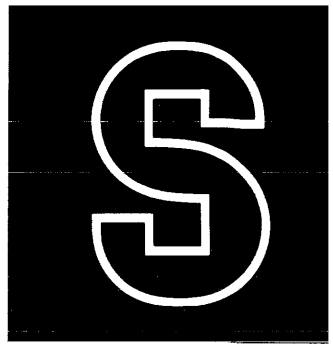

Reference Manual

Flow Charting and Block Diagramming Techniques

IBM Reference Manual

Flow Charting and Block Diagramming Techniques

Copies of this and other IBM publications can be obtained through IBM Branch Offices. Address comments concerning the contents of this publication to IBM Technical Publications Department, 112 East Post Road, White Plains, New York

Contents

Illustrations

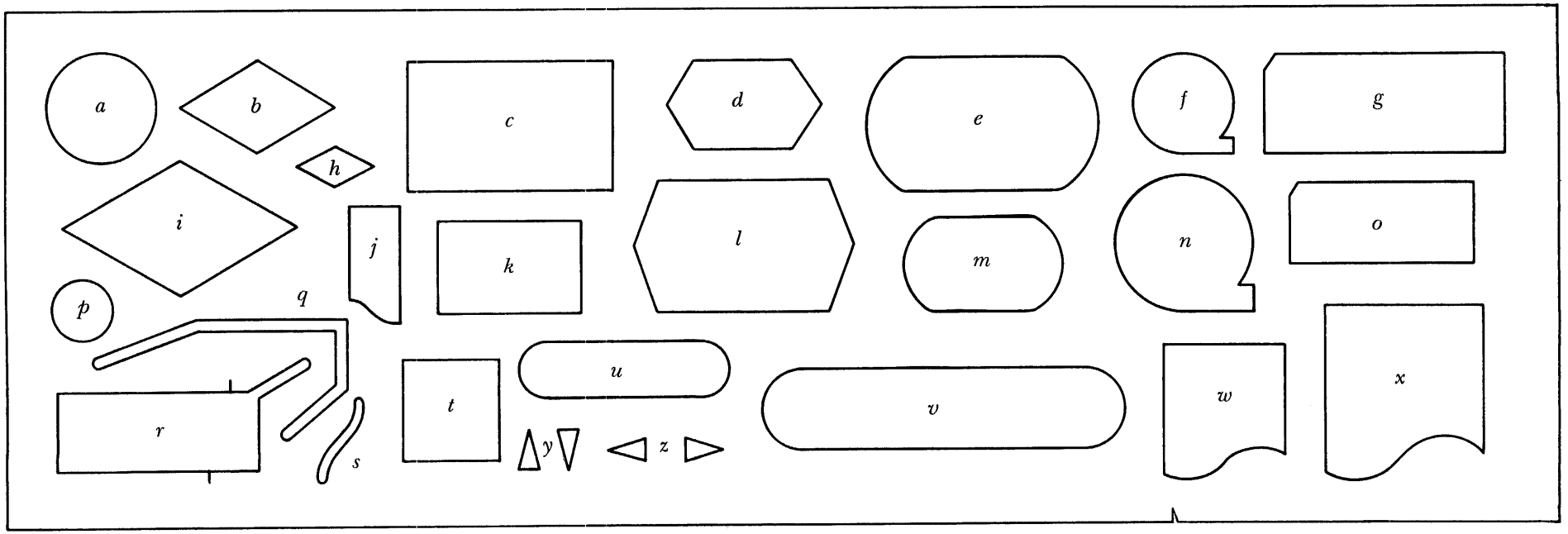

Figure 1. IBM Charting and Diagramming Template, Form X24-5884-6

Introduction

The old cliché which so strongly points up the quantitative relationship between words and pictures has particular truth in the development and documentation of data processing methods and procedures. The systems analyst finds use for many types of pictorial representations, including forms layouts, control panel diagrams, manpower loading charts, etc. The two to be discussed here are the work-flow chart and the block diagram.

A work-flow chart (or simply flow chart) is defined as a graphic representation of the *system* in which data provided by a source document is converted to final documents. (It should be noted here that in its broadest meaning a document is "any instrument conveying information." Such a definition is used in this discussion in order to include punched cards, magnetic tape and paper tape in the same category as printed forms.) A flow chart, then, provides a picture of the data processing application from the standpoint of what is to be accomplished. Such a picture gives primary emphasis to the documents involved and secondary emphasis to the work stations through which they pass.

A block diagram is defined as a graphic representation of the *procedures* by which data is processed within a system. In this picture the emphasis is on the operations and decisions necessary to complete the process.

To put it another way, a flow chart serves to show *what* job is to be done, while a block diagram shows *how* a job is done. This interpretation is derived from the fact that flow charts are frequently composed solely of symbols representing the form in which data appears at various stages in a process. The actual operations which must be performed are indicated only briefly or not at all.

In order to encourage standardization in the use of symbols, thereby simplifying the problem of interchanging information between IBM and its customers, IBM has made available the Charting and Diagramming Template shown in Figure 1. The cutouts can be used to draw symbols representing clerical functions, unit record machines and functions, data processing systems and functions, and various types of documents.

In the pages that follow, the standard symbols used in flow charts and block diagrams will be described. When preparing charts and diagrams for publication or presentation, it is common practice to introduce more artistic variations of the symbols. One frequent variation is the use of photographic reductions representing the actual forms and machines used in a procedure.

Flow Charting

Direction of Flow

The basic element of the flow chart (and the block diagram as well) is the line and arrow (Figure 2) indicating DIRECTION OF FLOW *(y, z)**. In order for one symbol to have a meaningful relationship to other symbols in a chart it is necessary that it be connected to one or more other symbols in such a way as to indicate the sequence in which the operations occur or the disposition of the documents.

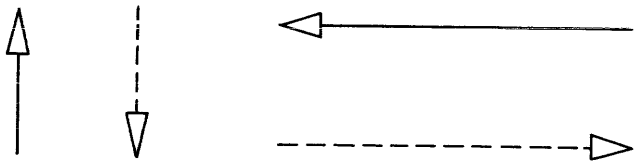

Figure 2. Direction of Flow

In flow charting, solid lines are normally used, but occasionally it is desirable to differentiate between the physical movement of work and the mere transfer of information. This can be done by using solid lines in the first case and dotted lines in the second. For example, dotted lines are used to illustrate accounting control functions.

Flow charts are constructed to conform to the natural tendency to read from left to right, top to bottom,

*Letters in parentheses following the name of a symbol refer to the location of the symbol on the template as illustrated in Figure 1.

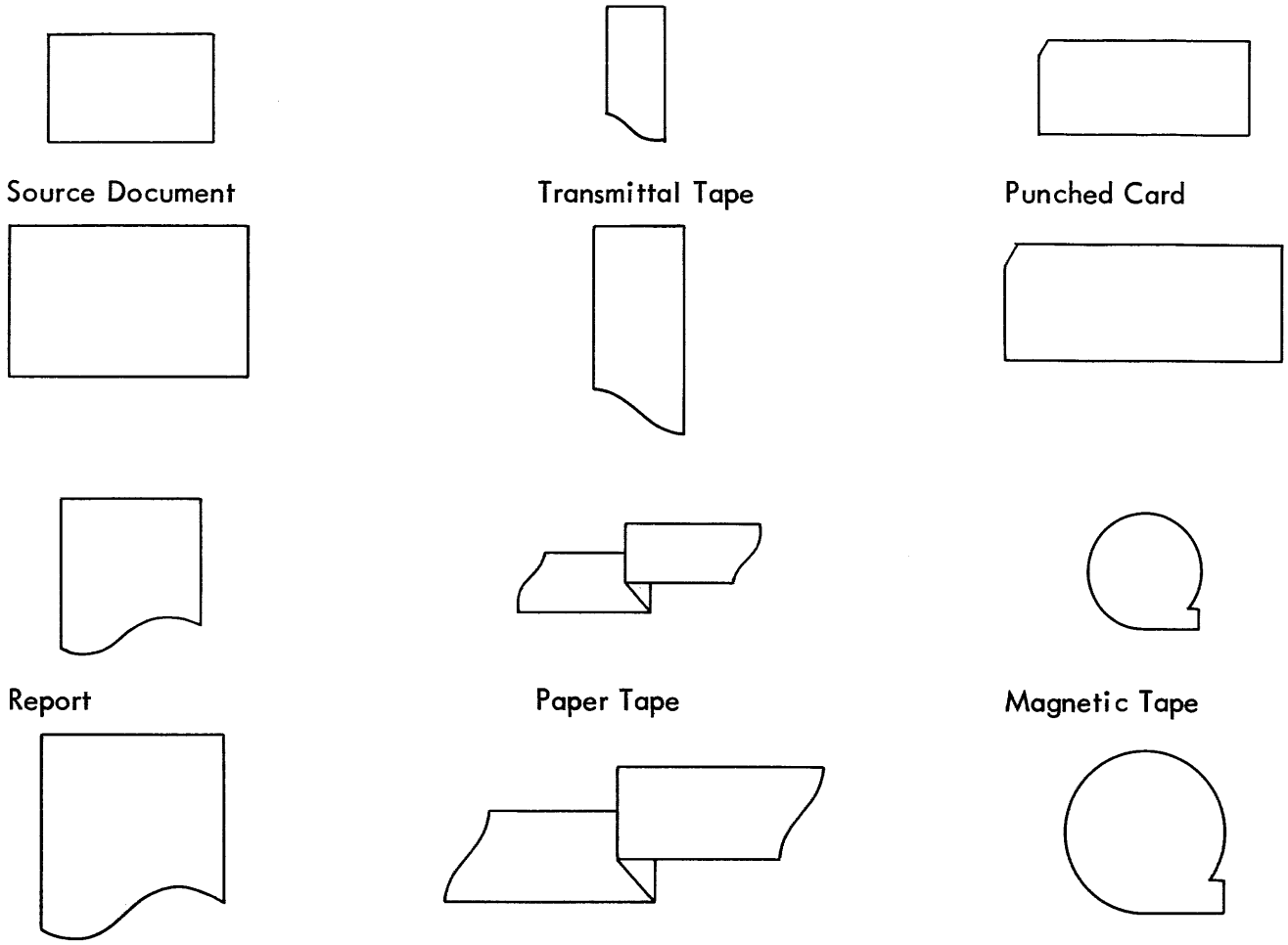

Figure 3. Document Symbols

though variations are sometimes desirable in order to achieve symmetry and to emphasize certain points. Within this framework, flow lines can be drawn horizontally, vertically and diagonally. The primary considerations are neatness and uniformity.

Document Symbols

The various types of documents encountered in data processing applications are represented by the symbols shown in Figure 3. Two sizes of each are provided. Normally, a single diagram will use either the small or the large symbols but not both.

The SOURCE DOCUMENT *(c, k)* symbol indicates any paper document from which information is extracted during a process. The REPORT *(w, x)* symbol, on the other hand, indicates a paper document prepared during a process. When a document serves both purposes (e.g., a permanent ledger card to which information is posted), the source document symbol is used.

The TRANSMITTAL FORM and PAPER TAPE *(j, r* plus *s)* symbols are constructed from the same cutouts. In drawing the large transmittal form, that portion of cutout "r" marked off by arrows provides the top and sides of the form, and cutout "s", the bottom. (The two "folds" of the paper tape are obtained from the transmittal form symbol by rotating the template 180°. A short diagonal line connecting the lower corners of the two halves gives the illusion of a folded strip of tape.) The transmittal form normally represents an adding machine tape or some other type of batch control sheet accompanying other forms.

The two remaining document symbols are the PUNCHED CARD *(g, o)* and MAGNETIC TAPE *(f, n)* symbols.

