LAUREL COBOL MAINTENANCE SUPPORT SYSTEM

This is a maintenance programmer's productivity improvement tool. It makes poor to middling quality programs more legible, thus making it easier to get a handle on what they're doing, thus shortening the time it takes to implement changes and/or enhancements.

With the foregoing disclaimer out of the way, be advised that the System has been in use for several years with no known bugs, but it should be viewed as a *computer assisted* source program reiteration tool rather than a *do it all* product. A given source program may have to be run through the System several times, with various control table changes, before the user is fully satisfied with the results.

F O R W A R D

This product is designed to lighten the program maintenance burden by improving program legibility before changes are applied. Illegible programs are difficult to upgrade.

Illegibility stems from poor organization, cryptic abbreviations, sequencing schemes that have collapsed over a period of time, and multiple coding styles, to name the major causes - and the LCMS System can fix them.

The System is not a Restructuring Engine in that it doesn't add code or change the logic of what is present - but it does reiterate the source program by applying various literary standards that will improve legibility anywhere from a modest to a remarkable degree, depending upon the original condition.

Experience has shown that this improved legibility often halves the time it takes to make a significant modification or upgrade to an average program.

The System has also proven useful in new program development, permitting the programmer an otherwise forbidden "quick & dirty" approach that can be cor-rected by a final-step recomposition; and especially useful in Assembler to COBOL conversions, allowing the programmer to employ as close to a one-for-one reiteration as the two languages permit - with a final-step recomposition thru the LCMS System to clean everything up.

I N T R O D U C T I O N

Perhaps one can't apply a scientific definition to the term *Literary Standards* but it's obvious to all that there's a difference between the works of Shakespear and those of the old *pulp western* authors, just as there's a difference in the organization of college text books and those produced for the grade school student.

Computer programs can be viewed as literary products too; differently structured because they're written for both people *and* computers - but they're not all that remote from the mainstream that similar literary standards can't be applied to them. Literary standards *should* be applied to them.

As an example, consider the following:

--- BEFORE ---

```
       FILE-CONTROL.
00001      SELECT FILE1          ASSIGN TO TRANFILE.                ARP910
00002      SELECT FILE2          ASSIGN TO VSAM-CUSTOMER            ARP910
00003                          ORGANIZATION IS INDEXED              ARP910
00004                          ACCESS MODE IS DIRECT                ARP910
00005                          FILE STATUS IS R2-STATUS             ARP910
00006                          RECORD KEY IS R2-RECORD-KEY.         ARP910
00007      SELECT FILE3          ASSIGN TO SYSPRINT.                ARP910
       DATA DIVISION.
       FILE SECTION.
       FD   FILE1 DATA RECORD IS RECORD1
            RECORD CONTAINS 35 TO 100 CHARACTERS
            BLOCK CONTAINS 0 RECORDS
            LABEL RECORDS ARE STANDARD
            RECORDING MODE IS V.
       01   RECORD1, PICTURE X(100).
```

--- AFTER ---

```
000190 FILE-CONTROL.                                                ARP910
000200                                                             ARP910
000210      SELECT FILE1, ASSIGN TO TRANFILE.                       ARP910
000220      SELECT FILE2, ASSIGN TO VSAM-CUSTOMER,                  ARP910
000230        ORGANIZATION IS INDEXED,                              ARP910
000240          FILE STATUS IS R2-STATUS,                           ARP910
000250            ACCESS MODE IS DIRECT,                            ARP910
000260              RECORD KEY IS R2-RECORD-KEY.                    ARP910
000270      SELECT FILE3, ASSIGN TO SYSPRINT.                       ARP910
000280                                                             ARP910
000290 DATA DIVISION.                                              ARP910
000300 FILE SECTION.                                               ARP910
000310                                                             ARP910
000320 FD   FILE1 DATA RECORD IS RECORD1,                           ARP910
000330        RECORD CONTAINS 35 TO 100 CHARACTERS,                 ARP910
000340          BLOCK CONTAINS 0 RECORDS,                           ARP910
000350            LABEL RECORDS ARE STANDARD,                       ARP910
000360              RECORDING MODE IS V.                            ARP910
000370   01 RECORD1, PICTURE X(100).                                ARP910
000380                                                                    ARP910
```

