72481 52567 57
$\left(\frac{\pi}{2}\right)^9$	=	58.2208 97135 63712 59
$\left(\frac{\pi}{2}\right)^{10}$	=	91.4531 71363 36231 53
$\left(\frac{\pi}{2}\right)^{11}$	=	143.6543 05651 31374 95
$\left(\frac{\pi}{2}\right)^{12}$	=	225.6516 55645 350
$\left(\frac{\pi}{2}\right)^{13}$	=	354.4527 91822 91051 47
$\left(\frac{\pi}{2}\right)^{14}$	=	556.7731 43417 624

CONSTANTS (Cont.)

π^2	= 9.86960	44010	89358	61883	43909	9988
$2\pi^2$	= 19.73920	88021	78717	23766	87819	9976
$3\pi^2$	= 29.60881	32032	68075	85680	31729	9964
$4\pi^2$	= 39.47841	76043	57434	47533	75639	9952
$5\pi^2$	= 49.34802	20054	46793	09417	19549	9940
$6\pi^2$	= 59.21762	64065	36151	71300	63459	9928
$7\pi^2$	= 69.08723	08076	25510	33184	07369	9916
$8\pi^2$	= 78.95683	52087	14868	95067	51279	9904
$9\pi^2$	= 88.82643	96098	04227	56950	95189	9892

$\sqrt{2}$	= 1.414	213	562	373	095	048	801	688
$1 + \sqrt{2}$	= 2.414	213	562	373	095	048	801	688
$(1 + \sqrt{2})^2$	= 5.828	427	124	746	18			
$(1 + \sqrt{2})^4$	= 33.970	562	748	477	08			
$(1 + \sqrt{2})^6$	= 197.994	949	366	116	30			
$(1 + \sqrt{2})^8$	= 1153.999	133	448	220	72			
$(1 + \sqrt{2})^{10}$	= 6725.999	851	323	208	02			
$(1 + \sqrt{2})^{12}$	= 39201.999	974	491	027	40			
$(1 + \sqrt{2})^{14}$	= 228485.999	995	622	956	38			
$(1 + \sqrt{2})^{16}$	= 1331713.999	999	246	711				
$(1 + \sqrt{2})^{18}$	= 7761797.999	999	884	751				

Sin .5	= 0.47942	55386	04203
Cos .5	= 0.87758	25618	90373
Tan .5	= 0.54630	24898	43790
Sin 1	= 0.84147	09848	07896
Cos 1	= 0.54030	23058	68140
Tan 1	= 1.55740	77246	5490
Sin 1.5	= 0.99749	49866	04054
Cos 1.5	= 0.07073	72016	67708
Tan 1.5	= 14.10141	99471	707

60-Bit Constants in Normalized Floating Point Form

Number	Octal Representation				
1	1720	4000	0000	0000	0000
10	1723	5000	0000	0000	0000
10^2	1726	6200	0000	0000	0000
10^3	1731	7640	0000	0000	0000
10^4	1735	4704	0000	0000	0000
10^5	1740	6065	0000	0000	0000
10^6	1743	7502	2000	0000	0000

CONSTANTS (Cont.)

Number	Octal Representation				
10^{7}	1747	4611	3200	0000	0000
10^{8}	1752	5753	6040	0000	0000
10^{9}	1755	7346	5450	0000	0000
10^{10}	1761	4520	1371	0000	0000
10^{11}	1764	5644	1667	2000	0000
10^{-1}	1714	6314	6314	6314	6315
10^{-2}	1711	5075	3412	1727	0244
10^{-3}	1706	4061	1156	4570	6520
10^{-4}	1702	6433	3427	2616	1031
10^{-5}	1677	5174	2654	2161	5510
10^{-6}	1674	4143	3675	0132	7554
10^{-7}	1670	6553	7624	6536	2572
10^{-8}	1665	5274	6167	0430	2142
10^{-9}	1662	4227	0137	2023	3265
10^{-10}	1656	6676	3376	6353	6756
10^{16}	2006	4341	5711	5760	2000
10^{32}	2073	4734	2655	5202	5561
10^{64}	2245	6047	4037	2237	7720
10^{128}	2572	4473	5107	6230	0352
10^{256}	3443	5247	7353	7671	6772
1/3	1716	5252	5252	5252	5253
1/5	1715	6314	6314	6314	6315
1/7	1715	4444	4444	4444	4445
1/9	1714	7070	7070	7070	7071
1/11	1714	5642	7213	5056	4272
1/13	1714	4730	4730	4730	4731
1/15	1714	4210	4210	4210	4211
1/17	1713	7417	0360	7417	0361
1/19	1713	6571	2065	7120	6571

OCTAL TO DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

0000	0000
to	to
0777	0511
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

1000	0512
to	to
1777	1023
(Octal)	(Decimal)

OCTAL TO DECIMAL INTEGER CONVERSION TABLE (Cont.)

