IBM

Reference Manual

7070 Data Processing System

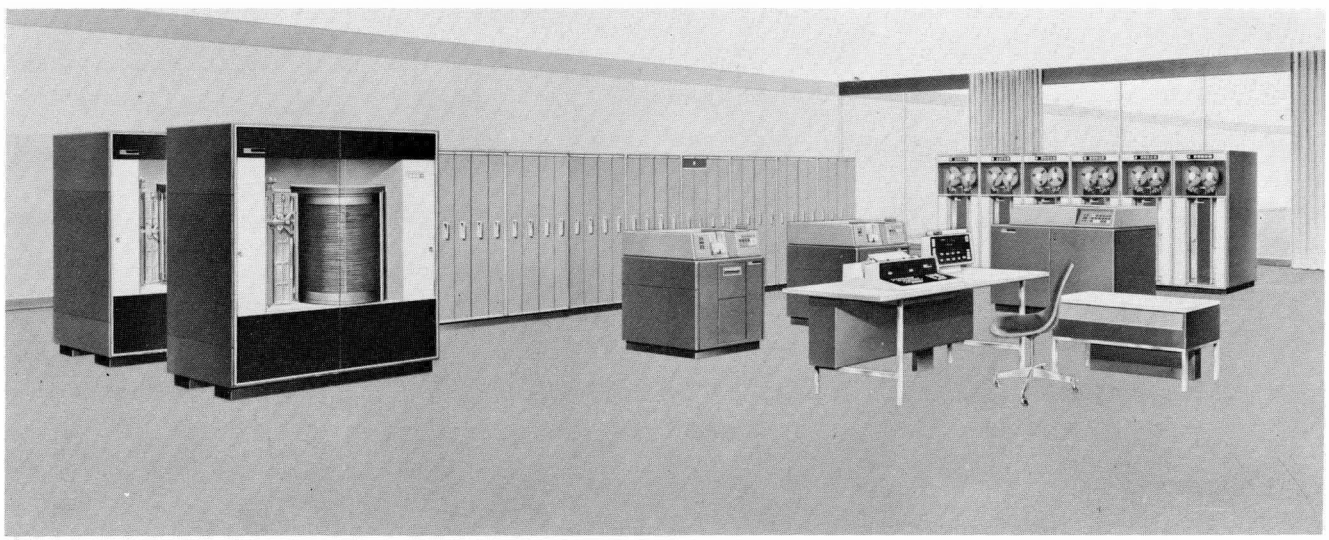
IBM Reference Manual

7070 Data Processing System

MINOR REVISION [January, 1960]

This edition, Form A22-7003-1, is a minor revision of the preceding edition but does not obsolete Form A24-7003-0. The changes in this edition reflect the increased number of tape units that can be attached to the system because of the addition of two more channel controls.

Contents



IBM 7070 DATA PROCESSING SYSTEM

IBM 7070 Data Processing System

The IBM 7070 is an electronic data processing system that covers the range from medium-scale through large-scale processors, by its own variety of configurations. It can be a card system only, an intermediate tape system, or a full-scale tape-disk system, depending on the requirements of the user. Moreover, its processing and storage capabilities can be increased as the requirements increase.

Solid-State Design

The electronic circuits of the IBM 7070 use *transistors,* instead of vacuum tubes. Because of the smaller size of transistors, (Figure 1), their use in a data processing system results in three types of saving:

1. Space requirements are reduced.
2. Air-conditioning requirements are reduced due to the lower heat output of transistors.
3. Power requirements are reduced.

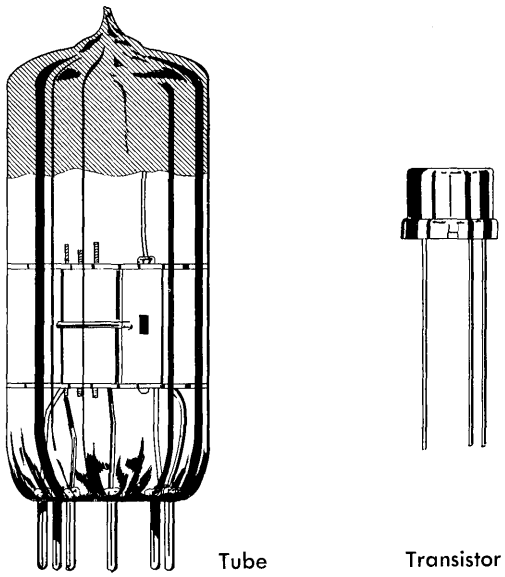


FIGURE 1. RELATIVE SIZES OF VACUUM TUBE AND TRANSISTOR

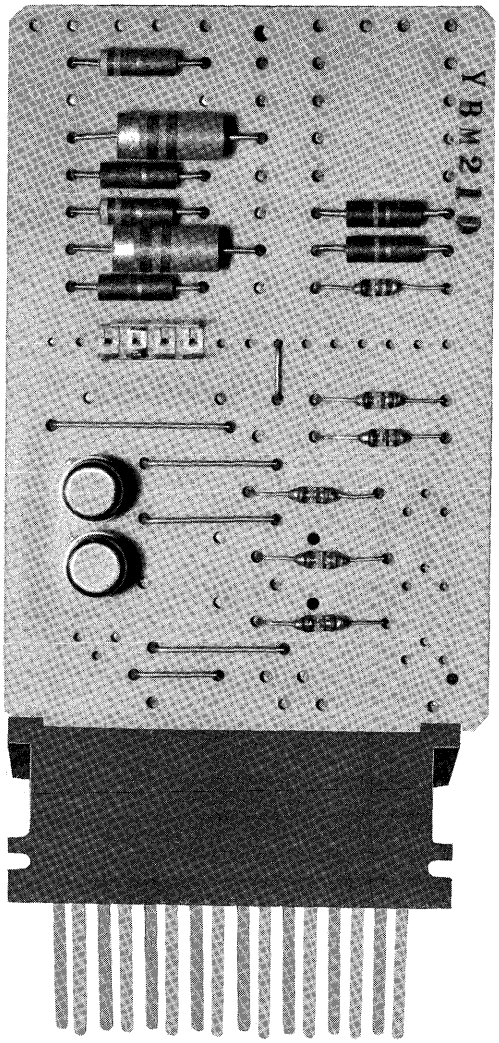


FIGURE 2. SMS CARD, FRONT SIDE, ACTUAL SIZE

Standard Modular System (SMS)

The circuitry of an IBM 7070 is comprised of many SMS cards. Each card can be easily inserted into the system or removed from it. Figure 2 shows a typical SMS card, and the various types of components that can be put on a single card. The prongs are the means of plugging the card into the system. Figure 3 shows the reverse side of a card, which contains a *printed circuit* connecting the components on the front side with one another and with the prongs at the bottom. Each of the 16 contacts at the bottom is connected to one of the prongs shown in Figure 2.

Functional Components

MAGNETIC-CORE STORAGE: The IBM 7070 contains magnetic-core storage in capacity of either 50,000 numerical digits or 99,900 digits, organized into 5000 or 9990 words of 10 digits each.

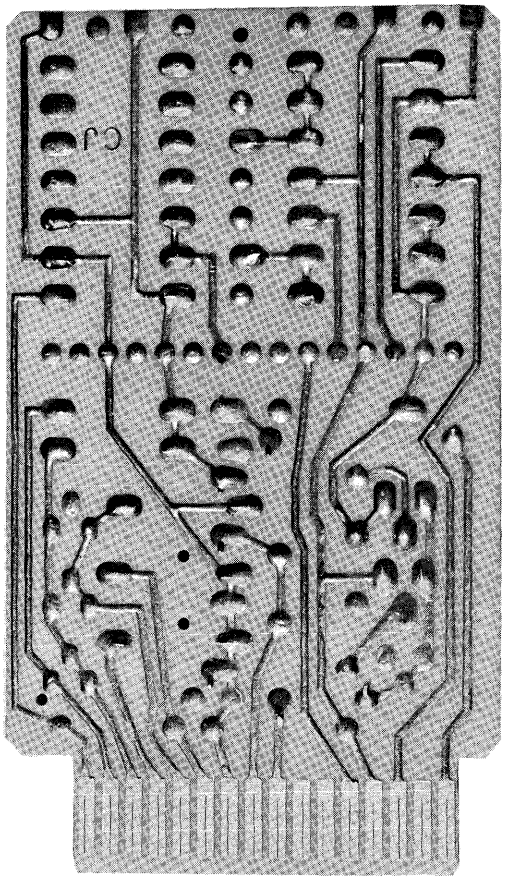


FIGURE 3. SMS CARD, REVERSE SIDE, ACTUAL SIZE

MAGNETIC TAPE: A full-capacity 7070 can have up to 40 IBM 729 Magnetic Tape Units attached.

DISK STORAGE: A 7070 system can include as many as four disk-storage units, making the available disk-storage capacity 48 million numerical digits.

UNIT RECORD INPUT AND OUTPUT: Up to three card readers can be used with a 7070, each with a rated speed of 500 cards a minute. Output consists of any combination of card punches or printers, up to a total of 3. Cards are punched at a rated speed of 250 a minute by each punch unit; the rated speed of the printer is 150 lines a minute.

INQUIRY: As many as ten manual-inquiry stations are available, for request of information and typed reply.

PROCESSING: The programming unit of the 7070 contains three accumulators, with registers and circuitry to perform stored-program instructions, addressing data and instructions, all arithmetic functions, and a wide variety of logic operations.

CONSOLE: The 7070 contains a console, for display, typed output, and manual control of the system.

Reference Manual

This manual gives complete instructions in the use of the operation codes; operation of the tape units, unit-record machines, inquiry stations and console; control-panel wiring of the unit-record machines; and descriptions of special features such as block transmission and automatic priority processing. Its purpose is to assist in planning, programming, and testing 7070 programs; and to aid in training sales representatives, systems personnel, planners, programmers, and test-center personnel, on the system. Although the manual is technical in its approach, it is not an engineering or highly scientific description of the 7070.

Units of the IBM 7070

A full-capacity 7070 system consists of a number of separate units. This is a list of these units, in approximate type-number order, showing the maximum number of each unit available in a single 7070 system:

IBM No.	Name	Maximum number in an IBM 7070 system
729	Magnetic Tape Unit	40
7150	Console	1
7300	Disk-Storage Unit	4
7301	Magnetic-Core Storage	1
7500	Card Reader	3
7400	Printer	3
7550	Card Punch	
7600	Input/Output Control	1
7601	Arithmetic and Program Control	1
7602	Core-Storage Control	1
7603	Input/Output Synchronizer	1
7604	Tape Control	1
7605	RAMAC® Control	1
7900	Inquiry Station	10

Figure 4 is a schematic representation of these components. Note that the two largest units are the Arithmetic and Program Control, IBM 7601; and the IBM 7301 Magnetic-Core Storage unit, with its IBM 7602 Core-Storage Control. These units comprise the nerve center of the system. The 7601 Arithmetic and Program Control executes the stored-program instructions, bringing each instruction from core storage for this purpose. The stored program operates *directly* on data in core storage only. It brings data to and from the cores by means of instructions to read a card, to punch, print, read tape, etc. Note that data transmitted to and from the unit-record equipment, the inquiry stations and the

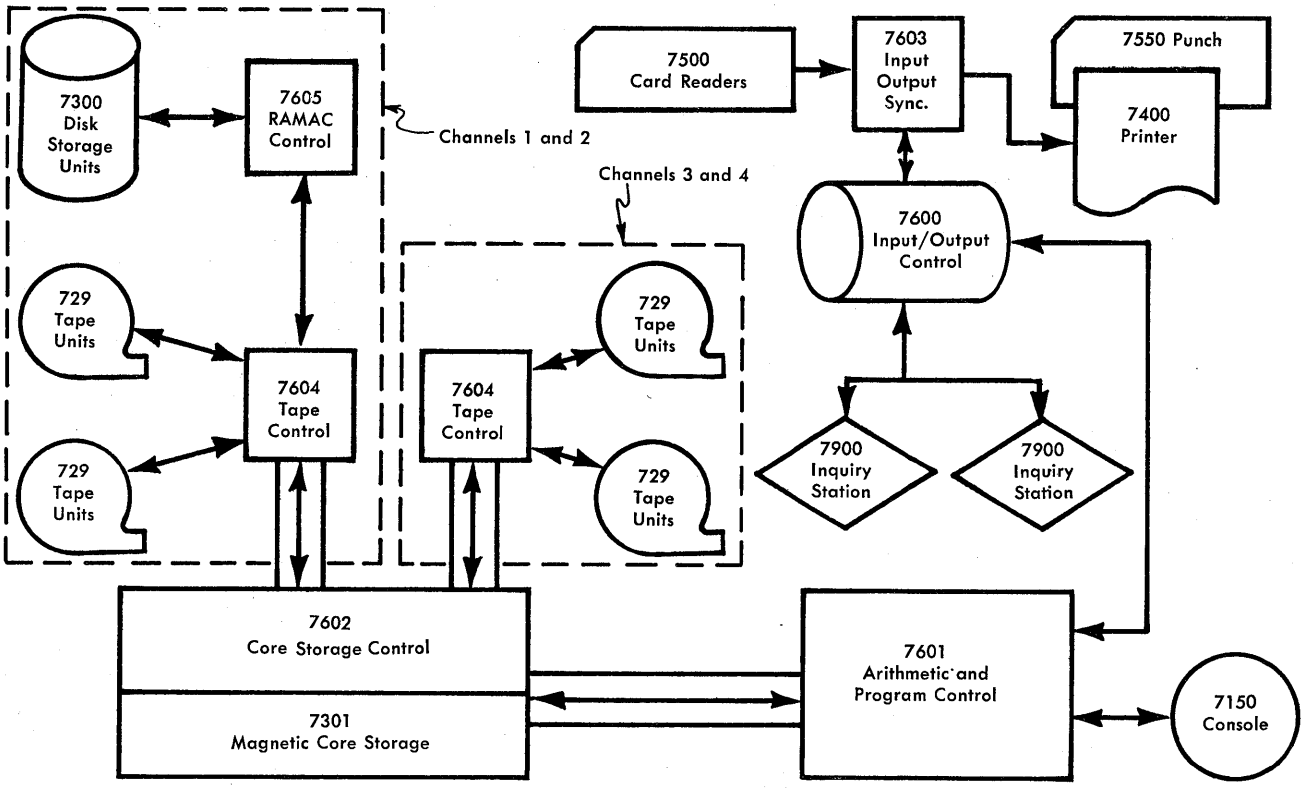


FIGURE 4. SCHEMATIC OF 7070 COMPONENTS

console, goes to and from core storage through the Arithmetic and Program Control.

IBM 729 TAPE UNITS: As many as 40 magnetic tape units, in four groups of 6 each, can be used with the system (see *Magnetic Tape* section).

IBM 7150 CONSOLE: This unit provides manual control of the operation, display of core-storage words, and typed output under control of the stored program (see *Console* section).

IBM 7300 DISK-STORAGE UNITS: As many as four disk files can be used in a 7070 system (see *Disk Storage* section).

IBM 7500 CARD READER, 7400 PRINTER, AND 7550 CARD PUNCH: (See *Unit-Record* section).

IBM 7600 INPUT/OUTPUT CONTROL: This unit contains a magnetic drum, revolving at a speed of 12,500 rpm. It is used as intermediate storage to synchronize core storage with the unit-record equipment, and the inquiry stations, as described in the sections on those units.

IBM 7602 CORE-STORAGE CONTROL: Data brought to or from core storage is addressed by this unit. The stored program sends the address to the 7602, which then causes the designated data to be read to or from core storage.

IBM 7603 INPUT/OUTPUT SYNCHRONIZER: This unit, between the unit-record equipment and the synchronizer drum, contains timing and translating circuitry.

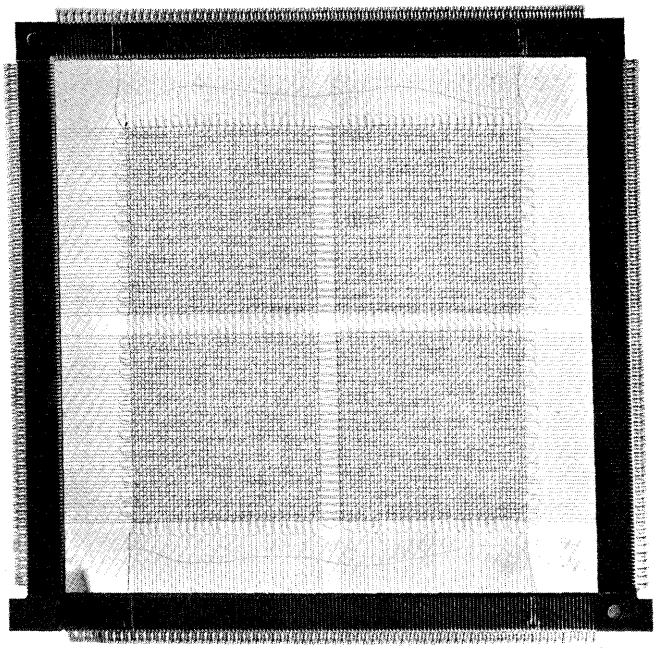


FIGURE 5. MAGNETIC-CORE STORAGE

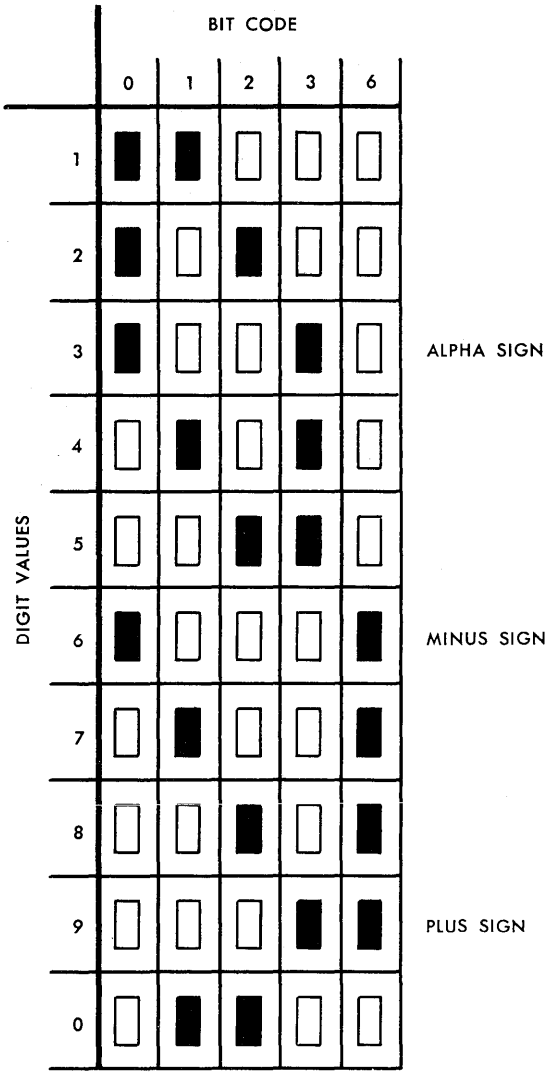


FIGURE 6. TWO-OUT-OF-FIVE FIXED-COUNT CODE

IBM 7604 TAPE CONTROL: Data to be written on tape is brought from core storage to this unit and thence to the tape unit. Data read from tape goes to this unit and thence to core storage. Data to and from the disk-storage units also comes through this unit. The unit contains the two tape/RAMAC channel controls, described under *Block Transmission.*

IBM 7605 RAMAC CONTROL: This unit contains timing, translating, and addressing circuity for all disk-storage operations. It controls data transmission between the disk-storage units and the 7604 Tape Control.

IBM 7301 Magnetic-Core Storage (Figure 5)

Core storage is best described as the *working storage* area of the 7070. It contains the stored program and all data that the stored program uses in its operations.

Data read from any of the input-output or storage units —the card readers, printers, punches, tape units, disk-storage units, inquiry stations, and the console—is brought to core storage. Similarly, any data brought to any of those units is brought from core storage.

CAPACITY: Model 1 of the IBM 7301 contains 5000 words of storage; Model 2 contains 9990 words. A word consists of 10 numerical digits and a sign, which can be plus, minus, or alpha. Because alpha coding requires two digits for each character, each alphabetic word contains five characters.

ADDRESSING: Each word in core storage is addressable. The addresses are 0000-4999 for a Model 1 unit, and 0000-9989 for a Model 2 unit.

BIT-CODE STRUCTURE: Each digit in core storage is represented by a combination of two *bits* out of a possible five. The total number of possible combinations is ten—one for each numerical digit. As shown in Figure 6, the bit positions are designated 0, 1, 2, 3, and 6. The digits 1 to 9 are each composed of two bits, the sum of which equals that number. Zero is designated by the 1-2 combination. A 9 code in the sign position denotes plus; a 6, minus; and a 3, alpha. Only the 0, 3, and 6 bits are used in the sign positions.

ALPHA CODING: An alphamerical word in core storage contains five characters, each represented by two digits. Figure 7 shows all the alphabetic, numerical and special characters included in this coding. Shown with these alphamerical codes are the magnetic-tape BCD code, the punched-card code, and core-storage coding for numerical words (digits 0-9). The characters are in two-digit-code sequence.

Character	Card Code	Core Storage 2-Digit Alphamerical Code	Magnetic Tape BCD Code	Core Storage 1-Digit Numerical Code	Notes
Blank		00	CA		
.	12-3-8	15	CBA821		
□ or)	12-4-8	16	BA84		
	12-5-8	17	CBA841		1
	12-6-8	18	CBA842		1
GM ‡	12-7-8	19	BA8421		1
& or +	12	20	BA		
$	11-3-8	25	B821		
*	11-4-8	26	CB84		
	11-5-8	27	B841		1
	11-6-8	28	B842		1
–	11	30	CB		
/	0-1	31	A1		
,	0-3-8	35	A821		
% or (	0-4-8	36	CA84		
	0-5-8	37	A841		1
	0-6-8	38	A842		1
SM	0-7-8	39	CA8421		1, 5
# or =	3-8	45	C821		
@ or '	4-8	46	84		
	5-8	47	C841		1
	6-8	48	C842		1
TM	7-8	49	8421		1, 6
+0	12-0	60	BA82		2
A	12-1	61	CBA1		
B	12-2	62	CBA2		
C	12-3	63	BA21		
D	12-4	64	CBA4		
E	12-5	65	BA41		
F	12-6	66	BA42		
G	12-7	67	CBA421		
H	12-8	68	CBA8		
I	12-9	69	BA81		
-0	11-0	70	CB82		2
J	11-1	71	B1		
K	11-2	72	B2		
L	11-3	73	CB21		
M	11-4	74	B4		
N	11-5	75	CB41		
O	11-6	76	CB42		
P	11-7	77	B421		
Q	11-8	78	B8		
R	11-9	79	CB81		
RM ‡	0-2-8	80	CA82		3
S	0-2	82	A2		
T	0-3	83	CA21		
U	0-4	84	A4		
V	0-5	85	CA41		
W	0-6	86	CA42		
X	0-7	87	A421		
Y	0-8	88	A8		
Z	0-9	89	CA81		
0	0	90	82	0	
1	1	91	C1	1	
2	2	92	C2	2	
3	3	93	21	3	
4	4	94	C4	4	
5	5	95	41	5	
6	6	96	42	6	
7	7	97	C421	7	
8	8	98	C8	8	
9	9	99	81	9	
Delta Δ	11-7-8		CB8421		4

FIGURE 7. 7070 CODING SYSTEM

NOTES:

1. Cannot be read into the IBM 7500 Card Reader, nor are they translated on output to the 7550 Card Punch, 7400 Printer, 7900 Inquiry Station, or the Console typewriter.
2. Cannot be read by the 7500 Card Reader unless they are wired as the units position of numeric words. Similarly on punching or printing, these codes (60, 70) are invalid.
3. This code cannot be wired to read or punch by the 7070 unit record equipment.
4. Generated by the 7070 controls on write operations, and not translated on read operations. This card code (11-7-8) cannot be entered through the 7500 Card Reader.
5. The tape segment mark is generally a single-character Tape record. This character is *not* translated and placed in 7070 storage if the CA8421 configuration appears as the first character of a tape record. In this case the End of Segment condition is signalled to the 7070 via a final status word condition code.

   It is possible to read the TSM as a character within a tape record (other than the first and to write the TSM from 7070 storage as part of a record.
6. The tape mark is handled in the same fashion as the tape segment mark.

VALIDITY CHECKING: Every digit that is moved to and from storage is tested to assure that it has two bits, neither more nor less. This is called *fixed count* checking.

PARALLEL TRANSMISSION: A feature of the 7070 is parallel transmission of data to and from core storage. An entire word, including sign, is moved all at once, instead of one bit or one digit at a time. A channel for parallel transmission consists of 53 lines, one for each bit in each of the ten digits (50), and the sign (3). This enables a word in core storage to be moved in 6 microseconds (6 millionths of a second). The channels in Figure 4 represent parallel transmission.

Information on tape, disk, and the synchronizer drum is stored serially,—each character read or written before the next character. Thus, transmission of data to and from these units is serial, as indicated by the lines in the figure.

IBM 7601 Arithmetic and Program Control

The programming feature of the 7070 is contained in the 7601 Arithmetic and Program Control Unit. Figure 8 is a simplified schematic of the programming unit. The unit contains three *accumulators,* the *auxiliary register,* the *arithmetic register,* the *program register,* the *instruction counter,* the *adder,* and the *synchronizer register.* Each of the registers and accumulators has a capacity of one word—10 digits and the sign. The instruction counter has a capacity of 4 digits. All arithmetic operations actually take place in the adder. There are several other special registers, and of course, much more circuitry than is shown. Figure 8 is a general, functional representation of data flow. The registers shown in the figure are referred to throughout the text, in the *Data Flow* and *Registers Affected* sections under each operation code.

The three accumulators, the program register, and the instruction counter have addresses:

9991	Accumulator 1
9992	Accumulator 2
9993	Accumulator 3
9995	Program register
9999	Instruction counter

The accumulators can be addressed by certain stored-program instructions, but the instruction counter and program register, 9995 and 9999, can be addressed from the console only.

The three busses in the figure are channels for parallel transmissions of data: a 10-digit word with sign, is moved all at once in a maximum of six microseconds, over these busses. The *information bus* moves data between core storage and the program register, instruction counter, arithmetic register, auxiliary register, and the tape channels. The *arithmetic bus* connects the three accumulators with the auxiliary register, arithmetic register, and synchronizer register. The *address bus* brings addresses to the 7602 Core Storage Control from the instruction counter and program register.

IBM 7070 Instruction

The program is normally sequential: each program step is located in a word with an address one higher than the last instruction. The address of each instruction is obtained by means of the instruction counter. Thus, each program step need not contain the location of the next step. This sequence can be broken by the program whenever it is desired to obtain the next program step from a word other than the one in the next sequential location. This is done by changing the contents of the instruction counter, either directly, or as the result of a logical decision.

Instruction Format

Each instruction in a 7070 program consists of 10 digits and the sign. The sign can be plus or minus, but not alpha. The digit positions are numbered 0 1 2 3 4 5 6 7 8 9 from left to right, or high-order to low-order. The general format of a 7070 instruction is:

S01 23 45 6789

S01	Operation code (S indicates sign)
23	Indexing word
45	Control
6789	Address

OPERATION CODE

The operation code (the sign and positions 0-1) denotes the operation to take place. For example: +24, *add to accumulator 2,* adds the contents of the word specified by the address portion of the instruction to the amount already in accumulator 2, with the result in the accumulator after the operation is completed.

Thus, 200 different operation codes are possible, 100 with a plus sign and 100 with a minus sign. Some of

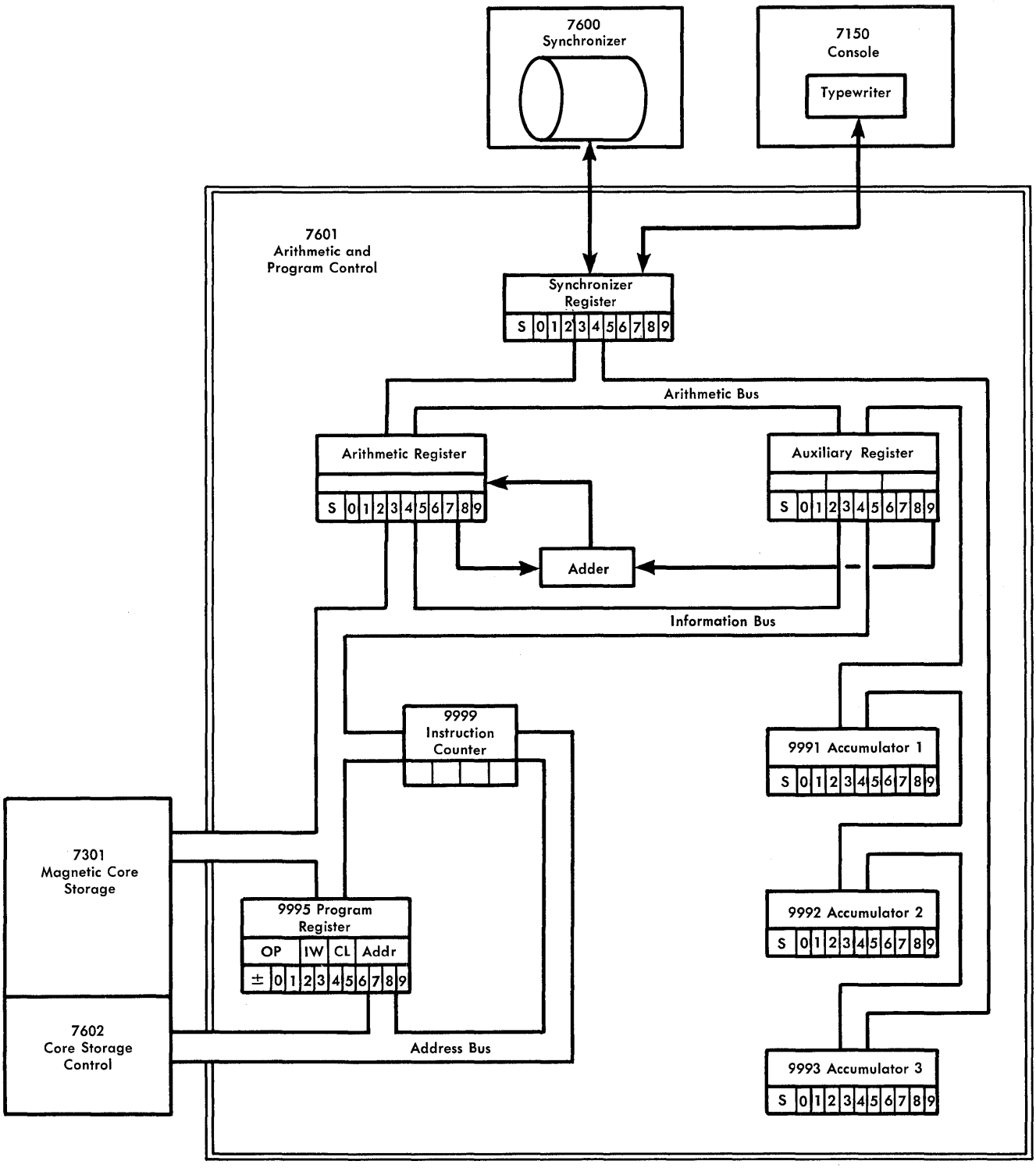


FIGURE 8. IBM 7601 DATA-FLOW SCHEMATIC

the operation codes, moreover, have multiple functions. They are called *augmented* codes; the operation code is augmented by some of the other digit positions in the instruction. An example of this is +69, *Card Control.* This code is used for all card input and punched/printed output, and the console typewriter. Positions 4 and 5 are used to define the operation further. Position 4 denotes the synchronizer, thus specifying the particular input or output unit involved. Position 5 defines the operation: read a card, punch/print, etc.

Indexing Word

Positions 2 and 3 of an instruction specify the indexing word to be used. Magnetic-core storage contains 99 index words, each of which contains a 10-digit number with sign. They are stored in locations 0001-0099. The IW portion of a program step determines which of these 99 words is to be used (00 means no indexing). Positions 2-5 and the sign of the designated indexing word are added algebraically to the address portion of the instruction, positions 6-9, considered plus, and this new address is used for the operation. The other six positions of index words are available to the programmer as storage; positions 6-9 are often used for constants, decrements, and limits. Index words not needed in the program can be used as normal core-storage words. If the indexing word is minus, the address in the instruction is reduced by the value in the indexing portion. If the indexing portion has a greater value and is minus, the 10's complement of the difference is obtained. For example, if positions 6-9 of an instruction contain 1875 and are indexed by the value of −2000, the resultant address is 9875, rather than 0125. (9875 is the 10's complement of 0125; 0125 + 9875 = 10,000).

Every instruction in the 7070 indexable, even if positions 6-9 are not used, or if they are used as a 4-digit factor rather than an address.

An indexing word can be plus or minus, but not alpha. An alpha indexing word specified in positions 2-3 of the instruction causes an error stop, whether positions 6-9 are used as an address, or not.

Any time there is a value other than 00 in the IW portion (positions 2-3) of an instruction, time is taken for indexing. This is true even if positions 6-9 are not used in the operation; indexing takes place at the beginning of each instruction, before the operation code itself has been interpreted.

For most operation codes, indexing adds 36 microseconds. There are 16 operations for which indexing adds only 24 microseconds:

−01	No operation	NOP
−03	Sense mode for sign change	SMSC
−03	Halt mode for sign change	HMSC
−03	Branch if sign change	BSC
−10, −20, −30	Branch if minus in accumulator #	BM1, BM2, BM3
+11, +21, +31	Branch if Overflow in Accumulator #	BV1, BV2, BV3
+40	Branch if low	BL
−40	Branch if high	BH
+41	Branch if field overflow	BFV
+41	Sense mode for field overflow	SMFV
+41	Halt mode for field overflow	HMFV
−41	Branch if equal	BE

Control

FIELD DEFINITION: In many of the instructions a portion of a word can be processed as easily as a full word. Positions 4 and 5 of an instruction determine the part of a word to be used. The digit in position 4 denotes the starting position, the high-order position of the field. The digit in position 5 specifies the low-order position. This is called *field definition.* The digit in position 4 of an instruction can never be higher than the digit in position 5, if field definition is used (—field definition does not extend over word boundaries). A single position is defined by the same digit in positions 4 and 5. For example, 99 in those positions of the instruction denotes the units position of the data word.

The field definition feature means that several fields, with like sign, can be stored in a single word, with no inconvenience to the programmer in processing an individual field. Whenever a portion of a word is used this way, its sign is the sign of the word.

OTHER THAN FIELD DEFINITION: With most of the augmented codes, the CL portion of an instruction does the augmenting—denoting the specific operation of the several that are defined by the operation code.

In the operation codes that specifically operate on index words, the CL portion denotes the index word to be operated on. (Positions 2-3 refer to the index*ing* word, and is used to modify the address, just as in other codes.)

Address

The address portion of an instruction, positions 6-9, usually refers to the storage locaton of the data (this data is sometimes called the operand). In an accumulator addition operation, for example, it is the address of the amount to be added; in a store operation, the location in which the data is to be stored. Another use of positions 6-9 is in branch operations, in which case the address portion contains what may be the location of the next instruction. An example of this is +30, *Branch if Zero in Accumulator 3.* If there is a non-zero number in accumulator 3 (regardless of sign), the address of the next instruction is the next sequential location. If the accumulator is entirely zero, the contents of the address portion of the instruction are moved to the instruction counter, and the next instruction comes from that location.

In some operations, the address portion of a program step contains the actual number to be processed, rather than a storage location. The 4-digit number in positions 6-9 of the instruction is used as a factor in the operation. This number is always considered plus, for these operations.

With some of the augmented codes, the address portion does the augmenting. Positions 6-9 of every instruction can be modified by indexing, regardless of whether they represent an address, a 4-digit factor, are part of the operation itself, or are not used at all.

EFFECTIVE ADDRESS: As many as eight positions of a program instruction may be used to define the specific digit positions of specific core-storage to be used by the instruction. Positions 6-9 contain an address. Positions 2-3 contain an indexing-word designation, or 00. Positions 4-5 define the digit positions of that word that are to be used (09 in these positions denotes a full word). The digit positions thus defined are sometimes referred to as the *effective address* of the instruction.

Indicators

The IBM 7070 contains 10 indicators, each of which is turned on automatically by a condition that arises during the stored program. They are:

Accumulator 1 overflow
Accumulator 2 overflow
Accumulator 3 overflow
Floating-decimal overflow
Floating-demical underflow
Sign change
Field overflow
High (compare)
Equal (compare)
Low (compare)

The indicators can be tested at any time by the program. With the exception of the compare indicators, each indicator is automatically turned OFF by the operation that tests it, if it was ON. Throughout this text, the name of each indicator is in italics.

Autocoder Mnemonics

Each operation has a mnemonic representation. For example, the operation code +22, *Store Accumulator 2,* is written as ST2; the programmer doesn't need to know that the operation code is +22. Each augmented code has a mnemonic representation for each of the several operations it performs. The +69 *Card-Control* code has 8 different mnemonics. These mnemonic representations are used for *Autocoder* programming.

IBM 7070 Basic Autocoder

The 7070 Basic *Autocoder* is a programming system developed to simplify the preparation of programs for the IBM 7070 Data Processing System. The major advantages of such a programming system are:

1. Operation codes are written in an easily remembered mnemonic form, rather than in the numerical language of the machine.
2. Every command is given a unique mnemonic representation, even though machine-language codes are the same.
3. Data to be processed is referred to symbolically, using names or other meaningful designations.
4. Instructions are not assigned core-storage locations by the programmer; thus the addition and deletion of instructions entail no re-assignment of addresses.
5. Each routine in a program can be written independently of the others with no loss of efficiency in the final program.

Writing a program in Basic *Autocoder* language relieves the programmer of most of the tedious clerical tasks. These tasks are turned over to the 7070 and the Basic *Autocoder* Processor. The processor takes the program in *Autocoder* language, translates the mnemonic codes into the machine-language codes, assigns core-storage addresses to the instructions and to the symbolic data references, and assembles a finished machine language program.

Also, the processor performs the added function of checking for certain common coding errors, and notes these by means of messages while continuing the translation process.

The IBM 7070 Basic *Autocoder* can assemble programs for use with any configuration of the 7070 system. The processor requires only a minimum of equipment: 5000 words of core storage, one card reader, and one card punch. The addition of a printer makes it possible to obtain a direct listing of the assembled program.

Basic *Autocoder* is described completely in the 7070 Data Processing System Bulletin *(Basic Autocoder Programming,* Form J28-6021).

IBM 7070 Autocoder

The full IBM 7070 *Autocoder* is an important tool for writing programs. It contains powerful macro-instructions, extensive control operations over processing, reassembly, multifile procedures, and many output options.

On one hand, *Autocoder* includes low-level statements that are very much like the 7070 machine language; on the other hand, it includes high-level statements, called *macro-instructions,* which bear no resemblance to machine language. The low-level statements offer more flexibility and control over each detail of the coding. The high-level statements provide a more

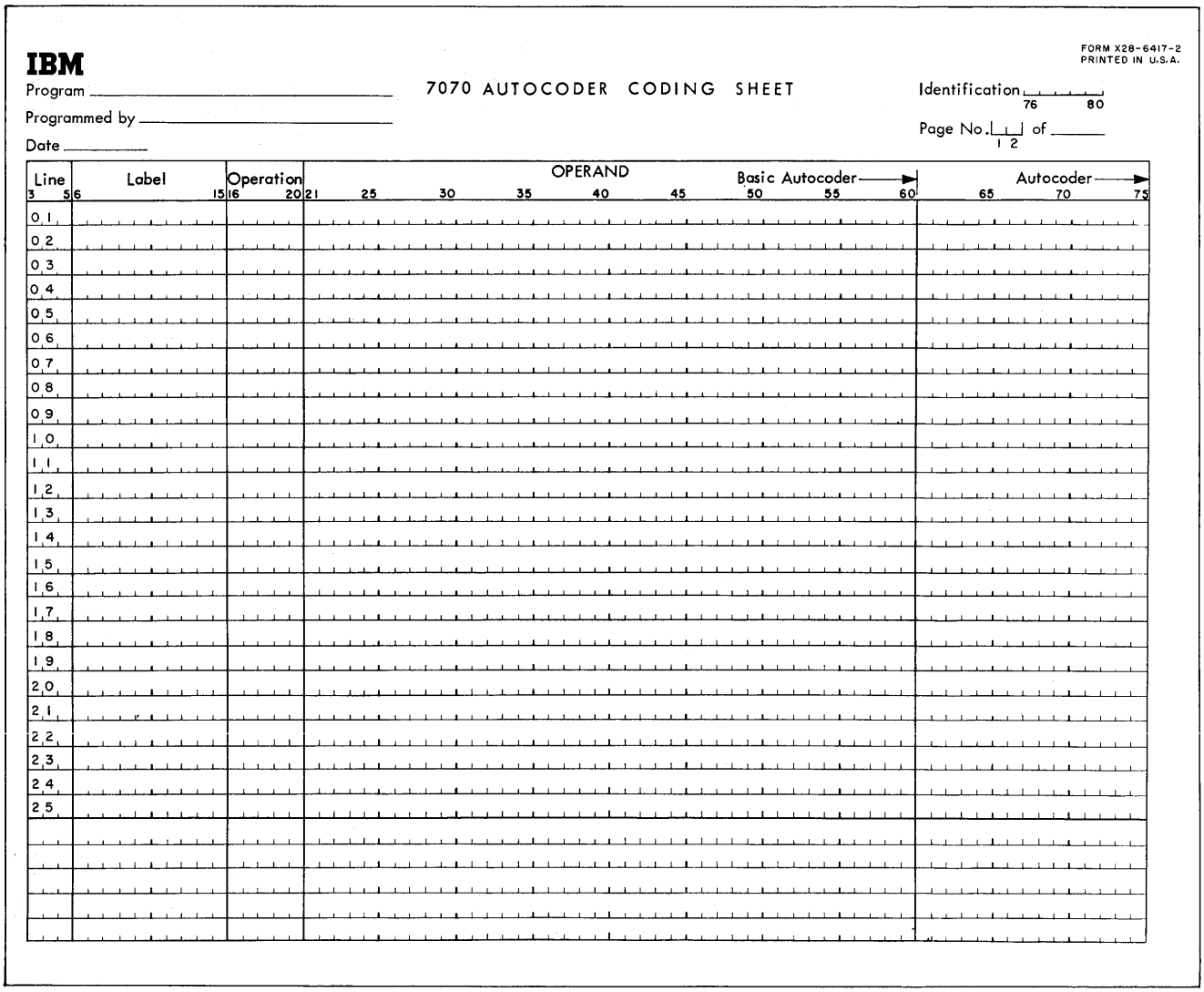


FIGURE 9. AUTOCODER CODING SHEET

convenient way to state a problem. They usually produce a number of machine-language instructions.

In addition to the macro statements provided by the IBM 7070 *Autocoder,* the user may add his own macro-instructions. Thus the language can be extended.

IBM 7070 *Autocoder* is described in the 7070 Data Processing System Bulletin *(IBM 7070 Autocoder,* Form J28-6032).

7070 Autocoder Coding Sheet

Source language programs are written on the IBM 7070 *Autocoder* Coding Sheet, Form X24-6417 (Figure 9). This form is used for both Basic *Autocoder* and full *Autocoder,* and its use is illustrated in the bulletins mentioned. The sheet indicates the column numbers of the *Autocoder load card* (Figure 10), punched from each line of the sheet.

IBM 7070 Basic Fortran

The *Fortran* language is a concise, convenient means of stating the steps to be carried out by the IBM 7070 Data Processing System in solving many types of problems, particularly in scientific and technical fields. As the language is simple, and the 7070, with the *Fortran* compiler program and the basic *Autocoder* assembly program, performs most of the clerical work, *Fortran* affords a significant reduction in the time required to write programs.

Virtually any numerical procedure can be expressed in the *Fortran* language. Arithmetic formulas are stated in a language closely resembling that of mathematics. Iterative processes can be easily governed by control statements and arithmetic statements. Input and output are simply handled by appropriate statements.

The basic 7070 *Fortran* system described here is designed for use with the basic IBM 7070 Data Processing System. The basic *Fortran* language is acceptable to the *Fortran* program produced for the expanded IBM 7070 Data Processing System.

GENERAL DESCRIPTION: The function of the basic *Fortran* system for the IBM 7070 is to convert a source program written in *Fortran* language into 7070 machine language. The system consists of two major parts:

1. The compiler, basic *Fortran,* which translates the *Fortran* language statements (source program) into basic *Autocoder* (symbolic) language.

2. The assembler, basic *Autocoder,* which converts the symbolic statements produced by the compiler into a machine language program (the object program).

The operation of the compiler and the assembler are automatic so that the programmer need use only the 7070 *Fortran* language. Debugging programs can be done using the *Fortran* source program. Accordingly, knowledge of the machine language or basic *Autocoder* is not required.

MACHINE REQUIREMENTS: Basic *Fortran* requires a system that consists of the IBM 7070, one IBM 7500 Card Reader, and one IBM 7550 Card Punch.

WRITING THE SOURCE PROGRAM: The *Fortran* language, which is used to write source programs for the basic *Fortran* system, is a language the structure of which closely resembles the language of mathematics. This is best described by an example of an arithmetic statement in the *Fortran* language. Consider the algebraic formula for one of the two roots of a quadratic equation:

$$\text{ROOT} = [-B + \sqrt{B^2 - 4AC}]/2A$$

The *Fortran* language statement that creates a machine language program for this calculation is:

```
ROOT =
(-B+SQRTF(B**2-4.0*A*C))/(2.0*A)
```

In this example the symbols denote the following:

1. The meaning of the entire statement is: evaluate the expression on the right side of the equal sign and make this the value of the variable on the left.
2. The symbol * denotes multiplication.
3. The symbol ** denotes exponentiation; e. g., A**3 means A^3.
4. SQRTF (arg.) is a subroutine that computes the square root of the argument enclosed in parentheses.

In addition to arithmetic statements, the basic *Fortran* system includes other statements to specify divisions, transfers, and input/output functions. Full description of basic *Autocoder* is found in the IBM 7070 Data Processing System Bulletin, Basic *Fortran*, Form J28-6037.

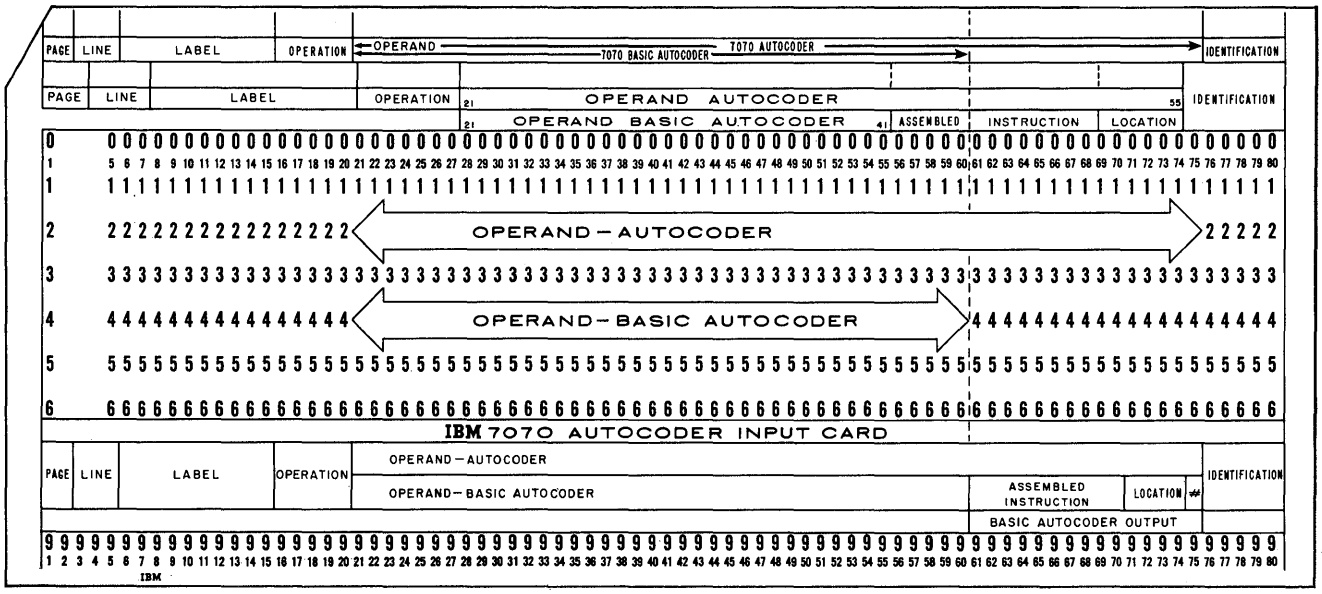


FIGURE 10. AUTOCODER LOAD CARD

Format of Operation-Code Text

The text discusses each operation code as an entity. For augmented codes, the *Autocoder* symbols are in the *Instruction Format* section.

The eight sub-paragraphs of the text describe the various aspects of the operation code, in this sequence:

MACHINE DESCRIPTION: A functional description of what the operation code accomplishes, what its purpose is, etc. If the programmer is familiar with the general format of instructions in the 7070, this paragraph can teach him how to use this operation code.

INSTRUCTION FORMAT: A breakdown of the 10 digits and sign of the instruction, showing the function of each group of digit positions.

EXAMPLE: A machine-language example of an instruction using this operation code.

DATA FLOW: A brief, general description of the actual operation taking place, using the schematic of the programming unit (Figure 8).

REGISTERS AFFECTED: A list of the registers used by the operation. This includes all accumulators, registers, the instruction counter, and storage locations that are used.

TIMING: The duration of the instruction. If the instruction involves the use of an input/output or storage unit, the duration of the operation of that unit is also given.

COMMENTS: Any miscellaneous data that might be helpful to the programmer — things that should be kept in mind concerning this operation code.

AUTOCODER EXAMPLE: An example of how this instruction might be written symbolically, showing the *Autocoder* instruction and the machine-language instruction assembled from it. In each example, the top line of the *Autocoder* Coding Sheet is used, to show the column headings.

Operations Involving Accumulators

Included in this section are all the operations that involve the use of one or more of the accumulators, with the exception of *Branch on Accumulator* contents or sign, and the floating-decimal instructions (table look-up and disk-storage operations use accumulator 3).

Figure 11 is a categorized list of the codes in this section, with the operation codes, names, and *Autocoder* mnemonics.

Zero Accumulator # and Add
+13, +23, +33 **ZA1, ZA2, ZA3**

MACHINE DESCRIPTION: The accumulator specified by the high-order digit of the operation code is set to zero. The field-defined portion of the word addressed by positions 6-9 (indexable) is brought to the accumulator, to the low-order portion if less than ten digits are defined. The sign of the accumulator is made the same as that of the data word.

INSTRUCTION FORMAT:

S	0	1	23	45	6789

S — Always +.

0 — Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.

1 — Always 3.

23 — Indexing word.

45 — Field definition.

6789 — Address of data word (indexable).

EXAMPLES: To move all of word 0500 to accumulator 1:

S01	23	45	6789
+13	00	09	0500

After this operation is completed, the contents of word 0500 and accumulator 1 are identical.

To zero accumulator 2 and then add positions 3-5 of word 1645:

S01	23	45	6789
+23	00	35	1645

Contents of word 1645: −81345 60193. Contents of accumulator 2: −00000 00456, regardless of contents prior to instruction.

DATA FLOW: The entire contents of the data word, including sign, are moved in parallel to the arithmetic register. If a full word is field-defined (09 in positions 4-5), the contents of the arithmetic register are moved, in parallel, to the designated accumulator. If less than a full word is field-defined, the field in the arithmetic register is moved, one digit at a time starting with the low-order digit, through the adder, back to the low-order portion of the arithmetic register. The contents of the arithmetic register, including sign, are then sent, in parallel, to the accumulator, with zeros inserted for all the high-order positions to the left of the field-defined digits.

REGISTERS AFFECTED: Arithmetic register, adder, and specified accumulator.

CATEGORIES	OP CODES	NAMES	MNEMONICS
Reset accumulator	+13, +23, +33	Zero accumulator # and add	ZA1, ZA2, ZA3
	−13, −23, −33	Zero accumulator # and subtract	ZS1, ZS2, ZS3
Addition and subtraction	+14, +24, +34	Add to accumulator #	A1, A2, A3
	−14, −24, −34	Subtract from accumulator #	S1, S2, S3
Multiplication	+53	Multiply	M
Division	−53	Divide	D
Absolute Value	+16	Zero accumulator 1 and add absolute	ZAA
	−16	Zero accumulator 1 and subtract absolute	ZSA
	+17	Add absolute to accumulator 1	AA
	−17	Subtract absolute from accumulator 1	SA
Store	−11, −21, −31	Zero storage and store accumulator #	ZST1, ZST2, ZST3
	+12, +22, +32	Store accumulator #	ST1, ST2, ST3
	−12, −22, −32	Store digits from accumulator # and ignore sign	STD1, STD2, STD3
Add to storage	+18, +28, +38	Add to storage from accumulator #	AS1, AS2, AS3
	−18, −28, −38	Subtract accumulator # from storage	SS1, SS2, SS3
	+19, +29, +39	Add to absolute storage from accumulator #	AAS1, AAS2, AAS3
Shift	+50	Shift control:	
		Shift right accumulator #	SR1, SR2, SR3
		Shift right and round accumulator #	SRR1, SRR2, SRR3
		Shift left accumulator #	SL1, SL2, SL3
		Shift left and count accumulator #	SLC1, SLC2, SLC3
	−51	Coupled shift control:	
		Shift right coupled	SR
		Shift right and round coupled	SRR
		Shift left coupled	SL
		Shift left and count coupled	SLC
		Shift right from point accumulator #	SRS
		Shift left from point accumulator #	SLS

FIGURE 11. ACCUMULATOR OPERATION CODES

TIMING: The duration of this code depends on the number of digits in the field. A full word takes only 36 microseconds, however, because the digits do not have to be shifted in the arithmetic register (Figure 12).

Number of Digit Positions	1	2	3	4	5	6	7	8	9	10
Microseconds	36	48	48	48	60	60	60	72	72	36

FIGURE 12. TIMING — ZERO ACCUMULATOR # AND ADD

COMMENTS: Although these codes involve accumulators, they are not really arithmetic operations; there is no addition or subtraction of two values. The accumulator always takes the sign of the data word; plus, minus or alpha, regardless of the size of the field defined.

Accumulator addresses (9991, 9992, or 9993) can be used, in which cases the addressed accumulator is treated as a core-storage word. If the same accumulator is used, as in ZA1 from accumulator 1, for example, the instruction has the effect of moving the field to the low-order portion and resetting the other positions. For example, if accumulator 1 has −54557 98643, and this instruction is given:

S01	23	45	6789
+13	00	03	9991

The result in the accumulator is −00000 05455. If field definition were 69 instead of 03, the result would be −00000 08643.

AUTOCODER EXAMPLES (Figures 13 and 14): The second example given for this code would be written symbolically as shown in Figure 13.

The assembly program produces code +23 from the ZA2 operation. Field definition (3, 5) specifies positions 3-5 of an area previously defined as GEORGE. If GEORGE is a complete word, as in this example, field definition in the assembled instruction is the same (35). If GEORGE had been defined as the six low-order positions of a word, field definition would be 79. The address of that word is placed in positions 6-9 of the assembled instruction.

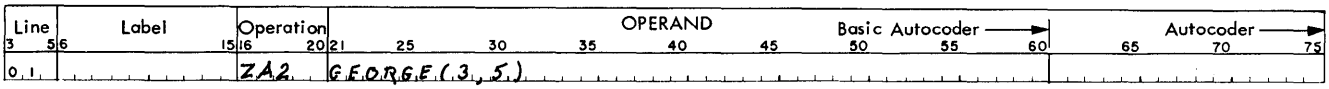


FIGURE 13.

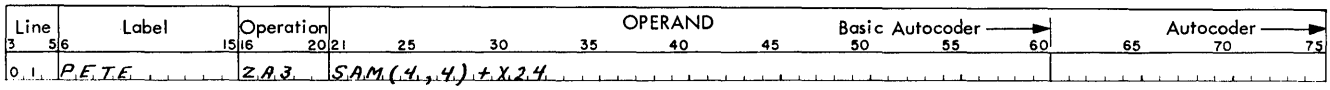


FIGURE 14.

To reset accumulator 3 and add position 4 of SAM, indexed, (instruction PETE; Figure 14):

In the example, assume that SAM has been defined as a complete word and its location is 1789. Instruction PETE is assembled as:

S01	23	45	6789
+33	24	44	1789

The number in positions 2-5 of IW 24 is added algebraically to the 4-digit number in 1789, each time this instruction is executed.

Zero Accumulator # and Subtract
−13, −23, −33 ZS1, ZS2, ZS3

MACHINE DESCRIPTION: The field-defined portion of the word addressed by positions 6-9 (indexable) is brought to the accumulator specified by position 0, to the low-order portion if less than 10 digits are defined.

If the operand is + or −, the accumulator gets the opposite sign; if the operand is alpha, the accumulator gets an alpha sign.

INSTRUCTION FORMAT: Same as ZA#, except for the sign.

EXAMPLES: To move all of word 0500 to accumulator 1 and change the sign:

S01	23	45	6789
−13	00	09	0500

Contents of 0500: +54380 02004. Contents of accumulator 1 after the operation: −54380 02004, regardless of previous contents.

To move the three high-order positions of word 1762 to accumulator 3, and change the sign:

S01	23	45	6789
−33	00	02	1762

Contents of 1762: −06000 00000. Contents of accumulator 3 after the operation: +00000 00060

DATA FLOW: This is the same as for *Zero Accumulator # and Add,* with this addition: After the high-order digit of the field is brought from the adder to the arithmetic register, the sign in the arithmetic register is tested. If it is 9 (+), it is made 6 (−), if it is 6, it is made 9. If it is 3 (a), it is retained as 3.

REGISTERS AFFECTED: Same as *Zero Accumulator # and Add.*

TIMING: Same as *Zero Accumulator # and Add*

COMMENTS: The sign of an accumulator can be changed by using this operation code with its own accumulator. The instruction:

S01	23	45	6789
−23	00	09	9992

changes the sign of accumulator 2 from + to − or vice-versa. There is no change if the sign is alpha.

Whenever the sign of the operand is alpha, ZS# and ZA# produce the same result.

AUTOCODER EXAMPLE (Figure 15): Assume that JOE has been defined as positions 0-5 of word 1551. Because no indexing or field definition is specified in the operand, the assembled instruction is:

S01	23	45	6789
−23	00	05	1551

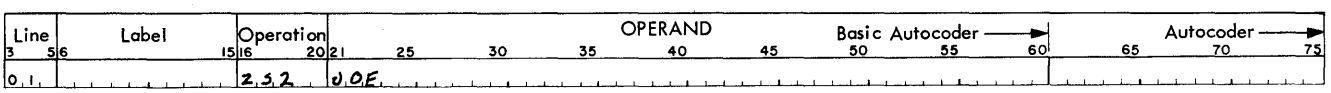


FIGURE 15.

Add to Accumulator #
+14, +24, +34 **A1, A2, A3**

MACHINE DESCRIPTION: The field-defined portion of the word addressed by positions 6-9 (indexable) is added to the amount in the accumulator, to the low-order portion if less than 10 digits are defined. The signs of both factors are taken into account. The result is in the accumulator after the operation.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S	Always +
0	Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.
1	Always 4
23	Indexing word
45	Field definition
6789	Address of data word (indexable)

EXAMPLES: To add the four low-order positions of word 0540 to accumulator 2:

S01	23	45	6789
+24	00	69	0540

To add all of word 1781 to accumulator 3:

S01	23	45	6789
+34	00	09	1781

DATA FLOW: The entire contents of the data word, including sign, are moved, in parallel, to the arithmetic register. The entire contents of the accumulator are moved in parallel to the auxiliary register. The units position of the field in the arithmetic register (indicated by the digit in position 5 of the instruction) is brought to the adder; the units position of the number in the auxiliary register is brought to the adder at the same time. The result goes from the adder to the units position of the arithmetic register; a carry 1, if any, is retained in the adder. If the operation is an actual subtraction (obtaining the difference in the two numbers), the digit from the arithmetic register is converted to its 10's complement as it enters the adder: a 1 becomes a 9, a 2 becomes an 8, etc; a 0 remains as a 0. (For all digits after the lowest-order nonzero digit, the 9's complement is used.) This is called *complement add.* Actual addition (obtaining the sum of two numbers, both plus or both minus) is called *true add.*

In the same way, the 10's digits of the two numbers are brought to the adder, and are added together, along with a carry 1, if any, from the addition of the units digits. The process continues until all the digits of the defined field in the arithmetic register, and all the significant digits in the auxiliary register have been added, and the complete result is in the arithmetic register. This takes as many add cycles as there are field-defined digits in the arithmetic register or significant digits in the auxiliary register, whichever is greater. If the result (in a complement-add operation) must be recomplemented, the result in the arithmetic register is brought through the adder and converted to its 10's complement (recomplement is described functionally in the *Timing* section). For all cases in which recomplementing is required, the sign of the result is changed from plus to minus, or vice-versa.

In either true add or complement add, if the original accumulator sign is alpha, it is not changed. If the storage-word sign is alpha, the result is made alpha.

The contents of the arithmetic register are sent to the accumulator, with zeros inserted for the high-order positions to the left of the result.

REGISTERS AFFECTED: The accumulator, arithmetic register, and auxiliary register.

TIMING: The duration of this instruction depends on the number of significant digits in the result, and on whether the operation is a *true add* or *complement add.* If the result is the sum of the two numbers, the operation is called *true add.* If the result is the difference of the two numbers, causing the digits in one of the factors to be complemented before entering the adder, the operation is called *complement add.* If a complement-add operation causes the accumulator to change its sign (because the original accumulator value was smaller in absolute value than the data from storage), a *recomplement* automatically takes place at the conclusion of the operation. For example:

Accumulator 2 contains the value	+123
The storage word value is	−127
The operation is: +24, add to accumulator 2	
The adder adds:	123
(10's complement of 127)	873
	996

This is complement-add. The value 996 is the 10's complement of the correct answer, −004. At the end of the operation, the 996 is recomplemented to −004.

Let's look at a complement-add operation that does *not* need recomplementing:

Accumulator 2 contains the value:	−456
The storage-word value is:	+421
The operation is: +24, add to accumulator 2	
The adder adds:	456
(10's complement of 421)	579
	−(1) 035

Note that the result does not need recomplementing if the accumuluator value is greater than that of the field-defined storage word. Note also that the carry 1 is not needed to obtain the correct result of +35, whereas a carry in a true-add operation is part of the total (+305 added to +759 is +1 064). Complement-add *without* recomplement thus takes less time than true-add with carry, which in turn takes less time than complement-add *with* recomplement.

Figure 16 indicates the duration in microseconds of accumulator addition/subtraction operations, as determined by field size.

Number of Digit Positions	1	2	3	4	5	6	7	8	9	10
True Add to Accumulator	48	48	48	60	60	60	72	72	72	72
Complement Add to Acc.	36	48	48	48	60	60	60	72	72	72
If Recomplement	60	60	72	84	84	96	108	108	120	132

FIGURE 16. TIMING — ADD TO ACCUMULATOR #

COMMENTS: An add operation obtains either the sum or the difference of the two factors, depending on the signs: the sum if the signs are the same, the difference if they are different. Three factors influence the value and sign of the result in an add instruction:

1. The sign of the storage word (+ or −)
2. The sign of the accumulator, prior to the operation (+ or −)
3. The operation (add or subtract)

Figure 17 is a chart showing the signs and values of the results of these combinations.

If either factor has an alpha sign, its value is considered plus in an arithmetic operation. Regardless of the value of the result, the sign of the result is alpha if either factor is alpha (Figure 18).

ORIGINAL SUM OF ACCUMULATOR	SIGN OF DATA WORD	OPERATION	VALUE OF RESULT	SIGN OF RESULT
+	+	Add	Sum	+
+	−	Add	Difference	Sign of Greater Value
−	+	Add	Difference	Sign of Greater Value
−	−	Add	Sum	−

FIGURE 17. ADD TO ACCUMULATOR # — PLUS AND MINUS FACTORS

Any time the contents of an accumulator are brought to zero (by adding an amount equal to it but opposite in sign if neither sign is alpha), the sign is not changed. If a +7 from storage is added to a −7 in an accumulator, the result in the accumulator is −0.

ORIGINAL SIGN OF ACCUMULATOR	SIGN OF DATA WORD	OPERATION	VALUE OF RESULT	SIGN OF RESULT
+	Alpha	Add	Sum	Alpha
−	Alpha	Add	Difference	Alpha
Alpha	+	Add	Sum	Alpha
Alpha	−	Add	Difference	Alpha
Alpha	Alpha	Add	Sum	Alpha

FIGURE 18. ADD TO ACCUMULATOR # — ALPHA FACTORS

AUTOCODER EXAMPLE (Figure 19): Assume that ALPHA has been defined as the five high-order positions of word 0660. The assembled instruction is:

S01	23	45	6789
+14	74	04	0660

Subtract from Accumulator #
−14, −24, −34 **S1, S2, S3**

MACHINE DESCRIPTION: The field-defined portion of the word addressed by positions 6-9 (indexable) is subtracted from the amount in the accumulator, from the low-order portion if less than 10 digits are field-defined. The signs of both factors are taken into account. The result is in the accumulator after the operation.

INSTRUCTION FORMAT: Same as A#, except for the sign.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		A 1	A L P H A + X 7 4		

FIGURE 19.

EXAMPLES: To subtract the five high-order positions of word 0881 from the amount in accumulator 3:

S01	23	45	6789
−34	00	04	0881

To subtract the units position of word 2320 from the amount already in accumulator 1:

S01	23	45	6789
−14	00	99	2320

DATA FLOW: Same as for A#

REGISTERS AFFECTED: Same as for A#

TIMING: Same as for A#

COMMENTS: A subtract operation obtains the difference of the two factors if the signs are the same, and the sum of the factors if the signs are different. As with the add operation codes, the result is influenced by the sign of the storage word and the sign of the accumulator, as well as by the subtract instruction (Figure 20).

ORIGINAL SIGN OF ACCUM.	SIGN OF DATA WORD	OPER.	VALUE OF RESULT	SIGN OF RESULT
+	+	Subtract	Difference	+ if acc. value greater − if storage value greater
+	−	Subtract	Sum	+
−	+	Subtract	Sum	−
−	−	Subtract	Difference	+ if storage value greater − if acc. value greater

FIGURE 20. SUBTRACT FROM ACCUMULATOR # — PLUS AND MINUS FACTORS

If either factor is alpha, its value is considered plus in determining true or complement add. It makes the sign of the result alpha in all cases (Figure 21).

ORIGINAL SIGN OF ACCUMULATOR	SIGN OF DATA WORD	OPERATION	VALUE OF RESULT	SIGN OF RESULT
+	Alpha	Subtract	Difference	Alpha
−	Alpha	Subtract	Sum	Alpha
Alpha	+	Subtract	Difference	Alpha
Alpha	−	Subtract	Sum	Alpha
Alpha	Alpha	Subtract	Difference	Alpha

FIGURE 21. SUBTRACT FROM ACCUMULATOR # — ALPHA FACTORS

Any time the contents of an accumulator are brought to zero (if neither factor is alpha), the sign is not changed. If a +7 in storage is subtracted from +7 in an accumulator, the result in the accumulator is +0.

An accumulator can be brought to zero by a subtract instruction, with the address of that accumulator, regardless of its sign. For example, to bring accumulator 1 to zero:

S01	23	45	6789
−14	00	09	9991

The sign of the accumulator is unchanged.

AUTOCODER EXAMPLE (Figure 22): Assume that BETA has been defined as positions 5-9 of word 3778. The assembled instruction is:

S01	23	45	6789
−24	00	59	3778

Multiply
+53 **M**

MACHINE DESCRIPTION: The arithmetic unit can multiply a 10-digit number by a 10-digit number to produce a 20-digit product. Either factor can be plus or minus, and the sign of the product conforms to the rules of algebra: plus times plus equals plus, plus times minus equals minus, and minus times minus equals plus. If either sign is alpha, the sign of the product is alpha.

The multiplicand must be placed in accumulator 3 in a previous instruction. The multiply operation multiplies this number by the field-defined portion of the word addressed by positions 6-9 (indexable). The product is developed in accumulators 1 and 2, but is not affected by the previous contents of either accumulator; they are automatically reset before the multiply operation begins.

At the completion of the operation, the low-order portion of the product is in accumulator 2, with the units position in the units position of the accumulator (if the product is 10 digits or less, the entire product is in accumulator 2). The multiplicand is in accumulator 3 at completion of the operation.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60	65 70 75	
0 1		S 2	B E T A		

FIGURE 22.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+53
23	Indexing word
45	Field definition of the multiplier
6789	Core-storage address of the multiplier

EXAMPLES: To multiply the 10-digit number in word 1230 by the 10-digit number in word 1735:

S01	23	45	6789	
+33	00	09	1230	(multiplicand to acc. 3)
+53	00	09	1735	(20-digit product, developed in acc. 1 and 2)

To multiply positions 0-6 of word 3310 by positions 3-9 of word 3316:

S01	23	45	6789	
+33	00	06	3310	(multiplicand to acc. 3)
+53	00	39	3316	(14-digit product, 10 low-order positions in acc. 2, 4 high-order positions in acc. 1)

DATA FLOW: Multiplication is accomplished by the *quadrupler* method. At the start of the operation, the multiplier is brought from core storage to accumulator 2, to the low-order portion if less than 10 digits are field-defined. Then, factors of 1, 2, and 4 times the value of the multiplicand in accumulator 3 are developed and stored in 3 registers (M = value of multiplicand):

M in the arithmetic register
2M in accumulator 3
4M in the auxiliary register

If M is a 10-digit number, 2M and 4M may become 11-digit numbers. Accumulator 3 and the auxiliary register each have an extra position, used in multiply operations, to take care of this possibility.

In the multiply operations, these factors are added together as determined by each multiplier digit, as follows:

Multiplier-digit value	Partial product developed
1	M
2	2M
3	2M + M
4	4M
5	4M + M
6	4M + 2M
7	4M + 2M + M
8	4M + 4M
9	4M + 4M + M

The multiplier digits are used in right-to-left (low-order to high-order) sequence. The M values are added as previously listed, for the units digit of the multiplier (in the units position of accumulator 2), to accumulator 1. Accumulators 1 and 2 are shifted one position to the right, dropping off the multiplier units digit and shifting the units position of the partial product from accumulator 1 to the high-order position of accumulator 2.

The process repeats for the 10's digit of the multiplier (now in the units position of accumulator 2). The M values of the multiplicand are added according to this digit, and this total is added to the partial product in accumulator 1. Another right-shift takes place, and the process repeats for each multiplier digit. If a multiplier digit is zero, there is a shift, but no adding of an M value. At the conclusion of the multiply operation, M is returned to accumulator 3.

REGISTERS AFFECTED: All accumulators, the arithmetic register, the auxiliary register, and the adder. (Accumulator 3 contains the multiplicand before and after the operation.)

TIMING: The duration of a multiply operation depends on the values of the multiplier digits as well as the number of them. The quadrupler method requires 1, 2, or 3 additions of the multiplicand multiples to obtain each partial product, depending on the value of the multiplier digit.

The formula in timing a multiply operation is as follows: Total number of microseconds = 48 [(number of 1's, 2's, and 4's in the multiplier) +2 (number of 3's, 5's, 6's, and 8's in the multiplier) +3 (number of 7's and 9's in the multiplier) + 4 (number of zeros in the multiplier) + 4 (number of zero groups in the multiplier)] + 180. If the total is not a multiple of 12, use next highest multiple of 12.

A zero group is a zero or two-or-more adjacent zeros. For timing purposes, the multiplier is considered a 10-digit number. Therefore, if field definition specifies a multiplier of less than 10 digits, the high-order zeros constitute a zero group.

COMMENTS: A number is multiplied by itself, if accumulator 3 is the address of the multiply instruction. If the value N is in word 0504, these two instructions obtain N^2:

	S01	23	45	6789
ZA3	+33	00	09	0504
M	+53	00	09	9993

If this instruction is next, N^3 is obtained:

M	+53	00	09	9992

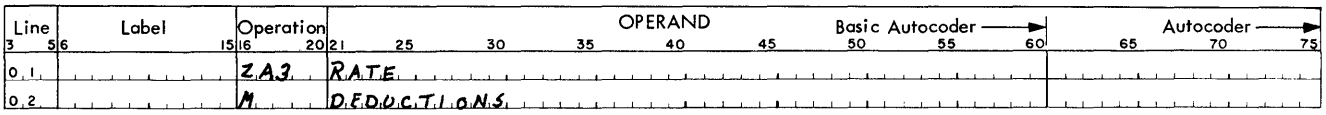


FIGURE 23.

Repeating this instruction obtains N^4, etc. If the contents of accumulator 2 are stored in accumulator 3 after each multiplication, the values obtained are N^2, N^4, N^8, etc., as long as the product values do not exceed 10 digit positions in size.

Polynomial nesting is the term given to adding variables to increasing powers of a constant:

$$a + bx + cx^2 + dx^3 + ex^4$$

This can be performed efficiently on the 7070, because of the fact that the multiplicand (x) needs to be brought to accumulator 3 only once. Each product is the multiplier to obtain the next product.

Instructions	Result in Accumulator #2
Zero-add x to Acc. #3	
Multiply by e	ex
Add d to Acc. #2	$d + ex$
Multiply x by contents of Acc. #2	$dx + ex^2$
Add c to Acc. #2	$c + dx + ex^2$
Multiply x by contents of Acc. #2	$cx + dx^2 + ex^3$
Add b to Acc. #2	$b + cx + dx^2 + ex^3$
Multiply x by contents of Acc. #2	$bx + cx^2 + dx^3 + ex^4$
Add a to Acc. #2	$a + bx + cx^2 + dx^3 + ex^4$

AUTOCODER EXAMPLE (Figure 23): Assume that RATE has been defined as positions 0-4 of word 2468, and DEDUCTIONS as positions 8 and 9 of the same word. The assembled instructions are:

S01	23	45	6789
+33	00	04	2468
+53	00	89	2468

Divide
−53 D

MACHINE DESCRIPTION: A 20-digit dividend is divided by a 10-digit divisor, to obtain a 10-digit quotient and a 10-digit remainder, with full sign control. Prior to the divide instruction, the dividend must be in accumulators 1 and 2, with the high-order portion in accumulator 1. The sign of accumulator 1 determines the sign of the dividend, in all cases. The divisor is the field-defined portion of the word addressed by positions 6-9 (indexable).

The absolute value of the divisor must be greater than the absolute value of the portion of the dividend in accumulator 1, at the start of the divide operation.

At the conclusion of the divide operation, the quotient is in accumulator 2, the remainder is in accumulator 1, and the divisor is in accumulator 3.

INSTRUCTION FORMAT: S01 23 45 6789

S01	−53
23	Indexing word
45	Field definition of divisor
6789	Address of divisor (indexable)

EXAMPLES: To divide a 20-digit number, in words 1656 and 1657 (high-order portion of dividend in 1656), by positions 5-9 of word 1707:

S01	23	45	6789
+13	00	09	1656
+23	00	09	1657
−53	00	59	1707

To divide a 12-digit number by a 4-digit number; the 10 low-order positions of the dividend are in word 2400, indexed by positions 2-5 of IW 62. The two high-order dividend digits are in positions 8-9 of the next word, 2401 indexed by IW 62; and the divisor is in positions 4-7 of that same word:

S01	23	45	6789
+23	62	09	2400
+13	62	89	2401
−53	62	47	2401

DATA FLOW: At the start of the divide operation, the divisor is reset-added into accumulator 3, to the low-order portion if less than 10 digits are field-defined (it is brought via the arithmetic register and the adder).

The signs of accumulators 1 and 3 are compared. If they are the same, accumulator 2, which will contain the quotient when the operation is completed, is given a plus sign. If they are different, accumulator 2 gets a minus sign. If either sign is alpha, accumulator 2 gets an alpha sign.

The high-order digit of accumulator 2 is joined with the 10 digits in accumulator 1, to form an 11-digit number; the digit from accumulator 2 is the units digit of this number. Accumulator 1 is able to contain 11 digits for this operation. The divisor in accumulator 3 is subtracted from this

11-digit number. (As the digits of the 11-digit number go to the adder for this operation, the 10 low-order digits go to the synchronizer register; they will be needed later.) A test is made to determine whether this subtraction brought the 11-digit number *below zero*. If it has not, a 1 is inserted into a special one-digit register, called the *quotient register*. The operation repeats — the divisor is again subtracted from the 11-digit number, and a 1 is added to the quotient register — until the *below-zero* condition is reached. (Each time, the 10 low-order digits of the 11-digit number are moved to the synchronizer register as they are brought to the adder for the subtract operation.) When the *below-zero* condition is reached, the correct high-order digit of the quotient is now in the one-digit register. This digit is moved to the high-order position of accumulator 2 (this position was vacated when the 11-digit number was formed).

Accumulator 2 is shifted one position to the left, moving the quotient digit around to the units position. The next 11-digit dividend consists of the 10-digit number in the synchronizer register; and the digit that was originally second high-order in accumulator 2, as the units digit. This number is brought to the adder along with the divisor, and the first subtraction operation toward obtaining the next quotient digit takes place. The result of this first subtraction goes to accumulator 1.

The operation continues until 10 quotient digits are developed. The synchronizer register contains the remainder, and this is moved to accumulator 1.

REGISTERS AFFECTED: All three accumulators, the arithmetic register, the adder, and the special quotient register.

TIMING: Division timing is determined by the sum of the quotient digits. For example, the sum of the digits in a quotient of 51326 is: $5 + 1 + 3 + 2 + 6 = 17$.

The formula for timing a division operation is as follows:

Total number of microseconds = 192 + 48 (10 + sum of the quotient digits).

COMMENTS: A divide operation always divides the entire contents of accumulator 3 into the entire contents of accumulators 1 and 2. For this reason, accumulators 1 and 2 must each be programmed to contain part of the dividend, or all zeros, prior to the divide instruction. Note that the sign of accumulator 2 in a divide operation is ignored; the sign of accumulator 1 is considered the sign of the entire 20-digit dividend. If the dividend is 10 digits or less, and goes entirely in accumulator 2 for the divide operation, accumulator 1 must be set to zero and given the correct sign.

Decimal-point alignment is often the determining factor in positioning the dividend in accumulators 1 and 2. The dividend can be positioned correctly by use of this formula, in which D represents the number of digit positions to the *right* of the decimal:

D in the divisor
+ D desired in the quotient
= D in the dividend

If the quotient is to be half-adjusted, the dividend should have one additional position to the right of the decimal. For example, total hours are divided into total miles, to obtain velocity:

The hours value has two decimal places:	hh.hh
It is desired to obtain velocity to three places:	vv.vvv
The miles are to two places:	mmm.mm

D in divisor h	= 2
D desired in quotient V	= 3
+ 1 for half-adjusting	= 1
total D needed for dividend M	= 6

The dividend itself has only two positions to the right of the decimal. Therefore, it should be positioned: mmm.mm0000.

The greater the dividend can be made in relation to the value of the divisor, the more exact the quotient will be. For example, the 10-digit number 00000 *22*000 divided by 00000 00007, is 00000 03142. The 10-digit number *22*000 00000 divided by 00000 00007, is 03142 85714. Keep in mind, however, that the absolute value of that portion of the dividend in accumulator 1 must be less than the absolute value of the divisor. (Accumulator 1 cannot be used for the divisor, because of this rule; accumulator 2 or 3 can, however.)

The sign of the quotient is determined by the signs of the divisor and dividend. If they are the same, the quotient is plus; if they are different, the

quotient is minus. The remainder always has the same sign as the dividend (Figure 24).

If the divisor is:	and the dividend is:	the quotient is:	and the remainder is:
+	+	+	+
+	−	−	−
−	+	−	+
−	−	+	−

FIGURE 24. DIVIDE — PLUS AND MINUS FACTORS

If either the divisor or dividend is alphabetic, the quotient is alphabetic (Figure 25).

If the divisor is:	and the dividend is:	the quotient is:	and the remainder is:
+	Alpha	Alpha	Alpha
−	Alpha	Alpha	Alpha
Alpha	+	Alpha	+
Alpha	−	Alpha	−
Alpha	Alpha	Alpha	Alpha

FIGURE 25. DIVIDE — ALPHA FACTORS

AUTOCODER EXAMPLE (Figure 26):

MON: Positions 6-9 of word 2101
TUES: Word 2102
WED: Positions 0-5 of word 1789

The assembled instructions are:

S01	23	45	6789
+13	00	69	2101
+23	00	09	2102
−53	00	05	1789

Zero Accumulator 1 and Add Absolute
+16 ZAA

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is brought to accumulator 1, to the low-order portion if less than 10 digits are field-defined. Regardless of the storage-word sign, +, −, or alpha, the accumulator becomes plus.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +16
23 Indexing word
45 Field definition
6789 Address of data word

EXAMPLES: To move the four high-order positions of word 1689 to accumulator 1, and give it a plus sign:

S01	23	45	6789
+16	00	03	1689

Contents of word 1689: −34517 78400. Contents of accumulator 1 after operation: +00000 03451, regardless of previous contents.

DATA FLOW: Same as for zero accumulator # and add, with this exception: After the high-order digit of the field is brought from the adder to the arithmetic register, the sign in the arithmetic register is set to 9 (+).

REGISTERS AFFECTED: The arithmetic register, the adder, and accumulator 1.

TIMING: Same as ZA#.

COMMENTS: This instruction can be used with accumulator 1 only. Thus, the number 1 is not needed in the *Autocoder* mnemonic.

The value in accumulator 1 can be guaranteed to have a plus value, regardless of its previous sign, if 9991 is the address of a ZAA instruction:

S01	23	45	6789
+16	00	09	9991

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
0 1		ZA1	MON		
0 2		ZA2	TUES		
0 3		D	WED		

FIGURE 26.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
0 1		ZAA	15371(5,9)		

FIGURE 27.

AUTOCODER EXAMPLE (Figure 27): The actual address 1537 is used. The assembled instruction is:

S01	23	45	6789
+16	00	59	1537

Zero Accumulator 1 and Subtract Absolute
−16 ZSA

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is brought to accumulator 1, to the low-order portion if less than 10 digits are field-defined. Regardless of the storage-word sign, +, −, or alpha, the accumulator becomes minus.

INSTRUCTION FORMAT: Same as ZAA, except for the sign.

EXAMPLE: To move all of word 1648 to accumulator 1, and make it minus:

S01	23	45	6789
−16	00	09	1648

Contents of word 1648: −46175 81421. Contents of accumulator 1 after the operation: −46175 81421, regardless of previous contents.

DATA FLOW: Same as zero accumulator # and add, with this exception: After the high-order digit of the field is brought from the adder to the arithmetic register, the sign in the arithmetic register is made minus.

REGISTERS AFFECTED: The arithmetic register, the adder, and accumulator 1.

TIMING: Same as ZA#.

COMMENTS: This instruction can be used with accumulator 1 only. Thus, the number 1 is not needed in the *Autocoder* mnemonic.

AUTOCODER EXAMPLE (Figure 28): The assembled instruction is:

S01	23	45	6789
−16	39	09	1919

Add Absolute to Accumulator 1
+17 AA

MACHINE DESCRIPTION: The field-defined portion of the word addressed by positions 6-9 (indexable) is added to the number already in accumulator 1. The number from storage is always considered plus, regardless of its sign in storage; the number in accumulator 1 can be plus, minus, or alpha. The result is in the accumulator after the operation. If accumulator 1 is alpha, the result is alpha.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+17
23	Indexing word
45	Field definition
6789	Address of data word

EXAMPLES: To add the absolute value of positions 5-9 of word 1617 to the amount already in accumulator 1:

S01	23	45	6789
+17	00	59	1617

Contents of accumulator 1 before the operation: +00000 05456. Contents of word 1617: −00000 00302. Contents of accumulator 1 after the operation: +00000 05758.

To add the absolute value of positions 0-2 of word 3735 to the amount already in accumulator 1:

S01	23	45	6789
+17	00	02	3735

Contents of accumulator 1 before the operation: −00000 00021. Contents of word 3735: −03500 00000. Contents of accumulator 1 after the operation: +00000 00014.

DATA FLOW: Same as add to accumulator #, with this exception: the value in the arithmetic register is considered plus, regardless of its sign. Thus, the sign of the accumulator value (now in the auxiliary register) determines whether the result will be the sum or the difference of the two values. This determines whether the digits from the arithmetic register are complemented before going to the adder.

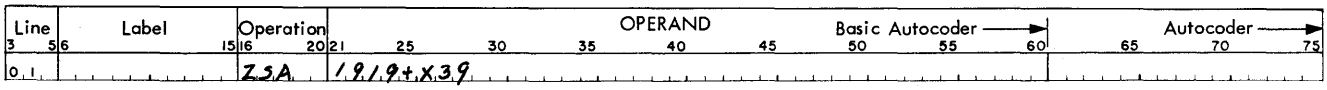


FIGURE 28.

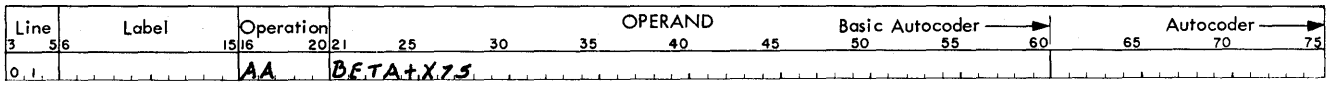


FIGURE 29.

REGISTERS AFFECTED: The arithmetic register, the adder, and accumulator 1.

TIMING: Same as add to accumulator #. True-add or complement-add is determined solely by the sign of the accumulator value: if it is plus or alpha, the operation is true-add; if it is minus, the operation is complement-add.

COMMENTS: This operation can be used with accumulator 1 only. The sum of the two values is obtained if the accumulator is plus or alpha prior to the operation, and the difference is obtained if the accumulator is minus. If the storage-word sign is alpha, its value is considered plus, but its sign is ignored. If the accumulator is alpha, its value is considered plus, and the resultant sign is alpha.

AUTOCODER EXAMPLE (Figure 29): Assume that BETA has been defined as positions 4-7 of word 1671. The assembled instruction is:

S01	23	45	6789
+17	75	47	1671

Subtract Absolute from Accumulator 1

−17 **SA**

MACHINE DESCRIPTION: The field-defined portion of the word addressed by positions 6-9 (indexable) is added to the number already in accumulator 1. The number from storage is considered minus, regardless of its sign in storage. The number in accumulator 1 can be plus, minus, or alpha. The result is in the accumulator after the operation.

INSTRUCTION FORMAT: Same as AA, except for the sign.

EXAMPLES: To subtract the absolute value of positions 7-9 of word 3331 from the amount already in accumulator 1:

S01	23	45	6789
−17	00	79	3331

Contents of accumulator 1 before the operation: +00000 00456. Contents of word 3331: +63457 19406. Contents of accumulator 1 after the operation: +00000 00050.

To subtract the absolute value of word 3518 from accumulator 1:

S01	23	45	6789
−17	00	09	3518

Contents of accumulator 1 before the operation: +00000 01234. Contents of word 3518: +00000 71234. Contents of accumulator after the operation: −00000 70000.

DATA FLOW: Same as add to accumulator #, with this exception: The value in the arithmetic register is considered minus, regardless of its sign. Thus, the sign of the accumulator value (now in the auxiliary register) determines whether the result will be the sum or the difference of the two values. This determines whether the digits from the arithmetic register are complemented before going to the adder.

REGISTERS AFFECTED: The arithmetic register, the adder, and accumulator 1.

TIMING: Same as add to accumulator #. True-add or complement-add is determined solely by the sign of the accumulator value: if it is minus, the operation is true-add; if it is plus or alpha, the operation is complement-add.

COMMENTS: The sum of the two values is obtained if the accumulator is minus prior to the operation, and the difference is obtained if the accumulator is plus or alpha. If the storage-word sign is alpha, its value is considered minus, and the sign is ignored. If the accumulator sign is alpha, its value is considered plus, and the resultant sign is alpha regardless of its value.

AUTOCODER EXAMPLE (Figure 30): GAMMA has been defined as word 2550. The assembled instruction is:

S01	23	45	6789
−17	00	09	2575

Field Overflow

Field overflow occurs in each of the following conditions:

1. A store or add-to-storage type of operation in which the number of significant digits in the accumu-

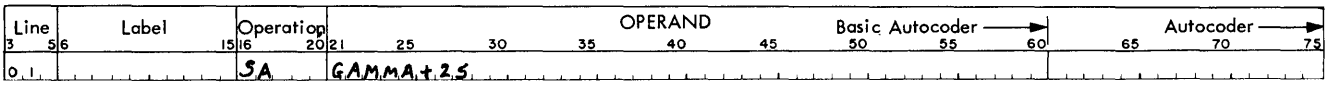


FIGURE 30.

lator is greater than field definition allows to be stored in the word.

2. An "add-to-storage" operation in which a carry is propagated beyond the high-order position of the field defined.

When either situation arises, only the digits that field definition allows are stored. The *field overflow* indicator is turned on, and the machine either continues or stops, depending on the setting of the field-overflow stop/sense switch. This switch is set to "sense" or "stop" by augmented code +41: sense mode for the field overflow (SMFV) and halt mode for field overflow (HMFV). If the *field-overflow* indicator is on, and the program changes the field-overflow stop/sense switch to "stop," the machine immediately stops. The operation that tests the indicator: BFV, branch if field overflow (+41), turns it off if it is on.

Sign Change

If a store or add-to-storage type of operation brings data to a field-defined portion of a storage word, and the remaining positions of the word are not set to zero, a difference in sign between the storage word and the data brought to it is detected by the machine. The field is stored in the word, and the sign is changed. The *sign-change* indicator is turned on, and the program either stops or continues, depending on setting of the sign-change sense/stop switch. This switch is set to "sense" or "stop" by an augmented operation code, −03: sense mode on sign change (SMSC) and halt mode on sign change (HMSC). If the *sign-change* indicator is on, and the program changes, the sign-control stop/sense switch to "stop," the machine immediately stops. The operation that tests the indicator: BSC, branch if sign change (−03), turns it off if it is on.

Zero Storage and Store Accumulator # −11, −21, −31 ZST1, ZST2, ZST3

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is replaced by an equivalent number of digits from the specified accumulator, from the low-order portion if less than 10 digits are field-defined. The remaining positions of the storage word are set to zero, and the sign is set to the sign of the accumulator. Field overflow is possible, but sign change is not.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S	Always minus
0	Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.
1	Always 1
23	Indexing word
45	Field definition
6789	Address of storage word in which the data is to be stored

EXAMPLES: To zero storage word 1860 and store data from accumulator 2, into positions 3-4.

S01	23	45	6789
−21	00	34	1860

Contents of accumulator 2: +00000 00023. Contents of 1860 after the operation: +00023 00000, regardless of previous contents.

To zero storage word 2816 and store data from accumulator 1, into positions 1-4.

S01	23	45	6789
−11	00	14	2816

Contents of accumulator 1: −00000 54231. Contents of 2816 after the operation: −04231 00000, regardless of previous contents. The *field-overflow* indicator is turned on, because there are five significant digits in the accumulator and only four digits are field-defined.

DATA FLOW: The arithmetic register is set to zeros. The contents of the specified accumulator are brought to the auxiliary register. The digit in the units position of the auxiliary register is brought to the arithmetic register, to the low-order position of the field (specified in position 5 of the instruction). The other digits are moved in the same manner, one at a time from low-order to high-order, until the high-order position of the field, (specified in position 4 of the instruction), is filled. If there are any significant digits in the auxiliary register to the left of

the last digit moved, the *field overflow* indicator is turned on. The sign of the arithmetic register is set to that of the auxiliary register. The contents of the arithmetic register are sent in parallel to the storage word. If 10 digits are field-defined, the entire contents of the auxiliary register are moved in parallel to the arithmetic register, and from there to the storage word.

REGISTERS AFFECTED: The arithmetic register. (The accumulator is unchanged by this instruction; the storage word is changed of course.)

TIMING:

Number of Digit Positions	1	2	3	4	5	6	7	8	9	10
Microseconds	48	48	60	60	60	72	72	72	84	36

COMMENTS: If there are more significant digits in the accumulator than field-definition allows to be stored, the *field-overflow* indicator is turned ON. The correct number of digits is stored.

Note that the contents of the storage word prior to this operation are lost completely and have no effect on the contents of the word after the operation.

AUTOCODER EXAMPLE (Figure 31): DELTA has been previously defined as word 1500. The assembled instruction is:

S01	23	45	6789
−31	43	04	1500

Store Accumulator #
+12, +22, +32 **ST1, ST2, ST3**

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is replaced by an equivalent number of digits from the specified accumulator, from the low-order portion if less than 10 digits are field-defined. The sign of the storage word is set to that of the accumulator. The remaining positions of the storage word are unchanged. Field-overflow and sign-change indications are both possible.

INSTRUCTION FORMAT: S 0 1 23 45 6789

- S — Always plus
- 0 — Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3
- 1 — Always 2
- 23 — Indexing word
- 45 — Field definition
- 6789 — Address of storage word in which the data is to be stored.

EXAMPLES: To store data from accumulator 1 in positions 4-7 of word 1820, set the sign to that of accumulator 1, and not disturb other positions of the word:

S01	23	45	6789
+12	00	47	1820

Contents of word 1820 before the operation: −22222 33333. Contents of accumulator 1: +00000 00678. Contents of word 1820 after the operation: +22220 67833. The *sign-change* indicator is turned on, because the accumulator and storage-word signs were different.

To store one digit from accumulator 3 into position 6 of word 2000, indexed by positions 2-5 of IW 54:

S01	23	45	6789
+32	54	66	2000

Contents of the storage word before the operation: +43444 53355. Contents of accumulator 3: +00010 00009. Contents of the storage word after the operation: +43444 59355. The *field-overflow* indicator is turned on, because the digit 1 in position 3 of the accumulator indicates that the accumulator number is larger than storage-location field.

DATA FLOW: The contents of the storage word are brought in parallel from core storage to the arithmetic register. From this point, data flow is the same as for zero storage and store. The contents of the specified accumulator are brought to the auxiliary register. The digits are moved, one digit at a time from low-order to high-order, from the accumulator to the field-defined digits of the arithmetic register, until the high-order position of the field has been filled. If there are any significant digits in the auxiliary register to the left of the last digit moved, the *field-overflow* indicator is turned on. The signs of the arithmetic register and auxiliary register are compared. If they are unequal, the *sign-change* indicator is turned on. The sign of the arithmetic register is made the same as that of the accumulator. The entire contents of the arithmetic register are moved in parallel to the storage word.

REGISTERS AFFECTED: Same as ZST#

TIMING: Same as ZST#

COMMENTS: Note that field definition locates the exact positions of the storage word to which the accumulator data goes. It also defines the number of

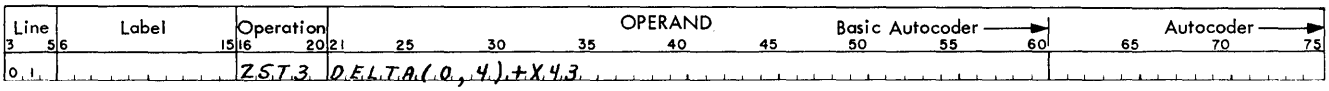


FIGURE 31.

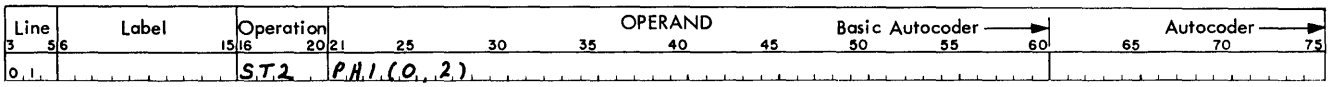


FIGURE 32.

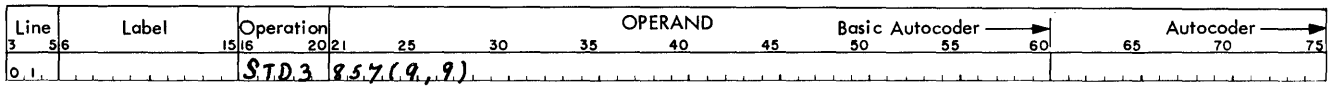


FIGURE 33.

positions in the accumulator, starting from the right, that are to be stored. This is true of all store-accumulator instructions.

AUTOCODER EXAMPLE (Figure 32): Assume that PHI has been previously defined as the 5 low-order positions of word 4430. The assembled instruction is:

S01	23	45	6789
+22	00	57	4430

Store Digits from Accumulator # and Ignore Sign −12, −22, −32 — STD1, STD2, STD3

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is replaced by an equivalent number of digits from the specified accumulator, from the low-order portion if less than 10 digits are field-defined. The remaining positions and sign of the storage word are unchanged.

This code differs from the store-accumulator instruction, in that the sign of the accumulator is ignored. Thus, the *sign-change* indicator cannot be turned on by this instruction. Field overflow is possible, however.

INSTRUCTION FORMAT: Same as ST#, except for sign.

EXAMPLES: To store digits from accumulator 1 in positions 6-9 of word 4147, and ignore the sign of accumulator:

S01	23	45	6789
−12	00	69	4147

Contents of word 4147 before the operation: @71857 59199. Contents of accumulator 1: +00000 09290. Contents of word 4147 after the operation: @71857 59290. The *sign-change* indicator is not turned on, even though the signs are different. (Field overflow does not occur in this particular example.)

To store digits from accumulator 2 in positions 8-9 of word 3214 and ignore the sign of accumulator 2:

S01	23	45	6789
−22	00	89	3214

Contents of 3214 before the operation: −55555 55555. Contents of accumulator 2: +00000 00123. Contents of 3214 after the operation: −55555 55523. The *field-overflow* indicator is turned on, but the *sign-change* indicator is not.

DATA FLOW: Same as store accumulator, with this exception: The sign of the arithmetic register is not made the same as that of the accumulator; it remains unchanged.

REGISTERS AFFECTED: Same as ZST#

TIMING: Same as ZST#

COMMENTS: The only difference between this code and store accumulator is that this code cannot turn on the *sign-change* indicator if the accumulator and storage-word signs are different. If the signs are the same, this instruction is exactly the same as store accumulator.

AUTOCODER EXAMPLE (Figure 33): The actual address 857 is used. The assembled instruction is:

S01	23	45	6789
−32	00	99	0852

Add to Storage from Accumulator #
+18, +28, +38 **AS1, AS2, AS3**

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is added to the entire contents of the specified accumulator, and the result is stored in the same field of the storage word. The storage word is given the sign of the result. The accumulator value is unchanged. The sign of each factor is taken into consideration. Both sign change and field overflow are possible.

The *sign-change* indicator is not turned on, however, if a full word is field-defined (09 in positions 4-5). In this case, the sign of the storage word can be changed by a complement-add operation, but the *sign-change* indicator is not turned on.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S	Always +
0	Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.
1	Always 8
23	Indexing word
45	Field definition
6789	Address of storage word that contains one of the factors prior to the operation, and the result, after the operation.

EXAMPLES: To add the contents of accumulator 1 to the five low-order positions of word 3279, and store the result in those positions:

S01	23	45	6789
+18	00	59	3279

Contents of 3279 before the operation: +22222 51434. Contents of accumulator 1: −00000 00121, unchanged by the operation. Contents of 3279 after the operation: +22222 51313. Neither the *field-overflow* nor *sign-change* indicator is turned on.

To add the contents of accumulator 2 to the three low-order digits of word 2754, and store the result in those positions:

S01	23	45	6789
+28	00	79	2754

Contents of 2754 before the operation: +01230 88454. Contents of accumulator 2: +00000 01111, unchanged by the operation. Contents of 2754 after the operation: +01230 88565. The *field-overflow* indicator is turned on, due to the presence of four significant digits in the accumulator. There is no sign change.

To add the contents of accumulator 3 to the entire contents of word 1547:

S01	23	45	6789
+38	00	09	1547

Contents of 1547 before the operation: +00000 00567. Contents of accumulator 3: −00000 00670, unchanged by the operation. Contents of 1547 after the operation: −00000 00103. Although the sign of the storage word is changed by the operation, the *sign-change* indicator is not turned on, because a full word was field-defined.

DATA FLOW: Data flow for this operation is the same as for add to accumulator #, with three exceptions:

1. The resultant digits developed by the adder are brought to the field-defined positions of the arithmetic register, rather than to the low-order portion.
2. After the result is developed in the arithmetic register, the entire contents of the arithmetic register are moved in parallel to the core-storage word, instead of to the accumulator. (The contents are sent "as is," with no insertion of high-order zeros.)
3. If the operation is complement-add, the accumulator value in the auxiliary register is converted to its complement (rather than the storage value in the arithmetic register).

In all complement-add operations, the value that is to be replaced by the result is complemented. In add/subtract to accumulator operations, it is the value in the auxiliary register; for add/subtract to storage operations, it is the value in the arithmetic register.

REGISTERS AFFECTED: Arithmetic register, adder, auxiliary register. The specified accumulator is unchanged.

TIMING: As in the add- and subtract-to-accumulator codes, the timing of an add-to-storage instruction is determined by the number of digits field-defined; and on whether the operation is true-add, complement-add, or complement-add with recomplement.

Number of Digit Positions	1	2	3	4	5	6	7	8	9	10
	MICROSECONDS									
True Add to Storage	48	48	60	60	60	72	72	72	84	84
Complement Add to Storage	48	48	60	60	60	72	72	72	84	84
If Recomplemented	60	72	84	84	96	108	108	120	132	132

FIGURE 34. TIMING — ADD TO STORAGE FROM ACCUMULATOR #

In add-to-storage operations, recomplement is required for complement-add operations in which the accumulator value is greater than the storage-word value.

Timing of add-to-storage operations is shown in Figure 34.

COMMENTS: An add-to-storage type of instruction reverses the normal functions of storage word and accumulator. The accumulator contains data to be added, and the storage word becomes the "accumulator." It contains one of the factors to be added, and after the operation, it contains the result.

The signs of both factors are considered; the sum is obtained if they are the same, and the difference is obtained if they are different. If either sign is alpha, its value is considered plus; but the sign of the result is alpha regardless of its value.

The storage word always takes the sign of the result. If less than a full word is specified by field definition, and the sign of the result is different than the sign of the original storage word, a sign-change indication is given. If the operation causes a carry to a position beyond the field defined, or the accumulator contains a larger number than field definition allows, a field-overflow indication is given.

AUTOCODER EXAMPLE (Figure 35): KAPPA has been defined as word 1929. The assembled instruction is:

S01	23	45	6789
+38	00	59	1929

Subtract Accumulator # from Storage
−18, −28, −38 SS1, SS2, SS3

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable) is subtracted from the entire contents of the specified accumulator, and the result is stored in the same field of the storage word. The accumulator value is unchanged. The sign of each factor is taken into consideration. The sign of the storage word is set to the sign of the result. Sign-change and field-overflow are both possible. However, as with the add-to-storage codes, the *sign-change* indicator is not turned on if a full word is field-defined. The sign of the storage word can be changed, but the indicator is not turned on.

INSTRUCTION FORMAT: Same as add-to-storage, except the sign.

EXAMPLES: To subtract the contents of accumulator 3 from all of word 0950:

S01	23	45	6789
−38	00	09	0950

Contents of 0950 before the operation: +00000 12300. Contents of accumulator 3: +00000 30000, unchanged by the operation. Contents of 0950 after the operation −00000 17700. Although the sign of the storage word is changed by the operation, the *sign-change* indicator is not turned on, because a full word was field-defined.

To subtract the contents of accumulator 2 from the two low-order positions of word 1781:

S01	23	45	6789
−28	00	89	1781

Contents of 1781 before the operation: +00000 00045. Contents of accumulator 2: +00000 00145, unchanged by the operation. Contents of 1781 after the operation: +00000 00000. The *field-overflow* indicator is turned on, because of three significant digits in the accumulator, with only two digits specified by field definition. There is no sign-change.

DATA FLOW: Same as AS#

REGISTERS AFFECTED: Same as AS#

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60	65 70 75	
0 1		A.S.3.	K.A.P.P.A.(5,9).		

FIGURE 35.

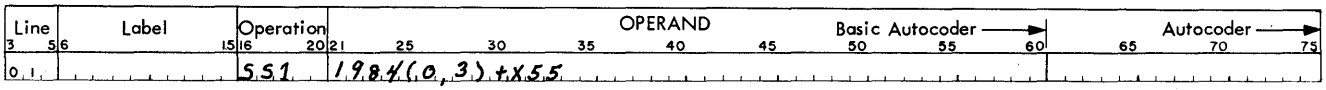


FIGURE 36.

TIMING: Same as AS#

COMMENTS: This instruction has the same relation with add-to-storage as a subtract operation has with add: the operation obtains either the sum or the difference of the two values, depending on the sign of each factor. If the signs are the same, this code obtains the difference. If the signs are different, it obtains the sum.

AUTOCODER EXAMPLE (Figure 36): Actual address 1984 is used. The assembled instruction is:

S01	23	45	6789
−18	55	03	1984

Add to Absolute Storage from Accumulator # +19, +29, +39 — AAS1, AAS2, AAS3

MACHINE DESCRIPTION: The field-defined portion of the storage word addressed by positions 6-9 (indexable), considered plus, is added algebraically to the entire contents of the specified accumulator. The result is stored in the same field of the storage word; the sign of the storage word is unchanged. The accumulator is unchanged by the operation. Field overflow is possible, but sign change is not.

INSTRUCTION FORMAT: S 0 1 23 45 6789

- S — Always +
- 0 — Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.
- 1 — Always 9
- 23 — Indexing word
- 45 — Field definition
- 6789 — Address of storage word that contains one of the factors prior to the operation; and the result, after the operation. (Value of the factor considered plus, regardless of sign of the storage word.)

EXAMPLES: To add the contents of accumulator 3 to the absolute value in positions 5-9 of word 2350, and store the result in those positions:

S01	23	45	6789
+39	00	59	2350

Contents of word 2350 before the operation: −98765 34343. Contents of accumulator 3: +00000 22222, unchanged by the operation. Contents of word 2350 after the operation: −98765 56565.

To add the contents of accumulator 1 to the absolute value in position 9 of word 1975, and store the result in that position:

S01	23	45	6789
+19	00	99	1975

Contents of 1975 before the operation: −00000 00006. Contents of accumulator 1: +00000 00008. Contents of 1975 after the operation: −00000 00004. The *field-overflow* indicator is turned on.

DATA FLOW: Same as AS#

REGISTERS AFFECTED: Same as AS#

TIMING: Same as AS#

COMMENTS: This is an unusual operation code, because in some circumstances it goes against the rules of arithmetic. The reason for this is that the value in the storage word is considered plus regardless of its sign, and is never changed, regardless of the result. Figure 37 is a chart of the combinations of two values, 8 and 6, and the results obtained by this code, when one position is field-defined.

Note that the results are the same for each combination of signs, when the values are reversed. If either factor is alpha, its value is considered plus.

VALUE IN STORAGE	VALUE IN ACCUMULATOR	RESULT IN STORAGE	
+6	+8	+4	Field Overflow
+6	−8	+2*	
−6	+8	−4*	Field Overflow
−6	−8	−2*	
+8	+6	+4	Field Overflow
+8	−6	+2	
−8	+6	−4*	Field Overflow
−8	−6	−2*	

*These results are arithmetically incorrect

FIGURE 37. RESULTS — ADD TO ABSOLUTE STORAGE FROM ACCUMULATOR

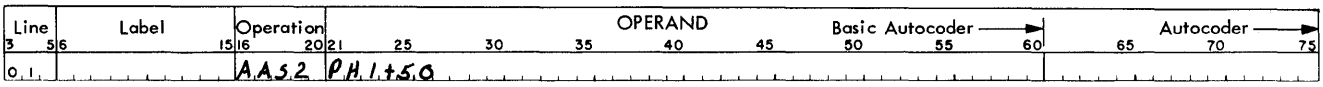


FIGURE 38.

The purpose of this operation code is modification of instructions. Because an instruction can have either a plus or minus sign, the value of the result would be affected by the sign, if an AS# or SS# instruction were used.

Note that neither factor ever has its sign changed by this operation.

Accumulator addresses can be used with this operation. If the instruction uses the same accumulator (AAS3 from accumulator 3 for example), the value in the accumulator is doubled if its sign is plus or alpha, or set to zeros if its sign is minus.

AUTOCODER EXAMPLE (Figure 38): PHI has been previously defined as positions 6-9 of word 2500. The assembled instruction is:

S01	23	45	6789
+29	00	69	2550

Shift Control
+50

MACHINE DESCRIPTION: The shift-control operations cause the contents of an accumulator to be moved to the right or left within the accumulator a specified number of positions. The digits moved off the end are lost, and the vacated positions at the other end are filled with zeros.

This operation code is augmented; the specific operation to be performed is defined in position 7 of the instruction.

INSTRUCTION FORMAT: S01 23 45 6 7 89

S01 +50

23 Indexing word. Even though positions 6-9 do not represent an address, they are modified by positions 2-5 of an IW, if one is specified here.

45 Index word, for shift and count operations. The number of places shifted (00-10) replaces positions 4-5 of the index word specified here (positions 2-3 of the index word are set to zero, and its sign is set to plus). These positions are not used in the other shift operations.

6 1-3, designating the accumulator to be shifted

7 0-3, designating the type of shift:
0. shift right accumulator # SR1, SR2, SR3
1. Shift right and round accumulator # SRR1, SRR2, SRR3
2. Shift left accumulator # SL1, SL2, SL3
3. Shift left and count accumulator # SLC1, SLC2, SLC3

89 00-10, designating the number of positions to be shifted. Not used in shift and count operations.

SHIFT RIGHT ACCUMULATOR #: The entire contents of the accumulator designated in position 6 are shifted to the right the number of positions specified in positions 8-9 of the instruction.

SHIFT RIGHT AND ROUND ACCUMULATOR #: This code is the same as shift right, with the additional feature that the amount shifted is rounded off. A 5 is added to the highest-order digit that is to be dropped off the edge; and if there is a carry, the next digit is increased by 1.

SHIFT LEFT ACCUMULATOR #: This is the same as shift right, except that the digits are moved to the left. Zeros are inserted into the vacated low-order positions.

SHIFT LEFT AND COUNT ACCUMULATOR #: The digits in the accumulator designated in position 6 are shifted to the left, until a digit other than a zero is in the high-order position. If the high-order digit is non-zero to start with, no shift takes place. Because the instruction does not determine the number of positions to be shifted, positions 8-9 are not used, but must be a valid shift number (00-10).

The number of positions shifted is recorded in an index word, specified in positions 4-5 of the instruction. This number (00-10) is recorded in positions 4-5 of the IW; positions 2-3 of the IW are set to zero, and the remaining positions are unchanged. The sign of the index word is set to plus.

EXAMPLES: To shift the contents of accumulator 2 five positions to the right:

S01	23	45	6	7	89
+50	00	xx	2	0	05

Contents of accumulator 2 before the operation: + 12345 67890. Contents of accumulator 2 after the operation: + 00000 12345.

To shift the contents of accumulator 2 five positions to the right, and round-off the number:

S01	23	45	6	7	89
+50	00	xx	2	1	05

Contents of accumulator 2 before the operation: + 12345 67890. Contents of accumulator 2 after the operation: + 00000 12346.

To shift the contents of accumulator 3 one position to the left:

S01	23	45	6	7	89
+50	00	xx	3	2	01

Contents of accumulator 3 before the operation: −12000 00000. Contents of accumulator 3 after the operation: −20000 00000.

To shift left and count in accumulator 1:

S01	23	45	6	7	89
+50	00	17	1	3	yy

Contents of accumulator 1 before the operation: −00002 40390. Although positions 8-9 are not used, they must contain a valid shift number (00-10). Contents of accumulator 1 after the operation: −24039 00000. Contents of IW 17 before the operation: −00 2425 0015. Contents of IW 17 after the operation: +00 0004 0015.

DATA FLOW: The contents of the specified accumulator are transferred in parallel to the synchronizer register, where the shifting actually takes place. The shifted contents of the synchronizer register are sent to the accumulator, replacing what was there.

REGISTERS AFFECTED: The synchronizer register and the accumulator.

TIMING: Figure 39

COMMENTS: A +50 shift operation never changes the sign of the accumulator. A shift of 10 positions sets the entire accumulator to zeros but does not change the sign. It has the same result as an operation that subtracts the contents of an accumulator from itself.

A shift in excess of 10 positions (11-99 in positions 8-9) is signalled as an error, even in shift and count operations. If positions 8-9 are 00, this instruction is the same as NOP except for shift and count.

ABBR.	NAME	NO. OF POS. SHIFTED	TIME IN MICROSECONDS
SR#	Shift Right Accumulator #	0-4	36
SL#	Shift Left Accumulator #	5-7	48
		8-10	60
SRR#	Shift Right and Round Accumulator #	0	36
		1-3	72
		4-6	84
		7-9	96
		10	108
SLC	Shift Left and Count Accumulator #	0-4	72
		5-7	84
		8-10	96

FIGURE 39. TIMING — SHIFT CONTROL

AUTOCODER EXAMPLES (Figures 40-43): The instruction assembled from Figure 40 is:

S01	23	45	6	7	89
+50	00	00	2	0	03

The instruction assembled from Figure 41 is:

S01	23	45	6	7	89
+50	00	00	3	1	02

The instruction assembled from Figure 42 is:

S01	23	45	6	7	89
+50	00	00	1	2	07

The instruction assembled from Figure 43 is:

S01	23	45	6	7	89
+50	00	91	2	3	00

Coupled Shift Control −50

MACHINE DESCRIPTION: This augmented code is used with accumulators 1 and 2 coupled, with accumulator 1 as the high-order accumulator. The operation is either normal-shift or split-shift. Normal-shift operates the same as the shift operation for a single accumulator (+50), except that it operates a single 20-position accumulator, composed of accumulators 1 and 2. Split-shift operations cause the 20-digit number to be "broken apart" between any two positions, with the positions, direction of shift and number of positions shifted all specified in the instruction (Figure 44).

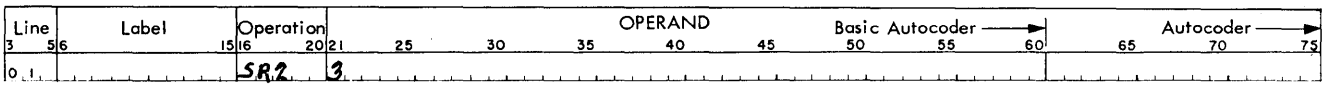


FIGURE 40.

