N-16044/000/01

This document was produced by SDC in performance of DOD-SD-97

AUTHOR [illegible]
J. A. Slaybaugh

DATE 17 May 1962

PAGE 1 OF 57 PAGES

NOTE *an internal working paper*

System Development Corporation/2500 Colorado Ave./Santa Monica, California

Author Delivered

THE AN/FSQ-32

A Description and Coding Manual for Experienced Programmers

The intent of this document is to provide senior programmers with a concise description of the Q-32. In general, instructions have been described in terms of SCAMP mnemonics and instruction effects rather than a detailed hardware analysis.

TABLE OF CONTENTS

Page

General Description

1. This synchronous ones' complement machine has a very powerful and flexible instruction list with which to operate on data contained in its 65K, 48 bit word memory. The memory unit consists of 4 - 16K banks, each bank being independently and simultaneously accessible during each 2.4 usec memory cycle and a test memory unit of 4 switch registers, 2 live registers, and 32 pluggable registers. This permits truly simultaneous I/O on its "1401" channel, typewriter channel, 1 to 4 tape channels (depending on specific configuration, but normally two), and high-speed channel (drums, I/O Register, or, on duplexed machines, inter-computer memory to memory, and two way drum-memory transfers). Further, instruction overlapping is possible by accessing the next instruction simultaneously with the last data cycle of an instruction. There is also a real-time clock, accurate to 1/2048 sec.

2. Instruction and Data Words

The instruction word, generally speaking, uses the left-half (24 bits) for the instruction and the various instruction modifiers, and the right-half (24 bits) for the address and addressing modifiers. The data word can be arithmetically operated upon in floating point or fixed point and in the latter case, as a function of the instruction modifiers, as a 48 bit signed number, two 24 bit signed numbers (dual), one 24 bit signed number (either half word), or a "twinned" dual number. Further, in fixed point, simultaneous with one of the above, the word may be treated as eight 6-bit "BYTEs" which are masked and cycled. In logical operations, BYTEs may also be manipulated, or a mask may be used on a word, or on a specific BYTE.

3. Addressing

Three different modes of addressing exist for most instructions, (1) direct, (2) indirect, and (3) immediate. Direct and indirect addressing include address modification via arithmetic addition of the contents of none, one or two index registers to the "address," using the sum for the effective address. There are eight index registers (designated 1-8) plus the right accumulator (designated 15 or the letter "A") and the program counter (designated 14 or the letter "C"). Direct indexing is specified by setting the "IX" modifier of the address half-word with the number of the index to be used (see Appendices A and E for coding formats). (NOTE: The right accumulator and program counter may NOT be specified as the index register for functions which modify the index.) Double indexing, the use of a second index in address modification, is accomplished by setting the "DI" modifier in the address portion of the word. (NOTE: The right accumulator may NOT be used as a Double Index.) The machine uses the contents of a special register to select the index register to be used for the second, or double, indexing sub-cycle. This register is called the Double Index Selection Register, or DISR. "Double indexing" may occur independently of direct indexing and indexing does not normally affect cycle time.

Indirect addressing is accomplished by setting the "I" modifier in the address half-word. Both forms of indexing apply in determining the effective address of the indirect address. Further, the entire addressing cycle is repeated for the indirect address, i.e., indexing and indirect addressing apply to the indirect address, thus permitting an infinite number of levels of indirect addressing with direct and double indexing at each level.

Immediate addressing permits using the right-half of the instruction as the operand and that operand is referred to as "real data." It is indicated by setting the "R" modifier in the instruction half-word, but is coded in the address field. When used in this manner, a full-word is assembled by duplicating the right-half into both halves of a machine register, then treating it as if the word had been accessed from core. That is, it is subjected to the manipulation described in section 4.2. Modification of real data via indexing applies ONLY for cycle, shift, LDX, and ATX instructions.

An additional feature is that all significant machine registers are addressable, most for both read and write (see Appendix A). However, Internal Addressable Registers (addresses 7776XX), are NOT addressable for instructions nor indirect addresses.

4. Arithmetic Operations

The Arithmetic Element includes three significant registers, (1) the Accumulator for holding results of operations and one operand, (2) the B Register for holding masks and low order bits of results of operations, and (3) the Logical Register[*] used in various logical operations.

4.1 Floating Point

A complete set of floating point instructions are available. The configuration, from left to right, is sign, 11 bit characteristic, and 36 bit mantissa. A characteristic of 2000_8 indicates a real point immediately to the left of the most significant bit of the mantissa. Positive powers of two are represented by characteristics exceeding 2000_8, etc. When the mantissa is in ones' complement form (negative sign), so also is the characteristic. In most operations, the B Register is set with the low order bits in floating point format. A modifier (UN) permits results to be left unnormalized.

[*] The Logical Register is actually a part of Test Memory, but is mentioned here because of its close association with arithmetic operations.

4.2 Fixed Point

The byte activity modifier (A, also called AC) permits bytes of the memory word to be masked as the word is brought from memory. Therefore, when packed data is brought from memory, if it coincides with a machine byte or bytes, it is masked as a function of the fixed point instruction. (In this context, a fixed point instruction is any instruction of the Load, Arithmetic, and Store classes.) The instruction modifier is actually an 8 bit mask, a "one" bit causing the corresponding byte to be cleared. This modifier is also available for some logical instructions. In store class instructions it refers to bytes of the effectively addressed register.

The byte displacement modifier (P, also called PS) permits the cycling of the memory word 0 to 7 byte positions to the right prior to the fixed point operation, and prior to when byte activity has occurred. The modifier itself is merely a 3-bit number. This function is also available with Store class instructions, but refers to displacement of the word to be stored. In coding, the activity and displacement modifiers are coded as a single modifier. This consists of coding, first, the active bytes of the register referred to in the instruction (numbered 0 - 7), a comma, then the active bytes of the register referred to in the director. For example:

```
LDA,012,567        10A
```

says, "Load the Accumulator, bytes 0, 1, and 2 with bytes 5, 6, and 7 of location 10A," setting the other Accumulator bytes to zero.

The mode modifier (M) sets the structure of the arithmetic element for the particular instruction. The modes are full word, dual, left, right, and twin. Full word permits operation on a signed 48 bit number. Dual operates on 2 signed 24 bit numbers, in a manner similar to the AN/FSQ-7. Left operates on the left-half of the arithmetic element only, leaving the right-half unaffected. Right, of course, is the converse of left. Twin, after displacement has taken place, duplicates the left-half word into the right-half word, then functions as dual. (NOTE: Byte activity takes place AFTER twinning.) Mode in NO way affects displacement. Mode, except for the twin mode, is also available with those Store class instructions which arithmetically affect the memory word (AØR, SØR, ATR).

The other fixed point arithmetic modifier is the signed data tag (T, also called SG). When this one bit modifier is set, all bits of all inactive (masked out) bytes are set equal to the sign bit of the leftmost active byte. This permits the setting of those bits to ones when a negative number is being brought into the arithmetic element. The operation of this modifier is a function of mode. That is, if in dual mode, this function occurs independently in each half-word, etc. The accessed word, after these modifications (A, P, and S), is generally referred to as the "configured contents of ....."

5. Interrupt System

An extremely complete interrupt system permits halt, branch, or ignore interrupt on 48 separate conditions, with both manual and program control of the options. Also, there are 12-48 bit registers available in the complete system to indicate the presence of various machine conditions. In addition to the machine malfunctions which could be reflected, there are such program relevant conditions as several types of inactivity, and various states of individual I/O devices.

6. I/O

Input-output operations have separate logic depending upon the speed. High speed (HS) operations include drums (2.75 usec per word), Input Memory, and I/O Register (2.4 usec per word) transfers, and output Burst Time Counters. All transfers must be either to or from central computer core memory. As is obvious from the transfer rates indicated above, machine registers are used for I/O word count and I/O address count. For low speed I/O, core memory registers are used to maintain this information, counts being established for each unit (see Appendix B). This block of registers is located at the end of one of the banks of memory, and they are referred to as Bookkeeping Address Registers or BAR's. The specific bank is selected by an "operate" instruction. A useful consequence of the low speed (LS) I/O logic is that the word count or address can be changed under program control at any time. In operation, when a word is ready to be transferred on LS operations, the LS I/O Word Counter and LS I/O Address Counter are loaded from the memory registers corresponding to the unit ready to send or receive a word, the word is transferred, then the core memory registers are set with the new values. For all cases except typewriter inputs, the word count is down from a positive number to + 0. As may be deduced, all LS I/O devices may be in operation simultaneously, except printer and punch. This one limitation is because the punch uses part of the printer buffer.

I/O operations are initiated by one of two instructions, SHØ or SLØ for HS and LS operations, respectively. Various modifiers indicate read or write, special operations, etc. The address portion indicates the starting address of a one or two word set which supplies the additional data for the transfer. Illegal operations, both non-existent and those pertaining to a temporary unit status, will NOT hang-up the machine. They are ignored except for the setting of certain indicators in the interrupt system.

The drum system consists of two units, DATOR and Storage. All drums have seventeen fields, 8192 words per field. The DATOR unit has but one drum, most fields of which are for output buffering. Drum registers are addressable on all fields. The Storage Unit may have up to four drums, used solely as auxiliary storage. The Q-32 has two auxiliary storage drums. A time saving feature possible with drum transfers is the block transfer. This permits a transfer to start immediately, i.e., access time of about 11 usec, and the entire drum field is transferred in one drum revolution. For this type of transfer, the starting memory location will be either the first or 8193rd register of a memory bank and the number of words must be 8192.

