COBOL Conversion Guide

Release: 2
July 1, 1980

800-1204CC-02

WANG LABORATORIES, INC., ONE INDUSTRIAL AVENUE, LOWELL, MA 01851 • TEL: 617/459-5000, TWX 710-343-6769, TELEX 94-7421

Disclaimer of Warranties and Limitation of Liabilities

The staff of Wang Laboratories, Inc., has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Wang purchase, lease, or license agreement by which this software package was acquired, nor increases in any way Wang's liability to the customer. In no event shall Wang Laboratories, Inc., or its subsidiaries be liable for incidental or consequential damages in connection with or arising from the use of the software package, the accompanying manual, or any related materials.

NOTICE:

All Wang Program Products are licensed to customers in accordance with the terms and conditions of the Wang Laboratories, Inc. Standard Program Products License; no ownership of Wang Software is transferred and any use beyond the terms of the aforesaid License, without the written authorization of Wang Laboratories, Inc., is prohibited.

This manual replaces and obsoletes the first edition of the VS COBOL Conversion Guide (800-1204CC-01). For a list of updates made to this manual since the previous edition, see the "Summary of Changes".

WANG

WANG LABORATORIES, INC., ONE INDUSTRIAL AVENUE, LOWELL, MA 01851 • TEL: 617/459-5000, TWX 710-343-6769, TELEX 94-7421

PREFACE

This VS COBOL Conversion Guide is intended as an aid for the programmer already familiar with COBOL. The conversion process of an IBM card batch system to a VS interactive system is specifically addressed. However, many of the topics discussed in this guide are discussed in terms sufficiently general to apply to any non-Wang to Wang VS COBOL conversion.

While the specific examples provided in this guide do apply specifically to an IBM-to-Wang conversion, many of the topics discussed, such as syntax, device considerations and file organization apply to any non-Wang to Wang conversion.

Appendix A is a table of syntactical conversion considerations for the IBM-to-Wang VS conversion. Appendix B is a sample VS COBOL program demonstrating the use of figurative constants, FACs, DISPLAY AND READ, SHARED files and display picture record entries. Appendix C is a demonstration of IBM COBOL programs and the resulting VS COBOL programs after conversion. Appendix D is list of reserved words recognized by the Wang compiler, but not the IBM COBOL compiler, and vice versa. Appendix E lists the ASCII and EBCDIC collating sequences.

This guide should be used in conjunction with the following VS manuals:

Manual	Wang Order Number
VS COBOL Language Reference	800-1201CB
VS Utilities Reference	800-1303UT
VS Programmer's Introduction	800-1101PI
VS Procedure Language Reference	800-1205PR

SUMMARY OF CHANGES
FOR THE 2nd EDITION OF THE VS COBOL CONVERSION GUIDE

TOPIC	DESCRIPTION	PAGES
Documentation Reference	Update	1,2,6, 8,9,14, 21,25, 26

ACKNOWLEDGEMENT

COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.

No warranty, expressed or implied, is made by any contributor or by the CODASYL Programming Language Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

The authors and copyright holders of the copyrighted material used herein

> FLOW-MATIC (trademark of Sperry Rand Corporation), Programming for the UNIVAC I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F 28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis Honeywell

have specifically authorized the use of this material in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTORY CONVERSION CONCEPTS

1.1 THE VS COBOL ENVIRONMENT

VS COBOL is a programming language which is compatible with ANSI standard COBOL and IBM COBOL. The COBOL compiler is supported by the programming environment of the user-oriented VS operating system. The VS COBOL language, extended to support interactive processing, is supported by many VS utilities to edit, compile, execute, debug and link programs. Other utilities provide capabilities to generate formatted screen displays and create and maintain data files.

The following is a list of major VS utilities with a brief description of their capabilities.

EDITOR: allows the development and maintenance of source programs and integrates all the functions needed to create, edit, compile and execute programs.
(See VS Programmer's Introduction.)

EZCOBOL: provides default source and object file locations to the EDITOR and allows the operator to execute other VS utilities such as COPY and DISPLAY.
(See VS Programmer's Introduction.)

CONTROL: creates and maintains description files (control files) which contain the file characteristics and field attributes of any data file.
(See VS Utilities Reference.)

DATENTRY: creates and maintains data files conforming to the file characteristics defined by the CONTROL file.
(See VS Utilities Reference.)

LINKER: links two or more program modules into a single executable program. The VS does not require a link for a self-contained object file (a program file that does not issue calls to external subroutines). Self-contained programs can be compiled and executed directly.
(See VS Programmer's Introduction.)

This second list contains a brief description of utilities and user aids that are mentioned in the text as being useful for the conversion programmer.

CONVERTC: makes global text changes to COBOL source files by file or library.
(User aid information is available through the International Society of Wang Users (ISWU)).

COPY2200: copies data files from a Wang 2200 diskette to a VS library, automatically converting from 2200 format to VS format or copies VS files from hard disk to diskette, automatically reformatting them to 2200 format.
(See VS Utilities Reference.)

TAPECOPY: copies data from 9-track tape to VS disk, converting record format and file organization, if necessary. TAPECOPY can also copy files from disk to tape, from tape to tape or from disk to disk.
(See VS Utilities Reference.)

TRANSL: converts EBCDIC to ASCII or ASCII to EBCDIC.
(See VS Utilities Reference.)

3740 Diskette Compatibility software: converts a 3740 formatted diskette file to a Wang TC formatted file accessible by the Wang 2200 series. This utility is executable only on the Wang 2200 series with a 3740 compatible diskette drive. The information on the TC formatted diskette can be transferred to the VS by the COPY2200 utility.
(See 3740 Diskette Compatibility Software Manual.)

TCCOPY: enables the VS system to imitate the protocol of an IBM 2780/3780 and receive and transmit data between two systems over telecommunications lines.
(See Data Communications User's Guide.)

In addition to these utilities, the Wang VS Command Processor of the operating system enables the user to:

execute a program or procedure,
set default constant values,
examine device status and volume contents, and
rename, scratch and protect files.
(See VS Programmer's Introduction.)

The American National Standards Institute (ANSI) standardized COBOL after many vendors had developed their own version of the language. The standard was determined by taking the best options and processing methods of each of these major vendors. Most vendors now use this standard to measure their versions of the language. The standard, however, is always being revised for improvement as better COBOL mousetraps are built. In attempting to improve it and test other verbs and usages which could be incorporated into the standard, each vendor offers extensions to its particular version of the language. These extensions represent the major differences between VS COBOL and other vendors' COBOL.

Wang VS COBOL supports 1974 ANSI Level 1 nucleus elements with many Level 2 features. Most Wang extensions to the COBOL language provide for interactive processing which is not currently part of the standard. The following is a complete list of verbs supported by VS COBOL.

ACCEPT	DIVIDE	OPEN
ADD	ENTER	PERFORM
ALTER	EXIT	READ
CALL	EXIT PROGRAM	REWRITE
CLOSE	GO TO	SET
COMPUTE	IF	START
COPY	INSPECT	STOP
DELETE	MOVE	SUBTRACT
DISPLAY	MOVE WITH CONVERSION *	USE
DISPLAY AND READ *	MULTIPLY	WRITE

*Wang extensions

This guide is intended as a source of information for programmers converting from an ANSI standard batch environment to the VS interactive environment. The examples used, therefore, are based on IBM to VS conversions.

The four major topics of conversion discussed in this manual are:

1. Syntax - The orderly arrangement of words and punctuation to form clauses, sentences, paragraphs, sections and divisions.

2. Data file considerations

   a. Code set - The character code set used to represent data on the external media, and
   b. File organization - Permanent logical file structure.

3. Control language - the commands which control program execution.

4. Logical design - The orderly flow of operational functions performed on data.

1.2 SYNTACTICAL DIFFERENCES

IBM COBOL and VS COBOL have syntactical differences, most of which can be converted with a global edit. For example, the IBM compiler accepts the abbreviated name "ID DIVISION" for "IDENTIFICATION DIVISION". The VS compiler does not. The IBM compiler recognizes the single quotation mark (') ; the VS compiler recognizes the double quotation mark (").

The user aid CONVERTC is specifically designed to globally edit character strings in COBOL source code. CONVERTC can make the following types of modifications:

source text global editing,
comment out specified lines of source text, and
add new source text to lines.

For a complete list of syntax differences between IBM COBOL and VS COBOL, refer to Appendix A.

1.3 DATA FILE CONSIDERATIONS

Before data files can be read and processed by a VS, they must be translated to machine-compatible code and format. Wang provides several utilities to copy, reorganize if necessary, and translate nonconforming files to an acceptable code and format.

The code set used by a computer system is transparent to the user. Both Wang and IBM provide a default code set. ANSI specifies ASCII (American Standard Code for Information Interchange) as the standard code set for use with COBOL. The VS code set is extended ASCII. The IBM code set is EBCDIC (Extended Binary Coded Decimal Interchange Code). Both are based on the 32-bit word. EBCDIC cannot be processed directly by VS hardware; therefore, the IBM data files must be translated to ASCII. The Wang VS utility TRANSL performs this conversion function.

The VS supports two file organizations, each with three access methods. The storage method used on the VS is virtual. Consecutive file organization (also known as sequential) where the logical file structure is composed of records identified by a predecessor-successor relationship, is supported. Indexed file organization, where each record of the logical file structure is identified by one or more keys, is also supported. Both consecutive and indexed files can be accessed sequentially, randomly or dynamically. Indexed files with sequential access method (similar to IBM ISAM files) are available.

FILE ORGANIZATION	ACCESS METHOD
Consecutive	Sequential Random Dynamic
Indexed	Sequential Random Dynamic

Figure 1-1. File Organization and Access Methods

1.4 CONTROL LANGUAGE MODIFICATIONS

The operator of a VS system can control program execution interactively at the workstation by invoking the Command Processor. Procedure language can also be used to perform these same functions, without constant operator intervention.

The Procedure language is functionally similar to IBM's Job Control Language (JCL) since it is used to write special routines which can execute system commands. Procedures are used to run a series of programs in a specified sequence. The operator is only required to execute the procedure. A special procedure may be written to be executed at logon time. This logon procedure can set default values for the duration of the logon session and execute specified programs.

RUN	-	executes a program or another procedure.
ENTER	-	provides run-time file information such as file, library and volume name. This verb can also be used to pass instream data to the executing program.
DISPLAY	-	allows the operator to modify default run-time file information.
SCRATCH	-	scratches a file or library.
PROTECT	-	protects a file or library.
RENAME	-	allows the operator to specify a new file name for an existing file.
SET	-	allows the user to specify default file information and print modes.
IF	-	controls conditional execution of procedure steps.
GOTO	-	performs forward branching within the procedure.
RETURN	-	terminates procedure execution.
LOGOFF	-	terminates procedure execution and logs the user off the system.

The above is a list of all of the Procedure language commands necessary and available to control program execution.

Instream data can be passed to an executing program either interactively or by the Procedure language. The programmer must define the data fields that will be passed to the program at run-time. The Procedure Division code should include an ACCEPT of these data fields. The name of the field to be accepted should not exceed eight characters and should contain no hyphens. The value for this field is supplied by the procedure.

If the program is executed interactively from the Command Processor, a prompt for the data field values is displayed. If the program is executed via a procedure, the ENTER verb provides the parameter value. If the data to be accepted by the program is not provided by the procedure, then a system prompt is issued to the screen. The following example demonstrates a sample program and procedure used to pass instream data to an executing program.

```
WORKING-STORAGE SECTION.
77  ACCPTFLD  PIC X.

PROCEDURE DIVISION.
    ACCEPT ACCPTFLD.
    IF ACCPTFLD = "0"
        THEN GO TO EXIT-CLOSE
        ELSE NEXT SENTENCE.
```

Sample Code

```
PROCEDURE
RUN DPLSAMPL IN DPLOBJ ON VOL444
ENTER ACCEPT ACCPTFLD=0
```

Procedure Language

Example 1-1. Sample Program Code and Procedure Language

For more information on the use of the Procedure language, refer to the VS Procedure Language Reference.

1.5 LOGIC DESIGN CHANGES

Interactive systems, by design, logically process data differently than batch systems. A sample card batch system goes through this cycle:

1. Entries are keypunched to create a card deck.
2. The deck is read onto disk or tape.
3. The disk or tape is used as input to program.
4. The program, which will sequentially process an accumulation of data, is executed.
5. Corrections are handled which were due to keypunch errors.

A sample online interactive system does not require the intermediary steps of first keying cards and then transferring the card information to tape or disk. The entry is keyed directly onto tape or disk. The VS programmer may design a system to accumulate all transactions onto disk or tape and then process this group of transactions (VS batch processing). The alternate system design is to interactively process/update each transaction as it is keyed (VS interactive processing).

The VS hardware configuration does not include any card-oriented devices. All data used by a program must be read from a disk or tape file or entered interactively from the workstation. Therefore the batch system, as enumerated above, must be redesigned to eliminate all card processing.

There are two approaches to this redesign:

1. Redesign the front-end programs that process card input. The main processing logic remains intact, but all card input is replaced. The new front-end could accumulate keyed transactions directly onto a disk file. This file can then be input to the rest of the system, design logic unchanged.

2. Incorporate interactive capability. This alternative requires a major redesign of the system logic. More lead development time is required, but the production processing time and methods would be totally interactive. Several VS utilities, such as EZFORMAT and DATA ENTRY, aid in the implementation of this approach.

The VS provides an interactive environment. Most programs being considered for conversion are based on a batch design. Batch processing does not necessarily infer card input, but it does require an accumulation of items of raw data. Interactive processing, during which each unit of data is processed immediately at execution time, is performed on-line.

CHAPTER 2
DEVICE AND DATA CONSIDERATIONS

Data files which are to be processed and read on the VS computer must conform to VS standards. There are several important considerations for the programmer who wishes to convert files from a non-Wang system to the Wang VS and insure data integrity. They include:

1. character code difference,

2. media compatibility, and

3. file assignment.

2.1 CHARACTER CODE CONVERSION

As noted in Chapter 1, IBM data files are represented in EBCDIC and must be converted to ASCII for use by the VS. The VS utility, TRANSL, described in the VS Utilities Reference, translates EBCDIC to ASCII. TRANSL may also be used to change the order of fields within the records of a file, delete existing fields, and insert new blank fields, without converting the character code set.

2.2 DEVICE ACCESSIBILITY

The storage medium of the data file must be accessible by VS equipment. Data files on IBM series 3740 diskette can be made accessible to the VS, if they are first transferred to a Wang formatted diskette. The VS can also emulate an IBM 2780/3780 remote job entry (RJE) terminal, and transmit and receive data from the host central processing unit over telecommunication lines. If the files to be converted are not on these devices, then they can be copied onto a 9-track tape, possibly by the IBM OS utility IEBGENER. The principle medium for data interchange between non-Wang and VS equipment is 9-track magnetic tape. Both 800 and 1600 bpi tape are supported.

