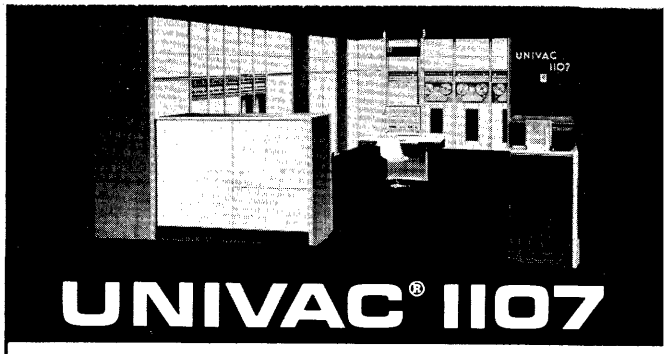


UNIVAC® 1107

TECHNICAL BULLETIN

CLAMP
Relative Load Routine

Programmers Reference

First Edition

April, 1962

TABLE OF CONTENTS

I. INTRODUCTION

CLAMP (Controlled Loading And Modification of Programs), is designed to transfer the object programs produced by the UNIVAC® 1107 Assembly System (or by the ALGOL or COBOL compilers) from their input media to their operating environments. CLAMP can operate as a part of the Executive System or in an independent status.

The object programs to be loaded by CLAMP, the UNIVAC 1107 Relative Load Routine, are in one of three object codes:

1. Absolute Object Code (AOC): Programs in this code are assembled to operate at an absolute core location. All input/output equipment, as well as data areas and data tables, is specified with absolute assignments. AOC programs run independently of the Executive System. These programs may only be loaded by CLAMP if the loading process does not cause CLAMP to be overwritten.

2. Relative Object Code - Executive System I/O (EXEC ROC): These programs are assembled with symbolic input/output references and must operate under Executive System control. Input/Output functions in these programs are submitted as requests to the Executive System and the Executive I/O Functional Routines are used to communicate with all peripheral equipment. All instructions and data areas are given relative symbolic assignments.

3. Relative Object Code - Direct I/O (DIRECT ROC): Input/Output references in these programs may be symbolic, but must be made absolute at assembly time. These absolute references may be reassigned at load time. Programs in this object code operate independently of the Executive System. All instructions and data areas are given relative symbolic assignments.

CLAMP will convert all symbolic references in ROC object programs to absolute assignments at load time.

ROC programs to be loaded by CLAMP are classified as either simple or complex programs. A simple program is one which is entirely contained on one input medium and in one program file. All subroutines referenced by the program are incorporated into the object program at assembly time.

A complex program is composed of a main program and one or more subroutines. The main program is contained in one program file while each subroutine is contained in an additional program file.

Two versions of the Relative Load Routine exist. The first is a part of the Executive System and loads object programs which operate under Executive Control, i.e., programs assembled in EXEC ROC format. All core allocations as well as I/O assignments are made by the Executive System before control is transferred to CLAMP. This version of the Relative Load Routine is called up by the Executive as part of its initiation procedure.

The second version of CLAMP is used to load object programs which are assembled in AOC or DIRECT ROC format. This version of the Relative Load Routine may be called either by the Executive System (if the program is to operate independent of the Executive System, but if the Executive is operating and a Job Request was submitted to it), or through manual initiation from the keyboard.

The Executive System may thus reference either of the two versions of CLAMP. Each Job Request submitted to the Executive will specify Executive control or independent operation.

When referenced either by the Executive System or through manual initiation, the Relative Load Routine will

1. Locate the object program on the input media (only for DIRECT ROC programs).

2. Accept relocation data and use it to modify and load the object program.

3. Modify data table lengths (in the core memory) to reflect new table length definitions introduced at load time.

4. Modify magnetic drum table lengths to reflect new table length definitions introduced at load time.

5. Modify symbolic I/O references to absolute assignments (only for EXEC ROC programs).

6. Assign the addresses for all symbolic references to the Executive System or to the Relative Load Routine.

7. Modify symbolic Selective Jump switch references to absolute assignments (only for EXEC ROC programs).

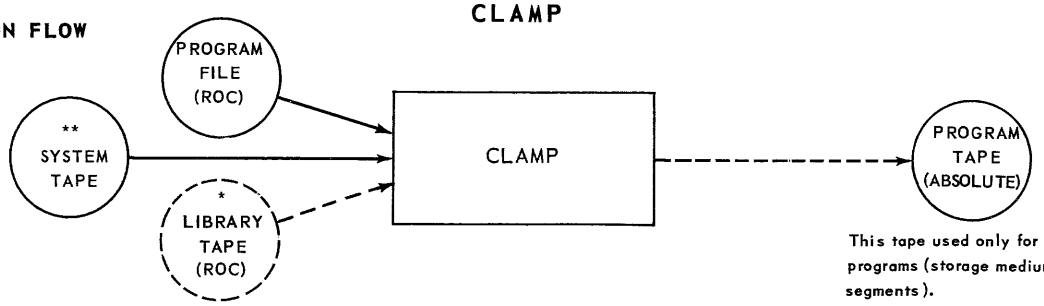
8. Load input parameters in the data area of the object program.

9. Incorporate subroutines into an object program at load time.

The following pages will serve to describe the manner in which the Relative Load Routine performs its required functions. It is assumed that the reader is familiar with the pertinent facts concerning the computer system and the 1107 Executive System. Appropriate references to these systems will be made. The chart of Figure 1 illustrates the operation of the Relative Load Routine.

CLAMP

INFORMATION FLOW



This tape used only for segmented programs (storage medium for segments).

N * This tape used only with complex programs

** Same tape as Executive System tape

NOTE: Magnetic Drum may be substituted for any (or all) tapes except Library Tape

BLOCK DIAGRAM

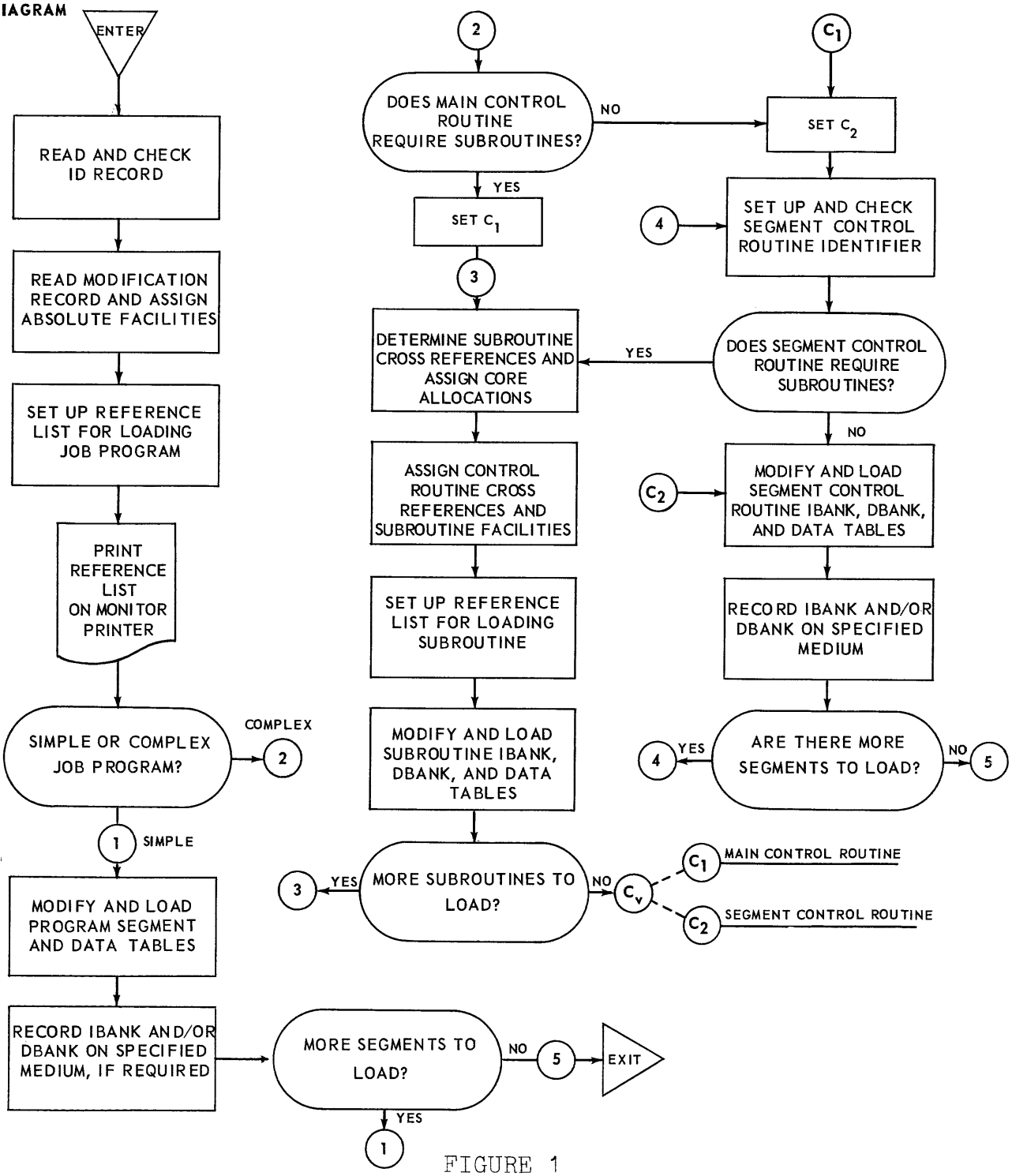


FIGURE 1

II. FORMAT OF PROGRAM FILE

A Program File is produced by the 1107 Assembly System, or by the appropriate compiler, and consists of the object program in the form required by the Relative Load Routine together with all of the necessary information for purposes of loading and modification. The Program File may be in either Relative Object Code or in Absolute Object Code.

Each ROC Program File is made up of blocks of 256 words which are grouped into four types of records. These records appear in the following order:

1. Identification Record
2. Modification Record
3. Program Record
4. Termination Record

Each record consists of one or more blocks as required by its specific function. The AOC Program Files contain all of the above records except for the Modification Record.

All but three types of tags are eliminated from programs which are assembled in Relative Object Code. These tags are:

1. Program Name: This is the unique symbolic name (12 Fieldata characters) assigned to the program instruction portion of an object program.[1]

   The Program Name is used by the Executive System and/or the Relative Load Routine to identify the program, and to serve as the relative zero address of the instruction portion (IBANK) of the program. The absolute assignment for this tag is given to the Relative Load Routine either by the Executive System in the case of EXEC ROC, or by the Location Input in the case of DIRECT ROC (see Section VI). This absolute assignment then becomes the first location assigned to the program instructions in core memory. All tags other than the Program Name which might appear in the IBANK of the source program are given relative assignments with respect to the Program Name.

---

[1]For object programs produced by SLEUTH this is the symbolic label which is obtained from the 6-character tag of the PRO line of the source program. This tag is expanded to 12 characters with the right-most 6 characters filled with space codes.

2. Data Table Tags: Each ROC program file may contain one or more Data Table Tags. This tag serves as the relative zero address of a data table. The absolute assignment for this tag, specified either by the Executive System or by the Location Input, becomes the first location assigned to the data table in the core memory or the magnetic drum memory. All other locations within the table are made relative to the Data Table Tag.

3. Data Table Length Tag: Each Data Table Tag in the ROC program file has a Data Table Length Tag associated with it which represents the minimum length of the data table. The table length may be increased at load time either through the Executive System or through the Location Input. The absolute assignment given to this tag becomes the current length of the data table in core or drum memory.

Programs assembled in AOC do not contain any symbolic assignments and do not have to be modified by CLAMP.

The four types of records which make up an object Program File are briefly described in the remainder of this section.[2]

A. Identification Record

The Identification Record contains the Program Name and a code identifying the class of program and the type of ROC (DIRECT or EXEC). The program is classed as a simple program, a complex program, or as a subroutine.

B. Modification Record

This record contains four types of tables which contain information necessary to modify the object program for its operating environment. The types of tables that may appear in the Modification Record are:

1. Input/Output References
2. System References
3. Drum References
4. Core References

---

[2]The Appendix contains a detailed description of the ROC and AOC Program Files.

As the source program is assembled, each symbolic reference within the program is assigned a reference number. These reference numbers replace all symbolic references in the Program Record.

The tables of the Modification Record contain lists of the symbolic references of the source program together with the reference numbers which have replaced them.

The tables of the Modification Record are used by the Relative Load Routine to generate a Reference List which will include the absolute assignments for all symbolic references. The Reference List is arranged in four sections corresponding to the four tables of the Modification Record. The reference numbers of the Modification Record correspond with entries in the Reference List. The operation of the Relative Load Routine with respect to the Modification Record is discussed in Section III. The following paragraphs contain a brief description of the contents of the Modification Record.

1. Input/Output References: This portion of the Modification Record contains a list of all symbolic references to the peripheral equipment.[3] These references are either for the Executive I/O Functional Routines (EXEC ROC) or for direct input/output (DIRECT ROC). The references for Executive I/O are two-word entries which contain the symbolic reference for the peripheral equipment, an equipment type field denoting the type of unit (UNISERVO* IIIA Tape Unit, Card Reader, etc.), and a logical channel grouping.

   All I/O references in DIRECT ROC programs must have absolute assignments at assembly time. The I/O reference table for these programs is in two parts, one containing channel references and their absolute assignments, and the other consisting of unit references and their absolute assignments.

2. System References: Four separate tables may be included in this section of the Modification Record. These tables include symbolic references to the Executive System, undefined symbols used in subroutines, and lists of associated subroutines for the object program.

---

[3]With the exception of magnetic drums.

*Trademark of Sperry Rand Corporation.

a. Executive System References: A list of one-word symbolic references to the Executive System, the Executive I/O Functional Routines, and the Relative Load Routine.

b. Subroutine List: A list of the Program Names of all subroutines referenced by the program, but not incorporated into it at assembly time.[4] These symbols are used at load time to load the required subroutines from the library tape. This list is contained in complex programs and subroutines only.

c. External Reference List: This portion of the System References table lists symbolic entrances to the subroutines in the Subroutine List. The absolute address for these entrances are assigned as the subroutines are loaded from information contained in the library tape directory. This list is contained in complex programs and subroutines only.

d. Entrance List: This table is contained only in the program files for subroutines. It contains a list of the symbolic entrances to the subroutine (excluding the subroutine Program Name). Each two-word entry contains the mnemonic symbol for the entrance and the address of the entrance relative to the subroutine Program Name.

3. Drum References: This table contains a list of all Data Table Tags and Data Table Length Tags which reference the magnetic drum(s) together with the assigned minimum length for each drum table.

4. Core References: Each three word entry in this table contains a core Data Table Tag, its associated Data Table Length Tag, and the assigned minimum length of the table.

C. Program Record

All object program instructions and data words are contained in the Program Record. This portion of the object program File is arranged in two parts: segment sections and data table sections.

---

[4] In SLEUTH the subroutine names are indicated in the XREF line of the source program.

1. Segment Sections: A segment section contains the set of instructions that comprise an object program segment as well as the associated data words (DBANK), if any. From the point of view of the source program, the program segment contains all words generated in the IBANK and DBANK.[5]

2. Data Table Sections: These parts of the Program Record contain the set of words for each variable length data table (DTABLE) assigned at assembly time.[5] CLAMP will place these data table words in the core memory locations allocated to the table.

Both the segment sections and the data table sections contain two types of words: program words and modification indicator words.[6] The program words are those instruction or data words which are generated by the Assembly System from the information contained in the source program. These words contain fields such as function codes which are fixed by the source program, and fields such as core memory address references which are to be modified by the Relative Load Routine. The modifiable fields will contain reference numbers which refer the Relative Load Routine to the Reference List which it generates from the information contained in the Modification Record.

The Modification indicator words contain fields to denote the type of modification necessary.

D. Termination Record

This record is the last block of the Program Record. It contains the address of the first object program word to be executed, relative to the Program Name. (This is the address given in the ENDPRO declarative of a SLEUTH source program.)

---

[5]See Data Table Conventions, Section IIIB.

[6]Modification indicator words are not contained in AOC programs.

III. PROGRAM MODIFICATION TECHNIQUES

A. Modifiable Fields

For programs assembled in Relative Object Code, certain fields within the generated program words are necessarily incomplete and must be modified by CLAMP at load time. From the source program, an assembler must construct modification indicators describing how a field is to be modified. The assembler must also produce the Modification Record which defines the source code tags from which the modifiable fields are generated.

The Relative Load Routine modifies references in program words to reflect internal program references, core data table references and their associated table lengths, drum table references and their associated lengths, I/O references, and external references, i.e., those pertaining to subroutines and other programs.

1. Internal Program References: Modifiable fields which contain internal program references are converted to their absolute form by adding the current address assigned to the Program Name. The word forms and word fields that can be modified are indicated in Table 1.

WORD FORM	MODIFIABLE FIELD(S) BIT POSITIONS
Whole Word	15-0
Half Word	15-0, 33-18
Instruction Word	15-0
I/O Access Word	15-0, 33-18
Variable Format Word	15-0[7], 33-18[7]

TABLE 1: MODIFIABLE FIELDS

2. Core Table References: The modifiable fields of the ROC program which contain core table references are of the forms indicated in Table 2. With the exception of the 3rd and 4th forms shown, the reference

---

[7] If these positions form a single section of the word.

numbers for the current tag assignments are contained within the field being modified. In the 5th form, the current value of the field is the sum or difference of the values associated with the reference numbers contained in that field.

1	Data Table Tag
2	Data Table Length Tag
3	Data Table Tag ± Constant
4	Data Table Length Tag ± Constant
5	Data Table Tag ± Data Table Length Tag

TABLE 2: MODIFIABLE FIELD FORMS

In the case of the 3rd and 4th forms shown, the current value of the field being modified is the sum or difference of the value associated with the reference number and the constant. In all field forms of this type, the constant is contained in the field being modified. The tag reference number is either in the field being modified or in the indicator field associated with that program word, depending on the value of the constant.

For core table references, both the tag reference number and the constant are in the field being modified if the constant is less than 2^6. If the constant is equal to or greater than 2^6, the tag reference number is carried in a nine bit area of the associated indicator field.

3. Drum Table References: Magnetic drum references are actually input/output references. They are discussed separately, however, because the manner in which drum tables are used parallels that for core tables.

   The modifiable field forms for drum references are as shown in Table 2. The constant and tag reference number are in the field being modified if the constant is less than 2^{12}. For other values of the constant the tag reference number is in a nine bit area of the indicator field associated with the program word.

The drum Table Length Tags are handled differently depending on their position in the program word. Length tags which occur in the right half of a program word are replaced by their current assignments in a 23-bit field (positions 22-0) at load time. When a length tag occurs in the left half of a program word, it is replaced by the 16 least significant bits of its current assignment (in positions 33-18 of the program word) at load time. The most significant seven bits of the assignment are ignored.

Drum references are discussed again in the following paragraphs.

4. Input/Output References: The absolute assignments for I/O references are made by the Executive System for EXEC ROC programs or through the Location Input for DIRECT ROC programs.

   a. EXEC ROC Programs: In object programs which operate under control of the Executive System, a single symbolic reference is used to denote both the unit and the channel for the peripheral equipment. At load time, this symbol is replaced by a 30-bit field (positions 29-0) which contains the current channel and unit assignment. The channel assignment is in positions 29-26.

      For drum Data Table Tag references, the current drum address is consigned to bit positions 22-0. Both of these assignments are shown in Figure 2. Note that bits 23-16 of the program word for peripheral equipment references are not modified The contents of this field can therefore be assigned at assembly time. The unit address is indicated by master bit selection.

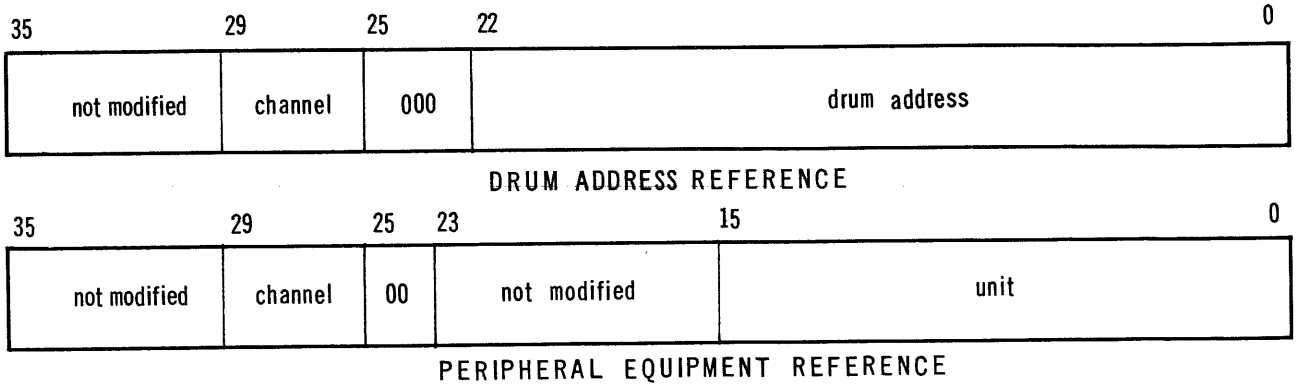


FIGURE 2: EXEC ROC REFERENCES

b. DIRECT ROC Programs: In object programs which operate independently of the Executive System, all I/O references are specified with a channel symbol and a unit symbol. Both of these symbols are given absolute assignments at assembly time, but these assignments may be changed at load time to assignments contained in the Location Input.

The symbolic channel tag is replaced by a four-bit absolute channel number in positions 25-22. Each channel used in the source program must be given a unique symbol.

A channel tag in the source code may be used to refer to an I/O Access Control Word, in film memory, associated with the specified channel. References of this type are converted at load time to the absolute address of the control word associated with the channel assignment given to the channel tag.

Symbolic unit references for the magnetic drum(s) are treated differently from those which reference other peripheral equipment. Symbolic drum address references are replaced by a 23-bit field (positions 22-0) at load time. References to peripheral equipment other than magnetic drums are replaced by a 16-bit field which contains the master bit selector for unit designation.

Two or more I/O references may be equated at assembly time.

5. Selective Jump Switch References: The symbolic Selective Jump Switch references are replaced by a 4-bit absolute switch number in positions 25-22 at load time. Each jump switch used must be given a unique symbol. Up to fifteen different Selective Jump switches may be specified. In DIRECT ROC programs only, they may be used interchangeably with channel symbols since both are of the same nature.