2000	1024
to	to
2777	1535
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2100	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1519	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

3000	1536
to	to
3777	2047
(Octal)	(Decimal)

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

OCTAL TO DECIMAL INTEGER CONVERSION TABLE (Cont.)

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

4000	2048
to	to
4777	2559
(Octal)	(Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

5000	2560
to	to
5777	3071
(Octal)	(Decimal)

OCTAL TO DECIMAL INTEGER CONVERSION TABLE (Cont.)

6000 (Octal) to 6777 (Octal) = 3072 (Decimal) to 3583 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

7000 (Octal) to 7777 (Octal) = 3584 (Decimal) to 4095 (Decimal)

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3496	3497	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

OCTAL TO DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

OCTAL TO DECIMAL FRACTION CONVERSION TABLE (Cont.)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL TO DECIMAL FRACTION CONVERSION TABLE (Cont.)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

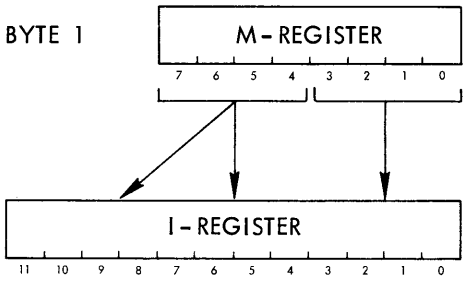
MODEL 33 ASR/KSR TELETYPE (ASCII) CODE
IN OCTAL FORM

Character	8-Bit Code (in octal)	Character	8-Bit Code (in octal)
A	301	!	241
B	302	"	242
C	303	#	243
D	304	$	244
E	305	%	245
F	306	&	246
G	307	'	247
H	310	(	250
I	311	)	251
J	312	*	252
K	313	+	253
L	314	,	254
M	315	-	255
N	316	.	256
O	317	/	257
P	320	:	272
Q	321	;	273
R	322	<	274
S	323	=	275
T	324	>	276
U	325	?	277
V	326	@	300
W	327	[	333
X	330	\	334
Y	331	]	335
Z	332	↑	336
		←	337
0	260		
1	261		
2	262	Line-Feed	212
3	263	Carriage-Return	215
4	264	Space	240
5	265	Rub-out	377
6	266	Blank	000
7	267		
8	270		
9	271		

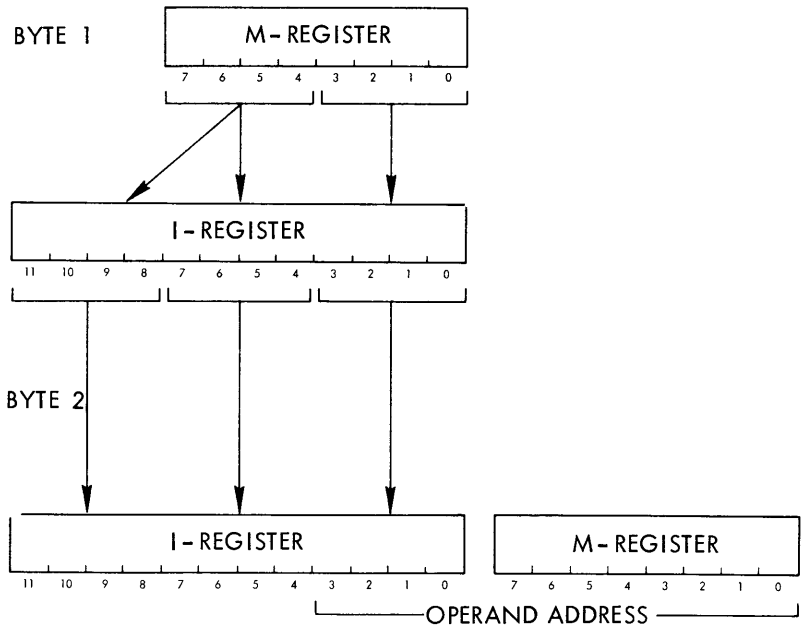
APPENDIX B
INSTRUCTION ASSEMBLY

Before an instruction can be executed, it must be fetched from memory and placed in the I-register. Since most of the instructions have a two-byte format, they must be assembled in the 12-bit I-register in a special way. The following diagrams show how the basic instruction types are assembled in the I-register.

