

MARC II AND COBOL

Henriette D. AVRAM and Julius R. DROZ:
Information Systems Office, Library of Congress, Washington, D. C.

A description of the machine processing of MARC II records using COBOL for an application on the Library of Congress System 360/30. Emphasis is on the manipulation by COBOL of highly complex variable length MARC records containing variable length fields.

Since the implementation of the MARC II format by the Library of Congress for the MARC Distribution Service, some potential users of machine readable data have expressed doubts that MARC formatted records could be effectively manipulated by programs in the COBOL language. Griffin (1) expressed his concern by stating, "Users will require programmers skilled in languages other than FORTRAN or COBOL to take advantage of MARC records." During the design phases of the MARC II format, the Information Systems Office staff concluded that the capability of COBOL should be accommodated. The relationship between the format and COBOL could not be tested until a data base was established. The data base is now an accomplished fact. The purpose of this article is to report that COBOL can be and is being used with MARC II data.

APPLICATION

The Science and Technology Division of the Library of Congress had a requirement from the U.S. Army Terrestrial Science Center to produce three reports (by subject heading, by author, and by corporate author) for ultimate photo-reduction and reproduction of the periodical literature in the Library's collection dealing with cold region research. The bibliographic data was processed through the MARC system and the resultant tape was in the Library of Congress' MARC processing format. It is at this point in the MARC Distribution Service that the MARC processing format is converted to the MARC II communications format for distribution to subscribers. Rather than create a completely simulated environ-

ment within which to test the use of COBOL, it was decided to integrate the analysis of the COBOL language with the task at hand, i.e., the programming effort required to produce the necessary reports from a magnetic tape file of MARC records.

Additional criteria for this investigation were established. Stress was placed on the development of COBOL programming techniques utilizing the smallest possible amount of computer core storage, thereby establishing the capability for potential users with a minimum hardware configuration. Furthermore, since COBOL compilers vary in the language power that they provide with the size and make of equipment, the subset of COBOL language used conformed with the basic level of COBOL that is being standardized for acceptance in the ADP world by the United States of America Standards Institute (USASI) Subcommittee X3.4 (Common Programming Languages).

MARC II COMMUNICATIONS FORMAT AND MARC PROCESSING FORMAT COMPARED

Before a description of what was done and how, a comparison of the MARC II communications format and the MARC processing format, with a brief statement of their differences, is in order.

The MARC II communications format (2) is schematically represented in Figure 1.

Leader	Record Directory	Control Fields	Variable Fields
--------	------------------	----------------	-----------------

Fig. 1. MARC II Communications Format.

The communications format may be recorded on either seven-level or nine-level tape and the term "byte" in the following discussion refers to either a six-bit or eight-bit character.

The MARC processing format is schematically represented in Figure 2.

Leader	Communications Field	Record Control Field	Fixed Field	Record Directory	Variable Fields
--------	----------------------	----------------------	-------------	------------------	-----------------

Fig. 2. Library of Congress MARC Processing Format.

MARC records at the Library of Congress are contained on magnetic tape in the form of undefined records. For System 360 purposes, these are unblocked, variable length records without the two four-byte block length and record length fields. In the processing format, fields may be recorded in binary, or packed decimal, hexadecimal or EBCDIC charac-

ters dependent on the characteristics of the field and the machine processing and storage efficiency required. Therefore, in the processing format the term "byte" does not necessarily refer to a character but rather describes a unit of eight bits. A brief description of the fields of both formats and a gross comparison of their differences is shown in Table 1.

Table 1. Comparison of MARC II Communications Format and MARC Processing Format

Communications Format

Leader

The leader is fixed in length for all records and contains 24 bytes (characters).

Logical Record Length	5 bytes
Record Status	1 byte
Type of Record	1 byte
Bibliographic Level	1 byte
Blanks	2 bytes
Indicator Count	1 byte
Subfield Code Count	1 byte
Base Address of Data	5 bytes
Blanks	7 bytes

Processing Format

Leader

The leader is fixed in length for all records and contains 12 bytes.