Low speed I/O on the Q-32 includes a 600 line a minute printer via an IBM 1401, 250 card a minute column binary or Hollerith reader, 100 card a minute row binary punch, I/O typewriter, output (FIX) typewriter, and up to four tape adapters (TA), each TA handling up to eight IBM 729-IV tape drives. The Q-32 has two TA's. The 729-IV handles high and low density tapes (200 or 555.5 characters per inch) in either binary or Hollerith format (binary and Hollerith differ in the nature of parity). Parity is two dimensional, i.e., character and track. The 1401 may also have up to 5 tapes and may be used as an off-line tape-to-printer unit capable of reformatting data.

7. Modifier and Instruction List

The following abbreviations are used in the descriptions that follows:

c(x)	the contents of the addressed register
C	configured
ACC	Accumulator

A summary of modifiers and instructions is in Appendices C and D.

7.1 Load Class

7.1.1 Modifiers

A and P	Activity and displacement	see 4.2
M	mode	see 4.2
T	signed data	see 4.2
R	real data	see 3

7.1.2 Instructions

LDA - Load Accumulator
Transfers Cc(x) into ACC.

LDB - Load B Register
Transfers Cc(x) into the B Register.

LBC
LBC - Load B Register Complement
Transfers the complement of Cc(x) into the B Register.

LDM - Load Magnitude
Transfers the magnitude of Cc(x) into ACC.

LMC - Load Magnitude Complement
Transfers the complement of the magnitude of Cc(x) into ACC.

LDC - Load Complement
Transfers the complement of Cc(x) to ACC.

LDL - Load Logical Register
Transfers Cc(x) to the Logical Register.

LLC - Load Logical Complement
Transfers the complement of Cc(x) to the Logical Register.

7.2 Arithmetic Class

7.2.1 Modifiers

Same as Load Class (see 7.1)

7.2.2 Instructions

ADD - Add
Adds Cc(x) to c(ACC), sum placed in ACC.

SUB - Subtract
Subtracts Cc(x) from c(ACC), difference placed in ACC.

ADM - Add Magnitude
Adds the magnitude of Cc(x) to c(ACC), sum placed in ACC.

SBM - Subtract Magnitude
Subtracts the magnitude of Cc(x) from c(ACC), difference placed in ACC.

DIM - Difference in Magnitude
Subtracts the magnitude of Cc(x) from the magnitude of c(ACC), placing the result in ACC.

MUL - Multiply
Multiplies Cc(x) by the c(ACC) placing the high order bits in the ACC and the low order bits in the B Register. The sign is placed in the sign bits of both the ACC and the B Register.

MLR - Multiply and Round
Same as MUL except upon completion of the instruction the magnitude of c(ACC) will be incremented by one if the most significant B Register magnitude bit is different from the sign bit. The B Register will be the same as in MUL.

DVD - Divide
Divides the contents of the accumulator - B Register (in the same format as following a MUL) by Cc(x). The quotient is placed in the ACC and the remainder in the B Register. However, the remainder takes on the sign of the quotient (being complemented, as necessary).

7.3 Floating Point

7.3.1 Modifiers

UN - Unnormalized (see 4.1)

7.3.2 Instructions

FLT - Float
Using the rightmost eleven bits of c(x) as the positive characteristic, floats and normalizes the fixed point word in the ACC. UN does NOT apply.

FRN - Floating Round
Increases the magnitude of the floating point word in the ACC by one if the most significant bit of the B Register mantissa differs from the sign. B Register is unaffected. UN does NOT apply.

FAD - Floating ADD
Using floating point arithmetic and word format adds c(x) to c(ACC), placing the most significant bits in the ACC and the least significant in the B Register in floating point format.

FAM - Floating Add Magnitude
Same as FAD except uses the magnitude of c(x) instead of signed c(x).

FSB - Floating Subtract
Same as FAD except subtracts instead of adds.

FSM - Floating Subtract Magnitude
Same as FAD except subtracts the magnitude of c(x) instead of adds c(x).

FMP - Floating Multiply
Similar to FAD except multiplies instead of adds. If either operand is not normalized, the result will not be normalized.

FDV - Floating Divide
In floating point arithmetic and using floating point formats, divides c(ACC) by c(x). The quotient is placed in the ACC and the remainder in the B Register. UN does not apply.

7.4 Store Class

All store class instructions involve two data cycles and are "selective store" instructions. A one cycle store instruction is included in the miscellaneous class.

7.4.1 Modifiers

A and P	Activity and displacement	see 4.2
M	mode	applies only to AØR, SØR, and ATR and twin does not apply. (see 4.2)
T	signed data	applies only to AØR, SØR, and ATR. (see 4.2)

7.4.2 Instructions

STA - Store Accumulator
Transfers Cc(ACC) to select bytes of x.

STB - Store B Register
Transfers Cc(B Register) to selected bytes of x.

STP - Store Program Register
Transfers Cc(Program Register) to selected bytes of x. For purposes of this instruction, the Program Register is treated as a 48 bit register having zeros in bytes 0 - 4. The Program Register is set as a result of a transfer of program control and is further discussed under branch instructions (see 7.5).

ECH - Exchange
Exchanges Cc(ACC) with selected bytes of x and vice versa. Inactive bytes of both registers are unaffected.

STL - Store Logical Register
Transfers Cc(Logical Register) to selected bytes of x.

AØR - Add One Right
Adds one to the rightmost bit of Cc(x), according to mode. Only active memory bytes are affected. ACC contains incremented Cc(x), according to mode.

SØR - Subtract One Right
Same as AØR except subtracts instead of adds.

ATR - Add to Register
Adds Cc(x) to c(ACC), transferring selected bytes back to x. ACC contains sum of Cc(x) plus c(ACC).

STX - Store Index
Transfers the c(index register "XR") into rightmost three bytes of x, unless H modifier is set (coded "L"), in which case stored into rightmost three bytes of left-half word (bytes 1, 2 and 3) of x. Index register to be stored is indicated by XR instruction modifier, coded as the appropriate number. The arithmetic configuration modifiers (A, P, N, and T) do NOT apply. ACC and Program Counter may NOT be used.

7.5 Branch Class

Whenever branching (transfer of program control) takes place, the Program Register is loaded with the contents of the Program Counter (location of instruction, plus 1). The nature of some of the modifiers is such that separate mnemonics are included in SCAMP which set those modifiers.

7.5.1 Modifiers

A	byte activity	Specifies which bytes to test. Coded as letter "A" followed by 1 to 8 digits (byte numbers).
B	Byte (of Logical Register)	Specifies byte of Logical Register whose contents are to be used for BAL, BLC, and BLN.

7.5.2 Instructions

Branch on Sign

Conditional branch depending on signs of <u>all</u> active bytes. (If "A" modifier is not coded, byte 0 will be active.)

BØP - Branch on Plus
Branches if all active bytes positive.

BNP - Branch not Plus
Falls through if all active bytes positive.

BØM - Branch on Minus
Branches if all active bytes negative (minus).

BNM - Branch not Minus
Falls through if all active bytes negative (minus).

Branch on Zero

Basic instruction examines all active bytes for zero. The Z modifier indicates plus only (coded P), minus zero only (coded M), or plus or minus zero but all having the same sign (coded "blank"). (If "A" modifier is not coded, all bytes will be active.)

BØZ - Branch on Zero
Branches if all active bytes zero.

BNZ - Branch not Zero
Falls through if all active bytes zero.

Branch on Sense

Basic instruction branches if unit tested is on. For some units, the unit is turned off after testing. Unit is specified by U modifier, coded as three octal digits. Units are listed in N-17334.

BSN - Branch on Sense Unit On
Branches if unit on.

BSF - Branch on Sense Unit Off
Branches if unit off. If used with a PER "U" code, branches and performs operation.

Branch on Logical Compare

Basic instruction compares Logical Register byte specified by B_2 modifier (coded as a single octal digit preceeded by the letter "L") with the BC modifier (coded as two octal digits).

BLC - Branch on Logical Compare
Branches on equality.

BLN - Branch on Logical No Compare
Branches on inequality.

BAL - Branch to Address Plus Logical Register

Unconditional Branch. Branch location is determined by (after indexing, etc.) taking the byte of the Logical Register specified by B_2 modifier (coded "L octal digit") masking out bits to the left of the bit specified by L_2 modifier, shifting right the number of places specified by L_3 modifier, and adding to effective address for final effective address. The L_2 and L_3 modifiers are coded as a single modifer, using the letter "G" followed by two digits each in the range 1-6, where the first digit indicates the first bit to be used and the second digit the last bit to be used.

BAR - Branch to Address Plus Register

Unconditional branch. Branch location is determined by (after indexing, etc.) adding the AF modifier (coded as two octal digits) to 777600 to arrive at the address of an addressable machine register (see Appendix A), then takes the rightmost 18 bits of that word (or rightmost 18 bits of left-half word, if H modifier is set, coded L) and adds it to the effective address for the final effective address.

7.6 Branch and Decrement Class

7.6.1 Modifiers

DF	Decrement Field	18 bit modifier used for modifying or comparing with index value (op code reduced to 6 bits for this class). Coded as one to six octal digits, or in decimal, or tag.
IX	Index	4 bit address modifier (see section 3) used to specify, for these instructions, the index to be modified or compared. Not used for address modification except on subsequent references of indirect addressing.

7.6.2 Instructions

The instructions may be segregated into three classes, those which conditionally branch based on comparing c(index) and c(DF), those which conditionally branch based on sign of index then modify by amount of c(DF), and those which unconditionally branch and modify c(index) with c(DF).

7.6.2.1 Conditional Branch

BXH - Branch on Index High
Branches if c(index) greater than c(DF).
($+0 > -0$ for this test.)