RECOMPOSITION EXAMPLES

--- BEFORE ---

```
000870 01   NUMERIC-CONVERSION-FIELDS.
000880       03   NCF-1.
000890            07   NCF-1A, PICTURE XXXX.
000900              07   NCF-1B, PICTURE S9, COMP-3, VALUE ZERO.
000910       03   NCF-2 REDEFINES NCF-1.
000920           07   NCF-2A, PICTURE S9(9), COMP-3.
000930       03   NCF-3, PICTURE 9(9).
000940       03   NCF-4 REDEFINES NCF-3.
000950         04   NCF-4A, PICTURE 999.
000960           04   NCF-4B, PICTURE 9999.
000970          04 NCF-4C, PICTURE 9.
000980           04   FILLER, PICTURE X.
000990       03   NCF-5, PICTURE 9(9).
   000      03 NCF-6 REDEFINES NCF-5.
   010           06   NCF-6A, PICTURE 9(6).
   020           06 NCF-6B, PICTURE 999.
```

--- AFTER ---

```
000840   01 NUMERIC-CONVERSION-FIELDS.                                ARP910
000850        03 NCF-1.                                             ARP910
000860             05 NCF-1A              PICTURE XXXX.             ARP910
000870             05 NCF-1B              PICTURE S9, COMP-3, VALUE ZERO. ARP910
000880        03 NCF-2 REDEFINES NCF-1.                             ARP910
000890             05 NCF-2A              PICTURE S9(9), COMP-3.    ARP910
000900        03 NCF-3                    PICTURE 9(9).             ARP910
000910        03 NCF-4 REDEFINES NCF-3.                             ARP910
000920             05 NCF-4A              PICTURE 999.              ARP910
000930             05 NCF-4B              PICTURE 9999.             ARP910
000940             05 NCF-4C              PICTURE 9.                ARP910
000950             05 FILLER              PICTURE X.                ARP910
000960        03 NCF-5                    PICTURE 9(9).             ARP910
000970        03 NCF-6 REDEFINES NCF-5.                             ARP910
000980             05 NCF-6A              PICTURE 9(6).             ARP910
000990             05 NCF-6B              PICTURE 999.              ARP910
```

The LCMS System reorganizes Data Division record layouts into five B-margin indents of 4 character positions, each, based on data level number.

Options exist for converting PICTURE to PIC or *vice-versa* and FILLER to FIL or *vice-versa*.

The PICTURE clause may abutt the data name, separated by a comma (the de-fault), or may be set to begin anywhere from columns 25 thru 45.

Comment lines will be passed *as-is*, but multiple spacer lines will be eliminated. A single spacer line will be emitted ahead of each 01-level data name.

--- AFTER ---

```
004110 2000-PROCESS-ROUTINE.                                          APXPS104
004120     IF DETAIL-MAINT-CODE = 'D'                                 APXPS104
004130         GO TO 2000-READ-NEXT.                                  APXPS104
004140     IF DETAIL-APCODE = CONTROL-PROCESSING-APCODE OR                APXPS104
004150             CONTROL-PROCESSING-APCODE = ZERO                       APXPS104
004160         NEXT SENTENCE                                          APXPS104
004170           ELSE                                                 APXPS104
004180         GO TO 2000-READ-NEXT.                                  APXPS104
004190     IF CONTROL-PROCESSING-DATE > DETAIL-PROCESS-DATE OR            APXPS104
004200             CONTROL-PROCESSING-DATE = DETAIL-PROCESS-DATE          APXPS104
004210         IF WS-FIRST-TIME-SW = 'Y'                              APXPS104
004220             MOVE DETAIL-VENDOR-NUMBER TO HOLD-VENDOR-NUMBER        APXPS104
004230             MOVE DETAIL-ACCOUNT-NUMBER TO                          APXPS104
004240                     HOLD-WHOLE-ACCOUNT-NUMBER                      APXPS104
004250             MOVE DETAIL-MATCH-CODE-NUMBER TO                       APXPS104
004260                     HOLD-MATCH-CODE-NUMBER                         APXPS104
004270             MOVE DETAIL-APCODE TO HOLD-APCODE                      APXPS104
004280             MOVE 'N' TO WS-FIRST-TIME-SW                       APXPS104
004290             PERFORM 8400-READ-GLADESC THRU 8400-EXIT               APXPS104
004300             PERFORM 8500-READ-VENDOR THRU 8500-EXIT                APXPS104
004310             PERFORM 3000-PROCESS-RECORD THRU 3000-EXIT             APXPS104
004320               ELSE                                             APXPS104
004330             PERFORM 3000-PROCESS-RECORD THRU 3000-EXIT.            APXPS104
```

