File Number 1401/1460-13
Form A24-3071-2

IBM Systems Reference Library

Special Feature Instructions
IBM 1401 Data Processing System
IBM 1460 Data Processing System

The special features described in this publication are available for the IBM 1401 and/or 1460 Data Processing Systems. Each feature is described and identified for the system to which it can be applied. These features offer additional flexibility in applications where special processing requirements exist.

Also included are the instructions for the special features on the IBM 1402, 1403, and 1009 when these units are used with the IBM 1401 or 1460 Data Processing System.

For a list of other publications and abstracts, see the *IBM 1401 and 1460 Bibliography,* Form A24-1495.

Preface

This publication is a portion of the reference text for the IBM 1401 and 1460 Data Processing Systems. The full set of manuals provides a detailed explanation of all the instructions used by the system to manipulate data. Detailed explanations of the instructions used with the required and available input/output units attached to the system are also included. The reader should be familiar with the *IBM 1401 System Summary,* Form A24-1401, or the *IBM 1460 System Summary,* Form A24-1496, and the various publications on programming material, such as Symbolic Programming System (SPS) and Autocoder.

The complete manual is divided functionally into these sections:

System Operation Reference Manual (A24-3067)

- Section A Introduction
- Section B System Operations
- Section C IBM 1406 Operations
- Section D IBM 1447 Operations
- Section E IBM 1402 and 1403 Operations

Tape Input/Output Instructions (A24-3069)

- Section F Tape Input/Output Operations

Disk Input/Output Instructions (A24-3070)

- Section G Disk Input/Output Operations

Miscellaneous Input/Output Instructions (A24-3068)

- Section H Miscellaneous Input/Output Operations

Special Feature Instructions (A24-3071)

- Section I Special Feature Operations

The sections are independent and do not have to be used in the order in which they appear. A System Reference Library can be compiled using those sections applicable to the user's machine configuration.

This publication is intended for programmers and systems personnel who have a general knowledge of the IBM 1401 or 1460 Data Processing Systems and who require a reference text for detailed information.

Other publications referenced here are, in most cases, prerequisites for a complete understanding of the material presented in this publication.

Major Revision, November 1964

This publication, Form A24-3071-2, is a major revision of A24-3071-1. It does not, however, obsolete the previous publication. The added material includes the instructions for the special features on the IBM 1402, 1403, and 1009 when these units are used with the IBM 1401 or 1460 Data Processing System.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

Address comments concerning the content of this publication to IBM Product Publications, Endicott, New York 13764.

Reprinted March 1966

Contents

Adapter, 51-Column Feed (1401, 1460)

This feature is required when the 51-column interchangeable-read-feed special feature is installed on the IBM 1402 Card Read-Punch, Model 1, for the 1401 system or the IBM 1402, Model 3, for the 1460 system.

The adapter furnishes the controlling circuitry necessary for the proper operation of the 51-column read-feed special feature.

For more detailed information on the 51-column interchangeable-read-feed special feature refer to IBM *1402 Card Read-Punch*, A24-3072.

Advanced Programming (1401)

The advanced-programming feature provides the 1401 system with greater program flexibility, by making address indexing, address storing, and record movement more automatic. The various parts of the advanced programming feature and their method of operation are described in this section.

Indexing

The indexing portion of the advanced-programming feature provides three 3-position index locations (registers) that can be used to modify addresses automatically. These three index registers are part of core storage and can be used as normal storage positions when not being used as index-register locations. The assigned core-storage addresses and index register numbers are:

Index-Register No.	*Core-Storage Positions*
1	087 - 089
2	092 - 094
3	097 - 099

The index factor can be placed in the index register by normal programming (add or move operations, for example) and the factor can be changed (add operations normally). In these instances, a word mark should be initially set in the high-order position of the index register before inserting or changing the index factor.

Both the A-address and/or the B-address can be modified by the index factor in any one of the three index registers; however, only core-storage addresses can be modified.

The A- and/or B-address specifies which index register will be used by a combination of A- and B-bits in the tens position of the address. The bit combinations and the registers they specify are:

Bit Combination	*Index Register No.*	*Zone Punch*
A-bit, No B-bit	1	Zero
B-bit, No A-bit	2	Eleven
A-bit, B-bit	3	Twelve

When the tens position of an A- and/or B-address contains one of these zone-bit combinations, the address is referred to as being *tagged.*

	Positions ↓	Tagged ↓
OP	AAA	BBB

The modification of the A- and/or B-address occurs in their respective address registers. For instance, if the A-address is indexed, the indexing occurs in the A-address register. This means the original instruction in storage is in no way changed or modified.

When a program is written using the symbolic programming system, indexing is indicated by writing a 1, 2, and 3 in the index column of the symbolic coding sheet (cc27 for (A) operand—cc38 for (B) operand).

Digit	Result
1	Index operand by contents of 087-089
2	Index operand by contents of 092-094
3	Index operand by contents of 097-099

The (A) and (B) operands can be symbolic of actual addresses. In the processing of instructions:

1. The A-address and B-address are analyzed for indexing as they are moved into the address registers.
2. The contents of the proper index location (indexing factor) are added to the contents of the address register and develops the effective address there, when indexing is indicated.
3. Three or four additional cycles are required for each address indexed.

Increasing an Address

To increase a core-storage address using the indexing feature, the contents of the index location are added to the selected address register. Figure I-1 illustrates various methods of address modification using the index locations.

		INSTRUCTION IN STORAGE	INDEX LOCATION 1	2	3	EFFECTIVE INSTRUCTION
1. INDEX THE B-ADDRESS	BEFORE	M 080 1A7	010	025	050	
	AFTER	M 080 1A7	010	025	050	M 080 167
2. INDEX A- AND B-ADDRESS	BEFORE	M 0S0 1J7	010	025	050	
	AFTER	M 0S0 1J7	010	025	050	M 030 142
3. INDEX A- AND B-ADDRESS	BEFORE	M JB0 8C0	010	025	050	
	AFTER	M JB0 8C0	010	025	050	M J70 880

Figure I-1. Indexing

Decreasing an Address

To decrease an address, the 16,000's complement of the amount to be subtracted from the address must be stored in the index location.

Example: Decreasing

Required Decrease a B-address by 10
Indexing factor 16000 − 10 (15990)
(Complement)

The 15990 converts to the 3-digit factor I9? (Figure I-2).

Using the modulus 16 rules, the arithmetic overflow adds an A-bit in the hundreds position (both the hundreds and units positions already contain A- and B-bits, the combination of which indicates a 15000-15999 block address). The addition of the A-bit increases the value of the zone bits to 16 which, according to modulus 16, has an address value of 0 (000-999 block address). Therefore, the new address is 927. With the indexing feature, even though there was an overflow, the arithmetic-overflow indicator is not turned on.

Store Address Register

The store-address-register portion of the advanced-programming special feature makes it possible to store the contents of the A- and B-address registers. Thus, the A- and B-addresses of program instructions can be modified directly in cases where variable-length records are being processed. This facility also makes it easier to re-enter the main program from a subroutine. Because the address of the next instruction in sequence can be retained, program re-entry is simplified.

A subroutine is a set of program instructions that are executed, if a particuler condition arises during the main routine. For example, if an unequal compare occurs during processing, the program branches to a subroutine in which a special set of instructions handles the condition.

Each time a subroutine is used, some method must be employed to link it with the main program. The function of the STORE A-ADDRESS REGISTER and STORE B-ADDRESS REGISTER instructions is to establish subroutine linkage so that upon leaving the sequence of the main program it is possible to execute the steps of the subroutine, and return to the main program where the sequence was interrupted.

	INSTRUCTION IN STORAGE	INDEX LOCATION 1	2	3	EFFECTIVE INSTRUCTION
BEFORE	L 123 9T7	I9?			
AFTER	L 123 9T7	I9?			L 123 927

937 + I9? = 927 (with overflow)
I = (AB9)
? = (AB0)

Figure I-2. Converting Address

Store A-Address Register

Instruction Format.

Mnemonic	*Op Code*	*A-address*
SAR	Q	xxx

Function. This instruction stores the contents of the A-address register from the previous operation, in the 3-position field that has its units position defined by the A-address of the STORE A-ADDRESS REGISTER instruction.

Word Marks. Word marks are not affected.

Timing. $T = .0115\ (L_I + 5)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	A-3	Ap

Example. Store the contents of the A-address register in area labeled AADRG (0625), Figure I-3.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
0,1,0			S A R	A A D R G								

Autocoder

Label	Operation	OPERAND
	SAR	AADRG

Assembled Instruction: Q 625

Figure I-3. Store Contents of A-Address Register

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
0,1,0			B	MULTRU								
0,2,0			S B R	LAST	+	3						
0,3,0												
0,4,0												
0,5,0	LAST		B	0000								

Autocoder

Label	Operation	OPERAND
	BU	MULTRU
	SBR	LAST+3
LAST	B	0

Assembled Instruction: B 495
H (address machine-assigned)
B (address in main routine following the B 495 instruction)

Figure I-4. Store Contents of B-Address Register (One Field)

Store B-Address Register (One Field)

Instruction Format.

Mnemonic	*Op Code*	*A-address*
SBR	H	xxx

Function. This instruction stores the contents of the B-address register resulting from the previous operation, in the 3-position field that has its units position defined by the A-address of the STORE B-ADDRESS REGISTER instruction.

Word Marks. Word marks are not affected.

Timing. T = .0115 (L_I + 4 or 5*) ms.

* Plus 4 or 5 depends on the presence or absence of zone bits in the units position of the address being stored.

Note. When indexing is installed in the IBM 1401, the functioning of all branch instructions is altered to simplify subroutine linkage. With these alterations, each time a branch occurs as a result of one of these instructions, the address of the next sequential instruction in the main routine is inserted in the B-address register.

Although the subroutine may be entered from many distinct points in the main program, this use of the SBR operation makes the subroutine linkage complete.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	A-3	Bp

Example. The main routine branches to a multiply subroutine labeled MULTRU (0495). This example shows the last step in the main routine and the first and last steps of the multiply routine, and illustrates subroutine linkage (Figure I-4). The last instruction (labeled LAST) plus three will contain the address of the next instruction in the main routine.

Store B-Address Register (Two Fields)

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*
SBR	H	xxx	xxx

Function. This instruction stores the present contents of the B-address register in the 3-position field that has its units position defined by the A-address of the STORE B-ADDRESS REGISTER instruction. The B-address register contains the number specified by the B-address portion of the STORE B-ADDRESS REGISTER instruction.

Word Marks. Word marks are not affected.

Timing. T = .0115 (L_I + 4 or 5*) ms.

* Plus 4 or 5 depends on the presence or absence of zone bits in the units position of the address being stored.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	A-3	Bp

Example. Store contents (456) of the B-address register in the area labeled BADRG (123), Figure I-5.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
0,1,0			S B R	B A D R G				4 5 6				

Autocoder

Label	Operation	OPERAND
	SBR	BADRG,456

Assembled Instruction: H 123 456

Figure I-5. Store Contents of B-Address Register (Two Fields)

Move Record (1401; Standard on 1460)

The move record portion of the advanced-programming special feature provides a special MOVE instruction that makes it possible to move a complete record from one storage area to another, without regard to word marks within the record. This instruction is especially desirable in magnetic tape operations.

This feature is standard on the 1460 system.

Move Characters to Record or Group Mark

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*
SPS MCM	P	xxx	xxx
A MRCM			

Function. This operation code makes it possible to move an entire record from one core-storage area to another, regardless of the presence of word marks in either field. The A- and B-addresses specify the high-order position of the respective areas. Transmission starts from the high-order addresses, and continues until a record mark (A82 bits) or a group-mark with a word-mark (WMBA8421 bits) is sensed in the A-field. The record mark or group mark transfers to the B-field.

Word Marks. Word marks within the area do not affect the MOVE CHARACTERS TO RECORD OR GROUP MARK operation. Any word marks in the B-field remain unchanged. A-field word marks are not transmitted to the B-field.

Timing. $T = N^* (L_I + 1 + 2L_A)$ ms.

* N = .0115 on 1401 system;
* N = .006 on 1460 system

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	$A + L_A$ (The length of the A-field includes the group-mark with a word-mark or record mark)	$B + L_A$

Example. Move the tape record that has its high-order character in the location labeled TARCIN (0679) to another area of core storage beginning at the label WTAREC (0985), Figure I-6).

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
3 5	6 7	8 13	14 16	17 23			27	28 34			36	39
0 1 0			MCM	TARCIN				WTAREC				

Autocoder

Label	Operation	OPERAND
6 15	16 20	21 25 30 35 40 45 50
	MRCM	TARCIN, WTAREC

Assembled Instruction: P 679 985

Figure I-6. Move Characters to Record or Group Mark

Binary Transfer (1460)

This special feature makes it possible to process cards and magnetic-tape data recorded in column-binary form. This provides compatibility between the IBM 1460 and IBM scientific data processing systems, such as the IBM 704, 709, and 7090.

Auxiliary operations that can be performed for large-scale binary systems are:

1. Card-to-tape.
2. Tape-to-card
3. Tape-to-tape
4. Card-to-card
5. Card-to-printer
6. Tape-to-printer
7. Tape sorting, editing and merging
8. File maintenance

This special feature is the same as the column-binary special feature on the 1401, with one exception. The binary transfer special feature does not have the BRANCH IF BIT EQUAL instruction as does the column-binary feature. For more detailed information on this feature, refer to the *Column Binary* section of this publication. Replace 1401 timing (.0115) with 1460 timing (.006) in all instructions.

Bit Test (1401, 1460)

This special feature is available on both the IBM 1401 and 1460 systems. It is a BRANCH instruction that causes the character located at the B-address to be compared, bit by bit, with the d-character. If any bit in the character located at the B-address matches any bit in the d-character, the program branches to the specified I-address (WM and C-bits are not compared).

For more detailed information on the bit-test special feature, refer to the BRANCH IF BIT EQUAL instruction in the *Column Binary* section of this publication. Replace 1401 timing (.0115) with .006 for 1460 timing.

Column Binary (1401)

This feature makes it possible for the IBM 1401 Data Processing System to process column-binary-coded cards and magnetic tapes used with IBM scientific data processing systems such as the IBM 704, IBM 709, and IBM 7090.

The reading, writing, and logic operational facilities of the 1401 can be used to process the binary-coded data.

The operation codes and instructions described in this section are used whenever:

1. The information to be read or written is in binary cards or binary-coded tapes.
2. There are invalid multiple punches in cards containing the standard IBM card code. This means that cards, coded with several punches in one column, that were designed for other machines can be entered by using this feature.

Read Column Binary

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SPS R *A* RCB	1	C

Function. This instruction causes the card at the read station to be read into the 1401 in the binary mode. During the reading of a card, the read cycle is divided into two parts, and three different areas in storage receive the data. Card cycle 9 through 4 time uses the normal read addresses 001-080, and column-binary-read area addresses 501-580. The other portion of the card cycle (3-12 time) uses addresses 001 through 080, and 401 through 480. Note that storage locations 001-080 are common read-in locations for both halves of the read cycle.

At the completion of this read operation a BCD coded image of the card is stored in addresses 001 through 080, just as in normal card reading. The portion of the card that contains column-binary information appears as *hash* in the corresponding addresses 001-080, and the portion of the card that contained alphamerical characters is stored in BCD code in addresses 001 through 080. Storage addresses 401-480 and 501-580 contain the true card image. In

	BCD CODE	PUNCHES IN CARD COLUMN
Storage	C	
	B	12
Addresses	A	11
	8	0
401-480	4	1
	2	2
	1	3
Storage	C	
	B	4
Addresses	A	5
	8	6
501-580	4	7
	2	8
	1	9
Storage	C	12
	B	11
Addresses	A	0
	8	1
	4	2
001-080	2	3
	1	4
	(BCD equivalent of the card punches if alphamerical, hash if true binary.)	5
		6
		7
		8
		9

Figure I-7. BCD Code and Column Binary Punches

these areas all alphamerical characters and all column-binary information appear as illustrated in Figure I-7.

If, for example, the following information is recorded in a binary card and appears in core storage:

Card-column 1 contains an IBM card-code H which is represented by a 12-punch and an 8-punch.

Storage Locations	*Contains*
001	BA8
401	B
501	2

Card-column 2 is part of a binary field and contains punches in rows 12, 0, 1, 3, 4, 5, 6, 7, and 9.

Storage Locations	*Contains*
002	hash
402	CB841
502	BA841

Word Marks. Word marks are not affected.

Timing. $T = .0115\ (L_I + 1)$ ms + I/O.

Note. The READ COLUMN BINARY instruction (1C) cannot be combined with any other operation.

Read checking of input data is unchanged, except for the validity check, which is not performed because all characters read are considered valid.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	*dbb*	481

Example. Read the card at the IBM 1402 read station in the column-binary mode (Figure I-8).

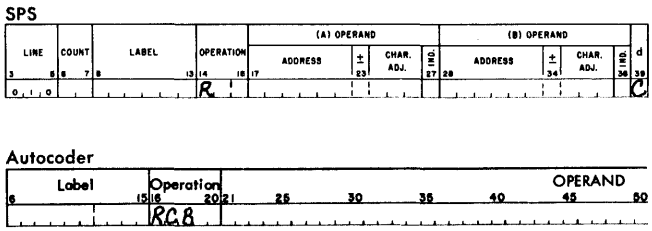


Figure I-8. Read Column Binary

Read Column Binary and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS R *A* RCB	1	xxx	C

Function. This is the same as READ COLUMN BINARY, except that the next instruction is at the I-address.

Word Marks. Word marks are not affected.

Timing. $T = .0115\ (L_I + 1)$ ms + I/O.

Note. The READ COLUMN BINARY AND BRANCH instruction (1xxxC) cannot be combined with any other operation.

Read checking of input data is unchanged, except for the validity check, which is not performed because all characters read are considered valid.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	481

Example. Read the card at the read station in column binary mode, and branch to BININ (0922) for the next instruction (Figure I-9).

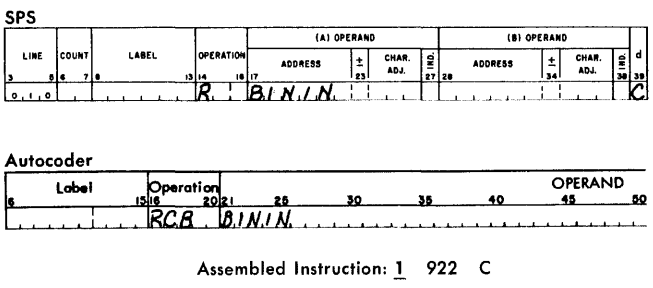


Figure I-9. Read Column Binary and Branch

Punch Column Binary

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SPS P *A* PCB	4	C

Function. This instruction, executed in two parts, requires that the information be stored in two different areas. Information that is to be punched in rows 12-3 (card columns 1-80) is stored in locations 401-480. Rows 4-9 of the card (columns 1-80) are punched from storage locations 501-580.

Using the same data shown in the READ COLUMN BINARY example, the card is punched:

Storage Locations	*BCD*	*Punches*
401	B	12
501	2	8

This combination causes the H to punch in card column 1.

Card column 2 is punched in *rows* 12, 0, 1, 3, 4, 5, 6, 7, and 9 as transferred from:

Storage Locations	*BCD*	*Punches*
402	CB841	12, 0, 1, 3
502	BA841	4, 5, 6, 7, 9

Word Marks. Word marks are not affected.

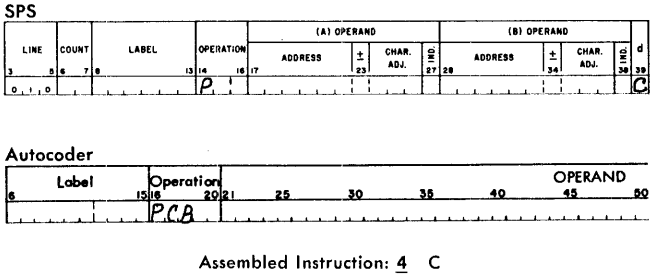
Timing. $T = .0115\ (L_I + 1)$ ms + I/O.

Note. The PUNCH COLUMN BINARY instruction (4C) cannot be combined with any other operation. The punch checking of output data is unchanged.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	*dbb*	181

Example. Punch the card at the punch station in the column-binary mode (Figure I-10).



Assembled Instruction: 4 C

Figure I-10. Punch Column Binary

Punch Column Binary and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS P *A* PCB	4	xxx	C

Function. This is the same as the PUNCH COLUMN BINARY instruction, except that the next instruction is taken from the I-address.

Word Marks. Word marks are not affected.

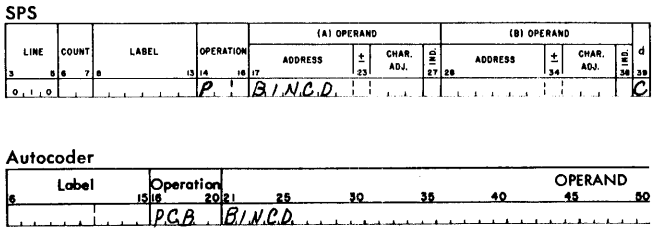
Timing. $T = .0115\ (L_I + 1)$ ms + I/O.

Note. The PUNCH COLUMN BINARY AND BRANCH instruction (4xxxC) cannot be combined with any other operation. The punch checking of output data is unchanged.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	181

Example. Punch a card in the column-binary mode, and branch to BINCD (0986) for the next instruction (Figure I-11).



Assembled Instruction: 4 986 C

Figure I-11. Punch Column Binary and Branch

Branch if Bit Equal

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*B-address*	*d-character*
BBE	W	xxx	xxx	x

Function. The d-character can contain any character or any combination of bits (BA 8421) that can exist in a single position of the 1401 core storage. If the character at the B-address contains any bit that matches any bit in the d-character, the program branches to the I-address. Otherwise, the program continues normally.

Word Marks. Word marks are not affected.

Timing. T = .0115 ($L_I + 2$) ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	B-1

Example. Examine the storage location labeled UNPOS (0759) for a match in the d-character bit configuration. The d-character is a 9 (8- and 1-bits). Therefore, if the character contains either an 8- or 1-bit, the program branches to BITEST (0985), Figure I-12.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND	d
			BBE	BITEST				UNPOS				9

Autocoder

Label	Operation	OPERAND
	BBE	BITEST,UNPOS,9

Assembled Instruction: W 985 759 9

Figure I-12. Branch if Bit Equal

Binary Tape Instructions

Column-binary information should be recorded on magnetic tape in its logical order. To do this it is necessary to arrange the data from storage locations 401-480 and 501-580 in the following sequence in a tape-write area:

> Address 401 followed by 501, followed by 402, 502, etc., until the entire area is so arranged.

This puts the 12-3 data next to the 4-9 data from the same card column, in the proper sequence for writing on tape.

This arranging can be done automatically by a MOVE AND BINARY DECODE instruction.

Conversely, data read from a tape unit can be arranged as a card image with 12-3 punches in the 401-480 area and 9-4 punches in the 501-580 area. The MOVE BINARY CODE operation performs this function.

Column-binary information must be written on magnetic tape in the odd-parity mode. This means that an odd number of bits must be recorded in each position of the tape record. The WRITE BINARY TAPE and READ BINARY TAPE instructions cause the data to be recorded in this manner.

Move and Binary Decode

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MCW	M	xxx	xxx	A
A MBD				

Function. This instruction arranges data in the correct order for tape writing. The A-address is usually 572 or 580, depending on whether the card has 72 or 80 columns of binary data. It specifies the units or low-order position of the record. The B-address specifies the low-order position of the area in core storage from which the record is to be written on tape by a WRITE BINARY TAPE instruction. The d-character specifies that this is a move and binary decode operation.

At the completion of this operation, the tape-write area (B-address) contains the data from both the 401-480 and 501-580 areas in this sequence:

401, 501, 402, 502, 403, etc.

Word Marks. A word mark must be preset in the 401-480 area to signify the high-order character of the record (normally in location 401). Any word mark encountered stops the transfer to the tape-write area.

Timing. T = .0115 ($L_I + 1 + 2L_B$) ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	Address of the 400 area preset word mark + 99.	B-L_B

Example. Write the data in 401-480 (labeled CLB14A) and 501-580 (labeled CLB15A) areas in the tape-write area labeled BITPAR (2080), Figure I-13.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND	d
			MCW	CLB15A				BITPAR				A

Autocoder

Label	Operation	OPERAND
	MBD	CLB15A,BITPAR

Assembled Instruction: M 580 180 A

Figure I-13. Move and Binary Decode

Move and Binary Code

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MCW *A* MBC	M	xxx	xxx	B

Function. Information read from magnetic tape is arranged into a coded card image in binary form, when this instruction is used. Data to be punched in rows 12-3 is transferred to core-storage-area locations 401-480, and data to be punched in rows 4-9 is stored in the 501-580 area for punching. The A-address specifies the units position of the tape read-in area, and the B-address is usually 572 or 580, depending on the number of columns to be punched. The d-character (B) specifies a move-binary-code operation.

Word Marks. A word mark must be preset in the high-order position of the B-field (normally 401) to stop the operation.

A word mark in the high-order position of the A-field can be set to stop the operation after the following B-cycle if desired.

Timing. $T = .0115\ (L_I + 1 + 2L_B)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	A-L_A	The address of the preset word mark in the 400 area + 99.

Example. Move the data from the tape read-in area, labeled BITPAR (2080), to the column-binary punch area CLB14A (0401-0480) and CLB15A (0501-0580), and store it in the proper sequence for punching (Figure I-14).

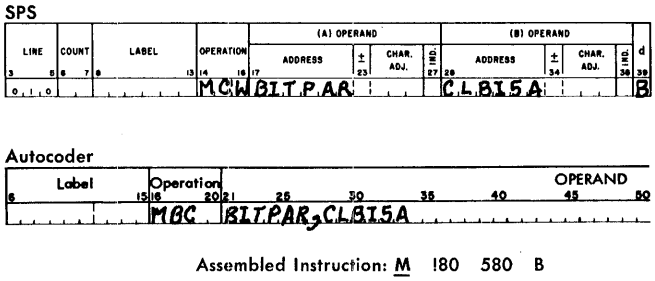


Figure I-14. Move and Binary Code

Write Binary Tape

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MU *A* WTB	M	%Bx	xxx	W

Function. This instruction writes a tape record in the odd-parity mode. The A-address specifies the tape unit to be selected, and signals that this is a column-binary-tape operation. The B-address specifies the high-order position of the tape record in core storage. The d-character indicates a tape-write operation. The sensing of a group-mark with a word-mark in core storage stops transmission from the system to the tape unit.

Word Marks. Word marks are not affected.

Timing. $T = .0115\ (L_I + 1)$ ms + T_M.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%2x	GM-WM + 1

Example. Write a tape record in the binary mode on the tape unit labeled 4, with the data stored in the area labeled BTPOUT (2001) and ending at the group-mark with a word-mark sensed in core storage (Figure I-15).