Table 2-1. Device Conversion Utilities

System	Utility	Input	Output	Manual
VS	TAPECOPY	9-track tape 800 bpi 1600 bpi	9-track tape 800 bpi 1600 bpi	VS Utilities Reference
2200	3740 Diskette Compatibility Software	3740 formatted diskette	2200 formatted diskette	3740 Diskette Compatibility Software Reference
VS	COPY2200	2200 formatted diskette	VS formatted diskette	VS Utilities Reference
VS	TCCOPY	2780/3780 files	VS disk file	VS Data Communications User's Guide

3740 Diskette Files

If data files are on IBM series 3740 diskette, the Wang 2200 Series 3740 Diskette Compatibility Software Utility can convert the IBM 3740 file to a Wang 2200 TC formatted file. This 2200 file can then be used as input to the VS utility COPY2200 to create a converted VS formatted disk file. COPY2200 provides the mechanism of data exchange between the VS and the Wang 2200 series. The two-system process described below requires both Wang 2200 hardware and VS hardware.

Step 1.
System: Wang 2200 with a 2270A diskette drive.
Input: 3740 diskette file.
Execute: 3740 Diskette Compatibility Software.
Output: Wang TC formatted file.

Step 2.
System: Wang VS.
Input: Wang TC formatted diskette file.
Execute: VS utility COPY2200.
Output: converted VS formatted disk file.

9-track Tape Files

If the files are on 9-track tape, the VS utility TAPECOPY can be used to copy the file from tape to VS disk. The TAPECOPY utility also has an option to translate from EBCDIC to ASCII during the COPY processing.

All tape files are organized consecutively. If an indexed disk file is copied to tape, the indices are not transferred, and the data portion of the file is written as a consecutive file. The DEFINE DISKFILE option of TAPECOPY allows the operator to specify file organization. A consecutive tape file may be changed to an indexed file with no alternate keys by specifying "I" in the FILEORG field and supplying the length and position of the primary key. The user is also given the opportunity to change other file characteristics (record format, packing density, printclass, and compression).

2780/3780 Emulation

The Wang TCCOPY utility's main function is to enable emulation. It enables the VS workstation to emulate the RJE terminal. Data received by the VS system via this utility is stored on disk in VS blocked format. These files, if in EBCDIC, must be translated to ASCII before further processing by the VS.

Data files can also be transmitted by the VS system by executing this emulation utility.

Print Records

Printer characteristics should also be reviewed for compatibility.

VS print records do not reserve the first byte for a write control character. All bytes described in the print record file description are usable for character display. Spacing and line positioning of VS print records are controlled by a two-byte write control area which is automatically added to the beginning of the print record. These two bytes are not counted with the total number of bytes for the print record definition. A print file that will produce 132 printed characters is defined in the COBOL program as containing 132 characters.

As long as the programmer writes print records BEFORE or AFTER ADVANCING a number of lines or PAGE, there is no need to provide values for the control area. The values in the control area are provided by the compiler when either reserved word, BEFORE or AFTER, is recognized.

SYNTAX: WRITE record-name [FROM identifier-1]

```
[ {BEFORE} ADVANCING {identifier-2} [LINE ]
  {AFTER }           {integer     } [LINES]
                     {PAGE                }
                     {user-figurative-constant} ]
```

If the programmer needs to specify non-standard spacing, the record can be written BEFORE or AFTER ADVANCING a user-figurative-constant. The user-figurative-constant must be defined in the FIGURATIVE-CONSTANTS paragraph of the Environment Division as one byte.

If bit 0 of byte 1 is zero, then bits 1-7 of byte 2 indicate the number of lines to be skipped. If bit 0 of byte 1 is one, then byte 2 indicates either TOP OF FORM (HEX "01") or VERTICAL TAB (HEX "02").

The bits indicated in Figure 2-1 must be zeroes.

Figure 2-1. Print Record Control Bytes

The VS implementation of the write control allows either standard hands-off line/page advancing or programmer-controlled line/page advancing.

The maximum length for any VS print record is limited by the printer device used. Most VS printers support print lines of up to 132 characters. Some models of the matrix line printer support up to 160 characters per line and the daisy printer can support up to 158 characters per line.

2.3 FILE ASSIGNMENT

VS files must be described in the COBOL program and they must be assigned a system name. This is done through the SELECT and ASSIGN clauses. The file-name associated with the SELECT clause is the file-name as it is used throughout the program. The name associated with the ASSIGN clause is called the parameter-reference-name. This parameter-reference-name is used to keep track of system related information for the file, such as record type, file organization and blocksize. Any system prompts to the operator for information about the file use the parameter-reference-name. To complete the assignment of the file, it must be associated with a device-type. Valid devices are DISK, TAPE, PRINTER and DISPLAY. If no device is supplied, the default is DISK.

SYNTAX: SELECT file-name
ASSIGN TO "parameter-reference-name" [,"device-type"][,NODISPLAY]

The NODISPLAY option of the ASSIGN clause suppresses prompts to the operator for file information as long as that information is provided elsewhere (from a procedure, for example).

For those familiar with IBM operating systems, the system name assigned to the file in the Data Control Block (DCB) is similar to the VS parameter-reference-name assigned to the Unit File Block (UFB). The DCB and UFB contain the same type of file information, although the UFB contains more file characteristic information. IBM uses Job Control Language (JCL) to control program execution and provide run-time file information. IBM's DDName is equivalent to the VS parameter-reference-name. The cataloged data set name (dsn) provides the same information to the IBM operating system that the data file-name, library name and volume name provide to the VS operating system. With the VS, this run-time data file information can be provided in any of four ways:

1. by an operator response to a system prompt,

2. through a procedure,

3. by a literal compiled with the program in the VALUE OF clause, or

4. by a data-name compiled with the program in the VALUE OF clause.

The examples of program code and procedures on the next page exhibit these four methods.

Four Examples of Assignment of Run-Time Information

```
ASSIGN "PUTIN"

       FILENAME=TESTDATA     LIBRARY=DPLDATA     VOLUME=VOL001
```

Example 2-1. Thru an Operator Response to System Prompting

```
PROCEDURE
SET INLIB=DPLDATA, INVOL=VOL001, PROGLIB=DPLOBJ,
    PROGVOL=VOL001
RUN SAMPLE
ENTER PUTIN FILE=TESTDATA
```

Example 2-2. Thru a Procedure

```
FD  PUTIN
    ...
    VALUE OF FILENAME IS "TESTDATA"
             LIBRARY IS "DPLDATA"
             VOLUME IS "VOL001".
```

Example 2-3. Thru Literals in the VALUE OF Clause

```
FD  PUTIN
    ...
    VALUE OF FILENAME IS FILE-NAME
             LIBRARY IS LIB-NAME
             VOLUME IS VOL-NAME.
...
...
WORKING-STORAGE SECTION.
77  FILE-NAME   PIC X(8).
77  LIB-NAME    PIC X(8).
77  VOL-NAME    PIC X(6).
...
...
PROCEDURE DIVISION.
...
...
    MOVE "TESTDATA" TO FILE-NAME.
    MOVE "DPLDATA" TO LIB-NAME.
    MOVE "VOL001" TO VOL-NAME.
```

Example 2-4. Thru Data-names in the VALUE OF Clause

CHAPTER 2 REFERENCES:

Topic	Text	Order Number
Compression	VS COBOL Language Reference Manual	800-1201CB
COPY2200	VS Utilities Reference	800-1303UT
Emulation	VS Data Communications User's Guide	800-1302DC
3740 Diskette Compatibility Software	3470 Diskette Compatability Software Manual	700-4369
TRANSL	VS Utilities Reference	800-1303UT

CHAPTER 3
FILE ORGANIZATION AND ACCESS METHODS

3.1 FILE ORGANIZATION GUIDELINES

The VS manages, maintains, and supports files through system-provided code known as the Data Management System (DMS). DMS is independent of individual user programs and is, in effect, transparent to the user. The Data Management System is invoked by the user's program whenever that program opens, accesses, or closes a file. DMS keeps track of all file characteristics, such as file organization, record length, access method, record count, parameter-reference-name, blocksize and format.

The Data Management System is resident in one common area of main memory. All user programs access this one area for file control. DMS is used by all languages, not just COBOL.

3.1.1 File Organizations

IBM COBOL offers five types of file processing with associated access methods. They are referred to as sequential (QSAM), direct (BSAM, BDAM), relative (BSAM, BDAM), indexed (QISAM, BISAM) and virtual (VSAM). The VS offers comparable counterpart processing to each of these five techniques.

IBM's standard sequential file is organized in serial order. As each record is written, it is added to the end of the file. Records are read in the order in which they were written. The VS equivalent is a file with consecutive organization and SEQUENTIAL access. This consecutive file, like the IBM sequential file, is ordered as the records were originally written.

Direct file processing in IBM COBOL permits records to be accessed by the relative track address provided through the ACTUAL KEY clause. An ACTUAL KEY value is composed of a relative track number and an identifying key utilized for the record search. A direct file cannot be allocated more space after it is created. Due to differences in data storage on disk, the VS Data Management System does not offer a keyed processing technique that is strictly compatible with IBM's direct file processing. A comparable effect may be achieved by use of VS indexed files. The VS programmer can directly access any specified record by identifying a key value in the RECORD KEY clause. The RECORD KEY clause specifies the data-name of a field in the record and has a unique value for each record. VS indexed files are discussed below.

IBM's relative file processing accesses records based on their ascending relative position in the file. The relative record number is provided via the NOMINAL KEY clause. The relative record number must be assigned to the NOMINAL KEY data-name before executing a READ, WRITE, or REWRITE. Relative file records are arranged in the same order that they were originally written to the file. The first record has a relative record number of zero (0). Once a file is created, its space allocation cannot be changed. A comparable VS technique to relative file processing utilizes consecutive file organization with RANDOM access. All records stored in a consecutive file are uniquely identified by a relative record number which specifies the record's ordinal position in the file. The relative record number must be assigned to the RELATIVE KEY data-name before executing a READ, WRITE or REWRITE. The first logical record has a relative record number of one (1) and subsequent logical records have relative record numbers of 2,3,4....

Indexed file processing techniques offered by IBM and Wang are very comparable, although there are some structural and implementation differences. Both IBM and Wang offer indexed file organization which permits RANDOM and SEQUENTIAL access to the file based on a record key. Both use a RECORD KEY clause to define a data item within the record by which the record can be uniquely identified. The data item is defined as any fixed length item from 1 to 255 bytes in length. The records are ordered in ascending sequence based on the key values. (Remember that the IBM collating sequence is EBCDIC and the Wang collating sequence is ASCII; see Appendix E.) Both offer indexed records that may be accessed sequentially; in such case each successive record obtained has a key value greater than the key value of the preceding record. Indexed records may also be accessed randomly by naming a specific key value. RANDOM access allows the user to specify which record is needed.

Both the IBM access methods (QISAM and BISAM) and the VS Data Management System offer system maintenance of indexes. However, there are notable structural differences. IBM defines three storage areas for an indexed file -- prime, index and overflow. The VS allocates one storage area only. When a new record is added to an IBM file, the actual record could be placed in the overflow area with an index pointer to that area. The new record added to the VS file will be placed in physical order in the file. This is performed through a process known as block-splitting.

Space is allocated differently. The IBM programmer must calculate space requirements dependent upon the number of records in the file, and the area required for index and overflow areas. The programmer specifies the primary and secondary space parameters in JCL. IBM's indexed files must have total space allocated at create time, and the amount specified must account for future file growth. VS indexed files, when opened in OUTPUT mode, automatically calculate and allocate required space depending upon the estimated number of records provided by the programmer. If additional space is needed as the file expands, secondary space is allocated automatically by Data Management.

File reorganization is sometimes necessary to consolidate and optimize space usage and to improve degraded input/output performance. One reason to reorganize IBM files is improvement of input/output performance which has been reduced because of chaining from index to prime to overflow areas. VS reorganizations may be necessary to improve performance, but because of the technique used by DMS to allocate file space, they are not required as frequently as with the IBM chaining.

The IBM method of data set cataloguing permits retrieval based on data set name and disposition only. VS Data Management does not support cataloguing. VS files must be retrieved by file, library and volume name.

The notable implementation differences for indexed file processing between IBM access methods and the VS Data Management System concern the flexibility and utility of the record key. To randomly access an IBM record by identifying key value, the programmer must move the identifying value to a data-name storage field. This data-name is identified as a NOMINAL KEY. A NOMINAL KEY is required when an indexed file is accessed randomly. The VS system does not use a NOMINAL KEY. The identifying VS key value is moved to the primary key field in the record area. As the record is read, it is moved into the record area, overlaying the primary key identifying value which was supplied by the programmer.

IBM COBOL does not allow direct REWRITE of a record with an altered length. The existing record must be deleted and the altered-length record then written. VS COBOL does not face this implementation restriction; REWRITE of a record with an altered length is allowed.

VS indexed files may be accessed along alternate key paths in addition to the primary key path. Each file may be defined with one primary and up to sixteen (16) alternate keys. Records in a file do not have to be part of all alternate access paths. A file with implicitly redefined records at the 01 level may have different data-names for the alternate key field, depending on the record type. This alternate key must still occupy the same left-most byte position in each record. Only those records with the data-name specified in the ALTERNATE RECORD KEY clause are indexed along that alternate key path. Alternate keys need not be unique if so defined at file creation: duplicate alternate key values are allowed. In order to randomly read a file to retrieve a record with a duplicate key value, the identifying value is first moved to the alternate key field in the record area. Next, the READ is issued specifying the alternate key data-name in the KEY clause. The record obtained will have the specified alternate key value, and the lowest primary key value associated with that alternate key, if the alternate key is duplicated. A READ NEXT will obtain the next record on this alternate key path. This READ NEXT process can continue until the AT END condition is encountered, indicating that there are no more records on this path. The file status of each READ NEXT is "00" unless the alternate key has one or more duplicates. In this latter case, the file status returned is "02". The file status of "00" is returned when the last of the duplicate key values is read.

For example, consider an inventory parts file with indexed organization and a primary key of parts number and two alternate keys -- supplier code and usage code. Supplier code allows duplicates. If the application requires a list of all parts supplied by Supplier-X, the list produced by randomly accessing the file on alternate duplicate key would be in primary key order within alternate key order.

VSAM processing of indexed files, as implemented through IBM COBOL, with direct and sequential access to indexed and sequential files, is most like the file processing techniques offered by the VS Data Management System. VS indexed file organization adds additional features to the READ, START and REWRITE I-O statements (as mentioned above).

3.1.2 Record Organizations

Disk files, by default, are blocked in 2K physical records. Spanned records, logical records that are stored in two or more physical records, are not supported. Because of the blocking considerations and other factors, there are limitations on record length.

	Consecutive	Indexed
Fixed	2048	2040
Variable	2024	2024
Compressed	2024	2024

Figure 3-1. Maximum Record Lengths Depend on File Organization and Record Type.

3.1.3 Compression

The Data Management System supports a special feature for both indexed and consecutive files called data compression. A Wang extension to the RECORD CONTAINS clause is used to specify data compression. The general format is:

RECORD CONTAINS [integer-1 TO] integer-2 [COMPRESSED] CHARACTERS

When COMPRESSED is used, all characters of three or more duplicate and consecutive characters are compressed into two bytes which contain the number of occurrences of the character and the character itself. Every COMPRESSED file contains compression indicator bytes which note (1) whether or not the bytes following the indicator bytes are COMPRESSED and (2) the number of bytes following the indicator bytes and preceding the next indicator bytes.