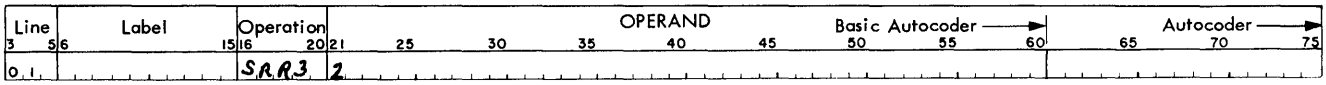


FIGURE 41.

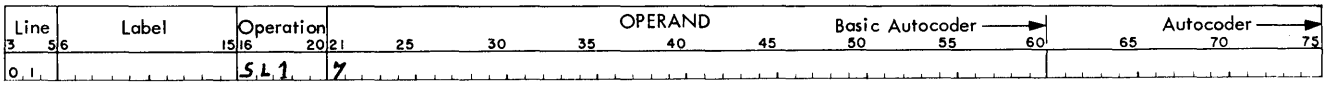


FIGURE 42.

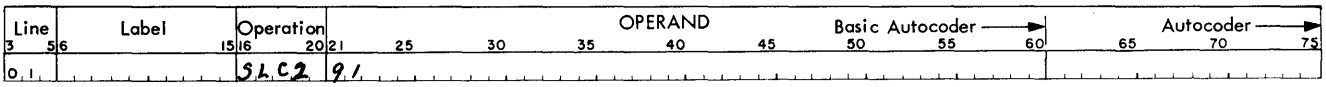


FIGURE 43.

For shift right coupled, or shift right and round coupled, the sign of accumulator 2 is replaced by the sign of accumulator 1. For shift left or a shift left and count, the sign of accumulator 1 is replaced by the sign of accumulator 2. This is true even for shifts of zero places.

INSTRUCTION FORMAT: S01 23 45 6 7 89

S01 −50

23 Indexing word. Even though positions 6-9 do not represent an address, they are modified by positions 2-5 of an IW, if one is specified here.

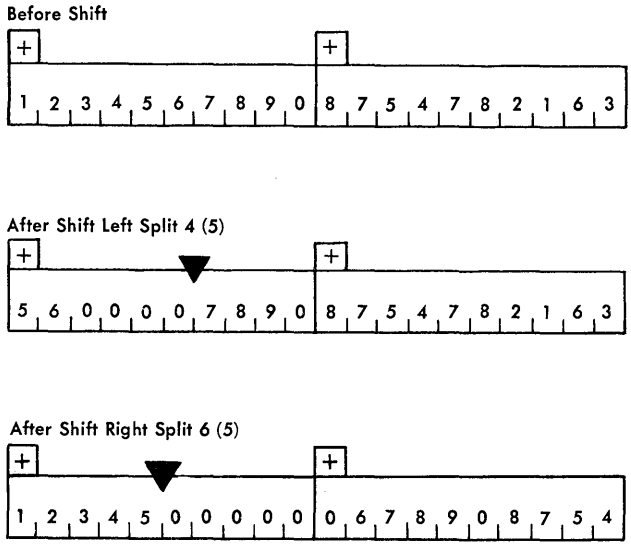


FIGURE 44. EXAMPLES OF SPLIT-SHIFTING

45 Index word, for shift and count. The number of places shifted (00-20) replaces positions 4-5 of the index word specified here (positions 2-3 of the index word are set to zero, and its sign is set to plus).

These positions are not used in the other shift operations.

6 0-9, designating the position in accumulator 1 or 2 from which the split-shift operation will take place. For shift-left split, it is the units position of the digits shifted; for shift-right split, it is the high-order position. This position is not used with normal-shift operations.

7 0-7, determining the type of shift:

0—Shift right coupled	SR
1—Shift right and round coupled	SRR
2—Shift left coupled	SL
3—Shift left and count coupled	SLC
4—Shift right from point accumulator 1	SRS
5—Shift left from point accumulator 1	SLS
6—Shift right from point accumulator 2	SRS
7—Shift left from point accumulator 2	SLS

89 00-20, designating the number of positions to be shifted. Not used in shift and count but must be valid.

EXAMPLES: To shift right five places in accumulators 1 and 2 coupled:

S01	23	45	6	7	89
−50	00	xx	x	0	05

Contents of the accumulators before the operation:

Acc 1	Acc 2
+12345 00000	−67890 00000

Contents of the accumulators after the operation:

Acc 1	Acc 2
+00000 12345	+00000 67890.

To shift the contents of accumulators 1 and 2 coupled, three positions to the right, and round off the value:

S01	23	45	6	7	89
−50	00	xx	x	1	03

Contents of the accumulators before the operation:

Acc 1	Acc 2
+34780 00022	+00001 78534.

Contents of the accumulators after the operation:

Acc 1	Acc 2
+00034 78000	+02200 00179.

To shift left 16 places in accumulators 1 and 2 coupled:

S01	23	45	6	7	89
−50	00	xx	x	2	16

Contents of the accumulators before the operation:

Acc 1	Acc 2
+75346 92188	−74310 25505.

Contents of the accumulators after the operation:

Acc 1	Acc 2
−55050 00000	−00000 00000.

To shift the contents of accumulators 1 and 2 coupled, to the left; until a significant digit is in the high-order position; and to store the number of positions shifted, in positions 2-5 of IW 25:

S01	23	45	6	7	89
−50	00	25	x	3	yy

Positions 8-9 are not used but cannot be greater than 20.

Contents of the accumulators before the operation:

Acc 1	Acc 2
−00007 00000	−12345 00222.

Contents of the accumulators after the operation:

Acc 1	Acc 2
−70000 01234	−50022 20000.

Contents of IW 25 before the operation: + 66 0155 6420. Contents of IW 25 after the operation: + 66 0004 6420.

To split-shift the contents of accumulators 1 and 2 coupled, 10 positions to the right, starting with position 5 of accumulator 1:

S01	23	45	6	7	89
−50	00	xx	5	4	10

Contents of the accumulators before the operation:

Acc 1	Acc 2
+11223 34455	−66778 89900

Contents of the accumulators after the operation:

Acc 1	Acc 2
+11223 00000	+00000 34455.

To split-shift the contents of accumulators 1 and 2 coupled, 3 positions to the left, starting with position 2 of accumulator 1:

S01	23	45	6	7	89
−50	00	xx	2	5	03

Contents of the accumulators before the operation:

Acc 1	Acc 2
+53560 00000	−00000 04475.

Contents of the accumulators after the operation:

Acc 1	Acc 2
−00060 00000	−00000 04475.

Note that the sign of accumulator 1 is made the same as that of accumulator 2.

To split-shift the contents of accumulators 1 and 2 coupled, 6 positions to the right, starting with position 0 of accumulator 2:

S01	23	45	6	7	89
−50	00	xx	0	6	06

Contents of the accumulators before the operation:

Acc 1	Acc 2
+12345 67890	+12345 67890.

Contents of the accumulators after the operation:

Acc 1	Acc 2
+12345 67890	+00000 01234.

To split-shift the contents of accumulators 1 and 2 coupled, 19 positions to the left, starting

with the units position of accumulator 2:

S01	23	45	6	7	89
−50	23	xx	9	7	19

Contents of the accumulators before the operation:

Acc 1		Acc 2	
−99887	76655	+00044	33221.

Contents of the accumulators after the operation:

Acc 1		Acc 2	
+10000	00000	+00000	00000.

DATA FLOW: The operations for this code vary, depending on the number of positions shifted and whether the operation is normal-shift or split-shift.

SHIFT LEFT OR SHIFT RIGHT, LESS THAN 10 POSITIONS: The contents of accumulator 2 are moved to the synchronizer register. The shifting takes place in accumulator 1 and the synchronizer register. Zeros are inserted, as required, into the high-order portion of accumulator 1 for shift right; or into the low-order portion of accumulator 2 for shift left. At the completion of the shifting, accumulator 1 contains its correct value. The contents of the synchronizer register are transmitted to accumulator 2.

SHIFT RIGHT OF GREATER THAN 10 POSITIONS: The arithmetic register is filled with zeros and given the sign of accumulator 1. The synchronizer register is shifted right the number of positions called for, minus 10. The contents of the synchronizer register are transmitted to accumulator 2. The contents of the arithmetic register are sent to accumulator 1.

SHIFT LEFT OF GREATER THAN 10 POSITIONS: This is the same as for shift right greater than 10 positions, except that the contents of accumulator 2 are sent to the synchronizer register, the shift is left, and the contents of the synchronizer register are transmitted to accumulator 1, with the zeros from the arithmetic register going to accumulator 2.

SPLIT SHIFT RIGHT IN ACC 2, OR SPLIT SHIFT LEFT IN ACC 1: These instructions do not involve both accumulators. The single accumulator to be shifted is brought to the synchronizer register, the sign of which is set to that of the unused accumulator. The split shift takes place there, and the resultant contents and sign are transmitted back to the accumulator.

OPERATION		NO. OF POS. SHIFTED	TIME IN MICROSECONDS
SR SL	Shift Right Coupled Shift Left Coupled	0-4, 20 5-7, 11-13 8-10, 14-16 17-19	36 48 60 72
SRR	Shift Right and Round Coupled	0 1-3, 11-13 4-6, 14-16 7-9, 17-19 10, 20	36 72 84 96 108
SLC	Shift Left and Count Coupled	0-4, 20 5-7, 11-13 8-10, 14-16* 17-19	72 84 96 108
SRS SLS	Shift Right Split Shift Left Split	0 1-3 4-6 7-9 10	36 108 120 132 144

FIGURE 45. TIMING — COUPLED SHIFT CONTROL

SPLIT SHIFT RIGHT IN ACC 1, SPLIT SHIFT LEFT IN ACC 2: The shifting must take place in each accumulator, and some of the digits from one accumulator must be in the other accumulator at completion.

REGISTERS AFFECTED: This depends on the specific operation, as described above. In some operations, both accumulators are affected; in others, only one. All of them, however, use the synchronizer register.

TIMING: Figure 45

COMMENTS: Because this instruction always uses accumulators 1 and 2 coupled, there is no accumulator number required in the *Autocoder* mnemonics.

A normal-shift left of 10 positions sets accumulator 2 to zeros and puts its contents and sign into accumulator 1. A normal-shift of 10 positions to the right does the reverse. A normal-shift of 10 or more positions always sets one accumulator to zeros and gives the other accumulator the sign of the reset accumulator. A normal-shift of 20 positions sets both accumulators to zeros.

In either normal-shift or split-shift instructions, the signs of both accumulators are always the same after the operation is completed. They both have the sign of accumulator 1 after a shift-right, or the sign of accumulator 2 after a shift-left operation. This is true even for shifts of zero places

(00 in positions 8-9, or no shift required in shift left and count). The sign of accumulator 2 can be made the same as that of accumulator 1, without changing the contents of either accumulator; for example, by the instruction:

S01	23	45	6	7	89
−50	00	00	9	4	00

The digit in position 7 could be 4 or 6. Position 6 can contain any digit (it can whenever positions 8-9 are 00). Code 5, shift left from point in accumulator 1, can never change the contents of accumulator 2. Similarly, code 7 can never change the contents of accumulator 1. Shift left or right from a point can specify a shift of more positions than there are available (e. g. shift right 20 positions from the units position of accumulator 2), as long as the number of positions does not exceed 20.

AUTOCODER EXAMPLES (Figures 46-52): The instruction assembled from Figure 46 is:

S01	23	45	6	7	89
−50	00	00	0	0	12

The instruction assembled from Figure 47 is:

S01	23	45	6	7	89
−50	00	00	0	1	05

The instruction assembled from Figure 48 is:

S01	23	45	6	7	89
−50	00	00	0	2	17

The instruction assembled from Figure 49 is:

S01	23	45	6	7	89
−50	00	47	0	3	00

Note that the number in the operand column designates the index word in which the shift-and-count position total is to be stored.

The instruction assembled from Figure 50 is:

S01	23	45	6	7	89
−50	00	00	5	4	02

The instruction assembled from Figure 51 is:

S01	23	45	6	7	89
−50	00	00	5	6	02

For *Autocoder* programming, the digits in the coupled accumulators are numbered from 00 through 19, from the high-order position of accumulator 1 to the low-order position of accumulator 2. The 15 here thus means position 5 of accumulator 2. The only difference between this assembled instruction and the previous one is in position 7.

The instruction assembled from Figure 52 is:

S01	23	45	6	7	89
−50	00	00	4	6	15

This instruction sets all of accumulator 1 and the 5 high-order positions of accumulator 2 to zero.

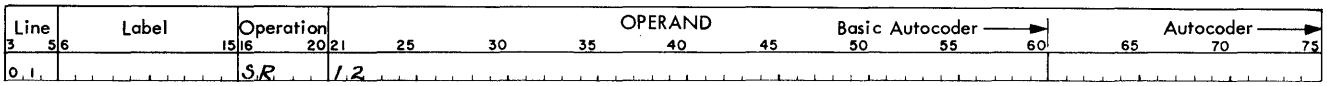


FIGURE 46.

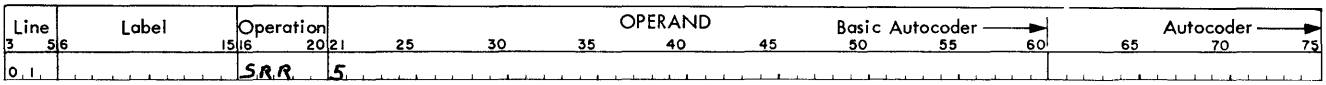


FIGURE 47.

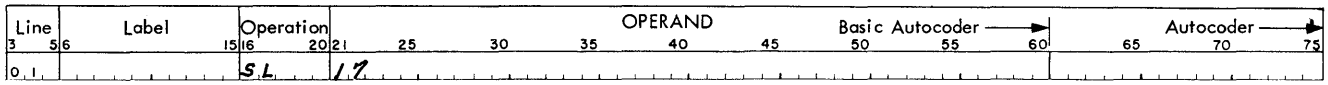


FIGURE 48.

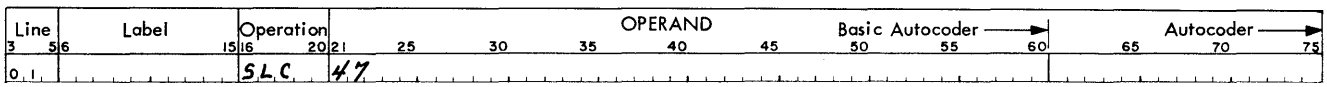


FIGURE 49.

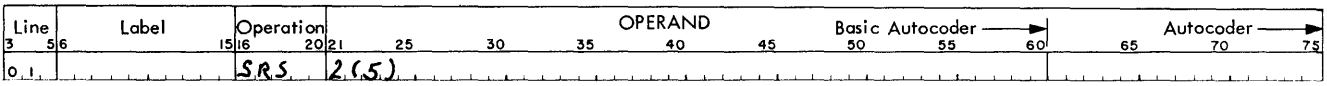


FIGURE 50.

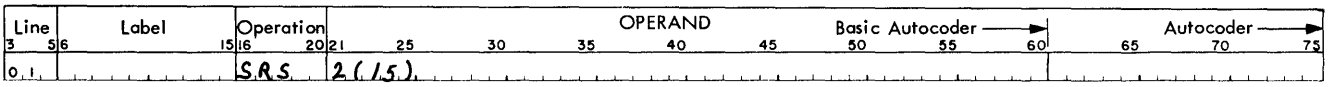


FIGURE 51.

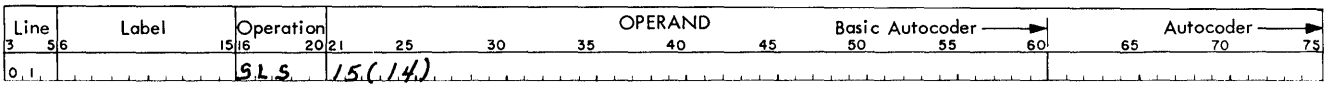


FIGURE 52.

Logic Codes

The 7070 stored program can make logical decisions based on a considerable number of different situations. The logic operations are classified into two types: *branch* and *compare*. Miscellaneous operations are also included in this section, because they are usually used in connection with logical decisions made by the program. The branch operations are logical decisions; the program either branches or does not, depending on the result of a test. A compare operation is also a test, but the result of the test merely turns on an indicator for later interrogation by a branch instruction. The miscellaneous codes perform no operation but are useful in connection with the logic operation codes, automatic checking features of the system, and program testing.

The branch operation codes in this section are listed in their categories in Figure 53; the compare codes and miscellaneous are listed in Figure 54.

CATEGORIES	OP CODES	NAMES	MNEMONICS
Branch			
Accumulators	+10, +20, +30	Branch if zero in accumulator #	BZ1, BZ2, BZ3
	−10, −20, −30	Branch if minus in accumulator #	BM1, BM2, BM3*
Alteration Switch	+51	Branch if alteration switch is on	BAS
Channel busy	+51	Branch if channel is busy	BCB
Electronic switches	+61, +62, +63	Electronic switch control:	
		Branch if electronic switch is on	BES
		Electronic switch set on	ESN
		Electronic switch set off	ESF
		Branch if electronic switch is on, and set on if off	BSN
		Branch if electronic switch is on, and set off if on	BSF
Indicator branch	+40	Branch if low	BL*
	−40	Branch if high	BH*
	−41	Branch if equal	BE*
	+11, +21, +31	Branch if overflow in accumulator #	BV1, BV2, BV3*
	+41	Field overflow control:	
		Branch if field overflow	BFV*
		Sense mode for field overflow	SMFV*
		Halt mode for field overflow	HMFV*

*These require only 24 microseconds for indexing, rather than 36.

This category also includes the branch on sign change instruction, BSC, included in the Compare category under −03, Sign Control. The *floating-decimal overflow* and *underflow* indicator-branch instructions are in the Floating Decimal section.

Other branch instructions are included in the Index Word, Disk Storage, and Priority Codes sections.

FIGURE 53. BRANCH OPERATION CODES

All of the *Autocoder* symbols that start with B are branch codes (but not all the branch codes start with B; HB, for example).

All the compare symbols start with C, and no other codes start with C.

Branch if Zero in Accumulator # +10, +20, +30 — BZ1, BZ2, BZ3

MACHINE DESCRIPTION: The contents of the specified accumulator are tested for zero. If it is zero, the program branches to the location in positions 6-9 (indexable). If it is not zero, the program continues sequentially, taking its next instruction from the location in the instruction counter. The sign of the accumulator is ignored.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S — Always +.

0 — Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.

1 — Always 0.

23 — Indexing word.

45 — Not used.

6789 — Address of the next instruction, if the accumulator is zero. Not used if the accumulator is non-zero.

EXAMPLE: To branch to location 1580 if accumulator 2 is zero:

S01	23	45	6789
+20	00	xx	1580

Contents of accumulator 2: −00000 00100. The program does not branch.

DATA FLOW: If the specified accumulator is zero, the contents of positions 6-9 go to the instruction counter (after indexing if indexing is specified).

REGISTERS AFFECTED: The instruction counter if the accumulator is zero.

TIMING: 36 microseconds

COMMENTS: Note that the sign of the accumulator is ignored. The accumulator is considered zero only if all 10 digits are zeros; there is no field definition. The accumulator is not changed by this operation.

CATEGORIES	OP CODES	NAMES	MNEMONICS
Compare			
Accumulators	+15, +25, +35	Compare accumulator # to storage	C1, C2, C3
	−15	Compare absolute in accumulator 1 to absolute in storage	CA
Digit	+03	Compare storage to digit	CD
Sign	−03	Sign Control:	
		Compare sign to alpha	CSA
		Compare sign to minus	CSM
		Compare sign to plus	CSP
		Make sign alpha	MSA
		Make sign minus	MSM
		Make sign plus	MSP
		Sense mode for sign change	SMSC*
		Halt mode for sign change	HMSC*
		Branch on sign change	BSC*
Miscellaneous			
Branch	+01	Branch (unconditional)	B
	+02	Branch and load location in index word	BLX
	+00	Halt and branch	HB
Other	−00	Halt and proceed	HP
	−01	No operation	NOP*

*These require only 24 microseconds for indexing, rather than 36.

FIGURE 54. COMPARE OPERATION CODE

AUTOCODER EXAMPLE (Figure 55): Assume that THETA has been previously defined as word 4100 (or any part of word 4100). The assembled instruction is:

S01	23	45	6789
+30	00	xx	4150

Branch if Minus in Accumulator # −10, −20, −30 — BM1, BM2, BM3

MACHINE DESCRIPTION: The sign of the specified accumulator is tested for minus. If it is minus, the program branches to the location in positions 6-9 (indexable). If it is not minus, the program continues sequentially, taking its next instruction from the location in the instruction counter. The contents of the accumulator are ignored.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S — Always −.

0 — Designates the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.

1 — Always 0.

23 — Indexing word.

45 — Not used.

6789 — Address of the next instruction, if the accumulator sign is minus. Not used if the sign is plus or alpha.

EXAMPLE: To branch to location 1670 if accumulator 3 is minus:

S01	23	45	6789
−30	00	xx	1670

Contents of accumulator 2: @ 618264 0000. The program does not branch.

DATA FLOW: If the specified accumulator is minus, the contents of positions 6-9 go to the instruction counter (after indexing if indexing is used).

REGISTERS AFFECTED: The instruction counter if the accumulator is minus.

TIMING: 36 microseconds.

COMMENTS: Note that the test is for minus or non-minus; if it is either plus or alpha, there is no branch. The contents of the accumulator are ignored, and neither sign nor contents are changed by the operation.

AUTOCODER EXAMPLE (Figure 56): The assembled instruction is:

S01	23	45	6789
−10	21	00	0645

Branch on Alteration Switch or Channel Busy +51

MACHINE DESCRIPTION: This is an augmented code, which performs two functions:

1. Tests any of the four alteration switches on the console and branches to the location in positions 6-9 (indexable) if that switch is ON.
2. Tests either of the two tape/disk channels, and branches to the location in positions 6-9 (indexable) if that channel is busy. Several causes can make a channel busy. The most common is the transmission of data between core storage and tape or disk storage, as the result of read or write instruction.

INSTRUCTION FORMAT: S01 23 4 5 6789

S01 — +51

23 — Indexing word.

4 — For alteration-switch test, 1-4, specifying the alteration switch.

For channel-busy test, 1-2, specifying the tape/disk channel.

5 — 0 or 1, determining the operation:

0-Branch if alteration switch is ON — BAS

1-Branch if channel is busy — BCB

6789 — Branch address if:

the specified alteration switch is ON, or if the specified channel is busy.

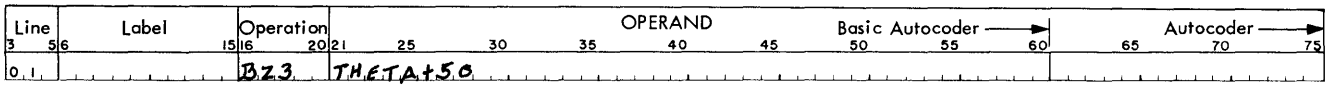


FIGURE 55.

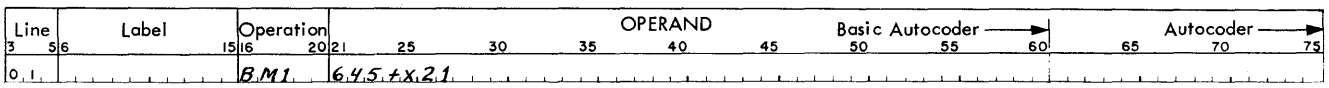


FIGURE 56.

EXAMPLES: To branch to location 2244, if alteration switch 3 is ON:

S01	23	4	5	6789
+51	00	3	0	2244

To branch to location 0888, if channel 2 is busy:

S01	23	4	5	6789
+51	00	2	1	0888

DATA FLOW: There is no flow of data initiated by these codes. They are tests of existing conditions at a particular point in the program.

REGISTERS AFFECTED: The instruction counter, if a branch is executed.

TIMING: 36 microseconds.

COMMENTS: The alteration switches afford operator control over the stored program, to the extent that they are used in the program. They are occasionally helpful in program testing. The stored program cannot turn an alteration switch on or off; this is done by console operation only.

The channel-busy test can be useful in saving time. A tape-read instruction may hold up the stored program; for example, if there is another tape unit on that channel in operation at the time the instruction is given. In many cases, the efficiency of the stored program can be increased if a channel-busy test is given before every tape read or write instruction, particularly if the tape records are long. If the channel is busy, the program can branch and do something else, instead of being interlocked until the channel is free.

This instruction can be used at the completion of a disk-storage seek operation to determine which channel is available for the subsequent disk read or write operation.

AUTOCODER EXAMPLES (Figures 57 and 58): CHI has been defined as word 2360. The instruction assembled from Figure 57 is:

S01	23	4	5	6789
+51	00	2	0	2360

OMEGA is word 3343, previously defined. The instruction assembled from Figure 58 is:

S01	23	4	5	6789
+51	00	1	1	3343

Electronic Switch Control +61, +62, +63

MACHINE DESCRIPTION: These augmented codes cover all of the operations involving the 30 electronic switches: testing whether one is ON or OFF, turning one on or off, or combining the test with turning one on or off. Code +61 provides for testing and/or setting the 10 switches, the settings of which are in word 0101; +62 for word 0102; and +63 for word 0103.

INSTRUCTION FORMAT: S 0 1 23 4 5 6789

S — Always +.

0 — Always 6.

1 — 1-3, designating the group of 10 switches. This position and position 5 determine the specific electronic switch of the 30 available.

23 — Indexing word for the branch address if the instruction is a test of an electronic switch. For non-test operations, these positions have no function and can contain any digits.

4 — 0-4, determining the specific operation:

0—Test	BES
1—Turn on	ESN
2—Turn off	ESF
3—Test and turn on	BSN
4—Test and turn off	BSF

5 — 0-9, Designating the switch number, of the 10 specified in position 1. These digits actually refer to positions 0-9 of word 0101, 0102, or 0103. The switches are numbered 1-30. Switch 1 has its designation in position 0 of word 0101; switch 2 in position 1, etc. The designation for switch 30 is

Line 3 5	Label 6 15	Operation 16 20	OPERAND 21 25 30 35 40 45 Basic Autocoder → 50 55 60	Autocoder → 65 70 75
0 1		BAS	2, CHI	

FIGURE 57.

Line 3 5	Label 6 15	Operation 16 20	OPERAND 21 25 30 35 40 45 Basic Autocoder → 50 55 60	Autocoder → 65 70 75
0 1		BCB	1, OMEGA	

FIGURE 58.

in position 9 of word 0103. Thus, electronic switch 7, for example, is ON if there is a digit 1-9 in position 6 of word 0101, and OFF if there is a zero in that position.

6789 Branch address (indexable) for the branch instructions (0, 3, or 4 in position 4). For the other two operations, these positions are not used and can contain any digits. If the specified electronic switch is OFF when a branch instruction is given, these positions are not used.

0—TEST: The electronic switch specified in positions 1 and 5 is tested. If it is ON, the next instruction is taken from the address in positions 6-9, after indexing if indexing is used. If it is OFF, the next instruction is taken from the location in the instruction counter.

The test is actually made of the digit in word 0101, 0102, or 0103, that corresponds to the specified electronic switch. If there is a digit 1-9 in that position, the switch is considered ON, and the branch is executed. If that position contains a zero, the switch is considered OFF, and no branch is executed.

1—TURN ON: The digit position in word 0101, 0102, or 0103, specified by positions 1 and 5 of the instruction, is set to 1. The program does not branch; the next instruction comes from the location in the instruction counter.

2—TURN OFF: This is the same as turn on, except that a 0 is inserted into the specified position of word 0101, 0102, or 0103, thus turning the switch off.

3—TEST AND TURN ON: If the specified switch is ON, the program branches to the location in positions 6-9 (indexable). Regardless of whether the branch took place, the specified switch is turned on after the test. A 1 is inserted into the specified position of word 0101, 0102, or 0103.

4—TEST AND TURN OFF: Same as test and turn on, except that a 0 is inserted into the specified position of word 0101, 0102, or 0103, after the test is made.

EXAMPLES: To branch in word 1352 if electronic switch 13 is ON:

S	0	1	23	4	5	6789
+	6	2	00	0	3	1352

Contents of word 0102: +01010 11001, unchanged by the operation. The program branches, because of the 1 in position 3.

To turn on electronic switch 24:

S01	23	4	5	6789
+63	xx	1	4	xxxx

Contents of word 0103 before the operation: −23408 50332; contents of word 0103 after the operation: −23401 50332.

To turn off electronic switch 29:

S01	23	4	5	6789
+63	xx	2	9	xxxx

Contents of word 0103: +11101 00110, the same before and after the operation, because position 9 was already at zero.

DATA FLOW: The storage word designated in position 1 (word 0101, 0102, or 0103), is brought to the arithmetic register. If the operation is test (0, 3, or 4 in position 4), the digit specified in position 5 of the instruction is tested for a zero or non-zero condition. If the operation is turn-on (1 or 3 in position 4), a 1 is inserted into the position of the arithmetic register, specified in position 5. If the operation is turn-off (2 or 4 in position 4), a 0 is inserted. The contents of the arithmetic register are returned to core storage. If the operation is test, and a non-zero is in the specified digit position of the arithemetic register, the contents of positions 6-9 of the instruction (indexed if so specified), are transmitted to the instruction counter.

REGISTERS AFFECTED: The arithmetic register, and the instruction counter if a branch instruction tests a switch that is ON.

TIMING: 36 microseconds for all electronic switch instructions.

COMMENTS: The test operations all test the specified electronic switch for an ON condition; the program branches if the switch is ON, and does not branch if it is OFF. The operations to test and turn on, and test and turn off, both make the test for ON, and then turn the switch on or off, as specified.

Words 0101, 0102, and 0103 can be changed by other instructions. They can be treated just like the other core-storage words.

AUTOCODER EXAMPLES (Figures 59-63): MONDAY has been previously defined as word 0427. The instruction assembled from Figure 59 is:

S01	23	4	5	6789
+63	00	0	6	0427

The instruction assembled from Figure 60 is:

S01	23	4	5	6789
+63	00	1	1	0000

The instruction assembled from Figure 61 is:

S01	23	4	5	6789
+61	00	2	8	0000

TUESDAY has been previously defined as word 2704. The instruction assembled from Figure 62 is:

S01	23	4	5	6789
+62	17	3	9	2704

The actual location 3110 is used. The instruc- instruction assembled from Figure 63 is:

S01	23	4	5	6789
+61	00	4	4	3110

Branch if Low
+40 **BL**

MACHINE DESCRIPTION: This instruction tests the *low* indicator. If it is ON, the program branches to the address in positions 6-9 (indexable). If it is not ON, the next instruction is taken from the location in the instruction counter. The indicator setting is not changed.

INSTRUCTION FORMAT:

S01	23	45	6789

S01 +40.

23 Indexing word for the branch address. Not used if there is no branch.

45 Not used.

6789 Branch address, used if the *low* indicator is ON (indexable). Otherwise, not used.

EXAMPLE: To branch to location 2300 if the *low* indicator is ON:

S01	23	45	6789
+40	00	xx	2300

DATA FLOW: There is no flow of data involved with this instruction. If the *low* indicator is ON, the contents of positions 6-9, indexed if indexing is specified, are transmitted to the instruction counter.

REGISTERS AFFECTED: The instruction counter if the *low* indicator is ON.

TIMING: 36 microseconds.

COMMENTS: This instruction does not affect the setting of the *low* indicator, nor the other two indicators;

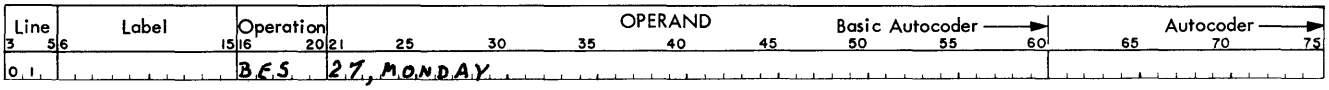


FIGURE 59.

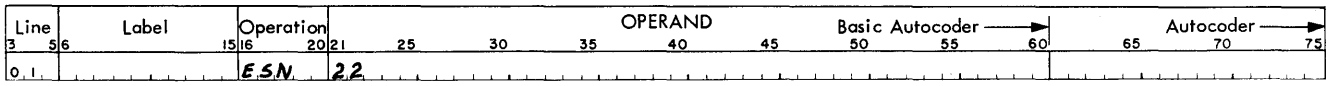


FIGURE 60.

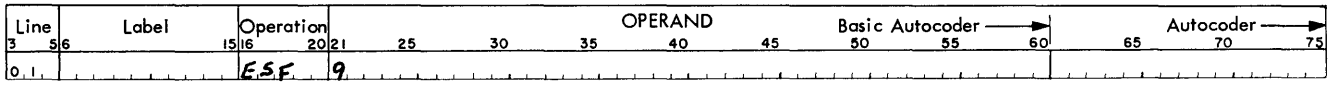


FIGURE 61.

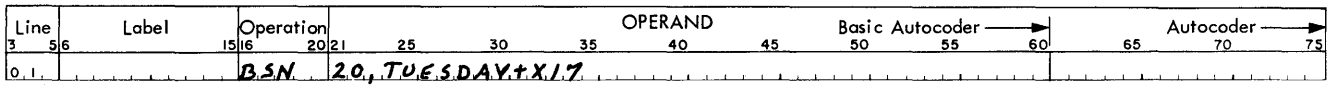


FIGURE 62.

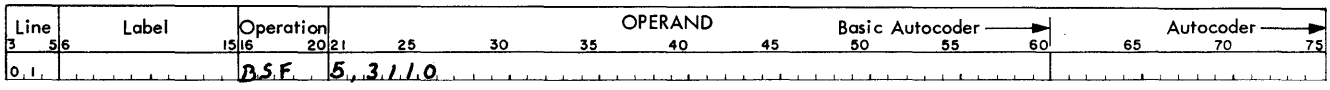


FIGURE 63.

the stored program normally changes these settings only with a compare instruction. (They can be changed by adjusting the priority indicator storage word, 0100, in a priority routine. The adjusted settings are returned to the indicators by a priority-release instruction. See section on *Automatic Priority Processing* for a complete description of the priority indicator storage word.)

The *high, equal,* and *low* indicators are all turned off by pressing COMPUTER RESET on the console operating keyboard. Pressing ERROR RESET on the operating keyboard turns them off, only if none, two, or three of them had been on, and the machine has stopped because a compare instruction has discovered this.

AUTOCODER EXAMPLE (Figure 64): WEDNESDAY has been previously defined as word 1100. The assembled instruction is:

S01	23	45	6789
+40	00	00	1100

Branch if High
−40 BH

MACHINE DESCRIPTION: This code is the same as branch if low, except that the *high* indicator is tested, and a branch is executed if it is ON.

INSTRUCTION FORMAT: Same as branch if low, except for sign.

EXAMPLE: To branch to location 3450 if the *high* indicator is ON:

S01	23	45	6789
−40	00	xx	3450

DATA FLOW: Same as branch if low.

REGISTERS AFFECTED: Same as branch if low.

TIMING: 36 microseconds.

COMMENTS: The comments under *Branch if Low,* also apply to branch if high.

AUTOCODER EXAMPLE (Figure 65): WEDNESDAY has been previously defined as word 1100. The assembled instruction is:

S01	23	45	6789
−40	00	00	1200

Branch if Equal
−41 BE

MACHINE DESCRIPTION: This code is the same as branch if low, except that the *equal* indicator is tested, and a branch is executed if it is ON.

INSTRUCTION FORMAT: Same as branch if low, except for positions S01, which are −41.

EXAMPLE: To branch to location 4101 indexed by positions 2-5 of IW 56, if the *equal* indicator is ON:

S01	23	45	6789
−41	56	xx	4101

DATA FLOW: Same as branch if low.

REGISTERS AFFECTED: Same as branch if low.

TIMING: 36 microseconds.

COMMENTS: The comments under branch if low also apply to branch if equal.

Note that these branch codes do not have to follow the compare instructions immediately; they can come at any time.

AUTOCODER EXAMPLE (Figure 66): WEDNESDAY has been previously defined as word 1100. The assembled instruction is:

S01	23	45	6789
−41	00	00	1300

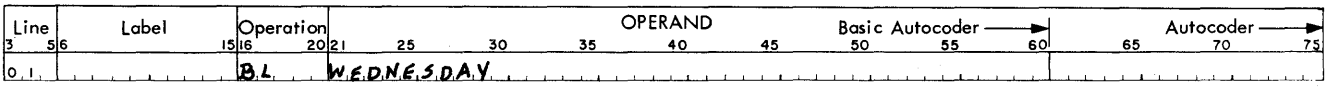


FIGURE 64.

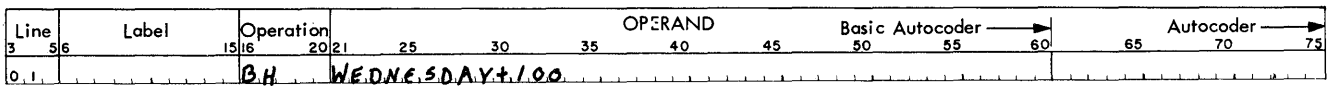


FIGURE 65.

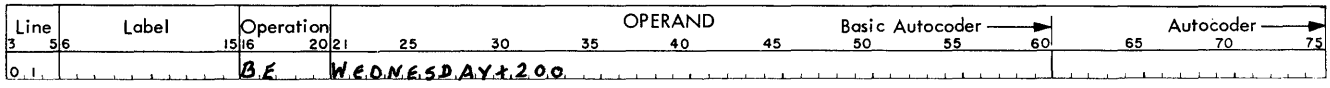


FIGURE 66.

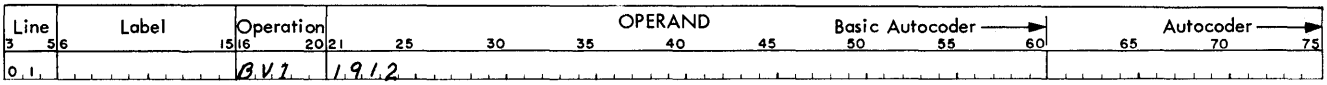


FIGURE 67.

Branch if Overflow in Accumulator # +11, +21, +31 **BV1, BV2, BV3**

MACHINE DESCRIPTION: These codes are similar to the branch if low, equal, and high codes, in that an indicator is tested, and the program branches to the address in positions 6-9 (indexable), if it is ON. The indicator may have been set previously in the program; it does not have to be tested immediately. The *overflow* indicator for an accumulator is turned on when a true-add operation obtains an 11-digit result. The high-order digit of this result is lost (it is a carry 1 in all cases), and the *overflow* indicator for the accumulator is turned ON.

Unlike the BRANCH IF LOW, EQUAL, and HIGH codes, a branch-overflow instruction turns off the indicator after testing it.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S — Always +.

0 — 1-3, designating the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3. (Each accumulator has its own *overflow* indicator.)

1 — Always 1.

23 — Indexing word for the branch address. Not used if there is no branch.

45 — Not used.

6789 — Branch address if the specified overflow indicator is on (indexable).

EXAMPLES: To branch to location 4501 if the *overflow* indicator for accumulator 3 is ON:

S01	23	45	6789
+31	00	xx	4501

DATA FLOW: There is no movement of data involved. If the indicator is ON, it is turned off, and the address in positions 6-9, indexed if so specified, is transmitted to the instruction counter.

REGISTERS AFFECTED: The instruction counter and the specified overflow indicator, if the indicator is ON.

TIMING: 36 microseconds.

COMMENTS: The only situation that can cause an accumulator to overflow and set the *overflow* indicator, is a true-add to accumulator operation in which at least one of the factors is a 10-digit number.

Overflow occurs if the sum of two numbers is 11 digits in length. When this happens, the 10 low-order digits of the correct total are in the accumulator, and the *overflow* indicator for that accumulator is turned on. The high-order digit of the 11-digit result is a 1, in all cases. If necessary, a program subroutine could be used to keep track of the overflows.

Each accumulator has an overflow light on the console. The light is ON whenever the *overflow* indicator for that accumulator is ON. The branch-if-overflow instruction turns off the console light, as well as the indicator. The console also contains an overflow key for each accumulator. If the key is not pressed, an overflow in that accumulator sets the indicator and stops the machine. If the key is pressed, an overflow in that accumulator merely sets the indicator; the program continues, with the indicator ON, until a BV instruction tests it and turns it off.

AUTOCODER EXAMPLE (Figure 67): The actual location 1912 is used. The assembled instruction is:

S01	23	45	6789
+11	00	00	1912

Field Overflow Control +41

MACHINE DESCRIPTION: This augmented code determines the setting of the field-overflow stop/sense switch. It also tests the *field-overflow* indicator, causing the program to branch to the address in positions 6-9 (indexable) if the indicator is on. This branch code then turns the indicator off if it is ON, just as the branch-if-overflow codes (BV#) do.

INSTRUCTION FORMAT: S01 23 4 5 6789

S01 — +41.

23 — Indexing word for the branch address. Not used if the branch instruction is not used, or if it is used but there is no branching.

4 — Not used.

5 — 0-2, designating the operation:

0—Branch if field overflow BFV

1—Sense mode for field overflow SMFV

2—Halt mode for field overflow HMFV

6789 — Branch address for the BFV instruction. For the other instructions, these positions are not used and can contain any digits.

0—BRANCH IF FIELD OVERFLOW: The *field-overflow* indicator is tested. If it is ON, the address in positions 6-9 (indexable) is transmitted to the instruction counter, and the next instruction is taken from that location. The *field-overflow* indicator is turned off if it was ON.

1—SENSE MODE FOR FIELD OVERFLOW: The field-overflow sense/stop switch is set to sense. If it is already at sense, this instruction has no effect. When the switch is at sense, a field-overflow condition arising in the program turns on the *field-overflow* indicator but does not stop the machine.

2—HALT MODE FOR FIELD OVERFLOW: The field-overflow sense/stop switch is set to stop. If it is already at stop, this instruction has no effect. When the switch is at stop, a field-overflow condition arising in the program turns on the *field-overflow* indicator, and stops the machine. If the *field-overflow* indicator is ON when this instruction is given, the machine stops.

EXAMPLES: To branch to location 2365 if the *field overflow* indicator is ON:

S01	23	4	5	6789
+41	00	x	0	2365

To set the field-overflow stop/sense switch to stop, regardless of its previous setting:

S01	23	4	5	6789
+41	xx	x	2	xxxx

DATA FLOW: There is no movement of data in these codes. If the *field-overflow* indicator is ON, and the BFV instruction is given, the contents of positions 6-9, indexed if so specified in positions 2-3, are transmitted to the instruction counter.

REGISTERS AFFECTED: If a branch is called for, the instruction counter is changed, and the *field-overflow* indicator is turned off.

TIMING: 36 microseconds, for all these operations.

COMMENTS: There are two devices involved with this operation code, and they have no direct relationship with each other. The *field-overflow* indicator is immediately turned ON whenever a field-overflow situation arises in a store or add-to-storage operation. The branch code, BFV, turns it OFF after making the test. There is no other way to turn the *field-overflow* indicator OFF, within the stored program.

The field-overflow stop/sense switch is constantly at either sense or stop, determined by whether a HMFV or SMFV instruction has been given more recently. If it is on stop, a field-overflow situation arising in the program stops the machine, after turning on the *field-overflow* indicator. To turn off the indicator and start the machine, press ERROR RESET and START on the console. If the switch is on sense, the program does not stop. The switch could thus be considered as having stop and "no-stop" settings.

Pressing COMPUTER RESET, on the console, sets the field-overflow stop/sense switch to stop.

AUTOCODER EXAMPLES (Figures 68 and 69): THURSDAY has been previously defined as word 1775. The instruction assembled from Figure 68 is:

S01	23	4	5	6789
+41	00	0	0	1755

The instruction assembled from Figure 69 is:

S01	23	4	5	6789
+41	00	0	1	0000

Note that no information is needed in the operand section.

Compare Accumulator # to Storage
+15, +25, +35 C1, C2, C3

MACHINE DESCRIPTION: The entire contents of the specified accumulator, including sign, are compared with the field-defined digits and sign of the storage

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		BFV	THURSDAY		

FIGURE 68.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		SMFV			

FIGURE 69.

word addressed by positions 6-9 (indexable). One of the three compare indicators, *high, equal,* or *low,* is turned on, and the other two turned off, depending on the relation of the accumulator value to the storage-word value. If the accumulator is higher, the *high* indicator is turned on; if the accumulator value is lower, the *low* indicator is turned on. If they are equal, the *equal* indicator is turned on.

The following is a chart of the relative values of signs and digits in compare operations.

Highest	+99999	99999
	to	
	+00000	00000
	−00000	00000
	to	
	−99999	99999
	@99999	99999
	to	
Lowest	@00000	00000

Note that plus is higher than minus, which is in turn higher than alpha, regardless of the numerical values. Note also that the greater a minus number is in absolute value, the smaller it is in a compare operation. Thus, a minus 9 is smaller than a minus 8, which is in turn smaller than a minus 7, etc.

The contents of the storage word and the accumulator are unchanged by this operation.

INSTRUCTION FORMAT: S 0 1 23 45 6789

S	Always plus.
0	1-3, designating the accumulator: 1 for accumulator 1, 2 for accumulator 2, 3 for accumulator 3.
1	Always 5.
23	Indexing word.
45	Field definition.
6789	Address (indexable) of the word in core storage, the sign and field-defined digits of which are to be compared with the specified accumulator.

EXAMPLES: To compare the entire contents of accumulator 1 with positions 0-4 of word 1806:

S01	23	45	6789
+15	00	04	1806

Contents of accumulator 1: +00001 23456. Contents of word 1806: +23456 78900. The *high* indicator is turned ON—123,456 is higher than 23,456.

To compare the entire contents of accumulator 3 with all of word 1919:

S01	23	45	6789
+35	00	09	1919

Contents of accumulator 3: −00000 11122. Contents of word 1919: +00000 11122. The *low* indicator is turned on, because the accumulator value is minus and the storage value is plus.

To compare the entire contents of accumulator 2 with position 9 of word 1750:

S01	23	45	6789
+25	00	99	1750

Contents of accumulator 2: −00000 00005. Contents of word 1750: −18555 02774. The *low* indicator is turned on, because the accumulator has the minus number that is greater in absolute value.

To compare the entire contents of accumulator 1 with all of word 1515:

S01	23	45	6789
+15	00	09	1515

Contents of accumulator 1: −61777 90091. Contents of word 1750: −18555 02774. The *low* indicator is turned on, because the accumulator has the minus number that is greater in absolute value.

To compare the entire contents of accumulator 1 with all of word 1515:

S01	23	45	6789
+15	00	09	1515

Contents of accumulator 1: @61777 90091. Contents of word 1515: @61777 90091. The *equal* indicator is turned on.

DATA FLOW: The contents of the storage word are moved in parallel to the arithmetic register, and the contents of the accumulator are sent to the auxiliary register. The signs of each are compared; and if they are different, the *high* or *low* indicator is turned on, depending on the result of the sign comparison. The field-defined portion of the number in the arithmetic register, and the entire contents (in significant digits) of the auxiliary register, are complement-added, just as in a subtract operation: the digits are brought in the adder in low-order to high-order sequence, with each digit from the arithmetic register changed to its complement value before entering the adder. The result of this operation is not stored. This operation has two purposes: detecting whether any digit other

than zero results from it; and determining whether a carry results from it. To show how these factors determine the result of the compare operation, let's take three examples. In each, the value in the storage-word field is 005. The accumulator values are: 006 (high comparison), 005 (equal), and 004 (low):

	Low	*Equal*	*High*
Storage word to arithmetic register	005	005	005
Accumulator to auxiliary register	004	005	006
Complement-add 005:	+995	+995	+995
	999	1000	1001
	not zero no carry	zero carry	not zero carry

The *low* indicator is turned on if there is a non-zero result and no carry. A non-zero result with a carry, turns on the *high* indicator. A zero result with a carry, is an *equal* condition.

If there are more significant digits in the accumulator than field definition specifies, the absolute value of the accumulator is, of course, greater. This condition automatically turns on the *high* indicator if the signs are both plus or alpha, or the *low* indicator if the signs are both minus.

REGISTERS AFFECTED: The arithmetic register, auxiliary register, adder, and the compare indicators.

TIMING: The duration of a compare instruction is determined by the number of significant digits in the accumulator, or the number of digits specified by field definition, whichever is greater (Figure 70).

Number of Digit Positions	1	2	3	4	5	6	7	8	9	10
Microseconds	36	48	48	48	60	60	60	72	72	72

FIGURE 70. TIMING — COMPARE ACCUMULATOR # TO STORAGE

These timings are the same as complement-add to accumulator.

COMMENTS: Note that the sign is more important than the value, in compare operations. If the signs are different, the values have no effect on the result of the comparison. Note, also, that the *entire* accumulator is always used; field definition is used with the storage word only.

Accumulators can be addressed by these codes. This means that two accumulators can be compared with each other; or an entire accumulator can be compared with a field-defined portion of itself, to determine whether the remaining positions are all zeros.

After a compare operation one of the three indicators, *high, low,* or *equal,* is ON and the other two are OFF.

AUTOCODER EXAMPLES (Figures 71 and 72): Assume that FRIDAY has been previously defined as positions 6-9 of word 2120. The instruction assembled from Figure 71 is:

S01	23	45	6789
+35	00	69	2120

The actual address 502 is used. The instruction assembled from Figure 72 is:

S01	23	45	6789
+25	39	56	0502

Compare Absolute in Accumulator 1 to Absolute in Storage

−15 **CA**

MACHINE DESCRIPTION: This code is similar to compare accumulator to storage, with two exceptions:

1. The signs of both factors are ignored. The comparison is made of the values only, both considered plus.

2. This code can be used with accumulator 1 only.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60		65 70 75
01		C,3	FRIDAY		

FIGURE 71.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60		65 70 75
01		C,2	502(5,6)+X39		

FIGURE 72.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −15.

23 Indexing word.

45 Field definition.

6789 Address of the word in storage whose field-defined digits are to be compared, in absolute value, with the entire contents of accumulator 1. Indexable.

EXAMPLES: To compare, in absolute values, the entire contents of accumulator 1 with positions 0-4 of word 2814:

S01	23	45	6789
−15	00	04	2814

Contents of accumulator 1: +00000 01235. Contents of word 2814: −02235 65222. The *low* indicator is turned on, even though the storage word is minus.

To compare, in absolute value, the entire contents of accumulator 1 with all of word 2005:

S01	23	45	6789
−15	00	09	2005

Contents of accumulator 1: +00000 00000. Contents of word 2005: −00000 00000. The *equal* indicator is turned on.

DATA FLOW: Same as compare accumulator # to storage, except that the signs are ignored, and the two factors are compared in all cases.

REGISTERS AFFECTED: Same as compare accumulator # to storage.

TIMING: Same as compare accumulator # to storage.

COMMENTS: As with all absolute-value instructions, this code can be used with accumulator 1 only, specified in the operation code. Accumulator addresses can be used in positions 6-9, however, for comparison with accumulator 1.

AUTOCODER EXAMPLE (Figure 73): Assume that SATURDAY has been defined as the six high-order positions of word 1509. The assembled instruction is:

S01	23	45	6789
−15	00	46	1509

Compare Storage to Digit
+03 **CD**

MACHINE DESCRIPTION: The contents of a single specified digit position of the storage word addressed by positions 6-9 (indexable), are compared with a single-digit value contained in the instruction itself. The comparison is of absolute values, and the *high, equal,* or *low* indicator is turned on, determined by the value of the digit in storage. If the storage digit is greater than the instruction digit, the *high* indicator is turned on; if it is less, the *low* indicator is turned on. If they are equal, the *equal* indicator is turned on.

INSTRUCTION FORMAT: S01 23 4 5 6789

S01 +03.

23 Indexing word.

4 0-9, specifying the digit value that the storage-word digit is to be compared with.

5 0-9, designating the position in the storage word that contains the digit to be compared.

6789 Address in storage of the word that contains the digit to be compared. Indexable.

EXAMPLE: To compare the units position in word 2932 with a zero:

S01	23	4	5	6789
+03	00	0	9	2932

Contents of word 2932: −12345 67890. The *equal* indicator is turned on.

DATA FLOW: The entire contents of the storage word are brought to arithmetic register. The digit to be tested is complement-added to the digit in position 4 in the program register. If the result is zero, the *equal* indicator is turned on. If the result is non-zero with a carry, the *low* indicator is turned on. If the result is non-zero with no carry, the *high* indicator is turned on.

REGISTERS AFFECTED: The arithmetic register, adder, and the compare indicators.

TIMING: 36 microseconds.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
01		CA	SATURDAY(4,6)		

FIGURE 73.

COMMENTS: This compare instruction doesn't necessarily involve an accumulator. Accumulators can be addressed, however, in positions 6-9.

If there is a 0 in position 4 of the instruction, only the *equal* or *high* indicator is turned on. Similarly, a 9 in position 4 can cause only the *equal* or *low* indicator to be turned on.

AUTOCODER EXAMPLE (Figure 74): Assume SUNDAY to have been defined as word 1948. The assembled instruction is:

S01	23	4	5	6789
+03	00	6	8	1948

This instruction compares the digit in position 8 of word 1948, with a 6.

Sign Control —03

MACHINE DESCRIPTION: This augmented code is used for four different purposes:

1. Comparing the sign of a storage word with the sign value specified in the instruction; and turning on the *high, equal,* or *low* indicator as a result.
2. Making the sign of a word plus, minus, or alpha, as specified in the instruction, regardless of its previous sign.
3. Setting the sign-change sense/stop switch to sense or stop.
4. Testing the *sign-change* indicator, and branching if it is ON. This operation automatically turns it off if it is on.

INSTRUCTION FORMAT: S01 23 4 5 6789

S01 —03.

23 Indexing word.

4 3, 6, or 9 in the operations for testing and making signs (No.'s 1 and 2 above):
- 3 — Alpha
- 6 — Minus
- 9 — Plus

Not used with the other operations.

5 0-4, specifying the operation:
- 0 — Compare sign*
- 1 — Make sign**
- 2 — Sense mode for sign change SMSC
- 3 — Halt mode for sign change HMSC
- 4 — Branch if sign change BSC

*The compare-sign instruction combines with each sign designation in position 4, to make up an instruction:

Position 4	5		
3	0	Compare Sign to Alpha	CSA
6	0	Compare Sign to Minus	CSM
9	0	Compare Sign to Plus	CSP

**Similarly, the make-sign instruction combines with the three sign designations:

Position 4	5		
3	1	Make Sign Alpha	MSA
6	1	Make Sign Minus	MSM
9	1	Make Sign Plus	MSP

6789 For comparing or setting the sign, the location of the word whose sign is to be set or compared with. For testing the sign-change indicator, these positions contain the branch address (indexable in both cases). Not used for the sense/stop switch operations.

COMPARE SIGN: The sign of the storage word addressed by positions 6-9 (indexable) is compared with the sign designation in position 4 of the instruction.

An alpha sign has a value of 3, a minus sign 6, and a plus sign 9. If the storage-word sign is greater than the sign in position 4, the *high* indicator is turned on; if it is less, the *low* indicator is turned on. If the signs are the same, the *equal* indicator is turned on. The storage word is not changed by this operation.

MAKE SIGN: The sign of the storage word addressed by positions 6-9 (indexable) is made the same as the sign designated in position 4 of the instruction, regardless of what it had been previously.

SENSE MODE FOR SIGN CHANGE: The sign-change sense/stop switch is set to sense. If it is already at sense, this instruction has no effect. When the switch is at sense, a sign-change condition arising in the program turns on the *sign-change* indicator but does not stop the machine.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
01		CD	SUNDAY(8),6		

FIGURE 74.

HALT MODE FOR SIGN CHANGE: The sign-change sense/stop switch is set to stop. If it is already at stop, this instruction has no effect. When the switch is at stop, a sign-change condition arising in the program turns on the *sign-change* indicator and stops the machine. If the *sign-change* indicator is on when this instruction is given, the machine stops.

To restart the program, press the error reset and start keys on the console operating keyboard.

BRANCH IF SIGN CHANGE: The *sign-change* indicator is tested. If it is on, the address in positions 6-9 (indexable) is transmitted to the instruction counter, and the next instruction is taken from that location. The *sign-change* indicator is turned off, if it is ON.

EXAMPLES: To compare the sign of word 1690 with plus:

S01	23	4	5	6789
−03	00	9	0	1690

Contents of word 1690: −02461 53794. The *low* indicator is turned on.

To make the sign of word 1690 plus:

S01	23	4	5	6789
−03	00	9	1	1690

Contents of 1690 before the operation: −02461 53794. Contents of 1690 after the operation: +02461 53794.

To set the sign-change sense/stop switch to sense:

S01	23	4	5	6789
−03	xx	x	2	xxxx

To branch to location 3000, indexed by positions 2-5 of IW 49, if the *sign-change* indicator is on:

S01	23	4	5	6789
−03	49	x	4	3000

DATA FLOW: For the compare-sign instruction, the contents of the data word are brought in parallel to the arithmetic register. From there, the sign value is compared with the value in position 4 of the instruction, just as in the compare-digit operation. All of the compare indicators are reset, and then the *high, equal,* or *low* indicator is turned on, depending on the comparison.

For the make-sign instruction, the contents of the storage word are brought in parallel to the arithmetic register. The sign designation in position 4 of the instruction is inserted into the sign position of the arithmetic register, the entire contents of which are then moved in parallel back to the storage location.

There is no movement of data involved in the SMSC and HMSC instructions.

For branch if sign change, the only movement of information is the transmission of the address in positions 6-9, indexed if so specified, to the instruction counter, if the *sign-change* indicator is ON. The indicator is turned off.

REGISTERS AFFECTED: For compare-sign, the arithmetic register and the compare indicators (possibly). For make-sign, the arithmetic register and the storage word (possibly). For branch if sign change, the instruction counter and *sign-change* indicator (possibly).

TIMING: 36 microseconds, for all these operations.

COMMENTS: A sign can be fully tested by the COMPARE SIGN-TO-MINUS instruction. If the storage-word sign is plus, the *high* indicator is turned on; if minus, the *equal* indicator; and if alpha, the *low* indicator. The CSA code can never turn on the *low* indicator, and the CSP code can never turn on the *high* indicator.

The *sign-change* indicator and stop/sense switch are two different features. Their relation to each other is the same as described under field overflow for the corresponding features.

AUTOCODER EXAMPLES (Figures 75-78): The actual address 455 is used. The instruction assembled from Figure 75 is:

S01	23	4	5	6789
−03	00	3	0	0455

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5 6	15	16 20	21 25 30 35 40 45 50 55 60		65 70 75
0 1		CSA	455		

FIGURE 75.

SUNDAY has been previously defined as word 1770. The instruction assembled from Figure 76 is:

S01	23	4	5	6789
−03	00	9	1	1770

No operand is required. The instruction assembled from Figure 77 is:

S01	23	4	5	6789
−03	00	0	3	0000

UPSILON is word 1690. The instruction assembled from Figure 78 is:

S01	23	4	5	6789
−03	00	0	4	1720

Branch
+01 **B**

MACHINE DESCRIPTION: The 4-digit address in positions 6-9, indexed if indexing is specified, is transmitted to the instruction counter. The next instruction in the program is taken from that address.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +01.

23 Indexing word.

45 Not used.

6789 Location of the next instruction.

EXAMPLES: To branch unconditionally to location 2000, indexed by positions 2-5 of IW 88:

S01	23	45	6789
+01	88	xx	2000

To branch unconditionally to the address in positions 2-5 of index word 83:

S01	23	45	6789
+01	83	xx	0000

IW 83 must be plus. If it is minus, program branches to the complement of the address.

DATA FLOW: There is no movement of data with this instruction, only the transmission of the address to the instruction counter.

REGISTERS AFFECTED: The instruction counter.

TIMING: 24 microseconds.

COMMENTS: The unconditional branch is usually used for those operations such as card read, punch, inquiry read and reply, in which an invalid character is detected during transmission of data. The next sequential instruction is taken in these cases, and is an unconditional branch to a sub-routine for that particular situation. If no invalid characters are detected, the program automatically bypasses the next sequential instruction location, and proceeds from the location 1 or 2 beyond it.

AUTOCODER EXAMPLE (Figure 79): Assume that KAPPA has been previously defined as word 1080 (or any part of word 1080). The assembled instruction is:

S01	23	45	6789
+01	08	00	1080

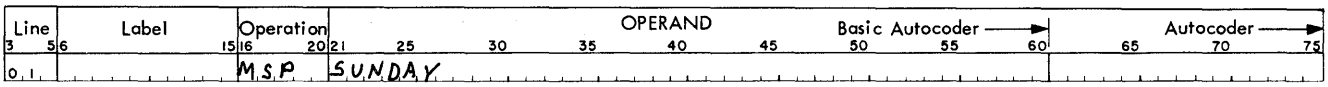


FIGURE 76.

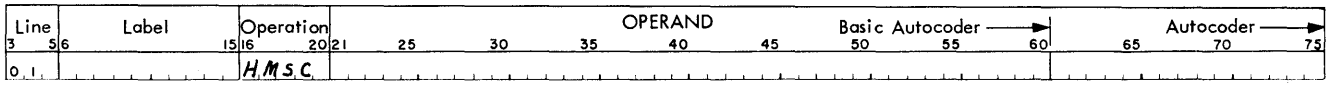


FIGURE 77.

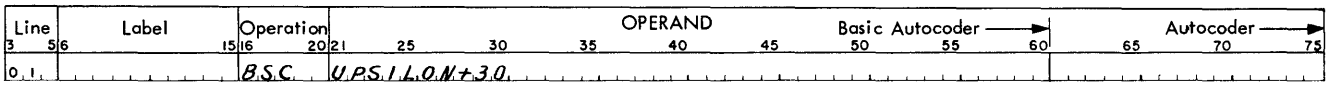


FIGURE 78.

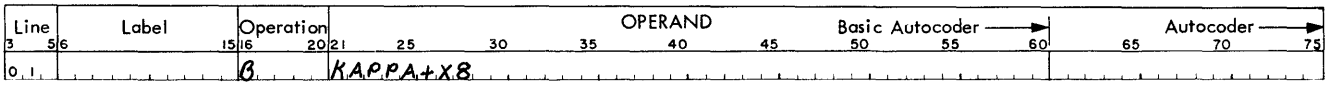


FIGURE 79.

Branch and Load Location in Index Word
+02 BLX

MACHINE DESCRIPTION: This is an unconditional branch and is the same as +01, with this addition:

Before the address (indexable) of this instruction goes to the instruction counter, the present contents of the instruction counter are stored in positions 2-5 of the index word specified in positions 4-5 of this instruction. The remaining positions of the IW are unchanged, and its sign is set to plus.

INSTRUCTION FORMAT:

S01	23	45	6789

S01 +02.

23 Indexing word.

45 Designates the index word that will contain, in positions 2-5, the present contents of the instruction counter.

6789 Location of the next instruction (indexable).

EXAMPLE: To branch unconditionally to location 3111 and store the contents of the instruction counter into positions 2-5 of IW 50:

S01	23	45	6789
+02	00	50	3111

DATA FLOW: The contents of the instruction counter are transmitted in parallel to the index word in core storage, and then the address from this instruction is sent to the instruction counter.

REGISTERS AFFECTED: The instruction counter and the specified index word.

TIMING: 36 microseconds.

COMMENTS: The instruction-counter address stored in the index word is the location of the next instruction that would have been taken if it were not for the branch instruction. It is the location of the branch instruction, +1.

The stored program can be returned to this point by an unconditional branch (+01) instruction with an address of 0000 in positions 6-9, indexed by the index word in which the instruction counter was stored.

Observe the following instruction, in location 1611:

S01	23	45	6789
+02	71	71	0000

Assume the contents of IW 71 to be: +00 1234 0000. Indexing takes place first; the 1234 in positions 2-5 of 0071 are added to the 0000 in positions 6-9 of the program register, and the result (1234) goes back to positions 6-9 of the program register. The 1612 in the instruction counter is then brought to positions 2-5 of 0071. The next instruction is taken from 1234. If the program later returns to 1611 without changing 0071, this instruction is just like a no-op; the BLX instruction in 1611 will cause a *branch* to 1612.

AUTOCODER EXAMPLE (Figure 80): The assembled instruction is:

S01	23	45	6789
+02	00	33	1901

Halt and Branch
+00 HB

MACHINE DESCRIPTION: This operation causes an unconditional machine stop. The console typewriter automatically types the contents of the instruction counter and program register. Pressing the start key on the operating keyboard restarts the program. The next instruction is taken from the location in positions 6-9 of the instruction (indexable).

INSTRUCTION FORMAT:

S01	23	45	6789

S01 +00.

23 Indexing word.

45 Not used.

6789 Branch address.

EXAMPLE: To stop the machine and resume the stored program with the instruction in word 1991:

S01	23	45	6789
+00	00	xx	1991

DATA FLOW: The contents of the instruction counter and program register are transmitted to the console typewriter. The contents of positions 6-9, indexed if so specified, are moved to the instruction counter.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		BLX	33, 1901		

FIGURE 80.

REGISTERS AFFECTED: The instruction counter.

TIMING: 36 microseconds is the duration of the instruction, but the program cannot continue until the operator presses the start key on the console.

COMMENTS: This code can be of considerable value in program testing. The program can be made to run to a predetermined point and stop there, by use of this code.

The program register contains the halt and branch instruction, when its contents are typed. The instruction counter contains the location of the HB instruction, +1.

AUTOCODER EXAMPLE (Figure 81): Assume that LAMBDA has been previously defined as word 2462. The assembled instruction is:

S01	23	45	6789
+00	65	00	2462

Halt and Proceed
−00 **HP**

MACHINE DESCRIPTION: This code is the same as HALT AND BRANCH, except that there is no branch when the program is restarted. The next instruction is taken from the location in the instruction counter.

INSTRUCTION FORMAT: S01 23 456789

S01	−00.
23	Indexing word. Even though there is no address with this instruction, positions 6-9 are modified by positions 2-5 of an indexing word if one is specified here.
456789	Not used.

EXAMPLE: To halt and proceed:

S01	23	456789
−00	00	xxxxxx

DATA FLOW: The contents of the instruction counter and program register are transmitted to the console typewriter.

REGISTERS AFFECTED: None.

TIMING: 36 microseconds is the duration of the instruction, but the program cannot continue until the operator presses the start key on the console.

COMMENTS: Like halt and branch, this code can be used in program testing, to stop the program at a predetermined point.

When its contents are typed, the program register contains the halt-and-proceed instruction. The instruction counter contains the address of the first instruction to be executed when the program is restarted.

AUTOCODER EXAMPLE (Figure 82): Note that no operand is necessary. The assembled instruction is:

S01	23	456789
−00	00	000000

No Operation
−01 **NOP**

MACHINE DESCRIPTION: No operation is performed. The next sequential instruction is executed.

INSTRUCTION FORMAT: S01 23 456789

S01	−01.
23	Indexing word. Indexing always takes place, if an IW is specified here, even though positions 6-9 are not used.
456789	Not used.

EXAMPLE: To perform no operation:

S01	23	456789
−01	00	xxxxxx

DATA FLOW: None.

REGISTERS AFFECTED: None.

TIMING: 36 microseconds.

COMMENTS: This code is used when the stored program is automatically brought to a location, but it is not desired to use that instruction. An example of this is an inquiry-read instruction, which takes the next sequential location as the next instruction, if an invalid character is detected on transmission of the request data from the synchronizer to core storage. If it is desired to ignore this validity check, the no-op instruction is used. The next sequential instruction is taken; this is the instruction automatically taken if there is no validity error.

AUTOCODER EXAMPLE (Figure 83): No operand is required. The assembled instruction is:

S01	23	456789
−01	00	000000

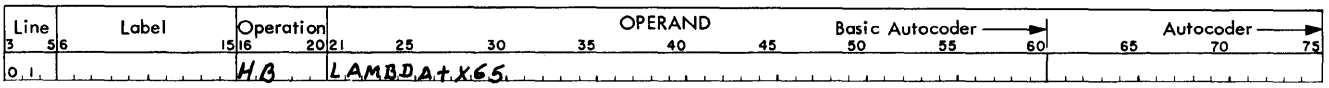


FIGURE 81.

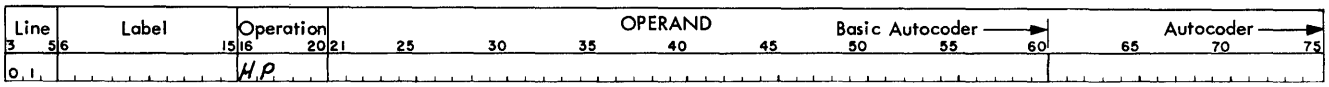


FIGURE 82.

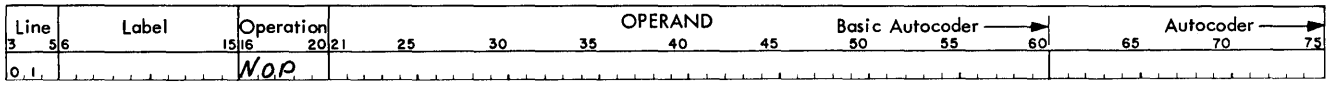


FIGURE 83.

Index-Word Codes

These operation codes are used directly with the 99 index words in the 7070—modifying them, testing them, loading and unloading them, etc. For all of these operations, the index word involved is indicated by positions 4-5 of the instruction. Positions 6-9 can be indexed by an indexing word specified by the IW portion (positions 2-3), just as any other instructions. There is no field definition in the index-word codes, because positions 4-5 of the instructions are always used to denote the index word being operated on.

The format of an index word is:

S	01	2345	6789
sign	not used	indexing portion	non-indexing portion

The non-indexing portion can be used for decrements, limits, and constants. An index word operated on by one of these codes can be plus, minus, or alpha. The sign of the the index word is controlled by the indexing portion; any operation that causes a sign change in the indexing portion changes the sign of the entire index word. Alpha index words are treated as plus in index-word operations that involve arithmetic. (An alpha index word cannot be used for indexing; however, an index word specified in positions 4-5 of these codes can be alpha, but an alpha index word can never be specified in digits 2-3 of any instruction.)

The index-word operation codes are categorized and presented in Figure 84.

The letter X is used in all *Autocoder* symbols for the index-word instructions; it is not used in any other instructions. All of the index-word operations are in the "40's"; position 0 of the instruction is always 4.

The *field-overflow* and *sign-change* indicators are never affected by an index-word instruction. Neither indicator can ever be turned on by an index-word load, unload, or modify instruction.

Index Word Load
+45 **XL**

MACHINE DESCRIPTION: The entire contents, including sign, of the word addressed by positions 6-9 (indexable) are brought to the index word specified by positions 4-5, replacing what was there.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+45.
23	Indexing word, for modifying the address in positions 6-9.
45	The index word that is to contain the contents of the storage word.
6789	Address of the word in storage (indexable).

EXAMPLE: To move the entire contents, including sign, of word 2442 to index word 33:

S01	23	45	6789
+45	00	33	2442

CATEGORY	OP CODES	NAMES	MNEMONICS
Load-unload	+45	Index word load	XL
	−45	Index word unload	XU
	−48	Index word load with interchange	XLIN
Modify	+46	Index word zero and add to indexing portion	XZA
	−46	Index word zero and subtract from indexing portion	XZS
	+47	Index word add to indexing portion	XA
	−47	Index word subtract from indexing portion	XS
	+48	Index word set non-indexing portion	XSN
Branch	−44	Branch if index word is minus	BXM
	+44	Branch if index word indexing portion is non-zero	BXN
	−43	Branch compared index word	BCX
Modify and Branch	+49	Branch incremented index word	BIX
	−49	Branch decremented index word	BDX

FIGURE 84. INDEX-WORD OPERATION CODES

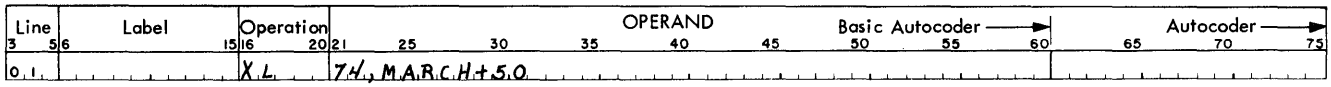


FIGURE 85.

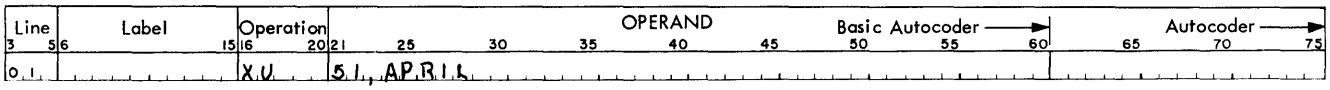


FIGURE 86.

After the operation, the contents of word 2442 and 0033 are identical.

DATA FLOW: The contents of the core-storage word are transmitted in parallel to the auxiliary register, and then back to core storage, to the specified index word location.

REGISTERS AFFECTED: The auxiliary register and the specified index word.

TIMING: 36 microseconds

COMMENTS: Another index word can be addressed by this instruction. In all cases, this code moves all 10 digits and the sign to the index word specified by positions 4-5. Accumulators can also be addressed by this instruction.

AUTOCODER EXAMPLE (Figure 85): Assume that MARCH has been defined as word 3250. The assembled instruction is:

S01	23	45	6789
+45	00	74	3300

Index Word Unload
−45 XU

MACHINE DESCRIPTION: The entire contents, including sign, of the index word specified by positions 4-5 are moved to the storage word addressed by positions 6-9 (indexable).

INSTRUCTION FORMAT: S01 23 45 6789

S01 −45.

23 Indexing word, for modifying the address in positions 6-9.

45 The index word, the contents of which are to be transmitted to the storage word.

6789 Address of the storage word (indexable).

EXAMPLE: To move the contents of index word 92 to word 1950; indexed by positions 2-5 of IW 37:

S01	23	45	6789
−45	37	92	1950

DATA FLOW: The contents of the index word are brought to the auxiliary register, and from there back to core storage, to the addressed location.

REGISTERS AFFECTED: The auxiliary register.

TIMING: 36 microseconds.

COMMENTS: The index word is not changed by this operation. Accumulator addresses and index-word addresses can be used. If an accumulator is addressed, this code is the same as zero-and-add, with a full word field-defined.

AUTOCODER EXAMPLE (Figure 86): Assume that APRIL has been defined as word 1830. The assembled instruction is:

S01	23	45	6789
−45	00	51	1830

Index Word Load with Interchange
−48 XLIN

MACHINE DESCRIPTION: The contents of the storage word addressed by positions 6-9 (indexable) are brought to the index word indicated in positions 4-5. Positions 2-5 and 6-9 of the storage word are interchanged in the index word; the contents of positions 6-9 of the storage word go to positions 2-5 of the index word and vice-versa. The sign and positions 0-1 of the storage word go to their corresponding positions in the index word.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −48.

23 Indexing word, modifying the address in positions 6-9.

45 The index word to be loaded.

6789 Address of the core storage word, the contents of which are to be transmitted to the index word, with positions 2-5 and 6-9 interchanged.

EXAMPLE: To load the contents of word 2358 into index word 62, with positions 2-5 and 6-9 interchanged:

S01	23	45	6789
−48	00	62	2358

Contents of 2358: −12 3456 7890. Contents of index word 62 after the operation: −12 7890 3456, regardless of previous contents.

DATA FLOW: The storage word is brought to the auxiliary register. Positions 2-5 of the auxiliary register are interchanged with positions 6-9. Each digit position shifts one position to the right, with the digit in position 9 going around to position 2; this takes place four times, thus accomplishing the interchange. The contents of the auxiliary register are transmitted to the index word.

REGISTERS AFFECTED: The auxiliary register and the index word.

TIMING: 48 microseconds.

COMMENTS: The storage word is unchanged. Accumulator and index-word addresses can be used with this instruction. The indexing and non-indexing portions of an index word can be reversed by using this instruction with the same index-word specified in the address (contents of positions 4-5 the same as 8-9, with 00 in positions 6-7).

AUTOCODER EXAMPLE (Figure 87): Assume APRIL to be defined as word 1830. The assembled instruction is:

S01	23	45	6789
−48	82	29	1830

Index Word Zero and Add to Indexing Portion +46 XZA

MACHINE DESCRIPTION: The 4-digit number in positions 6-9 of the instruction replaces the previous contents of the indexing portion of the specified index word. The sign of the index word is set to plus. The remaining positions of the index word are unchanged.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +46.

23 Indexing word. Positions 6-9 of this instruction are indexable, just as if they represented an address.

45 The index word, positions 2-5 of which are to be replaced with positions 6-9 of the instruction.

6789 Not an address, but the actual 4-digit number to be inserted into the index word.

EXAMPLE: To place the number 0365 into positions 2-5 of index word 78, and set its sign to plus:

S01	23	45	6789
+46	00	78	0365

DATA FLOW: The contents of the index word are brought from core storage to the auxiliary register. Positions 6-9 of the program register, after indexing if indexing should be called for, are inserted into positions 2-5 of the auxiliary register, replacing what was there. The sign of the auxiliary register is made plus. The contents of the auxiliary register are moved back to the index word.

REGISTERS AFFECTED: The auxiliary register and the index word.

TIMING: 48 microseconds.

COMMENTS: The contents of positions 6-9 are considered plus, and the sign of the index word is automatically set to plus, regardless of the previous sign.

This instruction automatically makes index-word 61 plus, regardless of its previous sign:

S01	23	45	6789
+46	61	61	0000

This instruction in *Autocoder* is shown in Figure 88.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
0 1		X L I N	2 9 , A P R I L + X 8 2		

FIGURE 87.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
0 1		X Z A	6 1 , 0 + X 6 1		

FIGURE 88.

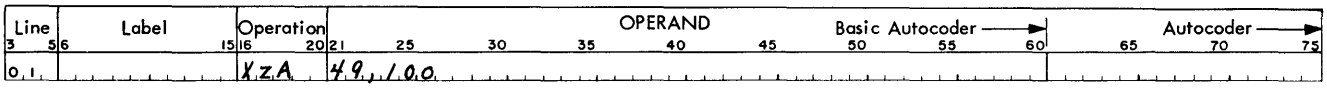


FIGURE 89.

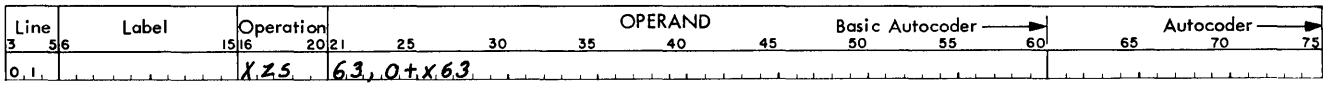


FIGURE 90.

AUTOCODER EXAMPLE (Figure 89): The assembled instruction is:

S01	23	45	6789
+46	00	49	0100

Index Word Zero and Subtract from Indexing Portion −46 XZS

MACHINE DESCRIPTION: The 4-digit number in positions 6-9 of the instruction replaces the previous contents of the indexing portion of the specified index word. The sign of the index word is set to minus. The remaining positions of the index word are unchanged.

INSTRUCTION FORMAT: Same as ZERO AND ADD TO INDEXING PORTION, except for the sign.

EXAMPLE: To place the number 2399 into positions 2-5 of index word 27, and set its sign to minus:

S01	23	45	6789
−46	00	27	2399

DATA FLOW: Same as ZERO AND ADD TO INDEXING PORTION, except that the sign of the auxiliary register is made minus.

REGISTERS AFFECTED: The auxiliary register and the index word.

TIMING: 48 microseconds.

COMMENTS: The index-word sign is set to minus, regardless of its previous sign. This is the only difference between this code and +46.

This instruction automatically makes index word 63 minus:

S01	23	45	6789
−46	63	63	0000

This instruction in *Autocoder* is shown in Figure 90.

AUTOCODER EXAMPLE (Figure 91): The assembled instruction is:

S01	23	45	6789
−46	00	08	0003

Index Word Add to Indexing Portion +47 XA

MACHINE DESCRIPTION: The 4-digit number in positions 6-9 of the instruction, considered plus, is algebraically added to the indexing portion of the specified index word, the sign of which determines whether the sum or difference of the values will be obtained. The result replaces the previous indexing portion of the index word, and the sign of the result becomes the sign of the index word. The remaining positions of the index word are unchanged.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +47.

23 Indexing word. Positions 6-9 of this instruction are indexable, just as if they represented an address.

45 The index word that contains, in positions 2-5, one of the factors in the operation, and the result of the operation.

6789 Not an address, but an actual 4-digit number, which is a factor in the operation.

EXAMPLES: To add the number 25 to the indexing portion of index word 53, and store the result in the indexing portion:

S01	23	45	6789
+47	00	53	0025

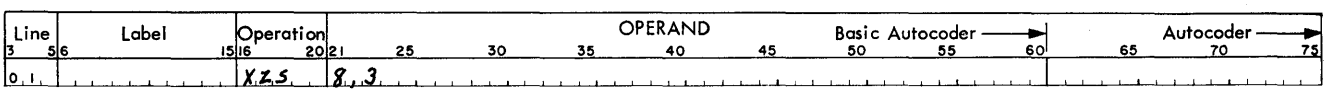


FIGURE 91.

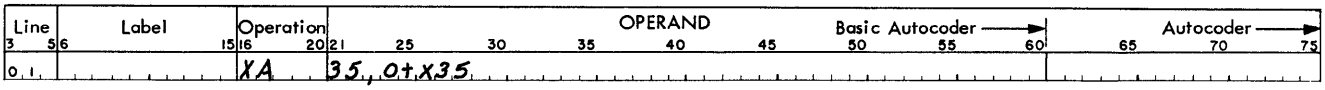


FIGURE 92.

Contents of index word 53 before the operation: +00 1234 5678. Contents after the operation: +00 1259 5678.

To add the number 150 to the indexing portion of index word 80 and store the result in the indexing portion:

S01	23	45	6789
+47	00	80	0150

Contents of index word 80 before the operation: −22 0125 0010. Contents after the operation: +22 0025 0010. The sign of index word 80 is changed.

DATA FLOW: The index word is brought to the auxiliary register.

The digits in positions 2-5 of the auxiliary register are added to the digits in positions 6-9 of the program register (after indexing if it should be specified), and the result is inserted back into positions 2-5 of the auxiliary register as it is developed. If the index-word sign is minus, its factor is complemented as it enters the adder. If the value from the instruction is greater and the index-word sign is minus, the result is recomplemented.

TIMING: 60 microseconds without recomplement, 84 microseconds with recomplement.

COMMENTS: If a carry to a fifth position results from the addition of the two factors, the high-order 1 is ignored, and the four low-order digits of the result are stored in the index word. There is no field-overflow indication given; it acts as if there had not been a carry.

If the sign of the index word is plus or alpha, this operation obtains the sum of the factors. If it is minus, the difference is obtained. A change in the sign of the index word can occur only if the index-word sign was originally minus and the factor in the instruction is greater than that in positions 2-5 of the index word. The sign change can be from minus to plus only. The *sign-change* indicator is never turned on by this operation.

This instruction doubles the value in the indexing portion of index word 35:

S01	23	45	6789
+47	35	35	0000

This holds true only if the value does not reach 5 digits when it is doubled. This instruction in *Autocoder* is shown in Figure 92.

If indexing is used with an XA instruction, and the word used for indexing is minus, a complement number is developed if the indexing value is greater than the number in positions 6-9 of the instruction. Take this instruction as an example:

S01	23	45	6789
+47	14	35	0000

Contents of IW 14: −xx 0007 xxxx. Contents of 0035: −xx 0004 xxxx. Because the IW is minus, the 4-digit factor after indexing becomes +9993 (complement of 0007); it is not recomplemented. The XA instruction adds this factor, considered plus, to the −0004 in the indexing portion of 0035, obtaining a result of +9989. Index word 35 is thus changed to +xx 9989 xxxx by this instruction.

AUTOCODER EXAMPLE (Figure 93): The assembled instruction is:

S01	23	45	6789
+47	00	21	0125

Index Word Subtract from Indexing Portion −47 XS

MACHINE DESCRIPTION: The 4-digit number in positions 6-9 of the instruction, considered plus, is algebraically subtracted from the indexing portion of the specified index word, the sign of which determines whether the sum or difference of the

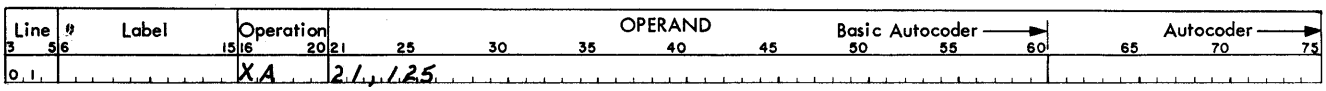


FIGURE 93.

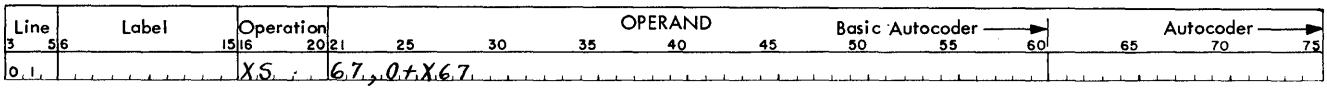


FIGURE 94.

values will be obtained. The result replaces the previous indexing portion of the index word, and the sign of the result becomes the sign of the index word. The remaining positions of the index word are unchanged.

INSTRUCTION FORMAT: Same as ADD TO INDEXING PORTION, except for the sign.

EXAMPLES: To subtract the number 27 from the number in positions 2-5 of index word 59, and store the result in the indexing portion:

S01	23	45	6789
−47	00	59	0027

Contents of index word 59 before the operation: +00 1321 7428. Contents after the operation: +00 1294 7428.

To subtract the number 50 from positions 2-5 of index word 9 and store the result in the indexing portion:

S01	23	45	6789
−47	00	09	0050

Contents of index word 9 before the operation: −00 0250 0000. Contents after the operation: −00 0300 0000.

DATA FLOW: Same as add to indexing portion, with one exception: The factor from the index word is complemented if it is plus, before entering the adder. Recomplement is necessary if the index-word sign is plus, and the instruction factor is greater.

REGISTERS AFFECTED: The auxiliary register, the adder, and the index word.

TIMING: 60 microseconds without recomplement, 84 microseconds with recomplement.

COMMENTS: The comments concerning carry, discussed under ADD TO INDEXING PORTION, apply here.

If the index-word sign is minus, this operation obtains the sum of the factors. If it is plus or alpha, the difference is obtained. If it is plus, sign change occurs if the instruction value is greater. If it is alpha, it remains alpha even if the instruction value is greater. Sign change can be from plus to minus only (the *sign-change* indicator is not turned on, however). In all cases, the correct arithmetic result is obtained.

This instruction automatically reduces the indexing portion of index word 67 to zero:

S01	23	45	6789
−47	67	67	0000

This instruction in *Autocoder* is shown in Figure 94.

AUTOCODER EXAMPLE (Figure 95): The assembled instruction is:

S01	23	45	6789
−47	00	85	0001

Index Word Set Non-Indexing Portion
+48 XSN

MACHINE DESCRIPTION: The non-indexing portion of the specified index word is replaced with the 4-digit number in positions 6-9 of the instruction. The sign and the remaining positions of the index word are not changed.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +48.

23 Indexing word. Positions 6-9 of this instruction are indexable, just as if they represented an address.

45 The index word, the non-indexing portion of which is to be set with the 4-digit number.

6789 Not an address, but the actual 4-digit number to be inserted into positions 6-9 of the index word.

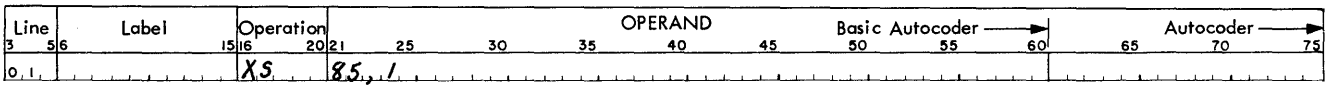


FIGURE 95.

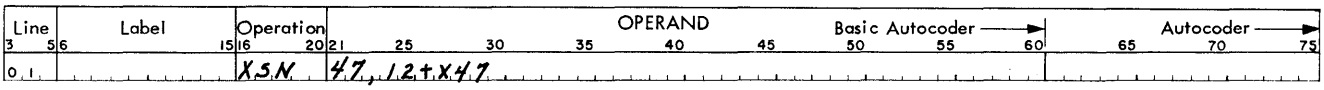


FIGURE 96.

EXAMPLES: To insert a 1 into the non-indexing portion of index word 16:

S01	23	45	6789
+48	00	16	0001

Contents of index word 16 before the operation: +00 0025 0124. Contents after the operation: +00 0025 0001.

DATA FLOW: The contents of the index word are brought to the auxiliary register. The contents of positions 6-9 of the program register, indexed if indexing should be called for, are inserted into positions 6-9 of the auxiliary register. The contents of the auxiliary register are transmitted to the index-word location in core storage.

REGISTERS AFFECTED: The auxiliary register and the index word.

TIMING: 36 microseconds.

COMMENTS: This instruction has almost the same function for the non-indexing portion of an index word, as the ZERO AND ADD TO INDEXING PORTION code (+46) has for the indexing portion. The only difference is with the signs: this instruction has no effect on the sign of the index word.

This instruction makes the non-indexing portion of index word 47 equal to the indexing portion if its sign is plus:

S01	23	45	6789
+48	47	47	0000

If there is a factor in positions 6-9, the instruction makes the non-indexing portion greater than the indexing portion, by that factor. This instruction makes the non-indexing portion of index word 47, 12 greater than the indexing portion (provided that it is plus):

S01	23	45	6789
+48	47	47	0012

If word 47 is minus, this instruction makes the non-indexing portion equal to: the four low-order positions of the sum of the 10's complement of the indexing portion, plus 0012. This instruction in *Autocoder* is shown in Figure 96.

This instruction sets positions 6-9 of index word 28 to 0000:

S01	23	45	6789
+48	00	28	0000

AUTOCODER EXAMPLE (Figure 97): The assembled instruction is:

S01	23	45	6789
+48	00	23	1000

Branch if Index Word Is Minus
−44 **BXM**

MACHINE DESCRIPTION: The sign of the index word specified by positions 4-5 is tested for minus. If it is minus, the program branches to the location specified in positions 6-9 of the instruction (indexable). If the sign is plus or alpha, the program does not branch. The contents of the index word have no effect on the operation.

INSTRUCTION FORMAT: S01 23 45 6789

S01	−44
23	Indexing word for modifying the branch address in positions 6-9.
45	The index word, the sign of which is to be tested.
6789	Branch address, if the index-word sign is minus.

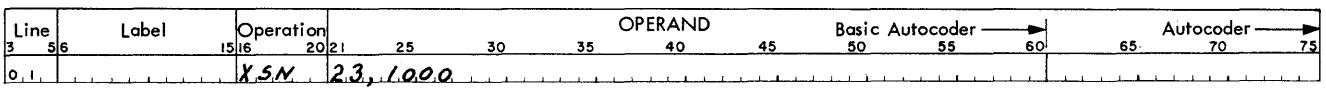


FIGURE 97.

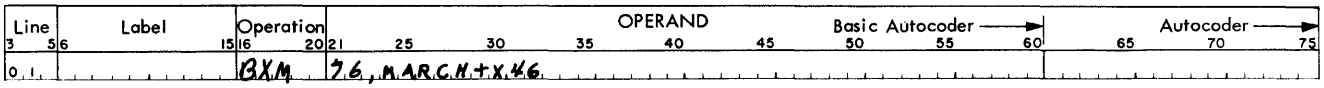


FIGURE 98.

EXAMPLE: To branch to location 2930, if index word 58 is minus:

S01	23	45	6789
−44	00	58	2930

DATA FLOW: The index word is brought to the auxiliary register, where the sign is tested. If it is minus, the contents of positions 6-9, after indexing if so specified in positions 2-3, are transmitted to the instruction counter.

REGISTERS AFFECTED: The auxiliary register and possibly the instruction counter.

TIMING: 36 microseconds.

COMMENTS: The contents of the index word are ignored. The index word is not changed by this operation.

AUTOCODER EXAMPLE (Figure 98): Assume MARCH to be defined as word 2050. The assembled instruction is:

S01	23	45	6789
−44	46	76	2050

Branch if Index Word Indexing Portion Is Non-Zero +44 BXN

MACHINE DESCRIPTION: The indexing portion (positions 2-5) of the index word specified in positions 4-5 of the instruction is tested for zero. If any of the four digits is other than zero, the program branches to the location specified in positions 6-9 of the instruction (indexable). If all four digits are zero, the program does not branch.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +44

23 Indexing word for modifying the branch address in positions 6-9.

45 The index word, the indexing portion of which is to be tested for non-zero.

6789 Branch address if the indexing portion is non-zero (indexable).

EXAMPLE: To branch to location 3817, if positions 2-5 of index word 56 are not all zeros:

S01	23	45	6789
+44	00	56	3817

Contents of index word 56: −00 0000 1234. The program does not branch.

DATA FLOW: The index word is brought to the auxiliary register. Positions 2-5 are tested for zero by being brought through the adder and tested for zero output, as in the compare operations. If any digit is non-zero, the contents of positions 6-9 of the instruction, indexed if specified by positions 2-3, are transmitted to the instruction counter.

REGISTERS AFFECTED: The auxiliary register, the adder, and possibly the instruction counter.

TIMING: 36 microseconds.

COMMENTS: The index word is not changed by this instruction. Like BRANCH COMPARED INDEX WORD, this code can be used to control a program loop. Each time, an instruction in the loop subtracts a one from the indexing portion. This instruction branches back to the start of the loop until the indexing portion is reduced to zero. The program then proceeds with the next sequential instruction.

This instruction tests the address in the indexing portion of index word 87 for zero and branches to that address if it is not 0000:

S01	23	45	6789
+44	87	87	0000

In *Autocoder,* this instruction is shown in Figure 99.

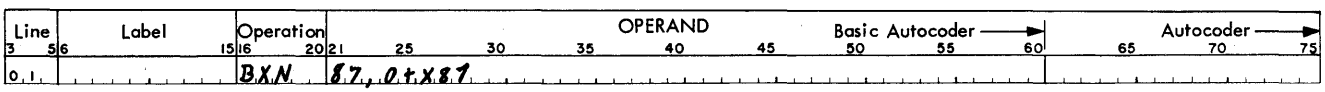


FIGURE 99.

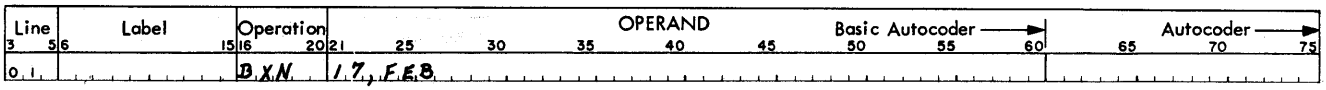


FIGURE 100.

AUTOCODER EXAMPLE (Figure 100): Assume that FEB has been defined as word 1803. The assembled instruction is:

S01	23	45	6789
+44	00	17	1803

Branch Compared Index Word
−43 BCX

MACHINE DESCRIPTION: The value of the indexing portion of the index word specified in positions 4-5 is compared to the value of the non-indexing portion. If the indexing portion is less than, or equal to, the non-indexing portion, the program branches to the location in positions 6-9 (indexable) for its next instruction. If the indexing portion is greater than the non-indexing portion, the program does not branch.

The absolute values of the two portions are compared; the sign of the index word is ignored.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −43

23 Indexing word for modifying the address in positions 6-9.

45 The index word on which the operation is to be performed.

6789 Branch address if the indexing portion is less than, or equal to, the non-indexing portion (indexable).

EXAMPLE: To branch to location 2930 if the contents of positions 2-5 of index word 77 are not greater than the contents of positions 6-9:

S01	23	45	6789
−43	00	77	2930

Contents of index word 77: +00 1234 1234. The two portions are equal; the program branches to location 2930.

DATA FLOW: The index word is brought from core storage to the auxiliary register. Positions 2-5 are brought to one side of the adder, in low-order to high-order sequence. Positions 6-9 are brought to the other side of the adder, converted to their 10's complement. This subtraction performs the comparison, as described under COMPARE ACCUMULATOR # TO STORAGE. If a branch is called for as a result of the comparison, the contents of positions 6-9, indexed if specified in positions 2-3 of the instruction, are transmitted to the instruction counter.

REGISTERS AFFECTED: The auxiliary register, the adder, and possibly the instruction counter.

TIMING: 48 microseconds.

COMMENTS: The index word itself is not changed by this instruction. Note that the index-word sign is ignored and the absolute values of the two portions are compared.

This instruction can be used when the program is in a loop to keep track of the number of times the loop has been executed and to branch out of it after a specified number of times. The difference between the values in the indexing and non-indexing portions represents the number of times the program should go through the loop. An instruction in the loop decreases the indexing-portion value by one, each time it is performed. When the two portions become equal, the program proceeds out of the loop.

AUTOCODER EXAMPLE (Figure 101): Assume that JAN has been previously defined as word 2711. The assembled instruction is:

S01	23	45	6789
−43	13	41	2711

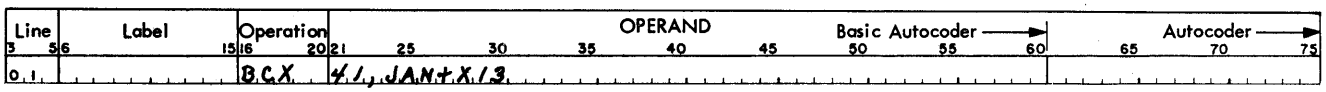


FIGURE 101.

Branch Incremented Index Word
+49 **BIX**

MACHINE DESCRIPTION: The absolute value of the contents of the indexing portion of the specified index word is increased by 1. This new indexing portion is compared with the non-indexing portion. If the absolute value of this new indexing portion is not greater than the absolute value of the non-indexing portion, the program branches to the location in positions 6-9 (indexable). If the indexing portion is greater, the program does not branch.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +49

23 Indexing word for modifying the branch address in positions 6-9.

45 The index word the indexing portion of which is increased by 1 and compared with the non-indexing portion.

6789 The branch address (indexable) if the new indexing-portion value is not greater than the non-indexing value.

EXAMPLES: To increase the indexing portion of index word 75 by 1, and branch to location 2201 if it is not greater than the non-indexing portion:

S01	23	45	6789
+49	00	75	2201

Contents of 0075 before the operation: +00 3329 3330. Contents after the operation: +00 3330 3330. The program branches, because the indexing portion is not greater than the non-indexing portion.

To increase the indexing portion of index word 19 by 1, and branch to location 1651 if it is not greater than the non-indexing portion:

S01	23	45	6789
+49	00	19	1651

Contents of index word 19 before the operation: −00 0012 0012. Contents after the operation: −00 0013 0012. The program does not branch. Note that the absolute value in positions 2-5 is increased; the minus sign is ignored.

DATA FLOW: The index word is brought to the auxiliary register. A digit 1 is generated in the adder and added to the indexing portion in positions 2-5 of the auxiliary register. Positions 2-5 are compared with positions 6-9 of the auxiliary register by the same complement-add operation used by the compare codes (the non-indexing value is complemented before entering the adder). If the indexing portion is less than, or equal to, the non-indexing portion, the contents of positions 6-9 of the program register, indexed if so specified, are transmitted to the instruction counter.

REGISTERS AFFECTED: The auxiliary register, the adder, the index word, and possibly the instruction counter.

TIMING: 60 microseconds.

COMMENTS: This code is an efficient means of controlling a program loop. The number of times the program executes the loop is determined by the difference in the values of the indexing and non-indexing portions of the index word (the non-indexing portion contains the greater value). For example, a program loop that uses data in word 0501, then 0502, etc., through 0525, (executing the loop 25 times), can use this instruction to control the addressing, and determine the end of the loop. At the start of the first loop, index word 45 is +00 0001 0025. An instruction referring to the data has an address of 0500, indexed by index word 45. At the end of the first loop, a BRANCH INCREMENTED INDEX WORD instruction changes IW 45 to +00 0002 0025 and branches back to the start of the loop. The loop is executed 24 more times; IW 45 is then +00 0026 0025, and the program does not again branch to start the loop. Each time the loop is executed, the correct data word, from 0501 to 0525, is used, because the address 0500 was indexed each time by IW 45.

The indexing portion is always increased in absolute value; the sign is ignored. If the indexing portion should be 9999, it is changed to 0000 (but the *field-overflow* indicator is not turned on).

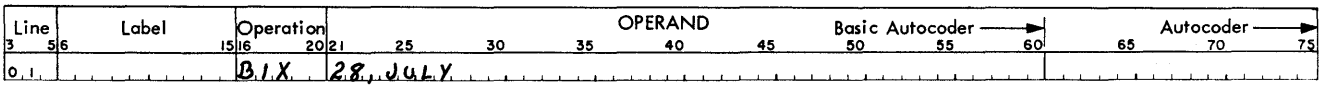


FIGURE 102.

AUTOCODER EXAMPLE (Figure 102): Assume JULY to be defined as location 2631. The assembled instruction is:

S01	23	45	6789
+49	00	28	2631

Branch Decremented Index Word
−49 BDX

MACHINE DESCRIPTION: The non-indexing portion of the specified index word is subtracted from the indexing portion. Because these factors are in the same word, they have the same sign, and this operation always obtains the difference of the factors. This result replaces the previous indexing portion. The sign of the result becomes the sign of the index word. The remaining positions of the index word are unchanged.

If the subtraction does not cause a zero result or sign change (if the non-indexing portion was smaller than the indexing portion), the program branches to the location in positions 6-9 (indexable). If the subtraction causes a zero result or sign change (if the non-indexing portion was equal to or greater than the indexing portion), the program does not branch.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −49

23 Indexing word, for the branch address

45 The index word, which contains the two 4-digit factors to be subtracted, and the result.

6789 Branch address (indexable), if the result of the subtraction does not produce a zero result or sign change.

EXAMPLES: To subtract the non-indexing portion of index word 52 from the indexing portion, place the result in the indexing portion, and branch to location 3700, unless the result is zero or less:

S01	23	45	6789
−49	00	52	3700

Contents of index word 52 before the operation: +00 0040 0005. Contents after the operation: +00 0035 0005. The program branches.

To subtract the non-indexing portion of index word 40 from the indexing portion, place the result in the indexing portion, and branch to location 1401, unless the result is zero or less:

S01	23	45	6789
−49	00	40	1401

Contents of index word 40 before the operation: −00 0005 0020. Contents after the operation: +00 0015 0020. The program does not branch. Note the sign change.

DATA FLOW: The contents of the index word are brought to the auxiliary register. The contents of positions 2-5 and 6-9 are subtracted; positions 6-9 are complemented before entering the adder. The result is entered into positions 2-5, replacing what was there. If the result is non-zero and no recomplement is necessary, the contents of positions 6-9 of the program register, after indexing if so specified, are transmitted to the instruction counter.

REGISTERS AFFECTED: The auxiliary register, the adder, and possibly the instruction counter.

TIMING: 60 microseconds, if no recomplementing is required; 84 microseconds, with recomplementing.

COMMENTS: This instruction is a means of controlling a program loop. The number of times the loop is executed is determined by the number of times the value in the non-indexing portion can be subtracted from the value in the indexing portion before reaching or passing zero. In the same manner as BRANCH INCREMENTED INDEX WORD, this instruction, BRANCH DECREMENTED INDEX WORD, can be used to modify addresses within the loop, as well as determine the number of times the loop is executed. Take, for example, a program that uses data in word 1727, then 1724, then 1721, etc., through 1700 (the loop is executed 10 times). Index word 68 contains +00 0030 0003, at the start of the first loop. Any instruction within the

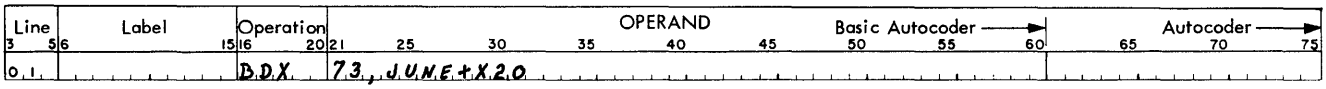


FIGURE 103.

loop that refers to the data has an address of 1697, indexed by IW 68. At the end of the first loop, BRANCH DECREMENTED INDEX WORD changes IW 68 to +00 0027 0003 and returns the program to the first instruction in the loop. Each time through the loop, the data word with an address 3 less than the previous one is used, because the address 1697 is indexed by IW 68. When IW 68 is reduced to +00 0000 0003, the program does not again branch to the start of the loop, and thus word 1697 is not used. If it is desired to process the words in ascending order, 1700-1727, make IW 68 minus: −00 0030 0003. Any instruction within the loop that refers to the data has an address of 1730, indexed by IW 68. The indexing portion will still be reduced to 0027, 0024, etc., by the BRANCH DECREMENTED INDEX WORD instruction.

This code differs from BRANCH INCREMENTED INDEX WORD in these respects:

1. The indexing portion is decreased in absolute value instead of increased.
2. The indexing portion is changed by the value in the non-indexing portion instead of by a constant of 1.
3. The sign of the index word can be changed (neither instruction can change the contents of positions 0-1 or 6-9, however).

AUTOCODER EXAMPLE (Figure 103): Assume that JUNE has been previously defined as word 3500. The assembled instruction is:

S01	23	45	6789
−49	20	73	3500

Block Transmission

One of the outstanding features of 7070 programming is the ability to move a group or several groups of words in a single instruction. The number of words moved and the number of blocks they are divided into are limited only by the capacity of core storage available.

This feature (called *Block Transmission)* is available in the following types of operations:

Core-to-Core Block Transmission—Blocks of data moved from core storage to other parts of core storage.

Table Lookup—Not a transmission of blocks of data in storage, but the blocks comprise the table to be searched.

Magnetic Tape—All reading and writing of magnetic tape.

Disk Storage—All reading and writing in the disk-storage units.

Unit Record—All movement of data from the input synchronizers, to the output synchronizers, and to the console typewriter.

Inquiry—All movement of data from and to the inquiry-control 1 and 2 synchronizers.

In all of these operations, the locations in core storage are divided into *blocks*. Data in a tape record, disk-storage track, or a synchronizer area is, of course, all in one record. In core-to-core block transmission, either the *from* locations or the *to* locations can be divided into blocks, but not both.

SCATTER READ/WRITE: The scattering of data from a tape record, disk track, or input/output synchronizer area into a number of blocks of core storage, and the gathering of data from the blocks to tape, disk, or synchronizer, are called *scatter read/write.*

RECORD DEFINITION WORDS: The locations of the various blocks of storage to be read into or written from are defined by *Record Definition Words* (also referred to in this text as RDW's). In block-transmission instructions, the address portion (positions 6-9) contains the location of the first record definition word to be used. A record definition word contains the first and last addresses of a block of storage words to be read into, or written from. When addressed by the program instruction, the first record definition word is brought to the *Record Definition Register,* which is in a *Channel-Control unit.* A channel control contains all of the circuitry needed to use the record definition words. A full-capacity 7070 has five channel-control units: 1 and 2 are in an IBM 7604 Tape Control Unit and regulate both tape and disk-storage read/write operations for data channels 1 and 2; 3 and 4 are in a second 7604 and regulate only tape, not disk, read/write operations for data channels 3 and 4; and 5, called *Process Channel Control,* is in the IBM 7601 Arithmetic and Program Control Unit. It regulates core-to-core block transmissions, table lookup, unit-record, and inquiry operations.

Channel Controls 1, 2, 3, and 4

Each channel-control unit of the 7070 connects as many as ten magnetic tape units with core storage. Control units 1 and 2 (but not 3 and 4) also connect as many as four disk-storage units; each can be programmed to use either units 1 or 2. A tape unit, however, must use the control to which it is connected. The control units operate independently of each other and of the programming unit. After a read/write operation is started the stored program continues while the operation takes place (unless interlocked by a second read/write instruction for the same channel-control unit).

Figure 104 is a schematic of either channel-control unit 1 or 2. Each unit contains the following:

Record definition register

Record definition word address register

Matching unit (equal-unequal)

Unit adder

Transmission registers A and B

Distributor bus — for parallel transmission within the unit

The function of a channel control is explained here, using a tape-read operation as an example.

1. The word specified by positions 6-9 (indexable) of the operation code is brought to the record definition register, and its address is brought to the RDW

address register. This word contains the address of the first word in the block (called the *starting address)* in positions 2-5, and the address of the last word in the block (called the *stop address)* in positions 6-9. These eight positions and the sign are brought to the record-definition register. The start address becomes the *working address* and specifies the core-storage location that the first word on the tape is to be read into.

2. The working and stop addresses in the record-definition register are matched. Although the working address is brought through the unit adder for the match operation, it is not changed by the adder for this initial test. If the addresses are equal, a special latch is set. The first word from tape is sent to the location specified by the working address via the transmission registers.
3. The special latch is tested. If it is not ON, the working address is incremented by one in the unit adder, and is returned to positions 2-5 of the record definition register; at the same time, it is brought to the matching unit, and matched with the stop address. If they are now equal, the special latch is turned on.
4. The next word from tape is sent to the location specified by the working address via the transmission registers. The process returns to paragraph 3.
5. This process continues until the special latch is set by the working and stop addresses becoming equal. When this happens, the last word is transferred, and the sign position in the record-definition register is tested. If it is plus, the record-definition word address register is increased by 1, a new record-definition word is brought in from storage to designate a new block of words to be read into, and the process

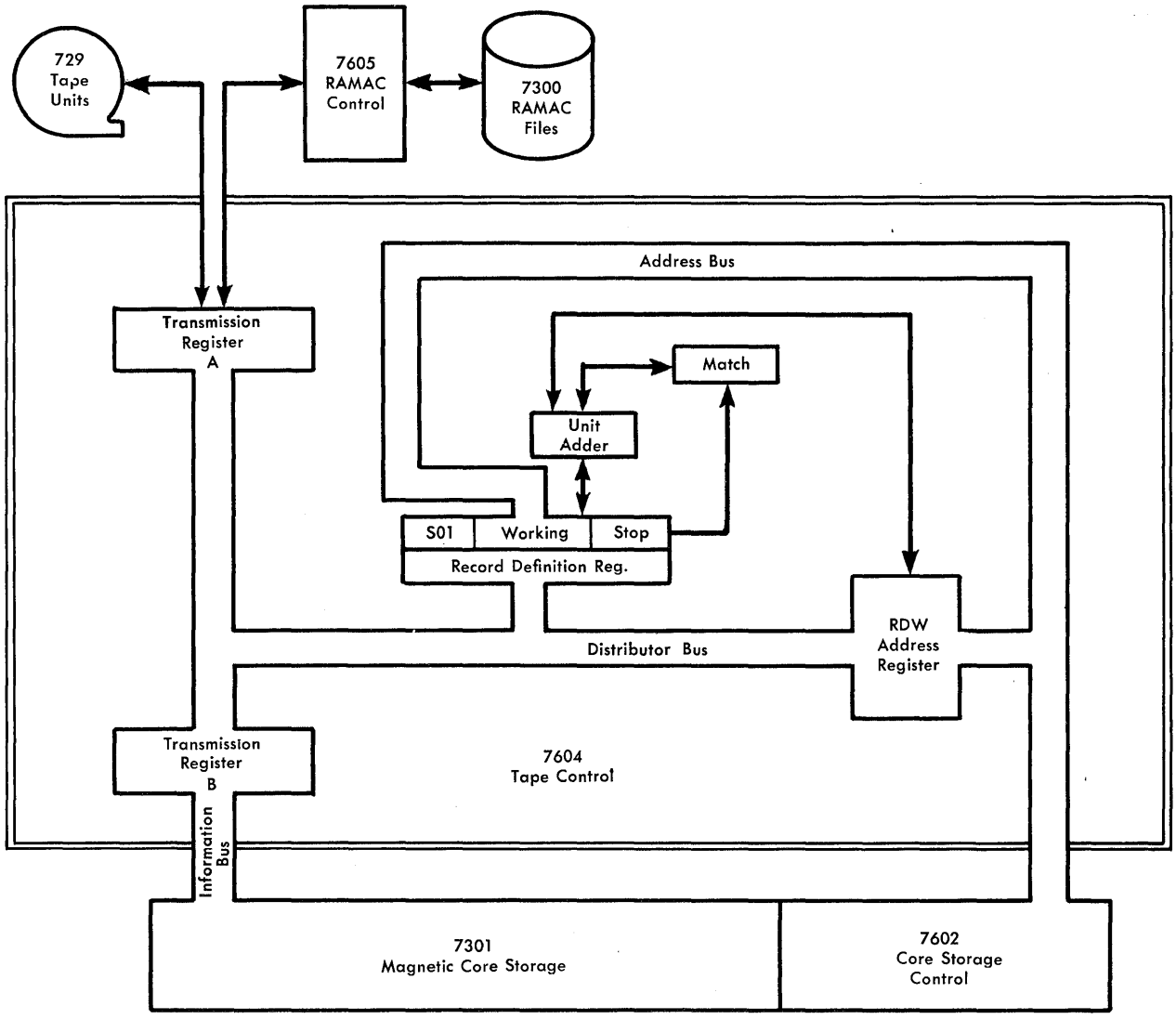


FIGURE 104. CHANNEL CONTROLS 1 AND 2 SCHEMATIC

begins again. If the sign in the record-definition register is minus, it means that this was the last record-definition word to be used in the operation.

The compare operation between the working and the stop addresses is on an equal-unequal basis only. If the start address of the record-definition word should be higher than the stop address, the record-definition register keeps right on working until the address capacity of core storage is reached (word 9990), at which time an error is indicated. Thus, the start address must be smaller than the stop address in all record-definition words.

Data is moved between core storage and tape or disk storage through the transmission registers. The purpose of these registers is to change the type of transmission from serial to parallel or vice versa, and to synchronize between the speeds of tape or disks and the speed of storage. Data must move to and from tape and disk storage serially but is always sent to and from core storage in parallel. In reading a tape or disk-storage record, the characters are read serially into transmission register A until the ten digit positions and the sign position are filled. They are then transmitted in parallel to transmission register B, and register A starts filling up serially again with the next characters on the tape or disk. The contents of register B are sent in parallel to the location specified by the working address of the record-definition register. In tape or disk-storage write operations, data goes in parallel to transmission register B, from there in parallel to transmission register A, and serially to the tape or disk-storage unit.

Process Channel Control

The fifth channel control is called the *Process Channel Control* and is used for block transmission of data to and from the synchronizers for card-reader input, punch and printer output, inquiry, and the console typewriter; and to define the blocks in core-to-core block transmission and table lookup operations. It works in much the same way as channel controls 1 through 4 but uses components of 7601 Arithmetic and Program Control.