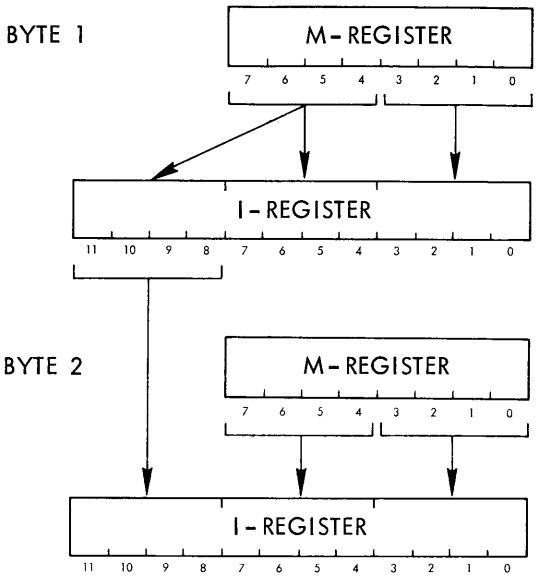
CONDITIONAL TRANSFER INSTRUCTIONS



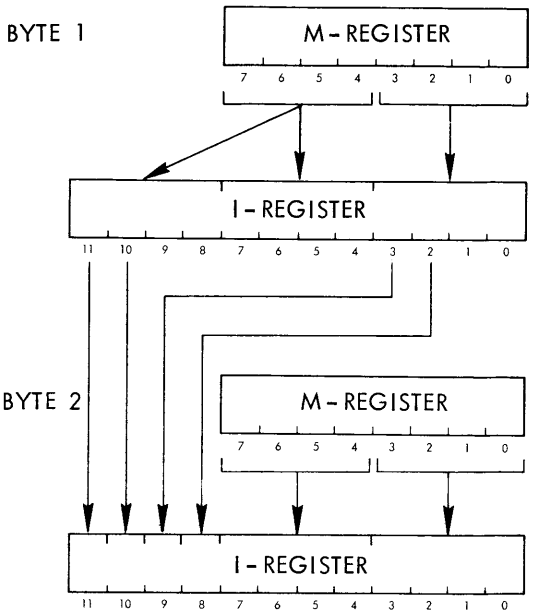
MEMORY ADDRESSING INSTRUCTIONS



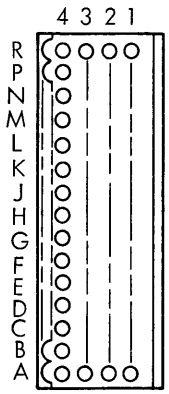
SHARED COMMAND ADDRESSING INSTRUCTIONS



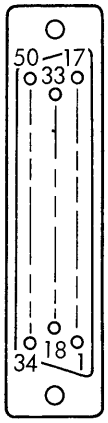
IMMEDIATE COMMAND ADDRESSING INSTRUCTIONS



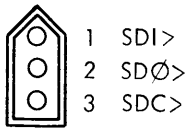
APPENDIX C
I/O CABLE CONNECTOR PIN ASSIGNMENTS



PARALLEL I/O CONNECTOR
USED ON SER. NO A0100
THROUGH A0196.
MATING CONNECTOR IS
MOLEX PRODUCTS CO.
PART 1725-P60.



PARALLEL I/O CONNECTOR
USED ON SER. NO. A0197
AND ABOVE.
MATING CONNECTOR IS
ITT CANNOT ELECT. PART DD-50P



SERIAL I/O CONNECTOR (TTY)
MATING CONNECTOR IS
MOLEX PRODUCTS CO.
PART 1396R-1/1381-TL

Parallel I/O Connector Pin Assignments

Signal Name	Connector Pin	
	Ser. No. A0100 Thru A0196	Ser. No. A0197 and Above
IØ00+I	K3	30
IØ00*I	J3	31
IØ01+I	K1	40
IØ01*I	J1	41
IØ02+I	L4	7
IØ02*I	K4	8
IØ03+I	L2	20
IØ03*I	K2	21
IØ04+I	M3	28
IØ04*I	L3	29
IØ05+I	M1	38

Parallel I/O Connector Pin Assignments (Cont.)

Signal Name	Connector Pin	
	Ser. No. A0100 Thru A0196	Ser No. A0197 and Above
IØ05*I	L1	39
IØ06+I	N4	5
IØ06*I	M4	6
IØ07+I	N2	18
IØ07*I	M2	19
IØ08+I	P3	1
IØ08*I	N3	2
IØ09+I	P1	36
IØ09*I	N1	37
IØ10+I	R4	3
IØ10*I	P4	4
IØ11+I	R2	34
IØ11*I	P2	35
FAP >-I	F3	32
FAP >*I	E3	33
TIP >-I	H1	42
TIP >*I	G1	43
TØP >-I	G2	24
TØP >*I	F2	25
SYRT-I	C4	15
SYRT*I	B4	14
EPIR-I	E2	26
EPIR*I	D2	27
TEST-I	C2	48
TEST*I	B2	47
SFEC-I	F1	44
SFEC*I	D1	46

Parallel I/O Connector Pin Assignments (Cont.)

Signal Name	Connector Pin	
	Ser. No. A0100 Thru A0196	Ser. No. A0197 and Above
SFRC-I[1]	J2	22
SFRC*I[1]	A2	50
+5VC	A4	17
GND	A3	16

[1]Available at I/O connector, but not normally carried through interface cable.