COMPRESSED is only specified when a file is being opened for output and the user wants the file to be compressed. Compression is not provided by default. DMS automatically expands compressed records to original format when they are read. Thus, compression is completely transparent to the user program. Compressed records are stored as variable length records. Variable length consecutive records cannot be updated via a REWRITE; therefore, a file that is to be updated must not be defined with consecutive organization and compressed records. This combination, consecutive compressed, is useful for record keeping logs. For example, this file may be used to record all update information which was processed against an updatable file. Because of this special function, compressed consecutive files are called log files.

3.2 SHARING

Sharing allows multiple users access to the same information, be it program or data files. There are two levels of sharing in the VS Operating System: automatic program sharing controlled by the operating system, and file sharing controlled by the Data Management System.

3.2.1 File Sharing

File sharing is a useful, easily implemented feature which can be used by the conversion programmer to redesign front-end programs or an entire system. Since files can be shared, system design can include concurrent multiple transactions to the shared file. One data file may be shared by many users. Shared files can even be simultaneously accessed by users performing different tasks. For example, one operator may update a shared file while another user is running inquiries against the file.

The single-user access constraint, which the system undergoing conversion had before, is no longer a design limitation.

File sharing allows multiple users update access to the same data file. File sharing is controlled by the sharing task of DMS. This task completely controls access to all shared files. If two users simultaneously attempt to access the same record, the sharing task puts one user in a "wait" state until the first user releases the record. When the record is released, the "waiting" user is then granted free access to it. This "wait" is generally not noticeable to the user.

The sharing task:

1. permits simultaneous multi-user update access to an indexed file,

2. prioritizes all user requests and resolves multi-user conflicts, and

3. executes I-O mode commands for file access requests.

3.2.2 COBOL Implementation

To implement file sharing, the VS programmer need only identify the open mode as SHARED and realize the effects of holding and releasing records.

SYNTAX: OPEN SHARED file-name.

The sharing task of DMS controls users' access to files which are in use. This control is established by DMS, not by the program code.

The COBOL programmer should be familiar with the standard INPUT, OUTPUT, I-O, AND EXTEND MODES and their access restrictions.

INPUT: Multiple users have READ only access.

OUTPUT: One user at a time may, depending on file organization, WRITE to or START the file.

I-O: One user at a time may READ or modify the file.

EXTEND: One user may WRITE records to an existing consecutive file.

SHARED mode, a Wang extension, allows multiple users concurrent access to READ and modify a file.

While one user is altering a record, that record should be protected from other user requests. This protection is insured by specifying a HOLD when obtaining the record. A record should be held only during modification.

SYNTAX: READ file-name WITH HOLD

When the user has completed processing that record, it must be released to allow access by other users. A record may be released by (1) issuing another READ WITH HOLD for the file, (2) issuing a REWRITE or DELETE for the held record, (3) issuing a WRITE for the record, or (4) issuing a CLOSE for the file. A user may hold only one record at a time.

The sample code in Appendix B demonstrates the Wang extensions to COBOL which are required to implement shared mode processing.

CHAPTER 3 REFERENCES:

Topic	Text	Order Number
Automatic File Sharing	VS Operating System Services	800-11070S
Automatic Program Sharing	VS Operating System Services	800-11070S
Data Management System	VS Operating System Services	800-11070S
File Organization	VS Operating System Services	800-11070S
	VS COBOL Language Reference	800-1201CB
File Access	VS Operating System Services	800-11070S
	VS COBOL Language Reference	800-1201CB
Shared Mode COBOL Code	VS COBOL Language Reference	800-1201CB

CHAPTER 4
CODING FOR THE WORKSTATION

4.1 INTERACTIVE WORKSTATION OVERVIEW

The VS system is an interactive system in which the primary means of program and data entry is the interactive workstation terminal. The user "converses" with the system via keyed entries and the system communicates to the user via screen-formatted displays.

In COBOL the workstation is associated with a workstation file. The workstation file, just like any file, has a file description in the Data Division. It can be read from or written to. The records of a workstation file, called screen display records, are similar in function to record description entries. These properties are common to all COBOL files. The uniqueness of the interactive extension to COBOL is the manipulation of data, as demonstrated by the Procedure Division statement DISPLAY AND READ.

4.2 CODING FOR THE WORKSTATION

Coding for the workstation in COBOL is supported by Wang extensions to the language. These extensions and coding requirements are described as follows and demonstrated in the sample program in Appendix B.

4.2.1 Workstation File

The workstation file is described in the COBOL FILE-CONTROL paragraph and file description. This file is assigned to the device DISPLAY. Access mode must be RANDOM and file organization must be consecutive. This file contains 1924 bytes -- 24 rows times 80 characters per row yields 1920 byte positions, plus a four byte order area. This order area can be used to control cursor positioning. The file-name for the workstation file is used by the COBOL program when manipulating the screen record.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CRT ASSIGN TO "SCREEN" "DISPLAY"
           ORGANIZATION IS SEQUENTIAL
           ACCESS MODE IS RANDOM.
...
DATA DIVISION.
FD  CRT.
01  DISPLAY-REC      PIC X(1924).
```

Example 4-1. Sample Code for a Workstation File

4.2.2 Screen Definition

Once the file has been defined, the screen record formats can be defined. A screen record entry defines positioning of data fields on the screen, field lengths, field sources, validation ranges and other field characteristics. There are three clauses- VALUE, SOURCE and OBJECT- used in the screen display record entry and by the screen manipulation verb DISPLAY AND READ.

```
01  RECORD-NAME  USAGE IS DISPLAY-WS.
    05  DATA-NAME-1 PIC X(5)  ROW 4 COLUMN 9
            VALUE IS "TOTAL".
    05  DATA-NAME-2 PIC X(5) ROW 4 COLUMN 15
            SOURCE IS FIELD-NAME-1
            OBJECT IS FIELD-NAME-2.
```

Example 4-2. Sample Screen Display Record Entry

The VALUE clause specifies the literal exactly as it is to appear on the screen. The SOURCE clause specifies a data-name of a Data Division item to be displayed on the screen. The value in the storage area identified by this data-name will be displayed on the screen when the DISPLAY AND READ is issued. The VALUE and SOURCE clauses are used exclusively of one another.

The OBJECT clause indicates that the described field is modifiable. This clause also specifies a data-name of an item to receive data from the screen. This value will be supplied by the operator and then moved to the storage area identified by the OBJECT data-name.

4.2.3 DISPLAY AND READ

When the DISPLAY AND READ is executed, the screen record is displayed. The values displayed are provided via the SOURCE or VALUE clauses. The system then waits for the operator to examine, add or change values and press the ENTER or one of the PFKEYS. This response is then read. If fields have been modified, the new values are validated and stored in the area determined by the OBJECT clause. Thus, DISPLAY AND READ could be considered as a five-process step: MOVE, DISPLAY, READ, Verify, and MOVE.

4.3 ADDITIONAL WORKSTATION CODING FEATURES

Figurative Constants and FACs

Another Wang extension to COBOL allows the programmer to define figurative constants. This feature can be used in conjunction with screen field attribute characters (FACs). FACs control the display mode characteristics for all characters in a field in the display, and are represented by hexadecimal values, but they appear on the display as a space. Default FACs are provided. The FAC value provided depends on the clauses used when defining the display picture definition.

The programmer may override the default FAC values by naming a FAC value in the figurative constants paragraph and then moving that FAC value to the FAC field of any data item described in the display picture definition entry.