When document symbols appear in a flow chart, the symbols are usually labeled to indicate the type of information contained therein. Frequently, the type of form is further indicated by drawing the general outline of the form. Figure 4 shows several ways in which this can be done. Included are ways in which

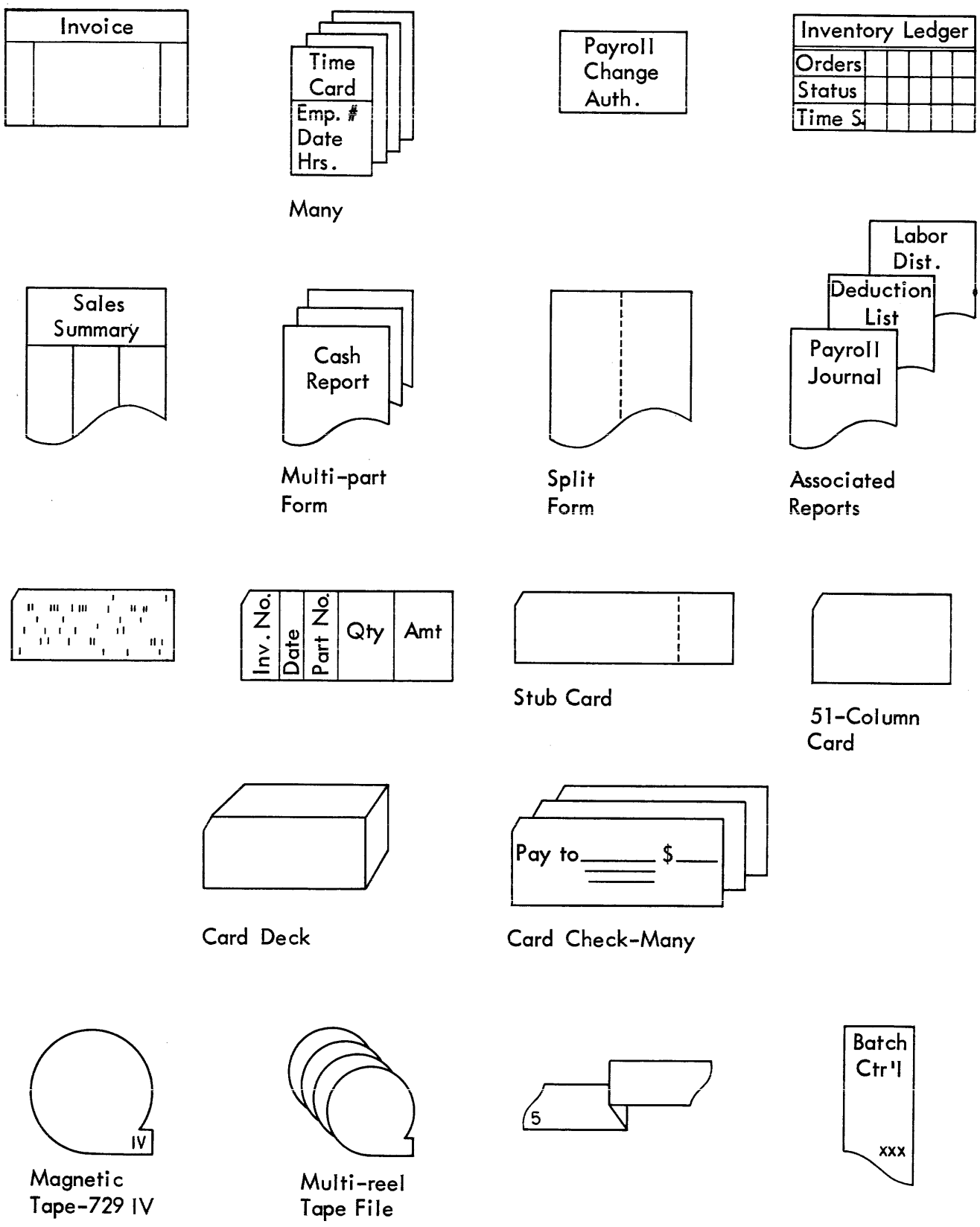

Figure 4. Variations in the Use of Document Symbols

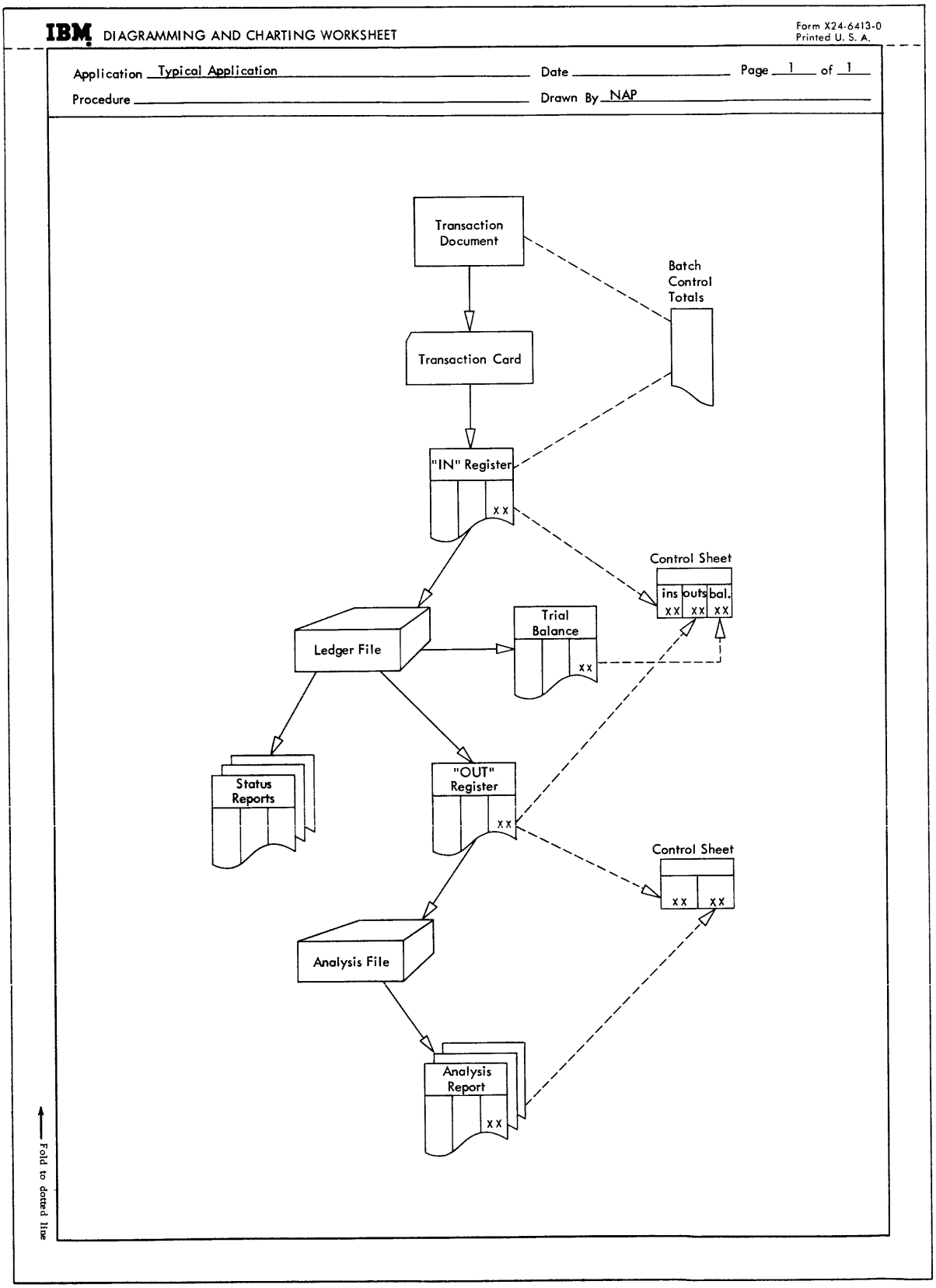

Figure 6. Data Flow Chart

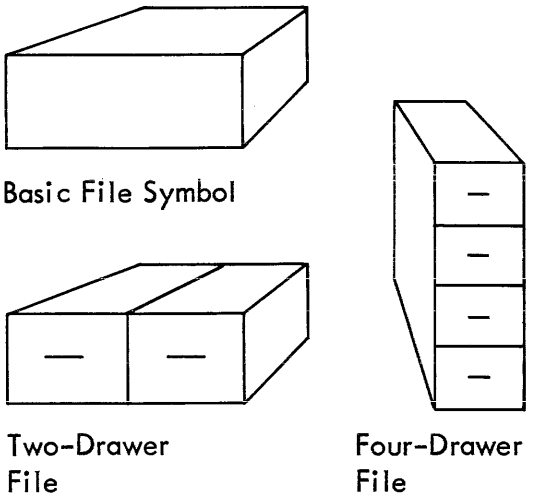

Figure 5. Document File Symbols

the impression of many documents or multiple forms can be given, various special types of punched cards represented, and paper and magnetic tape identified as to type.

In addition to the documents themselves, it is also necessary to represent files of documents. This is done with the FILE *(q* plus *r)* symbol shown in Figure 5.

Data Flow Charts

The symbols introduced thus far are all that are required to draw one simple type of chart — the data flow chart, so called because it describes an application in terms of the development of information.

Figure 6 is a data flow chart representing a portion of a punched card accounting application. A chart of this type might be used in depicting the overall application for management or for persons receiving the final reports. The source of the information is readily seen, the cards used in the procedures are evident, and the various reports relating to the job are shown.

In addition, the manner in which control is established and maintained throughout the process is shown by the use of dotted lines and a symbolic representation of control totals (XX) on the various reports.

To anyone even vaguely familiar with punched card accounting, it is obvious that a card punch, an accounting machine and, perhaps, a sorter or collator are involved in the procedures. Actually, as will be shown later, all of these machines, plus a card verifier, interpreter and reproducer, are required. A second type of flow chart which provides this information will be described.

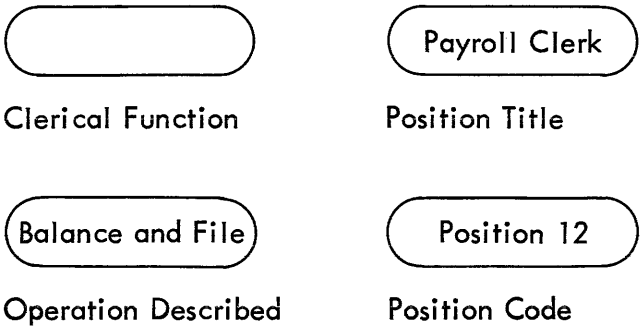

Figure 7. Clerical Function Symbol

Clerical Function Symbol

Operations performed at a clerical work station are represented by the CLERICAL FUNCTION *(u)* symbol shown in Figure 7 along with various ways in which the operation can be indicated. Specifying only a position title or code implies that a job description found elsewhere will provide the necessary information as to exactly what occurs at that station.

Unit Record Machine Symbols

Figure 8 illustrates the four basic symbols used to depict unit record machine operations. A few of the many ways in which these symbols are used appear in Figure 9.

The CARD PUNCH and CARD VERIFIER *(m)* symbol indicates card preparation and verification by key-driven devices (including machine types 10, 24, 26, 27, 28, 56). The Card Proof Machine generates an adding machine tape which is represented by the Transmittal Form (Figure 3).