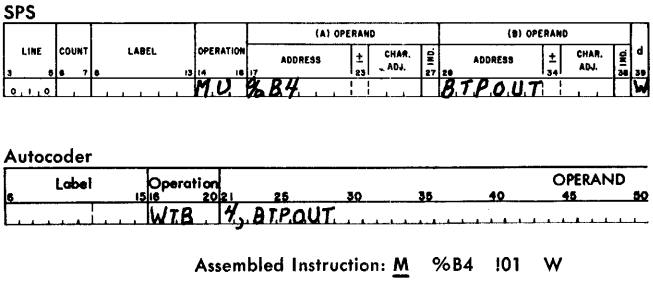


Figure I-15. Write Binary Tape

Read Binary Tape

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MU *A* RTB	M	%Bx	xxx	R

Function. A tape record written in binary form is read into core storage, beginning at the location specified by the B-address and ending at an inter-record gap between tape records or a group-mark with a word-mark in core storage. The A-address indicates the tape unit selected, and signals the column-binary tape operation. The d-character (R) specifies a read operation.

Word Marks. Word marks are not affected.

Timing. $T = .0115\ (L_I + 1)$ ms $+ T_M$.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%2x	Message Length + 1

Example. Read the binary tape record from the tape unit labeled 5 into the area of core storage labeled BTPIN (2080) and ending at the group-mark with a word-mark sensed in core storage or at the first inter-record gap encountered in the tape record (Figure I-16).

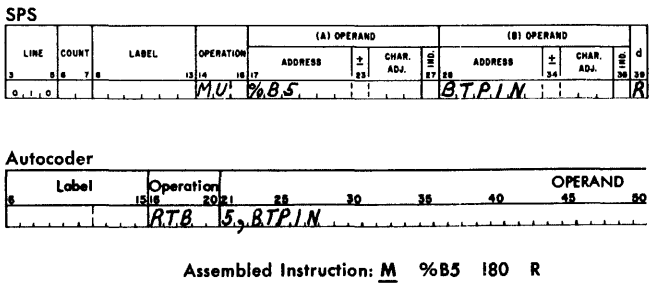


Figure I-16. Read Binary Tape

Compressed Tape (1401, 1460)

This feature makes it possible for the IBM 1401 and 1460 Data Processing Systems to read compressed tape prepared by the IBM 7070 Data Processing System, and to expand it within core storage for processing by the stored program.

The 7070 writes a compressed-tape record under control of the WRITE WITH ZERO ELIMINATION (TWZ or TWC) instruction. By using this 7070 instruction, as many as five high-order zeros in each numerical word in storage can be eliminated while the data is recorded on magnetic tape, thus conserving tape capacity and read-write time.

The READ COMPRESSED TAPE and MOVE AND INSERT ZEROS operation codes are incorporated in the 1401 to enable it to process compressed-tape records.

Read Compressed Tape

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
MU	M	%Cx	xxx	R

Function. This operation code causes an entire tape record to be read into core storage beginning at the B-address and ending with the detection of an inter-record gap (IRG) in the tape record. A group-mark without a word-mark is placed in storage at the right of the last data character transmitted when the IRG is sensed. At the end of the operation, the B-address register contains the address of the inserted group mark.

Mode changes (alpha to numerical and vice versa) are made automatically, and are controlled by the presence of mode change characters (Δ) in the tape record.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms $+ T_M$.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%3x	Address of the inserted group mark

Example. Read a tape record from tape unit 2 (labeled 2) in the area of core storage labeled TPAREA (0498), Figure I-17.

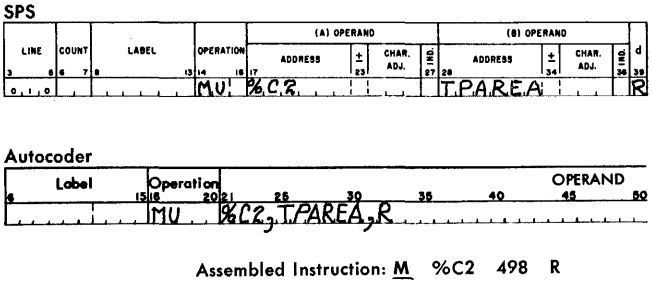


Figure I-17. Read Compressed Tape

Move and Insert Zeros

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*
MIZ	X	xxx	xxx

Function. The MOVE AND INSERT ZEROS instruction moves the compressed tape data that was read into core storage by a READ COMPRESSED TAPE instruction to another storage area, and expands each field to fill the storage locations allotted to it by the field-defining word marks. The A-address specifies the units

position of the compressed tape record in core storage. (To obtain the A-address, execute a STORE B-ADDRESS REGISTER instruction immediately following the execution of the READ COMPRESSED TAPE instruction. This stores the address that contains the group mark (‡) that indicated the end of the compressed tape record.) The B-address of the MOVE AND INSERT ZEROS instruction specifies the units position of the expanded area. The data moves from the compressed area to the expanded area, and zeros are inserted into the high-order positions of the expanded-area fields.

Word Marks. Word marks must be preset in the expanded area to indicate the high-order position of each field. A group-mark, with a word-mark that has also been preset by the program, must appear at the position immediately to the left of the high-order storage location of the A-field. It is this group mark that signals the end of the MOVE AND INSERT ZEROS operation.

Timing. $T = N\ (L_I + 1 + 2\ \Sigma\ L_A\ \Sigma\ L_Z)$ ms.

Σ = Number of fields included in an operation.

Note. When the IBM 7070 writes a tape record, it writes each word on tape in either the alphabetic or numerical mode. Each time the mode changes from alphabetic to numerical or vice versa, a mode change character, delta (Δ), is automatically written on tape. Each time a delta is read into core storage by the READ COMPRESSED TAPE instruction, it changes the setting of an internal switch to either the alphabetic or the numerical indication, corresponding to the mode. Thus, at the completion of the operation, the mode change switch indicates the mode setting of the last tape character read.

In the expand operation, the setting of the internal mode switch determines the method of operation. The machine works on the MOVE AND INSERT ZEROS operation from right to left, moving the data, field by field, from the compressed area to the expanded area. If the compressed-area field is alphabetic, it is moved, intact, to the expanded-area field, as defined by the preset word marks. To ensure proper operation, the expanded alphabetic fields should be equal in length to the alphabetic fields read from tape. If the data are numerical, they are moved digit-by-digit, low order to high order, until a zone bit (indicating sign position) or Δ (delta) is encountered. If any high-order positions in the expanded field are unfilled, zeros are inserted until a word mark is sensed. During this operation, the detection of a Δ in the compressed area changes the setting of the mode switch, and the mode of operation changes from alphabetic to numerical, or vice-versa.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	Address of preset group-mark with a word-mark −1 at immediate left of tape read-in area.	At last word-mark in B-field minus one.

Example. A 4-word compressed-tape record is prepared by the IBM 7070:

Field	*Mode*	*IBM 7070 Storage Words*
Part name	alpha	two words
Part number	numerical	one word, always plus
Unit Cost	numerical	one word, always plus

The part number can be from two to seven digits in length. The unit cost can be from three to six digits. The compressed-tape record, written by the 7070 for a specific part, looks like this:

EXTbbSHANKΔ0475C1154E

The letter C is a plus sign over units digit 3. The E is a plus sign over units digit 5. The mode switch is set to alpha mode during the compressed tape operation instruction time. Therefore, it was changed to the numerical mode by the Δ. It is necessary to perform the expand operation before the next READ COMPRESSED TAPE instruction.

Figure I-18 shows the three program steps that read and expand the compressed-tape record for this example. A group-mark with a word-mark has been preset in position 0699.

The READ COMPRESSED TAPE instruction reads into core-storage locations 0700-0721:

EXTbbSHANKΔ0475C1154E‡

After the operation, the B-address register contains the address of the group mark (0721). The STORE B-ADDRESS REGISTER instruction modifies the MOVE AND INSERT ZEROS instruction so that the A-address contains 721:

X	000	724	before
X	721	724	after

The maximum size of the compressed tape record is 24 positions (to accommodate a 10-character part name, a 7-digit part number, a 6-digit unit cost, and the mode-change character). Thus, the expanded

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
3 5	6 7	8 13	14 16	17 23			27	28 34			38	39
0,1,0			MU	%C2				0700				R
0,2,0			SBR	EXPAND+		3						
0,3,0		EXPAND	MIZ	0000				0724				

Autocoder

Label	Operation	OPERAND
6 15	16 20	21 25 30 35 40 45 50
	MU	%C2,700,R
	SBR	EXPAND+3
EXPAND	MIZ	0,724

Assembled Instruction:

M	%C2	700 R	located in storage positions 433-440
H	448		located in storage positions 441-444
X	000	724	located in storage positions 445-451

Figure I-18. Move and Insert Zeros

area is defined as locations 700-724 (the 25th position is for the group mark, ‡). Word marks are preset in positions 700, 711, and 718.

The MOVE AND INSERT ZEROS instruction first tests the mode switch, and then moves the unit cost field and the group mark (1154E ‡) from the compressed field locations, 716-721, to the expanded field locations, 719-724. The detection of the zone bits in the letter C of the part number indicates the units position of the next field. Because no mode-change character is detected, the mode switch continues to indicate numerical. A zero is inserted in position 718. The preset word mark in that position stops the insertion of additional zeros in the unit-cost expanded field. In a similar way, the 0475C part number moves from positions 711-715 to positions 713-717, and zeros are inserted in positions 712 and 711, halted by the word mark in location 711. The Δ in position 710 indicates the units position of the next field (part name).

The Δ changes the setting of the mode-change switch from numerical to alphabetic. In the alphabetic mode, characters are moved without insertion of zeros.

The expanded area in core storage after the operation looks like this:

EXTbbSHANKΔ000475C01154E‡

NOTE: To conserve storage, the compressed area overlaps with the expanded area in this example.

Console (1447, Model 2 or 4) Attachment (1460)

This special feature provides for the attachment of a console input-output printer to the 1460 system.

For more detailed information on the 1447, refer to IBM *1447 Console*, Form A24-3031.

Console Inquiry Station Adapter (1401)

This feature provides the additional circuitry to attach an IBM 1407 Console Inquiry Station to the 1401. For more detailed information on the 1407, refer to IBM *1407 Console Inquiry Station*, Form A24-3084.

Direct Data Channel (1401, 1460)

This feature provides for the attachment of the following data processing systems through their serial I/O adapters: 1401-1401; 1401-1440; 1401-1460; 1440-1440; 1440-1460; 1460-1460.

With the direct-data-channel special feature, the two processing systems are cable-connected through the serial I/O adapter feature on each system. When the direct data channel feature is in use, no other input-output unit can use the serial I/O adapter feature on either system.

The cable length between the two systems can be any length up to a maximum of 100 feet.

Data transmission takes place serially by character and parallel by bit (WM BA8421 plus a parity bit). Because word marks can be transmitted, the 1401 systems must have the engineering change installed that adds the WM bit line.

The type of data transmission operation that can be performed is varied and at the discretion of the customer. Depending on the written program, both systems can send and receive data, or one system can send data only while the other system can receive data only. To permit this flexibility, the direct-data-channel feature makes use of three types of instructions:

1. SIGNAL CONTROL instructions
2. BRANCH instructions
3. MOVE and LOAD instructions.

Signal Control Instructions

The SIGNAL CONTROL instructions are used by one processing system to:

1. inform the other processing system that it wants to perform a particular operation, or
2. actually perform a particular function in the other system.

NOTE: The Autocoder mnemonic for SELECT STACKER instruction (SS) can be used with the C, D, and E d-characters for direct-data-channel instructions.

The diagnostic phase of Autocoder will flag the instruction as an error, because it assumes that the d-character represents a stacker. However, the error flag can be ignored because the assembled instruction in the output object program will contain the intended (C, D, and E) d-characters.

Read Request

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	C

Function. This instruction informs the other system that the system initiating this instruction wants to read (receive) data from the other system. This condition is tested for in the other system with its B (III) 3 instruction.

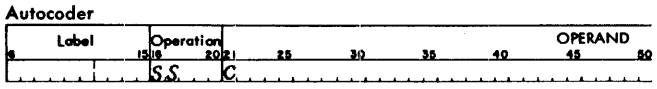
Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Example. The system executing this instruction signals the other system that it wants to receive data from the other system (Figure I-19).



Assembled Instruction: K C

Figure I-19. Read Request

Write Request

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	D

Function. This instruction informs the other system that the system initiating this instruction wants to send (write) data to the other system. This condition is tested for in the other system with its B (III) 4 instruction.

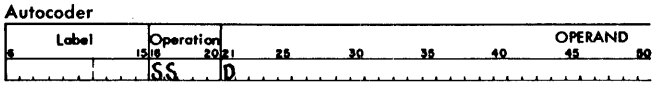
Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Example. The system executing this instruction signals the other system that it wants to send data to the other system (Figure I-20).



Assembled Instruction: K D

Figure I-20. Write Request

Reset

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	E

Function. This instruction originates a signal that resets the end of transmission circuitry in the other system. This reset operation cannot be tested for in the other system.

This instruction must be given prior to each execution of a read or write data instruction. It must be executed by the 1401 (if one of the machines is a 1401) every time the other machine is started or restarted. This is the only way to reset the end-of-transmission circuitry in the other machine, because the 1401 start-reset key does not include such a function.

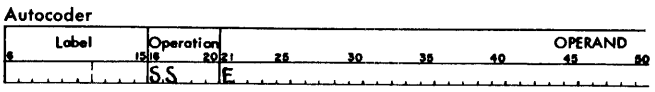
Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Example. Reset the end of transmission circuitry in the other system (Figure I-21).



Assembled Instruction: K E

Figure I-21. Reset

Branch Instructions (Direct Data Channel)

Branch if Indicator On

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
BIN	B	III	n

Function. This instruction and its associated d-characters are used by the system initiating these instructions to check for various conditions on the other system. When a tested condition is present, the program branches to the previously written subroutine. The BRANCH IF INDICATOR ON instruction d-characters, and the tests they perform, are shown in Figure I-22.

Word Marks. Word marks are not affected.

Timing.

No Branch:

$T = N (L_I + 1)$ ms.

Branch (without indexing):

$T = N (L_I + 1)$ ms.

Branch (with indexing):

$T = N (L_I + 2)$ ms.

Condition in System Initiating Line/Signal (System A)	Line/Signal		System B, Testing Conditions in System A with Branch-If-Indicator-On Instructions	
	Sent	Reset By	Branch Instruction *	Indicator Reset
Process Check due to detection of Transmission Error		Start Reset Key	B (III) 1	By executing the Branch Instruction in System B, or by pressing Start Reset Key in System A; both alternatives after pressing the Check Reset Key in System A.
End of Transmission. A GMWM was reached in the System A I/O — area during the previous data transfer. (The I/O — area in System A was smaller or equal in size to that of System B.)	I/O Disconnect	1401, 1440, 1460 Executing a K E instruction in the other system. 1440, 1460 Start Reset Key.	B (III) 2	By executing a K E Instruction in System B or by pressing the Start Reset Key in System A, if System A is a 1440 or 1460 System. Note: This Indicator must be off in both Systems before initiating any data transfer.
A Read Request Instruction (K C) has been executed in System A.	Read Request	Start Reset Key	B (III) 3	By executing a Write Data (with or without Word Marks) instruction, or pressing the Start Reset Key in System A.
A Write Request Instruction (K D) has been executed by System A.	Write Request	Start Reset Key	B (III) 4	By executing a Read Data (with or without Word Marks) Instruction, or pressing the Start Reset Key in System A.
A Write Data Instruction is being executed in System A.		Start Reset Key	B (III) 6	When System A ends the Write operation. (This is done when System B has executed a Read Instruction or by pressing the Start Reset Key in System A, or if the Indicator 2 was not reset in System A.)
A Read Data Instruction is being executed in System A.		Start Reset Key	B (III) 7	When System A ends the Read operation. (This is done when System B has executed a Write operation or by pressing the Start Reset Key in System A, or if the Indicator 2 was not reset in System A.)
System A stopped (Stop key pressed, STOP Instruction, error stop, etc.)		Start Reset Key	B (III) 8	When System A starts.

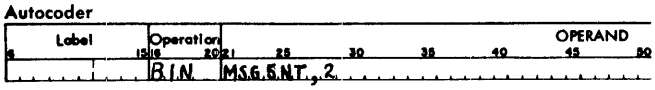
* The d-character must be in the operand field when using a BIN mnemonic.

Figure I-22. Branch if Indicator On Instruction Summary

Address Registers After Operation.

	I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
No Branch	NSI	BI	d*bb*
Branch (without indexing)	NSI	BI	Blank
Branch (with indexing)	NSI	BI	NSI

Example. Test for end of transmission by other system. If the other system did signal an end of transmission, branch to MSGSNT (0843), Figure I-23.



Assembled Instruction: B 843 2

Figure I-23. Branch if Indicator On

Move and Load Instructions

The MOVE or LOAD instruction (M or L (% H 1) (B B B) R or W) is used by the systems to transmit or receive the data in either the move mode or the load mode. The parts of the instruction and their use are:

M or L—The M or L operation code specifies whether the data transmission will be performed in the *move* mode or *load* mode. If the *move* mode is specified, up to 7 bits per character (CBA8421) are involved in the data transmission. If the *load* mode is specified, up to 8 bits per character (WM CBA8421) are involved in the data transmission. The same mode must be used by both systems for any one particular data transmission. Word marks would be lost if the message transmission were in the load mode, but the message reception were in the move mode.

% H 1 — The A-address (% H 1) specifies that the direct data channel feature is used in performing this instruction.

B B B — The B-address specifies the high-order position of the message in core-storage area involved in the data transmission.

R or W — A d-character of R specifies a read operation. This d-character is used when the other system is sending the data. A d-character of W specifies a write operation. This d-character is used when the other system is receiving the data.

The move and load instructions used, and the operations they initiate, are:

Read Data

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
MU	M	%H1	xxx	R

Function. This instruction causes the data sent from the other system to read into core storage, beginning at the core-storage location specified in the instruction.

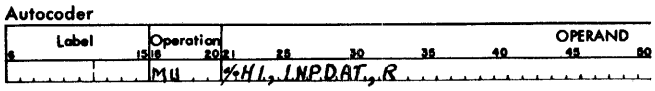
Word Marks. Word marks are not stored when operating in the *move* mode (M operation code).

Timing. T = N (L_I + 1) ms + transmission and start time.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%81	B + message length + 1

Example. Read data from the other system and place it in core storage, beginning at location 0633 (area is labeled INPDAT), Figure I-24.



Assembled Instruction: M %H1 633 R

Figure I-24. Read Data

Read Data with Word Marks

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
LU	L	%H1	xxx	R

Function. This instruction is similar to the READ DATA instruction except that word marks in the record area of core storage are removed, and word marks sent with the other data are written in core storage.

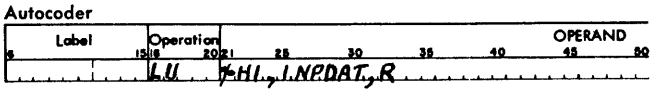
Word Marks. Word marks transmitted from other systems are written in core storage.

Timing. T = N (L_I + 1) ms + transmission and start time.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%81	B + message length + 1

Example. Read data from the other system, with its associated word marks, and place it in core storage, beginning at location 0633 (area is labeled INPDAT), Figure I-25.



Assembled Instruction: L %H1 633 R

Figure I-25. Read Data with Word Marks

Write Data

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
MU	M	%H1	xxx	W

Function. This instruction causes data to be sent to the other system from core storage, beginning at the core-storage location specified in the instruction.

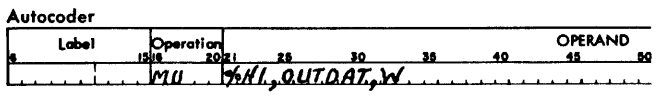
Word Marks. Word marks are not sent to the other system when operating in the *move* mode (M operation code).

Timing. $T = N\ (L_I + 1)$ ms + transmission and start time.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%81	B + message length + 1

Example. Send data to the other system from the core-storage area labeled OUTDAT (first position of the data located in 0633), Figure I-26.



Assembled Instruction: M %H1 633 W

Figure I-26. Write Data

Write Data with Word Marks

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
LU	L	%H1	xxx	W

Function. This instruction is similar to the WRITE DATA instruction except that word marks in the output area of core storage are transmitted with the associated data.

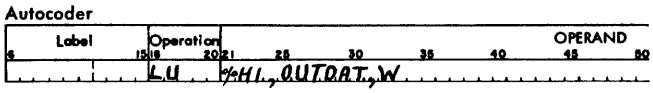
Word Marks. Word marks are sent to the other system.

Timing. $T = N\ (L_I + 1)$ ms + transmission and start time.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%81	B + message length + 1

Example. Send data to the other system, with its associated word marks, from the core-storage area labeled OUTDAT (first position of the data located in 0633), Figure I-27.



Assembled Instruction: L %H1 633 W

Figure I-27. Write Data with Word Marks

Instruction Utilization in the Program

With the instructions just described, the specific type of system-to-system data transmission can be set up. The type of operation performed is at the discretion of the user because the operation is completely programmed.

Each system has its own specifically designed program, using the previously described instructions. Some instructions are used in both programs, while other instructions might appear in only one program, if at all. The instructions used are completely dependent on the specific type of data transmission involved.

1-Way System-to-System Data Transmission

To illustrate one kind of system-to-system data transmission, assume a hypothetical case where one system sends data only, while the other system receives data only. The program procedure illustrated in Figure I-28 is meant only as an example to show the use of the various instructions and should not be considered the optimum procedure for this kind of operation.

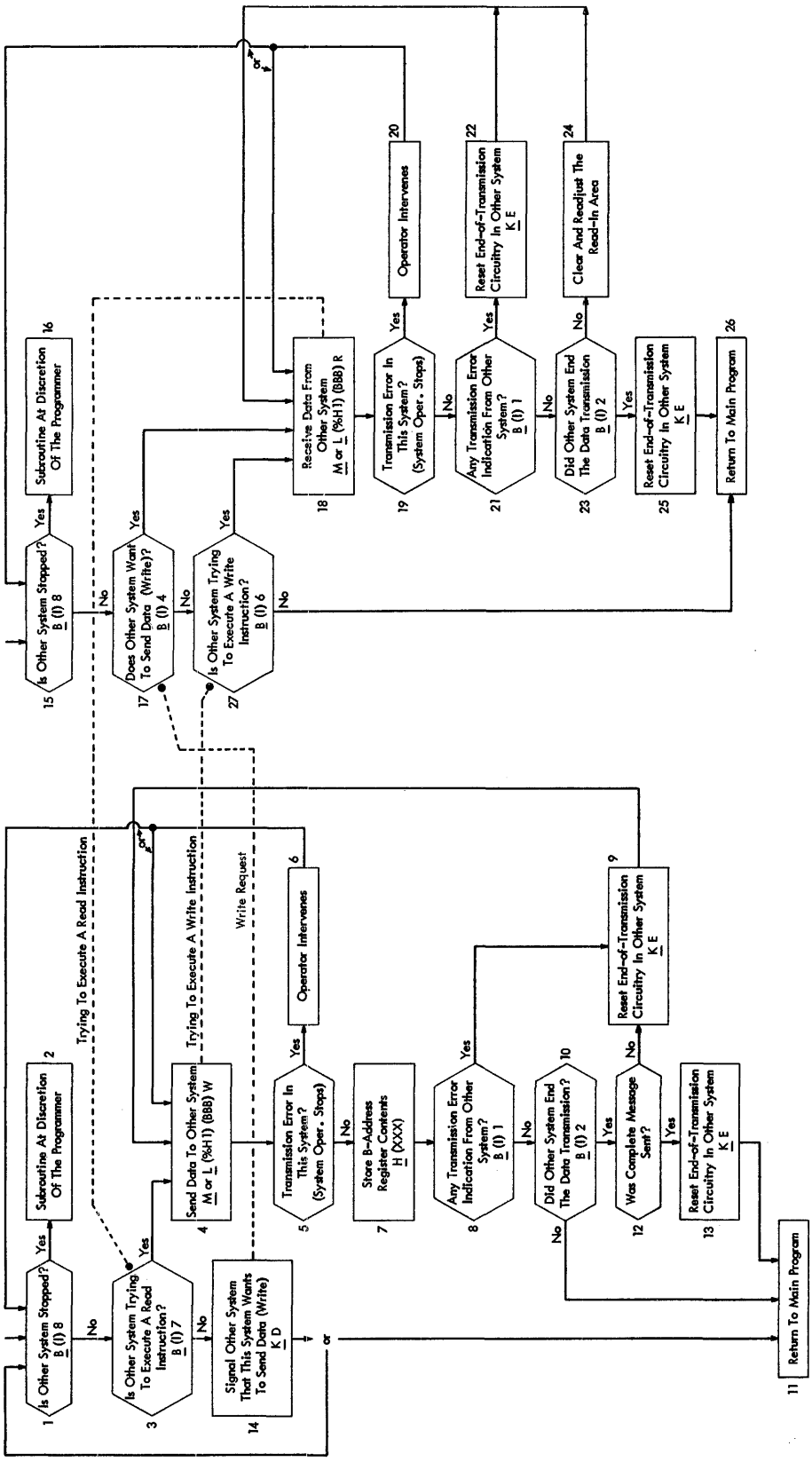


Figure I-28. Programming Example of 1-Way System to Data Transmission

Sending-System Operation

1. The sending system enters its system-to-system data transmission program and executes a BRANCH IF INDICATOR ON instruction, B (I I I) 8, which checks to see whether the other system is operating.
2. If the other system is stopped for any reason, the program branches into a previously specified subroutine, which may for example:
   a. permit processing of some other information, or,
   b. stop the system that initiated the instruction or,
   c. notify the system operator in some way.
3. If the other system is operating, another BRANCH IF INDICATOR ON instruction, B (I I I) 7, is executed, which checks the other system to see whether it is trying to execute a READ instruction. If the other system is trying to execute a READ instruction, it informs the sending system by setting the indicator tested by a B (I I I) 7 instruction.
4. When a B (I I I) 7 instruction results in a branch, the sending system immediately executes a WRITE instruction, M or L (% H 1) (B B B) W.

   The actual data transmission occurs between the two systems and continues until one of the systems encounters a preset group-mark with a word-mark in its core storage. The group-mark with a word-mark terminates the data transmission operation and sends a termination signal to the other system.
5. If any transmission error occurs in the sending system during the data transmission, the sending system stops at the end of the data transmission operation.
6. The system operator must start the system operating again and will either try to send the data again (step 4) or start at the beginning of the subroutine (step 1).
7. If no transmission error occurred in the sending system during the data transmission, then the core-storage address contained in the B-address register is stored in a location specified by the STORE B-ADDRESS REGISTER instruction, H (x x x). This information is used later to determine whether the complete message was transmitted.
8. A BRANCH IF INDICATOR ON instruction, B (I I I) 1 is executed, which checks to see whether any transmission errors occurred in the other system.
9. If any error occurred in the other system, the end-of-transmission circuitry in that system is reset (KE instruction) and the sending system tries to send the data again. The actual data transmission does not start until the operator corrects the error condition in the other (receiving) system and starts that system operating again.
10. If no transmission error occurred, a BRANCH IF INDICATOR ON instruction, B (I I I) 2, is executed, which checks to see whether the other system ended the data transmission.
11. If the other system did not end the data transmission, it means that the entire message was transmitted. The subroutine ends, and the system returns to its main program.
12. If the other system did end the data transmission, a check must be made to see whether the entire message was transmitted. One method that could be used is to compare the address stored in step 7 with the address known to be the last core-storage address in the sending system data area.
9. If the two addresses do not compare, then the end-of-transmission circuitry in the other system is reset, and the sending system tries to send the data again (step 4) because the receiving system did not receive the complete message.
13. If the two addresses do compare, it means that the entire message was transmitted. The end-of-transmission circuitry in the other system is reset.
11. The subroutine ends, and the system returns to its main program.