Logical Record Length	2 bytes
Date and Status	4 bytes
Blank	1 byte
Type of Record	1 byte
Bibliographic Level	1 byte
Blanks	3 bytes

Communications Field

The communications field is fixed in length for all records and contains 12 bytes.

Record Directory

Location	2 bytes
Directory Entry Count	2 bytes
Record Source	1 byte
Record Destination	1 byte
In Process Type	1 byte
In Process Status	1 byte
Blanks	4 bytes

Record Control Field

The record control field is fixed in length and contains 14 bytes. In the format for monographs, the Library of Congress catalog card number is recorded in this field.

Fixed Fields

The fixed fields are fixed in length for all records and contain 54 bytes.

Record Directory

The record directory is made up of a variable number of fixed length entries (12 bytes each) which contains the identification tag, the length and the starting character position in the record of each variable field. The record directory ends with a field terminator code.

Tag	3 bytes
Length	4 bytes
Starting Character Position	5 bytes

Control fields

The control fields contain alphanumeric data elements and are recorded like variable fields although many have a fixed length. These fields end with a field terminator code. Each control field is identified by a 3-byte numeric tag in the record directory and these tags are not repeated in the logical record. In the MARC format for monographs, the Library of Congress catalog card number and the fixed fields are recorded as control fields.

Variable Fields

The variable fields are made up of variable length alphanumeric data, and all fields end with a field terminator code except the last variable field in the logical record which ends with an end of record code. Each variable field is identified by a 3-byte numeric tag in the record directory, and tags may be repeated as required in the logical record. Each variable field begins with a constant number of indicators which provide descriptive infor-

Record Directory

The record directory is made up of a variable number of fixed length entries (12 bytes each). The record directory ends with a field terminator code.

Tag	3 bytes
Sequence Number	1 byte
Blanks	3 bytes
Action Code	1 byte
Length	2 bytes
Starting Character Position	2 bytes

(Record control field and fixed fields)

Variable fields

The variable fields are made up of variable length alphanumeric data, and all fields end with a field terminator code except the last variable field in the logical record which ends with an end of record code. Each variable field is identified by a 3-byte numeric tag in the record directory and tags may be repeated as required in the logical record. Each variable field begins with the number of indicators required for that field and the

mation about that field. The field contains data elements which may be separated by subfield codes to identify the data element. A subfield code is composed of a delimiter and a lower case alphabetic character.

A variable field for monographs can be described as follows:

ii\$a data \$b data FT

where i = indicator

\$a and \$b = subfield code

FT = field terminator

\$ = delimiter

number of lower case alphabetic characters which are a part of the subfield code.

The indicators and alphabetic characters are each followed by a delimiter. The data elements of the field are separated by delimiters only. A variable field can be described as follows:

i₁ i₂... i_n \$a₁ a₂... a_n \$ data

\$ data \$ data FT

where i = indicator

a = alphabetic character

FT = field terminator

\$ = delimiter

It should be noted that in the communications format those fields that are fixed in length (Library of Congress catalog card number and the fixed fields) for a particular form of material, e.g., monographs, are recorded as variable fields. This guarantees that the same format structure is able to be used to represent all forms of material, e.g., serials, maps, music, etc., where the contents of these fields may vary in length and meaning. The MARC II communications format was designed for the exchange of bibliographic information recorded on magnetic tape to be manipulable by the recipient's computer regardless of its characteristics, i.e., word machines, character machines, third-generation and second-generation computers (3). The MARC processing format was designed for the library of Congress System 360.

DEVELOPMENT

The use of the COBOL programming language provided straightforward language approach to the task. The use of natural language provided the program with its own documentation.

In order to relate the program to its use on the specific computer located at the Library of Congress, only the following were required in COBOL:

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. IBM-360 E30.

OBJECT-COMPUTER. IBM-360 E30.

The MARC input file was treated as containing "undefined" records. In the MARC file, records actually vary in length up to 2048 characters.

In COBOL, the MARC file and records, along with the option card file and printed output file were described as:

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT MARC-FILE ASSIGN TO 'SYS012' UTILITY 2400 UNITS
RESERVE NO ALTERNATE AREAS.