6. External References

a. Executive System References: Object programs which operate under control of the Executive System will make reference to it or to the Executive I/O Functional Routines during their operation. These references will be in the form of specific System Tags (e.g., $XIO) and will include I/O requests, closeout operations, and error procedures.

At assembly time, the System Tags are replaced by reference numbers to a system symbol table in the Reference List and the actual mnemonic tag is stored in that table. The reference numbers are replaced with absolute assignments at load time.

b. Control Routine References: Two system symbols are provided for DIRECT ROC programs. These symbols reference a program closeout sequence and an error routine. These routines are called by a control routine which can be loaded by CLAMP during loading of the object program. The control routine is discussed in Section IV.

B. Data Table Conventions

The core area assigned to an object program is divided into three sections: the instruction section (IBANK), the fixed-length data section (DBANK), and the variable-length data section (DTABLEs). The fixed-length data section contains items such as constant pools and fixed-length tables. Like the instruction section, the fixed-length data section is one continuous block of core whose length is determined at assembly time.

Data tables in the variable-length data section of core or drum are defined by a unique symbolic Data Table Tag and by a Data Table Length Tag. The length tags need not be unique for each table but only one value must be assigned for each unique length tag. The value assigned to the length tag at assembly time is the minimum length of the table. This minimum value may be incremented at load time by specifying a table length increment in the Job Request (for EXEC ROC programs) or in the Location Input (for DIRECT ROC programs).

The Relative Object Code defines each data table as an independent table. Symbolic internal table tags are not allowed. Provision is made for equating the starting addresses of two or more tables. If this is done, then the subject tables are listed consecutively in the Reference List produced by the Relative Load Routine (see below). The minimum length tag must be specified at assembly time.

The variable-length data section of core may contain as many tables as is desired subject to the conventions discussed in the preceding paragraph.

C. Reference List

The Relative Load Routine generates a Reference List which contains the absolute assignments for all symbolic references. The Reference List is composed of four sections corresponding to the sections of the Modification Record.

Each Reference List section contains a maximum of 128 one-word entries except for the drum reference section. The latter contains two-word entries (for a maximum of 256 words). The four sections of the Reference List are:

1) I/O References

2) System References

3) Drum References

4) Core References

IV. OPERATIONAL FUNCTIONS

The Relative Load Routine utilizes the Executive I/O Functional Routines in order to perform its own input/output operations.[8] The words generated by an assembler for a program segment are modified and read into core at the execution position. For each IBANK or DBANK area of the object program, one block is written on tape or on drum if storage is specified. Whenever a program segment is written on a storage medium, that area of core to which it is assigned is cleared. Any section not requiring storage remains in core and the initial operating section, i.e., the first program segment to be executed, must therefore be loaded last or not be written over by a succeeding section. After initial modification, loading, and storage of the program by the Relative Load Routine, the program itself must read succeeding segments.

When DIRECT ROC programs are loaded, the location of the object program is specified by the Location Input. The Relative Load Routine locates the object program on the input medium and then the loading process is initiated. When EXEC ROC programs are loaded, the location of the object program is specified in the Job Request. The program is then located on the input medium by the Executive System before control is transferred to the Relative Load Routine for loading.

A. Control Routine (DIRECT ROC Programs)

A control routine is provided by the Relative Load Routine to handle program termination and error occurrences for DIRECT ROC programs. The termination portion of the control routine checks that all input/output operations have been completed and informs the computer operator accordingly. The Executive System can then be reloaded. The error section of the control routine provides an error indication to the operator followed by a computer halt. The termination section can then be entered manually.

The control routine occupies a specific core memory area which must be reserved by object programs which make use of it. The loading of the control routine must be specified in the Location Input so that the Relative Load Routine may load it while loading the program.

---

[8] In the situation where the Executive System is not present (DIRECT ROC programs), the I/O Functional Routines are included in CLAMP.

B. $PARAM and $ERROR Tables

Two special tables can be defined for ROC programs. These are the Input Parameter Table and the Error Interrupt Table and are designated by the table names $PARAM and $ERROR respectively.[9]

1. $PARAM Table: A program may require a set of input parameters to determine or select options of execution. Accordingly, the Job Request or the Location Input may contain one or more input parameter records. These records, which may take the form of punched cards, contain 11 words of six alphanumeric characters each. A maximum of ten such records are permitted in an object program.

   The input parameter words are transferred into the $PARAM table of the object program if and only if such a table was defined at assembly time. This table must be defined in the source program by using the reserved label $PARAM as a Data Table Tag.

   If the $PARAM table is not large enough to accommodate the input parameters, or is not present and an attempt is made to load such parameters, then an appropriate type-out will notify the operator of this event.

2. $ERROR Table: Each program to be run under Executive System control must contain an image of the core memory Error Interrupt locations (addresses 192-199). This image is the $ERROR table and contains entrance addresses to error recovery subroutines within the object program.

   Entries for the $ERROR table must be specified at assembly time for programs to be run under Executive control. All unspecified entries are cleared to zero.

   The $ERROR table is assigned storage in the instruction area (IBANK) of core. No check is made by the Relative Load Routine for the presence of a $ERROR table for DIRECT ROC programs. The $ERROR table in this case is loaded as specified in the object program. It is recommended, however, that DIRECT ROC programs include a $ERROR table.

---

[9]The table names "$PARAM" and "$ERROR" must be present if these tables are to be acceptable to CLAMP.

C. Facility Assignment Notification

When all memory and input/output facilities have been assigned to a program, the operator is notified through a typeout. The format of the typeout is different for EXEC ROC and DIRECT ROC programs.

1. EXEC ROC Programs: The format for facility assignment notification is shown in Figure 3. Four different typeouts are represented for

   a) Selective Jump switch assignments

   b) I/O equipment assignment

   c) Drum memory assignments

   d) Core memory assignments

The symbols used in Figure 3 are as follows:

JOB REQUEST ID is the alphanumeric identity of the Job Request (see manual on 1107 Executive System).

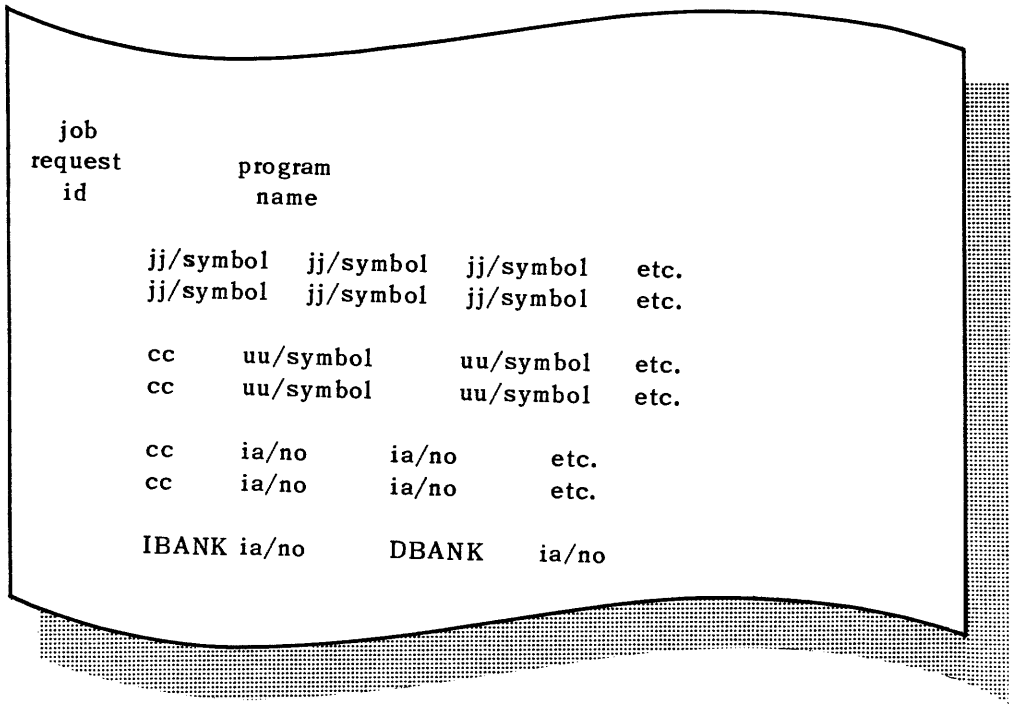


FIGURE 3: FACILITY ASSIGNMENT NOTIFICATION

PROGRAM NAME is the alphanumeric tag assigned to the program (see Section II).

jj is the current Selective Jump switch assignment. This is a decimal number from 01 to to 15.

symbol is the symbolic reference used in the program to reference jump switches or peripheral units.

cc is a decimal number from 00 to 15 denoting the current I/O channel assignment.

uu is a decimal number from 00 to 15 denoting the current peripheral unit assignment.

ia is the initial address for a drum table, or instruction block.

no is the number of words in the subject block

IBANK is the symbolic notation for a core memory instruction block.

DBANK is the symbolic notation for a core memory data block.

The Selective Jump switch assignments are printed five to a line. If any jump switch cannot be assigned the jj field is replaced by **.

Each different channel in the I/O equipment assignments starts a new line. The unit assignments associated with a specific channel follow the channel number five to a line. If any peripheral equipment cannot be assigned, the unit number is replaced by **.

The third group of typeouts shown in Figure 3 illustrates the format for drum allocation. Each block of drum, which may contain one or more drum tables, allocated to the program is typed out. Each different drum channel starts a new line. The assigned drum areas on a channel follow the channel number three to a line. The ia and no fields are each seven digits in length. The ia field is replaced by ******* if a drum block cannot be assigned.

The last line of Figure 3 is used to indicate the core areas assigned to the object program. The ia and no fields are each five digits in length. In the event the core area indicated in the Job Request is too small, the ia field is replaced by *****.

Following the facility assignments as indicated in Figure 3, a message is printed to verify the status of facilities. This message has one of the following forms:

a. FACILITIES ASSIGNED: This message is printed when all facilities have been assigned and the facility requirements contained in the Job Request were adequate.

b. FACILITIES ASSIGNED WITH CORRECTIONS: This message is typed when the facility requirements in the Job Request were insufficient to contain the program, but additional jump switches, I/O units, or drum areas were available to assign to the program. No indication is made as to which facility items were corrected.

c. FACILITIES CANNOT BE ASSIGNED: This message is used when Job Request facility requirements are insufficient and the additional assignments could not be made due to one or more of the following reasons:

(1) Insufficient peripheral units on assigned channel.

(2) Insufficient drum area on assigned channel.

(3) Requirements for core areas are insufficient.

When facilities cannot be assigned, the loading is terminated and control is returned to the Executive System for appropriate action.

2. DIRECT ROC Programs: The typeout for facility assignment notification for these programs is similar to the format shown above for Executive controlled programs. The first group shown in Figure 3 is used for both Selective Jump switch assignments as well as I/O channel assignments. The second group is used only for I/O unit assignments and the cc field is replaced by the symbol "UN".

The typeout for drum assignments (the third group of Figure 3) indicates drum blocks only, without channel designation. The cc field contains the symbol "DR".

The last line is used for core assignments. In this case, however, more than one IBANK or DBANK area may be assigned by Location Input so more than two indications may be typed out. These assignments are typed three to a line.

The message "PROGRAM LOADED" is typed following the loading of the program. The program is then initiated following receipt of an appropriate message from the operator indicating that the required I/O units are loaded.

V. LOADING AND MODIFICATION OF COMPLEX PROGRAMS

Provision is made in the Relative Load Routine for the incorporation of subroutines into an object program at load time.[10] A program requiring the addition of subroutines is called a main program. The main program and its associated subroutines is called a complex program. Complex programs may, or may not, be segmented.

A. Subroutines

The Relative Object Code for subroutines must have certain characteristics in order to facilitate their incorporation into a program at load time.

1. Directory Card: A Directory Card is produced for each subroutine when it is assembled. This record contains the following information in a compact list which becomes a part of the Library Tape Directory and is made available to the Relative Load Routine:

   a. Subroutine Name[11]

   b. Subroutine entrances

      (1) Symbolic entrance (tag)

      (2) Entrance address relative to the Subroutine Name.

   c. Names of all subsidiary subroutines referenced by this subroutine.

   d. Length of IBANK for this subroutine.

   e. Length of DBANK for this subroutine.

2. Symbol Definition: Each symbol used in a subroutine must be defined by that subroutine. The sub routine may consist at most of one IBANK, one DBANK and one set of variable-length data tables.

   a. DBANK: Only one fixed-length data section is allowed in a subroutine. This core section is assigned to the locations which precede the fixed-length data section of the referencing master program. The object program is thus given a single fixed-length data section.

---

[10] The Retrieval section of the 1107 LIBRARIAN is an integral part of CLAMP.

[11] The Subroutine Name is analogous to the Program Name of the main program.

b. Common Data Tables: All data tables used by the subroutine must be defined in the subroutine's variable-length core data section. All drum tables used by the subroutine must be similarly defined within the subroutine. All tables thus defined in either the variable-length core data section or on the magnetic drums are considered as common data tables.

The referencing main program must also define every common data table which is defined in a subroutine. All common core data tables must be defined in the variable-length data section of the main program. Common drum tables must be defined in the drum table section of the main program.

c. I/O Equipment: Like data tables, all I/O equipment used in common by a main program and the subroutines referenced by it must be defined in both the main program and the subroutine. I/O equipment which is used only by a subroutine need only be defined within that subroutine. The facility requirements for the total object program however, must include all I/O equipment referenced by the subroutines.

d. Subroutine Entrances: Provision is made in the Relative Object Code for multiple entrances to the subroutine. All of these entrances, excluding the Subroutine Name, must be defined symbolically as well as relative to the Subroutine Name. The Relative Object Code for subroutines contains a special section for these entrances.

e. External References: All subroutines may contain symbolic entrances to other subroutines.

3. Restrictions: Certain restrictions are made in order to facilitate the incorporation of subroutines into an object program at load time.

a. Closed Subroutines: The Relative Load Routine handles only closed subroutines at load time since the relative location of the subroutine in the program is not fixed. Open subroutines must be incorporated into an object program at assembly time.

b. Segmentation: Subroutines may not be segmented. All segmentation occurs within the main program.

B. Complex Programs

The Modification Record of the main program is loaded first and the Reference List is generated. The main program subroutine list (generated from the XREF line of the source code) is then submitted to the retrieval section of the 1107 LIBRARIAN. The subroutines are then loaded from the library tape followed by the Program Record of the main program.

1. Data Tables: Each complex program has only one fixed-length core data section. The fixed-length data section of the main program is preceded by the fixed-length data sections of all referenced subroutines. All common core data tables, i.e., tables used in common by both main program and subroutines, are placed in the variable-length core data section of the object program. All of these common core data tables must be defined within the main program. These definitions are incorporated into the subroutines during loading. The minimum table lengths specified at assembly time are honored. If the table length as specified in the main program is less than that given in the subroutine, an error indication is presented.

   The Data Table Tags defined in the main program are perpetuated throughout the entire loading procedure. All common data tables in the referenced subroutines, are therefore defined in terms of the main program specifications. The Data Table Tags defined within the subroutine are used only during loading and are not perpetuated. If a data table which was not defined in the main program is encountered during subroutine loading, an error indication will be given.

   The $PARAM and $ERROR tables, when present in the main program, are treated as common data tables. All drum data tables are also considered as common tables and are handled in the same manner as are core tables.

2. Instruction Section: The instruction section (IBANK) of the main program is preceded by the instruction sections of the subroutines. The object program will then contain only one instruction section after loading. All references to the main program are specified relative to the main Program Name.

3. I/O References: All common I/O references, i.e., those used in common by both main program and subroutines (or by two or more subroutines), are defined in the main program. These definitions are incorporated into the subroutines as they are loaded. Logical channel assignments specified in the main program are followed. Logical channel assignments in the subroutines are therefore not honored.

   Common I/O references are specified by using the same symbol in the main program and in the subroutine or by equating symbols in the main program. The equation of I/O reference symbols may be made only on the main program level. The I/O references defined in the main program are perpetuated throughout the loading procedure. This is not the case with I/O definitions within subroutines, however, and these are used only during subroutine loading.

   The I/O equipment necessary for the operation of a complex program is specified in the main program I/O facility requirements. These facilities include common I/O equipment as well as equipment used by a subroutine alone.

C. Segmented Complex Programs

Provision is made for loading segmented complex programs. The subroutines within the program are never segmented; only the main program may be segmented. In segmented complex programs, the main program is divided into a main control routine and one or more segment control routines.

1. Main Control Routine: This section of the object program must remain in core at all times during execution of the segmented complex program. This routine provides the control necessary for transferring program segments into core for execution.

   The main control routine contains all definitions associated with the entire main program. These definitions include:

   a. All common core and/or drum tables used by the main control routine, the segment control routines, and/or the associated subroutines.

b. All common I/O references used by the main control routine, the segment control routines, and/or the associated subroutines.

c. All noncommon I/O references used by the master control routine and/or the segment control routines.

d. The Subroutine Names of all subroutines associated with the main control routine.

e. The entrances to subroutines (other than Subroutine Name) associated with the main control routine.

The subroutines associated with the main control program are loaded and remain in core at all times during execution.

These subroutines may be referenced by the main control routine, the segment control routines, or by any of the subroutines within a segment.

The main control routine may reference any segment control routine as well as any subroutine associated with the main control routine. It may not, however, reference a subroutine associated with a segment control routine.

2. Segment Control Routine: Each segment of the main program contains a string of instructions which may be augmented by one or more subroutines (incorporated at load time) and which control the subroutines contained within that segment. This group of instructions comprises the segment control routine.

The segment control routine and all associated subroutines must be recorded on a program specified storage medium after they have been converted to absolute form. This storage medium should be referenced only by the main control routine. The segment control routine should not refer to the storage for that segment or for any other segment. All read-in and placement of segments is primarily under control of the main control routine.

The segment control routine contains definitions for the Subroutine Names and other entrances for all subroutines associated with it. Each segment control routine which has associated subroutines must specify a fixed-length core data section (DBANK). This section must include at least one non zero word. The DBANK of the segment control routine is located relative to the DBANK of the master control routine.

The DBANK of all associated subroutines is then located following the DBANK of the segment control routine.

The segment control routine may reference the main control routine or any other segment control routine. It may also reference any subroutine with it or with the main control routine. It may not, however, reference a subroutine associated with another segment control routine.

VI. ALLOCATION AND OPERATION OF PROGRAMS

The nature of the program is specified on the PTY card of the Job Request, i.e., whether the program is to be run under Executive control (EXEC ROC) or whether it is to be run independent of the Executive System (DIRECT ROC).

A. EXEC ROC Program Initiation

Programs which are to be run under control of the 1107 Executive System and which utilize the Executive I/O Functional Routines are initiated via a Job Request submitted to the Executive System. The Executive System will read the first block on the input medium specified by the PTY card of the Job Request; this block for EXEC ROC programs is the Program Identification Record. The Executive System will then transfer the peripheral equipment location of the input medium, the number of words in the block just read, and the core address of the block, to CLAMP.

B. EXEC ROC Core and Drum Allocation

All core and drum allocation for EXEC ROC programs is handled by the 1107 Executive System.

C. DIRECT ROC Program Initiation

If the Executive System is not in control, DIRECT ROC programs can be initiated by loading the Relative Load Routine by the standard bootstrap procedure. The peripheral equipment "address" of the program input medium is specified by means of a keyboard entry. The first block of the input medium may be the Program Identification Record or it may be Location Input. If the first block of input is not Location Input, then Location Input for the subject program does not exist.

The keyboard entry has the form

t, cc, uu.

where t denotes the type of input medium. This may be (1) the symbol "T" to designate UNISERVO IIIA, or

(2) the symbol "A" to designate UNISERVO IIA, or

(3) the symbol "D" to designate magnetic drum

(4) the symbol "P" to designate that the Location Input is on paper tape. In this case the uu field is omitted.

(5) the symbol "C" to designate that the Location Input is on punched cards. In this case, the uu field is omitted.

cc denotes the absolute channel for the input medium (one or two decimal digits)

uu denotes the input medium unit (one or two decimal digits.) In case of magnetic drum, this field is a decimal drum address of up to 7 digits.

For DIRECT ROC programs initiated through the Executive System, the Executive transfers the peripheral equipment "address" of the input medium to the Relative Load Routine. Section VII contains a description of Location Input.

D. DIRECT ROC Core and Drum Allocation

Core and drum allocations are assigned as indicated in the Location Input. Provision is made, however, for the Relative Load Routine to load programs and make assignments in cases where these specifications are omitted from the Location Input.

1. Instruction Area: The Program Name is assigned the core address indicated in the Location Input if a core address is specified on the LAB Card. In the absence of such specification, the Program Name is assigned the first available location in one core bank.

2. Core Data Tables: The core data tables are assigned addresses as indicated in the Location Input. In the absence of all or part of the address assignments, the core data tables specified in the Location Input are assigned first. All other tables are assigned to available locations, first in the core bank other than the one to which instructions have been assigned, and then to the same core bank if necessary.

3. Drum Tables: These tables are assigned drum addresses as indicated in the Location Input. In the absence of all or part of the address assignments, the tables specified by the Location Input are assigned first. All other drum tables are then assigned to available drum areas.

4. CLAMP Area: The core area used by CLAMP may be overlaid by the object program. In this case the portion of the object program which is to overlay the Relative Load Routine must be loaded as a segment of the object program and must be recorded on a storage medium. If a DTABLE is to use this area, it cannot have constants loaded in it at load time.

VII. LOCATION INPUT

Location Input contains information used to place a DIRECT ROC program in core memory together with instructions for modification of data tables and of program environment to suit a particular run. In the absence of Location Input, all table and I/O assignments made at assembly time are honored by CLAMP.

Location Input must contain the Program Name and the peripheral equipment "address" of the Program File. In addition, it may contain any or all of the following:

1. Absolute address for the Program Name
2. Absolute address for any core or drum table
3. Any table length incrementation
4. Any peripheral equipment reassignment
5. Any input parameters

Location Input may be in one of two formats: card format or paper tape format.

A. Location Input Card[12] Format

Location Input in card format must be preceded by a START Card and followed by a STOP Card (see manual on 1107 Systems Conventions.) The START Card is illustrated in Figure 4 and the STOP Card in Figure 12.