The Procedure Division normally employs varying amounts of compound and complex sentence syntaxt. The LCMS System attempts to separate dependent clauses into separate lines for the sake of improved legibility.

Dependent clauses begin with a COBOL reserved word (or phrase), and those that should prompt a new line at the next indent are contained in the table file INDENT1.LCS. Some of these reserved words are treated differently than others;
as for example note the special treatment of the ELSE in the preceeding speci-men. There are, in fact, nine different alogrythms for the various types of dependent clauses. These are summarized on page 14.

If a clause can't be contained in a single line, then the continuation is given a double indent. The preceeding specimen has three examples of this.

The first example shows the necessity of the double indent - because of the immediately following phrase. The second and third examples might look better with only a single indent, but the LCMS System doesn't know in advance what's going to follow. These can be manually adjusted should the user desire to do so.

RECOMPOSITION EXAMPLES

--- BEFORE ---

```
 00070 UPDATE-CUSTOMER-RECORD.
 00080       IF CS-TRAN=CODE = SPACES GO TO S1.
 00090       MOVE CHARGE-SEGMENT TO MONETARY-TRANSACTION-TABLE (1).
031000       MOVE +1 TO MONETARY-TRANSACTION-NBR-OCCURS.
031010       MOVE +2 TO SS2, GO TO S2.
031020 S1. MOVE +1 TO SS2.
031030 S2. MOVE +1 TO SS1, GO TO T2.
031040 T1. ADD +1 TO SS1.
             IF SS1 IS GREATER THAN +4 GO TO T3.
         T2. IF TS-TRANCODE (SS1) = SPACES GO TO T1.
031070       MOVE TRANSACTION-SEGMENT (SS1)
031080       TO MONETARY-TRANSACTION-TABLE (SS2).
031090       MOVE SS2 TO MONETARY-TRANSACTION-NBR-OCCURS.
003000       ADD +1 TO SS2, GO TO T1.
003010 T3. EXIT.
```

--- AFTER ---

```
002910                                                                  ARP910
002920 UPDATE-CUSTOMER-RECORD.                                           ARP910
002930           IF CS-TRAN=CODE = SPACES                               ARP910
002940               GO TO TAG012.                                      ARP910
002950           MOVE CHARGE-SEGMENT TO MONETARY-TRANSACTION-TABLE (1).  ARP910
002960           MOVE +1 TO MONETARY-TRANSACTION-NBR-OCCURS.             ARP910
002970           MOVE +2 TO SS2,                                        ARP910
002980               GO TO TAG013.                                      ARP910
002990 TAG012. MOVE +1 TO SS2.                                          ARP910
003000 TAG013. MOVE +1 TO SS1,                                          ARP910
003010               GO TO TAG015.                                      ARP910
003020 TAG014. ADD +1 TO SS1.                                           ARP910
003030           IF SS1 IS GREATER THAN +4                              ARP910
003040               GO TO TAG016.                                      ARP910
003050 TAG015. IF TS-TRANCODE (SS1) = SPACES                            ARP910
003060               GO TO TAG014.                                      ARP910
003070           MOVE TRANSACTION-SEGMENT (SS1) TO                      ARP910
003080               MONETARY-TRANSACTION-TABLE (SS2).                  ARP910
003090           MOVE SS2 TO MONETARY-TRANSACTION-NBR-OCCURS.           ARP910
003100           ADD +1 TO SS2,                                         ARP910
003110               GO TO TAG014.                                      ARP910
003120 TAG016. EXIT.                                                    ARP910
003130                                                                  ARP910
```

Many programs don't use them, but here you see an example of a program that employs *Statement names* distinct from *Paragraph names*. Both are *Procedure names*, but logically, statements, labeled or not, are part of a specific paragraph - and the LCMS System enforces a naming standard for statement labels that make their subordinant status obvious to the reader.