There is one other condition that could have occurred in the sending system. This is the condition that occurs when the other system is not trying to execute a READ instruction.

3. A BRANCH IF INDICATOR ON instruction, B (I I I) 7, is executed, which checks to see whether the other system is trying to execute a READ instruction.
14. If the other system is not trying to execute a READ instruction, then the sending system informs the other system that the sending system wants to send data by executing a WRITE REQUEST instruction, KD.

Receiving-System Operation

In the receiving system operation being used in this example, there are three conditions that might occur:

1. Sending system wants to send data, or,
2. Sending system is trying to execute a WRITE instruction, or,
3. Sending system does not want to send data and is not trying to execute a WRITE instruction.

Each one of these situations is discussed.

15. The receiving system enters its system-to-system data transmission program and executes a BRANCH IF INDICATOR ON instruction, B (I I I) 8, which checks to see whether the other system is operating.

16. If the other system is stopped for any reason, the program branches into a previously specified subroutine that may be similar to the subroutine described in step 2.

Condition 1—Sending system wants to send data.

17. If the other system is operating, another BRANCH IF INDICATOR ON instruction, B (I I I) 4 is executed. This instruction checks to see whether the other system wants to send data. This condition originated in step 14 of the sending system program. If the other system wants to send data, the receiving system immediately tries to execute a READ instruction, M or L (% H 1) (B B B) R. The data transmission does not take place immediately, however. In trying to execute a READ instruction, the receiving system signals the sending system that it is trying to execute a READ instruction. After a negligible time interval, the sending system enters its system-to-system data-transmission program and this condition initiates the data transmission previously described in step 4.

18. When the execution of a B (I I I) 4 instruction results in a branch, the receiving system immediately executes a READ instruction, M or L (% H 1) (B B B) R.

    The actual data transmission occurs between the two systems and continues until one of the systems encounters a preset group-mark with a word-mark in its core storage. The group-mark with a word-mark terminates the data transmission operation and sends a termination signal to the other system.

19. If any transmission error occurs in the receiving system during the data transmission, the system stops at the end of the data transmission.

20. The system operator must start the system operating again and will either try to receive the data again (step 18) or start at the beginning of the subroutine (step 15).

21. If no transmission error occurred in the receiving system during the data transmission, then a BRANCH IF INDICATOR ON instruction, B (I I I) 1 is executed, which checks to see whether any transmission errors occurred in the other system.

22. If any error occurred in the other system, the end-of-transmission circuitry in that system is reset and the receiving system tries to receive the data again (step 18).

23. If no transmission error occurred, a BRANCH IF INDICATOR ON instruction, B (I I I) 2 is executed, which checks to see whether the other system ended the data transmission.

24. If the other system did not end the data transmission, it means that the receiving system read-in area was not large enough to accept the incoming message. A subroutine is executed to readjust the read-in area so that it can accept the entire incoming message, and the receiving system tries to receive the data again (step 18).

At approximately this same time, steps 10, 12, and 9 are being executed in the other system. As previously described in these steps, the other system automatically tries to send the message again. As soon as the read-in area is adjusted, another data transmission operation takes place.

25. If the other system did end the data transmission, then the end-of-transmission circuitry in the other system is reset by initiating a RESET (KE) instruction.

26. The subroutine ends and the system returns to its main program.

Condition 2—Sending system trying to execute a WRITE instruction.

27. If the other system is operating, and is not trying to send data, another BRANCH IF INDICATOR ON instruction, B (I I I) 6 is executed. This instruction checks to see whether the other system is trying to execute a WRITE instruction as a result of an operator intervention or some other condition. If the other system is trying to execute a WRITE instruction, it informs the receiving system by setting the indicator tested by a B (I I I) 6 instruction. When the execution of a B (I I I) 6 instruction results in a branch, the program previously described in steps 18-26 is executed.

Condition 3—Sending system does not want to send data and is not trying to execute a WRITE instruction..

15, 17, 27, 26. If the other system is operating, but does not want to send data, and is not trying to execute a WRITE instruction, the subroutine ends and the system returns to its main program.

2-Way System-to-System Data Transmission

The programming involved in a 2-way system-to-system data transmission operation is, of necessity, more involved than the 1-way system-to-system programming just described. If both systems send and re-

ceive data, then the same program routine used by one system can also be used by the other system.

To permit maximum efficiency, each system must test the status of the other system at regular intervals. If the data transmission operations of one system have priority over the other system's operations, then the program must include routines that will terminate, or delay, the other system's operations.

If two duplicate programs are used, each program should include a dissimilar timing loop so that the systems do not re-enter their routines together after terminating an operation.

If both programs try to execute READ instructions at the same time, both the systems stop operating because all program execution stops. Each system then waits for the other system to start sending data, but neither one ever starts. This condition can be eliminated by proper programming.

If both programs try to execute WRITE instructions at the same time, the write operations are completed, but neither message is transferred, resulting in the loss of one message in each system. This condition can be eliminated by proper programming.

Direct Seek (1460)

This special feature reduces access time (250 ms maximum, 150 ms average) by allowing the access assembly to be positioned directly at a new setting without returning to the home position.

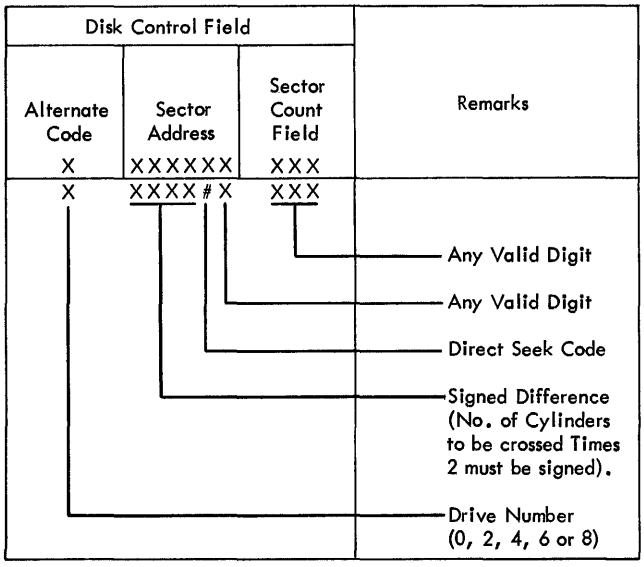


Figure I-29. Disk Control Field for Direct Seek

CHARACTER	BIT CODE
.	BA 8 21
≠	BA 8421
$	B 8 21
△	B 8421
'	A 8 21
≠	A 8421
√‾	8421

Figure I-30. Unacceptable Characters in Sixth Position of Disk Control Field

The instruction used for the direct seek operation is the same as that used with normal seek operation, M(%F0)(BBB)R. The B-address position of the instruction contains the core-storage address of the high-order position of the 6-digit disk-control field used.

Disk-Control Field

Direct seek operations use a 6-position disk-control field (Figure I-29).

The first position of the disk-control field contains the disk drive number (0, 2, 4, 6, or 8). An asterisk cannot be used for this operation.

The next four positions (2-5) contain a signed 4-digit number equal to twice the number of cylinders to be advanced (+) or retracted (−).

The sixth position contains a pound (#) sign to indicate a direct-seek operation.

NOTE: Any character with 8-2-1 bit combination will be taken to indicate a direct seek operation and cannot, therefore, be used in the sixth position. See Figure I-30 for a list of these characters.

The signed difference field can be calculated by the method shown in Figure I-31. This method uses the four high-order positions of the disk address at which the access arm is positioned and the four high-order

			Both Odd
New Address	004372	0043	0043
		+1	
Old Address	003291	0032	0033
		difference	0010

Signed Difference +0010 (+because increase in address)

Figure I-31. Calculating Signed Difference

Number of Cylinders Traveled	Time in Milliseconds	Number of Cylinders Traveled	Time in Milliseconds
1	54 Minimum	20	130
2	67	30	137
3	80	40	154
4	90	50	170
5	105	60	185
6	115	70	202
7	130	80	217
8	140	90	235
9	155	99	248 Maximum
10	165		

Figure I-32. Cylinder Seek Time with Direct Seek

positions of the disk address to be sought. Both fields must be changed to either odd or even (add one to even, subtract one from odd). The old address is then subtracted from the new address. The result of the subtraction has the correct sign to indicate that the mechanism is to advance (+) or retract (−).

If for some reason the address fails to specify the module in the alternate code position, direct seek will be executed on the master file.

Direct-Seek Timing

Figure I-32 provides seek times when direct seek feature is installed on the system.

Disk-Storage Control (1460)

This feature provides the controlling circuitry necessary for the proper operation of any IBM 1311 Disk Storage Drives attached to the system.

For more detailed information on the IBM 1311, refer to IBM *1311 Disk Storage Drive*, Form A24-3086.

Disk-Storage Drive Adapter (1401)

This feature provides the controlling circuitry to attach a 1311, Model 4, Disk Storage Drive to the 1401. For detailed information on the IBM 1311, refer to IBM *1311 Disk Storage Drive*, Form A24-3086.

Expanded Print Edit (1401 and 1460)

The basic operations of the MOVE CHARACTERS AND EDIT instruction can be increased by the expanded print edit feature. With this feature, asterisk protection, floating dollar sign, decimal control, and sign control left operations can be performed. The zero-suppression code in the control word should be in the position immediately to the left of the decimal, except as required in *Decimal Control.*

NOTE: Floating dollar sign and asterisk protection, or floating dollar sign and decimal control cannot be used in the same edit operation. When asterisk protection and decimal control are combined, and a blank data field is edited, the result is asterisks in all positions to the left of, but not including, the decimal-control position.

Asterisk Protection

When asterisks are to appear to the left of significant digits, the asterisk protection feature is used (Figure I-33). The control word is written with the asterisk immediately to the left of the zero-suppression code. Zero balances can be protected with asterisks by placing control zeros in the right-most position. In this instance, asterisks print in all positions including the decimal position.

Forward Scan:

1. The normal editing process proceeds until the asterisk is sensed.
2. The corresponding digit from the A-field replaces the asterisk (in the output field).
3. The editing process continues normally until the B-field word mark is sensed and removed.

Reverse Scan:

1. Asterisks replace zeros, blanks, and commas, to the left of the first significant digit.
2. The word mark (set during the forward scan) signals the end of editing. It is erased, and the operation is stopped.

A-field	00257426
Control word (B-field)	bbb, b*0. bb&CR
Forward scan	002,574.26 CR
Reverse scan	**2,574.26 CR
Results of edit	**2,574.26 CR

Figure I-33. Asterisk Protection

A-field	00257426
Control word (B-field)	bbbb, b$0. bb
First forward scan	002,574.26
Reverse scan	bbb 2,574.26
Second forward scan	$2,574.26
Results of edit	$2,574.26

Figure I-34. Floating Dollar Sign

Floating Dollar Sign

This feature causes the insertion of a dollar sign in the position at the left of the first significant digit in an amount field (Figure I-34). The control word is written with the $ immediately to the left of the zero-suppression code.

NOTE: The control word must be larger than the A-field. Three scans are necessary to complete this editing operation.

First Forward Scan:

1. The editing proceeds until the $ is sensed.
2. The corresponding digit from the A-field replaces the $ (in the output field).
3. Editing continues until the B-field word mark is sensed and removed.

Reverse Scan:

1. Blanks replace both zeros and commas to the left of the first significant digit.
2. The reverse scan continues until the word mark (set during the first forward scan) signals the start of the second forward scan.

Second Forward Scan:

1. The word mark is erased, and the scan continues until the first blank position is sensed. This blank position is replaced by $, and the operation stops.

Sign Control Left

CR or minus symbols can be placed at the left of a negative field, if the sign-control-left feature is used (Figure I-35). The control word is written with the CR or minus symbols in the high-order position.

Forward Scan:

1. The scan proceeds until the zero-suppression character in the control field is sensed.

A-field	00378940
Control word (B-field)	CR&bbb, bb0. bb
Forward scan	CRb003, 789.40
Reverse scan	CRbbb3, 789.40
Results of edit	CR 3,789.40

Figure I-35. Sign Control Left

2. The corresponding character from the A-field is placed in this position of the output field.
3. A word mark is automatically inserted in this position in the output field.
4. Editing continues, and the CR or minus symbols are undisturbed in their corresponding positions in the output field, only if the sign of the A-field is minus. If the sign is plus, they are blanked.

Reverse Scan:

1. Blanks in the output field replace zeros and commas. The scan continues until the automatically set word mark is sensed.
2. This word mark is erased and the operation ends.

Decimal Control

This feature ensures that decimal points print only when there are significant digits in the A-field (Figure I-36).

1. A-field	00000
Control word (B-field)	bbb. b0
First forward scan	000.00
Reverse scan	bbb. 00
Second forward scan	bbb
Results of edit	(Blank Field)
2. A-field	29437
Control word (B-field)	bbb. b0
First forward scan	294.37
Reverse scan	294.37
Result of edit	294.37
3. A-field	00001
Control word (B-field)	bbb. b0
First forward scan	000.01
Reverse scan	bbb. 01
Results of edit	.01

Figure I-36. Decimal Control

Two scans are sufficient to complete this editing operation, *unless* the field contains no significant digits. Then three scans are required.

First Forward Scan:

1. When the zero-suppression code (0) is sensed during editing, the corresponding digit from the A-field replaces this position.
2. A word mark is set automatically in this position in the B- (output) field.
3. Editing continues normally until the B-field word mark is sensed and removed.

Reverse Scan:

1. Blanks in the output field replace zeros and commas until the decimal point is sensed.
2. The decimal point and the digits at its right are unaltered. The automatically set word mark is erased. If there are no significant digits in the field, the second forward scan is initiated. Otherwise, the edit operation stops.

Second Forward Scan:

1. Blanks replace the zeros at the right of the decimal point and the decimal point itself.
2. The operation stops at the decimal column.

High-Low-Equal Compare Feature (1401; Standard on 1460)

This feature expands the compare operation to include indicators for high, low, or equal conditions. Additional d-characters are included so that the BRANCH IF INDICATOR ON instruction can test for these conditions. The basic machine permits testing for unequal conditions only.

This feature is standard on the IBM 1460 Data Processing System.

Branch if High, Low, or Equal Compare

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS B	B	xxx	x
A see chart			

Function. This instruction tests the compare indicator for the result of the previous compare operation and branches to the I-address, if the condition specified by the d-character is present:

Autocoder	*d-character*	*Branch to I-address if:*
BE	S	B = A (B equals A)
BL	T	B < A (B is less than A)
BH	U	B > A (B is greater than A)

If the condition is not present, the program continues with the next sequential instruction.

Word Marks. Word marks are not affected.

Timing.

No branch:

$T = N\ (L_I + 1)$ ms.

Branch (without indexing):

$T = N\ (L_I + 1)$ ms.

Branch (with indexing):

$T = N\ (L_I + 2)$ ms.

Address Registers After Operation.

	I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
No Branch	NSI	BI	d*bb*
Branch (without indexing)	NSI	BI	Blank
Branch (with indexing)	NSI	BI	NSI

Example. Compare the data at RECNO (0596) to the control number at CONTNO (0495). If the data at RECNO is higher than the control number at CONTNO, branch to GRTAN (0797) for the next instruction (Figure I-37).

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
010			C	CONTNO				RECNO				
020			B	GRTAN								U

Autocoder

Label	Operation	OPERAND
	C	CONTNO,RECNO
	BH	GRTAN

Assembled Instruction: C 495 596
B 797 U

Figure I-37. Branch if High Compare

Indexing and Store Address Register (1460)

The indexing-and-store-address-register special feature for the 1460 functions exactly the same as the indexing-and-store-address-register portions of the advanced-programming special feature for the 1401. For detailed information on indexing-and-store-address-register operation, refer to the appropriate sections of the *Advanced Programming* special feature write-up in this publication. Replace 1401 timing (.0115) with 1460 timing (.006).

Multiply-Divide (1401, 1460)

This feature makes it possible to perform direct multiplication and division in the IBM 1401 and/or 1460 Data Processing Systems.

Multiply

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*
M	@	xxx	xxx

Function. The multiplicand (data located in the A-field) is repetitively added to itself in the B-field. The B-field contains the multiplier in the high-order positions, and enough additional positions (low order) to allow for the development of the product. At the end of the multiply operation, the units position of the product is located at the B-address. The multiplier is destroyed in the B-field as the product is developed. Therefore, if the multiplier is needed for subsequent operations, it must be retained in another storage area.

The multiply-divide feature for the 1460 system has additional circuitry that automatically eliminates readdressing machine cycles when recomplementing is required during the operation.

Rules:

1. The product is developed in the B-field. The length of the B-field is determined by adding "1" to the sum of the number of digits in the multiplicand and multiplier fields.

Example:

```
  1246      4-digit multiplicand
× 543       3-digit multiplier
        +   1
  --------
        8 positions must be allowed in the B-field
```

2. A word mark must be associated with the high-order positions of both the multiplier and multiplicand fields.

3. A- and B-bits need not be present in the units positions of the multiplier and multiplicand fields. The absence of zone bits in these positions indicates a positive sign. At the completion of the multiply operation, the B-field will have zone bits in the units position of the product only. The multiply operation uses algebraic sign control (Figure I-38).

4. Zone bits that appear in the multiplicand field are undisturbed by the multiply operation. Zone bits in the units position of the multiplicand are interpreted for sign control.

Timing. The average time required for a multiply operation is:

$T = N (L_I + 3 + 2L_C + 5L_CL_M + 7L_M)$ ms.

L_C = length of multiplicand field.

L_M = length of multiplier field.

A chart of approximate timings is included in the section on *Timing.*

Note. The first addition within the multiply operation inserts zeros in the product field from the storage location specified by the B-address up to the units position of the multiplier.

The A-address register and the B-address register indicate positions within the A- and B-fields on which operations are currently being performed.

Word Marks. A word mark must be associated with the high-order positions of the multiplier and multiplicand fields.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	A minus the length of the multiplicand.	B minus the length of product field.

Multiplier Sign	+	+	−	−
Multiplicand Sign	+	−	+	−
Sign of Product	+	−	−	+

Figure I-38. Algebraic Sign Control for Multiplication

Example. Multiply:

Label	*Location of Data Word*	*Contents of Data Word*	*Description*
MULCAN	0502	1246	Multiplicand
MULIER	0065	543	Multiplier
PRODCT	0610		Product

The size of the product field is 4 + 3 + 1 = 8. The multiplier is placed in the three high-order positions of the PRODCT area (0603, 0604, and 0605). At the completion of the multiply operation, load the product in the area labeled OUT2 (0178). The units positions of the multiplier and multiplicand fields may be signed (Figure I-39).

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND	d
0,1,0			ZA	MULIER				PRODCT	-	5		
0,2,0			M	MULCAN				PRODCT				
0,3,0			LCA	PRODCT				OUT2				

Autocoder

Label	Operation	OPERAND
	ZA	MULIER,PRODCT-5
	M	MULCAN,PRODCT
	MLCWA	PRODCT,OUT2

Assembled Instruction: ? 065 605
@ 502 610
L 610 178

Figure I-39. Multiply

Divide

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*
D	%	xxx	xxx

Function. This instruction divides the data (dividend) in the low-order positions of the B-field by the divisor located in the A-field, and develops the quotient in the high-order positions of the B-field. The remainder is left in the low-order positions of the B-field.

Rules:

1. The quotient is developed in the B-field. The length of the B-field is determined by adding *1* to the sum of the number of digits in the divisor and dividend fields.

Example:

543 ⌐ 1246 — 4 digit dividend
3 digit divisor
\+ 1
8 positions must be allowed in the B-field

Divisor Sign	+	+	−	−
Dividend Sign	+	−	+	−
Quotient Sign	+	−	−	+
Remainder Sign	+	−	+	−

Figure I-40. Algebraic Sign Control for Division

2. A word mark must be associated with the high-order position of the A-field.
3. In all cases A- and B-bits (plus sign) or B-bit (minus sign) must appear in the units position of the dividend field. The divisor may be either signed or unsigned. If there are no bits in the units position of the divisor, the machine assumes the divisor factor is positive. The divide operation uses algebraic sign control (Figure I-40).

4. The dividend is loaded in the low-order positions of the B-field (Figure I-41) by a ZERO AND ADD instruction to ensure that zeros are present in the high-order positions of the B-field.
5. The B-address in the DIVIDE instruction specifies the high-order position of the dividend.

At the completion of division:

a. The quotient is in the high-order positions of the B-field. The location of the units position of the quotient, is the address of the units position of the dividend, minus the length of the divisor, minus one.
b. The remainder is in low-order positions of the B-field.
c. The sign of the quotient is over the units position of the quotient field.
d. Because only one quotient digit can be developed at a time, it is important to address the high-order position of the dividend (B-address of the DIVIDE instruction). This ensures that the first divide operation will result in a single high-order quotient digit. A dividend improperly addressed can cause an arithmetic overflow if the result of the first divide operation is greater than 9.

Word Marks. A word mark must define the high-order position of the divisor.

Timing. Average time required for the execution of a divide operation is calculated:

$$T = N\,(L_I + 2 + 7L_RL_Q + 8L_Q)\ \text{ms}.$$

L_Q = length of the quotient field.
L_R = length of the divisor field.

Dividend
±
0000XXXX

Figure I-41. Dividend in B-field

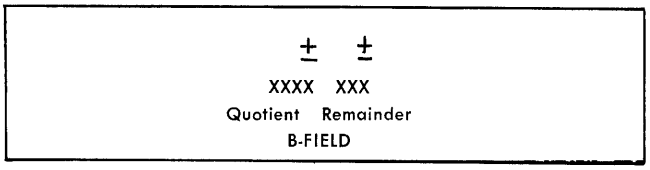


Figure I-42. Location of the Results of a Divide Operation

A chart of approximate timings is included in the section on *Timing*.

Note. A divide operation refers to the process of developing each quotient *digit*. If the quotient field is not large enough, no overflow is indicated. This is a programming error for which the machine does not check. Division by zero results in an arithmetic overflow condition. Figure I-42 shows the result of a divide operation.

Extra zeros can be added to the dividend prior to a divide operation when a larger quotient is required. For each additional quotient digit desired, place one zero to the right of the dividend as shown in Figure I-43. Note that in this example, the units position of the quotient is *not* located in the position previously described in the section *At The Completion of Division, a.*

The quotient field is not cleared before actual division begins.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	A minus the length of the divisor.	Tens position of quotient. If divisor has all zeros, the B-address register stands at the units position of the dividend minus the length of the divisor minus the length of the dividend minus 1.

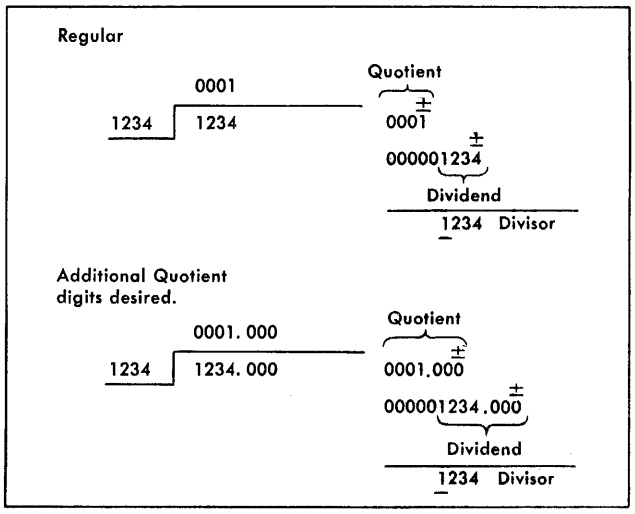


Figure I-43. Additional Quotient Digits

Example. DIVIDE (Figure I-44).

Label	*Location of Data Word*	*Data Word*	*Description*
DIVEND	0502	1246	Dividend
DIVSOR	0065	543	Divisor
QUOT	0985		Quotient

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	WM	(B) OPERAND ADDRESS	±	CHAR. ADJ.	WM	d
010			ZA	DIVEND				QUOT				
020			D	DIVSOR				QUOT	-	3		

Autocoder

Label	Operation	OPERAND
	ZA	DIVEND,QUOT
	D	DIVSOR,QUOT-3

Assembled Instruction: ? 502 985
% 065 982

Figure I-44. Division

1401 MULTIPLY TIMES* WITH SPECIAL FEATURE												
No. Pos. Multiplicand ⟶		1	2	3	4	5	6	7	8	9	10	11
Number Positions Multiplier ↓	1	.276	.347	.418	.489	.560	.631	.702	.773	.844	.915	.986
	2	.414	.552	.690	.828	.966	1.104	1.242	1.380	1.518	1.656	1.794
	3	.552	.748	.944	1.140	1.336	1.532	1.728	1.924	2.120	2.316	2.512
	4	.690	.943	1.196	1.449	1.703	1.956	2.209	2.462	2.715	2.968	3.221
	5	.828	1.139	1.450	1.761	2.072	2.383	2.694	3.005	3.316	3.627	3.938
	6	.966	1.334	1.702	2.070	2.438	2.806	3.174	3.542	3.910	4.278	4.646
	7	1.104	1.530	1.956	2.382	2.808	3.234	3.660	4.086	4.512	4.938	5.364
	8	1.242	1.725	2.208	2.691	3.174	3.657	4.140	4.623	5.106	5.589	6.072
	9	1.380	1.921	2.462	3.003	3.544	4.085	4.626	5.167	5.708	6.249	6.790
	10	1.518	2.116	2.714	3.312	3.910	4.508	5.106	5.704	6.302	6.900	7.498
* Multiply 1401 Times by Factor of .522 for 1460 Times												

Figure I-45. IBM 1401 and 1460 Multiply Times (with Special Feature)

Multiply and Divide Timing

The four timing charts give the approximate timings of multiply (Figures I-45 and I-46) and divide (Figures I-47 and I-48) operations. Two of the charts are based on the timings when a subroutine written in actual language is used. The other two charts are based on the timings required when the system is equipped with the special feature for multiply and divide.

Multiply and Divide Subroutines

These are subroutines for multiply and divide operations, discussed here to illustrate programming methods and to aid programming for machines not equipped with the multiply-divide special feature. These are not the only methods of performing these operations—they are typical methods.