```
ENVIRONMENT DIVISION.
.
.
.
FIGURATIVE-CONSTANTS.
    BLINK-MOD-NUMERIC IS "90".
.
.
.
PROCEDURE DIVISION.
.
.
.
    MOVE BLINK-MOD-NUMERIC TO
        FAC OF DATA-NAME-2.
```

Example 4-3. Sample Figurative Constants Code

The FAC display characteristics are a combination of one from each of the four columns in Table 4-1.

Table 4-1. Field Attribute Characteristics

BRIGHT	MODIFY	UPPERCASE	UNDERLINED
DIM	PROTECT	NUMERIC	NOT UNDERLINED
BLINK		ALPHANUMERIC UPPER AND LOWER	
BLANK			

The default FAC for a field with an OBJECT clause is bright modifiable ALL, not underlined. This combination has a hex code value of "80".

The default FAC for a field with a SOURCE or VALUE clause and no OBJECT clause is dim, protected, ALL, not underlined. This combination has a hex code value of "8C".

4.4 ADDITIONAL UTILITIES TO AID IN WORKSTATION CODING

The following is a list of additional utilities which may be useful to the conversion programmer:

EZFORMAT: generates the source code necessary to reproduce a screen display format defined by the user.

The COBOL option of EZFORMAT creates a compilable source file which can be copied into the working-storage section of the COBOL program. The DATA ENTRY option creates an entire data entry program based on the users graphic description of the screen format and the field characteristics. (See VS Programmer's Introduction.)

DISPLAY: displays any file on the screen. It can translate records from ASCII to HEX and locate records or text strings. (See VS Utilities Reference.)

COPY: copies files from one location to another. (See VS Utilities Reference.)

PRINT: produces a hard copy of a print file. (See VS Utilities Reference.)

CHAPTER 4 REFERENCES:

Topic	Text	Order Number
DISPLAY AND READ	VS COBOL Language Reference	800-1201CB
Field Attribute Characters	VS COBOL Language Reference	800-1201CB
Screen Definition	VS COBOL Language Reference	800-1201CB
Workstation File	VS COBOL Language Reference	800-1201CB

APPENDIX A
SYNTAX CONVERSION FACTORS

TABLE A-1. IDENTIFICATION DIVISION AND OVERALL CONVERSION FACTORS

ITEM		SYNTAX AND CONVERSION REMARKS*
IDENTIFICATION DIVISION	IBM:	IDENTIFICATION DIVISION. ID DIVISION.
	WANG:	IDENTIFICATION DIVISION.
	REMARKS:	The abbreviated form is not accepted by the Wang compiler.
PROGRAM-ID	IBM:	PROGRAM-ID. program-name. PROGRAM-ID. 'program-name'.
	WANG:	PROGRAM-ID. program-name.
	REMARKS:	The program name, in quotes, is not accepted by the Wang compiler.
REMARKS	IBM:	REMARKS. reserved word
	WANG:	(not supported).
	REMARKS:	REMARKS is not a Wang COBOL reserved word. Comment lines can be used by specifying an asterisk (*) in column 7.
RESERVED WORDS	REMARKS:	See Appendix D for IBM reserved words not recognized by the Wang compiler, and Wang reserved words not recognized by the IBM compiler.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM	SYNTAX AND CONVERSION REMARKS*	
EJECT SKIP	IBM:	EJECT SKIP2 SKIP1 SKIP3
	WANG:	(not supported).
	REMARKS:	These statements which generate spacing for IBM-compiler listings are not supported by the Wang compiler. An EJECT to top-of-form, next page is generated by a (/) in column 7. A SKIP condition is created by writing blank lines or blank comment lines in the source code.
QUOTES	IBM:	' ' (single quotation marks)
	WANG:	" " (double quotation marks)
	REMARKS:	The Wang compiler requires double quotation marks.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

TABLE A-2. ENVIRONMENT DIVISION CONVERSION FACTORS

ITEM		SYNTAX AND CONVERSION REMARKS*
SOURCE AND OBJECT COMPUTER	IBM:	SOURCE-COMPUTER. IBM... OBJECT-COMPUTER. IBM...
	WANG:	SOURCE-COMPUTER. WANG-VS. OBJECT-COMPUTER. WANG-VS.
OBJECT COMPUTER	IBM:	OBJECT-COMPUTER. ... [SEGMENT-LIMIT is priority-number].
	WANG:	OBJECT-COMPUTER. ... [PROGRAM COLLATING SEQUENCE is alphabet-name].
	REMARKS:	Wang supports the ANSI standard clause of PROGRAM COLLATING as a comment only. The VS collating sequence is ASCII. The IBM extension of SEGMENT-LIMIT is not supported.
FIGURATIVE CONSTANTS (system-defined)	IBM:	ZERO(E)(S) LOW-VALUE(S) SPACE(S) QUOTE(S) HIGH-VALUE(S) ALL
	WANG:	ZERO(E)(S) LOW-VALUE(S) SPACE(S) QUOTE(S) HIGH-VALUE(S)
	REMARKS:	The reserved word, ALL, is not supported as a system-defined figurative constant.
FIGURATIVE CONSTANTS (user-implemented)	IBM:	(not available).
	WANG:	user-defined.
	REMARKS:	The concept of implementor-defined figurative constants is mentioned in Chapter 4.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
SELECT CLAUSE	IBM:	SELECT [OPTIONAL] file-name Optional is a keyword required for input files that are not necessarily present each time the program is executed.
	WANG:	SELECT file-name
	REMARKS:	The IBM keyword OPTIONAL is omitted.
ASSIGN CLAUSE	IBM:	ASSIGN TO [integer-1] system-name-1 [system-name-2] ... [FOR MULTIPLE {UNIT}] {REEL}
	WANG:	ASSIGN TO "parameter-reference-name" [,"device-type"] [,NODISPLAY]
	REMARKS:	Parameter-reference-name can be a maximum of eight characters. The first character must be alphabetic and the rest must be alphanumeric. The parameter-reference-name associates the FD with run-time file specifications supplied via the command language. The parameter-reference-name is functionally equivalent to IBM's DD name in JCL. Device-type must be "DISPLAY", "DISK", "TAPE" or "PRINTER". The default value is "DISK". MULTIPLE REEL/UNIT is related to tapes and is not supported on the disk-based Wang VS. NODISPLAY is a VS option that suppresses system prompts for getparm information.
RESERVE	IBM:	{NO } [AREA] RESERVE {integer} ALTERNATE [AREAS]
	WANG:	[AREA] RESERVE integer [AREAS]
	REMARKS:	The NO option, used by IBM to control buffering, is not required by the WANG system since Data Management and the I-O processor handle these buffer requests. The RESERVE clause can be used to optimize buffers with buffer pooling. The RESERVE clause when used in conjunction with the SAME AREA clause indicates that the specified indexed files are sharing a buffer.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
FILE LIMIT	IBM:	{FILE-LIMIT IS } {data-name-1} {data-name-2} {FILE-LIMITS ARE} {lit-1 } THRU {lit-2 } [{data-name-3} {data-name-4}] ... [{lit-3 } THRU {lit-4 }]
	WANG:	(not supported).
	REMARKS:	This clause serves as documentation only on the IBM system.
PROCESSING MODE	IBM:	PROCESSING MODE IS SEQUENTIAL
	WANG:	(not supported).
	REMARKS:	This clause serves as documentation only on the IBM system.
ACTUAL KEY	IBM:	ACTUAL KEY IS data-name
	WANG:	RECORD KEY IS data-name
	REMARKS:	The ACTUAL KEY is used when processing a direct file. The ACTUAL KEY contains a track identifier and a record identifier. There is no one-to-one corresponding use of the ACTUAL KEY in VS COBOL because there is no direct file processing. However, the VS Data Management System offers indexed file organization which permits retrieval of a specific record by an identifying key value. The key field is identified in the RECORD KEY clause.
NOMINAL KEY	IBM:	NOMINAL KEY IS data-name This clause is required when an indexed or relative file is accessed randomly or when an indexed file is accessed sequentially using the START function.
	WANG:	(not supported).
	REMARKS:	Random access of an indexed file is controlled by the programmer. The desired record is accessed by placing the value of its record key in the record key data item in the record area.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
TRACK-AREA	IBM:	TRACK-AREA IS {data-name} {integer } CHARACTERS This clause is optional when adding records to an indexed file in the random access mode.
	WANG:	(not supported).
	REMARKS:	This clause is unnecessary since Data Management and the I-O processor supervise track allocation functions.
TRACK-LIMIT	IBM:	TRACK-LIMIT IS integer [TRACK] [TRACKS] This clause indicates the relative number of the last track to be initialized for the creation of files with direct organization.
	WANG:	(not supported).
	REMARKS:	This clause is unnecessary since Data Management and the I-O processor supervise track allocation functions.
CURSOR POSITION	IBM:	(not supported).
	WANG:	CURSOR POSITION is data-name
	REMARKS:	This clause is optional but only applicable when device-type is "DISPLAY" and a workstation file is being referenced. The data-name value, provided by the operating system, indicates current row and column position after every I-O operation on the workstation file.
BUFFER SIZE	IBM:	(not supported).
	WANG:	BUFFER SIZE IS integer BLOCKS
	REMARKS:	This optional clause is valid for sequential files only. It is another optimization tool used to create large buffers.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
PF KEY	IBM:	(not supported).
	WANG:	PFKEY is data-name
	REMARKS:	This optional clause is valid for workstation files only. It identifies a data area that receives the numeric value of the entered PFKEY following execution of the command DISPLAY AND READ.
PASSWORD	IBM:	PASSWORD IS data-name This optional clause defines a working-storage data item that controls object-time access to the file.
	WANG:	(not supported).
	REMARKS:	Files are protected by limiting user's access. Files can only be accessed by users who are authorized for that file's protection class. The protection class is defined when the file is created.
		RERUN ON system-name EVERY {integer RECORDS}
RERUN	IBM:	RERUN ON system-name {[END OF] {REEL} } OF file-name {UNIT}
	WANG:	[{ file name-1 }] RERUN [ON {implementor-name}] EVERY integer RECORDS OF file-name-2]
	REMARKS:	RERUN is treated as a comment by the Wang compiler.
MULTIPLE FILE	IBM:	MULTIPLE FILE TAPE CONTAINS file-name-1 [POSITION integer-1] [file-name-2 [POSITION integer-2]] ...
	WANG:	(not supported).
	REMARKS:	This clause is not supported by this disk-based system.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
APPLY	IBM:	APPLY WRITE-ONLY ON file-name-1 [file-name-2] ... APPLY CORE-INDEX ON file-name-1 [file-name-2] ... APPLY RECORD-OVERFLOW ON file-name-1 [file-name-2] ... APPLY REORG-CRITERIA TO data-name ON file-name
	WANG:	(not supported).
	REMARKS:	The APPLY clause is an IBM-implemented method of forcing optimal buffer and device space. Optimum space allocation is "automatically" handled by the VS Data Management System.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

TABLE A-3. DATA DIVISION CONVERSION FACTORS

ITEM		SYNTAX AND CONVERSION REMARKS*
LEVEL INDICATORS	IBM:	FD file description SD sort file description CD communication description RD report description
	WANG:	FD file description
	REMARKS:	Sorting on the Wang VS is supported by a Wang utility, SORT. This utility can be run directly or called dynamically. Direct COBOL SORT and REPORT WRITER features are not currently available. Reports can be produced by using the WRITE statement or the REPORT utility. A communication description is not used, since interactive workstation I/O functions are handled directly.
LEVEL NUMBERS	IBM:	Levels 01-49, 66, 77 and 88. Level 66 is used with RENAMES clause.
	WANG:	Levels 01-49, 77 and 88.
	REMARKS:	Special level number 66 is not currently supported.
DATA REFERENCE	IBM:	QUALIFICATION and CORRESPONDING
	WANG:	(not supported).
	REMARKS:	Currently, all data-names, paragraph-names and section-names must be unique (planned product enhancement).

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM	SYNTAX AND CONVERSION REMARKS*
BLOCK	IBM: BLOCK CONTAINS [integer-1 TO] integer-2 {CHARACTERS / RECORDS}
	WANG: BLOCK CONTAINS integer {RECORDS / CHARACTERS}
	REMARKS: IBM systems require this clause unless the physical record length equals the logical record length. Blocking is automatic for disk files with the Wang VS. Disk default blocking factor is 2K. Tape file blocking information is provided via this clause. If this clause is used, it must be in conjunction with a corresponding BUFFER SIZE clause.
RECORD	IBM: RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS
	WANG: RECORD CONTAINS [integer-1 TO] integer-2 [COMPRESSED] CHARACTERS
	REMARKS: Compression is a space-saving option. When specified, all character strings of 3 or more repeated characters are compressed into 2 bytes.
RECORDING MODE	IBM: RECORDING MODE IS mode If this clause is not coded, the IBM compiler will scan each record description entry to determine the correct mode.
	WANG: (not supported).
	REMARKS: The Wang compiler does not recognize this clause as it automatically scans each record description entry to determine the correct mode.
LABEL RECORDS	IBM: {RECORD IS / LABEL {RECORDS ARE}} {OMITTED / STANDARD / {data-name-1 [data-name-2]}} [TOTALING AREA IS data-name-3 TOTALED AREA IS data-name-4]
	WANG: {RECORD IS / LABEL {RECORDS ARE}} {STANDARD / OMITTED}
	REMARKS: This clause is required for every FD defined to the IBM system. The Wang compiler assumes standard label for all disk files. Tape labels may be either standard or omitted.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
VALUE OF	IBM:	VALUE OF data-name-1 IS {lit-1 / data-name-2} [data-name-3 IS {lit-2 / data-name-4}]...
	WANG:	VALUE OF [{FILENAME / SPACE / {data-name} / LIBRARY / VOLUME / POSITION / INDEX AREA / DATA AREA} IS {lit / data-name}]
	REMARKS:	This clause is documentation only on the IBM system; optional on the Wang VS. If the information specifying filename, space, volume, etc., is not provided via this clause, the operating system will prompt the operator for it.
CODE-SET	IBM:	(not supported).
	WANG:	CODE-SET IS alphabet-name
	REMARKS:	The CODE-SET clause is treated as a comment. All Wang VS data is stored in ASCII format.
PICTURE	IBM:	{PIC / PICTURE} IS character-string
	WANG:	{PIC / PICTURE} IS character-string
	REMARKS:	There are no syntax changes, but some symbols used in the picture clause require mention.
a.) alphabetic	IBM:	symbol A only
	WANG:	symbol A and B
	REMARKS:	Symbol B represents a space.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
b.) alphanumeric edited	IBM:	symbols A X 9 B 0
	WANG:	symbols A X 9 B 0 /
	REMARKS:	The stroke (/) represents a position where the character "/" is to be inserted.
c.) numeric edited	IBM:	symbols B P V Z 0 9 , . * + - CR DB $
	WANG:	symbols B / P V Z 0 9 , . * + - CR DB $
	REMARKS:	The stroke (/) represents a position where the character "/" is to be inserted.
SIGN	IBM:	{LEADING } SIGN IS {TRAILING} [SEPARATE CHARACTER]
	WANG:	{LEADING } SIGN IS {TRAILING} [SEPARATE CHARACTER]
	REMARKS:	If SEPARATE CHARACTER is not specified, the IBM default is to include the sign within the last byte; the Wang default adds another byte for the sign.
SYNC	IBM:	{SYNC } [LEFT] {SYNCHRONIZED} [RIGHT] Synchronization is performed, whether specified or not, for fields to be used in computation.
	WANG:	{SYNC } [LEFT] {SYNCHRONIZED} [RIGHT]
	REMARKS:	This clause is treated as a comment since the Wang compiler will, by default, synchronize all 01 levels to full-word boundaries. Level 77s are not synchronized.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
USAGE	IBM:	USAGE IS {DISPLAY } {COMPUTATIONAL } {COMP } {COMPUTATIONAL-1} {COMP-1 } {COMPUTATIONAL-2} {COMP-2 } {COMPUTATIONAL-3} {COMP-3 } {DISPLAY-ST } {INDEX } {COMPUTATIONAL-4} {COMP-4 }
	WANG:	USAGE IS {BINARY } {COMPUTATIONAL} {COMP } {DISPLAY } {DISPLAY-WS } {INDEX }
	REMARKS:	All COMP-3 and COMPUTATIONAL-3 usages may be changed to COMP or COMPUTATIONAL usage. All COMP or COMPUTATIONAL items of 32K or less may be changed to BINARY. Binary items are currently limited to two bytes (full word and double word support is planned). There is no corresponding Wang usage for COMP-1, COMP-2, and COMP-4, internal floating-point items. DISPLAY-WS is used when defining workstation screen records.
VALUE	IBM:	VALUE is lit {VALUE IS } lit-1 [THRU lit-2] [lit-3 [THRU lit-4]] {VALUES ARE}
	WANG:	{lit } VALUE IS {user-figurative-constant} {VALUE IS } {lit-1 } [{ THRU } { lit-2 }] {VALUE are } {user-figurative-constant} [{THROUGH} {user-figurative-constant-2}]
	REMARKS:	The VALUE can be defined as a user-figurative-constant.
RENAMES	IBM:	66 data-name RENAMES data-name-2 [THRU data-name-3]
	WANG:	(not supported).
	REMARKS:	RENAMES is functionally similar to REDEFINES which is accepted by the Wang compiler.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
OCCURS	IBM:	FORMAT 1: OCCURS integer-1 TIMES [{ASCENDING } / {DESCENDING} KEY IS data-name-1 [data-name-3] ...] [INDEXED BY ind-nm-1 [ind-nm-2]...] FORMAT 2: OCCURS integer-1 TO integer-2 TIMES [DEPENDING ON data-name-1] [{ASCENDING } / {DESCENDING} KEY IS data-name-2 [data-name-3] ...]... [INDEXED BY ind-nm-1 [ind-nm-2] ...] FORMAT 3: OCCURS integer-2 TIMES [DEPENDING ON data-name-1) [{ASCENDING } / {DESCENDING} KEY IS data-name-2 [data-name-3] ... [INDEXED BY ind-nm-1 [ind-nm-2] ...]
	WANG:	OCCURS integer-1 TIMES [INDEXED BY ind-nm-1 [,ind-nm-2] ...]
	REMARKS:	ANS COBOL LEVEL 1 FORMAT 1 is supported, but the LEVEL 2 and IBM extensions -- ASCENDING, DESCENDING, and DEPENDING ON -- are not.
SEGMENTATION	IBM:	COBOL segmentation in the form of static overlays is available with the IBM operating systems (DS/MFT, OS/MVT, DOS), although it is little used on the larger machines which accommodate paging.
	WANG:	COBOL-defined segmentation is not available with the Wang VS operating system.
	REMARKS:	The same effect of COBOL-defined segmentation is achieved automatically by the Wang VS virtual memory system.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

TABLE A-4. PROCEDURE DIVISION CONVERSION FACTORS

ITEM		SYNTAX AND CONVERSION REMARKS*
CORRESPONDING OPTION	IBM:	{CORR } ADD {CORRESPONDING} id-1 TO id-2 [ROUNDED] [ON SIZE ERROR imp-st] {CORR } SUBTRACT {CORRESPONDING} id-1 FROM id-2 [ROUNDED] [ON SIZE ERROR imp-st] {CORR } MOVE {CORRESPONDING} id-1 TO id-2
	WANG:	(not supported).
	REMARKS:	As noted in Table A-2, all data-names must be unique; therefore, the CORRESPONDING option is not available (planned product enhancement).
ADD/SUBTRACT MULTIPLE RESULTS	IBM:	{id-1 } [id-2] ADD {lit-1} [lit-2] ... TO id-m [ROUNDED] [id-n [ROUNDED] ... [ON SIZE ERROR imp-st] {id-1 } [id-2] SUBTRACT {lit-1} [lit-2] ... FROM id-m [ROUNDED] [id-n [ROUNDED]] ... [ON SIZE ERROR imp-st]
	WANG:	{id-1 } [id-2] ADD {lit-1} [lit-2] ... TO id-m [ROUNDED] [ON SIZE ERROR imp-st] {id-1 } [id-2] SUBTRACT {lit-1} [lit-2] ... FROM id-m [ROUNDED] [ON SIZE ERROR imp-st]
	REMARKS:	Values may only be added to/subtracted from one identifier, not to/from a series of identifiers. The Wang format follows the ANS COBOL LEVEL 1 standard.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM	SYNTAX AND CONVERSION REMARKS*
PERFORM (Format 4)	
IBM:	PERFORM procedure-name-1 [THRU procedure-name-2] VARYING {ind-nm-1} {id-1 } FROM {ind-nm-2} {lit-2 } {id-2 }
	{lit-3} {ind-nm-4} {ind-nm-5} BY {id-3 } UNTIL condition-1 [AFTER {id-4 } FROM {lit-5 } {id-5 }
	{lit-6} {ind-nm-7} {ind-nm-8} BY {id-6 } UNTIL condition-2 [AFTER {id-7 } FROM {id-8 } BY {lit-9} {id-9 } UNTIL condition-3]]
WANG:	PERFORM procedure-nm-1 [{THROUGH} procedure-nm-2] VARYING {ind-nm-1} {THRU } {id-1 }
	{ind-nm-2} {id-2 } {id-3 } FROM {lit-1 } BY {lit-2} UNTIL condition-1
REMARKS:	The command format that allows for testing of varying multiple conditions is not yet accepted by the Wang compiler. This function may be performed by coding multiple PERFORMs that will only be executed if a coded conditional test is passed (planned product enchancement).
	The Wang compiler will accept either form of THROUGH -- THRU or THROUGH.
MOVE WITH CONVERSION	
IBM:	(not supported).
WANG:	MOVE WITH CONVERSION data-name-1 TO data-name-2 [ON ERROR imp-st]
REMARKS:	This Wang-implemented verb will move free-form ASCII character strings containing a character representation of a number to a field with binary, computational or display usage. The move will decimal-align. Data-name-1 describes either an alphanumeric field or a numeric-edited field. The allowable edit characters include "+", "-", and the non-embedded blank. Data-name-2 describes a data item with a usage of binary, computational or display.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM	SYNTAX AND CONVERSION REMARKS*

EXAMINE/INSPECT

```
                                        {UNTIL FIRST}
                                        {ALL        }
IBM:     EXAMINE id-1 TALLYING {LEADING    } lit-1 [REPLACING BY lit-2]