BXL - Branch on Index Low
Branches if c(index) less than c(DF).
($+0 > -0$ for this test.)

BXE - Branch on Index Equal
Branches if c(index) equal to c(DF).
($+0 \neq -0$ for this test.)

7.6.2.2 Conditional Branch and Modify
(ACC and Program Counter may NOT be used.)

BPX - Branch on Plus Index
Branches if sign of index is positive.
If branching occurs, c(index) reduced by c(DF). (Note: if c(DF) is negative, the c(index) would be effectively incremented.)

BMX - Branch on Minus Index
Branches if sign of index is negative.
If branching occurs, c(index) increased by c(DF).

7.6.2.3 Unconditional Branch and Modify
(ACC and Program Counter may NOT be used.)

BSX - Branch and Set Index
Branches unconditionally and sets c(index) = c(DF).

BAX - Branch and Add to Index
Branches Unconditionally and set c(index) = c(index) + c(DF).

7.7 Logical Class

The diversity of the logical instructions is such that they are described herein as five sub-classes.

7.7.1 Sub-Byte Manipulation

These instructions operate upon bits within a particular byte.

7.7.1.1 Modifiers

B_1	Byte	Designates byte to be used by instruction. Coded "B" digit.
BS	Bits and Shift	Designates starting and ending bits of byte. Coded "G" and two digits (digits in range 1-6).
MK	Mask	Designates mask for byte. Coded "K" and two octal digits. (Ones designate active bits.)

7.7.1.2 Instructions

INS - Insert
The bits of the "V" modifier (coded as two octal digits) are inserted into the byte of the addressed register specified by the B_1 modifier as masked by the six bit "MK" modifier. Or, if the "MK" modifier is coded "L" and one octal digit, the three rightmost bits of the "MK" modifier are used to select the byte of the logical register to be used as a mask.

CØM - Complement Bits
Complements those bits of the addressed register in the byte specified by the B_1 modifier, according to the mask in the "MK" modifier. That is, complements only where there is a one bit in the mask.

LDS - Load and Shift
Clears the full accumulator, then loads, right justified, in the right accumulator, those bits specified by the "BS" modifier, of the byte specified by the B_1 modifier, of the addressed register. Real data may be used (see section 3).

STS - Store and Shift
Essentially the reverse of LDS. Takes the least significant bits of byte 7 of the accumulator and deposits them into the "BS" bits of "B_1" byte of the addressed register. It affects NO other bits of the addressed register, does NOT change the ACC, and may NOT use real data.

TST - Test
Increments the Program Counter by the contents of bits "BS" of byte "B_1" of the addressed register. In effect, is a test of from 1 to 6 bits resulting in a "skip" of 0 to 64 instructions.

7.7.2. Boolean Operations

The following instructions are various mnemonics used with the single "connect" instruction, the differences being various "pre-set" modifiers. The heart of the instruction is a 4 bit "truth table." Basically, the instruction generates a word, on a bit by bit basis, as a function of the addressed (or "storage") word, the ACC, and the truth table. The truth table is used as follows:

storage word bit	0 0 1 1
ACC word bit	0 1 0 1
resulting word bit	$b_1b_2b_3b_4$

where, for a given bit position, finding the column which represents the particular combination at a given bit position, the modifier bit (b_1, b_2, b_3, or b_4) designates how that bit will be in the resulting word. For example, a "CN" modifier (the truth table) "b" values of 0001 would result in an "AND" or "logical multiplication" operation.

The various mnemonics also include whether the generated word is placed in storage (the addressed register) or in the ACC. In all cases it is also possible to use byte activity OR a mask, but not both. In either case, the inactive bytes, or bits, of the receiving register are unaffected. Byte activity is coded by the letter "A," followed by the numbers of the active bytes. Masking is indicated by the letter "K." The B Register is used as the mask.

Real data may be used for those instructions affecting ACC.

CTA	Connect to Accumulator
CTS	Connect to Storage

These instructions are the general case of "Connect" and require the coding of the "CN" modifier (coded as four binary digits immediately following the mnemonic). CTA places the resulting word (or portions thereof, due to activity or masking) in the ACC. CTS, in a similar manner places the result in storage (addressed register).

ANA	AND to Accumulator
AND	AND to Storage

The truth table is automatically set to give a "logical multiply" operation (CN = 0001). Otherwise same as CTA, CTS.

ØRA	OR to Accumulator
ØRS	OR to Storage

The truth table is automatically set to give an "inclusive OR" operation (CN = 0111). Otherwise same as CTA, CTS.

DEP	Deposit

The truth table is automatically set to transfer ACC zeros as zeros and ACC ones as ones (CN = 0101). Otherwise identical to CTS. Note: unless a mask is used, the function may be performed by STA.

EXT	Extract

The truth table is automatically set to transfer c(x) to the ACC(CN = 0011), otherwise identical to CTA. Note: this instruction, using byte activity, differs from LDA in that inactive bytes of the ACC are unaffected.

7.7.3 Compare Operations

A single instruction is available which permits a wide range of comparison operations. The variety of these, and the affects, are so different that they are discussed herein as four separate instructions.

7.7.3.1 Modifiers

MS	Mask Selector	Causes the B Register to be used as a mask. When a mask is used, activity, displacement and signed data do not apply. (Coded "K")
A, P	Activity and Displacement	See section 4.2
T	Signed Data	See section 4.2

7.7.3.2 Instructions

CML Compare Logically

Compares Cc(x) with active bytes of ACC, or the masked c(x) with <u>the</u> <u>entire</u> c(ACC), and, if identical, skips one instruction. T modifier does not apply, and c(x) and c(ACC) are unaffected.

CDL Compare and Difference Logically

Same as CML plus, the "logical difference" is left in the ACC active bits, or bytes. (Logical difference is that value which results from a "CTA0110) instruction.) Inactive bits or bytes are unaffected.

CMA - Compare Arithmetically
Arithmetically compares Cc(x), or the masked c(x) with the original c(ACC). If $c(x) < c(ACC)$, the next instruction is operated, if $c(x) = c(ACC)$, one instruction is skipped, and if $c(x) > c(ACC)$, two instructions are skipped. c(x) and c(ACC) are unaffected.

CDA - Compare and Difference Arithmetically
Same as CMA, plus the masked or configured c(x) is subtracted from c(ACC), using "full" mode, the difference being placed in the ACC.

7.7.4 Index Instructions

The index instructions are of two types, those pertaining to the contents of index registers and those pertaining to the contents of the "Double Index Selection Register."

7.7.4.1 Index Modification

7.7.4.1.1 Modifiers

XR	Index Register	Specifies index to be modified. Not to be confused with IX which is the index register to be used in address modification (Both apply.) Coded as one decimal digit.
H	Half Word	Specifies half-word of addressed register to be used. Coded as "L" or "R". (Blank is equivalent to "R".)
R	Real Data	For LDX and ATX, if Real Data is specified, indexing and double indexing will modify the Real Data.

7.7.4.1.2 Instructions

LDX - Load Index
Transfers rightmost 18 bits of the "H" half-word of c(x) to index register "XR."

ATX - Add to Index
Modifies c(index "XR") by adding the rightmost 18 bits of the "H" half-word of c(x).

7.7.4.2 DISR Manipulating Instructions

7.7.4.2.1 Modifiers

B_1	Byte	Designates byte of addressed register to be used. Coded "B" one digit.
R	Real Data	see 3

7.7.4.2.2 Instructions

LDI - Load DISR
Sets c(DISR) = rightmost four bits of byte "B_1" of c(x).

SDI - Store DISR
Sets byte "B_1" of c(x), first two bits = 0, rightmost four bits = c(DISR).

7.7.5 Convert Instructions

Basically, the convert instructions sequentially use bytes of the ACC or B Register to modify an effective address, and replace the byte with data found in the referenced register. This permits very rapid conversion between various 6-bit coding schemes. Since they also set an index on the final iteration, they are very useful for multi-level table look-up operations.

7.7.5.1 Modifiers

XR	Index Register	Same as 7.7.4.1.1, above.
NC	Number of Converts	Number of iterations. Caution must be exercised as number used is modulo 128. Also, 0 is equivalent to 1. Coded as "N" and up to three decimal digits.
G_2	Direction of Operation	Direction in which operation proceeds. Applies only to CRA. Coded "L" or "R", blank is equivalent to "L."

7.7.5.2 Instructions

CRA - Convert by Replacement from Accumulator

Computes final effective address by adding c(ACC, byte 0) to effective address. Sets c(ACC, byte 0), = c(x, byte 0). Cycles ACC left one byte position. (c(byte 0) moved to byte 7.) Uses c(x, bytes 5, 6, and 7) as effective address for next iteration. Continues for "NC" iterations. Following last iteration, sets index "XR" = c(x final, bytes 5, 6, and 7). If H modifier indicates "right" direction uses ACC byte 7 and cycles right, but use of c(x) remains same.

CAB - Convert by Addition from B Register

Addresses in a manner similar to CRA except the B Register is used and only "left" is possible. However, instead of replacing B Register bytes, c(x, bytes 0-4) are added to the ACC, bytes 3-7, using full mode. "XR" and "NC" have same use as with CRA. ACC is cleared prior to first iteration.

7.8 Miscellaneous Class

Miscellaneous class instructions can be subdivided into three groups, (1) miscellaneous, (2) shifting and cycling, and (3) I/O.

7.8.1 Miscellaneous Group

PER - Operate
Performs an operation determined by the 9 bit "U" modifier (coded as three octal digits). These operations are listed in N-17334. Addressing does not apply.

