RCA 4102 PROGRAMMERS REFERENCE MANUAL

TP1145A

MAY 1963

RCA 4102

PROGRAMMERS REFERENCE MANUAL

TP1145A

FOREWORD

This manual presents the functional organization and the machine-language programming characteristics of the RCA 4102 Data Processing System. It will serve as an aid in the training of programmers and will be useful as a reference source for those familiar with the system.

Peripheral-equipment descriptions, operating procedures, software specifications, and programming examples will be found in other publications issued by RCA Data Systems Division for the RCA 4102 Data Processing System.

Address inquiries concerning this manual to:

Manager, Computer Applications
RCA, Data Systems Division
Van Nuys, California

TABLE OF CONTENTS

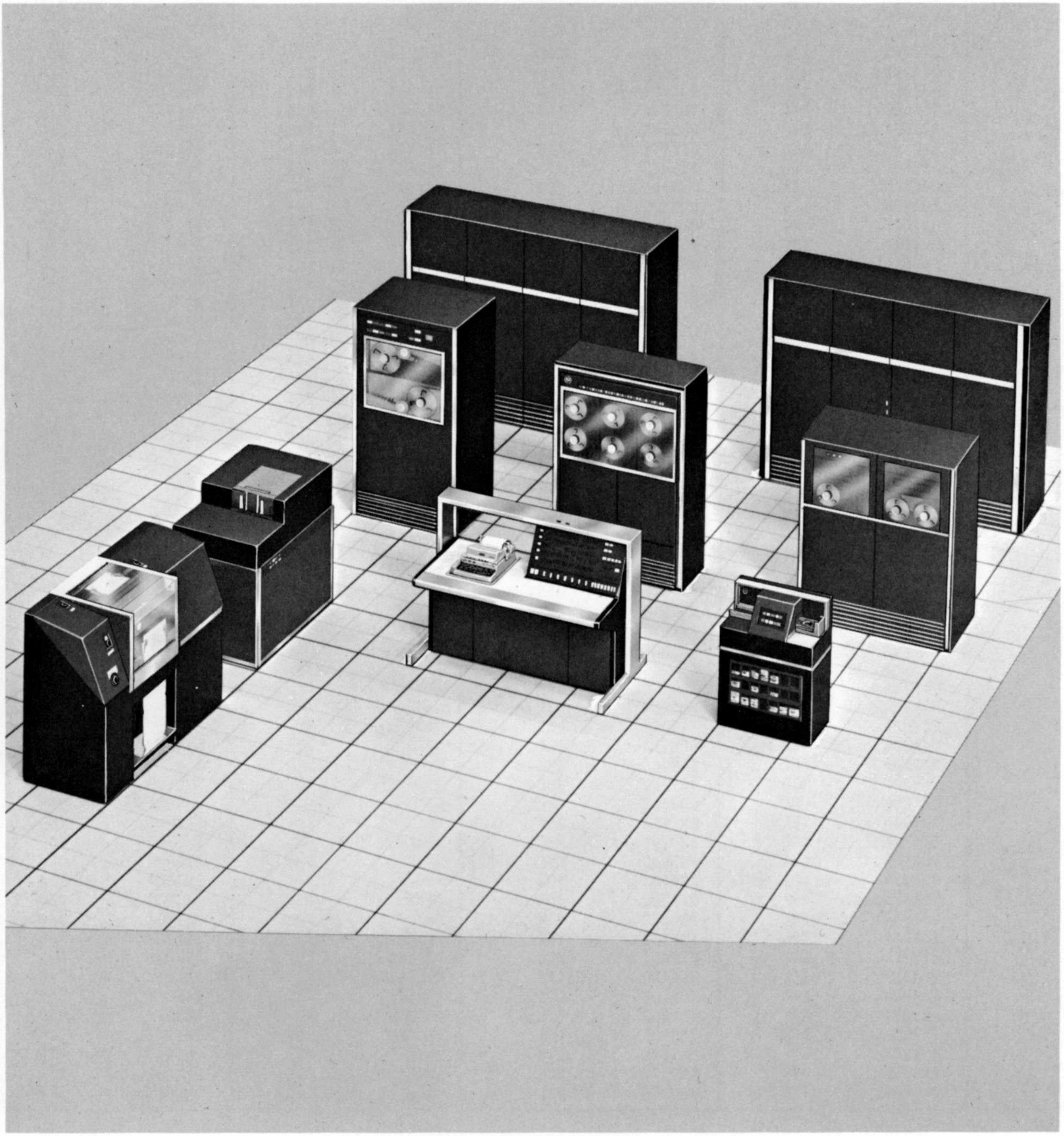
TABLE OF CONTENTS (CONT)

LIST OF ILLUSTRATIONS

LIST OF TABLES

RCA 4102 DATA PROCESSOR

RCA 4102 DATA PROCESSING SYSTEM



SECTION 1
INTRODUCTION

The RCA 4102 Data Processing System is a completely transistorized general-purpose digital computer system economically providing excellent performance in both data processing and real-time applications. This system, and its counterpart — the RCA 4101 (AN designation CP-685/GPQ) — are members of the RCA Data Systems Division's 4100 product line. Its components are selected, designed, and built to a combined military specification, based upon MIL-E-16400C, MIL-E-4158B, and XEL303.

A prime feature of the RCA 4102 is the priority-interrupt system, which provides for servicing a multiplicity of peripheral devices concurrently. Sixteen priorities are available for assignment to the computing programs and to peripheral devices.

When a program or device requires service, it sets a flag corresponding to the priority demanding attention. During each instruction, the priority flags are automatically examined, causing a computer transfer to the highest priority routine demanding attention. Each priority may be interrupted to allow servicing of a higher priority. When all higher priorities are dormant, control returns to the priority of the interrupted program.

The basic system consists of an RCA 4102 Central Processing Unit (CPU) with 4096 words of core memory and a Control Desk with Flexowriter. This system is expandable in modular fashion to provide memory capacities of 8192 or 16,384 words.

The RCA 4102 CPU is a parallel, bus-organized digital computer, capable of simultaneous communication with a multiplicity of peripheral devices. The standard complement of peripheral equipment includes card readers, card punches, magnetic tapes and drums, paper tapes, and line printers, with binary and alphanumeric formats and with operating speeds and characteristics compatible with data-processing industry standards. The RCA 4102 communicates efficiently with radars, data links, teletypes, telemetry equipments, analog-digital converters, plotters, and associated equipment.

On the Control Desk are installed the Flexowriter and a Control Panel. The Flexowriter can transmit data to the CPU through its paper-tape reader and can punch tape and print copy, either selectively or simultaneously, under computer or manual control.

The Control Panel provides the means for operator control of the CPU. It contains sets of indicators for displaying the contents of the CPU registers and a set of manual input switches for inserting information into the CPU. Controls are also provided to facilitate manual program-tracing and computer maintenance.

SECTION 2 CENTRAL PROCESSING UNIT

FUNCTIONAL ORGANIZATION

The inherent organizational simplicity of the RCA 4102 Central Processing Unit (CPU) is illustrated in the functional block diagram of Figure 2-1.

Control Console

On the RCA 4102 Control Console are a Flexowriter and a Control Panel for monitoring and controlling the Central Processing Unit.

The Flexowriter may be operated either on-line or off-line to permit:

- Manual preparation of typed copy
- Manual preparation of punched tape
- Automatic off-line listing from punched tape
- Automatic tape duplication
- Manual tape editing
- On-line reading of punched tape to CPU
- On-line preparation of punched tape by CPU

The Control Panel provides visual indicators and manual controls to enable the operator to:

- Insert and execute programs
- Observe the contents of the AC register, the Flags, various control registers, or any memory location
- Change the contents of any of the above
- Manually trace programs in various increments (through a single instruction, to the next program branch point, or to a preselected location)
- Perform diagnostic maintenance of the CPU

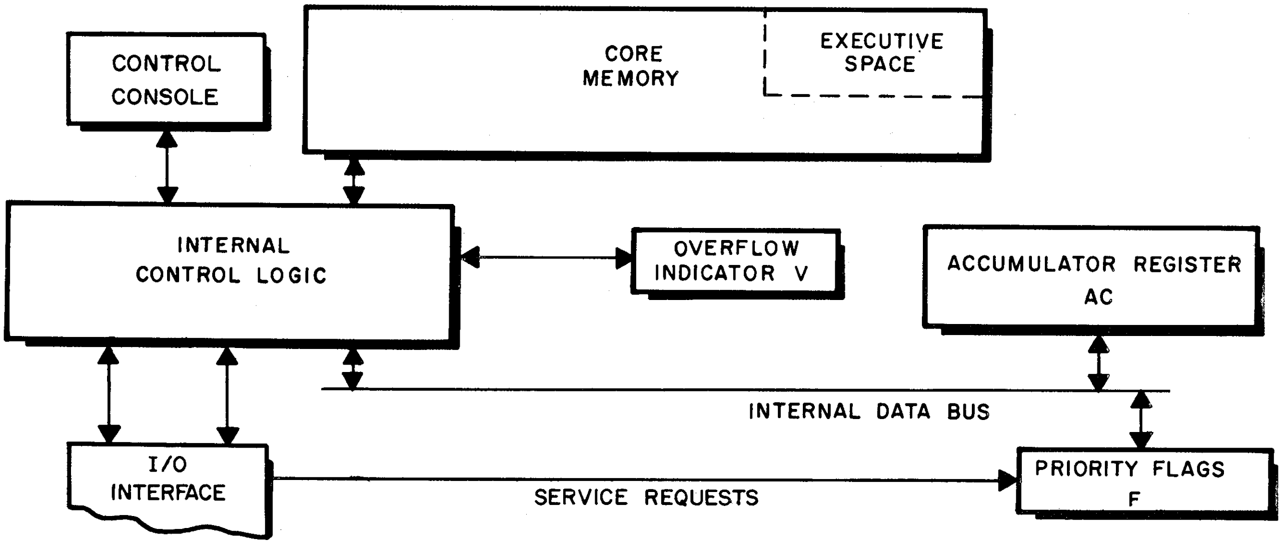


Figure 2-1. RCA 4102 CPU, Functional Block Diagram

I/O Interface

Communication between the RCA 4102 CPU and all peripheral devices or subsystems takes place through common line-matching circuits located in the Model 4111 Peripheral Equipment Control Unit Shell (PECUS).

Core Memory

The RCA 4102 uses a random-access magnetic-core memory for the internal storage of instructions, constants, data, and results. Dual significance is attached to the block of 128 core-storage words at the upper end of the memory, termed the "executive space" (see Appendix J). The executive space is subdivided into 16 sections of 8 words each. Although these locations can be addressed and treated as normal storage space, they are also used to control the 16 programs which the computer is capable of handling. (Refer to Priority Interrupt.) In installations containing 8192 words of memory, all addresses are automatically interpreted modulo 2^{13}; i.e., octal addresses 17654 and 37654 are equivalent and refer to the same memory location. Similarly, in installations with 4096 words of memory, addresses are automatically interpreted modulo 2^{12}.

Parity Check. — Two parity-check bits are generated and stored with every computer word in memory. Whenever a word is read from memory, each halfword is checked to assure that its parity is still "odd." If a "parity error" is sensed, control automatically transfers to a supervisory control program which is responsible for taking appropriate action to maintain system integrity. (If a "parity error" occurs OFF-LINE, the CPU stops with the PARITY ERROR indicator on the Control Panel illuminated.)

Instruction-Location Counters

There are 16 Instruction-Location Counters, IL, stored in the memory executive space (see Appendix J). The contents of these counters are used to specify the core-memory address of instructions executed by the CPU in each of the 16 priority levels. During the execution of each instruction, the appropriate IL value is modified to the address of the next instruction to be executed in the associated priority level. Unless the programmer has specified a program branch or jump action, this modification is achieved by automatically decrementing the IL Value by one (final value = initial value -1). Thus, instructions are usually performed in descending sequence. Refer to Instruction-Location Counter Word Format.

Index Registers

For each of the 16 priority levels, there are six Index Registers stored in the memory executive space (see Appendix J). The contents of these registers may be used by the programmer for automatic address modification, iterative loop control, or subroutine linkages. Refer to Index-Register Word Format.

Accumulator Register

The 30-bit Accumulator Register, AC, retains the result of most arithmetic, logic, and shift instructions, and generally supplies one operand for such instructions.

Overflow Indicator

The Overflow Indicator, V, is set whenever the result of an arithmetic instruction exceeds the RCA 4102 data-word capacity. Refer to Word Format and Overflow Detection.

Flags

The priority-interrupt feature is implemented by means of 15 flip-flops, comprising the Flag Register, each of which functions as a priority-request indicator, or "Flag." Each Flag may be set either under program control or by external equipment. A Flag can be cleared only by an instruction (with S = 1) executed in the corresponding priority level. An implied sixteenth stage (lowest priority) is considered permanently set, and assumes control whenever no other request is present.

PRIORITY-INTERRUPT SYSTEM

The priority-interrupt feature of the RCA 4102 provides for servicing a multiplicity of programs on a priority basis. The primary purpose of the priority-interrupt system is to eliminate the need for program-controlled scheduling and status-testing of input-output transfers.

The complete set of indicator Flags is examined, simultaneously with the execution of each instruction, to determine which demanding program has the highest priority. When the currently active program has the highest priority of those demanding attention, the program proceeds without interruption. Otherwise, the system automatically transfers, at the completion of the current instruction, to the highest demanding priority.

During the priority-interrupt cycle, the machine logic ensures that the address of the next instruction of the interrupted priority is saved in its Instruction-Location Counter. The only programming constraint is that each program which affects the Accumulator Register content is responsible for restoring the original value. When all the higher priority demands are satisfied, control returns to the interrupted program, and the latter proceeds oblivious of the interruption. In this way, the priority-interrupt feature enables the data processor to interleave a number of different programs.

Priority Assignment

Input-output routines for the RCA 4102 are generally short, often a single instruction which does not affect the contents of the arithmetic unit. Once the data have been transferred to memory, suitable programs are called on to process the new information. The various input-output and data-handling routines are assigned different priority levels; therefore, the programmer need not perform periodic tests for the arrival of data. The priority-interrupt system automatically activates the proper program to process the new data which have been transferred into the memory by other programs.