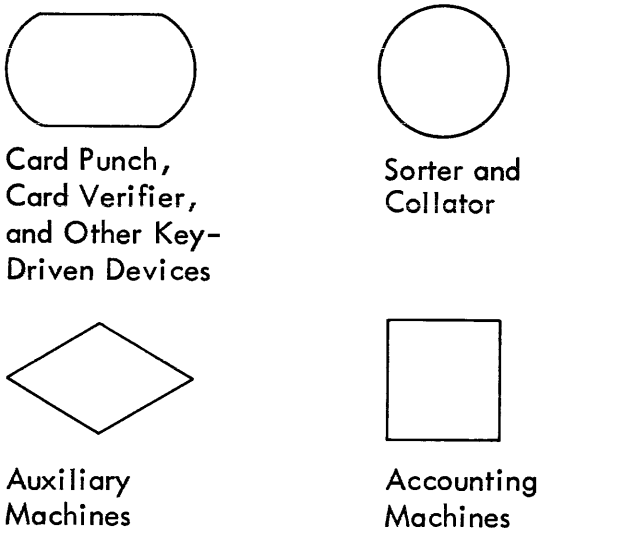

Figure 8. Unit Record Machine Symbols

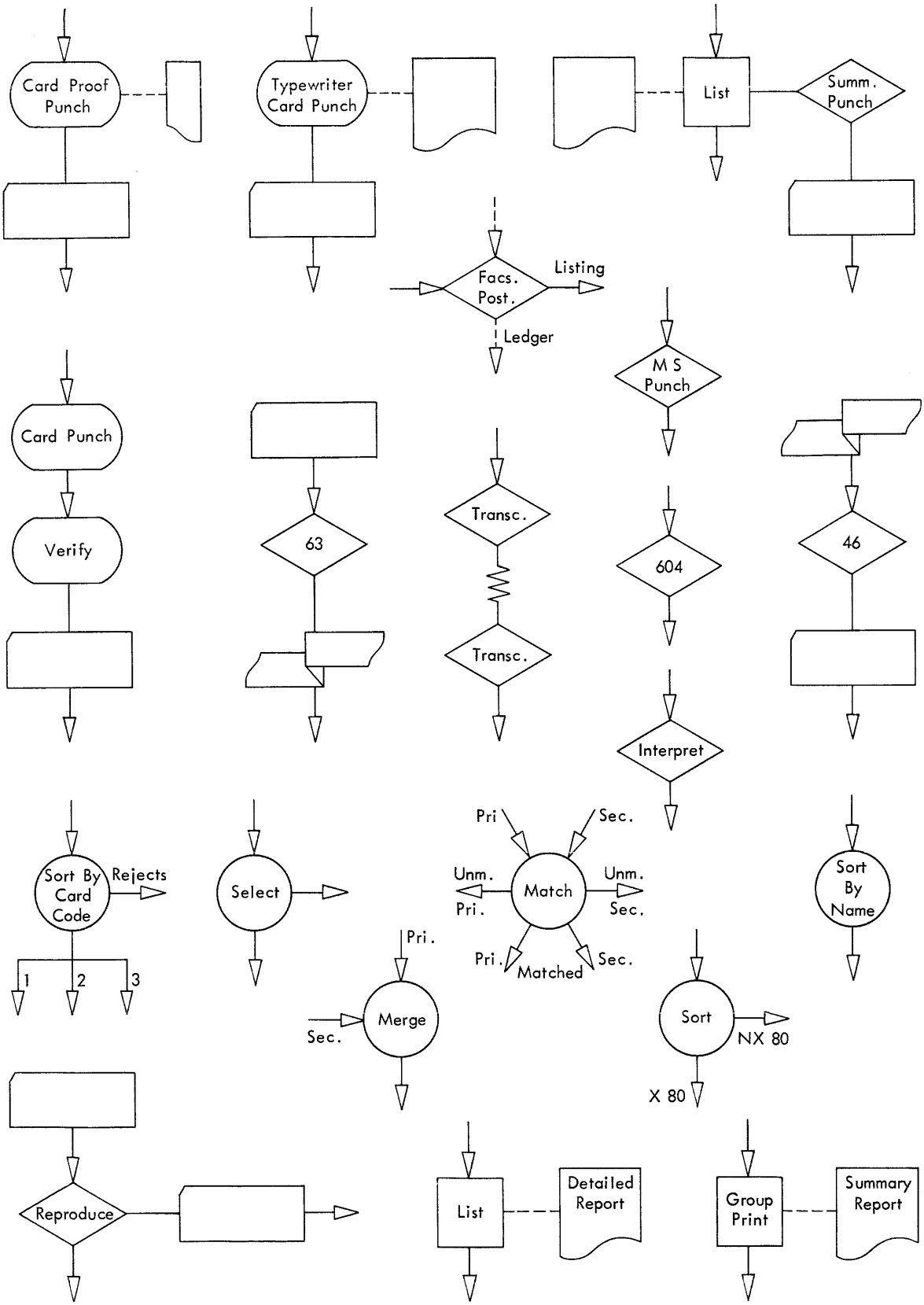

Figure 9. Typical Unit Record Machine Operations

The same symbol is also used to represent other key-driven devices such as the Cardatype®, Typewriter Card Punch, and Typewriter Tape Punch.

The SORTER and COLLATOR *(a)* symbol, representing operations of machine types 82, 83, 85, 87, 88, 101 and 108, may have one or more card files leading to the operation symbol and one or more card files leaving the symbol, depending on the type of operation being performed.

The ACCOUNTING MACHINE *(t)* symbol is usually associated with a Report symbol (Figure 3) illustrating the result of the operation. One of the examples in Figure 9 shows how printing with summary punching is illustrated.

The AUXILIARY MACHINE *(b)* symbol is used to represent operations performed on reproducing and summary punches, interpreters, calculators, facsimile posting machines, and tape-to-card and card-to-tape punches.

In Figure 9 it should be noted that documents produced as a result of an operation have a line leading to them but no arrow. This is done to indicate that the symbol does not represent a separate job step but is associated with the symbol to which it is connected. The line leading from the document to the next job step in which it is involved does have an arrow.

Data Processing System Symbols

The symbols in Figure 10 are used in flow charting to illustrate data processing system operations. The large number of machine components of a data processing system requires a somewhat different type of symbolism from that used with unit record machines. In this case the rectangle labeled CENTRAL PROCESSING UNIT *(c)* in Figure 10 is used to represent the basic system components. This is assumed to include the main storage facilities, the arithmetic unit,

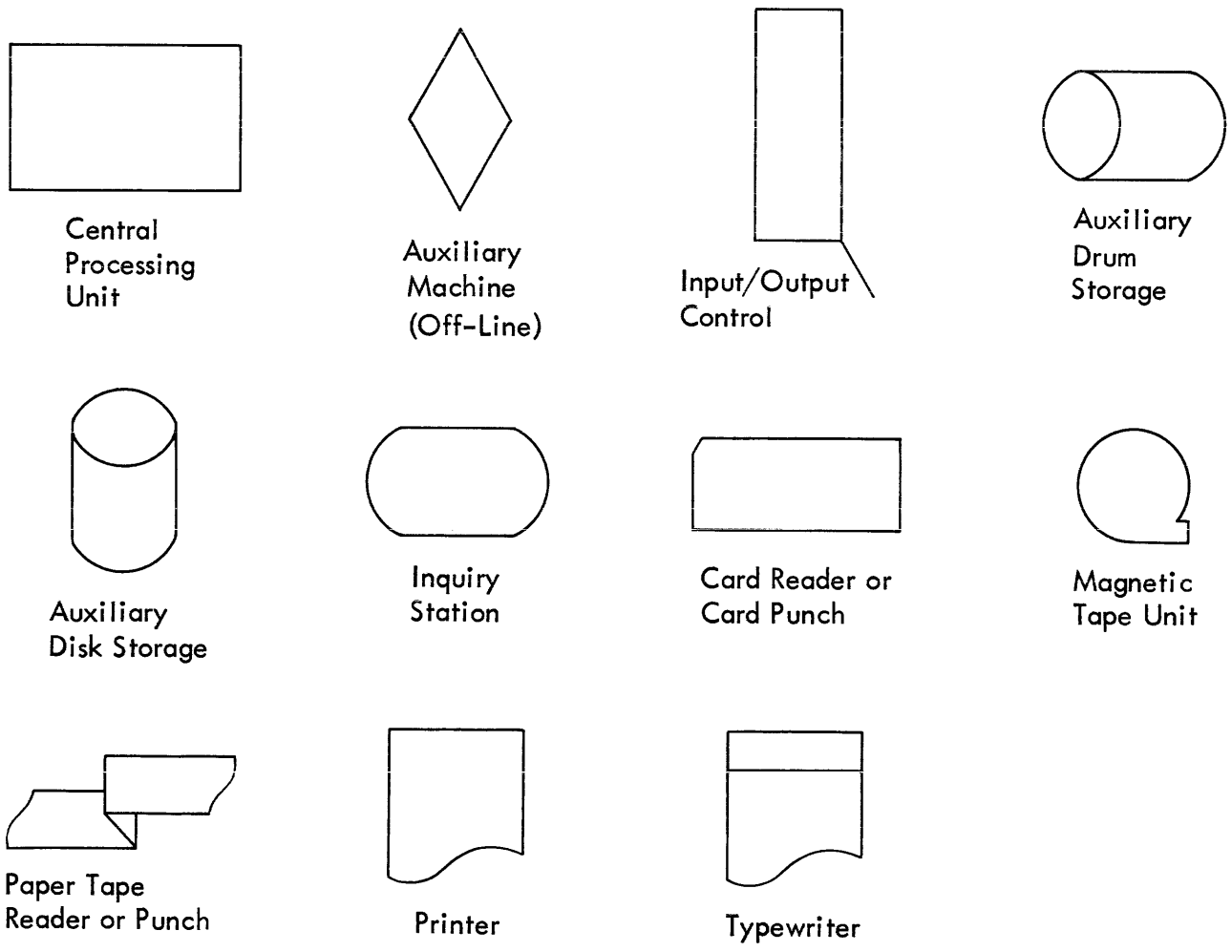

Figure 10. Data Processing System Symbols

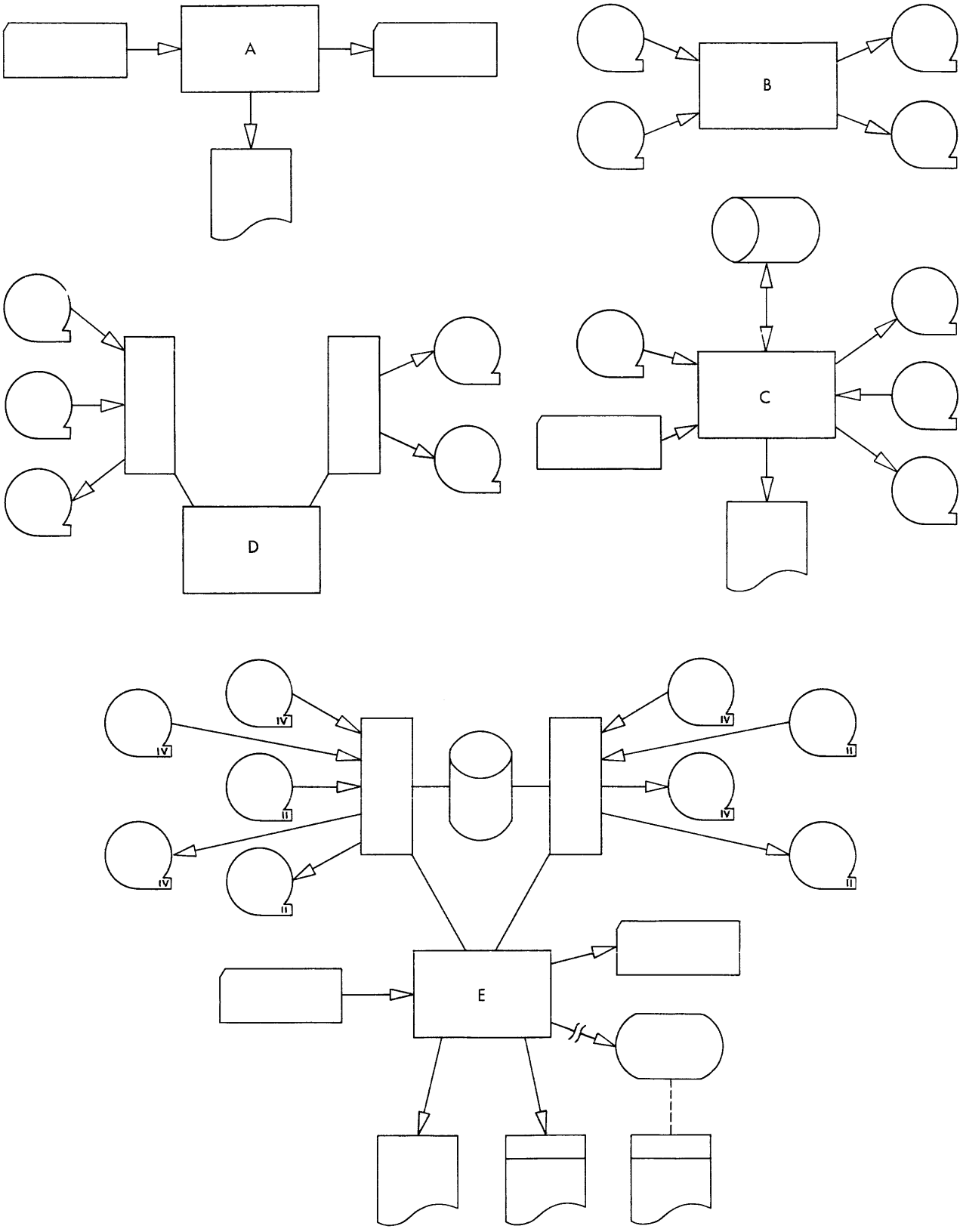

Figure 11. Typical Data Processing System Configurations

and the basic controls. To this are connected additional symbols representing peripheral components and the documents which they process. For example, a reel of magnetic tape indicates that a magnetic tape unit is connected; a punched card with an arrow leading to the CPU indicates a card reader; a punched card with an arrow leading from the CPU, a card punch; and so on.