HLT - Halt
Stops program operation. A manual "Continue" action restarts operation with next instruction. Addressing does not apply.

STE - Store Exchange Register
Transfers c(Exchange Register) to x. NO <u>instruction</u> modifiers apply.

STF - Store Full Word
Transfers c(ACC) to x. NO <u>instruction</u> modifiers apply. Uses only one data cycle.

STZ - Store Zeros
Transfers zeros to x. Special case of STF with SZ modifier set. No other <u>instruction</u> modifiers apply.

XEC - Execute
Operates the instruction at location x. If c(x) = active branch, branching occurs and the program register is set with the address of the instruction following the XEC. If c(x) = any type of "skip" instruction, location of the next instruction will be a function of the location of the XEC. If c(x) = XEC, the effect is the same as if the first XEC contained the addressing information of the second (or final) XEC instruction. All other instructions result in the instruction following the XEC to be operated after the object instruction. XEC does NOT cause the program counter to be stopped.

BUC - Branch Unconditionally
Unconditional transfer of program control to location x. (All address modifiers, except real data, apply.) If the "XR" modifier contains the number of an index, that index will be set

with the location of the instruction following the BUC. The Program Register is set with the address of the instruction following the BUC.

BUS - Branch Unconditionally with Same Program Register.

Special case of BUC with PR modifier set to 1. Inhibits changing of program register.

7.8.2 Shifting and Cycling Group

There are three basic instructions in this group, shift, cycle, and normalize. Shifting is moving all bits but the sign(s), losing bits at one end and obtaining sign duplicates at the other. Cycling is moving all bits as if the register were a ring register (no bits lost). See NØR below for normalizing.

7.8.2.1 Modifiers

M	Mode	See section 4.2. Twin does not apply.
G_1	Direction	Actually an "address" bit. Bit 30 (leftmost bit of the address "field" of the data word) indicates direction of movement of bits, positive-right, negative-left.
N	Number	Also "address" bits (41 - 47 of the data word). Specifies number of bit positions to shift (or cycle), modulo 128. Will be complemented if "G_1" is negative.
R	Real Data	If on, director of instruction, after indexing and double indexing, used to obtain "G_1" and "N". If off, c(x) are used, without further modification. Coded "R".
D	Rounding	Applies only to shifting. Effect same as round with MLR. Coded "N".

7.8.2.2 Instructions

SFA - Shift Accumulator
Shifts ACC according to mode, "N" positions.
Rounds ACC if "D" is set.

SFB - Shift B Register
Shifts B Register according to mode, "N" positions.
Rounds ACC if "D" is set.

SFC - Shift Combined
Shifts combined ACC - B Register, according to mode, "N" positions. ACC - B Register linkage is a function of mode, but for the general case, it can be said the most significant B Register <u>magnitude</u> bit is adjacent to the least significant ACC bit, i.e., the B Register sign bit is bypassed. Rounds ACC if "D" is set.

CYA - Cycle Accumulator
Cycles ACC "N" positions, according to mode.

CYB - Cycle B Register
Cycles B Register "N" positions, according to mode.

CYC - Cycle Combined
Cycles combined ACC - B Register "N" positions according to mode. ACC - B Register linkage is a function of mode, but for the general case it can be said, the B Register sign is adjacent to the least significant ACC bit and the ACC sign is adjacent to the least significant B Register bit.

CYI - Cycle Inverted
Same as CYC except as follows: only "full" mode, and, the B Register and ACC signs are adjacent, and the B Register and ACC least significant bits are adjacent. Direction refers to ACC.

NRA - Normalize Accumulator
Shifts the ACC left, according to mode, until the most significant magnitude bit is different from the sign. In dual mode, shifting stops when the above condition first occurs in either half. The number of shifts required is then placed in the right half of c(x). If there are no "1's", the operation stops after 127 shifts. "R" does NOT apply.

NRC - Normalize Combined
Same as NRA except ACC and B register are linked as for SFC.

7.8.2.2 Instructions

SFA - Shift Accumulator
Shifts ACC according to mode, "N" positions. Rounds ACC if "D" is set.

SFB - Shift B Register
Shifts B Register according to mode, "N" positions. Rounds ACC if "D" is set.

SFC - Shift Combined
Shifts combined ACC - B Register, according to mode, "N" positions. ACC - B Register linkage is a function of mode, but for the general case, it can be said the most significant B Register magnitude bit is adjacent to the least significant ACC bit, i.e., the B Register sign bit is bypassed. Rounds ACC if "D" is set.

CYA - Cycle Accumulator
Cycles ACC "N" positions, according to mode.

CYB - Cycle B Register
Cycles B Register "N" positions, according to mode.

CYC - Cycle Combined
Cycles combined ACC - B Register "N" positions according to mode. ACC - B Register linkage is a function of mode, but for the general case it can be said, the B Register sign is adjacent to the least significant ACC bit and the ACC sign is adjacent to the least significant B Register bit.

CYI - Cycle Inverted
Same as CYC except as follows: only "full" mode, and, the B Register and ACC signs are adjacent, and the B Register and ACC least significant bits are adjacent. Direction refers to ACC.

NRA - Normalize Accumulator
Shifts the ACC left, according to mode, until the most significant magnitude bit is different from the sign. In dual mode, shifting stops when the above condition first occurs in either half. The number of shifts required is then placed in ~~the~~ location X, bits 41-47, clearing S-40, ~~right half of c(x)~~. If there are no "1's", the operation stops after 127 shifts. "R" does NOT apply.

NRC - Normalize Combined
Same as NRA except ACC and B register are linked as for SFC.

7.8.3 Input - Output Group

A partial discussion of I/O is in Section 6. The various instruction modifiers take on different meaning depending on the instruction, and sometimes depending on other modifiers, therefore, they are discussed only in specific contexts.

SHØ - Start High-Speed Operation

Initiates an operation pertaining to a High-Speed (HS) I/O unit. The RD and WR modifiers are used to indicate read or write, (coded "R" and "W", respectively). The "S" modifier is used to select the specific unit as follows:

00	-	Storage Drum A
01	-	Storage Drum B
04	-	DATOR Drum
30	-	Input Memory
60	-	Burst Time Counters
70	-	H.S. I/O Register

The "U" (operate) modifier is really three one-bit modifiers, in that the values discussed below may be combined for multiple operations. "U1" locks the I/O address counter, causing the same register to be addressed for all words transferred. When the I/O Register is selected, "U4" inhibits clearing the I/O Register, thus permitting the generation of a logical sum when writing, or, on a subsequent read, transferring the same constant into two or more consecutive registers. When a drum is selected, "U4" inhibits field stepping, that is, when the last address on a field is reached, the next drum register to be used is the first one on the same field instead of the normal mode of the first register of the next higher numbered field. However, field stepping is always inhibited at the end of DATOR fields 5-17, that is, the buffer fields do not have field stepping. Mode is NOT related to arithmetic mode. "M00" (or not coded) is normal, "addressable" transfer. "M01" is inter-leaved by 2 (every other drum address). "M02" is inter-leaved by 4. "M04" is a block transfer (see Section 6). "M10" is destructive readout when Input Memory is selected. "M20" will cause all registers of all fields of the selected drum to be erased, however, the computer must be in "test mode" for this instruction. The balance of the data required for the transfer is contained in control words, the first of which is in the register addressed by the SHØ instruction (IX, I, and D apply), the second, if needed, immediately following the first. For I/O Register transfers, a single word is used containing the starting memory address in bytes 5, 6, and 7 and number of words in bytes 1, 2, and 3. (IX, I and D do NOT apply to the control word contents.)

See CTL, below, for coding. For drum transfers, the first word contains the field selection in bits 30 - 34 and drum address in bits 35 - 47. The second word has the same format as the I/O Register control word.

CTL - Control Word - General
A Pseudo instruction to properly format a "general purpose" control word. Director is, first the number of words to be transferred, a comma, and the starting memory address for the transfer. CTL may be used as an RC word.

DRC - Drum Control Words
Similar to CTL but used for drum control words. Address field contains (1) drum field, (2) starting drum address, (3) number of words, and (4) starting core memory address, each separated by a comma.

DRM - Start Storage Drum Operation
Equivalent to an SHØ00, or SHØ01, the letters "A" or "B", respectively, differentiating.

IØR - Start I/O Register Operation
Equivalent to an SHØ70.

SLØ - Start Low-Speed Operation
Initiates an operation pertaining to a low-speed (LS) unit. For LS transfers, the control information is automatically transferred to the appropriate "Bookkeeping Address Registers" (BAR) in core memory (see Section 6 and Appendix B), leaving the control word unchanged. As words are transferred, the LS I/O element obtains the needed information from the appropriate BAR, modifies it and tests word count for transfer completion, then restores modified values to the BAR. (The number of words is maintained as a positive number, reaching + 0 when all words are transferred, except as noted below for the I/O typewriter.) Reading and writing is as for SHØ, above. The unit selection codes ("S" modifier) are as follows:

01	Reader
02	1401 (Printer)
03	Punch
04	I/O Typewriter
06	Output (FIX) Typewriter
1x	Tape Adapter 1
2x	Tape Adapter 2

("x" represents the number of the Tape Drive (TD) selection. A given TD is associated with only the one Tape Adapter (TA). TD numbering is a function of manual selection on each TD, using 0 - 7 for TD1 - TD8.)

The I/O typewriter, as an output device, may be coded ("TM" modifier) "A" for alphanumeric mode, in which case the code in Appendix F is used. If the "A" is not coded, the octal mode is used wherein the contents of the registers transferred are automatically converted directly from binary to octal. The control word is identical to that for I/O Register transfers. The FIX typewriter operates only in alphanumeric.