1401 MULTIPLY TIMES* BASED ON MULTIPLY SUBROUTINE

No. Pos. Multiplicand ⟶	1	2	3	4	5	6	7	8	9	10	11
Number Positions Multiplier ↓ 1	3.51	3.63	3.74	3.86	3.97	4.08	4.20	4.32	4.43	4.55	4.67
Value of each 2	7.10	7.33	7.56	7.79	8.02	8.25	8.48	8.71	8.94	9.17	9.40
Multiplier Digit 3	10.74	11.08	11.43	11.77	12.12	12.46	12.81	13.15	13.50	13.84	14.19
assumed to be an 4	14.42	14.88	15.34	15.80	16.26	16.72	17.18	17.64	18.10	18.56	19.02
average of "5" 5	18.15	18.73	19.30	19.88	20.45	21.03	21.60	22.18	22.75	23.33	23.90
6	21.93	22.62	23.31	24.00	24.69	25.38	26.07	26.76	27.45	28.14	28.83
7	25.76	26.57	27.37	28.18	28.98	29.79	30.59	31.40	32.20	33.01	33.81
8	29.63	30.55	31.47	32.39	33.31	34.23	35.15	36.07	36.99	37.91	38.83
9	33.54	34.58	35.61	36.65	37.68	38.72	39.75	40.79	41.82	42.86	43.89

(Add 1ms to times shown above if signing of quotient is required)

* Multiply 1401 Times by Factor of .522 for 1460 Times

Figure I-46. IBM 1401 and 1460 Multiply Times (based on Multiply Subroutine)

1401 DIVIDE TIMES* WITH SPECIAL FEATURE

No. of Pos. in Quotient ⟶	1	2	3	4	5	6	7	8	9	10
No. of Pos. in Divisor ↓ 1	.276	.449	.621	.794	.966	1.139	1.311	1.484	1.656	1.829
2	.357	.610	.863	1.116	1.369	1.622	1.875	2.128	2.381	2.634
3	.438	.771	1.104	1.437	1.770	2.103	2.436	2.769	3.102	3.435
4	.518	.932	1.346	1.760	2.174	2.588	3.002	3.416	3.830	4.244
5	.598	1.093	1.588	2.083	2.578	3.073	3.568	4.063	4.558	5.053
6	.679	1.254	1.829	2.404	2.979	3.554	4.129	4.704	5.279	5.854
7	.759	1.415	2.071	2.727	3.383	4.039	4.695	5.351	6.007	6.663
8	.840	1.576	2.312	3.048	3.784	4.520	5.256	5.992	6.728	7.464
9	.920	1.737	2.553	3.370	4.186	5.003	5.819	6.636	7.452	8.269
10	1.001	1.898	2.795	3.692	4.589	5.486	6.383	7.280	8.177	9.074

* Multiply 1401 Times by Factor of .522 for 1460 Times

Figure I-47. IBM 1401 and 1460 Divide Times (with Special Feature)

1401 DIVIDE TIMES* BASED ON DIVIDE SUBROUTINE											
No. of Pos. in Quotient ⟶		1	2	3	4	5	6	7	8	9	10
No. of Pos. in Divisor ↓	1	8.502	13.703	18.904	22.762	26.411	30.060	33.709	37.358	41.007	44.656
	2	8.870	13.899	18.928	22.909	26.730	30.551	34.372	38.193	42.014	45.835
	3	9.238	14.094	18.950	23.060	27.054	31.048	35.042	39.036	43.030	47.024
Value of each Quotient Digit assumed to be an average of "5"	4	9.606	14.290	18.974	23.208	27.374	31.540	35.706	39.872	44.038	48.204
	5	9.974	14.485	18.996	23.359	27.698	32.037	36.376	40.715	45.054	49.393
	6	10.342	14.681	19.020	23.507	28.018	32.529	37.040	41.551	46.062	50.573
	7	10.710	14.876	19.042	23.658	28.342	33.026	37.710	42.394	47.078	51.762
	8	11.078	15.072	19.066	23.806	28.662	33.518	38.374	43.230	48.086	52.942
	9	11.446	15.267	19.088	23.957	28.986	34.015	39.044	44.073	49.102	54.131
	10	11.815	15.468	19.113	24.105	29.306	34.507	39.708	44.909	50.110	55.311
(Add 1ms to times shown above if signing of quotient is required)											
* Multiply 1401 Times by Factor of .522 for 1460 Times											

Figure I-48. IBM 1401 and 1460 Divide Times (based on Divide Subroutine)

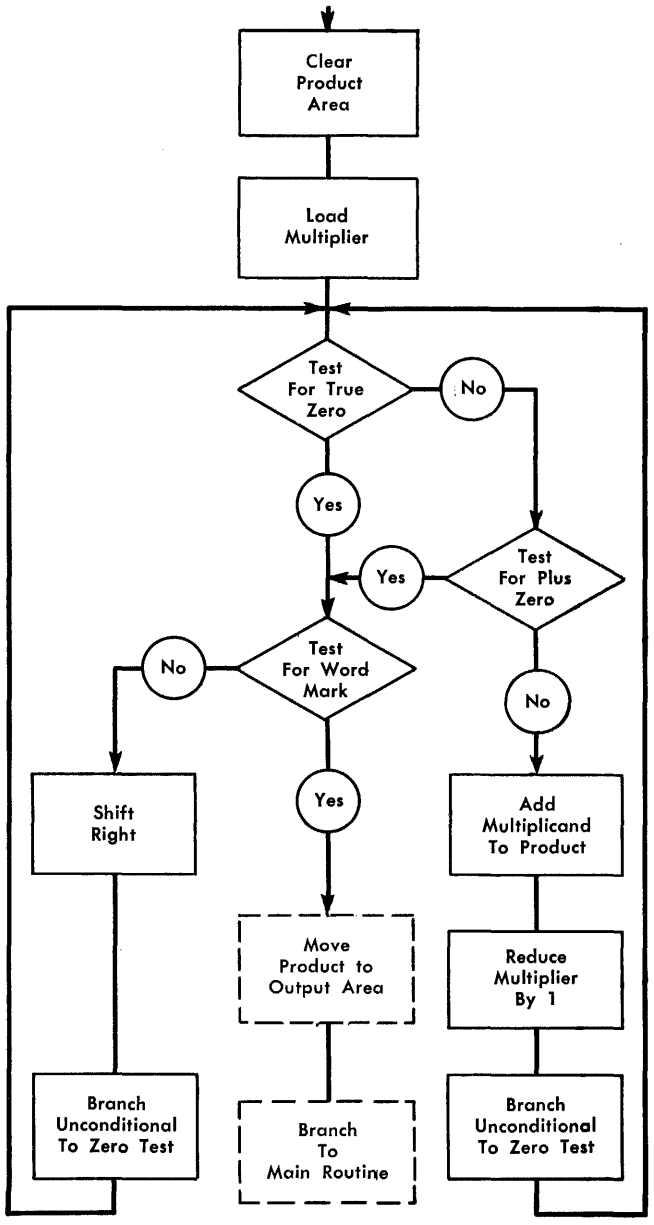


Figure I-49. Multiply Flow Chart

Multiply Subroutine

The block diagram in Figure I-49 illustrates the logic used in developing the two multiply subroutines discussed here. Both subroutines provide for a maximum of a 9-digit multiplier, 11-digit multiplicand, and a 20-digit product. Both routines use positive factors.

The subroutine written in actual language (Figure I-50) occupies the 900 block of storage. A multiplier area is provided in storage positions 901-909, and the product area is assigned in storage positions 910-929. The multiplicand can be located anywhere.

IBM 1401 PROGRAM CHART

Program: Multiply Subroutine

Programmer: _____ Date: _____

Step No.	Inst. Address	OP	Instruction A/I	Instruction B	Instruction d	Remarks	Effective No. of Characters Inst.	Data	Total
1	930	/	929			Clear Product Area	4		
2	934	L	XXX	909		Load Multiplier	7		
3	941	B	975	909	0	Test 909 for true zero	8		
4	949	B	975	909	?	No true zero - test for plus zero	8		
5	957	A	YYY	921		No zero - add Multiplicand to Product	7		
6	964	S	994	909		Reduce Multiplier by 1	7		
7	971	B	941			Branch to zero test	4		
8	975	V	995	909	1	Test 909 for Word Mark	8		
9	983	L	928	929		No Word Mark - Shift Right	7		
10	990	B	941			Branch to zero test	4		
11	994	1				Constant "1"	1		
12	995	B	ZZZ			Multiplication complete - Branch back to Program			
			XXX			Location of Multiplier			
			YYY			Location of Multiplicand			
			ZZZ			Address of next Program Step			
						9 digit Multiplier			
						11 digit Multiplicand			
						20 digit Product			

Figure I-50. Multiply Subroutine (actual)

IBM 1401 Symbolic Programming System
Coding Sheet

Program Multiply Subroutine Page No. 1 of 2

Programmed by Date Identification

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			LCA	ZERO,S				AREA					CLEAR PROD AREA
020			LCA	MPLIER				AREA	-	020			LOAD MPLIER
030		ZROTST	B	TESTWM				AREA	-	020		0	BRANCH TRUE ZERO
040			B	TESTWM				AREA	-	020		?	BRANCH PLUS ZERO
050			A	MCAND				AREA	-	008			ADD MCAND
060			S	ONE				AREA	-	020			SUB1 FROM MPLIER
070			B	ZROTST									BRANCH TO ZROTST
080		TESTWM	BWZ	LAST				AREA	-	020		L	TEST FOR WM
090			LCA	AREA	-	001		AREA					SHIFT AREA RT 1
100			B	ZROTST									BRANCH TO ZROTST
110		LAST	B	0000									MULT COMPLETE
120													
130													
140													
150													
160	30	ZEROS	DCW	*									
170	30	AREA	DCW	*									
180	9	MPLIER	DCW	*									
190	11	MCAND	DCW	*									
200	1	ONE	DCW	*		1							

Figure I-51. Multiply Subroutine (Symbolic)

Any program that uses this subroutine must include a step that moves the multiplier address (XXX) to location 937 and the multiplicand address (YYY) to location 952.

At the completion of the multiply subroutine, the program instruction step 12 can use a branch to the main program or stop instruction.

The routine starts in storage position 930. The product is found in 929 for a 9-digit multipler, 928 for 8-digit, 927 for 7-digit, 926 for 6-digit, etc.

The subroutine written using the symbolic programming system (Figure I-51) parallels the one written in actual. In this instance, the 1401 (through the processor program) controls the location of the instructions, and the data and work areas.

NOTE: The multiply subroutine results in blanks instead of zeros in the low-order position of a product when the multiplier contains low-order zeros. To correct this situation, set the product area to zeros.

Divide Subroutine

The restrictions placed on this subroutine (Figures I-52 and I-53) are:

1. The dividend and quotient fields must be of equal length.
2. The dividend and divisor must both be positive.
3. The divisor must have no zone for its positive indication. This is necessary only if the divisor could be zero.
4. The divisor cannot contain more than nine leading zeros.
5. All fields must be located completely below address 999.
6. At the completion of the subroutine, the address of the units position of the quotient can be found in the B-address of the instruction located in 651.
7. The remainder is left in the dividend field.
8. A word mark must be located immediately to the right of the units position of the dividend.
9. The quotient area must be preset to zeros or blanks to develop the correct quotient. If the area is not zeroed or blanked, then the quotient will be added to whatever is there. The positions added will depend on the number of leading zeros in the divisor.

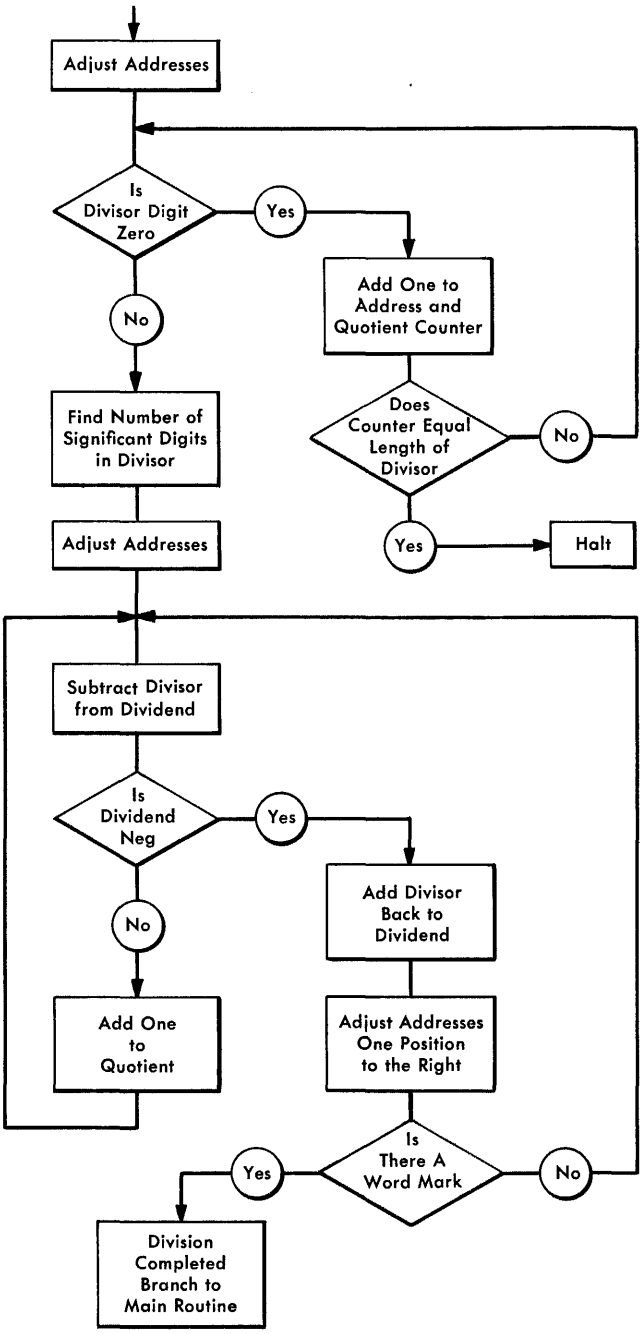


Figure I-52. Divide Flow Chart

10. The information shown in *Data for Division Subroutine* (Figure I-53), except the constant 1 in location 513, must be set initially for each desired execution of the divide subroutine. The two addresses in locations 507 and 510, associated with the divisor, are not altered. Thus, they do not have to be reinitialized if the divisor is contained in the same area.

Numerical Print Control (1401, 1460)

This feature is required when the numerical-print special feature is installed on the IBM 1403 Printer (Models 1 or 2 on 1401; Model 2 on 1460). This feature provides the checking circuitry necessary for the operation of the numerical-print special feature. For detailed information on this feature, refer to the IBM *1403 Printer*, Form A24-3073.

Print Control (1401)

This feature controls the additional 32 print positions of the IBM 1404 Printer, which includes the storage and checking circuitry when used with the 1401 only.

Print Control, Additional (1401)

This feature controls the additional 32 print positions of the IBM 1403 Printer, Model 2, attached to a 1401, and includes the storage and checking circuitry. For detailed information on the 1403, Model 2, refer to the IBM *1403 Printer*, Form A24-3073.

Print Storage (1401, 1460)

This special feature provides 100 or 132 non-addressable extra positions of core storage on the 1401 (132 positions are standard on the 1460). They are used as temporary storage for printer output data.

When this feature is installed, the WRITE LINE instruction transfers the data in the printer area (storage location 201-332) to print storage. Two milliseconds after the WRITE LINE instruction is given, the normal printer interlock is released and processing can continue. Thus, available processing time during a 100-ms cycle is 98 ms, as opposed to the 16-ms processing time available when the machine is not equipped with print storage. The print-storage area interlocks for 84 ms during printing on 1401 system. Therefore, another WRITE LINE instruction given during this 84 ms will cause the program to stop until the interlock is released.

IBM 1401 PROGRAM CHART

FORM X24-6437-0
PRINTED IN U.S.A.

Program: DIVIDE ROUTINE EXAMPLE

Programmer: _____ Date: _____

Step No.	Inst. Address	Instruction OP	A/I	B	d	Remarks	Effective No. of Characters Inst.	Data	Total
	516	M	507	529		Store Address of Word Mark Position (High Order) of Divisor			
	523	B	662	YYY	0	Branch if Divisor Digit Equals Zero			
	531	S	512	515		(Length of Divisor) - (Number of High Order Zeros)			
	538	A	515	501		Adjust Dividend Address			
	545	A	515	504		Modify Addresses			
	552	S	513	501					
	559	S	513	504					
	566	Y	755	501		Clear zone from low-order position to prepare for address assignment.			
	573	Y	755	504		Clear zone from low-order position to prepare for address assignment.			
	580	M	501	642		Set Modified Addresses into Divide Routine			
	587	M	501	649					
	594	M	501	698					
	601	M	501	705					
	608	M	501	719					
	615	M	504	657					
	622	M	510	639		Set Divisor Address			
	629	M	510	695					
	636	S	ZZZ	WWW		Subtract Divisor from Dividend			
	643	V	692	WWW	K	Branch if Negative Result			
	651	A	513	XXX		Add One to Quotient			
	658	B	636			Repeat Subtraction			
	662	A	513	529		Add One to YYY Address			
	669	A	513	512		Increase Counter by One			
	676	C	512	515		If Equal, Divisor Equals Zero			
	683	B	523		/	Branch Unequal			
	688	.	760			Halt - cannot Divide by Zero			
	692	A	ZZZ	WWW		Add Divisor to Dividend			
	699	Y	755	WWW		Move blank zone to word mark position of dividend			
	706	A	513	719					
	713	V	760	WWW	1	Test for End of Divide			
	721	A	513	642		Modify Addresses to Develop Next Quotient Digit			
	728	A	513	649		" " " "			
	735	A	513	657		" " " "			
	742	A	513	698		" " " "			
	749	A	513	705		" " " "			
	756	B	636			Return to Divide Calculations			
	760					Divide Complete			

DATA FOR DIVISION SUBROUTINE

Location of Data Word	Data Word	DESCRIPTION OF DATA
501	WWW	Address of word mark position (high order) of dividend
504	XXX	Address of word mark position (high order) of quotient
507	YYY	Address of word mark position of divisor
510	ZZZ	Divisor Address
512	00	Counter for number of zeros in divisor
513	1	Constant
515	NN	Length of the divisor

Figure I-53. Divide Subroutine

With this feature, two indicators can be tested by the BRANCH IF INDICATOR ON instruction, B(xxx)d.

The print-storage-busy indicator turns ON during the first 83⅓ ms of the print cycle. The d-character P tests the indicator. If it is ON, the program branches; if it is OFF, the next sequential program step is taken.

The printer-error indicator (d-character of =) should not be tested until the program has determined that the print-storage-busy indicator is OFF. If the error indicator is tested before the busy indicator turns OFF, the system interlocks until the print-storage operation in process is completed.

The carriage-busy indicator turns ON during form-movement time. The d-character R tests this indicator. If it is ON, the program branches; if it is OFF, the next sequential program step is taken.

On a 1460 system the print-storage feature provides a storage medium so that nearly all of the normal interlock time is released during printing (99 ms of processing time released for each 100-ms print cycle). Therefore, processing time is greatly increased for applications characterized by high printing requirements.

Programming Considerations

When the print-storage special feature is installed, an improperly placed BRANCH IF CARRIAGE CHANNEL #9 (or #12) instruction can cause incorrect carriage overflow operation on the 1403.

On a system equipped with print storage, the system is interlocked for only 2 ms during the 100-ms print cycle. The system is then free to continue processing for the rest of the print cycle. Because of this, it is possible to execute a BRANCH IF CARRIAGE CHANNEL #9 (or #12) instruction during this additional processing time. If this instruction is executed before the print operation is completed, the instruction actually checks for a channel 9 or 12 hole too soon.

When the print operation that spaces the carriage tape to a channel 9 or 12 finally takes place, the channel 9 or 12 hole will not be checked for until the next print operation is under way. This condition results in printing an extra line before the carriage overflow operation occurs.

Printer (1404) Adapter (1401)

This feature provides the controls and checking circuitry necessary to attach a 1404 printer to the IBM 1401.

Printer Adapter — 1403, Model 3 (1460)

This feature provides the circuitry necessary for attaching and controlling an IBM 1403 Printer, Model 3, on a 1460 system.

Printer Control Adapter — Second Printer (1460)

This feature provides the circuitry necessary to attach the first IBM 1462 Printer Control Unit and its associated 1403 printer to the 1460 system.

Processing Overlap (1401, 1460)

The processing-overlap special feature for the IBM 1401 and/or 1460 Data Processing Systems provides, for many applications, great reductions in over-all job time. The high-speed processing and input-output abilities of the 1401 and 1460 now can be used with maximum efficiency. Jobs requiring extensive input-output operations and lengthy programming now can be performed at or near maximum speeds. The actual increase in efficiency and consequent saving in job time varies with the specific program, and depends on the input-output requirements of the particular application.

The processing-overlap feature allows processing to be interrupted in order to take input-output cycles. A character can be read, written, or punched between processing cycles.

Serial input-output devices used with the IBM 1401 and/or 1460 systems, such as the IBM 1011 Paper Tape Reader and the IBM 1419 Magnetic Character Reader, can have their operations performed in overlap mode. In some cases, entire input-output operations can be overlapped; in others, partial overlapping can occur.

Job time required for card input-output applications is reduced because the processing operation is not interlocked during card reading or punching. This is also true when reading or writing magnetic tape. Tape operations and processing can occur on alternate cycles. This ability can result in appreciable increase in job time economy.

The processing-overlap special feature makes the IBM 1401 and 1460 Data Processing Systems more powerful tools for use in the field of data processing.

NOTE: On 1460 systems equipped with this feature and the IBM 1448 Transmission Control Unit, processing overlap cannot be used in any program that also uses the 1448.

Data Flow

The overlap special feature requires the addition of an O-register and an O (overlap)-address register to the processing unit (Figure I-54). These registers function in much the same manner as the A- and B-registers and the I-, A-, and B-address registers. That is, the O-register is used for movement of data when operating with an I/O unit. The O-address register is used

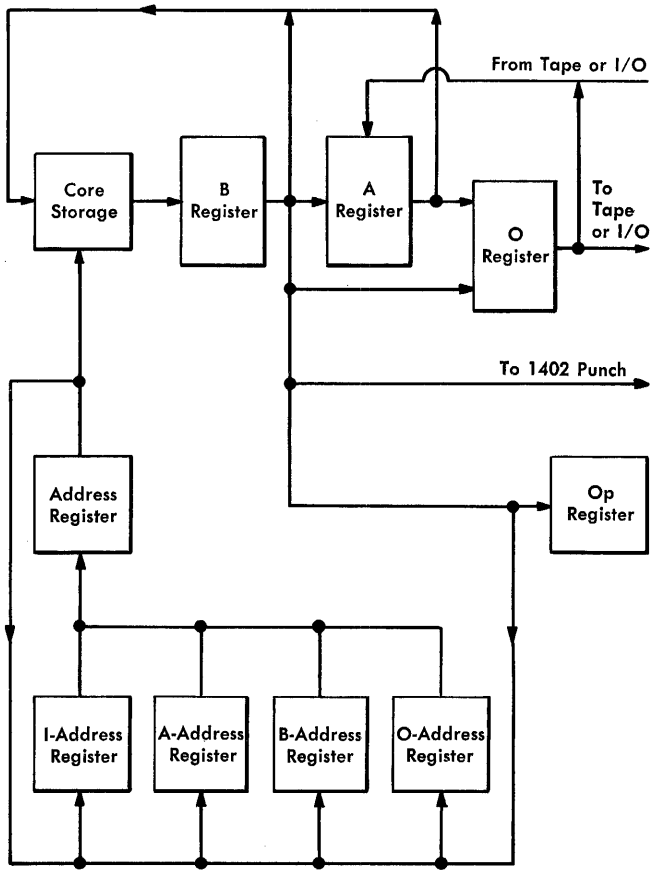


Figure I-54. IBM 1401 and 1460 Processing Overlap Data Flow

to keep track of the location of data that is moved by an I/O operation. These registers, when used by the program, operate on alternate cycles with the normal registers. Thus the 1401 takes processing cycles and then an input-output cycle, when required, instead of interlocking processing while an input-output operation is being performed. This means a savings in overall job time.

IBM 1402 Card Read-Punch (Model 1 on 1401; Model 3 on 1460)

When the system is in the overlap mode, processing can occur during the card cycle in which data is read from, or punched into, a card. Processing does not occur during the time used for either read or punch scanning, but alternates with scanning.

At the beginning of each read or punch scan, the character in the A-register is sent to the O-register (Figure I-55) and the O-address register is set at the column being scanned. When the read or punch scan of the character is completed, the character in the O-register is sent back to the A-register. At this point processing resumes.

Tape Operations

During a *read* operation, data from magnetic tape enters the A-register and then moves into core storage. During an overlap-read operation, data necessary for processing is contained in the A-register. In order to save the data, it is sent to the O-register until recalled for the next processing cycle, before data is read in from tape to the A-register (Figure I-56). The O-address register contains the storage address in which the data being read from magnetic tape is to be stored at the same time that the I-, A-, and B-address registers keep track of the processing operation being performed

During a *write* operation, data from core storage passes throught the B-register and enters the O-register (Figure I-57). It is sent to magnetic tape from the O-register. Processing cycles use the A- and B-register. The O-address register contains the storage address of the data being written on magnetic tape. At the same time the I-, A-, and B-address registers keep track of the processing operation being performed.

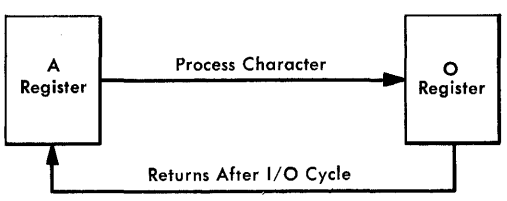


Figure I-55. IBM 1402 Overlap Operation

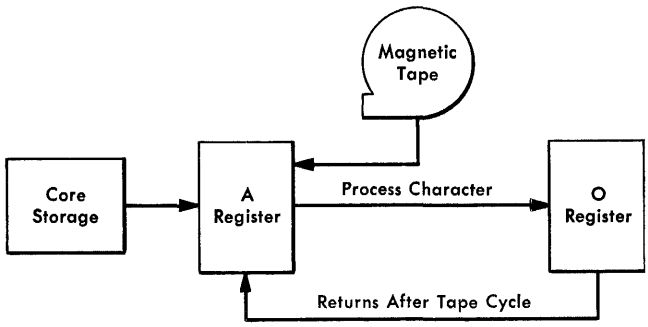


Figure I-56. Overlapped Magnetic-Tape Read Operation

To store the contents of the O-address register at the end of an overlapped tape operation the following operations must be performed:

B (xxx) J	BRANCH IF TAPE BUSY INDICATOR ON. This instruction allows the contents of the O-address register to be transferred to the B-address register (with the advanced-programming feature) if the indicator is not ON.
Then	
H xxx	Store B-address register

Other Input-Output Units

Input-output units such as the IBM 1419 Magnetic Character Reader, when operating in the overlap mode, function in the same manner as a magnetic-tape unit. Overlapping and processing occurs just as if the operation were a tape read or write operation.

Processing-Overlap Instructions

To signal the system that an operation to be performed should be done in the overlap mode, special instructions and A-address changes must be used in the program.

A-Address

The A-address hundreds position of a tape or input-output unit (not 1405 or 1407) instruction is changed from % to @. The @ (at) symbol signals the processing unit that the operation to be performed should be done as an overlap operation. The @ symbol can be used to signal an overlap operation with the:

IBM 1011 Paper Tape Reader
IBM 1419 Magnetic Character Reader
IBM 729 II, IV, V, VI Magnetic Tape Unit
IBM 7330 Magnetic Tape Unit

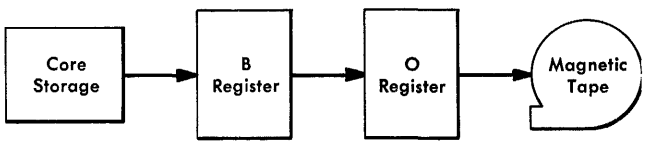


Figure I-57. Overlapped Magnetic-Tape Write Operation

Overlap On

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	$

Function. This instruction sets the system processing unit in the overlap mode. All 1402 read, punch, and combination card instructions that follow this instruction in the program are performed in the overlap mode.