Location input in card format consists of one or more cards. The Label Card (LAB) must be present in each Location Input, and must follow the START Card. An Address Card (ADD) is used to assign absolute addresses to data tables in core or drum. A Table Length Card (TAL) is used to indicate increments to core or drum Data Table Lengths. The Peripheral Cards (PER) contain absolute reassignments for I/O channel and unit references. Parameter Cards (PMn) are used for programs requiring input parameters.

All cards except for the LAB card are optional and are present in Location Input only if required. The order of card types following the LAB card is immaterial.

1. Label Card (LAB): Figure 5 illustrates the form of the LAB Card. The LAB Card is the basic card for every Location Input and consists of up to eleven ordered fields separated by commas. Two

[12] The term "card" is used here to indicate a unit record which can take the form of punched cards or magnetic tape blocks.

consecutive commas must be used to show the omission of a field except when trailing fields are ommitted. The end-of-card is indicated by a period following the last field. All spaces and blanks are ignored. The fields and information contained therein are as follows:

RUN ID: From one to six Fieldata characters which identify the program run.

CARD TYPE: The characters LAB to identify the card as a Label Card.

PROGRAM NAME: From one to twelve Fieldata characters which identify the program.

INPUT TYPE: This field defines the input medium of the program. It contains either:

1) The symbol "T" to denote a UNISERVO IIIA tape unit.
2) The symbol "A" to denote a UNISERVO IIA tape unit.
3) The symbol "D" to denote a magnetic drum.

INPUT CHANNEL: This field contains the absolute channel number of the input medium.

INPUT UNIT: The absolute tape unit number or drum address of the input medium.

IBANK LOCATION: Absolute address of the IBANK, one to five decimal digits.

DBANK LOCATION: Absolute address of the DBANK, one to five decimal digits.

SUBROUTINE INPUT TYPE: This field defines the type of input for subroutines associated with complex programs and may contain any of the symbols described above for INPUT TYPE.

SUBROUTINE CHANNEL: The absolute channel number of the subroutine input medium.

SUBROUTINE UNIT: The absolute tape unit number or drum address of the subroutine input medium.

The IBANK and DBANK LOCATION fields and the subroutine fields are optional. All other fields must be included on the LAB Card. If the DBANK assignment is specified, the IBANK must be assigned. In the absence of the IBANK or DBANK absolute assignments, they are placed in core memory in a dynamic manner (see Section VI.D.).

2. Address Card (ADD): Figures 6 and 7 illustrate the form of the ADD Card. This card must appear in Location Input when absolute core address assignments for data tables are to be made. The card contains a RUN ID field which is identical to the one on the LAB Card, a CARD TYPE field containing the symbol ADD to identify the card, and a number of ADDRESS ASSIGNMENT fields separated by commas. These latter fields are in the format:

Data Table Tag/Address

where the Data Table Tag is the one used in the source code and the address is from one to seven decimal digits indicating an absolute address assignment. The absolute addresses for core tables are treated as modulo 2^{16} and those for drum tables as modulo 2^{23}. These tables may be core or drum data tables according to their source code definition. If more than one card is needed to assign all Data Table Tags, the cards are repeated in the same format. The ADDRESS ASSIGNMENT fields can not be split between two cards. The last field on each card must be followed by a period. A maximum of 128 Data Table Tag assignments are allowed.

The ADDRESS ASSIGNMENT fields must be arranged on the cards in separate order groups, i.e., one group for core and one for drum. The address assignments within each group must appear in ascending order.

3. Table Length Card (TAL): Figure 8 illustrates the form of the TAL Card. This card specifies increments to the minimum data table lengths established at assembly time. The card contains a RUN ID field identical to the one on the LAB Card, a CARD TYPE field containing the symbol TAL, and a number of TABLE LENGTH INCREMENT fields separated by commas. These fields are in the format:

Data Table Length Tag/Increment

where the Data Table Length Tag is the one used in the source code and the increment is from one to seven decimal digits indicating an increment to the minimum table length. The length increments are treated as modulo 2^{16} for core tables and as modulo 2^{23} for drum tables. If more than one card is needed to assign all Data Table Length Tag increments, the cards are repeated in the same format. The TABLE LENGTH INCREMENT field can not be split between two cards. The last field on each card is followed by a period. A maximum of 128 Data Table Length Tags are allowed.

4. Peripheral Card (PER): Figure 9 illustrates the form of the PER Card. This card specifies reassignment of I/O channel and unit references. The card contains a RUN ID field identical to the one on the LAB Card, a CARD TYPE field containing the symbol PER, and a number of I/O ASSIGNMENT fields separated by commas. These fields are in the format:

Reference/Assignment

where the reference is the one used in the source code, and the assignment is one or two decimal digits indicating an absolute channel or unit number. All absolute assignments are treated as modulo 2^4. If more than one card is need to assign the I/O references, the cards are repeated in the same format. The I/O ASSIGNMENT field cannot be split between two cards. The last field on each card is followed by a period. A maximum of 128 I/O reference assignments are allowed.

5. Parameter Card (PMn): Figures 10 and 11 illustrate the form of the PMn Card. A program may require a set of input parameters to determine or select options of execution. These parameters are entered by the PMn cards. The cards contain a RUN ID field identical to the one on the LAB Card, a CARD TYPE field containing the symbol PMn, where n is a decimal digit from 0 to 9, and a PARAMETERS field which contains 66 Fieldata characters. The 66 characters are those which immediately follow the comma after the PMn field.

A maximum of ten PMn cards are allowed for any one Location Input. The cards must be numbered and appear sequentially as PMØ, PM1, ..., PM9. However, PMn cards within a sequence may be omitted. The parameters are transferred to the object program's $PARAM table, six characters per word, 11 words per card. Unspecified characters will be assigned the Fieldata space character. If a card is omitted in the sequence, the result will be 11 words of binary zero in the $PARAM table.

B. Location Input Paper Tape Format

The term "sentence" is used below to indicate a unit record on paper tape. A paper tape sentence may contain up to 480 Fieldata coded characters except where otherwise noted. The sentence must be followed by a period. Spaces, carriage returns, and non-printing characters are not included in the count of the total number of characters.

The Location Input paper tape must be headed by

LOCINP.

and followed by a carriage return. The information required in paper tape format is identical to that described above for card format.

Figure 13 contains an example of paper tape Location Input.

1. Label Sentence (LAB): The LAB Sentence contains the same information as the LAB Card plus a carriage return following the period at the end of the sentence. The information for the LAB Sentence must be contained within 80 characters.

2. Address Sentence (ADD): The ADD Sentence contains the same information as the ADD Card. A period followed by a carriage return designates the end of a sentence. If more than one sentence is needed to assign all Data Table Tag addresses, then another ADD sentence is typed.

3. Table Length Sentence (TAL): The TAL Sentence contains the same information as the TAL Card. A period followed by a carriage return designates the end of a sentence. If more than one sentence is needed to assign all table length increments, then another TAL Sentence may be punched.

4. Peripheral Sentence (PER): The PER Sentence contains the same information as the PER Card. Up to six lines of 80 characters each may be typed in one sentence. A period followed by a carriage return designates the end of a sentence. If all of the I/O references cannot be assigned in one sentence of six lines, another PER sentence is typed.

5. Parameter Sentence (PMn): The PMn Sentence contains the same information as the PMn card. The 66 characters transferred are those which immediately follow the comma after the PMn field. Since carriage return is a permissable parameter character, up to 67 frames may be necessary to complete a PMn sentence. All 66 characters must be specified for each PMn Sentence. Up to 10 PMn Sentences are allowed.

6. End of Location Input: The end of Location Input is signalled by the paper tape stop code (see Figure 13).

LOCATION INPUT CARD FORMAT

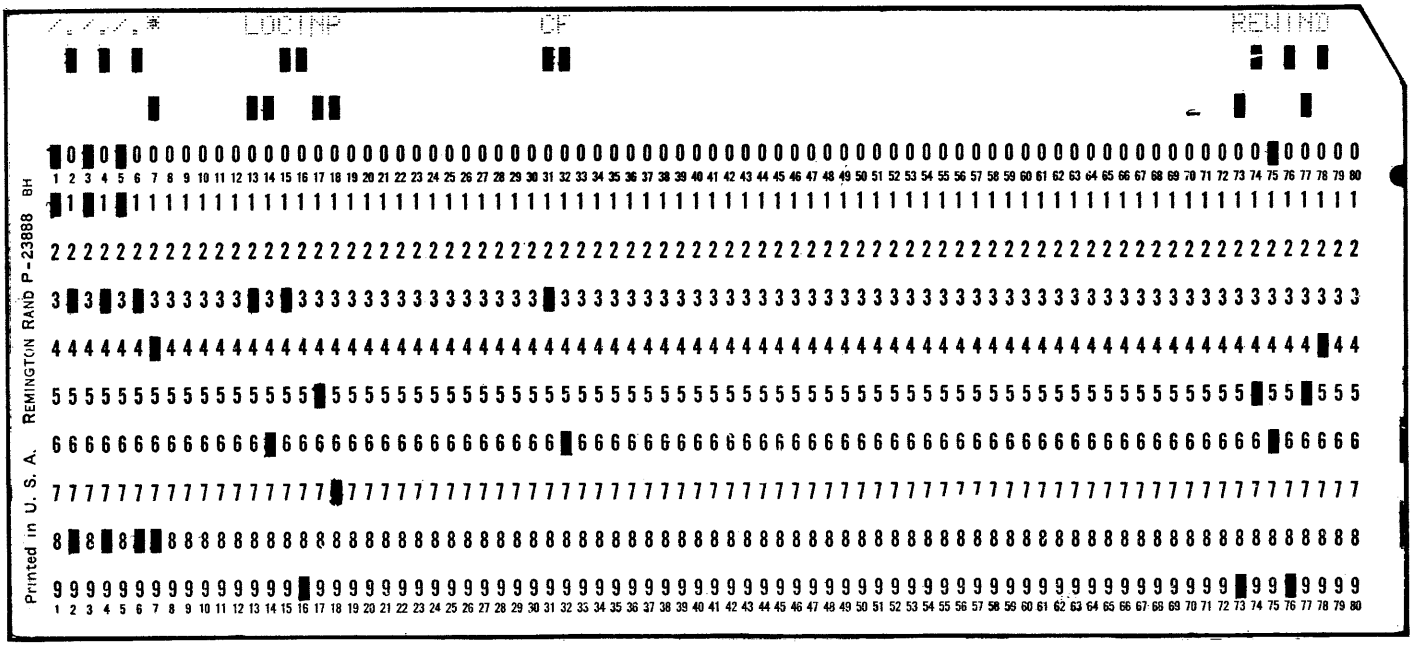


FIGURE 4: START CARD

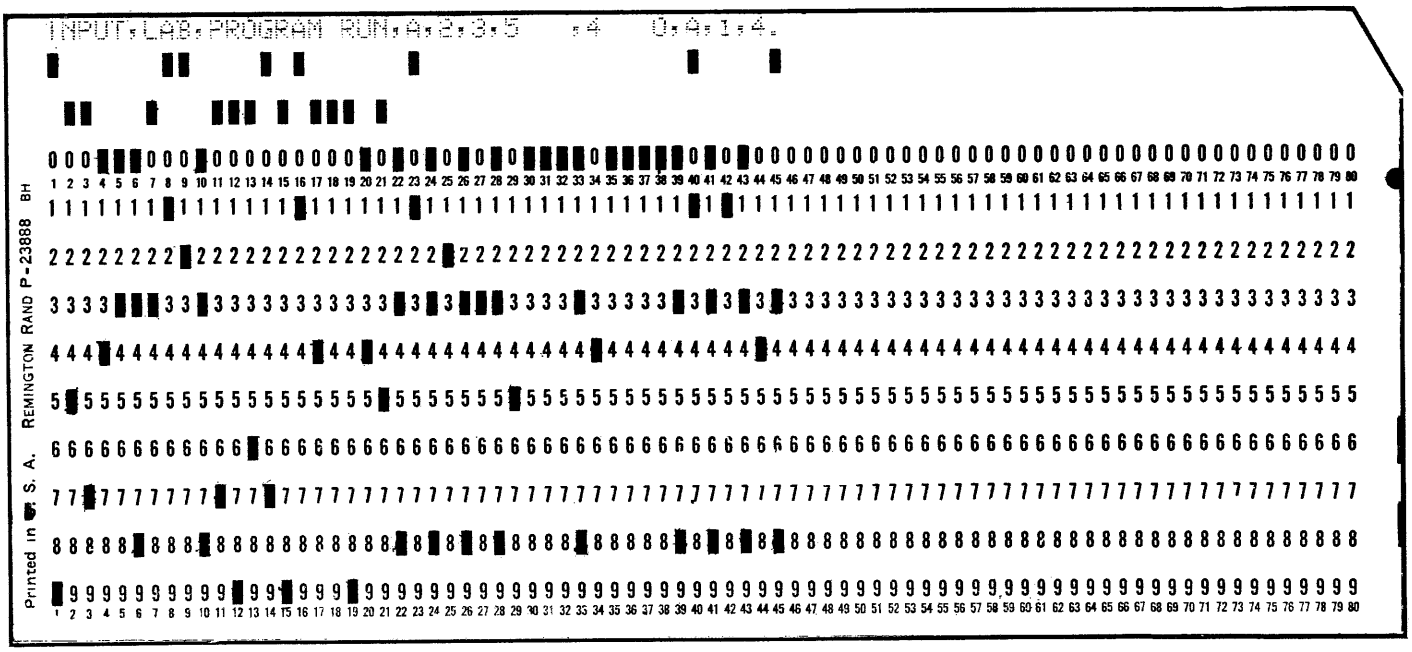


FIGURE 5: LAB CARD

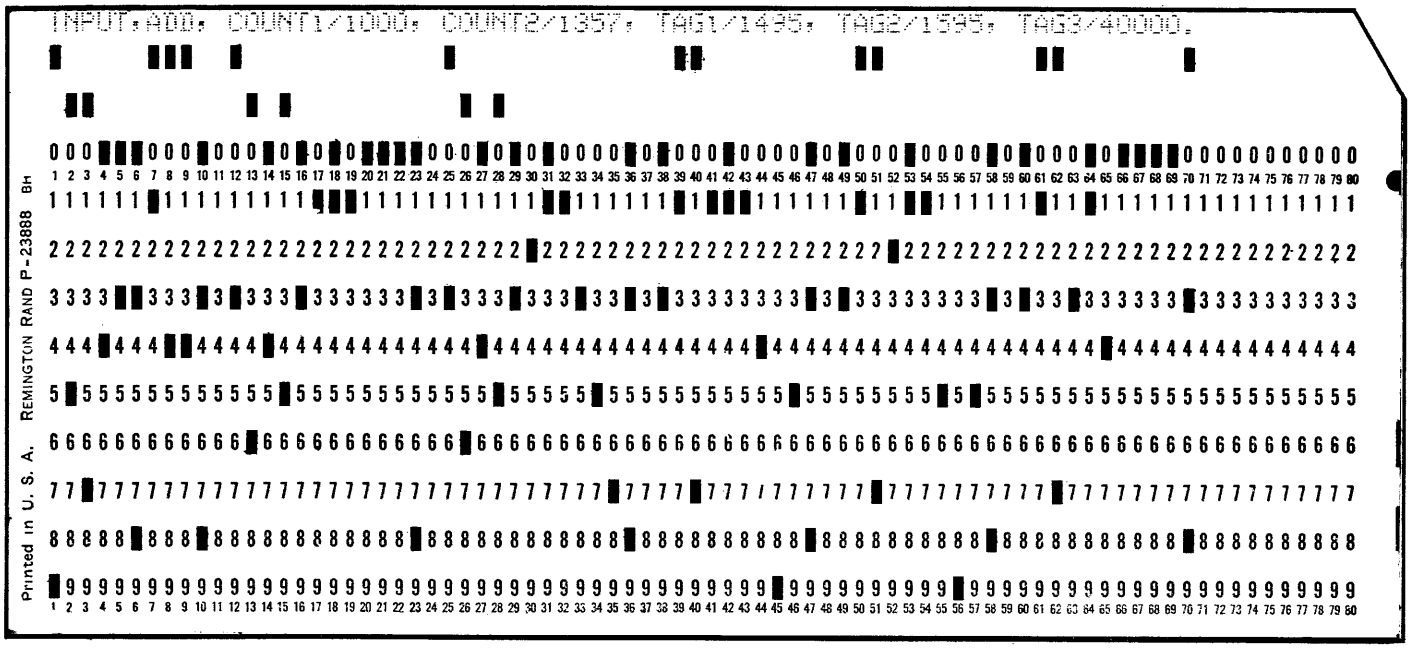


FIGURE 6: ADD CARD (1)

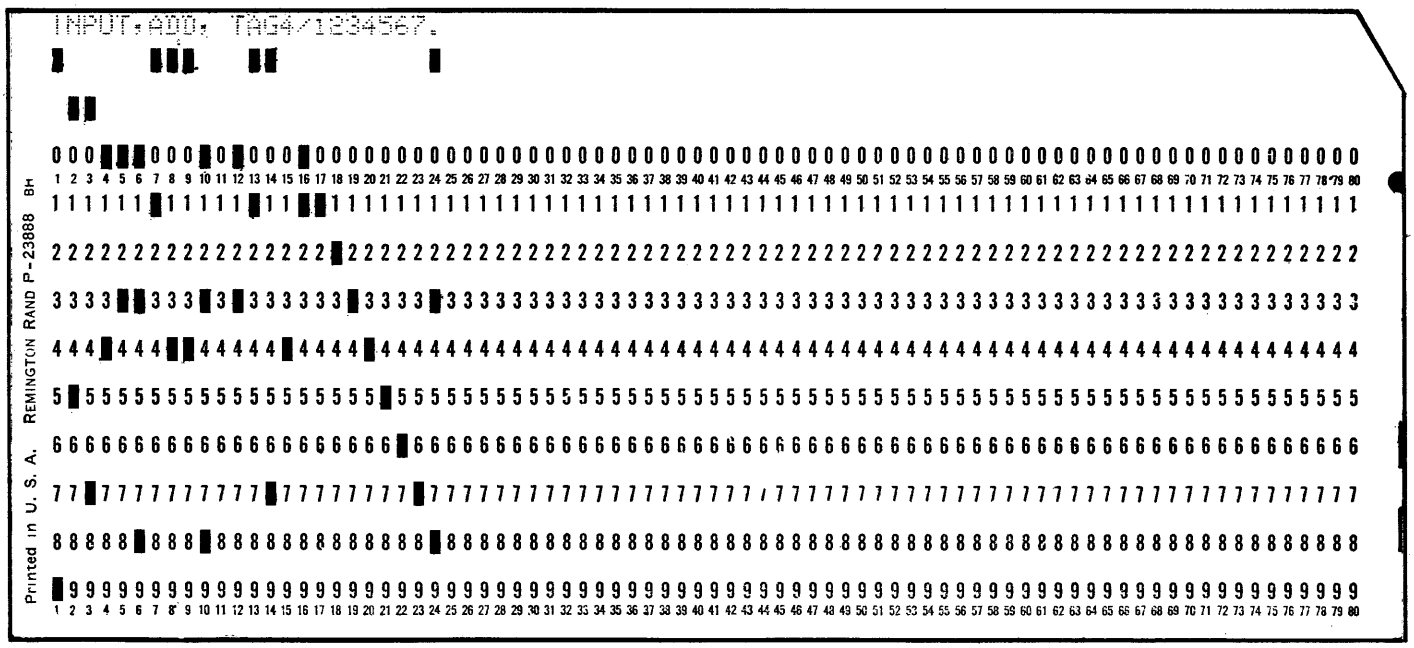


FIGURE 7: ADD CARD (2)