Because of these considerations, assignment of relative priority among the various input-output programs and devices is principally based on the allowable waiting time between data transfers. Computing programs are assigned lower priorities than those of input-output programs, and may be assigned arbitrarily, within this constraint, by the user. The lowest priority program, automatically executed when the Flag Register is cleared, is usually a self-check routine, which automatically exercises and tests the computer during idle time.

Priority-Interrupt Example

As an illustration of the operation of the priority-interrupt system, consider that the program in priority level N is currently active (see Figure 2-2).

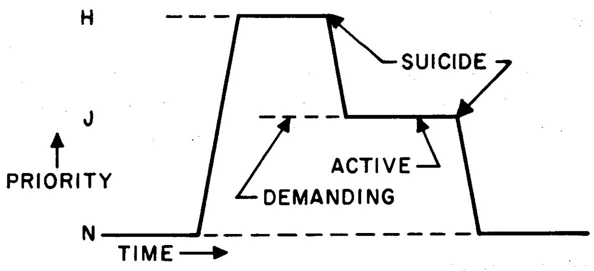


Figure 2-2. Priority Interrupt

Assume that one of the instructions in this program sets the higher priority Flag H. During the final timing steps of this instruction, the Instruction-Location Counter (stored in memory) is modified in the normal fashion to the value IL_N. The interrupt-control logic is interrogated, and the interrupt is sensed.

The next instruction to be executed by the system is located at an address specified by IL_H. Assume that this program will alter the AC; an early instruction in program H saves AC_N in a work space reserved by program H. At the conclusion of each instruction, the interrupt-control logic is interrogated to determine whether another interrupt is to occur. If a higher priority Flag is demanding, the system will enter the interrupt mode again.

Assume that no higher priority Flag is set; program H continues to operate. During the execution of program H, it is quite possible for a number of lower priority Flags to be set either by external devices or by an instruction within this program. These lower priority Flags do not interrupt program H. Before the conclusion of program H, an instruction retrieves AC_N from the work space of program H and places it in the AC register. The final instruction of this program clears priority Flag H.

The interrupt-control logic now indicates that a change in programs is required. Assume that Flag J is now the highest priority set and that the program associated with priority J will also modify the AC. As before, an early instruction saves AC_N in a priority J work space. Assume no higher priority interrupt occurs; the program executed in priority J operates until its final instruction clears Flag J. Near the end of this program, AC_N is again placed in the AC register.

This process of transferring AC_N from the AC register to a given program work space and back to the register continues until the system priority level returns to N. At this point, AC_N is in AC, IL_N is interrogated to gain access to the next instruction, and program N proceeds as if no interruption had occurred.

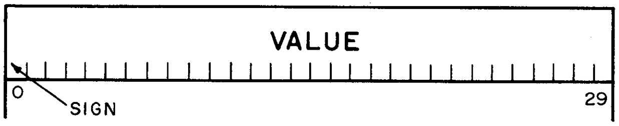
Note that no concurrent program was required to perform bookkeeping for other programs, except for the storage and retrieval of the AC content.

WORD FORMATS

All information in the RCA 4102 CPU is handled in parallel, binary words of 30 bits each. The various formats for these words are described in the following text.

Data Word

The RCA 4102 CPU data word is interpreted as a binary fraction using two's complement notation.

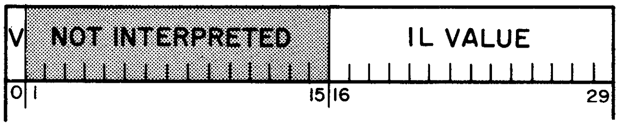


The range of numbers that can be represented by the data word extends from -1.0 to $1.0-2^{-29}$ (octal representation from 4000000000 to 3777777777 respectively). The octal representation of the number "zero" is always 0000000000 and can never have a negative sign. Numbers outside this range are handled by appropriate scaling.

Appendix A contains a discussion of the arithmetic used by the RCA 4102 CPU and the rules for the conversion of decimal, octal, and binary integers and fractions from one number system to another.

Instruction-Location Counter Word

Each of the 16 Instruction-Location words (when used as an IL) is interpreted according to the format below:



Bits 16-29 of the IL associated with the active priority contain the address at which the current instruction is stored. At the end of each instruction, the IL Value is suitably modified to contain the address of the next instruction.

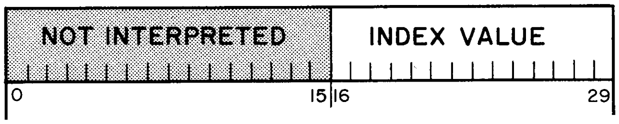
Bits 16-29 of the IL associated with each dormant priority contain the address of the instruction to be executed when its priority becomes active.

Bit 0 is automatically set to "one" whenever overflow occurs in an instruction executed in the associated priority (see Overflow). Bit 0 can be cleared automatically only by a suitably coded JAC or JSX instruction. This bit may also be modified by absolute addressing of the IL executive location.

Bits 1-15 are automatically set to "one's" at the end of each instruction executed in the associated priority level.

Index-Register Word

Each of the 96 Index-Register words (when used as an Index Register) is interpreted according to the format below:



Bits 16-29 are interpreted as the Index Value.

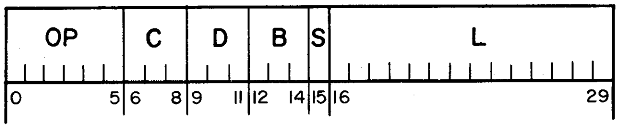
Bits 0-15 vary according to the manner in which the Index Value was last modified, as tabulated below:

INDEX VALUE (last changed by)	BIT 0	BITS 1-14	BIT 15
C = 1 or 2 (Table 2-1)	Indeterminate	Index Value	Indeterminate
ALX(C ≠ 1, 2)	77777		Indeterminate
JSX	77777		Indeterminate
PAX	$C(AC)_{0-15}$		
LDX	$C(L)_{0-15}$		

Instruction Word

Words used as instructions are interpreted according to the format below:

XXX name execution time



This diagram appears with the discussion of each operation in the repertoire, showing the mnemonic code (illustrated here as XXX) with the operation name and execution time above the diagram and with its octal number equivalent in the OP field.

With few exceptions, each instruction word specifies three independent operations to be executed simultaneously:

- The OP, B, and L fields comprise the "primary instruction".
- The C and D fields comprise the "secondary instruction".
- The S field controls the "suicide" action.

Operations may be referred to either by the equivalent octal number or by the descriptive mnemonic code. The mnemonic code is a three-letter abbreviation for the operation, and usually has a direct association with the operation name; such as ADD for the Add operation and STI for Store Input Word. The complete repertoire of operations in functional, alphabetic, and octal sequence is listed in the Appendixes for ready reference.

Primary Instruction. — A standard single-address format specifies the principal function to be performed:

- OP - bits 0-5 — The OP code specifies the primary operation to be performed by this instruction.
- L - bits 16-29 — The L field of the instruction word is termed the "base address", and usually contains the memory address of an item of data involved in the operation.
- B - bits 12-14 — The B field designates which Index Register, if any, is to be used in the execution of the primary instruction. Interpretation of this field is:

B = 0	No Index Register, Index Value = 0
B = 1-6	Designates one of the six Index Registers associated with the active-priority level.
B = 7	Designates, as the Index Value, the value of the Instruction-Location Counter associated with the active-priority level.

In most instructions, the specified Index Value is algebraically added -- modulo 2^{14} -- to the base address, L, of the instruction; the resulting sum is termed the effective address, E. The instruction is then executed as if the address field had contained E. The original value, L, in the address portion of the instruction word is unchanged by this operation. As a result, the same instruction may be used to operate on data stored in different areas of the memory.

Index Registers are also used to count and/or to control the number of times an instruction or group of instructions is executed. Each of 16 priority levels in the RCA 4102 has its own set of six Index Registers, for a total of 96, thereby making it unnecessary to store and retrieve the Index Values when a priority interrupt occurs.

Secondary Instruction. — In conjunction with certain primary OP codes, the secondary instruction fields have special interpretations. Unless otherwise noted, these fields are interpreted in the following "Normal" manner:

- C - bits 6-8 — The C field is the secondary operation code, which specifies conditions for branching and/or subsidiary functions (Table 2-1).

- D - bits 9-11 — The D field determines the destination address if the branch condition specified by the C field is met. When this occurs, the content of the D field is added to the content of the Instruction-Location Counter to yield the destination address.

  The value of D may range from zero through seven. D = 0 causes the current instruction to be repeated if the condition specified by C is satisfied. Other D values, with a satisfied normal C condition, cause the RCA 4102 to jump back D locations in memory to obtain the next instruction word.

Suicide. — "Suicide" is the term applied to the voluntary act of relinquishing control from the active priority level.

- S - bit 15 – A "one" in this bit of any instruction word causes the active-priority flag to be cleared. (Exceptions: see TST, SKP, STI, LDØ, IØC instructions.)

TABLE 2-1.
SECONDARY OPERATION CODES (NORMAL)

MNE-MONIC	OCTAL		BRANCH CONDITION	SUBSIDIARY FUNCTION
	C	D		
—	0	even	No jump	None
IØV	0	odd	No jump	Do not set V
JXC	1*	0-7	Initial $C(B)_{16-29} \neq 0$	C(B)-1 → C(B)
DIX	2*	0-7	No jump	C(B)-1 → C(B)
APZ	3	0-7	Final C(AC) $\geq$ 0	None
AGZ	4	0-7	Final C(AC) > 0	None
ANZ	5	0-7	Final C(AC) $\neq$ 0	None
AEZ	6	0-7	Final C(AC) = 0	None
ALZ	7	0-7	Final C(AC) < 0	None

* C = 1 or 2 must not be used with primary instructions LDZ, FLS, STX, LDX, PAX, and PXA.

OVERFLOW

Whenever the algebraic result of any addition, subtraction, division, or algebraic shift exceeds the capacity of the RCA 4102 data word, the Overflow Indicator, V, is set. Unless the instruction causing the overflow condition includes an IØV secondary instruction, a "one" is stored in the most significant (sign) bit position of the Instruction-Location Counter of the active-priority level. The IL Value is updated in the normal manner.

Once the overflow-memory bit has been set in the IL counter, V is automatically set prior to the execution of each instruction in this priority level. The overflow indication, both in V and in the IL sign bit, is thus perpetuated until a JAC or JSX instruction coded to clear the overflow condition is executed. Intervening instructions, regardless of C and D field coding, have no effect on the overflow indication.

The overflow indication can be used to control program branching in the active-priority level by employing a JAC or JSX instruction with a JØV condition. Overflow indications in <u>any</u> priority level can be sensed by programmed testing of the sign of the appropriate IL counter (See Appendix J).

Overflow Trap (Optional)

The Overflow Trap option provides a fail-safe response to unanticipated overflows. When this option is used, the occurrence of any overflow condition will automatically set the Flag of a preselected priority level, unless:

- The instruction causing the overflow is coded with an IØV secondary instruction to suppress the indication, or
- The next instruction executed in the same priority level is a JAC or JSX with a JØV condition, which performs a conditional branch dependent on the overflow indication.

TIMING

The execution time of any instruction in the RCA 4102 CPU requires a certain number of storage references. In general, the arithmetic, logic, and control actions required are performed simultaneously with these memory-access cycles, but some primary operations need additional logic time. The execution time is listed in the form: x cycles + y microseconds.

The duration of a "cycle" is 5.5 microseconds for the 4096/8192-word memory; 7.0 microseconds for the 8192/16384-word memory.

For any instruction which is indexed (B=1-6), one additional cycle should be counted. Note, however, that this cycle is included in the basic timing of the LDX, STX, PAX, PXA, JIX, JSX, and ALX instructions.

Any change in the active priority level (whether due to external interrupt, internal FLS instruction, or Suicide action) requires one additional cycle.

SECTION 3
INPUT-OUTPUT

DEFINITIONS

For the purpose of this discussion, three categories of input-output peripheral devices are defined:

- Autonomous Device — initiates or terminates communication, independent of computer control. Data links, radars, analog-digital converters, telemetry, teletypes, and humans are examples of Autonomous Devices.

- Unit-Transfer Device — communicates a single computer word, or less, in response to each control command it receives. Flexowriters, perforated-tape equipment, discrete-pulse channels, plot-boards, and teletypes are examples of Unit-Transfer Devices.

- Block-Transfer Device — communicates a finite sequence of computer words in response to each control command it receives. Magnetic-tape stations, punched-card equipment, line printers, and disk files are examples of Block-Transfer Devices.

These categories are not mutually exclusive: the decision to implement a particular device in one or another category depends on the transfer data rate, the manner in which the device is to be used, and the number and nature of other peripheral devices in the system. It should be noted that virtually all peripheral devices, regardless of category, are "asynchronous", in the sense that their transfer rates are independent of internal CPU timing.

For efficient asynchronous communication with, and control of, devices in each of the three categories, the RCA 4102 employs the following programming techniques.

AUTONOMOUS DEVICE

Each Autonomous Device generates a "service request" signal whenever it is ready for the transfer of a computer word to or from the CPU. This service request sets a preassigned priority Flag in the CPU. The routine executed in this priority contains a single-instruction loop to perform the required data transfer.

The single instruction (STI or LDØ) is executed once in response to each service request, transferring one word to or from memory, and returning control — by use of the S field — to the computing routines.

Using an Index Register to tally the service requests, the program can automatically notify another routine whenever a predetermined number of words has been transferred. This automatic notification relieves the programmer of any need for schedule control or periodic status-testing in the servicing of peripheral devices.

When several devices or channels are connected to the system, each responds only to STI and/or LDØ instructions executed in the priority level assigned to the device. The active-priority code thus serves as a "peripheral address" for input-output communication.

UNIT-TRANSFER DEVICE

Communication between the RCA 4102 CPU and a Unit-Transfer Device is similar to that described for Autonomous Devices. The distinction lies in the need for a command from the CPU to control the operation of the device. This command is encoded into the C field of the STI or LDØ instruction which performs the data transfer. For example, an LDØ instruction, which issues data to the Flexowriter, simultaneously directs whether the data should be punched or typed, or both.

After each service request has been processed and when the device is ready for the next data transfer, the Unit-Transfer Device sets its priority Flag, automatically notifying the CPU.