INDEX

E

F

H

I

J

L

M

N

O

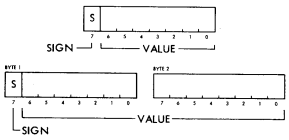
P

Q

R

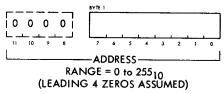
S

DATA WORD FORMATS

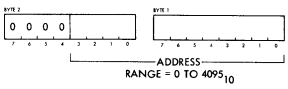


ADDRESS WORD FORMATS

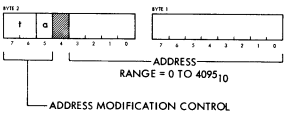
ONE-BYTE FORMAT



TWO-BYTE FORMAT

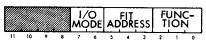


INDIRECT ADDRESS FORMAT



t = index register
a = autoincrementing

FUNCTION-ADDRESS WORD FORMAT



I/O Mode

0 0	Control signal
0 1	Data output
1 0	Data input
1 1	Test

REGISTER DESIGNATIONS

Octal Code	Register
0	A
1	X
2	Y
3	Z
4	P
5	B
6	YY
7	ZZ

SHARED COMMAND DEDICATED MEMORY

Octal Address	Instruction Group
0020 through 0037	Arithmetical/logical
0040 through 0057	Register transfer
0060 through 0077	Shift and control
0100 through 0117	Augmented memory addressing

EXTENDED MEMORY ADDRESSING

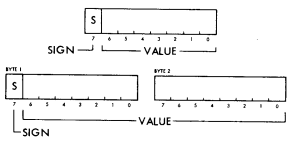
E-Register	Effective Memory
000	Block 0
001	Blocks 0 and 1
010	Blocks 0 and 2
011	Blocks 0 and 3
100	Blocks 0 and 4
101	Blocks 0 and 5
110	Blocks 0 and 6
111	Blocks 0 and 7

Each block = 2048 words

SPC-12 AUTOMATION COMPUTER

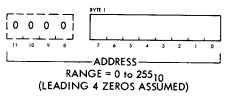
GENERAL AUTOMATION, INC.
Automation Products Division
706 West Katella, Orange, California 92668 (714) 633-1091

DATA WORD FORMATS

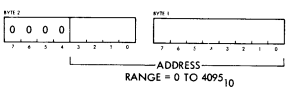


ADDRESS WORD FORMATS

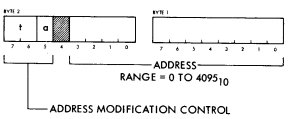
ONE-BYTE FORMAT



TWO-BYTE FORMAT

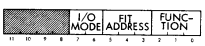


INDIRECT ADDRESS FORMAT



t = index register
a = autoincrementing

FUNCTION-ADDRESS WORD FORMAT



I/O Mode

0 0	Control signal
0 1	Data output
1 0	Data input
1 1	Test

REGISTER DESIGNATIONS

Octal Code	Register
0	A
1	X
2	Y
3	Z
4	P
5	B
6	YY
7	ZZ

SHARED COMMAND DEDICATED MEMORY

Octal Address	Instruction Group
0020 through 0037	Arithmetical/logical
0040 through 0057	Register transfer
0060 through 0077	Shift and control
0100 through 0117	Augmented memory addressing

EXTENDED MEMORY ADDRESSING

E-Register	Effective Memory
000	Block 0
001	Blocks 0 and 1
010	Blocks 0 and 2
011	Blocks 0 and 3
100	Blocks 0 and 4
101	Blocks 0 and 5
110	Blocks 0 and 6
111	Blocks 0 and 7

Each block = 2048 words

SPC-12 AUTOMATION COMPUTER

GENERAL AUTOMATION, INC.
Automation Products Division
706 West Katella, Orange, California 92668 (714) 633-1091

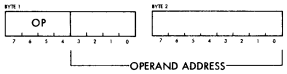
© 1968, 1969, General Automation, Incorporated