The process channel control does not work independently of the stored program. Any operation involving flow of data through the process channel control is completed before the next instruction takes place. (Operation of the card readers, punches, etc., themselves is independent of the program.)

Another important difference between the process channel control and channel controls 1 through 4 is the greater comparing capabilities of the process channel control. At the start of an operation, the start and stop addresses from the record-definition word are compared on a high-low-equal basis, to assure that the start address is not higher than the stop address. If it is higher, no data is moved and an error stop occurs. The adding of the addresses is done by the adder, rather than by a unit adder, and thus an address can be incremented by more than 1 each time. These features make the table lookup increment operations possible: instead of every word in a block being searched, every second word, every third word, etc., is searched, depending on the increment value specified by the operation, and used by the adder to modify the working address. The comparing unit is able to determine when the end of the block has been reached *or exceeded* by this incremented address. In other respects, the function of the process channel control is the same as that of channel controls 1-4.

Figure 105 compares the process channel control with channel controls 1-4, showing the units that perform the corresponding functions:

The function performed in channel control 1-4 by:	Is performed in the process channel control by:
1. Record definition register	1. The auxiliary register*
2. RDW address register	2. RDW address register
3. Matching unit (Equal-Unequal)	3. Comparing unit (High-Low-Equal)
4. Unit adder	4. Adder
5. Transmission register A	5. Synchronizer register*
6. Transmission register B	6. The arithmetic register*

*These units are shown schematically in Figure 8. The synchronizer register accepts data serially from the synchronizers or transmits data serially to the synchronizers, just as transmission register A does with tape and disk storage.

FIGURE 105. COMPARISON OF CHANNEL CONTROLS

Core-to-Core Block Transmission

It is possible to move an entire block of words from one part of magnetic-core storage to another, in a single instruction. There is no limit to the number of words moved, short of the capacity of core storage available. This operation uses the Process Channel Control. The scatter read/write feature in tape and disk file operations is also available in core-to-core block transmissions. It works in much the same way, with the blocks defined by record definition words. A block of words in one part of storage can be scattered into a number of smaller blocks in another part of storage, or can be gathered from a number of blocks. These operations are called *record scatter* and *record gather,* respectively. They are shown schematically in Figure 106.

Core-to-core transmissions differ from other operations that use record definition words, in that two sets of addresses are needed — the core-storage locations that data is moved from, and the locations that data is moved to. Record definition words are used to denote one of these sets of addresses — the *several* blocks that data is scattered *into,* or the *several* blocks that data is gathered *from.*

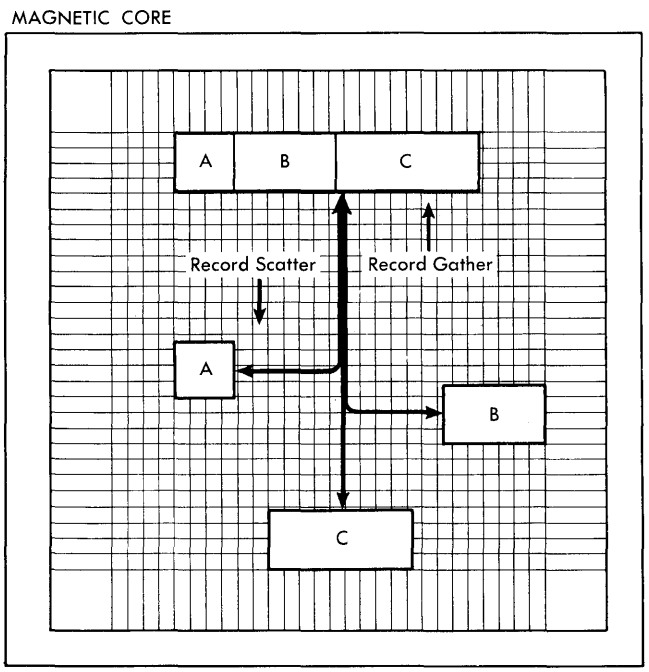


FIGURE 106. CORE-TO-CORE BLOCK TRANSMISSION

The *single* block of core-storage locations that data are *scattered from* or *gathered into* is defined in an index word, specified by positions 4-5 of the core-to-core block transmission instructions. The indexing portion of that word contains the address of the first word of the block.

ALPHA/NUMERICAL CONVERSION: Numerical words can be converted to alpha and vice versa, in core-to-core block transmission. In numerical-to-alpha conversion, every word becomes two words in alpha coding, with alpha signs.

In alpha-to-numerical conversion, every two alpha words are translated into one numerical word by taking the digits in positions 1, 3, 5, 7, and 9 of each word.

There are five operation codes in core-to-core block transmission, presented in the following sequence.

Categories	*Op Codes*	*Names*	*Mnemonics*
Simple transmission	+65	Record scatter	RS
	−65	Record gather	RG
Alpha/numerical conversion	+56	Edit numerical to alphanumerical	ENA
	−56	Edit numerical to alphamerical with sign control	ENS
	+57	Edit numerical to alphamerical with blank insertion	ENB
	−57	Edit alphamerical to numerical	EAN

The RG and RS symbols are the only *Autocoder* symbols that start with R.

Record Scatter
+65 **RS**

MACHINE DESCRIPTION: The contents of a single block of core-storage words, the starting address of which is defined by positions 2-5 of the index word specified by positions 4-5 of the instruction, are moved to one storage block or a series of blocks, under control of record-definition word(s), the first of which is located by positions 6-9 (indexable) of the instruction. Movement of data is under control of the process channel control.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +65.

23 Indexing word for modifying the RDW address in positions 6-9.

45 An index word, positions 2-5 of which contain the location of the first storage word, the contents of which are to be moved.

6789 Location of the first (or only) record-definition word, used to define the area to receive the transmitted data.

EXAMPLE: To transmit the contents of a block of storage words, the starting address of which is in positions 2-5 of index word 86, to two blocks of storage, the first of which is defined by the RDW in location 2841:

S01	23	45	6789
+65	00	86	2841

Contents of index word 86: +00 1721 0000. Contents of word 2841: +00 3511 3520. Contents of word 2842: —00 3101 3105. This instruction moves the contents of words 1721-1730 to words 3511-3520, and the contents of words 1731-1735 to words 3101-3105.

DATA FLOW: The contents of positions 6-9, after indexing if so specified, are brought to the RDW address register. The index word specified in positions 4-5 of the instruction is brought in parallel to the auxiliary register. Positions 2-5 are then brought to positions 6-9 of the program register.

1. The RDW addressed by the RDW address register is brought to the auxiliary register.

The contents of the RDW address register are incremented by 1 (this new address will be needed if the RDW sign is plus). The start and stop addresses are compared, to assure that the start address is not greater. If the addresses are equal, a special latch, called the *match latch,* is set ON.

2. The contents of the word addressed by positions 6-9 of the program register are brought to the arithmetic register. This word is stored in the location specified by the start address in the auxiliary register. (This address can now be considered the *working address.)*

3. The match latch is tested. If it is OFF, each address that is used to address core storage is increased by 1: the working address, in positions 2-5 of the auxiliary register (the *to* address), and the address in positions 6-9 of the program register (the *from* address). The new working address, in positions 2-5 of the auxiliary register, is compared with the stop address, in positions 6-9 of the same register. This comparison is made at the same time that the 1 is added to the working address. The working address is brought to the adder, where a 1 is added to it by the carry circuitry in the adder. As each digit leaves the adder, it is matched with the corresponding digit from positions 6-9 of the auxiliary register. If they are now equal, the match latch is turned on. The process returns to paragraph 2.

4. If the match latch is ON, the operation for this RDW is terminated. The sign of the RDW is tested; if it is minus, the entire operation is terminated. If the RDW sign is plus, the operation returns to paragraph 1.

REGISTERS AFFECTED: The auxiliary register, the arithmetic register, and the adder.

TIMING: 36 microseconds setup time, plus 36 microseconds for each RDW, plus 24 microseconds for each word moved.

COMMENTS: If only one record-definition word is used, this is not strictly a record-scatter operation. It is merely the movement of data from one block of core-storage words to another. This instruction can be used to move a storage block a few locations *up* in storage, replacing all but the last few words in the original block. For example, a storage block in locations 1217-1226 can be moved *up* two locations, to 1215-1224 (after the operation the contents of 1224 equal those of 1226, and 1223 = 1225). This cannot be done in the opposite direction, however; 1217-1226 could not be moved to 1219-1228 by a single RS instruction, for example. The same two words of data would be in 1217-1218, 1219-1220, 1221-1222, etc.

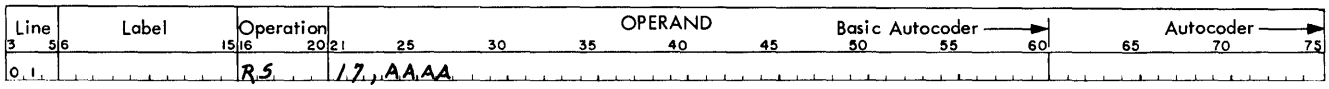


FIGURE 107.

AUTOCODER EXAMPLE (Figure 107): Assume that AAAA has been previously defined as word 2618. The assembled instruction is:

S01	23	45	6789
+65	00	17	2618

Record Gather
−65 RG

MACHINE DESCRIPTION: The address in positions 6-9 (indexable) specifies the first (or only) record-definition word that defines the *from* area.

The starting address of the *to* area is specified in positions 2-5 of the index word designated in positions 4-5 of the instruction. Movement of data is under control of the process channel control.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −65

23 Indexing word, for modifying the RDW address in positions 6-9.

45 An index word, positions 2-5 of which contain the location of the first storage word to be read into.

6789 Location of the first (or only) record-definition word, used to define the transmitting area.

EXAMPLE: To gather the contents of three blocks of storage words, the first of which is defined by the RDW in location 3441; into a single block, the starting address of which is specified in positions 2-5 of index word 77:

S01	23	45	6789
−65	00	77	3441

Contents of index word 77: +00 2001 1234. Contents of word 3441: +00 1401 1405. Contents of word 3442: +00 4110 4115. Contents of word 3443: −00 2710 2713. This instruction moves the contents of 1401-1405 to 2001-2005, the contents of 4110-4115 to 2006-2011, and the contents of 2710-2713 to 2012-2015.

DATA FLOW: Same as record-scatter, with one exception: data is brought from storage to the arithmetic register under control of positions 2-5 of the auxiliary register (the working address from the RDW), and transmitted from the arithmetic register *to* core storage under control of positions 6-9 of the program register (the address from the index word).

REGISTERS AFFECTED: The auxiliary register, the arithmetic register, and the adder.

TIMING: 36 microseconds setup time, plus 36 microseconds for each RDW, plus 24 microseconds for each word moved.

COMMENTS: Like RECORD-SCATTER, RECORD-GATHER using just one RDW is merely the movement of data from one block of core storage to another. A block can be moved *up* a few locations, as described under RECORD SCATTER.

AUTOCODER EXAMPLE (Figure 108): Assume that BBBB has been previously defined as word 4550 The assembled instruction is:

S01	23	45	6789
−65	21	09	4550

Edit Numerical to Alphamerical
+56 ENA

MACHINE DESCRIPTION: This is a record-scatter type of operation. The single block of storage words, located by the starting address in positions 2-5 of the index word specified in positions 4-5 of the instruction, is transmitted to the locations defined by record-definition words, the first of which (or only one) is addressed by positions 6-9 (indexable) of the instruction.

The first word of numerical data (10 digits) is converted to two words (20 digits) of 2-digit alpha representation. The signs of the numerical words are ignored. These two words, with alpha signs attached, are stored in the locations specified by the *start* and *start +1* addresses of the first record-definition word. The process is continued,

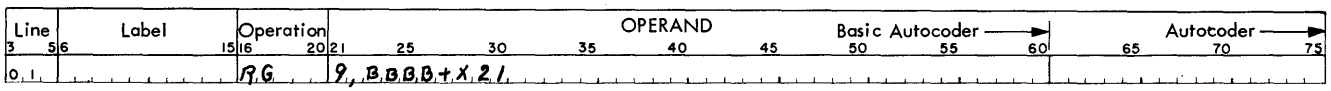


FIGURE 108.

converting words located sequentially in the numerical area and storing in consecutive locations, under the control of the process channel control. The operation terminates after the stop address of the last RDW is used.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +56.

23 Indexing word, for modifying the RDW address in positions 6-9.

45 An index word, positions 2-5 of which contain the location of the first numerical storage word to be read from.

6789 Location of the first (or only) record-definition word, used to define the receiving (alpha) area.

EXAMPLE: To scatter the contents of a block of storage words, the starting address of which is in positions 2-5 of index word 68, and convert each word into two alpha words, the locations of which are defined by RDW's starting in location 548:

S01	23	45	6789
+56	00	68	0548

Contents of index word 68: +00 1875 0000. Contents of word 548: +00 1201 1212. Contents of word 549: +00 2151 2176. Contents of word 550: −00 1700 1705. The 6 words in locations 1875-1880 are converted to alpha and stored in the 12 locations 1201-1212. The next 13 words, in 1881-1893, are converted to alpha and stored in the 26 locations 2151-2176. The next 3 words, in 1894-1896, are converted to alpha and stored in the 6 locations 1700-1705.

DATA FLOW: The movement of data as a result of an ENA instruction is described here, using a simple example: a single numerical word, +123456780, in location 2790, is converted to two alpha words and stored in locations 1601 and 1602. After the operation, 1601 contains @9192939495, and 1602 contains @9697989990. Index word 78 contains 2790 in positions 2-5. The RDW is in location 1575; its contents are −00 1601 1602. The instruction is:

S01	23	45	6789
+56	00	78	1575

1. The initial procedure is the same as for RECORD-GATHER and RECORD-SCATTER:

The 1575 in positions 6-9, after indexing if it were specified, are brought to the RDW address register. Index word 78 is brought to the auxiliary register; the 2790 in positions 2-5 are then brought to positions 6-9 of the program register. Program register: +56 00 78 2790. The contents of RDW 1575 are brought to the auxiliary register, and the 1575 in the RDW address register is incremented by 1 to 1576. Auxiliary register: −00 1601 1602.

The start and stop addresses in the auxiliary register are compared; in this example, 1601 is compared with 1602. If the stop address were *equal to* or less than the start address, an error would be signalled. This is because all of the alphamerical conversion codes must have at least two addresses specified by an RDW.

2. The contents of the address in positions 6-9 of the program register, 2790 in the example, are brought to the arithmetic register. Arithmetic register: +12345 67890.

3. The working address in the auxiliary register, 1601 in this example, is incremented by one, and this new address of 1602 is compared with the stop address, also 1602. The *match latch* is turned on by the equal comparison.

4. The 5 high-order positions of the arithmetic register are serially shifted into the synchronizer register, with 9's inserted to the left of each digit, to form a 10-digit number. The synchronizer register gets an alpha sign. Synchronizer register: @9192939495.

5. The contents of the synchronizer register move in parallel to the arithmetic register, and thence to the storage location addressed by positions 2-5 of the auxiliary register (1601 in this example). Arithmetic register and word 1601: @9192939495.

6. The contents of positions 6-9 in the program register (2790) are again brought to the arithmetic register: +12345 67890.

7. The working address in the auxiliary register (now 1602) is again incremented by one, and compared with the stop address. In this example, 1603 is compared with 1602, resulting in an unequal comparison (the match latch has already been set).

The increment and compare operation at this point in the operation should not cause an equal condition. If an equal is detected here, the total number of words specified by the RDW is an odd number. Because the RDW defines the alpha area, it must specifiy an even number of words. Thus, an equal comparison here would signal an error.

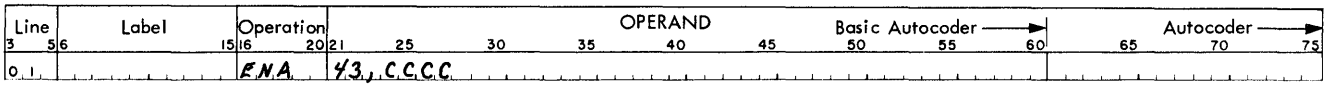


FIGURE 109.

8. The 5 low-order positions of the arithmetic register are serially shifted into the synchronizer register, with 9's inserted to the left of each digit, to form a 10-digit number. The synchronizer register has an alpha sign: @9697989990.

9. The contents of the synchronizer register move in parallel to the arithmetic register and thence to the storage location addressed by positions 2-5 of the auxiliary register (now at 1602). Arithmetic register and word 1602: @9697989990.

The address in positions 6-9 of the program register is incremented by 1. Program register: +56 00 78 2791.

10. The match latch is tested. If it is not ON, the process returns to paragraph 2. If it is ON, as in this example, the sign of the auxiliary register is tested. If it is plus, the next RDW is brought to the auxiliary register, and the process returns to the comparison of start and stop addresses, described in paragraph 1. In this example, the sign is minus, and the operation terminates.

REGISTERS AFFECTED: The auxiliary register, the arithmetic register, the adder, and the synchronizer register.

TIMING: In microseconds:
36 setup time
+36 for each RDW
+120 for each *numerical* word

COMMENTS: In numerical-to-alpha conversion, the lower of the two sequential alpha words converted from a numerical word contains alpha coding for the 5 high-order digits, and the second alpha word contains the alpha coding for the 5 low-order digits, of the numerical word.

The record-definition words define the *alpha* area—the *to* locations in numerical-to-alpha conversion. The total area defined by RDW's is twice as big as the numerical area used. Each RDW must define an even number of storage words. Because the start and stop addresses are inclusive (i. e. the RDW defines the start *through* stop addresses), the difference of the two addresses must be an odd number.

Note that each numerical word is brought from storage twice.

This operation does not distinguish between plus and minus signs in the numerical words; the numerical-word signs are ignored.

AUTOCODER EXAMPLE (Figure 109): Assume that CCCC has been previously defined as word 1905. The assembled instruction is:

S01	23	45	6789
+56	00	43	1905

Edit Numerical to Alphamerical with Sign Control −56 ENS

MACHINE DESCRIPTION: The code operates in the same way as EDIT NUMERICAL TO ALPHAMERICAL (ENA, +56) except:

For all but the units digit of each numerical word, a 9 is inserted into the position to its left in the alpha word, thus producing the alpha representation (90-99) for that digit. The units digit of the low-order alpha word has a 6 or 7 inserted next to it; 6 if the sign of the numerical word is plus, and 7 if it is minus. For example, the word −12345 67890 becomes @9192939495
@9697989970

If alphabetic words are included in the "numerical" area to be converted, the tens position of each resultant low-order alpha word gets a digit 9. For example, the word @6162636465 becomes @9691969296
@9396949695

INSTRUCTION FORMAT: Same as EDIT NUMERICAL TO ALPHAMERICAL, except for the sign.

EXAMPLE: To scatter the contents of a block of storage words, the starting address of which is in positions 2-5 of index word 34, convert each word into two alpha words, the locations of which are defined by RDW(s) starting in location 407; and convert the signs to 6's, 7's or 9's in the 10's digit of every second alpha word, 6 for plus, 7 for minus, and 9 for alpha:

S01	23	45	6789
−56	00	34	0407

Contents of index word 34: +00 2417 0000. Contents of word 407: −00 0901 0920. The 10 words in locations 2417-2426 are converted to alpha and stored in the 20 locations 901-920. The 10's digits of words 902, 904, 906, etc. are all either 6, 7, or 9 depending on the signs of the words from which they were converted.

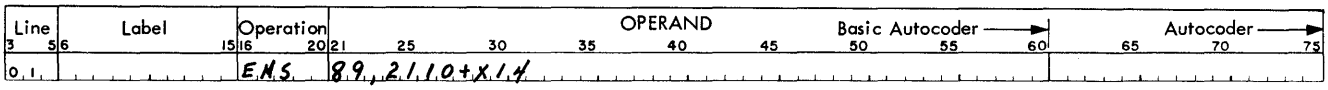


FIGURE 110.

DATA FLOW: Same as edit numerical to alphamerical, with one exception, in paragraph 6:

A 9 is inserted to the left of each of the four high-order digits (positions 5-8 of the original numerical word) as they are shifted into the synchronizer register. A 6 is inserted to the left of the units digit if the sign in the arithmetic register is plus; a 7 is inserted if the sign is minus; or a 9 is inserted if the sign is alpha.

REGISTERS AFFECTED: Same as EDIT NUMERICAL TO ALPHAMERICAL.

TIMING: Same as EDIT NUMERICAL TO ALPHAMERICAL.

COMMENTS: Use of this code enables the program to record the signs of the numerical records when they are converted to alpha. Note that the 10's digit of each second alpha word is 6, 7 or 9 if this code is used; it is always 9 if ENA is used. In all cases, the words are given alpha signs.

AUTOCODER EXAMPLE (Figure 110): Actual address 2110 is used. The assembled instruction is:

S01	23	45	6789
−56	14	89	2110

Edit Numerical to Alphamerical with Blank Insertion +57 ENB

MACHINE DESCRIPTION: This code operates in the same way as EDIT NUMERICAL TO ALPHAMERICAL (ENA, +56) except:

Zeros, instead of nines, are inserted in even digit positions (0, 2, 4, 6, 8) to the left of each leading zero from the numerical word. Thus, blank alphamerical characters (00) appear to the left of the most significant digits in the resulting alphamerical words. For example, the word +00002 57841 becomes @0000000092 @9597989491

INSTRUCTION FORMAT: Same as EDIT NUMERICAL TO ALPHAMERICAL, except for the operation code.

EXAMPLE: To scatter the contents of a block of storage words, the starting address of which is in positions 2-5 of index word 52, convert each word into two alpha words, the locations of which are defined by RDW(s) starting in location 1222; and convert each insignificant zero in the original numerical words to 00 instead of 90:

S01	23	45	6789
+57	00	52	1222

Contents of index word 52: −45 2300 0000. Contents of word 1222: +00 1001 1016. Contents of word 1223: −00 2713 2718. The 8 words in locations 2300-2307 are converted to alpha and stored in the 16 locations 1001-1016. The next 3 words, in locations 2308-2310, are converted to alpha and stored in the 6 locations 2713-2718. High-order insignificant zeros in the numerical words are each converted to 00 instead of 90.

DATA FLOW: Same as EDIT NUMERICAL TO ALPHAMERICAL, with one exception, in paragraph 4:

As the digits are shifted into the synchronizer register, the entire arithmetic register is tested for significant digits, and a digit 0 is inserted to the left of each insignificant zero; a 9 is inserted for all significant digits.

REGISTERS AFFECTED: Same as EDIT NUMERICAL TO ALPHAMERICAL.

TIMING: Same as EDIT NUMERICAL TO ALPHAMERICAL.

COMMENTS: A numerical-to-alpha operation can use blank insertion (+57, ENB), sign control (−56, ENS), or neither (+56, ENA). Blank insertion and sign control combined cannot be used by a single instruction.

If a numerical word is all zeros and the ENB instruction is used, two full alpha words of zeros are created.

AUTOCODER EXAMPLE (Figure 111): Assume that DDDD has been previously defined as word 1719. The assembled instruction is:

S01	23	45	6789
+57	00	54	1719

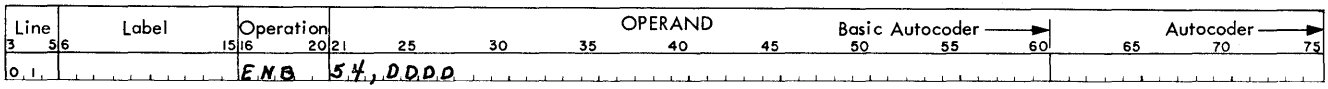


FIGURE 111.

Edit Alphamerical to Numerical
−57 **EAN**

MACHINE DESCRIPTION: This is a record-gather type of operation. Record-definition words, addressed by positions 6-9 (indexable) of the instruction, define blocks of alpha words, which are converted to numerical words and stored in a single storage block, the starting address of which is in positions 2-5 of the index word specified in positions 4-5 of the instruction.

Each pair of two sequential alpha words is converted into a single numerical word. Positions 1, 3, 5, 7, and 9 of the first alpha word become positions 0-4 of the numerical word; those same positions of the second alpha word become positions 5-9 of the numerical word. Position 8 of the second alphamerical word is interrogated. If it is anything but a 7, a plus sign is assigned to the resultant numerical word; if 7, a minus sign is assigned. The operation is under control of the process channel control.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −57.

23 Indexing word, for modifying the RDW address in positions 6-9.

45 An index word, positions 2-5 of which contain the first storage location to be filled with the numerical data.

6789 Location of the first (or only) record-definition word, used to define the transmitting area.

EXAMPLE: To gather the contents of one or several blocks of storage, the locations of which are defined by RDW's starting in location 2691, into a single block the starting address of which is in positions 2-5 of index word 96; converting each two words into a single word:

S01	23	45	6789
−57	00	96	2691

Contents of index word 96: +00 4651 0000. Contents of word 2691: +00 0746 0755. Contents of word 2692: −00 1301 1302. The 10 words in locations 746-755 are converted to numerical and stored into the 5 locations 4651-4655. The two words in locations 1301 and 1302 are converted to numerical and stored into word 4656. The 10's digit of each second word, 747, 749, 751, etc. and 1302, are tested for 7 or non-7. For each 7, the sign of the corresponding numerical word is made minus; for each non-7, it is made plus.

DATA FLOW: The concept of data flow for the alpha-to-numerical instruction is the same as that of the numerical-to-alpha codes. Data is brought from storage to the arithmetic register, shifted into the synchronizer register, and the converted data is brought from the synchronizer register to the arithmetic register, to storage. The RDW's function in the same manner as for numerical-to-alpha, also. The start and stop addresses are compared, to assure that the start address is less than the stop address. The RDW is brought to the auxiliary register to be used.

The differences in data flow between numerical-to-alpha and alpha-to-numerical are due to the fact that the former are record-scatter codes, whereas alpha-to-numerical conversion (EAN) is a record-gather operation:

1. Data is brought to the arithmetic register under control of the working address in the auxillary register.

2. For the first of the two alpha words, positions 1, 3, 5, 7, and 9 of the arithmetic register are serially shifted into positions 0-4 of the synchronizer register. Positions 0, 2, 4, 6, and 8 are ignored.

3. For the second of the two alpha words, positions 1, 3 ,5, 7, and 9 are serially shifted into positions 5-9 of the synchronizer. Positions 0, 2, 4, and 6 are ignored. Position 8 is tested for a 7. If it is 7, the synchronizer register gets a minus sign; otherwise, a plus sign.

4. The contents of the synchonizer register are moved to the arithmetic register, and thence to storage, under control of the address in positions 6-9 of the program register (originally the index-word address).

REGISTERS AFFECTED: The auxiliary register, the arithmetic register, the adder, and the synchronizer register.

TIMING: Same as numerical-to-alpha in microseconds:
36 setup time
+36 for each RDW
+120 for each *numerical* word

COMMENTS: The only kind of data that should be converted by this code is numerical information in the two-digit alpha coding (codes 90-99), information that may have been previously converted to two-digit coding by one of the numerical-to-alpha instructions. Note that positions 0, 2, 4, 6, and 8 of each word are lost, regardless of what they contained.

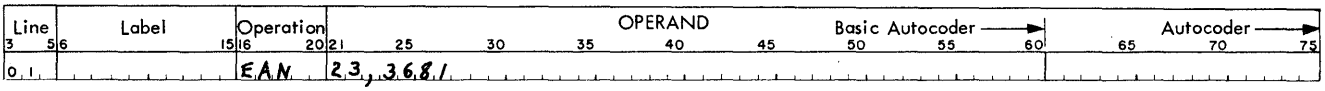


FIGURE 112.

Data from the first of each pair of alpha words goes to positions 0-4 of the numerical word, and data from the next sequential alpha word goes to positions 5-9. As with the numerical-to-alpha codes, each RDW must designate an even number of addresses; the RDW's define the alpha area in alpha-to-numerical conversion, as well as in numerical-to-alpha conversion.

Sign control is automatic with the EAN code. position 8 of each second alpha word is tested for 7 or non-7.

This instruction can place the resultant numerical data back into the first half of the areas occupied by the original alpha data. These are two reasons why it can do this, when numerical-to-alpha operations cannot:

1. The resultant numerical data takes only half as much storage as the original alpha data.

2. The EAN instruction obtains each word from storage only once, whereas the numerical-to-alpha codes obtain each word twice.

AUTOCODER EXAMPLE (Figure 112): The actual address 3681 is used. The assembled instruction is:

S01	23	45	6789
−57	00	23	3681

Table Lookup

An IBM 7070 program is able to search tables of information stored in blocks of magnetic-core storage. The search starts at the first word of the table and continues until it finds the word it is looking for, or until it has searched the entire table. Moreover, the table doesn't have to be completely in one storage block. It can be broken up into a number of blocks, and all blocks are searched in a single operation.

The blocks are defined by record definition words.

A table lookup operation consists of comparing a search value called the *search argument* with the table values, each of which is called a *table argument*. Accumulator 3 contains the search argument for the operation. The comparison is for a table value that is equal to the search argument, equal or higher, or lower, depending on the instruction.

INCREMENT: On all table lookup operations, the search address (working address part of the record definition register) is incremented by the value in the non-indexing portion of index word 98 (non-indexing portion considered plus no matter what sign the index word has). This means that, instead of every word on the table being searched, every second, third, etc., word is searched, depending on the increment value specified. If, for example, the increment value is 0004 and the start address is 0700, words 0700, 0704, 0708, etc., are searched. If the stop address should be 0783, word 0780 is searched, but 0784 is not. The first word in each block is always used, regardless of the increment. There must be an increment; 0000 in the non-indexing portion of 0098 causes an error stop.

RESULTS: When the operation is ended, the location of the found table argument is placed in the indexing portion of index word 98. The remaining positions of index word 98 are unchanged, and its sign is set to plus. If a table location is not found on table lookup operations before the stop address of the last record definition word is exceeded, the next sequential instruction is taken and the indexing portion and sign of index word 98 is not changed. If a table location is found, the next instruction will be taken from the location of the table lookup instruction, +2 .

There are three table lookup instructions as shown in Figure 113.

CATEGORY	OP CODES	NAMES	MNEMONICS
Table look-up	+66	Lookup lowest	LL
	+67	Lookup equal only	LE
	+68	Lookup equal or High	LEH

FIGURE 113. TABLE LOOKUP OPERATION CODES

These are the only codes whose *Autocoder* symbols start with the letter L.

Lookup Lowest
+66 LL

MACHINE DESCRIPTION: The search argument in accumulator 3 is compared with the field-defined portions of the storage words addressed by the RDW, the location of which is designated in positions 6-9 (indexable). Signs are included in the comparisons. The entire accumulator is used; a search argument of less than 10 digits must be in the low-order portion of the accumulator. The search continues through sequential or incremented locations, determined by the value in positions 6-9 of index word 98. The table can be in a single block of storage or several blocks, depending on the RDW signs(s).

The entire table is searched, and the lowest value and its address obtained, provided that the table value is lower than that of the original search argument. At completion of the operation, the lowest found value is in accumulator 3, if one was found; or the search argument is in accumulator 3, if a lower value was not found. The location of the lowest found value is in positions 2-5 of index word 98, and the next instruction is taken from the location two greater than that of the LOOKUP LOWEST instruction. Index word 98 is given a plus sign. If at the end of the search a lower value was not found, the next instruction is taken from the next sequential location, and index word 98 is unchanged.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +66.

23 Indexing word for modifying the RDW address in positions 6-9.

45 Field definition. Defines the portion of each table word to be compared with the value in accumulator 3.

6789 Address in storage of the first or only record-definition word, which defines the location(s) of the table to be searched.

EXAMPLES: To locate the lowest value in a table defined by the record-definition word in location 2740, searching each table word in positions 5-9 only:

S01	23	45	6789
+66	00	59	2740

Contents of RDW 2740: −00 1100 1110; the lowest value on this table is +xxxxx 00206, in word 1108. Contents of accumulator 3 before the operation: +00000 50000. Contents after the operation: +00000 00206. Contents of index word 98 before the operation:+00 0048 0001. After the operation: +00 1108 0001.

To obtain the lowest value on a table, defined by the record-definition word in 1732, searching each table word entirely:

S01	23	45	6789
+66	00	09	1732

Contents of the RDW in 1732: −00 0900 0948; the lowest value on this table is +00001 23456 in word 0918. Contents of accumulator 3 before the operation: +99999 99999. Contents after the operation: +00001 23456. Contents of index word 98 before the operation: +00 0944 0002. After the operation: +00 0918 0002. Note that the increment value is 2. This means that only the even-numbered words 0900-0948 are searched, 25 words in total. The odd-numbered table words are ignored.

DATA FLOW: Data flow in a table lookup operation is best described by use of an example. The first example above, the instruction +66 00 59 2740, is used here. Assume that word 1108 is the lowest, but that 1101 is also lower than the search argument; its contents, +xxxxx 00345.

1. The address 2740 is brought to the RDW address register (if indexing had been called for, the address would be modified first, of course). The increment value in index word 98 (positions 6-9) is brought to positions 6-9 of the program register. Program register: +66 00 59 0001.

2. Under control of the RDW address register, the record-definition word is brought to the auxiliary register. Auxiliary register: −00 1100 1110. The address in the RDW address register is upped by one to 2741 (it would be needed if the sign of the RDW in 2740 were plus). Positions 2-5 are compared with positions 6-9, to assure that positions 2-5 are not greater in absolute value (if they are, an error is signalled and the machine stops).

3. The contents of the first table word (1100) are brought to the arithmetic register. Under field definition, positions 5-9 of the arithmetic register are compared with the contents of accumulator 3. The signs are compared. The two values are complement-added, and the high, equal, or low result depends on zero result and carry, as in the compare instructions. In this example, the contents of 1100 are not lower.

4. The address in positions 2-5 of the auxiliary register is increased in absolute value by the increment in positions 6-9 of the program register and returned to the auxiliary register. Auxiliary register: −00 1101 1110.

Positions 2-5 and 6-9 of the auxiliary register are compared; when 2-5 is greater, end-of-block has been reached. In this example, end-of-block has not yet been reached, and the process returns to paragraph 3 above, for the second table word (1101). In this example, the contents of 1101 *are* lower; the process continues in paragraph 5.

5. The field-defined portion of the arithmetic register is brought to accumulator 3, to the low-order portion if less than 10 digits are field-defined. In this example, the value +xxxxx 00345 in the arithmetic register goes to accumu-

lator 3 as +00000 00345. This number is now the search argument.

6. The contents of the auxiliary register are transmitted to accumulator 2. Accumulator 2: −00 1101 1110.

If 1101 turns out to contain the lowest table value, that address will be needed at completion of the operation. A special latch is turned on, indicating that a table value lower than the original search argument has been found. The operation returns to paragraph 4, to increment the table address and compare the contents of 1102 with the value +00000 00345.

7. The process continues, comparing positions 5-9 of each sequential table word with the value in accumulator 3. Because the contents of 1108 are lower than +00000 00345, its contents, +00000 00206, are reset-added to accumulator 3, and become the search argument for words 1109-1110. Accumulator 2 is changed to +00 1108 1110, as a result.

8. When the incremented positions (2-5) of the auxiliary register reach 1112, the test in paragraph 4 detects this. The sign of the auxiliary register is tested. If it is plus, the next sequential RDW, addressed by the RDW address register, is brought to the auxiliary register, and the operation proceeds as described in paragraph 2. In this example, the entire table is defined in one 11-word block, 1100-1110; the sign of the RDW is therefore minus.

9. The special latch described in paragraph 6 is tested to determine whether any table value lower than the original search argument had been found. If the latch is not set, the operation terminates, and the next instruction is taken from the location of this LL instruction, +1. In this example, the special latch was turned on by the low value in word 1101.

10. The contents of accumulator 2 are brought to the auxiliary register. Auxiliary register: +00 1108 1110.

11. Positions 0-1 and 6-9 of index word 98 are brought from core storage to the corresponding positions of the auxiliary register. Auxiliary register: +00 1108 0001.

The entire contents of the auxiliary register are transmitted to index word 98: IW 98: +00 1108 0001.

12. The next instruction is taken from the location of this LL instruction, +2.

REGISTERS AFFECTED: The auxiliary register, the adder, and the arithmetic register. If a table value is found that is lower than the original search argument in accumulator 3; index word 98, accumulator 2, accumulator 3, the special *found* latch, and the instruction counter.

TIMING: In microseconds:

24

+36 per record-definition word

+108 per table word

+60 for each lower value found

COMMENTS: The table data does not have to be in low-to-high sequence for this operation; the values can be completely random. This instruction is a means of sorting the values in a table and putting them in lowest-to-highest sequence in another section of core storage. The programming procedure is as follows: The beginning search argument is all 9's. After the lowest value in the table is found, it is filled with 9's (IW 98 can be used to index an address of 0000 for this). The lookup lowest operation is repeated, again starting with all 9's, this time obtaining the second-lowest value in the original table, etc.

If the signs of the table values are not significant in the operation, this procedure can be used: Make all of the table signs alpha, and the search argument +00000 00000. After the LL operation obtains each table value, change the sign of that value to plus, thus making it higher than the search argument.

If there are two lowest values on a table (equal to each other), the first one found is used.

The comparison between the search argument and the table arguments includes the signs of both factors. The same relation of values is used for these operations as for compare operations:

Highest	+99999 99999
	to
	+00000 00000
	−00000 00000
	to
	−99999 99999
	@99999 99999
	to
Lowest	@00000 00000

Plus is higher than minus, which is in turn higher than alpha. Note than a minus 9 is lower than minus 8, which is lower than minus 7, etc. This means that, if the table values are minus, the highest in absolute value is obtained, regardless

of the sign (+ or −) of the original search argument. (A plus search argument always obtains a *lowest* value in this case; a minus search argument obtains one only if it finds one greater in absolute value.) A minus search argument can never obtain a plus table value with this code; an alpha search argument can never obtain a plus or minus table value.

Note that the field-defined portion of each table word is compared with the entire contents of accumulator 3. If the number of significant digits in accumulator 3 exceeds the number of digits specified by field definition, all table values have the same relation to the search argument, as determined by sign (Figure 114):

Search Argument containing more significant digits:	than	Table Argument digits field-defined:
+		All + values are lower
+		All − values are lower
+		All Alpha values are lower
−		All + values are higher
−		All − values are higher
−		All Alpha values are lower
Alpha		All + values are higher
Alpha		All − values are higher
Alpha		All Alpha values are lower

FIGURE 114. RELATIVE FIELD LENGTHS, LOOKUP LOWEST

AUTOCODER EXAMPLE (Figure 115): Assume that JULY has been previously defined as location 3023. The assembled instruction is:

S01	23	45	6789
+66	00	04	3023

Lookup Equal Only
+67 LE

MACHINE DESCRIPTION: The search argument in accumulator 3 is compared with the field-defined portions of the storage words addressed by the RDW, the location of which is designated in positions 6-9 (indexable). The entire accumulator is used; a search argument of less than 10 digits must be in the low-order portion of the accumulator. The search continues through sequential or incremented locations, determined by the value in positions 6-9 of index word 98. The table can be in a single block of storage or several blocks, depending on the RDW sign(s).

The table is searched until a word is found that is equal to the search argument, in sign and value, or until the entire table is searched. If an equal value is found, its location is in positions 2-5 of index word 98 at completion of the operation. The next instruction is taken from the location of the LOOKUP EQUAL instruction, +2. The sign of index word 98 is set to plus. If no equal value is found, index word 98 is unchanged, and the next instruction is taken from the next sequential location.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +67.

23 Indexing word for modifying the RDW address in positions 6-9.

45 Field definition. Defines the portion of each table word to be compared with the value in accumulator 3.

6789 Address in storage of the first or only record-definition word, which defines the location of the table to be searched.

EXAMPLE: To search for equal, a table defined by the record-definition word in location 4315, searching each table word in the two high-order positions only:

S01	23	45	6789
+67	00	01	4315

Contents of RDW 4315: −00 2201 2225. Contents of accumulator 3: +00000 00043, unchanged by the operation. Assume that word 2209 contains this value also. Contents of index word 98 before the operation: −00 2221 0001; after the operation: +00 2209 0001. Note that its sign is set to plus.

DATA FLOW: This operation is similar to LOOKUP LOWEST, with the following exceptions:

The comparison between the search argument in accumulator 3 and each table value in the arithmetic register is for equal, rather than for a lower value in the arithmetic register. Detection of an equal comparison moves positions 0-1 and 6-9 of index word 98 to the corresponding positions of the auxiliary register (positions 2-5 of the auxiliary register already contain the address of the table word, just as in LOOKUP LOWEST). The con-

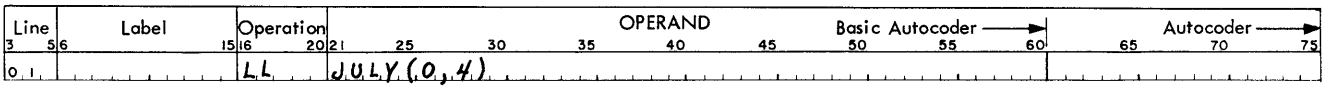


FIGURE 115.

tents of the auxiliary register are then transmitted to index word 98, as in the LOOKUP LOWEST operation.

REGISTERS AFFECTED: The auxiliary register, the adder, and the arithmetic register. If an equal value is found, the instruction counter.

TIMING: In microseconds:

24

+36 per record definition word

+108 per table word searched.

COMMENTS: Unlike LOOKUP LOWEST, this operation searches a table only until it finds an equal, rather than the entire table every time. The table data does not have to be in any particular sequence; the values can be completely random. If there are two equal values on a table, and both are equal to the search argument, the first value is used (the program does not "know" about the second one).

Signs are included in this operation. If they are unequal, the values are not considered equal.

AUTOCODER EXAMPLE (Figure 116): Assume that AUGUST is defined as location 1050. The assembled instruction is:

S01	23	45	6789
+67	64	09	1050

Lookup Equal or High
+68 LEH

MACHINE DESCRIPTION: The search argument in accumulator 3 is compared with the field-defined portions of the storage words addressed by the RDW, the location of which is designated in positions 6-9 (indexable). The entire accumulator is used; a search argument of less than 10 digits must be in the low-order portion of the accumulator. The search continues through sequential or incremented locations, determined by the value in positions 6-9 of index word 98. The table can be in a single block of storage or several blocks, depending on the RDW sign(s).

The table is searched until a word is found that is equal to, or greater than, the search argument, or until the entire table is searched. If an equal or higher value is found, its location is in positions 2-5 of index word 98 at completion of the operation. The next instruction is taken from the location of the LOOKUP EQUAL OR HIGH instruction, +2. The sign of index word 98 is set to plus. If no equal or higher value is found, index word 98 is unchanged, and the next instruction is taken from the next sequential location.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +68

23 Indexing word for modifying the RDW address in positions 6-9.

45 Field definition. Defines the portion of each table word to be compared with the value in accumulator 3.

6789 Address in storage of the first or only record-definition word, which defines the location of the table to be searched.

EXAMPLE: To search, for equal or higher, the table defined by the record-definition word in location 731, searching for four high-order positions only:

S01	23	45	6789
+68	00	03	0731

Contents of RDW 731: −00 2800 2899. Contents of accumulator 3: +00000 05250, unchanged by the operation. The table values are in ascending sequence; word 2856 is +5240 743529, and word 2857 is +5260 542123. Contents of index word 98 before the operation: +00 2888 0001. After the operation: +00 2857 0001.

DATA FLOW: Same as LOOKUP EQUAL ONLY, except that the comparisons between the value in accumulator 3 and the table values in the arithmetic register test for the arithmetic-register value being equal or higher, instead of for equal only. Otherwise, the operations are identical.

REGISTERS AFFECTED: The auxiliary register, the adder, and the arithmetic register. If an equal value is found, the instruction counter.

TIMING: In microseconds:

24

+36 per record definition word

+108 per table word searched

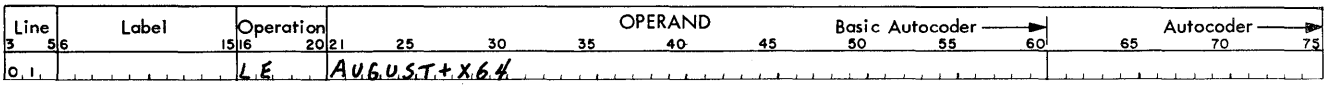


FIGURE 116.

COMMENTS: The table values should be in ascending sequence, if this operation is used.

The signs of both factors are considered, and the relative values are the same as for compare operations:

Highest	+99999 99999
	to
	+00000 00000
	−00000 00000
	to
	−99999 99999
	@99999 99999
	to
Lowest	@00000 00000

If the search argument and the table values are minus, the table values must be stored in *descending* order by absolute value for this code to be used. A plus search argument can never obtain a minus or alpha table value, and a minus search argument can never obtain an alpha table value with this code.

If the number of significant digits in accumulator 3 exceeds the number of digits specified by field definition, either the first table word is obtained by the LEH operation or none of them are, depending on the signs (Figure 117).

Search Argument containing more significant digits:	than	Table Argument digits field-defined:
+		No + values are obtained
+		No − values are obtained
+		No Alpha values are obtained
−		The first + value is obtained
−		The first − value is obtained
−		No Alpha value is obtained
Alpha		The first + value is obtained
Alpha		The first − value is obtained
Alpha		No Alpha value is obtained

FIGURE 117. RELATIVE FIELD LENGTHS LOOKUP EQUAL OR HIGH

AUTOCODER EXAMPLE (Figure 118): Assume that SEPT has been defined as location 1600. The assembled instruction is:

S01	23	45	6789
+68	00	05	1607

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60		65 70 75
0 1		L,EH	S,EPT,(,0,,5,),+,7		

FIGURE 118.

Magnetic Tape

Magnetic tape provides the IBM 7070 with high-speed, high-capacity storage of data. Because a reel of tape can be physically removed from one tape unit and brought to another, written by a different data processing system and used in the 7070, or written by the 7070 and read by another system, magnetic tape can be considered as a high-speed input/output feature as well as a means of storage.

A 7070 system can be served by as many as 40 tape units, IBM 729 II or IBM 729 IV (Figure 119). There are four data transmission channels, each of which connects up to ten tape units with the main system. Data read from tape to the system is read over a specified channel into magnetic-core storage, and a record is written from core storage over a specified channel to tape.

A single-channel 7070 system contains a maximum of ten tape units. (Throughout this section, tape is discussed in terms of a 40-unit, 4-channel system. A single-channel system uses only the instructions, special addresses, etc., that apply to channel 1, tape units 0-9.)

Successive records on tape are separated from each other by ¾ of an inch of blank space, called the inter-record gap. The size of a tape record has no limitation except the capacity of core storage that is read into or written from. Even that is not a rigid restriction in the case of tape reading if the entire tape record being read is not needed. The program defines the number of words of core storage to be read into, and any part of the tape record in excess of that is not accepted into storage.

Each tape-write operation writes one record, automatically creating ⅜ inch of space (inter-record gap) before the record and after it, thus providing the ¾ inch IRG between records.

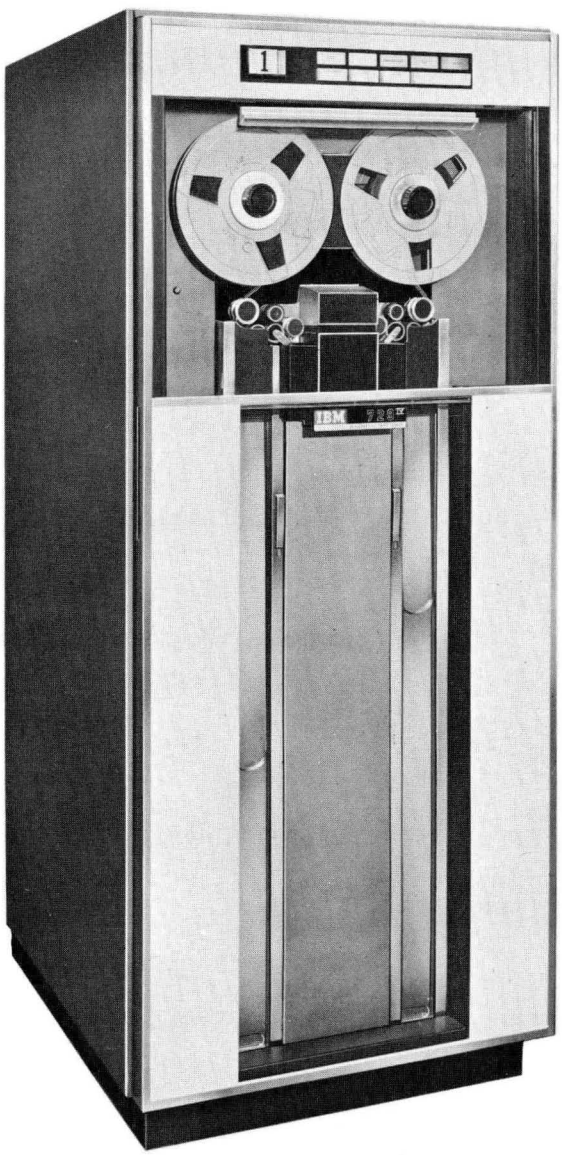


FIGURE 119. IBM 729 MAGNETIC TAPE UNIT

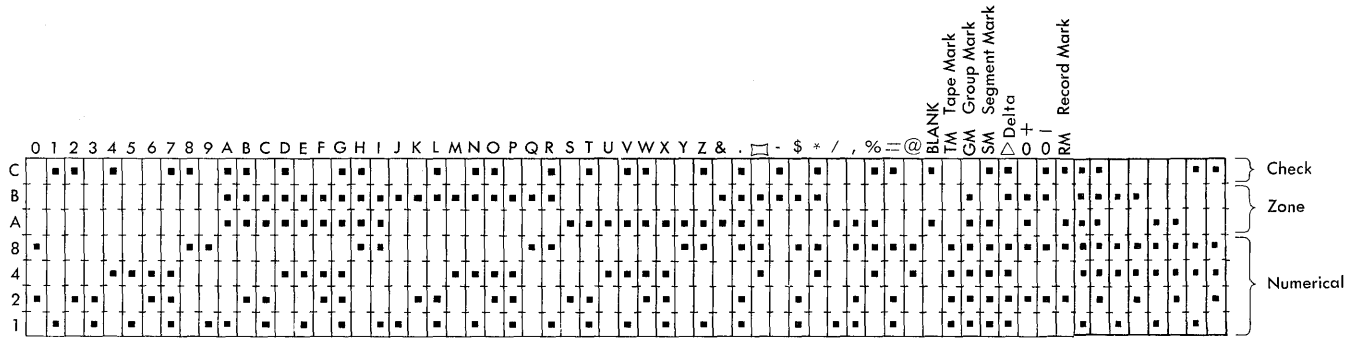


FIGURE 120. BCD 7-BIT CODE

BCD Code

The code structure used on tape is 7-bit alphamerical, called binary-coded decimal (BCD). Figure 120 shows the coding for each alpha, numerical, and special character that is acceptable to the 7070. There is a correlation between BCD and the IBM code used in punched cards: The AB bits are equivalent to 12-zones; the B bits, 11-zones, the A bits, 0-zones; and combinations of the 8421 bits comprise the digit values 0-9 (the 8 and 2 bits represent zero).

Validity Checking

The C bit is added to any character that would otherwise have an odd number of bits. In all reading and writing operations, each character is examined to establish that it is represented by an even number of bits. In the case of characters normally made up of an odd number of bits, the C bit makes the count even.

Also, there is a horizontal check of each record on tape. When a tape record is written, the bits in each channel (C, B, A, 8, 4, 2, 1) are tested for an odd or even total in the tape record; in each channel with an odd number, a bit is written at the end of the record. The check character thus created must itself have an even number of bits, for the over-all total number of bits to be even (Figure 121). Each tape-read operation tests the check character for an even number of bits.

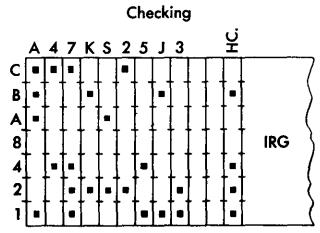


FIGURE 121. HORIZONTAL CHECK

IBM Magnetic Tape Units

Two models of IBM 729 Magnetic Tape Units, Model II and Model IV, are available in a 7070 system. There are no restrictions on number or configuration of models; 1-40 Model II's can be used, 1-40 Model IV's, or any combination adding up to 40. (Each channel can use 1-10 of either model, or any combination adding up to 10.) The difference between them is that the 729 IV has greater passing speed.

Each model can read or write tape at two densities of bit storage, 556 characters per inch and 200 characters per inch, and each tape unit in the system can be set at either high or lower density by a stored-program instruction. The densities affect character rate in read and write operations on the 729 II and 729 IV units as shown in Figure 122.

Each reel of tape (10½ inch diameter) is about 2400 feet in length. The capacity of a tape reel is determined by two factors: which character density is used, and how many individual records (how many IRG's) are contained.

Two-Gap Head

The read/write mechanism in an IBM 729 Magnetic Tape Unit is the two-gap head (Figure 123). Reading operations use the read head only. Writing operations, however, use both the read and write heads. The reading head reads each tape character after it has been written and checks it for validity. Thus, an invalid character is detected at the time it is written, instead of being first discovered at a later date, when the tape record is read.

Dual-Level Sensing

The ability of the two-gap head to read tape in both reading and writing operations affords another means of checking those operations — dual-level sensing. Bit impulses are sensed by the *read* head at two levels of pulse strength, high and low. Each character is read from tape into two 7-position registers, one position for each of the seven bit positions on tape. The *high* register requires a stronger impulse from a bit position than the *low* register does. This means that any bit impulse accepted by the *high* register is also accepted by the more sensitive *low* register, but a weak impulse is accepted by the *low* register only. The dual-level sensing feature functions in a slightly different manner for tape reading than for writing.

In a tape-read operation, after each character is read into the *high* register, it is checked for an even number of bits. If the number is not even, the contents of the *low* register are sent to the tape channel. If the *low* register has an invalid character, a validity-check signal is generated (condition code 1 — see section on *Automatic Priority Processing).* Figure 124 shows the various levels of signal strength that are acceptable and unacceptable. The solid lines indicate the levels of pulse strength acceptable to the high and low registers on a tape-read operation.

In a tape-write operation, the level of each register is increased, making them *harder to please.* As each tape character written is read back, it is checked for validity in the *high* register, and compared, bit by bit, with the contents of the *low* register. It must pass both of these tests to be acceptable; otherwise a validity-check signal is generated. The dotted lines in Figure 124 show the levels of pulse strength acceptable to the registers on a tape-write operation.

	MODEL II		MODEL IV	
Characters per inch (approx.)	556	200	556	200
× Inches per second (approx.)	75	75	112.5	112.5
= Characters per second (approx.)	41,700	15,000	62,550	22,500

FIGURE 122. COMPARISON CHART — IBM 729 MAGNETIC TAPE UNITS

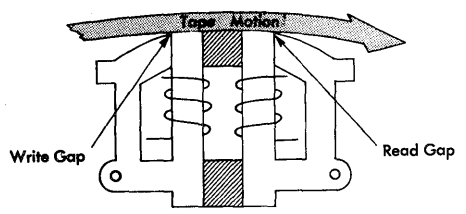
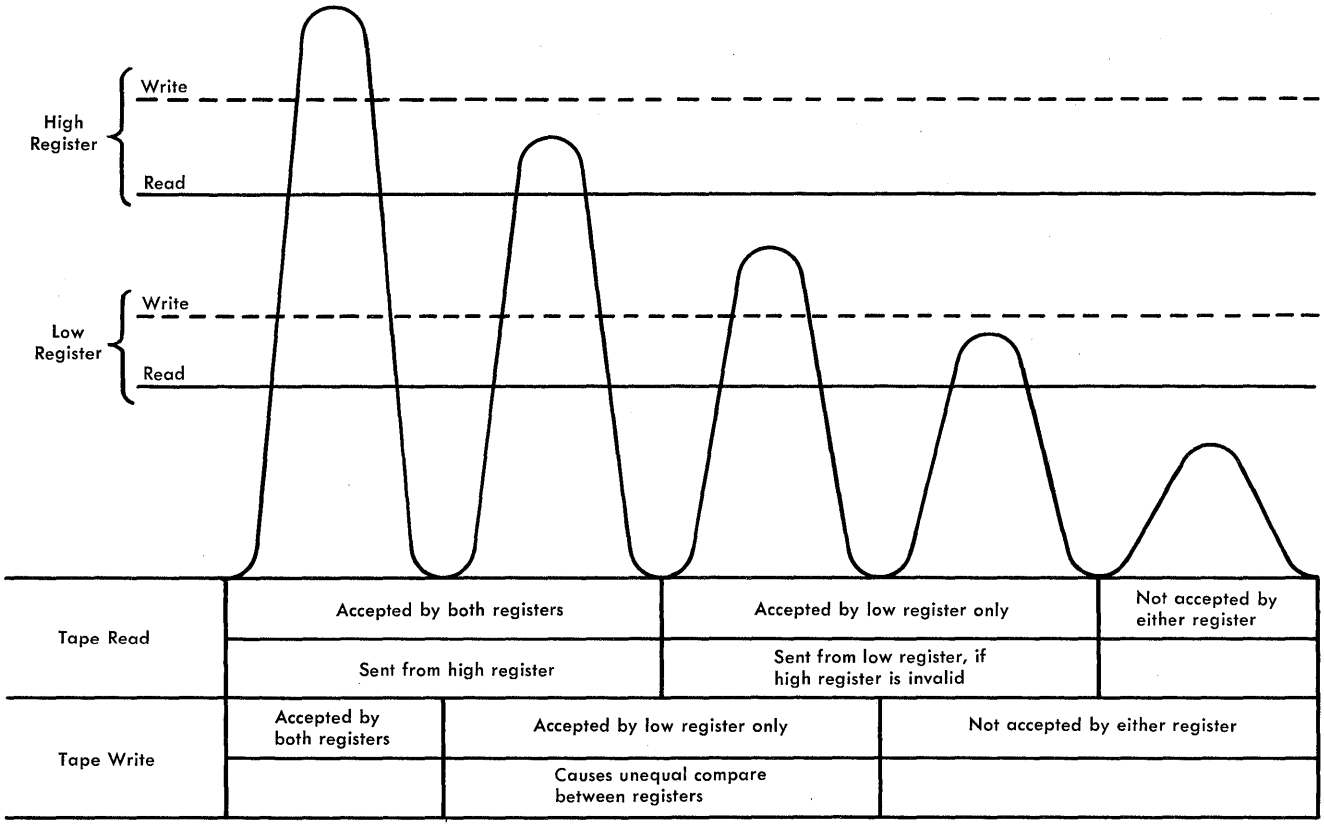


FIGURE 123. TWO-GAP READ/WRITE HEAD



Tape Read	Accepted by both registers	Accepted by low register only	Not accepted by either register
	Sent from high register	Sent from low register, if high register is invalid	
Tape Write	Accepted by both registers	Accepted by low register only	Not accepted by either register
		Causes unequal compare between registers	

FIGURE 124. DUAL-LEVEL SENSING

Operating Principles

Magnetic tape is a special plastic tape, coated on one side with a layer of magnetic oxide material. Data is recorded on the magnetic oxide of the tape in the form of magnetized spots or bits. Information written on tape remains there until the tape is used in a new write operation. This time period may extend indefinitely. When the recorded information is no longer needed, the tape may be used to record new data. Only when a tape is written is the previous information destroyed. The write operation automatically erases old information. Reflective spots manually placed on the tape are photoelectrically sensed to indicate the beginning (or load point) and the physical end of the useful portion of the tape. The load-point reflective spot is about 12 feet from the front end of the tape, and the reflective spot designating the end of the usable tape is 18 feet from the physical end. Tape is wound on plastic reels 10½ inches in diameter, and weighs about 12 ounces. A full reel contains about 2400 feet of usable tape, but lengths as short as 50 feet can be used.

Figure 125 shows schematically the physical location of the tape when it is mounted on a tape unit. During reading or writing, tape is moved from the file reel

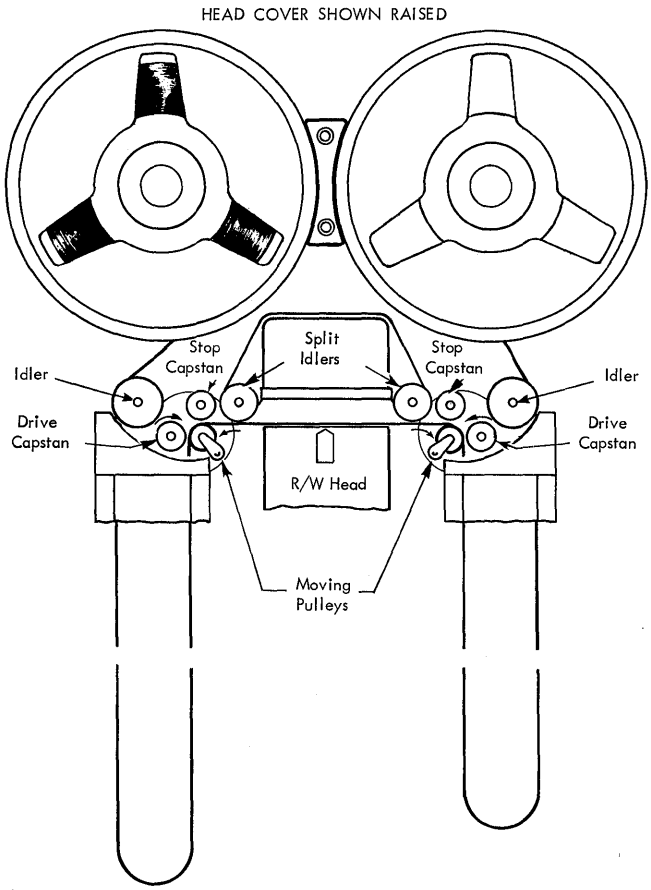


FIGURE 125. TAPE FEED SCHEMATIC

through the left vacuum column across the read-write head, through the right vacuum column, and to the machine reel.

Reading or writing on a tape takes place as the tape is moved across the read-write head. The vacuum columns control separate drive motors, which permit the read/write mechanism and each of the two tape reels to move tape independently of the other two units. The read/write mechanism feeds tape under instruction from the stored program. The file reel feeds tape when the tape reaches a minimum slack point in the vacuum column, and the machine reel winds tape when the slack tape reaches a point near the bottom of the vacuum column.

The head assembly, located between the vacuum columns, is built in two sections. The lower section is stationary, and the upper section can be moved up or down under control of the tape-unit keys. When the upper section is up, it allows the operator to thread tape. When down, it causes the read-write head to be in close contact with the tape for reading or writing. The tape reels and head are accessible by opening the reel door.

Reflective Spots

Reflective spots, also referred to as photo-sensing markers, are placed on the tape to enable the tape unit to sense where reading and writing are to begin and to stop. The markers are small pieces of plastic, one inch by 3/16 inch, coated with vaporized aluminum on one side and with adhesive on the other. They are fastened to the base (uncoated) side of the tape. The photo-electric cells sense them as either the load point marker where reading or writing is to begin on tape, or as the end-of-reel marker where reading or writing is to stop.

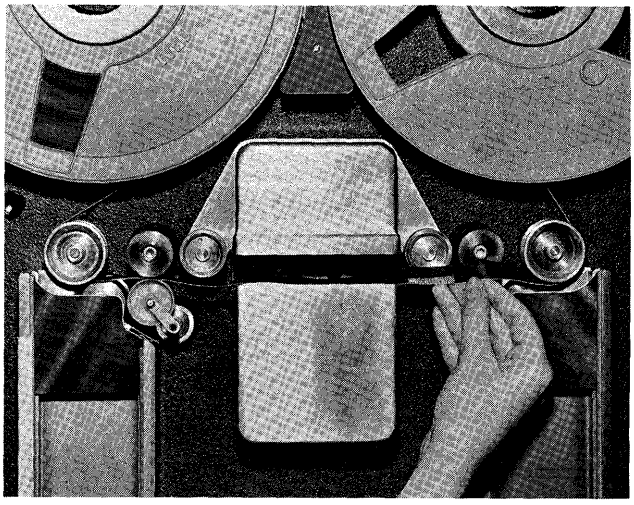
About 12 feet of tape must be allowed between the beginning of the reel and the load point marker as a leader for threading the tape over the feed rolls and the read-write head. Information must not be stored in this space. To indicate the load point, the one-inch dimension of the marker must be parallel to, but not more than 1/32 inch from, the channel 1 edge of the tape—the edge nearest the operator when the reel is mounted (Figure 126).

About 18 feet of tape should be reserved between the end-of-reel marker and the physical end of the tape attached to the hub of the machine reel. To indicate end of reel, the marker must be placed parallel to, but no more than 1/32 inch from, the C track edge of the tape (the edge nearest the tape unit when the reel is mounted; Figure 126).

Place load point and end-of-reel markers on tape with care. They should be properly aligned and pressed tightly onto the tape with the back of the fingernail. It is best to do this while the tape is loaded on a unit, to reduce the collection of dust on the unrolled tape. If this is done away from the unit, keep the unrolled end of tape off the floor and away from dusty areas.

File-Protection Ring

The back of the tape reel (machine side) has a circular groove in which a plastic ring may be inserted. To enable the machine to write on a tape, a plastic ring called the file protection ring, must be placed in the groove of the tape reel (Figure 127). A tape can be read whether the ring is inserted, or not. The file protection ring should be removed from the tape reel after writing on tape is completed. Doing this prevents accidental writ-



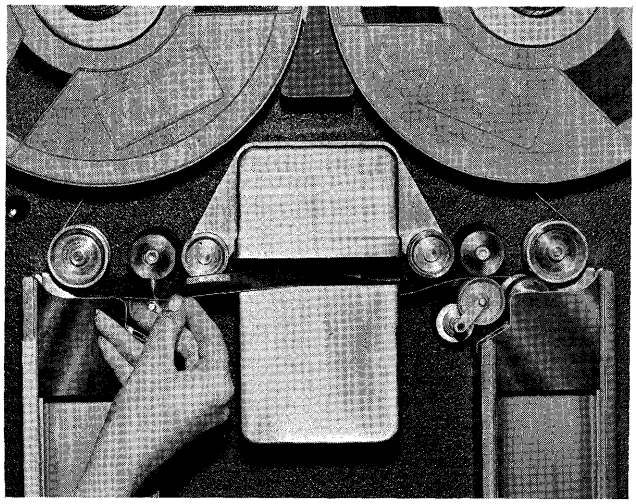


FIGURE 126. REFLECTIVE SPOTS ON TAPE

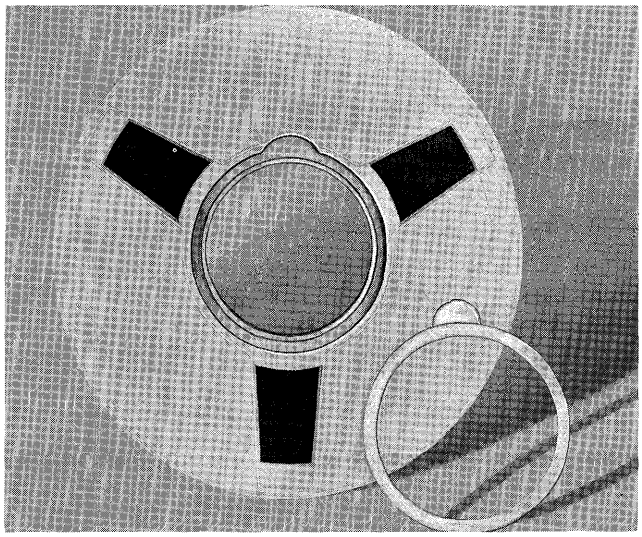


FIGURE 127. FILE-PROTECTION RING

ing and resultant loss of valuable tape records. Never remove the file protection ring while tape is loaded in the vacuum columns of the tape unit. Doing this could cause a broken or damaged tape.

Operating Keys and Lights

The operating keys and lights of the IBM 729 Tape Units are located at the top of the unit, above the tape reels (Figure 128). The lights are all on the upper row, and the keys are on the lower row. The address selection dial is at the left.

ADDRESS SELECTION DIAL: This dial assigns a number from 0 to 9 to the tape unit, to identify it to the stored program. If blank is set, the tape unit cannot be used by the stored program.

The setting should not be changed when a tape operation is in progress.

SELECT LIGHT: The select light is turned on automatically when the address selection dial is properly positioned and the unit is addressed by the computer, provided that the unit is ready.

READY LIGHT: This light, when ON, indicates that the tape unit is ready for operation. See *Start Key* for method of turning this light on. The reel door should never be opened when the ready light is ON.

TAPE INDICATE ON LIGHT: This indicator is turned on by any of the following conditions:

1. Sensing the end-of-reel marker while writing on tape.

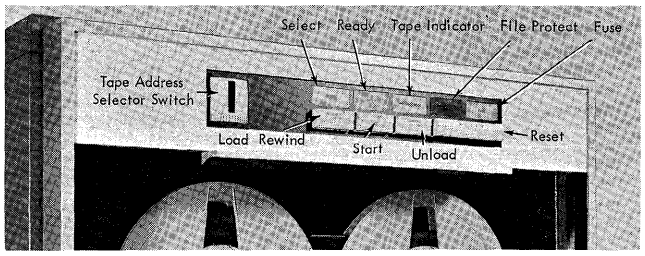


FIGURE 128. IBM 729 TAPE UNIT OPERATING KEYS AND LIGHTS

2. Sensing the tape mark while reading tape.

The indicator may be turned off by:

1. Depressing the unload key on the tape unit.
2. Executing a rewind-unload or tape end-of-file-off instruction in the stored program.

FILE PROTECTION LIGHT: This light automatically turns on if the unit is loaded with a reel that does not have the file protection ring inserted in the back of the reel. The tape cannot be written as long as the file protection light is ON. This light is ON whenever the tape unit is not in ready status.

FUSE LIGHT: This light turns on automatically whenever a fuse in the unit has blown.

LOAD REWIND KEY: This key is operative only when the reel door is closed and the ready light is OFF. Use of this key after tape has been properly mounted in the magnetic tape unit lowers tape into the columns, lowers the head assembly, and moves tape in the rewind direction until the load point reflective spot is sensed. If the reflective spot is not to the right of the read-write head when this key is pressed, the tape will unwind from the machine reel.

Caution: Do not open the reel door during rewind or load point searching.

START KEY: Use of this key places the tape unit in ready status and turns on the ready light provided that:

1. The reel door is closed.
2. Tape has been loaded into the columns.
3. The tape unit is not in the process of finding the load point (rewind or load point operation).

UNLOAD KEY: This key is operative only when the ready light is OFF, tape is in the vacuum columns, and the reel door is closed. Use of this key raises

the head assembly and removes the tape from the columns, regardless of the distribution of tape on the two reels. If the tape is not at load point when the operator wishes to change tape reels, a load point search should be initiated first by pressing the load-rewind key. Pressing the unload key will also turn off the tape-indicate-on light, if ON.

RESET KEY: Use of this key turns off the ready light if it is ON. If the reset key is used during high-speed rewind, the operation stops and then continues as a slow-speed rewind. If the reset key is used during a slow-speed rewind, the rewind stops.

REEL DOOR INTERLOCK: When the door is open, the interlock contact prevents any normal operation of the tape unit. The reel door should never be opened when the ready light is ON or during any load rewind operation.

REEL RELEASE KEY: When this key is depressed, both reels may be turned manually for threading tape or removing the file reel. To operate the reel release key, open the reel door.

Operating Pointers

Consider the following points whenever a tape unit is in operation:

1. Do not change the address of a tape unit by turning the address selection dial, during the execution of a program that uses other tape units. This applies whether the unit is in ready status or not.
2. Never set two tape units to the same address.
3. Do not open the door of a tape unit unless the tape inside is out of the vacuum columns and the read-write head is raised.
4. In the event of a power failure with tape units in ready status, have an IBM customer engineer remove the tape from the read-write head and the vacuum columns, of every unit in ready status, before power is restored.
5. Do not turn dc off on the 7070 with the tape units in ready status, as extraneous noise may be recorded on the tapes when dc is turned on.

Features of IBM 7070 Tape Operations

There are a number of features of tape operations in the 7070 that add considerably to the efficiency of the system. One of these is high-speed tape units, which were described previously. Other features add further to the effective speed of reading and writing tape records, the efficiency of arranging data read from tape into core storage, or written on tape from storage, and the ease of programming tape operations.

Simultaneous Operations

The IBM 7070 can perform four tape operations simultaneously: reading four tapes, writing four tapes, or any combination of these. Also, the stored program can continue while these operations are taking place. Tape units operating simultaneously must be connected to the

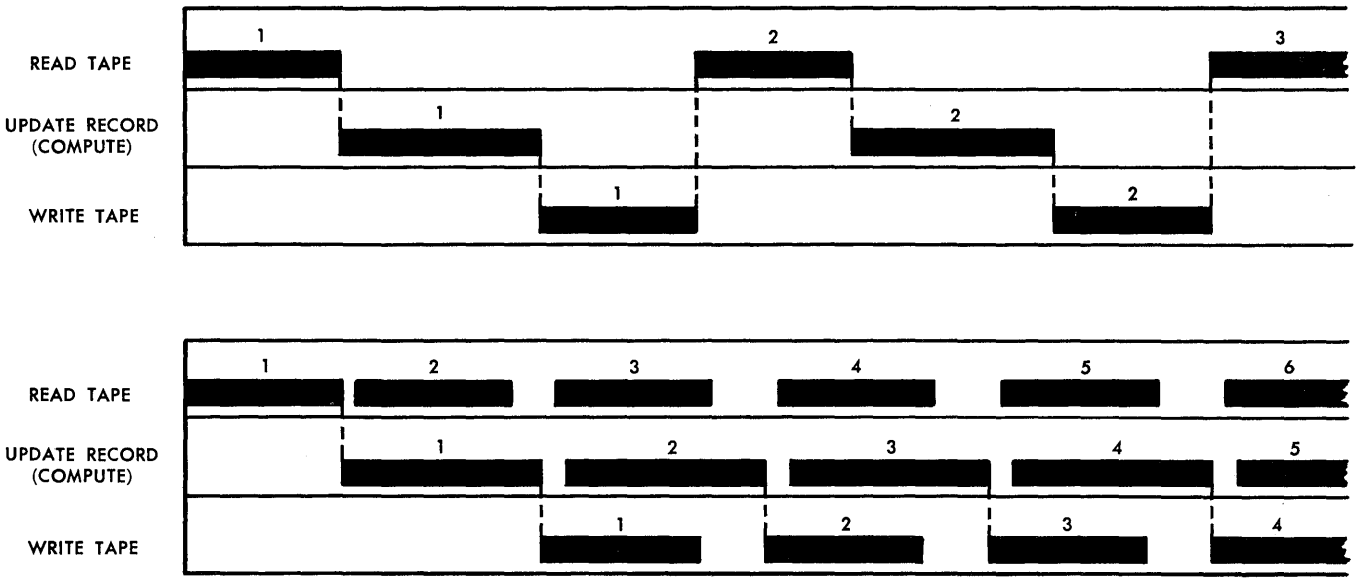


FIGURE 129. SIMULTANEOUS OPERATIONS

system via different channels. The simultaneous-operation feature is made possible by the ability of the four tape channels to work independently of the programming unit and of each other.

An example of the timesaving value of simultaneous operations is a file maintenance routine, which requires the reading of a tape record updating it, and writing it on another tape. Figure 129 shows the overlap of the operations made possible by the simultaneous operations feature. The timing of a sequential operation is shown for comparison.

Alphamerical and Numerical Modes

Because every word in core storage may be numerical or alphamerical, sign designation must be recorded when data is written on tape. Each word is written and read in either alphamerical or numerical mode. An alpha word written on tape consists of five characters. A numerical word takes one tape position for each digit, and appears on tape as it is in storage, except that the plus or minus sign is recorded as a sign-over-units zone. If the sign is plus, the equivalent of a 12-zone in IBM punched-card coding (the A and B bits) is combined with the low-order digit. If it is minus, an 11-zone (the B bit) is combined. Tape reading in the alpha mode automatically translates every five characters into a 10-digit word with an alphabetic sign. Numerical words are moved, without change, except for the units positions, the zone coding of which is interpreted to give plus or minus designation to the word in core storage. Figure 130 shows examples of the numerical and alpha modes.

Tape reading automatically starts in the alpha mode; the mode is changed to numerical by detection of the mode-change character, sometimes called the *delta* (Δ), on the tape. Detection of this character changes the mode from alpha to numerical or vice versa; the character signifies *change the mode.*

In tape writing, the sign of each core-storage word determines the mode. Every time the mode is changed from alpha to numerical or vice versa, the delta is automatically recorded on tape. (Tape writing automatically starts in the alpha mode.) The delta appears only on tape; it never enters core storage from tape. In BCD code, a Δ is CB8421.

Here is an example of a 3-word record written on tape. The three words are:

+0123301234
−5678856789
@7461799298

They are written on tape as:

Δ012330123D567885678RΔMAR28

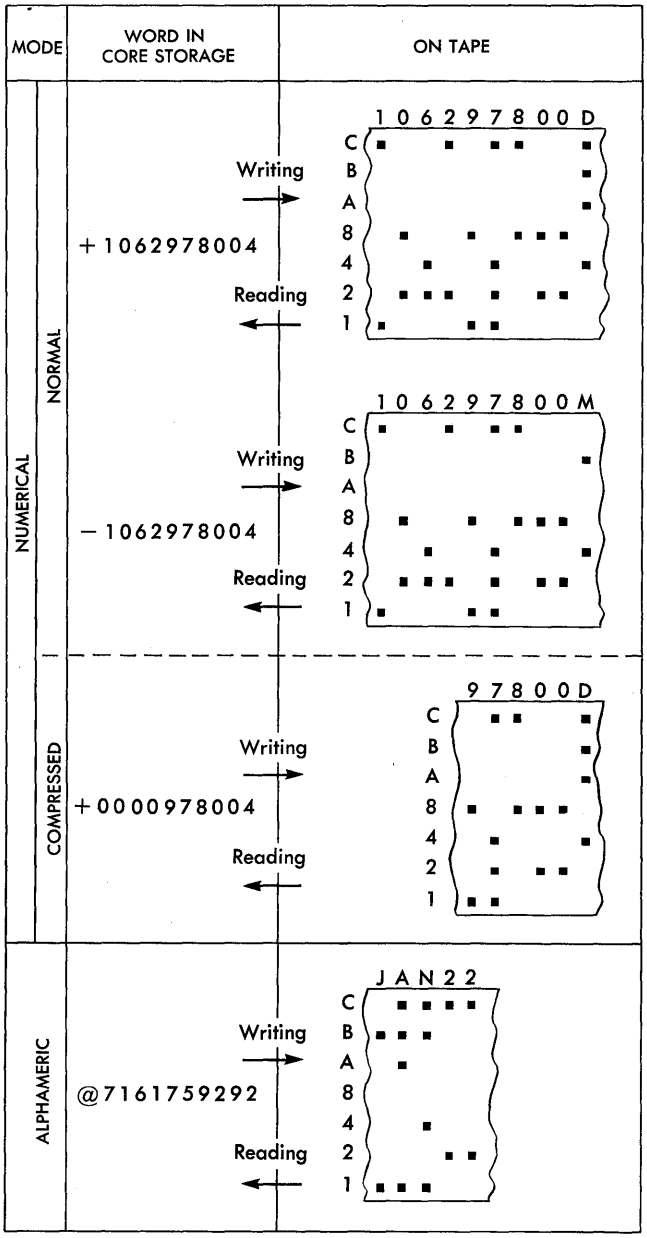


FIGURE 130. NUMERICAL AND ALPHAMERIC MODES

Because the first word is numerical, the first character on tape is the delta. It denotes a change in mode from alpha to numerical. The second Δ on the tape is created by the alpha sign in third word. The letter D is created by the units-digit 4 and the plus sign. The R is created by the digit 9 and the minus sign.

Scatter Read/Write

A powerful programming tool in IBM 7070 operations is scatter read/write. A single record read from tape can be divided into as many parts as desired by the programmer, and distributed into different blocks of core storage. This is done in only one program step, the

same instruction that initiates the tape-read operation. The blocks of storage are defined by record-definition words.

This feature can be used in writing tape as well as reading. The program can gather data from the core-storage blocks and automatically assemble it into one tape record. Figure 131 shows an inventory record separated by field and category as it is read from tape, or conversely, gathered from blocks of core storage and written as a single tape record.

A *grouped record* on tape is a series of records in storage, written on the tape as a single record. The scatter read-write feature can be used to read and write grouped records, while each individual record has its own core-storage section.

The scatter read/write feature can also be used for bringing data to and from disk storage, the unit-record input/output synchronizers, the inquiry control synchronizers, to the console typewriter, and for transmission of data from core storage to different parts of core storage.

Record-Mark Control

Record-mark words give additional flexibility to the scatter read/write feature. In many cases, the data read from tape into a core-storage block may be variable in length; a name and address field, for example. The block of storage reserved for a variable-length field must, of course, be large enough to accommodate the largest size that the field on tape can be — the longest name and address. To make all of the name and address fields in the tape the same size as the largest (by adding blanks to fill up to the allotted size) is impractical, and a waste of tape capacity and processing time. Record marks make this unnecessary.

A record-mark word on tape is an alpha word consisting of a special character (‡, CA82) in the low-order position, and any other four characters. Detection of this word automatically denotes the end of a block in a scatter read/write operation. In scatter read, a tape record being read into a block of storage does not continue to fill that block after a record-mark word has been read in. Instead, it goes on to the next block

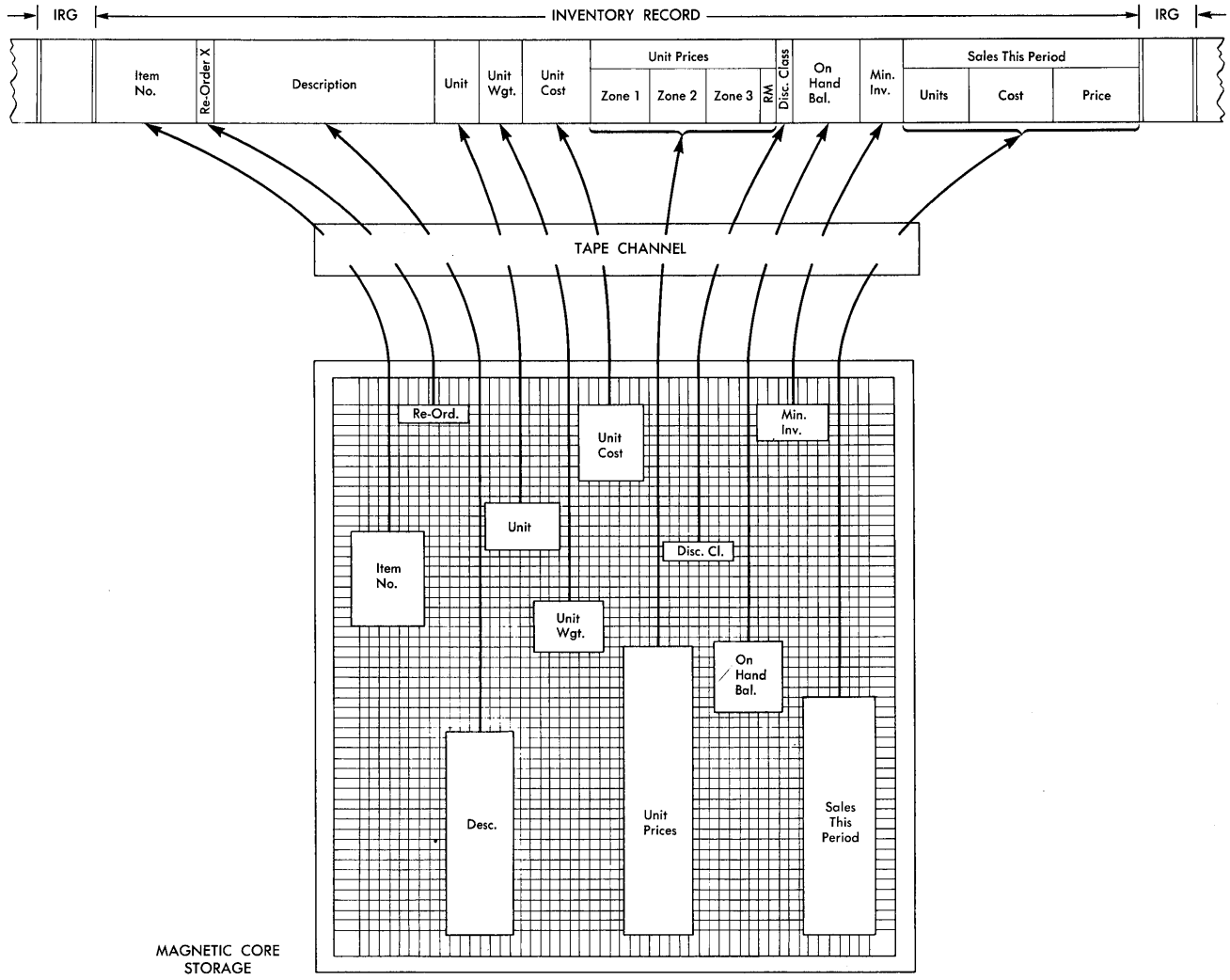


FIGURE 131. SCATTER READ-WRITE

and starts filling it with the characters on the tape that immediately follow the record-mark word. Detection of a record mark has the same effect as the working and stop addresses in the record-definition register being equal. (See section on *Block Transmission.)*

In scatter tape writing, a record-mark word in core storage signals to stop moving data from the block and start sending from the next block. The tape read or write instruction determines whether record-mark words will be operative. If not, a record-mark word is treated as a normal alphabetic word.

Zero Elimination

When a numerical word in storage is written on tape, as many as five high-order zeros can be eliminated (see Figure 130). When a tape record is read in the numerical mode, each core-storage word read into is automatically filled with zeros (up to 5) in the high-order positions with no lost time. The units digit of each word is identified by a sign-over-units zone. Zero elimination is not operative in reading or writing tape in the alphabetic mode.

Organization of Data on Tape

The characters stored on a reel of magnetic tape are arranged and grouped by several different categories. These categories are described here, starting with the smallest unit record — the alphamerical character — and building up to a complete tape reel or group of reels.

1. The alphamerical *character* consists of an even number of bits out of a possible seven.
2. A *word* on tape consists of 10 or less characters: five characters if it is alpha, 5-10 if it is numerical.
3. A tape *record* consists of any number of words. Adjacent records on tape are separated from each other by a blank space, called the Inter-Record Gap (IRG). Every read or write operation reads or writes one tape record.
4. A tape *segment* usually contains a number of records. Segments are defined as the records contained between *segment marks.* A segment mark is written only by the Tape Segment Mark Write (TSM) instruction. Each segment mark is a one-character record comprised of bits CA8421.
5. The data on a tape *reel* is between the load point, at the beginning, and a tape mark, at the end. When a tape mark is read, an EOF signal is given (see section on *Automatic Priority Processing).*
6. A *file* of tape data is all of the records in a group, or category — a customer list for accounts receivable, for example, or an employee list for payroll. A file may be part of a tape reel or several reels, in length.

Figure 132 shows the relationship of these categories to each other. Note that the reflective spot was detected before a complete segment was written on the tape reel. The tape mark is written, and the last record in that segment will be written as the first record of a second tape reel.

Tape Operation Codes

These codes govern all of the operations of the tape units: reading, writing, positioning, rewinding, etc. In all cases, the instruction initiates the operation, and the stored program can continue while the tape unit is operating. When the tape unit completes the operation, the main program can be signalled for priority.

MACHINE DESCRIPTION: All the operations that have anything to do with the magnetic tape units are incorporated in four augmented codes, ±81, ±82, ±83 and ±84. With each code, the sign determines whether there will be a normal-condition priority signal when the tape operation is completed; plus means that there will, minus means that there will not. (Unusual-condition priority

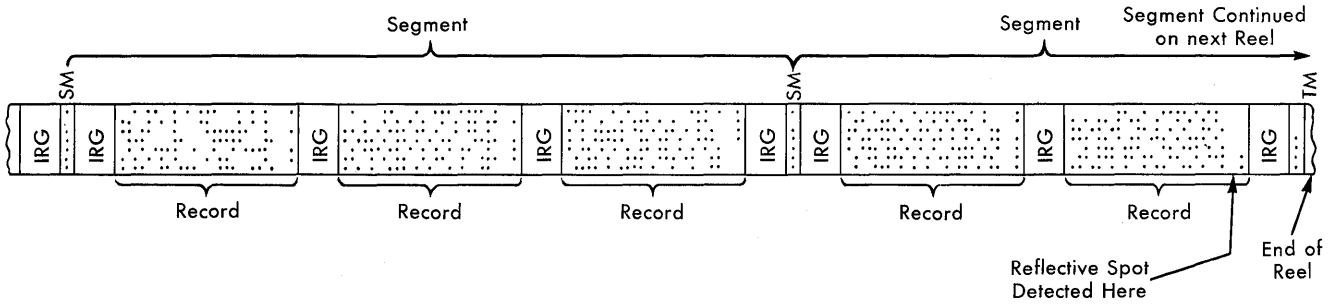


FIGURE 132. ORGANIZATION OF DATA ON TAPE

occurs regardless of the sign. See section on *Automatic Priority Processing.)* The only difference between the operation codes is the channel involved; 81 is used for tape units connected to the system via channel control 1, 82 is used for channel control 2, 83 for channel control 3, and 84 for channel control 4.

The operation itself is defined in position 5 of the instruction.

INSTRUCTION FORMAT: S 0 1 23 4 5 6789

S — Plus means that there will be a priority signal at the end of the tape operation, if it is a read, write or tape-spacing operation. Minus means that there will not be a priority signal, unless there is an unusual condition. (See section on *Automatic Priority Processing* for full descriptions of these conditions.) For those tape instructions that do not involve reading, writing, or spacing, the sign of the instruction has no effect.

0 — Always 8, identifying the instruction as an operation involving magnetic tape.

1 — Always 1, 2, 3, or 4 identifying the tape channel. This position and position 4 determine which of the 40 tape units is involved.

23 — Indexing word. Positions 6-9 can *in all cases* be modified by positions 2-5 of the indexing word specified here.

4 — Specifies the tape unit of the ten on the channel designated in position 1. The tape units are numbered 0-9.

5 — Digits 1-8 in this position specify the tape operation to be performed. Digit 9 is not used. Digit 0 indicates that the operation is defined by the digit in position 9 of the instruction.

The digit in position 5 defines the operation as follows:

1. Tape Read (P)TR
2. Tape Read per Record Mark Control (P)TRR
3. Tape Write (P)TW
4. Tape Write per Record Mark Control (P)TWR
5. Tape Write with Zero Elimination (P)TWZ
6. Tape Write per Record Mark Control and with Zero Elimination Combined (P)TWC
7. Tape Segment Forward Space per Count (P)TSF
8. Tape Segment Backspace per Count (P)TSB
9. Not used
0. Look at position 9 to further define the operation.

If the letter P is used in the mnemonic, the *Autocoder* assembly program gives the instruction a plus sign; if it is not used, the instruction gets a minus sign.

6789 — The function of these positions depends on the digit in position 5:

If position 5 contains digit 1-6, positions 6-9 contain the address of the first record-definition word, denoting the first group of core-storage locations to be read into or written from.

If there is a 7 or 8 in position 5, positions 6-9 contain the address of a storage word that defines the number of segments to be spaced over. The difference between the values of positions 2-5 and positions 6-9 of the specified word, denotes the number of segments to be spaced over. (Normally positions 2-5 are 0000, and positions 6-9 represent the number of segments.)

If there is an 0 in position 5, positions 6-9 do not represent an address; instead, position 9 contains the operation code, and positions 6-8 are not used. These operations do not require record-definition words. The digit in position 9 defines the operation as follows:

0 Tape Select TSEL
1 Tape Mark Write (P)TM
2 Tape Rewind TRW
3 Tape Rewind Unload TRU
4 Tape Backspace TRB
5 Tape Write Segment Mark (P)TSM
6 Tape Skip TSK
7 Tape End of File Off TEF
8 Tape Set Lower Density TSLD
9 Tape Set High Density TSHD

The digits in 6-8 can be anything; they are ignored if there is a 0 in position 5.

Operations

The following is a detailed description of the tape instructions. Discussed first will be the operations denoted by a digit 1-8 in position 5 of the instruction. All of these codes utilize positions 6-9 as the address of a core-storage word (indexable as per positions 2-3). All of these instructions can be used with priority (plus sign in the operation code).

Tape Read
1. **(P)TR**

This code specifies that the tape unit addressed by the digits in positions 1 and 4 of the instruction is to read a tape record. The record-definition word addressed by positions 6-9 (indexable) defines the core-storage area to be read into.

If the first character in the tape record is not a Δ, it is translated into the 2-digit alphamerical code, and placed in the two high-order positions of the first word in the RDW-defined area. Each succeeding tape character is similarly translated, with each group of five filling a storage word in high-order to low-order sequence. The storage-word addresses are serially advanced under control of the record-definition register.

When the first Δ on the tape is detected (it could be the first character in the tape record), the operation changes to the numerical mode. Each digit on the tape is translated to a single digit in the core-storage word. Each time a Δ is detected, the mode changes.

In the numerical mode, an alpha character on the tape automatically denotes the units position of the core-storage word to be read into. The bit coding determines the sign of the word: the A and B bits combined denote plus, the B bit alone denotes minus. The number of numerical characters to go into a storage word cannot be less than 5 nor, of course, more than 10 (not less than 4 digits and the sign-over-units alpha character, nor more than 9 digits and the alpha character). In the case of less than 10 digits, the high-order positions of the storage word are filled with zeros.

The high-order zeros are added to the word as the tape data enters transmission register A (see Figure 104). Just before the first digit enters it, the five *low-order* positions of transmission register A are filled with zeros. Each digit from tape is brought to the units position of the register, moving all of the other digits one position to the left. When an alpha character is read, its digit goes into the units position of the register, its zone coding is translated to plus or minus, and the contents of transmission register A are sent in parallel to transmission register B and thence to storage. Transmission register A is again reset to 00000 in the low-order positions, and the operation proceeds for the next digits from the tape.

EXAMPLE: To read a record from tape unit 3 on channel 2 into the core-storage area for which word 1715 is the first RDW, and to initiate a normal-priority signal at completion:

S01	23	4	5	6789
+82	00	3	1	1715

AUTOCODER EXAMPLE (Figure 133): Assume that JOE has been defined as word 2310. The assembled instruction is:

S01	23	4	5	6789
−81	00	0	1	2310

Tape Read per Record Mark Control
2. **(P)TRR**

This code is the same as tape read, with this addition:

The detection of a record mark (CA82) in the low-order position of an alpha word on tape causes a new RDW to be read into the record-defintion register, if the present one is plus; or terminates the movement of data from tape, if the present RDW is minus. It has the same effect as the working and stop addresses in the record-definition register being equal.

EXAMPLE: To read, per record-mark control, a record from tape unit 4 on channel 3 to the storage area for which word 1234 is the first RDW, with no normal-priority signal at completion:

S01	23	4	5	6789
−83	00	4	2	1234

AUTOCODER EXAMPLE (Figure 134): JIM is word 2341. The assembled instruction is:

S01	23	4	5	6789
+82	37	5	2	2341

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5 6	15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		TR	1.0, J.O.E.		

FIGURE 133.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5 6	15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		P.T.R.R	2.5, J.I.M+X.3.7.		

FIGURE 134.

Tape Write
3. (P)TW

The tape unit addressed by the digits in positions 1 and 4 is instructed to write a tape record. The record-definition word addressed by positions 6-9 (indexable) defines the core-storage area to be read from. If the sign of the first core-storage word is alpha, its contents are translated from two-digit alpha coding to the 7-bit BCD coding, and the 5 characters are written on tape, in high-order to low-order sequence.

If the sign of the first core-storage word is plus or minus, a Δ is written as the first character in the tape record. The sign of each succeeding word is tested; each time it is alpha when the preceding one was numerical (+ or −), or vice-versa, the Δ is automatically recorded on tape. The storage-word addresses are serially advanced under control of the record-definition register.

EXAMPLE: To write a record on tape unit 0 in channel 1 from the core-storage area for which 3321 is the first RDW, and to signal for normal-condition priority at the completion of the operation:

S01	23	4	5	6789
+81	00	0	3	3321

AUTOCODER EXAMPLE (Figure 135): JACK's address is 4341. The assembled instruction is:

S01	23	4	5	6789
+81	00	1	3	4341

Tape Write per Record Mark Control
4. (P)TWR

This code is the same as TAPE WRITE, with this addition:

The detection of a record mark (80) in the two low-order positions of an alpha storage word causes a new RDW to be read into the record-definition register, if the present one is plus; or terminates the tape-writing operation, if the present RDW is minus. It has the same effect as the working and stop addresses in the record-definition register being equal.

EXAMPLE: To write, per record-mark control, a record on tape unit 5 on channel 2, from the core-storage area for which 3635 is the first RDW, with no normal-condition priority signal:

S01	23	4	5	6789
−82	00	5	4	3635

AUTOCODER EXAMPLE (Figure 136): TOM is word 1517. The assembled instruction is:

S01	23	4	5	6789
+82	00	2	4	1567

Tape Write with Zero Elimination
5. (P)TWZ

This code is the same as TAPE WRITE, with this addition:

High-order insignificant zeros in numerical words (+ or − sign), up to a maximum of 5 per word, are not written on the tape.

NOTE: Tape *reading* automatically fills in high-order zeros for numerical words on tape; tape *writing* eliminates high-order zeros only if the instruction calls for it.

EXAMPLE: To write, with zero elimination, a record on tape unit 4 on channel 1, from the core-storage area defined by the RDW in: 1400 + positions 2-5 of IW 31. (No normal-condition priority signal is to be given):

S01	23	4	5	6789
−81	31	4	5	1400

AUTOCODER EXAMPLE (Figure 137): ALEX is 0440. The assembled instruction is:

S01	23	4	5	6789
+82	00	1	5	0440

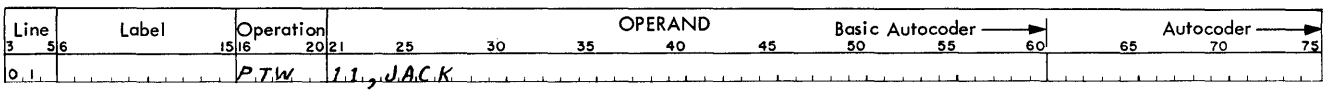


FIGURE 135.

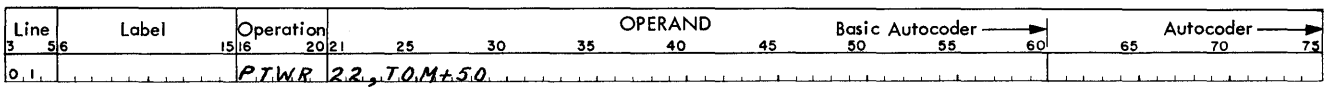


FIGURE 136.

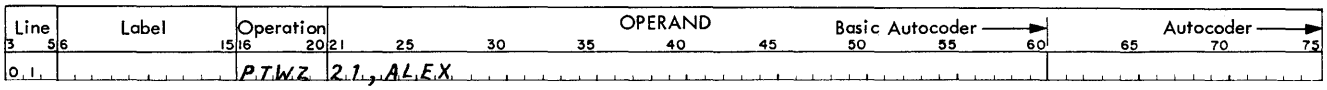


FIGURE 137.

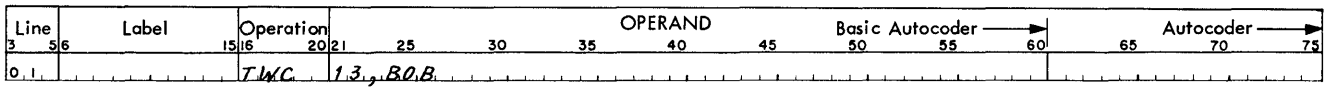


FIGURE 138.

Tape Write with Zero Elimination and per Record Mark Control Combined

6. (P)TWC

This code is the same as tape write, with the addition of both the record-mark control feature described above under code 4, and the zero elimination described above under code 5. Thus, maximum efficiency of tape-writing time and output-tape capacity is achieved by this code.

EXAMPLE: To write, with zero elimination, per record-mark control, a record on tape unit 1 in channel 4, from the core-storage area for which 1311 is the first RDW, and to signal for normal-condition priority at completion:

S01	23	4	5	6789
+84	00	1	6	1311

AUTOCODER EXAMPLE (Figure 138): BOB is word 1066. The assembled instruction is:

S01	23	4	5	6789
−81	00	3	6	1066

Timing of Tape Read and Write Operations

The speed of a tape read or write operation depends, of course, on whether a 729 II or a 729 IV unit is used, and on which density the specified tape unit is set to operate:

	Density	Character Rate	Milliseconds per Character
729 II	High	41,667 per second	.024
	Low	15,000 per second	.067
729 IV	High	62,500 per second	.016
	Low	22,500 per second	.044

An important factor in deriving operation times for tape operations is the time it takes for the tape unit to get to full speed at the start of the operation, and to complete the movement of tape after the operation is completed. In milliseconds, these times are:

729 II 10.8
729 IV 7.3

The length of time it takes to read or write on tape is obtained by adding this figure to the duration of reading or writing a tape record:

	Duration
729 II	10.8 + CN milliseconds
729 IV	7.3 + CN milliseconds

C = Milliseconds per character
N = Number of characters

Regardless of the 729 model used, the tape read or write instruction itself takes about 108 microseconds to be executed. This is the length of time taken before the next stored-program instruction can be executed.

Tape Segment Forward Space per Count

7. (P)TSF

The tape unit denoted by positions 1 and 4 of the instruction is instructed to move the tape forward until the specified number of tape segments has been spaced over. The number of segments to be spaced over is indicated by 1 plus the difference in the start and stop addresses of the record-definition word addressed by positions 6-9 (indexable).

The working and stop address portions of the record-definition register, although they don't represent core-store addresses to be read into or written from, work in much the same way. As each segment mark (CA8421) is spaced over, the *working address* is increased by one and compared with the *stop address*. When they become equal, the tape operation stops. The tape is positioned ready to read the first record in the next segment.

Only one record-definition word is used in this operation, regardless of whether its sign is plus or minus. If positions 2-5 of this word are 0000, positions 6-9 contain the number of segment marks to be spaced over.

EXAMPLE: To move tape unit 5 in channel 1 forward the number of segments specified by the difference of the values in positions 2-5 and 6-9 of word 1645. No normal-condition priority signal is to be given:

S01	23	4	5	6789
−81	00	5	7	1645

TIMING: The time it takes to space over several tape records is dependent on the number of records spaced over and the size of each record:

C = Milliseconds per character
N = Number of characters
R = Number of multi-character records
S = Number of single-character records

	Milliseconds
729 II	CN + 10.3 (R+S)
729 IV	CN + 6.8 (R+S)

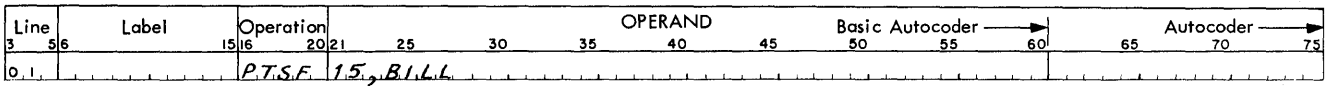


FIGURE 139.

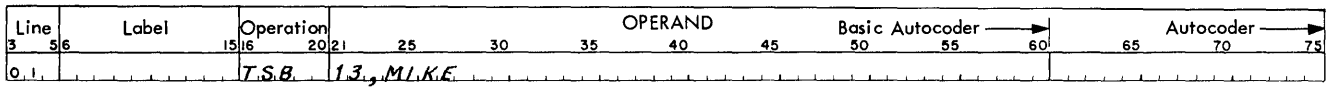


FIGURE 140.

AUTOCODER EXAMPLE (Figure 139): BILL is 1730. The assembled instruction is:

S01	23	4	5	6789
+81	00	5	7	1730

AUTOCODER EXAMPLE (Figure 140): MIKE is 3841. The assembled instruction is:

S01	23	4	5	6789
−81	00	3	8	3841

Tape Segment Backspace per Count
8. (P)TSB

This code works in the same manner as TAPE SEGMENT FORWARD SPACE PER COUNT, with this difference:

The tape is moved backward, instead of forward, until the specified number of segment marks has been sensed. This instruction can be used with an output tape, one that is being written; whereas TAPE SEGMENT FORWARD SPACE should be used only with input tapes.

EXAMPLE: To move tape unit 2 on channel 1 backward the number of segments specified by the difference of the values in positions 2-5 and 6-9 of word 1414, and to signal for normal-condition priority at completion:

S01	23	4	5	6789
+81	00	2	8	1414

TIMING: Tape characters cannot be read when the tape is being moved backwards. A single-character record may be a segment mark, or not; thus, when a single-character record is detected during a (P)TSB operation, the tape must be stopped and moved forward, to read the character. This takes 68.6 milliseconds for Model II units, and 47.5 for Model IV units:

C = Milliseconds per character
N = Number of characters
R = Number of multi-character records
S = Number of single-character records

	Milliseconds after Read
Model II	CN + 10.3 (R−1) + 68.6S
Model IV	CN + 6.8 (R−1) + 47.5S
	After Write,
Model II	Add 7.5 milliseconds to above
Model IV	Add 5.0 milliseconds to above

Digit 0 in Position 5

These codes are defined by a digit 0 in position 5 and digits 0-9 in position 9 of the tape-control instruction. Positions 6-8 can be of any value; they are ignored. In all cases, however, positions 6-9 are indexable, just as if they represented an address. These operations do not require record-definition words.

With the exceptions of TAPE MARK WRITE and TAPE SEGMENT MARK WRITE, these operations cannot be used to signal priority. Thus, the sign of the instruction can be either + or − with no effect on the operation.

Tape No-Op Select
xxx0 TSEL

This code is the means of testing for the availability of a specific tape unit. When this instruction is executed, the channel becomes busy if the specified tape unit is not available. The actual test is performed by a BRANCH CHANNEL BUSY operation (BCB +51), which normally follows it.

The purpose of this instruction is a test of the operator, to check whether a tape reel has been loaded on the unit and made ready to be read or written.

EXAMPLE: To set up tape unit 4 on channel 1, so that a BCB test will determine whether it is available to the stored program:

S01	23	4	5	6789
±81	00	4	0	xxx0

Sign can be either + or − Positions 6-8 can contain any digits.

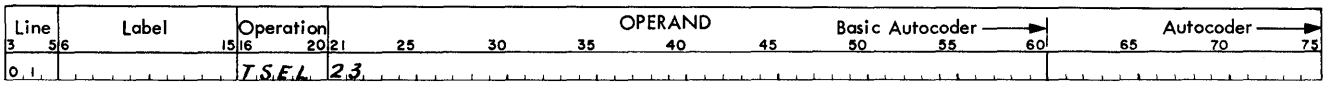


FIGURE 141.

TIMING: 108 microseconds.

AUTOCODER EXAMPLE (Figure 141): The assembled instruction is:

S01	23	4	5	6789
+82	00	3	0	0000

Tape Mark Write
xxx1 **(P)TM**

This code writes a tape mark (8421) on the tape unit addressed by the digits in positions 1 and 4. The tape mark is a single-character record, preceded and followed by an IRG. It is not in core storage; this instruction automatically produces the tape mark.

This instruction can be used to initiate a normal-priority signal (plus-sign in the operation code).

EXAMPLE: To write a tape mark on tape unit 4 of channel 2, with no signal for normal-condition priority:

S01	23	4	5	6789
−82	00	4	0	xxx1

Positions 6-8 can contain any digits.

TIMING: The equivalent of writing a single-character record:

729 II, High Density	10.8 + .024 milliseconds
729 II, Lower Density	10.8 + .067 milliseconds
729 IV, High Density	7.3 + .016 milliseconds
729 IV, Lower Density	7.3 + .044 milliseconds

AUTOCODER EXAMPLE (Figure 142): The assembled instruction is:

S01	23	4	5	6789
+81	00	2	0	0001

Tape Rewind
xxx2 **TRW**

The tape unit designated by the digits in positions 1 and 4 is instructed to rewind its tape. The unit is effectively disconnected from the system until the rewind operation is completed, at which time it again becomes available to the stored program. The ready light on the unit turns off while the tape is rewinding, and turns on again when the unit is again ready for use.

EXAMPLE: To rewind tape unit 0 on channel 4 and make it again available to the stored program after the rewind operation is completed:

S01	23	4	5	6789
±84	00	0	0	xxx2

Sign can be either plus or minus. Positions 6-8 can contain any digits.

TIMING: An entire 2400-foot reel of tape takes about 1.2 minutes to rewind on a 729 II Unit; .9 minute on a 729 IV Unit.

The tape channel is released and made available to the stored program after 35 milliseconds.

AUTOCODER EXAMPLE (Figure 143): The assembled instruction is:

S01	23	4	5	6789
+84	00	0	0	0002

Tape Rewind Unload
xxx3 **TRU**

This code is the same as TAPE REWIND except that the tape unit is *not* available to the system after the rewind operation is completed. It cannot be used again by the stored program until the operator makes it available.

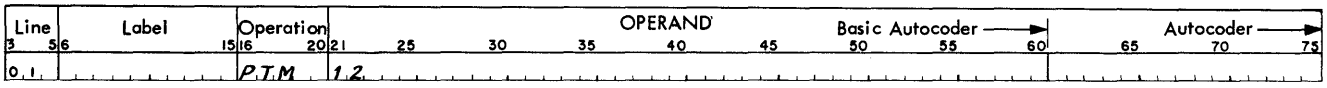


FIGURE 142.

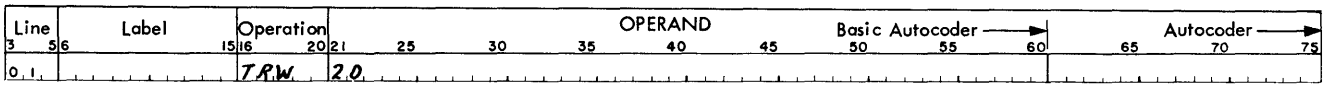


FIGURE 143.

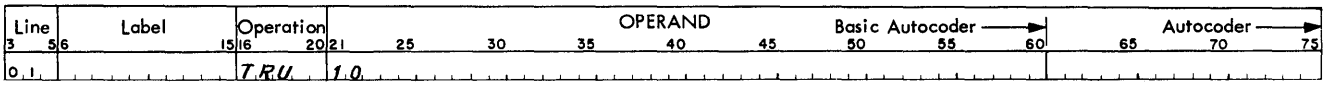


FIGURE 144.

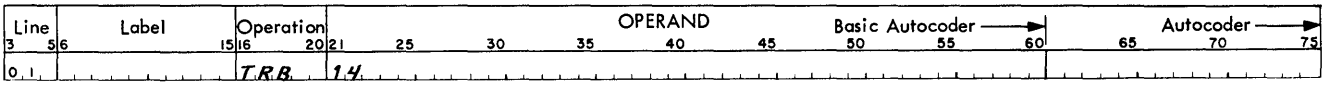


FIGURE 145.

EXAMPLE: To rewind tape unit 5 on channel 1 and make it unavailable to the stored program at the completion of the rewind operation:

S01	23	4	5	6789
±81	00	5	0	xxx3

Sign can be either plus or minus. Positions 6-8 can contain any digits.

TIMING: Same as TAPE REWIND.

AUTOCODER EXAMPLE (Figure 144): The assembled instruction is:

S01	23	4	5	6789
+81	00	0	0	0003

Tape Record Backspace
xxx4 TRB

The specified tape unit backspaces the tape one record (tape records are defined as separated by Inter-Record Gaps). If the tape backspaces over a segment mark or tape mark, this is considered as one record. If the tape is at load point at the time this command is given, no tape motion takes place; the instruction is the equivalent of NOP.

EXAMPLE: To backspace tape unit 2 on channel 2 one record:

S01	23	4	5	6789
±82	00	2	0	xxx4

Sign can be either plus or minus. Positions 6-8 can contain any digits.

TIMING:

	Milliseconds, after a read operation
729 II	46.4 + CN
729 IV	32.6 + CN

	After a write operation
729 II	Add 7.5 milliseconds to above
729 IV	Add 5.0 milliseconds to above

C = Milliseconds per character
N = Number of characters in the tape record.

AUTOCODER EXAMPLE (Figure 145): The assembled instruction is:

S01	23	4	5	6789
+81	00	4	0	0004

Tape Segment Mark Write
xxx5 (P)TSM

This instruction causes a tape segment mark (CA8421) to be written as a one-character record on the tape. The segment mark is automatically provided for this operation; it doesn't need to be in storage.

This instruction can be used to initiate a normal-priority signal (+ sign in the operation code).

EXAMPLE: To write a segment mark on tape unit 3 on channel 2 and to signal for normal-condition priority:

S01	23	4	5	6789
+82	00	3	0	xxx5

Positions 6-8 can contain any digits.

TIMING: The equivalent of writing a single-character record:

729 II, High Density	10.8 + .024 milliseconds
729 II, Lower Density	10.8 + .067 milliseconds
729 IV, High Density	7.3 + .016 milliseconds
729 IV, Lower Density	7.3 + .044 milliseconds

AUTOCODER EXAMPLE (Figure 146): The assembled instruction is:

S01	23	4	5	6789
−81	00	2	0	0005

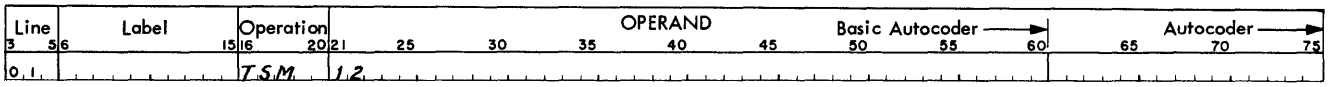


FIGURE 146.

Tape Skip
xxx6 **TSK**

The specified channel is conditioned so that the tape unit will space forward about four inches before writing in the next write instruction. The tape passed in the write operation is erased. The unit on which the spacing takes place is the first unit, on a channel, which is instructed to write following the skip instruction for any unit on that channel.

This command is used after rewriting a record several times has failed to correct an error detected by the two-gap head, usually due to a defection on the surface of the tape. It is usually followed immediately by a write operation. If the next instruction for that channel is a read command, no skipping takes place, and the skip instruction loses its effect on any subsequent write command. Any other type of operation that intervenes before a write command does not affect the function of skip on the write command.

EXAMPLE: To condition a tape unit on channel 3 to space forward about 4 inches prior to writing, on the next write instruction:

S01	23	4	5	6789
±83	00	Y	0	xxx6

Position four can designate any tape unit, on that channel, that is available to the stored program.