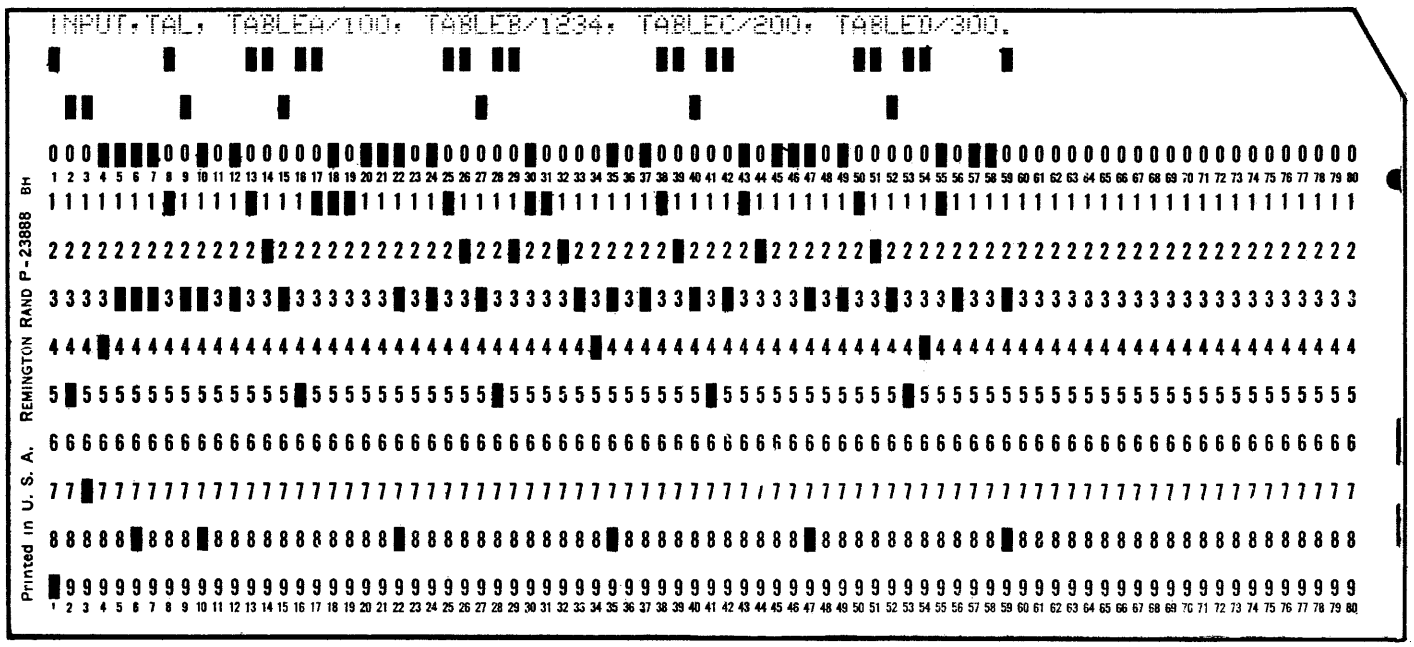


FIGURE 8: TAL CARD

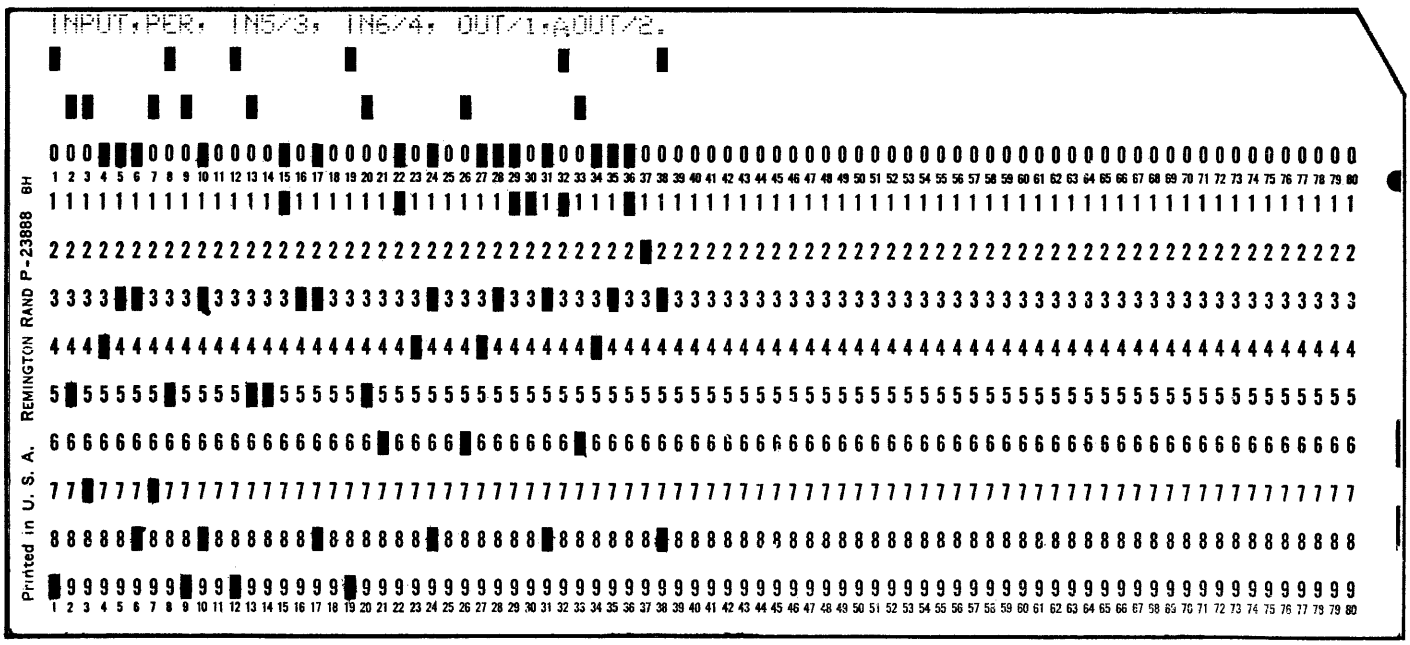


FIGURE 9: PER CARD

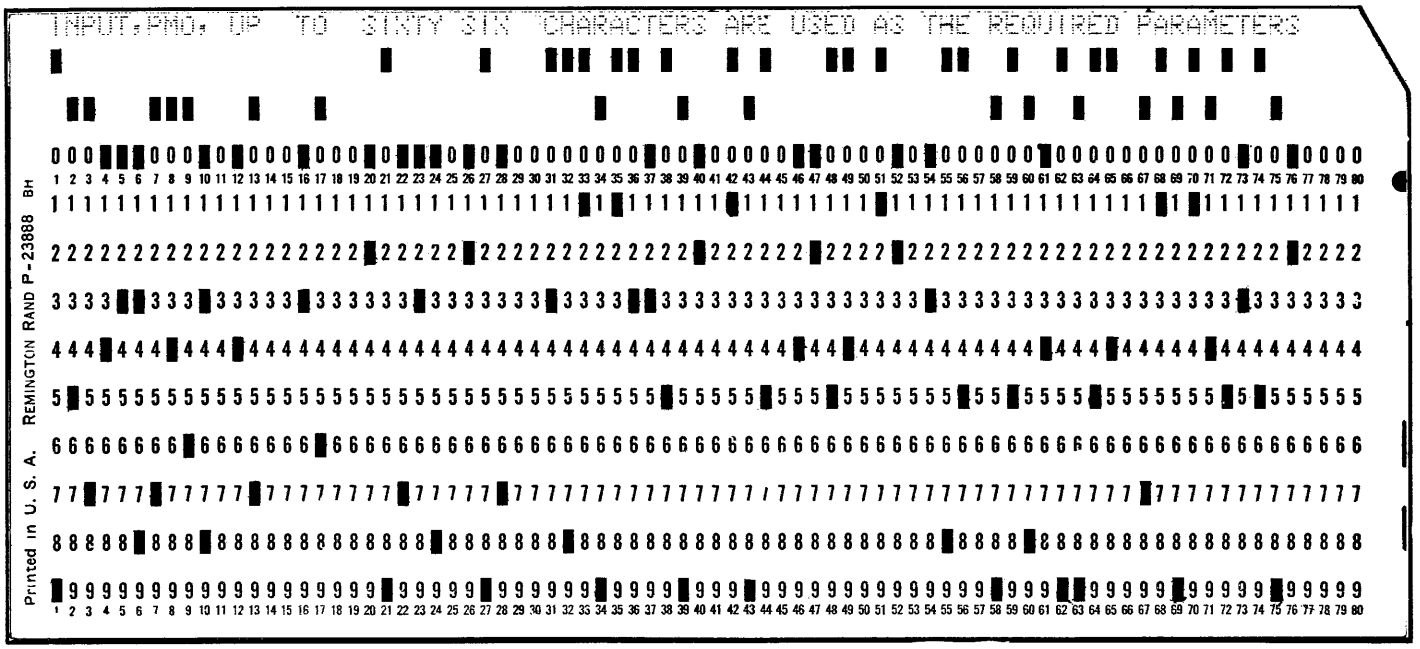


FIGURE 10: PMØ CARD

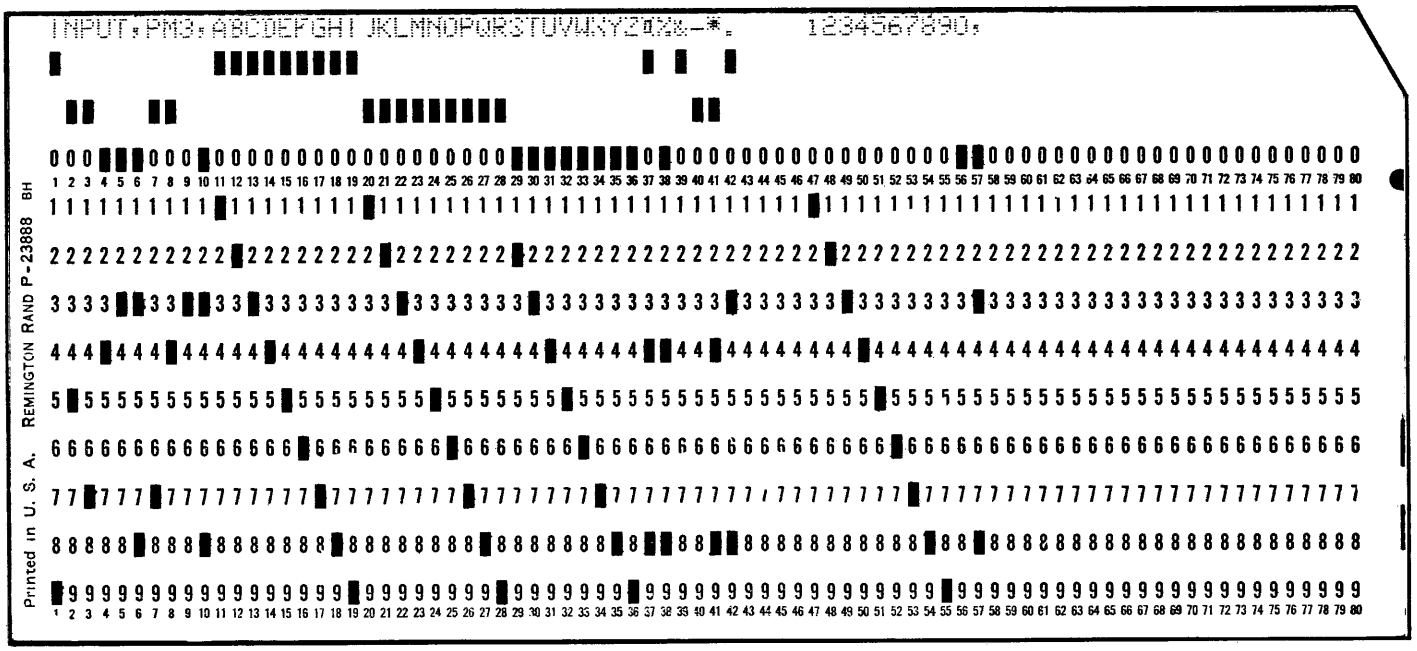


FIGURE 11: PM3 CARD

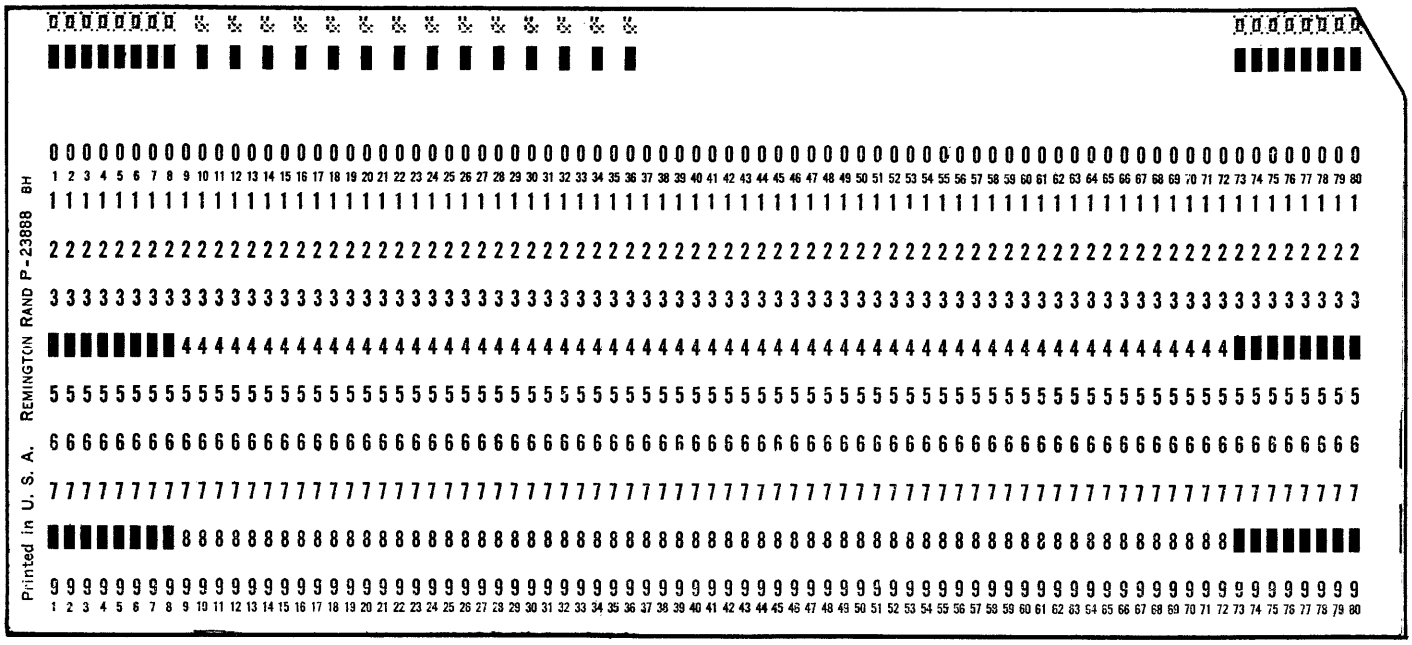


FIGURE 12: STOP CARD

CLAMP 40

An example of Paper Tape Location Input appears below. The first line is the header sentence "LOCINP." The second line is an LAB sentence which identifies the run by "INPUT" and the program by "PROGRAM-RUN." The program is located on UNISERVO IIA tape unit number 3 on channel number 2. The program IBANK is located at decimal address 5000 and its DBANK at 40000. The program requires subroutines located on channel 1, UNISERVO IIA unit 4. The third line is the beginning of an ADD sentence which occupies six lines. The last line contains the stop code symbol denoting the end of Location Input.

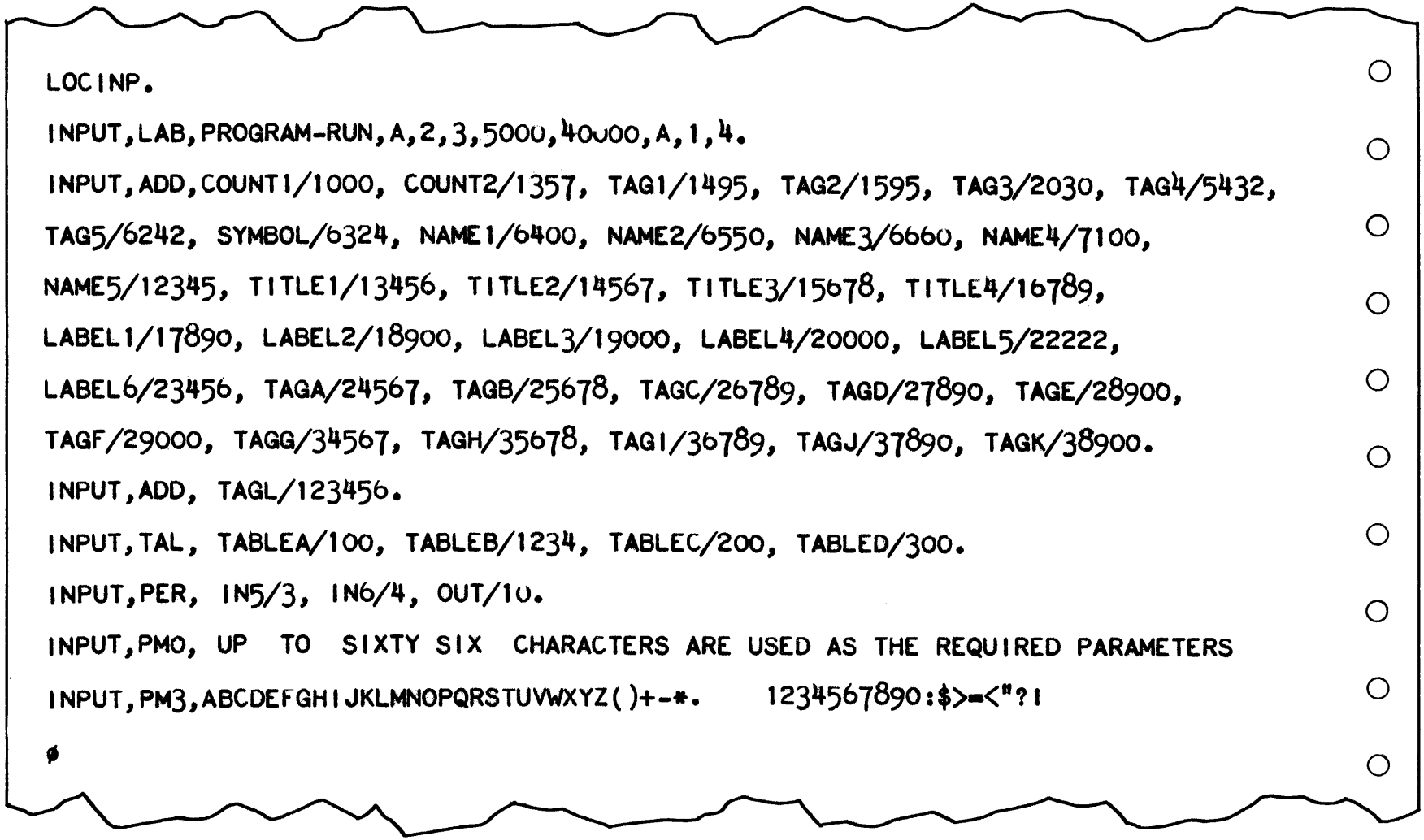


FIGURE 13: PAPER TAPE LOCATION INPUT

APPENDIX: RELATIVE OBJECT CODE

I. INTRODUCTION

This Appendix contains a detailed description of the 1107 Relative Object Code. For purposes of clarity and brevity, the following two symbols have been used throughout to represent a Data Table Tag and a Data Table Length Tag, respectively:

DTAG	The unique symbolic Data Table Tag assigned for each data table in the object program. The absolute assignment of this DTAG is the first location assigned the data table in core memory or on magnetic drum.
LTAG	The symbolic Data Table Length Tag. The absolute assignment of this LTAG is the current length of the data table in core memory or on magnetic drum.

II. DESCRIPTION OF ROC PROGRAM FILE

In general, an object program that is to be loaded by the Relative Load Routine is contained in a Program File. The Program File is an output of an assembly system and/or a compiler. A complete layout of the Program File is shown in Figure 1. Although the object program need not contain all of the sections listed, those sections which are present must be in the order shown.

The Program File consists of four records in the following order:

1) Identification Record
2) Modification Record
3) Program Record
4) Termination Record

Each record in the Program File consists of one or more blocks of 256 words each. The first word in each block identifies the record with which the block is concerned. The identifier word for the Identification Record is the first six characters of the Program Name. The identifier words for the other Program File records are:

Record Type	Identifier Word
Modification	*MODRC
Program	*PRORC
Termination	*TRMRC

Each block contains a checksum. The checksum for the Identification Record block is the next to last word of the block. For all other records, it is the last word of the block.

All unused words in the Program File blocks are disregarded. No special setting is required for these words. However, the contents of these disregarded words are included in the block checksum calculation.

The checksum is generated as follows:

1) The sum of all words, except the checksum word, in the block is calculated disregarding overflow.
2) The upper half of the checksum is added to the lower half (without sign extension) thus generating a 19-bit checksum in the least significant bits.
3) This 19-bit checksum is stored in the checksum word of the block.

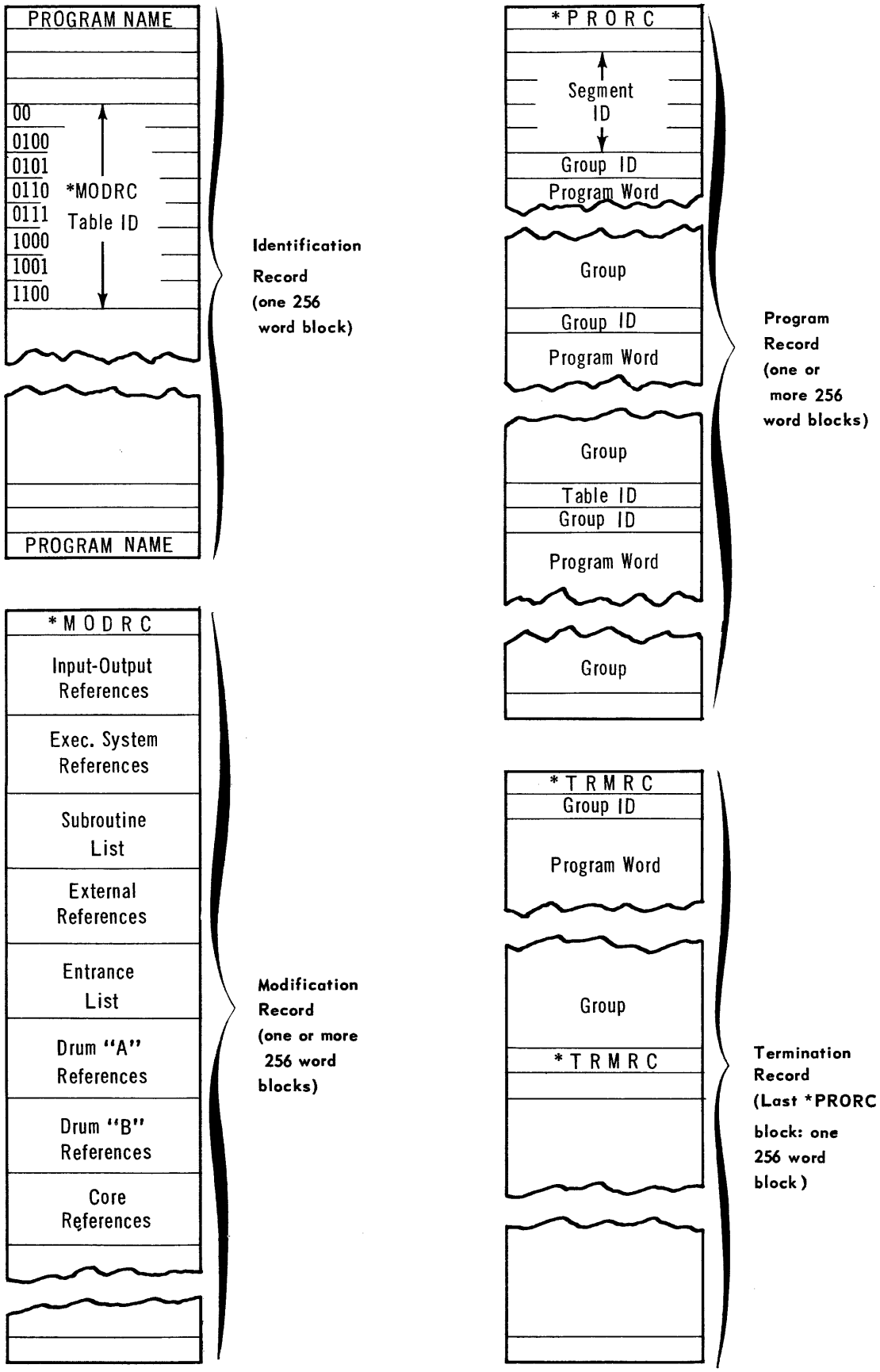


FIGURE 1: PROGRAM FILE FORMAT

III. IDENTIFICATION RECORD

The first record in a Program File is the Identification Record. This record is composed of one 256 word Label Block. The Label Block is illustrated in Figure 2.