SPC-12 INSTRUCTIONS

Mnemonic		Instruction Name	Memory Cycles Required	Ref Man Page
MEMORY ADDRESSING				
LDB	m	Load B-Register	3	3-2
LDB	m,X	Load B-Register Indexed	3	3-2
STB	m	Store B-Register	3	3-3
STB	m,X	Store B-Register Indexed	3	3-3
ELB	m	Extended Load B-Register	4	3-3
JMP	m	Jump Unconditionally	2	3-3
SKIP				
SKS	n	Skip if Link Set	1	3-4
SKR	n	Skip if Link Reset	1	3-4
SKZ	n	Skip if Zero	1	3-4
SKN	n	Skip if Not Zero	1	3-4
SKP	n	Skip if Plus	1	3-5
SKM	n	Skip if Minus	1	3-5
SKT	n	Skip if I/O Test True	1	3-5
SKF	n	Skip if I/O Test False	1	3-5
ARITHMETIC/LOGICAL				
AZE	r	Zero Register	2	3-6
AND	r,B	Logical AND (B with r)	2	3-6
AND	r,v	Logical AND (v with r)	3	3-6
AXR	r,B	Logical Exclusive OR (B with r)	2	3-7
AXR	r,v	Logical Exclusive OR (v with r)	3	3-7
AOR	r,B	Logical OR (B with r)	2	3-7
AOR	r,v	Logical OR (v with r)	3	3-7
ALD	r,v	Load Register (v into r)	3	3-8
ASU	r,B	Subtract (B from r)	2	3-8
ASU	r,v	Subtract (v from r)	3	3-8
AAD	r,B	Add (B to r)	2	3-9
AAD	r,v	Add (v to r)	3	3-9
REGISTER TRANSFER				
RTR	s,d	Register Transfer	2	3-9
RIC	s,d	Register Transfer and Increment	2	3-10
RDC	s,d	Register Transfer and Decrement	2	3-10
RLK	s,d	Register Transfer and Add Link	2	3-10
SHIFT				
SHR	r	Shift Right	2	3-11
SHC	r	Shift Circular	2	3-11
SHL	r	Shift Circular Through Link	2	3-12
SHI	r	Shift Serial Data In	2	3-12
SHIO	r	Shift Serial Data In and Out	2	3-12
SHRO	r	Shift Right and Serial Data Out	2	3-13
SHCO	r	Shift Circular and Serial Data Out	2	3-13
SHLO	r	Shift Circular Through Link and Serial Data Out	2	3-13
CONTROL				
PLR		Pulse Link Reset	2	3-14
PLS		Pulse Link Set	2	3-14
PSA		Pulse Stall Alarm (optional)	2	3-15
TBB		Transfer BB- to B-Register	2	3-15
TBE		Transfer B- to E-Register	2	3-15
FOB		Function-Address Out from B-Register	2	3-15
DOB		Data Out from B-Register	2	3-16
DIB		Data Into B-Register	2	3-16
INE		Interrupt Enable	2 or 3	3-16
AUGMENTED MEMORY ADDRESSING				
GnL	r,t,I	Augmented Load	3 or 5	3-19
GnA	r,t,I	Augmented Add	3 or 5	3-19
GnS	r,t,I	Augmented Store	3 or 5	3-21
GnZ	r,t,I	Augmented Store and Zero	3 or 5	3-21
SERIAL INPUT/OUTPUT				
SHI	r	Shift Serial Data In	2	3-12
SHIO	r	Shift Serial Data In and Out	2	3-12
SHRO	r	Shift Right and Serial Data Out	2	3-13
SHCO	r	Shift Circular and Serial Data Out	2	3-13
SHLO	r	Shift Circular Through Link and Serial Data Out	2	3-13
PARALLEL INPUT/OUTPUT				
SKT	n	Skip if I/O Test True	1	3-5
SKF	n	Skip if I/O Test False	1	3-5
FOB		Function-Address Out from B-Register	2	3-15
DOB		Data Out from B-Register	2	3-16
DIB		Data Into B-Register	2	3-16

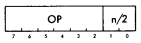
SPC-12 INSTRUCTION FORMATS

MEMORY ADDRESSING INSTRUCTIONS



Opcode	Instruction
1 1 1 0	Load B-Register
1 1 1 1	Load B-Register Indexed
1 1 0 0	Store B-Register
1 1 0 1	Store B-Register Indexed
0 1 1 1	Extended Load B-Register
0 1 1 0	Jump Unconditionally

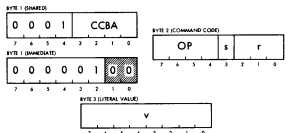
SKIP INSTRUCTIONS



Opcode	Instruction
0 1 0 1 0 1	Skip if Link Set
0 1 0 0 0 1	Skip if Link Reset
0 1 0 1 1 0	Skip if Zero
0 1 0 0 1 0	Skip if Not Zero
0 1 0 1 1 1	Skip if Plus
0 1 0 0 1 1	Skip if Minus
0 1 0 1 0 0	Skip if I/O Test True
0 1 0 0 0 0	Skip if I/O Test False

n = 0, 2, 4, or 6

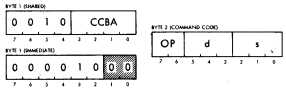
ARITHMETIC/LOGICAL INSTRUCTIONS



Opcode	S	Instruction
0 0 0 0	0	Zero Register
0 0 0 1	0	Logical AND (B with r)
0 0 0 1	1	Logical AND (v with r)
0 0 1 0	0	Logical Exclusive OR (B with r)
0 0 1 0	1	Logical Exclusive OR (v with r)
0 0 1 1	0	Logical OR (B with r)
0 0 1 1	1	Logical OR (v with r)
0 1 0 0	1	Load Register (v into r)
1 0 0 0	0	Subtract (B from r)
1 0 0 0	1	Subtract (v from r)
1 1 0 0	0	Add (B to r)
1 1 0 0	1	Add (v to r)

r = selected register
v = literal value

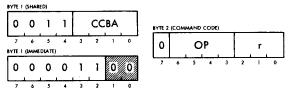
REGISTER TRANSFER INSTRUCTIONS



Opcode	Instruction
1 1	Register Transfer
0 1	Register Transfer and Increment
1 0	Register Transfer and Decrement
0 0	Register Transfer and Add Link

d = destination register s = source register

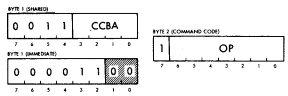
SHIFT INSTRUCTIONS



Opcode	Instruction
0 0 0 0	Shift Right
0 0 1 0	Shift Circular
0 0 0 1	Shift Circular Through Link
0 1 0 0	Shift Serial Data In
1 1 0 0	Shift Serial Data In and Out
1 0 0 0	Shift Right and Serial Data Out
1 0 1 0	Shift Circular and Serial Data Out
1 0 0 1	Shift Circular through Link and Serial Data out