Normally, the unit is the same one designated by the subsequent write instruction; it is the write instruction that spaces the specific tape unit. Sign can be either plus or minus. Positions 6-8 can contain any digits.

TIMING: If a tape-write operation has been preceded by a skip instruction, 40 milliseconds are added to write time for Model II units, and 27 are added for Model IV.

AUTOCODER EXAMPLE (Figure 147): The assembled instruction is:

S01	23	4	5	6789
+82	00	5	0	0006

Tape End of File Off
xxx7 **TEF**

Any time that a tape unit senses a foil strip while writing, or a tape mark while reading, the tape indicator light on that unit turns on, and an EOF condition code results (see section on *Automatic Priority Processing).* This instruction turns it off. Any time that the EOF indicator is on, a read or write operation involving that tape unit causes an EOF priority condition (see section on *Automatic Priority Processing).*

EXAMPLE: To turn off the EOF indicator for tape unit 5 on channel 1:

S01	23	4	5	6789
±81	00	5	0	xxx7

Sign can be plus or minus. Positions 6-8 can contain any digits.

TIMING: 108 microseconds.

AUTOCODER EXAMPLE (Figure 148): The assembled instruction is:

S01	23	4	5	6789
+81	00	1	0	0007

Tape Set Lower Density
xxx8 **TSLD**

The tape unit specified by positions 1 and 4 is set to operate at a density of 200 characters per inch. If the unit is a 729 II, its passing speed will be 15,000 characters per second on subsequent read or write operations; if the unit is a 729 IV, its speed will be 22,500 characters per second.

If the tape is to be read, its characters must have the same density. If not, an error (Priority condition code 1) is indicated.

EXAMPLE: To set tape unit 3 on channel 2 to a density of 200 characters per inch:

S01	23	4	5	6789
±82	00	3	0	xxx8

Sign can be either plus or minus. Positions 6-8 can contain any digits.

TIMING: 108 microseconds.

AUTOCODER EXAMPLE (Figure 149): The assembled instruction is:

S01	23	4	5	6789
+82	00	5	0	0008

Tape Set High Density
xxx9 TSHD

The tape unit specified by positions 1 and 4 is set to operate at a density of 556 characters per inch. If the unit is a 729 II, its character rate will be 41,667 characters per second on subsequent read or write operations; if the unit is a 729 IV, its rate will be 62,500 characters per second.

If the tape is to be read, its characters must have this same density. If not, an error (Priority condition code 1) is indicated.

EXAMPLE: To set tape unit 5 on channel 1 to a density of 556 characters per inch:

S01	23	4	5	6789
±81	00	5	0	xxx9

Sign can be either plus or minus. Positions 6-8 can contain any digits.

TIMING: 108 microseconds.

AUTOCODER EXAMPLE (Figure 150): The assembled instruction is:

S01	23	4	5	6789
+81	00	2	0	0009

Data Flow

Data flow for all tape read and write operations that involve movement of information between tape and core storage is described in the section on *Block Transmission.*

Included in these operations are:

Tape Read	(P)TR
Tape Read per Record Mark Control	(P)TRR
Tape Write	(P)TW
Tape Write per Record Mark Control	(P)TWR
Tape Write with Zero Elimination	(P)TWZ
Tape Write per Record Mark Control and with Zero Elimination	(P)TWC

The other tape operations do not involve the flow of data, but are instructions to a specific tape unit or channel; to backspace, rewind, write a segment mark, etc.

REGISTERS AFFECTED: For the read and write operations (listed above), the record-definition register, transmission registers A and B, the adder, and the RDW address register; for tape-mark write and tape segment-mark write, transmission registers A and B only (none of these registers is addressable).

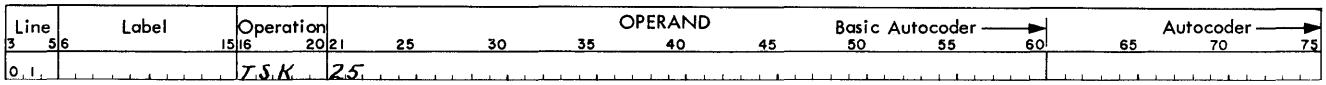


FIGURE 147.

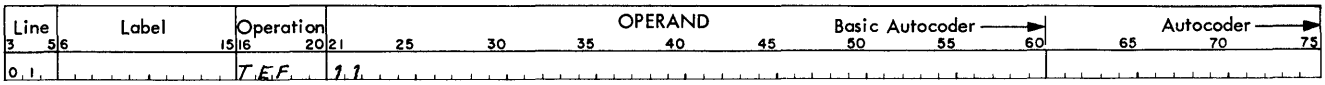


FIGURE 148.

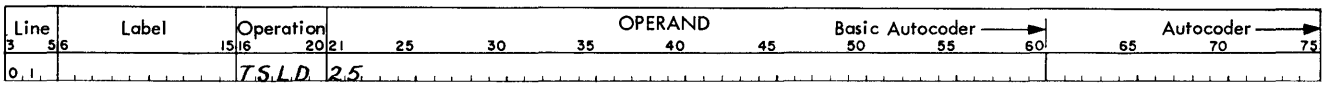


FIGURE 149.

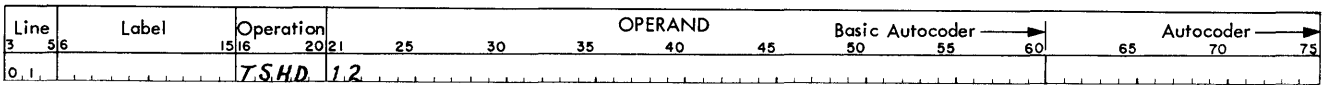


FIGURE 150.

Disk Storage

Large-capacity random-access storage is available to the IBM 7070 through the IBM 7300 Disk-Storage Unit (Figure 151). As many as four of these units can be attached to a 7070 system. The 7300 is available in two models: Model 1, with capacity of 6 million digits of information; and Model 2, with a capacity of 12 million digits.

The units are connected to the 7070 through a switching matrix, which allows any two of the four 7300 units to be connected at one time to the first two channel controls.

Data is stored in the form of magnetized spots on the surface of a circular disk, coated with ferrous oxide. Each storage unit contains 50 disks, mounted on a vertical shaft, which turn continuously at a speed of 1200 revolutions per minute. Because data can be recorded on both sides of each disk, each IBM 7300 unit contains 100 disk faces. Reading and writing data on the disk faces are performed by read-write heads, mounted on movable access arms. The arms seek records in storage by moving up or down to any disk, and horizontally across a disk face (Figure 152).

FIGURE 151. IBM 7300 DISK STORAGE UNIT

Each disk-storage unit has three mechanically independent access arms, all of which can be seeking at the same time. The stored program can continue while the seek operations are taking place.

Organization of Data in Disk Storage

When an access arm seeks, it seeks a *track* (Figure 152). Each track contains a record of data. Model 1 units have 100 tracks on a disk face. Each track contains 600 digits of data, comprised of 60 words with signs:

Model 1 Units	600	Digits per track
×	100	Tracks per disk face
=	60,000	Digits per disk face
×	100	Disk faces per disk-storage unit
=	6,000,000	Digits per disk-storage unit
×	4	Model 1 disk-storage units
=	24,000,000	Digits of disk storage maximum in a 7070 with Model 1 units.

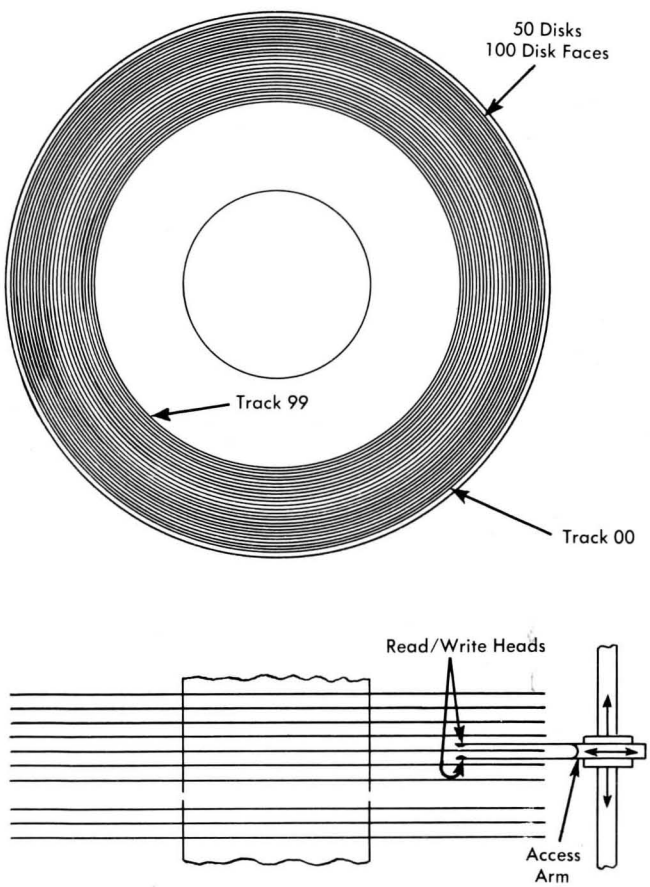


FIGURE 152. DISK STORAGE SCHEMATIC

Model 2 Units have 200 tracks on a disk face (double capacity). Each track contains 600 digits of data, comprised of 60 words with signs:

Model 2 Units	600	Digits per track
×	200	Tracks per disk face
=	120,000	Digits per disk face
×	100	Disk faces per disk-storage unit
=	12,000,000	Digits per disk-storage unit
×	4	Disk-storage units in a full-capacity 7070
=	48,000,000	Digits of disk storage in a full-capacity 7070

DISK FACE: The 100 disk faces in a file unit are numbered 00-99. Faces 00-49 are the top faces of the disk from top to bottom, and 50-99 are the bottom faces from top to bottom (Figure 153).

FILE UNIT: If only Model 1 units are used as disk storage for an IBM 7070 Data Processing System, the file unit identification ranges from 0-3 for the four (maximum) 7300 Model 1 units attached. On each disk face, the outside track is track 00, the next is track 01, etc.; and the inside track is track 99. Figure 154 is a schematic representation of this address arrangement.

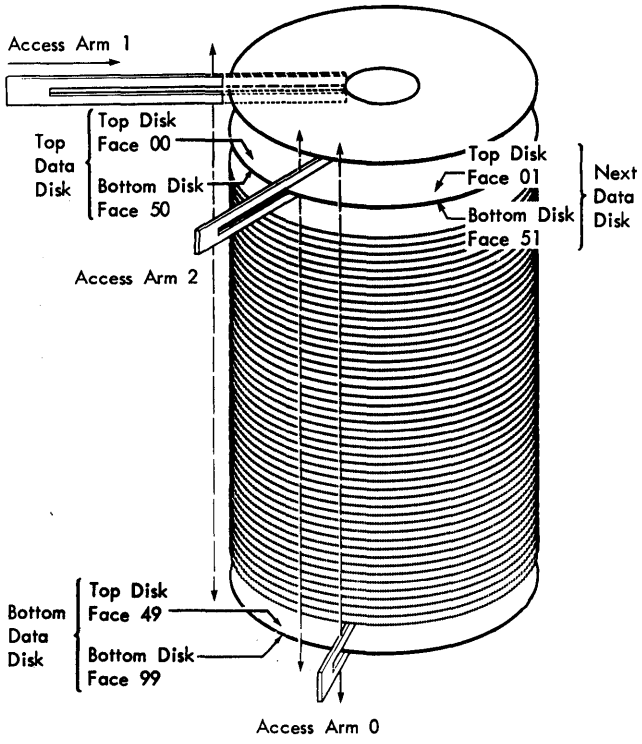


FIGURE 153. SCHEMATIC OF 7300 DISK FACE ARRANGEMENT

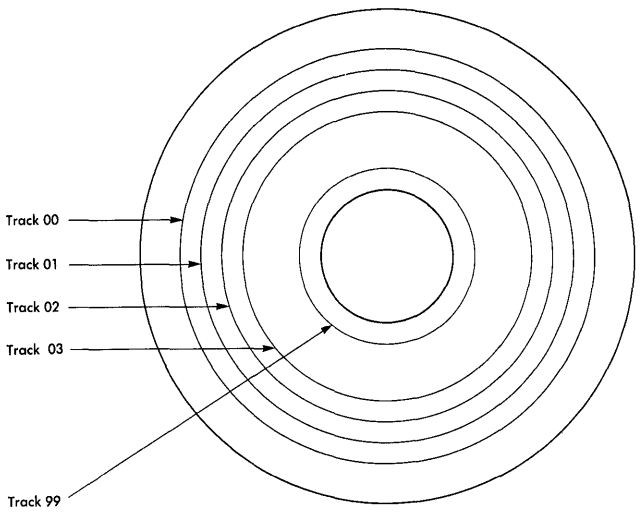


FIGURE 154. TRACK SCHEMATIC OF AN IBM 7300 MODEL 1 DISK

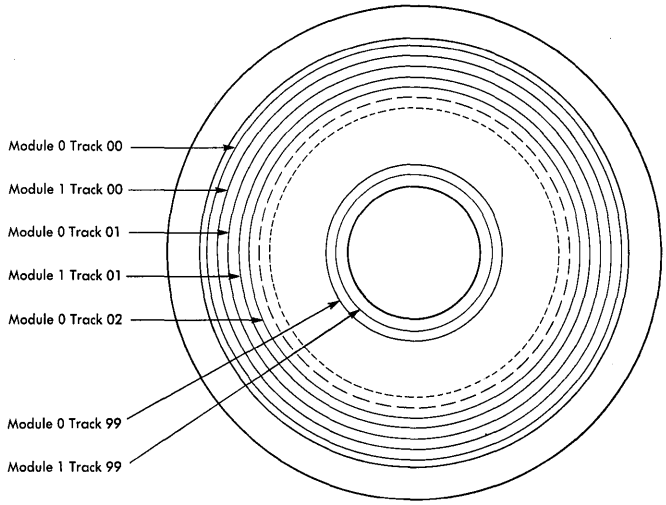


FIGURE 155. TRACK SCHEMATIC OF AN IBM 7300 MODEL 2 DISK

MODULE: There are two modules in each IBM 7300 Model 2 Disk-Storage Unit. In addressing, the modules are numbered 0-7; 0-1 are in the first file unit, 2-3 in the second, 4-5 in the third, and 6-7 in the fourth. The tracks for each of the two modules in a unit alternate. The outside track of the first disk face in the first file unit is track 00 of module 0. The next track is track 00 of module 1, the next track is track 01 of module 0, etc. Figure 155 is a schematic representation of this address arrangement. It shows the outside and inside tracks of a disk face in the first file unit.

ADDRESS: A *seek* operation code starts an access arm moving to one of the tracks in the disk files. The specific track is indicated by its address, which is in accumulator 3. The program must put the address into accumulator 3 for a seek, read, or write instruction.

A disk-storage address contains 7 digits. Its value in accumulator 3 ranges from 0000000 through 2379999 for a 7070 system that contains Model 2 units or both models, or from 0000000 through 2339999 for a 7070 with Model 1 units only (Figure 156).

Note that positions 4 and 5 both designate file unit, for a system with Model 1 units only. As described under the seek operation code, the file unit designation in position 4 is automatically provided by a seek instruction, by interpreting the module (Model 2) or file-unit (Model 1 only) designation in position 5.

COMBINED DISK STORAGE (MODELS 1 AND 2): A 7605 Disk Storage Control capable of handling the 7300 Model 2 can also accommodate the 7300 Model 1. Both models can be used in the same system. The 7300 Model 2 units are assigned the lower module numbers. For example, a 7070 that has two Model 2 units and one Model 1 unit has the addressing system shown in Figure 157 (not including access-arm designation). Location 0000 of each module should be reserved for customer engineer use.

S012 UNUSED	3 ACCESS ARM	4 FILE UNIT	5 MODULE	67 DISK FACE	89 TRACK
Model 2 or both	0 to 2	0 to 3	0 to 7	00 to 99	00 to 99
Model 1 Units Only	0 to 2	0 to 3	0 to 3	00 to 99	00 to 99

FIGURE 156. RANGE OF ADDRESSES — COMBINED MODELS, AND MODEL 1 ONLY

DISK UNIT	MODULE	FROM	TO
0-Model 2	0 1	000000 010000	009999 019999
1-Model 2	2 3	120000 130000	129999 139999
2-Model 1	4	240000	249999

FIGURE 157. ADDRESSES — MODELS 1 AND 2 COMBINED

Checking

There are a number of automatic self-checking features, which assure the accuracy of disk-storage seek, read and write operations. These checks insure:

a. That a seek operation is completed.
b. That the correct address is obtained.
c. That the program is reading or writing the same track that the seek operation obtained.
d. That all of the digits moved on a read or write operation satisfy the two-out-of-five validity check.
e. That the data written on a disk track is exactly the same as the data in the storage locations from which it was written.

All errors are detected on read or write operations. The checks are described here in the sequence in which they are made. Each type of error that can be detected has a condition code, which can be interrogated by the program. (See section on *Automatic Priority Processing*. The priority condition code is indicated here for each checking feature.)

COMPLETION OF SEEK: Each access arm is given a specified length of time in which to complete a seek operation. This duration is well above the maximum time that the operation should take. If it has not completed the seek within this time, an unusual-condition priority signal is given on the first read or write operation performed by that arm, following the seek (Priority condition code 7).

PROGRAMMING ADDRESS CHECK: When a seek operation is initiated for an access arm, the address of the module (Model 2) or file unit (Model 1), disk face, and track,—(the 5 low-order digits of the 7-digit address), is sent to the *access register* for that arm, and is retained there. Just prior to the first read or write operation after the seek, the address in accumulator 3 is automatically compared with the address in the access register, to assure that the disk record being read or written is the one called for by the program (Priority condition code 6).

MACHINE ADDRESS CHECK: A track does not take a complete circumference of the disk; there is a *two-word gap* between the end of the record and the beginning (Figure 158). Every time it writes a record, the machine automatically writes the address of that track in the two-word gap. This address includes disk face, track number, and module (Model 2) or file unit (Model 1); 5 digits in all. The first read or write operation after a seek automatically compares the address in the access register with the address in the two-word gap. This checking feature insures that the access arm obtained the correct track in the seek operation, before a read or write operation on that track is performed (Priority condition code 5).

VALIDITY CHECK: Information is recorded on the disks in the same two-out-of-five coding as in magnetic-core storage. Every character that is read or written on a track is checked for validity to assure that it has exactly two bits, no more and no less. If an invalid character is detected, the machine automatically attempts several times to read or write without a validity error, before setting up the error code and proceeding with the stored program (Priority condition code 1).

COMPARE CHECK: With every disk-write operation, there is an automatic compare check. The record just written on the track is compared, character-by-character and bit-by-bit, with the data in the core-storage locations from which it was written. This is done even if the data is written from sev-

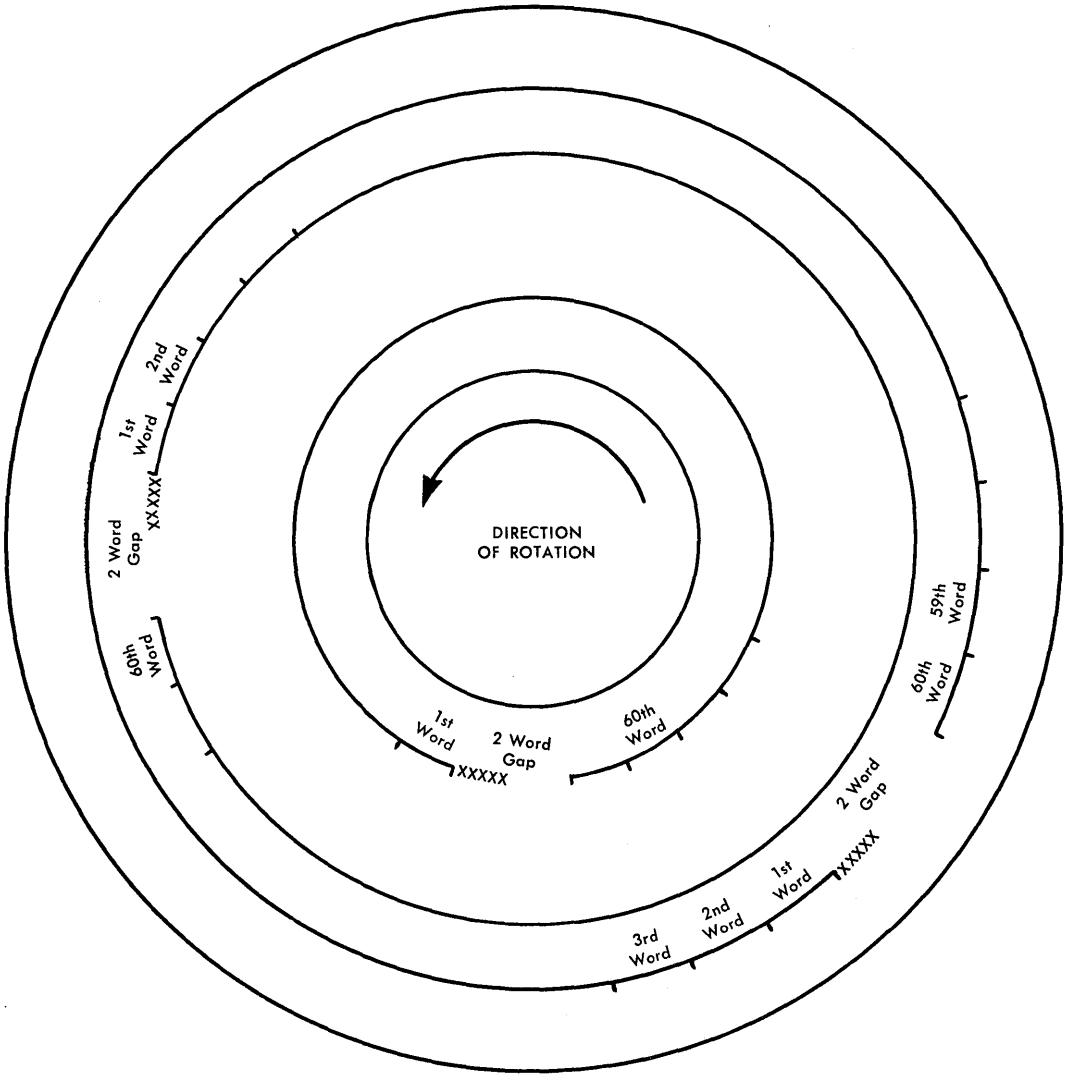


FIGURE 158. TWO-WORD GAPS

eral blocks of core storage, under control of record-definition words (Priority condition code 1). (The start address is obtained from the initial status word, as described in the section on *Automatic Priority Processing.)*

Figure 159 is a schematic representation of the checking features of disk-storage operations.

Scatter Read/Write

The scatter read/write feature available for tape operations can also be used in disk-storage read and write. It works in the same way, with the core-storage blocks defined by record-definition words. This feature can be used to scatter a record being read, or to "gather" a record from different storage blocks, just as in tape operations.

Simultaneous Operations

Seek operations for the 12 access arms (3 in each of the 4 files) are all totally independent of one another and of the stored program. All can be seeking simultaneously while the program continues.

The 7070 can perform two read/write operations simultaneously, by using both tape/disk-storage channels (channel controls 1 and 2). The total number of simultaneous read-write operations is two: two tape operations, two disk operations, or one of each. Because a file unit can be connected to only one channel at a time, two simultaneous disk-storage operations must use different file units.

Disk Storage Operation Codes

These instructions cover the following disk-storage operations:

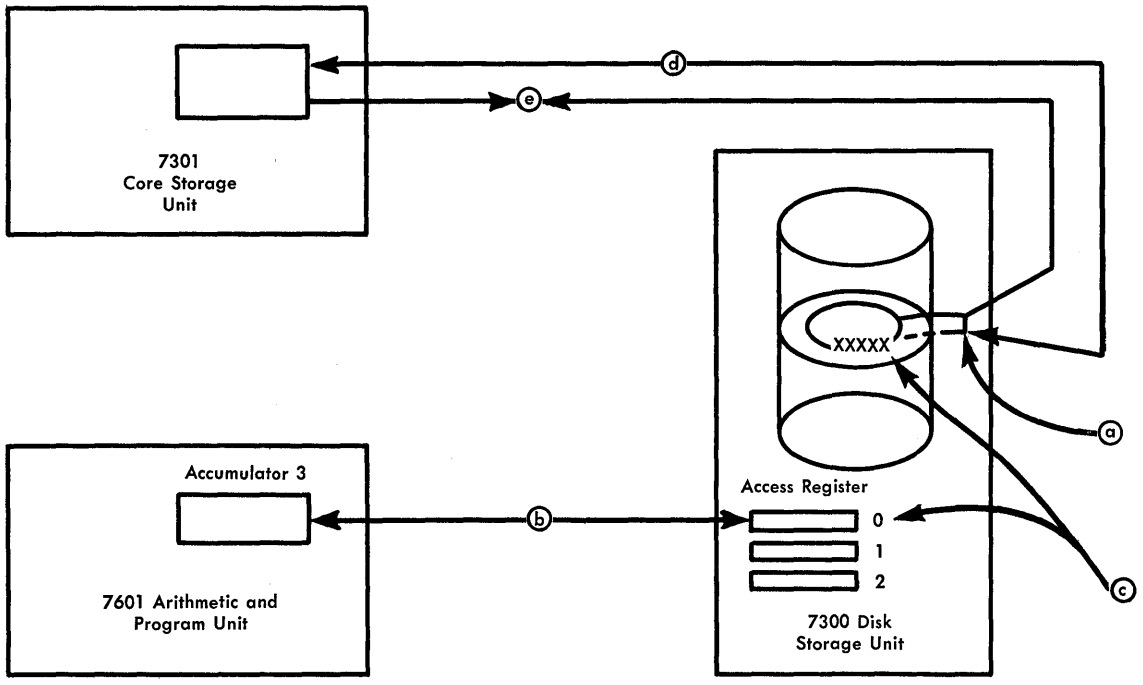
Read

Write

Provide an access arm and seek, or branch if arm not available.

Release access arm for further use.

All of these codes are in the 90's; position 0 is always 9. The read and write codes are ±91 and ±92. Seek is +95, and an access arm is released for further use by the program by a −95 instruction. In the case of read,



ⓐ Completion of Seek
ⓑ Programming Address Check
ⓒ Machine Address Check
ⓓ Validity Check
ⓔ Compare Check on Write

FIGURE 159. DISK STORAGE AUTOMATIC CHECKING FEATURES

write and seek, the stored program initiates the operation and can continue while the disk unit is operating. When a read or write operation is completed, the main program can be signalled for priority. A seek operation automatically initiates a priority signal at completion.

Disk Storage Control ±91, ±92

MACHINE DESCRIPTION: The disk-storage read/write operation codes are similar to the tape control codes. All read and write operations are incorporated in ±91 and ±92. With each code, the sign determines whether the read or write operation will initiate a normal-condition priority signal when the operation is completed; plus means that it will, minus means that it will not. (Unusual-condition priority occurs regardless of sign. See section on *Automatic Priority Processing.*)

The only difference between 91 and 92 in reading and writing is the channel involved; 91 specifies that channel control 1 is to be used for the flow of data in a read or write operation, and 92 is used for channel control 2. Position 5 of the instruction determines whether the operation is read or write.

INSTRUCTION FORMAT: S 0 1 23 4 5 6789

S — Plus means that there will be a priority signal at the end of the operation. Minus means that there will not be a priority signal unless there is an unusual condition (see section on *Automatic Priority Processing* for full descriptions of these conditions).

0 — Always 9.

1 — Always 1 or 2; depending on the channel to be used. Note that any disk-storage unit can read or write data through either channel. The channel is determined here.

(In tape operations, the channel designation helps to define which tape unit is to be used.)

23 — Indexing word. The address in positions 6-9 is modified by positions 2-5 of the indexing word specified here.

4 — Not used.

5 — Operation: 0.—Disk read (P)DR
1.—Disk Write (P)DW

6789 — The address of the first record-definition word to be read into or written from (indexable).

EXAMPLES: To read a disk-storage record through channel 2 into the core-storage area for which word 1204 is the first RDW, and signal for normal-condition priority at completion:

S01	23	4	5	6789
+92	00	x	0	1204

Position 4 can contain any digit.

To write a disk-storage record via channel 1 from the core-storage area for which the first RDW is: word 2700 indexed by positions 2-5 of IW 84; with no normal-priority signal at completion:

S01	23	4	5	6789
−91	84	x	1	2700

Position 4 can contain any digit.

DATA FLOW: The flow of data in disk read and write operations is described in the section on *Block Transmission.* Data flow is the same as for tape read and write operations.

REGISTERS AFFECTED: The record-definition register, transmission registers A and B, the unit adder, and the RDW address register. (None of these registers is addressable.)

TIMING: The disk read-write commands themselves take about 60 microseconds, just as the tape read-write codes. This is the amount of time that is taken before the next program step can be executed.

The duration of the actual read or write operation is determined by the setup time and by the speed of the disks. The setup time is the time it takes the access register to be set with the address in accumulator 3. The speed of the disk is 1200 rpm, or 50 milliseconds per revolution.

A read operation must wait for the beginning of the record on the track. This may be available immediately, or it may have just gone by the read/write head on the access arm when the read operation is ready to begin. The rotational delay is thus 0-50 milliseconds, for an average of 25. The timing of a disk-read operation is as follows:

Milliseconds	
38	setup time for the access register
+ 25	average rotational delay
+ 50	for the read operation itself (one complete revolution)
=113	average read time (88 minimum, 138 maximum)

A disk-write operation doesn't have to wait for the beginning of the record on the track. The operation begins as soon as the setup for the access register is completed. The automatic compare check for every write operation requires an extra disk revolution, during which the information written on the track is read back and compared with the core-storage words it was written from. The timing of disk-write operation is as follows:

Milliseconds	
38	setup time for the access register
+ 50	for the write operation itself (one revolution)
+ 50	for the automatic compare check (one revolution)
=138	write time

If a disk-write instruction is the first operation after the completion of a seek, additional time is taken for the machine-address check: The address in the 2-word gap is compared with the address in accumulator 3. In these cases, the write operation must wait for the two-word gap to pass under the read/write head. This adds 25 milliseconds average rotational delay to operation time, making a total of 163 milliseconds average time. The instance of write instead of read after a seek is rare, however. Except in file-loading routines, a seek operation is usually followed by a read.

AUTOCODER EXAMPLE (Figure 160): NORM is word 1850. The assembled instruction is:

S01	23	4	5	6789
+92	00	0	0	1856

Priority Disk Seek
+95 PDS

MACHINE DESCRIPTION: The seek operation code, +95, performs four functions:

1. Provides the file-unit number and places it into position 4 of accumulator 3. The module designation, in position 5 of the accumulator, determines this. There can be any digit in position 4 of accumulator 3 prior to the seek instruction.

2. Tests to see whether there is an access arm available in the file unit specified. The test can be for any access arm or for a specific arm. Each access arm has an *availability latch*, which is turned on when the arm is used by the program. An availability latch can be turned off only by −95 (DAR) instructions.

The digit in position 3 of accumulator 3 determines whether the +95 code will test for the availability of any access arm, or for a particular arm. If there is a digit 3-9 in that position, the test is for any arm. If the digit 0, 1, or 2 is used, the test is made for arm 0, 1, or 2, respectively.

3. If there is an arm available (or if the designated arm is available), a seek is initiated. In the case of testing for any arm, the digit (3-9) in position 3 of accumulator 3 is replaced by the digit representing the arm that has been provided (0, 1, or 2), and then the seek is initiated.

4. If there is no arm available (or if the designated arm is not available), the program branches to the location specified by positions 6-9 of the instruction (indexable), and a 9 is inserted in position 3 of accumulator 3.

A seek operation automatically sends a signal for priority when it is completed. (See section on *Automatic Priority Processing.)* For this reason, the *Autocoder* mnemonic for +95 is PDS.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+95
23	Indexing word. The address in positions 6-9 is modified by positions 2-5 of the indexing word specified here.
45	Not used.
6789	Branch address, used if *an* arm (3-9 in position 3 of accumulator 3), or *the specified* arm (0, 1, or 2 in position 3 of accumulator 3) is not available, and no seek is initiated. If a seek is initiated, this address is not used.

EXAMPLES: To seek track 47, disk face 28, with any arm available in module 6, and branch to location 4600 if there is no arm available:

Address in accumulator 3, prior to instruction:

S012	3	4	5	67	89
Sxxx	9	x	6	28	47

To test for any arm, position 3 can contain any digit 3-9. Sign can be plus, minus, or alpha, and positions 0-2 and 4 contain any digits.

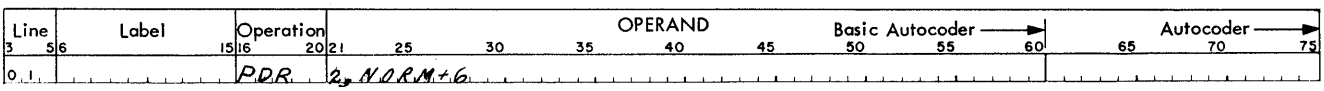


FIGURE 160.

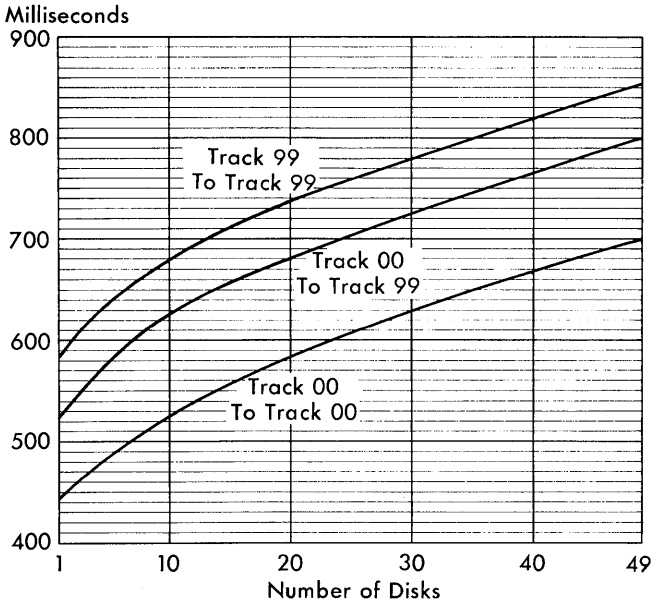


FIGURE 161. SEEK TIME — DIFFERENT DISKS

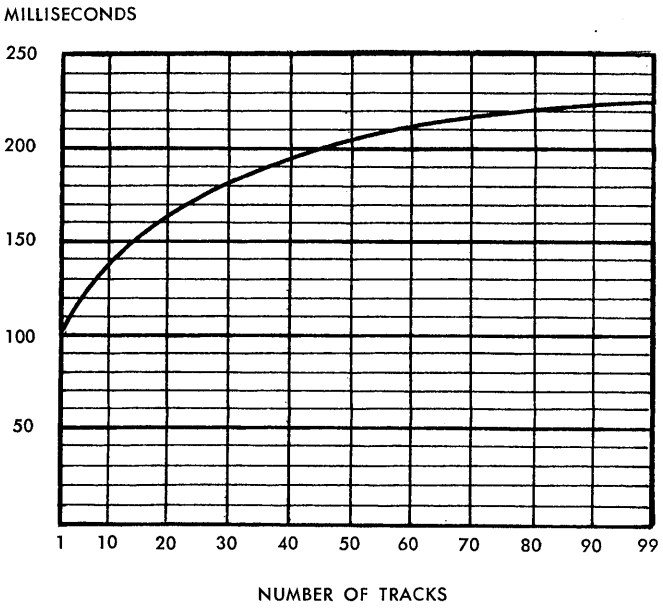


FIGURE 162. SEEK TIME — SAME DISK

Assume access arm 2 is available. Address in accumulator 3 after the instruction:

S012	3	4	5	67	89
Sxxx	2	3	6	28	47

(In any configuration of Model 1 and 2 units, module 6 designates file unit 3.) The program instruction to initiate the seek is:

S01	23	45	6789
+95	00	xx	4600

Positions 4-5 can contain any digits.

DATA FLOW: There is no movement of data in a seek operation.

If a seek is initiated, the address in accumulator 3 is moved to the access register for the arm that was provided, and the arm is sent to the track address.

If a seek cannot be initiated, the contents of positions 6-9, after indexing if so specified, are sent to the instruction counter.

REGISTERS AFFECTED: Accumulator 3 and the access register for the provided access arm. In the cases of any arm being provided (3-9 in position 4 of accumulator 3), the number of the arm that is made available replaces the digit in that position.

TIMING: The duration of a seek operation depends on how far the access arm has to go. To seek a track on another disk, the access arm moves horizontally, vertically, and horizontally again. The minimum time, from the outside track (00) of a disk to the outside track of an adjacent disk, is about 445 milliseconds. The maximum length of a seek operation is from track 99 of the top disk (face 00 or 50) to track 99 of the bottom disk (face 49 or 99), or vice-versa. This takes about 855 milliseconds. Figure 161 shows the maximum timings for seek operations using different disks.

For a seek to a different track on the same disk (disk face either the same or different by 50), the arm must move horizontally only. The timing ranges from 100 milliseconds for a movement of one track, to about 235 for 99 tracks (Figure 162).

The +95 operation itself takes 48 microseconds if an access arm is provided and a seek initiated. (This is the time taken before the next instruction can be executed.) If the program branches because of an arm being unavailable, the operation takes 24 microseconds.

AUTOCODER EXAMPLE (Figure 163): SUB1 is word 1561. The assembled instruction is:

S01	23	45	6789
+95	00	00	1561

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60		65 70 75
0 1		P.O.S.	SUB1		

FIGURE 163.

Disk Arm Release
−95 **DAR**

MACHINE DESCRIPTION: Once an access arm availability latch has been turned on by a +95 PDS instruction, it can be turned off only by a −95 DAR instruction. This means that it is unavailable for another seek operation until released.

INSTRUCTION FORMAT:

S01	23	4567	8	9

S01 −95

23 Indexing word. Although positions 6-9 do not represent an address, they can be modified by the value in positions 2-5 of the IW specified here.

4567 Not used.

8 0, 1, or 2; determines the access arm to be released

9 0, 1, 2, or 3; determines the file unit containing the arm.

EXAMPLE: To release access arm 2 in file unit 3:

S01	23	4567	8	9
−95	00	xxxx	2	3

Positions 4-7 can contain any digits.

REGISTERS AFFECTED: The access register for the designated access arm.

TIMING: 36 microseconds.

COMMENTS: The programmer never needs to know which arm was used in a seek operation, in order to release it with the DAR instruction. At the completion of each seek, read, or write operation, the access arm and file-unit number are automatically placed in positions 4 and 5, respectively, of index word 99. (They are actually part of the address of the final status word, as described in the section on *Automatic Priority Processing.*) Normally, the DAR instruction is given immediately after the last read or write operation for that arm, releasing it for a subsequent seek instruction. If the DAR instruction has 00 in positions 8-9, and is indexed by IW 99, the desired access-arm and file-unit number will be obtained. The DAR instruction looks like this:

S01	23	4567	8	9
−95	99	xxxx	0	0

In *Autocoder,* this would be as shown in Figure 164.

AUTOCODER EXAMPLE (Figure 165): The assembled instruction is:

S01	23	4567	8	9
−95	00	0000	0	3

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		DAR	0+X 99		

FIGURE 164.

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		DAR	03		

FIGURE 165.

Card Input-Output Operation Codes

All of the operations that involve the unit-record equipment: the IBM 7500 Card Reader, the IBM 7550 Card Punch, and the IBM 7400 Printer are incorporated in the +69 card control operation code. Data is moved from an input synchronizer to core storage on read instructions, or from core storage to an output synchronizer on punch or print instructions; and the unit-record machine connected with that synchronizer is operated for one cycle—reading a card, punching a card, or printing a line. Movement of data to and from core storage is under control of record definition words.

INSTRUCTION FORMAT: S01 23 4 5 6789

S01 These three positions of the card control instruction are always +69.

23 Indexing word. Position 6-9 can *in all cases* be modified by the value in positions 2-5 of the indexing word specified.

4 Specifies the synchronizer used. Synchronizers are numbered 1-3

5 Digits 0-3 in this position specify the unit record operation to be performed as follows.

0—Unit Record Signal	US
1—Read	UR
2—Write or punch	UW/UP
3—Write or punch invalid	UWIV/UPIV
4—Type	TYP

6789 Address of the initial record definition word.

Operations

The following are detailed descriptions of the unit record instructions.

Unit Record Read
1 **UR**

This command initiates a block transmission of from 1 to 16 words from the designated input synchronizer to storage locations specified by the process channel control system. Data to be entered is arranged by control-panel wiring. The address portion of the instruction specifies the location of the first record definition word (RDW).

Word 1 of the synchronizer is transmitted to the storage location specified by the first RDW start address. If the final RDW stop address is encountered before 16 words are transmitted, the last word transmitted is at that stop address location. If the start address of the RDW is greater than the stop address, the operation is terminated immediately. In no case are more than 16 words transmitted. There is no transmission of data if an end-of-file (EOF) condition is recognized.

After the transmission, an immediate re-start signal is created to allow the program to continue. A card read cycle is then performed to refill the synchronizer unless end of file has been sensed, or an error has been detected and RVI is wired to AUTO STOP. The instruction counter is advanced by:

1. One, if there is an invalid transmission.
2. Two, if there is an end-of-file condition.
3. Three, if the transmission of information is valid and no end-of-file condition exists.

Example: Read eight words from card reader synchronizer 1 to locations 0600-0607 (Figure 166).

	Location	Contents
Assembled Instruction	1000	+6900112000
Before	2000	−0006000607
Next Instruction is:		
1. if error	1001	Begin error routine
2. if EOF	1002	Begin EOF routine
3. if correct	1003	Continue processing

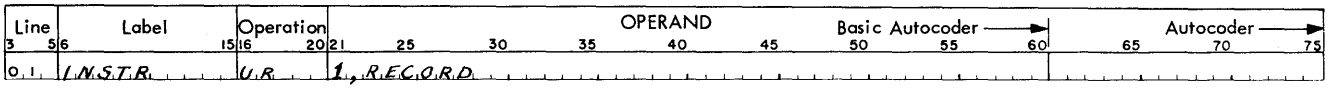


FIGURE 166

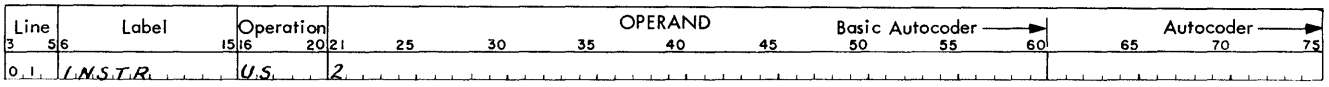


FIGURE 167

Unit Record Signal
0 **US**

This command does not cause any data transmission on the synchronizer specified. The only function executed is to turn on the program switch exit associated with the addressed input synchronizer. This control switch causes an early-timed impulse (PSE) to be emitted on the control panel of the associated input unit the next time that synchronizer is called for. If an end-of-file signal is given, no control panel impulse is available. This impulse is used to pick up selectors, impulse the offset stacker, etc. The instruction counter is always advanced by 1.

Example: Set up a signal for the next read cycle on card reader 2 (Figure 167).

	Location	Contents
Assembled Instruction	1000	+6900200000
Next Instruction is:	1001	Continue Processing

Unit Record Punch, Write
2 **UP, UW**

NOTE: Either punch or write is accepted by *Autocoder* for any card output unit. Both are assembled as a 2 in digit five.

This command initiates a block transmission of up to 16 words from the storage area specified by the RDW addresses. If the RDW is set for less than 16 words, the remaining synchronizer words are filled with zeros and assigned alpha signs. If the transmission of data from storage to synchronizer is valid, a punch/write cycle is then performed to empty the synchronizer while processing continues. Output format is determined by control-panel wiring.

The instruction counter is advanced by:

1. One, if the transmission of information from storage to synchronizer is invalid, and no printing or punching takes place.
2. Two, if the channel 9 latch is on.
3. Three, if the transmission of information is valid.

Example: Punch the 8 words contained in locations 0600-0607 on punch #1 (Figure 168).

	Location	Contents
Assembled Instruction	1000	+6900122000
Next Instruction is:		
1. if error	1001	Begin error routine
2. if channel 9	1002	Begin channel 9 routine
3. if correct	1003	Continue processing

Unit Record Punch Invalid, Write Invalid
3 **UPIV, UWIV**

NOTE: Either punch or write is acceptable for any card output unit. Both are assembled as a 3 in digit five.

This command has the function of the normal punch/write command except that the synchronizer-to-unit validity check is suppressed. It is a method to punch/write records containing invalid characters.

When the command is executed, the corresponding synchronizer-to-unit validity check is suppressed for one cycle only and a punch/write cycle is initiated. The normal storage to synchronizer-transmission is performed under control of the processing channel control system. The validity check circuits, in their suppressed condition, cause an early-timed error punch/write impulse (EWI) to be emitted on the control panel of the output unit. The instruction counter is advanced by one.

Output format is still under control of panel wiring.

Example: Print the 8 words contained in locations 0600-0607 on printer #2 (Figure 169).

	Location	Contents
Assembled instruction	1000	+6900232000
Next instruction	1001	Continue processing

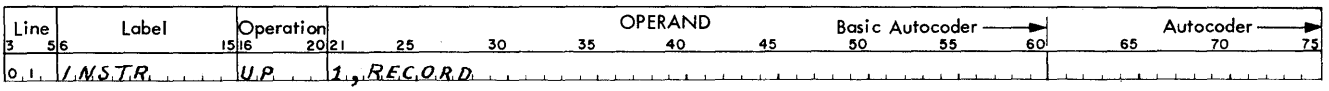


FIGURE 168

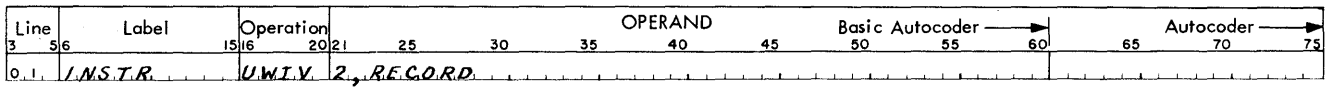


FIGURE 169

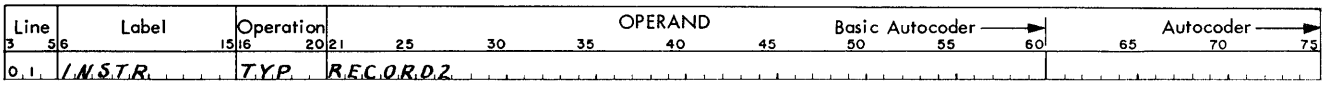


FIGURE 170

Type
4 **TYP**

This command initiates a block transmission of all words (specified by one or more RDW's) from core storage to the typewriter. The typing operation is preceded by a carriage return. Typing continues until the start and stop address in the record definition register are equal.

A carriage return is executed upon completion of the operation. All data words are typed under word sign control. The instruction counter is advanced:

1. If the transmission from storage to typewriter is invalid (typing is suspended).
2. If the transmission is valid.

Example: Type out the five words in locations 0810-0814 (Figure 170).

	Location	Contents
Assembled instruction	1000	+690042051
Next instruction		
if error	1001	Begin error routine
if correct	1002	Continue processing

Printer Channel 9 Test

The printer test for a punch in channel 9 of the carriage control tape is a function of the +69 card control command. This test is made at the time a print command is executed.

If a hole in channel 9 is sensed on a print cycle, the channel 9 latch is set ON. If a write invalid command is interpreted, the test of the channel 9 latch is not executed.

If the channel 9 latch is ON, the next printer write command is interpreted as a NOP instruction, and the program advances to IC+2. If the channel 9 latch is OFF, then the write command is executed and the program advances to IC+1 if invalid data was transmitted, or IC+3 if valid data was transmitted.

The channel 9 latch is reset at the beginning of the next printer operation even if the command is write invalid.

The channel 9 test signals the program when a specific line on a report has been printed and is used for printing page totals, page numbering, overflow sheet identifications, etc.

Timing

Transfers to and from the input-output synchronizers interrupt the execution of the 7070 program for a period of one to six milliseconds. The duration of this interruption (interlock time) is the same for all three units, but the compute time available to the program depends upon the specific unit being used.

In Figure 171, cycle time is the actual time it takes the corresponding unit to complete one operation, and compute time is the cycle time less the interlock time.

	MAXIMUM	MINIMUM
7500 Card Reader		
Cycle time — 120 ms		
Interlock time	6 ms	1 ms
Compute time	119 ms	114 ms
7550 Punch		
Cycle time — 240 ms		
Interlock time	6 ms	1 ms
Compute time	239 ms	234 ms
7400 Printer		
Cycle time — 400 ms		
Interlock time	6 ms	1 ms
Compute time	399 ms	394 ms

FIGURE 171. TIMING CHART

IBM 7500 Card Reader

One to three IBM 7500 Card Readers (Figure 172), each operating at a maximum speed of 500 cards per minute, can be attached to one IBM 7070 Data Processing System. Thus, maximum input speed can be up to 1500 cards per minute (120,000 characters per minute).

Control-panel signals can be initiated by the stored program to control format of subsequent cards or offset stacking.

FIGURE 172. 7500 CARD READER

An end-of-file key allows automatic transfer to programmed end-of-file procedures.

Reading with internally wired format control is accomplished by use of the load hub on the control panel.

SPOOL (Simultaneous Peripheral Operation On Line) routines can be performed by the card reader when priority signals for automatic program control are assigned by console setting.

The 7500 has three reading stations: The first is used to control format selection of alphabetic card fields and to control the *load* operation. The second is used to control format of numerical fields and to pre-read alphabetic data for checking purposes. The third station is used for data entry. Signs of numerical data can be punched in the card or generated by control-panel wiring.

Input Checking

An important consideration in any data processing system is the reliability of input data. The best programming unit in the world is without value if it processes incorrect or invalid information. The 7070 contains checking features as aids to assure this reliability.

VALIDITY CHECKING: The most important of these is validity checking. As each character is read from a card, it is translated into two-out-of-five bit coding on the synchronizer. When the data are moved from the synchronizer to storage, each digit is checked for validity.

In case an invalid character is detected, transmission of the input data from the synchronizer to storage is automatically repeated. If all the characters prove to be correct on this second try, the program continues normally; otherwise, an error indicator is set and the program branches.

Card Feeding

Cards are placed in the card hopper face down 9-edge first. The path of feeding (see *Card Feed Schematic*, Figure 173) is from the card hopper past first reading, second reading, third reading, the offset station to the stacker drum and then into the card stacker.

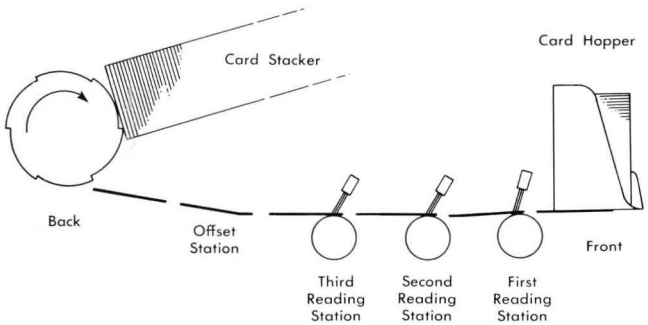


FIGURE 173. CARD FEED SCHEMATIC—7500

Operating Keys and Signal Lights

MASTER SWITCH: This switch (Figure 174) controls the power supply for the 7500. It is located on the left front of the machine.

The following keys and lights are located on the right front of the card reader:

START KEY: The start key feeds cards into the read feed. After the first card enters the input synchronizer, further feeding is under control of the 7070 program. This key is also pressed to re-start the machine if it has been stopped for any reason and for non process run-out when the hopper is empty.

STOP KEY: If the machine is in condition to accept read commands, pressing the stop key removes this condition and stops the motor. Also, if the end-of-file key has been previously pressed, the stop key may be used to nullify its effect.

RESET KEY: This key is used to extinguish lights caused by conditions that have stopped the machine after they have been noted by the operator. The machine is re-started by pressing reset and start. This key cannot extinguish the card advance light.

END-OF-FILE KEY: This key is pressed to allow the processing of cards remaining in the feed. (See *End-of-File* section.)

END-OF-FILE LIGHT: This light is turned on when the end-of-file key is pressed and there are cards in the hopper or if a card has been run in past the third reading station.

POWER ON LIGHT: This light is turned on when power is supplied to the 7500.

READY LIGHT: This light glows when the reader is in condition to respond to pressing the start key.

RING CHECK LIGHT: This light turns on when the number of significant characters to be read as indicated by the wiring from word exits to word size entry does not equal the number indicated by the wiring to entry end.

CLOCKING CHECK LIGHT: The machine makes an internal check to insure that the certain timing circuits are operating properly. If these circuits fail, the machine stops and the clocking check light comes ON indicating a possible error. This light is extinguished by pressing the reset key. Repeated clocking check lights indicate that the machine should be checked by a customer engineer.

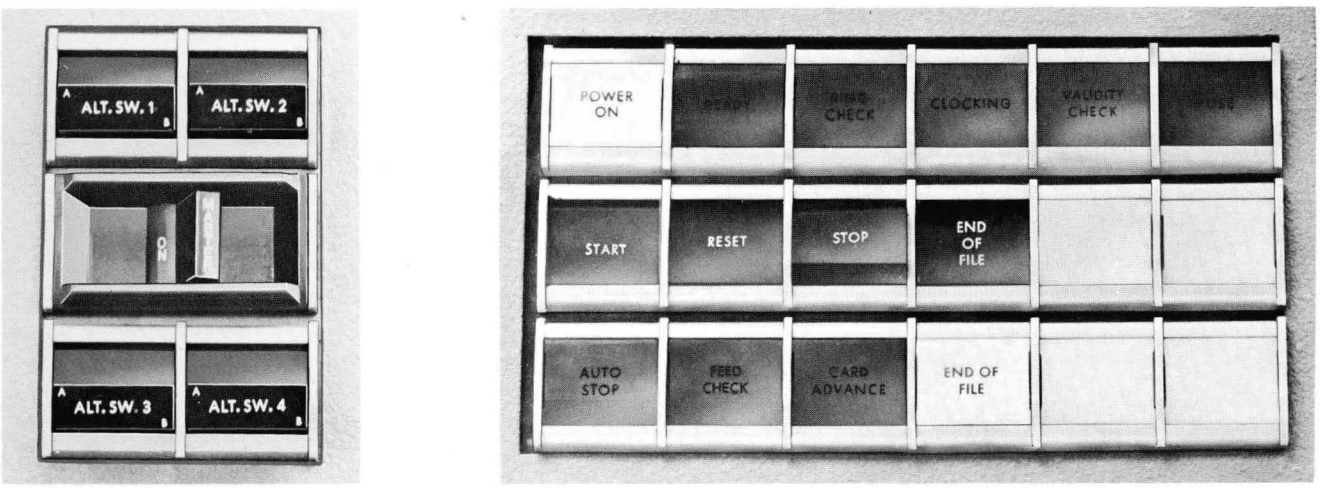


FIGURE 174. OPERATING KEYS AND SIGNAL LIGHTS

VALIDITY CHECK LIGHT: This light turns on when the validity-check circuits detect an invalid character. If RVI is *not* wired to AUTO STOP, the light goes out automatically. If RVI is wired to AUTO STOP, the light is turned off by pressing the reset key.

AUTO STOP LIGHT: This light glows whenever the reader stops as a result of a condition wired on the control panel to AUTO STOP. The machine is re-started by pressing the reset key and then the start key.

FEED CHECK LIGHT: This light turns on:

1. If the machine feeds a card without receipt of a read command (except on run-in or run-out).
2. If the machine fails to feed on receipt of a read command.

CARD ADVANCE LIGHT: This light turns on and the machine stops if a card fails to feed from the hopper into the machine or from one feed station to the next. The start key is inoperative until the cards have been removed from the hopper. The light is extinguished when the feed has been cleared. If there are cards in the feed unit when the power is turned off, the card advance light comes ON when the machine is next started. The light alerts the operator that there are cards in the feed.

FUSE LIGHT: This light turns on and the machine stops whenever a fuse burns out. *Note:* The main-line switch must be turned OFF while the fuse is replaced.

END OF FILE: If the End-of-File key is pressed at any time before the last card leaves the hopper, cards continue to feed until the last card passes third reading. If the end-of-file key is not pressed, card feeding stops when the last card leaves the hopper. If the end-of-file key is pressed then, the cards remaining in the read feed will advance until the last card passes third reading.

Whenever the last card passes third reading, an end-of-file (EOF) signal is sent to the program. In addition, the end-of-file latch is set ON as the information from the last card is being transferred from the input-synchronizer to core storage. Because the EOF latch is on, the next read command is interpreted as a NOP instruction, and the program advances to IC+2.

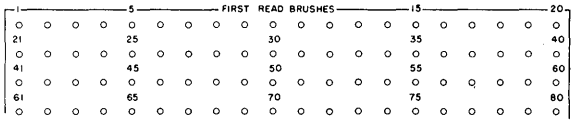
7500 Card Reader Control Panel

The IBM 7500 Card Reader control panel (Figure 175) has been designed to allow format flexibility of card input to the IBM 7070 Data Processing System. Data from the card is read and arranged in the synchronizer by control-panel wiring for entry into storage. All features and functions described in this section are included as standard equipment on the IBM 7500.

Planning Chart

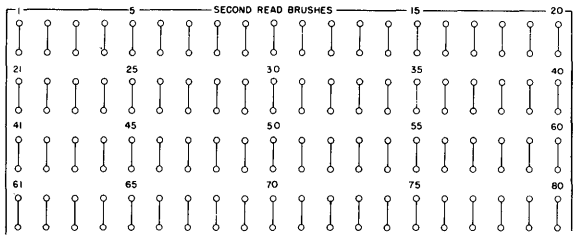
The reverse side of the IBM 7500 Card Reader control-panel diagram is the IBM 7500 Reader planning chart (Figure 176). If all notes that pertain to a given input format are written on this chart in their appropriate sections, the job of wiring the control panel is greatly simplified. The chart should be kept for use in control-panel testing and also serves as a permanent record of the job for which that control panel was designed.

Control Panel Hubs



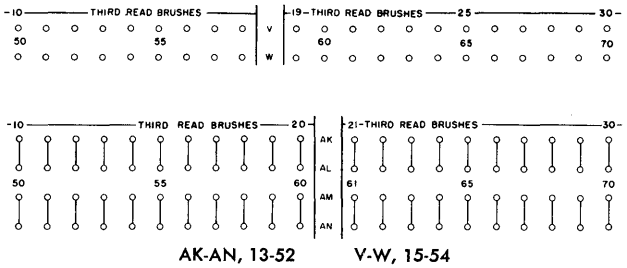
A-D, 1-20

FIRST READ BRUSHES: These 80 hubs are exits for impulses read from the card at the first reading station. They are normally used for control purposes.



A-H, 45-64

SECOND READ BRUSHES: These 80 common hubs are exits for impulses read from the card at the second reading station. They are normally used for control purposes or for alphabetic pre-read.



AK-AN, 13-52 V-W, 15-54

THIRD READ BRUSHES: The three sets of 80 hubs labeled THIRD READ BRUSHES are common.

These hubs are exits for impulses that originate as the card is read at the third reading station. These hubs are normally wired to storage entry but are *inactive on load cards.*

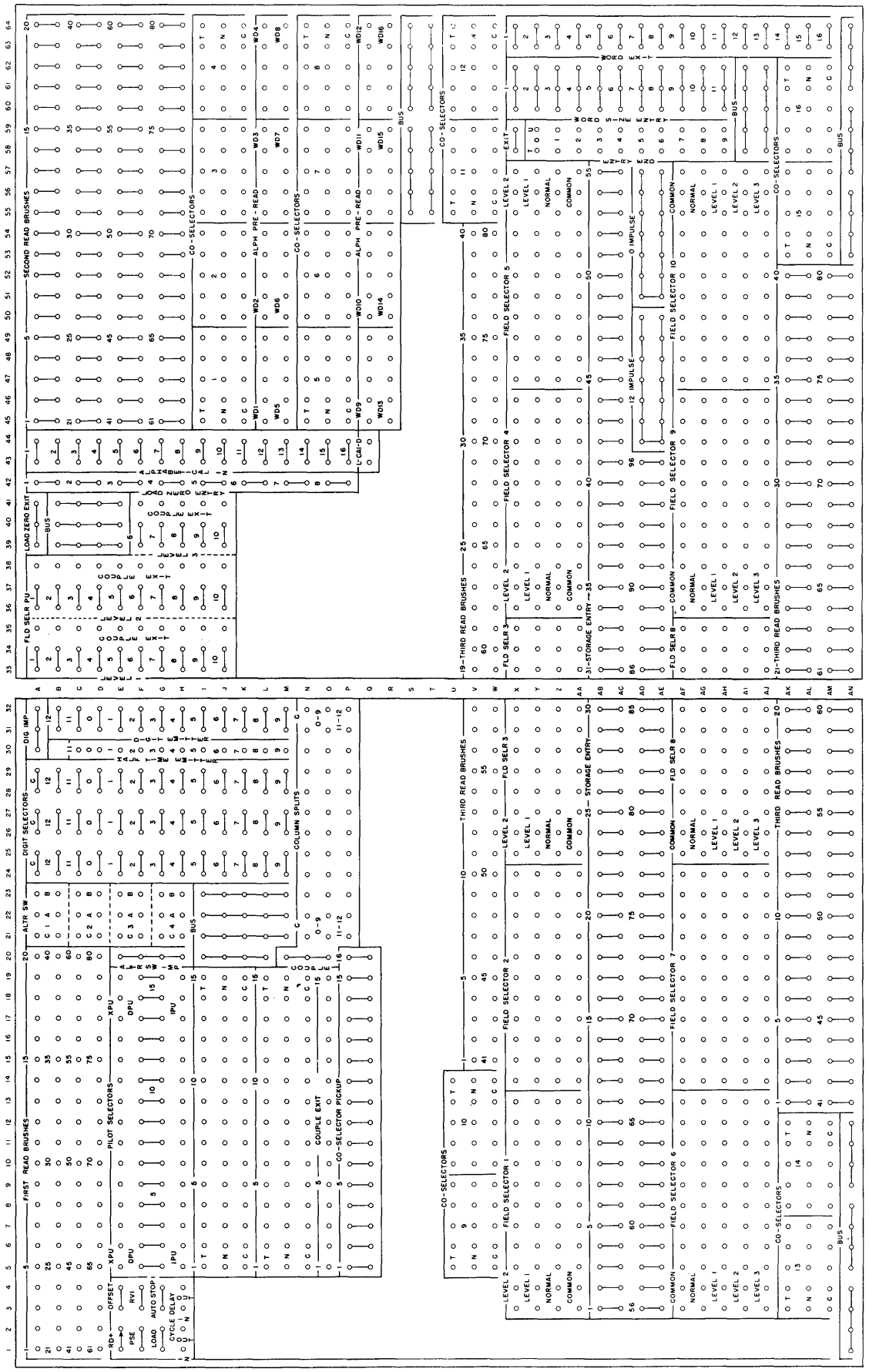


FIGURE 175. 7500 CARD READER CONTROL PANEL (FORM X24-6425)

FIGURE 176. READER PLANNING CHART

IBM 7500 READER PLANNING CHART

PREPARED BY _____ DATE _____

CARDS USED _____ APPLICATION _____

WORD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	REMARKS
NUMBER OF CHARACTERS TO BE READ (INCLUDE SIGN FOR NUMERIC)																	
ALPHABETICAL IN																	
ALPH PRE-READ																	
CONTROLS																	
FIELD HEADINGS																	
READING BRUSHES																	
STORAGE ENTRY																	

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	
PILOT SELECTORS	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	
	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
FIELD SELECTORS	3	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	2	2	2	2	2
	1	1	1	1	1	1	1	1	1	1
	N	N	N	N	N	N	N	N	N	N
	C	C	C	C	C	C	C	C	C	C

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
CO-SELECTORS	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C

MISCELLANEOUS

RD4 ☐

ALTERATION SWITCHES

	A	B		A	B
1	☐	☐	3	☐	☐
2	☐	☐	4	☐	☐

ENTRY END _____

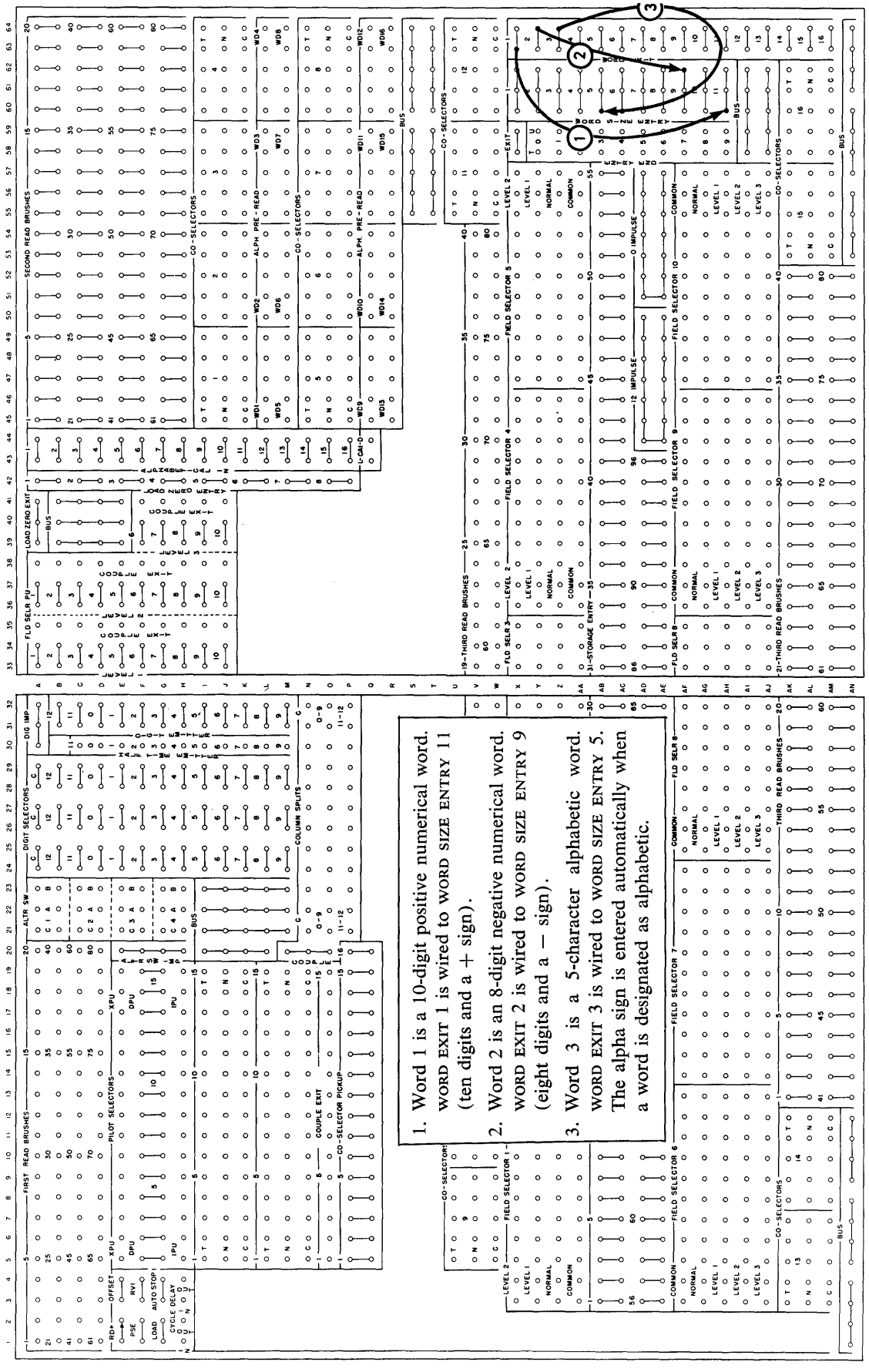
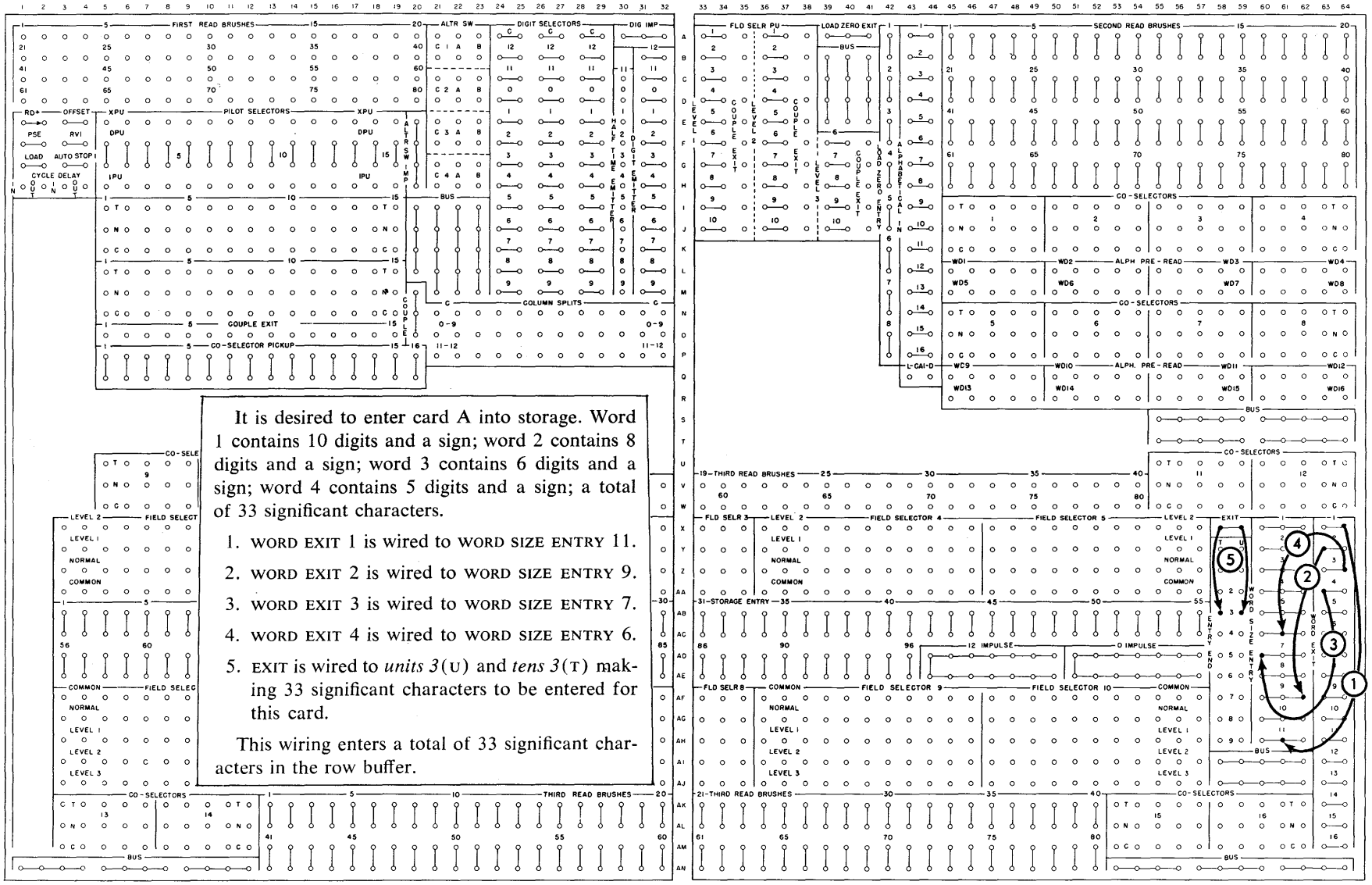


FIGURE 177. WORD SIZE WIRING

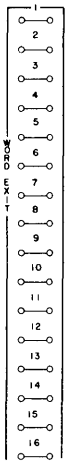
FIGURE 178. ENTRY END WIRING



Word Size Wiring

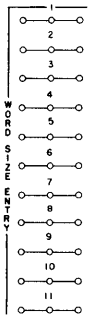
The word exit and word size entry hubs are used to determine the size of input words.

X-AM, 63-64



WORD EXIT, 1-16: These hubs indicate the number of character positions assigned to each input word. WORD EXITS are always wired to WORD SIZE ENTRY either directly or through properly controlled selectors.

X-AH, 60-62



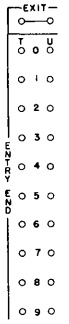
WORD SIZE ENTRY, 1-11: Impulses emitted by the WORD EXIT hubs are wired to WORD SIZE ENTRY to designate the actual number of wired character positions (including signs) that comprise a specific input word. Maximum word sizes are as follows:

		sign	word size
1. positive numerical word	10 digits	+	11
2. negative numerical word	10 digits	−	11
3. alphabetic word	5 characters	@	5

Figure 177 is an example of word size wiring.

Entry Control

X-AH, 58-59



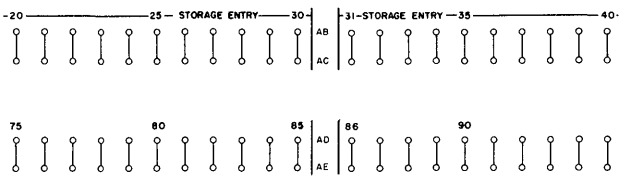
ENTRY END—UNITS, TENS, EXIT: These hubs are wired to indicate the total number of significant character positions that have been wired to storage entry for a given card format. An example is shown in Figure 178.

Sign Control

E, 1-2

RD+ (READ PLUS): This switch is wired when 12 punches are to be recognized as positive signs for numerical words. When RD+ is ON, each numerical word must receive an 11 or 12 impulse from the card or from the emitter. If the RD+ switch is unwired, any numerical word that does not receive an 11 impulse (minus sign) is treated as positive and a 12 impulse is automatically entered into the sign position. RD+ is automatically turned on for load cards.

Storage Entry



AB-AC, 3-57 AD-AE, 3-43

These 96 pairs of hubs are normally wired from the third reading brushes to enter data. They correspond to the 96 positions in the row buffer, which is a special synchronizer used for temporary storage. The words to be entered (as indicated by word size wiring) must be wired to these hubs in numerical sequence by word number.

Example: If the first word to be entered is numerical, then the sign of this word is wired to storage entry position 1; the high-order digit of the field is wired to storage entry position 2, etc. until the entire field is wired. The sign of the next word is entered in the storage-entry position immediately to the right of the low-order (units) position of the word preceding it.

It is not necessary to use column splits when signs are punched over numerical fields. Positions assigned to sign entry by word size wiring recognize only 11 and 12 timed impulses, and positions assigned to numerical entry recognize only 0-9 impulses.

Because signs are entered internally for alphabetic words, the high-order character of an alpha word is wired to the high-order storage-entry position for that word, and the next four character positions are wired to the next four storage-entry positions.

Wiring continues in this manner until the low-order digit (or character) of the last word to be entered is wired. Unless RD+ is *unwired,* there should be no unwired positions in storage entry from position 1 to the last position wired.

If wiring does not adhere to this pattern, invalid information will be transferred to core storage and a validity check will occur. However, if RD+ is unwired, the sign positions of numerical words can be left unwired, and automatic plus signs will be entered in these positions.

The number of the last storage entry position used should be the same as the number wired in ENTRY END. For example, if ENTRY END is wired for 33 characters, the last storage-entry position wired should be 33.

ERROR CONDITIONS: If error conditions in a given format exist as a result of control-panel wiring, one or more of the following can be the cause:

1. Ring Check. Reason—The number specified by entry end wiring does not agree with the number of characters specified by word size wiring.
2. Validity Check. Reason—STORAGE-ENTRY positions to the left of the position corresponding to ENTRY END are left unwired, or blank card columns are read at wired positions.

There are two exceptions to this that do *not* cause a validity check.

a. Unwired sign positions receive automatic plus signs if RD+ is unwired.

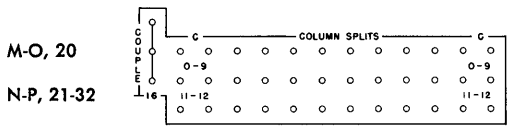
b. Valid coded characters (00) are entered automatically within alphabetic words that are either blank columns or are unwired at both STORAGE ENTRY and ALPHABETIC PRE-READ.

MULTIPLE CARD FORMATS: If two or more card formats are to be read using the same control panel, it is not necessary to select out unused positions to the right of the STORAGE ENTRY position that corresponds to the entry end wiring of the smallest card format, even though unwanted information is punched in these columns. However, word size and entry end wiring must be selected for each format. If word-size wiring does not agree with the specific card format, then the correct fields will not be directed to the desired positions in the synchronizer, and validity checks can occur.

Input Data Flow

Figure 179 shows data flow from the input card to the input synchronizer. Figure 180 is the planning chart for the operation. The card is read as it passes the third reading station. Data enters the 96-position row buffer through control-panel wiring from THIRD READ BRUSHES to STORAGE ENTRY. The wiring from WORD EXITS to WORD SIZE entry controls the number of characters that comprise each storage word. EXIT is wired to UNITS 1 and TENS 9 to indicate a total of 91 characters to be read for this card. Storage-entry hubs 1, 9, 17, 23, 32, 41, 55, 66, 72, and 83 are unwired on the control panel. Because the RD+ switch is unwired, a + sign is automatically placed in these positions of the row buffer. After the row buffer is checked for 91 significant characters, the data is transmitted to the input synchronizer. Unused portions of words 1-16 are automatically filled with zeros.

Control Panel Hubs



COLUMN SPLITS: This device is used to separate 12 and 11 impulses from 0-9 impulses, read from a card at the first, second, or third reading station. During the time that the 12 and 11 are being read from the card, the 12-11 and C hubs of the column split are connected. At 0-9 read-time, the 0-9 and C hubs are connected. An example of the use of column splits is shown in Figure 181.

COLUMN SPLIT COUPLE: An impulse emitted from these hubs may be wired to a co-selector pickup hub to expand the column split feature. The corresponding co-selector can then be used as an additional group of five column splits.

ALPHABETIC CONTROL

Alphabetic data may be read into storage by use of the alphabetic control hubs. As alphabetic information is read at second and third reading, it is compared and

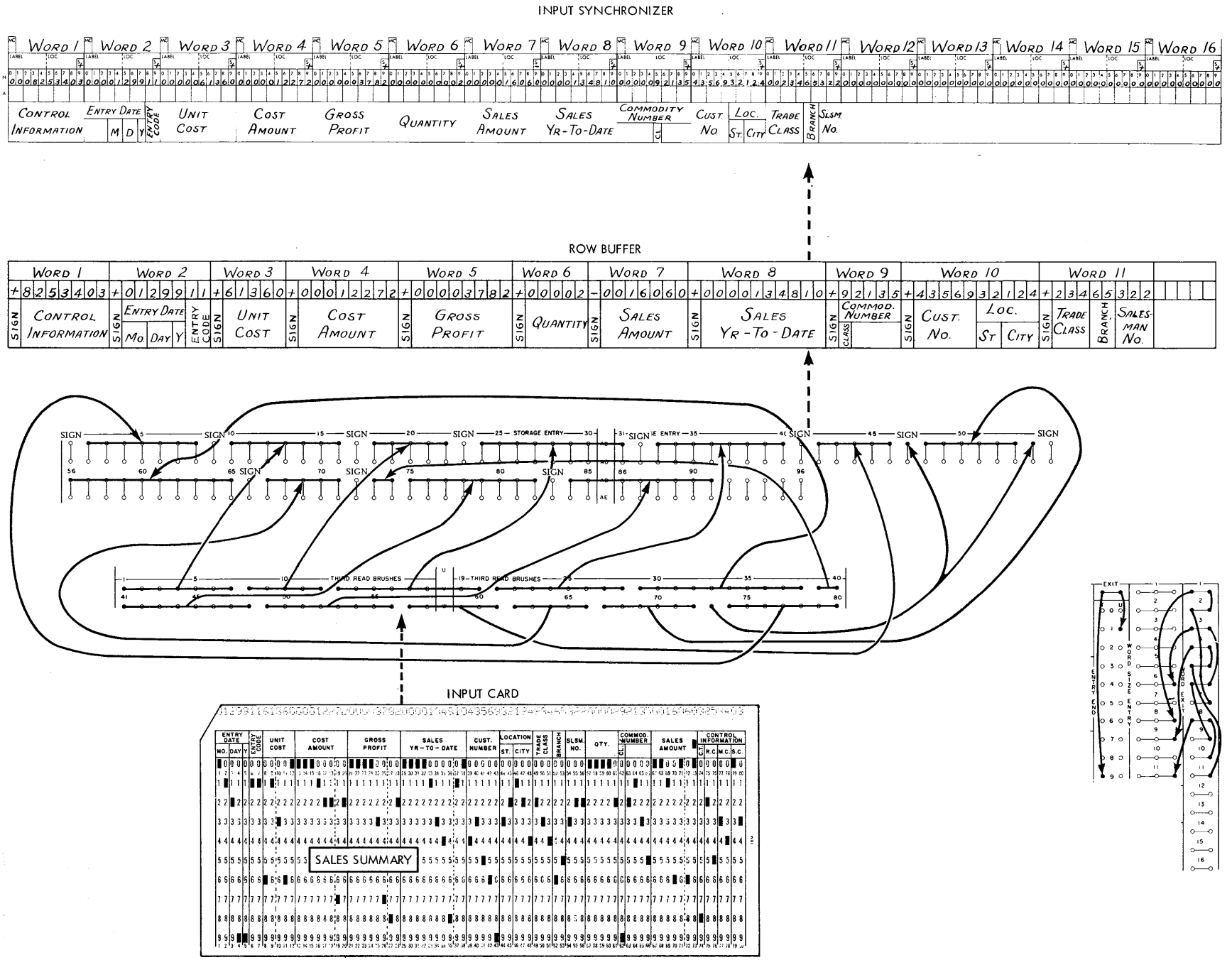


FIGURE 179. INPUT DATA FLOW IBM 7500

FIGURE 180. PLANNING CHART

IBM 7500 READER PLANNING CHART

PREPARED BY: GROUP 1 — DATE: 1-19-59

CARDS USED: SALES SUMMARY — APPLICATION: SALES ACCOUNTING

WORD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	REMARKS
NUMBER OF CHARACTERS TO BE READ (INCLUDE SIGN FOR NUMERIC)	8	8	6	9	9	6	8	11	6	11	9						
ALPHABETICAL IN																	
ALPH PRE-READ																	
CONTROLS																	
FIELD HEADINGS	CONTROL INFOR-MATION	ENTRY DATE, CODE	UNIT COST	COST AMOUNT	GROSS PROFIT	QUANTITY	SALES AMT.	SALES YR. TO D.	COMM. NO.	CUST NO. LOC. ST. CITY	TRADE CL. BRANCH S'MAN						
READING BRUSHES	74-80	1-7	8-12	13-20	21-28	57-61	67-73	29-38	62-66	39-48	49-56						
STORAGE ENTRY	2-8	10-16	18-22	24-31	33-40	42-46	48-54 SIGN 47	56-65	67-71	73-82	84-91						STORAGE ENTRIES 1, 9, 16, 23, 32, 41, 55, 66, 72, AND 83 ARE RESERVED FOR SIGNS OF NUMERIC WORDS.
PILOT SELECTORS	(1) T N C	(2) T N C	(3) T N C	(4) T N C	(5) T N C	(6) T N C	(7) T N C	(8) T N C	(9) T N C	(10) T N C	(11) T N C	(12) T N C	(13) T N C	(14) T N C	(15) T N C		
FIELD SELECTORS	(1) 2 1 N C	(2) 2 1 N C	(3) 2 1 N C	(4) 2 1 N C	(5) 2 1 N C				(6) 3 2 1 N C	(7) 3 2 1 N C	(8) 3 2 1 N C	(9) 3 2 1 N C	(10) 3 2 1 N C				
CO-SELECTORS	(1) T N C	(2) T N C	(3) T N C	(4) T N C	(5) T N C	(6) T N C	(7) T N C	(8) T N C	(9) T N C	(10) T N C	(11) T N C	(12) T N C	(13) T N C	(14) T N C	(15) T N C	(16) T N C	
MISCELLANEOUS	RD+ ☐ (UNWIRED)																

ALTERATION SWITCHES

	A	B		A	B
1	☐	☐	3	☐	☐
2	☐	☐	4	☐	☐

ENTRY END: 91

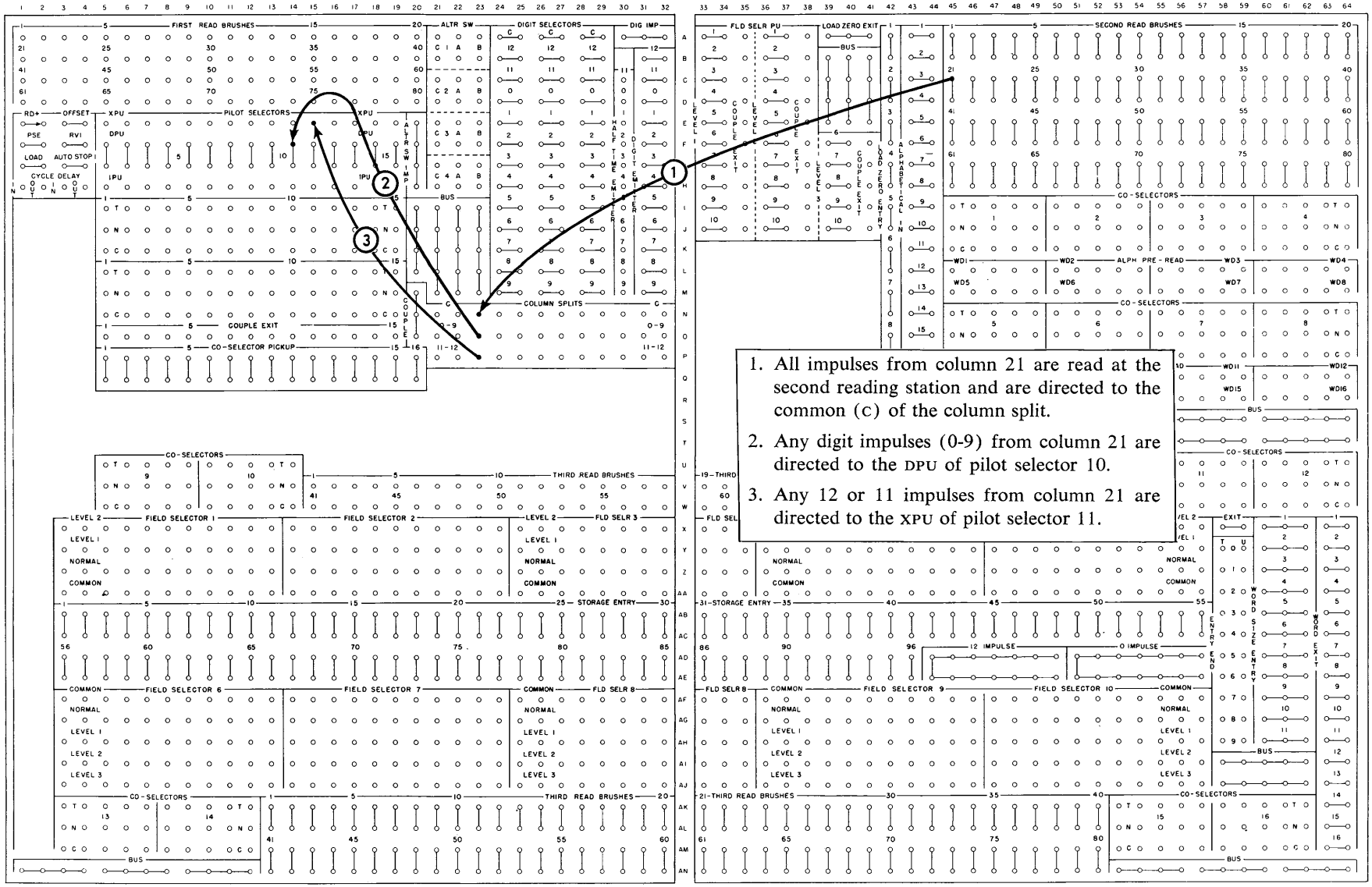
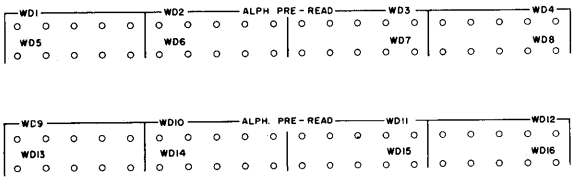


FIGURE 181. COLUMN SPLIT WIRING

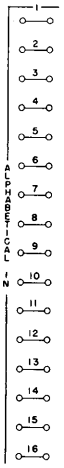
checked. The data read at second reading are stored and compared with the same data when the card advances past the third reading station. Each storage word can contain a maximum of five alphabetic characters. The sign is *not* counted as a character to be read but is automatically entered in the input synchronizer.



L-M, 45-64 Q-R, 45-64

ALPHA PRE-READ: Five hubs are provided for each of the sixteen words that may be alphabetic. The same alphabetic format that is wired to STORAGE ENTRY from third reading must be wired to ALPH PRE-READ from second reading. As a block is transferred from the input synchronizer to general storage a character-by-character comparison is made between alphamerical words read at second and third reading.

A-P, 43-44



ALPHABETICAL IN: These hubs are normally wired from the CAI hubs to activate the necessary translating circuitry for alphabetic conversion in the 7070. The combination zone and digit punches in the card are converted to the 2-digit representation used to store alphabetic information. Impulsing this hub also causes an alphabetic sign to be entered into the sign position of the designated word.

Q, 43-44

CAI (CONSTANT ALPHABETIC IMPULSE)

1. *D-Detail*—This hub emits an impulse on all read cycles except load cycles. This is a selectable impulse normally wired to ALPHABETICAL IN.

2. *L-Load*—This hub emits an impulse on load cycles only. This is a selectable impulse normally wired to ALPHABETICAL IN.

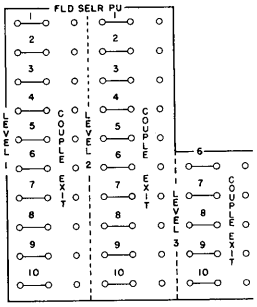
Figure 182 shows the wiring for entering alphabetic words into storage. Words 2-5 are alphabetic. Word 6 is a numerical word.

Selection

FIELD SELECTORS

Several different types of cards may be used as input to the 7070, without changing control panels, by use of the ten field selectors on the 7500.

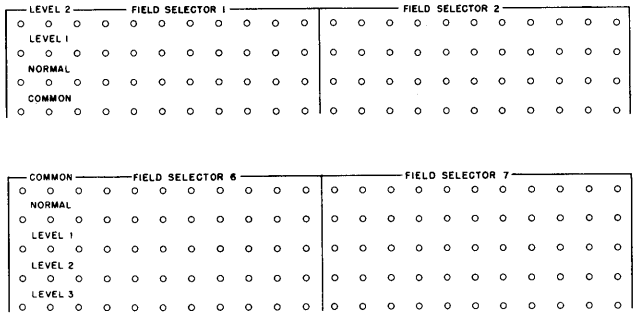
A-J, 33-34
A-J, 35
A-J, 36-37
A-J, 38
F-J, 39-40
F-J, 41



FIELD SELECTOR PICKUP LEVELS 1, 2, AND 3: These hubs are used to impulse the corresponding level of a field selector to transfer on the following read feed cycle. They are entries for any read-timed digit impulses (9-12) and are wired from first reading to cause the selector to transfer when that card passes second reading, or from second reading to transfer the selector for third reading.

Note: The higher-numbered levels have selective priority. That is, if both levels 1 and 2 are impulsed to pick up on the same cycle, then only level 2 is connected internally to common.

FIELD SELECTOR COUPLE EXIT: These hubs emit an impulse when the corresponding level is active (transferred). They are normally used to pick up co-selectors or pilot selectors to expand the capacity of the field selectors.



X-AA, 3-57 AF-AJ, 3-57

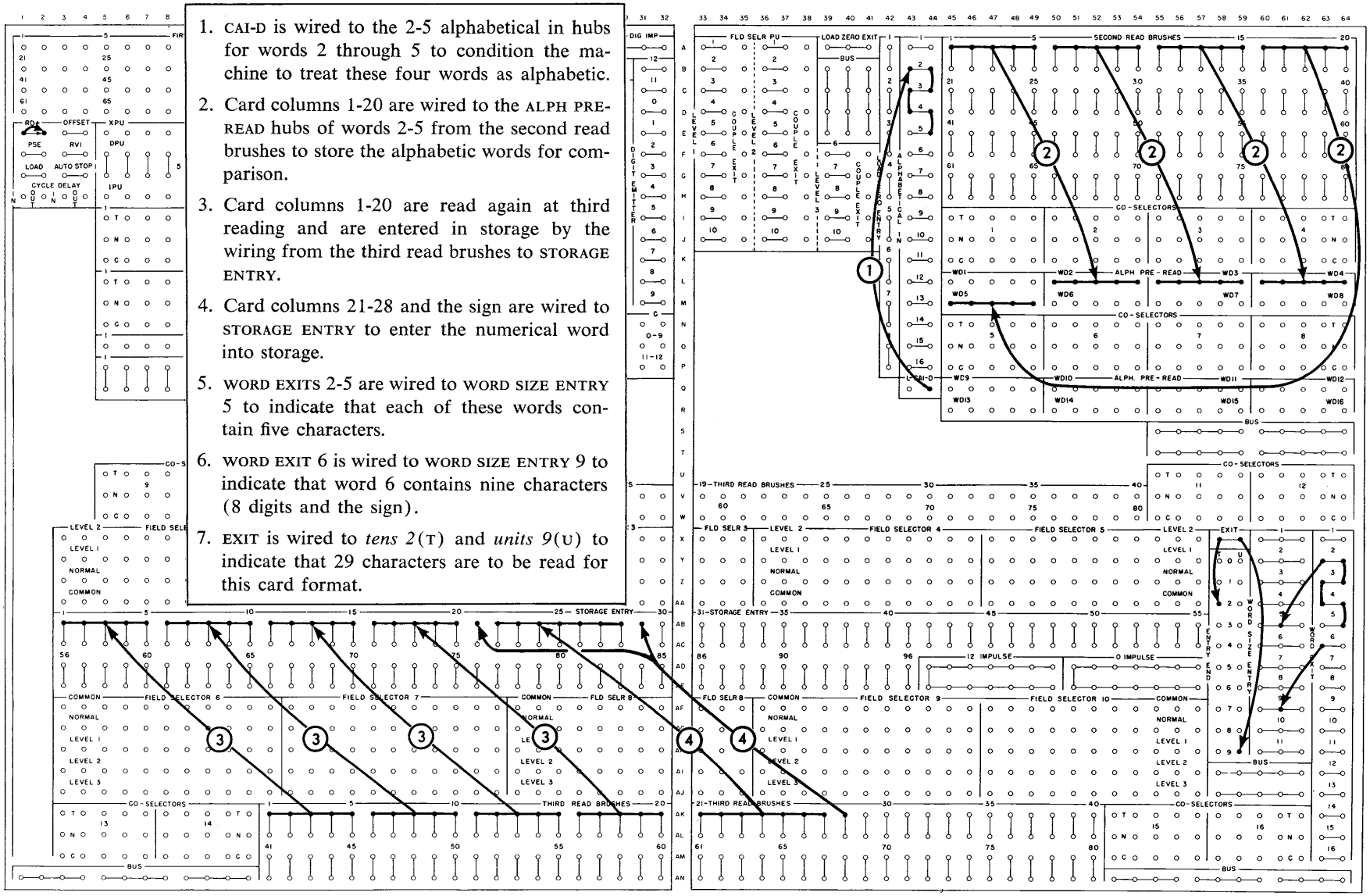
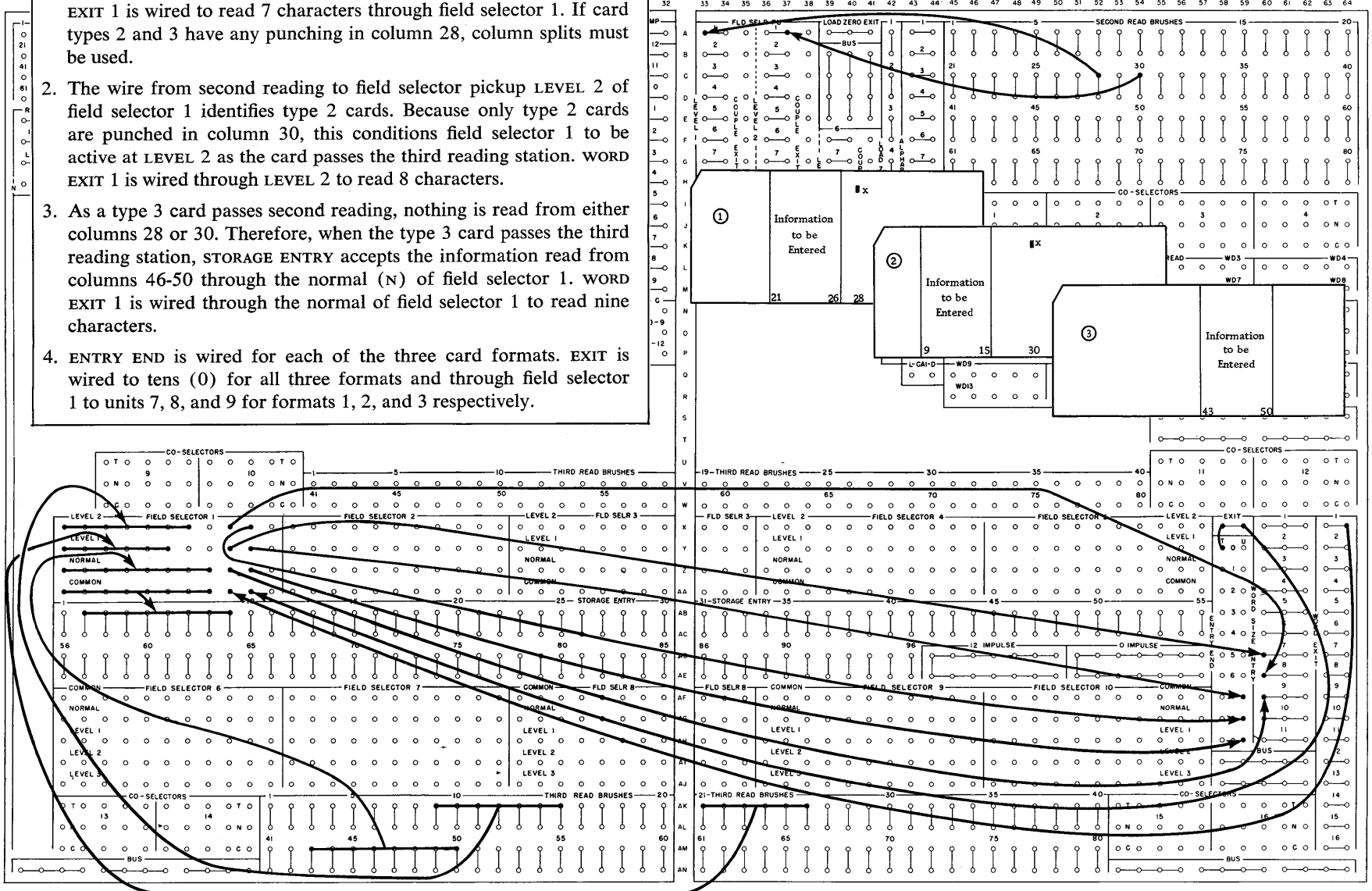


FIGURE 182. ALPHABETIC CONTROL WIRING

FIGURE 183. FIELD SELECTION

1. The wire from second reading column 28 to field selector pickup LEVEL 1 of field selector 1 detects cards with punching in column 28 and identifies them as type 1 cards. This conditions field selector 1 to be active at level 1, and STORAGE ENTRY accepts data wired through LEVEL 1 as this card passes the third reading station. WORD EXIT 1 is wired to read 7 characters through field selector 1. If card types 2 and 3 have any punching in column 28, column splits must be used.
2. The wire from second reading to field selector pickup LEVEL 2 of field selector 1 identifies type 2 cards. Because only type 2 cards are punched in column 30, this conditions field selector 1 to be active at LEVEL 2 as the card passes the third reading station. WORD EXIT 1 is wired through LEVEL 2 to read 8 characters.
3. As a type 3 card passes second reading, nothing is read from either columns 28 or 30. Therefore, when the type 3 card passes the third reading station, STORAGE ENTRY accepts the information read from columns 46-50 through the normal (N) of field selector 1. WORD EXIT 1 is wired through the normal of field selector 1 to read nine characters.
4. ENTRY END is wired for each of the three card formats. EXIT is wired to tens (0) for all three formats and through field selector 1 to units 7, 8, and 9 for formats 1, 2, and 3 respectively.



FIELD SELECTOR LEVELS 1, 2, AND 3—NORMAL AND COMMON: There are ten, 11-position field selectors provided. Field selectors 1-5 have 3 levels each (NORMAL, LEVEL 1 AND 2); selectors 6-10 have 4 levels each NORMAL, LEVEL 1, AND 2, AND LEVEL 3).

NOTE: The vertical arrangement of the four level field selectors 6-10 is the opposite of the arrangement of the three level field selectors 1-5.

In Figure 183, there are three types of cards to be entered into the system: Card 1 is identified by an X in column 28, card 2 is identified by an X in column 30, card 3 is identified by the absence of X's in both 28 and 30. From card 1, the information comes from card columns 21-26. From card 2, the information comes from columns 9-15, and from card 3, the information comes from columns 43-50.

PILOT SELECTORS AND CO-SELECTORS

Pilot and co-selectors are provided to increase the flexibility of input format. Each individual position of a selector is a switch composed of three hubs labeled T, N, C. The C *(common)* hub is always connected to either the N *(normal)* or T *(transferred)* hub, but never to both (Figure 184). Thus, impulses entering C are selectively available at either N or T, depending upon whether the pickup hubs have received an impulse. Conversely, impulses entering either N or T are available at C under control of the impulses to the pickup hubs.

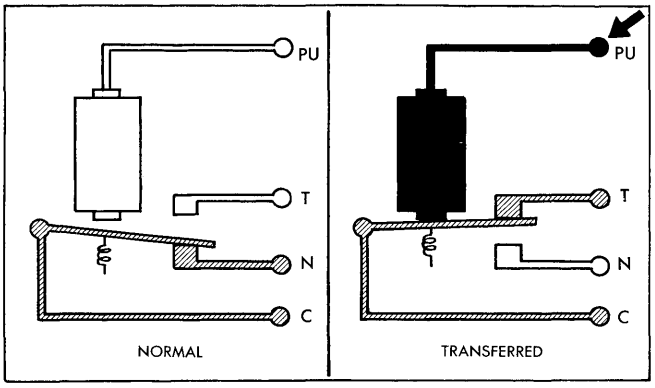
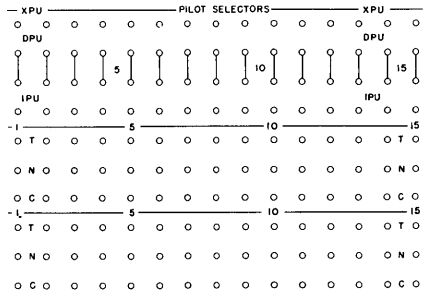


FIGURE 184. SELECTOR SCHEMATIC

Pilot selectors have multiple pickup hubs to meet varying needs. They can be impulsed on one cycle to transfer on the following cycle, or they can be impulsed and transferred on the same cycle. Pilot selectors hold for the cycle on which they transfer and remain transferred for that cycle only. Co-selectors have only one pickup. They transfer immediately when impulsed. If more flexible operation is desired, they can be coupled to the pilot selectors by wiring the couple exit of the pilot selector to the pickup of a co-selector. In this way, a co-selector is used to increase the size of the pilot selector.

E-N, 5-19



PILOT SELECTOR PICKUPS. Three types of pilot selector pickups are provided.

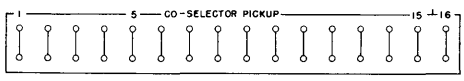
1. XPU. These hubs accept both 12 or 11 read impulses, to cause the corresponding selector to transfer on the following read cycle. There is a one-cycle delay in the transfer of the pilot selector (under automatic control of the read feed).
2. DPU. These hubs accept any read-digit impulse (12-9) to cause the corresponding selector to transfer on the following read cycle.
3. IPU. These hubs are entries for impulses timed to the read feed. An impulse entering one of these hubs causes the associated pilot selector to transfer immediately.

COUPLE EXIT: The impulse from these hubs is available at the beginning of the read cycle following the impulsing of the corresponding XPU or DPU. This impulse is usually used to control co-selectors or other pilot selectors.

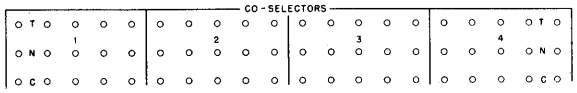
Be careful when using both XPU and IPU of the same selector as pickups. If both receive impulses during the same cycle, it can cause other pilot selectors receiving impulses into XPU or DPU during the same cycle to transfer one cycle early. The same precaution must be observed if both DPU and IPU of the same selectors are used as inputs.

PILOT SELECTORS: Fifteen pilot selectors are standard on the 7500. Each pilot selector has two switching positions.

P-Q, 5-20



CO-SELECTOR PICKUP: These hubs accept any impulse. The associated selector transfers immediately when an impulse reaches either one of these common pickup hubs.

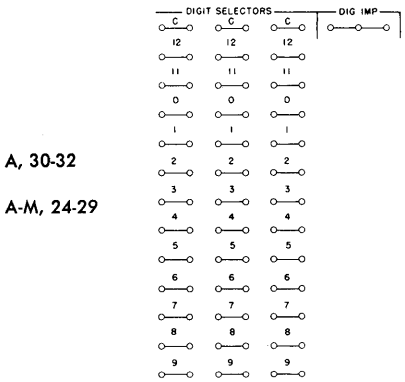


CO-SELECTORS: Sixteen co-selectors are standard on the 7500. Each co-selector has five positions.

Figure 185 illustrates a basic use of pilot and co-selectors. The same three cards used in Figure 183 are entered into the system without using the field selectors.

Digit Selection

The three cards shown in Figures 183 and 185 can be identified by different digit punching in the same card column. These digits can serve the same function that the separate X-punching did. The individual digits could be selected by using a digit selector.

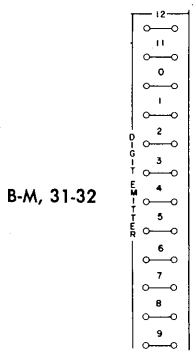


DIGIT SELECTORS: Three read-digit selectors are standard on the 7500. Digit selectors can be used to select specific digits from a card column, or to emit digits on read cycles. When a card column is wired to C (common), a specific digit impulse can be read from the corresponding hub of the digit selector. Thus, each digit that is punched is available for control purposes. If the DIG IMP (Digit Impulse) hub is wired to C, specific digit impulses are available from each of the exit hubs on each card feed cycle and the digit selector acts as a digit emitter.

In the example shown in Figure 186, the card types are identical to those used in Figures 183 and 185. The only difference in the cards is the identifying punching. For this example, card 1 is identified by a 3 in column 65, card 2 by a 4 in column 65, and card 3 by the absence of both 3 and 4 punches. All other conditions of Figures 183 and 185 remain the same. Therefore, only the necessary selection wiring is shown.

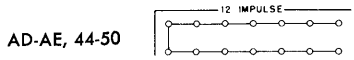
Emitted Impulses

Data can be supplied by the control panel for entry into the 7070 system. These emitted factors can be used for filling in zeros, supplying sign impulses, etc. Emitted impulses can come from the hubs labeled digit emitter or the digit selector wired as a digit emitter.

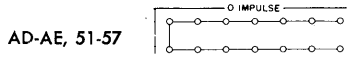


DIGIT EMITTER: These hubs emit read-digit impulses (12-9).

The example in Figure 187 shows emitting and selecting information from either the digit emitter or the digit selector wired as a digit emitter. For cards with an X in column 24, an 8 is entered into the second position of STORAGE ENTRY; for cards without an X in column 24 a 9 is entered into the second position of STORAGE ENTRY.



12 IMPULSE: These hubs emit twelve timed impulses that are normally wired to storage-entry sign positions when 12's are not punched over certain positive fields in the card. These impulses are available on all read cycles regardless of the setting of the RD+ switch.



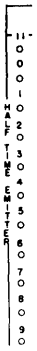
0 IMPULSE: These hubs emit zero timed impulses that are normally wired to storage-entry positions when zero impulses are not available from the card. These hubs emit impulses on all cycles.

Split Column Control

Multiple punching in one column can be split between any two punching positions by use of the half-time emitter. It differs from normal column split (which is always between 0 and X time in the card) in that the split can be between 9 and 8, 0 and X, etc. There is a

hub for all the digits 9-0 and for 11. They emit impulses at half after the number indicated. For example, 9 emits an impulse between 9 and 8; 0 emits an impulse between 0 and X time.

B-M, 30



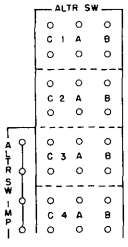
HALF-TIME EMITTER: These hubs are normally wired to a selector pickup (immediate) making the selector operate the same as a column split device. For example, if a half-after 1 is wired to a selector I (immediate) pickup, the C and N hubs of the selector are common from 9 through 1 time, and the C and T hubs are common from 0 through 12 time. Thus, zone impulses may be separated from digit impulses punched in the same column. If half-after 0 is wired to the selector I pickup, the C and N hubs are common from 9 through 0, and the C and T hubs are common for 11 and 12. This arrangement is the same as a normal column split. A column can be split between 11 and 12 by using the 11 half-after impulses as a pickup for the selector (Figure 188).

Alteration Switches

Within reasonable limitations, one control panel can be used for several different card formats without any change in control-panel wiring, by use of the alteration switches.

A-H, 21-23

E-H, 20



ALTR SW (ALTERATION SWITCH) 1, 2, 3, 4: Four alteration switches are located on the left front of the machine. Each alteration switch controls a corresponding two-position alteration switch selector. When the switch is set on A, a common connection exists between C and A; when the switch is set on B, a common connection exists between C and B. An alteration switch selector remains transferred as long as the corresponding alteration switch is set on B. Any impulse can be wired through C-A or C-B.

ALTR SW IMP (ALTERATION SWITCH IMPULSE): These hubs emit an impulse that is normally wired through an alteration switch selector to the D or I pickup of pilot selectors or to the pickup of co-selectors or field selectors. A selector impulsed through the A side of an alteration switch selector transfers each card cycle as long as the corresponding alteration switch is set to A. If a delay pickup is used (DPU of a pilot selector or PU of a Field Selector), the selector will transfer one card cycle after the switch is set to A and for one additional cycle after it is returned to B. An example of alteration switch is shown in Figure 189. In this illustration, two types of cards are run separately. Card type 1 has 12 punches over the units position of field B for positive words, and X punches over the units position of field B for negative words (column 55). Card type 2 has X punches over the units position of field B for negative words only. Type 1 cards are processed with alteration switch 2 set on B. Type 2 cards are processed with alteration switch 2 set on A.

Loading

Another method of input from the third reading station to the input-synchronizer is available on the IBM 7500. This method bypasses the control-panel wiring and sets up a direct internal connection between the third reading station and the input synchronizer. This direct connection is under control of the LOAD hubs.

G, 1-2

LOAD: These hubs accept only a read 12 impulse. When this hub is wired, internal wiring takes precedence over any external storage entry wiring. Card columns 1-80 are read automatically into read storage words 1-8. The LOAD hub is wired from first reading, RD+ is automatically on and words 9-16 are entered automatically as plus zeros. If a load card contains alphabetic information, the CAI LOAD hub is wired to alphabetical in, and conventional wiring to the ALPH PRE-READ hubs is necessary (Figure 190).