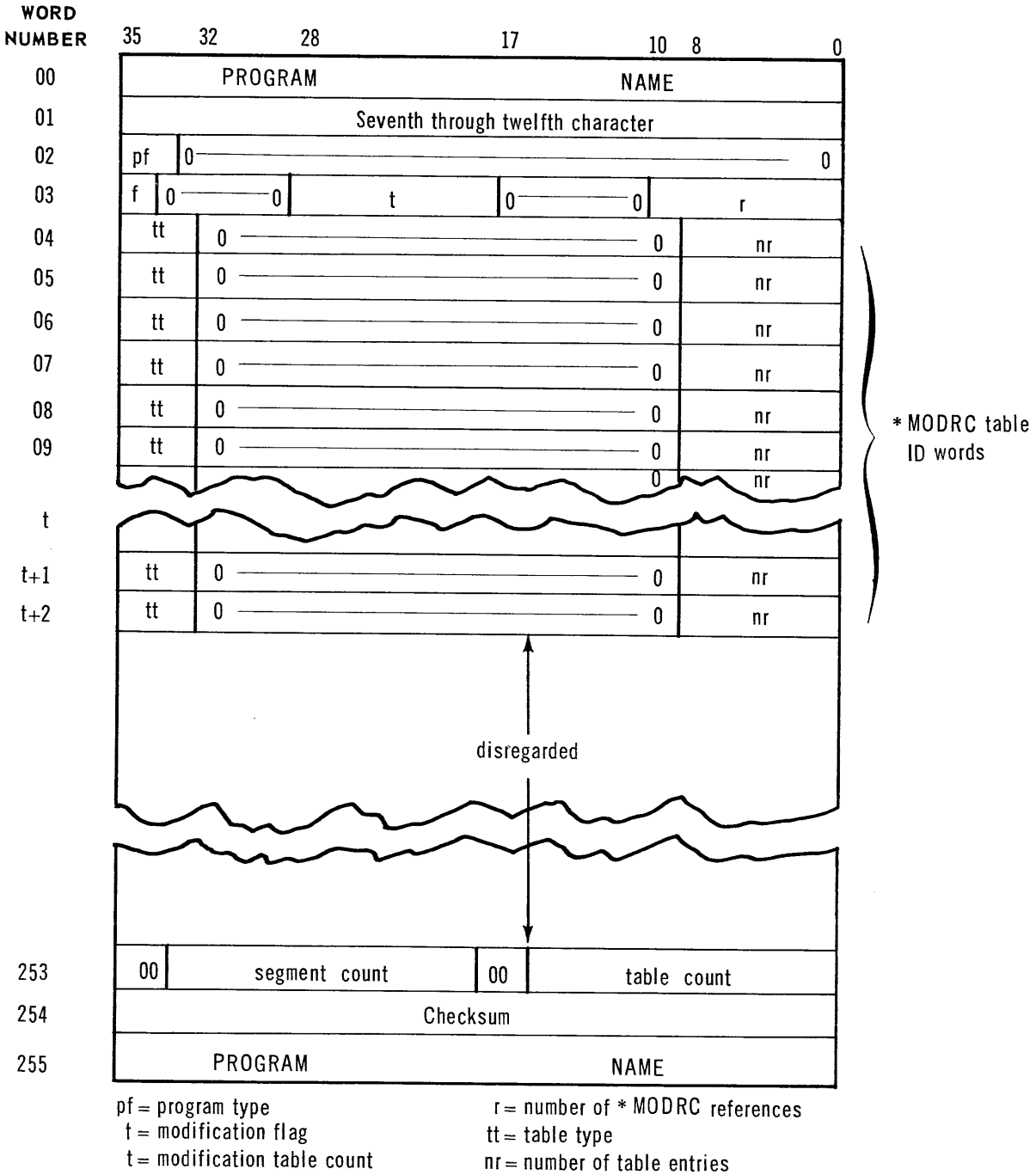


FIGURE 2: LABEL BLOCK

Words 00 and 01 contain the Program Name. The Program Name is a combination of 1 to 12 of the following characters, in Fieldata code:

A, B, ... , 0, ... , Z, Ø, 1, ... , 9

The Name is left justified and space filled to contain 12 characters. The hyphen (-) may also be used in the name, but it cannot be the first character.

Word 02 contains the program type flag in bits 35-33. This flag has the following values:

Program type flag (pf)	Object Code	Type ROC	Program Type
0 0 0	Absolute	DIRECT	Simple
0 0 1	Relative	DIRECT	Simple
0 1 0	Relative	DIRECT	Subroutine
0 1 1	Relative	DIRECT	Complex
1 0 0	Relative	None	Subroutine
1 0 1	Relative	EXEC	Simple
1 1 0	Relative	EXEC	Subroutine
1 1 1	Relative	EXEC	Complex

Programs and subroutines with flag fields 101, 110, 111 operate under Executive System control. Subroutines with flag field 100 do not contain input/output references and can be used in EXEC ROC or DIRECT ROC programs.

Word 03 describes the modification tables associated with the Program File. These tables are contained in the *MODRC. Bit 35 is the modification flag. This flag is 0 if table lengths may be incremented at load time. It is 1 if incrementation of table lengths is not allowed. The modification table count, bits 28-18, indicates the number of modification tables in the *MODRC. This value is the number of *MODRC table ID words which follow. Bits 10-00 indicate the number of entries in all modification tables. This number is the total of the values in the nr field of all *MODRC table ID words.

The *MODRC table ID words, words 04 and following, describe the *MODRC tables associated with the Program File. Bits 08-00 indicate the number of entries for the subject table. The table type is described by bits 35-32 as follows:

Type Flag (tt)	Table Type	Type References
0000	I/O reference	Selective jump switch
0001	I/O Reference	Executive I/O Functional Routines
0010	I/O reference	Direct I/O, channel reference
0011	I/O reference	Direct I/O, unit reference
0100	System reference	Executive System reference
0101	system reference	subroutine list
0110	system reference	external references
0111	system reference	entrance list
1000	drum reference	drum channel A*
1001	drum reference	drum channel B*
1010	drum reference	drum channel C*
1011	drum reference	drum channel D*
1100	core reference	

*Note: Bits 33-32 of the drum reference flag denote the logical channel assignments. If two or more drum channels are specified in the object program, there is a separate drum reference table for each channel. The difference is denoted by a change in bit 33 and/or bit 32 in the drum reference flag.

All the words from the word following the last *MODRC table ID word up to and including word 252 are disregarded.

Word 253 describes the Program Record (*PRORC) associated with the Program File. Bits 33-18 indicate the number of program segments in the program. Bits 15-00 indicate the number of variable-length tables which have entries in the *PRORC. The remaining bits of the word are set to zero.

Word 254 is the block checksum.

Word 255 is a repeat of the first six characters of the Program Name.

IV. MODIFICATION RECORD

The Modification Record (*MODRC) is composed of one or more blocks of 256 words each. Figure 3 illustrates a Modification Record block. Word 00 contains the identifier, *MODRC, in Fieldata code. Word 255 is the block checksum. Each block is composed of 254 modification table entry words. If an entry of more than one word will not fit at the end of a *MODRC block, it will be split between the present block and the next block. Unused words in the last MODRC block are disregarded.

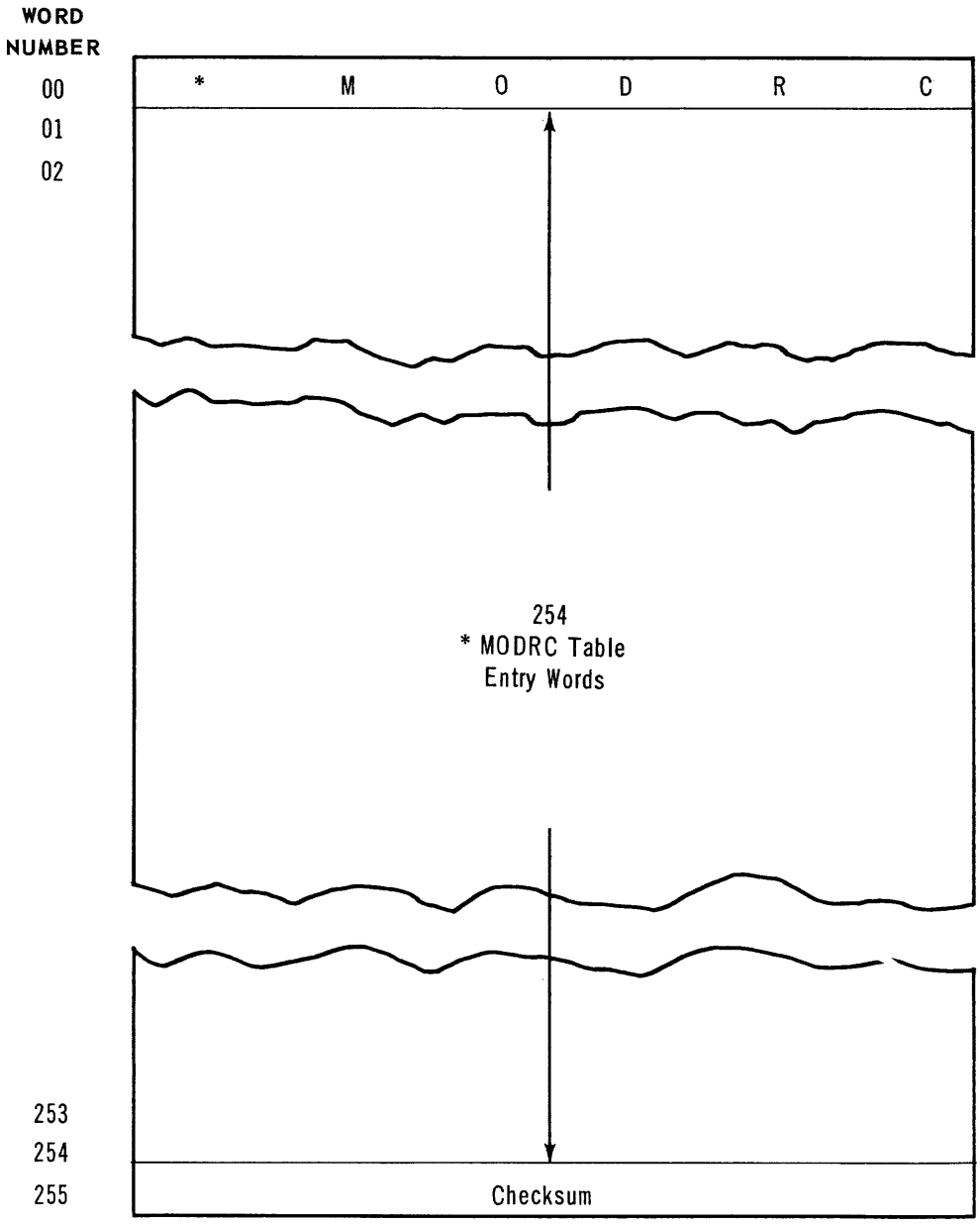


FIGURE 3: MODIFICATION RECORD BLOCK

The entries in the *MODRC blocks comprise four groups of modification tables. The contents of these tables are used to modify the object program for its operating environment. The four groups of tables that may appear are:

Input/Output References
System References
Drum References
Core References

All of these tables need not be included with every object program; however, those tables which are present must appear in the order shown above.

A. REFERENCE NUMBERS: As a program is assembled, each symbolic reference contained in the modification tables is assigned a reference number. Thus each symbolic reference in the source code is replaced by a reference number. These reference numbers will correspond to entries in one of the four modification table groups. The modification tables contain the absolute assignment for each symbolic reference. The reference numbers for each modification table group are an independent set.

B. INPUT/OUTPUT REFERENCES: Symbolic references to peripheral equipment and Selective Jump switches are contained in the input/output section of the *MODRC. This section of the *MODRC may contain up to 128 unique entries of two words each. The first word is the symbolic reference. The logical assignment of the equipment is in the second word. The jump switch entry is illustrated in Figure 4. Figure 5 illustrates the EXEC ROC entry. The DIRECT ROC entries are illustrated in Figures 6 and 7.

For object programs in EXEC ROC the 128 allowable references are available to represent Selective Jump switches and input/output units. All Selective Jump switch references must appear in the first 16 I/O references.

For DIRECT ROC programs, all symbolic input/output channel references must be included in the 128 allowable references. The channel references may only appear in the first sixteen entries in the input/output references. Selective Jump switch references and channel references must share the first 16 references. Symbolic channel references and Selective Jump switches may be used interchangeably.

The status flag is in bits 35-34 of the second word of the input/output *MODRC entry. The status flag applies to all EXEC ROC reference entries and DIRECT ROC unit reference entries. The status flag may also be associated with Selective Jump switch entries in EXEC ROC programs. The status flag may not be used for channel and jump switch entries in DIRECT ROC programs.

The status flag values are:

Status flag(s)	Definition
00	May not be deleted at Load time.
01	May not be deleted at Load time; equate this reference to last preceding reference.
10	May be deleted at Load time.
11	May be deleted at Load time; equate this reference to last preceding reference.

In addition to Selective Jump switch references, the input/output section of the *MODRC contains one of two types of input/output references. These references are either for the Executive I/O Functional Routines or for direct controlled input/output.

1. SELECTIVE JUMP SWITCH REFERENCES: In object programs which utilize the Selective Jump switches, the table containing these references appears first in the *MODRC input/output section. This table is limited to 16 entries. The reference numbers assigned to these entries must be the first numbers assigned in the input/output section.

   Entries in this table are composed of two words. The first word is the symbolic Selective Jump switch reference. The contents of the second word varies with the type of input/output operation used with the object program. For EXEC ROC programs the second word contains the logical switch number. For DIRECT ROC programs, the second word contains the absolute Selective Jump switch assignment made at assembly time.

   The second word contains three fields. The status flag is in bits 35-34. The Selective Jump switch number is in bits 25-22 and is repeated in bits 03-00. The Selective Jump switch entry is illustrated in Figure 4.

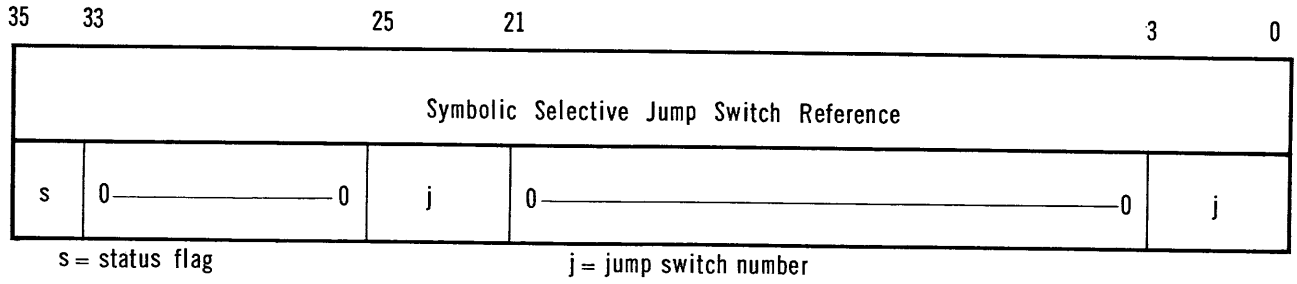


FIGURE 4: SELECTIVE JUMP SWITCH ENTRY

2. EXECUTIVE SYSTEM INPUT/OUTPUT REFERENCES: FOR EXEC ROC programs, each entry in the input/output reference table, Figure 5, contains two words. This table contains entries for all peripheral equipment with the exception of magnetic drums. The first word is the symbolic reference for the peripheral equipment. The second word contains the status flag in bits 35-34, the peripheral equipment type in bits 21-18, the logical channel grouping in bits 15-12, and the use flag in bit 00.

The logical channel field contains values from 1 through 15. A logical channel of zero is used to indicate that the unit may be assigned to any available channel. This field, together with the equipment type field, is used to define the absolute channel. A different set of logical channel numbers is assigned for each type of peripheral equipment.

The equipment type field has one of the following values:

Equipment Type Field	Equipment Referenced
0 0 0 1	UNISERVO IIA
0 0 1 1	UNISERVO IIIA
0 1 0 1	High-Speed Printer
1 0 0 1	Card Reader
1 0 1 0	Card Punch
1 1 0 1	Paper Tape Reader
1 1 1 0	Paper Tape Punch

The use flag, bit 00, indicates the use of the peripheral equipment such that if it is 0, the equipment is used for input. For output or buffer usage, the use flag is 1.

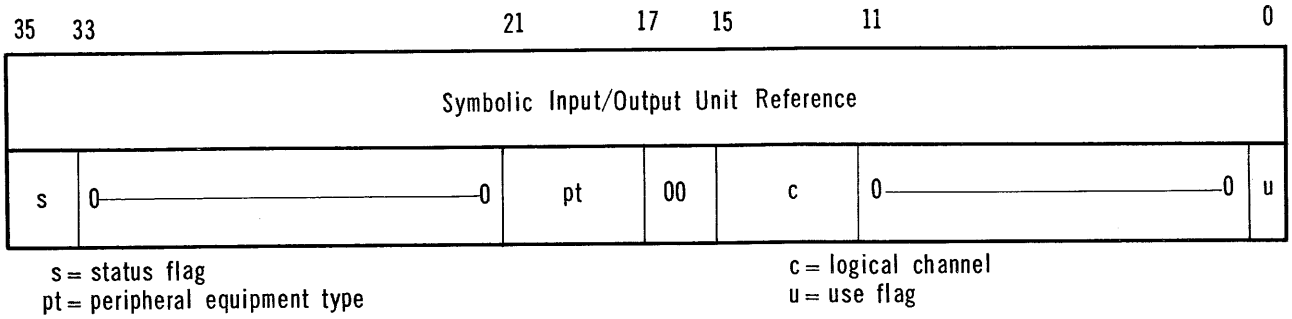


FIGURE 5: EXECUTIVE I/O REFERENCE ENTRY

3. DIRECT INPUT/OUTPUT REFERENCES: Two separate tables are used for input/output references in DIRECT ROC programs. All input/output references must have assignments at assembly time.

The first table contains channel references and their assignment. The second table contains unit references and their assignment. The status flag is contained in the second table only. No entries for magnetic drum reference(s) are contained in the second table.

a. DIRECT INPUT/OUTPUT CHANNEL REFERENCES: Entries in the first table (Figure 6), contain channel references. Each entry consists of two words. The first word is the symbolic channel reference. The second word contains the channel assignment. This channel assignment is in bits 25-22 (the a-field) and is repeated in bits 03-00.

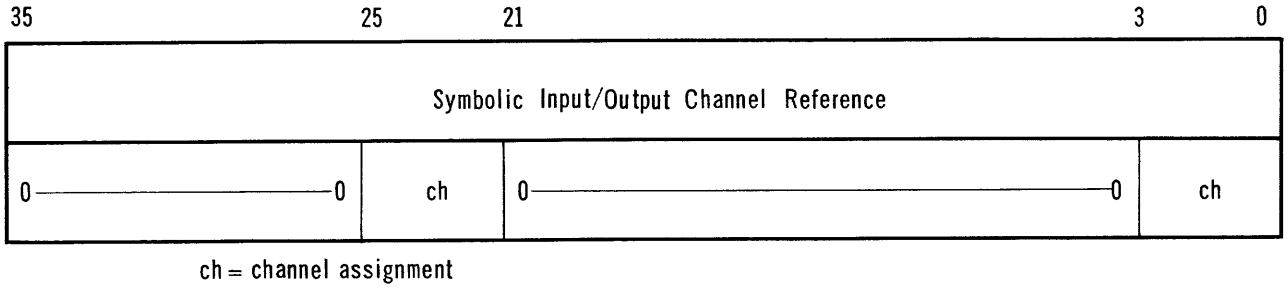


FIGURE 6: DIRECT I/O CHANNEL REFERENCE ENTRY

b. DIRECT INPUT/OUTPUT UNIT REFERENCES: The unit references, Figure 7, are in the second table. The first word in each two-word entry contains the symbolic unit reference. The second word contains the status flag in bits 35-34 and the unit assignment in bits 15-00.

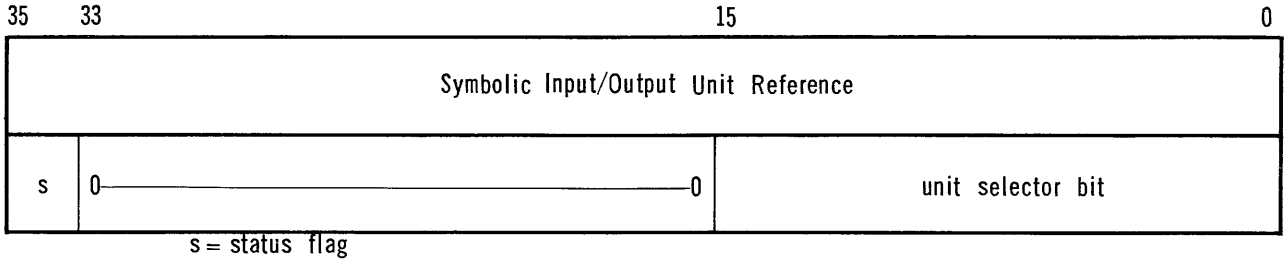


FIGURE 7: DIRECT I/O UNIT REFERENCE ENTRY

C. SYSTEM REFERENCES: The system reference section of the *MODRC may include up to four separate tables. These tables include:

Executive System References
Subroutine List
External References
Entrance List

All of these tables need not appear in each object program. However, those that do appear must be in the order shown.

Up to 128 unique references may be included in this section of the *MODRC.

1. EXECUTIVE SYSTEM REFERENCES: This table contains a list of references to the Executive System control, the Executive System I/O Functional Routines, and/or the Relative Load Routine. Each entry in this table will be one word, the symbolic reference.

2. SUBROUTINE LIST: This table contains a list of the names of all subroutines referenced by the object program, but not incorporated into the object program. Each entry in this table is one word, the Subroutine Name. These names are used to load the required subroutines from the Library at load time.

   This table may be included in the Program File for subroutines and complex programs only. Each entry in this table is assigned a reference number in the system reference section of the *MODRC.

3. EXTERNAL REFERENCES: This table contains a list of references to subroutines, other than their name, which are incorporated at load time. The references are entrances to the subroutines listed in paragraph C. 2. above.