                                        {ALL        }
                                        {LEADING    }
                                        {FIRST      }
         EXAMINE id-1 REPLACING {UNTIL FIRST} lit-1 BY  lit-2

WANG:    INSPECT id-1 TALLYING

        {       {{ALL     }   {id-3 } } [                        ]}
        {       {{LEADING}    {lit-1} } [{BEFORE}          {id-4 }]}
        {id-2 FOR{                    } [{AFTER } INITIAL {lit-2}]}
        {       { CHARACTERS          } [                        ]}

         INSPECT id-1 REPLACING

        {             {id-6 } [{BEFORE}          {id-7 }]       }
        {CHARACTERS BY {lit-4} [{AFTER } INITIAL {lit-5}]       }
        {{ALL     }                                             }
        {{LEADING} {id-5 }    {id-6 } [{BEFORE}          {id-7 }]}
        {{FIRST   } {lit-3} BY {lit-4} [{AFTER } INITIAL {lit-5}]}
```

REMARKS: EXAMINE is IBM-implemented. INSPECT is standard ANS COBOL used for counting and/or replacing an individual character. A third format is available which will tally and replace with one statement.

For example, if FIELD-ONE contains a value of bbb690, the IBM statement

///

EXAMINE FIELD-ONE TALLYING ALL SPACES REPLACING BY ZEROES

will yield a new value for FIELD-ONE of 000690 and the special register TALLY will contain the value 00003.

The VS and ANSI approach is the INSPECT statement:

INSPECT FIELD-ONE TALLYING COUNTER FOR ALL SPACES REPLACING BY ZEROES

which will yield a new value for FIELD-ONE of 000690 and COUNTER will contain a value of 3. COUNTER must be defined in the Data Division and initialized before each use or the COUNTER values will be accumulated.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
TRANSFORM	IBM:	TRANSFORM id-3 CHARACTERS FROM {id-1 / nonnumeric-literal-1 / figurative-constant-1} TO {id-2 / nonnumeric-literal-2 / figurative-constant-2}
	WANG:	(not supported).
	REMARKS:	TRANSFORM, IBM-implemented, can alter a string of characters. This verb may be replaced with multiple INSPECTs. For example, the IBM statement TRANSFORM FIELD-TWO CHARACTERS FROM "ABC" TO "XYZ" where FIELD-TWO has an initial value of BQARABCC, will result in a new value in FIELD-TWO of YQXRXYZZ. The VS implementation takes the form: INSPECT FIELD-TWO REPLACING ALL "A" BY "X". INSPECT FIELD-TWO REPLACING ALL "B" BY "Y". INSPECT FIELD-TWO REPLACING ALL "C" BY "Z". The FIELD-TWO value changes after each INSPECT from the original BQARABCC to BQXRXBCC, then to YQXRXYCC, and then to YQXRXYZZ, the desired result.
OPEN	IBM:	OPEN [INPUT {file-name-1 [WITH NO REWIND]} ...] [OUTPUT {file-name-2 [WITH NO REWIND]} ...] [I-O {file-name-3 } ...] [EXTEND {file-name-4 } ...]
	WANG:	OPEN {INPUT file-name-1 [file-name-6] ... } {OUTPUT file-name-2 [file-name-7] ... } {I-O file-name-e [file-name-8] ... } {EXTEND file-name-4 [file-name-9]}... } {SHARED file-name-5 [file-name-10]}... }
	REMARKS:	REVERSED and NO REWIND are used with tape files. They are not yet implemented for the OPEN statement tape support. SHARED MODE is discussed in Chapter 3.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
START	IBM:	START file-name [INVALID KEY imp-st] START file-name USING KEY data-name {EQUAL TO / =} id [INVALID KEY imp-st]
	WANG:	START file-name START file-name [KEY [data-name-1]] [/ {IS EQUAL TO / IS = / IS GREATER THAN / IS > / IS NOT LESS THAN / IS NOT <}] [data-name-2] [INVALID KEY imp-st]
	REMARKS:	IBM's FORMAT 2 requires data-name to be the same as the data-name specified in the RECORD KEY clause. Wang's FORMAT 2 requires data-name-1, if used, to be an alternate key.
SEEK	IBM:	SEEK file-name RECORD
	WANG:	(not supported).
	REMARKS:	This command serves as documentation only on the IBM system.
READ	IBM:	READ file-name [NEXT] RECORD [INTO id] {AT END / INVALID KEY} imp-st
	WANG:	READ file-name [NEXT] RECORD [WITH HOLD / MODIFIABLE / ALTERED] [INTO id] [{AT END / {INVALID KEY}} imp-st] READ file-name RECORD [WITH HOLD] [INTO id] [KEY IS data-name] [INVALID KEY imp-st]
	REMARKS:	These formats include all combinations for indexed and sequential files. For an explanation of WITH HOLD, MODIFIABLE and ALTERED, refer to Chapter 3 and the Wang VS COBOL Language Reference Manual.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
DISPLAY	IBM:	DISPLAY {id-1 } [id-2] ... [UPON {CONSOLE }] {lit-1} [lit-2] {SYSPUNCH }] {SYSOUT }] {mnemonic-name}
	WANG:	DISPLAY {id-1 } [id-2] {lit-1} [lit-2] ...
	REMARKS:	The choice of display medium is not available on the VS. All displays are directed toward the workstation screen. If a hard copy is needed, the screen display can be printed.
COPY	IBM:	COPY library-name [SUPPRESS] [REPLACING word-1 by{word-2} [word-3 BY {word-4}]...]...] {lit-1 } {lit-1 }
		{id-2 } {id-1 }
	WANG:	COPY file-name {IN} library-name. {OF}
	REMARKS:	The COPY statement incorporates text into a COBOL source program. Wang does not support the COPY ... REPLACING option.
WRITE	IBM:	WRITE record-name [FROM id-1] [{BEFORE} {integer-2 LINES}] {EOP } imp-st] {AFTER } ADVANCING {integer LINES }] [AT {END-OF-PAGE} {mnemonic-name }]
		WRITE record-name [FROM id-1] AFTER POSITIONING {id-2 } {integer} LINES [AT {END-OF-PAGE} {EOP } imp-st]
		WRITE record-name [FROM id-1] INVALID KEY imp-st
	WANG:	WRITE record-name [FROM id-1] [{BEFORE} ADVANCING {{id-2 } [LINE] }] [{AFTER } {{integer} [LINES] }] {{PAGE } }] {{user-figurative constant}
		WRITE record-name [FROM id-1] [INVALID KEY imp-st]
	REMARKS:	Wang printers, depending upon the model in use, can accomodate up to 158 characters per print line. Most VS printers support 132 character print lines. A file description of such a print file describes 132 usable characters. The first character need not be defined as unused filler to hold space for a write carriage control character. A WRITE, issued against a SHARED sequential file, releases the record to the buffer but does not write it to the disk until the buffer is filled. To prevent possible loss of these buffer-stored SHARED sequential records, a WRITE-THRU option is available which forces a write of the record as soon as the WRITE command is issued.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
REWRITE	IBM:	REWRITE record-name [FROM id] [INVALID KEY imp-st]
	WANG:	REWRITE record-name [FROM id] [INVALID KEY imp-st]
		REWRITE record-name [FROM id] AFTER [ALARM] [SETTING CURSOR COLUMN {id-1 } {ROW } {id-2 }] [{integer-1} {LINE} {integer-2}] [ROLL DOWN] [ROLL UP] [ERASE PROTECT] [ERASE MODIFY]
	REMARKS:	All additional clauses provided by the Wang format relate to screen-positioning control of the workstation files. Further discussion is in Chapter 4.
CLOSE	IBM:	CLOSE file-name-1 [REEL] [UNIT] [WITH {NO REWIND} {LOCK }] [file-name-2 [REEL] [UNIT] [WITH {NO REWIND} {LOCK }]]... CLOSE file-name-1 [WITH {NO REWIND} {LOCK } {DISP }] [file-name-2 [WITH {NO REWIND} {LOCK } {DISP }]]... CLOSE file-name-1 {REEL} {UNIT} [WITH {NO REWIND } {LOCK } {POSITIONING}] [file-name-2 {REEL} {UNIT} [WITH {NO REWIND } {LOCK } {POSITIONING}]]
	WANG:	CLOSE file-name-1 [WITH LOCK] {file-name-2} [WITH LOCK].... CLOSE file-name-1 [{REEL} [WITH NO REWIND]] [{UNIT} [FOR REMOVAL]] [file-name-2 [{REEL} [WITH NO REWIND]] [{UNIT} [FOR REMOVAL]]] [{NO REWIND}] [{NO REWIND}] [WITH {LOCK }] [WITH {LOCK }]
	REMARKS:	The DISP and POSITIONING options in the IBM format, not available with the Wang format, could be used to specify positioning of the current tape volume. These options are not supported for Wang tape processing.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
ACCEPT		{SYSIN }
		{CONSOLE }
	IBM:	ACCEPT identifier [FROM {mnemonic-name}]
		Each data item to be accepted cannot exceed 80 characters.
	WANG:	ACCEPT id-1 [id-2] ...
		The workstation operator is prompted for identifier-1 if it is not available from procedure specifications.
	REMARKS:	A maximum of 64 characters per data item may be made available to the Wang system using the ACCEPT command. Up to 16 data items may be accepted from operator entries at the screen on through a procedure ENTER statement.
		{lit-1} [lit-2] ...
CANCEL	IBM:	CANCEL {id-1 } [id-2]
	WANG:	(not supported).
	REMARKS:	A program can be cancelled through program code by calling a cancel subroutine.
ENTRY	IBM:	ENTRY lit-1 [USING id-1 [id-2] ...]
	WANG:	(not supported).
	REMARKS:	Each program can have only one entry point. If the programmer requires multiple entry points, a passed paramenter list could be utilized to indicate desired processing. (See Appendix C as an example.)
GOBACK	IBM:	GOBACK
	WANG:	(not supported).
	REMARKS:	EXIT PROGRAM provides the same end result action as does the GOBACK.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
NOTE	IBM:	NOTE. reserved word
	WANG:	(not supported).
	REMARKS:	NOTE is not an ANSI COBOL reserved word. All comment lines are indicated by a '*' in column 7.
IF	IBM:	IF condition {statement-2 } THEN {NEXT SENTENCE} {ELSE } {statement-2 } {OTHERWISE} {NEXT SENTENCE}
	WANG:	IF condition {statement-1 } THEN {NEXT SENTENCE} {ELSE } {statement-2 } {NEXT SENTENCE}
	REMARKS:	OTHERWISE is not accepted by the Wang compiler.
SEARCH	IBM:	SEARCH id-1 [VARYING {ind-nm} {id-2 }] [AT END imp-st-1] WHEN condition-1 {imp-st-2 } {NEXT SENTENCE} ... {imp-st-2 } SEARCH ALL id-1 [AT END imp-st-1] WHEN condition-1 {NEXT SENTENCE}
	WANG:	(not supported).
	REMARKS:	The SEARCH command, used with the table handling feature, is not yet available (planned product enhancement).
SORT	IBM:	ANS COBOL SORT/MERGE feature.
	WANG:	Utility
	REMARKS:	Sorting is handled in a Wang VS environment by the utility SORT. SORT enables the user to sort a file according to one or more record-embedded keys and to merge multiple sorted files. There is also a SORT subroutine which may be called from any COBOL program.

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

ITEM		SYNTAX AND CONVERSION REMARKS*
REPORT	IBM:	ANS COBOL REPORT WRITER feature.
	WANG:	Utility or "hard coding".
	REMARKS:	Reporting and report-formatting is performed by the Wang Utility REPORT. This utility allows the user to develop specialized reports, without programming, once the data file is available.
STRING/UNSTRING	IBM:	STRING {id-1 } [id-2] ... DELIMITED BY {id-3 } [{id-4 } [id-5] ... DELIMITED BY [{id-6 }] {lit-1} [lit-2] {lit-3} [{lit-4} [lit-5] {lit-6}] {SIZE } [{SIZE }] INTO id-7 [WITH POINTER id-8] [ON OVERFLOW imp-st] UNSTRING id-1 [DELIMITED BY [ALL] {id-2 } [OR ALL] {id-2 } {lit-1} {lit-2}] ...] INTO id-4 [DELIMITER IN id-5] [COUNT IN id-6] [id-7 [DELIMITER IN id-8] [COUNT IN id-9]]... [WITH POINTER id-10] [TALLYING IN id-11] [ON OVERFLOW imp-st] Functionally, STRING provides juxtaposition of partial or complete contents of 2 or more data items into a single data item. UNSTRING causes contiguous data in a sending field to be separated and placed into multiple fields.
	WANG:	(not supported).

* id = identifier imp-st = imperative-statement lit = literal ind-nm = index-name

APPENDIX B
SAMPLE VS COBOL PROGRAM

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. DATENTRY.
       ENVIRONMENT DIVISION.
       FIGURATIVE-CONSTANTS.
           ONE IS "01", NOTAB IS "80", DIM IS "8C", HIDE IS "9C",
           MOD-BRITA IS "81", BRITE IS "84", TAB IS "A0".
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT CRT ASSIGN TO "DISPLAY", "DISPLAY",
               ACCESS IS RANDOM,
               FILE STATUS IS FILESTAT.
           SELECT DEFILE ASSIGN TO "DATAFILE",
               ORGANIZATION IS INDEXED,
               RECORD KEY IS ACCTNO ,
               ALTERNATE RECORD KEY 01 IS CUSTOMER
                                    02 IS CITY     WITH DUPLICATES
                                    03 IS STATE    WITH DUPLICATES
                                    04 IS ZIP      WITH DUPLICATES
               ACCESS IS DYNAMIC.
       DATA DIVISION.
       FILE SECTION.
       FD  CRT
               LABEL RECORDS ARE STANDARD.
       01  DISPLAY-REC1 PIC X(1924).
       FD  DEFILE
               RECORD CONTAINS 0100 CHARACTERS,
               LABEL RECORDS ARE STANDARD.
       01  DE-RECORD-AREA.
           02  ACCTNO   PICTURE IS  X(10).
           02  CUSTOMER PICTURE IS  X(30).
           02  STREET   PICTURE IS  X(20).
           02  CITY     PICTURE IS  X(20).
           02  STATE    PICTURE IS  X(02).
           02  ZIP      PICTURE IS S9(05)       COMP.
           02  FILLER   PICTURE IS  X(15).
       WORKING-STORAGE SECTION.
       01  DESCREEN1 USAGE IS DISPLAY-WS.
           05  FILLER PIC X(43) ROW 6 COLUMN 20
               VALUE IS "FILE MAINTAINENCE PROGRAM FOR FILE DEMO    ".
           05  FILLER PIC X(54) ROW 9 COLUMN 15
               VALUE IS "SELECT THE APPROPRIATE PFKEY FOR THE FUNCTION D
      -                "ESIRED.".
           05  FILLER PIC X(23) ROW 12 COLUMN 30
               VALUE IS "PFKEY          FUNCTION".
           05  FILLER PIC X(37) ROW 14 COLUMN 31
               VALUE IS "PF3           ADD ENTRIES TO THE FILE".
           05  FILLER PIC X(46) ROW 15 COLUMN 31
               VALUE IS "PF4           MAINTAIN THE ENTRIES IN THE FILE
      -                " ".
           05  FILLER PIC X(40) ROW 16 COLUMN 31
               VALUE IS "PF5           DELETE ENTRIES IN THE FILE".
           05  FILLER PIC X(35) ROW 18 COLUMN 31
               VALUE IS "PF16          EXIT FROM THE PROGRAM".
       01  DESCREEN2 USAGE IS DISPLAY-WS.
           05  FILLER         PICTURE IS X(14)       ROW 07 COLUMN 41
```

```
        VALUE IS "ACCOUNT NUMBER".
    05  KEYFIELD1      PICTURE IS X(10)       ROW 07 COLUMN 57
        RANGE IS FROM "A"            TO "ZZZZZZZZZZ"
        SOURCE IS ACCTNO          OBJECT IS ACCTNO        .
    05  LINE232 PIC X(79) ROW 23 COLUMN 2,
            SOURCE IS LINE23S.
    05  LINE242 PIC X(79) ROW 24 COLUMN 2,
            SOURCE IS LINE24S.
01  FILESTAT.
    02  FILESTAT1 PIC X.
    02  FILESTAT2 PIC X.
01  POSITION-SCREEN1.
    02  FILLER PIC X VALUE ONE.
    02  SAFILLER PIC X VALUE TAB.
    02  FILLER PIC X(2) VALUE ONE.
01  DISPLAY-REC USAGE IS DISPLAY-WS.
    05  FILLER          PICTURE IS X(31)       ROW 01 COLUMN 26
        VALUE IS "***  WANG VS COPUTER SYSTEM ***".
    05  FILLER          PICTURE IS X(30)       ROW 05 COLUMN 03
        VALUE IS "SAMPLE DEMONSTRATION SCREEN...".
    05  FILLER          PICTURE IS X(14)       ROW 07 COLUMN 41
        VALUE IS "ACCOUNT NUMBER".
    05  ROW07-COL57     PICTURE IS X(10)       ROW 07 COLUMN 57
        RANGE IS FROM "A"            TO "ZZZZZZZZZZZZ"
        SOURCE IS ACCTNO          OBJECT IS ACCTNO        .
    05  FILLER          PICTURE IS X(08)       ROW 09 COLUMN 11
        VALUE IS "CUSTOMER".
    05  ROW09-COL21     PICTURE IS X(30)       ROW 09 COLUMN 21
        SOURCE IS CUSTOMER        OBJECT IS CUSTOMER      .
    05  FILLER          PICTURE IS X(06)       ROW 10 COLUMN 13
        VALUE IS "STREET".
    05  ROW10-COL21     PICTURE IS X(20)       ROW 10 COLUMN 21
        SOURCE IS STREET          OBJECT IS STREET        .
    05  FILLER          PICTURE IS X(04)       ROW 11 COLUMN 15
        VALUE IS "CITY".
    05  ROW11-COL21     PICTURE IS X(20)       ROW 11 COLUMN 21
        SOURCE IS CITY            OBJECT IS CITY          .
    05  FILLER          PICTURE IS X(05)       ROW 11 COLUMN 44
        VALUE IS "STATE".
    05  ROW11-COL51     PICTURE IS X(02)       ROW 11 COLUMN 51
        SOURCE IS STATE           OBJECT IS STATE         .
    05  FILLER          PICTURE IS X(03)       ROW 11 COLUMN 56
        VALUE IS "ZIP".
    05  ROW11-COL61     PICTURE IS Z(05)       ROW 11 COLUMN 61
        SOURCE IS ZIP             OBJECT IS ZIP           .
    05  LINE23          PICTURE IS X(77)       ROW 23 COLUMN 03
        SOURCE IS LINE23S       .
    05  LINE24          PICTURE IS X(77)       ROW 24 COLUMN 03
        SOURCE IS LINE24S       .
77  LINE23S         PICTURE IS X(77)       VALUE SPACES.
77  LINE24S         PICTURE IS X(77)       VALUE SPACES.
PROCEDURE DIVISION.
START-CODE.
    OPEN I-O CRT.
    MOVE POSITION-SCREEN1 TO ORDER-AREA OF DESCREEN1.
```

```
    OPEN SHARED DEFILE
MENU.
    DISPLAY AND READ DESCREEN1 ON CRT
        ONLY PFKEY 3, 4, 5, 16.
    IF FILESTAT2 = "P", CLOSE DEFILE, GO TO END-PROGRAM.
    MOVE SPACES TO LINE24S.
    IF FILESTAT2 = "D", MOVE DIM TO FAC OF ROW07-COL57
        GO TO MODIFYR.
    IF FILESTAT2 = "E", MOVE DIM TO FAC OF ROW07-COL57
        GO TO DELETER.
    MOVE "PRESS ENTER TO ADD ENTRY TO FILE, PF16 TO EXIT TO MENU
-        "." TO LINE23S.
    MOVE SPACES TO DE-RECORD-AREA.
ADDR-0.
    PERFORM RE-INIT.
ADDR-1.
    DISPLAY AND READ DISPLAY-REC ON CRT,
        ON PFKEY 16, PERFORM START-FILE, GO TO MENU.
    MOVE HIDE TO FAC OF LINE24.
    WRITE DE-RECORD-AREA INVALID KEY,
        MOVE BRITE TO FAC OF LINE24,
        MOVE "ENTRY ALREADY EXISTS." TO LINE24S, GO TO ADDR-1.
    GO TO ADDR-0.
MODIFYR.
    PERFORM RE-INIT.
        MOVE "ENTER=RETRIEVE BY KEY,     PF2=FIRST ENTRY,     PF3
-           "=NEXT ENTRY,     PF16=EXIT" TO LINE23S.
    MOVE HIDE TO FAC OF LINE242.
MCONT.
    DISPLAY AND READ DESCREEN2 ON CRT,
        PFKEY 2, 3, 16,
        ON PFKEY 16,
        MOVE MOD-BRITA TO FAC OF ROW07-COL57    , GO TO MENU.
    MOVE HIDE TO FAC OF LINE242.
    IF FILESTAT2 = "@", GO TO EXACT-KEY.
    IF FILESTAT2 = "C", GO TO READ-NEXT.
    PERFORM START-FILE, IF FAC OF LINE242 = BRITE, GO TO MCONT.
        GO TO READ-NEXT.
START-FILE.
    MOVE LOW-VALUES TO ACCTNO  .
    START DEFILE KEY NOT LESS THAN ACCTNO  ,
        INVALID KEY MOVE BRITE TO FAC OF LINE242,
          MOVE "NO ENTRIES IN FILE." TO LINE24S.
READ-NEXT.
    READ DEFILE NEXT WITH HOLD, INVALID KEY,
        MOVE BRITE TO FAC OF LINE242,
        MOVE "END OF FILE REACHED." TO LINE24S,
        GO TO MCONT.
    GO TO DISPLAY-MODSCREEN.
EXACT-KEY.
    READ DEFILE WITH HOLD INVALID KEY
        MOVE BRITE TO FAC OF LINE242,
        MOVE "ENTRY NOT FOUND." TO LINE24S,
        PERFORM START-FILE,
        GO TO MCONT.
```

```
DISPLAY -MODSCREEN.
    MOVE "MAKE THE MODIFICATIONS AND PRESS ENTER TO REPLACE THE E
-        "NTRY," TO LINE23S.
    MOVE "OR PRESS PF16 TO EXIT WITHOUT REPLACING THE ENTRY." TO
         LINE24S.
    MOVE DIM TO FAC OF LINE24.
    DISPLAY AND READ DISPLAY-REC ON CRT,
        ON PFKEY 16, GO TO MODIFYR.
    REWRITE DE-RECORD-AREA.
    GO TO MODIFYR.
DELETER.
    PERFORM RE-INIT.
        MOVE "ENTER THE KEY OF THE ENTRY TO BE DELETED, OR PRESS
-            "PF16 TO EXIT TO MENU." TO LINE23S.
    MOVE HIDE TO FAC OF LINE242.
DCONT.
    DISPLAY AND READ DESCREEN2 ON CRT,
        ON PFKEY 16, PERFORM START-FILE,
        MOVE MOD-BRITA TO FAC OF ROW07-COL57    , GO TO MENU.
    MOVE HIDE TO FAC OF LINE242.
    READ DEFILE WITH HOLD INVALID KEY,
        MOVE BRITE TO FAC OF LINE242,
        MOVE "ENTRY NOT FOUND." TO LINE24S,
        GO TO DCONT.
DISPLAY-DELSCREEN.
    MOVE "PRESS PF3 TO DELETE THIS ENTRY," TO LINE23S.
    MOVE "OR PRESS PF16 TO EXIT WITHOUT DELETING THE ENTRY." TO
         LINE24S.
    MOVE DIM TO FAC OF LINE24.
    DISPLAY AND READ DISPLAY-REC ON CRT,
        ONLY PFKEY 3, 16, ON PFKEY 16 GO TO DELETER.
    DELETE DEFILE.
    GO TO DELETER.
END-PROGRAM.
    CLOSE CRT.
    STOP RUN.
RE-INIT.
    MOVE SPACES TO ACCTNO      .
    MOVE SPACES TO CUSTOMER    .
    MOVE SPACES TO STREET      .
    MOVE SPACES TO CITY        .
    MOVE SPACES TO STATE       .
    MOVE ZERO   TO ZIP         .
```