When the CAI impulse is selected, it is normally wired through the transferred points of a pilot selector that has been impulsed from first reading at the X- or D-pickup. The wiring from second reading to ALPH PRE-READ can be selected through field selectors impulsed from first reading;

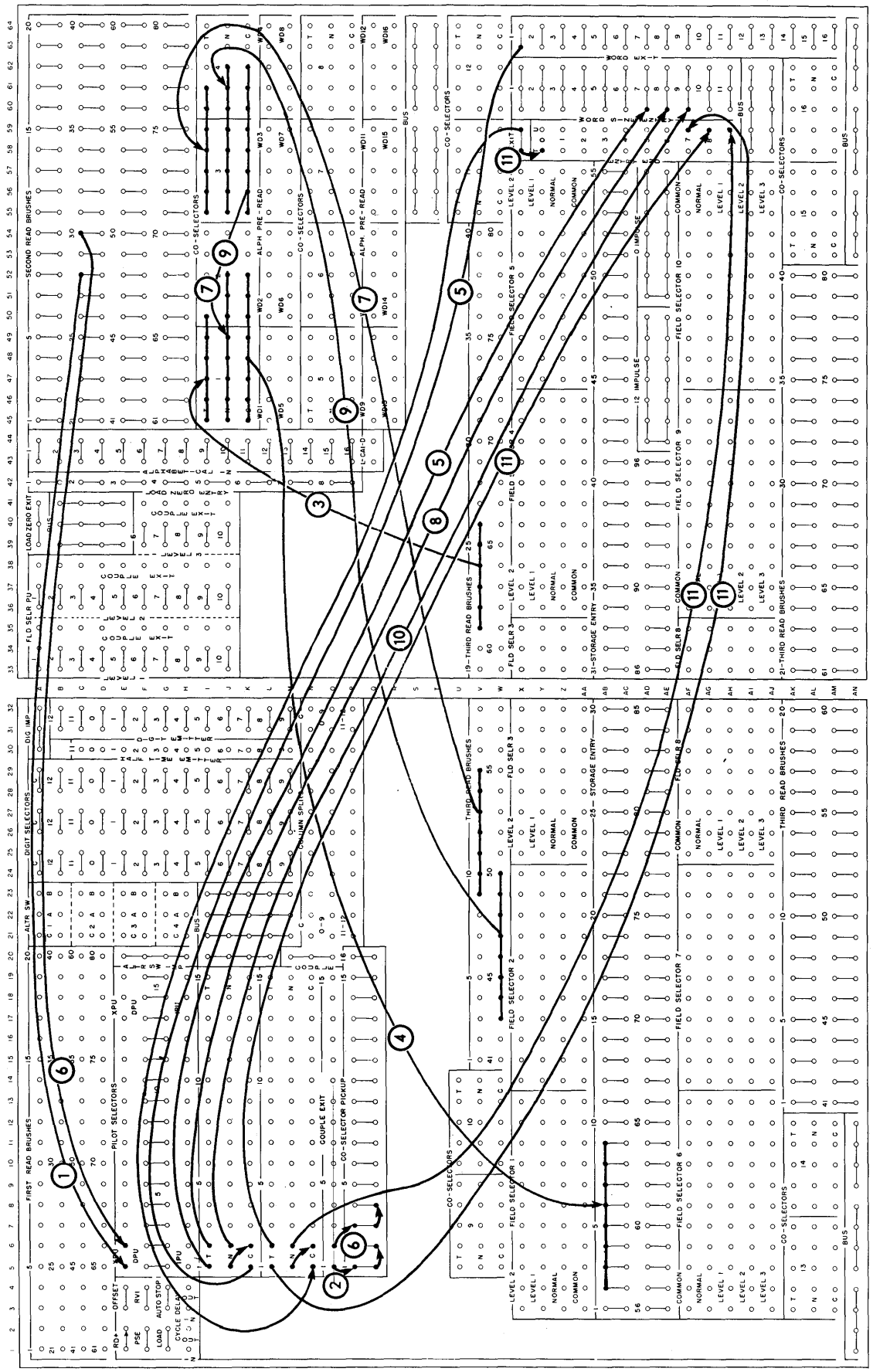
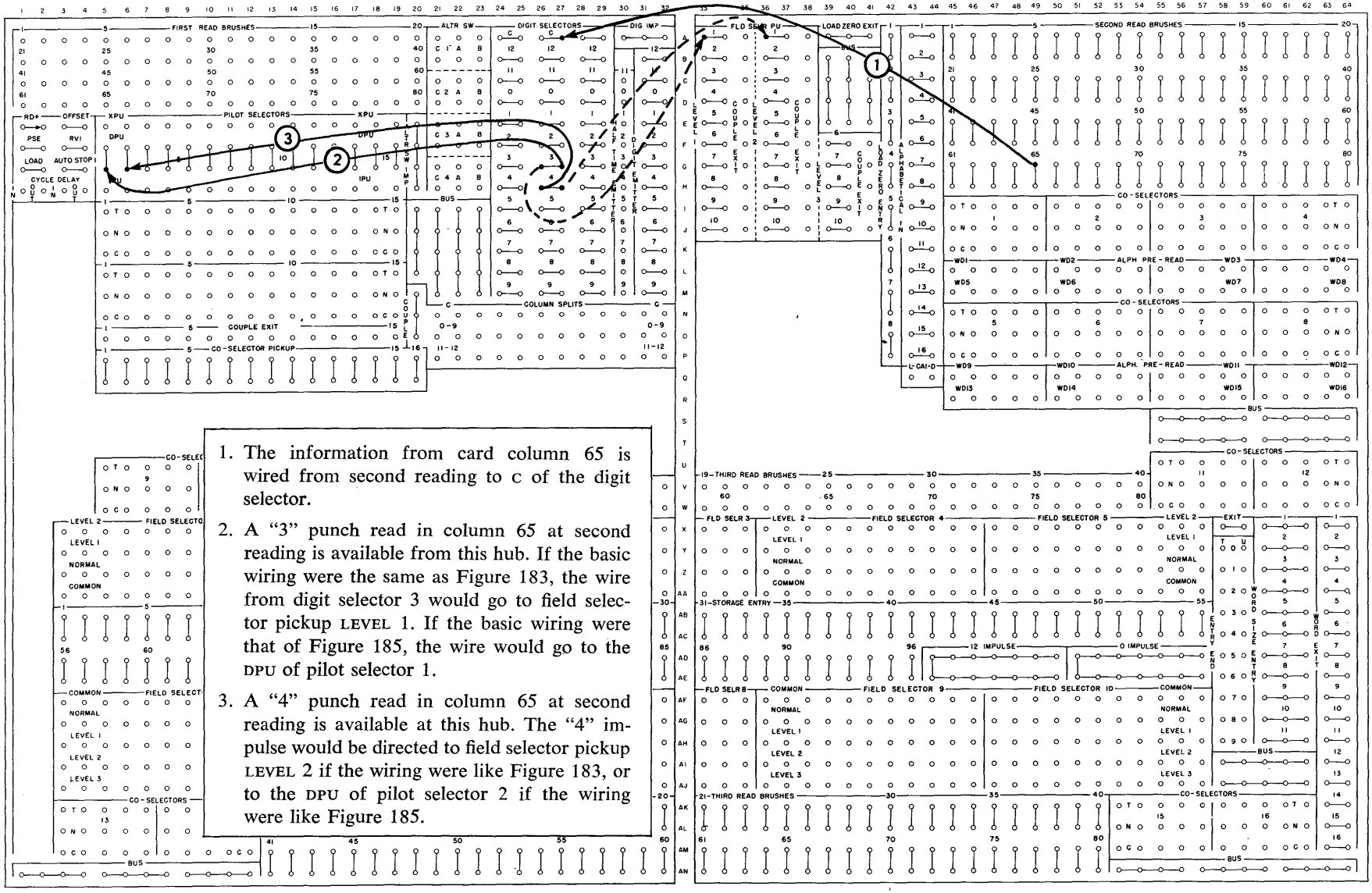


FIGURE 185. PILOT AND CO-SELECTOR OPERATION

1. Pilot selector 1 is impulsed by an X in column 28 from the second reading station. This identifies the type 1 card. Pilot selector 1 transfers on the following cycle as the type 1 card passes the third reading station.
2. Co-selectors 1 and 2 are transferred, during the same cycle that pilot selector 1 is transferred, by an impulse from the couple exit of pilot selector 1.
3. The information from card columns 21-26 is wired to the transferred side of co-selectors 1 and 2.
4. The common side of co-selectors 1 and 2 is wired to storage-entry positions 2-9. Because at this time co-selectors 1 and 2 are transferred, the information from card columns 21-26 is entered into word 1. A plus sign is entered into entry position 1 because RD+ is not wired.
5. WORD EXIT 1 is wired to WORD-SIZE ENTRY 7 through the transferred side of pilot selector 1. This pilot selector is transferred at the time the X-28 card passes the third reading station.
6. An X in column 30 of a card passing the second reading station impulses pilot selector 2. This causes this pilot selector to transfer one cycle later as the card passes the third reading station. The impulse from the couple exit of pilot selector 2 is used to pick up co-selectors 3 and 4. Co-selectors 3 and 4 are picked up on the cycle during which the X-30 card passes the third reading station.
7. The information from card columns 9-15 is wired to the transferred side of co-selectors 3 and 4. With co-selectors 3 and 4 transferred, this information is available at the common side of these co-selectors, and it is wired to the normal side of co-selectors 1 and 2. Co-selectors 1 and 2 should be normal when co-selectors 3 and 4 are transferred. This makes the information from card columns 9-15 available through the common side of co-selectors 1 and 2 to storage entry.
8. WORD EXIT 1 is wired to WORD SIZE ENTRY 8 through the normal side of pilot selector 1 and the transferred side of pilot selector 2.
9. The information from card columns 43-50 is wired to the normal side of co-selectors 3 and 4. When a card that is neither X-28 nor X-30 passes the third reading station, the information passes from columns 43-50, through co-selectors 3 and 4 normal, to co-selectors 1 and 2 normal, to storage entry.
10. WORD EXIT 1 is wired to WORD SIZE ENTRY 9 through the normal side of pilot selectors 1 and 2.
11. ENTRY END is wired for all three formats. EXIT is wired to tens 0, and to units 7, 8, and 9 for card types 1, 2, and 3 respectively.



1. The information from card column 65 is wired from second reading to c of the digit selector.
2. A "3" punch read in column 65 at second reading is available from this hub. If the basic wiring were the same as Figure 183, the wire from digit selector 3 would go to field selector pickup LEVEL 1. If the basic wiring were that of Figure 185, the wire would go to the DPU of pilot selector 1.
3. A "4" punch read in column 65 at second reading is available at this hub. The "4" impulse would be directed to field selector pickup LEVEL 2 if the wiring were like Figure 183, or to the DPU of pilot selector 2 if the wiring were like Figure 185.

FIGURE 186. DIGIT SELECTION

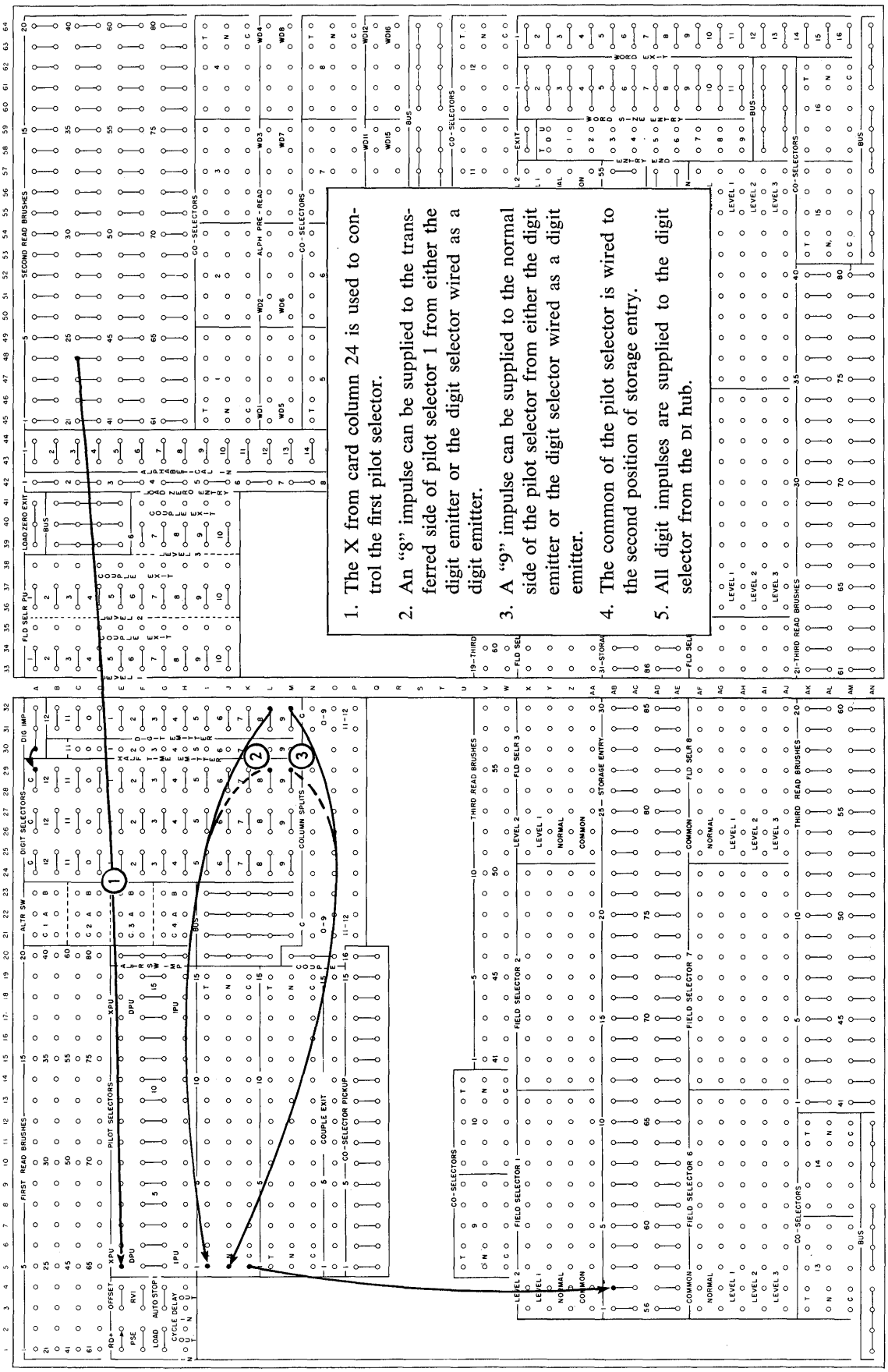


FIGURE 187. DIGIT EMITTING

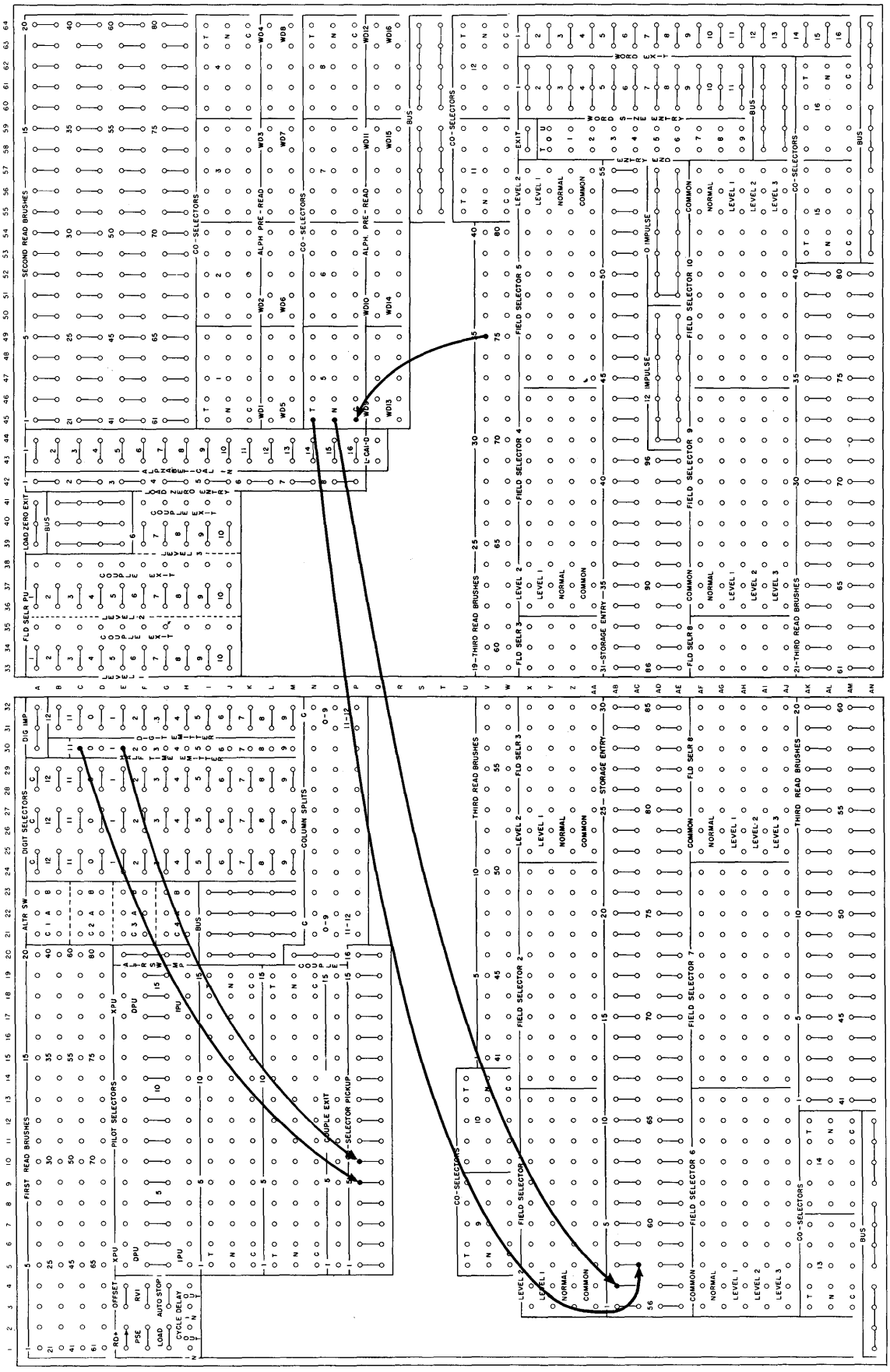
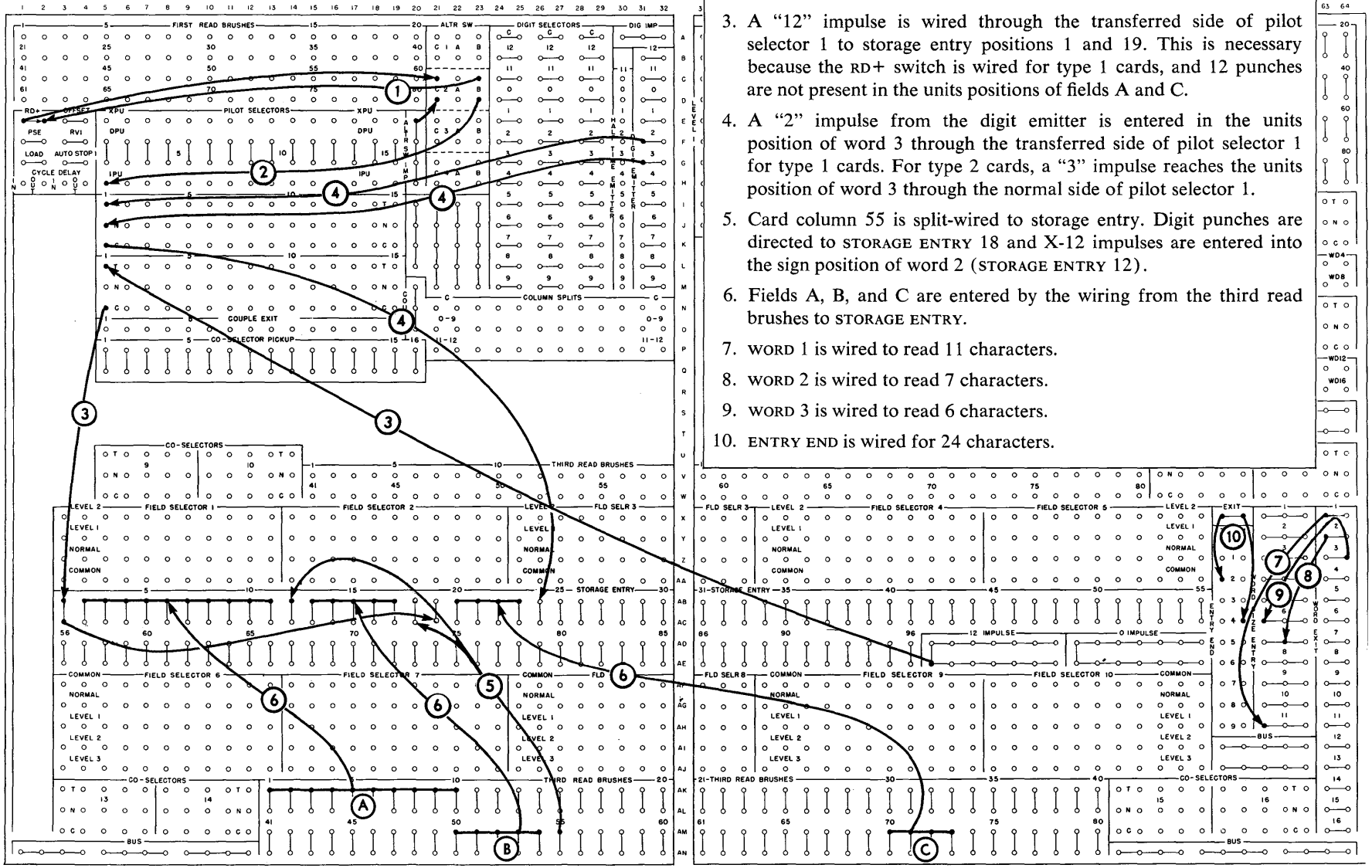
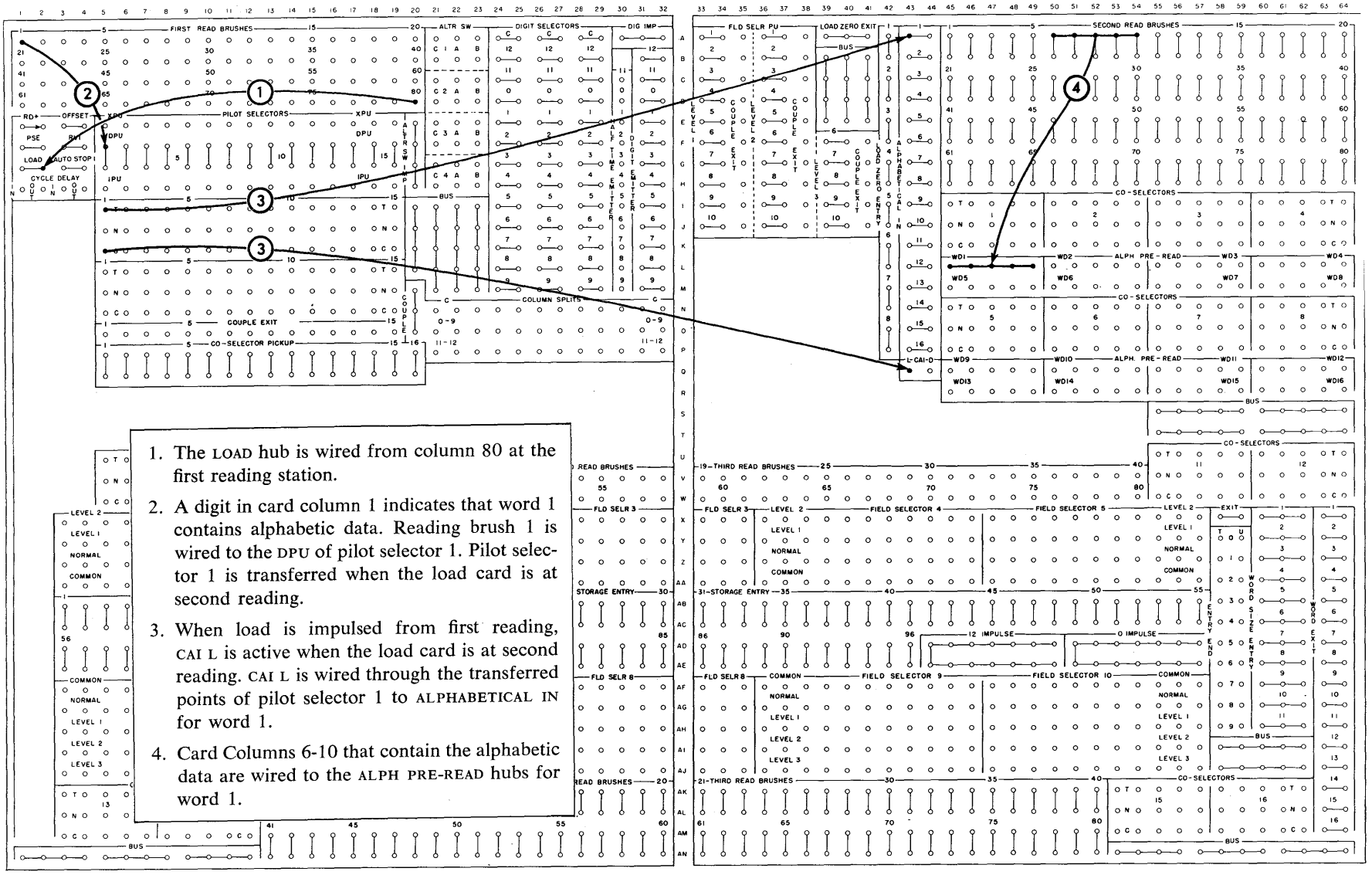


FIGURE 188. SPLIT COLUMN CONTROL USING THE HALF-TIME EMITTER



1. The RD+ switch is connected for type 1 cards through the B side of alteration switch selector 2. For type 2 cards the switch is unwired.
2. The alteration switch impulse is wired through the B side of alteration switch selector 2 to the D pickup of pilot selector 1. Pilot selector 2 is transferred when the alteration switch is set on B.
3. A "12" impulse is wired through the transferred side of pilot selector 1 to storage entry positions 1 and 19. This is necessary because the RD+ switch is wired for type 1 cards, and 12 punches are not present in the units positions of fields A and C.
4. A "2" impulse from the digit emitter is entered in the units position of word 3 through the transferred side of pilot selector 1 for type 1 cards. For type 2 cards, a "3" impulse reaches the units position of word 3 through the normal side of pilot selector 1.
5. Card column 55 is split-wired to storage entry. Digit punches are directed to STORAGE ENTRY 18 and X-12 impulses are entered into the sign position of word 2 (STORAGE ENTRY 12).
6. Fields A, B, and C are entered by the wiring from the third read brushes to STORAGE ENTRY.
7. WORD 1 is wired to read 11 characters.
8. WORD 2 is wired to read 7 characters.
9. WORD 3 is wired to read 6 characters.
10. ENTRY END is wired for 24 characters.

FIGURE 189. ALTERATION SWITCH WIRING



1. The LOAD hub is wired from column 80 at the first reading station.
2. A digit in card column 1 indicates that word 1 contains alphabetic data. Reading brush 1 is wired to the DPU of pilot selector 1. Pilot selector 1 is transferred when the load card is at second reading.
3. When load is impulsed from first reading, CAI L is active when the load card is at second reading. CAI L is wired through the transferred points of pilot selector 1 to ALPHABETICAL IN for word 1.
4. Card Columns 6-10 that contain the alphabetic data are wired to the ALPH PRE-READ hubs for word 1.

FIGURE 190. LOADING ALPHABETIC DATA

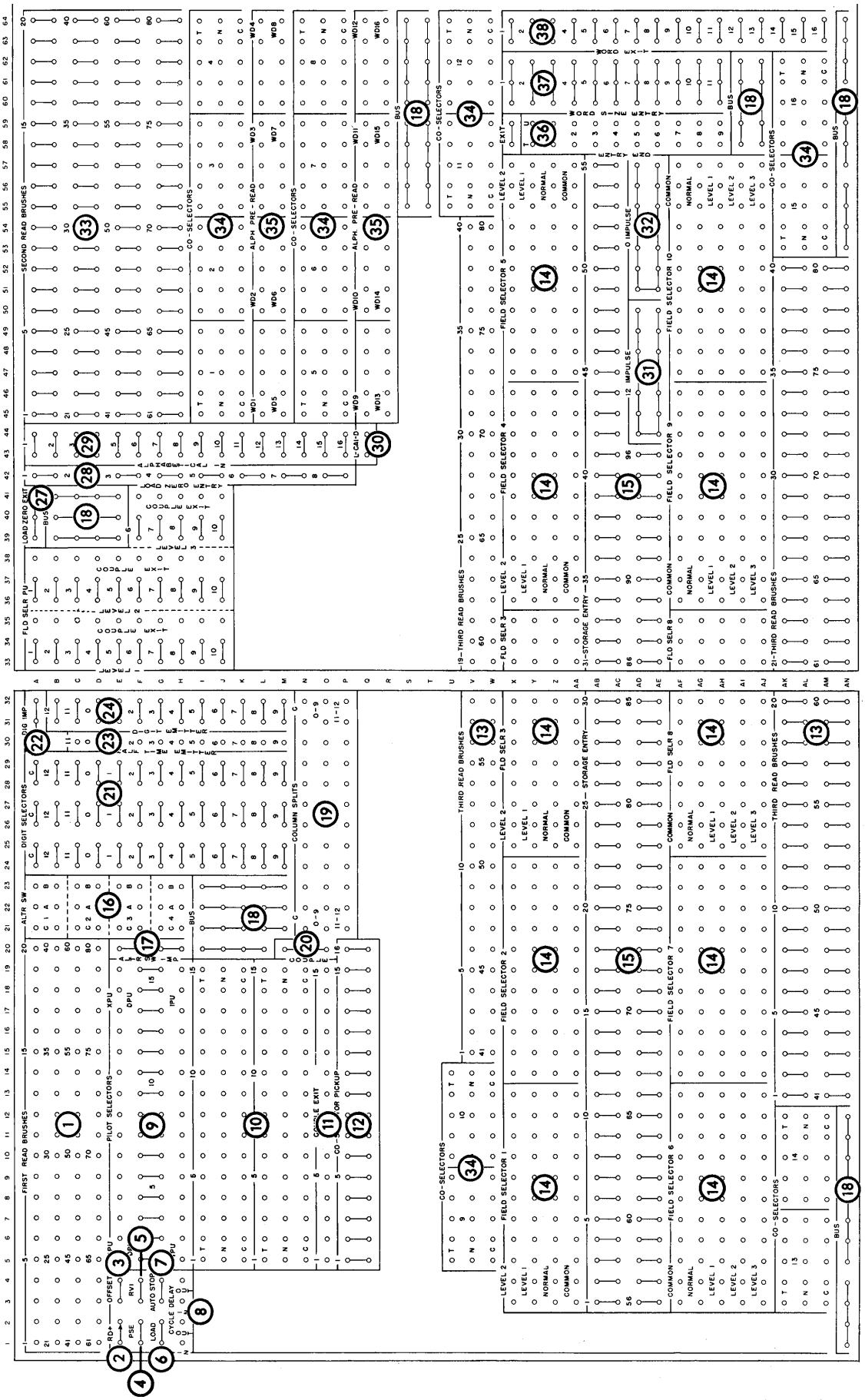


FIGURE 191. KEY TO CONTROL PANEL SUMMARY

or through co-selectors impulsed to pickup from the coupling exit of a pilot selector that has been impulsed at X- or D-pickup from first reading.

Specifications for punching load cards are as follows:

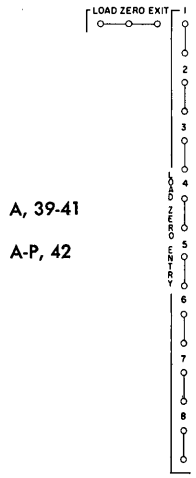
1. The sign (whether plus or minus) is punched over the units position of a numerical word. (Alpha signs are entered automatically by the wiring from CAI to ALPHABETICAL IN)
2. Words 1-8 are punched in the load cards as follows:

Word	Card Columns
1	1-10
2	11-20
3	21-30
4	31-40
5	41-50
6	51-60
7	61-70
8	71-80

If a word is alphabetic, the data are punched in the five low-order positions of the card columns assigned to that specific word.

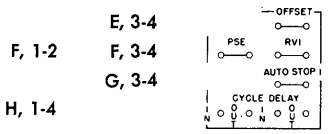
Zeros must be punched in blank positions of words containing numerical data.

Normally, zeros are punched in all ten positions of unused words. If they are not punched, they can be entered through control-panel wiring of the LOAD ZERO EXIT and LOAD ZERO ENTRY hubs.



LOAD ZERO ENTRY: These hubs are assigned to the eight input words that are active on load cards. If any word in a load card contains invalid data, the LOAD ZERO ENTRY hubs associated with that word must be wired from the LOAD ZERO EXIT hubs.

Other Control-Panel Hubs



RVI (READ VALIDITY IMPULSE): These hubs emit an impulse if the read validity check circuits detect an invalid character when data is transferred from synchronizer to core storage. If these hubs are unwired, the detection of a non-valid character causes the validity check light on the reader to flash. The RVI impulse is normally wired to OFFSET or AUTO STOP.

PSE (PROGRAM SWITCH EXIT): This exit hub is directly under control of the 7070 US command. PSE emits an early-timed impulse (prior to 9 time) on any read cycle if its associated switch is ON. The switch is interrogated at the time the read command is given. If the switch is ON, the PSE hub emits and resets the switch to OFF. The PSE hubs are normally used to pick up selectors, impulse OFFSET, or AUTO STOP.

AUTO STOP: An impulse directed to these hubs causes the reader to stop at the end of that read cycle and removes the reader from a ready status.

OFFSET: These hubs control the offset card-stacking device on the card reader. They must be impulsed early in the cycle in which the card is stacked. The OFFSET hubs accept an impulse from a pilot selector coupling exit, cycle delay out, etc.

CYCLE DELAY IN, OUT: An early-timed impulse wired to an In hub of the cycle delay unit causes an impulse to be available from the corresponding Out hub on the following cycle. These hubs are normally used to delay impulses for one read feed cycle for impulsing the offset stacker, picking selectors, etc.

7500 Card Reader Control-Panel Summary

1. *First Read Brushes.* These are exit hubs (Figure 191) for impulses originating as the card is read at the first reading station. These hubs are used to pick up selectors for alphabetic control, loading, etc.
2. *RD+ Read Plus Sign.* This switch must be connected when input cards are punched with 12's over positive numerical fields.

3. *Offset.* These hubs are impulsed early in the cycle in which a card is stacked. They are normally wired from cycle delay out, pilot selector coupling exit, etc.
4. *PSE—Program Switch Exit.* This hub is controlled by the 7070 program, US command. It emits an early-timed impulse (prior to 9 time) on any read cycle if its associated switch is ON. PSE hubs are normally used to pick up selectors, impulse OFFSET or AUTO STOP.
5. *RVI—Read Validity Impulse.* These hubs emit when the validity check circuits detect the reading of an invalid character. They are usually wired to the auto-stop or offset hubs.
6. *Load.* These hubs accept a 12 impulse from first reading to cause the machine to override external control-panel wiring. When load has been impulsed, the contents of the eighty columns in the card are transferred automatically to storage words 1-8.
7. *Auto Stop.* These hubs accept impulses to suspend reader operation at the end of that read cycle.
8. *Cycle Delay.* An impulse wired to the in hub of the cycle delay unit causes an impulse to emit from the corresponding out hub one cycle later.
9. *Pilot Selectors XPU, DPU, IPU*
   a. XPU. These hubs accept 11 or 12 read impulses to transfer the selector on the following read cycle.
   b. DPU. These hubs accept any read-digit impulse (9-12) to cause the selector to transfer on the following read feed cycle.
   c. IPU. These hubs accept any read-digit impulse (9-12) to cause the selector to transfer immediately.
10. *Pilot Selectors, T,N,C.* Whenever a pilot selector is transferred, an impulse wired to the C (common) of the selector is available from the corresponding T (transferred) hub. If the selector has not been impulsed at the pickup hub, an impulse wired to the C hub is available at the N (normal) hub.
11. *Couple Exit.* These hubs emit an impulse whenever the corresponding pilot selector is transferred. This impulse is normally used to control other pilot selectors or co-selectors.
12. *Co-Selector Pickup.* These hubs accept any read-timed impulse to transfer the co-selector on the same cycle (immediately).
13. *Third Read Brushes.* These are the 80 exit hubs for impulses read from the card at the third reading station. They are normally wired to storage entry directly or through properly controlled selectors.
14. *Field Selectors.* There are ten 11-position field selectors provided. Field selectors 1-5 have three levels each: (Normal, level 1 and level 2). Field selectors 6-10 have four levels each: (Normal, level 1, level 2, and level 3). When a field selector is impulsed to transfer at a given level, a connection exists between common and that level. If the field selector has not been impulsed, the connection is between common and normal.
15. *Storage Entry.* These are the 96 entry hubs to the input-synchronizer. They are normally wired from the third reading brushes to enter data in any desired format. Each wired storage entry position must receive a valid character from the reading brushes. In an alphabetic word only, a blank is a valid character. The ring check light comes on if less characters are read than the number indicated by the wiring to ENTRY END. The transfer of an invalid character activates the validity check and re-cycle circuits.
16. *Altr SW.* There are four alteration switch selectors on the control panel that correspond to the four alteration toggle switches on the machine. When the alteration toggle switches are turned on, the corresponding selectors transfer. The selectors can be used independently or in conjunction with co-selectors to change machine functions under the control of a toggle switch.
17. *Altr SW IMP—Alteration Switch Impulse.* These hubs provide an impulse that is normally wired through an alteration switch selector to the pickup of a co-selector to expand the alteration switch selector.
18. *Bus.* These hubs are located in convenient sections of the control panel and are used to expand either exits or entries, thereby eliminating the need for split wires.
19. *Column Splits.* This is a 12-position selector that is controlled to transfer automatically between 0 and X time as the card is read. This allows the 12 and X punches to be separated from 0-9 punches for storage entry, pilot, field, and co-selector control, etc.
20. *Column Split Couple.* An impulse is emitted from these common hubs between X and 0 read time. This impulse is used to pick up co-selectors to expand the column-split feature.
21, 22. *Digit Selector.* The DIG IMP (Digit Impulse) hub emits a series of impulses that are timed for reading the digits 9-12. If the DIG IMP hub is wired to the C of the digit selector, the digit selector becomes a digit emitter with a 12 impulse available at the 12 hub, an 11 impulse at the 11 hub, etc. If

a reading brush is wired to the C hub, whatever digit is read is available at the corresponding numbered hub of the digit selector.

23. *Half-Time Emitter.* These are off-time emitter hubs that emit an impulse at half after the number they represent. They are normally wired to pick up selectors that are then used as column splits.
24. *Digit Emitter.* These hubs emit read-digit impulses on every read-feed cycle. They are used to enter data that is not available in the input cards into storage.
25, 26. *Fld Sel PU. Field Selector Pickup.* These hubs are impulsed to cause the corresponding level of the field selectors to transfer on the following cycle. The field selector couple exit hubs emit impulses whenever the corresponding level of the field selector is active. They are normally wired to co-selector pickups to expand the field selectors for specific levels.
27, 28. *Load Zero Exit. Load Zero Entry.* These hubs are associated with the eight input words that are active on load cards. If any word in a load card contains invalid information, the load zero entry hubs associated with that word must be wired from the load zero exit hubs. That word will be filled with zeros in the input synchronizer.
29. *Alphabetical In.* When these hubs are wired from the CAI hubs, alphabetic translating circuitry is activated to convert the IBM card code to 2-digit numerical coding for the 7070. An alpha sign is entered into the sign position of the designated word.
30. *CAI—Constant Alphabetical Impulse*
    a. D-Detail. This hub emits an impulse on all read cycles except load cycles.
    b. L-Load. This hub emits an impulse on load cycles only.

    CAI-L and CAI-D can be selected and are normally wired to ALPHABETICAL IN.
31. *12-Impulse.* These hubs emit 12 impulses on all read cycles.
32. *0-Impulse.* A 0 impulse is emitted from these hubs on all read cycles.
33. *Second Read Brushes.* Impulses are emitted from these hubs as the input card passes the second reading station. These impulses are normally used for picking up selectors and for pre-reading alphabetic data.
34. *Co-Selectors.* Sixteen co-selectors are standard. Each selector has five positions. When the pickup hubs are impulsed, the selector transfers immediately and holds to the end of the same cycle.
35. *Alph Pre-Read.* These are entry hubs for impulses emitted from the second read brushes when alphabetic data is to be entered into storage. The same card columns that are wired to ALPH PRE-READ from second reading must be wired to STORAGE ENTRY from third reading.
36. *Entry End. Units, Tens, Exit.* These hubs are wired to set up for a ring check the total number of character positions that have been wired to storage entry for a given card format. If the count of characters in storage entry does not equal the number wired to ENTRY END, the ring check signal comes on.
37. *Word Exit.* These 16 hubs emit sequentially as words 1-16 are transmitted to the input synchronizer for entry to storage. These impulses are always wired to the word-size entry hubs to cause the proper number of characters to be entered.
38. *Word-Size Entry.* These hubs are wired from the word exit hubs to cause specific numbers of characters to be entered into the input synchronizer for each input word.

IBM 7550 Card Punch

One to three IBM 7550 Card Punch units (Figure 192) can be attached as output devices for the IBM 7070 Data Processing System. Information from the output synchronizer is directed by the control panel to punch cards in any desired format.

During continuous operation, a maximum of 250 cards a minute can be punched.

FIGURE 192. 7550 PUNCH

It is possible to perform gangpunching, double-punch and blank-column detection, column splitting, etc. SPOOL (Simultaneous Peripheral Operation on Line) routines can be accomplished when priority signals for automatic program control are assigned.

Validity checking can be suspended to allow punching after an invalid character has been detected.

Punching with internally wired format control is possible by use of the unload hubs on the control panel.

Punching and offset stacking can be controlled by control-panel signals initiated by the stored program instructions.

The functions of the 7550 are illustrated by examples with samples of data contained in the output synchronizer, punched cards, and wiring diagrams.

Card Feeding

The physical arrangement of the card punch consists of a hopper in which blank cards are placed, an 80-column punch station, a read station, a standard offset station, and a drum stacker (Figure 193). Cards are fed into the machine face down 9-edge first.

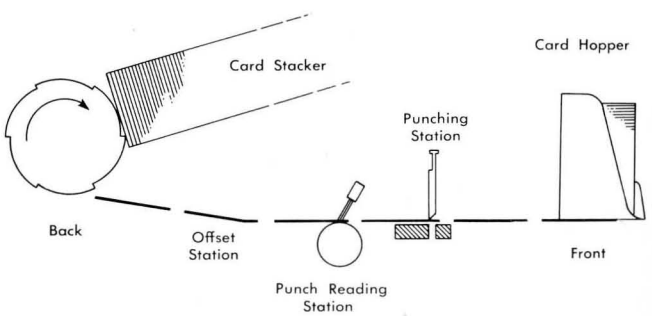


FIGURE 193. 7550 PUNCH SCHEMATIC

Operating Keys and Signal Lights

MASTER SWITCH: This switch (Figure 194) controls the power supply for the IBM 7550. It is located on the left front of the machine.

The following keys and lights are located on the right front of the card punch:

POWER ON LIGHT: This light glows when power is supplied to the 7550.

START KEY: The start key feeds cards into the punch feed. After the first card passes the punching station, further feeding is under control of the 7070 program. This key is also pressed to re-start the punch after it has been stopped for any reason.

STOP KEY: Pressing this key stops the machine and removes it from a ready status. If it is pressed during a punch cycle, that cycle will be completed before the 7550 stops.

RESET KEY: This key is used to extinguish lights caused by certain stop conditions after they have been noted by the operator. The punch is restarted by pressing RESET and START.

READY LIGHT: This light glows when the punch is ready to respond to pressing the start key.

RING CHECK LIGHT: This light turns on and the machine stops when the number of characters emitted by the storage exit hubs as controlled by word-size wiring on the control panel does not equal 80.

CLOCKING CHECK LIGHT: The machine makes an internal check to insure that certain timing circuits are operating properly. If these circuits fail, the machine stops and the clocking check light comes ON, indicating a possible error. This light is extinguished by pressing the reset key. Repeated clocking check lights indicate that the machine should be checked by a customer engineer.

VALIDITY CHECK LIGHT: This light turns on when the validity check circuits detect an invalid character. If WVI is not wired to AUTO STOP, the light goes out automatically. If WVI is wired to AUTO STOP, the light is turned off by pressing RESET.

AUTO-STOP LIGHT: This light glows whenever the punch stops as a result of a condition wired on the control panel to auto stop. The machine is restarted by pressing RESET and START.

FEED CHECK LIGHT: This light turns on:

1. If the machine feeds a card without receipt of a punch command, (except on run-in or run-out).
2. If the machine fails to feed on receipt of a punch command.

CARD ADVANCE LIGHT: This light turns on and the machine stops if a card fails to feed from the hopper into the machine or from one feed station to the next. The start key is inoperative until the cards have been removed from the hopper. The light is extinguished when the feed has been cleared. If there are cards in the feed unit when the power is turned off, the card advance light comes ON when the machine is next started. The light alerts the operator that there are cards in the feed.

FUSE LIGHT: This light turns on and the machine stops whenever a fuse burns out.

Note: The main line switch on the 7550 must be turned off while the fuse is replaced.

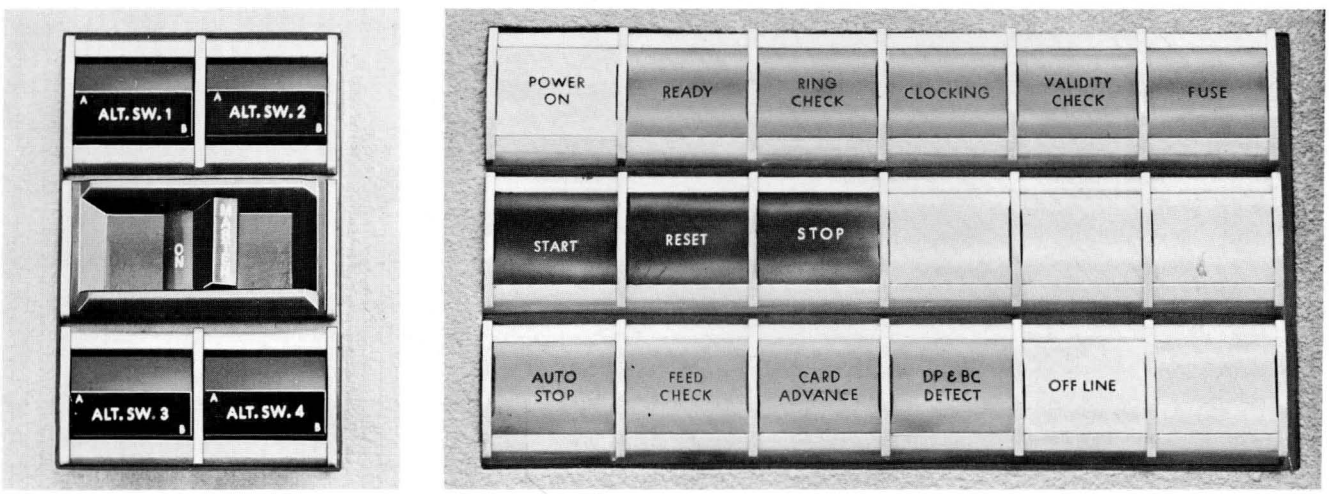


FIGURE 194. OPERATING KEYS AND SIGNAL LIGHTS

OFFLINE LIGHT: This light glows whenever the punch is being used as an independent gang punch machine. It is turned on by wiring INDP OPN (Independent Operation) on the control panel. It is turned off by removing the INDP OPN wire and pressing the Reset Key.

DPBC LIGHT: The punch control panel may be wired to stop the machine and turn this light on when either a double-punch or a blank-column error is detected.

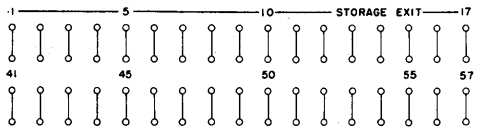
Control Panel

The IBM 7550 Control Panel (Figure 195) has been designed to allow format flexibility of punched-card output from the IBM 7070 Data Processing System. Data are transmitted from the output synchronizer and are arranged for punching by control-panel wiring. All features and functions described in this section are included as standard equipment on the 7550.

Planning Chart

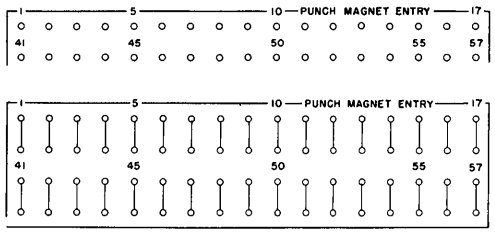
The reverse side of the IBM 7550 Punch control-panel diagram is the IBM 7070 Punch planning chart (Figure 196). If all notes that pertain to a given punch output format are written in the appropriate sections on this chart, the job of wiring the control panel is greatly simplified. The chart should be kept for use in control-panel testing and also serves as a permanent record of the job for which that control panel was designed.

Control-Panel Hubs



AB-AE, 16-55

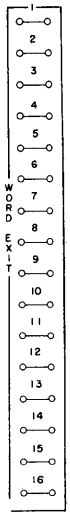
STORAGE EXITS: Any 80 characters from the sixteen words contained in the output-synchronizer can be controlled to exit through these hubs. Storage exit impulses are normally wired to the punch magnet entry hubs. The sign of a numerical word is emitted from the same storage exit position as the units (low-order) digit of the word.



V-W, 16-55 AK-AN, 16-55

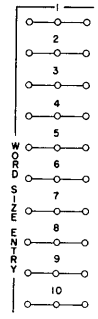
PUNCH MAGNET ENTRY: These three sets of 80 hubs are multiple entries to the 80 punch magnets on the 7550. Digit impulses directed to these hubs, from storage exits or from the emitter, punch in the corresponding card columns.

X-AM, 63-64



WORD EXIT: These hubs (1-16) are used to indicate the number of significant character positions assigned to each output word to be punched. They are normally wired to WORD SIZE ENTRY.

X-AG, 60-62



WORD SIZE ENTRY: These hubs are wired from the word exit hubs to punch the actual number of character positions that comprise a specific output word. They are labeled 1-10.

Note: All eighty positions of storage exit must be wired to emit impulses, but it is not necessary to wire all 80 to punch. If less than eighty positions are indicated by the wiring of word size entry, the ring check light on the punch comes ON.

Output Data Flow

Figure 197 shows data flow from the output-synchronizer to the output card. Figure 198 is the planning chart for the operation. Data are assembled in words 1-16 of the output-synchronizer through programmed instructions. On a punch command, as many as 80 characters from the synchronizer are available at the storage exit hubs on the control panel. Data are arranged and punched in the card by control-panel wiring from STORAGE exits to the punch magnets that correspond to specific card columns. The sign associated with a numercial output word is emitted from the low-order storage exit position of that word. The number of character positions to be punched for each word is controlled by the wiring from WORD EXITS to WORD SIZE ENTRY. Unused words in the output-synchronizer are automatically filled with zeros (the alphabetical special-character code for blank column) and the signs are set to Alpha.

OUTPUT SIGN CONTROL

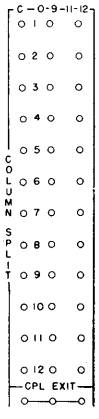
Signs of numerical words are emitted from the units position of the storage exit word. The setting of the PCH+ switch controls punching of positive signs.

AC, 1-2 

PCH+ (PUNCH PLUS SIGN): This switch is wired when it is desired to punch positive sign indications in the card. When this switch is *not* wired, only negative signs are available at the control panel. The positive sign is a 12-impulse, and the negative sign is an 11-impulse. The PCH+ switch can be selected.

Figure 199 shows wiring for punching word 1 in card columns 61-68 with the sign punched over position 68.

K-V, 56-58
W, 56-58



COLUMN SPLITS: This device is used to separate 0-9 impulses from 11 and 12 impulses, to be punched in a card. During the time that the 12 and 11 impulses are emitted from storage, the 12-11 and C hubs of the column split are connected. At 0-9 time, the 0-9 and C hubs are connected.

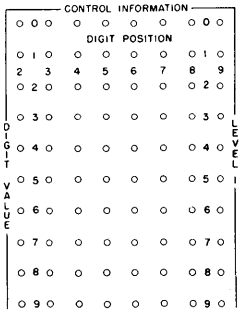
COLUMN SPLIT COUPLE: An impulse emitted from these hubs can be wired to a selector pickup hub to expand the column split feature. The corresponding selector can then be used as an additional group of column splits.

Figure 200 shows how output word 1 can be wired with plus and minus signs punched over a card column other than the units position of the field.

The Control Word

Word 1 of the output synchronizer is designated as the control word in the IBM 7070 Data Processing System. Included in the control word are ten digit positions, each of which can contain any digit value (0-9). Digit positions 2-9 each have 10 digit value exits on the control panel labeled CONTROL INFORMATION.

A-J, 1-8



CONTROL INFORMATION: An early-timed impulse is emitted from one hub in each of eight vertical columns. These columns represent the digit value in each of eight digit positions (2-9) in the control word. These impulses can be used to pick up co-selectors, field selectors, pilot selectors, etc.

Y, 3-4 

CI OFF (CONTROL INFORMATION OFF): These hubs are often impulsed from WVI or EWI. When they are wired, the control information hubs become inactive.

If the control word is alphabetic, the control information hubs emit the 7070 alphabetic code. For example: The alphabetic word SMITH causes 74 69 83 68 to emit.

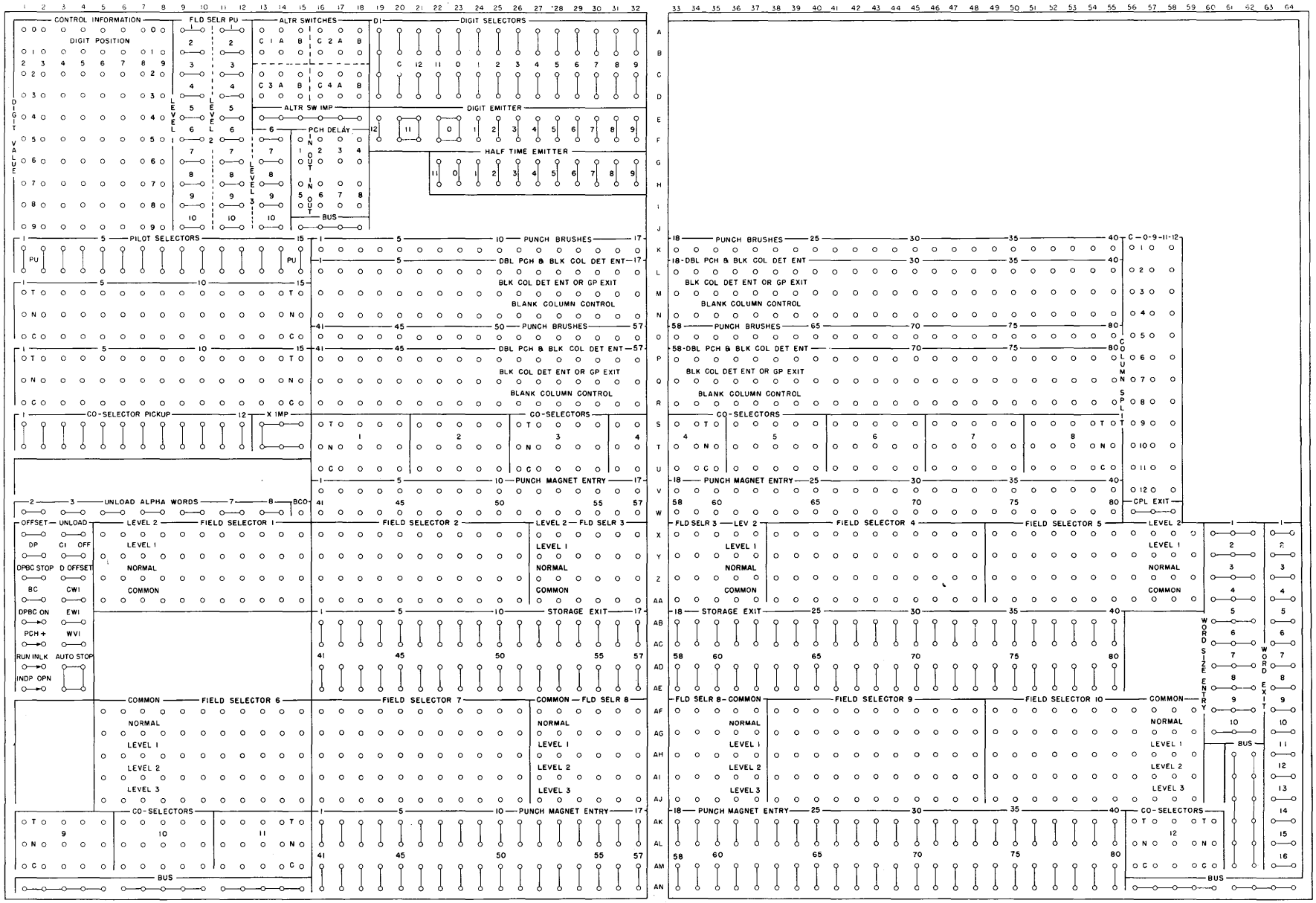


FIGURE 195. 7550 PUNCH CONTROL PANEL (FORM X24-6427)

FIGURE 196. PUNCH PLANNING CHART

IBM 7550 PUNCH PLANNING CHART

PREPARED BY _____ DATE _____

CARDS USED _____ APPLICATION _____

WORD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	REMARKS
CONTROL INFORMATION																	
NUMBER OF CHARACTERS TO BE PUNCHED																	
PUNCH BRUSHES																	
DPBC ENTRY																	
FIELD HEADINGS																	
STORAGE EXITS																	
PUNCH MAGNET ENTRY																	

PILOT SELECTORS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C

FIELD SELECTORS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
3					3	3	3	3	3
2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1
N	N	N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C

CO-SELECTORS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
T	T	T	T	T	T	T	T	T	T	T	T
N	N	N	N	N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C	C	C

MISCELLANEOUS

IND. OP ☐ ON

PCH + ☐

ALTERATION SWITCHES

	A	B		A	B
1	☐	☐	3	☐	☐
2	☐	☐	4	☐	☐

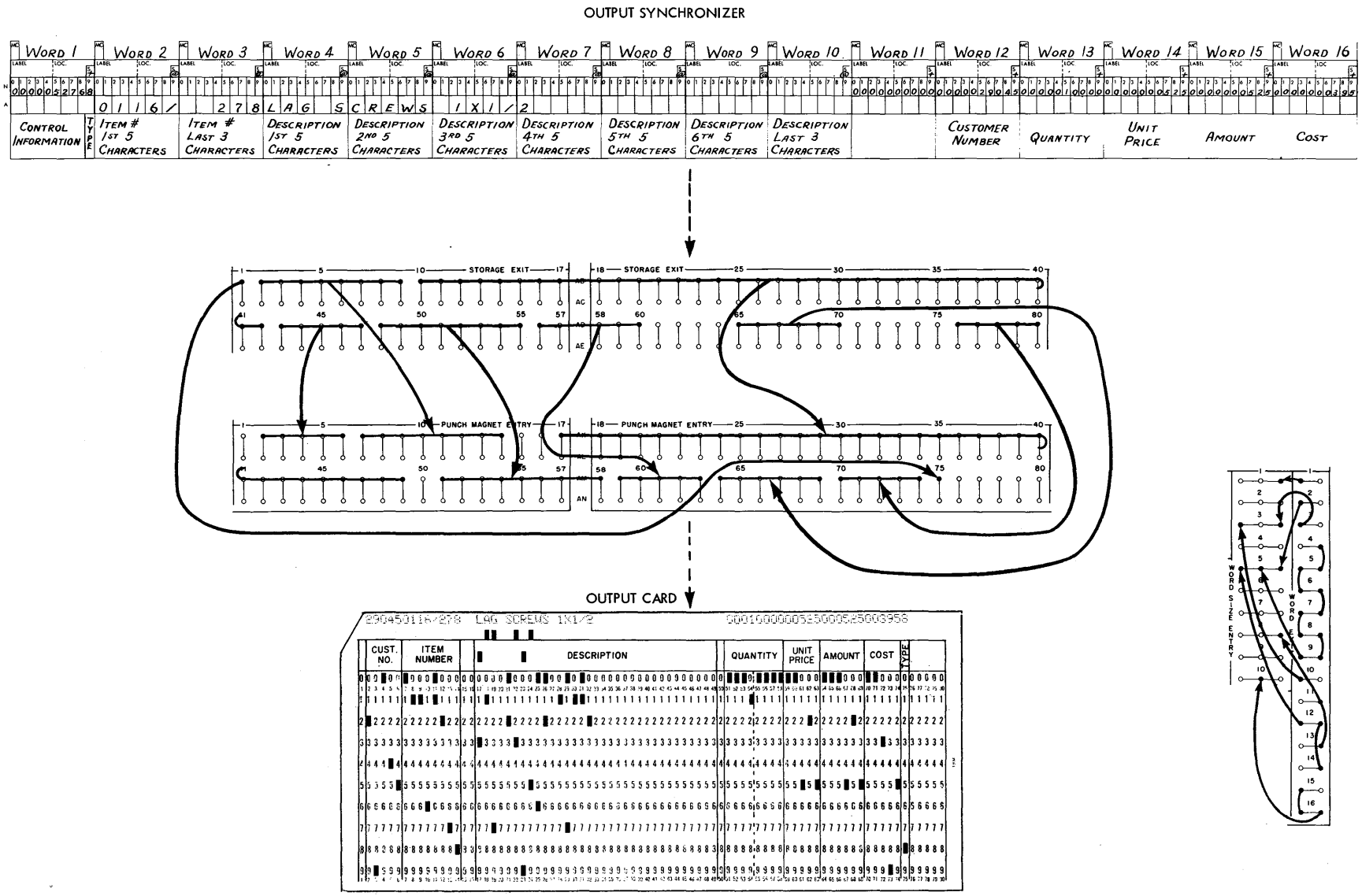


FIGURE 197. OUTPUT DATA FLOW

IBM 7550 PUNCH PLANNING CHART

PREPARED BY: GROUP 2 — DATE: 9-19-59

CARDS USED: SALES ACCOUNTING — APPLICATION: SALES ACCOUNTING

WORD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	REMARKS
CONTROL INFORMATION																	
NUMBER OF CHARACTERS TO BE PUNCHED	1	5	3	5	5	5	5	5	5	3	0	5	8	5	PUNCH 6 EMIT 10	PUNCH 5 EMIT 10	TOTAL 80 IMPULSES AT STORAGE EXIT HUBS
PUNCH BRUSHES																	
DPBC ENTRY																	
FIELD HEADINGS	TYPE	ITEM NO.		DESCRIPTION								CUST. NO.	QTY.	UNIT PRICE	AMT.	COST	
STORAGE EXITS	1	2-9		10-42								43-47	48-55	56-60	65-70	76-80	
PUNCH MAGNET ENTRY	75	7-14		17-49								2-6	51-58	59-63	64-69	70-74	

PILOT SELECTORS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C

FIELD SELECTORS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
3					3	3	3	3	3
2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1
N	N	N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C

CO-SELECTORS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
T	T	T	T	T	T	T	T	T	T	T	T
N	N	N	N	N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C	C	C

MISCELLANEOUS

IND. OP ☐ ON

PCH + ☐

ALTERATION SWITCHES

	A	B		A	B
1	☐	☐	3	☐	☐
2	☐	☐	4	☐	☐

FIGURE 198. PLANNING CHART FOR FIGURE 197

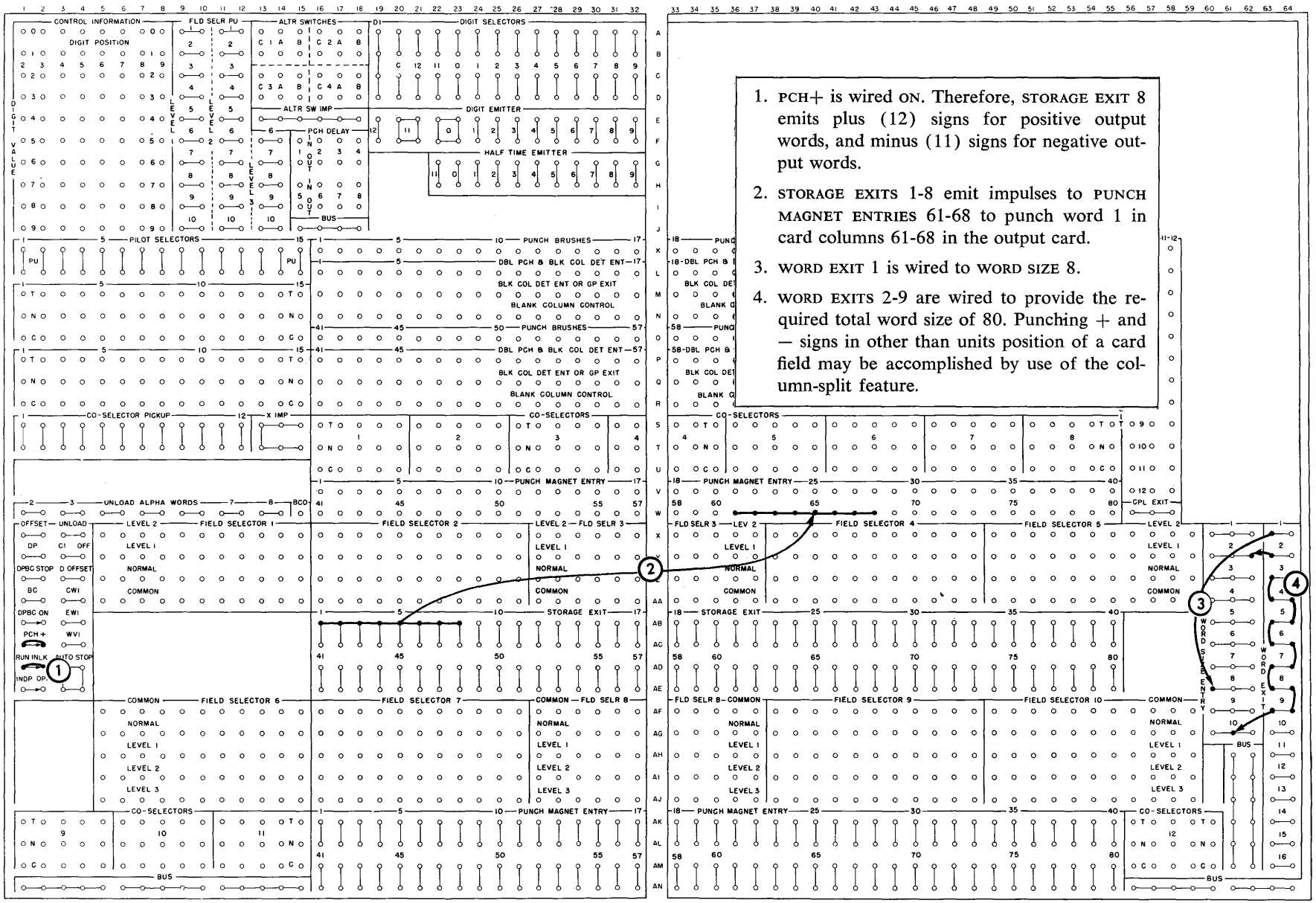


FIGURE 199. SIGN CONTROL—SIGN OVER UNITS POSITION

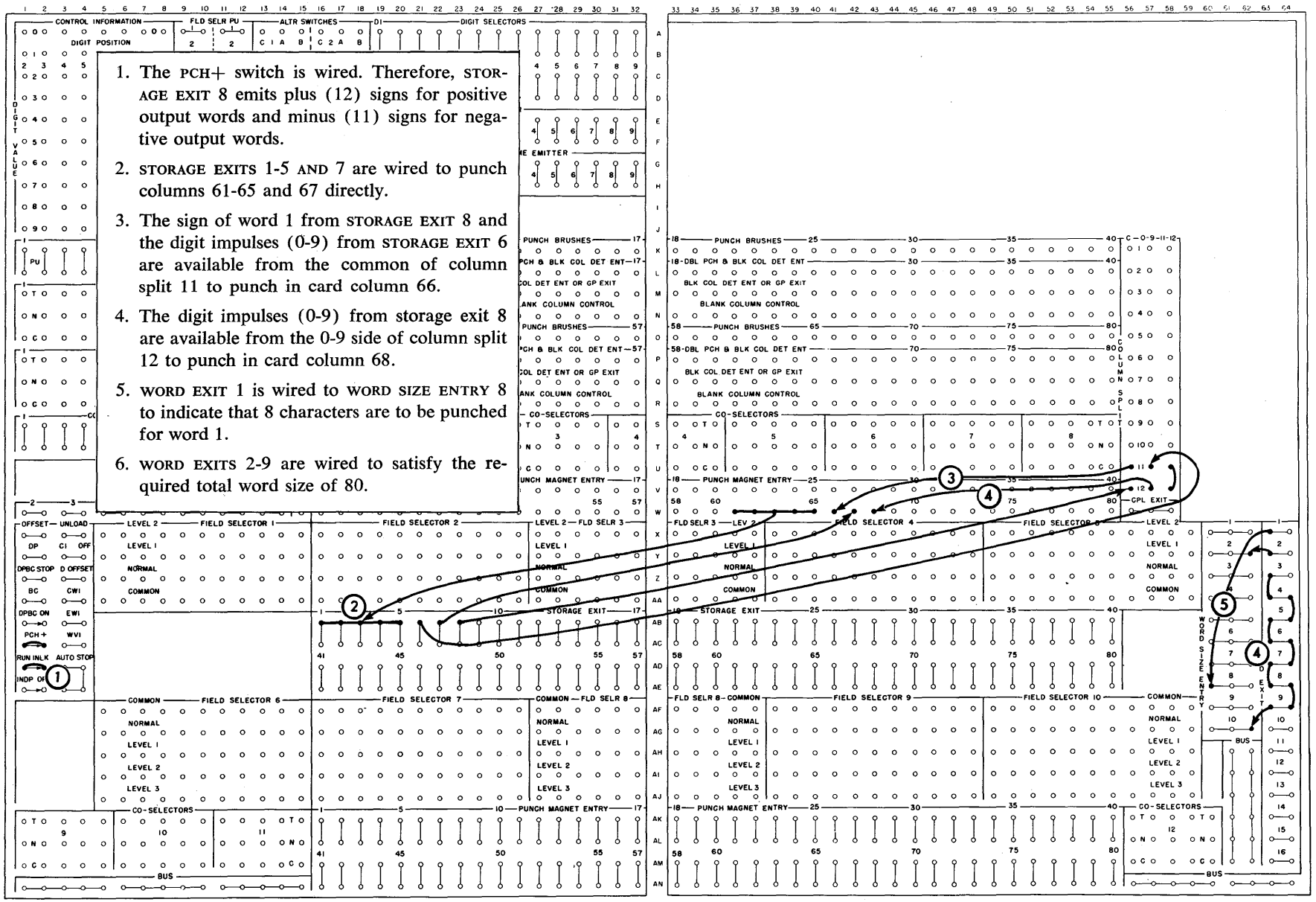
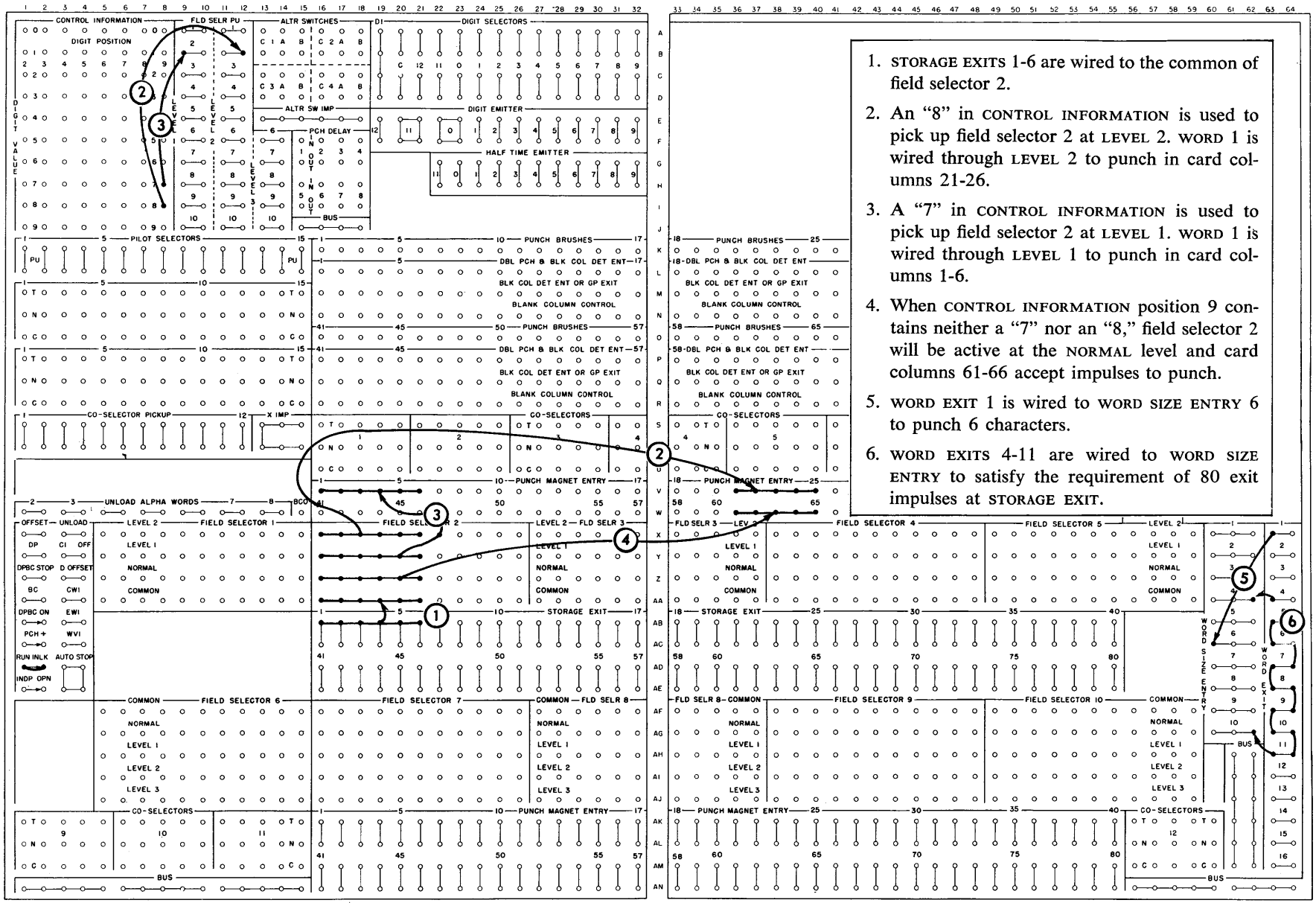


FIGURE 200. SIGN CONTROL—SIGN NOT OVER UNITS POSITION



1. STORAGE EXITS 1-6 are wired to the common of field selector 2.
2. An "8" in CONTROL INFORMATION is used to pick up field selector 2 at LEVEL 2. WORD 1 is wired through LEVEL 2 to punch in card columns 21-26.
3. A "7" in CONTROL INFORMATION is used to pick up field selector 2 at LEVEL 1. WORD 1 is wired through LEVEL 1 to punch in card columns 1-6.
4. When CONTROL INFORMATION position 9 contains neither a "7" nor an "8," field selector 2 will be active at the NORMAL level and card columns 61-66 accept impulses to punch.
5. WORD EXIT 1 is wired to WORD SIZE ENTRY 6 to punch 6 characters.
6. WORD EXITS 4-11 are wired to WORD SIZE ENTRY to satisfy the requirement of 80 exit impulses at STORAGE EXIT.

FIGURE 201. FIELD SELECTION

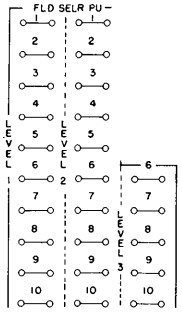
Selection

FIELD SELECTORS

Output from the 7070 can be arranged in several different punching formats without changing control panels by using the field selectors provided.

A-J, 9-12

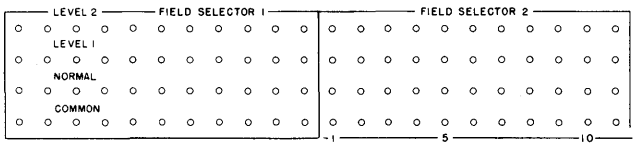
F-J, 13-14

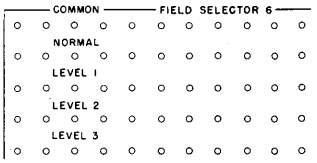


FIELD SELECTORS PICK UP LEVELS 1, 2, AND 3: These hubs when impulsed cause the corresponding level of a field selector to transfer on the same punch cycle. They are normally wired from CONTROL INFORMATION. An impulse to these hubs causes the field selectors to transfer immediately and remain transferred for the duration of that punch cycle.

Note: The higher-numbered levels have selective priority.

Example: If both levels 1 and 2 are impulsed to pick up on the same cycle, then only level 2 is connected to common.





X-AA, 5-59

AF-AJ, 5-59

FIELD SELECTOR LEVELS 1, 2, 3—NORMAL & COMMON: There are ten 11-position field selectors provided. Field selectors 1-5 have three levels each (NORMAL, LEVEL 1 AND LEVEL 2); selectors 6-10 have four levels each (NORMAL, LEVEL 1, LEVEL 2, AND LEVEL 3).

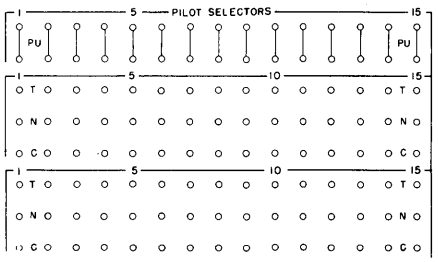
Note: The vertical arrangement of the four level field selectors 6-10 is the opposite of the three level field selectors 1-5.

Figure 201 shows the punching of three different card formats using control information and field selection. If there is an 8 in position 9 of the control information word, card columns 21-26 are punched from output word 1. If there is 7 in position 9 of the control information word, card columns 1-6 are punched from word 1. If control information contains neither 7 nor 8 in position 9, columns 61-66 are punched from word 1.

PILOT SELECTORS AND CO-SELECTORS

Pilot and co-selectors are provided to increase the flexibility of format on output. Each individual position of a selector is a switch composed of three hubs labeled T, N, C. The C (common) hub is always connected to either the N (normal) or T (transferred) hub, but never to both. Thus, impulses entering C are selectively available at either N or T, depending upon whether the pickup hubs have received an impulse. Conversely, impulses entering either N or T are available at C under control of the impulses to the pickup hubs.

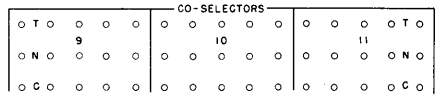
K-R, 1-15



PILOT SELECTORS: Fifteen 2-position pilot selectors are provided. Each position has three points, C (Common), N (Normal), and T (Transferred). When a pilot selector is transferred, there is an internal common connection between C and T. When it has not been impulsed to pick up, an internal common connection exists between the C and N hubs.

PILOT SELECTOR PICKUP: Each pilot selector is equipped with its own PU (pickup) hub. When an impulse reaches this hub, the corresponding selector transfers immediately and remains transferred for the remainder of that punch cycle.

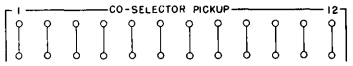
S-U, 16-55



CO-SELECTOR: The 7550 has twelve, 5-position co-selectors. Each co-selector position has three points, C (Common), N (Normal), and T (Transferred). When a co-selector is transferred, there is an internal common connection between C and T. When it has not been impulsed to pick up, an in-

ternal common connection exists between the C and N hubs.

S-T, 1-2



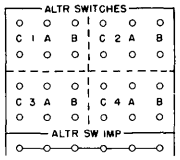
CO-SELECTOR PICKUP: Each co-selector has two common pickup hubs. When these hubs are impulsed, the selector transfers immediately and remains transferred for the duration of that cycle. They are normally wired from control information exit hubs.

Figure 202 illustrates the basic use of pilot and co-selectors. Cards 1, 2, and 3 are punched with information from word 1.

Alteration Switches

Within reasonable limitations, one control panel can be used for punching several different card formats without any change in control-panel wiring, by use of the alteration switches.

A-D, 13-18



ALTR SW (ALTERATION SWITCH) 1, 2, 3, 4: Four alteration switches are located on the front of the machine. Each alteration switch controls a corresponding 2-position alteration switch selector. When the switch is set on A, a common connection exists between C and A; when the switch is set on B, the connection is between C and B. An alteration switch selector remains transferred as long as the corresponding alteration switch is set on B. Any impulse may be wired through C-A or C-B.

ALTR SW IMP (ALTERATION SWITCH IMPULSE): These hubs emit an impulse that is normally wired through an alteration switch selector to the pickup of pilot selectors, field selectors, or co-selectors. A selector impulsed through the A side of an alteration switch selector transfers each card cycle as long as the corresponding alteration switch is set to A. If the selector is thus impulsed through the B side of an alteration switch selector, the pilot or co-selector remains transferred as long as the corresponding alteration switch is set to B.

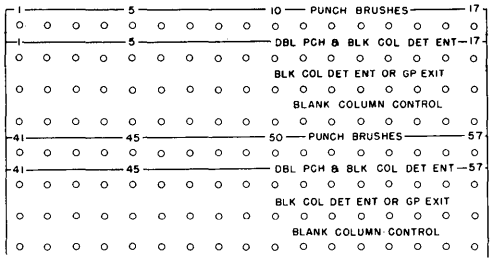
In Figure 203 the information from storage exits 1-5 is punched under control of alteration switch 3.

Double-Punch Blank-Column Detection (DPBC)

With this device it is possible to check output cards for both double-punched and blank columns. Also, card columns can be checked for either condition selectively. To activate this device, the punched output cards are read by the punch brushes and wired on the control panel for the appropriate check.

AB, 1-2

DPBC ON (DOUBLE-PUNCH BLANK-COLUMN DETECTION ON): These hubs are connected when it is desired to check a card for double punches and/or blank columns.



K-R, 16-55

PUNCH BRUSHES: These hubs are exits for digit impulses (12-9), read from the output card one cycle after the card is punched. These hubs are numbered to correspond to the card column being read.

DP AND BC DET ENTRY (DOUBLE-PUNCH AND BLANK-COLUMN DETECTION ENTRY): These are entry hubs for the information being read by the punch brushes. They are used to detect both double-punched and blank columns in the output card.

BC DET ENTRY OR GP EXIT (BLANK-COLUMN DETECTION ENTRY OR GANGPUNCH EXIT): These hubs can be used as either entries or exits. As entries, they are used to detect blank columns in the output cards, without using double-punch blank-column detection. As exits, they are wired to the punch magnets for gangpunching. When used as gangpunch exits, only the first digit read from any column wired to a DP and BC DET ENTRY is available at the associated GP EXIT.

BC CONTROL (BLANK-COLUMN CONTROL): These hubs provide a means for selective control of blank-column detection. All positions of DPBC are internally connected for blank-column detection. By control-panel wiring, it is possible to bypass one or more positions that are not to be checked for blanks.

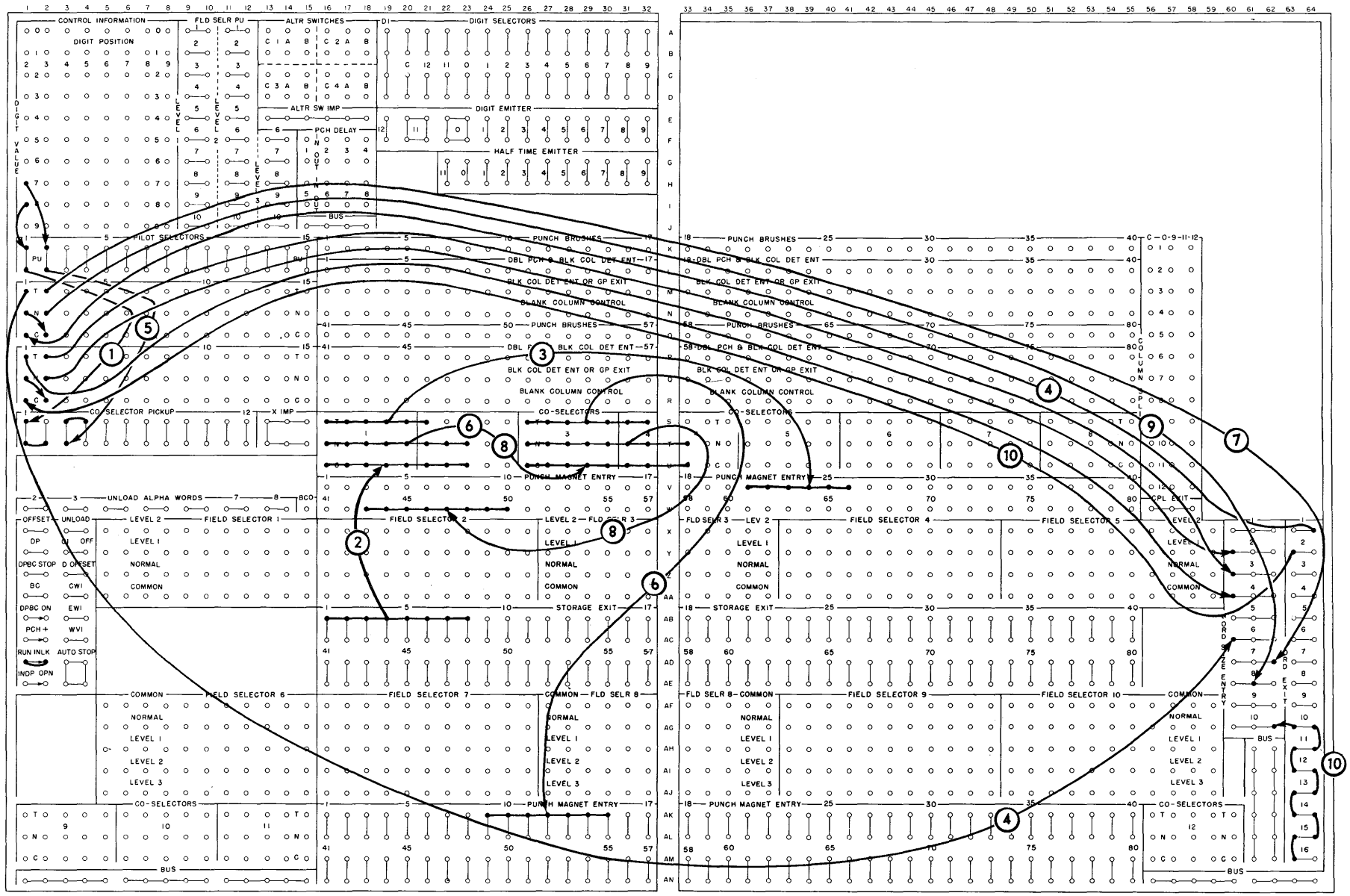


FIGURE 202. PILOT AND CO-SELECTOR OPERATION

1. Pilot selector 1 and co-selectors 1 and 2 are impulsed by an "8" in CONTROL INFORMATION 2. These selectors transfer at the beginning of the punch cycle and remain transferred for the duration of that cycle.
2. The common sides of co-selectors 1 and 2 are wired from STORAGE EXITS 1-8.
3. The information that is to punch in card columns 21-26 is wired from the *transferred* (T) sides of co-selectors 1 and 2 to PUNCH MAGNET ENTRY.
4. WORD EXIT 1 is wired to WORD-SIZE ENTRY 6 through the transferred side of pilot selector 1.
5. Pilot selector 2 and co-selectors 3 and 4 are impulsed by a "7" in CONTROL INFORMATION position 2. These selectors transfer immediately and remain transferred for the duration of that punch cycle.
6. Because co-selectors 1 and 2 should be normal when 3 and 4 are transferred, the common of co-selectors 3 and 4 accepts storage exit impulses from the normal of co-selectors 1 and 2. Then this information from STORAGE EXITS is available through the transferred side of co-selectors 3 and 4 to PUNCH MAGNET ENTRY 9-15.
7. WORD EXIT 1 is wired to WORD-SIZE ENTRY 7 from the *transferred* side of pilot selector 2. The word exit impulse is made available to the common of pilot selector 2 through the normal side of pilot selector 1. (Pilot selector 1 is normal when 2 is transferred.)
8. If CONTROL INFORMATION contains neither a "7" nor an "8" in position 2, co-selectors 1, 2, 3, and 4 and pilot selectors 1 and 2 are normal. Information passes from STORAGE EXIT through co-selectors 1 and 2, normal, and through co-selectors 3 and 4 normal to PUNCH MAGNET ENTRY positions 43-50.
9. WORD EXIT 1 is wired to WORD-SIZE ENTRY 8 from the normal side of pilot selector 2. The WORD EXIT impulse is made available to the common of pilot selector 2 through the normal side of pilot selector 1.
10. The word-size wiring necessary to total 80 storage exit impulses is completed through pilot selectors 1 and 2 from WORD EXIT 2, and direct from WORD EXITS 10-16.

Note: The word-size wiring using pilot selectors 1 and 2 could have been completed through the unused selector positions of co-selectors 2 and 4. Pilot selectors were used for purposes of illustration.

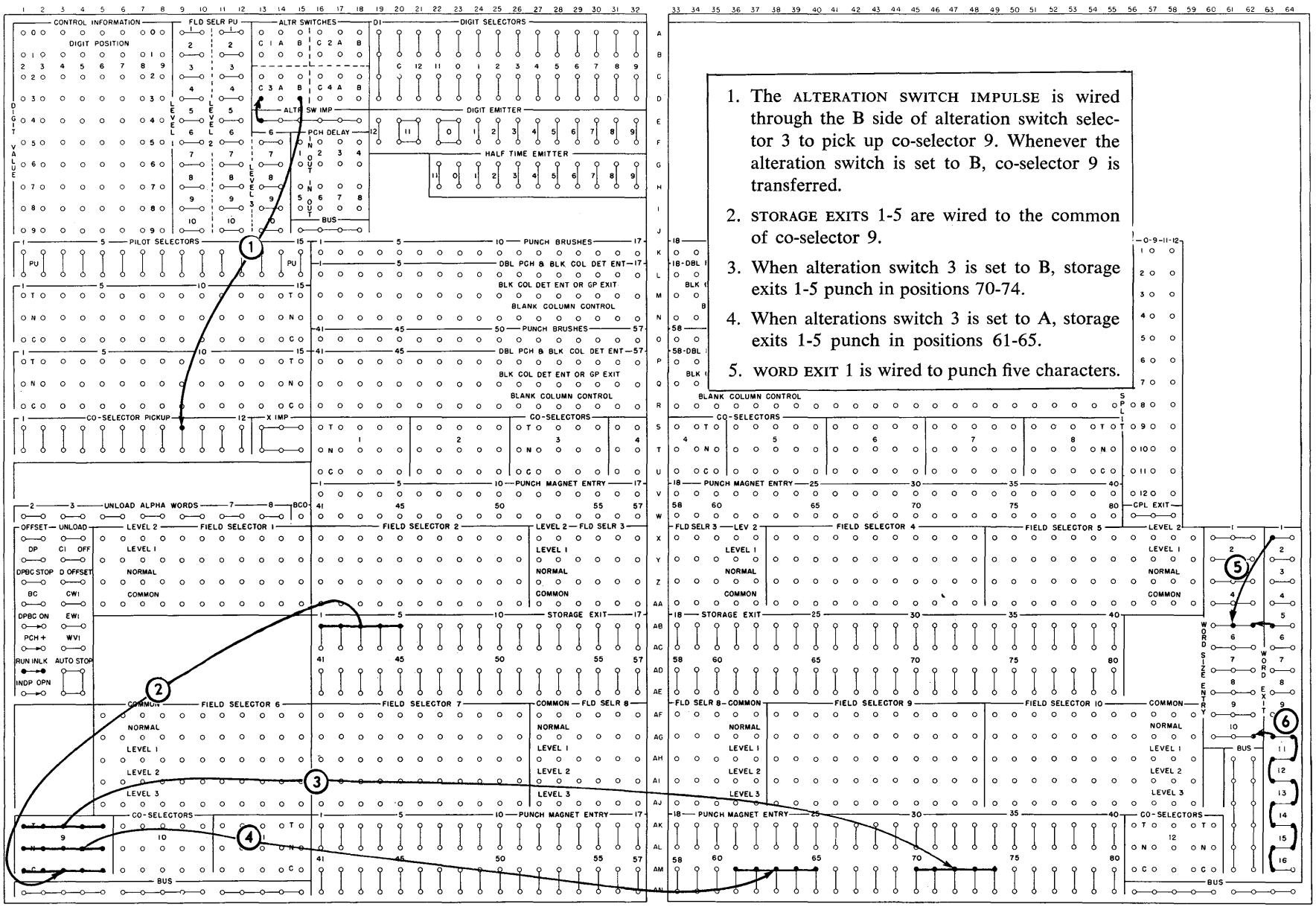


FIGURE 203. ALTERATION SWITCH WIRING

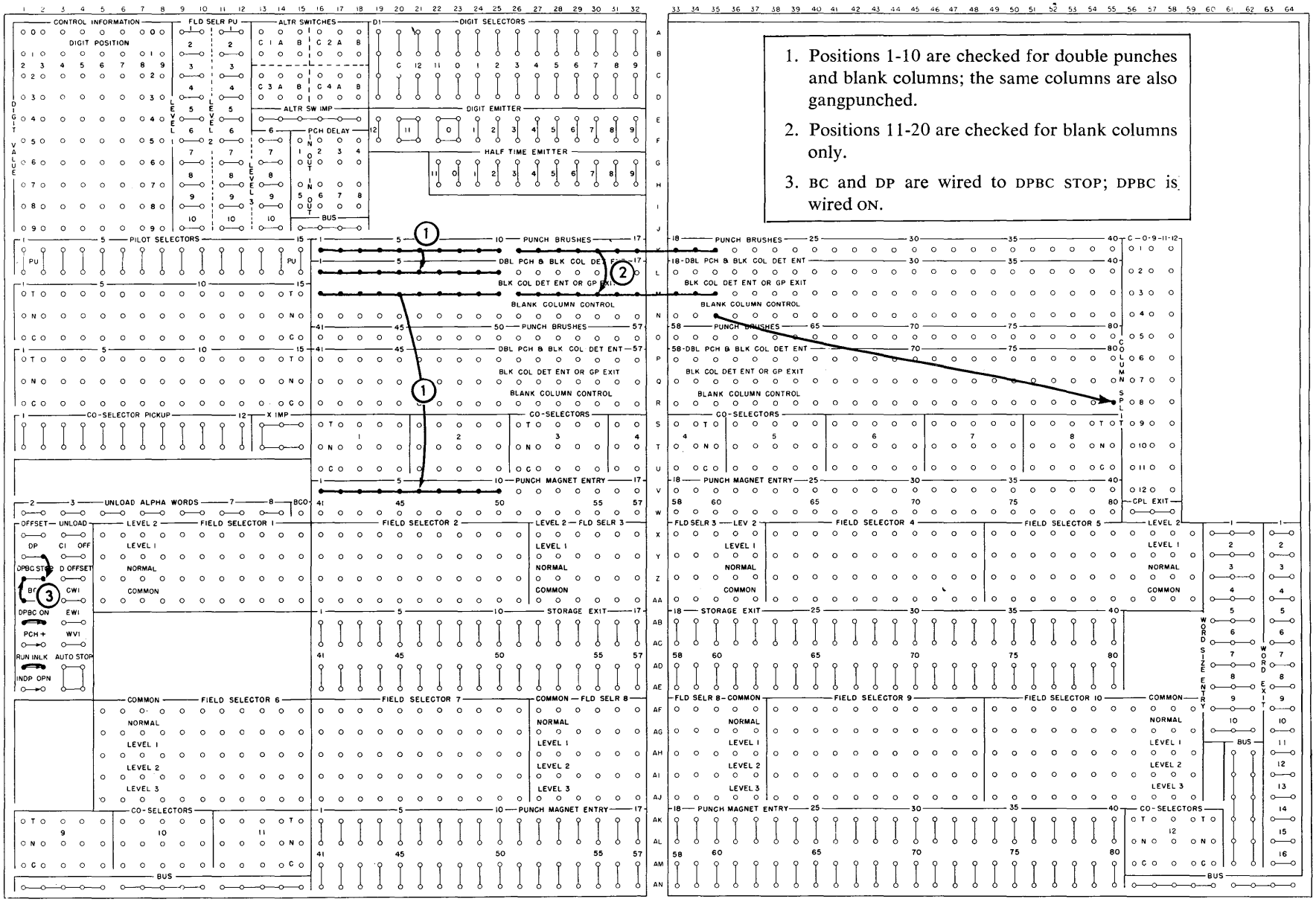
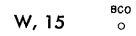
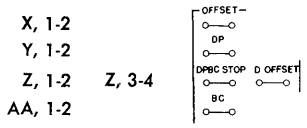


FIGURE 204. DOUBLE-PUNCH AND BLANK-COLUMN DETECTION, AND GANGPUNCHING



BLANK-COLUMN CONTROL ZERO: Bypassing is accomplished by wiring *around* those positions that are not to be checked. The position preceding the first position to be bypassed is wired to the last position to be bypassed. If position 1 is to be excluded, BC0 is considered the preceding position.



DPBC STOP: These hubs accept an impulse to cause the punch to stop at the end of the cycle during which a DPBC error is sensed. They are normally wired from the DP and/or BC hubs.

OFFSET: An impulse to this hub causes the card being read at the punch brushes to be offset in the stacker.

D OFFSET (DELAY OFFSET): When these hubs receive an impulse, the card then at the punching station is offset in the stacker.

DP: This hub emits an impulse at the end of the cycle during which a double punch is detected. It is normally wired to DPBC STOP or OFFSET.

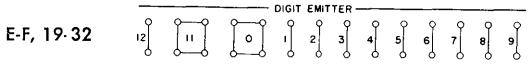
BC: This hub emits an impulse at the end of the cycle during which a blank column is detected.

Figure 204 shows double-punch and blank-column detection, and gangpunching.

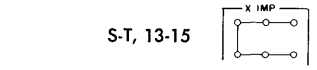
Figure 205 shows two separate control panels. In the first example, positions 1-20 are to be checked for double punches. Also, all positions except 10-13 are to be checked for blank columns. In the second example, positions 1-10 are to be checked for double punches. Also, only positions 4-10 are to be checked for blank columns.

Emitting Punch Impulses

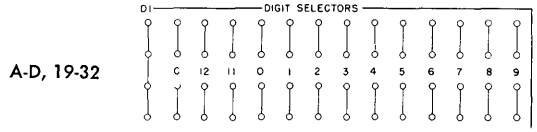
Data can be supplied by the control panel for punching into output cards. These emitted factors can be used for supplying distinguishing punches, etc. Emitted impulses can come from the hubs labeled digit emitter, the X-impulse hubs, or the digit selector wired to function as a digit emitter.



DIGIT EMITTER: These hubs emit digit impulses (9-12) on all punch-feed cycles.



X IMP: These hubs emit an 11 (X) impulse during each punch-feed cycle.

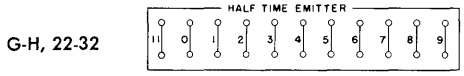


DIGIT SELECTORS—PUNCH: Two digit selectors are standard on the 7550.

A punch-digit selector can be used to separate or combine multiple digits. It may also be used as a digit-emitter by connecting the C hub to the DI hub. DI emits all digit impulses (9-12) every punch-feed cycle.

In Figure 206, there are two types of output cards. The information for one of the output cards is identified by the presence of an 8 in position 9 of control information. The other is identified by the absence of the 8 in the first position of control information. The card controlled by an 8 in position 9 of control information is to be identified by punching an 11 in column 80 and a 1 in column 55. The card controlled by the absence of an 8 in control information is to be identified by a 2 in column 55.

Split-Column Control



HALF-TIME EMITTER: With the half-time emitter, multiple punching in one column can be split between any two punching positions. It differs from normal column split (which is always between 0 and X time in the card) in that the split can be between 9 and 8, 0 and X, etc. There is a hub for all digits 9-0 and 11. They emit impulses at half after the number indicated. For example, 9 emits an impulse between 9 and 8; 0 emits an impulse between 0 and 11.

The half-time emitter hubs are normally wired to a selector pickup (immediate) making the selector operate the same as a column-split device. For example, if a 1 is wired to a selector pickup, the C and N hubs of the selector are common from 9-1 time, and the C and T hubs are common from 0-12 time. Thus, zone impulses may be separated

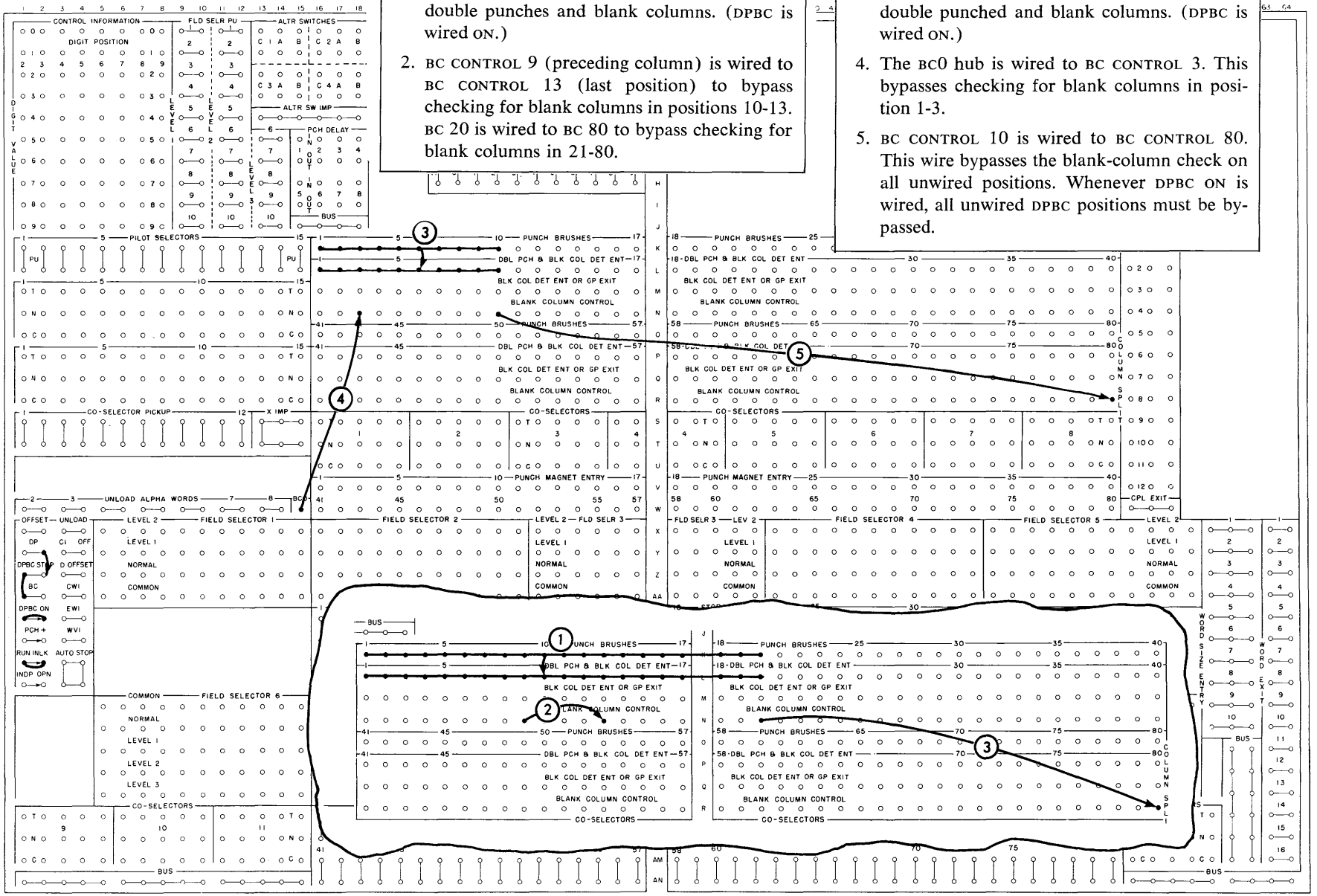
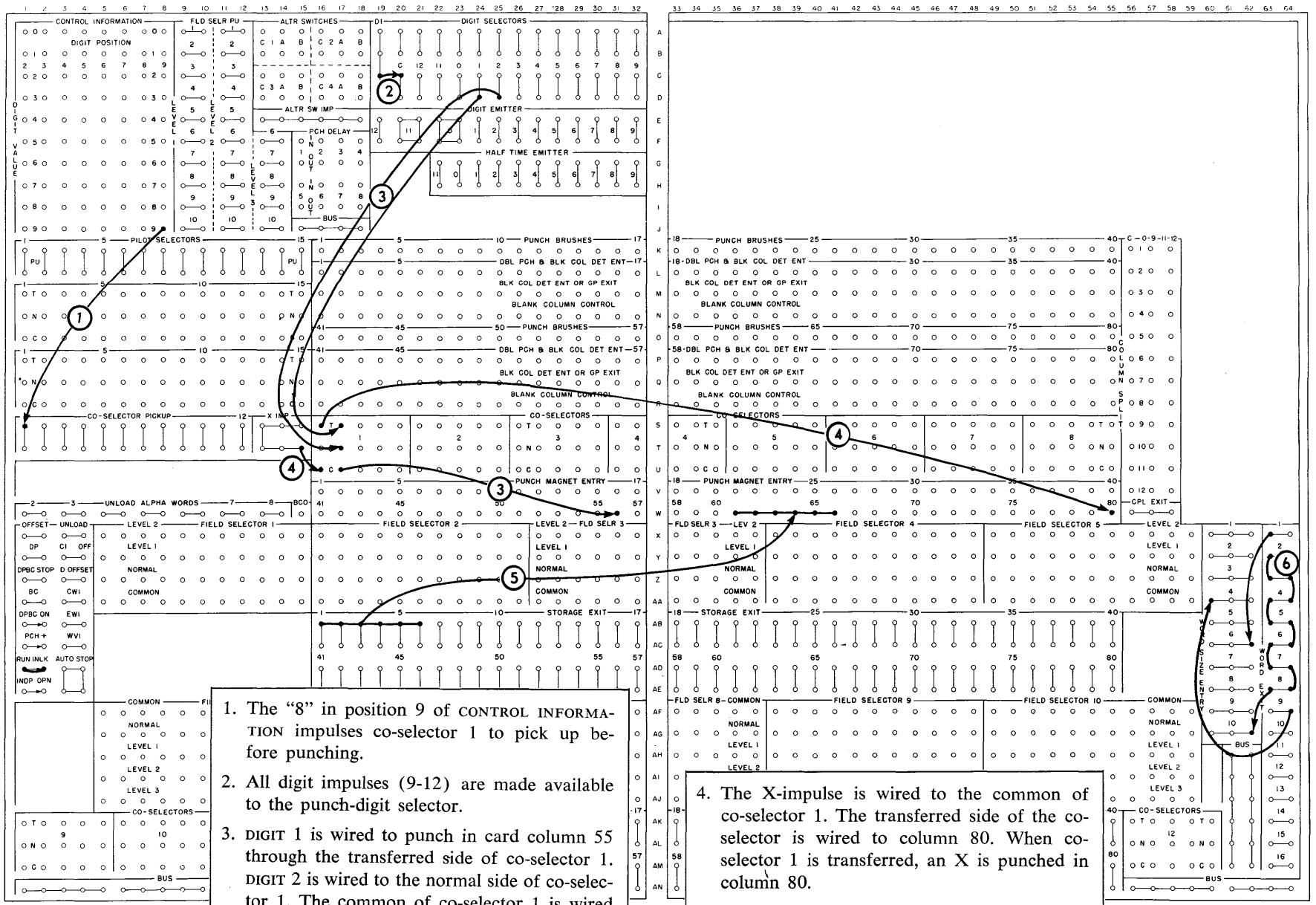


FIGURE 205. BYPASSING BLANK-COLUMN CHECKING

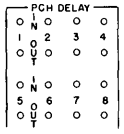
FIGURE 206. EMITTING DATA FOR PUNCHING



1. The "8" in position 9 of CONTROL INFORMATION impulses co-selector 1 to pick up before punching.
2. All digit impulses (9-12) are made available to the punch-digit selector.
3. DIGIT 1 is wired to punch in card column 55 through the transferred side of co-selector 1. DIGIT 2 is wired to the normal side of co-selector 1. The common of co-selector 1 is wired to punch card column 55. When co-selector 1 is normal, a 2 is punched in column 55 of the output card.
4. The X-impulse is wired to the common of co-selector 1. The transferred side of the co-selector is wired to column 80. When co-selector 1 is transferred, an X is punched in column 80.
5. The information from *storage-exit word 1* is punched in columns 61-66 of the output card.
6. Required 80-word-size wiring is shown.

from digit impulses punched in the same column. If the 0 is wired to the selector 1 pickup, the C and N hubs are common for 11 and 12. This arrangement is the same as a normal column split. A column can be split between 11 and 12 by using the 11 hub to pick up the selector.

F-I, 15-18



PUNCH DELAY: There are eight sets of punch delay hubs on the control panel. Each set of hubs consists of an in hub and out hub. The in hub accepts any impulse. The out hub emits an impulse just prior to punching time on the punch cycle following the one during which the in hub is impulsed. It is possible to delay up to eight cycles by wiring the OUT of one to the IN of another.

Figure 207 is based on Figure 206. The card identified by an 8 in position 9 of control information is to have column 80 checked for both double-punched and blank column. The card identified by the absence of an 8 does not check column 80 for either condition. Columns 55, 61-66 are to have double-punch and blank-column detection on both types of cards. Only the additional wiring for this checking is shown:

Unloading

Another method of output from storage is available. This method bypasses the control panel and sets up a direct internal connection between the output synchronizer and the punch magnets. This internal connection is under control of the UNLOAD hubs.

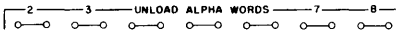
X, 3-4 UNLOAD

UNLOAD: These hubs accept early-timed impulses to cause words 1-8 to be transferred automatically to punch magnets 1-80 as follows:

Word	Punch Magnets
1	1-10
2	11-20
3	21-30
4	31-40
5	41-50
6	51-60
7	61-70
8	71-80

The sign is punched over the units position of each numerical output word (i.e. card column 10, 20, etc.). PCH+ is automatically on. Alphabetic words, when controlled by the wiring to UNLOAD ALPHA WORDS, are punched in the last five card columns assigned to those words. If word 1 is alphabetic or the unload alpha word hub for any other alpha word (2-8) has not been impulsed, this format will punch: *b* A *b* B *b* C *b* D *b* E (*b* = blank).

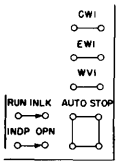
W, 1-14



UNLOAD ALPHA WORDS 2-8: These hubs are normally wired from CONTROL INFORMATION to control the punching of alphabetic words for an unloading operation.

Other Control-Panel Hubs

AA, 3-4
AB, 3-4
AC, 3-4
AD, 1-2 AD-AE, 3-4
AE, 1-2



CWI (CONSTANT WRITE IMPULSE): These hubs emit an impulse every punch cycle.

WVI (WRITE VALIDITY IMPULSE): These hubs emit an impulse if the write validity check circuits detect an invalid character when data is transferred from synchronizer to unit. If this pair of hubs is not wired, the detection of an invalid character causes the validity check light on the 7550 to flash. The impulse can be wired to AUTO STOP or OFFSET.

EWI: An early-timed impulse is emitted under control of the write invalid command (UWIV) in the 7070 program. It can be wired to AUTO STOP, to CI OFF or to pickup co-selectors for special format punching, etc.

INDP OPN (INDEPENDENT OPERATION): This switch is jackplugged to allow the punch to operate as an independent unit.

RUN INLK (RUN INTERLOCK): This switch must be wired to complete the punch unit run circuit. Its purpose is to prevent operation of the unit without a wired control panel in place.

AUTO STOP: These hubs accept any impulse to suspend punch operation at the end of that punch cycle. When AUTO STOP has been impulsed, the auto stop light on the front of the machine turns on.

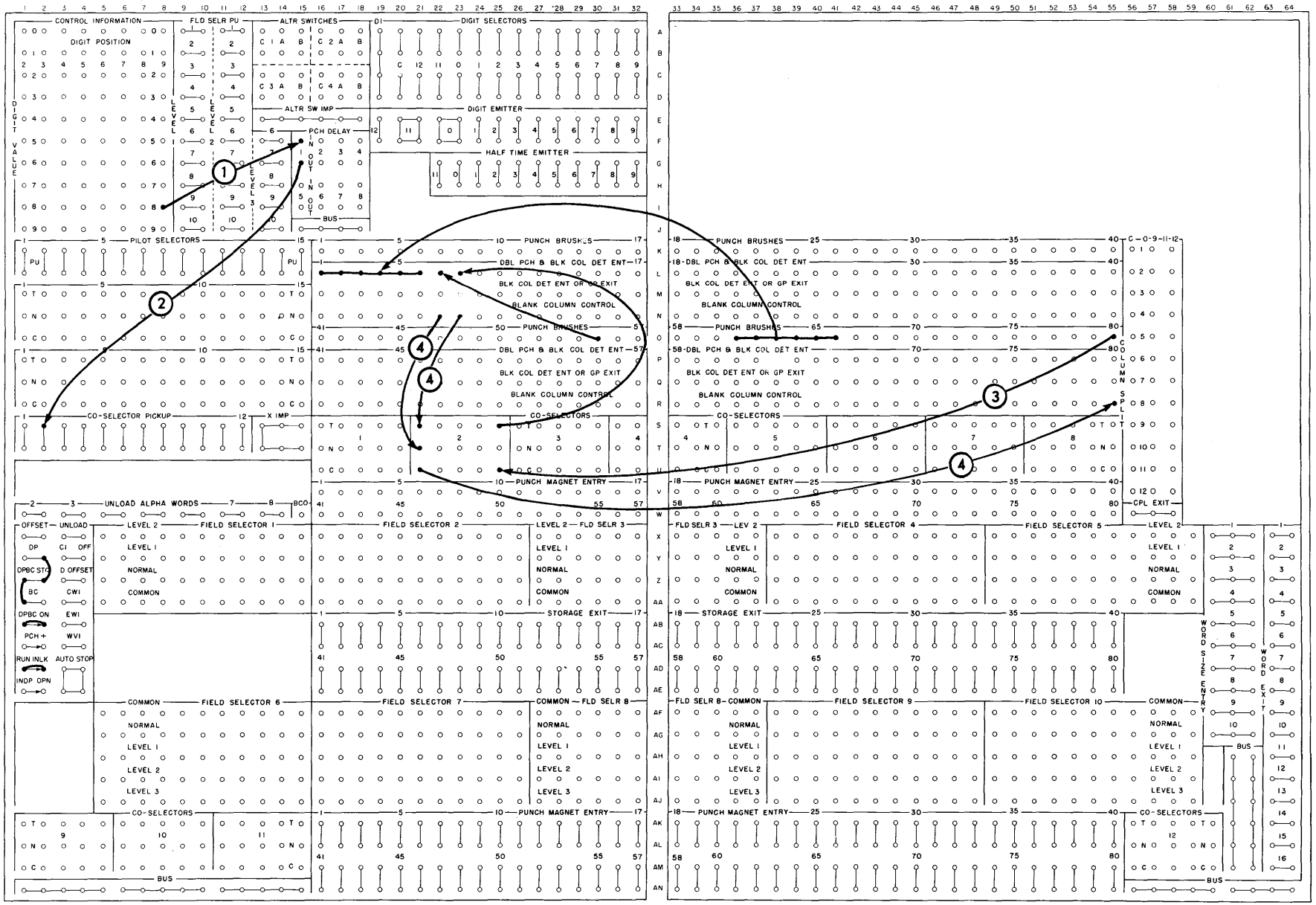
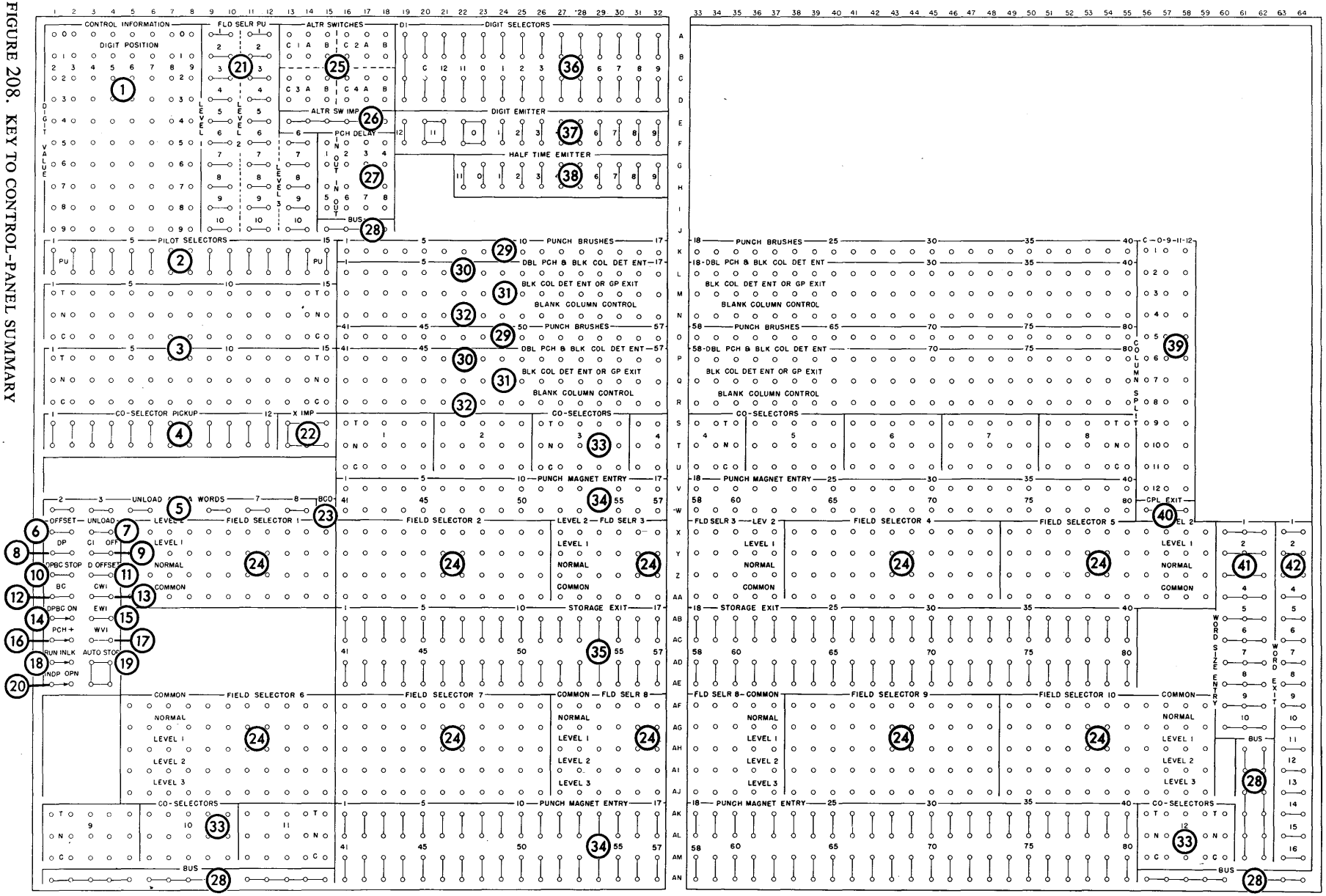


FIGURE 207. PUNCH DELAY WIRING

1. The CONTROL INFORMATION "8" impulse from position 9 is entered into the IN of PUNCH DELAY 1. This is at the beginning of the cycle during which the X-80 card is punched.
2. The output from PUNCH DELAY 1 is used to pick up co-selector 2. This is at the beginning of the cycle following the one during which the X-80 card was punched. Co-selector 2 remains transferred as the X-80 card passes the punch brushes.
3. The information from column 80 of the card is taken through co-selector 2 transferred to the DPBC DETECTION ENTRY. This checks column 80 for double punches and blanks. If the selector had been normal, no path would have existed between PUNCH BRUSH 80 and DPBC DETECTION ENTRY. Therefore, column 80 would not be checked for double punching.
4. This wiring shifts the last position for which blank-column detection is done. For the card identified by the control information impulse, blank-column checking must include that position to which the information from column 80 is wired. For the other card, this position must be bypassed.