For purposes of the SLØ instruction, the I/O typewriter is an output device only. However, when not being used as an output device, it may be used either as a variable or fixed entry device. As a variable entry device, it transfers octal words to any register(s) of core memory, independent of program control. As a fixed entry device, it transfers up to 32 words of 6 bit characters under limited program control. In this latter mode, the program must set the starting memory address and number of words into the FMBW (see Appendix B). For this operation only, word count proceeds in a positive direction to 32. For example, an original setting of 10 would permit 22 words (176 characters) to be entered. After setting the FMBW, a PER330 is necessary to "enable" the typewriter.

TYP - Typewriter

This mnemonic is equivalent to a SLØ,04.

Tape transfers are considerably more complex. The hierarchy of data on tapes is character (six data bits plus parity strung across the tape), record (a series of closely packed characters separated from other records by a gap of about 3/4 inch plus a track parity character), file (a series of records terminated by a special record consisting of a single character 1111000, plus "track parity"), and logical tape (a series of files terminated by a special record consisting of four of the "file" characters plus track parity). The end-of-logical-tape character has the same program effect as the permanent physical-end-of-tape mark, except it may be erased by writing over it. "U" modifier codes permit special operations as follows:

U1	Backspace one record
U2	Rewind to load point
U3	Write "end-of-file" (EOF)
U4	Write "logical-end-of-tape" (EOT)
U5	Backspace to end of previous file
U6	Set High Density
U7	Set Low Density

If none of the above are coded, it should be a data transfer. Density refers to 555.5 characters per inch or 200 characters per inch. Density control is set manually at the TD but may be overridden under program control, as above, but is effective ONLY if the tape is at load point, i.e., a single tape may be high or low density but not a mixture. Related to this, but serving a different function, is a modifier indicating the density expected. If the TD is set for the opposite density, no transfer will occur and an indicator will be set in the Interrupt System. Coding assumes high density unless an "L" is coded. The "normal" "code" for tapes is assumed to be binary. If it is a binary-coded-Hollerith tape, the letter"H" should be included as a modifier of the SLØ instruction. The difference between binary tapes and binary-coded-Hollerity (BCH) is that binary have odd character parity whereas BCH have even character parity and binary records have an additional, non-data, identification word. There are two control words used with tape I/O. However, the second word is used only with binary tapes. (Therefore, a "CTL" may be used in BCH transfers.) The second control word contains, in bytes 0, 1, 2 and 3, the "identity," and in bytes 5, 6 and 7 the "tape identity address." These may be coded with the mnemonic TPL and the following four pieces of information in the address field, separated by commas: (1) identity, (2) identity address, (3) number of words, and (4) starting memory address. "Identity" is used in reading and writing binary tapes. The first "word" of a <u>binary</u> record contains the "identity," but this word is not included as a data word for word count or memory addressing purposes. In writing, it is obtained from the second control word if the "SI" modifier is set, or is zeros if the "SI' modifier is not set. In reading, if the identity word is desired, the modifier "S" ("TA" modifier) is used so that the identity word from the tape will be stored in the register whose address is found in the "tape identity address" portion of the second control word. The identity word is also used in the file search function. If an "I" is coded ("SI" modifier), the TD will start reading the identity words of records, and, when it finds a <u>binary</u> record whose identity matches the "identity" portion of the control word, transfers the data in THAT record. If it reaches an EØF or EØT before finding such a record, no data is transferred and the EØF indicator is set. "TA" and "SI" may NOT both be set.

The following mnemonics apply for SLO for tape:

TAP - Tape
Same as SLØ, except will not accept "U" modifiers.

WEF - Write EØF
Equivalent to SLØU3.

WET - Write EØT
Equivalent to SLØU4.

REW - Rewind
Equivalent to SLØU2.

BKR - Backspace one Record
Equivalent to SLØU1.

BKF - Backspace to end of Previous File
Equivalent to SLØU5.

The card reader reads Hollerith or column binary cards, columns 1 - 72, at the rate of 250 cards per minute. The column binary mode is determined by a "7" and "9" punch in column 1, Hollerith being the absence of either or both of these punches. In the binary mode, each four columns determine a binary word, rows 12 - 9 of column 1 being S-11 of the first word, etc. In this mode, 18 words per card may be transferred into core memory. In the Hollerith mode, the punches of each column are converted into six-bit codes such that column 1 goes to byte 0 of the first word, column 2 to byte 1, etc., for a total of 9 words per card. To initiate a read card reader operation, a SLØ instruction is used with a select code of 01, and, of course, "R" to indicate read. The general purpose control word (CTL) is used for word count and number of words. The following mnemonic also applies:

RCD - Read a Card
Equivalent to SLØ,01,R.

The punch produces <u>row</u> binary cards at the rate of 100 cards per minute. For a <u>full</u> card, the punch uses an image of 24 words. The first word goes to the 9 row columns 1 - 48, the left-half of the second word goes to the 9 row columns 49 -72 and the right-half of the second word is not used. The third word goes to 8 row columns 1 - 48, etc. It is possible to offset columns 1 - 8 into columns 77 - 80 and to gang-punch columns 73 - 80. This is a function of plugboard wiring and is accomplished by the PER375 and PER376 instructions (see N-17474/009/00). The SLØ instruction for the punch uses a select code of 03. A general control word is used for number of words and starting address. The following mnemonic also applies:

PUN - Punch
Equivalent to SLØ03.

The 1401 - printer consists of, at present, a 4K memory 1401, a 1403 printer (600 lines per minute, 132 characters per line), and a 729 IV tape. To transfer data to the 1401, the program in the 1401 must first have set its "program ready" flip-flop. If this has been done, a SLØ02 will initiate the transfer of 6-bit coded information to the 1401. (NOTE: The 1401 can be written only.) The 1401 program must then initiate a "read" of the data. If the above mentioned 1401 "program ready" flip-flop was not set at the time the SLØ02 was executed, a 1401 (printer) not ready indication is set in the Q-32 and the SLØ02 is otherwise ignored. From the standpoint of the 1401, the character having an octal code of 77 (group mark) will terminate a transfer, but the 1401 program may initiate another read if the Q-32 has not reduced word count to zero. The consequence of this is two-fold: (1) a "block" of printing may be sent with group marks delineating the ends of lines of print and, (2) the last byte of the last word transferred must contain a group mark to avoid hanging-up the 1401. There are three flip-flop indicators sensible in the 1401 which are set by the Q-32, (1) the "P" flip-flop set by a Q-32 SLØ02 instruction when the 1401's program ready indication is on, (2) the "R" flip-flop set by a PER373, and, (3) the "Q latch" set by a PER 374. All three of these are cleared as a result of the word count going to zero. The 1401 "program ready" flip-flop can be either set or cleared by 1401 instructions. The 1403 printer being not ready will also clear the "program ready" flip-flop, causing a not ready indication if an SLØ02 is executed.

APPENDIX A

ADDRESSABLE REGISTERS

INTERNAL REGISTER	REGISTER* BITS	EXCHANGE REGISTER BITS	OCTAL ADDRESS	SCAMP TAG	READ ONLY	BAR[2] INSTR.
Positive Zero			777600	$Z	X	X
Index Register 1-8	S-17	30-47	777601 to 777610	$X1 to $X8		X
Program Register	1-18	30-47	777620	$P		X
Accumulator			777621	$A		X
B Register			777622	$B		X
Interrupt Control Reg.			777623	$I		
Real Time Clock	1-24	S-23	777624	$CLK		
Double Index Sel. Reg.	1-4	44-47	777651	$D	X	
@Switch Reg. A			777700	$AS	X	
@Switch Reg. B.			777701	$BS	X	
@Switch Reg. C			777702	$CS	X	
@Switch Reg. D			777703	$DS	X	
@TM Control Panel			777704 to 777743	$T1 to $T32	X	
@Live Register			777744	$LIV		
@Logical Register			777745	$L		
@Detail Cond. Reg. 1			777746	$DC1 [1]		
@Detail Cond. Reg. 2-11			777747 to 777760	$DC2 to $DC11	X	
@Detail Cond. Reg. 12			777761	$DC12 [1]		

FIX REGISTERS

Index Buffer Storage	S-17	6-23	777640		X	
P. C. Storage I	1-18	30-47	777640		X	
P. C. Storage II	1-18	6-23	777641		X	
P. C. Storage III	1-18	30-47	777641		X	
Instruction Storage	0-23	S-23	777642		X	
ACC Storage			777643		X	
B Reg. Storage			777644		X	
First Alarm Register	1-6	41-47	777645	$F	X	X
H.S. I/O Word Ctr.	S-17	6-23	777646	$HW	X	
H.S. I/O Addr. Ctr.	1-18	30-47	777646	$HA	X	
Drum Control Reg.	1-18	30-47	777650		X	
Angular Position Stor. Reg	1-13	35-47	777652	$APS	X	
Alarm Time Reg.	1-4	44-47	777653		X	
@System Status Reg.			777762		X	

(NOTE: See Page 34 for footnotes)

REGISTERS AS INDEX REGISTERS

REGISTER	OCTAL CODE	SCAMP TAG
Index Register 1-8	1-10	1-8
Program Counter	16	14 or C
Accumulator	17	15 or A

---

* If Not a 48-bit register.

@ Addressable for instruction words and indirect addresses.

[1] Storing into these registers causes bits of the register to be complemented where there is a "1" in the word being stored, and no action where there is a "0" in the word being stored.

[2] Addressable for Bar Instruction incrementing.