APPENDIX C
SAMPLE CONVERSION PROGRAMS

The following programs are an example of a conversion from a batch-designed IBM COBOL program to a VS COROL program with no interactive features. The IBM program consists of a main program called "SAMPLE" and a subprogram called "BDLEDIT". The main program reads a file of variable length records with different record descriptions. It accumulates information and performs calculations for each group of records based on man number. Each group of records must contain an "A-type" record. If not, an error record is written. This program also performs calculations based on information contained in the records and passed to the program via a parm.

The main program calls the edit program using the records and calculated data. The subprogram then produces an edited listing of the record information.

These two programs, when converted to VS COBOL, require the code changes which are highlighted in the following VS source listings. Most changes can be handled through a global edit. These changes include converting:

1. all single quotation marks (') to double quotation marks("),
2. all COMP-3 usage to COMP usage,
3. REMARKS section to remark lines (by coding an asterisk (*) in column 7),
4. the SOURCE and OBJECT computer name to WANG-VS,
5. all WRITE of print lines to include AFTER ADVANCING, and
6. WRITE AFTER POSITIONING 0 LINES, which will cause top-of-form advancing, to WRITE AFTER ADVANCING PAGE, which will also result in a page advance.

In addition, some logic changes are required to accomodate a single, rather than multiple, entry point in a subprogram, PARM passing and blocking factors.

1. By testing a flag indicator set in the main program, certain edit program code sections can be conditionally executed.

2. The parm definition is not set up automatically by the VS, but this parm information may be accepted by the program at run-time through a procedure or through an interactive response. The parm value is replaced by an ACCEPT statement in the VS program. The value accepted is described with usage of DISPLAY. It can then be moved to a numeric field with the VS extension MOVE WITH CONVERSION. This statement will convert the character string supplied in response to the ACCEPT getparm, into a number that can be used for computation. It will also decimal align and numerically justify the value. LENGTH-OF-PARM, a special value passed to the program from the IBM operating system, is not valid in the VS operating system environment since PARMS, in this program sense, are not supported. However, the value accepted by the VS program may be validity checked.

3. Blocking information must be provided in an IBM operating environment by the program in the BLOCK CONTAINS clause, by the JCL, or by the label information. VS disk files are blocked in 2K increments, by default.