Several similar or dissimilar Unit-Transfer Devices can be time-shared on a common priority level, provided no ambiguity or conflict arises in the interpretation of the C fields of the input-output instructions. As many as eight input and eight output devices can thus be serviced by each priority level. In this way, the C field of these instructions becomes a "subsidiary peripheral address", supplementing the active-priority code for selecting and controlling peripheral devices.

BLOCK-TRANSFER DEVICE

For efficient communication, each Block-Transfer Device is coupled to the RCA 4102 CPU through a Control Unit. A number of similar Block-Transfer Devices may communicate with the CPU through a single Control Unit. Each such Control Unit is connected to a specific CPU priority level, in which all data transfers (STI and LDØ) for this unit are executed.

Control commands, for directing the action of these Block-Transfer Control Units and their associated devices, are issued from the CPU by use of an IØC instruction. The IØC instruction, which may be executed in <u>any</u> priority level, transmits to the peripheral equipment a 15-bit Control Command which identifies:

- Priority level of the Control Unit addressed
- Specific device to be selected (if necessary)
- Action to be performed
- Type of data block

Once the action has been initiated, and for the duration of the specified data block, any required data transfers are performed as described for Autonomous or Unit-Transfer Devices.

Status Indicators

At the conclusion of this block action, the Block-Transfer Control Unit transmits to the CPU a "status indicator" word. The sign bit and the five least significant bits of this word are coded to indicate the status of the peripheral device and the validity of the data transmitted, if any. In some cases, the status indicators are requested by the program; in other cases, the control unit volunteers the status information.

CALL-BACK LOGIC

The Call-Back Logic contains 15 auxiliary flags, one of which is set whenever an IØC instruction is permitted to suicide. The auxiliary flag to be set corresponds to the priority level from which the IØC is issued. Whenever any Block-Transfer Control Unit completes its action, the auxiliary flags which are set cause the setting of the corresponding priority-interrupt Flags. The auxiliary flags are then cleared.

All priorities in which IØC commands have been rejected are thus re-entered, so that a new attempt may be made for each. In this way, the Call-Back Logic performs the task of notifying any priority, which is dormant and waiting to use a Block-Transfer Control Unit, that such a control unit has completed its action.

EXAMPLE

To illustrate the use of the above techniques, assume that priority A is connected through a Tape-Control Unit to six magnetic-tape stations, of which the third (A3) and fifth (A5) contain data needed by the programs being executed in priorities J and M, respectively. Priority P contains a self-exercise program, executed whenever the other priorities are temporarily dormant.

When the program in priority J requires, as input, the next data record (of unspecified length) from tape A3, priority J executes an IØC instruction which addresses the A3 tape. This IØC instruction is coded with D = 0 (i.e., repeat this instruction) and S = 1 (i.e., suicide). Assume the Tape-Control Unit connected to priority A is now available for use; a control signal, DSI, inhibits both the suicide and the repeat action; and the command issued by the IØC lays "claim" to Tape-Control Unit A and to tape A3.

Program J now proceeds to initialize the IL counter of priority A, specifying the location of the input routine to be used for the required data transfer. When this initialization is complete, the next instruction executed in priority J is an IØC which sets the tape in motion to read one record. Priority J now executes another IØC addressed to Tape-Control Unit A (e.g., attempting to release the claim on A3). Since Tape-Control Unit A is now busy, this command cannot be accepted. With D = 0 and S = 1 in this instruction, priority J prepares to repeat the attempt and clears its Flag to await completion of the tape operation. The combination of IØC rejection and S = 1 causes auxiliary flag J to be set in the Call-Back Logic. Lower-priority programs (M or P) may now proceed.

When the tape mechanism has attained operating speed and has transmitted the first data word to Tape-Control Unit A, the Tape-Control Unit automatically sets priority Flag A. Priority A becomes active and executes a single STI instruction. With D = 0, S = 1, and B = 1 - 6, the STI not only reads the data to memory, but also decrements the Index Register, prepares to repeat, and clears priority Flag A. Other programs may now proceed until the next data word from the tape is ready. This sequence is repeated for each word in the tape record.

Ultimately, Tape-Control Unit A senses the end of the tape record, stops the tape motion, and sets Flag A once more. The STI instruction is executed again, this time reading into memory the status indicators from Tape-Control Unit A. A DSI signal from Tape-Control Unit A causes the immediate execution of the next sequential instruction in priority A. The residual content of the Index Register (indicative of the record length) is saved for future use in priority J, and priority A suicides. Program J is reactivated by the Call-Back Logic, repeats the previous IØC (now accepted) to "release" its claim on tape A3, and processes the requested data.

If, when program J first tried to claim tape A3, Tape-Control Unit A had already been busy (e.g., processing tape A5 for priority M), no DSI signal would have been generated. The suicide of program J would be permitted, leaving auxiliary flag J set in the Call-Back Logic.

GLOSSARY

The following symbols are used in the description of the Instruction Repertoire:

Symbol	Meaning
A	A field, bits 15-20, of IØC command word
AC	Accumulator Register
B	B field, bits 12-14, of instruction word
C	C field, bits 6-8, of instruction word (secondary operation code)
D	D field, bits 9-11, of instruction word (secondary instruction address)
E	Effective address: If B = 0, E = L If B = 1-6, E = L + C(B) If B = 7, E = L + C(IL)
IL	Instruction-Location Counter of active priority
L	Base address, bits 16-29, of instruction word
R	R field, bits 23-26, of IØC command word
S	S field, bit 15, of instruction word (suicide control)
T	T field, bits 21-22, of IØC command word
U	U field, bits 27-29, of IØC command word
$(\ )_{16-29}$	Subscripts are used to reference particular bits (stages) of a word (register)
C( )	Content of
C(AC)	Content of Accumulator Register
C(B)	Content of Index Register specified by B; often means $C(B)_{16-29}$ when unambiguous
C(E)	Content of Memory location specified by E
C(L)	Content of Memory location specified by L
C(IL)	Content of Instruction-Location Counter of active priority; often means $C(IL)_{16-29}$ when unambiguous
$\longrightarrow$	Replaces
$\longleftarrow$	Is replaced by
$\cap$	AND operation
$\cup$	Inclusive OR operation
$\oplus$	Exclusive OR operation
+ -x/	Arithmetic operations
| |	Magnitude (absolute value) of
▨	Field not interpreted by RCA 4102; unless otherwise specified, any bits in this field will not affect, nor be affected by, the operation.
Not allowed	Undefined condition(s) which may conflict with instruction execution

SECTION 4 INSTRUCTION REPERTOIRE

WORD-TRANSMISSION OPERATIONS

STA — Store AC 3 cycles

41	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(AC) replaces the C(E).

RESULTS	VARIATIONS
C(AC) Unchanged	C - Normal (Appendix F)
C(E) ← C(AC)	D - Normal
C(B) Unchanged*	B - Normal
Overflow not possible	S - Normal

LDA — Load AC 3 cycles

61	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(E) replaces the C(AC).

RESULTS	VARIATIONS
C(AC) ← C(E)	C - Normal (Appendix F)
C(E) Unchanged	D - Normal
C(B) Unchanged*	B - Normal
Overflow not possible	S - Normal

XAS — Exchange AC and Storage 3 cycles + 1 μs

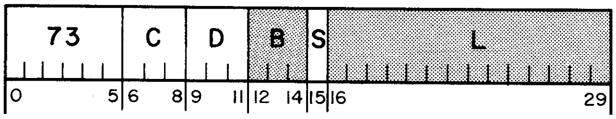
57	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(AC) is exchanged with the C(E).

RESULTS	VARIATIONS
C(AC) ← C(E)	C - Normal (Appendix F)
C(E) ← C(AC)	D - Normal
C(B) Unchanged*	B - Normal
Overflow not possible	S - Normal

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

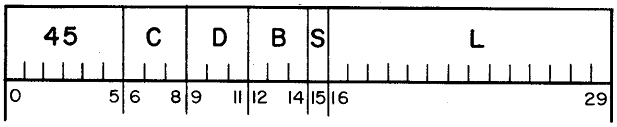
LDZ — Load Zero 2 cycles



The C(AC) is cleared to zero.

RESULTS	VARIATIONS
C(AC) ← 0	C - Normal (Appendix F)
Overflow not possible	D - Normal
	B - Not interpreted
	S - Normal

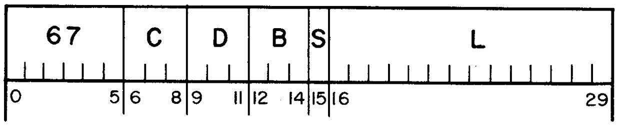
STZ — Store Zero 3 cycles



The C(E) is cleared to zero.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(E) ←	0	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

LDX — Load Index 4 cycles



If B = 1-6, the $C(L)_{0-29}$ replaces the $C(B)_{0-29}$.

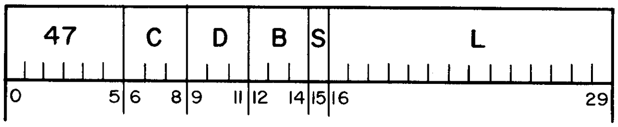
If B = 0, the $C(L)_{0-29}$ is placed in the executive work space for this priority.

B = 7 is not allowed.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(L)	Unchanged	D - Normal
C(B) ←	C(L)	B - See LDX text
Overflow not possible		S - Normal

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

STX — Store Index 4 cycles

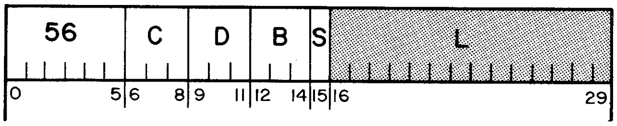


The $C(B)_{0-29}$ replaces the $C(L)_{0-29}$.

B = 0 or 7 is not allowed.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(L) ←	C(B)	D - Normal
C(B)	Unchanged	B - See STX text
Overflow not possible		S - Normal

PAX — Place AC in Index 3 cycles

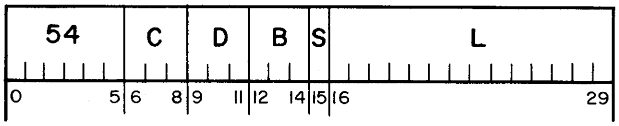


The $C(AC)_{0-29}$ replaces the $C(B)_{0-29}$.

B = 0 or 7 is not allowed.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(B) ←	C(AC)	D - Normal
Overflow not possible		B - See PAX text
		S - Normal

PXA — Place Index in AC 3 cycles



If L = 37777_8, the $C(B)_{16-29}$ (Index Value) replaces the $C(AC)_{16-29}$.

If L ≠ 37777_8, each bit of the $C(B)_{16-29}$ is matched with the corresponding bit position of the instruction word. For each "one" bit in the instruction word, the corresponding bit in the C(B) is placed in the corresponding bit position of the AC. For each "zero" bit in the instruction word, the complement of the corresponding bit in the C(B) is placed in the corresponding bit position of the AC.

If S = 1, the $C(B)_{15}$ replaces the $C(AC)_{15}$ and suicide occurs.

If S = 0, the complement of the $C(B)_{15}$ replaces the $C(AC)_{15}$.

In any case, the $C(B)_{0-14}$ is placed in the AC_{0-14}.

B = 0 or 7 is not allowed.

Example:

	OP	C	D	B	S	L
Instruction =	101 100	011	010	010	0	01 110 011 110 110
C(B) =	000 101	110	011	110	0	00 101 110 011 110
Final C(AC) =	000 101	110	011	110	1	10 100 010 010 111

RESULTS		VARIATIONS
C(AC)	See PXA text	C - Normal (Appendix F)
C(L)	Unchanged	D - Normal
C(B)	Unchanged	B - See PXA text
Overflow not possible		S - Normal

LOGIC OPERATIONS

ANA — AND to AC 3 cycles

34	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

Each bit of the C(AC) is matched with the corresponding bit of the C(E). For each "one" bit in the C(E), the corresponding bit in the C(AC) is unchanged. For each "zero" bit in the C(E), the corresponding bit in the C(AC) is cleared to "zero".

RESULTS		VARIATIONS
C(AC) ←	C(AC) ∩ C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

ANR — AND and Replace 3 cycles

35	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs an ANA, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	C(AC) ∩ C(E)	C - Normal (Appendix F)
C(E) ←	C(AC) ∩ C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

ERA — Exclusive ØR to AC 3 cycles

36	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

Each bit of the C(AC) is matched with the corresponding bit of the C(E). For each "zero" bit in the C(E), the corresponding bit in the C(AC) is unchanged. For each "one" bit in the C(E), the corresponding bit in the C(AC) is complemented.

RESULTS		VARIATIONS
C(AC) ←	C(AC) ⊕ C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

ERR — Exclusive ØR and Replace 3 cycles

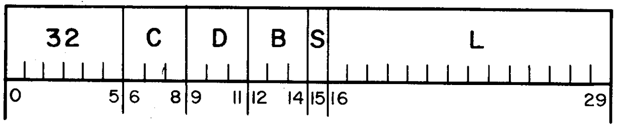
37	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs an ERA, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	C(AC) ⊕ C(E)	C - Normal (Appendix F)
C(E) ←	C(AC) ⊕ C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

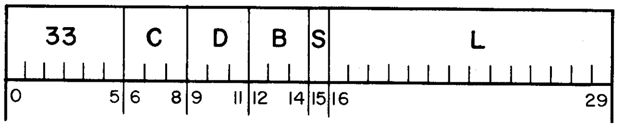
ØRA — ØR to AC 3 cycles



Each bit of the C(AC) is matched with the corresponding bit of the C(E). For each "zero" bit in the C(E), the corresponding bit in the C(AC) is unchanged. For each "one" bit in the C(E), the corresponding bit in the C(AC) is set to "one".