FIGURE 208. KEY TO CONTROL-PANEL SUMMARY



7550 Card Punch Control Panel Summary

1. *Control Information.* The impulses emitted from these hubs represent the digit value in each of eight digit positions (2-9) in the control word (word 1 of the output synchronizer). These impulses can be used to pick up co-selectors, field selectors, pilot selectors, etc.
2. *Pilot Selector Pickup.* These hubs are normally wired from control information to cause the associated pilot selector to transfer on the same cycle. Pilot selectors remain transferred for the cycle during which they are impulsed.
3. *Pilot Selectors, T,N,C.* Whenever a pilot selector is transferred, an impulse wired to the C (common) of the selector is available from the corresponding T (transferred) hub. If the selector has not been impulsed at the pickup hub, an impulse wired to the C hub is available at the N (normal) hub.
4. *Co-Selector Pickup.* These are the pickup hubs for the co-selectors. When impulsed, they cause the co-selector to transfer immediately. They are normally wired from CONTROL INFORMATION.
5. *Unload Alphabetic Words.* These hubs are impulsed from CONTROL INFORMATION whenever information contained in words 2-8 is designated as alphabetic in an unloading operation.
6. *Offset.* An impulse to this hub causes the card being read at the punch brushes to offset in the stacker.
7. *Unload.* These hubs accept early-timed impulses to cause words 1-8 of the output synchronizer to be transferred automatically to punch magnets 1-80.
8. *D.P.—Double Punch.* This hub emits at the end of a cycle during which a double punch is detected. It is normally wired to AUTO STOP or OFFSET.
9. *CI Off—Control Information Off.* These hubs are normally impulsed from EWI to make the control information hubs inactive.
10. *DPBC Stop.* These hubs accept impulses from the DP and/or BC hubs to cause the machine to stop whenever a blank column or double punch is detected.
11. *D Offset—Delay Offset.* These hubs accept impulses to offset the card then at the punch magnets when it reaches the stacker.
12. *B.C.—Blank Column.* This hub emits an impulse at the end of a cycle during which a blank column is detected.
13. *CWI—Constant Write Impulse.* These hubs emit impulses on every punch cycle.
14. *DPBC ON—Double-Punch, Blank-Column Detection On.* This switch must be wired to activate the double-punch, blank-column detection circuits.
15. *EWI—Error Write Impulse.* This exit hub is directly under control of the 7070 program (UWIV, UPIV commands). The EWI hubs are normally used to pick up selectors, impulse AUTO STOP, etc.
16. *PCH+—Punch Plus.* This switch is wired when 12 impulses (plus signs) are to punch for positive numerical words.
17. *WVI—Write Validity Impulse.* These hubs emit an impulse if the punch validity check circuits detect a non-valid character. They are normally wired to AUTO STOP. If they are left unwired, the detection of an invalid character causes the validity check light on the punch to flash.
18. *Run Inlk—Run Interlock.* This switch must be connected to complete the punch run circuit. Its purpose is to prevent punch operation without a wired panel in place.
19. *Auto Stop.* These hubs accept any impulse to suspend punch operation at the end of the punch cycle.
20. *Indp. Opn.—Independent Operation.* This switch is wired whenever the punch is to be used as an independent unit.
21. *Fld Selr PU—Field Selector Pickup.* These hubs are impulsed to cause the corresponding level of the field selectors to transfer immediately. If a field selector is not impulsed to transfer at level 1, 2, or 3, the normal level is active.
22. *X-Impulse.* These hubs emit X-timed impulses during each punch cycle.
23. *BC0—Blank Column Zero.* These hubs are used whenever column 1 is to be bypassed for blank-column checking.
24. *Field Selectors.* There are ten 11-position field selectors provided. Field selectors 1-5 have three levels each (normal, level 1 and level 2); selectors 6-10 have four levels each (normal level 1, level 2, and level 3). When a field selector is impulsed to transfer at a given level, a connection exists between common and that level. If the field selector pickup has not been impulsed, the connection is between common and normal.
25. *Altr. Switches.* There are four alteration switch selectors on the control panel. They correspond to the four alteration toggle switches on the machine,

which can be set to A or B. The selectors may be used independently or in conjunction with co-selectors to change machine functions under the control of a toggle switch.

26. *Altr SW Imp—Alteration Switch Impulse.* These hubs provide an impulse that is normally wired through an alteration switch selector to the pickup of a co-selector to expand the alteration switch selector.
27. *Punch Delay.* These hubs are used to delay impulses for 1 to 8 punch cycles. The in hub accepts any punch-timed impulse. The corresponding out hub emits an impulse just prior to punching time on the punch cycle following the one during which the in hub was impulsed. The out hub of one punch delay unit can be wired to the in hub of another.
28. *Bus.* These hubs are located in convenient sections of the control panel and are used to expand either exits or entries, thereby eliminating the need for split wires.
29. *Punch Brushes.* These hubs are exits for digit impulses (9-12), read from the output card one cycle after the card has been punched. They are normally used for double-punch, blank-column checking and/or gangpunching.
30. *DP and BC Det Entry—Double-Punch and Blank-Column Detection Entry.* These are entry hubs for information being read by the punch brushes. They are used to check for double punches and blank columns in output cards.
31. *Blk Col Det Entry or GP Exit—Blank-Column Detection Entry or Gangpunch Exit.* These hubs are both entries and exit hubs. As entries they receive impulses to check output cards for blank columns. As exits they are wired for gangpunching information wired to the DP and BC detection entry hubs.
32. *Blank-Column Control.* These hubs provide a means for bypassing one or more positions that are not to be checked for blank columns. The position preceding the first position to be bypassed is wired to the last position to be excluded.
33. *Co-selectors.* Twelve co-selectors are standard. Each selector has five positions. When the pickup hubs are impulsed, the selector transfers immediately and holds to the end of the same cycle.
34. *Punch Magnet Entry.* Three sets of hubs are multiple entries to the 80 punch magnets. Digit impulses wired to these hubs punch in the corresponding card columns.
35. *Storage Exits.* These are exits for as many as 80 characters from the sixteen words contained in the output-synchronizer. They are normally wired to the punch magnet entry hubs. The sign of a word (11, 12) emits from the low-order position of the word. PCH + must be wired if 12 punches (+ signs) are desired.
36. *Digit Selectors.* The DI hub emits a series of impulses that are timed for punching the digits 9-12. If the DI hub is wired to the C hub beside it, the digit selector becomes a digit emitter. If a storage exit position is wired to the C hub, whatever digit appears at that position is available at the corresponding hub of the digit selector.
37. *Digit Emitter.* These hubs emit punch digit impulses on every punch feed cycle. They are used to punch data that is not available from the storage exits.
38. *Half-Time Emitter.* These are off-time emitter hubs that emit an impulse at half after the number they represent. They are normally wired to pick up selectors that are then used as column splits.
39. *Column Splits.* This is a 12-position selector that is controlled automatically to transfer between 0 and X time as the card is punched. This allows the 9-0 punches to be separated from 11-12 punches for punching, pilot, field and co-selector control, etc.
40. *Column Split Couple.* An impulse is emitted from these common hubs between 0 and X punch time. This impulse is used to pick up co-selectors to expand the column split feature.
41. *Word Size Entry.* These hubs are wired from the word exit hubs to cause specific numbers of characters to be emitted from the storage exit hubs for each output word.
42. *Word Exit.* These 16 hubs emit sequentially as words 1-16 are transmitted from the output-synchronizer for punching. These impulses are always wired to the word size entry hubs to cause the proper number of characters to emit from storage.

IBM 7400 Printer

One to three IBM 7400 Printers (Figure 209) can be attached to one IBM 7070 Data Processing System. The 7400 Printer prepares printed reports from data transmitted from the system through the output synchronizer. The machine prints information from 120 print wheels that form a solid bank 12 inches wide. Each print wheel has 47 different characters, all the letters of the alphabet, all the numbers, and 11 special characters.

The 7400 can print a maximum of 150 lines a minute, or 9000 lines an hour.

Printing with internally wired format control is possible by use of the unload hubs on the control panel. On the 7400, validity checking can be suspended to allow printing after an invalid character has been detected.

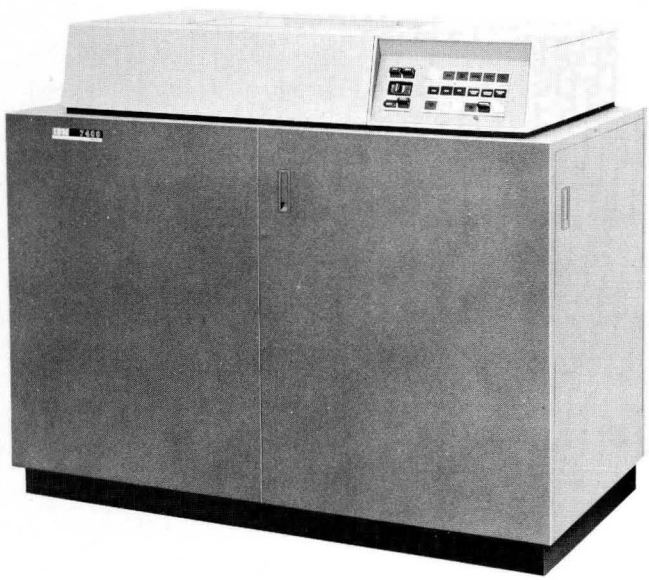


FIGURE 209. 7400 PRINTER

SPOOL (Simultaneous Peripheral Operation On Line) operations can be performed when priority signals for automatic program control are assigned by console settings.

Forms can be positioned in the machine automatically by the use of the carriage, which is set up for operation by inserting a prepunched paper tape in the tape-control mechanism.

It is possible to signal the program unit from the printer. A hole in channel 9 in the carriage tape allows a signal, which sets up a control in the 7070, to be tested by the program. This enables the stored program to know when a certain line on the printed form has been reached and to branch to a sub-routine for obtaining page totals, controlling forms skipping, etc. The control is reset on the next print command.

The functions of the machine are illustrated by examples with samples of data contained in the output synchronizer, reports, and wiring diagrams.

Print Unit

Printing on the IBM 7400 is accomplished by means of 120 printwheels arranged in a solid bank, which prints within a width of 12 inches, 10 characters to the inch. Each printwheel contains the following 47 separate character positions. *Note:* This is one printwheel configuration that can be specified for the 7400. Other character configurations are available.

10 digits: 0 through 9
26 letters: A through Z
11 special characters:
/ $ ⌑ * % @ — # . , +

As shown in Figure 210, the printwheel is divided into twelve equal parts:

Digits 1 through 9	9 parts
Combination of the digits 8 and 3 in one column	1 part
Combination of the digits 8 and 4 in one column	1 part
Zone only	1 part

Each of the twelve parts is in turn divided into four sections:

0 Zone
11 Zone
12 Zone
N (No) Zone

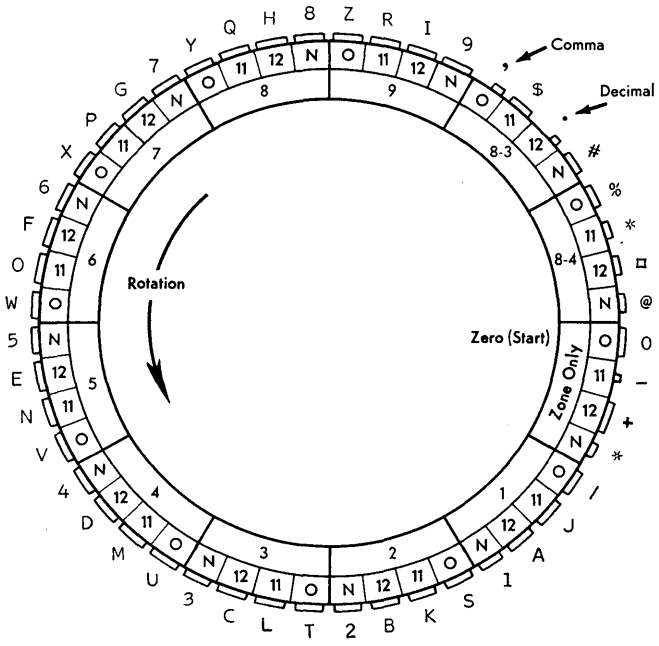


FIGURE 210. 7400 PRINTWHEEL SCHEMATIC

As shown in Figure 211, the 0, 11, and 12 zones control the printing of 26 letters, zero, and nine special characters. The N (No) zone controls the printing of nine digits and two special characters. An additional special character (*) position is provided for check protection.

The printwheels remain stationary until the digit emits, at which time one of the twelve sections is selected. A further selection of one of the four parts within that section is made when the zone is emitted. The printing wheel rotates at a high rate of speed until printing time, when its speed is reduced to 25 per cent of normal. At the actual time of printing, the wheel is moved forward against the platen in a straight line motion, which produces maximum legibility. The rotary motion of the wheel at print time is compensated for by a special cam.

LOWER PUNCH	ZONE 12	11	O	N
1	A	J	/	1
2	B	K	S	2
3	C	L	T	3
4	D	M	U	4
5	E	N	V	5
6	F	O	W	6
7	G	P	X	7
8	H	Q	Y	8
9	I	R	Z	9
8-3	.	$	,	#
8-4	□	[1] *	%	@
	+	−	0	[2] *

1. Total Symbol 2. Check Protection

FIGURE 211. CHARACTER CODES

Although one line is printed on one cycle, the wheels print at four different times within that cycle. All of the wheels zoned for 0 print first, followed in succession by those for 11, 12, and N.

Spacing Chart

The best way to determine which printwheels to use to print a field in the report column set aside for it is to superimpose the report itself on a spacing chart, as shown in Figure 212. The numbers across the top and bottom of the spacing chart represent printwheel positions, which are spaced 10 to the inch or the same as standard pica typewriter spacing. The large numbers represent the tens position, and the small numbers represent the units position of the printwheel number. The numbers down the sides of the spacing chart represent line numbers.

The report can be superimposed on the spacing chart. If the report is positioned as shown in Figure 212, then product number is printed by printwheels 34-37; schedule date, by printwheels 39-46; and order number, by printwheels 48-52. In this example, the report is centered, with the form alignment symbol (▽) in the center of the report lined up with printwheel 60 in the center of the spacing chart.

Operating Keys and Signal Lights

The keys and lights are located on the right front of the printer (Figure 213).

MASTER SWITCH: This switch controls the power supply for the IBM 7400.

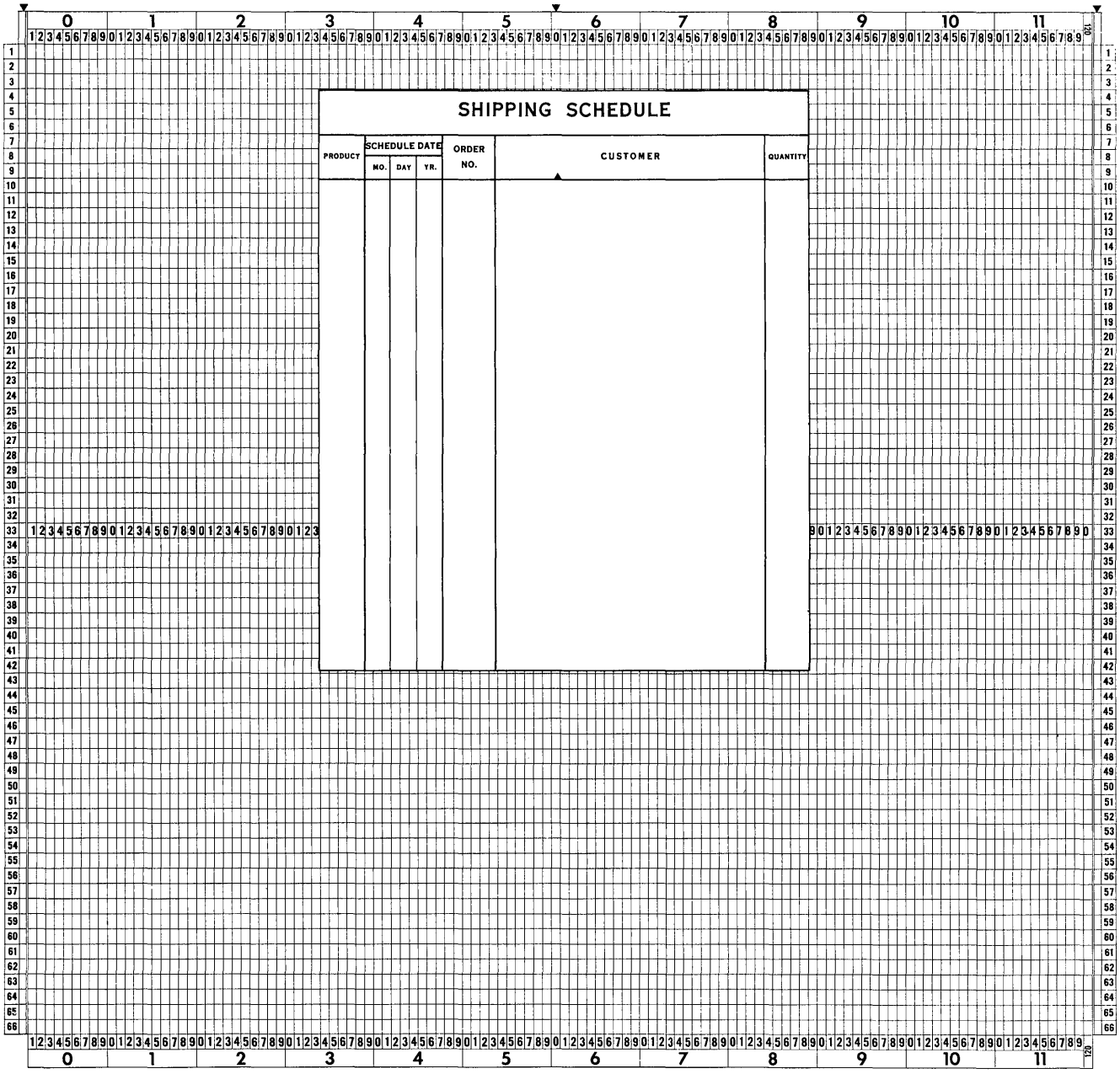


FIGURE 212. REPORT AND SPACING CHART

START KEY: The start key is pressed to condition the printer to accept data from the output-synchronizer. It is operative only if the ready light is on.

STOP KEY: If the 7400 is in condition to accept print commands, pressing the stop key nullifies this condition and stops the motor.

RESET KEY: This key is used to extinguish lights caused by certain conditions that have stopped the machine. After the condition has been noted by the operator, the light can be extinguished by pressing the reset and start keys.

POWER ON LIGHT: This light is turned on when power is supplied to the 7400.

READY LIGHT: This light glows when the form has been positioned, all error detecting devices are reset, and the printer is ready to respond to pressing the start key.

RING CHECK LIGHT: This light turns on when the number of characters emitted by the storage exit hubs on the control panel does not total 80.

CLOCKING CHECK LIGHT: The machine makes an internal check to insure that certain timing circuits are operating properly. If these circuits fail, the machine stops and the clocking check light comes ON, indicating a possible error. This light is extinguished by pressing the reset key. Repeated

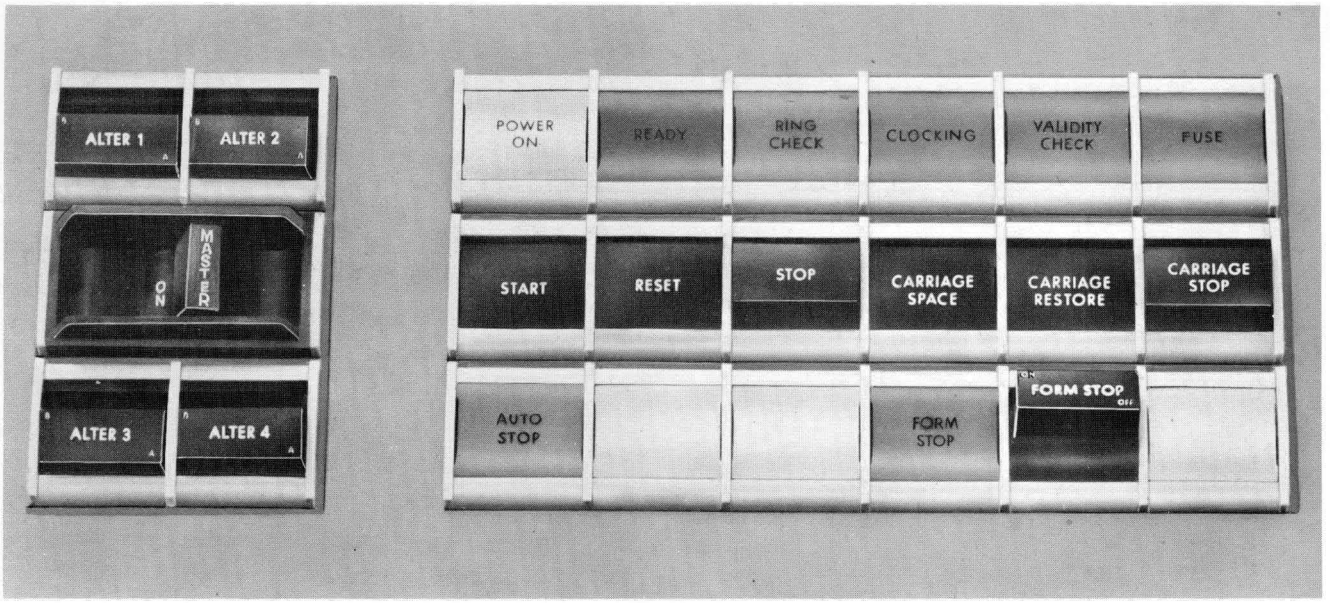


FIGURE 213. OPERATING KEYS AND SIGNAL LIGHTS

clocking check lights indicate that the machine should be checked by a customer engineer.

VALIDITY CHECK LIGHT: This light turns on when the validity-check circuits detect an invalid character. If WVI is not wired to AUTO STOP, the light goes out automatically. If WVI is wired to AUTO STOP, the machine stops. Turn off the light by pressing the reset key, and restart the machine by pressing the start key.

AUTO STOP: This light glows whenever the printer stops as a result of a condition wired on the control panel to AUTO STOP. The machine is restarted by pressing the reset and start keys.

FUSE LIGHT: This light turns on and the machine stops whenever a fuse burns out. The light is turned off when the master switch is turned off.
Note: Turn off the 7400 when replacing the fuse.

FORM STOP SWITCH: When the form stop switch is ON, the end of form stop, located in the center of the carriage, is operative and causes the printer to stop when the last form is within 13⅜ inches of the printing line. When the form stop switch is OFF, the end of form stop is inoperative.

FORM LIGHT: The form light goes ON and the printer stops whenever the last form is within 13⅜ inches of the platen, provided the form stop switch is ON. The form light is turned off by (1) inserting a new form, and (2) pressing the start key, or by turning off the form stop switch.

CARRIAGE SPACE KEY, CARRIAGE RESTORE KEY, AND CARRIAGE STOP KEY: These three keys are placed at the front of the machine for operator convenience. Their functions are exactly the same as the carriage control keys located beneath the top cover. They are described in the section entitled *Tape-Controlled Carriage, Operating Features.*

Control Panel

The IBM 7400 Control Panel (Figure 214) has been designed to allow maximum flexibility for printing documents and reports as output for the IBM 7070 Data Processing System. Data are transmitted from the output-synchronizer and arranged by control-panel wiring for printing. All features and functions described in this section are included as standard equipment on the IBM 7400.

Planning Chart

The reverse side of the 7400 Printer control-panel diagram is the IBM 7400 Printer planning chart (Figure 215). If all notes that pertain to a specific printer output format are written in the appropriate sections on this chart, the job of wiring the control panel is greatly simplified. Keep this chart for control panel testing, for a permanent record of the job for which that control panel was designed.

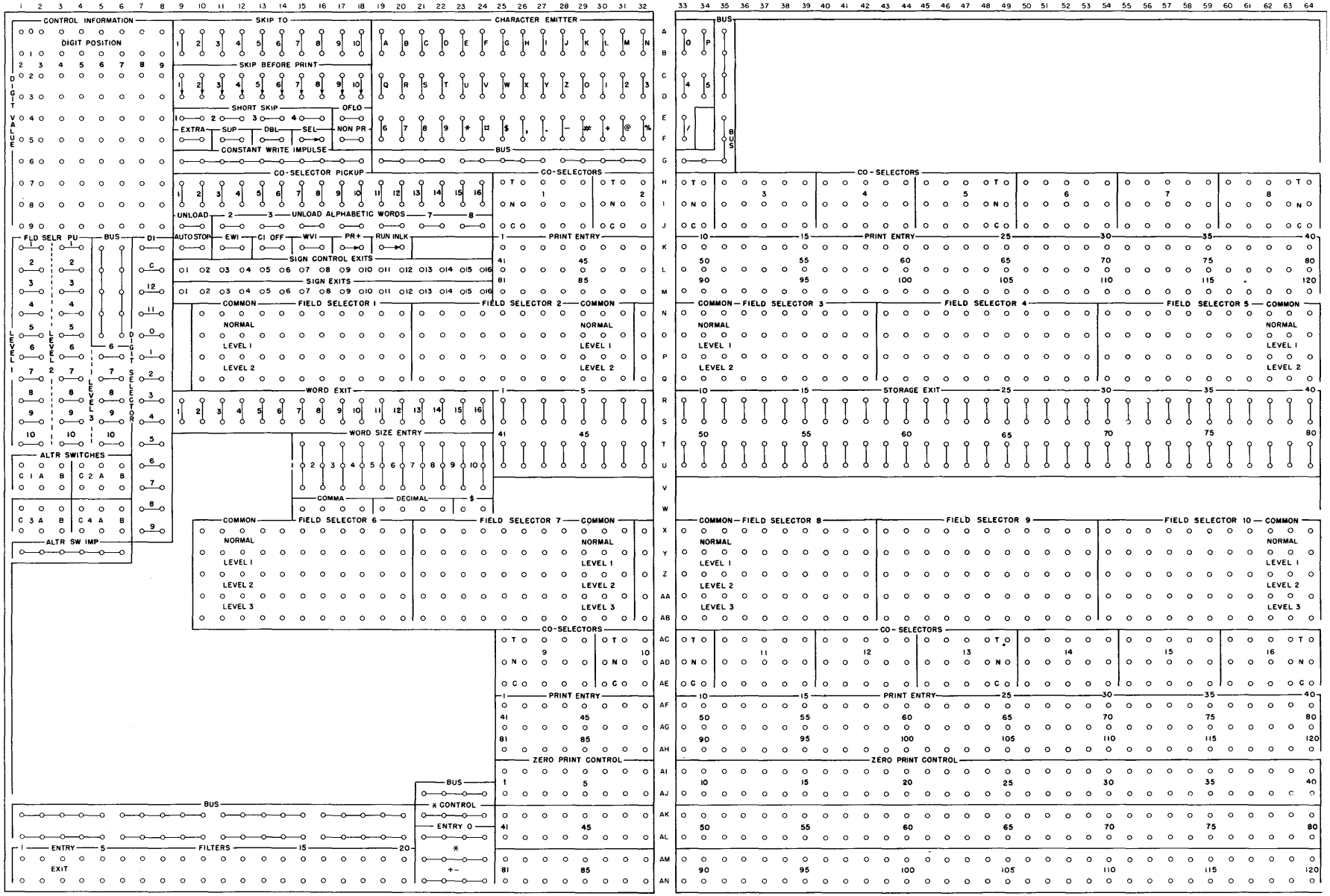


FIGURE 214. 7400 CONTROL PANEL (FORM X24-6426)

FIGURE 215. PRINTER PLANNING CHART

IBM 7400 PRINTER PLANNING CHART

PREPARED BY _____ DATE _____

REPORT NAME _____ APPLICATION _____

WORD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	REMARKS
CONTROL INFORMATION																	
NUMBER OF CHARACTERS TO BE PRINTED																	
SIGN CONTROL EXITS																	
SIGN EXITS																	
REPORT HEADINGS																	
STORAGE EXIT POSITIONS																	
PRINT POSITIONS																	
ZERO PRINT CONTROL																	

FIELD SELECTORS	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	REMARKS
	3	3	3	3	3	3	3	3	3	3	
	2	2	2	2	2	2	2	2	2	2	
	1	1	1	1	1	1	1	1	1	1	
	N	N	N	N	N	N	N	N	N	N	
	C	C	C	C	C	C	C	C	C	C	

CO-SELECTORS	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	REMARKS
	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	
	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	

MISCELLANEOUS

PR+ ☐

ALTERATION SWITCHES

	A	B		A	B
1	☐	☐	3	☐	☐
2	☐	☐	4	☐	☐

CARRIAGE SKIPS

1	_____	6	_____
2	_____	7	_____
3	_____	8	_____
4	_____	9	_____
5	_____	10	_____

SPACING

DBL SPACE _____

EXTRA _____

SUP _____

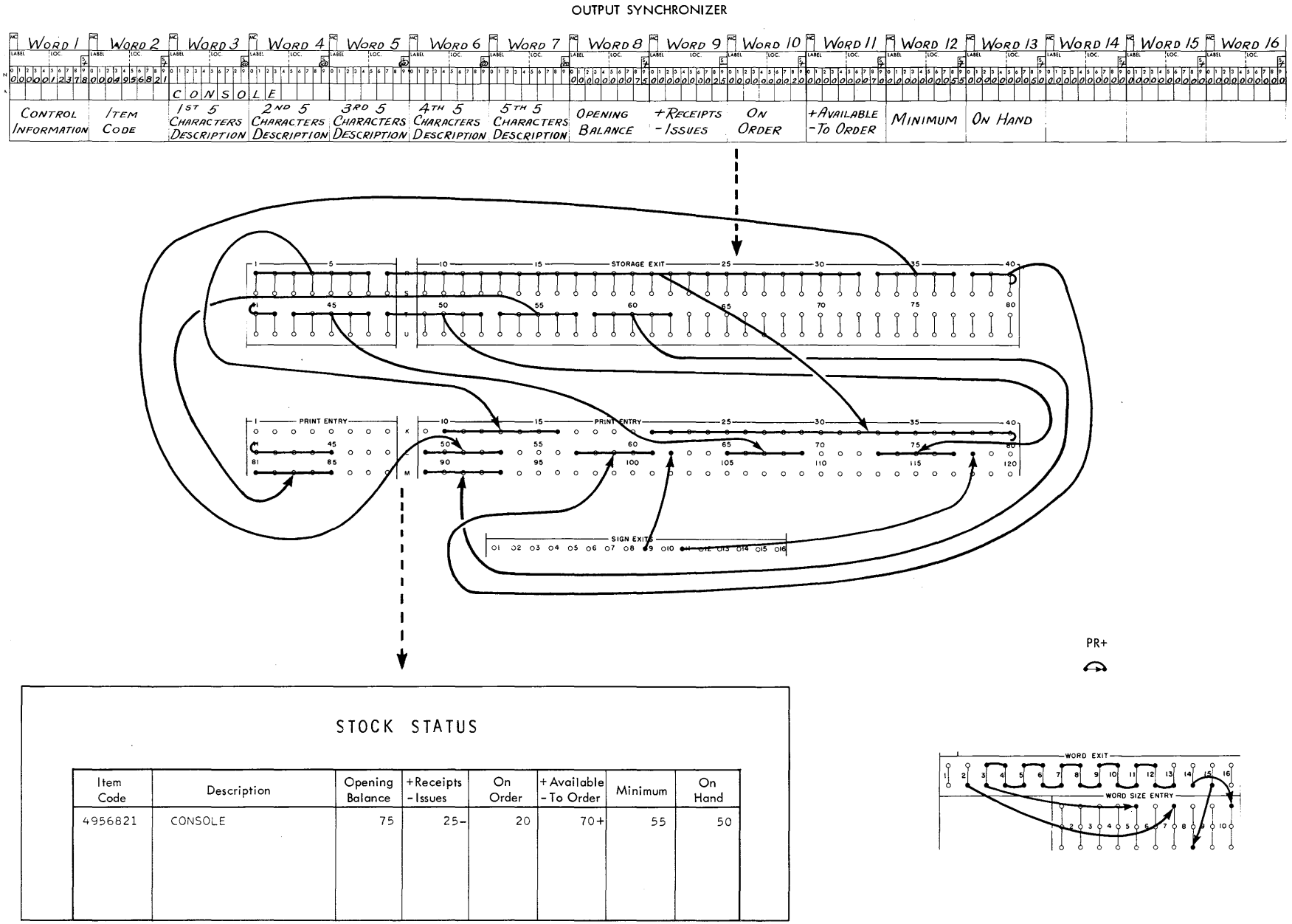


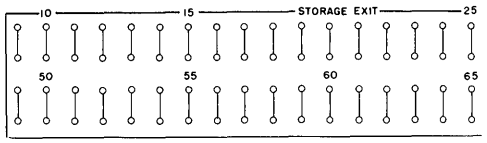
FIGURE 216. OUTPUT DATA FLOW

STOCK STATUS

Item Code	Description	Opening Balance	+Receipts -Issues	On Order	+Available -To Order	Minimum	On Hand
4956821	CONSOLE	75	25-	20	70+	55	50

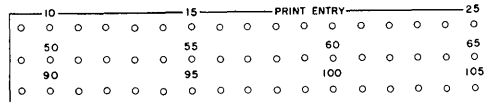
Control-Panel Hubs

R-U, 25-64

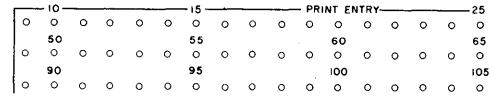


STORAGE EXITS: As many as 80 characters from the sixteen words contained in the output-synchronizer can be controlled to exit through these 80 hubs. STORAGE EXIT impulses are normally wired to the PRINT ENTRY hubs. Sign control is explained in a later section.

K-M, 25-64

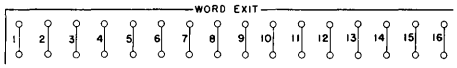


AF-AH, 25-64



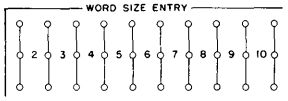
PRINT ENTRY: These hubs are entries to the 120 print wheels for impulses emitted by the STORAGE EXITS or the EMITTER. When a print entry hub is impulsed, the printwheel prints the numerical, alphabetic or special character information that emits from the storage exit wired to it. The print entry hubs can also be wired from the SIGN EXITS, the character emitter, and the comma, decimal, and dollar exit hubs.

R-S, 9-24



WORD EXIT: These hubs (1-16) are used to indicate the number of significant character positions assigned to each output word to be printed. They are normally wired to word size entry hubs.

T-V, 15-24



WORD SIZE ENTRY: These hubs, labeled 1-10, are wired from the word exit hubs to cause the actual number of character positions that comprise a specific output word to emit from the storage exit hubs. *Note*: All 80 positions of storage must be wired to emit impulses. If less than eighty positions emit, the ring check light on the printer comes ON and the machine stops.

Output Data Flow

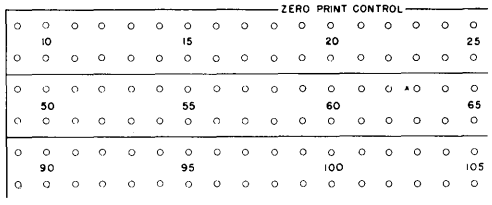
EXAMPLE: Figure 216 shows data flow from the output-synchronizer to the printed report. Figure 217 is the planning chart for the operation. Data are assembled in words 1-16 of the output-synchronizer through programmed instructions. On a print command, these data are available at the storage exit hubs on the control panel. Data are arranged and printed by wiring from the STORAGE EXITS to the print entry hubs that correspond to specific printing positions on the report. The sign associated with a given output word is emitted for printing from the sign exit hub of that word. The number of character positions to be printed for each word is controlled by the wiring from WORD EXITS to WORD SIZE ENTRY. Unused words in the output-synchronizer are automatically filled with zeros, the alphabetic special-character code for blank columns, and the signs are set to alpha.

Zero Print Control

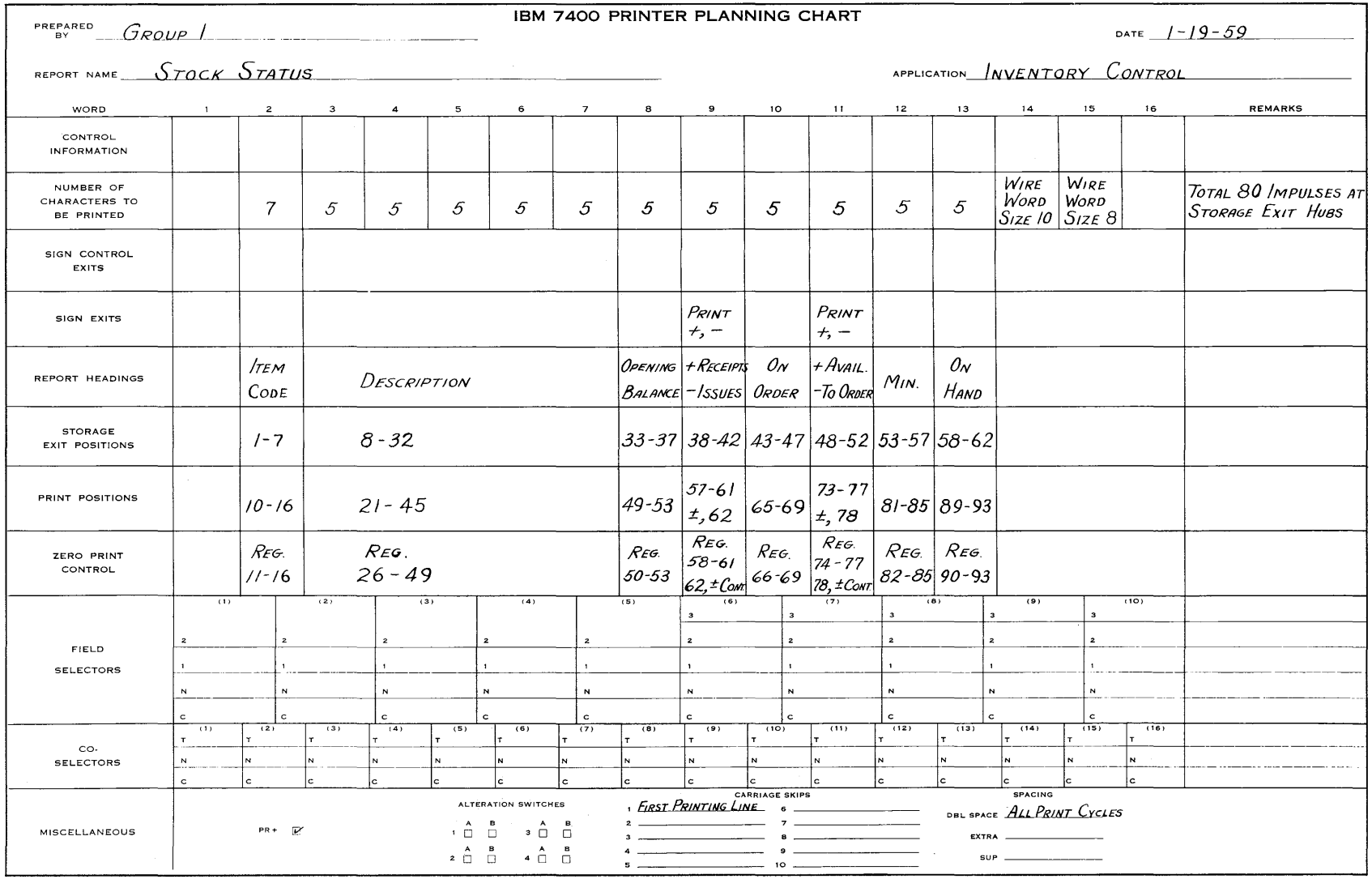
Printwheels are designed to print normally only the significant digits 1 to 9 and any letter or special character with a 1 to 9 combination.

The impulse, or impulses, wired to print entry must be able to reach the fuse to energize the print magnet and cause printing. There is an internal connection between the print magnet and the fuse at the time a digit 1 through 9 is emitted. However, that connection is broken after 1 time and before zero time, so that a 0, 11, and 12 cannot normally reach the fuse. A character that does not have a digit 1 to 9 does not normally reach the fuse for the same reason. Therefore zeros do not normally print. For example, a number like 010050 would print as (1 5). Zero printing is controlled by use of the zero print control hubs on the control panel. These hubs are described first, for controlling zeros, and second, for controlling the comma, decimal, dollar sign, plus (+), and dash (—) symbols.

AI-AN, 25-64



ZERO PRINT CONTROL: Each print entry position has a pair of zero print control hubs diagonally ar-



IBM 7400 PRINTER PLANNING CHART

PREPARED BY: GROUP 1 — DATE: 1-19-59

REPORT NAME: STOCK STATUS — APPLICATION: INVENTORY CONTROL

WORD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	REMARKS
CONTROL INFORMATION																	
NUMBER OF CHARACTERS TO BE PRINTED		7	5	5	5	5	5	5	5	5	5	5	5	WIRE WORD SIZE 10	WIRE WORD SIZE 8		TOTAL 80 IMPULSES AT STORAGE EXIT HUBS
SIGN CONTROL EXITS																	
SIGN EXITS									PRINT +, −		PRINT +, −						
REPORT HEADINGS		ITEM CODE	DESCRIPTION					OPENING BALANCE	+RECEIPTS −ISSUES	ON ORDER	+AVAIL. −TO ORDER	MIN.	ON HAND				
STORAGE EXIT POSITIONS		1-7	8-32					33-37	38-42	43-47	48-52	53-57	58-62				
PRINT POSITIONS		10-16	21-45					49-53	57-61 ±,62	65-69	73-77 ±,78	81-85	89-93				
ZERO PRINT CONTROL		REG. 11-16	REG. 26-49					REG. 50-53	REG. 58-61 62, ±CONT	REG. 66-69	REG. 74-77 78, ±CONT	REG. 82-85	REG. 90-93				

FIELD SELECTORS	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
	3	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	2	2	2	2	2
	1	1	1	1	1	1	1	1	1	1
	N	N	N	N	N	N	N	N	N	N
	C	C	C	C	C	C	C	C	C	C

CO-SELECTORS	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C

MISCELLANEOUS: PR+ ☑

ALTERATION SWITCHES: 1 A ☐ B ☐ — 2 A ☐ B ☐ — 3 A ☐ B ☐ — 4 A ☐ B ☐

CARRIAGE SKIPS: 1 FIRST PRINTING LINE; 2; 3; 4; 5; 6; 7; 8; 9; 10

SPACING: DBL SPACE ALL PRINT CYCLES; EXTRA; SUP

FIGURE 217. PLANNING CHART

ranged in two rows as illustrated. The hubs in the lower row are numbered from 1 to 120 on the control panel to correspond to the print entry positions. The hubs in the upper row are not numbered on the control panel, but they are associated diagonally with the hubs in the lower row.

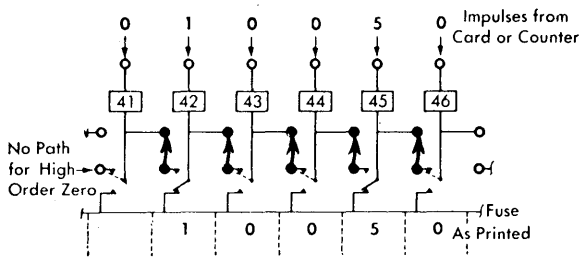


FIGURE 218. ZERO PRINT CONTROL—SCHEMATIC AND ZEROS TO RIGHT OF SIGNIFICANT DIGIT ONLY

The principle of zero print control can be explained by reference to Figure 218. Six zero print positions are shown for printwheels 41 to 46. It is assumed that STORAGE EXITS 1 through 6 (010050) are wired to PRINT ENTRY 41 through 46.

Each printwheel has a corresponding print magnet, which must be energized before a zero or a significant digit will print.

When a print entry position has been impulsed with a 9 to 1 digit (printwheels 42 and 45), the zero print contacts transfer and the connection between the two diagonal hubs is broken. The upper hub and print magnet are then connected directly to the fuse so that it can be used as an entry to the fuse. In positions that receive no significant digit (1 to 9) impulse, the zero print contacts do not transfer. A 0, 11, or 12 impulse entering that position does not find a path to the fuse but is available out of the lower hub. Printing of zeros can be accomplished in the desired pattern by correctly wiring the zero print control hubs on the control panel. For example, if the lower hub of print magnet 46 is jackplugged to the upper hub of 45, the zero impulse in position 46 goes through the normally closed zero print contact by control-panel wire to the upper hub of 45 and then to the fuse. The upper hub of 45 was connected directly to the fuse because the 5 impulse caused the zero print contact of that position to be transferred.

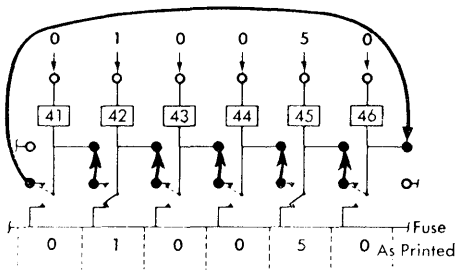


FIGURE 219. ZERO PRINT CONTROL—ZEROS TO LEFT OF SIGNIFICANT DIGIT

This arrangement affords complete flexibility in printing zeros. In the example above, the number may be wired to print either as 010050 or as 10050.

If zeros are to print only to the right of the significant digits, the zero print control hubs are wired for each printwheel in use except the position of the highest order, as shown in Figure 218. The zeros from hubs 43 and 44 reach the fuse over the external wires connecting 44 and 43 and the internal connection established because of the digit 1 in printwheel 42. The high-order zero cannot reach the fuse and therefore does not print.

Zeros can be printed to the left of significant digits as well as to the right by the wiring shown in Figure 219.

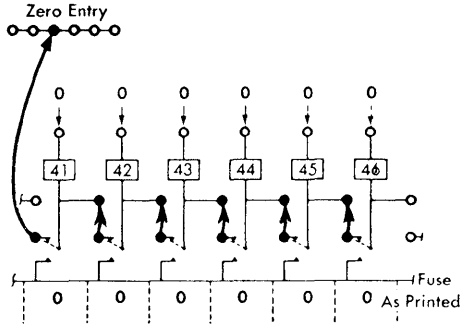
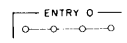


FIGURE 220. ZERO PRINT CONTROL—ZERO REGARDLESS OF SIGNIFICANT DIGIT

The high-order zero prints because it reaches the fuse the same way that the zero in the units position does. Zeros do not print, however, unless there is a significant digit in some position.

AL, 21-24 

ENTRY 0: These six common hubs provide a direct path to the fuse for zeros. They are normally wired from zero print control as shown in Figure 220 to cause zeros to print regardless of significant digits. As there are no significant digits in the field, every

lower zero print control hub is internally connected to its corresponding upper hub. The external wiring allows a zero anywhere in the field to reach the fuse by way of the entry 0 hubs.

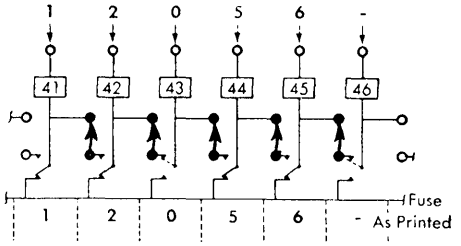


FIGURE 221. ZERO PRINT CONTROL—MINUS AND PLUS SIGNS

Zero print control need not be wired for alphabetic fields unless interspersed numerical or special-character information is to be printed. Five of the special characters (— + , . $) are controlled from zero print control.

ZERO PRINT CONTROL FOR MINUS (—) AND PLUS (+) SIGNS: Because zone impulses, by themselves, do not have a path to the fuse (11 for dash and 12 for plus) both the dash and the plus must be wired for zero print control and print under the same conditions in which the zero prints. It is seen in Figure 221 that the dash impulse reaches the fuse because of the presence of significant digits to the left. If the field emits all zeros, the dash does not print.

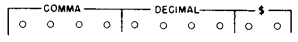
ZERO PRINT CONTROL—COMMA, DECIMAL, AND DOLLAR SIGN: There are three ways of printing commas, decimal points, and dollar signs:

a. From STORAGE EXITS to PRINT ENTRY.

b. From the corresponding hubs in the CHARACTER EMITTER to PRINT ENTRY.

c. From the COMMA, DECIMAL, and DOLLAR SIGN exits to PRINT ENTRY.

Under method A, the symbols print whenever they emit regardless of significant digits. Under method B, the symbols print every print cycle regardless of significant digits. Under method C, the symbols are controlled by zero print control.

W, 15-24 

COMMA, DECIMAL, $: These hubs emit comma, decimal, and dollar sign impulses every print cycle. They are specially timed so that they can be printed only under control of the zero print hubs, and in this way they differ from the comma, decimal, or dollar-sign impulses obtained from storage exits or from the emitter. They are normally wired directly to PRINT ENTRY to print the dollar sign and amount punctuation. They print with exactly the same type of control as that used for printing zeros; that is, they may be controlled to print to the right or left of significant digits by properly wiring the zero print control hubs.

The comma, decimal, and dollar sign hubs should never be split-wired directly to two or more different print entry hubs. If split-wiring is necessary, wire each split through a filter.

DOLLAR SYMBOL: The dollar symbol is normally printed to the left of significant digits. To do this, the same zero print control wiring must be used as that needed for printing zeros to the left of significant digits. Assuming that the $ hub is wired to PRINT ENTRY 40 and the amount field to PRINT ENTRY 41-46, the dollar sign prints to the left of significant digits by wiring lower ZERO PRINT CONTROL HUB 40 to the upper hub of ZERO PRINT CONTROL 46 (Figure 222). In this example, the dollar symbol reaches the fuse because of the digit 5. The high-order zero print control hub (41) is not wired, in order to prevent printing zeros between the dollar sign and the high-order digit 1.

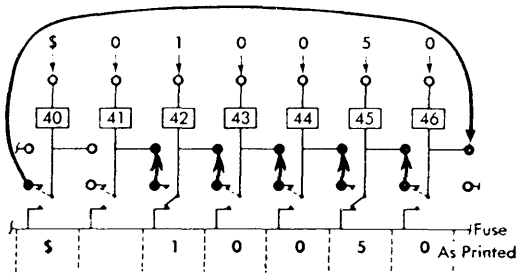


FIGURE 222. ZERO PRINT CONTROL—DOLLAR SYMBOL

COMMA: The comma is normally printed to the right of significant digits. Printing commas requires no special wiring other than that required for printing zeros. A comma impulse from the special-character emitter does not have a path to the fuse and, therefore, its printing depends upon the presence of significant digits. The comma shown in Figure 223 reaches the fuse because of the digit 2.

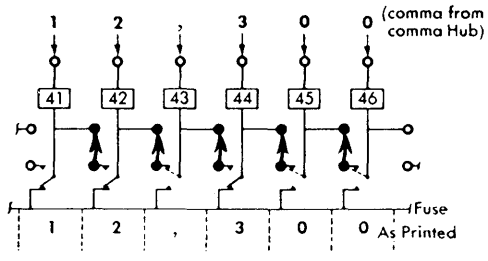


FIGURE 223. ZERO PRINT CONTROL—COMMA

DECIMAL: There are two methods of wiring for decimal printing as illustrated in Figures 224 and 225.

The decimal point prints to the right of significant digits the same as zeros, with normal zero print control wiring. This means that for amounts of 1.00 or over, normal zero print control wiring is sufficient. A decimal impulse does not have a path to the fuse and, therefore, its printing depends upon the presence of significant digits. The decimal shown in Figure 224 reaches the fuse because of the digit 1 in the hundreds position.

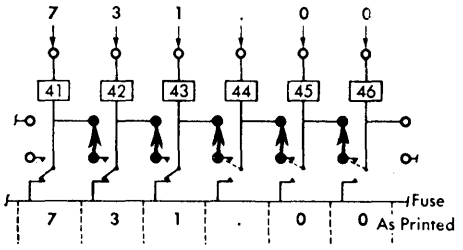


FIGURE 224. ZERO PRINT CONTROL—DECIMAL FOR AMOUNTS OF 1.00 OR MORE

For amounts of 1 to 99 cents, however, normal wiring fails to print the zero and the decimal point to the left of the significant digits. This can be accomplished by wiring the lower hub of the decimal position back to the upper hub of the units position as shown in Figure 225. Both the zero and the decimal in this example print because of the presence of the digit 1. In order to print an amount such as 1.00, it would appear that lower hub 44 should be split-wired to upper hub 43.

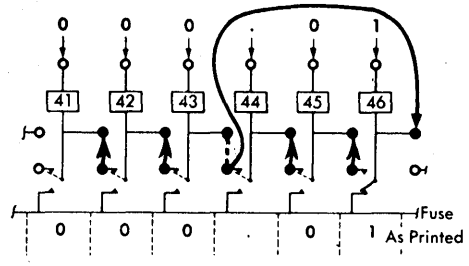
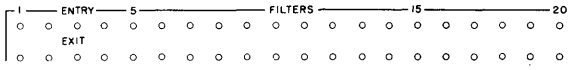


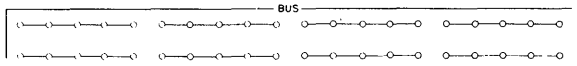
FIGURE 225. ZERO PRINT CONTROL—DECIMAL FOR AMOUNTS UNDER 1.00

Although this wiring would cause zeros to print to the right of even dollar amounts, it would also cause zeros to print to the left of any amount under 1.00. Any zero to the left of the decimal could find its way to the fuse over the wiring originally intended to print decimals and zeros for amounts of from 1 to 99 cents. Although the connection shown by the dotted line in Figure 225 must be made, it cannot be made directly. To prevent zeros in the dollar columns from backing up and reaching the fuse by way of connections established by the units and tens positions, use a filter.



AK-AL, 1-20

FILTER ENTRY-EXIT: These hubs permit the passage of an impulse in only one direction—into entry and out of exit. When any two functions are connected by split wires or through a bus, any impulse reaching one function also reaches the other. This back circuit can be eliminated by a filter. Twenty entry and twenty exit positions are standard. Do not wire the exit of one filter to the entry of another.



AM-AN, 1-20

BUS: When an impulse is entered into one of the hubs in a set, it is available out of the remaining hubs. Bus hubs are used in place of split wires.

Figure 226 shows the connection between lower 44 and upper 43 made through a filter. This allows zeros and the decimal for even dollar amounts to reach the fuse because of the presence of a significant digit anywhere to the left. It prevents zeros in 41, 42, 43, or 44 from reaching the fuse for amounts ranging from .01 to .99, since an impulse cannot pass through a filter from exit to entry. In such operations, one filter should be used for every two positions to the right of the decimal to prevent an undue load on one filter.

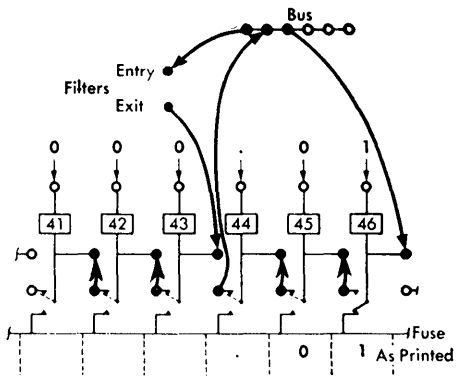
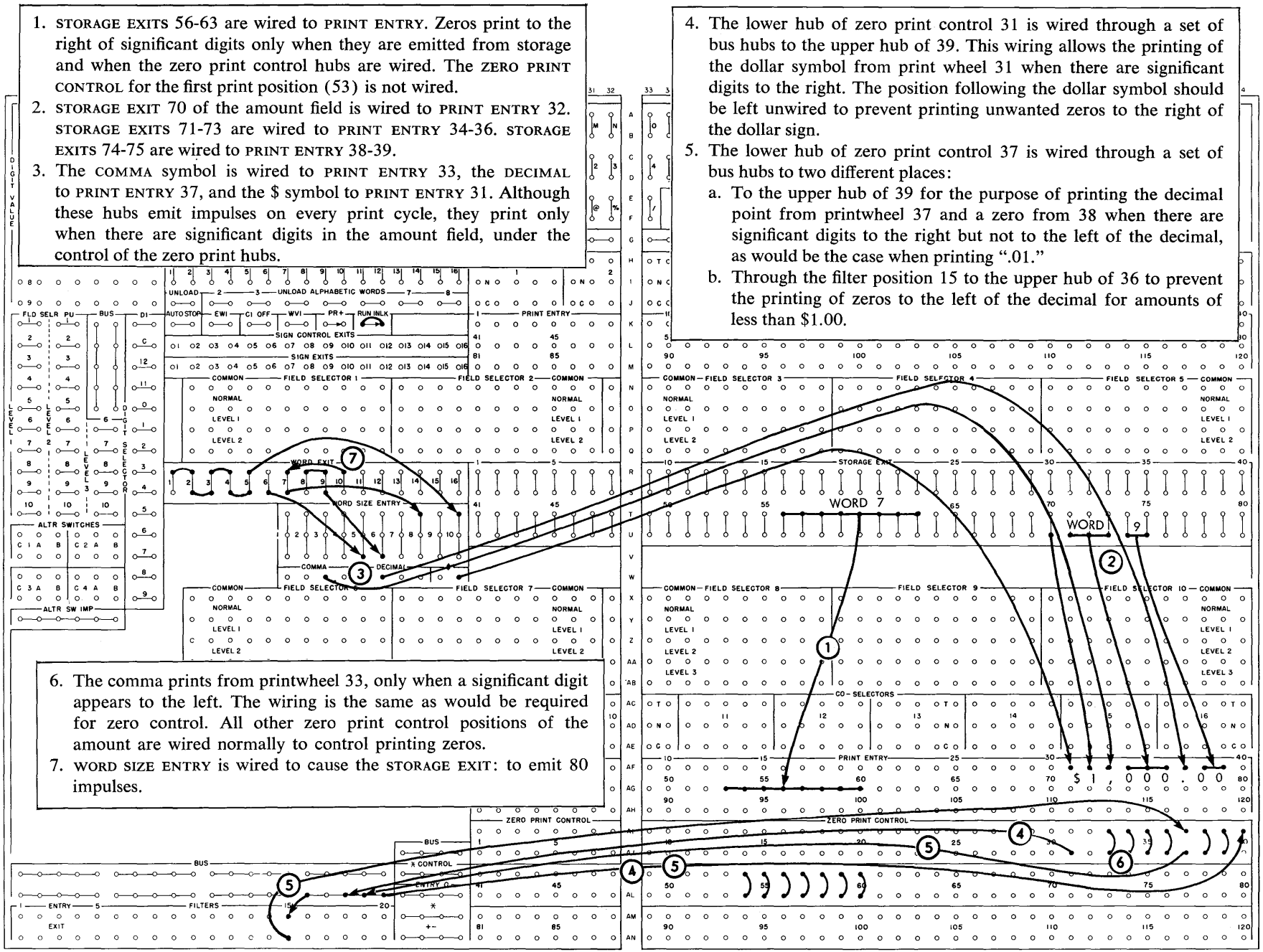


FIGURE 226. ZERO PRINT CONTROL—DECIMAL FOR ALL AMOUNTS

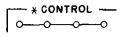
In Figure 227 only that wiring is shown that concerns printing amounts, zeros, commas, decimals, dollar signs, and word size.

FIGURE 227. WIRING FOR ZERO PRINT CONTROL



PRINT CONTROL FOR ASTERISKS AND PLUS OR MINUS SIGNS:

6	7	8	9	10	11	12	13
*	9	2	5	4	.	9	2
*	*	7	5	6	.	7	6
*	*	*	9	2	.	4	3
*	*	*	*	5	.	2	1
*	*	*	*	*	.	1	0

AK, 21-24 

*CTRL: These hubs are used primarily to control printing of check-protecting asterisks. They are always wired to the pickup of a co-selector. All the zero print control hubs for the positions in which the asterisks are to print are connected through the normal side of the co-selector, and the transferred hubs are laced from one to the other. Once a printwheel begins to turn because of the presence of a significant digit, no asterisk can print to its right. They print to the left of the high-order significant digits.

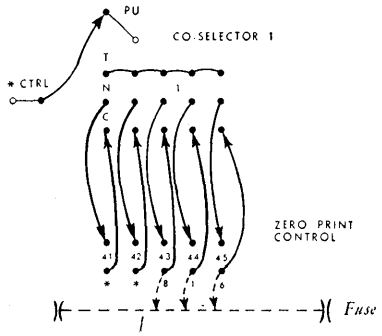
Operating principles of the asterisk control feature are described below:

1. An impulse is emitted from every lower zero print control hub at no-zone time. If the no-zone impulse can reach the fuse, a check-protecting asterisk will print.

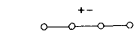
2. When a significant digit is printed, an internal connection is made from the lower zero print control position to the fuse.

3. In the schematic shown here, the no-zone impulse from zero print control position 42 can reach the fuse as follows:

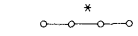
   a. through the transferred side of the selector
   b. to the transferred selector position at the right over the crosswiring
   c. through the transferred side of the selector to the lower zero print control hub for position 43, where it can reach the fuse over the internal path set up by the digit 8.



The impulse from position 41 can reach the fuse in a similar manner, by passing over two positions of the transferred side of the selector to zero print control position 43. The selector is picked up from *CTRL so that the lower zero print control hubs will be connected at no-zone time only; at all other times they are wired to the upper hubs through the normal side of the selector.

AM, 21-24 

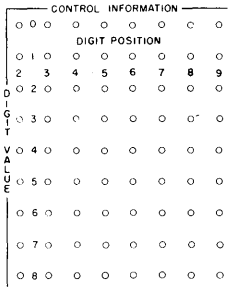
+ OR −: The four common + or − entry hubs are direct connections to complete the circuit for 11 or 12 impulses only. They are normally wired from zero print control hubs to control the printing of the plus symbol when only a 12 is sensed and a minus symbol when only the 11 is sensed by the printwheels, regardless of the presence of significant digits. They are sometimes wired to print other symbols containing an 11 or 12 impulse, when those symbols are wired to the printwheels.

AN, 21-24 

*ENTRY: These hubs are wired from the zero print control hubs to print asterisks to the left of high-order significant digits or to print asterisks for zero balances.

THE CONTROL WORD

Word 1 of the output synchronizer is designated as the control information word in the IBM 7070 Data Processing System. Included in the control information word are ten digit positions, each of which can contain any digit value (0-9). Digit positions 2-9 each have 10 digit value exits on the control panel labeled *Control Information.*

AJ, 1-8 

CONTROL INFORMATION: An early-timed impulse is emitted from one hub in each of eight vertical columns on every print cycle unless CI is off. These columns represent the digit value in each of eight digit positions (2-9) in the control word. These im-

pulses can be used to pick up co-selectors, field selectors, control the carriage, etc. If the control word is alphabetic, the control information hubs emit the 7070 alphabetic code. For example: The alphabetic word JONES causes 76 75 65 82 to emit at CI time from digit positions 2-9.

K, 13-14 

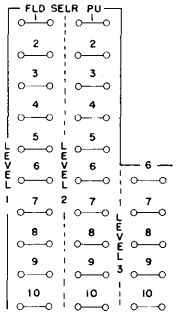
CI OFF (CONTROL INFORMATION OFF): These hubs are normally impulsed from EWI. When they are impulsed before control information time, the control information hubs become inactive.

Selection

FIELD SELECTORS

Output from the IBM 7070 to the 7400 can be arranged in several different formats without changing control panels, by use of the field selectors provided.

K-T, 1-4
P-T, 5-6

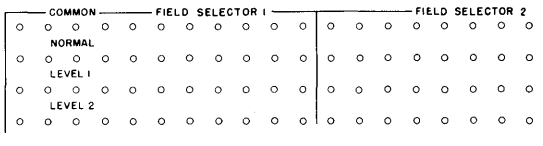


FIELD SELECTORS PICKUP LEVELS 1, 2, AND 3: These hubs, when impulsed, cause the corresponding level of a field selector to transfer on the same print cycle. They are normally wired from CONTROL INFORMATION. An impulse to these hubs causes the field selectors to transfer immediately and remain transferred for the duration of that print cycle.

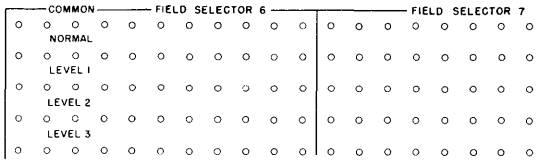
Note: The higher-numbered levels have selective priority.

Example: If both levels 1 and 2 are impulsed to pick up on the same cycle, then only level 2 is connected internally to common.

N-Q, 10-64



X-AB, 10-64



FIELD SELECTOR LEVELS 1, 2, AND 3—NORMAL AND COMMON: There are ten 11-position field selectors provided. Field selectors 1-5 have three levels each *(normal, level 1 and level 2);* selectors 6-10 have 4 levels each *(normal, level 1, level 2, and level 3).* *Note:* The vertical arrangement of the four level field selectors 6-10 is the same as that of the three level field selectors 1-5.

In Figure 228 word 1 is to print in four different fields on the report under the following conditions:

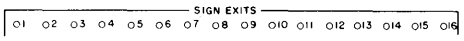
Control Information

Digit Position 9	Print Wheels
Digit Value 1	1-5
Digit Value 2	6-10
Digit Value 3	12-16
Other than 1, 2, or 3	18-22

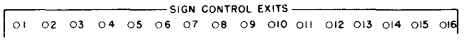
SIGN CONTROL

K, 17-18 

PR+ (PRINT +): These hubs must be connected when a 12 (+) impulse is to be emitted from the sign exit hub for positive output words. If the PR+ switch is unwired, only minus symbols (−) emit from the SIGN EXITS.

M, 9-24 

SIGN EXITS: One hub for each of the 16 output words emits an 11 (−) impulse for a negative word and a 12 (+) impulse for a positive or alphabetic word. These exits are normally wired to PRINT ENTRY and work in conjunction with the PR+ switch.

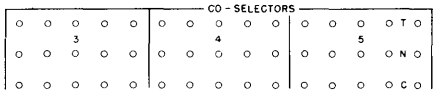
L, 9-24 

SIGN CONTROL EXITS: These hubs emit an early timed impulse from the pair of hubs associated with any output word with a *negative* sign. These impulses can be used to pick up co-selectors, field selectors, etc.

CO-SELECTORS

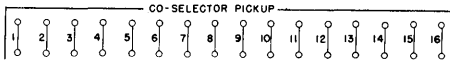
Additional capacity for printed format control is available through the use of the co-selectors on the 7400 control panel.

H-J, 25-64
AC-AE, 25-64



CO-SELECTORS: There are 16 co-selectors in the 7400. Each selector has five positions, each position having a C (common), N (normal), and a T (transferred) hub. When they are impulsed to pick up, there is a common connection between C and T; and when they are not impulsed to pick up, there is a common connection between C and N.

H-I, 9-24



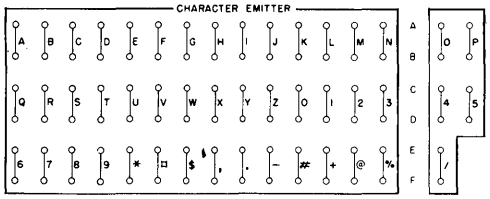
CO-SELECTOR PICKUP: Each selector has two common pickup hubs. When these hubs are impulsed, the selector transfers immediately and holds for the remainder of the cycle.

Figure 229 shows printing from the storage exits and selection using sign control.

CHARACTER EMITTER PRINTING

All 47 characters can be printed from the character emitter when they are not available from a storage exit.

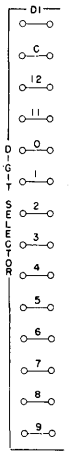
A-F, 19-32



CHARACTER EMITTER: All digits, letters, and special characters are emitted from the character emitter hubs during every print cycle. They may be wired to PRINT ENTRY directly or through properly controlled selectors.

In Figure 230 the date is being emitted for printing under the control of the 7070 program.

K-X, 7-8



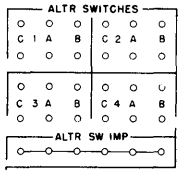
DIGIT SELECTORS: One digit selector is provided. Digit-timed impulses wired to the C (Common) hub of a digit selector are distributed internally to the appropriate exit hub (12-9).

The DI (digit impulse) hub furnishes all digit impulses automatically at their appropriate times. When DI is wired to C of a digit selector, the selector becomes a digit emitter and can be used to print numbers, letters, or special characters.

Alteration Switches

Within reasonable limitations, one control panel can be used for several different printing formats without any change in control-panel wiring, by use of the alteration switches.

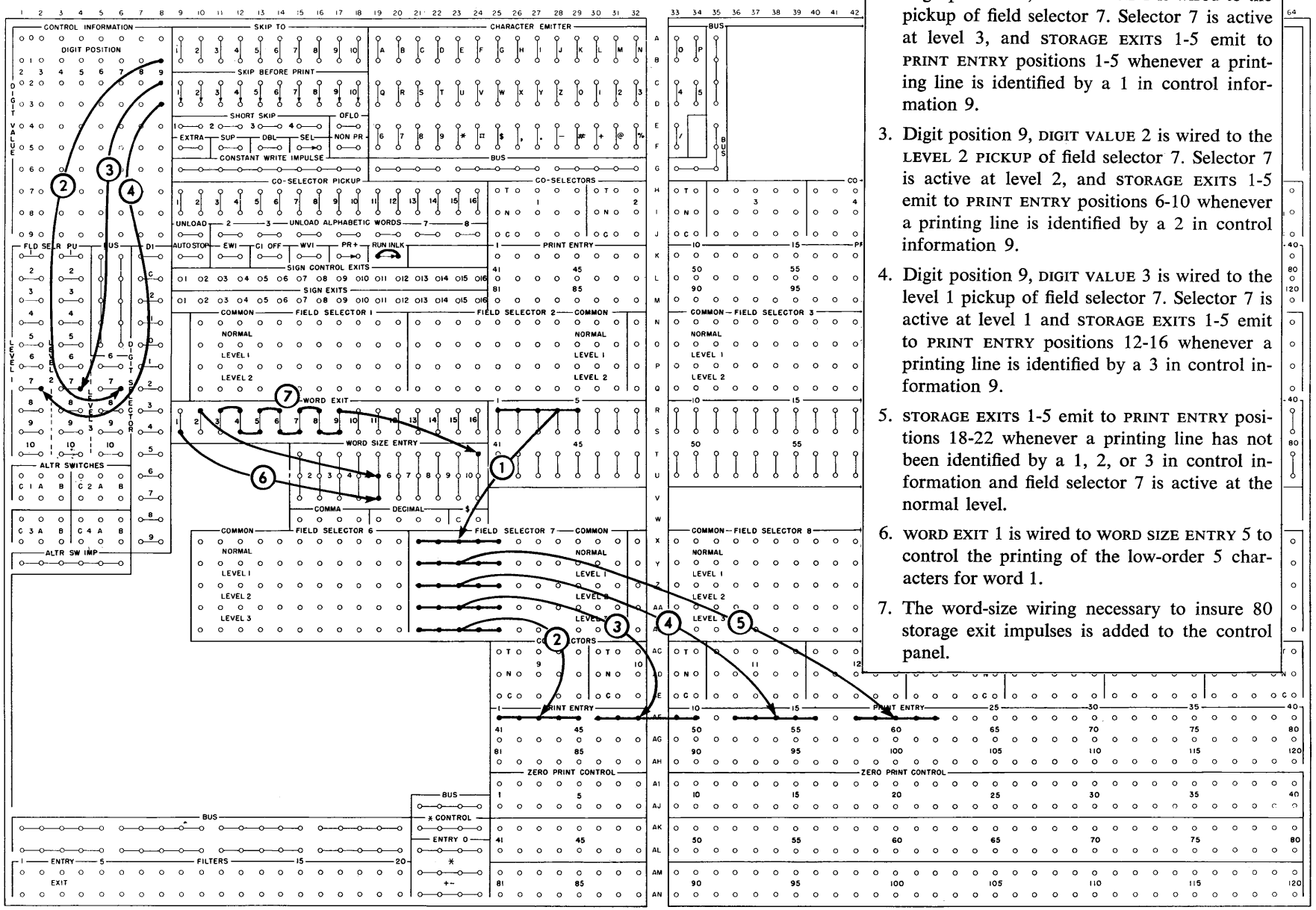
U-X, 1-6



ALTR SW (ALTERATION SWITCH) 1, 2, 3, 4: Four alteration switches are located on the front of the machine. Each alteration switch controls a corresponding two-position alteration switch selector. When the switch is set on A, a common connection exists between C and A; when set on B, the connection is between C and B. An alteration switch selector remains transferred as long as the corresponding alteration switch is set on B. Any impulse can be wired through C-A or C-B.

ALTR SW IMP (ALTERATION SWITCH IMPULSE): These hubs emit an impulse that is normally wired through an alteration switch selector to the pickup of co-selectors. A selector impulsed through the A side of an alteration switch selector remains transferred as long as the corresponding alteration switch is set to A. If the selector is impulsed through the B side of an alteration switch selector, the co-selector will transfer on each print cycle as long as the corresponding alteration switch is set to B.

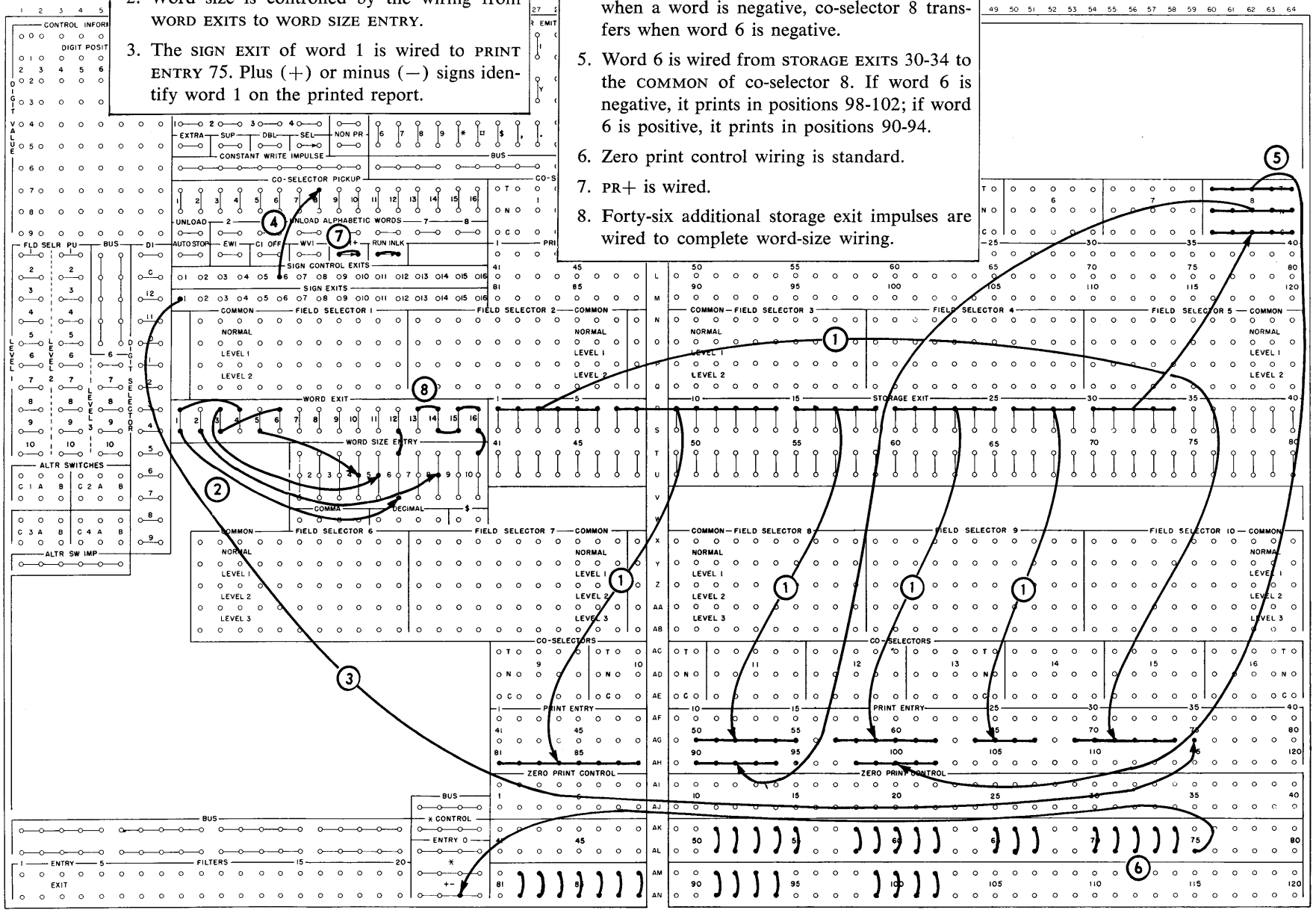
In Figure 231, the information from storage exits 1-5 is printed under control of alteration switch 3.



1. STORAGE EXITS 1-5 are wired to the COMMON of field selector 7.
2. Digit position 9, DIGIT VALUE 1 is wired to the pickup of field selector 7. Selector 7 is active at level 3, and STORAGE EXITS 1-5 emit to PRINT ENTRY positions 1-5 whenever a printing line is identified by a 1 in control information 9.
3. Digit position 9, DIGIT VALUE 2 is wired to the LEVEL 2 PICKUP of field selector 7. Selector 7 is active at level 2, and STORAGE EXITS 1-5 emit to PRINT ENTRY positions 6-10 whenever a printing line is identified by a 2 in control information 9.
4. Digit position 9, DIGIT VALUE 3 is wired to the level 1 pickup of field selector 7. Selector 7 is active at level 1 and STORAGE EXITS 1-5 emit to PRINT ENTRY positions 12-16 whenever a printing line is identified by a 3 in control information 9.
5. STORAGE EXITS 1-5 emit to PRINT ENTRY positions 18-22 whenever a printing line has not been identified by a 1, 2, or 3 in control information and field selector 7 is active at the normal level.
6. WORD EXIT 1 is wired to WORD SIZE ENTRY 5 to control the printing of the low-order 5 characters for word 1.
7. The word-size wiring necessary to insure 80 storage exit impulses is added to the control panel.

FIGURE 228. FIELD SELECTION

FIGURE 229. PRINTER SIGN CONTROL



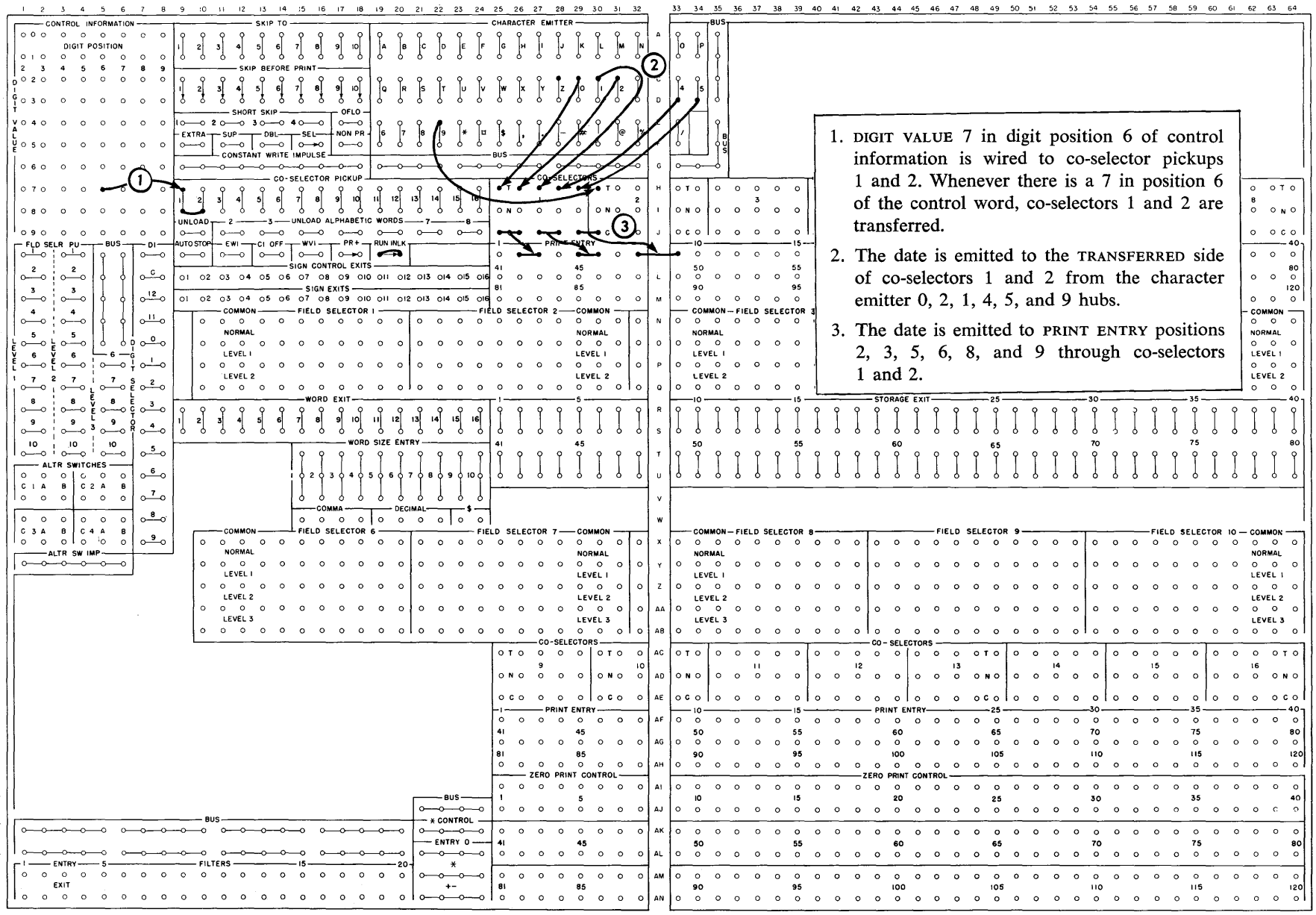
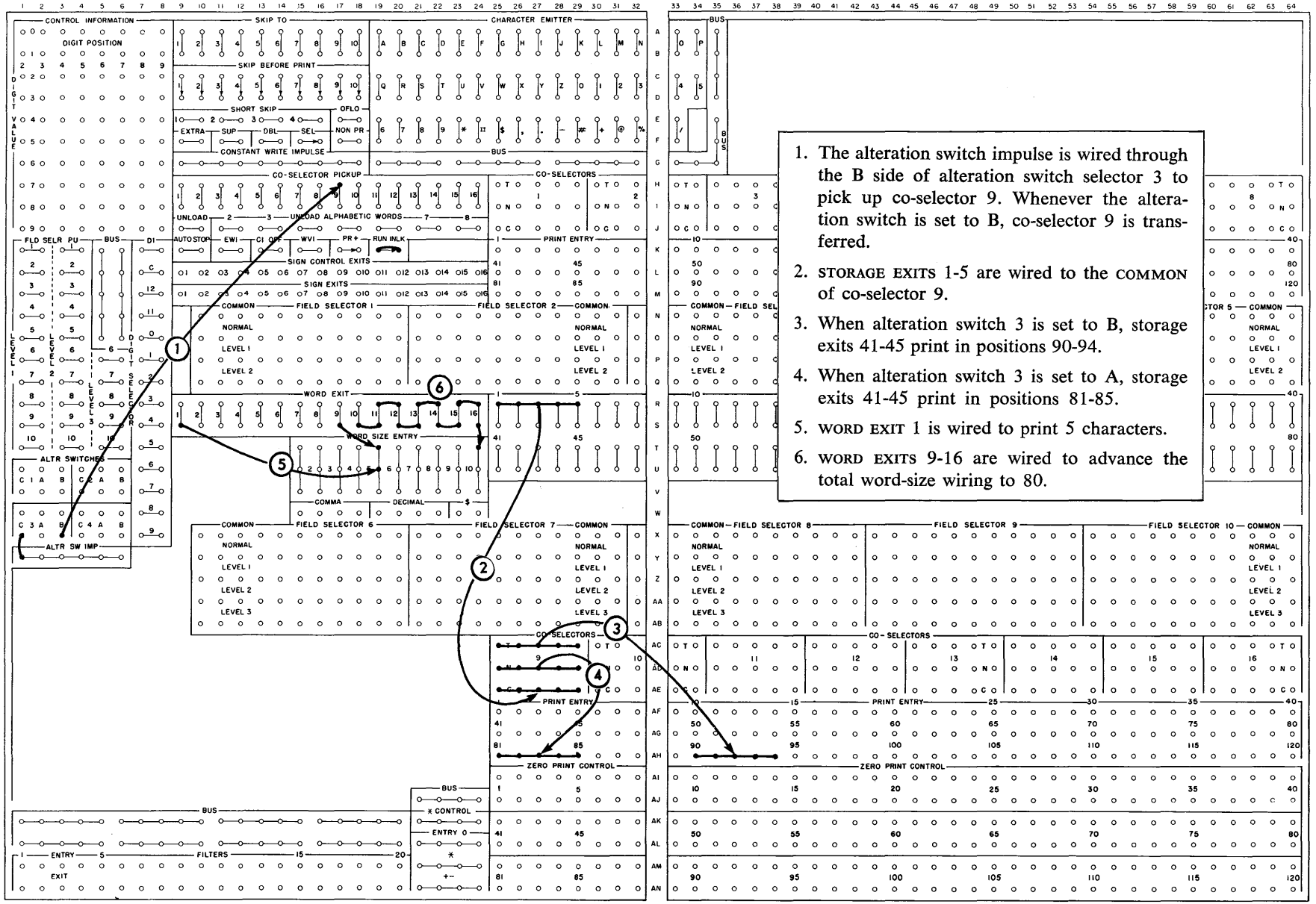


FIGURE 230. EMITTING DATA FOR PRINTING

FIGURE 231. ALTERATION SWITCH WIRING



Tape-Controlled Carriage

The Tape-Controlled Carriage controls feeding and spacing of forms at high speed while documents or reports are being prepared on the IBM 7400 Printer. The carriage is controlled by punched holes in a narrow paper tape that exactly corresponds in length to the length of one or more forms. Holes punched in the tape stop the form when it reaches any predetermined position. One of the punched holes in the tape can be used to control the carriage to start overflow skipping to the next form.

The carriage accommodates continuous forms measured in 6ths of an inch up to a maximum of 22 inches in length and 16¾ inches in width including punched margins (Figure 232). While forms of any size within these limits can be handled by the carriage, forms of standard sizes available from the forms manufacturers can be obtained more quickly and economically.

Forms can be designed to permit printing in practically any desired arrangement. Skipping can be controlled to ten different sections of the form.

Variable-Line Spacing and Uniform Skipping

Single, double, or triple spacing can vary between lines as controlled by wiring on the control panel. For example, the heading section of a form can be single spaced and the body double spaced.

Any other spacing that is required must be controlled by the tape. Spaces up to two and a half inches between

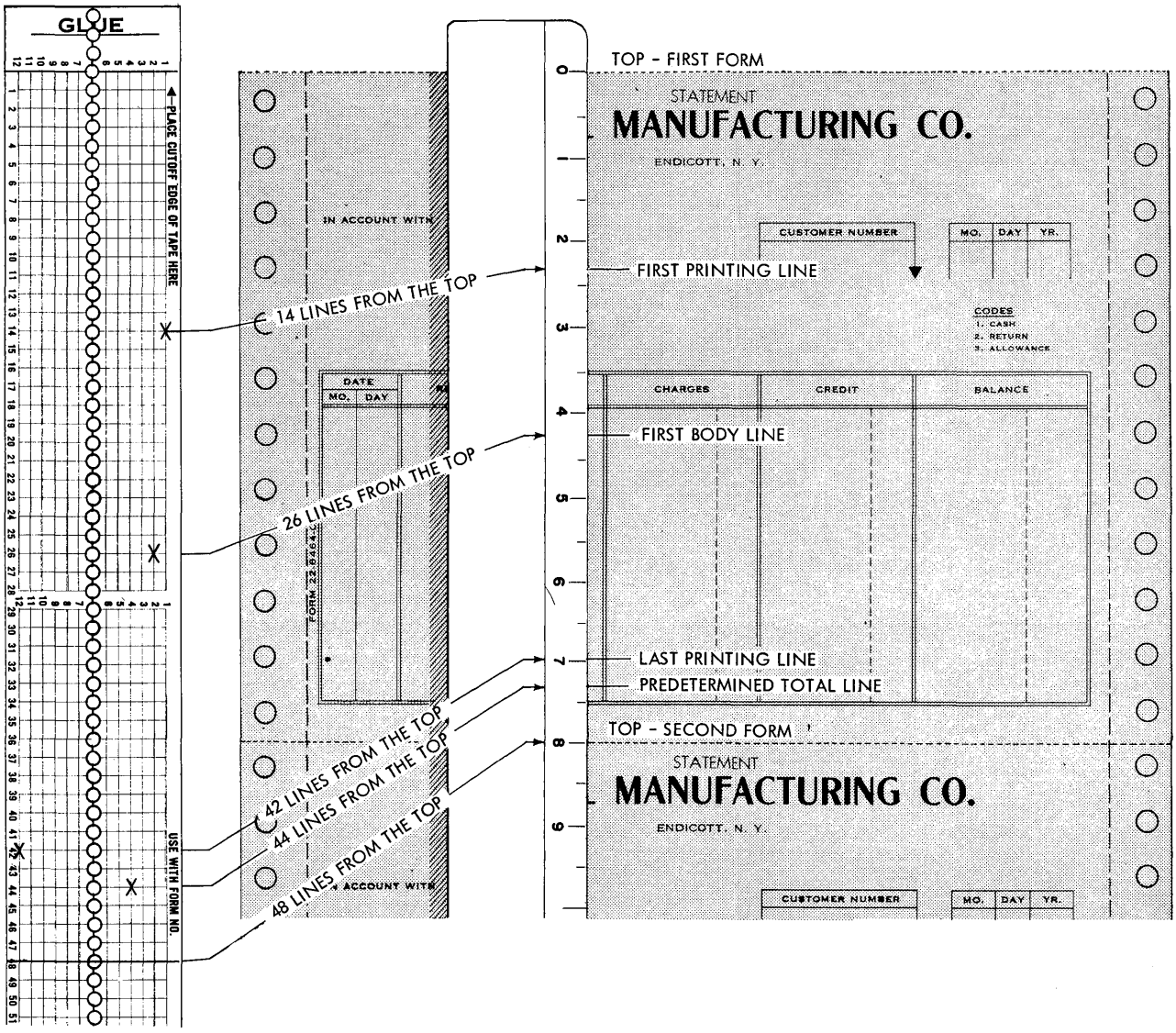


FIGURE 232. FORMS

lines can be skipped at the same rate of speed as normal spacing (Figure 232). This skipping is a smooth, high-speed advance of the form allowing successive lines to be printed up to two and a half inches apart at the rate of 150 lines a minute, the normal printing speed of the machine.

Overflow Skipping

When one form is completely filled, it can be ejected, and the next form can advance to the first printing line or to the first body line. This overflow skipping is caused by sensing a punch in a specific position of the tape, which starts advancing the paper to the required line on the next form. Overflow is slower than other skipping; therefore, it is desirable to reduce overflow skipping to a minimum.

Overflow Sheet Identification

Several lines of numerical or alphabetic identifying information can be printed on an overflow sheet. Invoice and page numbering can also be printed on the overflow sheets.

Predetermined Total Line

Totals can be printed on a predetermined total line whether the form is completely filled or not. For example, although only two or three items have been printed on a form, the total of these items can be printed on a designated line of the form, instead of directly beneath the last item printed.

Control Tape

The control tape (Figure 233) has 12 columnar positions indicated by vertical lines. These positions are called channels. A maximum of 132 lines can be used for control of a form, although for convenience the tape blanks are slightly longer. Horizontal lines are spaced six to the inch for the entire length of the tape. Round holes in the center of tape are pre-punched for a pin-feed drive in a tape sensing mechanism that controls the carriage. The tape advances through the mechanism in synchronism with the movement of the printed form through the carriage. The effect is exactly the same as though the control holes were punched along the edge of each form.

Twelve brushes, one for each channel, are positioned over the tape to sense the holes that are punched. As viewed from the front of the machine, they are numbered 1 through 12 from left to right. Brush 1 rests on channel 1, brush 2 on channel 2, and so on. A hole in the channel allows the brush to make contact with a metal roll and set up the necessary circuits that are normally used to stop skipping or to initiate an overflow.

Tape Channels

Tape channels are punched to control the following functions:

FIRST PRINTING LINE STOP: Channel 1 is normally punched for the first printing line of a form. This is the starting or home position.

NORMAL SKIP STOPS: Channels 2 through 10 are used to stop a form at one of nine positions including first body line. They can be used in any order or sequence. Early-timed impulses such as CONTROL INFORMATION, CWI, and EWI can be used to start skipping to any position of the form.

OVERFLOW CONTROL: The 12th channel of the tape must be punched in a position corresponding to the last printing line of a form. This punch is normally used to cause immediate overflow skipping. A hole in channel 12 causes an impulse to be emitted from the OFLO (overflow) hubs. This impulse can be directed to the skip to channel 1 hub only.

Short Skip

Normally, the tape-controlled carriage interlocks the printer during every skip regardless of its length. The 7070 can accept read commands after the skip is completed, but at least one machine cycle is lost for every skip taken. This is sometimes referred to as interlocking, and its primary purpose is to prevent printing in flight for skips longer than two and a half inches. If a skip is two and a half inches or less, and skipping is to occur before printing, it is called a short skip, and the control panel may be wired to release the interlock, thereby permitting continuous operation of the printer.

In designing forms, distances that are to be skipped frequently should, if possible, be kept within two and a half inches for most efficient operation. These distances may or may not be between two successive sections of a form. For example, in a billing form with a section for the heading and section for the body, the distance from the last heading line printed to the first body line should be kept within two and a half inches for most efficient operation.

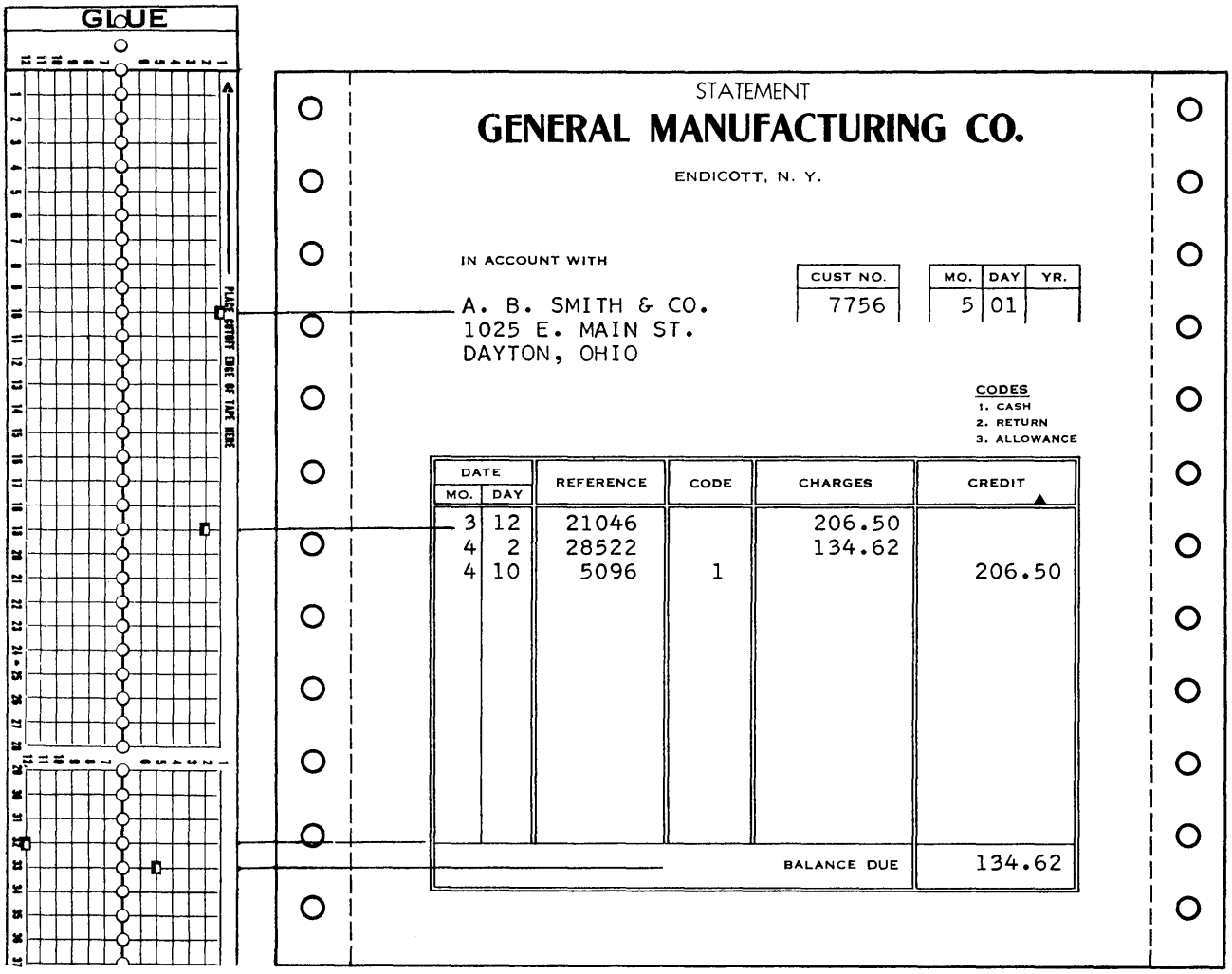


FIGURE 233. CONTROL TAPE

TAPE PUNCHING

A small, compact punch (Figure 234) is provided for punching the tape. The tape is first marked in the channels in which the holes are to be punched. This can be done easily by laying the tape beside the left edge of the form that it is to control with the top line (immediately under the glue portion) even with the top edge of the form. A mark is then made in the first channel on the line that corresponds to the first printing line of the form. Additional marks are made in the appropriate channels for each of the other skip stops and the overflow signal required for the form.

The marking for one form should be repeated as many times as the usable length of the tape (22 inches) allows. With the tape thus serving to control several forms in one revolution through the sensing mechanism, the life of the tape is increased. Finally, the line corresponding to the bottom edge of the last form should be marked for cutting after the tape is punched.

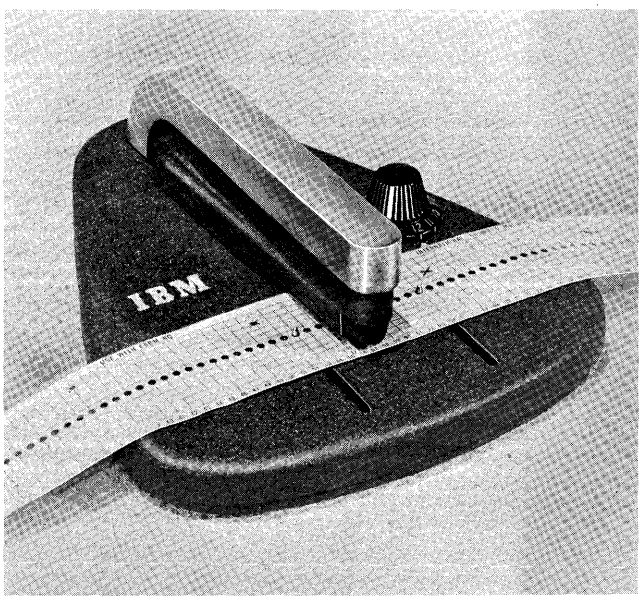


FIGURE 234. TAPE PUNCH

The tape is inserted in the punch by placing the line to be punched over a guide line on the base of the punch and placing the center feed holes of the tape over the pins projecting from the base. The dial is then turned until the arrow points at the number of the channel to be punched. Pressing on the top of the punch, toward the back, cuts a rectangular hole at the intersection of a vertical and horizontal line in the required channel of the tape.

The tape can be punched with holes in more than one channel on the same line. This is advantageous in many cases, when several skip impulses are directed to the same skip stop. Punching two holes in one channel is necessary in some instances.

After the tape is punched, it is cut and looped into a belt. The bottom line is glued to the top line by the use of the section marked *glue,* after the glaze has been removed by an ink eraser. If the glaze is not removed, the tape ends may come apart. The center feed holes should coincide when the two ends of the tape are glued together.

The last hole punched in the tape should not be less than four lines from the cut edge, as approximately the last half inch of the tape overlaps the glue section when the two ends are spliced. If it is necessary to punch a hole lower than four lines from the bottom of the form, place the tape with the top line (immediately under the glue portion) four lines lower than the top edge of the form before the channels are marked. To compensate for the loss, cut the tape four lines lower than the bottom edge of the form.

Inserting Tape in Carriage

Tilt back the cover of the carriage to gain access to the tape reading mechanism. Turn the platen clutch to a disengaged position, and raise the brushes by moving to the left the latch located on the side of the brush holder. With the tape held so that the printed captions can be read, place one end of the loop over the nearest half circle guide piece. Remove the excess slack from the tape by lifting the lever away from the notched bar and by moving the guide piece unit to the right. Have the tape just tight enough so that it gives slightly when the top and bottom portions of the loop are pressed together. If the tape is too tight, the pin feed holes will be damaged.

After the tape is in position, press down the brushes and close the cover. Press the restore key to bring the tape to its home position and turn back the platen clutch to its engaged position. The carriage is then ready to operate.

Tapes can be changed readily and used repeatedly over a considerable period of time.

Operating Features

Platen Clutch

When the platen clutch is positioned so that arrow below it is pointing to IN (Figure 235), the platen is engaged and can be turned manually only by the vernier knob. To disengage the platen from machine control, turn the platen clutch to the right. The platen can then be turned manually by the platen knob.

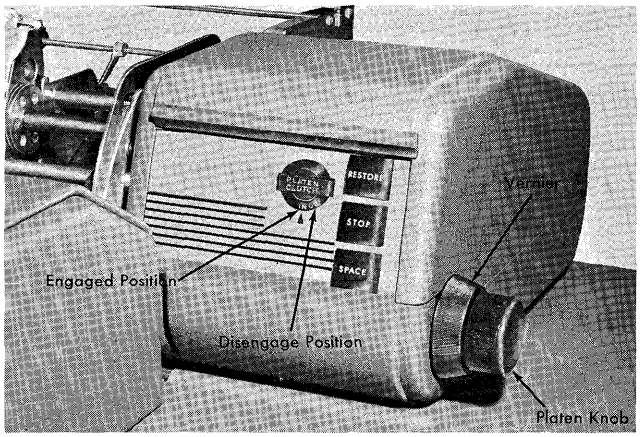


FIGURE 235. PLATEN CLUTCH

Restore Key

Press the restore key to set the carriage at the start or home position (channel 1). This is done while the platen is disengaged. Restoring is necessary because the distance that each form travels through the carriage, as it is being printed, is measured by the tape. Starting from the first printing line of one form, the tape moves in synchronism with the form.

Stop Key

Pressing this key stops the carriage operation immediately. Use it to stop undesired carriage skipping. It also stops the printer, but it should not be used for that purpose alone.

Space Key

When the printer is stopped, advance a form by pressing the space key. The form advances one space each time the space key is pressed, regardless of the spacing for which the space control is wired. During processing, spacing is controlled by control-panel wiring. Feed the first form into position by pressing the space key if the platen clutch is engaged, but disengage the platen clutch to permit restoring the tape without advancing the form.

PLATEN KNOB

Turn this knob backward or forward to position the form only when the platen clutch is disengaged.

VERNIER KNOB

Use the vernier knob to obtain exact registrations in relation to the horizontal lines. The platen advances, thus lowering the printing on the form, when the knob is turned counterclockwise. Turning the knob in a clockwise direction causes the printing to occur higher on the form. In either case, the carriage tape is not affected and adjustments can be made while the platen is engaged or while the machine is in operation.

FORM THICKNESS ADJUSTMENT DEVICE

The distance between the printwheels and the platen is adjustable, for thickness of paper stock or for varying number of copies, by the use of the form thickness adjustment device located on the left side of the carriage. This device contains seven positions numbered from 0 through 6. When the dial is in the 0 position, the printwheels are ⅛ (.125) of an inch from the platen. Each of the remaining six positions adds to the ⅛ inch distance by about the thickness of 1½ cards. When the dial is set to 6, the distance is increased to approximately .178 of an inch. Set the dial wherever the best results are obtained. To adjust for varying thicknesses, pull out the dial and turn it to increase or decrease the distance between the platen and the printwheels.

PRESSURE RELEASE LEVER

When the lever is pushed back, the feed rolls are released so that the paper can be moved freely around the platen. Always release pressure when the forms tractor for the 7400 is in use. Apply pressure when the forms tractor is not in use.

PLATEN SHIFT WHEEL

The platen can be shifted laterally a total of four inches by turning the platen shift wheel. For example, with the carriage in the extreme left position, the carriage can be moved four inches to the right. Do not make this adjustment while the machine is in operation.

END-OF-FORM STOP

The end-of-form stops located in the center and near either end of the carriage, stop the machine when the carriage runs out of paper, if the form stop switch is turned on. The forms feed under the end of form stops; and when the bottom edge of the last form passes them, the machine stops. The distance between the end of form stops and the printing line is about 13⅜ inches. If the end-of-form stop is not desired, the form stop switch is turned off.

Forms Tractors—The IBM F-2 Forms Tractor— 6- or 8-Lines Per Inch Spacing

This device is used for feeding marginally punched continuous forms and has two adjustable tractor-type pin-feed units, one for each side of the forms. It can be freely inserted in the carriage by first positioning the carriage paper guides to the far left and right sides of the carriage, and by hooking the rear pin of the forms tractor in position and then lowering the front.

The F-2 provides the choice of 6- or 8-lines-to-the-inch spacing. Make this adjustment by moving the shift cam until its pointer is positioned between the two scribed lines at either the 6 or the 8 on the side frame. If the pointer cannot be positioned between the scribed lines, a tooth-on-tooth condition exists between the platen gear and the forms tractor drive gear. In this case, move the shift cam to release the pressure on the 6- or 8-line drive gear and turn the platen slightly to allow the teeth to engage fully.

6- OR 8-LINES-PER-INCH SPACING: Regardless of the setting of the forms tractor at 6 or 8 lines per inch, the platen moves at 6 lines per inch. When the F-2 is spacing at 8 lines per inch, the platen has a tendency to move the back copies of a multi-copy form faster than the original and front copies, even though the pressure release lever is disengaged. For this reason, single spacing at 8 lines per inch is not recommended where the accuracy of line spacing is critical.

The control tape for 8-lines-per-inch spacing is punched, as it would be for normal 6-lines-per-inch spacing. Each line on the tape always equals one line on the form regardless of whether the latter be 6- or 8-lines-per-inch. In measuring a control tape for a document printed 8 lines to the inch, every ⅛ inch on the form represents one line on the tape.

Normal spacing is 6 lines per inch.

STEPS IN USING THE FORMS TRACTOR

1. After the forms tractor is in position, disengage the platen clutch. Make sure that the platen and the forms tractor can be moved freely by hand.
2. If the form to be used is less than 7 inches wide, remove the center paper guide and paper-tension device.
3. Move the left lower paper guide and tractor slightly to the left of the first printing position, place the

first form between the left and right lower paper guides, and move the right guide in against the edge of the form. Allow a slight clearance so that the form slides freely between both guides. Tighten both locking rings to hold the guide assemblies in place.

4. With the pressure rolls engaged, insert the form over the forms tractor paper guides, under the round rod, and then into the pressure rolls and platen. Turn the platen by hand until the end of the form can be grasped.
5. Raise the pressure plates away from the tractor pins.
6. Release the pressure rolls and position the pin-feed hole in the tractor pins.
7. Lower the pin-feed pressure plates.
8. Set the form so that the first printing line is even with the first printing line indicator mark on the lower part of the pressure plates; then turn the form back fourteen spaces if spacing is set for 6 lines per inch, and nineteen spaces if set for 8 lines per inch. Finer adjustments may be achieved by use of the vernier knob.
9. Insert control tape, restore carriage, and engage the platen clutch.

Tractor Adjustments

Tractor adjustment wheels can be turned to provide a ⅛ inch lateral movement of the tractors. These wheels make the tractor pins line up exactly with the center of the marginal holes in the paper after the paper guides have been set.

Outfold Guide Bar

The outfold guide bar fits across the front of the forms tractor. It is an aid in the feeding of forms. It is spring-loaded and can be easily removed by the operator, if desired.

Paper Tension Device

The paper tension device is adjustable and exerts a slight pressure on the paper as it feeds through the forms tractor.

Platen

The carriage is equipped with an easily removable solid platen as a standard feature.

Platen hardness requirements vary with the number of parts and type of paper in each form. The following platen hardnesses are recommended:

Hard platen (100-durometer), two-part forms and over.

Medium platen (90-durometer), most single-part forms or very thin multiple-part forms.

If there is doubt as to the platen hardness required or to the quality of the carbon paper to be used, make test runs with sample sets of forms.

Remove the platen by raising the platen lock on the left side and lifting the platen from the bearing housing. When the platen is inserted, drop the end with the gear wheel into the platen bearing housing. The platen must then be moved to the right, turning it back and forth to fit the platen drive key into the carriage drive mechanism. The platen lock is then closed.

Tear Bar

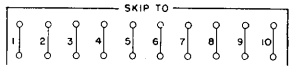
The tear bar can be used in place of the forms tractor whenever feeding is under the control of the pressure rolls.

It is generally used for single-sheet operations or for roll paper. It is easily inserted in the carriage by placing the ends into the forward slots on the carriage frame.

Skip Hubs

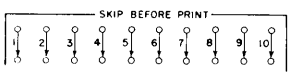
Skipping is started by control-panel wiring and is stopped by holes in the tape. Skipping is required whenever continuous forms are used.

A-B, 9-18

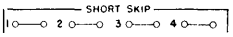


SKIP TO: There are 10 carriage skip positions representing the first 10 channels on the carriage tape. Each of the 10 positions has a SKIP TO hub. If an early-timed impulse is introduced into one of these hubs and a hole is punched in the corresponding channel of the tape, a skip will take place to the position where the hole is punched in the tape. For example, if hub 3 is impulsed on the control panel and a hole is punched in channel 3 of the tape, the impulse starts the skip and the hole in the tape stops it. Channels 1-10 can be used in any order to identify printing lines on a form. However, channel 1 is normally used to identify the first printing line. Channel 12 is *always* used to initiate overflow.

C-D, 1-8



SKIP BEFORE PRINT: These 10 switches cause skipping to a predetermined channel in the control tape before the print cycle occurs. The corresponding SKIP TO hub must be activated before these switches become effective.



E, 9-16

SHORT SKIP: These hubs accept early-timed impulses to release the interlock that occurs whenever a skip takes place. They can be wired if a skip does not exceed 2½ inches. If a carriage skip exceeds 2½ inches and short skip is wired, improper printing registration can result. Wiring to short skip is effective only on a skip before print operation.