The procedure to execute the program converted to VS COBOL follows.

```
PROCEDURE
SET PROGLIB=DPLOBJ, PROGVOL=VOL444, INLIB=DPLDATA,
    INVOL=VOL444, OUTLIB=#DPLPRT, OUTVOL=VOL444
RUN DPLCONL
ENTER ACCEPT FACTOR1="32.9"
ENTER ACCEPT FACTOR2="16.3"
ENTER RECORDS FILE=DPLCONV
ENTER ERROUT FILE=DPLERR
ENTER PRTREC FILE=DPLPRT
```

```
ID DIVISION.
PROGRAM-ID.  DPLCONL.
AUTHOR.  J.P. PROGRAMMER.
DATE-WRITTEN.  FEB. 28, 1979.
DATE-COMPILED. MARCH 1, 1979.
REMARKS.  THIS PROGRAM WILL SCREEN CASES, CALCULATE TAXES
    AND TOTALS, AND YIELD  A LIST SHOWING ERRORS (MISSING TAX
    MULTIPLIERS OR RECORDS NOT PRECEDED BY AN A-TYPE RECORD).
    THIS MAIN CONTROL MODULE WILL CALL IN AN EDIT MODULE.



ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370-168.
OBJECT-COMPUTER.  IBM-370-168.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT OUTPUT-1 ASSIGN TO UT-S-RECORDS.
    SELECT ERRORS ASSIGN TO UT-S-ERROUT.



DATA DIVISION.
FILE SECTION.
FD  OUTPUT-1
    RECORDING MODE IS V
    RECORD CONTAINS 9 TO 35 CHARACTERS
    BLOCK CONTAINS 0 CHARACTERS
    LABEL RECORDS ARE OMITTED.
01  A-VALIDS.
    05  A-TIPE              PICTURE X.
    05  A-RECINFO                          COMPUTATIONAL-3.
        10  A-DEPTNO        PICTURE S999.
        10  A-MANNO         PICTURE S9(5).
        10  A-TAX-STATUS    PICTURE S9.
        10  A-EARNINGS      PICTURE S999V99.
    05  A-NAME              PICTURE X(25).
01  B-VALIDS.
    05  B-TIPE              PICTURE X.
    05  B-RECINFO                          COMPUTATIONAL-3.
        10  B-DEPTNO        PICTURE S999.
        10  B-MANNO         PICTURE S9(5).
        10  B-BONUS         PICTURE S999V99.

FD  ERRORS
    RECORDING MODE IS F
    RECORD CONTAINS 80 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE OMITTED.
01  ERROR-LIST.
    05  E-TIPE      PICTURE X.
    05  E-DEPTNO    PICTURE 999.
    05  E-MANNO     PICTURE 9(5).
    05  E-BONUS     PICTURE 9(5).
    05  FILLER      PICTURE X(66).
01  ERR-LIST        PICTURE X(80).
```

```
WORKING-STORAGE SECTION.
77  PRIOR-DEPT-NO PICTURE S999     COMPUTATIONAL-3 VALUE ZEROES.
77  RECS-LEFT     PICTURE XXX                     VALUE 'YES'.
77  TAXE          PICTURE S9      COMPUTATIONAL-3 VALUE ZERO.
77  CHECK         PICTURE S999V99 COMPUTATIONAL-3 VALUE ZEROES.
77  A-REC-SW      PICTURE X                       VALUE SPACE.
    88  A-READ VALUE 'A'.

01  WSREC.
    05  DEPT-NO    PICTURE S999     COMPUTATIONAL-3  VALUE ZEROES.
    05  MAN-NO     PICTURE S9(5)    COMPUTATIONAL-3  VALUE ZEROES.
    05  NAME       PICTURE X(25)                     VALUE SPACES.
    05  EARNINGS   PICTURE S999V99 COMPUTATIONAL-3  VALUE ZEROES.
    05  BONUS-1    PICTURE S999V99 COMPUTATIONAL-3  VALUE ZEROES.
    05  BONUS-2    PICTURE S999V99 COMPUTATIONAL-3  VALUE ZEROES.
    05  GROSS      PICTURE S999V99 COMPUTATIONAL-3  VALUE ZEROES.
    05  FED-TAX    PICTURE S999V99 COMPUTATIONAL-3  VALUE ZEROES.

01  TOTREC                              COMPUTATIONAL-3.
    05  T-DEPTNO    PICTURE S999                     VALUE ZEROES.
    05  T-EARNINGS  PICTURE S9999V99                 VALUE ZEROES.
    05  T-BONUS-1   PICTURE S9999V99                 VALUE ZEROES.
    05  T-BONUS-2   PICTURE S9999V99                 VALUE ZEROES.
    05  T-GROSS     PICTURE S9999V99                 VALUE ZEROES.
    05  T-FED-TAX   PICTURE S9999V99                 VALUE ZEROES.

LINKAGE SECTION.
 01  PARMINFO COPY PARMINFO.
C01  PARMINFO.
C    02 LENGTH-OF-PARM PICTURE S99 COMP SYNC.
C    02 FACTOR1 PICTURE S99V99.
C    02 FACTOR2 PICTURE SV99.



PROCEDURE DIVISION USING PARMINFO.
100-START-HERE.
    OPEN INPUT OUTPUT-1
         OUTPUT ERRORS.
    CALL '100-ENTRY1'.

200-READ.
    READ OUTPUT-1
        AT END MOVE 'NO' TO RECS-LEFT.
200-EXIT.
    EXIT.
```

```
300-PARM-TEST.
    IF LENGTH-OF-PARM NOT EQUAL 6
        THEN MOVE SPACES TO ERR-LIST
             MOVE 'ERROR MESSAGE: FACTORS NOT COPIED.' TO ERR-LIST
             WRITE ERR-LIST
             MOVE 3 TO RETURN-CODE
             GO TO 550-STOP
             ELSE MOVE SPACES TO ERR-LIST
             MOVE 'ERRORS: TYPES B AND C NOT PRECEDED BY TYPE A.'
                    TO ERR-LIST
             WRITE ERR-LIST.
    MOVE A-DEPTNO TO PRIOR-DEPT-NO.
    MOVE B-MANNO TO MAN-NO.

400-CONTROL.
    IF A-DEPTNO NOT = PRIOR-DEPT-NO
        THEN PERFORM 525-COMPUTE THRU 525-EXIT
             CALL '300-ENTRY3' USING TOTREC RECS-LEFT
             MOVE A-DEPTNO TO PRIOR-DEPT-NO
             MOVE B-MANNO TO MAN-NO
             GO TO 425-BADRECCHECK
        ELSE NEXT SENTENCE.
    IF B-MANNO NOT = MAN-NO
        THEN PERFORM 525-COMPUTE THRU 525-EXIT
             MOVE B-MANNO TO MAN-NO
        ELSE NEXT SENTENCE.

425-BADRECCHECK.
    IF A-TIPE = 'A'
        THEN PERFORM 530-WSREC-FORM THRU 530-EXIT
        ELSE NEXT SENTENCE.
    IF A-READ
        THEN NEXT SENTENCE
        ELSE PERFORM 535-ERROR THRU 535-EXIT
             GO TO 430-PERFREAD.
    IF A-TIPE = 'B'
        THEN MOVE B-BONUS TO BONUS-1
        ELSE NEXT SENTENCE.
    IF A-TIPE = 'C'
        THEN MOVE B-BONUS TO BONUS-2
        ELSE NEXT SENTENCE.

430-PERFREAD.
    PERFORM  200-READ THRU 200-EXIT.
    IF RECS-LEFT = 'NO'
        THEN NEXT SENTENCE
        ELSE GO TO 400-CONTROL.
    IF A-READ
        THEN PERFORM 525-COMPUTE THRU 525-EXIT
        ELSE NEXT SENTENCE.
    CALL '300-ENTRY3' USING TOTREC RECS-LEFT.

500-STOP.
    CLOSE OUTPUT-1 ERRORS.
    CALL '400-ENTRY4'.
```

```
 510-COPY.
     COPY DISPEOJ.
C    DISPLAY 'YOUR PROGRAM HAS REACHED END OF JOB.'.
     STOP RUN.

 525-COMPUTE.
     IF A-READ
         THEN NEXT SENTENCE
         ELSE GO TO 525-EXIT.
     COMPUTE GROSS
         ROUNDED = EARNINGS + BONUS-1 + BONUS-2.
     COMPUTE CHECK
         ROUNDED = (GROSS - (TAXE * FACTOR1)).
     IF CHECK NOT  GREATER THAN 0
         THEN MOVE ZEROES TO FED-TAX
         ELSE COMPUTE FED-TAX
             ROUNDED = CHECK * FACTOR2.
     MOVE DEPT-NO TO T-DEPTNO.
     ADD EARNINGS TO T-EARNINGS.
     ADD BONUS-1  TO T-BONUS-1.
     ADD BONUS-2  TO T-BONUS-2.
     ADD GROSS    TO T-GROSS.
     ADD FED-TAX  TO T-FED-TAX.
     MOVE SPACE   TO A-REC-SW.
     CALL '200-ENTRY2' USING WSREC.
 525-EXIT.
     EXIT.

 530-WSREC-FORM.
     MOVE A-MANNO      TO MAN-NO.
     MOVE A-TAX-STATUS TO TAXE.
     MOVE A-EARNINGS   TO  EARNINGS.
     MOVE A-NAME       TO NAME.
     MOVE 'A'          TO A-REC-SW.
     MOVE ZEROES       TO BONUS-1.
     MOVE ZEROES       TO BONUS-2.
     MOVE A-DEPTNO     TO DEPT-NO.
 530-EXIT.
     EXIT.

 535-ERROR.
     MOVE SPACES   TO ERR-LIST.
     MOVE B-TIPE   TO E-TIPE.
     MOVE B-DEPTNO TO E-DEPTNO.
     MOVE B-MANNO  TO E-MANNO.
     MOVE B-BONUS  TO E-BONUS.
     WRITE ERROR-LIST.
 535-EXIT.
     EXIT.

 575-STOP.
     CLOSE OUTPUT-1 ERRORS.
     CALL '400-ENTRY4'.
     STOP RUN.

*** END OF LISTING ***********************************************************************
```

```
ID DIVISION.
PROGRAM-ID.  DPLEDIT.
REMARKS.     THIS IS A SUBPROGRAM OF DPLCONL AND IS USED TO
        PROVIDE HEADINGS AND AESTHETICALLY FORMAT OUTPUT.



ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370-168.
OBJECT-COMPUTER. IBM-370-168.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT EDOUT ASSIGN TO UT-S-PAYROLL.



DATA DIVISION.
FILE SECTION.
FD  EDOUT
    RECORDING MODE IS F
    RECORD CONTAINS 133 CHARACTERS
    BLOCK CONTAINS 0 CHARACTERS
    LABEL RECORDS ARE OMITTED.
01  DETAIL-REC.
    05  FILLER              PICTURE X(3).
    05  DEPT-NO             PICTURE 9(3).
    05  FILLER              PICTURE X(10).
    05  MAN-NO              PICTURE 9(5).
    05  FILLER              PICTURE X(10).
    05  NAME                PICTURE X(25).
    05  FILLER              PICTURE X(5).
    05  EARNINGS            PICTURE $$$$.99.
    05  FILLER              PICTURE X(8).
    05  BONUS-2             PICTURE $$$$.99.
    05  FILLER              PICTURE X(8).
    05  BONUS-1             PICTURE $$$$.99.
    05  FILLER              PICTURE X(8).
    05  GROSS               PICTURE $$$$.99.
    05  FILLER              PICTURE X(8).
    05  FED-TAX             PICTURE $$$$.99.
    05  FILLER              PICTURE X(5).
01  TOTAL.
    05  EMPTY               PICTURE X(3).
    05  T-DEPTNO            PICTURE 9(3).
    05  T-MESSAGE           PICTURE X(24).
    05  FILLER              PICTURE X(31).
    05  T-EARNINGS          PICTURE $$,$$$.99.
    05  FILLER              PICTURE X(6).
    05  T-BONUS-1           PICTURE $$,$$$.99.
    05  FILLER              PICTURE X(6).
    05  T-BONUS-2           PICTURE $$,$$$.99.
    05  FILLER              PICTURE X(6).
    05  T-GROSS             PICTURE $$,$$$.99.
    05  FILLER              PICTURE X(6).
    05  T-FED-TAX           PICTURE $$,$$$.99.
    05  FILLER              PICTURE X(3).
```

```
WORKING-STORAGE SECTION.
77  LINE-CTR     PICTURE S99 COMPUTATIONAL-3 VALUE +61.
01  MAIN-HEADING.
    05  FILLER  PICTURE X(51) VALUE SPACES.
    05  MH PICTURE X(29) VALUE 'LISTING OF DEPT  PAYROLL DATA'.
    05  FILLER PICTURE X(53)  VALUE SPACES.
01  SUB-HEAD.
    05  FILLER        PICTURE X(3)    VALUE SPACES.
    05  A1            PICTURE X(8)    VALUE 'DEPT-NO'.
    05  FILLER        PICTURE X(5)    VALUE SPACES.
    05  A2            PICTURE X(6)    VALUE 'MAN NO'.
    05  FILLER        PICTURE X(9)    VALUE SPACES.
    05  A3            PICTURE X(4)    VALUE 'NAME'.
    05  FILLER        PICTURE X(26)   VALUE SPACES.
    05  A4            PICTURE X(12)   VALUE 'ST. EARNINGS'.
    05  FILLER        PICTURE X(3)    VALUE SPACES.
    05  A5            PICTURE X(12)   VALUE 'SPEC  BONUS '.
    05  FILLER        PICTURE X(3)    VALUE SPACES.
    05  A6            PICTURE X(12)   VALUE 'BASIS BONUS '.
    05  FILLER        PICTURE X(3)    VALUE SPACE.
    05  A7            PICTURE X(14)   VALUE 'GROSS EARNINGS'.
    05  FILLER        PICTURE X       VALUE SPACES.
    05  A8            PICTURE X(12)   VALUE 'FTX WITHHELD'.

LINKAGE SECTION.
77  RECS-LEFT PICTURE XXX.
01  WSREC.
    05  DEPT-NO  PICTURE S999      COMPUTATIONAL-3.
    05  MAN-NO   PICTURE S9(5)     COMPUTATIONAL-3.
    05  NAME     PICTURE X(25).
    05  EARNINGS PICTURE S999V99   COMPUTATIONAL-3.
    05  BONUS-2  PICTURE S999V99   COMPUTATIONAL-3.
    05  BONUS-1  PICTURE S999V99   COMPUTATIONAL-3.
    05  GROSS    PICTURE S999V99   COMPUTATIONAL-3.
    05  FED-TAX  PICTURE S999V99   COMPUTATIONAL-3.
01  TOTREC                         COMPUTATIONAL-3.
    05  T-DEPTNO   PICTURE S999.
    05  T-EARNINGS PICTURE S9999V99.
    05  T-BONUS-1  PICTURE S9999V99.
    05  T-BONUS-2  PICTURE S9999V99.
    05  T-GROSS    PICTURE S9999V99.
    05  T-FED-TAX  PICTURE S9999V99.
    EJECT
```

```
PROCEDURE DIVISION.
    ENTRY '100-ENTRY1'.
    OPEN OUTPUT EDOUT.
    GOBACK.
    ENTRY '200-ENTRY2' USING WSREC.

210-WRITE-A-LINE.
    IF LINE-CTR GREATER THAN 57
        THEN PERFORM 220-WRITE-HEADINGS
        ELSE MOVE SPACES TO DETAIL-REC
             MOVE CORRESPONDING WSREC TO DETAIL-REC
             WRITE DETAIL-REC AFTER POSITIONING 2 LINES
             ADD 2 TO LINE-CTR.
    GO TO 230-GOBACK.

220-WRITE-HEADINGS.
    WRITE DETAIL-REC FROM MAIN-HEADING AFTER POSITIONING 0 LINES.
    WRITE DETAIL-REC FROM SUB-HEAD AFTER POSITIONING 2 LINES.
    MOVE SPACES TO DETAIL-REC.
    MOVE CORRESPONDING WSREC TO DETAIL-REC.
    WRITE DETAIL-REC AFTER POSITIONING 3 LINES.
    MOVE 9 TO LINE-CTR.

230-GOBACK.
    GOBACK.
    ENTRY '300-ENTRY3' USING TOTREC RECS-LEFT.
    IF T-GROSS OF TOTREC = 0
        THEN GO TO 230-GOBACK
        ELSE NEXT  SENTENCE.
    MOVE SPACES TO TOTAL.
    MOVE CORRESPONDING TOTREC TO TOTAL.
    MOVE 'DEPARTMENT TOTALS FOLLOW' TO T-MESSAGE.