RESULTS		VARIATIONS
C(AC) ←	C(AC) ∪ C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

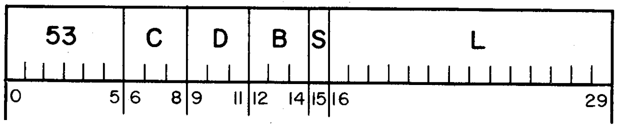
ØRR — ØR and Replace 3 cycles



This instruction performs an ØRA, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	C(AC) ∪ C(E)	C - Normal (Appendix F)
C(E) ←	C(AC) ∪ C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

ØRS — ØR to Storage 3 cvcles



Each bit of the C(E) is matched with the corresponding bit of the C(AC). For each "zero" bit in the C(AC), the corresponding bit in the C(E) is unchanged. For each "one" bit in the C(AC), the corresponding bit in the C(E) is set to "one".

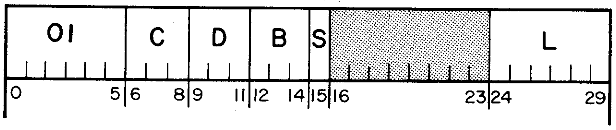
RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(E) ←	C(E) ∪ C(AC)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

SHIFT OPERATIONS

Bits 24-29 of the effective address, E, control the shift direction and distance:

- Bit 24 of the effective address is interpreted as an algebraic sign: positive (zero) for right shift, negative (one) for left shift.
- Bits 25-29 of the effective address are interpreted as the number of places to be shifted. If bit 24 is "zero", bits 25-29 specify the right shift distance; if bit 24 is "one", bits 25-29 specify the two's complement of the left shift distance.
- Shifts of 0 through 31 places right or 1 through 32 places left are possible.
- Execution time is 2 cycles + 2 μs for shifting one or two places left, or zero or one place right. Each additional two places require 2.0 microseconds.
- No discontinuity occurs when using an Index Register to modify the shift distance. For example, a shift distance of minus two (-2) modified by an Index Value of plus five (+5) will result in a right shift of three places.
- Bits 16-23 of shift instruction words are not interpreted.
- B = 7 is not allowed.

SAL — Shift AC Logical 2 cycles + 2-32 μs



This instruction shifts the $C(AC)_{0-29}$ right or left E places.

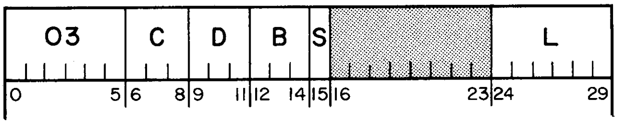
When shifting left, bits shifted out of AC_0 are discarded; vacated positions of AC are cleared to zero.

When shifting right, bits shifted out of AC_{29} are discarded; vacated positions of AC are cleared to zero.

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

RESULTS		VARIATIONS
C(AC)	See SAL text	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal, except B=7 not allowed
Overflow not possible		S - Normal

SAC — Shift AC Circular 2 cycles + 2-32 µs



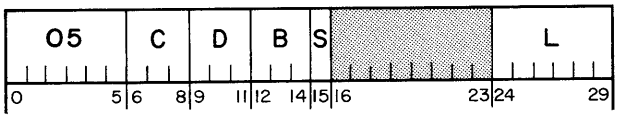
This instruction rotates the $C(AC)_{0-29}$ right or left E places.

When rotating left, bits shifted out of AC_0 enter AC_{29}.

When rotating right, bits shifted out of AC_{29} enter AC_0.

RESULTS		VARIATIONS
C(AC)	See SAC text	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal, except B=7 not allowed
Overflow not possible		S - Normal

SAA — Shift AC Algebraic 2 cycles + 2-32 µs



This instruction shifts the $C(AC)_{0-29}$ right or left E places.

When shifting left, bits shifted out of AC_0 are discarded; vacated positions of AC are cleared to zero. The OVERFLOW indicator is set if the $C(AC)_0$ changes at any time during the process.

When shifting right, bits shifted out of AC_{29} are discarded; the $C(AC)_0$ is unchanged; and vacated positions of AC are replaced by the $C(AC)_0$. Overflow is not possible.

RESULTS		VARIATIONS
C(AC)	$\leftarrow C(AC)/2^E$	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal, except B=7 not allowed
Overflow possible during left shift		S - Normal
Overflow not possible during right shift		

The examples shown in Table 4-1 illustrate the effect of the various shift operations. Each instruction operates on the C(AC) remaining after the previous instruction.

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

TABLE 4-1. SHIFT OPERATION EXAMPLES

PRIMARY INSTRUCTION	C(AC)											EXECUTION TIME
	OCTAL	BINARY										
(Initial AC)	6102451357	110	001	000	010	100	101	001	011	101	111	—
01...03	0610245135	000	110	001	000	010	100	101	001	011	101	2 cycles + 4 µs
01...72	1024513500	001	000	010	100	101	001	011	101	000	000	6
03...14	3500102451	011	101	000	000	001	000	010	100	101	001	14
03...75	5001024513	101	000	000	001	000	010	100	101	001	011	4
05...03	7500102451	111	101	000	000	001	000	010	100	101	001	4
05...75	5001024510	101	000	000	001	000	010	100	101	001	000	4
05...75	0010245100	000	000	001	000	010	100	101	001	000	000	4
		(overflow)										
05...67	0245100000	000	010	100	101	001	000	000	000	000	000	2 cycles + 10 µs
		(overflow)										

ARITHMETIC OPERATIONS

ADD — Add 3 cycles

10	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(E) is algebraically added to the C(AC), and the resulting sum is placed in the AC.

If the algebraic sum exceeds $+1-2^{-29}$ or is less than -1, the OVERFLOW indicator, V, is set, and the residue (modulo 2.0) replaces the C(AC). Refer to Number System discussion in Appendix A.

RESULTS		VARIATIONS
C(AC) ←	C(AC) + C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

ADR — Add and Replace 3 cycles

11	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs an ADD, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	C(AC) + C(E)	C - Normal (Appendix F)
C(E) ←	C(AC) + C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

ADM — Add Magnitude 3 cycles

12	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(E) is added to the magnitude of the C(AC), and the resulting sum is placed in the AC.

If the algebraic sum exceeds $+1-2^{-29}$, the OVERFLOW indicator, V, is set, and the residue (modulo 2.0) replaces the C(AC). Refer to Number System discussion in Appendix A.

RESULTS		VARIATIONS
C(AC) ←	|C(AC)|+C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged *	B - Normal
Overflow possible		S - Normal

AMR — Add Magnitude and Replace 3 cycles

13	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs an ADM, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	|C(AC)|+C(E)	C - Normal (Appendix F)
C(E) ←	|C(AC)|+C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

SUB — Subtract 3 cycles

14	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(AC) is algebraically subtracted from the C(E), and the resulting difference is placed in the AC.

If the algebraic difference exceeds $+1-2^{-29}$ or is less than -1, the OVERFLOW indicator, V, is set, and the residue (modulo 2.0) replaces the C(AC). Refer to Number System discussion in Appendix A.

RESULTS		VARIATIONS
C(AC) ←	-C(AC)+C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

SBR — Subtract and Replace 3 cycles

15	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs an SUB, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	-C(AC) + C(E)	C - Normal (Appendix F)
C(E) ←	-C(AC) + C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

SBM — Subtract Magnitude 3 cycles

16	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The magnitude of the C(AC) is algebraically subtracted from the C(E), and the resulting difference is placed in the AC.

If the algebraic difference is less than -1, the OVERFLOW indicator, V, is set, and the residue (modulo 2.0) replaces the C(AC). Refer to Number System discussion in Appendix A.

RESULTS		VARIATIONS
C(AC) ←	-|C(AC)| + C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

SMR — Subtract Magnitude and Replace 3 cycles

17	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs an SBM, and replaces both the C(AC) and the C(E) with the result.

RESULTS		VARIATIONS
C(AC) ←	-|C(AC)| + C(E)	C - Normal (Appendix F)
C(E) ←	-|C(AC)| + C(E)	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

MPY — Multiply 3 cycles + 59 μs

22	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(AC) is multiplied by the C(E), and the most significant 30 bits of the product are placed in the AC. The less significant part of the product is discarded.

Note: The product of (-1) x (-1) is represented as $(+1-2^{-29})$.

RESULTS		VARIATIONS
C(AC) ←	C(AC) x C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

MPH — Multiply Half-Word 3 cycles + 29 μs

23	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The $C(AC)_{0-14}$ is multiplied by the C(E), and the most significant 30 bits of the product are placed in the AC. The less significant part of the product is discarded.

Note: The product of (-1) x (-1) is represented as $(+1-2^{-29})$.

RESULTS		VARIATIONS
C(AC) ←	$C(AC)_{0-14}$ x C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow not possible		S - Normal

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

DIV — Divide 3 cycles + 57 μs

24	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

If the magnitude of the C(E) exceeds the magnitude of the C(AC), the C(AC) is divided by the C(E), and the most significant 30 bits of the resulting quotient are placed in the AC.

If the magnitude of the C(AC) exceeds the magnitude of the C(E), the quotient sign replaces the $C(AC)_0$, the $C(AC)_{1-29}$ is meaningless, and the OVERFLOW indicator, V, is set.

If the magnitudes of the C(E) and the C(AC) are equal, the quotient replaces the C(AC). The OVERFLOW indicator, V, is set if the C(E) is negative. The quotient is -1 if the operand signs were unlike; the quotient is $+1-2^{-29}$ if the operand signs were alike.

RESULTS		VARIATIONS
C(AC) ←	C(AC)/C(E)	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - Normal
Overflow possible		S - Normal

DVH — Divide Half-Word 3 cycles + 29 μs

25	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

This instruction performs a DIV, but forms only the most significant 15 bits of the quotient, which are placed in the AC_{0-14}. The AC_{15-29} is cleared to zero. (See DIV for overflow conditions.)

RESULTS		VARIATIONS
$C(AC)_{0-14}$ ←	C(AC)/C(E)	C - Normal (Appendix F)
$C(AC)_{15-29}$ ←	0	D - Normal
C(E)	Unchanged	B - Normal
C(B)	Unchanged*	S - Normal
Overflow possible		

BRANCH OPERATIONS

With these five instructions, whose primary function is program branch control, the C-D fields are not interpreted as a secondary instruction. The special uses of these fields are presented with the description of each primary operation.

JSX — Jump and Set Index 3 cycles

72	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

If the branch condition specified by the C and D fields (see Table 4-2) is satisfied, the C(IL) replaces the C(B), and the next instruction to be executed in the current priority level is taken from memory location L.

If the branch condition specified by the C and D fields is not satisfied, no action is taken, the execution time is reduced to 2 cycles, and the RCA 4102 proceeds to the next instruction in sequence.

B = 0 or 7 is not allowed.

TABLE 4-2. JSX CONDITIONS

SYMBOL	OCTAL CD†	BRANCH CONDITION	AUXILIARY FUNCTION
—	02	Unconditional	
JØV	03	V = 1	0→V→ $C(IL)_0$
APZ	32	$C(AC) \geq 0$	
AGZ	42	$C(AC) > 0$	
ANZ	52	$C(AC) \neq 0$	
AEZ	62	$C(AC) = 0$	
ALZ	72	$C(AC) < 0$	
† Other C-D combinations are not allowed			

RESULTS		VARIATIONS
C(AC)	Unchanged	C - See Table 4-2
C(L)	Unchanged	D - See Table 4-2
C(B)	See JSX text	B - See JSX text
Overflow not possible		S - Normal

JAC — Jump on AC 2 cycles

72	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

If the branch condition specified by the C and D fields (see Table 4-3) is satisfied, the next instruction to be executed in the current priority level is taken from memory location E.

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

If the branch condition specified by the C and D fields is not satisfied, no action occurs, and the RCA 4102 proceeds to the next instruction in sequence.

TABLE 4-3. JAC CONDITIONS

SYMBOL	OCTAL CD*	BRANCH CONDITION	AUXILIARY FUNCTION
—	00	Unconditional	
JØV	01	V = 1	$0 \rightarrow V \rightarrow C(IL)_0$
APZ	30	C(AC) ≥ 0	
AGZ	40	C(AC) > 0	
ANZ	50	C(AC) ≠ 0	
AEZ	60	C(AC) = 0	
ALZ	70	C(AC) < 0	
*Other C-D combinations are not allowed			

RESULTS		VARIATIONS
C(AC)	Unchanged	C - See Table 4-3
C(E)	Unchanged	D - See Table 4-3
C(B)	Unchanged	B - Normal
Overflow not possible		S - Normal

JIX — Jump on Index 3 cycles

72	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

If the branch condition specified by the C and D fields (see Table 4-4) is satisfied, the C(B) is decremented by one, and the next instruction to be executed in the current priority level is taken from memory location L.

If the branch condition specified by the C and D fields is not satisfied, the C(B) is decremented by one, and the RCA 4102 proceeds to the next instruction in sequence.

B = 0 or 7 is not allowed.

TABLE 4-4. JIX CONDITIONS

SYMBOL	OCTAL CD*	BRANCH CONDITION	AUXILIARY FUNCTION
JXC	10	Initial C(B) ≠ 0	C(B)-1 → C(B)
JXD	20	Unconditional	C(B)-1 → C(B)
*Other C-D combinations are not allowed			

RESULTS		VARIATIONS
C(AC)	Unchanged	C - See Table 4-4
C(L)	Unchanged	D - See Table 4-4
C(B)	← C(B)-1	B - See JIX text
Overflow not possible		S - Normal

TST — Test Storage 3 cycles

77	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(E) is tested as specified by the C field (see Table 4-5).

If the condition is satisfied, the RCA 4102 jumps back D instructions -- i.e., C(IL) + D → C(IL) -- and proceeds from there. The Suicide field is interpreted normally.

If the condition is not satisfied, the RCA 4102 proceeds to the next instruction in sequence. Suicide is inhibited.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - See Table 4-5
C(E)	Unchanged	D - Normal
C(B)	Unchanged	B - Normal
Overflow not possible		S - See TST text

SKP — Skip 3 cycles

76	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

The C(E) is tested as specified by the C field (see Table 4-5).

If the condition is satisfied, the RCA 4102 skips the next D instructions -- i.e., C(IL) - 1 - D → C(IL) -- and proceeds from there. The Suicide field is interpreted normally.

If the condition is not satisfied, the RCA 4102 proceeds to the next instruction in sequence. Suicide is inhibited.