There are two reiteration programs. One generates 2-character statement labels in the manner shown in the *before* example, and the other generates 6-byte labels as seen in the *after* example, with a B-Margin option of either column 12 or column 16.

RECOMPOSITION EXAMPLES

--- BEFORE ---

```
019840           IF TH-CODE-FTRS-LANGUAGE = 'F' AND (TH-CODE-FTRS-GTA = 'NSCC3C000
```

```
019850-              " OR SPACES)                                        SCC3C000
019860               MOVE 'TCL-2' TO WS-PARAMETER                        SCC3C000
019870               PERFORM 67000-MOVE-PARAMETER.                       SCC3C000
019880             IF TH-CODE-FTRS-LANGUAGE = 'F' AND TH-CODE-FTRS-GTA = 'Y'SCC3C000
019890               MOVE 'TCL-9' TO WS-PARAMETER                        SCC3C000
019900               PERFORM 67000-MOVE-PARAMETER.                       SCC3C000
019910             IF TH-CODE-FTRS-CALL-FOLLOWING = 'Y'                  SCC3C000
019910             IF TH-CODE-FTRS-CALL-FOLLOWING = 'Y'                  SCC3C000
019910             IF TH-CODE-FTRS-CALL-FOLLOWING = 'Y'                  SCC3C000
019920               MOVE 'MRF-3' TO WS-PARAMETER                        SCC3C000
                     ELSE
019930               PERFORM 67000-MOVE-PARAMETER                        SCC3C000
                     ELSE
019930               PERFORM 67010-MOVE-PARAMETER                        SCC3C000
                     ELSE
019930               PERFORM 67050-MOVE-PARAMETER.                       SCC3C000
019940             IF TH-CODE-FTRS-CALL-FORWARDING = 'Y'                 SCC3C000
019950               MOVE 'CTR-2' TO WS-PARAMETER                        SCC3C000
019960               PERFORM 67000-MOVE-PARAMETER.                       SCC3C000
```

--- AFTER ---

```
001550       IF TH-CODE-FTRS-LANGUAGE = 'F'                              SCC3C000
001560       AND (TH-CODE-FTRS-GTA = 'N' OR SPACES)                      SCC3C000
001570            MOVE 'TCL-2' TO WS-PARAMETER                           SCC3C000
001580            PERFORM 67000-MOVE-PARAMETER.                          SCC3C000
001590       IF TH-CODE-FTRS-LANGUAGE = 'F'                              SCC3C000
001600       AND TH-CODE-FTRS-GTA = 'Y'                                  SCC3C000
001610            MOVE 'TCL-9' TO WS-PARAMETER                           SCC3C000
001620            PERFORM 67000-MOVE-PARAMETER.                          SCC3C000
001630       IF TH-CODE-FTRS-CALL-FOLLOWING = 'Y'                        SCC3C000
001640            IF TH-CODE-FTRS-CALL-FOLLOWING = 'Y'                   SCC3C000
001650                 IF TH-CODE-FTRS-CALL-FOLLOWING = 'Y'              SCC3C000
001660                      MOVE 'MRF-3' TO WS-PARAMETER                 SCC3C000
001670                        ELSE                                       SCC3C000
001680                      PERFORM 67000-MOVE-PARAMETER                 SCC3C000
001690                        ELSE                                       SCC3C000
001700                 PERFORM 67010-MOVE-PARAMETER                      SCC3C000
001710                   ELSE                                            SCC3C000
001720            PERFORM 67050-MOVE-PARAMETER.                          SCC3C000
001730       IF TH-CODE-FTRS-CALL-FORWARDING = 'Y'                       SCC3C000
001740            MOVE 'CTR-2' TO WS-PARAMETER                           SCC3C000
001750            PERFORM 67000-MOVE-PARAMETER.                          SCC3C000
```

There are multiple uses of the *OR* operator, and the LCMS System chooses to manipulate it as shown in the preceeding example. Some nested IFs may have to be manually corrected, however, to get them to appear as the user wants them.