APPENDIX B

BOOKKEEPING ADDRESS REGISTER ASSIGNMENTS

Unit	Bookkeeping Word	Control Bookkeeping Word	Identity or Fixed Mode Bookkeeping Word
Reader	37751	37750	-
1401	37753	37752	-
Punch	37747	37746	-
Tape Adapter 1	37775	37774	37773
Tape Adapter 2	37772	37771	37770
I/O Typewriter	37745	37744	37743
FIX Typewriter	37740	-	-

APPENDIX C

SUMMARY OF MODIFIERS

I. Modifiers coded in address field

Abbreviation	Bit Positions	Name	Coding
IX	26-29	Index	One or two decimal digits, A or C
DI	24	Double Index	D
I	25	Indirect Address	I
R	8	Real Data (Immediate Addressing)	(n)R where "n" represents the value to be placed in address field of instruction word.

II. Arithmetic and Store Instruction Modifiers

A-P	Activity 16-23 Displacement 9-11	Byte Activity and Displacement	Two sets of octal digits separated by a comma (omission = 01234567, 01234567)
MD	12-14	Mode	L,R,D,F, or T (omission = F.)
SG	15	Signed Data	S

Abbreviation	Bit Positions	Name	Coding
III. Logic Class Instruction Modifiers			
CN	12-15	Connective (truth table)	four binary digits
CP	9	Convert Position (Convert Direction)	L or R (omission = L)
BS	start bit 18-20 shift 21-23	Bit and Shift (Starting and Ending Bit)	G and two digits in range 1-6. Second digit must be ≥ first digit.
LB	9-11	Logical Register Byte	"L" and one octal digit
	(15-17 for INS)		(May NOT be used if MK is used)
MB	9-11	Memory Byte	"B" and one octal digit
NC	11-17	Number of Converts	"N" and up to three decimal digits.
MK (form a)	9 (12 for compare instructions)	Mask Selector (use B Register for mask)	"K"
(form b)	12-17	Byte - Mask	"K" and two octal digits (may NOT be used if LB is used)
VL	12-17 (18-23 for INS)	Value	Two octal digits

Abbreviation	Bit Positions	Name	Coding
IV. Input-Output Instruction Modifiers			
CD	15	Coding	H (Hollerith, as opposed to binary)
RD-WR	R-11 W-12	Read-Write	R or W, or neither if "U" is used
SI	14	Status-Identity	I
TA	16	Tape Identity Address (Store Identity)	S
M	13-17	Mode	"M" and two octal digits (omission = M00)
U	8-10	Operate Code	"U" and one octal digit (omission = U0)
S	18-23	Selection Select Drum Selection	two octal digits A or B (with DRM instruction) I (for I/O Register with SHO)
TM	13	Density of Tape Operation Typewriter mode	L (for low density) A (for alphanumeric)

V. Other Modifiers

Abbreviation	Bit Positions	Name	Coding
AR	18-23	Addressable Machine Register	Two octal digits
H	9	Half-Word	"L" or "R", blank = "R"
UT	15-23	Unit (PER-SEN code)	Three octal digits
XR	20-23	Index Register (to be manipulated)	One digit (1-8), blank = none, where legal.
RD	15	Round	N
UN	8	Unnormalized (do not normalize)	U
ZD	10-11	Zero Definition	P or M or blank (either)
AC (or A)	16-23	Byte Activity	"A" and up to eight octal digits (Omission = A01234567 except for BNM, BOM, BNP and BØP where omission = AO)
DF	6-23	Decrement Field	One to six optionally signed decimal digits or one to six octal digits. If octal, followed by a "prime." May also be an address tag.

VI. Modifiers Inherent in Instruction Mnemonics

Abbreviation	Bit Positions	Name	Remarks
NB	8	No Branch	Causes conditional branch to reverse logic of testing conditional branch instructions not pertaining to index registers.
SV	9	Sign Variation	Differentiates between positive and negative test on branch on sign instructions.
Y	8	Mask Selector	Indicates MK used to select logical register byte rather than being used directly. Implied by coding MK as if it were LB.
RS	10	Results Stored	Select ACC or x to store results for connect instructions.
DA	13	Difference in Accumulator	Permits storage of difference in ACC for connect instructions.
CT	14	Compare Type	Logical or Arithmetic comparison.
SZ	8	Store Zeros	Zeros instead of c(ACC) allowing STF to become STZ.
RC	9-11	Register Con-nective	ACC and/or B Register for shift, cycle and normalize instructions.
PR	8	Program Register	If set, unconditional branch will not cause Program Register to be changed.
SX	10	Store Exchange Register	c(Exchange Reg) stored if SX and SZ modifiers both set, making STF instruction an STE.

APPENDIX D

Summary of Instructions

The following abbreviations are used:

-	minus or complemented
+	plus
*	multiplied by
/	divided by
(/ /)	absolute magnitude
AND	Boolean AND
OR	Boolean inclusive OR
$\underline{\text{OR}}$	Boolean exclusive OR
=	is set equal to
=F	is set equal to, and calculations and format are according to floating point arithmetic
=M	is set equal to, according to fixed point arithmetic mode
⊛	cycled
NQ	is not equal to
EQ	is equal to
GR	is greater than
LS	is less than
c( )	the contents of
C	configured
m	masked bits of
$\underline{\text{m}}$	unmasked bits of
C/m	configured $\underline{\text{OR}}$ masked bits of
$\underline{\text{C/m}}$	the inactive bytes or unmasked bits of
$\underline{\text{C}}$	the inactive bytes of
bi( )	bits
by( )	bytes
" "	as determined by the indicated modifier
"AP1"	as determined by the first set of digits coded in the A-P modifier
"AP2"	as determined by the second set of digits coded in the A-P modifier
ACC	Accumulator
B	B Register
ACCB	Combined ACC and B
L	Logical Register
PR	Program Register
PC	Program Counter
DISR	Double Index Selection Register
x	effectively addressed register

Mnemonic	Name	Octal Code	Coded Modifiers	Description
I.	Load Instructions			
LBC	Load B Register Complement	224	A-P,MD,SG,R	B = M -Cc(x)
LDA	Load Accumulator	200	"	ACC = M Cc(x)
LDB	Load B Register	220	"	B = M Cc(x)
LDC	Load Complement	210	"	ACC = M -Cc(x)
LDL	Load Logical	230	"	L = M Cc(x)
LDM	Load Magnitude	204	"	ACC = M (/Cc(x)/)
LLC	Load Logical Complement	234	"	L = M -Cc(x)
LMC	Load Magnitude Complement	214	"	ACC = M -(/Cc(x)/)
II.	Arithmetic			
ADD	Add	100	A-P,MD,SG,R	ACC = M c(ACC) + Cc(x)
ADM	Add Magnitude	104	"	ACC = M c(ACC) + (/Cc(x)/)
DIM	Difference in Magnitude	130	"	ACC = M (/c(ACC)/) - (/Cc(x)/)
DVD	Divide	134	"	ACC = M c(ACCB) / Cc(x) B = M (/remainder/) if new ACC sign = + or - (/remainder/) if new ACC sign = -
MLR	Multiply and Round	124	"	ACCB = M c(ACC) * Cc(x) and c(ACC) is increased in magnitude by "one" if the most significant B magnitude bit differs from the new sign.
MUL	Multiply	120	"	ACCB = M c(ACC) * Cc(x)
SBM	Subtract Magnitude	114	"	ACC = M c(ACC) - (/Cc(x)/)
SUB	Subtract	110	"	ACC = M c(ACC) - Cc(x)

Mnemonic	Name	Octal Code	Code Modifiers	Description
III.	Floating Point			
FAD	Floating Add	300	UN	ACCB = F c(ACC) + c(x)
FAM	Floating Add Magnitude	304	-	ACCB = F c(ACC) + (/c(x)/)

FDV	Floating Divide	334	-	ACC = F c(ACC) / c(x)
				B = F (/remainder/) if new ACC sign = +
				or
				- (/remainder/) if new ACC sign + -

FLT	Float	320	-	ACCbi(1_11) (characteristic) = (/c(xbi(37_47)) -n/) if ACC sign = + or -(/c(xbi(37_47))-n/) if ACC sign = - ACC bi (12_47) = c(ACCbi(1+n_36+n)) (mantissa) Bbi(0) = c(ACCbi(0)) (sign) Bbi(1_11 = new c(ACC bi(1_11)) -36 (characteristic) Bbi(12_47) = c(ACCbi(37+n_47)) "n" in the above represents the number of leading zero-magnitude bits in the original c(ACC).
FMP	Floating Multiply	330	UN	ACCB = F c(ACC) * c(x)

FRN	Floating Round	324	-	ACC = F c(ACC) if c(Bbi(1)) EQ c(ACCbi(0))
				or
				c(ACC)+1 if c(Bbi(1)) NQ c(ACCbi(0))

FSB	Floating Subtract	310	UN	ACCB = F c(ACC) - c(x)
FSM	Floating Subtract Magnitude	314	UN	ACCB = F c(ACC) - (/c(x)/)