E, 17-18

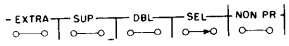
OFLO (OVERFLOW): The overflow hubs emit whenever a punch is sensed in channel 12 of the control tape on a print cycle. This impulse must be wired to skip to channel 1. Skipping starts after the line which corresponds to the channel 12 punch is printed. During overflow skipping an interlock occurs regardless of the wiring to skip before print and short skip.

In Figure 236 the following tape channels and control information assignments have been made.

	Digit Value	Tape Channel	Description
Digit Position 9	1	1	First Printing Line
	2	2	Miscellaneous Data Line
	4	3	First Body Line
	5	4	Predetermined Total Line

Space Hubs

Single spacing (six lines to the inch) on the 7400 is automatic. Double spacing, space suppression, and extra spacing can be accomplished through the use of the space control hubs on the control panel.



F, 9-10 F, 11-12 F, 13-14 F, 15-16 F, 17-18

EXTRA: An impulse wired to these hubs causes the machine to take an extra space after printing. If the machine is single spacing and extra space is impulsed, two spaces occur, one before and one after printing. If the machine has been impulsed to double space and EXTRA has been impulsed, then triple spacing occurs—two before and one after printing. These hubs are normally wired from CONTROL INFORMATION or CWI.

SUP (SUPPRESS): When these hubs are impulsed, all spacing is suppressed before printing for that cycle. These hubs are normally wired from CONTROL INFORMATION or CWI.

DBL (DOUBLE): An early-timed impulse is wired to these hubs to cause double spacing (three lines to the inch). These hubs are normally wired from CONTROL INFORMATION or CWI.

SEL (SELECTIVE SPACE): All line spacing may be controlled by the SEL hubs and a hole in channel 11 of the tape. More particularly, however, these hubs and channel 11 are used to control triple spacing or selective line spacing up to a maximum of seven lines.

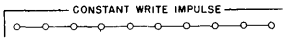
When the selective space hubs are connected, the carriage is stopped by reading a hole in channel 11. The automatic single spacing or wiring of the double space hubs normally perform this function. The printing of every line is then normally under the control of the 11 punch in the tape. Whenever the desired spacing (including overflow skipping) exceeds three lines but no more than seven, both SPACE SUPPRESS and EXTRA must be impulsed from CWI. This causes the selective spacing to occur after printing.

Triple spacing on the 7400 can be accomplished by connecting the the SEL hubs and by punching a hole in channel 11 on every third line of the tape, including the line on which the total prints. If the SEL hubs are wired and DBL is impulsed, single spacing occurs. By this means, CONTROL INFORMATION can be used to override selective spacing and cause single spacing without using a selector.

Figure 237 shows the use of the space control hubs.

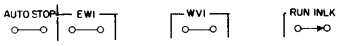
NON PR (NON PRINT): These hubs are impulsed when a particular line is not to print on a report. Impulsing these hubs suppresses all printing for that print cycle but has no effect on carriage operations.

Other Control-Panel Hubs



G, 9-18

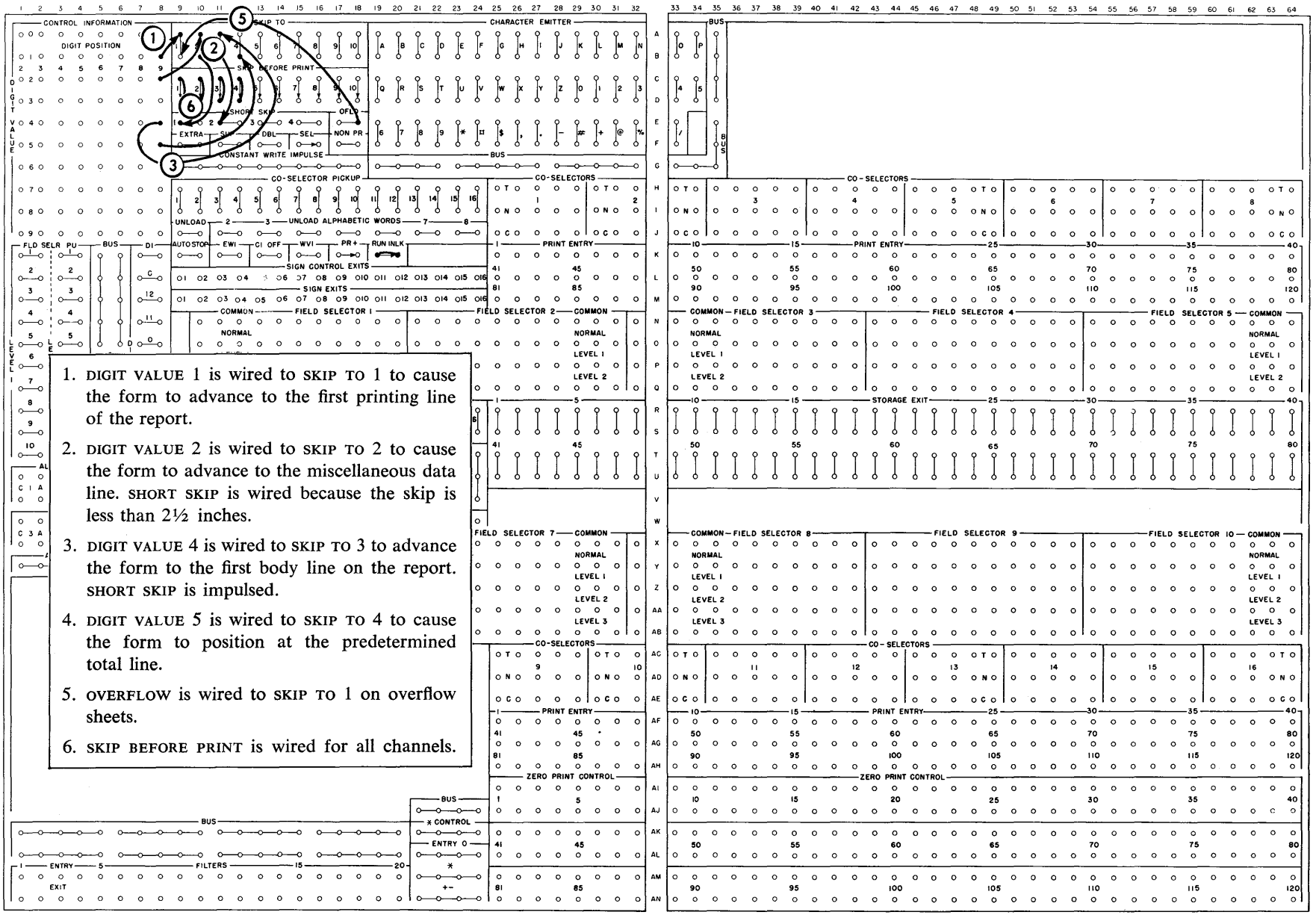
CONSTANT WRITE IMPULSE: These hubs emit early-timed impulses on every print cycle.



K, 9-12 K, 15-16 K, 19-20

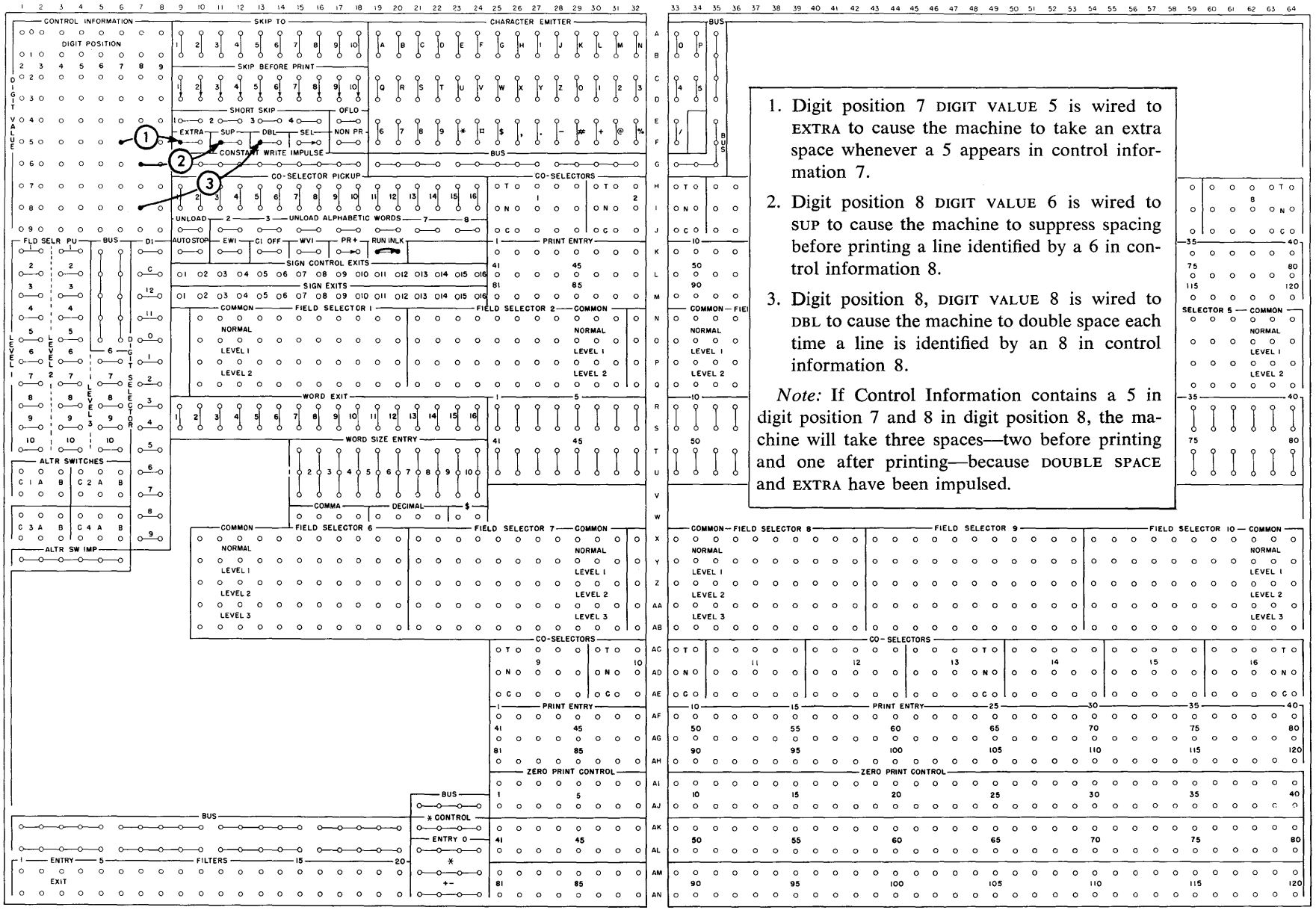
WVI (WRITE VALIDITY IMPULSE): An impulse emits from these hubs if the print validity check circuits detect a non-valid character. If this pair of hubs is unwired, the detection of a non-valid character causes the validity check light on the printer to flash. These hubs can be wired to AUTO STOP.

RUN INLK (RUN INTERLOCK): This switch must be wired to complete the print unit run circuit. Its purpose is to prevent printer operation unless a wired control panel is inserted in the machine.



1. DIGIT VALUE 1 is wired to SKIP TO 1 to cause the form to advance to the first printing line of the report.
2. DIGIT VALUE 2 is wired to SKIP TO 2 to cause the form to advance to the miscellaneous data line. SHORT SKIP is wired because the skip is less than 2½ inches.
3. DIGIT VALUE 4 is wired to SKIP TO 3 to advance the form to the first body line on the report. SHORT SKIP is impulsed.
4. DIGIT VALUE 5 is wired to SKIP TO 4 to cause the form to position at the predetermined total line.
5. OVERFLOW is wired to SKIP TO 1 on overflow sheets.
6. SKIP BEFORE PRINT is wired for all channels.

FIGURE 236. SKIP CONTROL



1. Digit position 7 DIGIT VALUE 5 is wired to EXTRA to cause the machine to take an extra space whenever a 5 appears in control information 7.
2. Digit position 8 DIGIT VALUE 6 is wired to SUP to cause the machine to suppress spacing before printing a line identified by a 6 in control information 8.
3. Digit position 8, DIGIT VALUE 8 is wired to DBL to cause the machine to double space each time a line is identified by an 8 in control information 8.

Note: If Control Information contains a 5 in digit position 7 and 8 in digit position 8, the machine will take three spaces—two before printing and one after printing—because DOUBLE SPACE and EXTRA have been impulsed.

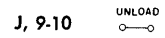
FIGURE 237. SPACE CONTROL

EWI (ERROR WRITE IMPULSE): This exit hub is directly under control of the 7070 program UWIV command. The EWI hubs are normally used to pick up selectors, impulse AUTO STOP, etc.

AUTO STOP: These hubs accept any impulse to suspend printer operation at the end of the print cycle.

Unloading

Another method of output from storage is available. This method bypasses the control panel and sets up a direct internal connection between the output-synchronizer and the printwheels. This internal connection is under control of the UNLOAD hubs.



UNLOAD: These hubs are normally wired from control information or CWI to cause words 1-8 to be transferred automatically to the printwheels as follows:

	Printwheels	
Word	Data	Sign
1	1-10	11
2	13-22	23
3	25-34	35
4	37-46	47
5	49-58	59
6	61-70	71
7	73-82	83
8	85-94	95

When UNLOAD is impulsed, all zero print control is automatic. Zeros, plus, and minus signs are printed for numerical words. The PR+ switch is automatically on. If an alphabetic word is to be printed, the unload alphabetic word hubs should be wired. Then the alphabetic word is printed in the five low-order print positions assigned to that word, and the sign position is left blank. If word 1 is alphabetic, print positions 1-11 will print the alpha characters contained therein like this: *A*B*C*D*E+. This same format results if the unload hub for a given word has not been impulsed and the word is alphabetic.

UNLOAD ALPHABETIC WORDS 2-8: These hubs are impulsed from CONTROL INFORMATION whenever information contained in words 2-8 is designated as alphabetic.

IBM 7400 Control-Panel Summary

1. *Control Information.* The impulses emitted from these hubs (Figure 238) represent the digit value in each of eight digit positions (2-9) in the control word (Word 1 of the synchronizer). These impulses can be used to pick up co-selectors or field-selectors, initiate carriage skips, etc.
2. *Field Selector Pickup—Fld Selr Pu.* These hubs are impulsed to cause the corresponding level of the field selectors to transfer immediately. If a field selector is not impulsed to transfer at level 1, 2, or 3, the normal level is active.
3. *Alter SW.* There are four alteration switch selectors on the control panel that correspond to the four alteration toggle switches on the machine. When the alteration toggle switches are turned on, the corresponding selectors transfer. The selectors may be used independently or in conjunction with co-selectors to change machine functions under the control of a toggle switch.
4. *Altr Sw Imp—Alteration Switch Impulse.* These hubs provide an impulse that is normally wired through an alteration switch selector to the pickup of a co-selector to expand the alteration switch selector.
5. *Bus.* These hubs are located in convenient sections of the control panel and are used to expand either exits or entries, thereby eliminating the need for split wires.
6. *Filters.* There are 20 filters standard, each with an entry and exit hub. They allow an impulse to travel in only one direction (into entry, out of exit) and are used to control a circuit in one direction only.
7. *Digit Selector.* The DI hub emits a series of impulses that are timed for printing the digits 12-9. If the DI hub is wired to the C hub beneath it, the digit selector becomes a digit emitter. If a storage exit position is wired to the C hub, whatever digit appears at that position is available at the corresponding hub of the digit selector.
8. *Skip to.* These hubs correspond to channels 1-10 on the tape. They receive impulses to initiate skips that are to be stopped by holes in the corresponding channels in the tape.
9. *Skip before print.* These ten switches cause skipping to a predetermined channel in the tape before the print cycle occurs. The corresponding SKIP TO hub must be activated before these switches become effective.

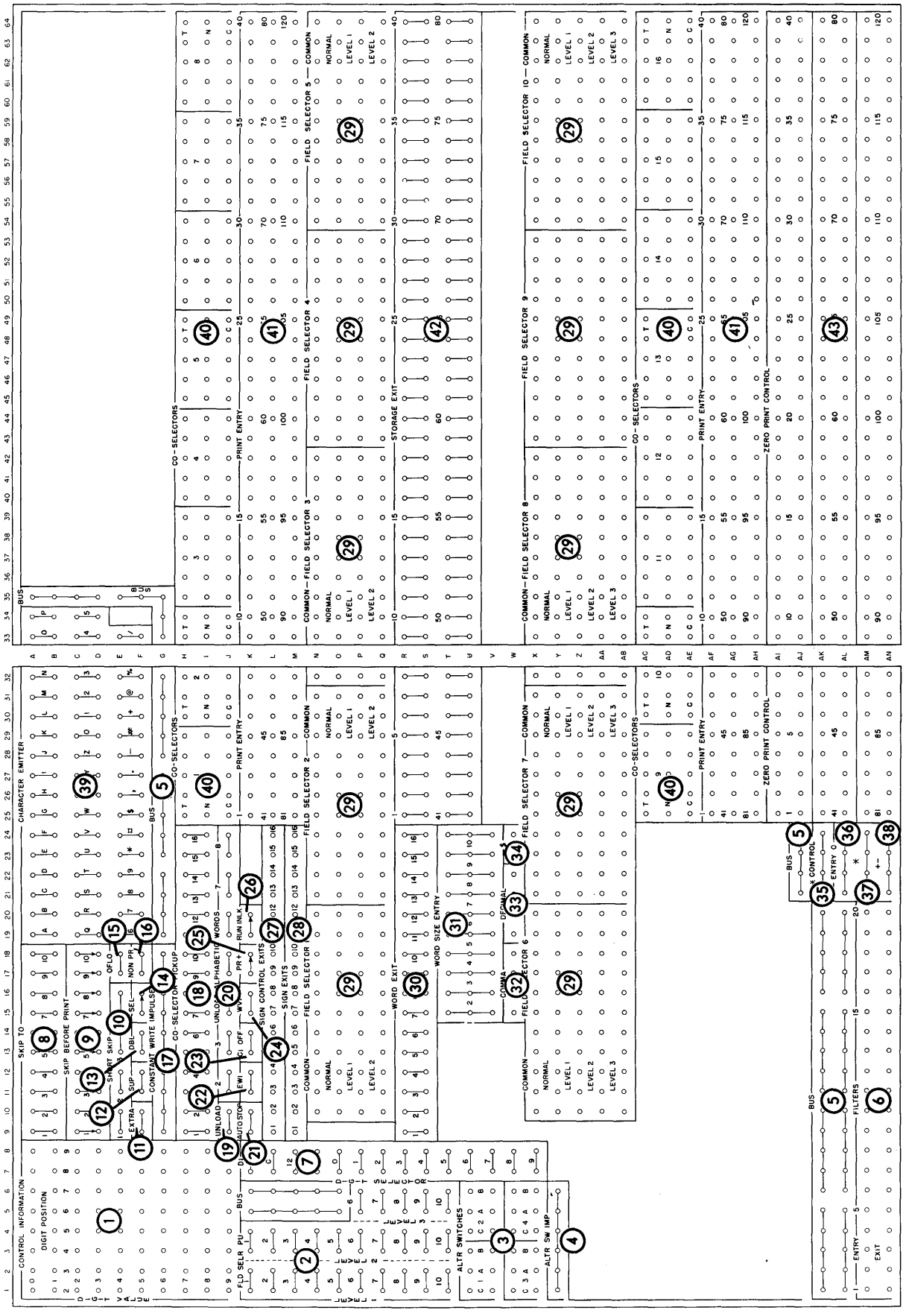


FIGURE 238. KEY TO CONTROL-PANEL SUMMARY

10. *Short Skip.* These are entry hubs normally wired from control information when the skip does not exceed two and a half inches. When these hubs are impulsed, the internal interlock is released and the skip takes place without lost time if skip before print is wired.

11. *Extra.* These hubs accept CI-timed impulses to cause single or double space after printing. If SEL and SUP are also wired, the selective spacing occurs after printing.

12. *Sup.* These hubs accept impulses to suppress spacing before a line prints. Space suppression takes precedence over all normal spacing but does not suppress extra spacing.

13. *Dbl—Double Space.* An early-timed impulse is wired to these hubs to cause double spacing (three lines to the inch). These hubs are normally wired from control information or CWI.

14. *Sel.* When these hubs are connected, spacing is under the control of channel 11 in the carriage control tape. Spacing is stopped by a hole in channel 11, thus allowing variable spacing within any section of the form. The select space hubs and the punching in channel 11 of the control tape must be used for triple spacing before printing. Up to seven lines can be spaced if SUP and EXTRA are also wired to cause selective spacing to take place after printing.

15. *OFLO—Overflow.* These hubs emit an impulse as the last body line of a form is printed. The last body line is identified by a punch in channel 12 of the control tape. These hubs are always wired to SKIP TO 1.

16. *Non Pr—Non Print.* These hubs are normally impulsed from CONTROL INFORMATION to suppress printing, but not carriage operation, for that print cycle.

17. *Constant Write Impulse.* These hubs emit early-timed impulses on every print cycle.

18. *Co-Selector Pickup.* These are the pickup hubs for the co-selectors. When impulsed, they cause the co-selector to transfer immediately. They are normally wired from CONTROL INFORMATION.

19. *Unload.* These hubs accept early-timed impulses to cause words 1-8 of the output-synchronizer to be transferred automatically to the printwheels.

20. *Unload Alphabetic Words.* These hubs are impulsed from CONTROL INFORMATION whenever information contained in words 2-8 is designated as alphabetic in an unloading operation.

21. *Auto Stop.* These hubs accept any impulse to suspend printer operation at the end of the print cycle.

22. *EWI—Error Write Impulse.* This exit hub is directly under control of the 7070 program (UWIV command). The EWI hubs are normally used to pick up selectors, impulse AUTO STOP, etc.

23. *CI Off—Control Information Off.* These hubs are normally impulsed from EWI to make the control information hubs inactive.

24. *WVI—Write Validity Impulse.* These hubs emit an impulse if the print validity check circuits detect a non-valid character. They are normally wired to AUTO STOP. If they are left unwired, the detection of an invalid character causes the validity check light on the printer to flash.

25. *Pr+.* This switch is wired when 12 impulses (plus signs) are to print for positive numerical words.

26. *Run Interlock.* This switch must be connected to complete the print run circuit. Its purpose is to prevent printer operation without a wired panel in place.

27. *Sign Control Exits.* These hubs emit early-timed impulses when the corresponding output word is negative. These impulses can be used to pick up selectors, etc.

28. *Sign Exits.* These hubs emit 12 impulses (+ signs) for positive or alphabetic words and 11-impulses (− signs) for negative output words. These impulses are wired to the print entry hubs, and signs are printed in connection with the zero print control wiring. *Note:* + signs emit only when PR+ is wired.

29. *Field Selectors.* There are ten 11-position field selectors provided. Field selectors 1-5 have 3 levels each (Normal, level 1 and level 2); selectors 6-10 have 4 levels each (normal, level 1, level 2, and level 3). When a field selector is impulsed to transfer at a given level, a connection exists between common and that level. If the field selector pickup has not been impulsed, the connection is between common and normal.

30. *Word Exit.* These hubs emit impulses that are wired to word-size entry to determine which portions of the 16-word synchronizer are to be directed to the 80-position storage exit hubs. They can be selected but should never be directed to any control panel location except WORD SIZE ENTRY.

31. *Word Size Entry.* These hubs are always wired from the word exit hubs to cause specific numbers of characters to be emitted from the storage exit hubs for each output word.

32, 33, 34. *Comma, Decimal, Dollar Symbol.* These symbols are emitted on every print cycle. They differ

from the same hubs in the character emitter in that they can be controlled by zero print wiring.

35. **Ctrl—Asterisk Control.* These hubs emit an impulse for asterisk control and are wired to co-selector pickup hubs to control zero print control hubs when check-protecting asterisks are being printed. They differ from the regular asterisk in that they print from the check-protecting asterisk impulse rather than an 8-4-11 impulse.
36. *Entry 0.* The zero entry hubs are direct connections to complete a circuit from any lower zero print control hub wired to them. Normally, they are wired from zero print control hubs to print zeros for zero balances. However, more than eight zeros should not be printed in this manner.
37. **Entry.* The asterisk entry hubs are direct connections to complete the circuit for printing check-protecting asterisks (N-impulse). They are normally wired from zero print control.
38. *+ − Entry.* The plus and minus entry hubs are direct connections to complete the circuit for printing a plus (12 only) or a minus (11 only). They are normally wired from zero print control.
39. *Character Emitter.* All digits, letters, and special characters are emitted from the character emitter hubs during every print cycle.
40. *Co-selectors.* Sixteen co-selectors are standard. Each selector has five positions. When the pickup hubs are impulsed, the selector picks up immediately and holds to the end of the same cycle.
41. *Print Entry.* The 120 print entry hubs are multiple entries to the printwheels. They are usually wired from storage exits or from emitters to print either numerical or alphabetic information.
42. *Storage Exits.* These are exits for any 80 characters from the sixteen words contained in the output-synchronizer. They are normally wired to the print entry hubs.
43. *Zero Print Control.* Each printwheel has a pair of zero print control hubs. They are jackplugged to print zeros to the right of significant digits. The high-order position of a field is always left unplugged. To print zeros, zero impulses must be received by the printwheel from the storage exits or from an emitter. A maximum of eight zeros may be controlled to print to the left or right (or four on each side) of a significant digit, or for zero balances. The zero print control hubs also control printing commas, decimals, and dollar signs as they do zeros, when these symbols are wired to print entry from the comma, decimal, and dollar special emitter.

Inquiry

The IBM 7900 Inquiry Station (Figure 239) is a means of sending a message to the 7070, having a stored-program routine process it, and obtaining a typed reply when it comes back. As many as 10 inquiry stations can be used by a 7070 system. They are organized into two groups of five stations each, called inquiry control 1 and inquiry control 2. Each group is completely independent of the other. The five (or less) stations in each group are interdependent; when one is being used, the others are inoperative until the reply has been typed.

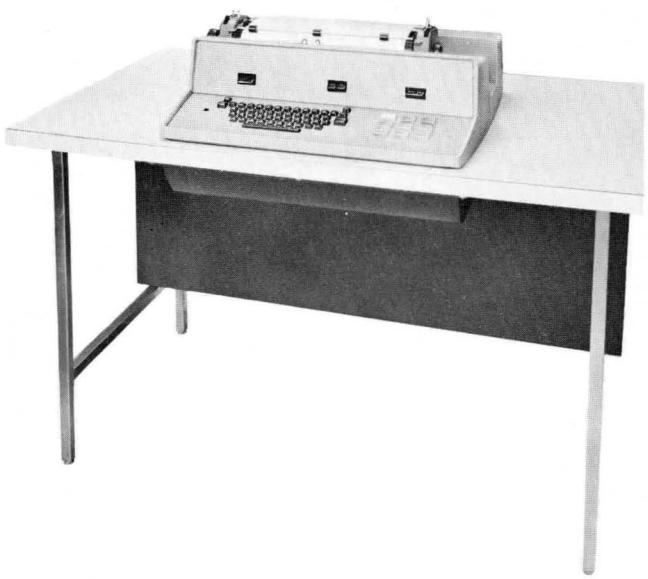


FIGURE 239. IBM 7900 INQUIRY STATION

Each group has a 10-word synchronizer, which is used for the inquiry input and output of the reply. Data is moved between main core storage and the synchronizers through the 7601 Arithmetic and Program Control, in a manner similar to the movement of data between the unit-record equipment and core storage. The movement of data to and from the synchronizers is controlled by record-definition words. Formats for the inquiry and reply between the station and the synchronizer are controlled by a perforated control tape at the inquiry station. Three different formats can be made available for each station. Figure 240 is a schematic representation of the 10 stations in the two control groups connected to the synchronizers. Each inquiry unit has 50 feet of cable, but additional lengths can be obtained. The maximum cable length connection of a station from the central processing unit is 2500 feet. Each inquiry typewriter can be used for manual typing. Single-case typewriters are used.

Inquiry is automatically on a priority basis. When an inquiry is made, the program receives a priority signal and branches to a sub-routine to process the inquiry data. The program branches to location 0106 on a priority signal from an inquiry control 1 station, and to 0107 on a signal from an inquiry control 2 station.

CONTROL WORDS: The first word on the synchronizer is the control word. Its data is emitted from the control tape and contains the format number (1-3) in position 4 and station number (1-5) in position 5. (The other 8 positions can have any digits, but must contain digits. They can come from the keyboard or be emitted from the tape.) The control word must be in word one for the reply, to enable the 7070 to send the reply to the desired station and format.

Operation

The manual operating features of an IBM 7900 Inquiry Station are the typewriter keyboard and the control keys and lights. The typewriter keyboard is on the left side of the panel (Figure 241). It has all of the features of an IBM electric typewriter, except that it types in single-case only.

The master power off key is located above the control keys and lights and should be used only in an emergency. Pressing this key cuts off immediately all ac and dc power to the inquiry station. Power is restored by pushing the key up.

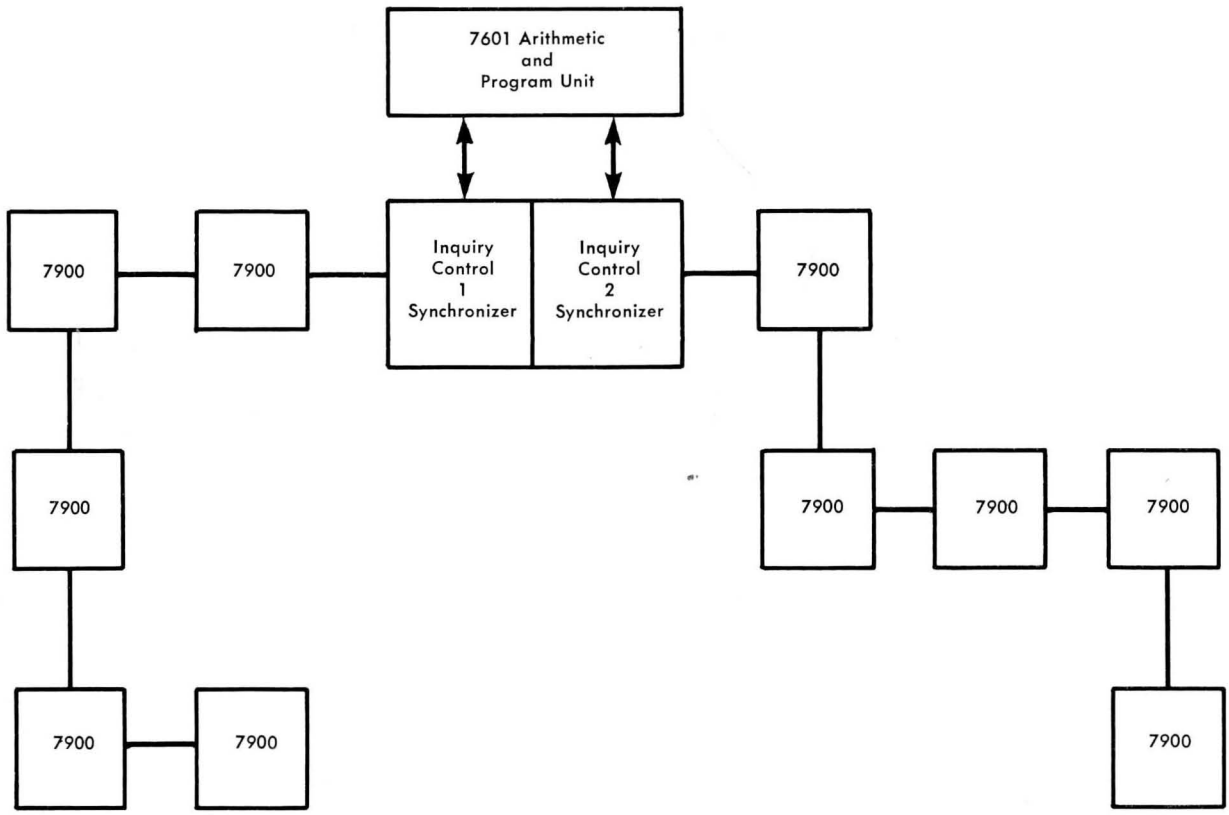


FIGURE 240. INQUIRY CONTROL GROUP SCHEMATIC

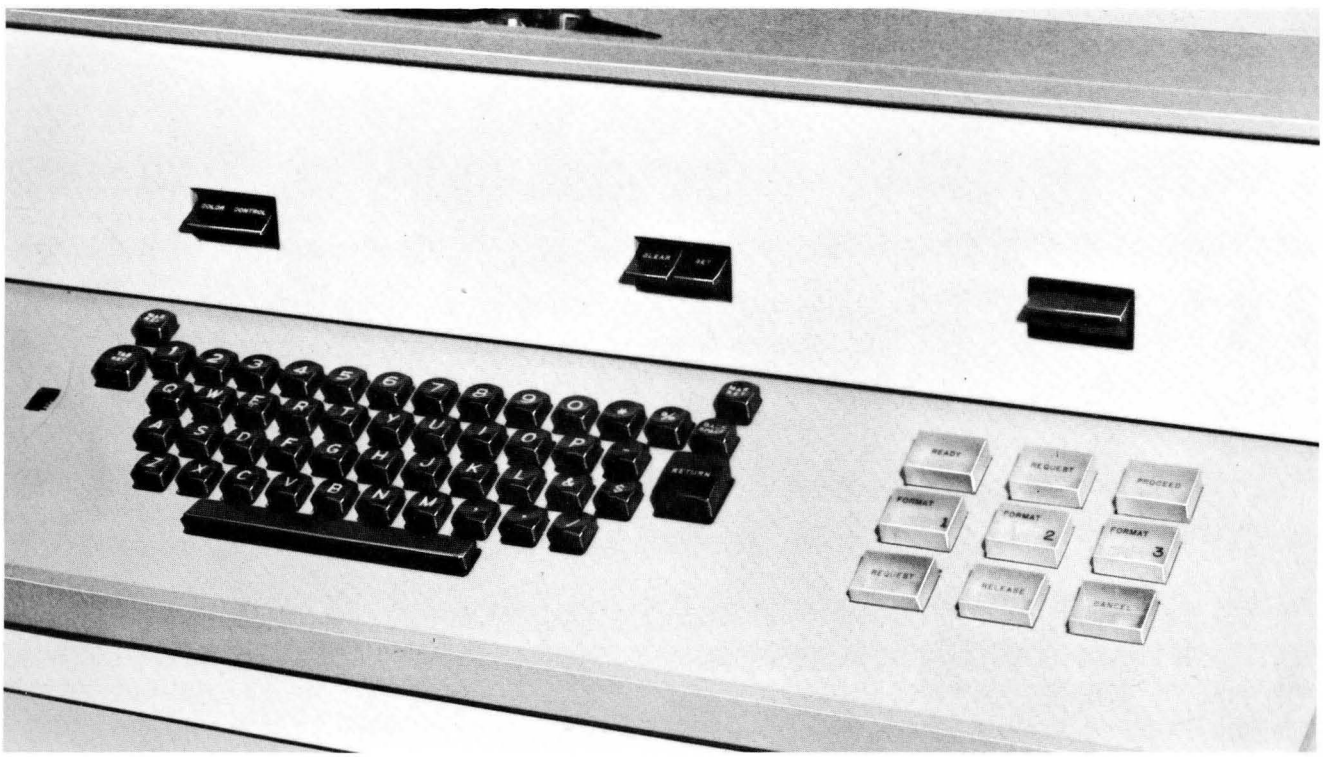


FIGURE 241. INQUIRY STATION TYPEWRITER KEYBOARD, AND OPERATING KEYS AND LIGHTS.

The nine keys and lights on the right side of the panel are the means of signalling that an inquiry is to be made, selecting the format, indicating and cancelling errors, and signalling that the inquiry has been completed. Figure 241 shows the keys and lights. REQUEST and PROCEED are lights, and the other seven are *illuminated keys*. An illuminated key is both a key and a light. The light comes ON as a signal that the key has been pressed ON, in the case of the ready and format keys, or as a signal that the key can be pressed.

If the main 7070 system is not ON when the ready key-light is pressed, the key-light is amber color, when it comes ON. If the 7070 is ON when READY is pressed, the key-light is green, indicating that inquiries can now be made. If the inquiry-station is ON, but the main 7070 system is not (ready key-light amber) and the 7070 is then turned on, the ready key-light turns green.

The operation of the control keys and lights is shown in Figure 242, which illustrates the manual operations and what happens as a result of them. The manual operations (four of them if no errors) are indicated by asterisks. Machine operations that take place at the same time are included in brackets.

READY Light on — Station is available for use.

*Press one of the three FORMAT Keys. This selects one of the three formats on the control tape.

The control tape positions at the proper format position.

[The FORMAT Key light comes on.
[The REQUEST Key light comes on.

*Press the REQUEST Key.

[The central processing unit is signalled that an inquiry is coming.
[The REQUEST Key light turns off.
[The REQUEST light (top row) comes on.

If no other inquiry station in the control group is being used (if the synchronizer is available).

The PROCEED light comes on.

The REQUEST light turns off.

The control word is automatically entered into the synchronizer from the control tape.

*Type the inquiry message. If the message is the correct length, the RELEASE Key light comes on.

*Press the RELEASE Key.

[The RELEASE Key light turns off.
[The PROCEED light turns off.
[The FORMAT Key light turns off.
[The Inquiry Control (1 or 2) stacking latch is set on, thus signalling the 7070 for priority. (See section on Automatic Priority Processing.)

FIGURE 242. INQUIRY PROCEDURE

Checking

There are several automatic checking features in 7070 inquiry operations to assure the correct operation of the keyboard, the transmission of correct data from and to inquiry station, and the proper station and format for the reply.

REQUEST

There are four automatic check operations for each request operation at an inquiry station. They are described here in the sequence in which they occur.

KEYBOARD VALIDITY CHECK: As each key is pressed at the inquiry station keyboard, it is translated into the 2-out-of-5 coding for that character and recorded on the inquiry-control synchronizer at the position designated by the control tape. (Numerical digits are translated into one digit each, alphamerical characters are translated into two digits each.) There is an automatic validity check on each character as it is moved from the translator to the synchronizer, to assure that it was translated into a valid 2-out-of-5 bit combination. If an invalid character is detected, the cancel key-light comes on.

COMPARE CHECK: After each typed character is recorded on the synchronizer, it is automatically compared with the relays set by pressing the keyboard key. This assures that the character on the synchronizer is the same as the one keyed in. If the compare operation detects an inequality, the cancel key-light comes on.

KEYBOARD OPERATION CHECK: An error in typing may be detected by the machine or by the operator. It is detected by the machine if the typed message is too short or too long; it is too short if the release key is pressed before its light comes ON (before the end-of-message indication on the tape); it is too long if the operator attempts to continue typing after the release key light has come ON. (End of message is indicated on the tape by detection of the start position of the next format.) If either of these errors occurs, the cancel key-light comes on.

Pressing the space bar as the last "character" in a request operation is considered an operation error by the machine. This is to prevent an operator from attempting to fill out the inquiry by pressing the space bar, with the possibility that the typed information has not gone to the correct synchronizer locations. The cancel key-light comes on.

CANCEL KEY-LIGHT(S): Whenever the cancel key-light comes on, the operator must begin the inquiry operation again. Pressing the cancel key turns off the light and reverts the inquiry station to the *ready* status. The operator begins the inquiry procedure again by pressing a format key.

If an operator discovers an error in typing, he can start over again by pressing the cancel key.

SYNCHRONIZER VALIDITY CHECK: There is a validity check on the data moved from the synchronizer to magnetic-core storage. If an invalid character is detected, the next sequential program instruction is taken. If there are no invalid characters, the next instruction is taken from the location of the inquiry-read instruction, + 2.

REPLY

There is no validity check on the movement of data from core storage to an inquiry-control synchronizer. Any invalid character on the synchronizer is detected during the data transmission from the synchronizer to the inquiry station itself, and is automatically typed as a red asterisk.

INQUIRY STATION AND FORMAT: The inquiry station and format number, in positions 4 and 5 of the control word respectively, are tested for validity and correctness when they are in the synchronizer. The station number must be 1-5, and the format number 1-3. If the station and format are correct, the typing operation is initiated, and the program takes its next instruction from the location of the reply instruction, +2. If they are not both correct, there is no typing, and the program takes the next sequential instruction. While a reply is being typed, the specified format key-light is ON.

Control Tape

The control tape at an inquiry station is a 16-channel perforated tape and provides for the following:

1. Arrangement of data for entry to the synchronizer from the typewriter during the inquiry, and to the typewriter from the synchronizer during the reply.
2. Control and identification of the inquiry station and format being used (control word).
3. Forms control through carriage tabulation, line spacing, etc.
4. Zero suppression on reply.

The tape is continuous, and a complete revolution can be from 12 to 48 inches in length. A sample section of tape is shown in Figure 243.

The tape moves one position for each character being typed in both inquiry and reply, selecting the synchronizer-word position to be read into or read from, respectively.

The synchronizer positions not read into are automatically filled with zeros. (An exception to this is the first word of the synchronizer in the inquiry operation, which must be completely filled with data from either the control tape or the keyboard.)

FUNCTIONS OF THE CHANNELS

Channels 1-5 specify the synchronizer word that the typed digit is to go to (inquiry) or come from (reply). The ten words are represented by the digits 1-9 for the first 9, and 0 for word 10, in two-out-of-five code, with code values 0, 1, 2, 3 and 6 represented by channels 1, 2, 3, 4, and 5 respectively:

Synchronizer Word	2-out-of-5 Code	Tape Channels
1	0-1	1-2
2	0-2	1-3
3	0-3	1-4
4	1-3	2-4
5	2-3	3-4
6	0-6	1-5
7	1-6	2-5
8	2-6	3-5
9	3-6	4-5
10	1-2 (= 0)	2-3

Channels 6-10 specify the digit position of the synchronizer word, in the same manner as channels 1-5 specify the word; 0, 1, 2, 3, and 6 are represented by

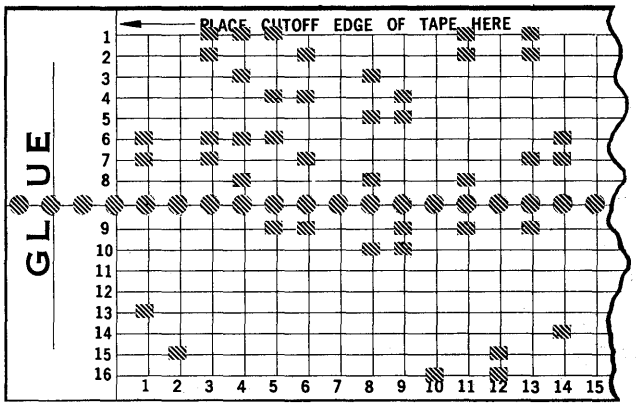


FIGURE 243. INQUIRY STATION CONTROL TAPE

channels 6, 7, 8, 9, and 10, respectively. The digit positions are 0-9 from left to right, or high-order to low-order:

Digit Position	2-out-of-5 Code	Tape Channels
0	1-2	7-8
1	0-1	6-7
2	0-2	6-8
3	0-3	6-9
4	1-3	7-9
5	2-3	8-9
6	0-6	6-10
7	1-6	7-10
8	2-6	8-10
9	3-6	9-10

Channel 11 designates an alphabetic character. The synchronizer-word position represented by channels 1-10 is the low-order position of the two that the letter is to go into (inquiry) or come from (reply). Whenever channel 11 is punched for alpha, channels 6-10 must designate position 1, 3, 5, 7, or 9. The word designated by channels 1-5 automatically gets an alpha sign.

Channel 11 is also used in conjunction with position code zero, to allow operator control over the sign of a numerical word. The tape has the word number in channels 1-5 and a zero designation in channels 6-10 (specifically, channels 7 and 8). When the tape is at a position with this coding, the operator determines the sign of the specified word by pressing either the "&" key for plus or the "-" key for minus. When a reply is being typed, this coding causes a minus sign to be typed for the designated word, if that word is minus. Plus and alpha signs are not typed.

Any word, the sign of which is not made alpha or minus in the request operation, is automatically given a plus sign on the synchronizer. In all cases of both request and reply, channel 11 is used to create alpha signs and allow minus signs.

Channel 12 has two functions. In inquiry, it specifies that the typed character is to go into the control word, in the position designated by channels 6-10. Channels 1-5 are thus not needed to indicate synchronizer word. They are used instead to specify the actual digit that is to go into the control-word position. Channel 12 can be thought of as making the control tape a digit emitter. When a punch in channel 12 is sensed, the tape advances automatically, and no typing takes place.

In a reply operation, channel 12 is used for zero suppression. When a tape position is punched with a 12, no zeros will type until a significant digit is reached.

Channel 13 indicates the start position for the tape format in an inquire (request) operation. Channels 6-10 indicate the format by containing a 1 (channels 6-7), 2 (6-8), or 3 (6-9). When one of the format keys is pressed, the tape moves to the specified format position. When channels 12 and 13 are both punched, the program tape skips to the next 13 punch in the tape. The main use of this feature is to skip the lapped section of the program tape.

Channel 14 has the same function for reply that channel 13 has for inquire (request). It indicates the start position for the tape format, with the format specified in channels 6-10. When a reply instruction is given by the stored program, the inquiry station (specified by the digit in position 5 of the control word) moves its control tape to the desired format (specified by the digit in position 4 of the control word), if there is a punch in channel 14 for that format.

The combination of punches in channels 13 and 14 has a special function. Used at the end of an inquire (request) section of a tape format, it signifies that there is not to be a reply for this inquiry. It is used when an inquiry is made in order to insert data into storage.

A hole in channel 15 causes an automatic carriage return. The tape suspends data transmission within the 7070 until the carriage return is complete. Variable-field-length applications may require the operator to press the carriage-return key before channel 15 is reached. In this case, the program tape is automatically advanced to the channel 15 column.

The function of channel 16 is to cause the typewriter to tabulate automatically to the next tab stop. The advance of the tape and manual operation of the tab key for variable-length fields is identical to the previously described carriage-return operation by channel 15. Tabulations should not be programmed for fewer than three spaces.

When both channels 15 and 16 are punched, the typewriter is caused to space one column. The spacing function suspends any transmitting or receiving of data. Spacing from the program tape functions on both the input and output modes of operation.

Channels 15 and 16 must not be punched in columns with word or digit punching.

Operation Code

An inquiry routine is usually started by an inquiry-read instruction. This brings to magnetic-core storage the information that had been typed at the inquiry station. When the inquiry sub-routine has been completed, an inquiry-reply instruction is given, which transmits the reply data to the inquiry station and automatically initiates typing the reply data. The stored program does not have to wait for the request data to be typed by the

inquiry-station operator, nor for the reply to be typed by the reply instruction.

Inquiry Control +54

MACHINE DESCRIPTION: All of the instructions for read and reply from and to the inquiry stations are included in the augmented code +54, INQUIRY CONTROL. Positions 4 and 5 of the instruction define the operation, and which of the two inquiry-control groups is involved. (As described, the specific inquiry station within the control group is identified in word 1 of the request and reply data.)

INSTRUCTION FORMAT:

S01	23	4	5	6789

S01	+54
23	Indexing word
4	Inquiry control group; 1 or 2
5	Operation: 0. Inquiry Read QR 1. Inquiry Write QW
6879	Address of first record-definition word (indexable). This word defines the core-storage area to be read into on a read operation, or read from, on a write operation.

INQUIRY READ: This instruction initiates the transmission of data from the synchronizer specified in position 4, to the core-storage words defined by the RDW(s) addressed by positions 6-9. Transmission is stopped by the RDW(s), or when 10 words have been moved, whichever comes first. The stored program proceeds when this transmission is completed.

The data is validity-checked as it moves from the synchronizer to core storage. If there is an invalid character, the next instruction is taken from the address in the instruction counter. If there are no invalid characters, the instruction counter is incremented by 1, and the next instruction comes from that address.

INQUIRY WRITE: This code works in the same way as inquiry read, except that the flow of data is from core storage to the synchronizer, and from there to the inquiry station. The stored program can continue after the data has been transmitted to the synchronizer.

All of the reply data does not require a validity check because an asterisk is automatically typed in red for any invalid characters detected on data movement from the synchronizer to the inquiry station. There are two tests made of the reply data: inquiry station number and format are checked for validity, and to see whether an inquiry station of that number exists on the 7070 system. If there is an invalid character, the next instruction is taken from the address in the instruction counter. If there are no invalid characters, the instruction counter is incremented by 1, and the next instruction comes from that address.

EXAMPLES: To read request information from a station in control group 2, into the core storage area for which word 4400 is the first RDW:

S01	23	4	5	6789
+54	00	2	0	4400

Assume that at the completion of the inquiry sub-routine, the reply data has been put into the same storage area as the request data. To send this reply to the inquiry station:

S01	23	4	5	6789
+54	00	2	1	4400

DATA FLOW: The movement of data between the synchronizer and core storage is the same as for unit-record read and punch/print operations. For a read operation, a word on the synchronizer is read serially into the synchronizer register, from there in parallel to the arithmetic register, and from there in parallel to magnetic-core storage under control of the record-definition register. The flow is opposite for a write operation.

REGISTERS AFFECTED: The synchronizer register and the arithmetic register.

TIMING: An inquiry-read (QR) instruction is timed as follows:

Microseconds
64
+36 if indexing is used
+132 for each word moved—1320 for 10 words

An inquiry-write (QW) operation takes longer, due to the necessity of clearing the synchronizer before the data is moved, and because of the automatic station and format check:

Microseconds
156
+36 if indexing is used
+1320 for clearing the synchronizer
+132 for each word moved (1320 maximum)
+5000 station-format check

COMMENTS: An inquiry routine is automatically a priority routine; it is started as the result of an inquiry-priority signal. Therefore, a priority release instruction (PR) is necessary; this is usually the next instruction following the QW instruction.

The inquiry-station number and format number are automatically put into word 1 of the request data by the control tape at the inquiry station. The reply data must have a valid station and format number in these same positions. Normally, the station number is the same, so that the reply is typed at the same station that made the inquiry (it doesn't have to be, however). The format number can be different, so that the reply data need not conform to the same format as the request data.

A reply is required for all inquiries, unless the control tape for the inquiry format has both channels 13 and 14 punched in its last position.

AUTOCODER EXAMPLE (Figure 244): ZETA has been previously defined as word 2330. The assembled instruction is:

S01	23	4	5	6789
+54	00	1	0	2330

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		QR	1,ZETA		

FIGURE 244.

Automatic Priority Processing

One of the most important considerations in planning and programming an application for a data processing system is that of deriving the most efficient use of all components of the system. In some programs this may be card reading or card punching, in others it may be tape reading or writing, etc. Programs are said to be input-bound, printer-bound, seek-bound, compute-bound, etc., referring to the phase of the application that the other phases must wait for, at some point or points in the program. Sometimes it is desirable to combine two or more independent programs, so that one is functioning while the other is waiting for one of its operations to be completed.

An outstanding feature of the IBM 7070 is *Automatic Priority Processing,* which makes it possible to combine programs and virtually eliminate any lost time waiting for an operation to be completed. There need not be a delay; one program or the other is constantly functioning.

For example, one program, called the main routine, may have a comparatively large number of program steps. Another called the priority routine has relatively few instructions but involves almost continuous use of a card reader, card punch, printer, tape unit, or disk file.

The priority routine can be a sub-routine that serves the main program, or it may be a complete program in itself, totally independent of the main program. The latter case is described by the term SPOOL, for Simultaneous Peripheral Operations On-Line. It is called this because it is a separate operation that is performed concurrently with the main program, and in timing it is almost the equivalent of two simultaneous operations.

The main routine functions normally, while the tape, disk storage, or card unit of the priority routine is operating—reading a card, punching a card, printing a line, reading tape, writing tape, seeking a disk-storage record, reading a disk record, or writing a disk record. As soon as that operation is completed, the main routine is automatically signalled. The location of the next main-routine instruction is stored in positions 2-5 of index word 97, and the priority routine carries out its comparatively short program, starts its input/output or storage unit, and releases priority. The main routine then takes up exactly where it left off, by bringing the address of its next instruction from index word 97 to the instruction counter.

The number of individual programs need not be limited to two; it is possible to have more than one tape, disk storage, or card input/output unit operating on a priority basis during the main routine. However, only the main routine can be signalled for priority; it is not possible to do this to a priority routine. If a second priority is ready while a first one is in progress, it will wait until the first is completed. The main routine is resumed only when there are no priority routines waiting.

Several devices are used by the 7070 in Automatic Priority Processing. They are:

Stacking Latches

A stacking latch is the means by which an input/output or storage unit signals the main routine for priority. When the unit has finished its operation, it sets a stacking latch. The main routine is constantly testing to see whether any stacking latches have been set, with no loss of time.

There are 56 stacking latches in a full-capacity 7070, assigned as follows:

Two for any two of the six unit-record machines	2
One for each inquiry control group of 5 stations	2
One for each tape unit	40
One for each of the 3 access arms in each of the 4 disk-storage units	12
	56

Sequencing Scanner

The stacking latches are continuously scanned by the *sequencing scanner*. Figure 245 is a representation of how this is done. The scanner is constantly scanning, testing each of the stacking latches in turn. If two latches should be turned on at the same time, the one detected first depends on the position of the sequencing scanner at that time. Figure 245 shows the sequencing scanner stopped by the stacking latch for tape unit 0, channel 1.

If a stacking latch is turned on and it has not been masked, the sequencing scanner stops at that latch the next time it gets to it. When the priority routine is started, the stacking latch is turned off and the sequencing scanner proceeds again.

A scan of all 56 stacking latches takes about 1.2 milliseconds.

Priority Waiting Latch

The setting of this latch causes a priority signal to be generated and a priority routine to be initiated. It is set only when all three of these things have occurred:

1. A stacking latch has been set, provided that
2. it is not masked, and
3. the sequencing scanner has reached it.

Priority Control Masks

A stacking latch can be masked so that it will not cause a priority signal when it is set. The mask prevents a stacking latch from stopping the sequencing scanner,

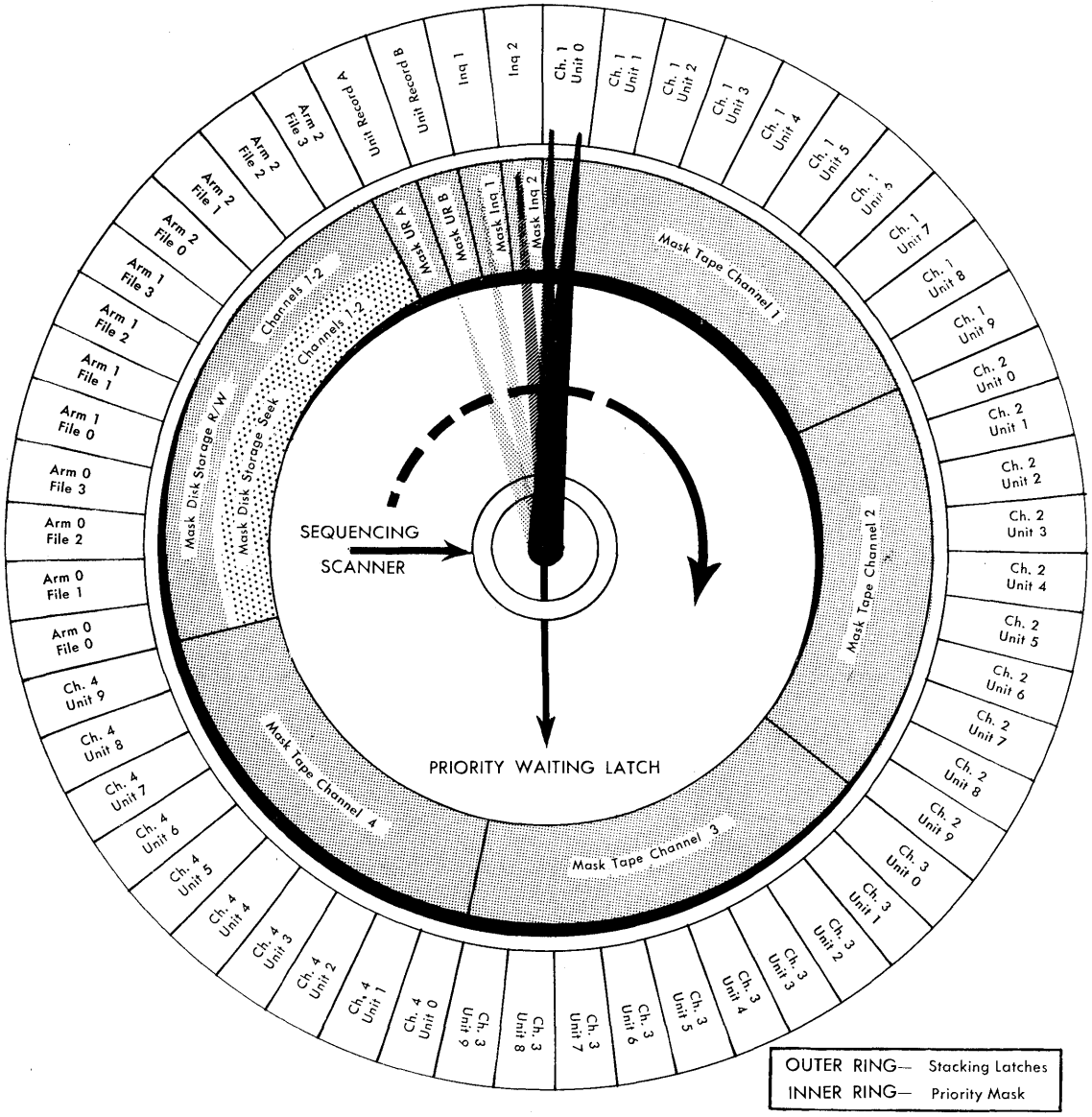


FIGURE 245. STACKING LATCHES, PRIORITY MASKS AND SEQUENCING SCANNER

and thus setting the priority waiting latch. Figure 245 shows how the latches are masked. Note that an entire tape channel is masked, thus prohibiting any of the six tape units connected to it from setting the priority waiting latch. In the illustration, tape channel 2 is masked. Thus, the stacking latch set by tape unit 5 on channel 2 cannot stop the sequencing scanner. Unit-record B stacking latch is also masked; the stacking latch is not set, however. The twelve access arms in the disk storage units are all either masked or not for read/write operations, and are all either masked or not for seek operations, with each mask independent of the other. The priority masks are set by the PRIORITY CONTROL code, +55.

Index Word 97

When the main routine is signalled for priority, the contents of the instruction counter are automatically sent to positions 2-5 of index word 97. The instruction counter contains the address of the next program step to be executed in the main routine. Thus, when the priority routine is completed, the PRIORITY RELEASE command returns the main routine to the point where it left off. The sign of index word 97 is set to plus.

Priority Indicator Storage Word, 0100

There are 10 indicators and two sense/stop switches that are set during a stored program:

Accumulator overflow (one indicator for each accumulator)

High-Low-Equal compare

Floating-decimal overflow and underflow

Field overflow, and setting of field-overflow sense/stop switch

Sign change, and setting of sign change sense/stop switch

At the beginning of a priority routine, the settings of these 10 indicators and the two sense/stop switches are automatically stored in word 0100. The settings are restored to the indicators and switches by the PRIORITY-RELEASE command at the end of the priority routine. Thus, any changes in the indicators by the priority routine have no effect on the main routine.

The values in word 0100 are dependent on the indicator settings as follows:

Word 0100,
position:

9. Accumulator 3 overflow
   0. No overflow
   1. Overflow
8. Accumulator 2 overflow
   0. No overflow
   1. Overflow
7. Accumulator 1 overflow
   0. No overflow
   1. Overflow
6. Low
   0. Not low
   1. Low
5. Equal
   0. Not equal
   1. Equal
4. High
   0. Not high
   1. High
3. Floating point overflow
   0. No overflow
   1. Overflow
2. Floating point underflow
   0. No underflow
   1. Underflow
1. Field Overflow
   0. No overflow, and field-overflow sense/stop switch is at stop.
   5. No overflow, and field-overflow sense/stop switch is at sense.
   9. Overflow, and field-overflow sense/stop switch is at sense.
0. Sign change
   0. No sign change, and sign-change sense/stop switch is at stop.
   5. No sign change, and sign-change sense/stop switch is at sense.
   9. Sign change, and sign-change sense/stop switch is at sense.

If any of these digits are changed in the priority routine, the corresponding indicator setting will be affected when priority is released.

Priority Mode Latch

A 7070 program is always in priority mode or non-priority mode. This latch is set when a priority routine is started and is reset by the priority release instruction. It prevents a priority routine from being interrupted by another priority signal.

Status Words

These are formed by tape and disk-storage reading and writing operations, to enable the program to reference a priority condition back to the initiating command and to ascertain the type of priority signal. The status words are explained in detail in the texts on tape and disk-storage priority.

Priority Operation

During each program step in the main routine a test is made to determine whether any stacking latches have been set, by testing the priority waiting latch. If it is not ON, the program continues normally. If the priority waiting latch has been set, by a tape or disk read-write or disk-seek operation, the status words have been formed and stored.

The following takes place as the result of the test having found the priority waiting latch set:

1. The operation currently taking place in the main routine is completed—recomplementing, etc.
2. The contents of the instruction counter—location of the next main-routine instruction, are stored in positions 2-5 of index word 97, with the other positions of 0097 set to zero and its sign to plus.
3. The indicator settings are stored in word 0100.
4. The priority mode latch is set.
5. The stacking latch and priority waiting latch are turned off and the sequencing scanner starts again.
6. If the interrupt was caused by tape or disk, the address of the final status word for the particular unit or arm is placed in index word 99.
7. The program branches to the location of the first instruction in the priority routine. These locations depend on what caused the priority signal, and they are covered in the texts on the types of priority.

Types of Priority

There are four types of priority:

Unit Record
Inquiry
Tape
Disk Storage

Unit-record priority is initiated by the completion of a card-read, card-punch, or print operation. Inquiry priority is initiated by the release of an inquiry message by the operator. The completion of a tape read or tape write operation can start a tape-priority routine, and disk-storage priority can be caused by the completion of read, write, or seek.

Unit-Record Priority

A priority routine caused by the completion of a read, punch, or print operation is called Unit-Record Priority.

There can be two unit-record operations available for priority at any one time. Console switches determine whether or not there will be priority, and which unit or units will be involved. The two priority controls are called A and B, and each can be assigned to any of the six input-output units. Assignment is made by the console dials. The right-hand dial in each set determines the unit that will set that stacking latch each time an operation is completed (see Figure 279).

In operations involving magnetic tape, such as card-to-tape, tape-to-printer, etc., priority signalling of the main program can be automatically delayed until both the card unit and the tape channel are free. This is done by setting the left-hand dial to channel control 1, 2, 3 or 4. When the unit record priority A dial is set to channel control 2, for example, the sequencing scanner, when stopped by the unit record A stacking latch, will not set the priority waiting latch until channel 2 is free.

The first instruction in a priority routine initiated by priority control A is located in word 0104; for priority control B, it is in word 0105. The instruction in each of those locations is normally an unconditional branch to the location of the first instruction in the priority routine.

Inquiry Priority

Inquiry priority is automatic. A manual inquiry to storage from any of the five stations in each inquiry control group automatically starts a priority routine (unless, of course, the stacking latch is masked). A stacking latch for inquiry control 1 or 2 is set by pressing the release key at the inquiry station, after the inquiry request has been typed.

A priority routine resulting from an inquiry control 1 station starts in location 0106; resulting from an inquiry control 2 station, in 0107. As with unit-record priority, the instructions in these locations are usually unconditional branches.

Tape Priority

The priority signalling of the main program by the completion of a tape operation is called tape priority. There are two causes of tape priority: normal and unusual-condition. Normal tape priority means that the main program will be automatically signalled at the conclusion of the tape operation, just as in card priority. Tape

priority, however, is under control of the stored program rather than the console. A plus sign in the tape control instruction determines that there will be a priority signal at the conclusion of the operation; a minus sign determines that there will not be a signal for normal condition. In *Autocoder* symbols, the letter P is used if the operation is to signal normal priority. Unusual-condition priority occurs regardless of the sign of the instruction.

If the stacking latch for a tape unit is ON and another instruction involving that unit is called for, the program is interlocked until the stacking latch is turned off. Then, the program proceeds with the tape instruction. If the stacking latch is masked, or if the program is in the priority mode when this happens, the machine stops.

Tape Status Words

Status words are automatically provided, with no loss in time, for tape priority operations to help the programmer determine what operations caused the priority signal. There are two status words formed: *initial* and *final*.

TAPE INITIAL STATUS WORDS: Every tape instruction that might cause a priority signal when it is completed, automatically creates an initial status word at the beginning of the operation. The format of the initial-status word is as follows:

Tape Initial Status Word S 0 1 2345 6789

S — The sign of the tape instruction: plus, priority; minus, no priority.

0 — Always 8, from position 0 of the instruction, identifying it as a tape operation.

1 — Defines the specific operation—contains the digit in position 5 of the tape instruction.

2345 — Contents of the instruction counter when the tape instruction was given—the location of the next sequential instruction after the tape command.

6789 — Contents of the corresponding positions of the tape instruction. If position 5 of the instruction (position 1 of this word) contains a digit 1-9, it is the location of the first RDW. If position 5 contains 0, position 9 contains the operation code, and positions 6-8 can contain any digits.

Each tape unit has a location for an initial status word: 0160-0169 for the ten units for channel 1, words 0170-0179 for the ten units for channel 2, 0180-0189 for channel 3, and 0190-0199 for channel 4. Each time an initial status word is created by a tape control instruction, it is stored in the location corresponding to the tape unit and channel specified.

TAPE FINAL STATUS WORDS: A final status word is automatically created at the conclusion of each tape read or write operation. The final status word for each tape unit is stored in a location whose address is 50 *less* than that of the initial status word: 0110-0119 for channel 1, 0120-0129 for channel 2, 0130-0139 for channel 3 and 0140-0149 for channel 4. The ten's position specifies the channel, 1-4; and the units position specifies tape unit, 0-9. The format of the final status words is as follows:

Tape Final Status Word S 0 1 2345 6789

S — Always plus.

0 — This position contains a digit put in by the program itself and is not disturbed by formation of the rest of the word. It is used to determine the exact address that the program will branch to in starting the priority routine.

1 — Condition code, indicating the cause of the priority signal:

1. Error
2. CLR
3. SLR
4. LLR
5. EOF
6. EOS
7. SCLR

2345 — Working address of the record definition register at completion of the operation—the last core-storage location used by the last RDW in the operation (except for LLR).

6789 — Contents of the RDW address register—the location of the last record-definition word used, +1.

The location of this word (0110-0149) is automatically stored in the indexing portion (positions 2-5) of index word 99 when the priority routine is started, the sign of 0099 is set to plus, and the remaining positions of the word are set to zeros.

The final status word developed as a result of (P)TM and (P)TSM is a special case. Only the condition code (position 1) resulting from the operation is inserted. The remaining positions and the sign are left unchanged. This is because RDW's are not used by these instructions.

TAPE PRIORITY ROUTINE START ADDRESS: To start a tape-priority routine, the program branches to one of the ten locations 0150-0159, determined by the digit in position 0 of the final status word.

Condition Codes

The condition code in position 1 indicates whether the priority signal is due to a normal or an unusual condition, and specifically what the condition is. All but CLR (2) are unusual-condition codes (they may be errors, or not). The digits in this position have the following meaning:

1. Error. These are machine checks on itself.

Read
1. Vertical redundancy check on tape character.
2. Longitudinal redundancy check on a tape record.
3. Two-out-of-five validity check on translated data.
4. A tape character not *translated* to 2-out-of-5 coding.
5. Detection of a Δ not at the end of a word.
6. In reading numerical data, the last word read has less than 5 digits.
7. In reading numerical data, 10 digits are read with no sign indication.
8. Insufficient signal strength.

Write
9. A two-digit combination in an alpha word that does not represent an alphamerical character (probably a programming error).
10. Validity error detected by the two-gap head.

The occurrence of an error condition takes precedence over any other priority conditions that may exist at the same time.

2. CLR—Correct Length Record. This is *normal* priority—called for by a plus sign in the tape instruction. This code is used only if none of the unusual conditions has occurred.

3. SLR—Short Length Record. Occurs if the working and stop addresses in the record definition register are not the same when the end of the tape record (the IRG) is reached, in tape reading. The address in positions 2-5 of the final status word is that of the last core-storage word filled.

   Short-length record occurs in TAPE-SEGMENT BACKSPACE if the load point (beginning of the tape) is reached before the specified number of segments has been spaced over.

4. LLR—Long Length Record. Occurs if the working and stop addresses in the record definition register coincide, but the end of the tape record (the IRG) has not yet been reached, in tape reading. The contents of positions 2-5 of the final status word is the sum of the address of the last core-storage word filled plus the number of excess words in the tape record. This will indicate to the programmer the number of words by which the record is greater than the storage area provided.

5. EOF—End of File. Occurs when reading a tape mark or writing into the foil strip, indicating that the end of the tape reel has been reached.

   Either of these operations turns on the end-of-file indicator. Until the indicator is turned off by a TEF operation, every subsequent read or write command will result in the EOF priority condition.

   It is possible to "coast" into the reflective spot after a write operation has been completed. If this happens, an EOF condition is indicated at the completion of the following write instruction.

   End-of-file occurs in TAPE-SEGMENT FORWARD SPACE and TAPE-SEGMENT BACKWARD SPACE if the tape mark is read before the specified number of segments has been passed over.

6. EOS—End of Segment. Occurs on reading a tape segment mark during a read command.

7. SCLR — Short Character Length Record. Occurs only in reading alpha data from tape if the number of characters do not fill an exact number of core-storage 10-digit words; that is, the number of alphamerical characters is not a multiple of five. When this condition occurs, the remainder of the word (low-order positions) is filled with zeros (alpha coding for blanks). The address in positions 2-5 of the final status word is that of the last core-storage word filled.

Precedence of Tape Priority Conditions

It is possible for more than one tape priority condition to occur at any time. For example, in writing a tape record, Correct Length Record and End of File may both occur.

In all cases, error priority takes precedence over all of the other conditions. Figure 246 shows the sequence of precedence of the condition codes and which codes apply to each tape operation. Note that status words are formed for every operation that might initiate a priority signal.

MNEMONIC	NAME	OPER.	PRIORITY POSSIBLE?	STATUS WORDS?	CONDITION CODES, IN ORDER OF PRECEDENCE HIGHEST ← UNUSUAL CONDITION 1. ERROR	5. EOF	6. EOS	4. LLR	7. SCLR	3. SLR	→ LOWEST NORMAL 2. CLR
		Position 5:									
TR	Tape Read	1	Yes	Yes	Error	EOF	EOS	LLR	SCLR	SLR	CLR*
PTR	Priority Tape Read	1	Yes	Yes	Error	EOF	EOS	LLR	SCLR	SLR	CLR
TRR	Tape Read per Record Mark Control	2	Yes	Yes	Error	EOF	EOS	LLR	SCLR	SLR	CLR*
PTRR	Priority Tape Read per Record-Mark Control	2	Yes	Yes	Error	EOF	EOS	LLR	SCLR	SLR	CLR
TW	Tape Write	3	Yes	Yes	Error	EOF					CLR*
PTW	Priority Tape Write	3	Yes	Yes	Error	EOF					CLR
TWR	Tape Write per Record-Mark Control	4	Yes	Yes	Error	EOF					CLR*
PTWR	Priority Tape Write per Record-Mark Control	4	Yes	Yes	Error	EOF					CLR
TWZ	Tape Write with Zero Elimination	5	Yes	Yes	Error	EOF					CLR*
PTWZ	Priority Tape Write with Zero Elimination	5	Yes	Yes	Error	EOF					CLR
TWC	Tape Write with Zero Elimination and per Record Mark Control Combined	6	Yes	Yes	Error	EOF					CLR*
PTWC	Priority Tape Write with Zero Elimination and per Record Mark Control Combined	6	Yes	Yes	Error	EOF					CLR
TSF	Tape Segment Forward Space per Count	7	Yes	Yes	Error	EOF					CLR*
PTSF	Priority Tape Segment Forward Space per Count	7	Yes	Yes	Error	EOF					CLR
TSB	Tape Segment Backward Space per Count	8	Yes	Yes	Error	EOF				SLR	CLR*
PTSB	Priority Tape Segment Backward Space per Count	8	Yes	Yes	Error	EOF				SLR	CLR
		Position 9:									
TSEL	Tape No-op Select	0	No	No							
TM	Tape Mark Write	1	Yes	Yes	Error	EOF					CLR*
PTM	Priority Tape Mark Write	1	Yes	Yes	Error	EOF					CLR
TRW	Tape Rewind	2	No	No							
TRU	Tape Rewind Unload	3	No	No							
TRB	Tape Record Backspace	4	No	No							
TSM	Tape Write Segment Mark	5	Yes	Yes	Error	EOF					CLR*
PTSM	Priority Tape Write Segment Mark	5	Yes	Yes	Error	EOF					CLR
TSK	Tape Skip	6	No	No							
TEF	Tape End of File Off	7	No	No							

*CLR does not initiate a priority condition.

FIGURE 246. PRECEDENCE OF TAPE PRIORITY CONDITIONS

Disk Storage Priority

Disk-storage priority operates on the same principle as tape. Normal priority on reading and writing is under control of the stored program (a plus sign in the disk-storage read or write instruction), and unusual-condition priority occurs automatically as the result of machine self-checking, programming errors, or situations that occur in the program.

The completion of a seek operation automatically initiates priority. However, the sequencing scanner, when stopped by a stacking latch at completion of a seek operation, does not set the priority waiting latch unless one of the two tape/disk channels (channel 1 or 2) is available for use. No status words are created in seek operations.

Disk Status Words

DISK INITIAL STATUS WORDS: The initiation of a disk-storage read or write operation automatically creates an initial status word, just as with tape, and with the same format:

Disk Initial Status Word S 0 1 2345 6789

S The sign of the disk-storage read or write instruction — plus, priority; minus, no priority.

0 Always 9, from position 0 of the instruction, identifying it as a disk-storage operation.

1 0 or 1, defining the specific operation as either read or write—contains the digit in position 5 of the disk instruction.

2345 Contents of the instruction counter when the disk instruction was given—the location of the next sequential instruction after the disk-storage command.

6789 Contents of the corresponding positions of the disk instruction—the location of the first RDW used by the operation.

Each of the 12 access arms has a location for initial status words: 0250-0253 for arm 0, files 0-3 respectively; 0260-0263 for arm 1, files 0-3 respectively; and 0270-0273 for arm 2, files 0-3 respectively.

DISK FINAL STATUS WORDS: The completion of a disk-storage read or write operation automatically creates a final status word, which for each access arm is stored in a location, the address of which is 50 *less* than that of its initial status word: 0200-0203 for arm 0, 0210-0213 for arm 1, and 0220-0223 for arm 2. The two low-order positions of each address define the exact access arm (ten's position, 0-2) and file unit (units position, 0-3). The format of the final status words is the same as for tape:

Disk Final Status Word S 0 1 2345 6789

S Always plus.

0 This position contains a digit put in by the program itself, and is not disturbed by the formation of the rest of the word. It is used to determine the exact address to which the program will branch in starting the priority routine.

1 Condition code, indicating the cause of the priority signal:
1. Error
2. CLR
3. SLR
4. LLR
5. Machine address check
6. Addressing error
7. Incomplete seek

2345 Working address of the record/definition register at completion of the operation—the last core-storage location used by the last RDW in the operation.

6789 Contents of the RDW address register—the location of the last record-definition word used, + 1.

The location of this word (0200-0203, 0210-0213, or 0220-0223) is automatically stored in the indexing portion of index word 99 when the priority routine is started. The remaining positions of 0099 are set to zeros, and the sign is set to plus.

Condition Codes

The condition code in position 1 indicates the cause of the priority signal. All but CLR are unusual-condition codes.

1. Error. These are machine checks on itself.
   a. A validity check (2 out of 5) in reading or writing.
   b. The automatic compare check on disk writing.
2. CLR—Correct Length Record. As in tape operations, this is *normal* priority. It is used if none of the other conditions has occurred. If the disk-storage read or write instruction had a minus sign and no other condition has occurred, this code goes to the final status word, but no priority signal is given.

3. SLR—Short Length Record. As in tape operations, this code means that the record on the disk track is shorter than the number of words specified by the RDW(s). Since there are always 60 words on a track, SLR in disk operations means that more than 60 words were specified by the RDW(s). SLR is possible on either read or write in disk operations.

   The address in positions 2-5 of the final status word is that of the last core-storage word filled in a read operation, or read from in a write operation.

4. LLR—Long Length Record. The record on the disk track is longer than the number of words specified by the RDW(s). The disk record is too long if less than 60 words are so specified. LLR is possible in either disk-read or disk-write operations. The address in positions 2-5 of the final status word is that of the last core-storage word filled in a read operation, or read from in a write operation.

5. Machine Address Check—The address in the two-word gap on the track to be written or read is not the same as the address put in the access register by the seek operation (machine error).

6. Programming Address Check—The address in accumulator 3 is not the same as the address put in the access register by the seek operation (programming error).

7. Incomplete Seek—The access arm has not completed a seek operation in the maximum time that should be necessary (machine error).

   Codes 1-4 are created at *completion* of a read or write operation. Codes 5-7 are created at the *beginning* of the first read or write operation after a seek. If condition 5-7 exists, no read or write operation takes place.

   If the addressed disk storage unit has been temporarily disconnected, an incomplete seek code is generated (if a unit that has never been part of the system is addressed, an error stop results).

DISK PRIORITY ROUTINE START ADDRESS: To start a priority routine from a disk-storage read or write operation, the program branches to one of the ten locations 0240-0249, determined by the digit in position 0 of the final status word.

Precedence of Disk Priority Conditions

It is possible, of course, for a disk-storage read or write operation to produce more than one priority condition. As with the tape condition codes, the disk-priority condition codes have an order of precedence, determining which condition code is used if two or more arise in the operation. Incomplete seek, code 7, takes precedence over all of the other conditions. CLR is last, and is used only if no other condition exists and the 91 or 92 operation code had a plus sign. The order of precedence of the disk-priority condition codes is as follows:

Precedence	Condition Code	Name
1st	7	Incomplete seek
2d	6	Addressing error
3d	5	Machine address check
4th*	3	SLR
	4	LLR
5th	1	Error
6th	2	CLR

*SLR and LLR cannot both occur, of course, in the same operation.

Seek Priority

A seek operation automatically initiates a priority signal when completed, but is conditioned to do so only if a channel control is available (either one). If the signal is caused by a seek error, the cause can be made known when a read or write operation is to be performed, and the condition code in the final status word is tested by the program.

At the time that a priority routine is initiated by the completion of a seek, index word 99 contains the address of the final status word for the appropriate file and arm, even though the final status word itself is not affected by the seek operation. Note that the file number (0-3) is in position 5 and the arm number (0-2) is in position 4 of the final status word address. The other six positions of index word 99 are set to zero, and the sign is set to plus.

Although final status words are not formed by seek operations, digit position 0 in these words is used to determine the exact address to which the program will branch in starting the priority routine, just as with read and write operations. A seek priority routine starts in one of the ten locations 0290-0299, determined by this digit.

Priority Codes

These instructions are used specifically with the Automatic Priority Processing feature of the 7070. (Other operation codes, such as +81, +91, etc. instruct an operation to signal priority at completion, but they do not affect priority directly.)

These codes are presented in the sequence shown in Figure 247.

If the first letter in an *Autocoder* symbol is P, it stands for priority. These instructions are PC and PR as shown above; all of the tape and disk read-write instructions that are to cause priority at completion; and the disk seek instruction, which automatically causes priority at completion.

CATEGORIES	OP CODES	NAMES	MNEMONICS
Priority mask control	+55	Priority Control	PC
Stacking latches	+60	Stacking latch test:	
		Any stacking latch	BAL
		Unit-record	BUL
		Inquiry controls	BQL
		Tape channels	BTL
		Disk access arms	BDL
	−60	Stacking latch set:	
		Unit-record	ULN
		Inquiry controls	QLN
		Tape channels	TLN
		Disk access arms	DLN
	−62	Stacking latch reset:	
		Unit record	ULF
		Inquiry controls	QLF
		Tape channels	TLF
		Disk access arms	DLF
Release	+64	Priority release	PR

FIGURE 247. PRIORITY OPERATION CODES

Priority Control

+55 **PC**

MACHINE DESCRIPTION: This operation determines which stacking latches will be masked and which will not. The mask is contained in the word addressed by positions 6-9 (indexable). This word determines which stacking latches can stop the sequencing scanner and thus initiate a priority routine. A zero in a specific position of this word allows priority, and a 1 masks priority, for the stacking latches as follows:

DIGIT POSITION	STACKING LATCHES
Sign	not used
0	Tape channel 4
1	Tape channel 3
2	Inquiry control group 2
3	Inquiry control group 1
4	Disk read and write
5	Disk seek
6	Tape channel 2
7	Tape channel 1
8	Unit record B
9	Unit record A

A mask is effective until another priority-control instruction resets it.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+55
23	Indexing word for modifying the address of the mask word.
45	Not used.
6789	Address of the mask word

EXAMPLES: To set the priority masks according to the ones and zeros in word 1590:

S01	23	45	6789
+55	00	xx	1590

Contents of word 1590: +00111 10011. After this instruction, only the tape units can cause priority signals.

To set the priority masks according to the ones and zeros in word 1500, indexed by positions 2-5 of IW 73:

S01	23	45	6789
+55	73	xx	1500

Contents of the storage word: +00000 00000. After this instruction, all of the stacking latches can cause priority signals.

DATA FLOW: The word addressed by positions 6-9, after indexing if indexing is specified, is brought from storage to a special register, called the *priority mask register*. The digit 1's turn on the corresponding masks, and 0's turn off the corresponding masks as described above.

REGISTERS AFFECTED: None, except the priority mask register.

TIMING: 36 microseconds.

COMMENTS: If there is a digit other than zero or one in positions 2-9 of the mask word, an error is signalled and the machine stops. The sign of the word is ignored.

A mask does not prevent a stacking latch from being set; nothing can prevent that. The mask prevents the stacking latch from setting the priority waiting latch, and thus sending a priority signal and initiating a priority routine. If a stacking latch is masked and is set, and later a PC instruction is given which unmasks that stacking latch, the sequencing scanner will stop at the stacking latch and signal priority.

Note that there are 10 masks for the 56 stacking latches. The inquiry-control and the unit-record masks, in digit positions 2-3 and 8-9 of the mask word respectively, each control one stacking latch. Each tape-channel mask controls the 10 stacking

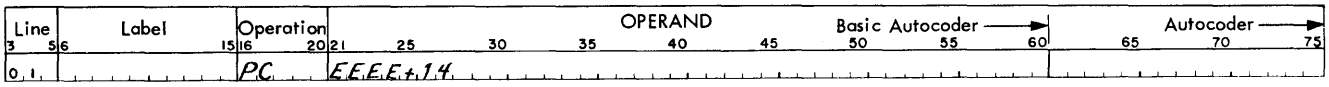


FIGURE 248.

latches for the 10 tape units on that channel. The disk read-write mask covers all 12 disk-storage stacking latches, one for each access arm, and the disk-seek mask covers those same 12 stacking latches for seek operations. This means that *all* disk read and write operations are either masked or not, and *all* disk seek operations are either masked or not.

The masks have no effect on the operations to test, set, or reset the stacking latches.

AUTOCODER EXAMPLE (FIGURE 248): Assume that EEEE has been previously defined as word 2300. The assembled instruction is:

S01	23	45	6789
+55	00	00	2314

Stacking Latch Test
+60

MACHINE DESCRIPTION: This augmented instruction tests a specified stacking latch and branches to the address in positions 6-9 (indexable) if it is on. The stacking latch is designated in positions 4-5, and its setting is not changed by the operation.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +60

23 Indexing word for modifying the branch address

45 Designates the particular stacking latch to be tested:

00	Any stacking latch	BAL
01, 02	Unit-record Latch A, B	BUL
03, 04	Inquiry Controls 1, 2	BQL
10-19	Tape Units 0-9 on Channel 1	BTL
20-29	Tape Units 0-9 on Channel 2	BTL
30-39	Tape Units 0-9 on Channel 3	BTL
40-49	Tape Units 0-9 on Channel 4	BTL
50-53	Access Arm 0, Disk Storage Unit 0-3	BDL
60-63	Access Arm 1, Disk Storage Unit 0-3	BDL
70-73	Access Arm 2, Disk Storage Unit 0-3	BDL

6789 Branch address if the specified stacking latch is ON.

EXAMPLES: To branch to location 3200 indexed by positions 2-5 of IW 63, if any of the 56 stacking latches is ON:

S01	23	45	6789
+60	63	00	3200

To branch to location 2111, if the stacking latch for tape unit 0 on channel 2 is ON:

S01	23	45	6789
+60	00	20	2111

DATA FLOW: The only movement of data involved is the transmission of the address in positions 6-9, indexed if so specified, to the instruction counter, if the specified stacking latch is ON.

REGISTERS AFFECTED: Possibly the instruction counter.

TIMING: 36 microseconds

COMMENTS: This instruction can be used as a means of branching to a priority routine at a point determined by the main program, instead of when the sequencing scanner detects the stacking latch.

Note that the setting of the stacking latch is not changed by this operation.

AUTOCODER EXAMPLES (FIGURES 249-253): Actual address 1340 is used. The instruction assembled from Figure 249 is:

S01	23	45	6789
+60	00	00	1340

Assume that FFFF has been defined as word 2650. The instruction assembled from Figure 250 is:

S01	23	45	6789
+60	00	01	2662

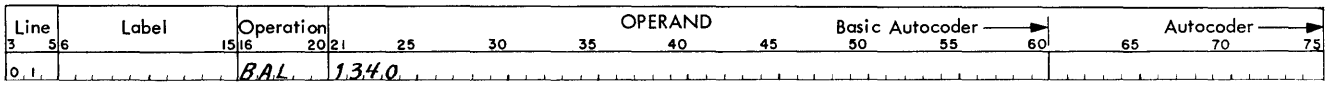


FIGURE 249.

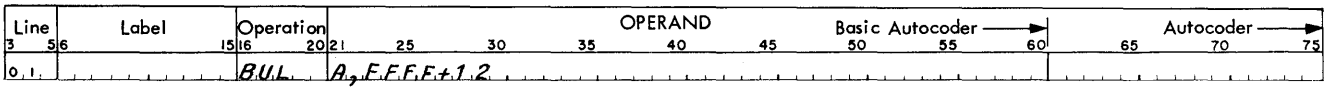


FIGURE 250.

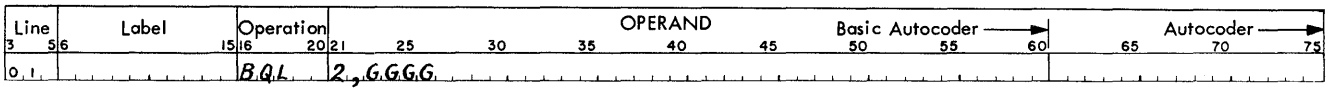


FIGURE 251.

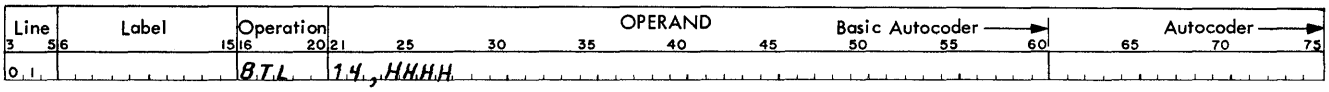


FIGURE 252.

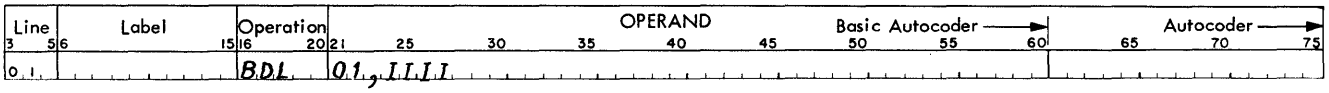


FIGURE 253.

Assume that GGGG has been defined as word 2946. The instruction assembled from Figure 251 is:

S01	23	45	6789
+60	00	04	2946

Assume that HHHH has been defined as word 1505. The instruction assembled from Figure 252 is:

S01	23	45	6789
+60	00	14	1505

Assume that IIII has been defined as location 4154. The instruction assembled from Figure 253 is:

S01	23	45	6789
+60	00	51	4154

Stacking Latch Set
−61

MACHINE DESCRIPTION: This augmented code turns ON a specified stacking latch. Positions 4-5 designate the latch to be set, in the same manner as the stacking-latch test command designates the latch to be tested.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −61

23 Indexing word. Even though positions 6-9 are not used, they are modified by positions 2-5 of an indexing word, if one is specified.

45 Designates the particular stacking latch to be turned on:

01, 02	Unit-record latch A, B	ULN
03, 04	Inquiry controls 1, 2	QLN
10-19	Tape units 0-9, ch. ctrl. 1	TLN
20-29	Tape units 0-9, ch. ctrl. 2	TLN
30-39	Tape units 0-9, ch. ctrl. 3	TLN
40-49	Tape units 0-9, ch. ctrl. 4	TLN
50-53	Access arm 0, disk storage unit 0-3	DLN
60-63	Access arm 1, disk storage unit 0-3	DLN
70-73	Access arm 2, disk storage unit 0-3	DLN

6789 Not used

EXAMPLE: To turn on the stacking latch for inquiry control group 1:

S01	23	45	6789
−61	00	03	xxxx

DATA FLOW: None

REGISTERS AFFECTED: Only the specified stacking latch, if it had been OFF.

TIMING: 36 microseconds.

COMMENTS: If the specified stacking latch is not masked, priority will be initiated when it is detected by the sequencing scanner. If a stacking latch is set, it causes a branch on a stacking-latch test instruction, whether it is masked or not.

AUTOCODER EXAMPLE (FIGURE 254): The assembled instruction is:

S01	23	45	6789
−61	00	70	0000

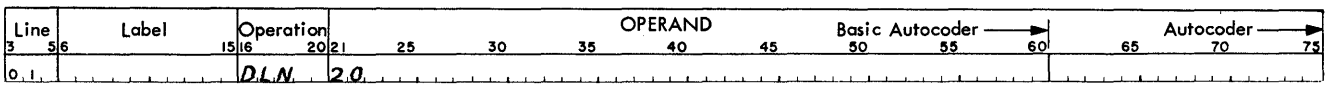


FIGURE 254.

Stacking Latch Reset

−62

MACHINE DESCRIPTION: This code is the same as stacking-latch set, except that the stacking latch specified in positions 4-5 is turned off.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −62

23 Indexing word. Even though positions 6-9 are not used, they are modified by positions 2-5 of an indexing word, if one is specified.

45 Designates the particular stacking latch to be turned off:

01, 02	Unit-record latch A, B	ULF
03, 04	Inquiry controls 1, 2	QLF
10-19	Tape units 0-9, ch. ctrl. 1	TLF
20-29	Tape units 0-9, ch. ctrl. 2	TLF
30-39	Tape units 0-9, ch. ctrl. 3	TLF
40-49	Tape units 0-9, ch. ctrl. 4	TLF
50-53	Access arm 0, disk-storage unit 0-3	DLF
60-63	Access arm 1, disk-storage unit 0-3	DLF
70-73	Access arm 2, disk-storage unit 0-3	DLF

6789 Not used.

EXAMPLE: To reset the stacking latch for tape unit 5 on channel 2:

S01	23	45	6789
−62	00	25	xxxx

DATA FLOW: None.

REGISTERS AFFECTED: Only the specified stacking latch if it was on.

TIMING: 36 microseconds.

COMMENTS: The only difference between the *Autocoder* symbols of this code and stacking-latch set is the letter F instead of N.

Regardless of whether the stacking latch is masked, this instruction is effective.

AUTOCODER EXAMPLE (FIGURE 255): The assembled instruction is:

S01	23	45	6789
−62	00	02	0000

Priority Release

+64 **PR**

MACHINE DESCRIPTION: This code is usually the last instruction in a priority routine. It releases priority, so that the main routine can continue from where it left off when interrupted by the priority signal.

Index word 97, called the *priority word,* contains in positions 2-5 the address of the next instruction to be executed in the main program (it was put there when the priority routine was started).

The location of the next instruction after a priority-release operation is determined by two factors:

1. Is there another priority routine waiting?

2. Is the address of the PRIORITY RELEASE instruction 0097?

There are four courses of action that can be taken, dependent on combinations of the two factors:

1. No priority waiting, address is 0097: The address in positions 2-5 of index word 97 is sent to the instruction counter, and the next instruction is taken from that address.

2. No priority waiting, address is not 0097, 1234 for example: the instruction counter is set to 1234, and the main routine starts with that instruction. Index word 97 is unchanged.

3. Another priority routine waiting, address is 0097. The main routine is not resumed. The waiting priority routine is started instead. Index word 97 is unchanged.

4. Another priority waiting, address is not 0097, 1234 for example: the waiting priority routine is started. The address in positions 2-5 of index word 97 is replaced by the address 1234. This means that the return address to the main routine is lost, unless it was stored elsewhere during the first priority routine.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +64

23 Indexing word. The address in positions 6-9, whether it be positions 2-5 of 0097 or some other address, is modified by positions 2-5 of an IW, if one is specified.

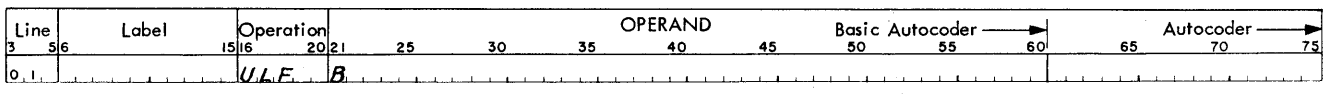


FIGURE 255.

45 Not used.

6789 Contains the location of the next instruction, dependent on whether this address is 0097 or not, and on whether there is another priority routine waiting, as described above.

EXAMPLES: To release priority, and return to the main program at the point where it was interrupted by the priority routine:

S01	23	45	6789
+64	00	xx	0097

If there is no other priority routine waiting, the program takes its next instruction from the address in positions 2-5 of index word 97. If there is another priority routine waiting, it is started, and index word 97 is unchanged.

To release priority, and branch to location 1144:

S01	23	45	6789
+64	00	xx	1144

If there is no other priority routine waiting, the program takes its next instruction from 1144, and index word 97 is unchanged. If there is another priority routine waiting, it is started, and the address 1144 is transmitted to positions 2-5 of index word 97.

DATA FLOW: This is dependent on the situation as outlined under *Machine Description:*

1. No priority routine waiting, address is 0097. The contents of 0097 are brought to the arithmetic register, and positions 2-5 are then moved to the instruction counter.

2. No priority routine waiting, address is not 0097. The contents of positions 6-9 of the program register are moved to the instruction counter.

3. Another priority routine waiting, address is 0097. The waiting priority routine started by bringing its starting address to the instruction counter.

4. Another priority routing waiting, address is not 0097. The contents of 0097 are brought to the arithmetic register. The contents of positions 6-9 of the program register are brought to the instruction counter and to positions 2-5 of the arithmetic register. The contents of the arithmetic register are then moved to index word 97.

REGISTERS AFFECTED: The instruction counter, possibly the arithmetic register, and possibly index word 97.

TIMING: 36 microseconds.

COMMENTS: A distinction is made between an address of 0097 and an address of any other number. If the address is other than 0097, this instruction acts much like a branch code; the next instruction is taken from that location.

If the address, after indexing if so specified, is 0097, the next instruction is not taken from 0097 but from the location in positions 2-5 of 0097.

If a PR instruction is given in non-priority mode, it is considered a no-op; the next sequential instruction is given.

AUTOCODER EXAMPLES (FIGURES 256, 257): The instruction assembled from Figure 256 is:

S01	23	45	6789
+64	00	00	0097

Assume that JJJJ has been defined as word 2272. The instruction assembled from Figure 257 is:

S01	23	45	6789
+64	17	00	2272

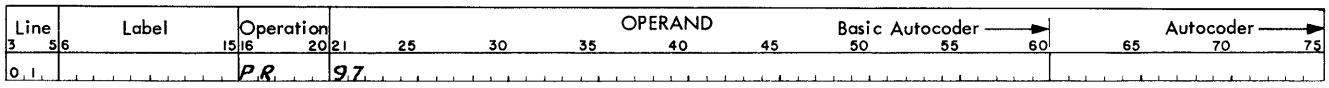


FIGURE 256.

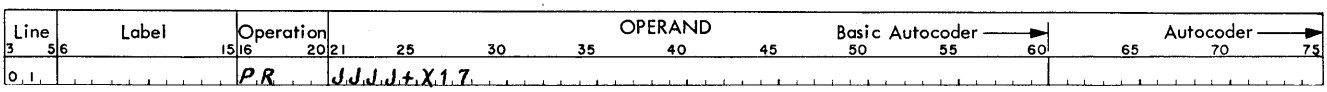


FIGURE 257.

Floating Decimal

The computing capabilities of the IBM 7070 are greatly increased by use of floating-decimal operations, available on an optional basis. In floating-decimal notation, a 10-digit word of data is divided into the *modified characteristic,* in positions 0-1, and the number itself, in positions 2-9. The 8-digit number called the *mantissa,* has a value between +1 and −1. The modified characteristic is 50 plus a power of ten, by which the mantissa would be multiplied to equal its actual value. A modified characteristic of 50 in positions 0-1 equals 0 in actual value (10^0), 51 equals +1, 49 equals −1, etc. The floating-decimal-notation range of modified characteristics 00-99 thus equals actual-value exponents of 10 from −50 through +49.

This notation affords a very wide range in values that can be represented by a 10-digit word. The smallest absolute value (other than zero) of a normalized floating-decimal number is:

Mod. Ch.	Mantissa
00	10000000

This represents a 1 in the 51st position to the right of the decimal. The largest value of a floating-decimal number is:

Mod. Ch.	Mantissa
99	99999999

This equals a whole number of 99,999,999 followed by 41 zeros. A number in floating-decimal notation may be plus or minus. The sign of a word is the sign of the mantissa.

Figure 258 shows some examples of numbers in actual notation (left column) and 7070 floating-decimal notation (right column). The other columns show how the floating-decimal values were obtained from the actual values.

All arithmetic operations can be performed by the floating-decimal codes. In all cases except the absolute-value codes, the signs of the factors are considered in the operations.

All numbers referred to in descriptions of floating-decimal codes are floating-decimal values; that is, each 10-digit word consists of an 8-digit mantissa and a 2-digit modified characteristic. There is no field definition with these codes; positions 4 and 5 are not used and can contain any digits.

Accumulators 1 and 2 are coupled for some floating-decimal operations. In these operations, the sign of accumulator 2 is considered the same as that of accumulator 1, and the modified characteristic of accumulator 2 is considered to be 8 less than that of accumulator 1. The actual sign and modified characteristic of accumulator 2 are ignored. Because each accumulator retains its characteristic, the mantissa is 16 digits in size. The coupled accumulators can be used for addition and subtraction and contain the multiplication product and dividend in division. In multiplication, two 8-digit mantissas can be multiplied to give a 16-digit product. In division, an 8-digit divisor can be divided into a 16-digit dividend to obtain an 8-digit quotient and an 8-digit remainder. This is called *modified double-precision floating-decimal arithmetic.*

One starting factor must always be placed in accumulator 1 on a previous program step (except for ZERO AND ADD) and the result of an operation is automatically placed in accumulator 1. If so specified, the result is extended into accumulator 2.

ABSOLUTE VALUE	SCIENTIFIC NOTATION	MANTISSA	EXPONENT OF 10	MODIFIED CHARACTERISTIC	FLOATING-DECIMAL NOTATION
+46,230	$+.4623 \times 10^{+5}$	+.46230000	+5 + 50	55	+5546230000
+3.1416	$+.31416 \times 10^{+1}$	+.31416000	+1 + 50	51	+5131416000
+0.5	$+.5 \times 10^{0}$	+.50000000	0 + 50	50	+5050000000
+0.00006	$+.6 \times 10^{-4}$	+.60000000	−4 + 50	46	+4660000000
−1234.0	$-.1234 \times 10^{4}$	−.12340000	+ 4 + 50	54	−5412340000
−0.4239	$-.4239 \times 10^{0}$	−.42390000	0 + 50	50	−5042390000
−0.0005862	$-.5862 \times 10^{-3}$	−.58620000	−3 + 50	47	−4758620000

FIGURE 258. FLOATING-DECIMAL EXAMPLES

In floating-decimal arithmetic, alpha signs cannot be used. If either factor is alpha, an error stop results.

Whenever a number is obtained as the result of a floating-decimal operation, it is automatically *normalized.* All high-order zeros are eliminated by shifting left, and the modified characteristic is decreased by one for each position shifted. In these discussions of floating-decimal operation codes, all factors are assumed to be normalized.

When an operation tries to develop a characteristic greater than 99, the *floating-decimal overflow* indicator is turned on, and the machine either stops or continues, depending on the setting of the floating-decimal overflow stop/sense switch on the console. The term *floating-decimal underflow* is used to define an attempt to develop a modified characteristic less than 00 (minus 50 in actual value). The *floating-decimal underflow and overflow* indicators can be interrogated by the program. Only the *overflow* indicator can cause a program stop.

A mantissa of zero carries a modified characteristic of 00. Accumulators 1 and 2 are set to zero if: (1) in double precision results, the mantissa of accumulator 1 is zero or (2) if a zero quotient is developed in division. A zero mantissa in accumulator 2 is retained with its exponent if it contains the low-order mantissa digits of a double-precision result in which some higher-order mantissa digit is non-zero. If the modified characteristic of accumulator 1 is greater than 00, but that of accumulator 2 is less than 00, an underflow condition is not indicated but accumulator 2 is set to zero.

CATEGORIES	OP CODES	NAMES	MNEMONICS
Add-subtract	+75	Floating zero and add	FZA
	+74	Floating add	FA
	−74	Floating subtract	FS
	+76	Floating add double precision	FAD
	−76	Floating add double precision and suppress normalization	FADS
	+77	Floating add absolute	FAA
	−77	Floating subtract absolute	FSA
	+71	Floating round	FR
Multiply-divide	+73	Floating multiply	FM
	−73	Floating divide	FD
	−75	Floating divide double precision	FDD
Branch	+70	Floating branch overflow	FBV
	−70	Floating branch underflow	FBU

FIGURE 259. FLOATING DECIMAL OPERATION CODES

The floating-decimal codes are presented in Figure 259.

The *Autocoder* mnemonics for all of the floating-decimal codes, and only those codes, start with F.

Floating Zero and Add

+75 **FZA**

MACHINE DESCRIPTION: The contents of the storage word addressed by positions 6-9 (indexable) is brought to accumulator 1, replacing its previous contents. The number is then normalized, if necessary.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +75

23 Indexing word, for the address in positions 6-9

45 Not used.

6789 Address of the floating-decimal number to be brought to accumulator 1, and normalized if necessary.

EXAMPLES: To bring the contents of word 2662 to accumulator 1, and normalize it if necessary:

S01	23	45	6789
+75	00	xx	2662

Contents of word 2662: −24 00000001. Contents of accumulator 1, after the operation −17 10000000, regardless of previous contents.

To bring the contents of word 622 to accumulator 1, and normalize it if necessary:

S01	23	45	6789
+75	00	xx	0622

Contents of word 62: +80 21426384. Contents of accumulator 1, after the operation: +80 21426384, regardless of previous contents. The numbers are the same because no normalizing was required.

DATA FLOW: The contents of the storage word are brought to the arithmetic register and thence to accumulator 1. Accumulator 2 is set to zeros. The coupled accumulators are shifted left two positions, to eliminate the modified characteristic. Then a shift-and-count operation takes place. The total number of positions shifted is subtracted from the modified characteristic in the arithmetic register. A shift right of two positions then allows the new modified characteristic to be inserted in positions 0-1 of accumulator 1.

REGISTERS AFFECTED: The arithmetic register, accumulator 1, and accumulator 2.

TIMING: 108-156 microseconds, depending on the number of positions to be normalized.

COMMENTS: A storage word can be normalized by using this instruction and then storing accumulator 1 into the same location. A STORE ACCUMULATOR 1 instruction (ST1) is used. If it is desired to bring a word to accumulator 1 but not normalize it, use ZERO ACCUMULATOR 1 AND ADD. If the high-order digit of the mantissa is non-zero, this instruction has the same result as ZA1.

Whenever a floating-decimal number is normalized, its modified characteristic is reduced by one, for each high-order zero in the original mantissa. For example, the number 27 00750121 is normalized to 25 75012100. This means that the *floating-decimal underflow* indicator can be turned on by any operation, such as FZA, that normalizes a floating-decimal value.

AUTOCODER EXAMPLE (FIGURE 260): Assume that OHIO has been defined as word 1735. The assembled instruction is:

S01	23	45	6789
+75	00	00	1735

Floating Add

+74 **FA**

MACHINE DESCRIPTION: (8-digit number) + (8-digit number) = (16-digit result).

The contents of the location addressed by positions 6-9 (indexable) are added to the amount already in accumulator 1, and the result is normalized and developed in accumulators 1 and 2. Accumulator 2 is reset before the operation, and there is a difference of 8 in the resultant modified characteristics of the two accumulators.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+74
23	Indexing word for modifying the address in positions 6-9.
45	Not used
6789	Address of the floating-decimal number to be added to the number in accumulator 1.

EXAMPLES: To add, in floating-decimal notation, the contents of word 2359 to the amount already in accumulator 1:

S01	23	45	6789
+74	00	xx	2359

Contents of accumulator 1 before the operation: +57 20415855. Contents of word 2359: +51 72885343. Contents of accumulators 1 and 2 after operation:

Acc. 1	Acc. 2
+57 20415927	+49 88534300

To add, in floating-decimal notation, the contents of word 987 to the amount already in accumulator 1:

S01	23	45	6789
+74	00	xx	0987

Contents of accumulator 1 before the operation: −33 21467522. Contents of word 987: +36 23013333. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+36 22991865	+28 47800000

To add, in floating-decimal notation, the contents of word 1164 to the amount already in accumulator 1:

S01	23	45	6789
+74	00	xx	1164

Contents of accumulator 1 before the operation: −69 67714400. Contents of word 1164: −69 84002217. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−70 15171661	−62 70000000

DATA FLOW: A complete detailed description of all the operations involved with this code is too lengthy and intricate to be included here. A general description of the functions will be given, using the first example given above:

The smaller factor, as determined by comparing the two modified characteristics, is brought to the arithmetic register. In the example, arithmetic register: +51 72885343. The larger is brought to the auxiliary register: +57 20415855. The difference in modified characteristics (6 in the example), sets up field definition in the program

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60		65 70 75
0 1		FZA	OHIO		

FIGURE 260.

register to control a series of additions, as follows:
1. Positions 4-9 of the arithmetic register are added to the zeros in positions 0-5, accumulator 2. Accumulator 2: +885343 0000.
2. Positions 2-3 of the arithmetic register are added to the contents of the auxiliary register, and the result is stored in accumulator 1: +00 20415927. The modified characteristic from the auxiliary register is ignored.

The number of high-order zeros in accumulator 1 must now be determined. A true-add operation may produce a carry 1 in the second high-order position (the last example shown above, the one with the two minus factors, produces this carry). A complement-add operation, performed when the factors have different signs, may produce any number of high-order zeros. The original exponent of the greater factor must be changed if there are other than two high-order zeros in the result; increased by one if there is one high-order zero, or decreased by one for each high-order zero beyond the first two.

This test is performed by a shift-left-and-count operation, using both accumulators coupled. Accumulators 1 and 2 in the example are shifted, and thus contain: + 20415 92788 +53430 00000. The accumulators are then shifted two positions to the right, with the modified characteristic of the result inserted in the two high-order positions. In the example, the accumulators thus contain: +57 20415927 +88534 30000. Accumulator 2 is shifted right 2 positions, and a modified characteristic of 8 less than that of accumulator 1 is inserted in the two high-order positions: +49 88534300.

The example used in this explanation had a difference of 6 in the modified characteristics of the two original factors. This procedure is used in all cases in which there is a difference of 1-7 in the two modified characteristics. If there is a difference of 8-15 in the modified characteristics, no accumulation takes place in a true-add operation.

If a complement-add is called for by a difference in signs, the entire operation takes place as described above, because there is no way of knowing how many digits of the greater factor might be affected by the subtraction.

If the difference in the two original modified characteristics is greater than 16, the greater factor, unchanged, is considered the result, for either true-add or complement-add.

If the original modified characteristics are the same, the accumulation takes place in accumulator 1 only. If a carry 1 is produced, the units digit of the result is shifted to the high-order mantissa position of accumulator 2, as in the last example above.

REGISTERS AFFECTED: All 3 accumulators are used, along with the arithmetic register, the auxiliary register, and the program register. Positions 6-9 of the program register are used to contain the difference in the values of the two modified characteristics. Positions 4-5 are used to control the digits from the arithmetic register during the add operation, as determined by the difference in modified characteristics.

TIMING: 204-288 microseconds. The variance of as many as 84 microseconds is due to the shifting caused by the difference in modified characteristics, and to normalizing the result.

COMMENTS: This instruction, as all floating-decimal codes, uses accumulator 1 as the *base* accumulator; the more important part of the result, the high-order portion of the 16-digit mantissa, is in accumulator 1.

If the difference in modified characteristics is greater than 15, the result is as if no operation had taken place. The contents of the larger factor is in accumulator 1 at completion, and accumulator 2 is set to plus zeroes. This is not correct arithmetically in the case of complement-add (the mantissa in accumulator 2 should be all 9's, and the mantissa in accumulator 1 should be decreased by one). However, since no part of the smaller factor gets to accumulator 2, true-add and complement-add are made to produce the same result.

At the conclusion of the operation, positions 6-9 of the program register contain the modified characteristic of accumulator 2 (00xx), and positions 4-5 contain the digits 01, from the insertion of that modified characteristic.

Note that accumulator 3 is used in this operation. Note too, that the contents of accumulator 2 have no effect on the result; it is reset prior to the operation.

The floating-decimal *overflow* and *underflow* indicators are set by the modified characteristic of the result in accumulator 1. If the operation attempts to develop a modified characteristic greater than 99 in accumulator 1, the *overflow* indicator is turned on and can be tested later by a floating-branch-overflow instruction. The modified characteristic in positions 0-1 of accumulator 1 is the units and tens digits of the resultant modified characteristic. (This is also true of the modified characteristic in accumulator 2, if it also exceeds 99.)

If the operation attempts to develop a modified characteristic less than 00, the *underflow* indicator is turned on. Both accumulators are reset to zero. If the modified characteristic of accumulator 1 is greater than 00, but that of accumulator 2 is less than 00, an underflow condition is not indicated but accumulator 2 is set to zero.

AUTOCODER EXAMPLE (FIGURE 261): Assume that TEXAS has been defined as word 2580. The assembled instruction is:

S01	23	45	6789
+74	42	00	2580

Floating Subtract

−74 **FS**

MACHINE DESCRIPTION: (8-digit number) − (8-digit number) = (16-digit result).

The contents of the location addressed by positions 6-9 (indexable) are subtracted from the amount already in accumulator 1, and the result is normalized and developed in accumulators 1 and 2. Accumulator 2 is reset before the operation, and there is a difference of 8 in the resultant modified characteristics in the accumulators.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −74

23 Indexing word for modifying the address in positions 6-9.

45 Not used

6789 Address of the floating-decimal number to be subtracted from the number in accumulator 1.

EXAMPLES: To subtract, in floating-decimal notation, the contents of word 4714 from the amount already in accumulator 1:

S01	23	45	6789
−74	00	xx	4714

Contents of accumulator 1 before the operation: +91 10214889. Contents of word 4714: +88 7145 0024. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+91 10143438	+83 97600000

To subtract, in floating-decimal notation, the contents of word 1300 indexed by positions 2-5 of IW 48, from the amount already in accumulator 1:

S01	23	45	6789
−74	48	xx	1300

Contents of accumulator 1 before the operation: +75 44162810. Contents of the data word: −75 79100026. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+76 12326283	+68 60000000

To subtract, in floating-decimal notation, the contents of word 1907 from the amount already in accumulator 1:

S01	23	45	6789
−74	00	xx	1907

Contents of accumulator 1 before the operation: −32 28184000. Contents of word 1907: −39 10120000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+39 10119997	+31 18160000

DATA FLOW: Same as floating-add, except that true-add takes place if the signs are different, and complement-add takes place if the signs are the same. In all other respects, they are the same.

REGISTERS AFFECTED: Same as floating-add.

TIMING: Same as floating-add.

COMMENTS: The rules of algebra apply to floating add and subtract, just as for fixed-decimal add and subtract. Alpha factors cannot be used, however, with the floating-decimal operations

AUTOCODER EXAMPLE (FIGURE 262): Assume that IDAHO has been defined as word 785. The assembled instruction is:

S01	23	45	6789
−74	00	00	0785

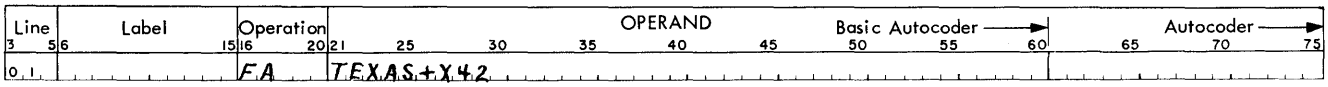


FIGURE 261.

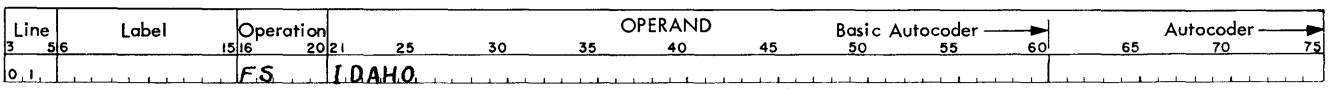


FIGURE 262.

Floating Add Double Precision

+76 **FAD**

MACHINE DESCRIPTION: (16-digit number) + (8-digit number) = (16-digit result).

This code is the same as FLOATING ADD except that accumulator 2 is not set to zero before the operation starts. The program, of course, must have placed the values in the accumulators prior to this instruction.

The modified characteristic of the number from storage must be equal to, or greater than, the modified characteristic of the number in accumulator 1. The modified characteristic in accumulator 2 should, of course, be 8 less than that of accumulator 1.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+76
23	Indexing word for the address in positions 6-9
45	Not used
6789	Address of the floating-decimal number to be added to the number in accumulators 1 and 2 coupled.