In some applications the manner in which auxiliary storage units are used (e.g., magnetic drum storage on a 705 system or magnetic disk storage on a 650 system) is an important feature of the system. The AUXILIARY DISK STORAGE and AUXILIARY DRUM STORAGE *(m)* symbols, connected to the central processing unit by a two-pointed arrow, provide a means of illustrating such features.

For many data processing systems, the INPUT-OUTPUT CONTROLS *(r)* must be shown in order to depict clearly the system's capabilities. A specific example is the IBM 7070 with the 7604 Tape Control. The manner in which magnetic tape units are assigned to the two data channels is a significant factor in the data processing procedure in that processing time will depend partly on the assignment. The same thing is true in the case of the 705 III or 709 with two or more Data Synchronizers and the 705 I and II with two or more Tape Record Coordinators.

With some other systems — a 705 II with 754 Tape Controls, for example — the manner in which the magnetic tape units attach to the central processing unit has no obvious effect on the application approach or timing, hence the tape controls need not be shown.

Figure 11 illustrates typical ways in which data processing system operations are flow-charted. The Central Processing Unit symbol is sometimes labeled with the type number to identify the system. When no confusion as to the system type can exist, the space can be used to identify the processing step.

The illustrations in Figure 11 can represent any of the following system configurations:

A. Card reader, card punch and printer providing direct input and output to:
   1. an IBM 305
   2. an IBM 650 Data Processing System
   3. any IBM 700 series system
   4. an IBM 7070 Data Processing System

B. Four magnetic tape units operating with:
   1. an IBM 650 Data Processing System
   2. an IBM 705 Data Processing System with one or more 754 Tape Controls or one 777 Tape Record Coordinator
   3. an IBM 705 III Data Processing System with one 767 Data Synchronizer
   4. an IBM 704 Data Processing System with 753 Tape Control
   5. an IBM 709 Data Processing System with one 766 Data Synchronizer

C. Four magnetic tape units, auxiliary drum storage, a card reader and a printer operating with:
   1. an IBM 705 Data Processing System with one or more 754 Tape Controls or one 777 Tape Record Coordinator
   2. an IBM 705 III Data Processing System with one 767 Data Synchronizer
   3. an IBM 704 Data Processing System with 753 Tape Control
   4. an IBM 709 Data Processing System with one 766 Data Synchronizer

D. Five magnetic tape units operating with:
   1. an IBM 705 Data Processing System with two 777 Tape Record Coordinators
   2. an IBM 705 III Data Processing System with two 767 Data Synchronizers
   3. an IBM 709 Data Processing System with two 766 Data Synchronizers
   4. an IBM 7070 Data Processing System with 7604 Tape Control
   5. an IBM 7090 Data Processing System with two 7607 Data Channels

E. An IBM 7070 Data Processing System with 7604 Tape Control, five 729 IV Magnetic Tape Units, four 729 II Magnetic Tape Units, 7300 Disk Storage, 7500 Card Reader, 7150 Console Control Unit (typewriter), 7400 Printer, 7550 Card Punch and 7900 Inquiry Station. (The remote location of the inquiry station is indicated by the broken line.)

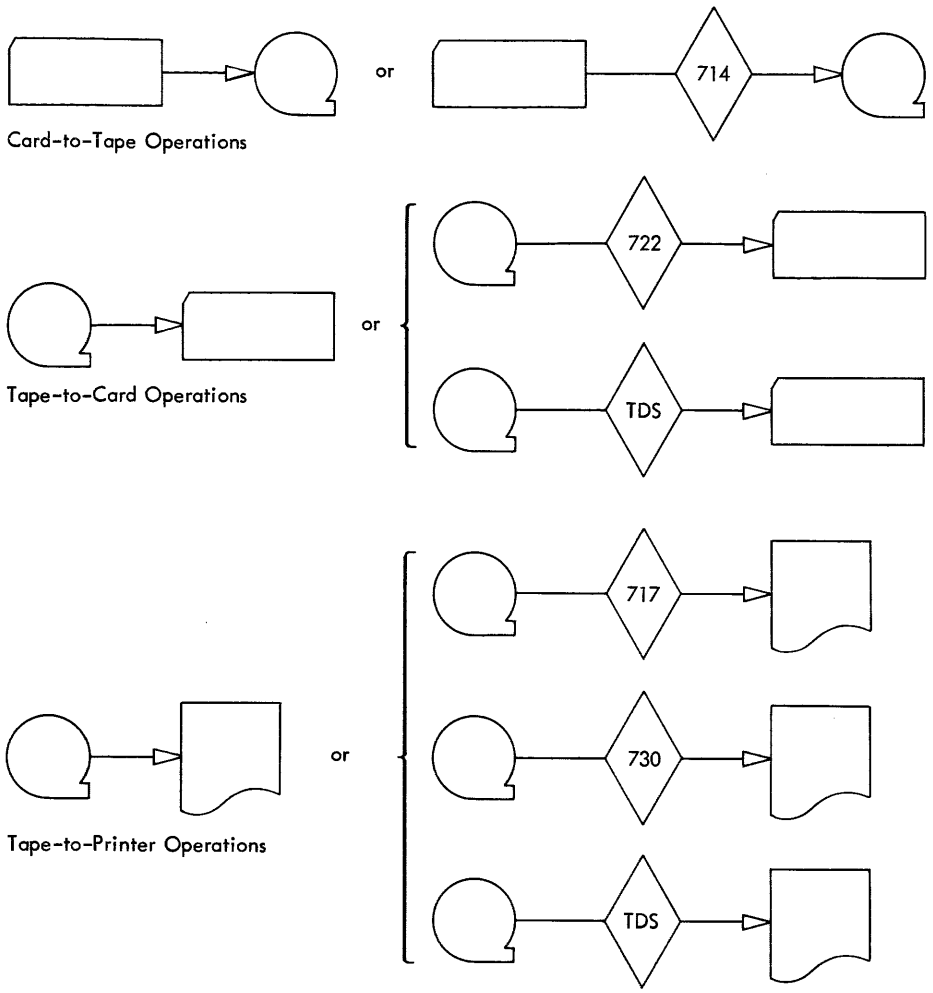

Figure 12. Typical Auxiliary Machine Operations

The AUXILIARY MACHINE *(b)* symbol in Figure 10 is used to identify the units involved in off-line operations. Figure 12 shows a simplified symbolic representation for card-to-tape, tape-to-card, and tape-to-printer operations as well as variations in which the specific units are identified by type number.

Operational Flow Charts

The operational flow chart is an extension of the data flow chart in that it shows the job steps involved in the development of information as well as the documents themselves. Figure 13 is an operational flow chart representing the first few steps in the punched card application illustrated by the data flow chart in Figure 6. The manner of presentation in this example might be categorized as "formal" in that the basic flow charting rules have been closely followed.

Note that the primary direction of flow of the application is in a vertical line from the top of the page to the bottom. Secondary functions (e.g., balancing original control totals against totals developed elsewhere in the procedures and filing cards prepared for later processing) are shown to one side.

The information provided by Figure 13 is sufficient to satisfy the needs of all but the person responsible for installing and maintaining the system and the clerks and operators performing the various clerical and machine functions. In the case of punched card and clerical procedures the latter requirements can be satisfied by providing:

1. a narrative description of the manual operations to be performed at each point in the procedures
2. control panel wiring diagram, carriage control tapes, etc., for each machine function which requires them.

This material, assembled in a notebook or file folder together with forms layouts, job schedules, and so on, comprises a manual of procedures. The circled numbers at each work station or job step symbol provide a simple means of referencing such material to the flow chart.