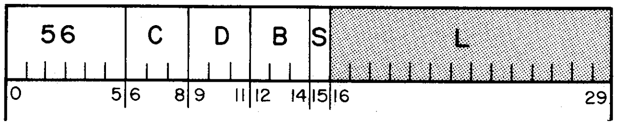
RESULTS		VARIATIONS
C(AC)	Unchanged	C - See Table 4-5
C(E)	Unchanged	D - See SKP text
C(B)	Unchanged	B - Normal
Overflow not possible		S - See SKP text

TABLE 4-5. SKP/TST CONDITIONS

SYMBOL	OCTAL C	BRANCH CONDITION
EGZ	0	C(E) > 0
ENZ	1	C(E) ≠ 0
EEZ	2	C(E) = 0
ELZ	3	C(E) < 0
EGA	4	C(E) > C(AC)
ENA	5	C(E) ≠ C(AC)
EEA	6	C(E) = C(AC)
ELA	7	C(E) < C(AC)

CONTROL OPERATIONS

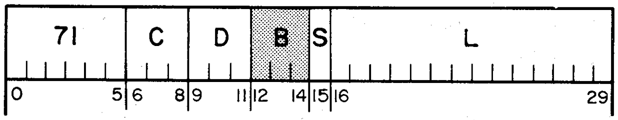
NØP — No Operation 3 cycles



No action occurs, and the RCA 4102 proceeds to the next instruction in sequence, unless the secondary instruction causes branching.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
Overflow not possible		D - Normal
		B - Must be zero
		S - Normal

FLS — Flag Set 2 cycles



The bits of L_{16-29} are matched with priority Flags B through Ø, respectively (Table 4-6). For each bit of L which contains a "zero", the corresponding Flag is unchanged. For each bit of L which contains a "one", the corresponding Flag is set. The bit of L corresponding to the priority Flag of the current program has no effect.

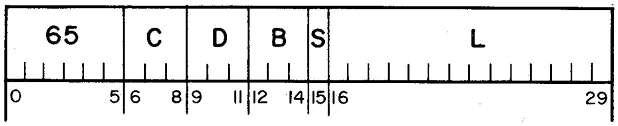
More than one Flag may be set with an FLS instruction. For example: if L = 13201_8, Flags C, E, F, H, and Ø would be set.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(L)	Unchanged	D - Normal
Overflow not possible		B - Not interpreted
		S - Normal

TABLE 4-6. FLS ADDRESSES

PRIORITY	L FIELD
A	(Not affected)
B	20000
C	10000
D	04000
E	02000
F ✓	01000
G	00400
H	00200
I	00100
J	00040
K ✓	00020
L ✓	00010
M	00004
N	00002
Ø	00001
P	(Never cleared)

ALX — Add L to Index 3 cycles



The value of L (interpreted as a 14-bit unsigned integer) is added to the $C(B)_{16-29}$, and the resulting sum (modulo 2^{14}) replaces the $C(B)_{16-29}$. This action may produce a net increase or decrease in the index value. Refer to Number System discussion in Appendix A.

The $C(B)_{15}$ is meaningless.

If the secondary operation code in this instruction is JXC (C = 1) and $C(B)_{16-29} + L > 37777_8$ (net decrease in index value), the RCA 4102 jumps back D instructions -- i.e., C(IL) + D→ C(IL) -- and proceeds from there. Otherwise (with C = 1), the RCA 4102 proceeds to the next instruction in sequence.

The DIX secondary operation code (C = Z) in this instruction has no effect, except for the "twinning" of the index register. Refer to Index-Register Word Format.

Other secondary operation codes are interpreted normally.

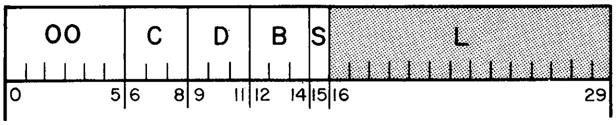
B = 0 or 7 is not allowed.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - See ALX text
C(L)	Unchanged	D - Normal
C(B) ←	C(B) + L	B - See ALX text
Overflow not possible		S - Normal

TABLE 4-7. ALX EXAMPLES

Instruction Word	Initial C(B)	Final C(B)	Remarks
6500300001	5432103245	7777703246	
6500340001	5432103245	7777743246	Note effect of S-bit in instruction word
6514300001	5432103245	0324603246	No branch
6514300001	2345677777	0000000000	Jump back 4
6512300020	3141637756	3777637776	No branch
6512300020	3141637766	4000640006 (effectively +6)	Jump back 2
6510337777	5432103245	4324443244	Repeat instruction
6510337777	0000000000	3777737777	No branch*
6510337777	4000040000	7777777777	No branch*
*These two examples are indistinguishable.			

HLT — Halt 3 cycles



The interpretation of this instruction depends on the position of the ON-LINE/OFF-LINE console switch.

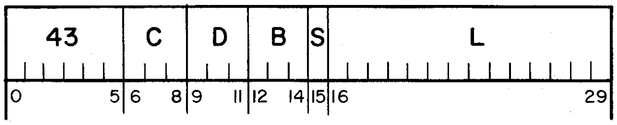
If the RCA 4102 is ON-LINE, suicide occurs regardless of the S-field coding. An Overflow indication is forced, setting the $C(IL)_0$ = 1, and Overflow Trapping (optional) automatically occurs. The IL value is updated in the normal manner.

If the RCA 4102 is OFF-LINE, the CPU stops with the values of E and the C(E) displayed on the Control Console. No instruction will be executed in any priority level until the CPU is manually restarted. When restarted, the RCA 4102 proceeds to the next instruction in sequence, unless the C field has caused branching or the S field has caused suicide of this priority level.

RESULTS		VARIATIONS
C(AC)	Unchanged	C - Normal (Appendix F)
C(E)	Unchanged	D - Normal
C(B)	Unchanged*	B - May specify index register to be tested
Overflow not possible OFF-LINE		S - Normal OFF-LINE
Overflow forced ON-LINE		Forced ON-LINE

INPUT-OUTPUT OPERATIONS

STI — Store Input Word 3 cycles



This instruction replaces the C(E) with the input word.

The secondary instruction executed simultaneously with the STI is that which would normally occur with C = 1. The actual C field is used as a "subsidiary peripheral address" for device and/or function selection (see STI/LDØ C Field Assignments) and controls the transmission of data from an external device to the CPU.

If a DSI signal is received from a Block-Transfer Control Unit during execution of this instruction, the normal actions of the D and S fields are inhibited, and the RCA 4102 proceeds to the next instruction in sequence (see Block-Transfer Devices).

RESULTS		VARIATIONS
C(AC)	Unchanged	C - See STI text
C(E) ←	Input word	D - See STI text
C(B) ←	C(B) -1	B - Normal
Overflow not possible		S - See STI text

*By primary instruction; C(B) may be changed by secondary instruction (see Appendix F)

LDØ — Load Output Word 3 cycles + 8 μs

63	C	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

If B ≠ 7 the C(E) is transmitted to an output device.

If B = 7, the C(AC) is transmitted to an output device and L is not interpreted.

The secondary instruction executed simultaneously with the LDØ is that which would normally occur with C = 1. The actual C field is used as a "subsidiary peripheral address" for device and/or function selection (see STI/LDØ C Field Assignments), and controls the acceptance of data from the CPU by an external device.

If a DSI signal is received from a Block-Transfer Control Unit during execution of this instruction, the normal actions of the D and S fields are inhibited, and the RCA 4102 proceeds to the next instruction in sequence (see Block-Transfer Devices).

RESULTS

C(AC)	Unchanged
C(E)	Unchanged
C(B) ←	C(B) - 1
Overflow not possible	

VARIATIONS

C - See LDØ text
D - See LDØ text
B - Normal, except B=7 transmits the C(AC)
S - See LDØ text

IØC — Input-Output Control 3 cycles + 8 μs

62	O	D	B	S	L
0 5	6 8	9 11	12 14	15	16 29

If B ≠ 7, the C(E) is transmitted to a Block-Transfer Control Unit.

If B = 7, the C(AC) is transmitted to a Block-Transfer Control Unit, and L is not interpreted.

The control unit interprets the transmitted word as a control command which it may "accept" or "reject" (see IØC Control Commands).

If the control unit "accepts" the command, it transmits a DSI signal (see Block-Transfer Devices) to the CPU, and the peripheral action proceeds to completion without further program intervention. The DSI signal inhibits the normal actions of the D and S fields of the IØC instruction issuing the command, and the CPU proceeds to the next instruction in sequence.

If the control unit "rejects" the command, no action is taken by the peripheral equipment, no DSI signal is generated, and the CPU proceeds as specified by the D and S fields of the IØC instruction. In this case, if the IØC instruction contains S = 1 (suicide), an auxiliary flag is set in the Call-Back Logic.

C must be zero. Branching, as specified by the D field of the IØC instruction word, is conditional only on receipt of the DSI signal.

RESULTS

C(AC)	Unchanged
C(E)	Unchanged
C(B)	Unchanged
Overflow not possible	

VARIATIONS

C - Must be zero
D - Normal
B - Normal, except B = 7 transmits the C(AC)
S - See IØC text

SECTION 5 PERIPHERAL EQUIPMENT CONTROL CODING

INTRODUCTION

This section presents the addressing and control techniques used with the STI, LDØ, and IØC instructions and illustrates their use with the present RCA 4102 catalogue line of peripheral devices. The information included here will be supplemented as the catalogue is expanded.

STI/LDØ C-FIELD ASSIGNMENTS

As previously noted, the C field of the STI and LDØ instructions is used as a "subsidiary peripheral address", supplementing the active-priority code (see Unit-Transfer Devices) for selecting and controlling peripheral devices. The codes to be used in this field for the control of any specific peripheral channel may, in general, be assigned arbitrarily at the time of design of the associated buffer or control unit.

Flexowriter

The assignments which have been made for interpretation of C fields in STI and LDØ instructions, when executed in the priority level(s) associated with the standard Flexowriter are:

- STI, C = 3 — Advance paper tape one character space; store previous character in E_{24-29}, clearing the $C(E)_{0-23}$; set input priority flag when the next character is ready.

- LDØ, C = 2 — Type the character contained in E_{24-29} or AC_{24-29}; do not change the C(E) nor the C(AC); set output priority flag when the next character can be accepted. (See Appendix I for character codes.)

- LDØ, C = 4 — Punch the character contained in E_{24-29} or AC_{24-29}; do not change the C(E) nor the C(AC); set output priority flag when the next character can be accepted.

- LDØ, C = 6 — Type and punch the character contained in E_{24-29} or AC_{24-29}; do not change the C(E) nor the C(AC); set output priority flag when the next character can be accepted. (See Appendix I for character codes.)

High-Speed Paper-Tape Reader-Punch

The assignments which have been made for interpretation of C fields in STI and LDØ instructions, when executed in the priority level associated with the High-Speed Paper-Tape Reader and Punch are:

- STI, C = 2 — Start (continue) advancing paper tape; store current character in E_{23-29}, clearing the $C(E)_{1-22}$, and inserting a "one" in E_0 only if a parity or timing error has been detected; set priority flag when the next character is ready.
- STI, C = 1 — Stop tape motion at next character; store current character in E_{23-29}, clearing the $C(E)_{1-22}$, and inserting a "one" in E_0 only if a parity or timing error has been detected; set priority flag when the stopping point has been reached.
- LDØ, any C value — Punch the character contained in E_{23-29} or AC_{23-29}; do not change the C(E) nor the C(AC); set priority flag when the next character can be accepted.

Block Transfer Devices

Table 5-1 lists the C field assignments which have been made for the RCA 4102 standard line of Block-Transfer peripheral devices and their control units. Further details appear in the discussions of the IØC commands to which these devices respond. In general, the C field specifies:

- Whether STI instruction samples status or reads a data word.
- Whether control unit is to continue or discontinue the transmission of data for the remainder of the data block.

TABLE 5-1. BLOCK-TRANSFER CONTROL CODES

C	INTERPRETATION	
VALUE	STI	LDØ
0	Sample status, discontinue	Transfer data word, discontinue
1	Sample status, continue	Transfer data word, discontinue
2	Read data word, discontinue	Transfer data word, discontinue
3	Read data word, continue	Transfer data word, discontinue
4	Sample status, discontinue	Transfer data word, continue
5	Sample status, continue	Transfer data word, continue
6	Read data word, discontinue	Transfer data word, continue
7	Read data word, continue	Transfer data word, continue

IØC CONTROL COMMANDS

At any given time, each Block-Transfer Control Unit is in one of three mutually exclusive states:

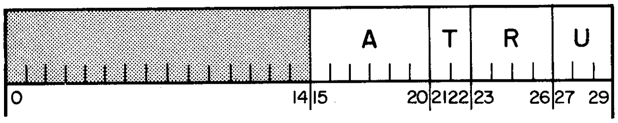
- Available — not reserved by any program.
- Claimed — reserved, but not currently busy.
- Busy — action in process.

The state of a control unit at any time is determined solely by the last IØC command it has accepted. The ability of a control unit to "accept" an IØC command and to issue a DSI signal is determined by the nature of the command, by the present state of the control unit, and, at times, by the condition of the selected peripheral device.

If the control unit "rejects" the command, no action is taken by the peripheral equipment, no DSI signal is generated, and the CPU proceeds as specified by the D and S fields of the IØC instruction. In this case, if the IØC instruction contains S = 1 (suicide), an auxiliary flag is set in the Call-Back Logic.

Command Word Format

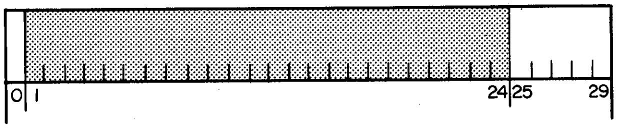
The command words transmitted by the IØC instruction are interpreted according to the following format:



The R field of this command specifies the priority level of the control unit being addressed. The A and T fields specify the action to be taken; the U field specifies either the device number or the data format.

Status Indicators

Upon completion of the action required by most IØC commands, the Block Transfer Control Unit automatically sets the associated Flag R. When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory a "Status Indicator" word with the following format:



Bit 0 contains a "one" only if a data error or timing error or inoperable condition has been detected.

Bits 1-24 always contain "zero's".

Bit 25 contains a "one" only if a data error has been detected.

Bit 26 contains a "one" only if a timing error has been detected.

Bit 27 contains a "one" only if the controlled device has sensed "Start of Medium" (e.g., magnetic tape load point).