Many programmers choose to key their indent structure on the ELSE operator rather than on the operative statement that follows it, but the LCMS System feels that it should be just the opposite.

In other words, the indent is keyed to the operative statement with the ELSE tucked out of the way so as not to interfere with the reader's comprehension of the syntaxt.

The reader is reminded that COBOL is a syntaxt-oriented language translator, and it pays no attention whatever to any indent structure that may or may not be present. Indents are for people, and whatever works best is that which should be used.

--- BEFORE ---

```
021170          EXEC CICS READ DATASET('SCVTA008')                      SCC3C000
021180                               INTO (TRANSACTION-HISTORY-RECORD)  SCC3C000
021190               RIDFLD(TH-RECORD-KEY) EQUAL NOHANDLE               SCC3C000
021200               RESP(RESPONSE-CODE) END-EXEC.                      SCC3C000
```

--- AFTER ---

```
002260      EXEC CICS READ                                              SCC3C000
002270           DATASET ('SCVTA008')                                   SCC3C000
002280           INTO (TRANSACTION-HISTORY-RECORD)                      SCC3C000
002290           RIDFLD(TH-RECORD-KEY)                                  SCC3C000
002300           EQUAL                                                  SCC3C000
002310           NOHANDLE                                               SCC3C000
002320           RESP(RESPONSE-CODE)                                    SCC3C000
002330           END-EXEC.                                              SCC3C000
```

The LCMS System reiteration programs employ two external table files for indent processing. INDENT1 (previously described) controls the basic COBOL dependent clause recognition, and INDENT2 is to be used for whatever special handling may be appropriate for external calls to a communications monitor or data base management system, etc.

Both indent tables are supplied with the System, with INDENT2 being set up for CICS. Table layouts are described at the end of the manual.

Basically, the LCMS System has three operations:

(1) RELEVEL Renumbers data name level numbers so that they are consistant throughout the entire Data Division. Specification of the six low-order level numbers (beyond 01) is requested by the RELEVEL program at startup time.

(2) REPLACE Replaces cryptic abbreviations and any other data or procedure names with whatever is deemed to be more appropriate. Word replacement is controlled by a table file which must be prepared prior to REPLACE program run time.

(3) REWRITE Reiterates the source program according to the literary standards and various options described before and below.

There are two reiteration programs: REWRITE1 employs a six-character statement label in the form TAGnnn with a B-Margin option of either columns 12 or 16. REWRITE2 employs a two-character statement label and a B-Margin fixed at column 12. The statement labels run in the range A1 thru Z9, then AA thru ZZ excepting the reserved COBOL words of IF, TO, OR, etc. A control table called LABELS is required by REWRITE2.

The following utility programs are also supplied with the System:

(1) MP021 Resequences the COBOL source program and emits the program name in columns 72 thru 80 of each statement (expands trun-cated records to 80 characters).

(2) MP023 Compresses 80-characrter records down to the last non-blank character (PC VBS Format). Can save significant disk space while modifying a source program with a word processor.

Whenever a reiterated program requires one or more manual revisions to what-ever the LCMS did to it, then MP021 may be used to resequence it again.

Required control tables:

LABELS The legal 2-character statement labels used by REWRITE2.
INDENT1 All the basic COBOL verbs that require continued line indents. May cover both VS COBOL and COBOL-II (and include DB2 verbs as well).
INDENT2 Auxilliary indent table for communications monitor or database management system, etc. Supplied version is for CICS.

The structure of the Indent tables is described on page 14.

The word replacement table required by REPLACE is specific to each program be-ing reiterated. Its layout is *Old Word* beginning in column 1 followed by *New Word* with at least one blank separating them.