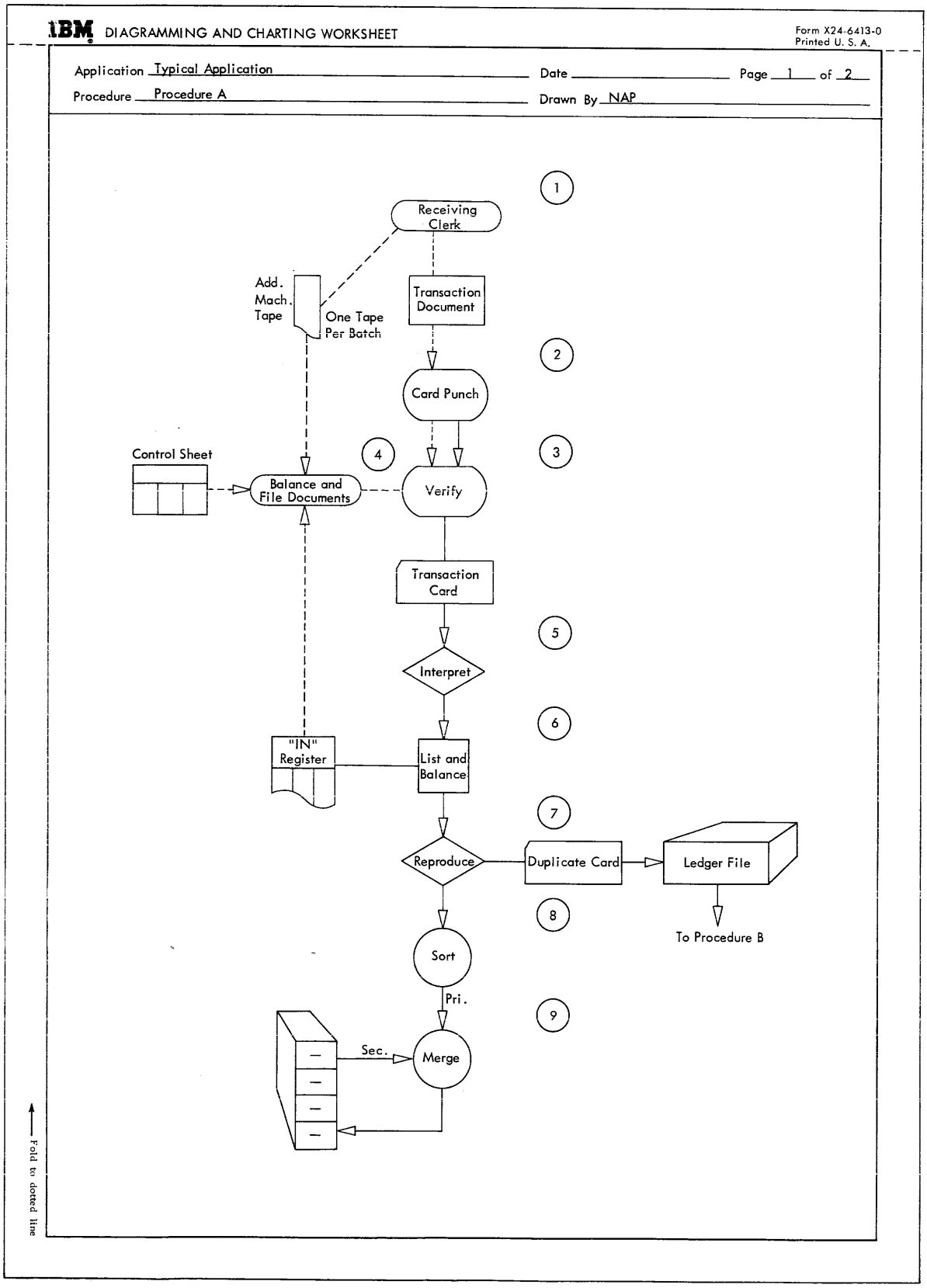

Figure 13. Operational Flow Chart (Formal), Unit Record Procedures

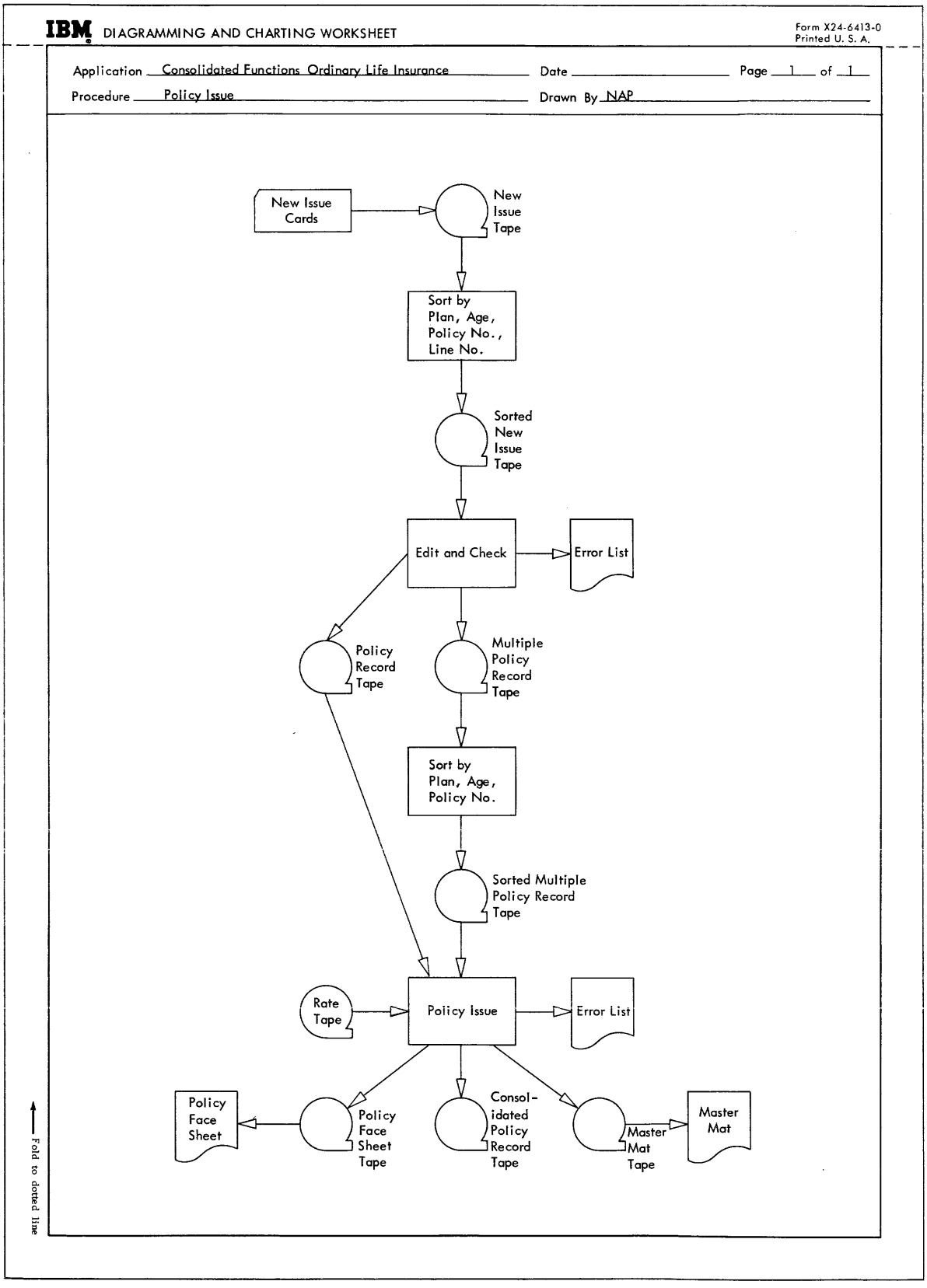

Figure 14. Operational Flow Chart (Formal), IBM 705 Procedures

Figure 14 is an illustration of a typical operational flow chart of an application of a data processing system – in this case the 705. This chart also falls in the formal category. Figures 15 and 16 are informal operational flow charts illustrating the use of line drawings and photographs in place of the template symbols.

Flow charts of data processing system applications usually convey much less information as to what is

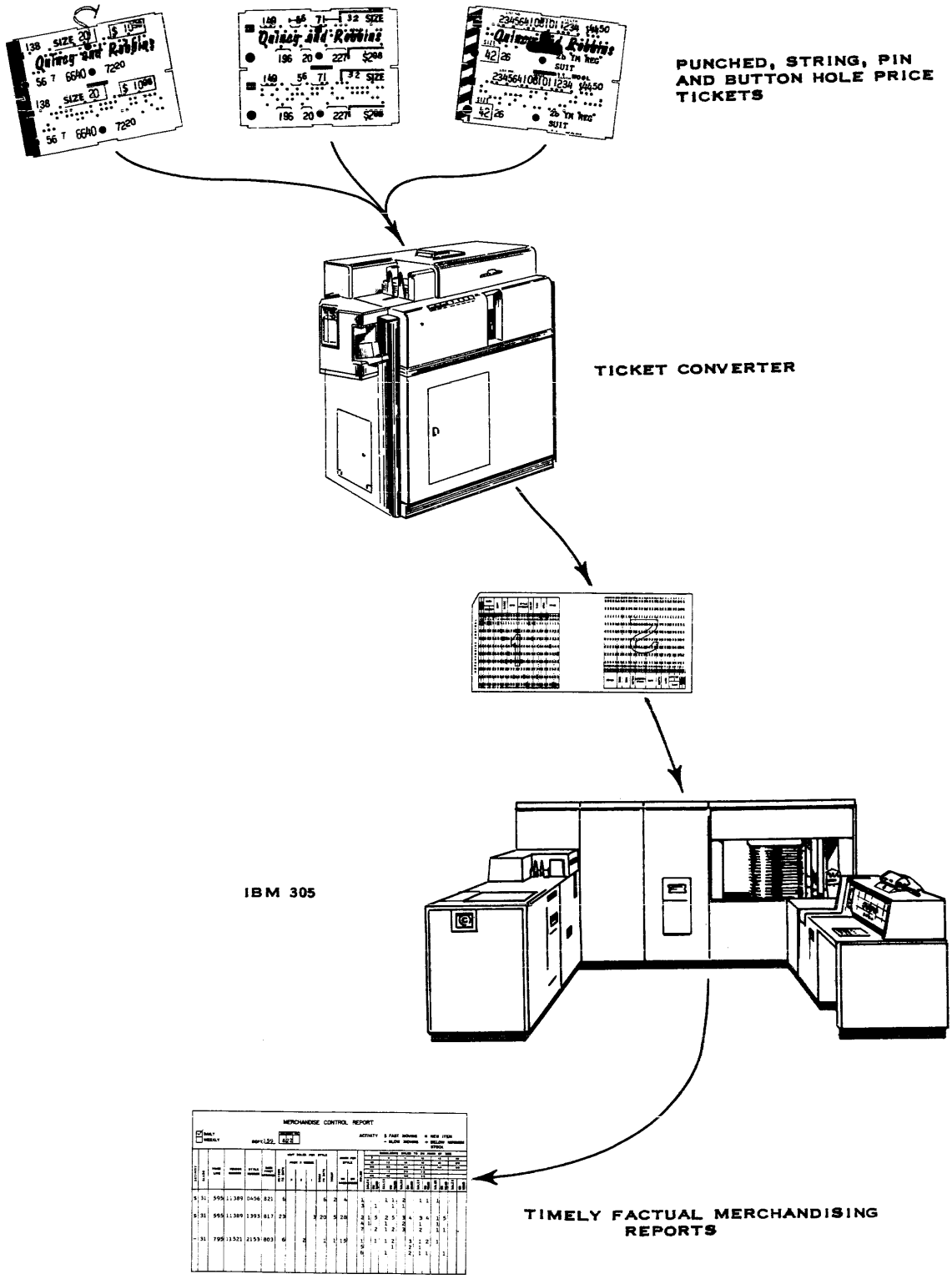

Figure 15. Operational Flow Chart (Informal), RAMAC 305 Procedures

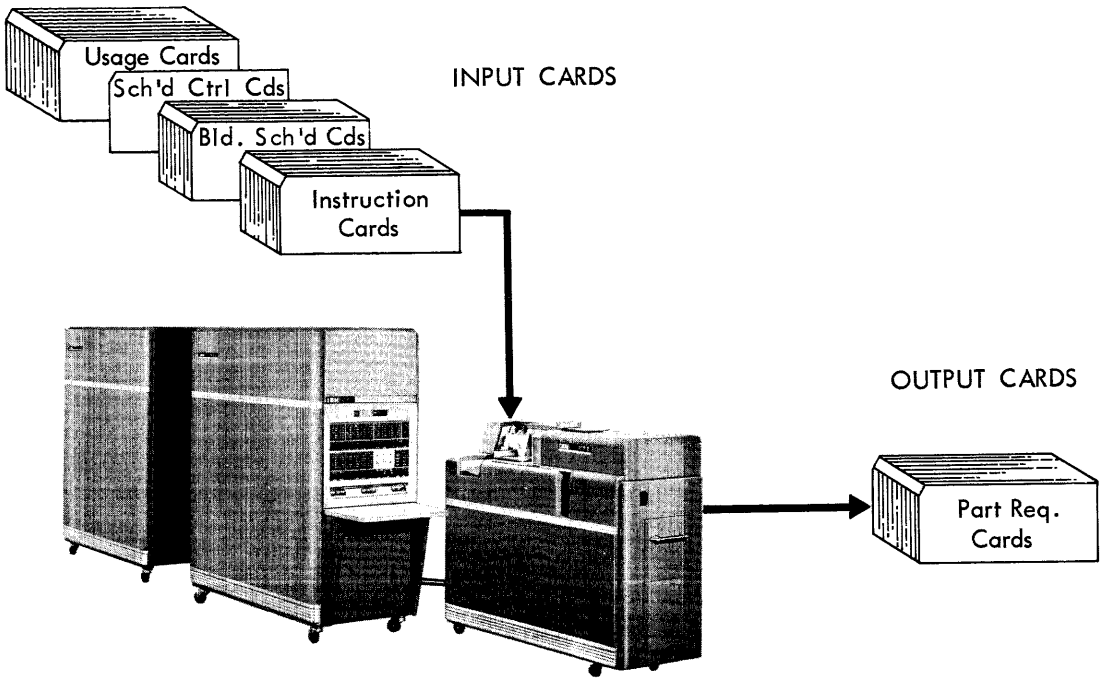

Figure 16. Operational Flow Chart (Informal), IBM 650 Procedures

actually being done than a flow chart of a punched card procedure, because each step is considerably more involved. For this reason, complete documentation of the application must include block diagrams of each processing step. The manner in which these are constructed will be described in the next section.