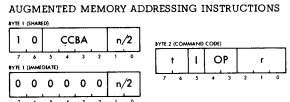
r = selected register

CONTROL INSTRUCTIONS



Opcode	Instruction
0 0 0 0 0 0 0	Pulse Link Reset
0 0 0 0 0 0 1	Pulse Link Set
1 0 0 0 0 0 0	Pulse Stall Alarm
0 0 0 0 1 0 1	Transfer BB- to B-Register
0 0 0 0 1 1 0	Transfer B- to E-Register
0 0 0 0 0 1 0	Function-Address Out from B-Register
0 0 0 0 0 1 1	Data Out from B-Register
0 0 0 0 1 0 0	Data Into B-Register
0 0 0 0 1 1 1	Interrupt Enable

AUGMENTED MEMORY ADDRESSING INSTRUCTIONS



Opcode	Instruction
0 0	Augmented Load
1 0	Augmented Add
1 1	Augmented Store
0 1	Augmented Store and Zero

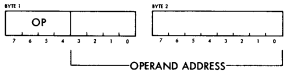
n = 0, 2, 4, 6 I = indirect addressing
t = index register r = selected register

SPC-12 INSTRUCTIONS

Mnemonic		Instruction Name	Memory Cycles Required	Ref Man Page
MEMORY ADDRESSING				
LDB	m	Load B-Register	3	3-2
LDB	m,X	Load B-Register Indexed	3	3-2
STB	m	Store B-Register	3	3-3
STB	m,X	Store B-Register Indexed	3	3-3
ELB	m	Extended Load B-Register	4	3-3
JMP	m	Jump Unconditionally	2	3-3
SKIP				
SKS	n	Skip if Link Set	1	3-4
SKR	n	Skip if Link Reset	1	3-4
SKZ	n	Skip if Zero	1	3-4
SKN	n	Skip if Not Zero	1	3-4
SKP	n	Skip if Plus	1	3-5
SKM	n	Skip if Minus	1	3-5
SKT	n	Skip if I/O Test True	1	3-5
SKF	n	Skip if I/O Test False	1	3-5
ARITHMETIC/LOGICAL				
AZE	r	Zero Register	2	3-6
AND	r,B	Logical AND (B with r)	2	3-6
AND	r,v	Logical AND (v with r)	3	3-6
AXR	r,B	Logical Exclusive OR (B with r)	2	3-7
AXR	r,v	Logical Exclusive OR (v with r)	3	3-7
AOR	r,B	Logical OR (B with r)	2	3-7
AOR	r,v	Logical OR (v with r)	3	3-7
ALD	r,v	Load Register (v into r)	3	3-8
ASU	r,B	Subtract (B from r)	2	3-8
ASU	r,v	Subtract (v from r)	3	3-8
AAD	r,B	Add (B to r)	2	3-9
AAD	r,v	Add (v to r)	3	3-9
REGISTER TRANSFER				
RTR	s,d	Register Transfer	2	3-9
RIC	s,d	Register Transfer and Increment	2	3-10
RDC	s,d	Register Transfer and Decrement	2	3-10
RLK	s,d	Register Transfer and Add Link	2	3-10
SHIFT				
SHR	r	Shift Right	2	3-11
SHC	r	Shift Circular	2	3-11
SHL	r	Shift Circular Through Link	2	3-12
SHI	r	Shift Serial Data In	2	3-12
SHIO	r	Shift Serial Data In and Out	2	3-12
SHRO	r	Shift Right and Serial Data Out	2	3-13
SHCO	r	Shift Circular and Serial Data Out	2	3-13
SHLO	r	Shift Circular Through Link and Serial Data Out	2	3-13
CONTROL				
PLR		Pulse Link Reset	2	3-14
PLS		Pulse Link Set	2	3-14
PSA		Pulse Stall Alarm (optional)	2	3-15
TBB		Transfer BB- to B-Register	2	3-15
TBE		Transfer B- to E-Register	2	3-15
FOB		Function-Address Out from B-Register	2	3-15
DOB		Data Out from B-Register	2	3-16
DIB		Data Into B-Register	2	3-16
INE		Interrupt Enable	2 or 3	3-16
AUGMENTED MEMORY ADDRESSING				
GnL	r,t,I	Augmented Load	3 or 5	3-19
GnA	r,t,I	Augmented Add	3 or 5	3-19
GnS	r,t,I	Augmented Store	3 or 5	3-21
GnZ	r,t,I	Augmented Store and Zero	3 or 5	3-21
SERIAL INPUT/OUTPUT				
SHI	r	Shift Serial Data In	2	3-12
SHIO	r	Shift Serial Data In and Out	2	3-12
SHRO	r	Shift Right and Serial Data Out	2	3-13
SHCO	r	Shift Circular and Serial Data Out	2	3-13
SHLO	r	Shift Circular Through Link and Serial Data Out	2	3-13
PARALLEL INPUT/OUTPUT				
SKT	n	Skip if I/O Test True	1	3-5
SKF	n	Skip if I/O Test False	1	3-5
FOB		Function-Address Out from B-Register	2	3-15
DOB		Data Out from B-Register	2	3-16
DIB		Data Into B-Register	2	3-16