Mnemonic	Name	Octal Code	Coded Modifiers	Description
IV. Store				
A. Non-Arithmetic Store				
ECH	Exchange	524	A-P	xby("AP2") = c(ACCby("AP1")) ACCby("AP1") = c(xby("AP2"))
STA	Store Accumulator	500	A-P	xby("AP2") = c(ACCby("AP1"))
STB	Store B Register	504	A-P	xby("AP2") = c(Bby("AP1"))
STE	Store Exchange Register	050	-	x = c(Exchange Register)
STF	Store Full Word	050	-	x = c(ACC)
STL	Store Logical Register	510	A-P	xby("AP2") = c(Lby("AP1"))
STP	Store Program Register	514	A-P	xby("AP2") = c(PRby("AP1")) Note: if AP1 NQ 5,6,7, zeros will be stored for the other byte positions.
STX	Store Index	520	H, XR	xby(5,6,7) = c(index "XR") if H EQ R or blank xby(1,2,3) = c(index "XR") if H EQ L
STZ	Store Zero	050	-	x = 0
B. Arithmetic Store				
AØR	Add One Right	530	A-P, SG, M (M ≠ T)	ACC = M Cc(x) + 1 xby("AP2") = new c(ACCby("AP1"))
ATR	Add to Register	540	A-P, SG, M (M ≠ T)	ACC = M c(ACC) + Cc(x) xby("AP2") = new c(ACCby("AP1"))
SØR	Subtract One Right	534	A-P, SG, M (M ≠ T)	ACC = M Cc(x) - 1 xby("AP2") = new c(ACCby("AP1"))

Mnemonic	Name	Octal Code	Coded Modifiers	Description

V. Branch Class

A. Unconditional

Mnemonic	Name	Octal Code	Coded Modifiers	Description

BAL	Branch to Address Plus Logical	624	LB, BS	PR = c(PC) + 1
				PC = effective address + c(Lby(LB) bi(BS))

BAR	Branch to Address Plus Register	614	H, AF	PR = c(PC) + 1
				PC = effective address + c((777600_8 + AF) by(h))
				"h" represents 5,6,7 if H = R or blank and 1,2,3 if H = L

BUC	Branch Unconditionally	014	XR	PR = c(PC) + 1
				index "XR" = c(PC) + 1 if c(XR) NQ 0
				PC = effective address

BUS	Branch Unconditionally with same Program Register	014	XR	index "XR" = c(PC) + 1 if c(XR) NQ 0 PC = effective address
XEC	Execute	010	-	Operates as if c(x) were at location c(PC)

B. Conditional

Where conditions for branching are not met, these instructions are equivalent to a "no operation." Where conditions for branching are met, PR = c(PC) + 1 and PC = effective address.

Mnemonic	Name	Octal Code	Coded Modifiers	Branch if:
BLC	Branch on Logical Compare	620	LB, VL	c(Lby("LB")) EQ c("VL")
BLN	Branch on Logical No Compare	620	LB, VL	c(Lby("LB")) NQ c("VL")
BNM	Branch on Not Minus	610	A	c(ACCby("A")bi(0)) not all EQ 1 (minus)
BNP	Branch on Not Plus	610	A	c(ACCby("A")bi(0)0 not all EQ 0 (plus)
BNZ	Branch on Not Zero	600	A, ZD	c(ACCby("A")) not all EQ zero zero is +0 only or -0 only or all the same form of 0 depending on c(ZD)
BØM	Branch on Minus	610	A	c(ACCby("A") bi(0)) all EQ 1
BØP	Branch on Plus	610	A	c(ACCby("A") bi(0)) all EQ 0
BØZ	Branch on Zero	600	A, ZD	c(ACCby("A") all EQ zero see BNZ

mnemonic	Name	Octal Code	Coded Modifiers	Description
				Branch if:
BSF	Branch on Sense Unit Off	604	U	Unit "U" is off, see BSN
BSN	Branch on Sense Unit On	604	U	Unit "U" is on For certain units, BSN and BSF also turn the unit off. See N-17334.

Mnemonic	Name	Octal Code	Code Modifiers	Description	

VI. Branch Decrement

For all instructions in this class, when branch occurs, PR = C(PC) + 1 and PC = effective address. If branching does not occur, the instruction has the same effect as "no operation." Also, for these instructions, the IX modifier, in first level addressing only, is used NOT for address modification, but to specify the index used by the instruction.

A. Unconditional Branch and Modify

BAX	Branch and Add to Index	740	DF	Index "IX" = c(index "IX") + c(DF)	
BSX	Branch and Set Index	730	DF	Index "IX" = c(DF)	

B. Conditional Branch

For all instructions of this sub-class, +0 GR -0

				Branch If:	
BXE	Branch on Index Equal	720	DF	c(index "IX") EQ c(DF)	
BXH	Branch on Index High	700	DF	c(index "IX") GR c(DF)	
BXL	Branch on Index Low	710	DF	c(index "IX") LS c(DF)	

C. Conditional Branch and Modify

				Branch If:	
BMX	Branch on Minus Index	760	DF	c(index "IX") LS +0	index ("IX")= c(index "IX")+ c(DF)
BPX	Branch on Plus Index	750	DF	c(index "IX") GR -0	index ("IX")= c(index "IX")- c(DF)

Mnemonic	Name	Octal Code	Coded Modifiers	Description
VII.	Logical and Miscellaneous			
A.	Manipulate Bits of a Byte			
CØM	Complement Bits	410	MB, MK (MK = form b)	xby("MB") = c(xby("MB") OR c(MK)
INS	Insert Bits	404	MB, MK, VL	xby("MB") = (c(xby("MB")) AND -c(MK)) OR (c(VL) AND c(MK))
LDS	Load and Shift	434	MB, BS, R	ACC = c(xby("MB")bi("BS"))
STS	Store and Shift	440	MB, BS	xby("MB")bi("BS") = c(ACCby(7) bi(n_6)) n is in the range 1 through 6 according to the range between the first and second digits of c(BS).
B.	Skip Instructions			

CDA	Compare and Difference, Arithmetic	400	A-P, SG or MK (form a)	ACC = c(ACC) - C/m c(x) (note: Full mode used)
				IF C/m c(x) LS c(ACC), PC = c(PC) + 1
				IF C/m c(x) EQ c(ACC), PC = c(PC) + 2
				IF C/m c(x) GR c(ACC), PC = c(PC) + 3

CDL	Compare and Difference Logican	400	A-P or MK, (form a)	ACC = Cc(ACC) OR (Cc(ACC) OR Cc(x)) unless MK is set, in which case:
				ACC = c(B) AND (c(ACC) OR c(x)) OR -c(B) AND c(ACC), in any event:
				IF C/mc(x) NQ C/mc(ACC), PC = c(PC) + 1
				IF C/mc(x) EQ C/mc(ACC), PC = c(PC) + 2

CMA	Compare, Arithmetic	400	A-P, SG or MK (form a)	same as CDA except ACC = c(ACC)

Mnemonic	Name	Octal Code	Coded Modifiers	Description
CML	Compare, Logical	400	A-P or MK (form a)	same as CDL except ACC = c(ACC)
TST	Test	414	MB, BS, R	PC = c(PC) +1 + c(xBY("MB")bi("BS"))
C.	Boolean Operations			
ANA	AND to Accumulator	430	A, R or MK, R	C/mACC = (C/mc(ACC) AND C/mc(x))
ANS	AND to Storage	430	A or MK	C/mx = (C/mc(ACC) AND C/mc(x))
CTA	Connect to Accumulator	430	A, CN, R or MK, CN, R	C/mACC = (C/mc(ACC) "CN" C/mc(x)) where CN determines the truth table for the Boolean operation
CTS	Connect to Storage	430	A, CN or MK, CN	C/mx = (C/mc(ACC) "CN" C/mc(x)) see CTA
DEP	Deposit	430	A or MK	C/mx = C/mc(ACC)
EXT	Extract	430	A, R or MK, R	C/mACC = C/mc(x)
ØRA	OR to Accumulator	430	A, R or MK, R	ACC = c(ACC) OR C/mc(x)
ØRS	OR to Storage	430	A or MK	x = c(x) OR C/mc(ACC)
D.	Index Manipulation			
ATX	Add to Index	424	XR, H, R	index "XR" = c(index "XR") + c(xby(h)) "h" is 1,2,3, if H EQ L or 5,6,7 if H EQ R or blank
LDI	Load Double Index	444	MB, R	DISR = c(xby("MB")bi(3_6))

Mnemonic	Name	Octal Code	Coded Modifiers	Description
LDX	Load Index	420	XR,H,R	index "XR" = c(xby(h)) "h" is k,2,3 if H EQ L or 5,6,7 if H EQ R or blank.
SDI	Store Double Index Selection Register	450	MB	xby("MB")bi(3_6) = c(DISR) xby("MB")bi(1_2) = 0

E. Convert

The convert instructions set-up a loop within themselves, where the number of iterations is indicated in the "NC" modifier. After the first data reference "x" is determined from bytes 5, 6, and 7 of the most recent data word referenced, plus byte 0 of the ACC or B. On the last iteration: index "XR" = c(xby(5,6,7)). On each iteration:

CAB	Convert, by Addition from the B Register	460	XR, NC	ACC = 0 prior to first cycle
				ACC = c(xby(0_4)right justified)+ACC
				B = c(B by (1,2,3,4,5,6,7,0))

CRA	Convert by Replacement from the Accumulator	454	XR,NC,CP	ACC by(0_6) = c(ACC by(1_7) ACC by $(\overline{7})$ = c(xby(0)) unless CP = R, in which case ACC by(1_7) = c(ACC by(0_6)) ACC by $(\overline{0})$ = c(xby(0))

F. Shift and Cycle

For the shift and cycle instructions, the data word has a special format, bytes 5, 6, and 7 only, are used. The leftmost bit (30) is used as a sign for bits 41 to 47 and indicates direction (positive is right), and the seven rightmost bits (in ones' complement if a negative sign) are used for "bit count," i.e., modulo 128. In immediate addressing only, the IX, and D modifiers will change the effective "bit count" and possibly the sign.