Bit 28 contains a "one" only if the controlled device has sensed "End of Medium" (e.g., end of magnetic tape warning point).

Bit 29 contains a "one" only if an End-of-File signal has been sensed (e.g., empty Card Reader feed hopper).

Bits 25-29 are forced to "zero's" if the controlled device has been found inoperable.

Inoperable Condition

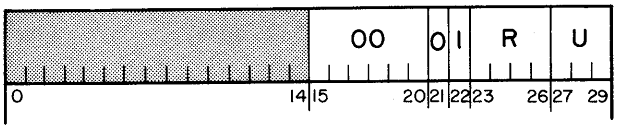
If a Block-Transfer Device is "inoperable" (off-line, fuse blown, tape broken, etcetera), the following special actions are taken:

- Acceptance or rejection of any IØC command is not affected by inoperability.
- If any specific operational command (e.g., read, write, advance) is accepted, the control unit will immediately set its associated Flag.
- When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The program is thus notified of the "inoperability" condition, without the use of special instructions.

GENERIC COMMANDS

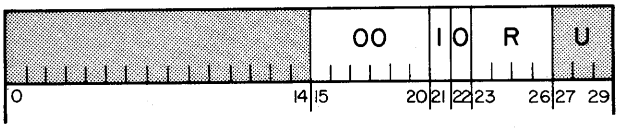
PEC – Peripheral-Equipment Claim



This command is accepted only if Control Unit R and device U are "available". (U may specify any of (up to) 8 devices associated with Control Unit R.)

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R enters the "claimed" state, and device U is selected and connected to it.

PER — Peripheral-Equipment Release



This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R and the previously selected device are left in the "available" state, a call-back is initiated, and the status indicators of Control Unit R are cleared.

MAGNETIC-TAPE COMMANDS

TWF — Tape Write Forward

	04	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport starts to write tape in the forward direction.

Each CPU word is expanded to 36 bits by suffixing six "zero's" to the least significant end, and is transmitted from the control unit to the tape transport as six sequential 6-bit characters, from most significant to least significant. A seventh bit is generated by the control unit and recorded on the tape as a parity-check bit for each character, as specified by the U field. The parity is made "even" if U is even, "odd" if U is odd. (If even parity is used, and an all-zero character is encountered, it is transmitted as a "blank", octal code 20.)

Priority Flag R is set whenever the control unit is ready to accept a CPU word. When the CPU responds to this demand, it must execute an LDØ instruction in order to cause the transfer of data. If an LDØ instruction with C < 4 (see Table 5-1) is executed in response to a demand, an end-of-record gap will be recorded and the action of the tape transport will cease. If no LDØ instruction is executed in this priority level within a prescribed interval* following the demand, an end-of-record gap will be recorded, the action of the tape transport will cease, and the "timing error" status indicator will be set.

When the tape motion has stopped, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the end-of-record gap just written.

TWE — Tape Write End-of-File

	14	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport advances, records an end-of-file record, and stops. The parity of this record is made "even" if U is even, "odd" if U is odd.

When the tape motion has stopped, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI instruction will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the gap following the end-of-file just written.

TRF — Tape Read Forward

	02	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport starts to read tape in the forward direction. Six sequential 6-bit characters are assembled in the control unit to form a 36-bit word. The parity of each character is checked as specified by the U field to assure that it is "even" if U is even, "odd" if U is odd. The sixth character is then discarded.

*This interval is a function of the recording frequency of the tape station.

Priority Flag R will subsequently be set whenever a computer word has been assembled in the control unit for transmission to the CPU. When the CPU responds to this demand, it must execute an STI instruction in order to cause the transfer of data. C codes for this STI instruction are presented in Table 5-1.

If no STI instruction is executed in this priority level within a prescribed interval* following the demand, the remainder of the record will not be read, and the "timing error" status indicator will be set.

When an end-of-record is sensed, the action of the tape transport will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the end-of-record gap following the record just read.

TAR — Tape Advance Record

▨	40	0	0	R	▨ U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport starts to scan tape in the forward direction.

When an end-of-record is sensed, the action of the tape transport will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the end-of-record gap following the record just scanned.

TBR — Tape Backspace Record

▨	41	0	0	R	▨ U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport starts to scan tape in the reverse direction.

When an end-of-record or load point is sensed, the action of the tape transport will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the end-of-record gap preceding the record just scanned.

TAF — Tape Advance File

▨	50	0	0	R	▨ U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed."

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport starts to scan tape in the forward direction.

When an end-of-file (or an end-of-record gap, following the end-of-tape warning point) is sensed, the action of the tape transport will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

*This interval is a function of the recording frequency of the tape station.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the gap following the end-of-file just sensed.

TBF — Tape Backspace File

0 – 14	15 – 20	21	22	23 – 26	27 – 29
	51	0	0	R	U

This command is accepted only if Control Unit R is "claimed."

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected tape transport starts to scan tape in the reverse direction.

When an end-of-file or load point is sensed, the action of the tape transport will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

The final position of the tape will be such that the read-write head is in the gap preceding the end-of-file just sensed.

TRW — Tape Rewind

0 – 14	15 – 20	21	22	23 – 26	27 – 29
	20	0	0	R	U

This command is accepted only if Control Unit R is "claimed."

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R enters the "busy" state, and the previously selected tape transport starts to rewind. When the rewind motion has started, Control Unit R will return to the "available" state and initiate call-back. The tape station continues to rewind until the load point is reached. The tape motion will then stop, and another call-back signal will be initiated.

LINE-PRINTER COMMANDS

LAP — Line Advance and Print

0 – 14	15 – 20	21	22	23 – 26	27 – 29
	03	0	0	R	U

This command is accepted only if Control Unit R is "claimed."

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected printer starts to advance the paper.

Priority Flag R is set immediately and will be set again whenever the control unit is ready to accept another word from the CPU. When the CPU responds to each of these demands, it must execute an LDØ instruction in order to cause the transfer of data. Each such instruction transmits to the control unit one CPU word, containing five 6-bit characters (see Appendix I) to be printed in consecutive columns. Each LDØ instruction with $C \geq 4$ (see Table 5-1) indicates the availability of additional output data for this line of print. When the number of characters transmitted equals the number of print positions available, the line is "full". An LDØ instruction with $C < 4$ indicates that the remainder of this line is to be blank, and the line is then considered "full". When the line is "full", Control Unit R does not set priority Flag R again until after the line is printed.

Meanwhile, the paper advances until the perforated control tape within the printer mechanism reaches the next perforation in the track (0-7) specified by U. When the paper-advance action has been completed and the line is "full", the line of characters will be printed, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators. (The "End-of-File" indicator is used to denote that the paper is positioned for printing the last line of a page; i.e., "Bottom of Form".)

LPA — Line Printer Advance

	01	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected printer starts to advance the paper. The paper advances until the perforated control tape within the printer mechanism reaches the next perforation in the track (0-7) specified by U.

When the paper-advance action has been completed, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators. (The "End-of-File" indicator is used to denote that the paper is positioned for printing the last line of a page; i.e., "Bottom of Form".)

LPR — Line Print

	02	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state. Priority Flag R is set immediately and will be set again whenever the control unit is ready to accept another word from the CPU. When the CPU responds to each of these demands, it must execute an LDØ instruction in order to cause the transfer of data. Each such instruction transmits to the control-unit one CPU word, containing five 6-bit characters (see Appendix I) to be printed in consecutive columns. Each LDØ instruction with C $\geq$ 4 (see Table 5-1) indicates the availability of additional output data for this line of print. When the number of characters transmitted equals the number of print positions available, the line is "full". An LDØ instruction with C $<$ 4 indicates that the remainder of this line is to be blank, and the line is then considered "full".

When the line is "full", the line of characters will be printed, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators. (The "End-of-File" indicator is used to denote that the paper is positioned for printing the last line of a page; i.e., "Bottom of Form".)

CARD-PROCESSING COMMANDS

CRA — Card Read Alphanumeric

	01	0	0	R	0
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected card reader starts to read a card. Alphanumeric card columns are read, converted to 6-bit characters (see Appendix I) and are automatically assembled by the control unit into computer words of five characters each. Each CRA command generates up to 16 full words (80 characters).

Priority Flag R will subsequently be set whenever a word is ready for transmission to the CPU.

When the CPU responds to this demand, it must execute an STI instruction in order to cause the transfer of data. C code interpretations for this STI are presented in Table 5-1.

When the end of the card has passed the reading station, the action of the card reader will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators. (The "Data Error" indicator is used to denote that at least one card column contained an invalid code; i.e., not one of the 63 characters listed in Appendix I. The "End-of-File" indicator is used to denote that the card feed hopper is empty.)

CRB — Card Read Binary

	01	0	0	R	I
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected card reader starts to read a card. Starting with card column 1, each group of three columns is assembled into a 30-bit computer word. (Row 12, column 1 is the most significant bit; and row 3, column 3 is the least significant bit of the first word, etc.) Rows 4 through 9 of each column divisible by three are deleted. If any of the deleted bits is punched, the "data error" status indicator will be set. Each CRB command generates up to 27 computer words, bits 24 through 29 of the 27th word being "zeros".

Priority Flag R will subsequently be set whenever the control unit has assembled a computer word for transmission to the CPU. When the CPU responds to this demand, it must execute an STI instruction in order to cause the transfer of data. C-field interpretations for this STI are presented in Table 5-1.

When the end of the card has passed the reading station, the action of the card reader will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators. (The "End-of-File" indicator is used to denote that the card feed hopper is empty.)

CPR — Card Punch by Row

	01	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected card punch starts to feed a card past the punch position.

Priority Flag R will subsequently be set whenever the control unit is ready to accept data from the CPU to be punched in a card row.

When the CPU responds to this demand, it must execute an LDØ instruction for each portion (columns 1-30, 31-60, 61-80) of the card row to be punched. Bit 0 of each CPU word is punched in column 1, 31, or 61. C codes for these instructions (see Table 5-1) are:

C VALUE	CARD COLUMNS	NOTES
0	None	No further data for this row, punch row
1	1-30	Last word for this row, punch row
2	31-60	Last word for this row, punch row
3	61-80	Last word for this row, punch row
5	1-30	More data to follow
6	31-60	More data to follow
C = 6 may not precede C = 1 or C = 5		

If no LDØ with C < 4 is executed before a row must be punched, the "timing error" status indicator is set.

When the last card row has been punched, the action of the card punch will cease, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

DRUM COMMANDS

DRW — Drum Write

TRACK/SECTOR	04	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected drum starts to scan for the track and sector specified by bits 0 through 14 of the DRW word.

Each CPU word is transmitted from the control unit to the drum as thirty sequential bits, from most significant to least significant. A 31st bit is generated by the control unit and recorded on the drum as a parity-check bit for each word.

Priority Flag R is set whenever the control unit is ready to accept a CPU word. When the CPU responds to this demand, it must execute an LDØ instruction in order to cause the transfer of data. If an LDØ instruction with C < 4 (see Table 5-1) is executed in response to a demand, the remainder of the drum sector will be cleared. If no LDØ instruction is executed in this priority level within a prescribed interval* following the demand, the remainder of the drum sector will be cleared and the "timing error" status indicator will be set.

When the last word of the data block has been recorded, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

DRR—Drum Read

TRACK/SECTOR	02	0	0	R	U
0 14	15 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed".

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state, and the previously selected drum starts to scan for the track and sector specified by bits 0 through 14 of the DRR word. Thirty sequential bits are assembled in the control unit to form a computer word. The parity of each word is checked, and the "data error" status indicator is set if incorrect parity is detected.

Priority Flag R will subsequently be set whenever a computer word has been assembled in the control unit for transmission to the CPU. When the CPU responds to this demand, it must execute an STI instruction in order to cause the transfer of data. C codes for this STI instruction are presented in Table 5-1.

If no STI instruction is executed in this priority level within a prescribed interval* following the demand, the remainder of the record will not be read, and the "timing error" status indicator will be set.

When an end-of-record is sensed, Control Unit R will automatically enter the "claimed" state and initiate call-back, and priority Flag R will be set.

When the CPU responds to this demand, any STI or LDØ instruction will generate a DSI signal, and any STI will read into memory the control-unit status indicators.

INTERCOM COMMAND

ICT— INTERCOM Transmit

	ADDRESSEES		0	0	R	U
0	1 8	9 20	21	22	23 26	27 29

This command is accepted only if Control Unit R is "claimed" and no higher precedence subsystem has "simultaneously" claimed its INTERCOM Control Unit.

*This interval is a function of the recording frequency of the drum.

If the command is accepted, a DSI signal is generated, and the RCA 4102 proceeds to the next instruction in sequence. Control Unit R is left in the "busy" state.

Priority Flag R must subsequently be set once under program control, and will be set automatically whenever the control unit is ready to accept another word from the CPU. When the CPU responds to each of these demands, it must execute an LDØ instruction in order to cause the transfer of data. Each such instruction transmits to the control-unit one CPU word, to be transmitted continuously on the INTERCOM trunk until all addressed subsystems (specified by the presence of "one's" in bits 1-8 of the ICT word) have acknowledged receipt of the data.

Each LDØ instruction with $C \geq 4$ (see Table 5-1) indicates the availability of additional output data. An LDØ instruction with $C < 4$ indicates that the end-of-message is being transmitted, and a DSI signal is generated.

When the end-of-message transmission has been acknowledged, Control Unit R will automatically enter the "claimed" state and initiate call-back. (If U is odd, the addressee(s) will subsequently reply, via INTERCOM, with the message requested by the CPU; e.g., bulk storage subsystem.)

APPENDIX A NUMBER SYSTEMS

A number system is a set of symbols and rules used to represent and manipulate values. The basic rules remain the same for the different number systems, but, since familiar symbols are used in an unfamiliar manner, some explanation is required.

There are three number systems that are of interest to a computer programmer.

1. The DECIMAL (radix 10) number system, using the symbols (digits) 0 through 9, is of course required since problems and their results will usually be stated using this number system.

2. The BINARY (radix 2) number system, using the symbols (bits) 0 and 1, is used internally by the computer and should be understood by the programmer. Computer components are inherently binary in nature - that is, transistors are conducting or not, materials are magnetized in one direction or the other, etc. Therefore, the binary number system is the most efficient for machine operation.