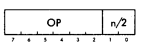
SPC-12 INSTRUCTION FORMATS

MEMORY ADDRESSING INSTRUCTIONS



Opcode	Instruction
1 1 1 0	Load B-Register
1 1 1 1	Load B-Register Indexed
1 1 0 0	Store B-Register
1 1 0 1	Store B-Register Indexed
0 1 1 1	Extended Load B-Register
0 1 1 0	Jump Unconditionally

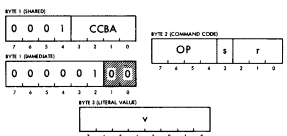
SKIP INSTRUCTIONS



Opcode	Instruction
0 1 0 1 0 1	Skip if Link Set
0 1 0 0 0 1	Skip if Link Reset
0 1 0 1 1 0	Skip if Zero
0 1 0 0 1 0	Skip if Not Zero
0 1 0 1 1 1	Skip if Plus
0 1 0 0 1 1	Skip if Minus
0 1 0 1 0 0	Skip if I/O Test True
0 1 0 0 0 0	Skip if I/O Test False

n = 0, 2, 4, or 6

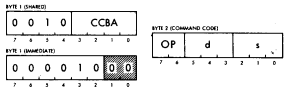
ARITHMETIC/LOGICAL INSTRUCTIONS



Opcode	S	Instruction
0 0 0 0	0	Zero Register
0 0 0 1	0	Logical AND (B with r)
0 0 0 1	1	Logical AND (v with r)
0 0 1 0	0	Logical Exclusive OR (B with r)
0 0 1 0	1	Logical Exclusive OR (v with r)
0 0 1 1	0	Logical OR (B with r)
0 0 1 1	1	Logical OR (v with r)
0 1 0 0	1	Load Register (v into r)
1 0 0 0	0	Subtract (B from r)
1 0 0 0	1	Subtract (v from r)
1 1 0 0	0	Add (B to r)
1 1 0 0	1	Add (v to r)

r = selected register
v = literal value

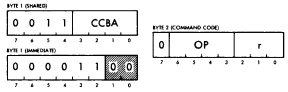
REGISTER TRANSFER INSTRUCTIONS



Opcode	Instruction
1 1	Register Transfer
0 1	Register Transfer and Increment
1 0	Register Transfer and Decrement
0 0	Register Transfer and Add Link

d = destination register s = source register

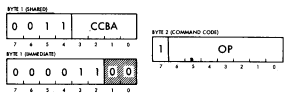
SHIFT INSTRUCTIONS



Opcode	Instruction
0 0 0 0	Shift Right
0 0 1 0	Shift Circular
0 0 0 1	Shift Circular Through Link
0 1 0 0	Shift Serial Data In
1 1 0 0	Shift Serial Data In and Out
1 0 0 0	Shift Right and Serial Data Out
1 0 1 0	Shift Circular and Serial Data Out
1 0 0 1	Shift Circular through Link and Serial Data out

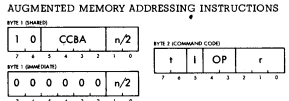
r = selected register

CONTROL INSTRUCTIONS



Opcode	Instruction
0 0 0 0 0 0 0	Pulse Link Reset
0 0 0 0 0 0 1	Pulse Link Set
1 0 0 0 0 0 0	Pulse Stall Alarm
0 0 0 0 1 0 1	Transfer BB- to B-Register
0 0 0 0 1 1 0	Transfer B- to E-Register
0 0 0 0 0 1 0	Function-Address Out from B-Register
0 0 0 0 0 1 1	Data Out from B-Register
0 0 0 0 1 0 0	Data Into B-Register
0 0 0 0 1 1 1	Interrupt Enable

AUGMENTED MEMORY ADDRESSING INSTRUCTIONS

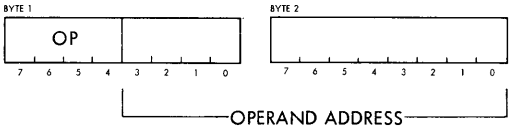


Opcode	Instruction
0 0	Augmented Load
1 0	Augmented Add
1 1	Augmented Store
0 1	Augmented Store and Zero

n = 0, 2, 4, 6 I = indirect addressing
t = index register r = selected register

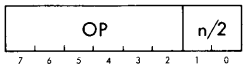
SPC-12 INSTRUCTION FORMATS

MEMORY ADDRESSING INSTRUCTIONS



Opcode	Instruction
1 1 1 0	Load B-Register
1 1 1 1	Load B-Register Indexed
1 1 0 0	Store B-Register
1 1 0 1	Store B-Register Indexed
0 1 1 1	Extended Load B-Register
0 1 1 0	Jump Unconditionally