SELECT OPTION-CARD ASSIGN TO 'SYS013' UNIT-RECORD
2540R RESERVE NO ALTERNATE AREAS.

SELECT INDEX-REPORT ASSIGN TO 'SYS014' UNIT-RECORD
1403.

DATA DIVISION.

FILE SECTION.

FD MARC-FILE

DATA RECORD IS MARC-RECORD
LABEL RECORDS ARE STANDARD
RECORDING MODE IS U.

01 MARC-RECORD.

02 MARC-BYTE PICTURE X OCCURS 2048 TIMES.

FD OPTION-CARD

DATA RECORD IS OPTION-RECORD
LABEL RECORDS ARE OMITTED
RECORDING MODE IS F.

01 OPTION-RECORD PICTURE X(80).

FD INDEX-REPORT

DATA RECORD IS INDEX-RECORD
LABEL RECORDS ARE OMITTED
RECORDING MODE IS F.

01 INDEX-RECORD.

02 CARRIAGE-CONTROL PICTURE X.
PICTURE X(130).

02 FILLER

The next step was to manipulate the fields of the MARC processing format which would provide access to the variable field information. These fields included: 1) the first 92 bytes of each record containing fixed-formatted items (leader, communication field, record control field, fixed fields); 2) a 12-byte directory entry that had been established for each variable field in the record containing the identification tag, the tag sequence number, the length of the field that this directory entry corresponded to, and the starting position of that field relative to the first position of the MARC record; and 3) the number of directory entries contained in each record and carried in the fixed portion of the record.

The specific programming approach taken was to move the fixed fields to a work area, calculate the locations of the directory entries and their corresponding fields, extract the desired field from the record and place

it in a work area for the appropriate processing. This technique was used to overcome the word boundary alignment considerations in the System 360, i.e., the use of binary arithmetic for certain data fields in the processing format. Since in some cases character-by-character scanning of the record data would be required, the programming routines were modularized to provide for economic core storage utilization and simple accessing.

The three COBOL modules (which could be used repetitively) were developed as follows, preceded by supporting work areas and subscripts:

Work Areas and Subscripts

WORKING-STORAGE SECTION.

01	FIXED-MARC.	
02	LENGTH	PICTURE 99 USAGE COMPUTATIONAL.
	.	.
02	DIRECTORY-COUNT	PICTURE 99 USAGE COMPUTATIONAL.
	.	.
01	MARC-FIXED REDEFINES FIXED-MARC.	
02	FIXED	PICTURE X OCCURS 92 TIMES.
01	HOLD-DIRECTORY.	
02	D-TAG	PICTURE X(3).
02	FILLER	PICTURE X(5).
02	D-LENGTH	PICTURE 99 USAGE COMPUTATIONAL.
02	D-ADDRESS	PICTURE 99 USAGE COMPUTATIONAL.
01	DIRECTORY-HOLD REDEFINES HOLD-DIRECTORY.	
02	D-HOLD	PICTURE X OCCURS 12 TIMES.
01	SUBSCRIPTS USAGE COMPUTATIONAL.	
02	DSUB	PICTURE 9(4) VALUE ZEROS.
02	TSUB	PICTURE 9(4) VALUE ZEROS.
01	HOLD-AREAS.	
02	HOLD-TAG	PICTURE X(3) VALUE SPACES.
02	HOLD-DATA.	
03	DATA-HOLD	PICTURE X OCCURS 1000 TIMES.
02	PERSW	PICTURE 9 VALUE ZERO.
01	OPTIONS.	
02	OPTION-TAG1	PICTURE X(3).
02	OPTION-TAG2	PICTURE X(3).
02	OPTION-TAG3	PICTURE X(3).
02	OTHER-OPTIONS	PICTURE X(71).

Continued to end of work areas and subscripts.

Modules

MOD-1.

NOTE

SUB-ROUTINE TO MOVE FIXED ITEMS FROM RECORD INTO WORK AREA. ENTER WITH DSUB AND TSUB SUBSCRIPTS SET TO ZERO. EXIT WITH FIXED ITEMS, ADDRESSABLE BY DATA-NAMES, IN WORK AREA LABELLED 'FIXED-MARC'.