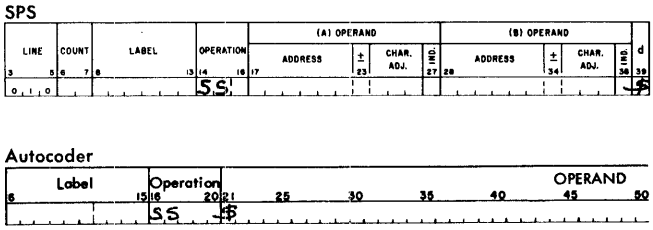
Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Example. Set the system in the overlap mode (Figure I-58).



Assembled Instruction: K $

Figure I-58. Overlap On

Overlap On and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS SS	K	xxx	$
A SSB			

Function. This is the same as OVERLAP ON, except that the next instruction is taken from the I-address.

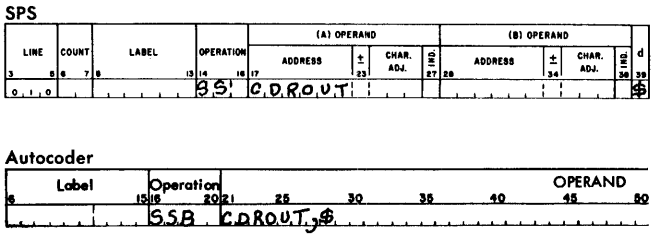
Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	d*bb*

Example. Set the system in the overlap mode and branch to CDROUT (0950) for the next instruction (Figure I-59).



Assembled Instruction: K 950 $

Figure I-59. Overlap On and Branch

Overlap Off

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	•

Function. This instruction returns the system processing unit to normal operation. All card input-output instructions following this instruction are performed without the overlap feature.

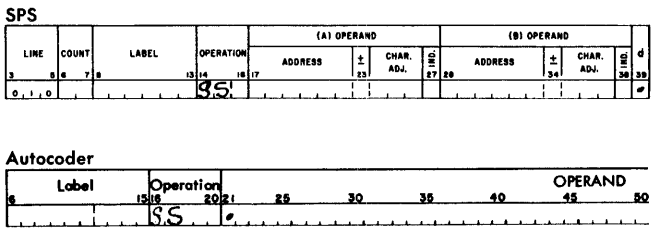
Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Example. Place the system in normal operation (Figure I-60).



Assembled Instruction: K •

Figure I-60. Overlap Off

Overlap Off and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS SS	K	xxx	•
A SSB			

Function. This instruction is the same as OVERLAP OFF, except that the next instruction is taken from the I-address.

Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	d*bb*

Example. Place the system in normal operation, and branch to the routine labeled NORCRD (1771) for the next instruction (Figure I-61).

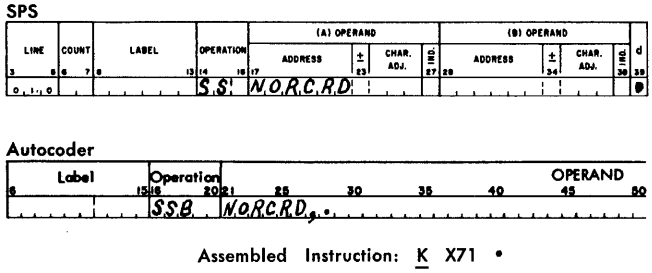


Figure I-61. Overlap Off and Branch

Reset Overlap

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	□

Function. This instruction is operative only when the serial input-output adapter special feature is installed. If the system is performing an overlapped serial I/O operation when this instruction is given, the I/O device is disconnected from the system.

Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Note. If an overlap magnetic-tape operation is in process when this instruction is given, the tape unit is disconnected and the tape transmission error indicator is turned ON.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Example. Reset overlap mode and return the system to normal operation (Figure I-62).

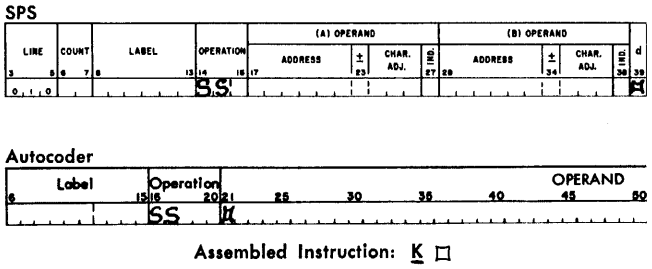


Figure I-62. Reset Overlap

Reset Overlap and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS SS	K	xxx	□
A SSB			

Function. This instruction is the same as RESET OVERLAP, except that the next instruction is taken from the I-address.

Word Marks. Word marks are not affected.

Timing. $T = N (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	d*bb*

Example. Reset overlap mode and branch to routine labeled NORMOP (1755) for the next operation (Figure I-63).

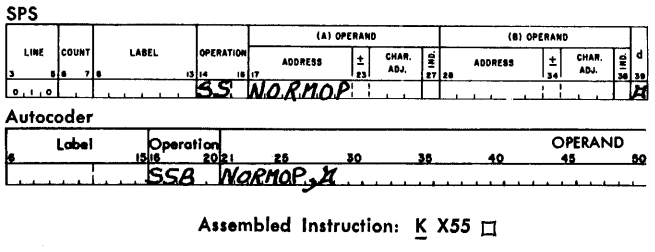


Figure I-63. Reset Overlap and Branch

Read Tape (with or Without Word Marks) in Overlap Mode

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MU	M	@xx	xxx	R (without word marks)
SPS LU	L	@xx	xxx	R (with word marks)

NOTE: The move and load operations differ only in the handling of the word-separator characters on tape.

Function. The tape unit specified in the A-address is started. The d-character specifies a tape-read operation. The @ in the hundreds position of the A-address indicates that this operation is to be performed in the overlap mode.

The B-address specifies the high-order position of the tape read-in area of storage. The machine reads magnetic tape until either an inter-record gap in the tape record or a group-mark with a word-mark in core storage is sensed. The inter-record gap indicates the end of the tape record, and a group mark (code CBA 8421) is inserted in core storage at this point.

Word Marks. Word marks are not affected in *move* mode. However, a word separator character read from tape causes a word mark to be associated with the next character transferred to core storage in *load* mode.

Timing. $T = N\ (L_I + 1)$ ms + T_M. (See section on *Processing Overlap Timing.*)

Note:

1. The internal circuitry of the processing-overlap special feature temporarily forces the R d-character of the instruction into the first position of the specified read-in area. If this first position is used to check for a blank field before the first character arrives from the I/O unit, the R d-character gives a false indication of the read-in area's contents. To ensure proper system operation, any blank field checking should use some other read-in area position than the first position.

2. Some of the tape start time is available for processing when a tape read operation is being performed in the overlap mode. Refer to Figures I-73 (1401) and I-77 (1460) for the specific times.

3. Depending on the tape unit, the tape speed, and the tape density involved, some of the record transfer time may also be available for processing while the tape-read operation is being performed in the overlap mode. Refer to Figures I-74 (1401) and I-78 (1460) for the specific times.

Address Registers After Operation.

I-Add. Reg.	*O-Add. Reg.*
NSI	Group mark + 1

Example. Read the record from the tape unit labeled 2 to the core-storage area labeled OUTPAR (0419). The tape-record characters are moved into core storage until the transfer is stopped by an inter-record gap in the tape record, or a group-mark with a word-mark is sensed in core storage (Figure I-64).

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	(A) OPERAND CHAR. ADJ.	(A) OPERAND IND	(B) OPERAND ADDRESS	(B) OPERAND CHAR. ADJ.	(B) OPERAND IND	d
			MU	@U2			OUTPAR			R

Autocoder

Label	Operation	OPERAND
	RT	@U2,OUTPAR

Assembled Instruction: M @U2 419 R

Figure I-64. Read Tape in Overlap Mode (Without Word Marks)

Write Tape (with or Without Word Marks) in Overlap Mode

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MU	M	@xx	xxx	W (without word marks
SPS LU	L	@xx	xxx	W (with word marks

NOTE: The move and load operations differ only in the handling of the word marks in the storage area.

Function. The tape unit designated in the A-address is started. The d-character specifies a tape-write operation. The data from core storage is written on the tape record. The @ in the hundreds position of the A-address indicates that this operation is to be performed in the overlap mode.

The B-address specifies the high-order position of the record in storage. A group-mark with a word-mark in core storage stops the operation. The group-mark with a word-mark causes an inter-record gap.

Word Marks. Word marks are not affected in the move mode. However, a word mark associated with any position in core storage causes a word-separator character (A841) to be written automatically on tape, one character ahead of that which contained the word mark. Thus, word marks are translated to word-separator characters for tape storage.

Notes:

1. Some of the tape start time is available for processing when a tape-write operation is being performed in the overlap mode. Refer to Figures I-73 (1401), and I-77 (1460) for the specific times.

2. Depending on the tape unit, the tape speed, and the tape density involved, some of the record transfer time may also be available for processing while a tape-write operation is being performed in the overlap mode. Refer to Figures I-74 (1401) and I-78 (1460) for the specific times.

Address Registers After Operation.

I-Add. Reg.	*O-Add. Reg.*
NSI	Group mark + 1

Example. Transfer the contents of the core-storage area labeled TPOUT (0525) to the tape unit labeled 3. The transfer of data is stopped by a group-mark with a word-mark in core storage (Figure I-65).

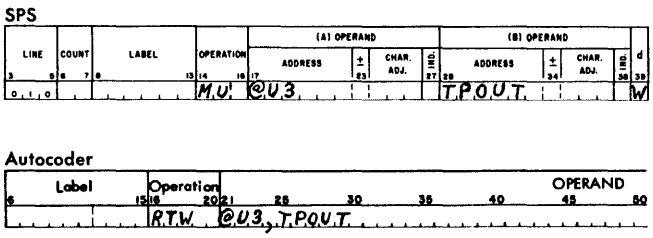


Figure 1-65. Write Tape in Overlap Mode (Without Word Marks)

Read Card in Overlap Mode

Instruction Format.

Mnemonic	*Op Code*
R	1

Function. This code, if overlap is ON, causes a card to feed, and causes all 80 columns of information to be read into core-storage locations 001 through 080 while processing continues.

Word Marks. Word marks are undisturbed.

Timing. $T = N\ (L_I + 1)$ ms + I/O.

Note: Processing is interrupted each time a row in the card is scanned (9-row, 8-row, etc.). The bits in a column are developed in the A-register, and the O-address register keeps track of which column is being scanned. While the scanning operation is being performed, the data required for processing is temporarily stored in the O-register. When the row scan is completed, the data in the O-register is sent back to the A-register and processing continues.

The read-in area of core storage *must not* be addressed while an overlap operation is being performed.

Address Registers After Operation.

I-Add. Reg.	*O-Add. Reg.*
NSI	081

Example. Read a card in the overlap mode (Figure I-66).

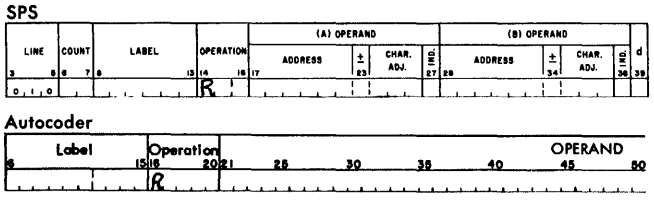


Figure I-66. Read Card in Overlap Mode

Read Card in Overlap Mode and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*
R	1	xxx

Function. This is the same as the READ CARD instruction, except that the next instruction is taken from the I-address instead of from the next, sequential instruction address. The program branch occurs when the instruction is processed; that is, before card reading is completed.

Word Marks. Word marks are not affected.

Timing.

Without indexing:

$T = N\ (L_I + 1)$ ms + I/O.

With indexing:

$T = N\ (L_I + 2)$ ms + I/O.

Address Registers After Operation.

I-Add. Reg.	*O-Add. Reg.*
NSI	081

Example. Read a card in the overlap mode, and branch to the location labeled OVERCD (1500), Figure I-67.

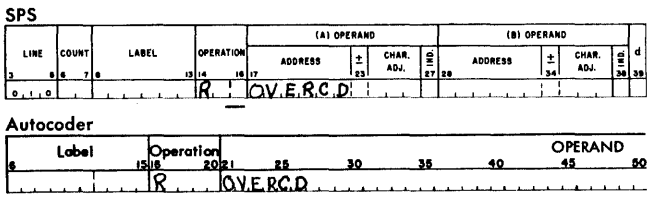


Figure I-67. Read Card in Overlap Mode and Branch

Punch Card in Overlap Mode

Instruction Format.

Mnemonic	*Op Code*
P	4

Function. This code, if overlap mode is ON, causes the data in storage locations 101 through 180 to be punched into an IBM card while processing continues.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms + I/O.

Note. Processing is interrupted each time a row in the card (12-row, 11-row, etc.) is scanned for punching. The data to be punched is developed in the B-register, and the O-address register keeps track of which column is being punched. While the punch-scan operation is being performed, the data required for processing (from the A-register) is temporarily stored in the O-register. When the row punch-scan is completed, the data in the O-register is sent back to the A-register, and processing continues.

The punch-out area of core storage *must not* be addressed while an overlap operation is being performed.

Address Registers After Operation.

I-Add. Reg.	*O-Add. Reg.*
NSI	181

Example. Feed a card, and punch in overlap mode (Figure I-68).

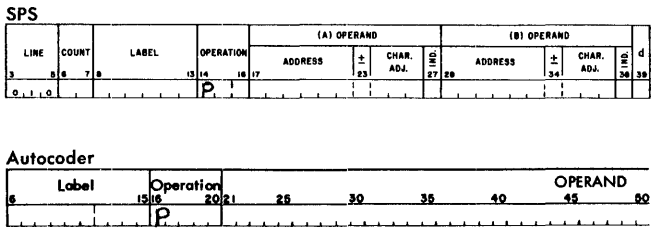


Figure I-68. Punch Card in Overlap Mode

Punch Card in Overlap Mode and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*
P	4	xxx

Function. This is the same as the PUNCH A CARD instruction, except that the next instruction is taken from the I-address instead of from the next, sequential instruction address. The branch occurs when the instruction is processed; that is, before punching has been completed.

Word Marks. Word marks are not affected.

Timing.

Without indexing:

$T = N\ (L_I + 1)$ ms + I/O.

With indexing:

$T = N\ (L_I + 2)$ ms + I/O.

Address Registers After Operation.

I-Add. Reg.	*O-Add. Reg.*
NSI	181

Example. Punch a card in overlap mode, and branch to the location labeled OVSTAT (1758), Figure I-69.

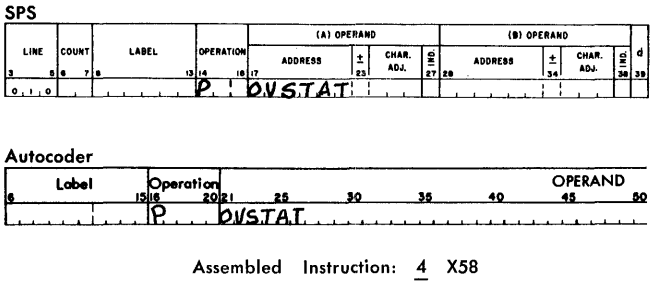


Figure I-69. Punch Card in Overlap Mode and Branch

Branch if Indicator On

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS B *A* BIN	B	xxx	x

Function. The d-character specifies the indicator tested. If the indicator is ON, the next instruction is taken from the I-address. If the indicator is OFF, the next sequential instruction is taken. Figure I-70 shows the symbols that are valid d-characters and the indicators they test.

d-CHARACTER	INDICATORS
H	Reader Busy
I	Punch Busy
J	Tape or Input-Output Busy

Figure I-70. Processing Overlap Indicators

Indicators. Reader Busy. This indicator turns on when the IBM 1401 is performing an overlapped read operation. The indicator automatically turns off when the overlapped read operation is completed.

Punch Busy. This indicator turns on when the 1401 is performing an overlapped punch operation. The indicator automatically turns off when the overlapped punch operation is completed.

Tape or Input-Output Busy. This indicator turns on when the 1401 is performing an overlapped magnetic tape or input-output (1419, or 1011) operation. The indicator automatically turns off when the overlapped operation is completed.

Word Marks. Word marks are not affected.

Timing.

Reader or Punch Busy

No branch:

$T = N (L_I + 1)$ ms.

Branch (without indexing):

$T = N (L_I + 1)$ ms.

Branch (with indexing):

$T = N (L_I + 2)$ ms.

Tape or I/O Busy

No branch (without indexing):

$T = N (L_I + 1)$ ms.

No branch (with indexing):

$T = N (L_I + 2)$ ms.

Branch (without indexing):

$T = N (L_I + 1)$ ms.

Branch (with indexing):

$T = N (L_I + 2)$ ms.

Note. When the tape or I/O busy indicator is *not* ON, and indexing (part of another 1401 or 1460 special feature) is installed on the system, an O-Add. Reg.-to-B-Add. Reg. operation takes place before going to the next instruction. This operation stores the last address used in the normal manner for operations such as read-compressed-tape. If indexing is not installed, the B-Add. Reg. will contain *dbb*.

Address Registers After Operation.

	I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
Reader or Punch Busy			
No branch	NSI	BI	*dbb*
Branch (without indexing)	NSI	BI	Blank
Branch (with indexing)	NSI	BI	NSI
Tape or I/O Busy			
No branch (without indexing)	NSI	BI	*dbb*
No branch (with indexing)	NSI	BI	O-Add. Reg. Contents
Branch (without indexing)	NSI	BI	Blank
Branch (with indexing)	NSI	BI	NSI

Example. Test the reader-busy indicator, and branch to a routine beginning at the location labeled RDBUSY (0891), if the indicator is ON (Figure I-71).

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND	d
3 5	6 7	8 13	14 16	17 23			27	28 34			38	39
0 1 0			B	RDBUSY								H

Autocoder

Label	Operation	OPERAND
6 15	16 20	21 25 30 35 40 45 50
	BIN	RDBUSY,H

Assembled Instruction: B 891 H

Figure I-71. Branch if Reader Busy Indicator On

Programming Considerations

1. A 1-character B-field arithmetic operation, where recomplementing can occur, *must not* be executed while an input-output unit is operating in the overlap mode.

2. During overlap operation the processing-overlap internal circuitry initiates a dummy B-cycle at the beginning of an overlap execute operation. This B-cycle transfers the B-address register contents to the O-address register. During this address-register transfer operation, the specified core-storage position also reads out to the B-register. This condition could cause various problems (see note 1 of the READ TAPE IN OVERLAP MODE instruction). Another example would be during binary tape read operation. The first character position could easily contain a group-mark with a word-mark (sent during the last operation). When the group-mark with a word-mark is sensed in the B-register during this B-cycle, the read-in operation would end, and the record would not be transferred.

3. In general, no other input-output operation can be initiated until the preceding overlapped I/O function is completed. The exceptions to this rule are:

   a. If the 1401 and/or 1461 is equipped with the print-storage special feature, the following print

instructions can be executed simultaneously with an overlapped input-output function:

PRINT
PRINT AND BRANCH
PRINT WORD MARKS
PRINT WORD MARKS AND BRANCH

b. When performing a normal CARD READ instruction in the overlap mode, the following output instructions can be performed:

PUNCH
PUNCH AND BRANCH
PRINT AND PUNCH
PRINT, PUNCH AND BRANCH
PRINT WORD MARKS AND PUNCH
PRINT WORD MARKS, PUNCH AND BRANCH

NOTE: The last four instructions involving the printer would require the print storage feature to be installed on the system in order for it to be overlapped.

It is possible to maintain an approximate ratio of three card-read operations to one card-punch operation with the proper placement of BRANCH ON BUSY instructions. That is, after a punch operation, test for punch busy. If it is not busy, branch back to punch. If it *is* busy, test for reader busy. If the reader is not busy, perform a read operation and branch back to the test-punch-busy operation. If the reader was busy, go to the main program (which eventually goes to the test-punch-busy operation).

c. When performing a normal PUNCH instruction, the following additional input-output instructions can be performed:

READ
READ AND BRANCH
PRINT AND READ
PRINT, READ AND BRANCH
PRINT WORD MARKS AND READ
PRINT WORD MARKS, READ, AND BRANCH

NOTE: The last four instructions would require print storage in order to be overlapped.

d. If performing a tape operation in the overlap mode, a READ or PUNCH instruction in the overlap mode can be given successfully so long as the total tape operation is completed before the first read or punch scan (inability to store address of last position of tape record can result).

e. The last card indicator is turned on during the time the last card is being read. Therefore, to ensure proper reading of the card, test for a reader busy condition before the branch on last card test is given

The last card indicator is turned off if the system is in an overlap mode, and the program completes a successful branch if the last card indicator is ON.

System-Interlock Conditions

The following conditions cause the 1401 system to interlock:

1. When the IBM 1401 and/or 1460 is operating in an overlap mode, the system interlocks under certain conditions.

B (xxx) ? — BRANCH ON READER ERROR. If the reader is operating when this instruction is given, the 1401 or 1460 interlocks until the overlapped operation is completed, before making the error test. If punching is being overlapped at the same time, the 1401 or 1460 interlocks until the end of the read and the punch operations.

B (xxx) ! — BRANCH ON PUNCH ERROR. If the punch is operating when this instruction is given, the 1401 or 1460 interlocks until the overlapped operation is completed before making the error test. If the reader is being overlapped at the same time, the 1401 or 1460 interlocks until the end of the read and the punch operations.

B (xxx) L — BRANCH ON TAPE TRANSMISSION ERROR. If the tape adapter unit is busy when this instruction is given, the 1401 or 1460 interlocks until the TAU is not busy, before making the test for a tape transmission error.

B (xxx) H — BRANCH ON READER BUSY. If the reader is busy when this instruction is given, and the interlock-stop condition is ON in the 1402, the 1401 or 1460 stops. The stop occurs for these conditions:
a. a card jam in the 1402 read or punch feed
b. a full stacker
c. an empty hopper
d. the I/O check stop switch on the 1401 or 1460 console is ON, and one of the following conditions occurs:
Hole-count check
Validity check
Punch check

B (xxx) I — BRANCH ON PUNCH BUSY. If the punch is busy when this instruction is given, and the interlock-stop condition is ON in the 1402, the 1401 or 1460 stops. The stop occurs for these conditions:
a. a card jam in the 1402 read or punch feed
b. a full stacker
c. an empty hopper
d. the I/O check stop switch on the 1401 or 1460 console is ON, and one of the following conditions occurs:
Hole-count check
Validity check
Punch check

2. When operating in the overlap mode, the following conditions cause a processing unit stop:

a. When a storage-address error occurs, processing stops.

b. If a process error occurs during an overlapped I/O cycle, processing stops immediately; however, the input-output operation continues.

OPERATION	1402 CYCLE ms	PROCESSING TIME AVAILABLE		
		STANDARD ms	WITH READ RELEASE ms	WITH PROCESSING OVERLAP ms
Read	75	10	31	63
Read Column Binary	75	10	31	52
Punch	240	22	59	224
Punch Column Binary	240	22	59	212
Read and Punch	240	22	56	213
Read-Punch Feed	240	22	56	209

Figure I-72. Summary of Overlapped Card Operation Timing (1401 system)

c. If a process error occurs between overlapped I/O cycles, processing stops immediately, and the input-output operation continues.

3. Read and punch-release operation codes (8 and 9) are not operative in the overlap mode. However, the read or punch (1 and 4) operation codes, when used in the overlap mode, perform the same function. If a tape operation overextends the time allotted it (data to or from tape is still being processed when data from a card is available), a tape transmission error occurs and the data from tape does not enter/leave core storage. Normal tape-error procedure can correct this type of error. If a tape operation is given after a read or punch (1 or 4) operation, the tape operation and processing are interlocked until the read or punch operation is completed.

Processing-Overlap Timing

The addition of the processing-overlap special feature can provide significant reductions in over-all job times. To gain the greatest advantage from this feature, careful timing consideration should be given to the development of each program and to the operations that will be overlapped. Charts showing the increase of available processing time by using this feature are included to aid in the efficient placement of the overlapped instructions.

IBM 1401 System Timing Considerations

The input-output units that can make use of the processing-overlap special feature on the 1401 system are:

1. IBM 729 Magnetic Tape Units, Models II, IV, V
2. IBM 1011 Paper Tape Reader, Model 1
3. IBM 1402 Card Read-Punch, Model 1
4. IBM 1419 Magnetic Character Reader, Model 1
5. IBM 7330 Magnetic Tape Unit

IBM 1011 Paper Tape Reader

One core-storage cycle is required for the transfer of each character read from the 1011. The core-storage cycle is 11.5 μsec. The character rate for the 1011 is 1 character every 2000 μsec. This leaves an available processing time of 1989.5 μsec between 1011 characters.

IBM 1419 Magnetic Character Reader

One core-storage cycle is required for the transfer of each character read from the 1419. The core-storage cycle is 11.5 μsec. The actual 1419 speed is dependent upon the document length and the stored program.

IBM 1402 Card Read-Punch

When reading cards in the overlap mode, processing is interrupted each time a row in the card is scanned. During a normal 75-ms card read cycle, 63 ms are available for other processing. The available processing time for this operation, and other card read-punch operations, are shown in Figure I-72.

Magnetic Tape

Without the processing-overlap special feature, the processing unit is interlocked during the entire tape operation. However, the addition of processing overlap releases the processing unit so that other processing can occur during the tape start times of a high-density tape operation as shown in Figure I-73.

Tape Unit	Operation	Total Start Time (ms)	Start Time Available For Processing (ms)
7330	Read	10.3	10.3
	Write	5.0	5.0
729 II/V	Read	10.5	6.6
	Write	7.5	7.3
729 IV	Read	6.7	4.4
	Write	5.0	4.9

Figure I-73. Magnetic-Tape Start Times Available for Processing on 1401 System

Tape Unit	Tape Density	Operation	Tape Character Rate (ms)	Average Transfer Per Char (ms)	Average Processing Time Available (ms)
7330	200 CPI	Read	.139	.0115	.127
		Write	.139	.0115	.127
	556 CPI	Read	.050	.014	.036
		Write	.050	.0115	.038
729 II/V	200 CPI	Read	.067	.0115	.055
		Write	.067	.0115	.055
729 IV/VI	200 CPI	Read	.044	.014	.030
		Write	.044	.0115	.032

Figure I-74. Magnetic-Tape Record Transfer Times Available for Processing on 1401 System

In addition to making tape operation start times available for processing, the processing-overlap special feature also makes some time available during the record transfer time at the tape speeds and tape densities shown in Figure I-74.

Figure I-75 can be used to help determine the amount of processing time available during overlapped tape read and/or write operations.

An example of available processing time is: Reading a 1000-character tape record written in high-density (556 CPI) on an IBM 7330 Magnetic Tape Unit, with the processing-overlap special feature.

Tape Operation Time	*Total Time (ms)*	*Time Available For Processing (ms)*
Start Time	10.3	10.3
Record Time	50.0	36.0
Remaining TAU Interlock Time	12.9	12.9*
Total Time	73.2	59.2

*Assumes that the tape error indicator is tested at the proper time.

Without the processing-overlap special feature, only 12.8 milliseconds of the total time is available for processing.

IBM 1460 System Timing Considerations

The input-output units that can make use of the processing-overlap special feature on the 1460 system are:

1. IBM 729 Magnetic Tape Units, Models II, IV, V, VI
2. IBM 1011 Paper Tape Reader, Model 1
3. IBM 1402 Card Read-Punch, Model 3
4. IBM 1419 Magnetic Character Reader, Model 1
5. IBM 7330 Magnetic Tape Unit

IBM 1011 Paper Tape Reader

One core-storage cycle is required for the transfer of each character read from the 1011. The core-storage cycle is 6 μsec. The character rate for the 1011 is 1 character every 2000 μsec. This leaves an available processing time of 1994 μsec between 1011 characters.