   Each table entry is one word, the symbolic external reference. Each entry is assigned a reference number in the system reference section of the *MODRC. This table may be included in the Program File for subroutines and complex programs only.

4. ENTRANCE LIST: This table is contained only in the Program Files for subroutines. It contains a list of the symbolic entrances to the subroutine. Each table entry contains two words. The first word is the mnemonic symbol for the entrance. The second word contains the address of the entrance relative to the Subroutine Name. The entry is illustrated in Figure 8.

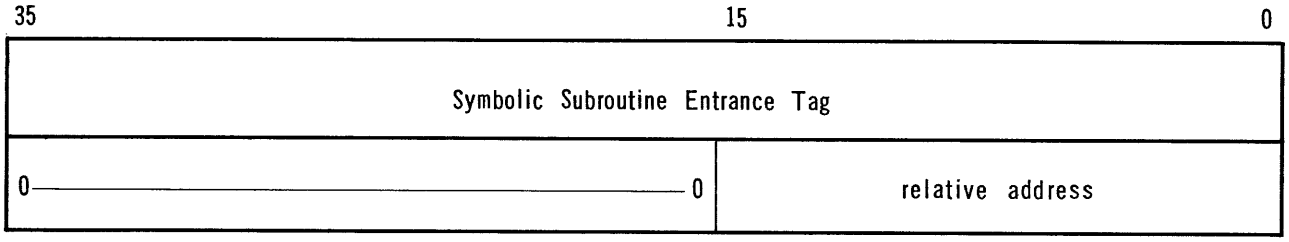


FIGURE 8: ENTRANCE LIST ENTRY

D. DRUM REFERENCES: The drum reference table contains a list of all DTAG and LTAG references to the magnetic drum(s) together with the assigned minimum length for each drum table. Each entry contains three words. The first word is the symbolic DTAG associated with the drum table. The symbolic LTAG is in the second word. Figure 9 illustrates the drum reference entry in the *MODRC.

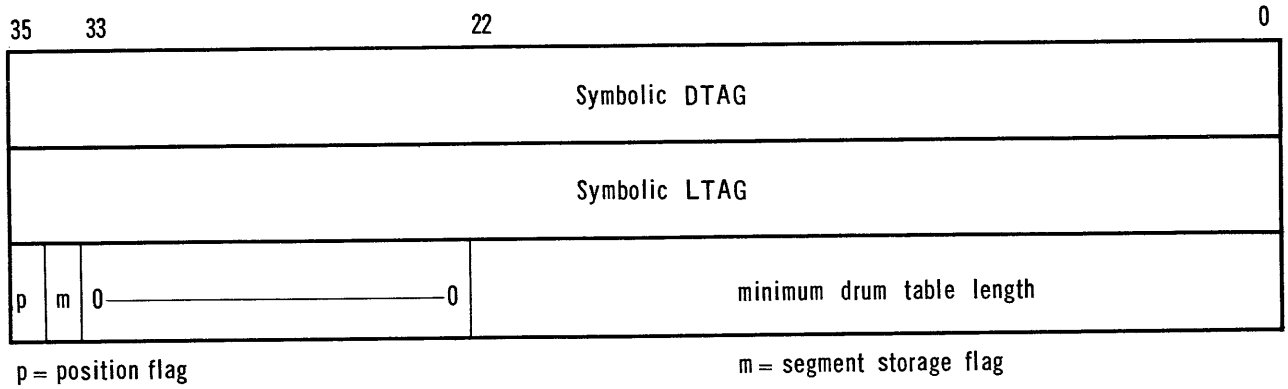


FIGURE 9: DRUM REFERENCE ENTRY

The third word contains the minimum drum table length in bits 22-00. The relationship of the subject drum table to the last preceding table is contained in bit 35. Bit 34 contains a flag to indicate whether or not the drum table is used for segment storage in segmented programs. Bit 34 contains a 1 if the drum table is used for segment storage. In all other cases this bit is 0.

The position flag, bit 35 of word 3, is Ø for independent tables, and has the value 1 if the subject table starts at the same location as the last preceding table.

If the length of any table had been assigned its absolute value, i.e., table length incrementation is not permitted at load time, the word containing the LTAG must have been cleared to zero.

Up to 128 drum tables may be contained in an object program. Reference numbers are arranged so that LTAG reference numbers are odd and DTAG reference numbers are even. Up to 256 references may be included in this section of the *MODRC.

E. CORE REFERENCES: The core reference table contains a list of all DTAG and LTAG references to the data section of core memory. Each entry in the core reference table contains three words. The first word is the symbolic DTAG. The symbolic LTAG is in the second word. The third word contains the minimum table length in bits 15-00. Bit 35 of the third word is the position flag. The position flag gives the position of the subject core table to the last preceding core table. The values of the position flag are assigned in the same manner as for drum tables (see Section IV.D.). The entry is illustrated in Figure 10.

The first entries in the core reference table are:

Fixed Length Data Table (DBANK)
$ERROR Table
$PARAM Table.

All of these tables need not be included with every object program; however those tables which are present must appear in the order shown above. The position flag for each of these tables is zero.

Both the LTAG and the DTAG for the DBANK are cleared to zero. For complex programs, the length assigned to this table includes the area needed by the main program and any included subroutines. It does not include the area(s) necessary for subroutines incorporated at load time.

The LTAG for the $ERROR table is cleared to zero. This table is assigned in the instruction area of the object program. The $PARAM table, when present, is assigned in the variable-length core data section.

All other tables listed in the core reference table are in the variable-length core data section. The absolute length for any of these tables may be assigned at assembly time. In this case the LTAG for the subject table is cleared to zero.

Both the symbolic LTAG and DTAG for a specific core table are replaced by the same reference number in the object program word. The field position of the reference number and/or an indicator determines if the current table length or address is to be used in the modification of the program word.

Up to 128 separate core tables may be defined in an object program.

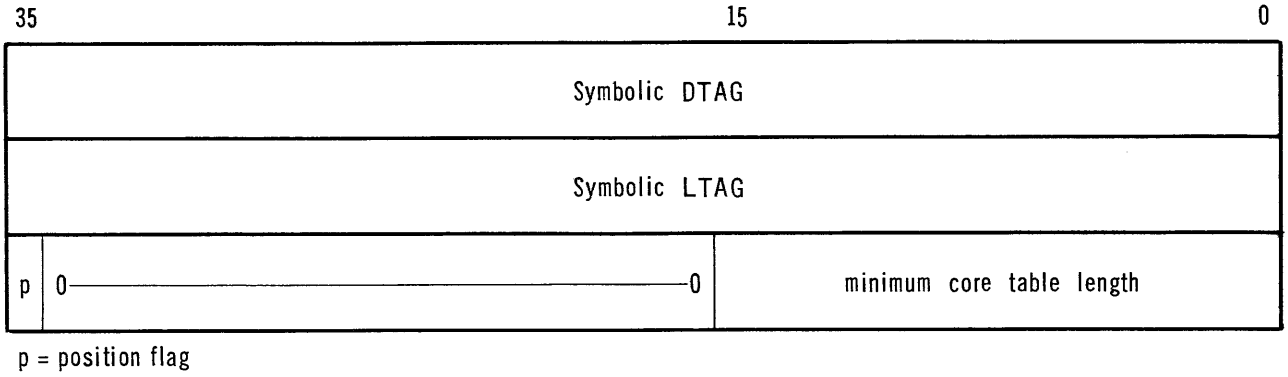


FIGURE 10: CORE REFERENCE ENTRY

V. PROGRAM RECORD

The Program Record is composed of one more blocks of 256 words each. A Program Record block is illustrated in Figure 11. Word 00 contains the identifier *PRORC, in Fieldata code. The identifier for the last block of the Program Record is *TRMRC.

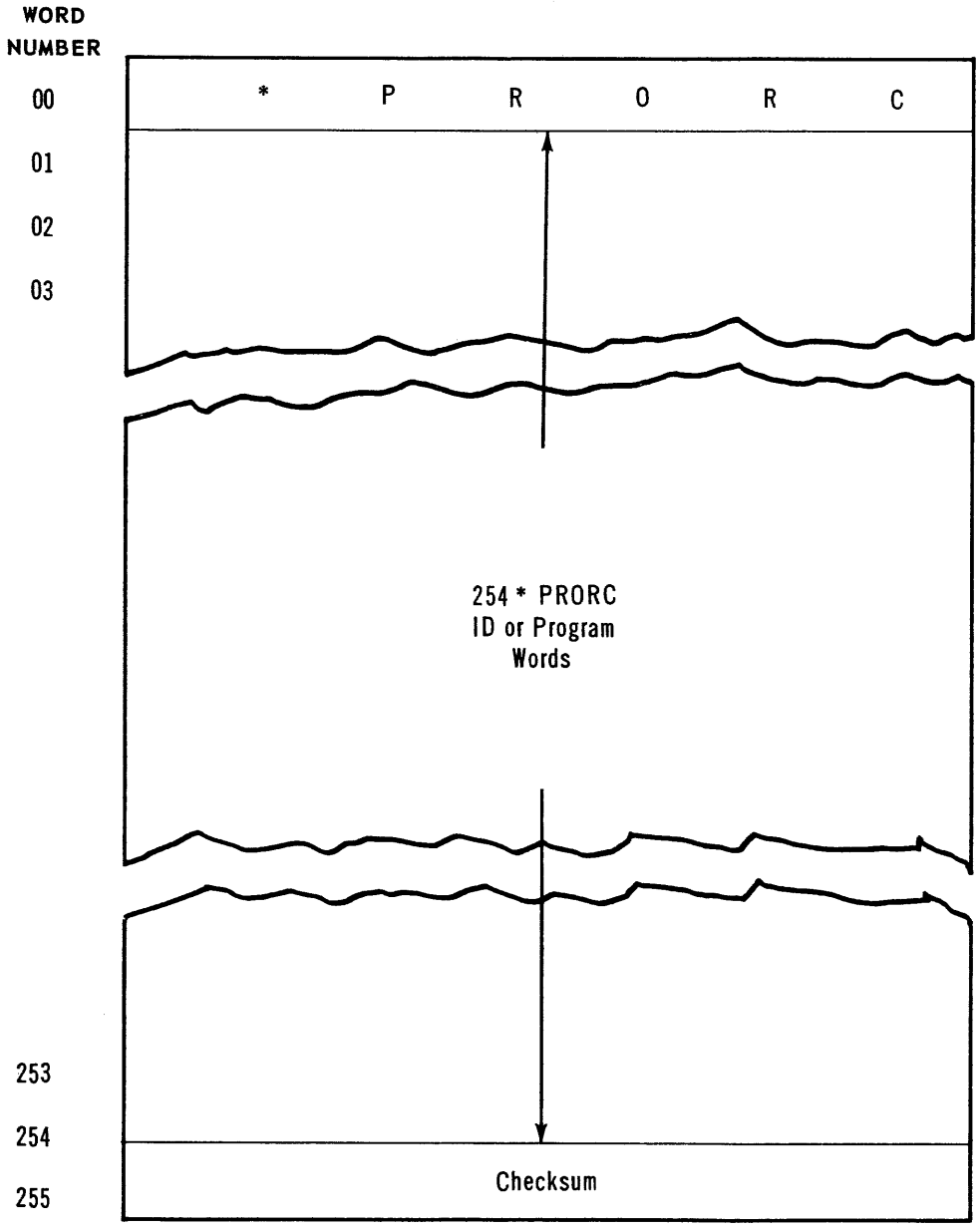


FIGURE 11: PROGRAM RECORD BLOCK

Word 255 contains the block checksum. Each block contains from 246 to 254 program words, indicator words, and/or identifier words. A balance of eight or less words in a *PRORC block is disregarded. Likewise any balance in the *TRMRC block (the last *PRORC block) is disregarded.

The Program Record contains all object program instructions and data table entries assigned at assembly time. These words are arranged in program sections. Program sections are one of two types; segment sections or table sections.

Each program section is headed by a section identifier. These words describe the program section which follows. Individual program sections are divided into program word groups. Each program word group is headed by a group identifier word.

A. PROGRAM SECTIONS: The segment sections contain the IBANK of an object program segment. This section also contains the DBANK associated with the program segment.

For segmented complex programs, the segment section may contain a subroutine reference portion. This portion appears in each segment section (with the exception of the main control routine) which has subroutine references. A subroutine list and an external reference list make up this portion of the segment section.

The subroutine list contains the Names of the subroutines to be incorporated into the program segment at load time. The references to these subroutines, in the program segment, are contained in the external reference list. The subroutine list and external reference list for the main control routine are contained in the program *MODRC.

The table sections contain the set of words to be loaded into the variable-length tables from the Program File.

B. PROGRAM SECTION IDENTIFIERS: The section identifier is used to identify each program section in the Program Record. The first entry in the first *PRORC block is a section identifier. A section identifier also heads each program segment and/or group of data table entries.

The section identifier for program segments is illustrated in Figure 12. This identifier is four words in length. Words 00 and 01 are bank descriptors. Word 00 refers to the IBANK. The DBANK is described in word 01. Bit 35 is the storage flag. If bit 35 is 1, the subject bank is recorded on a storage medium after modification. Bits 33-18 indicate the number of words in the IBANK or DBANK.

The relative address field, bits 15-00, indicates the address for the first word in the IBANK or the first word in DBANK.

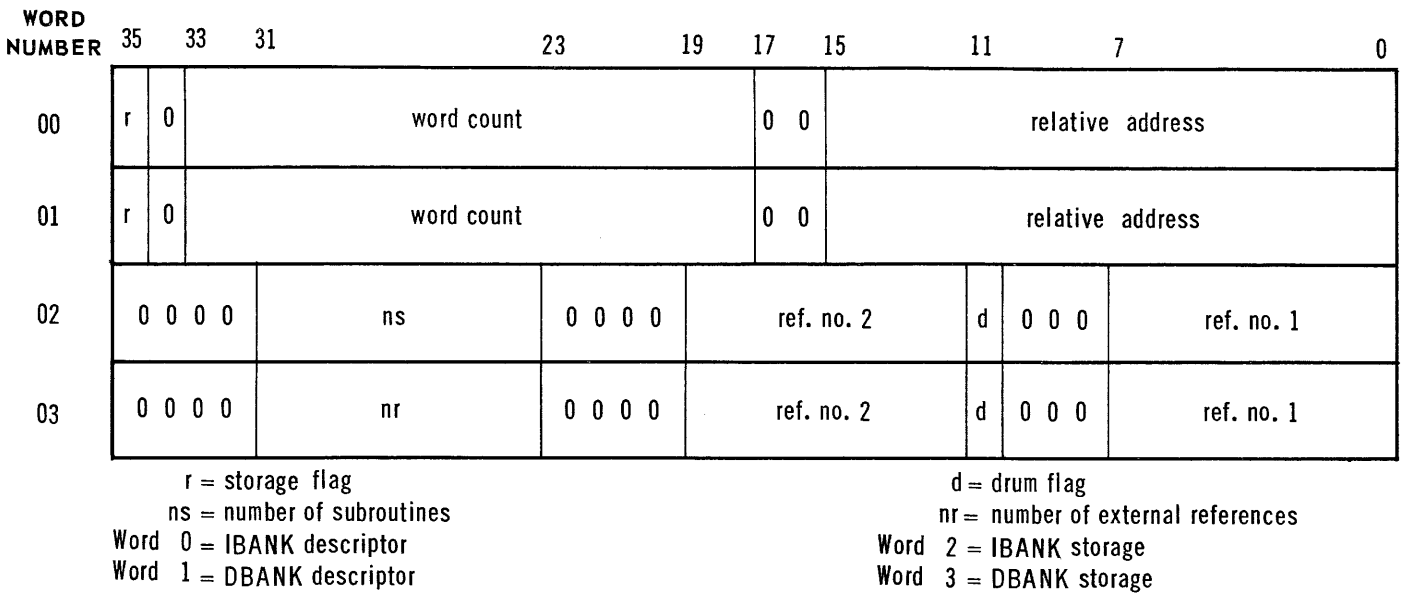


FIGURE 12: SEGMENT SECTION IDENTIFIER

For complex segmented programs, the number of entries in the subroutine list is indicated by bits 31-24 of word 02. Bits 31-24 of word 03 indicate the number of external references in the segment section. For simple programs, these fields are disregarded. They must be cleared to zero in complex segmented programs when the segment does not have load time included subroutines.

Bits 19-00 of words 02 and 03 are the segment storage descriptors. Word 02 is used for the IBANK and word 03 is used for the DBANK. Bits 19-12 contain the reference number for the storage channel for DIRECT ROC programs only. Bits 07-00 contain the unit or drum address reference number. Bit 11 is set to 1 when the storage medium is a drum. These fields are disregarded if the storage flag for the subject bank is not set in word 00 or 01.

The table section identifier is illustrated in Figure 13. This is a one word identifier. Bit 35 is 0 and bit 34 must always be 1. The word count, bits 33-18, indicates the number of data table words which are to be loaded. Bits 07-00 contain the reference number for the subject table in the core reference section of the *MODRC. Storage of variable-length data tables on an external medium is not provided.

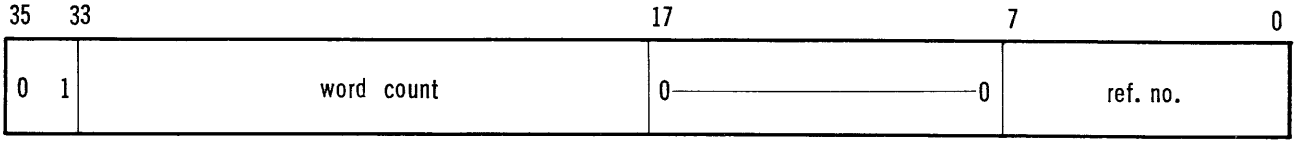


FIGURE 13: TABLE SECTION IDENTIFIER

C. GROUP IDENTIFIER WORDS: Each group of consecutive instruction words, consecutive data table words, and subroutine reference in a program section is headed by a group identifier word. A group identifier follows the section identifier for the first group in a section. The second word in each *PRORC block is either a group identifier or a section identifier word. The group identifier word is illustrated in Figure 14.

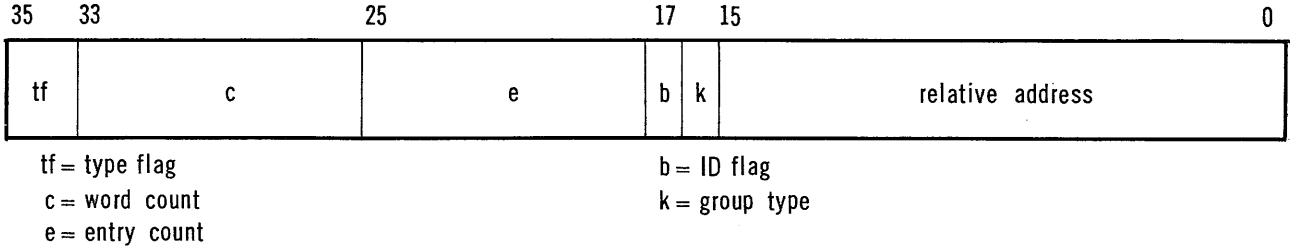


FIGURE 14: GROUP IDENTIFIER

The type flag, bits 35-34, is used to indicate the type of program word group that follows. This flag has the following values:

Type Flag	Word Group
1 0	instruction or data table words
1 1	subroutine references

1. INSTRUCTION AND DATA TABLE WORD GROUPS: These word groups are composed of program words and indicator words. Figure 15 illustrates the layout of an instruction or data table word group. The half words K thru T are either instruction half words or data table half words; only one kind is found in any given word group. There are two indicator fields

associated with each program word. The indicator fields are packed into the latter words of the word group. The indicator field for the upper half of the first program word (K in Figure 15) is located in the most significant bits of the last word in the word group (the k-field in Figure 15). This indicator field is followed by the indicator field for the lower half of the first program word, and so forth. If an indicator field does not fit in the least significant bit of a word, it is split between the present word and the preceding word in the word group; i.e., the field consisting of s1 and s2 is the indicator field for the half program word S in Figure 15 (s1 contains the most significant indicator bits). All bits of the word within the instruction and data table word groups are used for program words or indicator fields, except for the possibility of the least significant bits of the indicator word following the last program word.

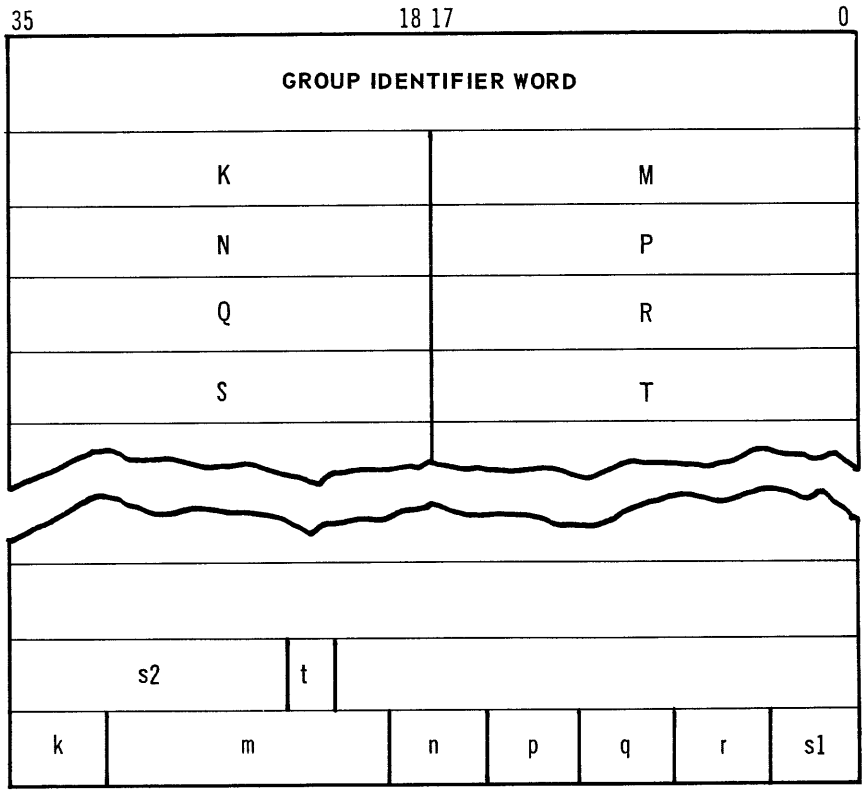


FIGURE 15: INSTRUCTION AND DATA TABLE WORD GROUP

All IBANK and DBANK word groups for a specific program segment must appear in one program section. The IBANK and DBANK word groups need not be segregated within the section, however.