The labeling of an operational flow chart for a data processing system application is important when it is to serve as a guide to programming and operating. It is at this point that tape and program identification must be introduced. Schemes have been developed which lead to great efficiency in systems planning and operating, particularly with regard to:

1. programming job assignments
2. tape labeling and library maintenance
3. job scheduling.

A brief description of the highly efficient system used at one installation can be found in the IBM General Information Manual, "The IBM 705 System of Accounting at Southern Railway" (Form E20-6025).

Block Diagramming

A block diagram has been defined as a graphic representation of the procedure by which data is processed. Hence, one could assume that block diagrams would be used to illustrate clerical, punched card machine and data processing system operations. While it is theoretically true that clerical and punched card machine functions can be block-diagrammed, this practice is not followed since many of the steps are assumed to be understood by the clerk or machine operator. Furthermore, in the case of machine operations, control panel wiring diagrams and planning charts are more descriptive of the functions than a block diagram would be.

This discussion, then, will be concerned only with block diagramming data processing systems functions. It might be argued that the coding sheet (or programming form) is to the data processing systems operation what the wiring diagram and planning chart are to the punched card procedure. This is true in only a limited sense because the coding sheet lacks the graphic qualities that the others have.

One of the most important uses of the block diagram is to provide the programmer with a means of visualizing, during the developmental stages of programming, the sequence in which logical and arithmetic operations should occur, and the relationship of one portion of a program to another. The coding sheet does not offer this advantage.

The amount of detail included in a block diagram will vary according to the purpose it is to serve. In the early stages of program development the primary purpose of the diagram is to experiment with and verify the accuracy of different approaches to coding the application. In this instance large segments of the program are represented by a single symbol.

Once an approach has been established, a block diagram is prepared to serve as a basis for coding. At this

point the nature of the system being programmed must be taken into account. For the IBM 305, such a block diagram will usually show in detail both the stored program steps and the control panel functions. For the 650, 705, and 7070 systems, however, such detail is neither necessary nor desirable. In order to preserve its usefulness as a clear picture of the overall program logic, the block diagram should not, in these cases, be a replica of the coding. Because of this and other differences, block diagramming techniques for the 305 will be discussed separately.

Block Diagramming Symbols, Stored-Wired Program

A procedure for a system which utilizes a combination of stored and wired programming is characterized by two main features:

1. functions initiated by control panel wiring
2. functions initiated by stored program instructions.

The symbols chosen for graphically illustrating such procedures serve to differentiate between these two types of functions.

In order for the block diagram to fulfill its purpose certain other aspects of the program must also be shown, namely:

1. the program exits which cause the change from stored program to wired program control
2. points at which signals are generated to initiate operations on the console, punch or printer
3. the condition which causes a particular branch of a program to be taken.

The recommended symbols for use in block diagramming procedures for a system combining the stored program and wired program concepts (as does the 305) are shown in Figure 17.

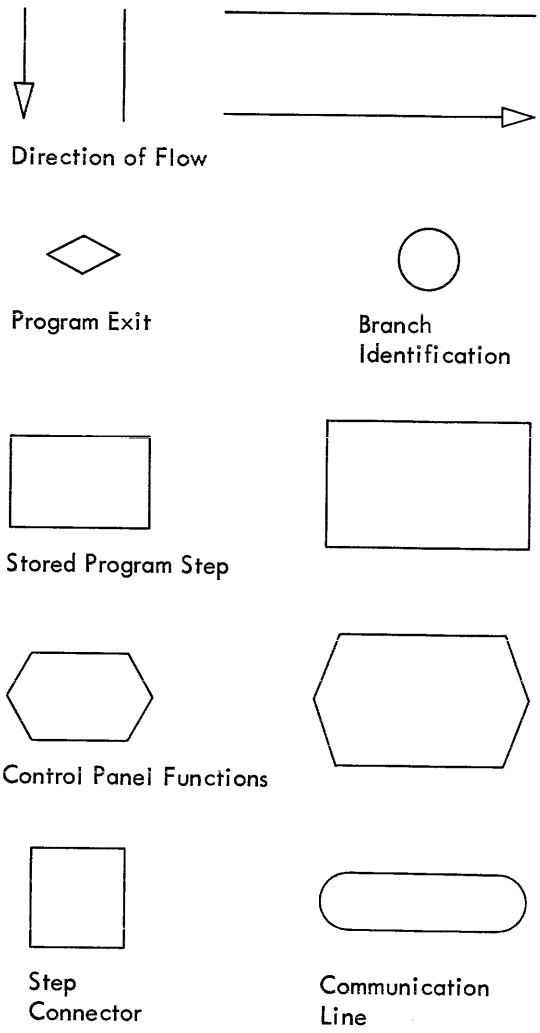

Figure 17. Block Diagramming Symbols, Stored-Wired Program

The DIRECTION OF FLOW *(y, z)* symbol has the same function as it did in the flow chart. Since the average block diagram contains a large number of symbols and follows the "top-to-bottom" rule closely, the arrowheads are frequently omitted except where necessary for clarity. Dotted lines are used occasionally but have no standard meaning.

The STORED PROGRAM STEP *(c, k)* symbol corresponds to program instructions. In a detailed block diagram each such symbol corresponds to a single instruction step and the address of the drum storage location assigned to that step is indicated at the top left corner of the rectangle.

The CONTROL PANEL FUNCTION *(d, l)* symbol indicates operations defined by control panel wiring. The only time this symbol refers to other than a process control panel function is when it is preceded by a communication symbol.

Two sizes of each of these two symbols are provided on the template, but the appearance of a diagram is improved if different sizes of the same symbol are not intermixed.

The PROGRAM EXIT *(h)* symbol indicates that the last program instruction contained a control code which will cause an impulse to be emitted at a program exit. The specific program exit involved is indicated by placing the corresponding letter, number or special character within the symbol. A program exit symbol will always be preceded by a stored program step symbol.

The BRANCH IDENTIFICATION *(p)* symbol is used to indicate the condition which must exist for one of two or more alternate paths in the program to be taken. The conditions are established as a result of a compare, a test of the accumulator or a selector, or character selection. These are all process control panel functions, and the branch identification symbol will

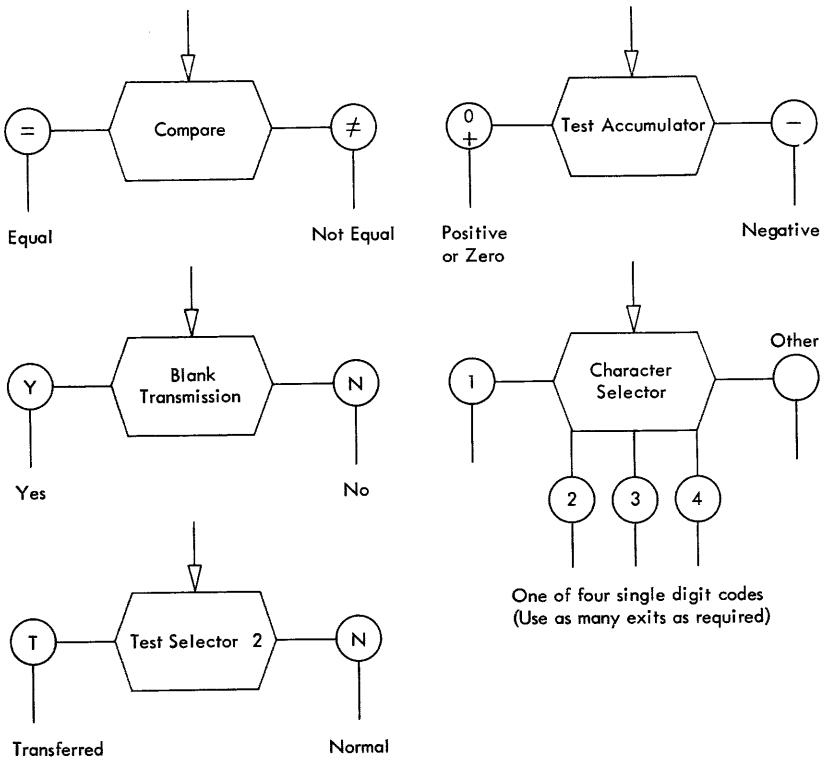

Figure 18. RAMAC Decision Symbols

therefore always follow a control panel function symbol. The various conditions are indicated by labeling the symbol as shown in Figure 18.

The COMMUNICATION *(u)* symbol indicates the use of a program exit impulse to signal the console, printer or punch over the corresponding communication line. The particular unit and line involved are indicated by the label on the symbol. The function initiated is indicated by a control panel function symbol following it.

The STEP CONNECTOR *(t)* symbol, like the communication symbol, does not represent a specific instruction or control panel function. It is used to indicate:

1. points at which a program will terminate
2. the identity of the next step in the program when that step does not immediately follow on the block diagram.

RAMAC® Block Diagramming Techniques

The basic techniques for preparation of IBM RAMAC 305 block diagrams are easily illustrated by an example. Figure 19 is a small part of a diagram of an inventory and billing application. A close examination of this example will bring out most of the important features and techniques of a good block diagram:

1. The main direction of flow is from the top of the page to the bottom.
2. The starting point of the program is indicated by an arrow labeled "copy out."
3. The stored program and control panel function symbols are clearly labeled to indicate the operation (a single operation in the case of the detailed diagram shown).
4. Notes have been used to indicate unusual situations (e.g., cause of stops).
5. A dotted line has been used to connect a branch point which actually does not exist in the program. Including this condition in the diagram provides a positive indication that all cases have been considered.
6. Arrowheads are drawn only when they add to the clarity of the diagram.
7. Control panel function symbols for other than the process control panel need not have an exit.
8. A step connector symbol labeled "Go to Step 00" has been used at the bottom of the page rather than a direction-of-flow line all the way to the top of the page.
9. The two major program segments (all that could be conveniently shown on a single page) are clearly separated. To avoid crowding (and the resulting confusion), other segments have been relegated to subsequent pages.