3. The OCTAL (radix 8) number system, using the symbols (digits) 0 through 7, is used as a shorthand method of recording long binary numbers. For example, a binary number containing 30 bits becomes less unwieldy when written as a 10-digit octal number.

Note: In examples where the number system being used could be misunderstood, the radix, or number system base (10, 2, or 8) is shown as a subscript.

These number systems are positional in nature. That is, each symbol is implicitly assigned a weighting factor, determined by the radix and by the position of the symbol relative to the radical point (e.g., decimal point, binary point, or octal point). For example, the decimal representation, 652.54 is interpreted as follows:

$$652.54_{10} = 6 \times 10^2 + 5 \times 10^1 + 2 \times 10^0 + 5 \times 10^{-1} + 4 \times 10^{-2}$$
$$= 600 + 50 + 2 + .5 + .04$$

The general formula for positional representation is given by the power series:

$$N = \cdots + (A_k r^k) + \cdots + (A_2 r^2) + (A_1 r^1) + (A_0 r^0)$$
$$+ (A_{-1} r^{-1}) + (A_{-2} r^{-2}) + \cdots$$

Where

N is the value of the number

r is the radix (i.e., the total number of symbols permissible in the number system)

k is the position of the symbol (k = 0 for the digit immediately to the left of the radical point, with positive values of k for positions further to the left and negative values of k for positions to the right)

A_k may be any permissible symbol (0, 1, . . . , r-1)

Note: Mathematically $r^0 = 1$ regardless of the value of r.

In practice, it is usually necessary to limit the power series to a finite set of terms. If only negative values of k are retained, only fractions can be directly represented. If only non-negative values of k are retained, only integers can be directly represented.

NUMBER CONVERSIONS

The following are some rules for converting values from one number system to another.

Octal-to-Binary

Use the accompanying table to convert each octal digit to three bits (binary digits).

Octal Digit	0	1	2	3	4	5	6	7
Binary Equivalent	000	001	010	011	100	101	110	111

Binary-to-Octal

Group the bits by three in each direction from the binary point, and use the preceding table to convert each group to its equivalent octal digit. (Supply 0's as needed to complete groups of three at either end.)

Examples:

$$\begin{array}{ccccccc} \underbrace{10} & \underbrace{001} & \underbrace{111} & . & \underbrace{000} & \underbrace{011} & \underbrace{1_2} \\ \underbrace{2} & \underbrace{1} & \underbrace{7} & . & \underbrace{0} & \underbrace{3} & \underbrace{4_8} \\ 010 & 001 & 111 & . & 000 & 011 & 100_2 \end{array}$$

Decimal-to-Octal

Integers are converted by dividing the number (and each succeeding quotient) by eight. The remainders from these divisions are the desired octal digits, in reverse order. For example, to convert 143_{10} to octal:

$$\begin{array}{rl} & 0 \ \text{r} \ 2 \\ 8 & \overline{)2} \ \text{r} \ 1 \\ 8 & \overline{)17} \ \text{r} \ 7 \\ 8 & \overline{)143} \end{array}$$

Therefore, $143_{10} = 217_8$

Fractions are converted by multiplying the number repeatedly by eight. The integer portions of the products are the desired octal digits, and do not enter subsequent multiplications. For example, to convert $.0547_{10}$ to octal:

$$\begin{array}{rr} & .0547 \\ & \underline{\text{x}8} \\ 0 & .4376 \\ & \underline{\text{x}8} \\ 3 & .5008 \\ & \underline{\text{x}8} \\ 4 & .0064 \end{array}$$

Therefore, $.0547_{10} = .034^+_8$

Mixed Numbers are converted by treating the integer and fraction portions separately, according to the above rules:

$$143.0547_{10} = 217.034^+_8$$

Octal-to-Decimal

Integers are converted by alternate multiplication by eight and addition of successive digits, from the left. For example, to convert 217_8 to decimal:

$$\begin{array}{l} 2 \\ \underline{\text{x}8} \\ 16 \ + 1 = 17 \\ \qquad\qquad \underline{\text{x}8} \\ \qquad\qquad 136 \ + 7 = 143 \end{array}$$

Therefore, $217_8 = 143_{10}$

Fractions are converted by alternate division by eight, and addition of successive digits, from the right. For example, to convert $.034_8$ to decimal:

$$\begin{array}{ll} 8 & \underline{|4\quad} \\ & \quad .5 \\ & +3 \\ 8 & \underline{|3.5\quad} \\ & \quad .4375 \\ & +0 \\ 8 & \underline{|0.4375} \\ & \quad .0547^- \end{array}$$

Therefore, $.034_8 = .0547^-_{10}$

Mixed Numbers are converted by treating the integer and fraction portions separately, according to the above rules:

$$217.034_8 = 143.0547^-_{10}$$

Decimal-to-Binary and Binary-to-Decimal

Conversions between decimal and binary representations are most easily performed in two steps, converting to octal notation in the first step; i.e., decimal-to-octal-to-binary or binary-to-octal-to-decimal:

$$10\ 001\ 111.000\ 011\ 1_2 = 217.034_8 = 143.0547^-_{10}$$

ALGEBRAIC VALUES

The preceding paragraphs have presented the principles used for representing a non-negative real number in any radix system. In practice, the power-series representation (on page 37) is modified. One term (called the "sign bit") is prefixed to the truncated series, allowing the representation of negative, as well as positive, values. Regardless of the radix used, only two symbols are permissible in the sign-bit term: 0 or -1. (This constraint is not universally applicable, but is convenient for the current purpose.)

In dealing with signed numbers, the symbol (-1) should be written with the "minus" sign above the "one" to avoid confusion:

$\bar{1}290_{10}$ means $-1000 + 200 + 90 = -710_{10}$

$\bar{1}.367_{10}$ means $-1 + .3 + .06 + .007 = -.633_{10}$

$\bar{1}.001_2$ means $-1 + 2^{-3} = -7/8$

When the magnitude (absolute value) of a negative number is desired, or vice versa, the most convenient technique is "complementing". Complementing is performed by (1) subtracting each digit of the given representation from the largest symbol (r-1) permitted in the radix system, and then (2) adding unity to the rightmost digit of the resulting number, propagating carries as necessary. Thus, in the preceding examples:

$$\begin{array}{rl}
\_999 & \\
\bar{1}290_{10} & \text{given negative number} \\
\hline
709 & \text{subtract each digit from 9} \\
+1 & \\
\hline
710 &
\end{array}$$

$$\begin{array}{rl}
\_\ 999 & \\
\bar{1}.367_{10} & \text{given negative number} \\
\hline
.632 & \text{subtract each digit from 9} \\
+1 & \\
\hline
.633 &
\end{array}$$

$$\begin{array}{rl}
\_\ 111 & \\
\bar{1}.001_{2} & \text{given negative number} \\
\hline
.110 & \text{subtract each digit from 1} \\
+1 & \\
\hline
.111 & \text{absolute value} = 7/8
\end{array}$$

It must be noted that negative binary numbers are often written without an explicit "minus" sign over the sign bit, which then becomes indistinguishable from any binary digit. For arithmetic purposes, the presence of this implicit sign is vital.

OCTAL REPRESENTATION VERSUS OCTAL VALUE

When using octal notation for the representation of a 30-bit signed binary number, it must be recognized that this octal representation differs from the octal value. Octal representation is generated by grouping bits from either end of the computer word, rather than from the binary point.

Thus, the sign bit becomes part of the first digit in octal representation, whereas it remains a separate symbol when writing the octal value.

This distinction must be remembered when reading the Control Panel displays and when entering data words in a program.

Examples:

$$\begin{aligned}
+ .75_{10} &= 0.6_8 \\
&= 0.11000000000000000000000000000_2 \\
&\quad \text{Octal representation } 3000000000 \\
- .5_{10} &= \bar{1}.4_8 \\
&= \bar{1}.10000000000000000000000000000_2 \\
&\quad \text{Octal representation } 6000000000
\end{aligned}$$

ARITHMETIC

Because numbers are represented in computer words of finite length, it is necessary to define the result of arithmetic processes which tend to exceed this length. The general rule applicable to such situations is defined in the language of residue arithmetic.

In residue arithmetic, all numbers are made to correspond to a set of m numbers by repeated addition or subtraction of the "modulus", m (expressed or implied). As a familiar example, any angle has a "residue equivalent" in the range from zero through 359° by use of the modulus, m = 360°:

$$\begin{aligned}
370^\circ - 1\,(360^\circ) &= 10^\circ \\
1962^\circ - 5\,(360^\circ) &= 162^\circ \\
-735^\circ + 3\,(360^\circ) &= 345^\circ
\end{aligned}$$

Numbers which differ by an exact multiple of the modulus are thus considered equivalent in residue arithmetic (i.e., "congruent, modulo m").

For example, this principle is applied in the computer representation of index values which range from 0 through 16383, modulo 16384. Thus, -5 + 1 (16384) = 16379, the computer representation of the index value, -5. Similarly, the addition of 16382 to an index value (e.g., 16379) is equivalent to subtracting two from the index value:

$$\begin{aligned}
16379 + 16382 &= 32761 \\
&= 16377 + 1(16384)
\end{aligned}$$

or

$$16379 - 2 = 16377$$

In analogous fashion, data words in the RCA 4102 CPU may have 2^{30} discrete values: $-1, -1 + 2^{-29}, \ldots, -2^{-29}, 0, + 2^{-29}, \ldots, + 1-2^{-29}$.

Arithmetic results are interpreted as residues with respect to the modulus, $m = 2.0 = 2^{30} (2^{-29})$.

Where

2^{30} is the size of the number set

2^{-29} is the quantum size

The arithmetic rules for processing numbers are analagous in any radix system. The addition and multiplication tables for the three systems discussed are illustrated in Table A-1.

No special treatment is afforded the "sign bit" position of an algebraic representation. Carries from lower order terms propagate into this position in normal fashion. The addition table for this position is:

+	$\bar{1}$	0
$\bar{1}$	$\overline{2}$*	$\bar{1}$
0	$\bar{1}$	0

+	$\bar{1}$	0
$\bar{1}$	$\bar{1}$	0
0	0	1*

(with carry from adjacent column)

*The starred entries are invalid symbols for this term, and represent an "overflow" situation.

TABLE A-1. ADDITION AND MULTIPLICATION TABLES

Decimal Addition

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Decimal Multiplication

x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

Binary Addition

+	0	1
0	0	1
1	1	10

Binary Multiplication

x	0	1
0	0	0
1	0	1

Octal Addition

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Octal Multiplication

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Addition Examples:

	Conventional	Algebraic Representations		
	Decimal	Decimal	Octal	Binary
1)	+13	013	015	0 001 101
	+28	028	034	0 011 100
	+41	041	051	0 101 001
2)	-13	$\bar{1}87$	$\bar{1}63$	$\bar{1}$ 110 011
	+28	028	034	0 011 100
	+15	015	017	0 001 111
3)	+13	013	015	0 001 101
	-28	$\bar{1}72$	$\bar{1}44$	$\bar{1}$ 100 100
	-15	$\bar{1}85$	$\bar{1}61$	$\bar{1}$ 110 001
4)	-13	$\bar{1}87$	$\bar{1}63$	$\bar{1}$ 110 011
	-28	$\bar{1}72$	$\bar{1}44$	$\bar{1}$ 100 100
	-41	$\bar{1}59$	$\bar{1}27$	$\bar{1}$ 010 111
5)	+46	046	056	0 101 110
	+57	057	071	0 111 001
	103	103	147	1 100 111
		↑	↑	↑
		Invalid symbol -- overflow condition		
6)	-46	$\bar{1}54$	$\bar{1}22$	$\bar{1}$ 010 010
	-57	$\bar{1}43$	$\bar{1}07$	$\bar{1}$ 000 111
	-103	$\bar{2}97$	$\bar{2}31$	$\bar{2}$ 011 001
		↑	↑	↑
		Invalid symbol -- overflow condition		

In the last two examples (overflow conditions), the rules of residue arithmetic cause ambiguity to arise in the sign bit. Thus, in Example 5, the sign bit of "1" would be indistinguishable from "$\bar{1}$". In the last example, the sign bit of "$\bar{2}$" would similarly appear as "0".