The identifiers for these groups use the following fields in addition to the type flag:

Field	Bit Positions	Use
Word count	33-26	Number of words in group
Entry count	25-18	Number of program words in group
ID flag	17	Denotes last group in program section
Group type	16	Denotes if group contains instructions or data words.

The ID flag, bit 17, is set to 1 only when the word group identifier refers to the last group in a program section. A section identifier or the *TRMRC notation follows the subject word group. Bit 16 is set to 1 for word groups which contain data table words (either DBANK or variable-length table entries). It is set to Ø for word groups containing instruction words. The relative address field locates the program words in the word group in the IBANK, DBANK, or DTABLE.

2. SUBROUTINE REFERENCE WORD GROUPS: These word groups may appear only in segment sections of complex segmented programs. These word groups are used to specify subroutines incorporated at load time, and form an extension to the system reference section of the *MODRC. Two types of word groups appear in this portion of the segment section. The subroutine list group precedes the external reference group. These word groups follow the segment section identifier and precede the instruction and data table word groups in the segment sections in which they appear.

The identifiers for these groups use the following fields, in addition to the type flag:

Field	Bit Positions	Use
Entry count	25-18	Number of words in group
Group type	16	Denotes group contents
Relative address	15-00	Reference number

Bit 16, the group type, is Ø for the subroutine list and 1 for the external reference list. The relative address field contains the reference number of the first group word in the system reference modification table. The remainder of the fields in the identifier word are disregarded.

The subroutine list contains the names of all subroutines to be incorporated into the program segment at load time. Each name is contained in one word.

The external reference list contains the symbolic references to the subroutines in the program segment. Each reference is contained in one word.

D. PROGRAM AND INDICATOR WORDS: The major portion of the Program Record is composed of program and indicator words. These words appear in instruction and data table word groups. Each program word has two indicator fields associated with it.

1. INDICATOR FIELDS: Each indicator field denotes the modification for one half of a program word. The indicator for the left half precedes the indicator for the right half. An indicator field contains 1, 4 or 13 bits. Each indicator field can contain three subfields: a modification flag, a modification type, and a reference number. The indicator values are tabulated in Figure 16.

   The modification flag indicates if the half program word is to be modified. The modification flag is Ø for no modification and 1 when modification is necessary. When the modification flag is Ø, the indicator contains only one bit. If the flag is 1, the next 3 or 12 bits denote the type of modification.

   The modification type subfield indicates the modification that is to be performed. Values in the modification type subfield take on different meanings based on the object program format. These meanings are determined by the type of ROC used in the object program. The number of bits modified in the half program word is affected by these different meanings.

INDICATOR		LEFT HALF WORD MODIFICATION	RIGHT HALF WORD MODIFICATION
MODIFICATION FLAG	MODIFICATION TYPE		
0		No Modification	No Modification
1	0 0 0	Add current Program Name address (16-bit field)	Add current Program Name address (16-bit field)
1	0 0 1	Core DTAG ± LTAG (16-bit field)	Core DTAG ± LTAG (16-bit field)
1	0 1 0	Core TAG, Core TAG ± constant (16-bit field)	Core TAG, Core TAG ± constant (16-bit field)
1	0 1 1	System symbol (16-bit field)	System symbol (16-bit field)
1	1 0 0	I/O { Selective Jump Switch (4-bit field) / Channel Reference (4-bit field)* }	Not Available
1	1 0 1	Not Available	Drum DTAG ± LTAG ***
1	1 1 0	Drum LTAG ± constant (16-bit field)	Drum TAG, Drum TAG ± constant ****
1	1 1 1	I/O Unit Reference (16-bit field)*	I/O Unit Reference **

* *Available only for DIRECT ROC object programs*

** *16-bit field for DIRECT ROC object programs.*

For EXEC ROC object programs, modifiable field is 30 bits.

*** *For DIRECT ROC object programs, 23-bit field.*

For EXEC ROC object programs, 30-bit field.

**** *For LTAG, 23-bit field*

For DTAG, DIRECT ROC object program, 23-bit field

For DTAG, EXEC ROC object program, 30-bit field

FIGURE 16: . MODIFICATION INDICATOR FIELDS

The reference number subfield is used when the modification is a special case of the "TAG ± constant" form. In this case, the constant has a value equal to or greater than 2^6 for core references, or the constant has a value equal to or greater than 2^{12} for magnetic drum references.

a. INDICATORS FOR CORE REFERENCES: The indicators for IBANK references, core data table references, and system symbol references refer to a 16-bit field in the half program word. Figure 17 illustrates the indicators and their associated program word fields. This type of indicator may be used to denote modification in either the left or right half program word.

The modification type subfield for core references has one of the following values:

Subfield	Definition
000	Internal Program Reference Add current address of Program Name to value in program word field and place sum in program word field.
001	Core DTAG ± LTAG Add values referenced by core table reference numbers in program word field and place sum in program word field.
010	Core TAG, Core TAG ± constant Add value referenced by core table reference number to value in program word field and place sum in program word field. The size of the constant determines location of reference number.
011	System Symbol References Replace the system reference number in program word field by current address of system symbol.

b. INDICATORS FOR DRUM LTAG REFERENCES: The indicators, Figure 18, for drum table length modification refer to either a 16-bit field or a 23-bit field in the program word. The size of the program word field depends on the part of the program word (either left half or right half) which is modified. The value of the modification type subfield is 110. The reference number for the LTAG is always odd.

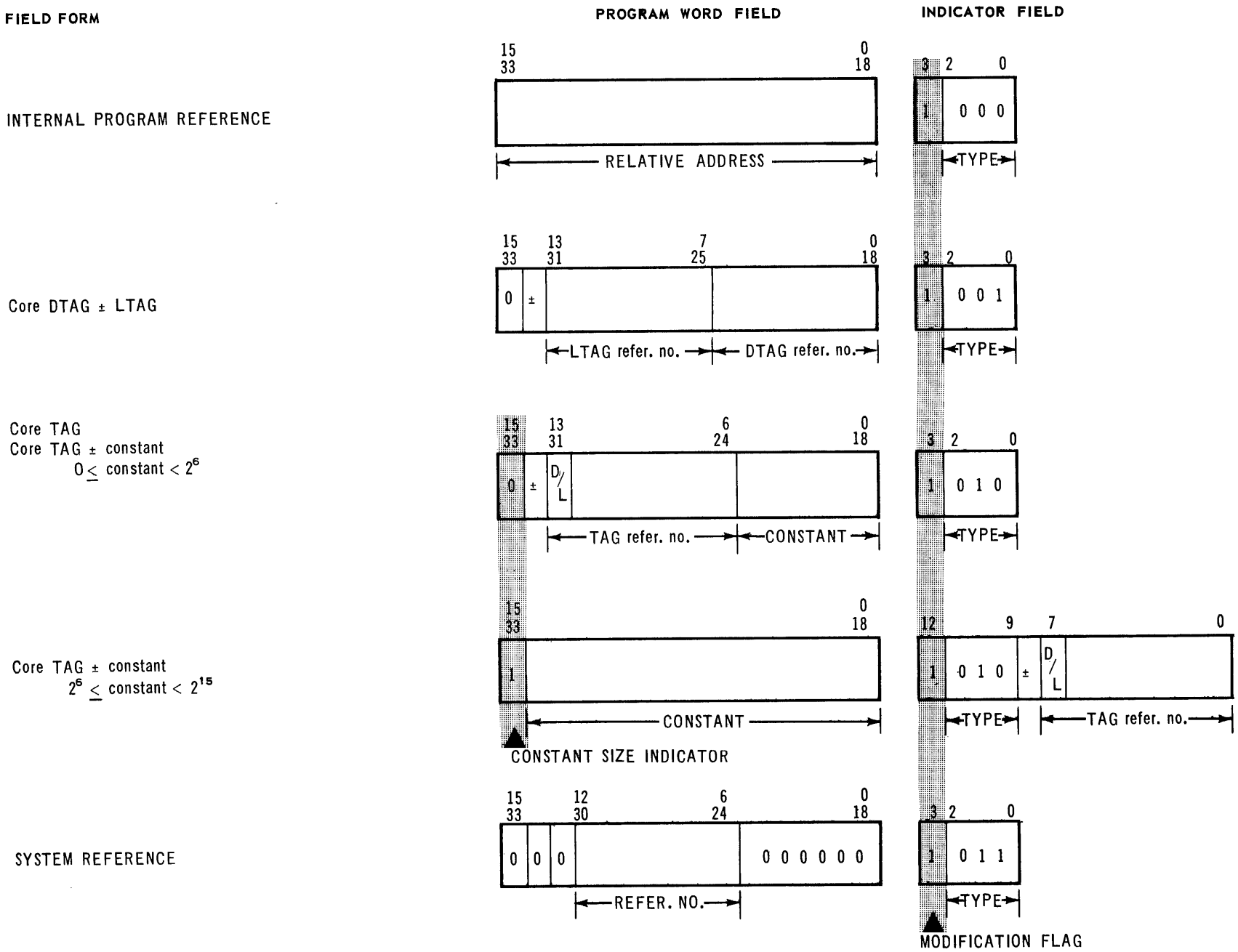


FIGURE 17: IBANK, CORE, AND SYSTEM REFERENCES

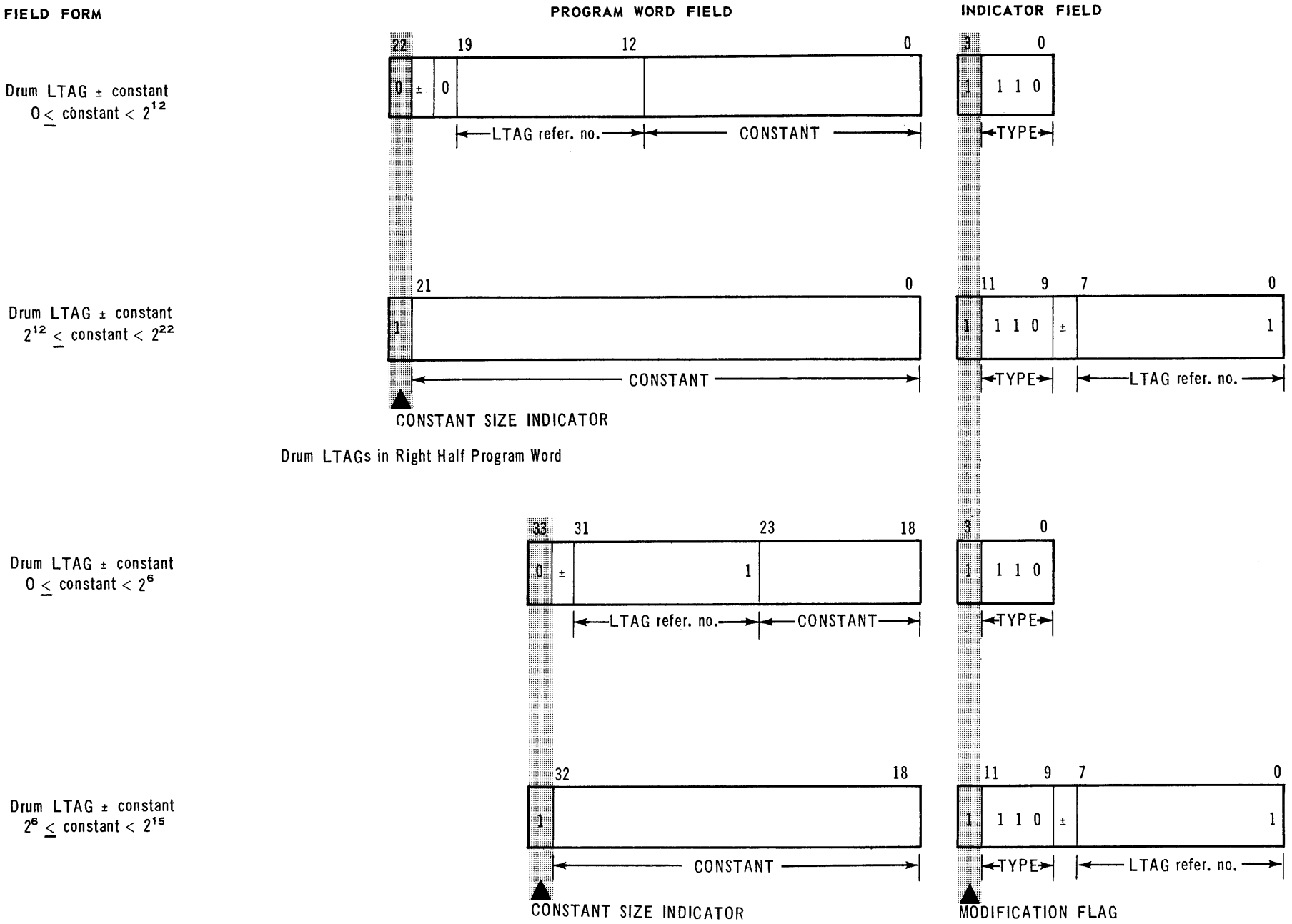


FIGURE 18: DRUM LTAG REFERENCES

The current value for the program word field is the sum or difference of the value associated with the drum reference number and the value contained in the program word field. The drum reference number is either in the program word field or in the indicator field depending on the value of the constant.

When a drum LTAG reference indicator is associated with a left half program word, a 16-bit field is modified. This 16-bit field occupies bits 33-18. The 16 least significant bits of the value generated are placed in the program word field. The conventions for this reference are the same as those for a core LTAG.

If a drum LTAG reference indicator is associated with a right half program word, a 23-bit field is modified. This 23-bit field is in bits 22-00. When this modification is used, the left half of the program word cannot be modified.

c. INDICATORS FOR EXECUTIVE SYSTEM INPUT/OUTPUT REFERENCES: The indicators for Executive System input/output references refer to a 30-bit field. (see Figure 19). This type of indicator is associated with right half word modification only The field modified occupies bits 29-00. If this modification is used, no modification can be made in the left half program word.

The modification type subfield has one of the following values:

Subfield	Definition
101	Drum DTAG ± LTAG Add values referenced by drum reference numbers in program word field and place sum in program word field.
110	Drum DTAG, Drum DTAG ± constant Add value referenced by drum reference number to value in program word field and place sum in program word field. The size of constant determines reference number location.
111	Equipment Reference Replace equipment reference number by current channel and unit assignment.

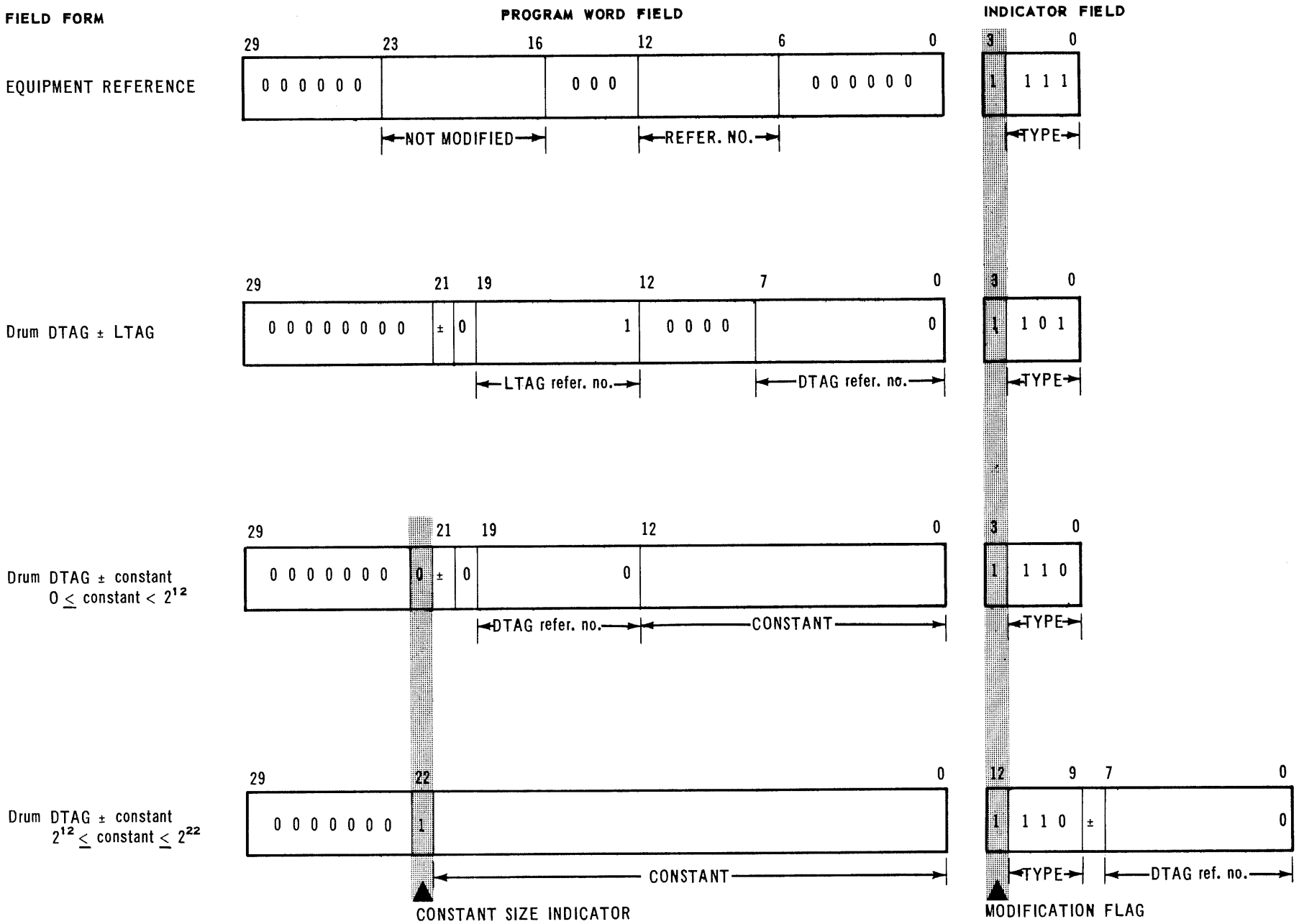


FIGURE 19: EXEC ROC I/O REFERENCES

d. INDICATORS FOR SELECTIVE JUMP SWITCH REFERENCES: The indicator for Selective Jump Switch references refers to a four-bit field. See Figure 20. The four-bit field is in bits 25-22, the a-field. The indicator is used with left half program words only. The modification type subfield value is 100. The jump switch reference number in the program word is replaced by its current assignment.

e. INDICATORS FOR DIRECT INPUT/OUTPUT REFERENCES: The indicators for Direct Input/Output references refer to 4, 16, or 23-bit fields. See Figure 20. These indicators are used for channel references, unit references, and drum references.

The channel reference indicators refer to the four bit a-field, bits 25-22. This indicator is used with left half program words only. The modification type subfield value is 100. The channel reference number in the program word is replaced by its current assignment. This modification is identical to the Selective Jump Switch modification.

The indicators for unit references refer to a 16-bit field. This indicator may be associated with either a left half program word or a right half program word. The modification is made in the low order 16-bits of the half program word. The value of the modification type subfield is 111.

The unit reference indicator may be used for input/output access-control word references. In this case the reference number refers to a channel rather than a unit. The sum of the value associated with the input/output reference number and the value in the 6 least significant bits of the half program word is placed in the 16 least significant bits of the half program word.

The indicators for drum references refer to a 23-bit field. This 23-bit field is in bits 22-00. This indicator is associated with right half program words only. When the right half program word has this modification, no modification may be made in the left half program word.

The modification type subfield values are:

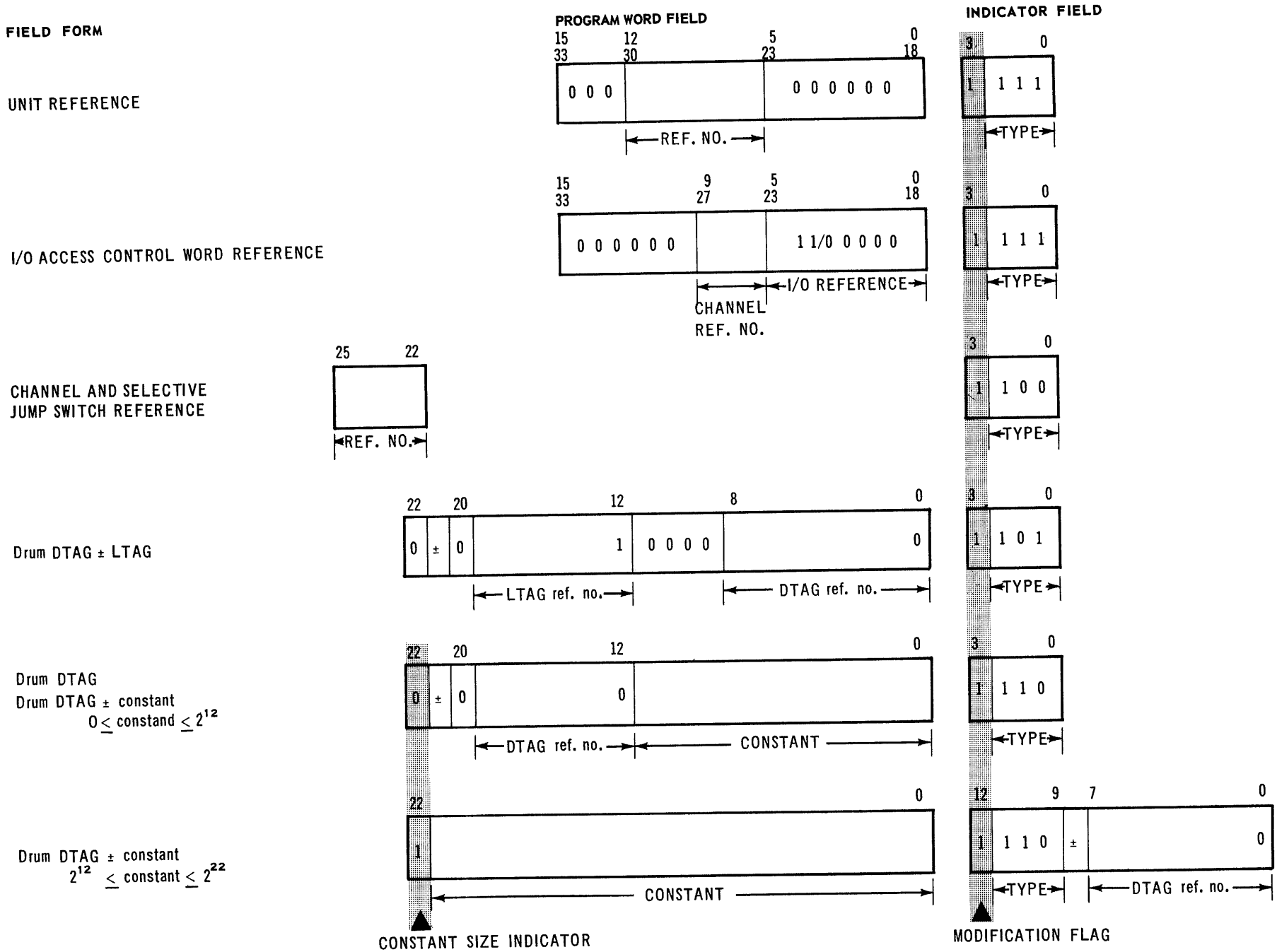


FIGURE 20: DIRECT ROC I/O AND SELECTIVE JUMP REFERENCES

Subfield	Definition
101	Drum DTAG ± LTAG Add values referenced by drum table reference numbers in program word field and place sum in program word field.
110	Drum DTAG, Drum DTAG ± constant Add value referenced by drum table reference number to value in program word field and place sum in program word field The size of the constant determines location of reference number.