Some Download software copy programs in their original 80-character format (plus the CR/LF symbols) and some truncate the records to their last non-blank character. The LCMS System can handle either form, but it isn't designed to handle embedded tab skip symbols, such as might be emitted by a PC word processor.

Any program with embedded tab skip symbols must be passed through MP021 before it is submitted to the LCMS System. By the same token, if the output program is retouched by a word processor it must be passed through MP021 before being uploaded.

RESEQUENCING DATA DEFINITION LEVEL NUMBERS

This function is required when a program has inconsistent level number assign-ments. Given the following data definitions ...

```
000750 01   BREAK-CONTROL.
000760       02   BC-MERCHANT-NBR, PICTURE 999.
000770       02   BC-CUSTOMER-NBR, PICTURE 9999.
000780       02   BC-CHECK-DIGIT, PICTURE 9.
000790
000800 01   WORKING-DATE.
000810       10   WD-CENTURY, PICTURE 99, VALUE 19.
000820       10   WD-YYMMDD.
000830            15   WD-YEAR, PICTURE 99.
000840            15   WD-MONTH, PICTURE 99.
000850            15   WD-DAY, PICTURE 99.
```

... you will note that level 02 for BREAK-CONTROL has the same relative in-dent as level 10 for WORKING-DATE. This presents a problem for the REWRITE because it requires complete consistency throughout the entire program in level number assignments for indent selection.

Level numbers may be in the series 02, 03, 04, 05, etc., or 03, 05, 07, etc., but if they aren't all the same for all data definitions then the RELEVEL pro-gram must be used ahead of the REWRITE program. Setup is as follows:

```
RELEVEL
MP024 Starting
Adjusts COBOL Data Definition Level Numbers
Specify Input File DSNAME: C:ARP910.COB
Specify Work File DSNAME: C:ARP910.WRK
Specify Output File DSNAME: C:ARP910.REL
Specify the lowest 6 data level-numbers beyond 01 to be used in the program
by overkeying where required: 03,05,07,09,11,13
MP024 Finished
```

The requests for DSNAMEs show specimen responses, but the message requesting data level numbers emits the most commonly used numbers. If they are satis-factory as is, then simply press the ENTER key and RELEVEL will execute.

REPLACING CRYPTIC ABBREVIATIONS

In addition to cryptic abbreviations the REPLACE program may be used to simply change data names and procedure names to more appropriate selections. Anything that makes a program easier to follow is appropriate.

Getting set up for a REPLACE run takes a little effort. You must peruse the original source listing and create a word replacement table for the words that are to be changed. You may use a word processor or the EDLIN utility. Each table record must be in the form...

OLDWORD NEWWORD

... each entry may be 30 characters in length, and the pair must be separated by at least one blank. A columnar alignment for "NEWWORD" is optional, and the table does not have to be in sequence.

Setup for the REPLACE program is as follows:

```
REPLACE
MP025 Starting
COBOL Word Replacement Program
Specify Single or Double Quotes (S or D): S
Specify Input File DSNAME: C:ARP910.REL
Specify Word Replacement Table: C:APR910.TBL
Specify Output File DSNAME: C:ARP910.REP
MP025 Finished
```

In the normal course of events, several passes by REPLACE may be necessary be-fore the replacement table covers everything you want fixed.

TSO and PC word processors have word replacement facilities, but word proces-sors don't care if they do an overset past column 72, and TSO will refuse an overset past column 72. REPLACE will process all replacement requests, and if an overset would otherwise occur it creates a continuation line - and unlike TSO and word processors, it will not replace words occurring within a literal string. Where such is necessary, you will have to manually process each one individually.

Where a given program is to have no statement names (as distinct from para-graph names), then all procedure names must be eight characters or more in length (and even eight character names could be viewed as being cryptically abbreviated).