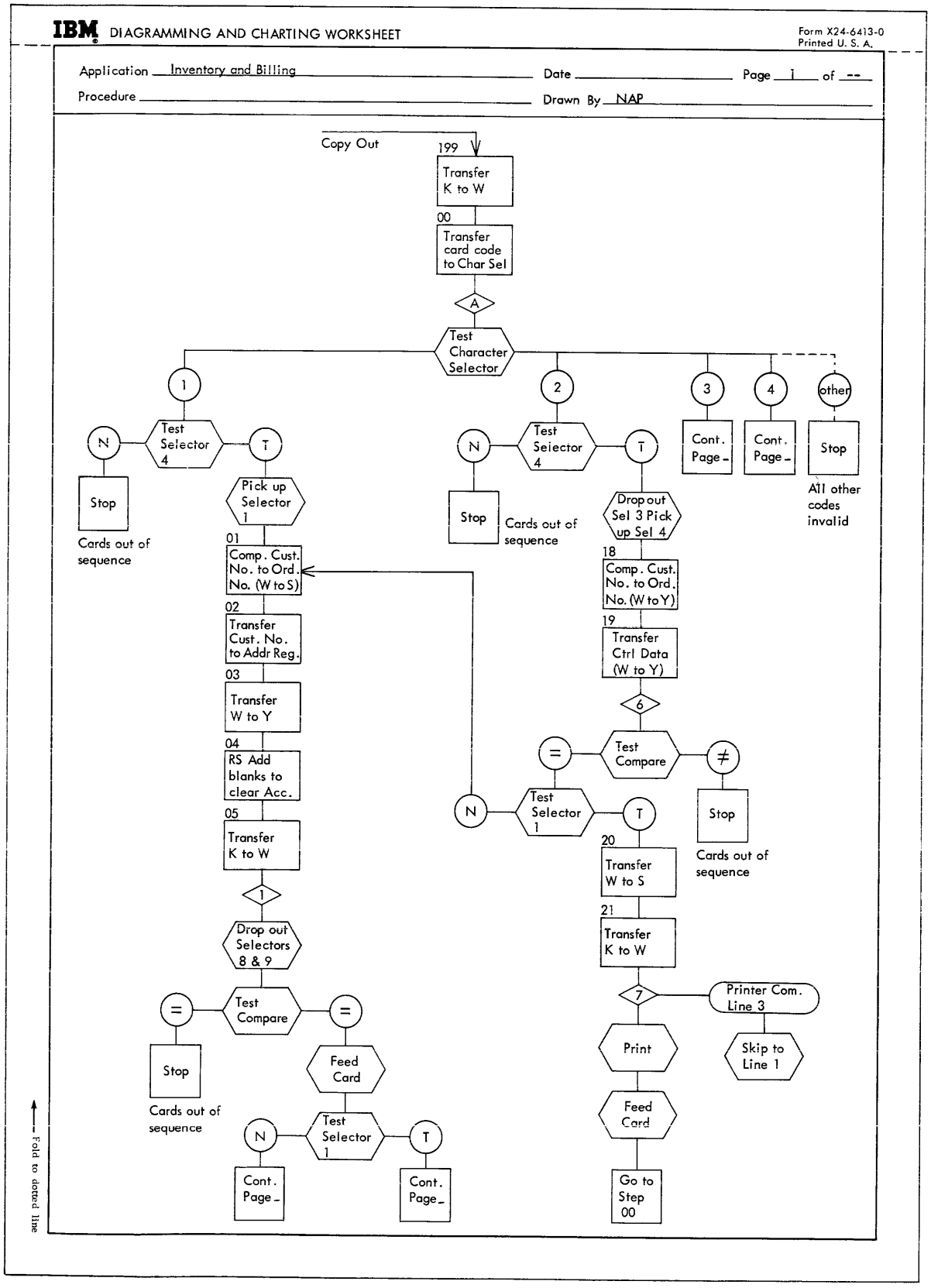

Figure 19. Detailed Block Diagram, Stored-Wired Program (RAMAC 305)

Block Diagramming Symbols, Stored Program

The block diagramming symbols used with stored program systems, including the 650, 705, and 7070, are shown in Figure 20. (SHARE, an organization of 704, 709 and 7090 users, has published standards for their members.)

As before, the DIRECTION OF FLOW *(y, z)* symbol is the basic element of the diagram. It is the nature of most programs prepared for systems of this type to involve many decisions, that is, tests to determine which of two or more paths should be taken. This leads to complex block diagrams and hence to the requirement that flow lines be drawn with an arrow whenever the direction is not immediately clear. "Looping" in a program (that is, repeating an instruction sequence) is also a common occurrence and in some cases leads to violation of the basic rule that diagrams should run from the top of the page to the bottom.

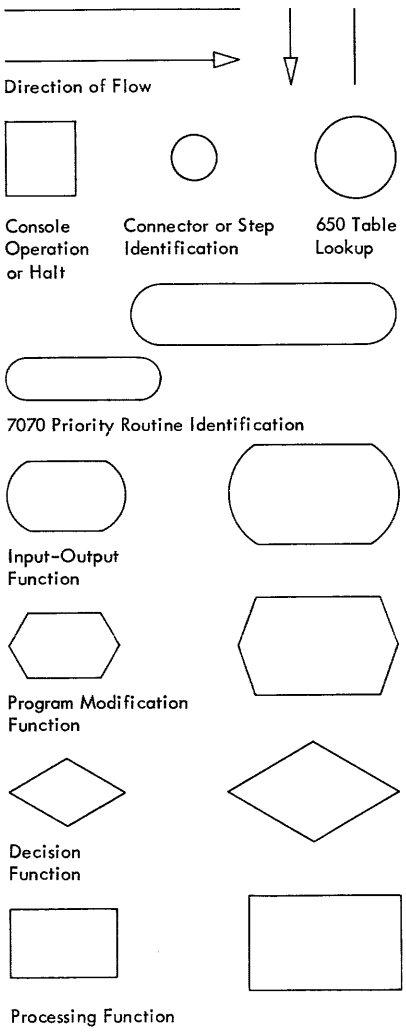

Figure 20. Block Diagramming Symbols, Stored Program

The CONNECTOR OR STEP IDENTIFICATION *(p)* symbol, is used to indicate a direct connection between sections of a block diagram which cannot be easily connected by arrows (i.e., sections widely separated on the page or on different pages). In this way it also provides an indication of and a label for, points in a program to which other portions of the program may branch. When a connector symbol refers to a point on another page, the page number should be shown adjacent to the symbol.

The TABLE LOOKUP *(a)* symbol is used to denote an instruction sequence based on the TLU function of the IBM 650. Since the LE, LL and LEH functions of the 7070 cause branching, the decision function symbol is used in these cases.

The CONSOLE OPERATION and HALT *(t)* symbol represents either a point in a program at which data is introduced via the console typewriter or switch settings, or any point at which a program terminates — either normally at the completion of the job, or because of an error condition.

The PRIORITY ROUTINE IDENTIFICATION *(u, v)* symbol is used in block diagramming for the IBM 7070. The symbol does not represent a program function, but merely serves as a label for the routine that follows and identifies the routine as one which will be executed in the priority mode.

The INPUT-OUTPUT FUNCTION *(e, m)* symbol is used to denote any function of an input-output device. Included in this category are magnetic tape (e. g., reading, writing, backspace, rewind, etc.), card reader, card punch and printer functions, as well as those involving multiple-word communication between auxiliary storage units (such as drum or disk storage) and the main storage.

The PROGRAM MODIFICATION FUNCTION *(d, l)* symbol is used to denote an instruction sequence which effects a change in the stored program itself. Included are:

1. address alteration or modification
2. operation code modification
3. setting of program switches (by above techniques or by bit or digit manipulation, etc.)
4. setting of indexing registers or words
5. setting or resetting an indicator.

(Each particular system provides unique ways in which program modification can be implemented. However, the purpose of the block diagram is to indicate the function and not the specific technique.)

The DECISION FUNCTION *(b, i)* symbol is used to depict a point in a program at which a branch to one of two or more alternate paths is possible. The manner in which the choice is made is indicated by the label and can be based on:

1. a comparison or the test of a compare indicator

TEST CONDITIONS, LOGICAL CHOICES, DECISIONS

ENGLISH STATEMENT	SHORTHAND STATEMENT
Compare A with B (where B is the common factor or constant value)	A: B
A is greater than B	A > B
A is less than B	A < B
A is equal to B	A = B
A is not greater than B	A ≦ B
A is not less than B	A ≧ B
A is not equal to B	A ≠ B
Compare indicator settings	HI LO EQ
Check indicator settings	ON OFF
Balance test results (plus, zero, minus)	+ 0 −
End of file	EOF
End of Job	EOJ
Program switch settings (No Operation and Transfer in the 705)	NOP TR

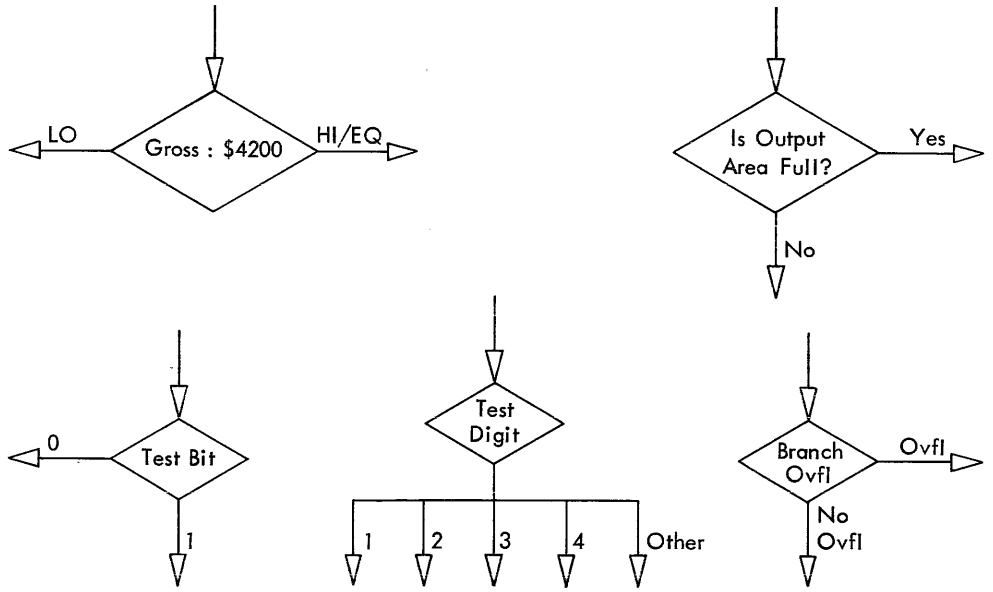

Figure 21. Decision Symbols Uses and Shorthand Notation

(depending on the manner in which the compare operation functions)

2. a balance (sign) test
3. a program switch setting
4. an indexed branch instruction
5. a check indicator setting, etc.

The condition upon which each of the possible paths will be taken must be identified. In addition, all possible conditions should be accounted for. This is done by labeling each of the flow lines leading from the decision symbol as shown in Figure 21. A list of commonly used shorthand notations for program conditions is also shown.

The PROCESSING FUNCTION *(c, k)* symbol is most easily defined as that symbol used to represent a function which cannot be represented by one of the other symbols. Generally, such operations are those concerned with the actual processing functions of the program. Some programs – the assignment section of a generalized sort, for example – are concerned with the modification of another program. The operations which the first program performs on the second are processing functions, and the processing function symbol should be used in illustrating them. The modifications which the first program performs on itself, however, are program modification functions and the corresponding symbol is used.

Stored Program Block Diagramming Techniques

Earlier in this discussion reference was made – but without explanation – to "detailed" and "general" block diagrams. The amount of detail presented in a block diagram will depend upon the use that is to be made of it. The three most important uses are:

1. as an aid to program development
2. as a guide to coding
3. as documentation of a program.

In the program development stage, the block diagram serves as a means of experimenting with various approaches to the mechanization of the application.

At this point, logical program segments will have been established, at least tentatively, through flow charting. Starting with blocks representing the major functions of the proposed program, the programmer develops the overall logic by adding blocks to depict input and output functions, steps for the identification and selection of records, and decision functions.

After the overall logic of the program has been tentatively established, the large segments are extracted from the "general" block diagram and analyzed in the same fashion. Proceeding in this manner, the block diagram becomes more detailed. The eventual goal is to produce a diagram which clearly shows all decision points in the program and which can be used to verify that the procedure satisfies all possible conditions which can arise during program execution.

Once the procedure is established, and proved sound, the block diagram becomes a guide to coding. Unless the person who developed the diagram was an experienced programmer for the data processing system to be used, it is almost certain that the peculiarities of the machine logic will necessitate changes in the program logic. Thus, the diagram may need to be redrawn and reverified at this stage.

Upon completion of the coding, testing and installing of the procedure, the program must be documented to facilitate future modifications which are bound to become necessary. At this stage, the block diagram can greatly simplify the problem by serving as a "map" of the program listing or coding sheets. To be useful in this respect, the block diagram must be related, by labeling, to the instruction steps.

Conventions for labeling instructions and data have been established in connection with the several automatic programming systems in use (e. g., AUTOCODER, SOAP). These same labels should be used to identify the blocks on the diagram which correspond to the instruction sequence on the coding sheets. The labels are placed just above the symbol to the left of the flow line.

Final documentation of a program should include both general and detailed block diagrams. The general block diagram helps in understanding the more detailed one and also provides an easily understood picture of the procedure for those who might be interested in the concept only.