MOVE-FIXED.

ADD 1 to DSUB. ADD 1 to TSUB.

MOVE MARC-BYTE (DSUB) TO FIXED (TSUB).

MOD-2.

NOTE

SUB-ROUTINE TO FIND A DESIRED IDENTIFICATION TAG IN THE RECORD DIRECTORY. SEARCH MUST BE CONTROLLED BY THE GIVEN NUMBER OF ENTRIES IN THE RECORD DIRECTORY. ENTER WITH DSUB SET TO 92 AND PERSW SET TO ZERO. SEARCH TAG MUST BE STORED IN 'HOLD-TAG'. EXIT WITH 12 CHARACTER DIRECTORY ENTRY IN WORK AREA LABELLED 'HOLD-DIRECTORY'.

SIFT-DIRECTORY.

IF PERSW EQUALS 1 GO TO SIFT-EXIT.

MOVE ZERO TO TSUB.

PERFORM DIRECTORY-MOVE 12 TIMES.

IF D-TAG EQUALS HOLD-TAG MOVE 1 TO PERSW.

SIFT-EXIT.

EXIT.

DIRECTORY-MOVE.

ADD 1 TO DSUB. ADD 1 TO TSUB.

MOVE MARC-BYTE (DSUB) TO D-HOLD (TSUB).

MOD-3.

NOTE

SUB-ROUTINE TO FIND DESIRED RECORD FIELD AND MOVE IT TO A WORK AREA FOR PROCESSING. ENTER WITH DESIRED DIRECTORY ENTRY IN WORK AREA LABELLED 'HOLD-DIRECTORY'. EXIT WITH DESIRED FIELD IN WORK AREA LABELLED 'HOLD-DATA'.

MOVE-DATA.

MOVE ZEROS TO TSUB.

MOVE SPACES TO HOLD-DATA.

MOVE D-ADDRESS TO DSUB.

PERFORM MOVE-A D-LENGTH TIMES.

MOVE-A.

ADD 1 TO DSUB. ADD 1 TO TSUB.

MOVE MARC-BYTE (DSUB) TO DATA-HOLD (TSUB).

Once access to any field in the record was accomplished by the use of sub-routine modules, what remained was to establish the "mainline coding" and to apply the specifications and logic for the reports that were

to be produced. With this objective, the following COBOL statements were written:

PROCEDURES DIVISION.

HOUSEKEEPING.

OPEN INPUT MARC-FILE OPTION-CARD OUTPUT
INDEX-REPORT.

READ OPTION-CARD INTO OPTIONS TO END GO TO A.

A.

READ MARC-FILE RECORD AT END GO TO ENDJOB.

MOVE ZEROS TO DSUB TSUB.

PERFORM MOVE-FIXED 92 TIMES.

MOVE OPTION-TAG1 TO HOLD-TAG.

PERFORM B.

MOVE OPTION-TAG2 TO HOLD-TAG.

PERFORM B.

MOVE OPTION-TAG3 TO HOLD-TAG.

PERFORM B.

GO TO A.

B.

MOVE ZEROS TO PERSW.

MOVE 92 TO DSUB.

PERFORM SIFT-DIRECTORY THRU SIFT-EXIT
DIRECTORY-COUNT TIMES.

PERFORM MOVE-DATA.

(At this point, the field is formatted for output and printed.)

ENDJOB.

CLOSE MARC-FILE OPTION-CARD INDEX-REPORT.

STOP RUN.

RESULTS

Figure 3 is a sample of the output of the program. The character-by-character scan, implicit in the programming required of a variable length field, also gave rise to extremely flexible data report formatting. All the data printed is "floating" in nature, restricted only by the user's print format requirements which were introduced by option card in this case.

Table 2 shows the amount of core occupied during processing, and Table 3 lists man weeks expended in producing the program.