REWRITE1
MP029 Starting
Laurel COBOL Recomposition System Reiteration
Specify Program Name:
Specify Quote Symbol (S or D):
Recompose Procedure Division Only (Y/N)?
Specify the lowest 6 data level-numbers beyond 01 to be used in the program
by overkeying where required: 03,05,07,09,11,13
Use Short Form of FILLER (Y/N)?
Use Short Form of PICTURE (Y/N)?
Specify Picture Clause Margin (25-45):
Specify Procedure Division B Margin (12/16):
Specify Primary Indent Table DSNAME: C:INDENT1.LCS
Specify Auxilliary Indent Table DSNAME: C:INDENT2.LCS
Specify Input File DSNAME:
Specify Workfile DSNAME: C:REWRITE.WRK
Specify Output File DSNAME:
So you want Statement XREF Report (Y/N)?
MP029 Beginning Reiteration Run
Specify XREF File DSNAME: [if response was Yes]
MP029 Finished

Pressing the ENTER key without first keying a response to the requests for Quote Symbol and PICTURE Clause Margin will result in system defaults of "S" and no PICTURE clause indent, respectively.

Most reiteration runs require more than one pass through the System before the Procedure Division fits the required results. Intermediate stages may be shortened by bypassing reiteration of the other divisions.

All requests for file names should be preceeded by the drive ID. If not specified, the default will be assigned to drive B rather than the system disk.

All Yes/No setup requests default to No. Predefined requests default to the values shown, unless overkeyed.

This program requires the INDENT1.LCS and INDENT2.LCS control tables.

If the program being reiterated has statement labels, then a cross-reference file may optionally be written. If requested, it may be listed by the DOS PRINT facility.

REWRITE2
MP030 Starting
Laurel COBOL Recomposition System Reiteration
Specify Program Name:
Specify Quote Symbol (S or D):
Recompose Procedure Division Only (Y/N)?
Specify the lowest 6 data level-numbers beyond 01 to be used in the program
by overkeying where required: 03,05,07,09,11,13
Use Short Form of FILLER (Y/N)?
Use Short Form of PICTURE (Y/N)?
Specify Picture Clause Margin (25-45):
Specify Labels File DSNAME: C:LABELS.LCS
Specify Primary Indent Table DSNAME: C:INDENT1.LCS
Specify Auxilliary Indent Table DSNAME: C:INDENT2.LCS
Specify Input File DSNAME:
Specify Workfile DSNAME: C:REWRITE.WRK
Specify Output File DSNAME:
So you want Statement XREF Report (Y/N)?
MP030 Beginning Reiteration Run
Specify XREF File DSNAME: [if response was Yes]
MP030 Finished.

Pressing the ENTER key without first keying a response to the requests for Quote Symbol and PICTURE Clause Margin will result in system defaults of "S" and no PICTURE clause indent, respectively.

Most reiteration runs require more than one pass through the System before the Procedure Division fits the required results. Intermediate stages may be shortened by bypassing reiteration of the other divisions.

All requests for file names should be preceeded by the drive ID. If not specified, the default will be assigned to drive B rather than the system disk.

All Yes/No setup requests default to No. Predefined requests default to the values shown, unless overkeyed.

This program requires all three control tables.

If the program being reiterated has statement labels, then a cross-reference file may optionally be written. If requested, it may be listed by the DOS PRINT facility.

REWRITE ERROR MESSAGES

Preferably, the source program to be reiterated should be error free - but the REWRITE program has a limited error diagnostic facility. When an error is de-tected, the offending source statement will be sent to the output file with the following message:

***ERROR CONDITION 'x' (SKIPPING REMAINDER OF INPUT PROGRAM)

Error Code Index

A = IDENTIFICATION DIVISION not found.
C = INPUT-OUTPUT SECTION not followed by FILE-CONTROL or I-O CONTROL.
D = FILE-CONTROL not followed by a SELECT statement.
E = ENVIRONMENT DIVISION not followed by:
(1) CONFIGURATION SECTION or
(2) INPUT-OUTPUT SECTION or
(3) DATA DIVISION
F = I-O-CONTROL encountered without preceeding FILE-CONTROL.
G = DATA DIVISION not found.
H = DATA DIVISION found but not immediately followed by:
(1) FILE SECTION or
(2) WORKING-STORAGE SECTION or
(3) LINKAGE SECTION or
(4) COMMUNICATION SECTION or
(5) REPORT SECTION
I = FILE SECTION found but is not followed by an FD, SD, RD statement.
J = Invalid contents in A-Margin.
K = PROCEDURE DIVISION not found or not in required location.
L = Unbalanced quotes or missing sentence delimiters have caused composition overset and loss of data - Data Division.
M = Composition overset caused by over-length procedure (paragraph).