2. PROGRAM WORD FIELDS: The program word fields which are subject to modification contain reference numbers and/or constants.

   a. FIELDS FOR IBANK REFERENCES: The modifiable fields for IBANK references are in bits 33-18 or in bits 15-00. These fields contain an address which is relative to the Program Name. The current address of the Program Name is added to this value. The sum is placed in the program word field. See Figure 17.

   b. FIELD FOR CORE TABLE REFERENCES: The modifiable fields for core table references of the form "DTAG ± LTAG" are in bits 33-18 or in bits 15-00. The sign is in bit 32 or bit 14. The LTAG reference number is in bits 31-25 or bits 13-07. The DTAG reference number is in bits 24-18 or bits 06-00. See Figure 17.

      The modifiable fields for core table references of the form "TAG" or "TAG ± constant" are either in bits 33-18 or bits 15-00. The most significant bit in the modifiable field indicates the location of the TAG reference number. This bit is 1 if the reference number is in the indicator. If this bit is Ø, the TAG reference number is contained in the modifiable field. The sign is in bit 14 or bit 32. If the TAG reference number is in the indicator, the sign is in bit 03. The TAG reference number is in bits 31-24 or bits 13-06 of the program word. When the indicator contains the TAG reference number, it is in bits 07-00. The most significant bit of the reference number indicates the type of TAG. This bit is Ø for DTAGs and 1 for LTAGs. See Figure 17.

c. FIELD FOR SYSTEM REFERENCES: The modifiable field for system references is in bits 33-18 or in bits 15-00. The system reference number is either in bits 30-24 or in bits 12-06. The current address associated with the reference number is placed into the 16-bit modifiable field. See Figure 17.

d. FIELD FOR DRUM REFERENCES: The modifiable field for drum references is in 16, 23, or 30 bit fields. The reference number for drum TAGs occupies 8 bits. The least significant bit indicates the type of TAG. If this bit is Ø, a DTAG is referenced. This bit is 1 for LTAG references.

The modifiable field for drum LTAG references is in 16 or 23 bit fields. If the modifiable field is in a left half program word, a 16-bit modifiable field, bits 33-18, is referenced. For a modifiable field in a right half program word, a 23-bit field, bits 22-00, is referenced. See Figure 18.

The modification for the 16-bit drum LTAG reference field is handled in the same manner as core table references. The modification for the 23-bit field is handled in the same manner as DIRECT ROC drum DTAG references.

The modifiable fields for drum DTAG references is 23 bits or 30 bits. The modifiable field is always in the right half program word.

The 30-bit field, bits 29-00, applies to EXEC ROC programs. Bits 29-25 contain the channel assignment. See Figure 19.

The 23-bit field, bits 22-00, applies to DIRECT ROC programs. See Figure 20.

In the "DTAG" or "DTAG ± constant" forms, bit 22 indicates the location of the reference number. If this bit is Ø, bits 19-12 of the program word contain the drum reference number. If bit 22 is 1, the drum reference number is in bits 07-00 of the indicator. The sign is in bit 21 of the program word or bit 08 of the indicator. The current value for the modifiable field is the sum or difference of the value associated with the reference number and the value in the constant part of the modifiable field.

For the "DTAG ± LTAG" form, bit 21 contains the sign, 19-12 contain the LTAG reference number, and the DTAG reference number is in bits 07-00. The current value of this field is the sum or difference of the values associated with the DTAG and LTAG reference numbers.

e. FIELD FOR INPUT/OUTPUT REFERENCES: The modifiable fields for input/output references contain 4, 16, or 30 bits. The size of the field is dependent on the type of input/output control and the kind of reference.

The modifiable field for Selective Jump Switches is four bits, bits 25-22. This field contains the reference number for the current switch assignment which replaces it. This type of modifiable field is associated only with left half program words. See Figure 20.

The modifiable field for EXEC ROC input/output references contains 30 bits. The reference number is in bits 12-06. The current channel assignment is placed in bits 29-25 and the current unit assignment is placed in bits 15-00. Bits 23-16 are not modified. All other bits in the 30-bit field are zero. This type of modifiable field is only associated with a right half program word. See Figure 19.

The modifiable fields for DIRECT ROC input/output references contain 4 or 16 bits. The modifiable field for channel references is four bits, bits 25-22. This field contains the reference number for the current absolute channel assignment which replaces it. See Figure 20.

The modifiable fields for unit references are in bits 33-18 or in bits 15-00. The reference number is in bits 30-24 or bits 12-06. This reference number may refer to a unit reference, a channel reference, or input/output access-control word reference. In the case of input/output access control words, bits 05-00 or bits 23-18 contain either 40 (octal) for input or 60 (octal) for output. See Figure 20.

The modifiable field is replaced by the sum of the value in bits 05-00 or bits 23-18 and the value associated with the input/output reference number.

VI. TERMINATION RECORD

The Termination Record is the last *PRORC block. It is identical to a *PRORC block except that the first word in the block contains *TRMRC, in Fieldata code. The block contains program section words in the same manner as *PRORC blocks.

The last program section is followed by a two-word entry. This entry is illustrated in Figure 21. The first word is the identifier *TRMRC, in Fieldata code. The second word contains the address, relative to the Program Name of the first instruction to be executed. This field is in bits 15-00.

Any balance in this block is disregarded. The last word in the block is the checksum.

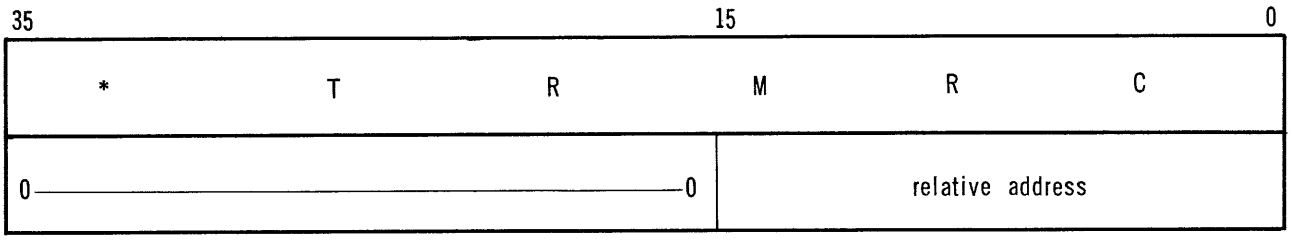


FIGURE 21: TERMINATION ENTRY

VII. PROGRAM FILE FOR ABSOLUTE OBJECT CODE PROGRAMS

The Program File for Absolute Object Code (AOC) programs is slightly different from the Program File for ROC programs. The Program File consists of three records, in the following order:

1) Identification Record

2) Program Record

3) Termination Record

The Identification Record consists of one block. The Program Record contains as many blocks as are necessary to include all program instructions and data tables.

The last word in each block of the Program File, with the exception of the Identification Record block, contains the block checksum. Each block in the Program File contains 256 words.

The first word in each block is the identifier word. The identifier for the Identification Record is the first six characters of the alphanumeric Program Name. The identifiers for the other records are:

Record Type	Record Label
Program	*PRORC
Termination	*TRMRC

All unused words in the Program File blocks are disregarded. No special setting is required in these words.

A. IDENTIFICATION RECORD: The first record in a Program File is the Identification Record. This record is composed of one 256-word Label Block. The Label Block is illustrated in Figure 22.

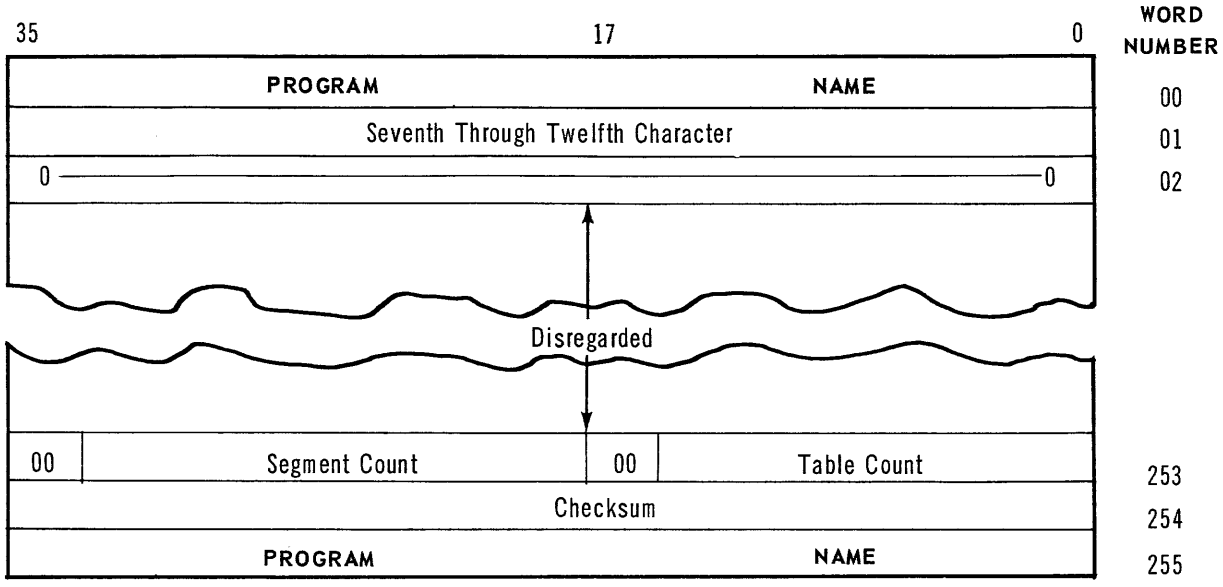


FIGURE 22: LABEL BLOCK, AOC

Words 00 and 01 contain the Program Name. The conventions for the Program Name are discussed in Section III. The first six characters of the Program Name are repeated in word 255. Word 254 contains the block checksum. Word 02 is cleared to zero.

Word 253 describes the *PRORC of the Program File. Bits 33-18 indicate the number of program segments. Bits 15-00 indicate the number of data tables which are to be loaded from the *PRORC. The balance of the block is disregarded.

B. PROGRAM RECORD: The Program Record is composed of one or more blocks of 256 words each. A Program Record block is illustrated in Figure 23. Word 00 contains the identifier *PRORC, in Fieldata code. The identifier for the last block of the Program Record is *TRMRC.

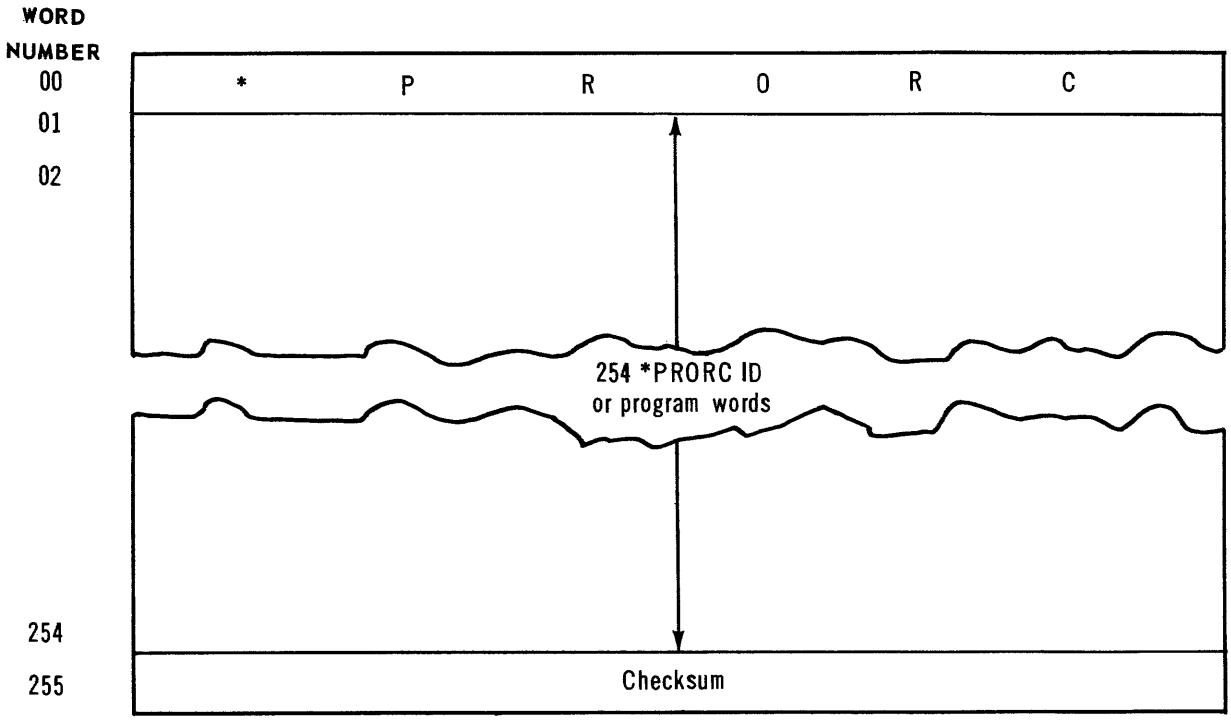


FIGURE 23: PROGRAM RECORD BLOCK, AOC

Word 255 contains the block checksum. Each block contains from 246 to 254 program words and/or identifier words. A balance of eight or less words in a *PRORC block are disregarded. Likewise, any balance in the *TRMRC block (the last *PRORC block) is disregarded.

The Program Record contains all programs instructions and data table entries assigned at assembly time. These words are arranged in program sections. Program sections are one of two types: segment sections or table sections.

Each program section is headed by a section identifier. These words describe the program section which follows. Individual program sections are divided into program word groups. Each program word group is headed by a group identifier word.

1. PROGRAM SECTIONS: Segment sections contain the segment IBANK and its associated DBANK. Table sections contain the DTABLE entries from the Program File.

2. PROGRAM SECTION IDENTIFIER: The section identifier is used to identify each program section in the Program Record. A section identifier heads each program segment and/or set of data table entries. The first entry in the first *PRORC block is a section identifier.

   The section identifier for program segments is illustrated in Figure 24. This identifier is four words in length. Words 00 and 01 are bank descriptors. Word 00 refers to the IBANK and word 01 refers to the DBANK. Bit 35 is the storage flag. This bit is 1 if the bank is recorded on a storage medium after loading. Bits 33-18 indicate the number of words in the bank. The address field, bits 15-00, give the address of the first word in the bank.

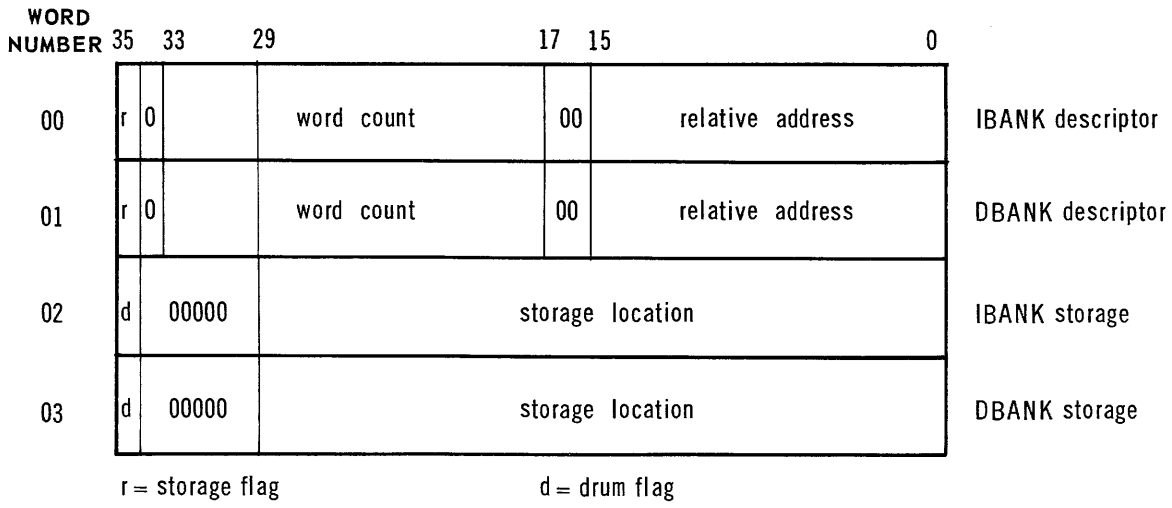


FIGURE 24: SEGMENT SECTION IDENTIFIER, AOC

Words 02 and 03 are segment storage descriptors. The IBANK storage is in word 02 and DBANK storage is in word 03. Bit 35 is set to 1 if the storage is on drum. The storage location for the bank is in bits 29-00. The storage descriptor is in EXEC ROC format.

The table section identifier is illustrated in Figure 25. It is one word long. Bit 35 is 0 and bit 34 is 1. The word count, bits 33-18, indicate the number of table words to be loaded. The remainder of the word is disregarded.

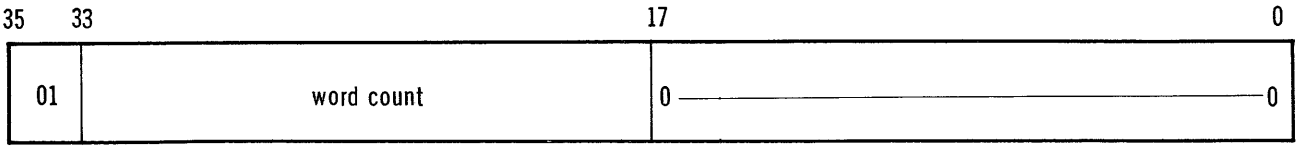


FIGURE 25: TABLE SECTION IDENTIFIER, AOC

3. GROUP IDENTIFIER WORDS: Each group of consecutive instruction words and/or consecutive table words, in a program section, is headed by a group identifier word. A group identifier follows each section identifier. The second word in each *PRORC block is either a group identifier or a section identifier word. The group identifier word is illustrated in Figure 26.

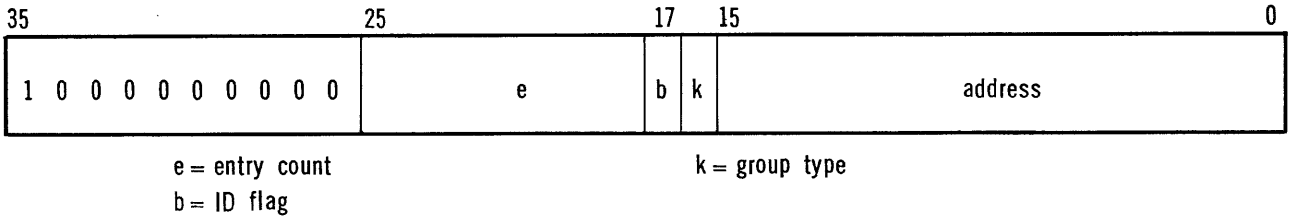


FIGURE 26: GROUP IDENTIFIER, AOC

The entry count field, bits 25-18, indicates the number of words in the word group. The ID flag, bit 17, is 1 when the word group following is the last in the program section. Bit 16 is 0 for instruction word groups and 1 for table word groups. The address field, bits 15-00, contains the address for the first word in the word group.

All IBANK and DBANK word groups for a single program segment must appear in one program section. The IBANK and DBANK word groups need not be segregated within the section, however.

4. PROGRAM WORDS: The balance of the *PRORC blocks contain either instruction words or data table words. Modification indicators are not used in Absolute Object Code.

C. TERMINATION RECORD: The Termination Record is the last *PRORC block. It is identical to a *PRORC block except that the first word contains *TRMRC, in Fieldata code. The block contains program sections in the same manner as *PRORC blocks.

The last program section is followed by a two-word entry. This entry is illustrated in Figure 27. The first word contains *TRMRC in Fieldata code. The second word contains the execution address in bits 15-00.

Any balance in this block is disregarded. The last word in the block is the checksum.

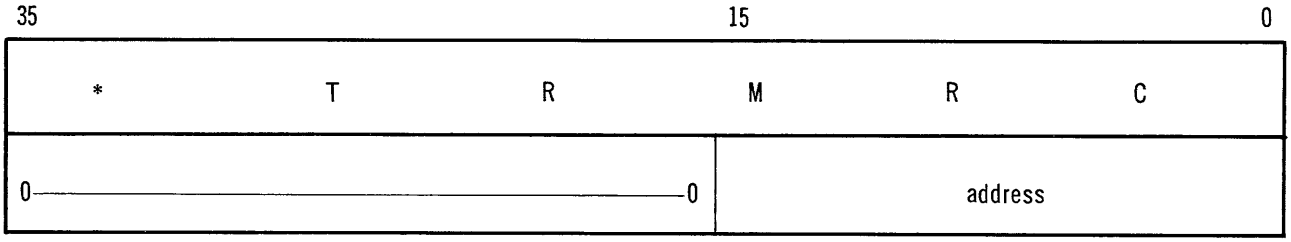


FIGURE 27: TERMINATION ENTRY, AOC

INDEX

L

M

O

P

R

S

INDEX

T

UNIVAC

DIVISION OF SPERRY RAND CORPORATION

PRINTED IN U.S.A. UP 2575