Anyone who has seriously attempted to program a data processing system knows that a considerable amount of scratch paper is consumed before anything resembling a finished block diagram is obtained. For this reason, the illustrations which accompany this text may not resemble very closely the diagrams which will be found in, on and around the desk of the typical programmer. They do, however, illustrate the goal he should be trying to achieve in appearance.

Figure 22 is a general block diagram of a consolidated life insurance file maintenance program.* This particular diagram is arranged on the worksheet so as to show clearly the logical relationship among the three major segments of the program. Because the diagram reads from the top of the page to the bottom, for the most part, arrowheads have been omitted except where lines join and where the direction of flow may not be immediately obvious.

*In this application ungrouped 80-character transaction records are processed against long, ungrouped master records. Unmatched transactions may be errors or represent new business and require generation of a new master record. Master records may be active for reasons other than the presence of a transaction record.

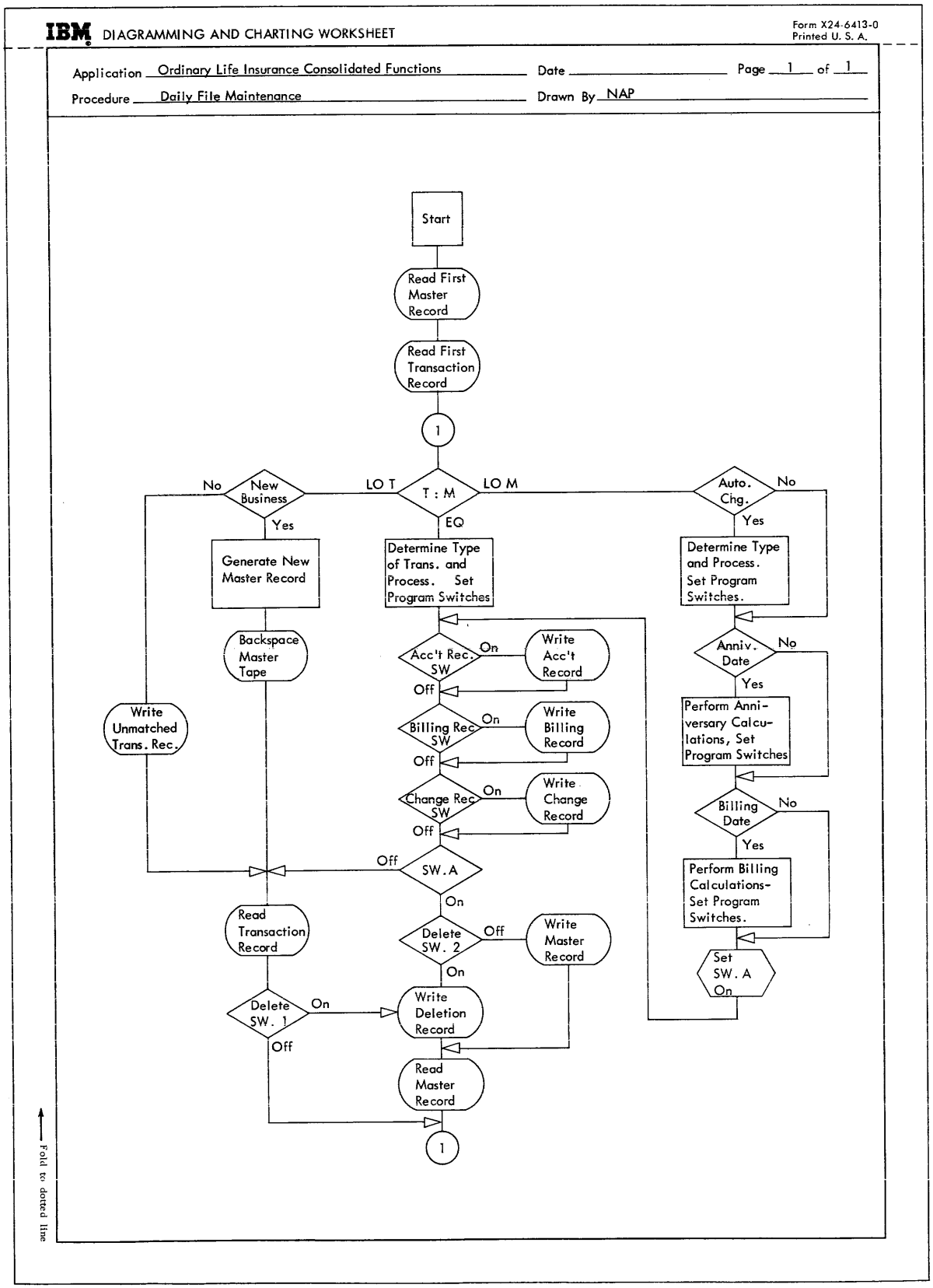

Figure 22. General Block Diagram, Stored Program System

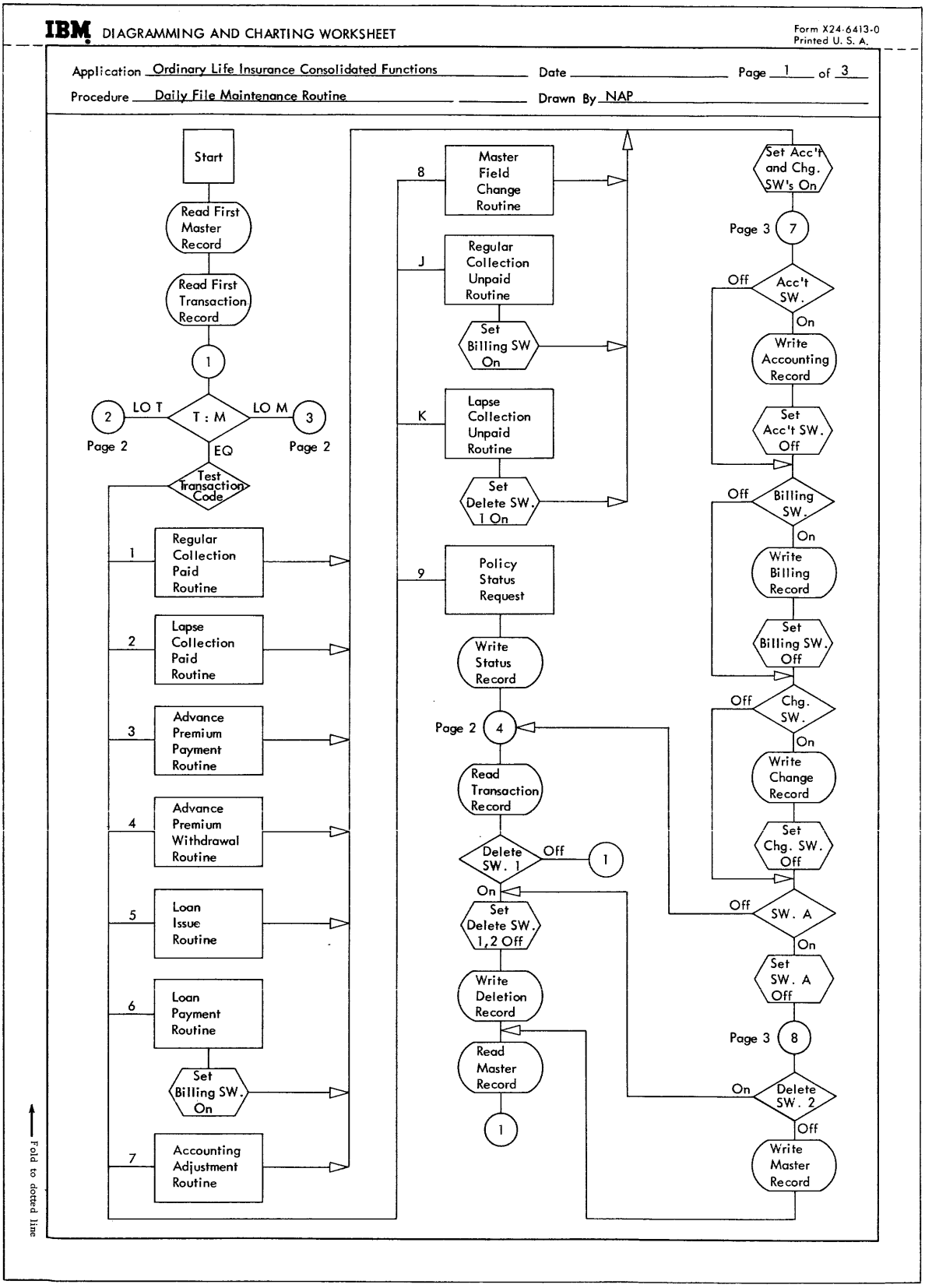

Figure 23. Detailed Block Diagram, Stored Program System

Figure 23 is part of the same procedure shown in more detail. In this instance an arrangement which best utilizes the available space has been used. A diagram such as this one is sufficiently complete to serve as a guide in coding even though certain program functions are still not represented (e. g., read-write error routines, end-of-file and end-of-job routines). A detailed diagram documenting a completed program would include all such functions.

During program development, the "open" arrangement of the diagram on the page, as shown in Figure 22, is preferable to that used in Figure 23 not only because it shows more clearly the alternate program paths, but also because it leaves more room for inserting additional steps. The arrangement used in Figure 23 is most suitable for use in documenting the program because it requires less space.

Block diagrams prepared for an IBM 7070 will differ in some respects from those illustrated because of the Automatic Priority Processing feature and the unique manner in which input-output functions are handled. Since the main processing functions will usually be executed in the non-priority mode, the corresponding block diagram will follow the pattern outlined. Priority routines, however, cannot be shown as linked directly to the non-priority routines since the point at which the linkage will occur is not known. To call attention to this in the diagram the priority routine identification symbol is always used at the beginning of such an instruction sequence.

A priority routine is terminated by the execution of a priority release instruction, the address of which determines whether control will return to the point at which the interruption occurred or to some specified other point. In the first case, the block diagram of the sequence should end with a dotted line and arrow from the last symbol. If control is to be transferred to a specified routine, then a solid line and a properly labeled step connector symbol should be used. Figure 24 illustrates the manner in which priority routine entry and exit points are diagrammed.

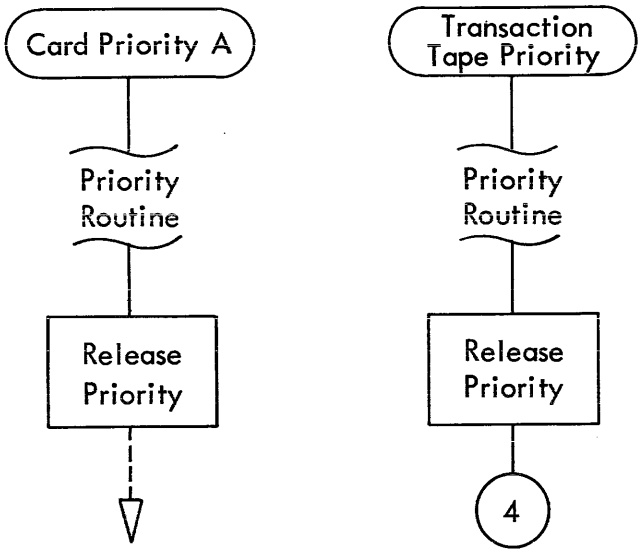

Figure 24. Priority Routine Entry and Exit Symbols

The 7070 performs a number of unique operations which, in preparing block diagrams, should be considered to be program modification functions. Included in this category are:

1. setting and modification of indexing words
2. setting priority stacking latches
3. loading the priority masking register
4. establishing Halt or Sense mode for field overflow or sign changing conditions
5. setting of electronic switches.

Perhaps the most important aspect of systems and procedures development is the communication of ideas between an analyst and

... the individuals involved in current systems and procedures

... the managers responsible for the approval and implementation of new systems and procedures

... other analysts working on the same or related systems and procedures.

Adherence to the flow charting and block diagramming standards outlined in this manual will greatly improve the effectiveness of communication in this endeavor.

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York