IBM 1419 Magnetic Character Reader

One core-storage cycle is required for the transfer of each character read from the 1419. The core-storage cycle is 6 μsec. The actual 1419 speed is dependent upon the document length and the stored program.

IBM 1402 Card Read-Punch

When reading cards in the overlap mode, processing is interrupted each time a row in the card is scanned. During a normal 75-ms card read cycle, 68 ms are

Tape Unit	Tape Density	Operation	Tape Adapter Unit Interlocked (ms)	Approximate Processing Time Available (ms)		Approximate Gain In Processing Time (ms)
				Without Overlap	With Overlap	
7330	200 CPI	Read	21.1 + CN	10.5	21.1 + .127N	10.6 + .127N
		Write	20.9 + CN	15.9	20.9 + .127N	5.0 + .127N
	556 CPI	Read	20.5 + CN	10.1	20.5 + .036N	10.4 + .036N
		Write	20.3 + CN	15.3	20.3 + .038N	5.0 + .038N
729 II/V	200 CPI	Read	11.1 + CN	0.6	11.1 + .055N	10.5 + .055N
		Write	12.1 + CN	4.6	12.1 + .055N	7.5 + .055N
	556/800 CPI	Read	10.7 + CN	0.2	6.8	6.6
		Write	11.7 + CN	4.2	11.5	7.3
729 IV/VI	200 CPI	Read	7.1 + CN	0.4	7.1 + .030N	6.7 + .030N
		Write	8.1 + CN	3.1	8.1 + .032N	5.0 + .032N
	556 CPI	Read	6.8 + CN	0.1	4.5	4.4
		Write	7.8 + CN	2.8	7.7	4.9

Figure I-75. Summary of Available Processing Time During Magnetic Tape Operations on the 1401 System

Operation	1402 Cycle Time (ms)	Available Processing Time (ms)		
		Standard	With Read Release	With Processing Overlap
Read	75	10	31	68
Punch	240	22	59	230
Read Column Binary	75	10	31	63
Punch Column Binary	240	22	59	224
Read and Punch	240	22	56	224
Punch Feed Read	240	22	56	223

Figure I-76. Summary of Overlapped Card Operation Timing (1460 System)

available for other processing. The available processing time for this operation, and other card read-punch operations, are shown in Figure I-76.

Magnetic Tape

Without the processing-overlap special feature, the processing unit is interlocked during the entire tape operation. However, the addition of processing overlap releases the processing unit so that other processing can occur during the tape start times of a high-density tape operation as shown in Figure I-77.

In addition to making tape operation start times available for processing, the process-overlap special feature also makes some time available during record transfer time on some tape units as shown in Figure I-78.

Figure I-79 can be used to help determine the amount of processing time available during overlapped tape read and/or write operations.

Tape Unit	Operation	Total Start Time (ms)	Start Time Available For Processing (ms)
7330	Read	10.3	10.3
	Write	5.0	5.0
729 II	Read	10.5	10.5
	Write	7.5	7.5
729 IV	Read	6.7	4.4
	Write	6.0	4.9
729 V	Read	10.5	6.6
	Write	7.5	7.3
729 VI	Read	6.7	4.4
	Write	5.0	4.9

Figure I-77. Magnetic-Tape Start Times Available for Processing on the 1460 System

Tape Unit	Tape Density	Operation	Character Rate (ms)	Average Time/Char (ms)	Average Processing Time Available/Char (ms)
7330	200 CPI	Read	.139	.006	.133
		Write	.139	.006	.133
	556 CPI	Read	.050	.006	.044
		Write	.050	.006	.044
729 II/V	200 CPI	Read	.067	.006	.061
		Write	.067	.006	.061
	556 CPI	Read	.024	.007	.017
		Write	.024	.006	.018
729 IV/VI	200 CPI	Read	.044	.006	.038
		Write	.044	.006	.038

Figure I-78. Magnetic-Tape Record Transfer Times Available for Processing on the 1460 System

Tape Unit	Tape Density	Operation	Tape Adapter Unit Interlocked (ms)	Approximate Processing Time Available (ms)		Approximate Gain In Processing Time (ms)
				Without Overlay	With Overlay	
7330	200 CPI	Read	21.1 + CN	10.5	21.1 + .133N	10.6 + .133N
		Write	20.9 + CN	15.9	20.9 + .133N	5.0 + .133N
	556 CPI	Read	20.5 + CN	10.1	20.5 + .044N	10.4 + .044N
		Write	20.3 + CN	15.3	20.3 + .044N	5.0 + .044N
729 II/IV	200 CPI	Read	11.1 + CN	0.6	11.1 + .061N	10.5 + .061N
		Write	12.1 + CN	4.6	12.1 + .061N	7.5 + .061N
	556 CPI	Read	10.7 + CN	0.2	10.7 + .017N	10.5 + .017N
		Write	11.7 + CN	4.2	11.7 + .018N	7.5 + .018N
729 V	800 CPI	Read	10.7 + CN	0.2	6.8	6.6
		Write	11.7 + CN	4.2	11.5	7.3
729 IV/VI	200 CPI	Read	7.1 + CN	0.4	7.1 + .038N	6.7 + .038N
		Write	8.1 + CN	3.1	8.1 + .038N	5.0 + .038N
	556 CPI	Read	6.8 + CN	0.1	4.5	4.4
		Write	7.8 + CN	2.8	7.7	4.9
729 VI	800 CPI	Read	6.8 + CN	0.1	4.5	4.4
		Write	7.8 + CN	2.8	7.7	4.9

Figure I-79. Summary of Available Processing Time During Magnetic Tape Operations on the 1460 System

Some examples of available processing time are:

1. Reading a 1,000-character tape record written in high density on an IBM 729 II tape unit, with the processing-overlap special feature.

Tape Operation Time	*Total Time (ms)*	*Time Available for Processing (ms)*
Start Time	10.5	10.5
Record Time	24.0	17.0
Remaining TAU Interlock Time	.2	.2*
Total Time	34.7	27.7

*Assumes that the tape error indicator is tested at the proper time.

2. Writing a 1,000-character tape record in high density on an IBM 7330 tape unit, with the processing-overlap special feature.

Tape Operation Time	*Total Time (ms)*	*Time Available for Processing (ms)*
Start Time	5.0	5.0
Record Time	50.0	44.0
Remaining TAU Interlock Time	15.3	15.3*
Total Time	70.3	64.3

*Assumes that the tape error indicator is tested at the proper time.

3. Writing a 1,000-character tape record in high density (800 CPI) on an IBM 729 VI tape unit, with the processing-overlap special feature.

Tape Operation Time	*Total Time (ms)*	*Time Available for Processing (ms)*
Start Time	5.0	4.9
Record Time	11.1	—
Remaining TAU Interlock Time	2.8	2.8*
Total Time	18.9	7.7

*Assumes that the tape error indicator is tested at the proper time.

Punch-Feed Read Control (1401, 1460)

This feature is required when the punch-feed-read special feature is installed on the IBM 1402 Card Read-Punch, Model 1 or Model 3. The feature provides the controlling circuitry necessary for the proper operation of the punch-feed-read special feature. For detailed information on the feature, refer to the IBM *1402 Card Read-Punch*, A24-3072.

Read-Compare Adapter (1401)

This feature provides the controlling circuitry for the read-compare feature on the IBM 1404 Printer. For detailed information, refer to the IBM *1404 Printer*, A24-1446.

Read-Punch Release (1401, 1460)

With this feature, it is possible to release the read-start-time and punch-start-time interlocks that normally occur during card-read and card-punch cycles, thus providing more processing time during input-output operations.

Start Read Feed

Instruction Format.

Mnemonic	*Op Code*
SRF	8

Function. This instruction works in conjunction with the read-release special feature. It releases the interlock that occurs during read start time, and permits a gain of 21 milliseconds of processing time between card-read cycles. Also, it activates the card-read feed and moves the next card into reading position.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Note. After the START READ FEED instruction is executed, a READ A CARD instruction must be given before the reader is ready to read the 9-row of the card. If the READ CARD instruction comes too late, then the card is not read, and feeds into the NR pocket, and the machine stops. The reader light comes ON, and the I-address register is at the location of the instruction following the one on which read release time was overextended. To ensure optimum processing time, a START FEED READ instruction should follow the READ CARD instruction, within 10 ms, if continuous card reading is desired. Then the machine is ready to accept the next read instruction on the following cycle.

START READ FEED instructions can also be given in cases other than those that cause continuous card feeding, provided that a READ A CARD instruction follows within the next 21 milliseconds. For this reason subroutines that can be executed after the START READ FEED instruction has been given should be timed to determine whether a READ CARD instruction is necessary in the subroutine as well as in the main routine.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	Ap	Bp

Example. Release the interlock on the card reader and feed a card (Figure I-80).

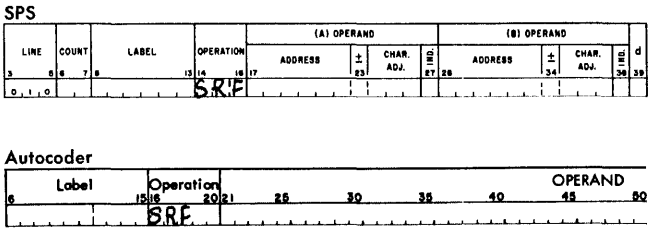


Figure I-80. Start Read Feed

Start Punch Feed

Instruction Format.

Mnemonic	*Op Code*
SPF	9

Function. This instruction works in conjunction with the punch-release special feature. It releases the interlock that occurs during punch start time, and allows a gain of 37 milliseconds of processing time between card punch cycles. It also activates the card feed, and moves the card into punching position.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Note. After a START PUNCH FEED instruction is executed, a PUNCH A CARD instruction must follow before the machine is ready to punch the 12-row of the card. If no PUNCH CARD instruction is interpreted, the card feeds past the punch station without being punched and the machine stops. (The card punched on the previous cycle is not checked.) The I-address register is at the location of the instruction following the one on which punch release time was overextended. For this reason, if cards are to be punched every cycle, a START PUNCH FEED instruction should be given within 22 ms after the PUNCH CARD instruction, so that the machine is ready to punch the next card on the following punch cycle. If cards are not to be punched every cycle, a PUNCH CARD instruction should always follow a START PUNCH FEED instruction within 37 ms to ensure proper machine operation.

When the punch-release special feature (Op code 9) is used in conjunction with punch-read-feed special feature (Op code 4R), the 9 Op code must have an R d-modifier. The R d-modifier activates the punch-feed-read brushes.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	Ap	Bp

Example. Release the punch interlock, and feed a card (Figure I-81).

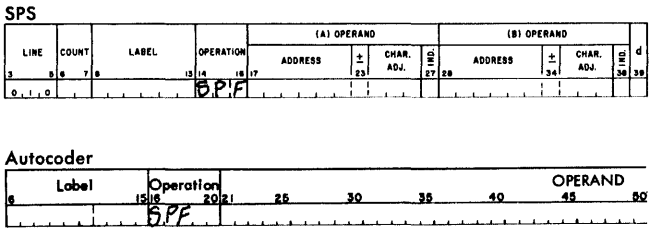


Figure I-81. Start Punch Feed

Scan Disk (1460)

The scan-disk special feature provides an automatic search of 1311 and/or 1301 (Models 11, 12, 21, 22) disk data for a specific identifier or conditions predetermined by the program.

Scan Disk

Instruction Format.

Mnemonic		*Op Code*	*A-address*	*B-address*	*d-character*
SPS	MU	M or L	7	xxx	W
	LU		%F8		
			9		

A SDL (core contents ≦ disk-record contents)
SDE (core contents = disk-record contents)
SDH (core contents ≧ disk-record contents)
SDLW (core contents ≦ disk-record contents — word marks)
SDEW (core contents = disk-record contents — word marks)
SDHW (core contents ≧ disk-record contents — word marks)

Function. This instruction compares a specified search argument in core storage (factor B) to the records within a specified group of sectors in disk storage (factor A).

The A-address units position controls the operation. A 7 in the units position specifies a scan operation that stops when the search argument in core storage is either less than (B < A), or equal to (B = A), a record in the specified section of disk storage. An 8 specifies a scan operation that stops when the search argument in core storage is equal to (B = A), a record in the specified section of disk storage. A 9 specifies a scan operation that stops when the search argument in core storage is either higher than (B > A), or equal to (B = A), a record in the specified section of disk storage. (The operation also stops when the end of the cylinder is reached, or when the sector count reaches zero.)

The B-address of the instruction specifies the high-order core-storage position of the disk-control field that specifies the starting address in disk storage. The record area associated with the disk-control field contains the search argument. The search argument must be placed in the same positions of the core-storage record as it appears in the disk-storage record. Skip codes ($) are used in those positions of the core storage read-in that are not a part of the search argument (Figure I-82). The search argument can be variable in length, but must be no longer than 99 characters. The last character of a sector (100th) cannot be included as part of the search argument. The units positions of the search argument should be followed by a group-mark with a word-mark that signals the end of the search argument.

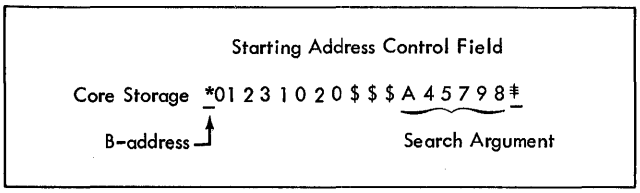


Figure I-82. Record in Core Storage for Scan Operation

Scanning begins at the disk record specified by the B-address and ends

1. when the specified comparison is found. The sector-count field may, or may not, be all zeros at this time.

2. when the operation reaches the end of a cylinder. The sector-count field may, or may not, be all zeros at this time.

3. when the sector-count field is reduced to all zeros.

Word Marks. A group-mark with a word-mark must be set one position to the right of the last character of the search argument.

Timing. $T = .006\ (L_I + 1) + 2N_S +$ disk rotation. 400 ms is the maximum time for scanning one 1311 cylinder (200 sectors); 1,332 ms is the maximum time for scanning one 1301 cylinder (800 sectors).

Note. The result of the scan is determined by testing the high, low, or equal compare indicators with the BRANCH IF INDICATOR ON instruction.

The scan operation can be performed only on disk records written in sector format.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	B + 6	$B + 11 + L_F$

Example. Scan disk storage for an equal compare beginning at sector-address 012510 and continue scan-

ning until the record with part number A24537 is found. The disk-control field is located in the high-order positions of the area of core storage labeled SCANAR (966-974), Figure I-83.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND	d
3 5	6 7	8 13	14 16	17 23			27	28 34			38	39
0 1 0			MU	%F8				SCANAR				W

Autocoder

Label	Operation	OPERAND
6 15	16 20 21	25 30 35 40 45 50
	SDE	SCANAR

Assembled Instruction: M %F8 966 W

Figure I-83. Scan Disk Equal

Seek-Overlap Adapter (1460)

This feature is required when the seek-overlap feature is installed on the IBM 1311 Disk Storage Drives attached to a 1460 system. This feature provides some of the circuitry necessary for proper operation of the seek-overlap feature.

Selective-Tape-Listing Control (1401, 1460)

This feature is required when the selective-tape-listing special feature is installed on the IBM 1403 Printer, Model 1, 2, or 3, and it provides the controlling circuitry necessary for proper operation of the selective-tape-listing feature.

Sense Switches (1401)

Six sense switches make it possible to control the stored program from the 1401 console (Figure I-84). The BRANCH IF INDICATOR ON instruction expands to allow the program to test each switch.

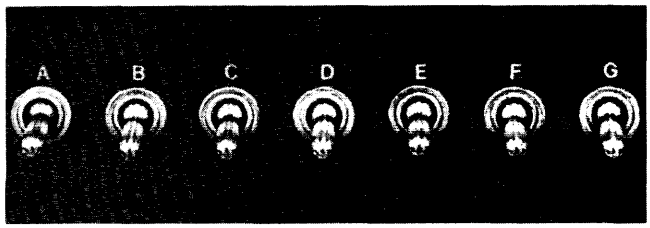


Figure I-84. Sense Switches

For example, a sense switch is turned ON if printing only is required for a given job. If the switch is OFF, the data is to be printed and punched.

The program can be written to interrogate the setting of this sense switch, to determine whether the data is to be printed and punched or just printed. The setting of a sense switch should not be changed while the system is operating.

Branch if Indicator On

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS B	B	xxx	x
A BIN + d-character			

Function. This instruction tests the position of the toggle switch specified by the d-character. If the switch is in the ON position, the next instruction is taken from the I-address. If it is OFF, the program continues with the next sequential instruction.

d-character	*Branch to the I-address if:*
B	Sense switch B is on
C	Sense switch C is on
D	Sense switch D is on
E	Sense switch E is on
F	Sense switch F is on
G	Sense switch G is on

Word Marks. Word marks are not affected.

Timing.

No branch:

$T = N\ (L_I + 1)$ ms.

Branch (without indexing):

$T = N\ (L_I + 1)$ ms.

Branch (with indexing):

$T = N\ (L_I + 2)$ ms.

Note. Sense switch A (last card switch) is standard in all systems equipped with an IBM 1402 Card Read-Punch. When the last card passes the second reading brushes, and switch A is in the ON position, the last-card indicator is turned ON and a B xxx A instruction causes a branch to the I-address. It is turned OFF; on a run-in; if the switch is physically turned OFF; or if the start reset key is pressed.

Address Registers After Operation.

	I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
No Branch	NSI	BI	d*bb*
Branch (without indexing)	NSI	BI	Blank
Branch (with indexing)	NSI	BI	NSI

Example. Branch to the subroutine labeled CLWARE (beginning at 0585) if sense switch G is ON (Figure I-85).

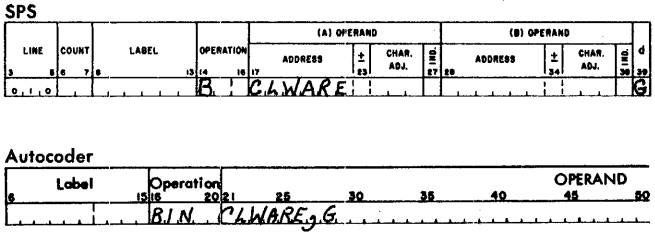


Figure I-85. Branch if Indicator (Sense Switch G) On

Serial Input/Output Adapter (1401, 1460)

This special feature is required to attach these components to the 1401 and 1460 systems:

1009 Data Transmission Unit
1011 Paper Tape Reader
1012 Tape Punch
1412/1419 Magnetic Character Reader
1418 Optical Character Reader
1428 Alphameric Optical Reader
7710 Data Communication Unit (1401 only)
Direct Data Channel
7040/7044 System (1401 only)

One adapter can be used for any of these attachments, but only one attachment can be connected at a time.

Space Suppression (1401; Standard on 1460)

This feature provides program control for space suppression on the printer attached to the system. The character S is used as a d-modifier character with any WRITE instruction (operation codes 2, 3, 6, 7). The S prevents the automatic spacing operation after the next print operation. The next line printed appears on the same line as the line printed by the SPACE SUPPRESS instruction—2 S.

Tape Intermix (1401, 1460)

This special feature makes it possible to have 729 II, IV, V, VI and 7330 magnetic tape units in any combination, connected to the same 1401 and/or 1460 system at one time. Thus, the tape units best suited for the job to be done can be used. Some jobs may require a pair of high-speed tape units (729 IV's) while the remainder of the operation can be performed on a slower unit (7330).

Now, any combination of tape units can be provided to best suit the need of the application, thereby greatly increasing the flexibility of the IBM 1401 and 1460 Data Processing Systems to meet the requirements of any application.

Track Record (1460)

The track-record special feature provides for reading or writing an entire disk track with or without the track address. For IBM 1311, a single, 6-digit address is used, followed by 2,980 characters (2,543 for IBM 1301) in the move mode and 2,682 characters (2,261 for IBM 1301) in the load mode. Track records can be used for storing programs, tables, blocked records, and other data requiring a single large storage block.

When this feature is installed on the system, it provides the track-record function to all the attached 1301 and 1311 drives.

Read Disk-Track Record

Instruction Format.

	Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS	MU	M or L	%F2	xxx	R
	LU (word marks)				
A	RDTR				
	RDTRW (word marks)				

Function. This instruction causes data to be read from a disk track into core storage. The digit 2 in the A-address (%F2) specifies that a track-record operation is to be performed. From the 1311 disk track

2,980 characters (2,543 for 1301) are read in *move* mode or 2,682 characters (2,261 for 1301) in *load* mode. The additional characters read are accounted for by using the normal gap between disk sectors and the sector-address positions. Reading from the disk is stopped by a group-mark with a word-mark in core storage.

Reading from the track begins following the address specified by the core-sector address. This address is located at the beginning of the track, directly after the index pulse.

The core-sector address field in core storage is not modified, but the sector-count field in core storage is reduced by one as the track is read. The sector-count field must be set at 001 before the operation begins, so that reducing it by one can signal an end-of-operation (000 in sector-count field).

The B-address specifies the high-order position in core storage of the disk-control field, and the area in storage reserved for data read from the disk track.

The R in the d-character position signifies that this is a read operation.

Word Marks. A group-mark with a word-mark must be one position to the right of the last position reserved in core storage for the track record. If a group-mark with a word-mark is detected before reading of the track is completed, the wrong-length record and any-disk condition indicators turn on and reading stops. For the 1311, the position of the group-mark with a word-mark is determined by adding 2,991 (2,554 for IBM 1301) to the B-address.

1311 Timing. T = .006 (L_I + 1) + 40 ms + disk rotation.*

*42 ms is maximum time for disk rotation.
22 ms is average time for disk rotation.
2 ms is minimum time for disk rotation.

1301 Timing. T = .006 (L_I + 1) + 33.3 ms + disk rotation.*

*35 ms is maximum time for disk rotation.
18.3 ms is average time for disk rotation.
1.7 ms is minimum time for disk rotation.

Note. Track-record read operations can be performed only on a track written with a TRACK RECORD instruction.

Before reading starts, an automatic check is made of the record address in storage with the record address on the disk. If the addresses are not the same, the unequal-address compare and any-disk condition indicators are turned on, and the data in storage cannot be read from the disk.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*(1311) B-Add. Reg.*	*(1301) B-Add. Reg.*
NSI	B + 6	B + 11 + 2980 or B + 11 + 2682	B + 11 + 2543 or B + 11 + 2261

Example. Read disk track 012540 in core storage beginning at location 976 (area is labeled TRSEC1). The high-order position of the disk address is in the first ten positions of the label (966-975), Figure I-86.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND	d
3	6	7 8	13 14	16 17		23	27	29	34		38	39
0 1 0			MU	%F2				TRSEC1				R

Autocoder

Label	Operation	OPERAND
6	15 16 20 21	25 30 35 40 45 50
	RDTR	TRSEC1

Assembled Instruction: M %F2 966 R

Figure I-86. Read Disk Track Record

Read Disk-Track Record with Address

This instruction is the same as READ DISK-TRACK RECORD with these exceptions:

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS MU	M or L	%F@	xxx	R
LU (word marks)				
A RDTA				
RDTAW (word marks)				

Function. The @ in the A-address (%F@) specifies that the address of the track record in disk storage is also read into core storage with the data on the disk track.

When a disk-track record operation is initiated, an automatic check is made of the record address in storage with the record address on the disk. If the addresses are equal, reading begins immediately following the index pulse on the disk track. (The index pulse signals the system that the beginning of a track is about to come under the access assembly.) The track-record address in the high-order position of the disk data field in core storage is written in the first sector-address position after the index pulse.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*(1311) B-Add. Reg.*	*(1301) B-Add. Reg.*
NSI	B + 9	B + 11 + 2986 or B + 11 + 2988	B + 11 + 2549 or B + 11 + 2267

Example. Read the address and data from disk track 012540 into core storage beginning at location 476

(area is labeled TRECAD). The high-order position of the disk-control field is in the first ten positions of the label (466-475), Figure I-87.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	d
3 5	6 7	8 13	14 16	17 23		27	28 34		36	39
0 1 0			MU	%F@			TRECAD			R

Autocoder

Label	Operation	OPERAND
6 15	18 20	21 25 30 35 40 45 80
	RDTR	TRECAD

Assembled Instruction: M %F@ 466 R

Figure I-87. Read Disk Track Record with Address

Write Disk-Track Record

Instruction Format.

	Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS	MU	M or L	%F2	xxx	W
	LU (word marks)				
A	WDTR				
	WDTRW (word marks)				

Function. This instruction causes data from core storage to be written on a disk track. The digit 2 in the A-address (%F2) specifies that a track-record operation is to be performed. From core storage an entire 1311 disk track of 2,980 characters (2,543 for 1301) is written in *move* mode or 2,682 characters (2,261 for 1301) in *load* mode. The additional characters are accounted for by writing in what is normally the gap between disk sectors and the sector address positions. Writing of a disk track is stopped by sensing a group-mark with a word-mark in core storage and the end of track.

Writing begins at the track address specified by the core-sector address field. This address is located at the beginning of the track, directly after the index pulse.

The core-sector address field in storage is not modified, but the sector-count field in core storage is reduced by one as the track is written. Set the sector-count field to 001 before the operation begins so that reducing it by one can signal an end-of-operation (000 in sector-count field).

The B-address specifies the high-order position in core storage of the disk-control field, and the area in storage where the data to be written on the disk track is stored.

The W in the d-character position signifies that this is a write operation.

Word Marks. A group-mark with a word-mark must be one position to the right of the last character of the data in core storage. The writing of data stops when the end of track is reached on the disk and a group-mark with a word-mark is sensed in core storage. If the group-mark with a word-mark is sensed before the end of track, the remainder of the disk track is filled with data from core storage, and the wrong length record and any-disk condition indicators are turned ON. For the 1311, position of the group-mark with a word-mark is determined by adding 2,991 (2,554 for 1301) to the B-address.

1311 Timing. $T = .006 (L_I + 1) + 40$ ms + disk rotation.*

*42 ms is maximum time for disk rotation.
22 ms is average time for disk rotation.
2 ms is minimum time for disk rotation.

1301 Timing. $T = .006 (L_I + 1) + 33.3$ ms + disk rotation.*

*35 ms is maximum time for disk rotation.
18.3 ms is average time for disk rotation.
1.7 ms is minimum time for disk rotation.

Note. Before writing starts, an automatic check is made of the core-sector address in storage with the record address on the disk. If the addresses are not the same, the unequal-address compare and any-disk condition indicators are turned ON, and the data in storage cannot be written on the disk.

A WRITE DISK CHECK instruction must be performed following a write operation unless an error occurred during the write operation. No other disk-storage operation can be performed until the check of data written on the disk is accomplished.

If the data in core storage contains characters with word marks and the write operation is performed in the move mode, only the CBA 8421 portion of the character is written on the disk (the word mark is ignored).