Table 2. Computer Core Usage

<i>Program Sections</i>	<i>Bytes Occupied</i>
Software I/O Routines	1,155
COBOL Compiler Sub-routines	2,071
Program I/O Buffers and Constants	4,110
Object Instructions	3,944
TOTAL	11,280

construction	23-2660
Increasing frost resistance of road toppings in the Tiumen' regio	23-3164
Washing concrete aggregates in freezing weather	23-3161
CONTACT PHOTOGRAPHY	
Contact method of photographing snow and firn samples	23-2799
CONTINUOUS LOADING EFFECTS	
Snow creep under continuous loading	23-2708
CONSTRUCTION	
Building embankments in freezing weather	23-3150
CONVECTION	
Calculating thawing depth of frozen moist soil	23-2217
Convective heat exchange in radiant gas	23-2473
Convective heat exchange in water saturated ground	23-2483
Laminar thermal convection in a rotating void between two discs	23-2477
CONVECTIVE HEAT EXCHANGE	
Convective heat exchange in radiant gas	23-2473
COOLING SYSTEMS	
Optimal parameters of cooling systems operating on gas regeneration with heat release	23-2445
COORDINATES	
Ice movement determination from astronomical observations	23-3070
CORES	
Analysis of ice cores from Byrd Station	23-3113
Deep-core drilling program at Byrd Station	23-3112
Results of Antarctic core hole to bedrock	23-3126
COSMIC DUST	
Analysis of magnetic particles from Greenland ice	23-3213
COUNTERMEASURES	
Countermeasures for icing	23-2567
Frost heave countermeasures in road construction	23-2308
Investigating frost heave areas	23-2770
Investigating frost heave areas on railroad tracks	23-2729
Landslide countermeasures	23-2602
Mass movement and effective countermeasures	23-2731
Preventing the appearance of nailed near small bridges and pipes	23-2584
Protecting roads from landslides by building outlets for solifluction material	23-2595
Rock streams	23-2604
CRITICAL SOUND PRESSURE	
Critical value of sound pressure for the processes of heat and mass transfer taking place under the influence of acoustic vibrations	23-2506
CRYOGENIC PROCESSES	
Permafrost in Tien Shan	23-3018
Thermal regime of soils in permafrost regions	23-2226
CRYOGENIC RELIEF	
Locating minor structures from cryogenic relief and permafrost indications	23-3100
Supergene distribution of radium and thorium in the Transbaikal taiga	23-2862
Soils of Urals, West and Central Siberia	23-3193
Tundra and forest-tundra soils	23-3195
CRYOGENICS	
Cryogenic engineering	23-2276
CRYOTURBATION	
Cryogenic fossil crevasses in L'islet County	23-2314
CRYSTAL GROWTH	
Patterns on the ice surface of a lake	23-2714
CRYSTAL LATTICES	
Growth of an ice crystal in analogy with an electrostatic field	23-2624
CRYSTAL STRUCTURE	
Crystal structure of water	23-2245
Growth of snowflakes	23-2874
Silver iodide nucleating sites	23-2269
Snow crystals in Pushimi District, Kyoto	23-2878
CRYSTAL STUDY TECHNIQUES	
Complexities of the three-dimensional shape of individual crystals in glacier ice	23-2943
CRYSTALS	
Physical properties of molecular crystals, liquids, and glasses	23-2231
CUBIC ICE	
Hexagonal and cubic ice at low temperature	23-2651
Tensile strength of cubic crystals under pressure	23-2928
CYCLONE BLOWING SNOW METER	
"Cyclone" blowing snow meter and its use at Mirny	23-3073
CRYSTAL STUDY TECHNIQUES	
Contact method of photographing snow and firn samples	23-2799
DAMAGE	
Avalanches on Rebun Island, Japan	23-2933
Damage by snowstorm of Jan. 1963 in Japan	23-2867
Forest damage caused by avalanches	23-2875
Snow and ice damage on electric communication lines in Hokkaido	23-2881
DAMAGE TO FOREST TREES	
Comparative studies of avalanche injury and wind damage to forests	23-2424
DAMS	
Building dams of moraine deposits	23-2556
Building embankments in freezing weather	23-3150
Changing the hydrological regime of a river by controlling its flow	23-2429
The year-round construction of the Vilyuy power station dam in the Extreme North	23-2982
DEFORMATION	
Building deformations caused by frost heave	23-2607
Concrete deformation due to shrinkage at minus temperatures	23-2558
Deformation of bridge abutments erected on permafrost	23-2599
Roadbed deformation due to ground thawing and frost heave	23-2865
Stability of foundations built on frost heaving ground	23-2598
Strains in concrete due to negative temperatures	23-2845
DEGREE DAYS	
Development of shore ice in the Lazarev station region	23-3037