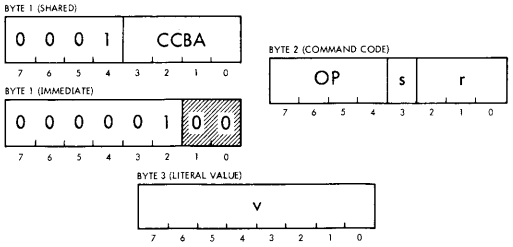
SKIP INSTRUCTIONS



Opcode	Instruction
0 1 0 1 0 1	Skip if Link Set
0 1 0 0 0 1	Skip if Link Reset
0 1 0 1 1 0	Skip if Zero
0 1 0 0 1 0	Skip if Not Zero
0 1 0 1 1 1	Skip if Plus
0 1 0 0 1 1	Skip if Minus
0 1 0 1 0 0	Skip if I/O Test True
0 1 0 0 0 0	Skip if I/O Test False

n = 0, 2, 4, or 6

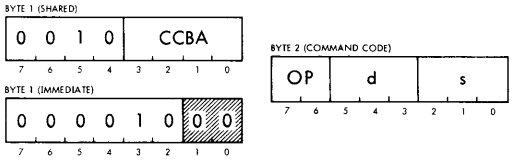
ARITHMETIC/LOGICAL INSTRUCTIONS



Opcode	S	Instruction
0 0 0 0	0	Zero Register
0 0 0 1	0	Logical AND (B with r)
0 0 0 1	1	Logical AND (v with r)
0 0 1 0	0	Logical Exclusive OR (B with r)
0 0 1 0	1	Logical Exclusive OR (v with r)
0 0 1 1	0	Logical OR (B with r)
0 0 1 1	1	Logical OR (v with r)
0 1 0 0	1	Load Register (v into r)
1 0 0 0	0	Subtract (B from r)
1 0 0 0	1	Subtract (v from r)
1 1 0 0	0	Add (B to r)
1 1 0 0	1	Add (v to r)

r = selected register
v = literal value

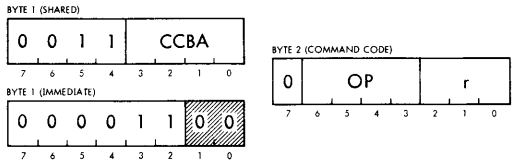
REGISTER TRANSFER INSTRUCTIONS



Opcode	Instruction
1 1	Register Transfer
0 1	Register Transfer and Increment
1 0	Register Transfer and Decrement
0 0	Register Transfer and Add Link

d = destination register s = source register

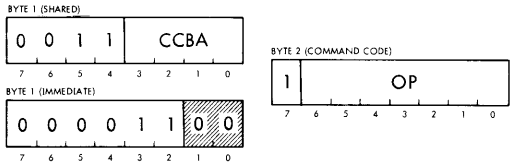
SHIFT INSTRUCTIONS



Opcode	Instruction
0 0 0 0	Shift Right
0 0 1 0	Shift Circular
0 0 0 1	Shift Circular Through Link
0 1 0 0	Shift Serial Data In
1 1 0 0	Shift Serial Data In and Out
1 0 0 0	Shift Right and Serial Data Out
1 0 1 0	Shift Circular and Serial Data Out
1 0 0 1	Shift Circular through Link and Serial Data out

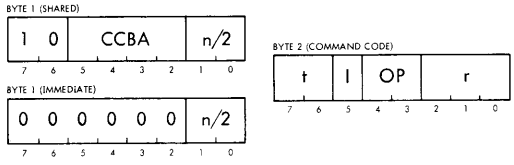
r = selected register

CONTROL INSTRUCTIONS



Opcode	Instruction
0 0 0 0 0 0 0	Pulse Link Reset
0 0 0 0 0 0 1	Pulse Link Set
1 0 0 0 0 0 0	Pulse Stall Alarm
0 0 0 0 1 0 1	Transfer BB- to B-Register
0 0 0 0 1 1 0	Transfer B- to E-Register
0 0 0 0 0 1 0	Function-Address Out from B-Register
0 0 0 0 0 1 1	Data Out from B-Register
0 0 0 0 1 0 0	Data Into B-Register
0 0 0 0 1 1 1	Interrupt Enable

AUGMENTED MEMORY ADDRESSING INSTRUCTIONS



Opcode	Instruction
0 0	Augmented Load
1 0	Augmented Add
1 1	Augmented Store
0 1	Augmented Store and Zero

n = 0, 2, 4, 6 I = indirect addressing
t = index register r = selected register