Disk tracks adjacent to, but not above or below, a disk track written with the WRITE DISK-TRACK RECORD instruction must be either unused or set up as a *track record.* If the adjacent tracks are written using WRITE DISK SECTOR or WRITE DISK SECTOR WITH ADDRESSES instructions, interference occurs to the track-record data stored in what is normally the gap between sectors.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*(1311) B-Add. Reg.*	*(1301) B-Add. Reg.*
NSI	B + 6	B + 11 + 2980 or B + 11 + 2682	B + 11 + 2543 or B + 11 + 2261

Example. Write a disk-track record from the data in the core-storage area labeled TRSEC1 (the first position of data is at 976). The high-order position of the disk address is in the first ten positions of the label (966-975), Figure I-88.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	d
			MU	%F2			TRSEC1			W

Autocoder

Label	Operation	OPERAND
	WDTRW	TRSEC1

Assembled Instruction: M %F2 966 W

Figure I-88. Write Disk Track Record

Write Disk-Track Record with Address

This instruction is the same as WRITE DISK-TRACK RECORD with these exceptions:

Instruction Format.

	Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
SPS	MU	M or L	%F@	xxx	W
	LU (word marks)				
A	WDTA				
	WDTAW (word marks)				

Function. The @ in the A-address (%F@) specifies that the address of the track record in core storage is also written on the disk.

When a disk track-record operation is initiated, an automatic check is made of the record address in storage with the record address on the disk. If the addresses are equal, writing begins immediately following the index pulse on the disk track. (The index pulse signals the system that the beginning of a track is about to come under the access assembly.) The track-record address in the high-order position of the disk data field in core storage is written in the first sector-address position after the index pulse.

Word Marks. A group-mark with a word-mark must be one position to the right of the last character of data in core storage. The writing of data stops when the end of track is reached on the disk and a group-mark with a word-mark is sensed in core storage. If the group-mark with a word-mark is sensed before the end-of-track, the remainder of the disk track is filled with valid blanks (C-bit), and the wrong-length record, and any-disk condition indicators are turned on. Processing is interlocked until the end of the sector. For the 1311, the position of the group-mark with a word-mark is determined by adding 2,997 (2,560 for 1301) to the B-address.

Note. Before writing starts, an automatic check is made of the core-sector address in storage with one of the sector addresses on the pack. If the address is not found the unequal-address compare and any-disk condition indicators are turned on, and the data in storage cannot be written on the disk.

A WRITE DISK CHECK instruction must be performed following a write operation. No other disk-storage operation can be performed until the check of data written on the disk is accomplished.

If the data in core storage contains characters with word marks and the write operation is performed in the move mode, only the CBA 8421 portion of the character is written on the disk (the word mark is ignored).

Disk tracks adjacent to, but not above or below, a disk track written with the WRITE DISK TRACK RECORD or WRITE DISK TRACK RECORD WITH ADDRESS instructions must be either unused or set up as a track record. If the adjacent tracks are written using WRITE DISK SECTOR or WRITE DISK SECTOR WITH ADDRESSES instructions, interference occurs to the track-record data stored in what is normally the gap between sectors.

The write-address key on disk-storage-drive zero must be ON to perform this operation.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*(1311) B-Add. Reg.*	*(1301) B-Add. Reg.*
NSI	B + 9	B + 11 + 2986 or B + 11 + 2688	B + 11 + 2549 or B + 11 + 2267

Example. Write a disk-track record with its new address from the data in the core-storage area labeled TRECAD (the first position of the address is at 476). The high-order position of the disk-control field is in the first ten positions of the label (466-475), Figure I-89.

SPS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	d
			MU	%F@			TRECAD			W

Autocoder

Label	Operation	OPERAND
	WDTA	TRECAD

Assembled Instruction: M %F@ 466 W

Figure I-89. Write Disk Track Record with Address

Translate (1460)

This special feature provides the data processing system with the capability of fast, flexible translation of codes to and from the code of the system.

The feature uses stored-program instructions to initiate the code translation and subsequent record movement. (IOCS for this feature is described in the SRL publication *IOCS for the 1440/1448: Specifications*, Form C24-3024.) One translate instruction translates a complete record, moving left to right as it replaces each record character with a character from a translate table in core storage until a group-mark with a word-mark is detected in the field being translated.

Each code translation requires a table in storage beginning at an even-hundreds address for TRANSLATE-WITH-WORD-MARKS and at *any* hundreds address for TRANSLATE-WITHOUT-WORD-MARKS. The number of various code translations that can be handled at one time in a system is limited only by the core storage available for tables.

The LOAD RECORD instruction (included in the translate feature) moves characters and word marks from an A-field to a B-field, moving left to right up to and including an A-field group-mark with a word-mark. Original B-field word marks are cleared.

Translate with Word Marks

Op Code	*A-address*	*B-address*	*d-character*
T	xxx	x00	>

The TRANSLATE-WITH-WORD-MARKS instruction consists of T (C-A-2-1-WM) for the operation code, a 3-character A-address representing the initial address of the record to be translated, a 3-character B-address representing the initial address of the translate table, and a d-character with a bit configuration of 8-4-2.

The record to be translated must end with a group-mark with a word-mark. The initial address of the translate table is restricted to any available *even*-hundreds address such as 200, 400, 600, 800, or 1000. Two table sizes can be accommodated: a 78-character table provides as many as 64 usable positions, and a 156-character table provides as many as 128 usable positions. (Figures I-90 and I-91 show that the digits 8

Baudot Character	Enters 1440 or 1460 Storage as	Generates Translate-Table Address	Baudot Character	Enters 1440 or 1460 Storage as	Generates Translate-Table Address
T	2	X02	Z	CB2	X42
5 (FIGS-T)	C21	X03	"(FIGS-Z)	B21	X43
CR	4	X04	D	CB4	X44
CR (FIGS)	C41	X05	$(FIGS-D)	B41	X45
0	C42	X06	B	B42	X46
9 (FIGS-0)	421	X07	?(FIGS-B)	CB421	X47
SPACE	8	X10	S	CB8	X50
FIGS-SPACE	C81	X11	Bell (FIGS-S)	B81	X51
H	C82	X12	Y	B82	X52
#(FIGS-H)	821	X13	6(FIGS-Y)	CB821	X53
N	C84	X14	F	B84	X54
,(FIGS-N)	841	X15	!(FIGS-F)	CB841	X55
M	842	X16	X	CB842	X56
.(FIGS-M)	C8421	X17	/(FIGS-X)	CBA421	X57
LF	A	X20	A	CBA	X60
FIGS-LF	CA1	X21	-(FIGS-A)	BA1	X61
L	CA2	X22	W	BA2	X62
)(FIGS-L)	A21	X23	2(FIGS-W)	CBA21	X63
R	CA4	X24	J	BA4	X64
4 (FIGS-R)	A41	X25	'(FIGS-J)	CBA41	X65
G	A42	X26	FIGS	Deleted From Input by Adapter	
&(FIGS-G)	CA421	X27			
I	CA8	X30	U	BA8	X70
8 (FIGS-I)	A81	X31	7(FIGS-U)	BA81	X71
P	A82	X32	Q	CBA82	X72
0 (FIGS-P)	CA821	X33	1(FIGS-Q)	BA821	X73
C	A84	X34	K	CBA84	X74
:(FIGS-C)	CA841	X35	((FIGS-K)	BA841	X75
V	CA842	X36	LTRS	Deleted From Input by Adapter	
;(FIGS-V)	A8421	X37			
E	B	X40	BLANK	C	X00
3(FIGS-E)	CB1	X41	FIGS-BLANK	1	X01

(Note: X in table address represents any hundreds-position digit.)

Figure I-90. Baudot Code to Translate-Table Address

and 9 are not used in the units or hundreds position of any generated address of the translate table.) The 156-character table consists of the 78-character table, beginning at an even-hundreds address (for example 200-277), combined with another 78-character table, beginning at the next sequential hundreds address (300-377).

Characters and word marks from the translate table (initial address specified by B-address) replace the characters and word marks in the record being translated, beginning at the address specified by the A-address of the translate instruction.

The TRANSLATE-WITH-WORD-MARKS instruction interprets word marks in the A-address field as DATA or SHIFT bits, and the word marks actually take part in the translate function.

The translate instructions are interruptible.

The B-field address (initial translate-table address) cannot be indexed.

Translate Without Word Marks

Op Code	*A-address*	*B-address*
T	xxx	x00

The TRANSLATE-WITHOUT-WORD-MARKS instruction format is the same as the TRANSLATE-WITH-WORD-MARKS instruction, but without a d-character. It functions the same, with these exceptions:

1. One table size can be accommodated: the 78-character table, which provides as many as 64 usable positions.
2. A-field word marks do not take part in the actual translation, and are *not* altered by the translation.

General Description of Translate

The program assigns locations and contents in the translate tables depending on the desired translation. Each position of the translate table contains the BCD character to which a particular character is to be translated. For each character to be translated, the translate feature automatically selects the appropriate position of the translate table. The contents of that position replaces the character in the A-field. The contents of the translate table are undisturbed.

The translate-table characters must contain odd parity. The input/output device, channel or adapter (for example, the IBM 1448 Transmission Control Unit) performs any necessary parity conversion for input and output. The tables also must contain required SHIFT bits or shifted codes. The adapter detects SHIFT-bit transitions, generates the required SHIFT character, and removes the SHIFT bit.

1440 or 1460 BCD Character	Generates Translate-Table Address	1440 or 1460 BCD Character	Generates Translate-Table Address
BLANK	X00	–	X40
1	X01	J	X41
2	X02	K	X42
3	X03	L	X43
4	X04	M	X44
5	X05	N	X45
6	X06	O	X46
7	X07	P	X47
8	X10	Q	X50
9	X11	R	X51
0	X12	I	X52
#	X13	$	X53
@	X14	*	X54
:	X15	)	X55
>	X16	;	X56
√	X17	Δ	X57
(b)	X20	&	X60
/	X21	A	X61
S	X22	B	X62
T	X23	C	X63
U	X24	D	X64
V	X25	E	X65
W	X26	F	X66
X	X27	G	X67
Y	X30	H	X70
Z	X31	I	X71
ǂ	X32	?	X72
,	X33	.	X73
%	X34	□	X74
=	X35	(	X75
'	X36	<	X76
(+++)	X37	ǂ	X77

Note: Word marks with the BCD characters will generate the same sequence of addresses at (X+1)00, (X+1)01, etc.

Figure I-91. BCD Character Generates Translate-Table Address

Figure I-90 is an example of how a code (Baudot telegraph code in this case) enters the system and the addresses that are generated (assuming Baudot 1-2-3-4-5 bits respectively, are equal to BCD bits B-A-8-4-2 with the 1-bit designating figures shift).

Figure I-91 shows the BCD character and the table address generated.

Translation of each character is accomplished in a 3-cycle sequence (first A-cycle, B-cycle, and second A-cycle). Refer to Figure I-92 while reading the sequence described here. Here again Baudot code is used as an example.

Although the examples show the function of the translate with incoming data, it functions the same way for outgoing data using another translate table.

First A-Cycle

The first A-cycle generates the appropriate translate-table address specified by the character to be translated.

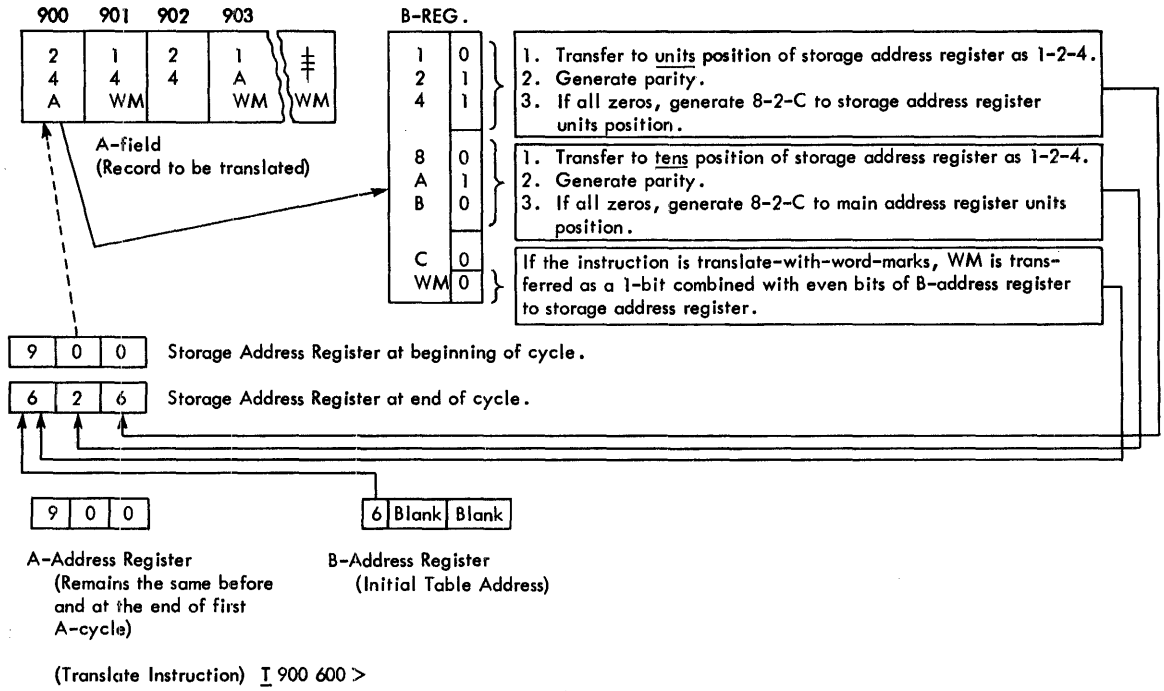


Figure I-92. First A-cycle of Translate Sequence

The translate instruction acts upon all the characters of the A-field. Figure I-92 shows how the instruction T 900 600 (as an example) translates one character. In this case the first character of the A-field is used as the example. The contents of position 900 has a bit configuration of 2-4-A (Baudot-code G).

The first A-cycle moves this first character in the A-field to the B-register. From the B-register, the operation moves the 1-2-4 bits of this character (with proper parity) to the *units* position of the storage-address register together with the A-B bits from the units position of the B-address register. A-B bits in the *units* position of the generated address designate core-storage blocks over 3999. In Figure I-92 the 2-4 bits produce a 6 in the *units* position of the storage-address register. If the 1-2-4- bits are all blanks (no-bits), the operation generates an 8-2-C bit configuration (zero) into the *units* position of the storage-address register.

The first A-cycle also moves the 8-A-B bits (interpreted as 1-2-4 bits) of the same character (with proper parity) to the *tens* position of the storage-address register. In Figure I-92 the A-bit, which becomes a 2-bit, produces a 2 in the *tens* position of the storage-address register. Again, if the 8-A-B bits are all blanks, the operation generates an 8-2-C bit configuration (zero) into the *tens* position of the storage-address register.

If the instruction is a TRANSLATE-WITH-WORD-MARKS, a word-mark bit in the B-register is interpreted as a 1-bit and is combined with the bits already selected for transfer to the hundreds position of the storage-address register. This adds 1 to the hundreds position of the storage-address register and thus generates an address from the second half of a 156-character table. If the table's base address was 600 (as in Figure I-92) the generated address would be 726. Because in the example the B-register has no word mark, the generated address is 626.

A TRANSLATE-WITHOUT-WORD-MARKS instruction blocks a word mark in the B-register so it takes no part in the generation of the address.

B-Cycle

The B-cycle of the operation reads a character or character with word mark out of a specified translate-table address in storage into the A-register. It uses the address generated in the first A-cycle. The translate table is unaltered.

Second A-Cycle

The second A-cycle of the operation again reads out the character being translated and replaces it in core storage with the contents of the A-register, which contains the character read out of the translate table on the B-cycle. The original A-field character is destroyed.

A TRANSLATE-WITH WORD-MARKS instruction destroys original A-field word marks but moves word marks from the translate table to the A-field. A TRANSLATE-

WITHOUT-WORD-MARKS instruction regenerates original A-field word marks into the A-field and moves translate-table word marks to the A-field.

Figures I-93 and I-94 show how the translate instruction affects processor-storage areas. No particular codes are specified.

Timing. The formula for the translate-operation execution time (T) for the 1460 is:

$$T = .006\ (L_I + 2 + 3N)\ \text{ms}.$$

N = The number of characters in A-field to be translated.
L_I = 7 for TRANSLATE-WITHOUT-WORD-MARKS
8 for TRANSLATE-WITH-WORD-MARKS

Example: N = 100-character record
T = 1.8 ms

	900	901	902	903	904	905	906	907	908	909	910
1	0	1	0	1	0	0	0	1	0	1	1
2	0	0	1	1	1	1	0	0	0	0	1
4	1	1	1	0	1	0	1	0	1	0	1
8	0	1	0	0	1	1	0	0	1	0	1
A	0	1	0	0	1	1	0	0	1	0	1
B	0	1	0	0	1	1	0	0	1	0	1
C	0	1	1	0	0	0	1	0	0	1	0
WM	0	1	0	1	0	1	1	0	0	1	1

Original A-field (record to be translated)

	600	601	602	603	604	605	606	607	608	674	675	676	677	
									⊠					Translate Tables 600-677
	700	701	702	703	704	705	771	772	773	774	775	776	777	700-777

(Coded characters are assigned by the program)

⊠ Represents an unused table address (X08, X09, X18, X19, etc.)

	900	901	902	903	904	905	906	907	908	909	910
	©	©	©	©	©	©	©	©	©	©	
	6	7	6	7	6	7	7	6	6	7	‡
	0	7	0	0	7	7	0	0	7	0	†
	4	5	6	3	6	2	4	1	4	1	WM

Resultant A-field. © 604 means the contents of 604 (character and word mark).

I 900 600 >

Figure I-93. Translate-with-Word-Marks

Load Record

The translate feature includes a special move instruction that moves characters and associated word marks in a record from one storage area to another.

Op Code	*A-address*	*B-address*	*d-character*
P	xxx	xxx	>

The LOAD RECORD instruction consists of P (B-4-2-1-WM) for the operation code, a 3-character A-address representing the address of the record to be moved, a 3-character B-address representing the address of the field to which the record is moved, and a d-character with a bit configuration of 8-4-2.

The operation moves characters and word marks in the A-field to the B-field, moving from low-numbered to high-numbered storage position up to and including an A-field group-mark with a word-mark, which stops the operation. Original B-field word marks are cleared.

	900	901	902	903	904	905	906	907	908	909	910
1	0	1	0	1	0	0	0	1	0	1	1
2	0	0	1	1	1	1	0	0	0	0	1
4	1	1	1	0	1	0	1	0	1	0	1
8	0	1	0	0	1	1	0	0	1	0	1
A	0	1	0	0	1	1	0	0	1	0	1
B	0	1	0	0	1	1	0	0	1	0	1
C	0	1	1	0	0	0	1	0	0	1	0
WM	0	1	0	1	0	1	1	0	0	1	1

Original A-field (record to be translated) (Note: Same bit configurations as example translate-with-word-marks.)

	600	601	602	603	604	605	606	607	608	672	673	674	675	676	677
									⊠						

Translate table 600-677 (coded characters are assigned by the program)

⊠ Represents an unused table address (X08, X09, X18, X19, X28, X29, etc.)

	900	901	902	903	904	905	906	907	908	909	910
	©	©	©	©	©	©	©	©	©	©	
	6	6	6	6	6	6	6	6	6	6	
	0	7	0	0	7	7	0	0	7	0	‡
	4	5	6	3	6	2	4	1	4	1	‡
		WM		WM		WM	WM			WM	WM

Resultant A-field. © 604 means the contents of 604 (character and word mark).

T 900 600

Figure I-94. Translate Without Word-Marks

Transmission Control-Unit Attachment (1460)

This feature is required when the IBM 1448 Transmission Control Unit is attached to an IBM 1460 Data Processing System. For detailed information on the 1448, refer to the IBM *1448 Transmission Control Unit,* Form A24-3010, and IBM *1448 Transmission Control Unit with* IBM *1460 Data Processing System,* Form A24-3050.

800 CPI Feature (1401)

This feature provides the facility to operate 729 v magnetic-tape unit at 800 cpi density with the 1401. The feature includes a tape-densities-option switch on the 1401 console, which is used to select one of the following pairs of densities: 200/556, 200/800, or 556/800. The high or low density selected by the change-density switch on each tape unit operates in conjunction with the pair of densities selected by the tape-densities-option switch.

This feature is standard on an IBM 1460 Data Processing System with an IBM 1461 Input/Output Control, Model 2, and is effective for both the 729 v and 729 VI magnetic-tape units.

IBM 1402 Special Feature Instructions and Timing

Early Card Read (Model 1 Only; Standard on Model 3)

The early-card-read feature for the IBM 1402 Card Read-Punch minimizes the decrease in card-reading speed caused by lengthy processing routines. In such routines, the card-reading mechanism can engage sooner, thus reducing the time between the reading of cards.

The card reader operates at a rated speed of 800 cycles per minute (one cycle every 75 milliseconds). The card reading speed depends on the timing of the READ CARD instructions in the program. To effect continuous card-reading at the rate of 800 cards per minute, a READ CARD instruction must be given within 10 milliseconds after the preceding card has been actually read into core storage (labeled *Processing Time* in Figure I-95).

Normally, if processing time exceeds 10 ms in a basic card-read cycle, the rated card-reading speed decreases. This occurs because of the mechanical structure of the card-read feed. There is only one time during the read cycle when the feeding mechanism can be engaged. If a READ CARD instruction is given too late (processing time exceeds 10 ms), a card-read cycle is skipped, thus reducing the input speed from 800 to 400 cards per minute (Figure I-96). Similarly, if the time required for processing exceeds 85 ms, two read cycles are skipped, and the input speed is reduced to 266 cards per minute.

The early-card-read feature provides two additional points (clutch points) where the feeding mechanism can engage. When processing time between cards exceeds 10 ms, the feed mechanism can engage 50 ms sooner than before. The time between card feeding is reduced to 100 ms rather than 150 ms. Instead of a 50 percent reduction in the rated speed (to 400 cpm), there is only a reduction of 25 percent (to 600 cpm) as shown in Figure I-97.

Similarly, when processing time between cards exceeds 35 ms, the feed mechanism can engage 25 ms sooner than before. The time between card feeding is reduced to 125 ms rather than 150 ms. Instead of a 50 percent reduction in the rated speed (to 400 cpm), there is only a reduction of 40 percent (to 480 cpm) as shown in Figure I-98.

Figure I-99 is a chart showing the card feed cycle times and the resultant cards per minute output.

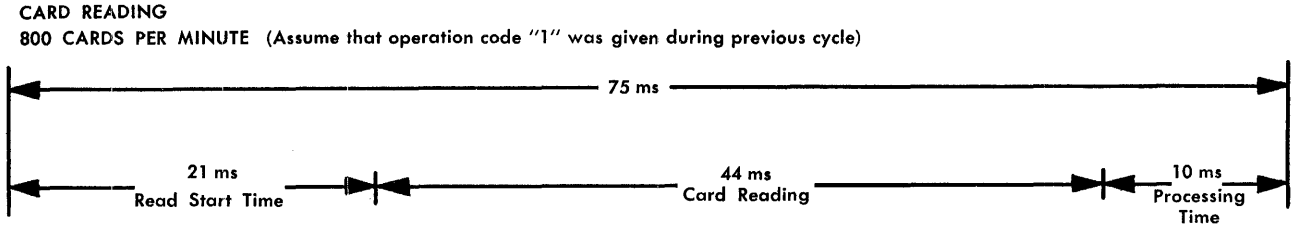


Figure I-95. Card Read Cycle

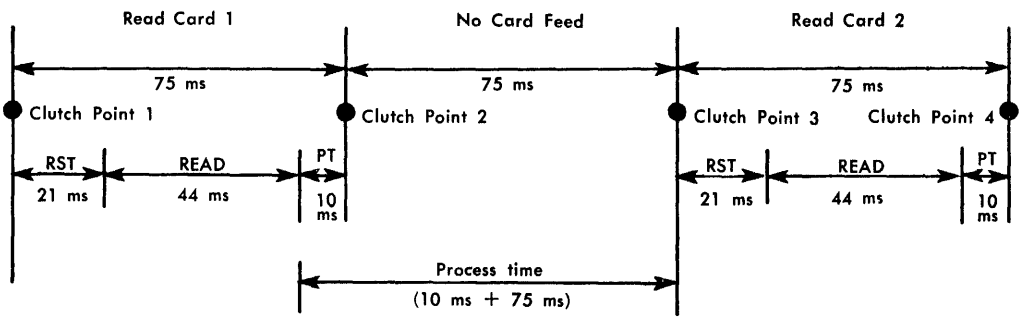


Figure I-96. 400 CPM Operation

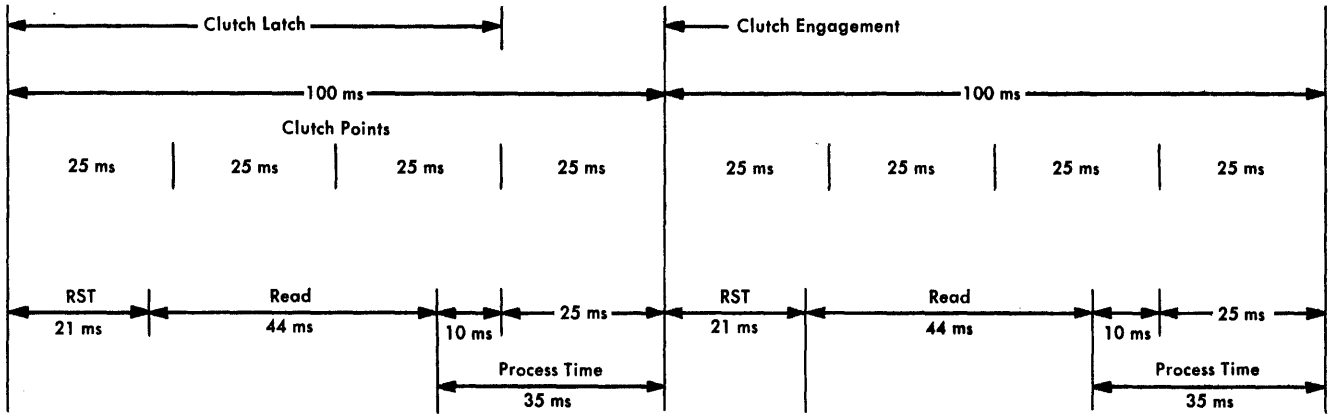


Figure I-97. 600 CPM with Early Card Read

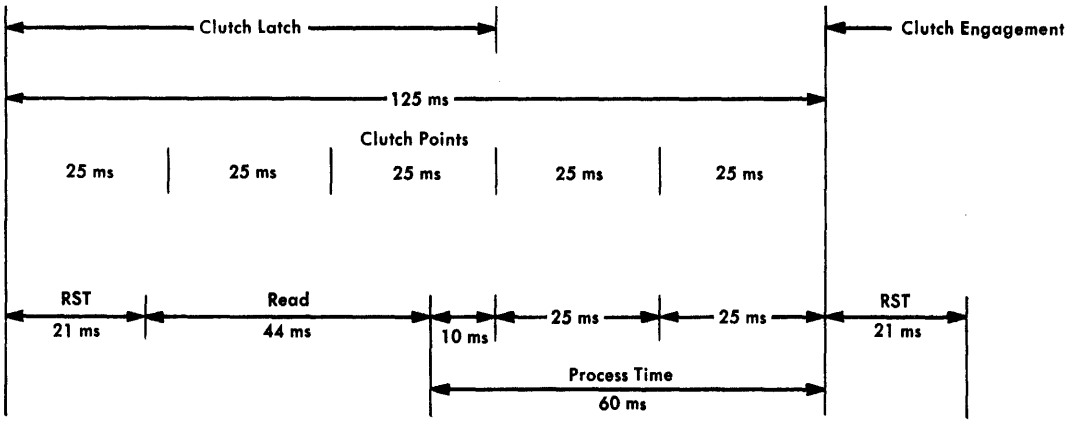


Figure I-98. 480 CPM with Early Card Read

Card Feed Cycle Timing	Cards Per Minute
75 ms	800
100 ms	600
125 ms	480
150 ms	400
175 ms	342
200 ms	300
225 ms	266
250 ms	240

Figure I-99. Card Feed Timings

The hole-count check of prepunched data is begun at the punch-read station and is completed at the punch-check station after punching has occurred.