CYA	Cycle Accumulator	024	M, R	ACC =M $\circledast$ c(ACC)
CYB	Cycle B Register	024	M, R	B =M $\circledast$ c(B)
CYC	Cycle Combined	024	M, R	ACCB =M $\circledast$ c(ACCB)
CYI	Cycle Inverted	024	M, R	ACCB =M $\circledast$ c(ACCB) using inverted ACC to B connections
NRA	Normalize Accumulator	034	M	ACC =M c(ACC $\circledast$ 2^n where n is the number of consecutive zero magnitude bits in ACCbi(1_n) x~~by(4_7)~~ = n

Mnemonic	Name	Octal Code	Coded Modifiers	Description
NRC	Normalize Combined	034	M	ACCB = M c(ACCB) * 2^n x ~~by (4_7)~~ = n see NRA for "n"
SFA	Shift Accumulator	020	M,RD,R	ACC = M c(ACC)*s where "s" = 2 c(xbi(30_47)), Modulo 128
SFB	Shift B Register	020	M,RD,R	B = M c(B) * s see SFA for "s"
SFC	Shift Combined	020	M,RD,R	ACCB = M c(ACCB) * s see SFA for "s"

G. Input-Output

Mnemonic	Name	Octal Code	Coded Modifiers	Description
BKF	Backspace File	040	S	equivalent to SLØU5
BKR	Backspace Record	040	S	equivalent to SLØU1
DRM	Start Drum	044	RD-WR, U,M,S	equivalent to SHØ
IØR	Start I/O Register Operation	044	RD-WR,U	equivalent to SHØ70
REW	Rewind	040	S	equivalent to SLØU2
SHØ	Start High Speed Operation	044	U,M,RD-WR,S	Starts operation on selected high-speed device.
SLØ	Start Low Speed Operation	040	U, RD-WR, TM, SI,CD,TA,S	Starts operation on selected low-speed device
TAP	Start Tape Operations	040	RD-WR,TM SI,CD,TA,S	equivalent to SLØ
WEF	Write end-of-file	040	S	equivelant to SLØU3
WET	Write end-of-tape	040	S	equivalent to SLØU4

H. Miscellaneous

Mnemonic	Name	Octal Code	Coded Modifiers	Description
HLT	Halt	000	-	Stops central computer operation
PER	Operate	004	U	Operates unit U. U codes listed in N-17334.

APPENDIX E

SUMMARY OF SCAMP CODING

The coding sheet consists of four fields;

1. Location field - columns 2 - 7 fixed length field used for "tagging" instructions.

2. Operation field - starting in column 8, operation code in columns 8 - 10, modifiers, if any, starting in column 11. Modifiers consisting of digits only must start in column 11. Modifiers are separated by commas. The first blank terminates the field.

3. Address field - starting one or more blanks after operation field, but not to the left of column 24. Modifiers follow the address and are preceeded by single commas. $ is used for "self reference".

4. Comments filed - starting one or more blanks after address field. Columns 67 - 72 reserved for numbering and identification. 73 - 80 is not used.

Program Organization Pseudos

BEG - first card of every program

END - last instruction of every program indicating termination of inputs. If address field contains an address, program will start at that location.

ORG (Origin) - contents of address field indicates the starting location for the instructions that follow.

Pseudos Pertaining to Tags

SYN (Synonym) - location field contains location label being defined. Address field indicates definition of label. This instruction is used to define address tags only.

SYM (Symbolic Modifier) - location field contains label to be used in place of some instruction modifier or address modifier or group of such modifiers. The address field contains the code for the modifiers represented by the label. This instruction permits the use of tags to replace modifiers which may be used repeatedly.

TØP - Used to initiate a program area wherein tags may be exclusively defined for that area. That is, a given label can be defined more than one time in a program if each occurrance is in a different area of the program and that area is bracketed with TØP and BØT instructions. These instructions may be nested.

BØT - terminates the zone initiated by a TØP.

Data Representation

Numbers, except where specifically required to be otherwise, are assumed to be decimal. An octal number may be coded by immediately following the number with a "prime."

"Prime" is written on the coding sheet as an inverted delta (∇) to avoid confusion with the digit "1."

Constants must be enclosed in parenthesis and the open parenthesis must be in column 8. Constants representing a dual word have the two half words separated by a double comma. Two byte numbers within a half word are separated by a single comma. One byte numbers within a half word are also separated by single commas and there must be four values represented for each half word. In general, constants will be right justified.

Floating point constants must contain a period, or decimal point, and may include, following the constant, the letter E followed by a number. The number following the E represents the power of 10 by which the constant is to be multiplied before conversion to machine format.

Fixed point constants are written in the same manner as floating point with the additional requirement that they are followed by the letter B and a number. The number following the letter B indicates the number of bit positions to the left of the end of the word or half word that the binary point is to be located. Single Hollerith constants are labelled as one byte constants and the Hollerith nature of the constant is indicated by preceding the character with an asterisk.

Integer constants may be indicated as decimal or octal with the mnemonics DEC and ØCT respectively. The number desired appears in the address field and if it is not to be a full word constant, the modifier R, L or D must precede, or follow, the number, indicating right half word, left half word, or a dual word constant. More than one constant can be designated with this pseudo by separating the constants, one from another with commas. The place in which these integer constants will be stored depends on which type constant is used, but in general they will occupy the next available word or half word. If it is desired to have a string of half word constants wherein they alternate left-half word, right half-word, etc., the R and L modifiers in the address field may be omitted and the letter A, indicating alternate, placed in the instruction modifier field.

Binary coded information for a card image on the printer may be set up with a single BCI instruction. Since blanks are regular Hollerith characters, the string of characters which are placed in the address field is terminated with a "$". A carriage control character may be indicated immediately following the BCI pseudo as follows: S (suppress space), A (single space), B (double space), C (triple space), and D (quadruple space). If a termination character other than $ is desired it may be indicated as the second modifier following the BCI pseudo. That is, a single modifier would always be carriage control and if there are two modifiers the first taken as the carriage control character and the second as the termination character. When a special termination character is used $ may be used as one of the characters to be converted. The necessary record mark will be placed in byte 7 of the last memory location used for the converted information.

A recurring set of constants may be designated with the DUP pseudo. This instruction uses two numbers in the address field, separated from each other by a comma. The first number represents the number of input cards immediately following the DUP instruction which are in effect to be duplicated. The second number indicates the number of times they are to be duplicated. The instructions or constants being duplicated should not contain a location tag as it will result in multiple definition. However, the DUP instruction may have a location tag.

A block of registers internal to a program may be set aside with the BLK pseudo. The address field contains the number of registers being reserved for this block. This pseudo instruction may have a location tag.

Symbolic Tags

Tags or location labels consist of a string of 1 - 6 alpha numeric characters of which at least one must be alphabetic. In addition, an address may consist of a string of tags or constants arithmetically combined. The tags, of course, must be defined within the program. To indicate the arithmetic operation, + is used for addition, - for subtraction, * for multiplication and / for division. In computing such addresses any fractions that are generated are dropped at each step of the computation.

APPENDIX F

I/O TYPEWRITER CODES

Character Shift Down	Shift Up	Octal Code
A	a	61
B	b	62
C	c	63
D	d	64
E	e	65
F	f	66
G	g	67
H	h	70
I	i	71
J	j	41
K	k	42
L	l	43
M	m	44
N	n	45
O	o	46
P	p	47
Q	q	50
R	r	51
S	s	22
T	t	23
U	u	24
V	v	25

Character Shift Down	Shift Up	Octal Code
W	w	26
X	x	27
Y	y	30
Z	z	31
0 (Input)	$	00
0 (Output)	$	12
1	]	01
2	[	02
3	ⓥ	03
4	⊙	04
5	&	05
6	<	06
7	>	07
8	?	10
9	Δ	11
/	'	21
)	#	74
(	%	34
-	:	40
+	√	60
.	=	73
,	;	33
*	±	54

Functions

Carriage Return	77
Shift Up	16
Space Bar	20
Tabulate	17
Shift Down	32

APPENDIX G

INTERRUPT SYSTEM DATA FOR CENTRAL PROCESSING AND I/O

Condition	DCR1 Bit	BSN Sense & Clear	ICR/ICS Bit
Selected Unit Not Ready			
Reader	28	675	36
Punch	30	677	36
1401	31	700	36
TA1	39	705	41
TA2	47	706	41
I/O Type	29	676	36
FIX Type	DCR2-23	-	-
Operation Complete			
Reader	24	671	37
Punch	26	673	37
1401	27	674	37
TA1	32	704	42
TA2	40	712	43
I/O Type (Output)	25	672	37
I/O Type (Input)	10	-	38
Selected Uint Busy			
All but Tapes	23	743	40
TA1	35	-	41
TA2	43	-	41
E.O.T.			
TA1	33	701	42
TA2	41	707	43
E.O.F.			
TA1	34	702	42
TA2	42	710	43
Selected Tape Rewinding			
TA1	37	703	41
TA2	45	711	41

Condition	DCR1 Bit	BSN Sense & Clear	ICR/ICS Bit
Illegal Motion			
TA1	36	-	41
TA2	44	-	41
File Protected Write or Wrong Density			
TA1	38	-	41
TA2	46	-	41
Overflow			
Left, Full	12	657	31
Floating Acc	12	657	32
Right	13	660	31
Floating - BReg	13	660	32
Precision	16	663	32
Underflow			
Floating - B Reg	15	662	32
Floating - Acc	14	661	32
Illegal Divide			
Left, Full	17	664	31
Right	18	665	31
Floating	18	665	32
Illegal Selection			
Memory Address	4	-	33
Drum Field	5	-	33
Index Register	6	-	33
High Speed I/O			
Complete	7	-	35
Conflict	11	-	39
Protected Field Write	21	667	34
Channel in Use	22	670	34