APPENDIX B POWERS-OF-TWO TABLE

2^{η}	η	$2^{-\eta}$
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

APPENDIX C INTEGER CONVERSION TABLE

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

	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
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

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

	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
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

APPENDIX C INTEGER CONVERSION TABLE (Cont)

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

	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
2110	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
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	1509	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 to 3777 1536 to 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
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

INTEGER CONVERSION TABLE (Cont)

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

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	0253	2054	2055
401C	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
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

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

	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
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

APPENDIX C INTEGER CONVERSION TABLE (Cont)

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

	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
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 to 7777 (Octal) 3584 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	3596	3597	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
7261	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
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

APPENDIX D 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

APPENDIX D 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

APPENDIX D 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

APPENDIX E PRIMARY INSTRUCTIONS

MNE-MONIC	OCTAL CODE	INSTRUCTION NAME	PRIMARY FUNCTION	CYC + µs	B VALUE	REFERENCE APPENDIX F	PAGE
WORD TRANSMISSION INSTRUCTIONS							
STA	41	Store AC	C(AC)→C(E)	3*	0-7	A	15
LDA	61	Load AC	C(E)→C(AC)	3*	0-7	A	15
XAS	57	Exchange AC and Storage	C(E)⇄C(AC)	3 + 1*	0-7	A	15
STZ	45	Store Zero	0→C(E)	3*	0-7	A	16
LDZ	73	Load Zero	0→C(AC)	2	None	A	16
LDX	67	Load Index	C(L)→C(B)	4	0-6	A	16
STX	47	Store Index	C(B)→C(L)	4	1-6	A	16
PAX	56	Place AC in Index	C(AC)→C(B)	3	1-6	A	16
PXA	54	Place Index in AC	C(B)→C(AC) (See PXA Text.)	3	1-6	A	16
LOGIC INSTRUCTIONS							
ANA	34	AND to AC	C(AC)∩C(E)→C(AC)	3*	0-7	A	17
ANR	35	AND and Replace	C(AC)∩C(E)→C(AC)→C(E)	3*	0-7	A	17
ERA	36	Exclusive ØR to AC	C(AC)⊕C(E)→C(AC)	3*	0-7	A	17
ERR	37	Exclusive ØR and Replace	C(AC)⊕C(E)→C(AC)→C(E)	3*	0-7	A	17
ØRA	32	ØR to AC	C(AC)∪C(E)→C(AC)	3*	0-7	A	18
ØRR	33	ØR and Replace	C(AC)∪C(E)→C(AC)→C(E)	3*	0-7	A	18
ØRS	53	ØR to Storage	C(AC)∪C(E)→C(E)	3*	0-7	A	18
SHIFT INSTRUCTIONS							
SAL	01	Shift AC Logical	Shift C(AC), E places right	2+2*	0-6	A	18
SAC	03	Shift AC Circular	Rotate C(AC), E places right	to	0-6	A	19
SAA	05	Shift AC Algebraic	C(AC) x 2^{-E}→C(AC)	2+32*	0-6	A	19
ARITHMETIC INSTRUCTIONS							
ADD	10	Add	C(AC) + C(E)→C(AC)	3*	0-7	A	20
ADR	11	Add and Replace	C(AC) + C(E)→C(AC)→C(E)	3*	0-7	A	20
ADM	12	Add Magnitude	|C(AC)| + C(E)→C(AC)	3*	0-7	A	20
AMR	13	Add Magnitude and Replace	|C(AC)| + C(E)→C(AC)→C(E)	3*	0-7	A	20
SUB	14	Subtract	-C(AC) + C(E)→C(AC)	3*	0-7	A	20
SBR	15	Subtract and Replace	-C(AC) + C(E)→C(AC)→C(E)	3*	0-7	A	21
SBM	16	Subtract Magnitude	-|C(AC)| + C(E)→C(AC)	3*	0-7	A	21
SMR	17	Subtract Magnitude and Replace	-|C(AC)| + C(E)→C(AC)→C(E)	3*	0-7	A	21
MPY	22	Multiply	C(AC) xC(E)→C(AC)	3+59*	0-7	A	21
MPH	23	Multiply Half-Word	$C(AC)_{0-14}$xC(E)→C(AC)	3+29*	0-7	A	21
DIV	24	Divide	C(AC)/C(E)→C(AC)	3+57*	0-7	A	22
DVH	25	Divide Half-Word	C(AC)/C(E)→$C(AC)_{0-14}$	3+29*	0-7	A	22
BRANCH INSTRUCTIONS							
JAC	72	Jump on AC	E→C(IL) } If C condition is satisfied.	2*	0-7	C	22
JIX	72	Jump on Index	L→C(IL)	3	1-6	D	23
JSX	72	Jump and Set Index	L→C(IL)	3	1-6	E	22
TST	77	Test Storage	C(IL) + D→C(IL)	3*	0-7	F	23
SKP	76	Skip	C(IL) -D -1→C(IL)	3*	0-7	F	23
CONTROL INSTRUCTIONS							
FLS	71	Flag Set	C(F)∪L→C(F)	2	None	A	24
ALX	65	Add L to Index	C(B) + L→C(B)	3	1-6	B	24
HLT	00	Halt	Stop CPU or Suicide	3*	0-7	A	25
NØP	56	No Operation	None	3	0	A	24
INPUT-OUTPUT INSTRUCTIONS							
STI	43	Store Input Word	Input→C(E)	3*	0-7	G	25
LDØ	63	Load Output Word	C(E)→Output Device	3+8*	0-6	G	26
			C(AC)→Output Device		7		
IØC	62	Input Output Control	C(E)→Control Unit	3+8*	0-6	G	26
			C(AC)→Control Unit		7		

*Indexable; add 1 cycle if B = 1-6.
Also, add 1 cycle for each priority interruption.

APPENDIX F SECONDARY INSTRUCTIONS

SYM-BOL	OC-TAL C	D VALUE	BRANCH CONDITION	BRANCH DESTINATION	SUBSIDIARY ACTIONS	
					CONDITIONAL	UNCONDITIONAL
Table A. NORMAL						
——	0	even	None	None	None	None
IØV	0	odd	None	None	None	Do not set V
JXC	1*	0-7	Initial C(B) ≠ 0	C(IL) + D	None	C(B) -1→C(B)
DIX	2*	0-7	None	None	None	C(B) -1→C(B)
APZ	3	0-7	Final C(AC) ≥ 0	C(IL) + D	None	None
AGZ	4	0-7	Final C(AC) > 0	C(IL) + D	None	None
ANZ	5	0-7	Final C(AC) ≠ 0	C(IL) + D	None	None
AEZ	6	0-7	Final C(AC) = 0	C(IL) + D	None	None
ALZ	7	0-7	Final C(AC) < 0	C(IL) + D	None	None
*C≠1, 2 with OP codes LDZ, FLS, LDX, STX, PAX, and PXA.						
Table B. ALX						
——	0	——	None	None	None	C(B) + L→C(B)
JXC	1	0-7	C(B) + L > 37777_8	C(IL) + D	None	C(B) + L→C(B)
DIX	2	——	None	None	None	C(B) + L→C(B)
APZ	3	0-7	C(AC) ≥ 0	C(IL) + D	None	C(B) + L→C(B)
AGZ	4	0-7	C(AC) > 0	C(IL) + D	None	C(B) + L→C(B)
ANZ	5	0-7	C(AC) ≠ 0	C(IL) + D	None	C(B) + L→C(B)
AEZ	6	0-7	C(AC) = 0	C(IL) + D	None	C(B) + L→C(B)
ALZ	7	0-7	C(AC) < 0	C(IL) + D	None	C(B) + L→C(B)
Table C. JAC						
——	0	0	Unconditional	E	None	None
JØV	0	1	Initial V = 1	E	None	Clear V, C(IL)$_0$
APZ	3	0	C(AC) ≥ 0	E	None	None
AGZ	4	0	C(AC) > 0	E	None	None
ANZ	5	0	C(AC) ≠ 0	E	None	None
AEZ	6	0	C(AC) = 0	E	None	None
ALZ	7	0	C(AC) < 0	E	None	None
Table D. JIX						
JXC	1	0	Initial C(B) ≠ 0	L	None	C(B) -1→C(B)
JXD	2	0	Unconditional	L	None	C(B) -1→C(B)
Table E. JSX						
——	0	2	Unconditional	L	Initial C(IL)→C(B)	None
JØV	0	3	Initial V = 1	L	Initial C(IL)→C(B)	Clear V, C(IL)$_0$
APZ	3	2	C(AC) ≥ 0	L	Initial C(IL)→C(B)	None
AGZ	4	2	C(AC) > 0	L	Initial C(IL)→C(B)	None
ANZ	5	2	C(AC) ≠ 0	L	Initial C(IL)→C(B)	None
AEZ	6	2	C(AC) = 0	L	Initial C(IL)→C(B)	None
ALZ	7	2	C(AC) < 0	L	Initial C(IL)→C(B)	None
Table F. TST/SKP						
EGZ	0	0-7	C(E) > 0	TST: C(IL) + D SKP: C(IL) -1 -D	None	None
ENZ	1	0-7	C(E) ≠ 0		None	None
EEZ	2	0-7	C(E) = 0		None	None
ELZ	3	0-7	C(E) < 0		None	None
EGA	4	0-7	C(E) > C(AC)		None	None
ENA	5	0-7	C(E) ≠ C(AC)		None	None
EEA	6	0-7	C(E) = C(AC)		None	None
ELA	7	0-7	C(E) < C(AC)		None	None
Table G. STI, LDØ						
0-7 (Subsidiary peripheral address)		0-7	C(B) ≠ 0 and no DSI	C(IL) + D	None	C(B) -1→C(B)

APPENDIX G IØC COMMAND WORDS

MNEMONIC CODE	COMMAND NAME	A FIELD (OCTAL)	T FIELD (BINARY)	U FIELD INTERPRETATION	PAGE NO.
			GENERIC COMMANDS		
PEC	Peripheral-Equipment Claim	00	01	Device Number	29
PER	Peripheral-Equipment Release	00	10	Not Interpreted	29
			MAGNETIC-TAPE COMMANDS		
TWF	Tape Write Forward	04	00	Parity Select	30
TWE	Tape Write End-of-File	14	00	Parity Select	30
TRF	Tape Read Forward	02	00	Parity Check	30
TAR	Tape Advance Record	40	00	Not Interpreted	31
TAF	Tape Advance File	50	00	Not Interpreted	31
TBR	Tape Backspace Record	41	00	Not Interpreted	31
TBF	Tape Backspace File	51	00	Not Interpreted	32
TRW	Tape Rewind	20	00	Not Interpreted	32
			LINE-PRINTER COMMANDS		
LPA	Line Printer Advance	01	00	Paper Advance Control	33
LPR	Line Print	02	00	Not Interpreted	33
LAP	Line Advance and Print	03	00	Paper Advance Control	32
			CARD-PROCESSING COMMANDS		
CPR	Card Punch	01	00	Not Interpreted	34
CRA	Card Read Alphanumeric	01	00	Must be Even	33
CRB	Card Read Binary	01	00	Must be Odd	34
			DRUM COMMANDS		
DRW	Drum Write	04	00	Not Interpreted	35
DRR	Drum Read	02	00	Not Interpreted	35
			INTERCOM COMMAND		
ICT	INTERCOM Transmit	Not Interpreted	00	Reply Request	35

APPENDIX H
BLOCK TRANSFER STATUS INDICATOR WORD FORMAT

Status Word (Octal Value)	IØC Commands Affected ⟶	T W F	T W E	T R F	T A R	T B R	T A F	T B F	L P A	L P R	L A P	C P R	C R A	C R B	D R W	D R R
	Interpretation															
0000000005	End of File at Load Point					X		X								
0000000004	Load Point					X		X								
0000000003	End of File at End of Medium		X	X	X		X		X	X	X					
0000000002	End of Medium	X		X	X		X		X	X	X					
0000000001	End of File		X	X	X	X	X	X	X	X	X		X	X		
0000000000	Normal Operation	X		X	X	X			X	X	X	X	X	X	X	X
4000000030-37	Data Error & Timing Error	X		X									X	X		X
4000000020-27	Data Error	X	X	X									X	X		X
4000000010-17	Timing Error	X		X								X	X	X	X	X
4000000000	Inoperable	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

APPENDIX I ALPHANUMERIC CODES

CARD CODE	BINARY CODE	LINE PRINTER	FLEXO-WRITER	CARD CODE	BINARY CODE	LINE PRINTER	FLEXO-WRITER
Invalid	000 000	Blank		11	100 000	-	-
1	001	1	1	11-1	001	J	J
2	010	2	2	11-2	010	K	K
3	011	3	3	11-3	011	L	L
4	100	4	4	11-4	100	M	M
5	101	5	5	11-5	101	N	N
6	110	6	6	11-6	110	Ø	Ø
7	111	7	7	11-7	111	P	P
8	001 000	8	8	11-8	101 000	Q	Q
9	001	9	9	11-9	001	R	R
0	010	0	0	11-0	010	±	$\boxed{K}$
8-3	011	=	=	11-8-3	011	$	$\boxed{L}$
8-4	100	~	$\boxed{4}$	11-8-4	100	*	*
8-5	101	"	$\boxed{5}$	11-8-5	101	'	$\boxed{N}$
8-6	110	<	$\boxed{6}$	11-8-6	110	:	$\boxed{Ø}$
8-7	111	≤	$\boxed{7}$	11-8-7	111	;	$\boxed{P}$
Blank	010 000	Blank	Space	12	110 000	+	+
0-1	001	/	/	12-1	001	A	A
0-2	010	S	S	12-2	010	B	B
0-3	011	T	T	12-3	011	C	C
0-4	100	U	U	12-4	100	D	D
0-5	101	V	V	12-5	101	E	E
0-6	110	W	W	12-6	110	F	F
0-7	111	X	X	12-7	111	G	G
0-8	011 000	Y	Y	12-8	111 000	H	H
0-9	001	Z	Z	12-9	001	I	I
0-8-2	010	↓	Car Ret	12-0	010	→	Tab
0-8-3	011	,	$\boxed{T}$	12-8-3	011	.	
0-8-4	100	(	$\boxed{U}$	12-8-4	100	)	$\boxed{D}$
0-8-5	101	[	$\boxed{V}$	12-8-5	101	]	$\boxed{E}$
0-8-6	110	≥	$\boxed{W}$	12-8-6	110	≠	$\boxed{F}$
0-8-7	111	>	$\boxed{X}$	12-8-7	111	↑	?

Flexowriter characters enclosed by box are printed in red.

APPENDIX J EXECUTIVE SPACE ASSIGNMENTS

PRIORITY	IL	INDEX REGISTERS						WORK* SPACE
		6	5	4	3	2	1	
(Low) P	37777	37776	37775	37774	37773	37772	37771	37770
Ø	37767	37766	37765	37764	37763	37762	37761	37760
N	37757	37756	37755	37754	37753	37752	37751	37750
M	37747	37746	37745	37744	37743	37742	37741	37740
L	37737	37736	37735	37734	37733	37732	37731	37730
K	37727	37726	37725	37724	37723	37722	37721	37720
J	37717	37716	37715	37714	37713	37712	37711	37710
I	37707	37706	37705	37704	37703	37702	37701	37700
H	37677	37676	37675	37674	37673	37672	37671	37670
G	37667	37666	37665	37664	37663	37662	37661	37660
F	37657	37656	37655	37654	37653	37652	37651	37650
E	37647	37646	37645	37644	37643	37642	37641	37640
D	37637	37636	37635	37634	37633	37632	37631	37630
C	37627	37626	37625	37624	37623	37622	37621	37620
B	37617	37616	37615	37614	37613	37612	37611	37610
(High) A	37607	37606	37605	37604	37603	37602	37601	37600

*May be addressed by the LDX instruction as well as by absolute addressing.

GENERAL INDEX

OPERATION CODE INDEX

ALPHABETIC

NUMERIC

THE MOST TRUSTED NAME IN ELECTRONICS

DATA SYSTEMS DIVISION DEFENSE ELECTRONIC PRODUCTS VAN NUYS, CALIFORNIA

Printed in U.S.A.