Composition Restrictions

(1)The Procedure Division is recomposed paragraph by paragraph (meaning named procedure). In the event a paragraph exceeds 10,000 characters it must be broken by the insertion of dummy paragraph names that can later be removed from the output program. 10,000 characters averages about 180 statements.

(2)No character string other than a literal or on a comment statement may If such is encountered REWRITE will go into a loop can be aborted by depressing the ESCAPE key. This will prove necessary if a loop occurs because of a missing quote symbol or other coding violation.

LAUREL COBOL MAINTENANCE SUPPORT SYSTEM
INDENT CONTROL TABLE STRUCTURE

```
01 INDENT-RECORD.
    02 IR-PHRASE          PIC X(20).
    02 IR-CTL-CODE        PIC X.
    02 FILLER             PIC X.
    02 IR-LENGTH          PIC 99.
```

PHRASE may contain one or more words beginning in the 2nd position. The 1st position must be blank (or contain an asterisk to be treated as a comment).

LENGTH must include the leading blank and the terminating delimiter (either a blank or a left parenthesis). Legal range: 03 - 20.

INDENT CONTROL CODES:

The System employs a maximum of five 4-byte indents beginning at the B-Margin. At the end of a sentence (occurrance of a period) the current indent is reset to the B-Margin.

Sentences are built word by word until either the right margin is reached and the sentence is continued at the next indent or there is a match against one of the words or phrases in the INDENT1 table, in which case the current line is immediately written out and the appropriate indent is set according to the following control codes:

1 Permanent shift to next indent except at beginning of a new sentence.

2 Temporary shift of 2 bytes, insert word, write line, and set ELSE count.

3 Temporary shift to next indent and set table pointer to INDENT2.

4 Temporary shift to next indent.

5 Temporary shift to next indent and set table pointer to INDENT1.

6 Permanent shift to next indent except at beginning of new sentence and add 1 to IF-counter.

7 Temporary shift of 2 bytes, insert work/phrase, write line, and subtract 1 from IF-counter.

8 Begin new line at current indent.

9 Permanent shift to next indent (until end of sentence).

LAUREL COBOL MAINTENANCE SUPPORT SYSTEM

ADDITIONAL COMPOSITION RESTRICTIONS

It is possible to recompose a portion of a program as long as all division headers are present. This permits the recomposition of segments that may be extracted out into copy book members.

In all cases, there can be no missing periods or quote symbols.

There should be no EJECT statements in the A-Margin.

Sentences must be delimited by a period. Long sentences will be presented in multiple lines, with an indent for continued lines separate from the indents caused by reserved words in the two indent tables.

Comments embedded within long sentences (such as nested IFs) will cause the indent structure to be restarted following such comments as if that portion was the beginning of a new sentence. The program will compile OK, but its appearance will be degraded.

Comment lines should be preceeded and followed by spacer lines with asterists in column 7 because the REWRITE programs emit spacer lines only ahead of 01 level data names in the Data Division and ahead of paragraph names in the Procedure Division.

No Procedure Division-type statements (such as EXEC SQL) in the Data Division will be accepted. They must either be commented before and restored afterward or replaced by COPY statements.

In the event the source program has comment headers in the following form ..

```
*****************************************************************
*              P R O C E D U R E    D I V I S I O N             *
*****************************************************************
```

... they should immediately FOLLOW the relavant division heading, EXCEPT for the Identification Division. Any such comment for the Identification Division statement must PRECEDE it.

Any large block of comment statements detailing maintenance history and other general information narratives appearing at the beginning of a program must appear AFTER the REMARKS statement.