EXAMPLES: To add, in floating-decimal notation, the contents of word 645 to the amount already in accumulators 1 and 2:

S01	23	45	6789
+76	00	xx	0645

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
+51 23000740	+43 02232306

Contents of word 645: +55 71774321. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+55 71776621	+47 07400223

To add, in floating-decimal notation, the contents of word 1707 to the amount already in accumulators 1 and 2:

S01	23	45	6789
+76	00	xx	1707

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
+21 76714350	+13 14414388

Contents of word 1707: −22 11220000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−21 35485649	−13 85585620

Note that the accumulator signs are changed.

DATA FLOW: Same as floating add, except:

1. The comparison of the modified characteristic of the storage word with that of accumulator 1 is not for the purpose of placing the larger value in the auxiliary register and the smaller value in the arithmetic register; such re-arrangement is not possible. The modified characteristics are compared to assure that the storage value is not smaller; if it is, an error is signalled and the machine stops.

2. Accumulator 2 is not set to zeros; its mantissa is a factor in the operation.

In all other respects, the operations are the same as FLOATING ADD.

REGISTERS AFFECTED: Same as FLOATING ADD.

TIMING: Same as FLOATING ADD.

COMMENTS: The modified characteristic of the word from storage must not be smaller than that of accumulator 1. It is advisable to compare the two factors that are to be added by a FAD instruction, so as to put the smaller one into the accumulators first if they are unequal.

The modified characteristic of accumulator 2 before the operation is ignored, as is the sign. Its sign is considered to be that of accumulator 1. The mantissa in accumulator 1 before the operation should be normalized, but the mantissa in accumulator 2 needn't be. Note that if the modified characteristics are different, some of the low-order digits in accumulator 2 are lost to the operation. In the first example shown above, the 4 low-order digits of accumulator 2, 2306, are lost. The number of digits lost is determined by the difference in modified-characteristic values of the storage word and accumulator 1.

AUTOCODER EXAMPLE (FIGURE 263): Assume that UTAH has been defined as word 1700. The assembled instruction is:

S01	23	45	6789
+76	00	00	1742

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
01		FAD	UTAH+42		

FIGURE 263.

Floating Add Double Precision and Suppress Normalization

−76 **FADS**

MACHINE DESCRIPTION: (16-digit number) + (8-digit number) = (16-digit-or-less result).

This code is the same as FLOATING ADD DOUBLE PRECISION except that the result in accumulators 1 and 2 is not normalized after the operation is completed.

INSTRUCTION FORMAT: Same as FLOATING ADD DOUBLE PRECISION, except for the sign.

EXAMPLE: To add, in floating-decimal notation, the contents of word 4582 to the amount already in accumulators 1 and 2, but not normalize the result:

S01	23	45	6789
−76	00	xx	4582

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
+37 95791886	+29 00507500

Contents of word 4582: −38 12720011. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−38 03140822	−30 39949250

DATA FLOW: Same as FLOATING ADD DOUBLE PRECISION, except that the result is not normalized. The shift-left-and-count operation is used only to test for a carry 1 and not for high-order zeros beyond the second position.

REGISTERS AFFECTED: Same as FLOATING ADD.

TIMING: This instruction takes less time than FLOATING ADD, due to the fact that the result is not normalized. Its duration is 168-252 microseconds.

COMMENTS: The modified characteristic of the result (in accumulator 1) can never be less than that of the value from storage. If there is a carry 1, it is greater than the storage-word modified characteristic by one; otherwise, it is the same.

The main purpose of this instruction is to align decimal positions of several numerical values, by using this code with a storage value consisting of a modified characteristic based on the desired decimal location, and a mantissa of eight zeros. Take, for example, the two values +53 12345012 and +51 78754614. If brought to accumulator 1, and if accumulator 2 is set to zero, these values look like this:

Acc. 1	Acc. 2
+53 12345012	+00 00000000
+51 78754614	+00 00000000

Perform a FADS operation on each, using a storage value of +55 00000000, and the results are:

+55 00123450	+47 12000000
and +55 00007875	+47 46140000

The two mantissas are now decimally aligned. They can be stored and printed as:

1234501200
78754614

The accumulators are not set to zero if the mantissa in accumulator 1 is set to zero by this operation (using complement-add).

AUTOCODER EXAMPLE (FIGURE 264): The actual address 1114 is used. The assembled instruction is:

S01	23	45	6789
−76	00	00	1114

Floating Add Absolute

+77 **FAA**

MACHINE DESCRIPTION: (8-digit number) + (8-digit number, considered plus) = (16-digit result).

This instruction is the same as FLOATING ADD, except that the sign of the storage word is considered plus. Thus, the sign of accumulator 1 determines whether the operation is true-add or complement-add.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+77
23	Indexing word, for the address in positions 6-9.
45	Not used
6789	Address of the floating-decimal number the contents of which, considered plus, are to be added to the number in accumulator 1.

EXAMPLES: To add, in floating-decimal notation, the absolute value of the contents of word 1585 to the amount already in accumulator 1 and develop the result in accumulators 1 and 2:

S01	23	45	6789
+77	00	xx	1585

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45	50 55 60	65 70 75
0 1		FADS	1114		

FIGURE 264.

Contents of accumulator 1 before the operation: +25 74026849. Contents of word 1585: −26 54000600. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+26 61403284	+18 90000000

To add, in floating-decimal notation, the absolute value of the contents of word 4830 to the amount already in accumulator 1 and develop the result in accumulators 1 and 2:

S01	23	45	6789
+77	00	xx	4830

Contents of accumulator 1 before the operation: −43 56490066. Contents of word 4830: −47 29955000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+47 29949350	+39 99340000

DATA FLOW: Same as FLOATING ADD, except that the number from storage is considered plus in determining whether true-add or complement-add will be performed.

REGISTERS AFFECTED: Same as FLOATING ADD.

TIMING: Same as FLOATING ADD.

COMMENTS: This code has the same relation to FLOATING ADD as ADD ABSOLUTE TO ACCUMULATOR 1 has to the ADD TO ACCUMULATOR 1 instruction. Since one factor is considered plus and the operation is add, a plus sign in accumulator 1 causes a true-add operation, and a minus sign causes a complement-add operation.

The comments that apply to FLOATING ADD also apply to this code.

AUTOCODER EXAMPLE (FIGURE 265): Assume that MAINE has been defined as word 1650. The assembled instruction is:

S01	23	45	6789
+77	00	00	1672

Floating Subtract Absolute

−77 **FSA**

MACHINE DESCRIPTION: (8-digit number) − (8-digit number, considered plus) = (16-digit result).

This instruction is the same as FLOATING ADD ABSOLUTE except that the value from storage is always subtracted.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −77

23 Indexing word for modifying the address in positions 6-9.

45 Not used.

6789 Address of the floating-decimal number the contents of which, considered plus, are to be subtracted from the number in accumulator 1.

EXAMPLE: To subtract, in floating-decimal notation, the absolute value of the contents of word 841 from the amount already in accumulator 1 and develop the result in accumulators 1 and 2:

S01	23	45	6789
−77	00	xx	0841

Contents of accumulator 1 before the operation: +22 52507225. Contents of word 841: +24 70502575. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−24 69977502	−16 75000000

DATA FLOW: Same as FLOATING ADD, except that true-add is performed if the original sign of the accumulator is minus, and complement-add is performed if it is plus.

REGISTERS AFFECTED: Same as FLOATING ADD.

TIMING: Same as FLOATING ADD.

AUTOCODER EXAMPLE (FIGURE 266): Actual address 747 is used. The assembled instruction is:

S01	23	45	6789
−77	52	00	0747

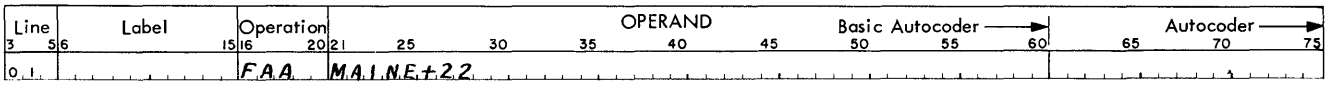


FIGURE 265.

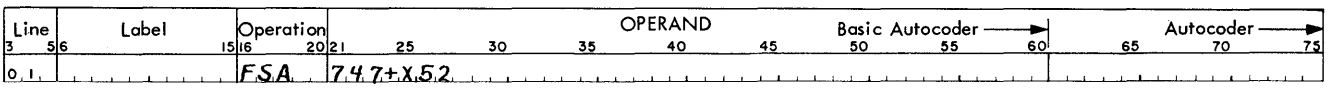


FIGURE 266.

Floating Round

+71 **FR**

MACHINE DESCRIPTION: The mantissa in accumulator 1 is rounded off, by adding a 5 to the high-order position of the mantissa in accumulator 2, and carrying the result into accumulator 1 if that position contained a 5 or higher. If it carries all through accumulator 1 (if accumulator 1 stood at all 9's), the mantissa of accumulator 1 becomes 10000000, and its modified characteristic is increased by one. Accumulator 2 is set to zero and made plus. The operation is always true-add; the signs of the accumulators are ignored.

INSTRUCTION FORMAT: S01 23456789

S01	+71
23456789	Not used.

EXAMPLE: To round off the floating-decimal value in accumulators 1 and 2:

S01	23456789
+71	xxxxxxxx

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
+33 01234567	+25 82124719

after the operation:

+33 01234568	+00 00000000

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
−91 75502300	−83 49990000

after the operation:

−91 75502300	+00 00000000

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
−54 99999999	−46 50000000

after the operation:

−55 10000000	−00 00000000

DATA FLOW: The contents of accumulator 1 are brought to the auxiliary register. The contents of accumulator 2 are brought to the arithmetic register. A 5 is added to the high-order mantissa digit from the arithmetic register; the result of this addition is ignored. The digits from the auxiliary register are brought to the adder, in low-order to high-order sequence; the low-order digit picks up a carry from the adder if it had been produced by the original addition of the 5 to the high-order mantissa digit from the arithmetic register. If this addition produces a carry (if the units digit in the auxiliary register had been a 9), the 10's digit from the auxiliary register is increased by 1; etc.

If, as in the last example above, all of the mantissa digits in accumulator 1 are 9's, the carry is continued into the low-order digit of the modified characteristic (in the example, accumulator 1 is now +55 00000000). If the carry gets this far, a 1 is inserted into the high-order digit of the mantissa at completion of the add operation. Concurrent with the add operation, the arithmetic register is set to zeros and given a plus sign.

The contents of the auxiliary register are moved to accumulator 1, and the contents of the arithmetic register are moved to accumulator 2.

REGISTERS AFFECTED: The auxiliary register, the arithmetic register, the adder, accumulator 1, and accumulator 2.

TIMING: 72 microseconds.

COMMENTS: The modified characteristic of accumulator 2 is ignored, as are the sign and all but the high-order mantissa digit. The modified characteristic of accumulator 1 is increased by one in this operation, only if its mantissa was all 9's, and the high-order digit of the mantissa in accumulator 2 was 5 or greater, before the operation.

Although positions 2-9 are not used, positions 2-3 cannot designate an indexing word that is alpha. If it does, an error stop results. In all cases, indexing takes place if positions 2-3 are not 00.

AUTOCODER EXAMPLE (FIGURE 267): Note that no operand is needed. The assembled instruction is:

S01	23456789
+71	00000000

Line	Label	Operation	OPERAND	Basic Autocoder →	Autocoder →
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60	65 70 75	
0 1		FR			

FIGURE 267.

Floating Multiply

+73 **FM**

MACHINE DESCRIPTION: (8-digit multiplier) × (8-digit multiplicand) = (16-digit product).

Prior to this instruction, the multiplier must be in accumulator 1. The location of the multiplicand is specified by positions 6-9 (indexable). At the start of the operation, the multiplicand is placed in accumulator 3. The 16-digit product is developed in accumulators 1 and 2 and is normalized one position if the high-order digit is zero. The product is not affected by what was in accumulator 2 prior to the operation. When the multiply operation is completed, the multiplicand is in accumulator 3.

The modified characteristic of the result is determined by adding the modified characteristics of the two factors and subtracting either 50 or 51. If the product of the two mantissas has 16 significant digits, subtract 50; if 15, subtract 51. (Since both original factors had to be normalized to start with, the product must be either 15 or 16 digits in size.)

The product of the original mantissas becomes the mantissa of the product in accumulators 1 and 2.

INSTRUCTION FORMAT: S01 23 45 6789

S01	+73
23	Indexing word for modifying the address in positions 6-9.
45	Not used.
6789	Address of the multiplicand.

EXAMPLES: To multiply, in floating-decimal notation, the contents of word 2680 by the number in accumulator 1:

S01	23	45	6789
+73	00	xx	2680

Contents of accumulator 1 before the operation: +39 21427685. Contents of word 2680: −45 10000000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−33 21427685	−25 00000000

Contents of accumulator 3 after the operation: −45 10000000.

Using the same instruction, with different factors: Contents of accumulator 1 before the operation: +55 22200000. Contents of word 2680: +55 50000000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+60 11100000	+52 00000000

Contents of accumulator 1 before the operation: +35 22200000. Contents of word 2680: +46 50000000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+31 11100000	+23 00000000

Contents of accumulator 1 before the operation: −24 12340000. Contents of word 2680: +72 20000000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−45 24680000	−37 00000000

DATA FLOW: The function of the FLOATING-MULTIPLY operation is to add the modified characteristics and multiply the mantissas. The multiplier is brought from accumulator 1 to the auxiliary register, and the multiplicand is brought from storage to the arithmetic register. The modified characteristics of the two factors are added (true add), and the result stored in positions 6-9 of the program register. Zeros are inserted in the high-order positions of the arithmetic and auxiliary registers, replacing the original modified characteristics.

The two mantissas can now be multiplied, as in fixed-decimal multiplication. The multiplier in the auxiliary register is moved to accumulator 2, and the multiplicand in the arithmetic register is brought to accumulator 3. The multiply operation is then carried out, using the quadrupler method as described under the multiply code (M, +53). Because each original factor is an 8-digit number, either a 15- or 16-digit result is obtained in accumulators 1 and 2. This means that accumulator 1 contains either 4 or 5 high-order zeros.

Accumulators 1 and 2 coupled are shifted left four positions. Then the operation becomes shift left and count (SLC); the number of positions shifted, 0 or 1, goes to position 5 of the auxiliary register (000 goes to positions 2-4). The sum of the original modified characteristics, previously stored in positions 6-9 of the program register, is decreased by 50 or 51, as determined by the 0 or 1 in position 5 of the auxiliary register. This result is inserted into positions 0-1 of accumulator 1, as the contents of accumulators 1 and 2 coupled are shifted two positions to the right. Then 8 is subtracted from this modified characteristic, and this result is placed into positions 0-1 of accumulator 2, as the contents of accumulator 2 are shifted two positions to the right.

REGISTERS AFFECTED: The auxiliary register, the arithmetic register, the adder, the program register, and all three accumulators.

TIMING: Total number of microseconds = 48 [(number of 1's, 2's, and 4's in the multiplier) + 2 (number of 3's, 5's, 6's and 8's in the multiplier) + 3 (number of 7's and 9's in the multiplier) +4 (no. of zeros in the multiplier) + 4 (no. of zero groups in the multiplier)] +366. If the total is not a multiple of 12, use the next highest multiple of 12. Consider multiplier-digit positions 0-1 to be a zero group.

COMMENTS: Note that if the modified characteristics of both original factors are below 50, the modified characteristic of the result in accumulator 1 will be farther below 50. Similarly, if they are both above 50, the resultant modified characteristic in accumulator 1 will be farther above 50. Observe the first three operations under *Examples:*

	Mod. Ch. of one factor	Mod. Ch. of other factor	Mod. Ch. of result in acc. 1
1st ex.	39 (50−11)	45 (50−5)	33 (50−11−5, −1)
2d ex.	55 (50+5)	55 (50+5)	60 (50+5+5)
3d ex.	35 (50−15)	46 (50−4)	31 (50−15−4)

The differences from 50 are added to each other, and the resultant difference is the sum of these differences. If a product of 15 digits is developed, the modified characteristic of the result is one less, as in the first example.

Now observe the last example:

	Mod. Ch. of one factor	Mod. Ch. of other factor	Mod. Ch. of result in acc. 1
4th example	24	72	45 (50−26+22, −1)

The same principle works here. The resulting difference from 50 of −26 and +22, is −4. Because the product is 15 digits instead of 16, the modified characteristic of the result becomes 45 instead of 46.

Note that the signs of the original factors do not affect the value of the modified characteristic of the result. They affect the sign of the result, however, according to the rules of algebra: plus × plus = plus, plus × minus = minus, and minus × minus = plus (Alpha values cannot be used).

The rules for floating-decimal overflow and underflow are the same as for FLOATING ADD. Note, however, that FLOATING MULTIPLY can more easily cause an overflow or underflow condition. If the sum of the modified characteristics of the two original factors is greater than 150, an overflow will result. Similarly, if their sum is less than 50, an underflow will result.

At completion of the operation, the program register has 01 in positions 4-5, and the modified characteristic of the product in accumulator 2 in positions 6-9 (00xx).

AUTOCODER EXAMPLE (FIGURE 268): Actual address 804 is used. The assembled instruction is:

S01	23	45	6789
+73	00	00	0804

Floating Divide

−73 **FD**

MACHINE DESCRIPTION: (8-digit dividend) ÷ (8-digit divisor) = (8-digit quotient) with (8-digit remainder).

The divisor, located by positions 6-9 (indexable), goes to accumulator 3. Accumulator 2 is reset to zero and given the same sign as accumulator 1, which by a prior instruction contains the dividend. The mantissa portion of the dividend is divided by the mantissa portion of the divisor in the same manner as fixed-point divide.

At the conclusion of the operation, the quotient is in accumulator 1 and the remainder is in accumulator 2, each with its corrected modified characteristic.

In floating-decimal division operations, the absolute value of the mantissa of the divisor does not have to be smaller than the absolute value of the mantissa of the dividend. In all cases, regardless of the relative absolute values of divisor and dividend, a correct 8-digit quotient with correct modified characteristic is developed.

The relative absolute values of the divisor and dividend mantissas do, however, help to determine the modified characteristics of the quotient and remainder. If the dividend mantissa is greater than, or equal to, the divisor mantissa, the modified characteristic of the quotient is equal to the dividend characteristic minus the divisor characteristic, plus 51. The modified characteristic of the remainder is 7 less than the modified characteristic of the dividend.

If the dividend mantissa is smaller than the divisor mantissa, the modified characteristic of the

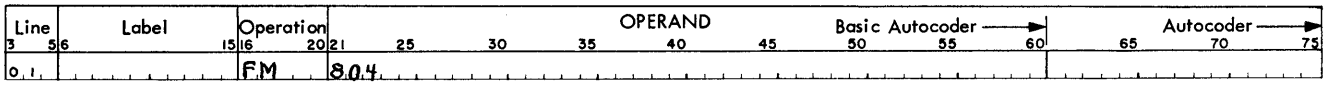


FIGURE 268.

quotient is equal to the dividend characteristic minus the divisor characteristic, plus 50. The modified characteristic of the remainder is 8 less than the modified characteristic of the dividend.

INSTRUCTION FORMAT: S01 23 45 6789

S01	−73
23	Indexing word for modifying the address in positions 6-9.
45	Not used.
6789	Address of the divisor.

EXAMPLES: To divide, in floating-decimal notation, the number in accumulator 1 by the number in word 1565:

S01	23	45	6789
−73	00	xx	1565

Contents of accumulator 1 (dividend) before the operation: +49 70000000. Contents of word 1565 (divisor): +28 30000000. Contents of accumulator 1 (quotient) after the operation: +72 23333333. Contents of accumulator 2 (remainder) after the operation: +42 10000000.

Using the same instruction, with different factors:

Contents of accumulator 1 (dividend) before the operation: −41 22000000. Contents of word 1565 (divisor): −45 70000000. Contents of accumulator 1 (quotient) after the operation: +46 31428571. Contents of accumulator 2 (remainder) after the operation: −33 30000000.

DATA FLOW: The function of this operation is to subtract the modified characteristic of the divisor from that of the dividend, and divide the mantissa of the divisor into that of the dividend. The dividend is brought from accumulator 1 to the arithmetic register, and the divisor is brought to the auxiliary register and to accumulator 3. Accumulator 2 is set to zero.

The contents of the auxiliary register (divisor) and arithmetic register (dividend) are compared. The comparison of the mantissas will be used later to help determine the modified characteristics of quotient and remainder. The difference of the modified characteristics of divisor and dividend, obtained by the compare operation, is stored in positions 6-9 of the program register (this is a complement number if the divisor characteristic is greater).

Positions 0-1 of each factor (the modified characteristics) are set to zeros, and the operation proceeds as in fixed-point divide, with two exceptions:

1. Fixed-point divide indicates error if the dividend from accumulator 1 is greater than the divisor. FLOATING DIVIDE merely adjusts the number of divide cycles if this is the case, and thus determines that the factors 51 and 7 will be used later to obtain the modified characteristics of the quotient and remainder respectively, instead of the factors 50 and 8.

2. Each factor is treated as an 8-digit number, instead of a 10-digit number.

At completion, the quotient is in positions 0-7 of accumulator 2, and the remainder is in positions 0-7 of accumulator 1. Each must now be shifted two positions to the right, have its correct modified characteristic inserted into the two high-order positions, and then be moved to the accumulator location of the other:

The (quotient) contents of accumulator 2 are shifted two positions to the right and placed in the auxiliary register. The difference in the original modified characteristics, stored in positions 6-9 of the program register, is added to either 50 or 51, depending on the comparison of the original mantissas; and this result is inserted into positions 0-1 of the auxiliary register during the shift operation. The (remainder) contents of accumulator 1 are shifted two positions to the right and placed into accumulator 2. The modified characteristic of the dividend, in the arithmetic register, is complement-added to either 7 or 8 depending on the comparison of the original mantissas; and this result is inserted into positions 0-1 of accumulator 2 during the shift operation. Then, the contents of the auxiliary register are moved to accumulator 1.

REGISTERS AFFECTED: The arithmetic and auxiliary register, the adder, the program register, and all three accumulators.

TIMING: Fixed-point divide time, plus 168 microseconds.

Total number of microseconds = 360 + 48 [10 + (Sum of the quotient digits)]. The quotient is 8 digits in size.

COMMENTS: The modified characteristic of the quotient can be determined by the same means described under *Floating Multiply;* keep in mind that the dividend is equal to the divisor times the quotient. Observe the factors given under *Examples:*

Mod. Ch. of dividend	Mod. Ch. of divisor	Mod. Ch. of quotient	Mod. Ch. of remainder
1st example			
49(50−1)	28(50−22)	72(50+21, +1)	42(49−7)
2d example			
41(50−9)	45(50−5)	46(50−4)	33(41−8)

The rules of algebra concerning signs is the same for FLOATING-DIVIDE as for fixed-point divide. If the signs of the dividend and divisor are the same, the quotient is plus; if they are different, the quotient is minus. The remainder always has the sign of the dividend. Alpha factors cannot be used.

The rules for FLOATING-DECIMAL OVERFLOW AND UNDERFLOW in this operation are the same as for FLOATING ADD. A good rule-of-thumb is that the modified characteristics of the dividend and divisor should not be more than 50 apart (approximately). Specifically, they should meet these qualifications:

	Dividend Mod. Ch. greater	Divisor Mod. Ch. greater
Dividend mantissa equal or greater than divisor mantissa	Should not be more than 48 greater.	Should not be more than 51 greater.
Dividend mantissa smaller than divisor mantissa	Should not be more than 49 greater.	Should not be more than 50 greater.

If it is necessary to normalize the quotient, these figures are reduced accordingly.

At completion of the operation, the program register has 01 in positions 4-5, and the modified characteristic of the remainder in positions 6-9 (00xx).

AUTOCODER EXAMPLE (FIGURE 269): Assume that KANSAS has been previously defined as word 750. The assembled instruction is:

S01	23	45	6789
−73	38	00	0750

Floating Divide Double Precision

−75 **FDD**

MACHINE DESCRIPTION: (16-digit dividend) ÷ (8-digit divisor) = (8-digit quotient) with (8-digit remainder).

Same as FLOATING DIVIDE except that accumulator 2 is not reset to zero initially and its mantissa digits become the 8 low-order digits of the dividend, making a 16-digit dividend.

INSTRUCTION FORMAT: Same as FLOATING DIVIDE, except for the sign.

EXAMPLES: To divide, in floating-decimal notation, the number in accumulator 1 by the number in word 1635:

S01	23	45	6789
−75	00	xx	1635

Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
+75 12151822	+67 24273032

Contents of word 1635: +56 30000000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
+69 40506070	+61 12427303

Using the same instruction, with different factors: Contents of accumulators 1 and 2 before the operation:

Acc. 1	Acc. 2
−27 66664446	−19 55550000

Contents of word 1635: +45 22220000. Contents of accumulators 1 and 2 after the operation:

Acc. 1	Acc. 2
−33 30002001	−25 35000000

DATA FLOW: Same as FLOATING DIVIDE, with the exception that accumulator 2 is not set to zero at the start of the operation.

REGISTERS AFFECTED: Same as FLOATING DIVIDE.

TIMING: Same as FLOATING DIVIDE.

COMMENTS: The comments for FLOATING DIVIDE also apply to this code.

For this operation, accumulator 2 contains the eight low-order digits of a 16-digit mantissa, whose modified characteristic is specified in positions 0-1 of accumulator 1.

The sign and modified characteristic of accumulator 2 are ignored by the operation. If the sign is not the same, or the modified characteristic eight less than that of accumulator 1, the result is obtained as if they were.

AUTOCODER EXAMPLE: (FIGURE 270): Actual address 1305 is used. The assembled instruction is:

S01	23	45	6789
−75	00	00	1305

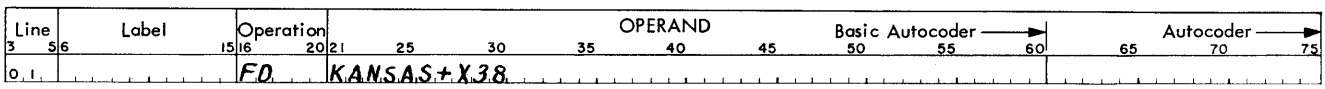


FIGURE 269.

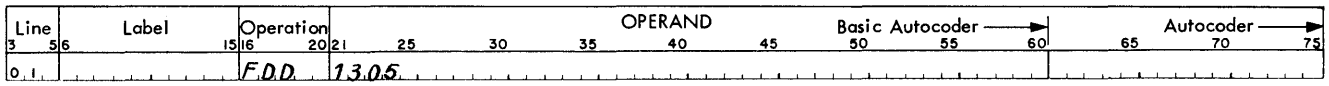


FIGURE 270.

Floating Branch Overflow

+70 **FBV**

MACHINE DESCRIPTION: The *floating-decimal overflow* indicator is tested. If it is ON, the program branches to the location in positions 6-9 (indexable), and the indicator is turned off. If the indicator is not ON, the next sequential instruction is taken.

INSTRUCTION FORMAT: S01 23 45 6789

S01 +70

23 Indexing word for modifying the branch address in positions 6-9.

45 Not used.

6789 Branch address, if the *floating-decimal overflow* indicator is ON.

EXAMPLE: To branch to location 2701 if the *floating-decimal overflow* indicator is ON, and turn it off:

S01	23	45	6789
+70	00	xx	2701

DATA FLOW: If the indicator is ON, the address in positions 6-9, after indexing if so specified, is transmitted to the instruction counter, and the indicator is turned off.

REGISTERS AFFECTED: Possibly the instruction counter and the *floating-decimal overflow* indicator.

TIMING: 36 microseconds.

COMMENTS: After this instruction, the *floating-decimal overflow* indicator is always OFF. Whenever this indicator is ON, the *Exponent Overflow* light on the console is ON.

AUTOCODER EXAMPLE (FIGURE 271): Assume that OHIO has been defined as word 3800. The assembled instruction is:

S01	23	45	6789
+10	00	00	3812

Floating Branch Underflow

−70 **FBU**

MACHINE DESCRIPTION: The *floating-decimal underflow* indicator is tested. If it is ON, the program branches to the location in positions 6-9 (indexable), and the indicator is turned off. If the indicator is not ON, the next sequential instruction is taken.

INSTRUCTION FORMAT: S01 23 45 6789

S01 −70

23 Indexing word for modifying the branch address in positions 6-9.

45 Not used.

6789 Branch address, if the *floating-decimal underflow* indicator is ON.

EXAMPLE: To branch to location 1842 if the *floating-decimal underflow* indicator is ON, and turn it off:

S01	23	45	6789
−70	00	xx	1842

DATA FLOW: If the indicator is ON, the address in positions 6-9, after indexing if so specified, is transmitted to the instruction counter, and the indicator is turned off.

REGISTERS AFFECTED: Possibly the instruction counter and the *floating-decimal underflow* indicator.

TIMING: 36 microseconds.

COMMENTS: After this instruction, the *floating-decimal underflow* indicator is always OFF. Unlike the overflow indicator, this indicator has no light on the console to signal when it is ON.

AUTOCODER EXAMPLE (FIGURE 272): Actual address 2355 is used. The assembled instruction is:

S01	23	45	6789
−70	00	00	2355

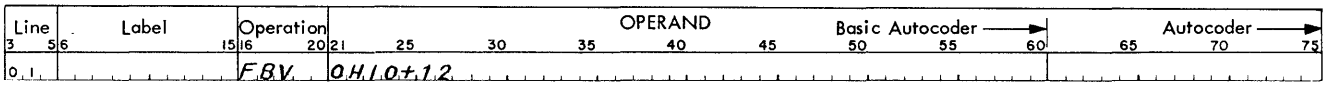


FIGURE 271.

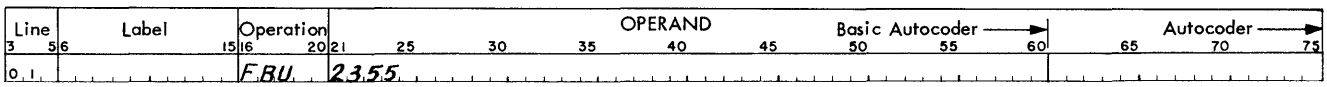


FIGURE 272.

Programming Summaries

This section contains a functional chart of the IBM 7070 operation codes; lists of the operation codes and addresses; and a routine for clearing all of core storage, or a part of it, to zeros. They are presented in this sequence:

1. Functional Chart of 7070 Operation Codes*
2. List of IBM 7070 Operations by Category
3. Core Storage and Register Addresses
4. Op Codes that Allow Accumulator Addresses
5. Op Codes that Use Field Definition
6. Store and Add-to-Storage Codes
7. Index of 7070 Operation Codes by *Autocoder* Mnemonics*
8. Clearing a Specified Portion of Core Storage to Zeros.

*These include the page number containing the description of each code.

Functional Chart of 7070 Operation Codes

This is a chart of the 7070 program instructions in operation-code order. The 10-digit instruction is divided into its four parts: Operation Code, Indexing Word, Control, and Address. Where necessary, the Control or Address portion is further divided into individual digit positions. Included in the chart are the *Autocoder* mnemonics and names of the operations, and a column for comments. The letter S and digits 0-9 at the top refer to the sign and digit positions of each instruction.

The pages that contain the description of these operations are included, and this chart thus serves as an index by operation code.

S01 ±0P	ABBR	NAME	23 IW	45 CL	6789 ADDRESS	PAGE	COMMENTS
+00	HB	Halt and Branch	IW	Not used	Branch address	57	Unconditional branch
−00	HP	Halt and Proceed	IW	Not used	Not used	58	Next instruction from IC
+01	B	Branch	IW	Not used	Branch address	56	Next instruction from location in positions 6-9
−01	NOP	No Operation	IW	Not used	Not used	58	Next instruction from IC
+02	BLX	Branch and Load Location in Index Word	IW	Operand Index Word	Branch address	57	Contents of IC are stored in index word, positions 2-5
+03	CD	Compare Storage to Digit	IW	Value compared with | Position to be compared	Address of data	53	Turns on *high*, *low*, or *equal* indicator
−03		Sign Control	IW	Sign value 3, 6 or 9 | Operation 0. Compare 1. Make	Address of data for operations 0-1	54	Compare operations turn on *high*, *low*, or *equal* indicator
	CSA			Six operations: Compare sign to alpha	Branch address for operation 4		
	CSM			Compare sign to minus			
	CSP			Compare sign to plus			
	MSA			Make sign alpha			
	MSM			Make sign minus			
	MSP			Make sign plus			
	SMSC			2. Sign change sense			
	HMSC			3. Sign change stop			
	BSC			4. Sign change test			
+10	BZ1	Branch if Zero in Accumulator 1	IW	Not used	Branch address	43	Sign is ignored
−10	BM1	Branch if Minus in Accumulator 1	IW	Not used	Branch address	44	Contents are ignored
+11	BV1	Branch if Overflow in Accumulator 1	IW	Not used	Branch address	49	If *overflow* indicator is ON, branch and turn it off
−11	ZST1	Zero Storage and Store Accumulator 1	IW	Field Definition	Address of data	29	
+12	ST1	Store Accumulator 1	IW	Field Definition	Address of data	30	*Field overflow* and *sign change* indicators might be set
−12	STD1	Store Digits from Accumulator 1 and Ignore Sign	IW	Field Definition	Address of data	31	*Field overflow* indicator might be set
+13	ZA1	Zero Accumulator 1 and Add	IW	Field Definition	Address of data	17	
−13	ZS1	Zero Accumulator 1 and Subtract	IW	Field Definition	Address of data	19	
+14	A1	Add to Accumulator 1	IW	Field Definition	Address of data	20	
−14	S1	Subtract from Accumulator 1	IW	Field Definition	Address of data	21	
+15	C1	Compare Accumulator 1 to Storage	IW	Field Definition	Address of data	50	Sets *high*, *low*, or *equal* indicator
−15	CA	Compare Absolute in Accumulator 1 to Absolute in Storage	IW	Field Definition	Address of data	52	Sets *high*, *low*, or *equal* indicator
+16	ZAA	Zero Accumulator 1 and Add Absolute	IW	Field Definition	Address of data	26	
−16	ZSA	Zero Accumulator 1 and Subtract Absolute	IW	Field Definition	Address of data	27	
+17	AA	Add Absolute to Accumulator 1	IW	Field Definition	Address of data	27	
−17	SA	Subtract Absolute from Accumulator 1	IW	Field Definition	Address of data	28	
+18	AS1	Add to Storage from Accumulator 1	IW	Field Definition	Address of data	32	*Field overflow* and/or *sign change* indicators might be set
−18	SS1	Subtract Accumulator 1 from Storage	IW	Field Definition	Address of data	33	*Field overflow* and/or *sign change* indicators might be set
+19	AAS1	Add to Absolute Storage from Accumulator 1	IW	Field Definition	Address of data	34	*Field overflow* indicator might be set
+20	BZ2	Branch if Zero in Accumulator 2	IW	Not used	Branch address	43	Sign is ignored
−20	BM2	Branch if Minus in Accumulator 2	IW	Not used	Branch address	44	Contents are ignored

S01 ±0P	ABBR	NAME	23 IW	45 CL	6789 ADDRESS	PAGE	COMMENTS
+21	BV2	Branch if Overflow in Accumulator 2	IW	Not used	Branch address	49	If *overflow* indicator is ON, branch and turn it off
−21	ZST2	Zero Storage and Store Accumulator 2	IW	Field Definition	Address of data	29	
+22	ST2	Store Accumulator 2	IW	Field Definition	Address of data	30	*Field overflow* and/or *sign change* indicators might be set
−22	STD2	Store Digits from Accumulator 2 and Ignore Sign	IW	Field Definition	Address of data	31	*Field overflow* indicator might be set
+23	ZA2	Zero Accumulator 2 and Add	IW	Field Definition	Address of data	17	
−23	ZS2	Zero Accumulator 2 and Subtract	IW	Field Definition	Address of data	19	
+24	A2	Add to Accumulator 2	IW	Field Definition	Address of data	20	
−24	S2	Subtract from Accumulator 2	IW	Field Definition	Address of data	21	
+25	C2	Compare Accumulator 2 to Storage	IW	Field Definition	Address of data	50	Sets *high, low,* or *equal* indicator
+28	AS2	Add to Storage from Accumulator 2	IW	Field Definition	Address of data	32	*Field overflow* and/or *sign change* indicators might be set
−28	SS2	Subtract Accumulator 2 from Storage	IW	Field Definition	Address of data	33	*Field overflow* and/or *sign change* indicators might be set
+29	AAS2	Add to Absolute Storage from Accumulator 2	IW	Field Definition	Address of data	34	*Field overflow* indicator might be set
+30	BZ3	Branch if Zero in Accumulator 3	IW	Not used	Branch address	43	Sign is ignored
−30	BM3	Branch if Minus in Accumulator 3	IW	Not used	Branch address	44	Contents are ignored
+31	BV3	Branch if Overflow in Accumulator 3	IW	Not used	Branch address	49	If *overflow* indicator is ON, branch and turn it off
−31	ZST3	Zero Storage and Store Accumulator 3	IW	Field Definition	Address of data	29	
+32	ST3	Store Accumulator 3	IW	Field Definition	Address of data	30	*Field overflow* and/or *sign change* indicator might be set
−32	STD3	Store Digits from Accumulator 3 and Ignore Sign	IW	Field Definition	Address of data	31	*Field overflow* indicator might be set
+33	ZA3	Zero Accumulator 3 and Add	IW	Field Definition	Address of data	17	
−33	ZS3	Zero Accumulator 3 and Subtract	IW	Field Definition	Address of data	19	
+34	A3	Add to Accumulator 3	IW	Field Definition	Address of data	20	
−34	S3	Subtract from Accumulator 3	IW	Field Definition	Address of data	21	
+35	C3	Compare Accumulator 3 to Storage	IW	Field Definition	Address of data	50	Sets *high, low,* or *equal* indicator
+38	AS3	Add to Storage from Accumulator 3	IW	Field Definition	Address of data	32	*Field overflow* and/or *sign change* indicator might be set
−38	SS3	Subtract Accumulator 3 from Storage	IW	Field Definition	Address of data	33	*Field overflow* and/or *sign change* indicator might be set
+39	AAS3	Add to Absolute Storage from Accumulator 3	IW	Field Definition	Address of data	34	*Field overflow* indicator might be set
+40	BL	Branch if Low	IW	Not used	Branch address	47	Tests the *low* indicator
−40	BH	Branch if High	IW	Not used	Branch address	48	Tests the *high* indicator
+41	BFV SMFV HMFV	Field Overflow Control	IW	Not used | Operation: 0. Test 1. Sense 2. Stop	Branch address in Operation 0 Not used for operations 1-2	49	Test operation tests *field overflow* indicator
−41	BE	Branch if Equal	IW	Not used	Branch address	48	Test the *equal* indicator
−43	BCX	Branch Compared Index Word	IW	Operand index word	Branch address	68	
+44	BXN	Branch if Index Word Indexing Portion is Non-zero	IW	Operand index word	Branch address	67	Sign and the other six digit positions are ignored
−44	BXM	Branch if Index Word is Minus	IW	Operand index word	Branch address	66	Contents are ignored
+45	XL	Index Word Load	IW	Operand index word	Address of data	60	
−45	XU	Index Word Unload	IW	Operand index word	Address of data	61	
+46	XZA	Index Word Zero and Add to Indexing Portion	IW	Operand index word	Four-digit factor	62	Sign is set to plus and the other six digit positions are unchanged
−46	XZS	Index Word Zero and Subtract from Indexing Portion	IW	Operand index word	Four-digit factor	63	Sign is set to minus and the other six digit positions are unchanged
+47	XA	Index Word Add to Indexing Portion	IW	Operand index word	Four-digit factor	63	Sign may change. Other positions are unchanged

S01 ±0P	ABBR	NAME	23 IW	45 CL	6789 ADDRESS	PAGE	COMMENTS
−47	XS	Index Word Subtract from Indexing Portion	IW	Operand index word	Four-digit factor	64	Sign may change. Other positions are unchanged
+48	XSN	Index Word Set Non-indexing Portion	IW	Operand index word	Four-digit factor	65	Sign and other six digit positions are unchanged
−48	XLIN	Index Word Load and Interchange	IW	Operand index word	Address of data	61	Positions 2-5 and 6-9 are interchanged
+49	BIX	Branch Incremented Index Word	IW	Operand index word	Branch address	69	Branch if incremented indexing portion is not greater than non-indexing portion. Incremented by 1
−49	BDX	Branch Decremented Index Word	IW	Operand index word	Branch address	70	Branch if decremented indexing portion is not brought to zero or beyond zero by non-indexing value
+50	SR# SRR# SL# SLC#	Shift Control	IW	Index word for shift and count	Accumulator (1-3) | Opn * | Length of shift (00-10)	35	*Operation: 0—Shift right 1—Shift right and round 2—Shift left 3—Shift left and count
−50	SR SRR SL SLC SRS SLS SRS SLS	Coupled Shift Control	IW	Index word for shift and count	Digit Position for split, 0 for normal | Opn * | Length of shift (00-20)	36	*Operation, normal: 0—Shift right 1—Shift right and round 2—Shift left 3—Shift left and count Operation, split: 4—Shift right from point Acc 1 5—Shift left from point Acc 1 6—Shift right from point Acc 2 7—Shift left from point Acc 2
+51	BAS BCB	Branch on Alteration Switch or Channel Busy	IW	Operation: Switch 1-4 | 0. Alt sw Chan 1-4 | 1. Ch bsy	Branch address	44	
+53	M	Multiply	IW	Field definition of multiplier	Address of multiplier	22	
−53	D	Divide	IW	Field definition of divisor	Address of divisor	24	
+54	QR QW	Inquiry Control	IW	Inquiry ctrl group (1 or 2) | Operation: 0. Request 1. Reply	Address of first record definition word	213	
+55	PC	Priority Control	IW	Not used	Address of priority mask	224	0—allow; 1—mask
+56	ENA	Edit Numerical to Alphameric	IW	Index word-locates first numerical word	Address of first record definition word	77	RDW defines alpha area
−56	ENS	Edit Numerical to Alphameric with Sign Control	IW	Index word-locates first numerical word	Address of first record definition word	79	RDW defines alpha area
+57	ENB	Edit Numerical to Alphameric with Blank Insertion	IW	Index word-locates first numerical word	Address of first record definition word	80	RDW defines alpha area
−57	EAN	Edit Alphameric to Numerical	IW	Index word-locates first word to be filled with numerical data	Address of first record definition word	81	RDW defines alpha area
+60	BAL BUL BUL BQL BQL BTL BTL BTL BTL BDL BDL BDL	Stacking Latch Test	IW	Stacking latch: 00 Any stacking latch 01 Unit Record A 02 Unit Record B 03 Inquiry Control 1 04 Inquiry Control 2 10-19 Tape units 0-9 on Channel 1 20-29 Tape units 0-9 on Channel 2 30-39 Tape units 0-9 on Channel 3 40-49 Tape units 0-9 on Channel 4 50-53 Access arm 0, disk stor. 0-3 60-63 Access arm 1, disk stor. 0-3 70-73 Access arm 2, disk stor. 0-3	Branch address	225	

S01 ±0P	ABBR	NAME	23 IW	45 CL		6789 ADDRESS	PAGE	COMMENTS
+61		Electronic Switch Control	IW	Operation:	Switch Number	Branch address	45	Controls the ten switches in word 0101
	BES			0-Test				
	ESN			1-Turn on	(0-9)			
	ESF			2-Turn off				
	BSN			3-Test and turn on				
	BSF			4-Test and turn off				
−61		Stacking Latch Set	IW	Stacking latch:		Not used	226	
	ULN			01 Unit Record A				
	ULN			02 Unit Record B				
	QLN			03 Inquiry Control 1				
	QLN			04 Inquiry Control 2				
	TLN			10-19 Tape units 0-9 on Channel 1				
	TLN			20-29 Tape units 0-9 on Channel 2				
	TLN			30-39 Tape units on Channel 3				
	TLN			40-49 Tape units on Channel 4				
	DLN			50-53 Access Arm 0 disk stor. 0-3				
	DLN			60-63 Access Arm 1 disk stor. 0-3				
	DLN			70-73 Access Arm 2 disk stor. 0-3				
+62		Electronic Switch Control		Same as +61, except for the ten switches that are controlled			45	Controls the ten switches in word 0102
−62		Stacking Latch Reset	IW	Stacking latch:		Not used	226	
	ULF			01 Unit Record A				
	ULF			02 Unit Record B				
	QLF			03 Inquiry Control 1				
	QLF			04 Inquiry Control 2				
	TLF			10-19 Tape units 0-9 on Channel 1				
	TLF			20-29 Tape units 0-9 on Channel 2				
	TLF			30-39 Tape units on Channel 3				
	TLF			40-49 Tape units on Channel 4				
	DLF			50-53 Access arm 0, disk stor. 0-3				
	DLF			60-63 Access arm 1, disk stor. 0-3				
	DLF			70-73 Access arm 2, disk stor. 0-3				
+63		Electronic Switch Control		Same as +61, except for the ten switches that are controlled			45	Controls the ten switches in word 0103
+64	PR	Priority Release	IW	Not used		Branch address	227	Branch address is usually 0097
+65	RS	Record Scatter	IW	Index word-locates first transmitting word		Address of first record definition word	76	RDW defines receiving area
−65	RG	Record Gather	IW	Index word-locates first receiving word		Address of first record definition word	77	RDW defines transmitting area
+66	LL	Lookup Lowest	IW	Field Definition		Address of first record definition word	83	
+67	LE	Lookup Equal Only	IW	Field Definition		Address of first record definition word	86	Index Word 98: positions 2-5, found address
+68	LEH	Lookup Equal or High	IW	Field Definition		Address of first record definition word	87	positions 6-9, increment
+69		Card Control	IW	Syn. (1-3)	Oper:	Address of first record definition word	117	
	US				0—Set PES			
	UR				1—Read			
	UW/UP				2—Write or Punch			
	UWIV/ UPIV				3—Write or Punch invalid			
	TYP				4—Type			

S01 ±0P	ABBR	NAME	23 IW	45 CL	6789 ADDRESS	PAGE	COMMENTS
+70	FBV	Floating Branch Overflow	IW	Not used	Branch address	242	Tests the *floating decimal overflow* indicator
−70	FBU	Floating Branch Underflow	IW	Not used	Branch address	242	Tests the *floating decimal underflow* indicator
+71	FR	Floating Round	IW	Not used	Not used	237	
+73	FM	Floating Multiply	IW	Not used	Address of multiplicand	238	
−73	FD	Floating Divide	IW	Not used	Address of divisor	239	
+74	FA	Floating Add	IW	Not used	Address of data	231	Accumulator 2 cleared at start
−74	FS	Floating Subtract	IW	Not used	Address of data	233	Accumulator 2 cleared at start
+75	FZA	Floating Zero and Add	IW	Not used	Address of data	230	
−75	FDD	Floating Divide Double Precision	IW	Not used	Address of divisor	241	
+76	FAD	Floating Add Double Precision	IW	Not used	Address of data	234	Accumulator 2 not cleared at start
−76	FADS	Floating Add Double Precision Suppress Normalization	IW	Not used	Address of data	235	
+77	FAA	Floating Add Absolute	IW	Not used	Address of data	235	
−77	FSA	Floating Subtract Absolute	IW	Not used	Address of data	236	
±81		Tape Control 1	IW	Tape Unit (0-9) | Oper:	If position 5 is not zero, address of first record definition word	98	
	(P) TR			1—Read			+ Priority
	(P) TRR			2—Read per RM			− No priority
	(P) TW			3—Write			Code 81 means Channel 1
	(P) TWR			4—Write per RM			
	(P) TWZ			5—Write with zero elimination			
	(P) TWC			6—Write per RM and zero elimination			
	(P) TSF			7—Segment forward space per count			
	(P) TSB			8—Segment backspace per count			
				0—Operation in position 9	If pos. 5 is zero, operation is in pos. 9:		
	TSEL				0—no-op select		
	(P) TM				1—write tape mark		
	TRW				2—rewind		
	TRU				3—rewind unload		
	TRB				4—backspace		
	(P) TSM				5—write tape segment mark		
	TSK				6—skip		
	TEF				7—turn off end of file		
	TSLD				8—set lower density		
	TSHD				9—set high density		
±82		Tape Control 2		Same as ±81, except that Channel 2 is used		98	
±91		Disk Storage Control 1	IW	Not used | Oper:	Address of first record definition word	113	+ Priority
	(P) DR			0—read			− No priority
	(P) DW			1—write			Code 91 means Channel 1
±92		Disk Storage Control 2		Same as ±91, except that Channel 2 is used		113	
+95	PDS	Priority Disk Seek	IW	Not used	Branch address	114	
−95	DAR	Disk Storage Arm Release	IW	Not used	Not used | arm (0-2) | disk storage unit 0-3	116	Provides any arm in the file unit, or a specific arm. Branch if arm not available.

List of IBM 7070 Instructions by Category

Many of the IBM 7070 operation codes fall into several different categories. Each category is listed here with all of the operations that it includes. Thus, each operation appears on the list once for each category to which it applies. There are 15 categories:

1. *Arithmetic.* All instructions that involve any type of arithmetic operation—adding, subtracting, multiplying, or dividing, including reset-add and subtract.
2. *Shift.* All instructions that, include, or may include, the shifting of the number in an accumulator.
3. *Branch.* Instructions that may cause the program to take its next instruction from the address in positions 6-9 (indexable).
4. *Compare.* All instructions that turn on either the *high, equal,* or *low* indicator as the result of a test.
5. *Data to core storage—one word or less.* All operations that bring information to a single storage word or part of a word, as specified by field definition (positions 4 and 5). These operations all cause previous data in a storage word to be replaced by new data. (Codes to load index words are not included.)
6. *Data to core storage—one word or more.* All operations that bring information to one or more storage words, under control of record-definition words (addressed by positions 6-9). These operations cause data in core storage to be replaced by new data.
7. *Index Words.* All operations that use index words for other than indexing, specifying the operand index word in positions 4 and 5.
8. *Magnetic Tape.* Instructions that involve the use of a tape channel or unit.
9. *Disk Storage.* Instructions that involve the use of a disk-storage channel or unit.
10. *Unit Record.* All instructions that involve the use of a card reader, punch, printer, the console typewriter, or the input/output synchronizers.
11. *Core-to-core block transmission.* The codes that move one or more words of core storage to other locations in core storage, under control of record-definition words.
12. *Table lookup.* The three table lookup codes.
13. *Inquiry.* The codes that involve the processing of inquiry requests and formulation of replies.
14. *Priority.* All of the instructions that have anything to do with Automatic Priority Processing.
15. *Floating Decimal.* The instructions made available by the optional floating-decimal arithmetic feature.
16. *Miscellaneous.* All instructions that don't fully come under the other categories.

Each list is given in sequence by *Autocoder* mnemonics.

Arithmetic

+14, +24, +34	A#	Add to Accumulator #
+17	AA	Add Absolute to Accumulator 1
+18, +28, +38	AS#	Add to Storage from Accumulator #
+19, +29, +39	AAS#	Add to Absolute Storage from Accumulator #
+49	BIX	Branch Incremented Index Word
−49	BDX	Branch Decremented Index Word
−53	D	Divide
+74	FA	Floating Add
−74	FS	Floating Subtract
+73	FM	Floating Multiply
−73	FD	Floating Divide
+77	FAA	Floating Add Absolute
−77	FSA	Floating Subtract Absolute
+75	FZA	Floating Zero and Add
+76	FAD	Floating Add Double Precision
−75	FDD	Floating Divide Double Precision
−76	FADS	Floating Add Double Precision and Suppress Normalization
+53	M	Multiply
−14, −24, −34	S#	Subtract from Accumulator #
−17	SA	Subtract Absolute from Accumulator 1
−18, −28, −38	SS#	Subtract Accumulator # from Storage
+50	SRR#	Shift Right and Round Accumulator #
+50	SLC#	Shift Left and Count Accumulator #
−50	SR	Shift Right and Round Coupled
−50	SLC	Shift Left and Count Coupled
+47	XA	Index Word Add to Indexing Portion
−47	XS	Index Word Subtract from Indexing Portion
+46	XZA	Index Word Zero and Add to Indexing Portion
−46	XZS	Index Word Zero and Subtract from Indexing Portion
+13, +23, +33	ZA#	Zero Accumulator # and Add
−13, −23, −33	ZS#	Zero Accumulator
+16	ZAA	Zero Accumulator 1 and Add Absolute
−16	ZSA	Zero Accumulator 1 and Subtract Absolute

Shift

+71	FR	Floating Round
+75	FZA	Floating Zero and Add
+50	SR#	Shift Right Accumulator #

+50	SRR#	Shift Right and Round Accumulator #
+50	SL#	Shift Left Accumulator #
+50	SLC#	Shift Left and Count Accumulator #
−50	SR	Shift Right Coupled
−50	SRR	Shift Right and Round Coupled
−50	SL	Shift Left Coupled
−50	SLC	Shift Left and Count Coupled
−50	SRS	Shift Right Split
−50	SLS	Shift Left Split

Branch

+01	B	Branch
+02	BLX	Branch and Load Location in Index Word
−10, −20, −30	BM#	Branch if Minus in Accumulator #
+10, +20, +30	BZ#	Branch if Zero in Accumulator #
+11, +21, +31	BV#	Branch if Overflow in Accumulator #
+41	BFV	Branch if Field Overflow
−03	BSC	Branch if Sign Change
−40	BH	Branch if High
−41	BE	Branch if Equal
+40	BL	Branch if Low
+51	BAS	Branch if Alteration Switch is On
+61, +62, +63	BES	Branch if Electronic Switch is On
+61, +62, +63	BSF	Branch if Electronic Switch is On and Set Off if On
+61, +62, +63	BSN	Branch if Electronic Switch is On and Set On if Off
+51	BCB	Branch if Channel is Busy
+69	BCT	Branch if Carriage Tape Sensed Channel 9 I/O Sync
+60	BAL	Branch if Any Stacking Latch is On
+60	BDL	Branch if Disk Latch is On
+60	BQL	Branch if Inquiry Control Latch is On
+60	BTL	Branch if Tape Latch is On
+60	BUL	Branch if Unit Record Latch is On
−44	BXM	Branch if Index Word is Minus
+44	BXN	Branch if Index Word is Nonzero
−43	BCX	Branch Compared Index Word
+49	BIX	Branch Incremented Index Word
−49	BDX	Branch Decremented Index Word
+70	FBV	Floating Branch Overflow
−70	FBU	Floating Branch Underflow
+00	HB	Halt and Branch
+95	PDS	Priority Disk Seek

Compare

+15, +25, +35	C#	Compare Accumulator # to Storage
−15	CA	Compare Absolute in Accumulator 1 to Absolute in Storage
+03	CD	Compare Storage to Digit
−03	CSA	Compare Sign to Alpha
−03	CSP	Compare Sign to Plus
−03	CSM	Compare Sign to Minus

Data to Core Storage—One Word or Less (Field Definition)

+18, +28, +38	AS#	Add to Storage from Accumulator #
+19, +29, +39	AAS#	Add to Absolute Storage from Accumulator #
−18, −28, −38	SS#	Subtract Accumulator # from Storage
+12, +22, +32	ST#	Store Accumulator #
−12, −22, −32	STD#	Store Digits from Accumulator # and Ignore Sign
−45	XU	Index Word Unload
−11, −21, −31	ZST#	Zero Storage and Store Accumulator #

Data to Core Storage—One Word or More (Block Transmission)

−91, −92	DR	Disk Read
−57	EAN	Edit Alphameric to Numerical
+56	ENA	Edit Numerical to Alphameric
+57	ENB	Edit Numerical to Alphameric with Blank Insertion
−56	ENS	Edit Numerical to Alphameric with Sign Control
+91, +92	PDR	Priority Disk Read
+81, +82	PTR	Priority Tape Read
+81, +82	PTRR	Priority Tape Read Per Record Mark Control
+54	QR	Inquiry Read
−65	RG	Record Gather
+65	RS	Record Scatter
−81, −82	TR	Tape Read
−81, −82	TRR	Tape Read Per Record Mark Control
+69	UR	Unit Record Read

Index Words

+02	BLX	Branch and Load Location in Index Word
−44	BXM	Branch if Index Word is Minus
+44	BXN	Branch if Index Word is Nonzero
−43	BCX	Branch Compared Index Word
+49	BIX	Branch Incremented Index Word
−49	BDX	Branch Decremented Index Word
−57	EAN	Edit Alphameric to Numerical
+56	ENA	Edit Numerical to Alphameric
+57	ENB	Edit Numerical to Alphameric with blank insertion
−56	ENS	Edit Numerical to Alphameric with Sign Control
−65	RG	Record Gather
+65	RS	Record Scatter
+50	SLC#	Shift Left and Count Accumulator #
−50	SLC	Shift Left and Count Coupled
+45	XL	Index Word Load
−48	XLIN	Index Word Load and Interchange
−45	XU	Index Word Unload
+47	XA	Index Word Add to Indexing Portion
−47	XS	Index Word Subtract from Indexing Portion
+46	XZA	Index Word Zero and Add to Indexing Portion
−46	XZS	Index Word Zero and Subtract from Indexing Portion
+48	XSN	Index Word Set Nonindexing Portion

Magnetic Tape

+51	BCB	Branch if Channel is Busy
+60	BTL	Branch if Tape Latch is On
+55	PC	Priority Control
+81, +82	PTR	Priority Tape Read
+81, +82	PTRR	Priority Tape Read Per Record Mark Control
+81, +82	PTW	Priority Tape Write
+81, +82	PTWZ	Priority Tape Write with Zero Elimination
+81, +82	PTWR	Priority Tape Write Per Record Mark Control
+81, +82	PTWC	Priority Tape Write with Zero Elimination and Per Record Mark Control
+81, +82	PTSB	Priority Tape Segment Backward Space
+81, +82	PTSF	Priority Tape Segment Forward Space
±81, ±82	PTM	Priority Tape Mark Write
±81, ±82	PTSM	Priority Tape Segment Mark Write
−81, −82	TR	Tape Read
−81, −82	TRR	Tape Read Per Record Mark Control
−81, −82	TW	Tape Write
−81, −82	TWR	Tape Write Per Record Mark Control
−81, −82	TWZ	Tape Write with Zero Elimination
−81, −82	TWC	Tape Write with Zero Elimination and Per Record Mark Control Combined
±81, ±82	TRB	Tape Record Backspace
−81, −82	TSB	Tape Segment Backward Space
−81, −82	TSF	Tape Segment Forward Space
±81, ±82	TRW	Tape Rewind
±81, ±82	TM	Tape Mark Write
±81, ±82	TSM	Tape Segment Mark Write
±81, ±82	TEF	Tape End of File Off
±81, ±82	TSK	Tape Skip
±81, ±82	TSEL	Tape No-Op Select
±81, ±82	TRU	Tape Rewind Unload
±81, ±82	TSHD	Tape Set High Density
±81, ±82	TSLD	Tape Set Lower Density
−61	TLN	Tape Latch Set On
−62	TLF	Tape Latch Set Off

Disk Storage

+51	BCB	Branch if Channel is Busy
+60	BDL	Branch if Disk Latch is On
−91, −92	DR	Disk Read
−91, −92	DW	Disk Write
−95	DAR	Disk Arm Release
−61	DLN	Disk Latch On
−62	DLF	Disk Latch Off
+55	PC	Priority Control
+95	PDS	Priority Disk Seek
+91, +92	PDR	Priority Disk Read
+91, +92	PDW	Priority Disk Write

Unit Record

+60	BUL	Branch if Unit Record Latch is On
+69	TYP	Type
+55	PC	Priority Control
+69	UR	Unit Record Read I/O
+69	UP	Unit Record Punch I/O
+69	UW	Unit Record Write I/O
+69	UPIV	Unit Record Punch Invalid I/O
+69	UWIV	Unit Record Write Invalid I/O
−61	ULN	Unit Record Latch Set On
−62	ULF	Unit Record Latch Set Off
+69	US	Unit Record Signal I/O

Core-to-Core Block Transmission

−57	EAN	Edit Alphameric to Numerical
+56	ENA	Edit Numerical to Alphameric
+57	ENB	Edit Numerical to Alphameric with Blank Insertion
−56	ENS	Edit Numerical to Alphameric with Sign Control
−65	RG	Record Gather
+65	RS	Record Scatter

Table Lookup

+67	LE	Lookup Equal Only
+66	LL	Lookup Lowest
+68	LEH	Lookup Equal or High

Inquiry

+60	BQL	Branch if Inquiry Control Latch is On
+55	PC	Priority Control
+54	QR	Inquiry Read
+54	QW	Inquiry Write
−62	QLF	Inquiry Control Latch Set On
−61	QLN	Inquiry Control Latch Set Off

Priority

+60	BAL	Branch if Any Stacking Latch is On
+60	BDL	Branch if Disk Latch is On
+60	BQL	Branch if Inquiry Control Latch is On
+60	BTL	Branch if Tape Latch is On
+60	BUL	Branch if Unit Record Latch is On
−61	DLN	Disk Latch On
−62	DLF	Disk Latch Off
+55	PC	Priority Control
+64	PR	Priority Release
+95	PDS	Priority Disk Seek
+91, +92	PDR	Priority Disk Read
+91, +92	PDW	Priority Disk Write
+81, +82	PTR	Priority Tape Read
+81, +82	PTRR	Priority Tape Read Per Record Mark Control
+81, +82	PTW	Priority Tape Write
+81, +82	PTWZ	Priority Tape Write with Zero Elimination
+81, +82	PTWR	Priority Tape Write Per Record Mark Control
+81, +82	PTWC	Priority Tape Write with Zero Elimination and Per Record Mark Control Combined
+81, +82	PTSB	Priority Tape Segment Backward Space
+81, +82	PTSF	Priority Tape Segment Forward Space

+81, +82	PTM	Priority Tape Mark Write
+81, +82	PTSM	Priority Tape Segment Mark Write
−62	QLF	Inquiry Control Latch Set On
−61	QLN	Inquiry Control Latch Set Off
−61	TLN	Tape Latch Set On
−62	TLF	Tape Latch Set Off
−61	ULN	Unit Record Latch A Set On
−62	ULF	Unit Record Latch A Set Off
−61	ULN	Unit Record Latch B Set On
−62	ULF	Unit Record Latch B Set Off

Floating Decimal

+74	FA	Floating Add
−74	FS	Floating Subtract
+73	FM	Floating Multiply
−73	FD	Floating Divide
+71	FR	Floating Round
+77	FAA	Floating Add Absolute
−77	FSA	Floating Subtract Absolute
+75	FZA	Floating Zero and Add
+76	FAD	Floating Add Double Precision
−75	FDD	Floating Divide Double Precision
−76	FADS	Floating Add Double Precision and Suppress Normalization
+70	FBV	Floating Branch Overflow
−70	FBU	Floating Branch Underflow

Miscellaneous

+51	BAS	Branch if Alteration Switch is On
+61, +62, +63	BES	Branch if Electronic Switch is On
+61, +62, +63	BSF	Branch if Electronic Switch is On and Set Off if On
+61, +62, +63	BSN	Branch if Electronic Switch is On and Set On if Off
+61, +62, +63	ESN	Electronic Switch On
+61, +62, +63	ESF	Electronic Switch Off
+00	HB	Halt and Branch
−00	HP	Halt and Proceed
+41	HMFV	Halt Mode for Field Overflow
−03	HMSC	Halt Mode for Sign Change
−03	MSA	Make Sign Alpha
−03	MSP	Make Sign Plus
−03	MSM	Make Sign Minus
−01	NOP	No Operation
+41	SMFV	Sense Mode for Field Overflow
−03	SMSC	Sense Mode for Sign Change

Core Storage and Register Addresses

Core Storage Addresses

0000-4999	(5000-word capacity)
0000-9989	(9990-word capacity)

Core Storage Locations With Special Functions

0001-0099	Indexing words 01-99
0097	Priority address word
0098	Table lookup indexing value and found address
0099	Address of priority final status word
0100	Address at which indicator settings are stored prior to priority routine
0101-0103	Electronic switches
0104	Unit-record priority A branch address
0105	Unit-record priority B branch address
0106	Inquiry-control 1 priority branch address
0107	Inquiry-control 2 priority branch address
0110-0119	Final-status words, tape units 0-9, channel 1
0120-0129	Final-status words, tape units 0-9, channel 2
0130-0139	Final-status words, tape units 0-9, channel 3
0140-0149	Final-status words, tape units 0-9, channel 4
0150-0159	Tape priority branch addresses
0160-0169	Initial-status words, tape units 0-9, channel 1
0170-0179	Initial-status words, tape units 0-9, channel 2
0180-0189	Initial-status words, tape units 0-9, channel 3
0190-0199	Initial-status words, tape units 0-9, channel 4
0200-0203	Final-status words, arm 0, disk-storage units 0-3
0210-0213	Final-status words, arm 1, disk-storage units 0-3
0220-0223	Final-status words, arm 2, disk-storage units 0-3
0240-0249	Disk read and write priority branch addresses
0250-0253	Initial status words, arm 0, disk-storage units 0-3
0260-0263	Initial status words, arm 1, disk-storage units 0-3
0270-0273	Initial status words, arm 2, disk-storage units 0-3
0290-0299	Disk seek priority branch addresses

Register Addresses

9991	Accumulator 1
9992	Accumulator 2
9993	Accumulator 3
9995	Program register—addressable from console only.
9999	Instruction counter—addressable from console only

OP Codes that Allow Accumulator Addresses

+03	Compare Storage to Digit	CD
−03	Sign Control:	
	Compare Sign to Alpha	CSA
	Compare Sign to Minus	CSM
	Compare Sign to Plus	CSP
	Make Sign Alpha	MSA
	Make Sign Minus	MSM
	Make Sign Plus	MSP
−11, −21, −32	Zero Storage and Store Accumulator #	ZST1, ZST2, ZST3
+12, +22, +32	Store Accumulator #	ST1, ST2, ST3
−12, −22, −32	Store Digits from Accumulator # and Ignore Sign	STD1, STD2, STD3
+13, +23, +33	Zero Accumulator # and Add	ZA1, ZA2, ZA3
−13, −23, −33	Zero Accumulator # and Subtract	ZS1, ZS2, ZS3
+14, +24, +34	Add to Accumulator #	A1, A2, A3
−14, −24, −34	Subtract from Accumulator #	S1, S2, S3
+15, +25, +35	Compare Accumulator # to Storage	C1, C2, C3
−15	Compare Absolute Accumulator 1 to Absolute in Storage	CA
+16	Zero Accumulator 1 and Add Absolute	ZAA
−16	Zero Accumulator 1 and Subtract Absolute	ZSA
+17	Add Absolute to Accumulator 1	AA
−17	Subtract Absolute from Accumulator 1	SA
+18, +28, +38	Add to Storage from Accumulator #	AS1, AS2, AS3
−18, −28, −38	Subtract Accumulator # from Storage	SS1, SS2, SS3
+19, +29, +39	Add to Absolute Storage from Accumulator #	AAS1, AAS2, AAS3
+45	Index Word Load	XL
−45	Index Word Unload	XU
−48	Index Word Load and Interchange	XLIN
+53	Multiply	M
−53	Divide	D

OP Codes that Use Field Definition

−11, −21, −31	Zero Storage and Store Accumulator #	ZST1, ZST2, ZST3
+12, +22, +32	Store Accumulator #	ST1, ST2, ST3
−12, −22, −32	Store Digits from Accumulator # and Ignore Sign	STD1, STD2, STD3
+13, +23, +33	Zero Accumulator # and Add	ZA1, ZA2, ZA3
−13, −23, −33	Zero Accumulator # and Subtract	ZS1, ZS2, ZS3
+14, +24, +34	Add to Accumulator #	A1, A2, A3
−14, −24, −34	Subtract from Accumulator #	S1, S2, S3
+15, +25, +35	Compare Accumulator # to Storage	C1, C2, C3
−15	Compare Absolute in Accumulator 1 to Absolute in Storage	CA
+16	Zero Accumulator 1 and Add Absolute	ZA
+17	Add Absolute to Accumulator 1	AA
−17	Subtract Absolute from Accumulator 1	SA
+18, +28, +38	Add to Storage from Accumulator #	AS1, AS2, AS3
−18, −28, −38	Subtract Accumulator # from Storage	SS1, SS2, SS3
+19, +29, +39	Add to Absolute Storage from Accumulator #	AAS1, AAS2, AAS3
+53	Multiply	M
−53	Divide	D
+66	Lookup Lowest	LL
+67	Lookup Equal Only	LE
+68	Lookup Equal or High	LEH

Store and Add-to-Storage Codes

All store and add-to-storage codes can turn on the *field-overflow* indicator:

−11, −21, −31	Zero Storage and Store Accumulator #	ZST1, ZST2, ZST3
+12, +22, +32	Store Accumulator #	ST1, ST2, ST3
−12, −22, −32	Store Digits from Accumulator # and Ignore Sign	STD1, STD2, STD3
+18, +28, +38	Add to Storage from Accumulator #	AS1, AS2, AS3
−18, −28, −38	Subtract Accumulator # from Storage	SS1, SS2, SS3
+19, +29, +39	Add to Absolute Storage from Accumulator #	AAS1, AAS2, AAS3

These codes *can* turn on the *sign-change* indicator:

+12, +22, +32	Store Accumulator #	ST1, ST2, ST3
+18, +28, +38*	Add to Storage from Accumulator #	AS1, AS2, AS3
−18, −28, −38*	Subtract Accumulator # from Storage	SS1, SS2, SS3

*Only if less than a full word is field-defined

These codes *cannot* turn on the *sign-change* indicator:

−11, −21, −31	Zero Storage and Store Accumulator #	ZST1, ZST2, ZST3
−12, −22, −32	Store Digits from Accumulator # and Ignore Sign	STD1, STD2, STD3
+19, +29, +39	Add to Absolute Storage from Accumulator #	AAS1, AAS2, AAS3

Index of 7070 Operation Codes by Autocoder Mnemonics

			Page
SRR	−50	Shift Right and Round Coupled	37
SRR1, 2, 3	+50	Shift Right and Round Accumulator #	35
SRS	−50	Shift Right Split	37
SS1, 2, 3	−18, −28, −38	Subtract Accumulator # from Storage	33
ST1, 2, 3	+12, +22, +32	Store Accumulator #	30
STD1, 2, 3	−12, −22, −32	Store Digits from Accumulator # and Ignore Sign	31
TEF	±81, ±82	Tape End of File Off	106
TLF	−62	Tape Latch Set Off	227
TLN	−61	Tape Latch Set On	226
TM	−81, −82	Tape Mark Write	104
TR	−81, −82	Tape Read	100
TRB	±81, ±82	Tape Record Backspace	105
TRR	−81, −82	Tape Read per Record Mark Control	100
TRU	±81, ±82	Tape Rewind Unload	104
TRW	±81, ±82	Tape Rewind	104
TSB	−81, −82	Tape Segment Backward Space	103
TSEL	±81, ±82	Tape No-op Select	103
TSF	−81, −82	Tape Segment Forward Space	102
TSHD	±81, ±82	Tape Set High Density	107
TSK	±81, ±82	Tape Skip	106
TSLD	±81, ±82	Tape Set Lower Density	106
TSM	−81, −82	Tape Segment Mark Write	105
TW	−81, −82	Tape Write	101
TWC	−81, −82	Tape Write with Zero Elimination and per Record Mark Control Combined	102
TWR	−81, −82	Tape Write per Record Mark Control	101
TWZ	−81, −82	Tape Write with Zero Elimination	101
TYP	+69	Type	119
ULF	−62	Unit Record Latch Set Off	227
ULN	−61	Unit Record Latch Set On	226
UP	+69	Unit Record Punch	118
UPIV	+69	Unit Record Punch Invalid	118
UR	+69	Unit Record Read	117
US	+69	Unit Record Signal	118
UW	+69	Unit Record Write	118
UWIV	+69	Unit Record Write Invalid	118
XA	+47	Index Word Add to Indexing Portion	63
XL	+45	Index Word Load	60
XLIN	−48	Index Word Load and Interchange	61
XS	−47	Index Word Subtract from Indexing Portion	64
XSN	+48	Index Word Set Non-Indexing Portion	65
XU	−45	Index Word Unload	61
XZA	+46	Index Word Zero and Add to Indexing Portion	62
XZS	−46	Index Word Zero and Subtract from Indexing Portion	63
ZA1, 2, 3	+12, +23, +33	Zero Accumulator # and Add	17
ZAA	+16	Zero Accumulator 1 and Add Absolute	26
ZS1, 2, 3	−13, −23, −33	Zero Accumulator # and Subtract	19
ZSA	−16	Zero Accumulator 1 and Subtract Absolute	27
ZST1, 2, 3	−11, −21, −31	Zero Storage and Store Accumulator #	29

Clearing a Specified Portion of Core Storage to Zeros

It is often desirable to clear all of magnetic-core storage, or a specified portion of it, to zeros. This is often done during program testing; it can also be used for 7070 systems that use more than one stored program, clearing core storage after one program has been used, before loading the second program. Areas used for storage of data such as tables can be cleared by this method as well. The zeros can be either all plus or all minus.

Instructions and Addresses

Two instructions are loaded from the console. They are a read-card instruction, and the record-definition word defining the storage words to be read into. The instruction is stored in location 0000, and the RDW is stored in location 0001, as follows:

Location	Contents	
0000	+69 00 1 1 0001	UR
0001	−00 0002 0006	(RDW)

The card to be read contains the instructions for the operation, and the addresses that specify the portion of core storage to be cleared. Assume, for example, that storage locations 1700-1799 are to be cleared. Columns 1-50 are brought to words 0002-0006, and contain the following:

Card Columns	Location	Contents	
1-10	0002	±0000000000	zeros to be used
11-20	0003	−4505020000	instructions
21-30	0004	+6500050006	instructions
31-40	0005	+0017000000	addresses
41-50	0006	−0017011799	addresses

The card can be completely prepunched except for the addresses in columns 33-36 and 43-50 and the zone punch in column 10 for plus or minus zeros. Figure 273 is an example of the prepunched card (note the prepunched X in column 50).

Operation

Place the card in the 7500 feed, and start the program from the console at 0000. The card is read, bringing columns 1-50 to words 0002-0006. If there is no validity error or end of file, the next instruction is taken from 0003. (End-of-file does not occur unless there are no other cards to be read; the end-of-file key on the 7500 must be used, however, if there are no cards in back of this one.)

The instruction in 0003 is as follows.

	S01	23	45	6789	
Index-word unload	−45	05	02	0000	XU

This instruction puts the zeros in word 0002 into location 0000, indexed by positions 2-5 of word 0005. These positions contain 1700; thus, word 1700 is loaded with zeros.

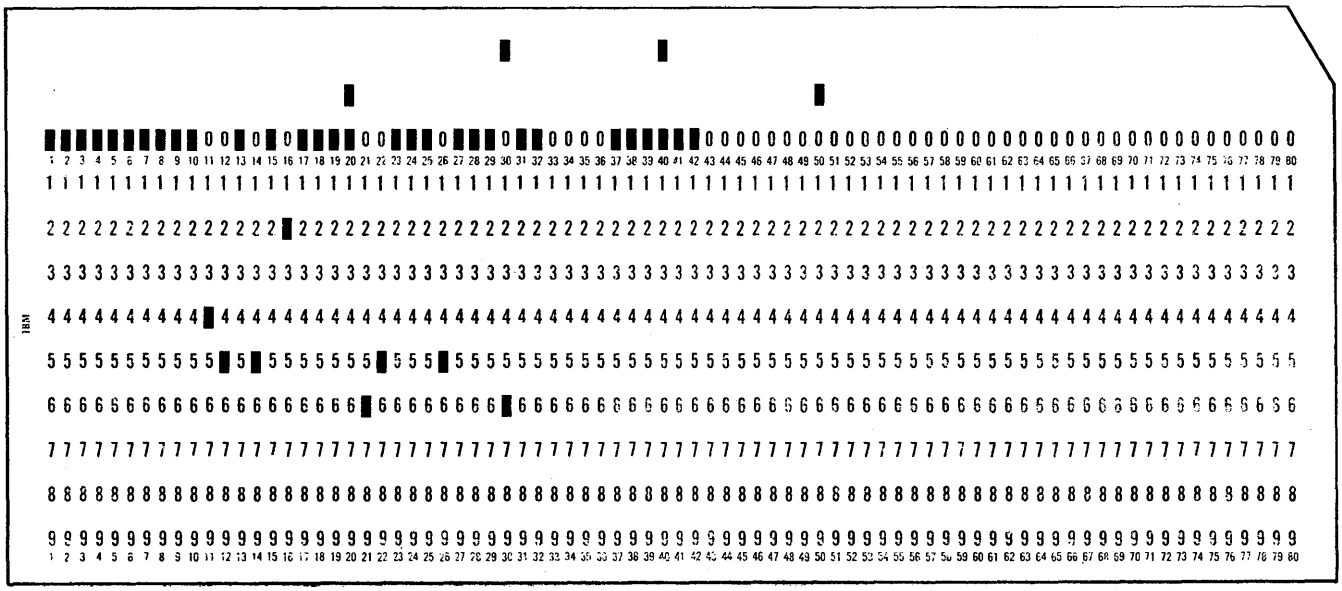


FIGURE 273. INPUT CARD FOR CLEARING A PORTION OF STORAGE

The instruction in word 0004 is as follows:

	S01	23	45	6789	
Record scatter	+65	00	05	0006	RS

Word 0006 is an RDW, defining the remainder of the area to be cleared (1701-1799). This is the receiving area. Positions 2-5 of index word 05 define the first word of the transmitting area (1700). Thus, this instruction "scatters" the contents of 1700 to 1701, then the contents of 1701 to 1702, etc., until the contents of 1798 are brought to 1799, which is the *stop* address of the RDW.

The operation code in word 0005 is +00, which is halt and proceed (HP), thus ending the routine. In this example, the contents of the instruction counter and program register are typed as follows:

IC	PR
0006	+0017000000

In an operation where 0005 is zeroized, the program register would be typed as ±0000000000.

TIMING: All of core storage, locations 0000-9989, can be cleared to zeros by this method, in about 2½ seconds, including card-read and typing time.

Console

The IBM 7150 Console contains operating keys, lights, and switches for the central processing unit of the IBM 7070. As shown in Figure 274, it has three operating components: the *operating panel,* the *operating keyboard,* and the *typewriter keyboard.* The two keyboards are beside each other in the center portion of the console, and the operating panel is vertically mounted at the right. The console typewriter is in the same unit as the keyboards.

There are no lights on the console for the purpose of displaying the contents of a storage word. This is done by the console typewriter, providing a permanent record of all storage words that have been displayed.

Another feature of the IBM 7150 Console is the *illuminated key.* An illuminated key is both a key and a light. When the key is pressed, it illuminates, indicating that it is ON. The light goes OFF when its feature is turned off.

Automatic Typing on Machine Stop

Every time the 7070 program stops, regardless of the reason, the console typewriter automatically types the contents of the instruction counter (4 digits) and the program register (10 digits with sign). There is a carriage return before and after the typing.

For example, if the instruction in location 1449 is index-word load, but no index word is specified in positions 4-5, it is an invalid instruction:

S01	23	45	6789	
+45	21	00	3112	XL

The machine stops, and the console typewriter types:

IC		PR
1450	(tab)	+4521003112

Note that the instruction counter contains the location of the next instruction to be executed, not the location of the instruction presently in the program register.

FIGURE 274. IBM 7150 CONSOLE

In a branch instruction, the instruction counter contains the location of the next sequential instruction, regardless of whether branching is to take place. Take, for example, a halt-and-branch instruction, in location 0749:

S01	23	45	6789
+00	00	00	2451

The program stops, and the typewriter types:

IC		PR
0750	(tab)	+0000002451

The ADDRESS light on the operating panel is on, indicating that the operation is a branch. When the program is re-started, the branch address 2451 will be transmitted to the instruction counter, and the next instruction will be taken from that location.

Operating Panel

Figure 275 is a picture of the operating panel. The surface of the panel is black, enabling a light to stand out clearly when it is ON; when an indicator light comes ON, the printing is illuminated against the black background. On the right are the main power keys and lights. At the left are illuminated keys for the alteration switches, and accumulator and exponent overflow. In the center are the control lights and the dials for address stop and unit-record priority control.

Power Keys and Lights (Figure 276)

The power keys and lights are located at the right of the operating panel. They are the main ON and OFF operating features in the IBM 7070 system.

FIGURE 275. OPERATING PANEL

EMERGENCY PULL: This should be used *only* in case of emergency, when all power must be shut off immediately to prevent injury to an individual or damage to the machine. If this device is used to turn off power, only a customer engineer should turn on the power again.

Pulling this switch removes all power from all units except the disk files and unit-record equipment. Direct current only is removed from the disk files. The unit-record equipment is not affected by this switch.

POWER ON: This is an illuminated key. Pressing it provides full operating power to the 7070 system, either from a power-off condition or dc-off condition. When the key is pressed, its light comes on. It indicates that power has been supplied to the system and is turned off only by pressing the *power-off key*.

READY: When the 7070 is ready for operation, this light comes on. It takes a short time for the machine to be ready after the power has been turned on, because power must be supplied to various components of the system in a specified sequence. The *ready* light comes on immediately if *power on* is pressed from a dc-off condition.

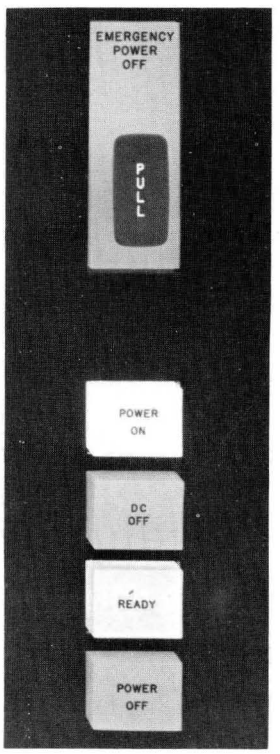


FIGURE 276. POWER KEYS AND LIGHTS

DC OFF: Pressing this key turns off the dc power only. It is used when the machine is to be idle for a short time. When this key is pressed, the *ready* light turns off, but the *power on* key-light stays ON. Full power is restored immediately by pressing the *power on* key.

POWER OFF: This key is the means of turning off the machine. When it is pressed, the *ready* light and the *power-on* key-light turn off. The 7070 is turned on again by pressing the *power-on* key.

Alteration Switches and Overflow Keys (Figure 277)

These features are on the left of the operating panel. They afford control by the console operator over certain features in the stored program.

ALTERATION SWITCHES: The top four keys on the left side of the operating panel are the alteration switches. They are illuminated keys, which come ON when the key is pressed. Pressing one of these keys turns on the corresponding alteration switch, which can be interrogated by the stored program instruction +51, BAS. Each key latches in its pressed position. To turn off an alteration switch, press the key. It will unlatch from its pressed position and return to its normal position, and the light will turn off.

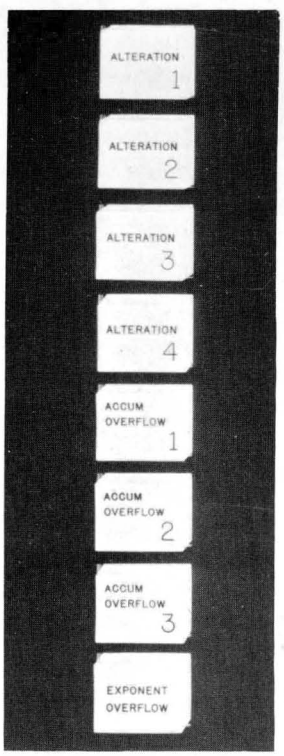


FIGURE 277. ALTERATION SWITCHES AND OVERFLOW KEYS

ACCUMULATOR OVERFLOW: Like the alteration switches, these keys latch when they are pressed and are turned off by being pressed again. Each key illuminates while it is ON. The accumulator overflow keys are the means of determining whether an overflow in an accumulator will set the internal *overflow* indicator for that accumulator for later interrogation by the program, or will stop the machine as well, when it occurs. If the key for an accumulator is not pressed ON, an overflow in that accumulator stops the machine. If the key is ON (illuminated), an overflow does not stop the machine, but turns on the *overflow* indicator for that accumulator. This indicator can be tested later in the program by a BV# instruction: +11 (BV1), +21 (BV2) or +31 (BV3).

The accumulator overflow *lights* on the right of the panel are associated directly with the *overflow* indicators for the accumulators. Each light indicates that the corresponding *overflow* indicator is on. The light goes out when the indicator is turned off. Thus, an accumulator overflow light indicates either why the machine has stopped, if its accumulator overflow key-light is OFF; or that the program is continuing after it has caused an accumulator to overflow, if its accumulator overflow key-light is ON.

EXPONENT OVERFLOW: This illuminated key has the same function for exponent overflow in floating-decimal arithmetic operations that the accumulator overflow keys have for normal overflow. Exponent overflow occurs when an operation attempts to develop a modified characteristic greater than 99.

Control Lights

The lights on the operating panel are called control lights, rather than display lights, because they don't display the contents of storage words or registers. They indicate the status that the system is in, the components that are in operation, and the type of error detected by the checking features of the system.

Figure 278 shows all of the control lights ON (normally, only a few of them are ON).

STATUS

The 7070 is always in the normal or priority status. Normal status is when the main program is operating. Priority status is when a priority routine is functioning. Inquiry is the status that is set by pressing the *inquiry only* key on the operating keyboard. In normal

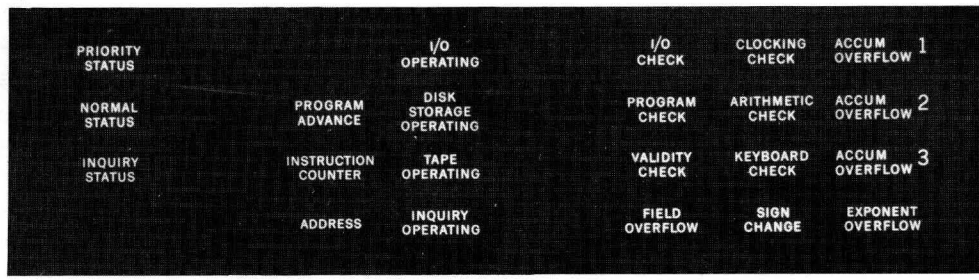


FIGURE 278. CONTROL LIGHTS

status, an inquiry from any one of the inquiry stations sends a priority signal, and the program branches to location 0106 or 0107, depending on the inquiry control group. In inquiry-only status, the program is not operating. An inquiry starts its inquiry routine at word 0106 or 0107.

OPERATING

These lights indicate which types of components are put into operational status. There are five component types: the programming unit, the unit-record machines, the disk files, the tape units, and the inquiry stations. In the case of all but the programming unit, the appropriate light flashes briefly when an instruction involving the operation of one of the units of that type is sent to the program register.

These lights serve their main purpose when an operation involving a particular type of unit can't be executed. This may be due to machine error, programming error, or normal operations, (the card reader having run out of cards, for example).

PROGRAM ADVANCE: The function of this light is a little different from that of the other lights. It is ON whenever stored-program instructions are being executed. If the program stops for any reason other than pressing the *stop* key on the operating keyboard (if it stops by itself, due to accumulator overflow, HB or HP operation codes, validity check, etc.), this light flashes ON and OFF, calling the console operator's attention to the fact that the program has stopped. The operator can stop the flashing light by pressing the *stop* key on the operating keyboard.

I/O: This light comes on whenever an operation involving a unit-record machine is sent to the program register. Note that it must be an actual operation; the US code (unit-record signal) is not included. All of the read, write, punch, and type codes are included: UR, UW, UWIV, UP, UPIV, and TYP.

DISK STORAGE: This light flashes whenever a disk seek, read or write instruction is sent to the program register (codes ± 91, ± 92, and $+95$). The code for release (-95, DAR) has no effect.

TAPE: Any instruction that involves the operation of a tape unit turns on this light when it is in the program register. Not included are tape select (TSEL), skip (TSK) and turn off end of file (TEF).

INQUIRY: Whenever a $+54$ inquiry-control instruction (QR or QW) is in the program register, this light comes ON.

ADDRESS AND INST COUNTER: Whenever the stored program stops, one of these lights is on. If the next instruction in the stored program is to come from the instruction counter, the inst counter light is on. This light is on even if the next instruction is one or two locations beyond the next sequential location, as in a UR operation not involving validity-check or end-of-file. The address light is on if the next instruction is to come from the address in the instruction, in a branch operation. This could be an unconditional branch, or the result of a test, such as BM3 when accumulator 3 is minus.

CHECK

A checking light comes ON whenever the IBM 7070 stops processing, due to the discovery of an error by an automatic checking feature. The error may be caused by incorrect programming, operation, control-panel wiring, or by the machine itself. (Whenever the machine stops due to a checking feature, the *program advance* light blinks, as previously described.)

I/O CHECK: An error in operation of any of the machines that transmit data to and from the synchronizer drum causes this light to come ON, and the 7070 to stop. Human errors that cause this include failing to fill the input synchronizer because of improper control-panel wiring; or improper alphabetic-input wiring, detected by the automatic alphabetic check. Machine checks include validity check on transmission of data between the synchronizer drum and the synchronizer register, or

any check on the operation of the input/output units.

CLOCKING CHECK: This light indicates a failure in the internal timing circuits of the 7070. It denotes a machine error only, not a programming or operating error.

The correction procedure after a clocking check is important, however. In all cases, the program should be re-started back at some previous point. A clocking error may affect totals that are being updated or processed, and the operator must restart the program at a point that again starts development of those totals.

PROGRAM CHECK: Either a programming error or a machine error can cause this light to come on. Example of programming errors are an invalid address, an invalid operation code, an alpha sign for a program instruction, etc. A machine error occurs if it fails to process a correct operation code.

ARITHMETIC CHECK: This light indicates an error in the arithmetic circuits. It may be a machine error, or programming error, such as divide overflow. Like the clocking check, restart after detection of this type of error should revert the program to a point that starts the development of the totals involved when the arithmetic error was detected.

VALIDITY CHECK: The detection of an invalid character on any transfer of data that the stored program must complete before being able to continue, turns on this light. This includes flow of data to and from magnetic-core storage, to and from the registers and accumulators, and to and from the synchronizers on the drum. (Validity checks on data flow to and from the magnetic-tape units and the disk files are indicated by the final-status words. Validity checks between the synchronizer drum and the card units and between inquiry stations and inquiry control synchronizers are indicated by the I/O check light.)

KEYBOARD CHECK: This light is used to indicate an error in operation of the console typewriter keyboard. When the console typewriter is being used to enter data manually into the 7070, only the numerical keys can be used. (Alpha data must be entered in the two-digit numerical code.) If an alpha key is pressed while the console is in manual-entry status, the keyboard check light comes ON. If the operator presses too many keys or too few keys in an alter operation, this light comes ON.

FIELD OVERFLOW: Whenever a field overflow occurs on a store or add-to-storage type of operation, the *field-overflow* indicator is turned on, and the machine either continues or stops, depending on the setting of the internal field-overflow stop-sense switch. Whenever the *field-overflow* indicator is ON, regardless of the setting of the stop-sense switch, the field overflow console light is on. Thus, it indicates either why the machine has stopped (switch at stop) or that the program is continuing after it has caused a field overflow (switch at sense). The instruction BRANCH IF FIELD OVERFLOW, BFV, turns off the indicator if it is ON, and turns off the field overflow light as well.

SIGN CHANGE: In the same manner as field overflow, this light indicates that the *sign-change* indicator is ON; a sign change has resulted from a store or add-to-storage type of operation. The BRANCH IF SIGN CHANGE instruction, BSC, turns off the indicator and the sign change light.

ACCUMULATOR OVERFLOW AND EXPONENT OVERFLOW: The function of these lights is explained in the text on *Overflow Keys*.

ADDRESS STOP (FIGURE 279)

There are four dials for address stop, located in the center of the operating panel. Each dial has a digit setting (0-9) and can be turned to any digit. The setting of each dial is shown in the small window above the dial.

The operator can have the program stop at a desired address by setting the address stop dials at that address, and pressing the address stop key on the operating keyboard. The stored program continues normally until it comes to that address, for either instruction or data. If it stops because of an instruction address, the stop occurs before that instruction takes place. If the stop results from a data address (positions 6-9), it comes immediately after the instruction involving the address has been executed. In all cases of address stop, the address is the one after indexing, if indexing is used.

With the index-word codes, the operand index word (specified by positions 4-5 of the instruction), causes an address stop if its address is set in the dials (0001-0099). For a code whose address represents the first of a series of record-definition words, address stop is effective only for the first RDW—the one addressed by the program instruction itself.

If positions 6-9 of an instruction are the same as the address in the address stop dials, the program stops only if positions 6-9 actually represent a usable address. For example, a halt and proceed (−00, HP) instruction cannot cause the program to stop on address stop, because positions 6-9 are not used for an address. Similarly, the index-word codes that use positions 6-9 as a 4-digit factor cannot cause an address stop.

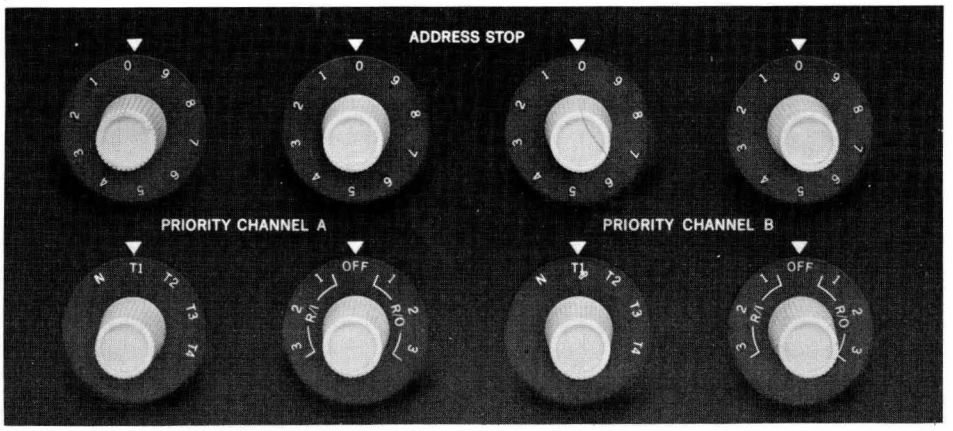


FIGURE 279. ADDRESS STOP AND UNIT-RECORD PRIORITY CONTROLS

UNIT-RECORD PRIORITY CONTROL (FIGURE 279)

These dials are the means of assigning unit-record priority controls A and B to the desired card readers, punches, or printers. The right-hand dial under each control (A and B) has seven positions, one designating each of the six (maximum) unit-record machines, and one for OFF. When a unit-record machine is thus designated, the completion of a cycle (read, punch, or print) automatically sets the priority stacking latch for unit record A or B. If the stacking latch has not been masked, the program branches to a subroutine, as a result of the unit-record cycle having been completed (see section on *Automatic Priority Processing*).

The left-hand dial under each control is for tape channel 1, 2, 3, 4, or none. If a tape channel is specified, the stacking latch is set only when the unit-record machine has completed a cycle, and the specified channel is available for use. This increases the efficiency of card-to-tape, tape-to-printer, etc. operations that use the automatic priority processing feature.

Console Typewriter

The typewriter on the IBM 7150 Console is used for five different purposes:

1. Typing output data under control of the stored program.
2. Automatic display of the contents of the instruction counter and program register whenever the stored program is stopped.
3. Displaying the contents of a core-storage word or addressable register.
4. Changing the contents of a core-storage word or addressable register.
5. Manual typing, independent of the rest of the 7070 system.

The flow of data between console typewriter and core storage is as follows: Typed information is recorded in the synchronizer register as each digit is typed (see Figure 8). A word is moved from the synchronizer register to the arithmetic register and thence to core storage. Data from core storage to the console typewriter is transmitted to the arithmetic register, thence to the synchronizer register, and from there to the typewriter.

A typewriter interlock is provided to prevent a data movement to the synchronizer if the typewriter has not completed printing on a previous type operation.

Forty-four characters are available for typing: 26 letters, 10 digits, and 8 special characters: plus sign (+), minus sign (—), comma (,), period (.), slash (/), asterisk (*), dollar sign ($), and pound sign (#). Any character other than these 44 is typed as a pound sign (#).

As shown in Figure 280, the keyboard is the same as that of other IBM electric typewriters. The alpha keys are used only when typing independently of the rest of the 7070 system (except the A for an alpha sign, as described under *Display*). A pin-feed platen is provided, to enable multi-carbon paper forms to feed accurately.

TYPEWRITER OPERATIONS UNDER PROGRAM CONTROL

Data is transmitted from core storage to the console typewriter under control of record-definition words, as described under *Block Transmission*. The stored program continues with the next instruction (see Page 119) after the typing operation is completed.

An automatic carriage return is performed before starting printing when a TYP command is executed. This assures printing starting in position 1.

Typing is in sequence, starting from the high-order position of the first word defined by the first RDW and continuing to the units position of the last word defined by the last RDW. Carriage return at the end of each line is automatic.

The sign position of a word is scanned first, and on numerical words, a dash (−) is printed for negative words and a plus sign for positive words. An alpha sign is neither printed nor spaced but sets up circuits for conversion of two-out-of-five code to alpha.

If the word is numerical, the sign and digit positions 0-9 are typed in that order. A space follows position 9. Thus, a series of numerical words are printed in 11-character groups (sign and 10 digits), separated by single spaces.

On alpha words the converted alpha characters are printed in sequence beginning with positions 0-1. The contents of a second alpha word are printed directly following the previous one, with no spaces for sign. Thus, alpha data words are printed out in a continuous series. Characters coded 00 in alpha words cause a space.

Operating Keyboard

The operating keyboard (Figure 280) is located to the right of the console-typewriter keyboard. It contains the console operational features that are used for program testing or for operator control of data and instructions. For the purpose of explanation, the keys are grouped in three categories:

1. Status keys: *Run, Address Stop, Single Cycle* and *Inquiry Only*
2. Control keys: *Start, Stop, Computer Reset, Check Reset* and *Program Reset*
3. Operation keys: *Display, Alter, Store, Log* and *Type Reset*

Status Keys

The status keys are located at the top of the operating keyboard. They are latch type—at all times, one is in its pressed position and the other three are not. When one of these keys is pressed, the key that had been pressed previously automatically resets to its off position. Thus, the 7070 is always in *run, address-stop, single-cycle,* or *inquiry-only* status.

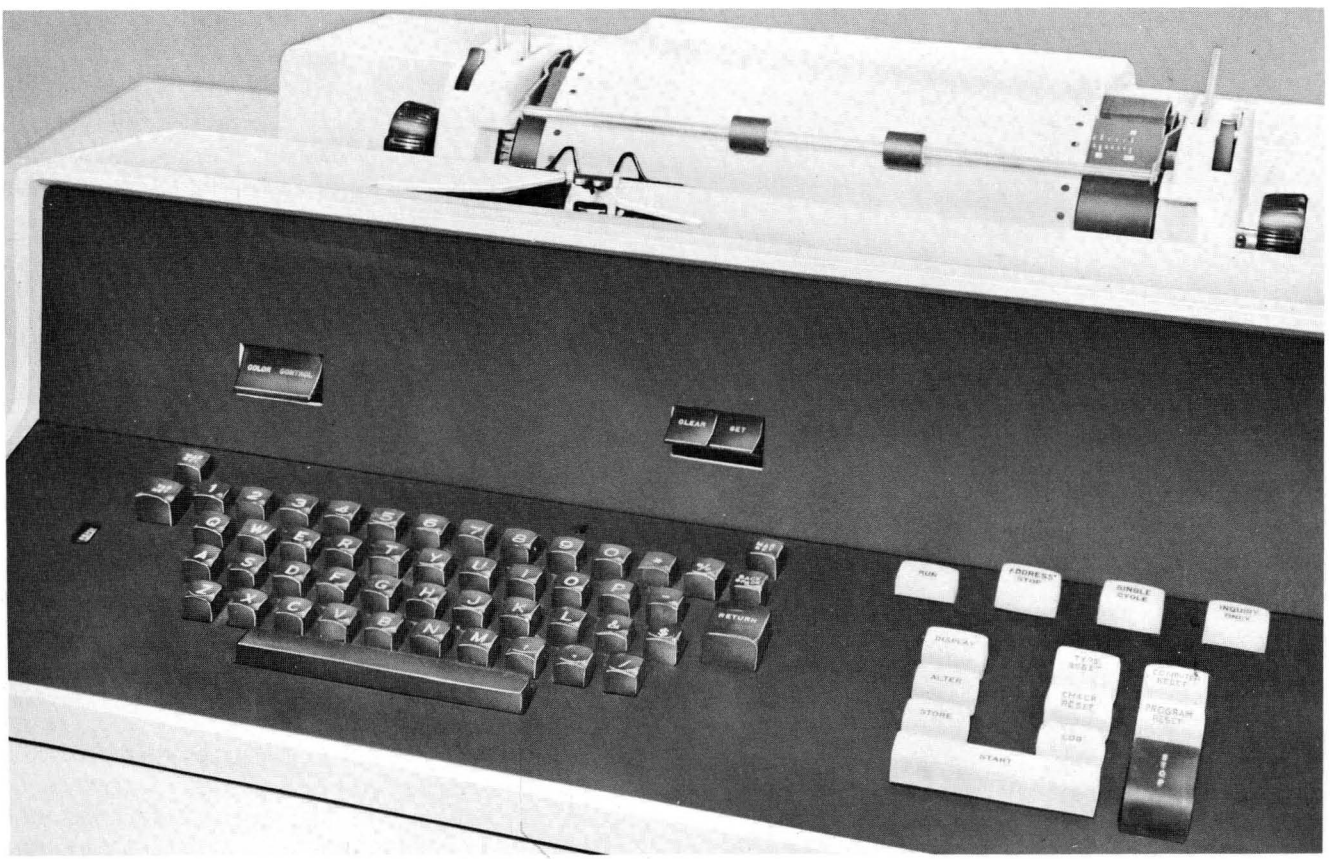


FIGURE 280. CONSOLE TYPEWRITER KEYBOARD AND OPERATING KEYBOARD

RUN: When this key is pressed, the stored program runs continuously in a normal manner. This can be considered "normal" status, the one used almost exclusively once the program has been tested and is performing on a regular basis.

ADDRESS STOP: In this status, the stored program runs continuously until it uses the address set on the address stop dials in the operating panel. This address can be for either instruction or data; the stop occurs when the address enters the program register, from either the instruction counter, or storage (after indexing). If it is an instruction address, the program step is the next one to be executed in the program. If it is the address of data, the stop occurs immediately *after* that instruction has been executed. When an address stop occurs, the contents of the instruction counter and program register are automatically typed.

SINGLE CYCLE: Each pressing of the program start key causes the stored program to execute one instruction if the 7070 is in this status. After the instruction is completed, the contents of the instruction counter and program register are automatically typed.

INQUIRY ONLY: When this key is pressed, the stored program does not function, but an inquiry can be processed. A priority signal from an inquiry station starts an inquiry subroutine at either 0106 or 0107, depending on the inquiry-control group involved.

The following is the sequence of operations for using the inquiry-only feature.

1. Stop the stored program by pressing the stop key (contents of the instruction counter and program register are typed).

2. Press the inquiry-only key. The inquiry-only status light on the operating panel turns on.

3. Press the start key. The 7070 goes into an idle condition, and the program-advance light flashes ON and OFF.

4. Release of an inquiry causes the contents of the instruction counter to be stored in positions 2-5 of index word 97, the inquiry routine to start in 0106 or 0107, and the priority status light on the operating panel to come on. (The instruction counter contains the location of the next stored program instruction, as typed in paragraph 1.)

5. If the priority-release instruction at completion of the inquiry sub-routine has an address of 0097, the address in positions 2-5 of that word is returned to the instruction counter, the 7070 returns to the idle condition, and the priority status light on the operating panel turns off. If the address of the PR instruction is other than 0097, the address itself goes to the instruction counter.

When the machine is taken out of the inquiry-only status, and the program is started again, it starts at the point where it had left off, whether or not any inquiry sub-routine had been processed.

Control Keys

The control keys are used to start and stop the program and reset the various registers and error-detection circuits prior to starting. The start and stop keys are at the lower portion of the keyboard. Computer reset and program reset are above the stop key, and check reset is beside the program reset key.

START: Pressing this key starts the stored program. The first instruction is taken from the location in the instruction counter at that time. In single-cycle status, pressing this key causes one program instruction to be executed.

STOP: Pressing this key stops the stored program. The instruction being executed at the time is completed, and the contents of the instruction counter and program register are automatically typed. The stop key is also used to stop the program advance light from blinking; any time the program stops other than by pressing the stop key, the program advance light starts blinking.

CHECK RESET: The function of the check reset key is to reset a detecting circuit that has stopped the 7070. This may be due to a machine error, or to a situation that has arisen in the stored program, which caused the program to stop.

Any time that one of the checking lights on the operating panel comes ON, the 7070 stops. (The functions of these lights are explained in the section on the *Operating Panel.*) Pressing the check reset key turns off the light and resets the error-detecting circuits. The program can be continued from that point by pressing the start key.

As described in the *Operating Panel* section, use of the accumulator overflow key determines whether an accumulator overflow will set an indicator and stop the machine, or merely set the indicator and allow the program to continue. If an overflow stops the machine, pressing CHECK RESET turns off the indicator and allows the program to continue, when the start key is pressed. The *exponent overflow* indicator, used in floating decimal operations, also has a stop/sense key on the operating panel, and check reset has the same function for this indicator as for accumulator overflow.

The *sign-change* and *field-overflow* indicators are made to stop the machine or not when they are turned on, by stored program instructions, rather than by keys on the operating panel. The function of check reset in the case of these two indicators, however, is the same. If the *sign-change* indicator, for example, is in the halt mode and a sign change occurs, the machine stops, and pressing CHECK RESET turns off the indicator. Check reset does not change the indicator to the sense mode; only the SMSC instruction can do that. Field overflow works in the same way as sign change. Note that the sequence of an indicator being put in the halt mode (HMSC, HMFV), and being set, could be reversed. An indicator can be in the sense mode, and ON; if it is then put into the halt mode, the machine stops, the console light comes ON, and check reset must be used.

The *high, low,* and *equal* indicators are reset by pressing the check reset key, if none of them, two of them, or all three are ON. For the normal condition of one of them being ON, check reset has no function. (A compare operation automatically resets all these indicators and then turns on one of them as a result of the comparison.)

COMPUTER RESET: The computer reset key resets the program controls and resets all registers except the accumulators. After this key has been pressed, the program should be started over again, because virtually all of the results of the program are destroyed. The following is a list of the results of pressing the computer reset key.

1. Forces check reset
2. Resets any of the indicators that may be ON:
   Accumulator overflow
   Floating-decimal overflow and underflow
   Sign change
   Field overflow
   High, low, and equal compare
3. Resets any of the stacking latches that may be ON and the priority waiting latch, if it is ON.
4. Sets the priority masks (so that all are masked)
5. Resets the priority mode latch
6. Resets all access-arm availability latches
7. Sets the instruction counter to 0000
8. Sets the program register to zeros
9. Sets the arithmetic register and the auxiliary register to zeros
10. Resets the system from priority status to normal or inquiry-only status, depending on which status key is in its pressed position.
11. Sets the field-overflow and sign-change sense/stop switches to stop
12. Resets the end-of-file latch in the IBM 7500.
13. Resets all three accumulators to plus zeros.

It can be seen that a complete program re-start is required after pressing COMPUTER RESET.

PROGRAM RESET: Pressing this key performs the same function as pressing COMPUTER RESET, with the exception of numbers 7 and 8. The contents of the instruction counter and program register are unchanged when PROGRAM RESET is pressed. Thus, all of the things that may have been affected by the program are reset, but the program can be continued at the point where it left off.

Operation Keys

These keys are the means by which an operator can display the contents of any core-storage location, by having it typed on the console typewriter. He can also change a storage word by using these keys.

The display, alter, and store keys are operative only when the stored program is *not* running. These keys are located on the left of the operating keyboard. The type reset key is above the check reset key. The log key, which can be used when the program is running or not, enables the operator to use the console typewriter without affecting the rest of the 7070 system. It is located below the check reset key.

To display the contents of a storage word, the operator merely types the 4-digit address. If the operator did not hit tab or carriage return first, this address is typed directly below the IC contents that were typed when the program was halted. There is an automatic tab, and the contents of that address are typed (directly below the contents of the program register, if tab or carriage return had not been used). If the word is numerical, it is typed as the sign (+ or −) followed by the 10 digits. If it is alpha, the five alpha characters are typed without sign indication:

Location		Contents
(typed by operator)		
1234	TAB	±1234512345 (if numerical) CR
2345	TAB	ABCDE (if alphabetic) CR

There is an automatic carriage return after the word is typed.

DISPLAY: The display key is used to display an alphabetic word in its 10-digit core-storage notation, after it has been displayed as five characters. Assume, for example, that the five characters JAN 15 in word 8245 have just been displayed:

8245 TAB JAN15 CR

The operator presses the display key. There is an automatic tab, and the 10 digits of word 8245 are typed, preceded by an A for alpha:

TAB A7161759195 CR

ALTER: When a word has been displayed, it can be changed by the operator if he first presses the alter key. This causes an automatic tab and a ribbon shift to red, and the operator types the sign and ten digits that are to go into that word. The complete word must be typed in numerical digits. An alpha word need not be displayed numerically but can be altered numerically only.

STORE: After visually checking what he has typed, the operator presses the store key, and the new data is stored in the core-storage word. If there were no errors, such as validity errors, a red pound sign (#) is typed at the right of the 10 digits, and a carriage-return is executed.

LOG: When the stored program is not operating, and it is desired to use the typewriter for normal typing, press this key. This prevents the automatic display of the storage word the address of which is the first four digits typed, or keyboard error if an alpha key is pressed. Typing can continue until the stop key is pressed, resetting the typewriter to normal condition.

When the stored program is operating, manual typing can be performed without pressing the log key. If a stored-program TYP instruction is given while manual typing is taking place, the program instruction takes precedence, and the typewriter is removed from manual control. After the programmed type operation is completed, the operator can again resume manual typing.

The alphabetic keys can be used manually only after pressing the log key. Any other use of the alpha keys results in a keyboard error (except the A for an alpha sign in an alter operation).

TYPE RESET: Any time that the typewriter is typing from a TYP instruction, pressing this key stops the typing operation at the end of the word being typed. A red * is typed, the carriage returns, and any interlocks set up by the typing operation are released. The contents of the instruction counter and program register are then typed.

Checking

The checking features of IBM 7150 Console are in two main categories, checks on machine operation (validity checks), and checks on manual operation of the typewriter keyboard. An error in keyboard operation may be detected by the machine or by the operator himself. The transmission of data from the 7070 to the console does not, of course, involve operator errors but is checked for validity. All of the checking features are discussed here in the sequence of operations to display, alter, and store a word of data. Validity checking of all movement of data from the 7070 to the console typewriter is then discussed in a section by itself.

In all cases of an error being made when the operating keyboard and the typewriter keyboard are used to display and/or alter a storage word, the operation must be started over again. This is done by first pressing the stop key, to reset the error-detection circuits, and again displaying the storage word by typing its address. In keying in an address, or in altering a word, only the numerical keys on the typewriter can be used. If an alphabetic key is pressed in either of these operations, the keyboard locks, and the stop key must be pressed.

If the operator discovers that he has made an error in typing the address of a word to be displayed, he presses the stop key. There is an automatic carriage return, and he can attempt again to type the correct address. The error must be detected before the last digit of the address has been typed; once four digits have been keyed in, the contents of that word are immediately and automatically typed.

If there is a validity error on transmission of a digit from the typewriter to the arithmetic register, the keyboard locks. The operator presses the stop key in order to be able to start again.

The operator may discover that he has made an error in typing the sign and ten digits of a word in an alter operation. He presses the stop key, the typewriter automatically types a red asterisk (*), and carriage returns. The operator must then start over by keying in the 4-digit address of the word to be altered.

A typing error undetected by the operator may occur if less than or more than ten digits and sign are typed in an alter operation. In either case, pressing the store key does not store the typed data in the storage location, but locks the keyboard instead. Pressing the stop key then causes typing of a red asterisk and a carriage return.

The movement of data from the arithmetic register to core storage when the store key is pressed is checked for validity. If an invalid character is detected, a red asterisk is typed, but there is no carriage return. The validity check light on the operating panel turns on, and the error reset key must be pressed, before the stop key is pressed.

Bit Typing on Validity Check

As described in the section on the TYP operation code, there is a validity check on data moved from core storage to the arithmetic register, and on movement of data from the arithmetic register to the console typewriter. Typing always takes place, except in the case of a validity error on transmission from cores by the TYP instruction. If an invalid character is detected between the arithmetic register and console typewriter for a TYP command, bit typing of the invalid character automatically takes place.

Bit typing is the typing of the bit representations (01236) of the invalid digit. It is preceded by an asterisk and followed by the letter D, all automatically typed in red. Assume for example, that +0123456789 in storage is to be typed, but the digit 6, which should be made up of the 0 and 6 bits, has an extraneous 3 bit. The typed word appears as follows:

+012345*630D789
red

If there are no bits, only the * and the D are typed.

The purpose of the D is to define it as a digit. In an alpha word, positions 0, 2, 4, 6, and 8 are identified by Z for zones, and positions 1, 3, 5, 7, and 9 are identified by a D for digits. If either digit of an alpha character is invalid, both the zone and digit are bit-typed. Assume an alpha word containing ABCDE, @6162636465 in core storage, in which the 3 digit of the C contains an extraneous 2 bit added to its 0 and 3 bits. The word types as follows:

AB*60Z320DDE
red

The bits of both the 6 and the 3 of the 63 are typed, even though only the 3 was invalid.

Executing an Instruction from the Console

It is often desirable, in program testing, to create an instruction from the console and then have the stored program execute it. This is done as shown in Figure 281.

xxxx TAB	±xxxxxxxxxx	Contents of instruction counter and program register automatically typed on machine stop.
9995	±xxxxxxxxxx	Display the program register, repeating the type-out of its contents.
	±yyyyyyyyyy#	Alter the program register, by pressing Alter, typing the sign and ten digits, and pressing Store. #Indicates it was stored correctly.
9999	xxxx	Display the instruction counter. (Note only 4 digits, with no sign, are typed.)
	9995#	Alter the instruction counter, giving it the address of the program register. #Indicates it was stored correctly. Press the Start key.

FIGURE 281. EXECUTING AN INSTRUCTION FROM THE CONSOLE

If the instruction counter contains an address of 9995, it does not increase with each instruction—it stops at 9995. Thus, if the program is *run* status, it continuously performs a one-instruction loop. If the status is single-cycle, the instruction is executed once for each pressing of the start key.

Index

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York

Printed in U. S. A. A22-7003-1