NOTE: Punching in prepunched columns is acceptable, provided that the resultant character is valid and that the punches read at the punch-feed-read station are not repeated. For example, an X can be punched in a card column that already contains a 2, but punching a K (X and 2 punches) at the punch station if either an X or a 2 was already in the card, results in a hole-count check.

The d-character R activates the punch-feed-read brushes. It can be used with the operation codes PUNCH CARD (4), WRITE AND PUNCH (6), and START PUNCH FEED (9). If the combination instruction READ AND PUNCH (5), or WRITE READ AND PUNCH (7) is given, read and punch errors occur.

Punch Feed Read (Models 1 and 3 Only)

In some applications it is desirable to read information into the system, calculate, and punch the results in the same card from which the input data was read. By using the punch feed read feature, the card at the punch-feed-read station can be read while the card ahead of it is being punched. To permit this type of operation, a special set of 80 reading brushes, called *punch feed read,* is added to the IBM 1402 Card Read-Punch feed, one station ahead of the punch station (Figure I-100). The R d-character specifies that the card is to be read from the punch side of the 1402. The normal read area (storage locations 001-080) receives the information from the punch feed read in the same manner as information is read from the read feed. A validity and a columnar hole-count check is made on each card column read from the punch-feed-read brushes. MLP card codes cannot be read by the punch-feed-read brushes.

The punching operation for machines equipped with punch feed read is the same as in the basic 1401. Storage positions 101-180 are specified as the punch area, and a hole-count check is made the punch brushes.

Read-Punch Feed

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SPS P *A* RF	4	R

Function. When this instruction is used, the punch feed operates and reads the card entering the read station on the punch side. It also causes the card at the punch station to be punched. The R character modifier makes this instruction effective.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms + punch start time (37 ms) and punching time of 184 ms (Figure I-101). (Punch start time can be used for processing if the punch release special feature is installed.)

Note. An additional 3 ms is required in excess of the normal punch time of 181 ms when the punch feed-read-feature is used. Processing time available is 19 ms.

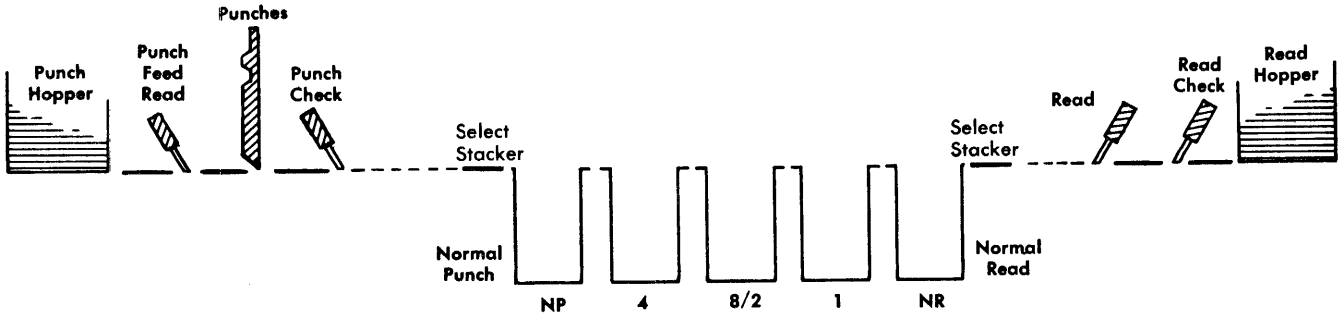


Figure I-100. Punch Feed Read Schematic

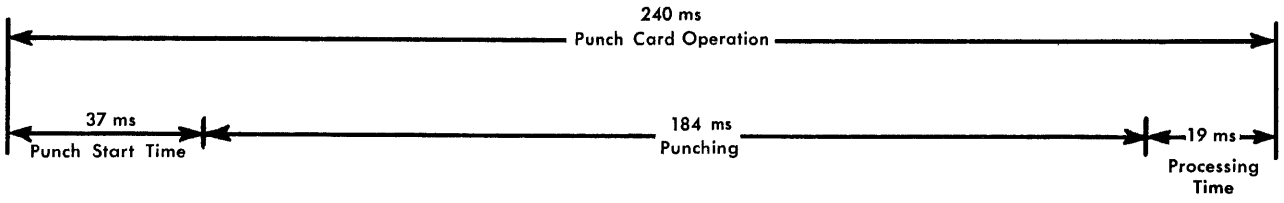


Figure I-101. Punch Timing (Read Punch Feed)

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	*dbb*	181

Example. Read the card at the punch feed read station and punch a card (Figure I-102).

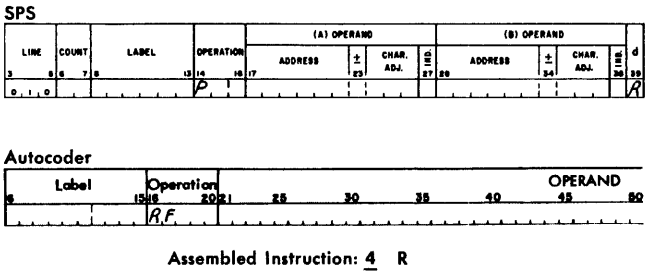


Figure I-102. Read Punch Feed

Read-Punch Feed and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS P	4	xxx	R
A RF	—		

Function. This instruction causes the same function as READ-PUNCH FEED, except that an automatic branch to the I-address is effected.

Word Marks. Word marks are not affected.

Timing. T = N (L_I + 1) ms + punch start time (37 ms) and punching time of 184 ms. (Punch start time can be used for processing if the punch-release special feature is installed.)

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	081

Example. Read the card at the punch-feed-read station, punch a card, and branch to START6 (0598) for the next instruction (Figure I-103).

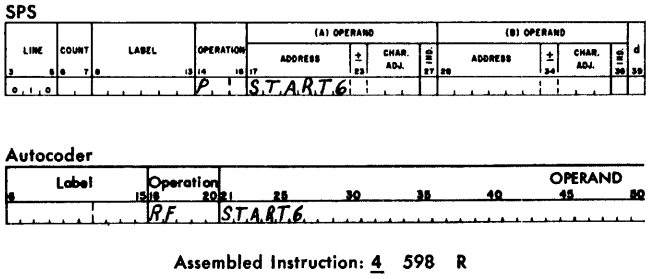


Figure I-103. Read-Punch Feed and Branch

Read-Punch Feed and Write

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SPS WP	6	R
A WRF	—	

Function. This instruction causes the printer to operate and print a line, and the punch unit to read a card, and also causes the card at the punch station to be punched. The d-character R specifies that the card at the punch feed station is to be read. The printer takes priority and operates first, but the signal to start the punch feed read is automatically given before the end of the print operation, so that actual card reading starts soon after the print cycle is complete.

Word Marks. Word marks are not affected.

Timing. T = N (L_I + 1) ms + the timing conditions for print and punch overlap (see *Write and Punch Operation Timing Chart* – Figure E-44). The print operation normally takes 84 ms. Punch start time is 37 ms and the punch reading time is 184 ms. An

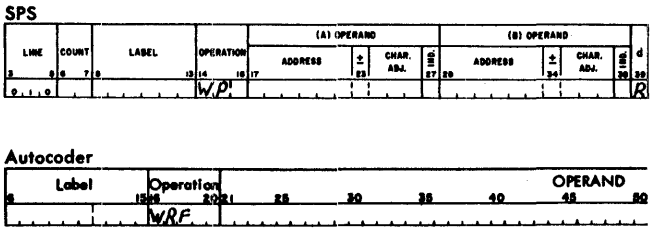
additional 3 ms are added to the normal punching time of 181 ms. Normal processing time available is 19 ms.

Note. If the print-storage special feature is installed in the system, the automatic signal to start the punch-feed-read operation is given shortly after the transfer of data to the print storage area. Thus, additional processing time can be gained by using print storage.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	081

Example. Write a line, read a card at the punch-feed-read station, and punch a card (Figure I-104).



Assembled Instruction: 6 R

Figure I-104. Read-Punch Feed and Write

Read-Punch Feed, Write and Branch

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
SPS WP	6	xxx	R
A WRF			

Function. Same as the READ-PUNCH FEED AND WRITE except that the program branches to the I-address for the next instruction.

Word Marks. Word marks are not affected.

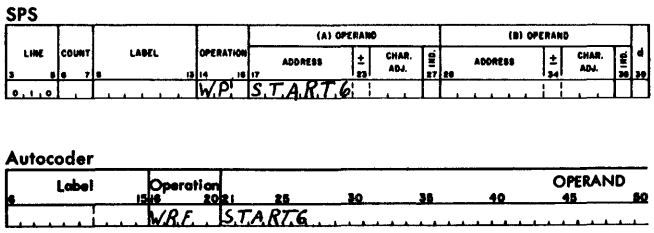
Timing. T = N (L_I + 1) ms + the timing conditions for print and punch overlap (see *Write and Punch Operation Timing Chart* — Figure E-44). The print operation normally takes 84 ms. Punch start time is 37 ms and the punch reading time is 184 ms. An additional 3 ms are added to the normal punching time of 181 ms. Normal processing time available is 19 ms.

Note. If the print-storage special feature is installed in the system, the automatic signal to start the punch-feed-read operation is given shortly after the transfer of data to the print-storage area. Thus, additional processing time can be gained by using print storage.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	BI	081

Example. Print a line, read and punch a card from the punch side of the IBM 1402, and branch to START6 (0895) for the next instruction (Figure I-105).



Assembled Instruction: 6 895 R

Figure I-105. Read-Punch Feed Write and Branch

IBM 1403 Special Feature

Selective-Tape-Listing Feature

The selective-tape-listing feature can be attached to any model of the IBM 1403 Printer so that output results of data processed on a system can be printed on adding-machine style paper tapes.

Operation

An IBM 1403 Printer with the selective-tape-listing feature installed continues to operate at regular 1403 speeds. Each tape is individually linespaced, one line at a time (no skipping or ejecting is possible). Tape is spaced by using modified 1403 CONTROL CARRIAGE Op code (F), when used with a d-character of A through H, which signals a single linespace for the corresponding tape. (The space operation takes place after the next print operation.) The modifier characters and the tapes they control are:

A – Tape 1
B – Tape 2
C – Tape 3
D – Tape 4
E – Tape 5
F – Tape 6
G – Tape 7
H – Tape 8

When a double-width tape is used, two tape linespace instructions are given, using the d-characters corresponding to the positions occupied by the double-width tape. If additional linespacing is wanted, a tape linespace instruction (CONTROL CARRIAGE) Op code and the specific d-character and a PRINT instruction (2) are given. The print operation is a dummy print operation, and the print area in core storage should be clear so that nothing is printed. The linespace operation occurs after the print operation.

To equalize the ribbon wear, the customer can vary the location of the master tape. This can be done by using the same width tape in another location and altering the program (changing the d-character to the character that corresponds to the new location). Tapes can be pulled up manually before tearing them against the tear-bar portion of the tape-advancing mechanism (Figure I-106).

An end-of-tape condition, sensed at the tape-spool tray, stops the printing operation and turns ON the 1403 end-of-forms light.

When programming selective-tape-listing, the program should not select more than four tape-feeding solenoids simultaneously.

Control (Feature Mode Switch)

A feature-mode switch is provided on the 1403. This switch temporarily disconnects the standard carriage-control circuits and activates the tape-feeding circuits when the tape-spool assembly tray is latched in the operating position.

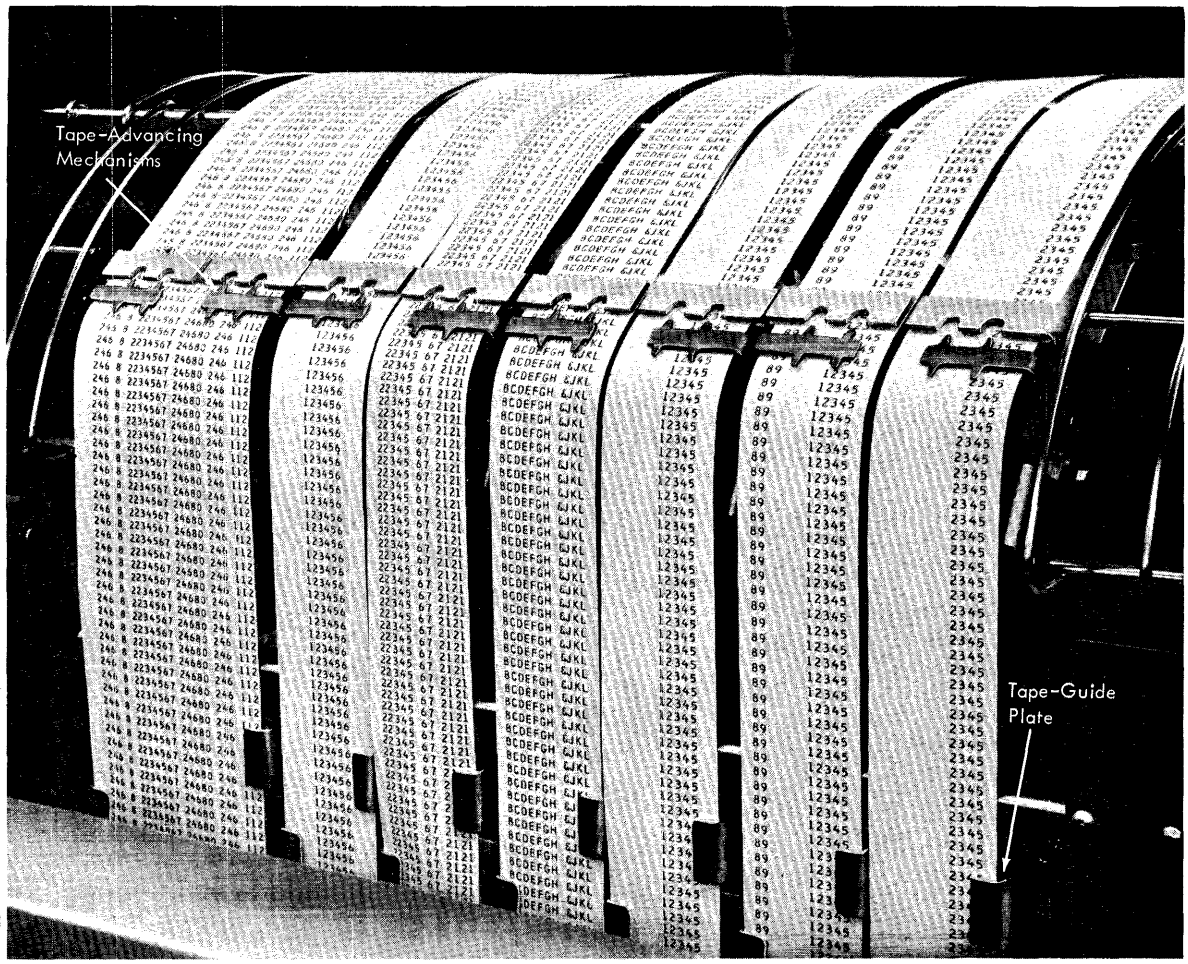


Figure I-106. Selective Tape Listing Feature Mechanism

Buffer Feature for the IBM 1009 Data Transmission Unit

The IBM 1009 Data Transmission Unit, with the buffer feature installed, can be attached to an IBM 1401 or 1460 Data Processing System. With this feature installed on the IBM 1009, data can be loaded and unloaded to and from the transmission line while the processor is free to perform any other function.

The buffer feature provides a 400-character core-storage buffer that is divided into four 100-character blocks. Blocks of 100 characters are loaded into buffer storage from the processor or unloaded from buffer storage into the processor by a single MOVE or LOAD instruction. After a block of 100 characters has been moved into the core-storage area of the processor (receive mode), testable latches can be interrogated by the program to determine whether more data is stored and available for transfer to the processor. After a block of 100 characters has been moved into the 1009 buffer from the processor core storage (transmit mode), the program routine can interrogate the buffer to determine whether it can accommodate another block of 100 characters.

If additional data is available in the 1009 buffer (receive mode), or additional space is available in the 1009 buffer (transmit mode), the program can initiate another MOVE or LOAD instruction. The B-address of the MOVE or LOAD instruction should be increased by 100 before the instruction is executed. This procedure is followed until a group-mark with a word-mark (end-of-message) is detected in the processor core storage (transmit mode) or an end-of-message is detected in the 1009 buffer storage (receive mode). The processor program must determine the validity of the data received by testing the appropriate latch. The data is processed if it is valid. If the data is invalid, it is automatically retransmitted a maximum of two more times. A counter in the 1009 buffer keeps track of the number of transmissions made during an error routine; however, the retransmission of data is under the control of the program. If the data is still invalid after three transmissions the transmission of data ceases, and an alarm sounds, indicating operator intervention is required.

The buffer feature also provides the 1009 with the ability to answer automatically and establish a telephone connection for transmission of data and to disconnect at the end of transmission without operator intervention. The direction of the transmission can be automatically controlled through appropriate programming.

Maximum Processor Time Required for Movement of Data

Blocks of 100 characters, with or without word marks, are transferred by each MOVE or LOAD instruction. The following formula can be used to calculate the time involved in moving a 100-character block of data to or from the buffer.

Timing. $T = 9N + 1N + 1L + 99N$ or $109N + 1L$ (without indexing) ms.

$T = 9N + 1N + 3N + 1L + 99N$ or $112N + 1L$ (with indexing) ms.

L = Line speed in ms.

$N = .0115$ (1401); $.006$ (1460) ms.

IBM 1009 with Buffer (Instructions)

Several processor instructions are expanded to provide program control for operations that involve the IBM 1009 Data Transmission Unit with buffer feature installed.

Initialize a Message Transmission (XMIT)

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*d-character*
CU	U	%D1	E

Function. This instruction initiates a start-of-message signal if the 1009 is in a send-run condition (transmit-receive switch is set to TRANSMIT). If the 1009 is in a receive-run condition (transmit-receive switch set to RECEIVE) the instruction causes the processor to interlock and an alarm to sound, signaling that operator intervention is necessary. This instruction is also used when the transmit, receive, and automatic switch is set to the AUTOMATIC position(buffer feature installed) and a K E (SET DIRECTION TO TRANSMIT) instruction has been issued.

The A-address specifies the 1009, and the d-character specifies the start transmission operation.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%41	d41

Initialize a Reply from Receiver (RCV)

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*d-character*
CU	U	%D1	D

Function. The receiving 1009 signals the transmitting station that it is ready to receive and indicates the status of the previous message (see BRANCH IF INDICATOR ON instruction).

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Note. The transmit-receive switch on the receiving 1009 should be set to RECEIVE. If it is set to TRANSMIT, the processor is interlocked and an alarm is sounded to signal the operator. This instruction is also used when the transmit, receive, and automatic switch is set to the AUTOMATIC position (buffer feature installed) and a K D instruction has been issued.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%41	d41

Move Character to the Transmitting 1009

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
MCW	M	%D1	xxx	W

Function. The transmitting processor sends 100 characters to the 1009 buffer, starting with the position in core storage specified by the B-address. The d-character, W, specifies a transmit operation.

Word Marks. Word marks are not affected.

Timing. T = 109N + 1L (see *Maximum Processor Time Required for Movement of Data*)
Timing. T = 112N + 1L (with indexing)

Note. If a group-mark with a word-mark is sensed in processor core storage, an end-of-message transmit condition is recognized.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%41	B + 100 or GM + 1

Move Character from the Receiving 1009

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
MCW	M	%D1	xxx	R

Function. This instruction transfers 100 characters in the receiving 1009 buffer to the receiving processor core-storage location starting with the position specified by the B-address. The d-character specifies a receive operation.

Word Marks. Word marks are not affected.

Timing. T = 109N + 1L (see *Maximum Processor Time Required for Movement of Data*)
Timing. T = 112N + 1L (with indexing)

Note. When the 1009 recognizes the end-of-message condition, the receiving processor gets an end-of-message-receive signal and inserts a group mark in the core-storage location immediately beyond the location containing the last character of the message.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%41	B + 100 or GM + 1

Load Character to the Transmitting 1009

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
LCA	L	%D1	xxx	W

Function. The transmitting processor sends 100 characters, starting with the location specified by the B-address, to the transmitting 1009 buffer. The d-character, W, specifies a transmit operation.

Word Marks. If a word mark is associated with a character, it is placed with the character in 1009 buffer storage during one transfer cycle. When the character is placed on the transmission line, the word mark is converted to a word separator. Placing the character and word separator on the transmission line takes two transmission cycles.

Timing. T = 109N + 1L (see *Maximum Processor Time Required for Movement of Data*)
Timing. T = 112N + 1L (with indexing)

Note. A group-mark with word-mark in processor core storage signals an end-of-message-transmit condition.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%41	B + 100 or GM + 1

Load Character from the Receiving 1009

Instruction Format.

Mnemonic	*Op Code*	*A-address*	*B-address*	*d-character*
LCA	L	%D1	xxx	R

Function. This instruction transfers 100 characters in the receiving 1009 buffer to receiving processor core storage starting with the location specified by the B-address. The d-character, R, signals a receive operation.

Word Marks. If a word mark is associated with a character, it is received as a word separator followed by its associated data character, but converted to a word mark and placed with its associated character in the receiving 1009 buffer storage. This operation takes two transmission cycles. The character and associated word mark are then transferred to processor core storage during one transfer cycle.

Timing. T = 109N + 1L (see *Maximum Processor Time Required for Movement of Data*)

Timing. T = 112N + 1L (with indexing)

Note. When the 1009 recognizes an end-of-message condition, the receiving processor interprets an end-of-message-receive signal and inserts a group mark in the core-storage location immediately beyond the location containing the last character of the message.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	%41	B + 100 or GM + 1

Branch If Indicator On

Instruction Format.

Mnemonic	*Op Code*	*I-address*	*d-character*
B	B	xxx	X

Function. This instruction tests the indicator specified by the d-character. If the indicator is ON, the program branches to the I-address for the next instruction. If it is OFF, the program continues with the next sequential instruction.

d-character	*Indicator*	*Station*
1	1009 Ready	RCV or XMIT
2	Buffer Service	RCV or XMIT
3	Reply Good	XMIT
4	Reply Bad	XMIT
5	Receive Error	RCV
6	Attention 1009	RCV or XMIT
7	Receive EOM	RCV
8	Receive EOF	RCV or XMIT

Indicators.

B xxx 1 — This indicator turns on when the 1009 is in a run condition. If the 1009 is not in a run condition, the program should stop, or loop until the run condition is established.

B xxx 2 — This indicator, when on, indicates to the processor that the 1009 is in a buffer-available condition when in either the transmit or receive mode. This indicator turns on the first time when 100 characters have been placed in the receive 1009 buffer or when a U%D1 E instruction has been issued when in a transmit mode.

B xxx 3 — This indicator turns on if the signal sent to the transmitting station by the U%D1 D instruction specified that a good transmission occurred. The transmitting processor should test this indicator and branch to the routine for the next message, if it is on. If the indicator is not on, the program should advance to test the transmission-error indicator.

B xxx 4 — The reply-bad indicator turns on if the signal sent to the transmitting station by the U%D1 D instruction specified that a transmission error occurred. The transmitting processor should test this indicator and branch to an error subroutine if an error occurred.

B xxx 5 — This indicator, when on, indicates to the processor that the condition for acknowledgment that has been set in the 1009 is error-reply. This indicator turns on when the first error character is detected in a message and remains on until the next U%D1 D instruction is issued.

B xxx 6 — This indicator turns on when any one of the following indicators turns on: branch-2, branch-3, branch-4, branch-7, and branch-8. When the branch-6 indicator is on, the 1009 requires program attention.

B xxx 7 — This indicator, when on, indicates to the processor that the conditions for an acknowledgment have been set (reply-good or reply-bad) in the 1009. This indicator turns on when the processor receives an end-of-message signal from the 1009 and remains on until the next U%D1 D instruction is issued.

B xxx 8 — This indicator, when on, indicates to the processor that the 1009 has received an EOF (end-of-file) signal from the remote terminal.

Word Marks. Word marks are not affected.

Timing. T = N (L_I + 1) ms.

Address Registers After Operation.

	I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
		With Indexing	
Branch	BI	BI	NSI
No Branch	NSI	BI	d*bb*
		Without Indexing	cleared to
Branch	BI	BI	blanks
No Branch	NSI	BI	d*bb*

Suppress 3-Second Alarm

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	A

Function. This instruction prevents the 3-second alarm from sounding during a delay (such as tape rewind). Normal alarm functions will be restored when any subsequent instruction addresses the 1009. This instruction can be given when a delay in processing can be foreseen.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Set Direction to Receive

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	D

Function. This instruction is incorporated in the processor receive program routine to set the line direction to RECEIVE. This instruction is effective only if the transmit, receive, and automatic switch is set to AUTOMATIC.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Set Direction to Transmit

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	E

Function. This instruction is incorporated in the processor transmit program routine to set the line direction to TRANSMIT. This instruction is effective only if the transmit, receive, and automatic switch is set to AUTOMATIC.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Send EOF (End-of-File)

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	F

Function. This instruction is incorporated in the processor transmit or receive program routine to initiate sending an EOF (end-of-file) code signal to the remote 1009.

Word Marks. Word marks are not affected.

Timing. $T = N\ (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Operate in Load Mode

Instruction Format.

Mnemonic	*Op Code*	*d-character*
SS	K	L

Function. This instruction is placed ahead of the U %D1 D instruction in the processor receive program routine if the receiving 1009 is operating in the load mode. This instruction causes all word separators to be converted to word marks as they are received at the receiving 1009 and before they are placed in the receiving 1009 buffer. The receive station operates in a load mode until an end-of-message signal is received.

Timing. $T = N\ (L_I + 1)$ ms.

Address Registers After Operation.

I-Add. Reg.	*A-Add. Reg.*	*B-Add. Reg.*
NSI	d*bb*	d*bb*

Index

IBM

Technical Newsletter

File Number 1402-03

Re: Form No. A24-3072-2

This Newsletter No. N21-0098

Date November 22, 1968

Previous Newsletter Nos. None

This Technical Newsletter provides replacement pages for *Component Description: IBM 1402 Card Read-Punch,* Form A24-3072-2. Pages to be inserted and/or removed are listed below.

Cover and Edition Page
1-2
5-6
17-18

A change to the text or a small change to an illustration is indicated by a vertical line to the left of the change; a changed or added illustration is denoted by the symbol • to the left of the caption.

Summary of Amendments: (1) Only Models 3, 4, and 5 are equipped with a punch idle control timer. (2) When adding punches into prepunched columns, the previously punched characters must be valid.

File this cover letter at the back of the manual to provide a record of changes.

International Business Machines Corp., Product Publications Dept., Rochester, Minn. 55901

Printed in U.S.A.

IBM 1401 and 1460 Printed in U. S. A. A24-3071-2

IBM

International Business Machines Corporation

Data Processing Division

112 East Post Road, White Plains, N. Y. 10601