Fig. 3. Output of COBOL Language Program Using MARC II Data.

Table 3. Manpower Expenditure

Activity	Man Weeks
Analysis and Programming	1
Debugging and Checkout	2
TOTAL	3

Since the processing time of a print program is usually a function of the speed of the printer, no accurate internal processing times were recorded. However, there was no noticeable time difference between this program and other MARC print programs written at the Library of Congress in assembly language.

COMMUNICATION FORMAT PROCESSING

The aforementioned techniques are equally adaptable for use with the MARC II communications format (3) with the following changes in format conventions: 1) The communication format has a 24-character leader rather than 92 characters of fixed length items in the processing format. In the program, under the "WORKING-STORAGE SECTION", the group item labelled "FIXED-MARC" would have to be redefined to conform with the 24-character leader. The COBOL statements that are noted with "..." would require a change of their value from "92" to "24". 2) The communication format has no total count of entries in the record directory. A calculation would have to be made to arrive at the total count and that figure stored in a new hold area labelled "DIRECTORY-COUNT".

The base address of the data in the communication format is not relative to the first position of the record as defined in the processing format, but to the first position of the first variable field. This base address is carried in the record leader, and is available for the calculation required for the Directory Entry Count (Base address -24/12).

In the program, after the record directory had been searched and the proper entry placed in the work area, the "MOVE-DATA" sub-routine would move the appropriate field to the work area for processing with the one alteration noted below with an asterisk.

MOVE-DATA.

```

MOVE ZEROS TO TSUB.
MOVE SPACES TO HOLD-DATA.
MOVE D-ADDRESS TO DSUB.
ADD BASE-ADDRESS TO DSUB.*
```

```
PERFORM MOVE-A D-LENGTH TIMES.
```

MOVE-A.

```

ADD 1 TO DSUB. ADD 1 TO TSUB.
MOVE MARC-BYTE (DSUB) TO D-HOLD (TSUB).
```

Programming techniques naturally are dependent on the processing required and the format characteristics at the individual institution. If the MARC II communications format were to be manipulated in the form

in which it is received (each byte equal to a character with a 24-character leader followed by 12-character directory entries) an alternate approach to that suggested above could be to work in the record area and not move data to a work area.

CONCLUSION

The only MARC II data available to users up to the writing of this article (October 1968) has been the MARC II test tape released by the Library of Congress in August 1968. Therefore, it is probable that most people expressing doubts about the use of COBOL with MARC records have done so without the experience of actually using the language. We now have this experience at the Library of Congress. COBOL was successfully used for the computer processing of MARC records. The complexity of the record did not detract from ease in programming.

Although the programs written were for a report function, the data accessing modules of COBOL nevertheless can be used for many other functions. File maintenance and retrieval algorithms could be defined and programmed in COBOL with facility equal to that in programming the subject function.

REFERENCES

1. Griffin, Hillis: "Automation of Technical Processes in Libraries," In *Annual Review of Information Science and Technology*, edited by Carlos A. Cuadra (Chicago: Encyclopaedia Britannica) 3 (1968), 241-262.
2. U. S. Library of Congress, Information Systems Office: *Subscriber's Guide to the MARC Distribution Service* (Washington, D. C.: Library of Congress, 1968).
3. Avram, Henriette D.; Knapp, John F.; Rather, Lucia J.: *The MARC II Format: A Communications Format for Bibliographic Data* (Washington, D. C.; Library of Congress, 1968), pp. 1, 2, 10.