    WRITE TOTAL AFTER POSITIONING 3 LINES.
    MOVE ZEROES TO T-DEPTNO OF TOTREC.
    MOVE ZEROES TO T-EARNINGS OF TOTREC.
    MOVE ZEROES TO T-BONUS-1 OF TOTREC.
    MOVE ZEROES TO T-BONUS-2 OF TOTREC.
    MOVE ZEROES TO T-GROSS OF TOTREC.
    MOVE ZEROES TO T-FED-TAX OF TOTREC.


310-WRITE-HEADINGS.
    IF RECS-LEFT = 'NO '
        THEN GO TO 230-GOBACK
        ELSE NEXT SENTENCE.
    WRITE DETAIL-REC FROM MAIN-HEADING AFTER POSITIONING 0 LINES.
    WRITE DETAIL-REC FROM SUB-HEAD AFTER POSITIONING 2 LINES.
    MOVE 6 TO LINE-CTR.
    GO TO 230-GOBACK.
    ENTRY '400-ENTRY4'.
    CLOSE EDOUT.
    GOBACK.

*** END OF LISTING ************************************************************************
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  DPLCONL.
AUTHOR.  J. P. PROGRAMMER.
DATE-WRITTEN.  FEB. 20, 79.
DATE-COMPILED.
    FRIDAY     MAR 16, 1979
*REMARKS.   THIS PROGRAM WILL SCREEN CASES, CALCULATE TAXES
*         AND TOTALS, AND YIELD A LIST OF ERRORS (AN ERROR
*         OCCURS WHEN A TAX MULTIPLIER IS MISSING OR A GROUP
*         OF RECORDS IS NOT PRECEDED  BY AN A-TYPE RECORD).
*         THIS MAIN CONTROL MODULE WILL CALL IN AN EDIT MODULE.


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  WANG-VS.
OBJECT-COMPUTER.  WANG-VS.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT OUTPUT-1 ASSIGN TO "RECORDS" "DISK".
    SELECT ERRORS   ASSIGN TO "ERROUT"  "PRINTER".


DATA DIVISION.
FILE SECTION.
FD  OUTPUT-1
    RECORD CONTAINS 9 TO 35 CHARACTERS
    LABEL RECORDS ARE STANDARD.
01  A-VALIDS.
    05  A-TIPE             PICTURE X.
    05  A-RECINFO                          COMPUTATIONAL.
        10  A-DEPTNO       PICTURE S999.
        10  A-MANNO        PICTURE S9(5).
        10  A-TAX-STATUS   PICTURE S9.
        10  A-EARNINGS     PICTURE S999V99.
    05  A-NAME             PICTURE X(25).
01  B-VALIDS.
    05  B-TIPE           PICTURE X.
    05  B-RECINFO                        COMPUTATIONAL.
        10  B-DEPTNO     PICTURE S999.
        10  B-MANNO      PICTURE S9(5).
        10  B-BONUS      PICTURE S999V99.


FD  ERRORS
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS.
01  ERROR-LIST.
    05  E-TIPE         PICTURE X.
    05  E-DEPTNO       PICTURE 999.
    05  E-MANNO        PICTURE 9(5).
    05  E-BONUS        PICTURE 9(5).
    05  FILLER         PICTURE X(66).
01  ERR-LIST           PICTURE X(80).
```

```
WORKING-STORAGE SECTION.
77   PRIOR-DEPT-NO     PICTURE S999     COMPUTATIONAL VALUE ZEROES.
77   RECS-LEFT         PICTURE XXX                    VALUE "YES".
77   TAXE              PICTURE S9       COMPUTATIONAL VALUE ZERO.
77   CHECK             PICTURE S999V99  COMPUTATIONAL VALUE ZEROES.
77   A-REC-SW          PICTURE X                      VALUE SPACE.
     88  A-READ                                       VALUE "A".
77   ENTRY-FLAG        PICTURE 9                      VALUE ZERO.
01   WSREC.
     05   DEPT-NO      PICTURE S999     COMPUTATIONAL VALUE ZEROES.
     05   MAN-NO       PICTURE S9(5)    COMPUTATIONAL VALUE ZEROES.
     05   NAME          PICTURE X(25)                  VALUE SPACES.
     05   EARNINGS      PICTURE S999V99 COMPUTATIONAL VALUE ZEROES.
     05   BONUS-1       PICTURE S999V99 COMPUTATIONAL VALUE ZEROES.
     05   BONUS-2       PICTURE S999V99 COMPUTATIONAL VALUE ZEROES.
     05   GROSS         PICTURE S999V99 COMPUTATIONAL VALUE ZEROES.
     05   FED-TAX       PICTURE S999V99 COMPUTATIONAL VALUE ZEROES.
01   TOTREC                                COMPUTATIONAL.
     05   T-DEPTNO      PICTURE S999                   VALUE ZEROES.
     05   T-EARNINGS    PICTURE S9999V99               VALUE ZEROES.
     05   T-BONUS-1     PICTURE S9999V99               VALUE ZEROES.
     05   T-BONUS-2     PICTURE S9999V99               VALUE ZEROES.
     05   T-GROSS       PICTURE S9999V99               VALUE ZEROES.
     05   T-FED-TAX     PICTURE S9999V99               VALUE ZEROES.

77   FACTOR1           PICTURE X(5)                   VALUE SPACE.
77   FACTOR2           PICTURE X(3)                   VALUE SPACE.
77   FACTOR1-CONV      PICTURE S99V99   COMPUTATIONAL VALUE ZEROES.
77   FACTOR2-CONV      PICTURE SV99     COMPUTATIONAL VALUE ZEROES.
```

```
/
PROCEDURE DIVISION.
100-START-HERE.
    OPEN INPUT OUTPUT-1
         OUTPUT ERRORS.
    MOVE 1 TO ENTRY-FLAG.
    CALL "DPLEDIT" USING ENTRY-FLAG.

200-READ.
    READ OUTPUT-1
        AT END MOVE "NO" TO RECS-LEFT.
200-EXIT.
    EXIT.

300-PARM-TEST.
    ACCEPT FACTOR1.
    ACCEPT FACTOR2.
    MOVE WITH CONVERSION FACTOR1 TO FACTOR1-CONV
        ON ERROR MOVE 999 TO RETURN-CODE
           DISPLAY "BAD FACTOR1"
           GO TO 575-STOP.
    MOVE WITH CONVERSION FACTOR2 TO FACTOR2-CONV
        ON ERROR MOVE 999 TO RETURN-CODE
           DISPLAY "BAD FACTOR2"
           GO TO 575-STOP.
    IF FACTOR1-CONV IS LESS THAN 0
             OR FACTOR2-CONV IS LESS THAN 0
        THEN MOVE SPACES TO ERR-LIST
            MOVE "ERROR MESSAGE: FACTORS NOT COPIED." TO ERR-LIST
            WRITE ERR-LIST AFTER ADVANCING 1 LINE
            MOVE 3 TO RETURN-CODE
            GO TO 575-STOP
        ELSE MOVE SPACES TO ERR-LIST
            MOVE "ERRORS: TYPES B AND C NOT PRECEDED BY TYPE A."
                    TO ERR-LIST
            WRITE ERR-LIST AFTER ADVANCING 1 LINE.
    MOVE A-DEPTNO TO PRIOR-DEPT-NO.
    MOVE B-MANNO TO MAN-NO.


400-CONTROL.
    IF A-DEPTNO NOT = PRIOR-DEPT-NO
        THEN PERFORM 525-COMPUTE THRU 525-EXIT
          MOVE 3 TO ENTRY-FLAG
          CALL "DPLEDIT" USING ENTRY-FLAG RECS-LEFT TOTREC
          MOVE A-DEPTNO TO PRIOR-DEPT-NO
          MOVE B-MANNO TO MAN-NO
          GO TO 425-BADRECCHECK
        ELSE NEXT SENTENCE.
    IF B-MANNO NOT = MAN-NO
        THEN PERFORM 530-WSREC-FORM THRU 530-EXIT
        ELSE NEXT SENTENCE.
```

```
425-BADRECCHECK.
    IF A-TIPE = "A"
        THEN PERFORM 530-WSREC-FORM THRU 530-EXIT
        ELSE NEXT SENTENCE.
    IF A-READ
        THEN NEXT SENTENCE
        ELSE PERFORM 535-ERROR THRU 535-EXIT
             GO TO 430-PERFREAD.
    IF A-TIPE = "B"
        THEN MOVE B-BONUS TO BONUS-1
        ELSE NEXT SENTENCE.
    IF A-TIPE = "C"
        THEN MOVE B-BONUS TO BONUS-2
        ELSE NEXT SENTENCE.

430-PERFREAD.
    PERFORM 200-READ THRU 200-EXIT.
    IF RECS-LEFT = "NO"
        THEN NEXT SENTENCE
        ELSE GO TO 400-CONTROL.
    IF A-READ
        THEN PERFORM 525-COMPUTE THRU 525-EXIT
        ELSE NEXT SENTENCE.
    MOVE 3 TO ENTRY-FLAG.
    CALL "DPLEDIT" USING ENTRY-FLAG RECS-LEFT TOTREC.

500-STOP.
    CLOSE OUTPUT-1 ERRORS.
    MOVE 4 TO ENTRY-FLAG.
    CALL "DPLEDIT" USING ENTRY-FLAG.
510-COPY.
    COPY DISPEOJ.
    DISPLAY "YOUR PROGRAM HAS REACHED END OF JOB.".
    STOP RUN.

525-COMPUTE.
    IF A-READ
        THEN NEXT SENTENCE
        ELSE GO TO 525-EXIT.
    COMPUTE GROSS
        ROUNDED = EARNINGS + BONUS-1 + BONUS-2.
    COMPUTE CHECK
        ROUNDED = (GROSS - (TAXE * FACTOR1-CONV)).
    IF CHECK NOT GREATER THAN 0
        THEN MOVE  ZEROES TO FED-TAX
        ELSE COMPUTE FED-TAX
            ROUNDED = CHECK * FACTOR2-CONV.
```

```
    MOVE DEPT-NO TO T-DEPTNO.
    ADD EARNINGS TO T-EARNINGS.
    ADD BONUS-1  TO T-BONUS-1.
    ADD BONUS-2  TO T-BONUS-2.
    ADD GROSS    TO T-GROSS.
    ADD FED-TAX  TO T-FED-TAX.
    MOVE SPACE   TO A-REC-SW.
    MOVE 2       TO ENTRY-FLAG.
    CALL "DPLEDIT" USING ENTRY-FLAG RECS-LEFT TOTREC WSREC.
525-EXIT.
    EXIT.
530-WSREC-FORM.
    MOVE A-MANNO       TO MAN-NO.
    MOVE A-TAX-STATUS TO TAXE.
    MOVE A-EARNINGS    TO EARNINGS.
    MOVE A-NAME        TO NAME.
    MOVE "A"           TO A-REC-SW.
    MOVE ZEROES        TO BONUS-1.
    MOVE ZEROES        TO BONUS-2.
    MOVE A-DEPTNO      TO DEPT-NO.
530-EXIT.
    EXIT.

535-ERROR.
    MOVE SPACES   TO ERR-LIST.
    MOVE B-TIPE   TO E-TIPE.
    MOVE B-DEPTNO TO E-DEPTNO.
    MOVE B-MANNO  TO E-MANNO.
    MOVE B-BONUS  TO E-BONUS.
    WRITE ERROR-LIST AFTER ADVANCING 1 LINE.
535-EXIT.
    EXIT.

575-STOP.
    CLOSE OUTPUT-1 ERRORS.
    MOVE 4 TO ENTRY-FLAG.
    CALL "DPLEDIT" USING ENTRY-FLAG.
    STOP RUN.
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. DPLEDIT.
*REMARKS.    THIS IS A SUBPROGRAM OF DPLCONL AND IS USED TO
*    PROVIDE HEADINGS AND AESTHETICALLY FORMAT OUTPUT.
*
*


 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. WANG-VS.
 OBJECT-COMPUTER. WANG-VS.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT EDOUT ASSIGN TO "PRTREC" "PRINTER".


 DATA DIVISION.
 FILE SECTION.
 FD  EDOUT
     RECORD CONTAINS 132 CHARACTERS
     LABEL RECORDS ARE STANDARD.
 01  DETAIL-REC.
     05  FILLER       PICTURE X(2).
     05  DEPT-NO-ED   PICTURE 9(3).
     05  FILLER       PICTURE X(10).
     05  MAN-NO-ED    PICTURE 9(5).
     05  FILLER       PICTURE X(10).
     05  NAME-ED      PICTURE X(25).
     05  FILLER       PICTURE X(5).
     05  EARNINGS-ED PICTURE $$$$.99.
     05  FILLER       PICTURE X(8).
     05  BONUS-2-ED   PICTURE $$$$.99.
     05  FILLER       PICTURE X(8).
     05  BONUS-1-ED   PICTURE $$$$.99.
     05  FILLER       PICTURE X(8).
     05  GROSS-ED     PICTURE $$$$.99.
     05  FILLER       PICTURE X(8).
     05  FED-TAX-ED   PICTURE $$$$.99.
     05  FILLER       PICTURE X(5).
 01  TOTAL.
     05  EMPTY          PICTURE X(2).
     05  T-DEPTNO-ED    PICTURE 9(3).
     05  T-MESSAGE      PICTURE X(24).
     05  FILLER         PICTURE X(31).
     05  T-EARNINGS-ED PICTURE $$,$$$.99.
     05  FILLER         PICTURE X(6).
     05  T-BONUS-1-ED   PICTURE $$,$$$.99.
     05  FILLER         PICTURE X(6).
     05  T-BONUS-2-ED   PICTURE $$,$$$.99.
     05  FILLER         PICTURE X(6).
     05  T-GROSS-ED     PICTURE $$,$$$.99.
     05  FILLER         PICTURE X(6).
     05  T-FED-TAX-ED   PICTURE $$,$$$.99.
     05  FILLER         PICTURE X(3).
```

```
WORKING-STORAGE SECTION.
77  LINE-CTR     PICTURE S99 COMPUTATIONAL VALUE +61.
01  MAIN-HEADING.
    05  FILLER  PICTURE X(50)  VALUE SPACES.
    05  MH  PICTURE X(29)  VALUE "LISTING OF DEPT  PAYROLL DATA".
    05  FILLER  PICTURE X(53)  VALUE SPACES.
01  SUB-HEAD.
    05  FILLER  PICTURE X(2)   VALUE SPACE.
    05  A1      PICTURE X(8)   VALUE "DEPT-NO ".
    05  FILLER  PICTURE X(5)   VALUE SPACE.
    05  A2      PICTURE X(6)   VALUE "MAN NO".
    05  FILLER  PICTURE X(9)   VALUE SPACE.
    05  A3      PICTURE X(4)   VALUE "NAME".
    05  FILLER  PICTURE X(26)  VALUE SPACE.
    05  A4      PICTURE X(12)  VALUE "STD EARNINGS".
    05  FILLER  PICTURE X(3)   VALUE SPACE.
    05  A5      PICTURE X(12)  VALUE "ORD VARIABLE".
    05  FILLER  PICTURE X(3)   VALUE SPACE.
    05  A6      PICTURE X(12)  VALUE "GRP VARIABLE".
    05  FILLER  PICTURE X(3)   VALUE SPACE.
    05  A7      PICTURE X(14)  VALUE "GROSS EARNINGS".
    05  FILLER  PICTURE X      VALUE SPACE.
    05  A8      PICTURE X(12)  VALUE "FTX WITHHELD".

LINKAGE SECTION.
77  ENTRY-FLAG      PICTURE 9.
77  RECS-LEFT       PICTURE XXX.
01  WSREC.
    05  DEPT-NO     PICTURE S999     COMPUTATIONAL.
    05  MAN-NO      PICTURE S9(5)    COMPUTATIONAL.
    05  NAME        PICTURE X(25).
    05  EARNINGS    PICTURE S999V99  COMPUTATIONAL.
    05  BONUS-2     PICTURE S999V99  COMPUTATIONAL.
    05  BONUS-1     PICTURE S999V99  COMPUTATIONAL.
    05  GROSS       PICTURE S999V99  COMPUTATIONAL.
    05  FED-TAX     PICTURE S999V99  COMPUTATIONAL.
01  TOTREC                           COMPUTATIONAL.
    05  T-DEPTNO     PICTURE S999.
    05  T-EARNINGS   PICTURE S9999V99.
    05  T-BONUS-1    PICTURE S9999V99.
    05  T-BONUS-2    PICTURE S9999V99.
    05  T-GROSS      PICTURE S9999V99.
    05  T-FED-TAX    PICTURE S9999V99.
```

```
/
PROCEDURE DIVISION USING ENTRY-FLAG RECS-LEFT
                        TOTREC WSREC.

SECTION0 SECTION.
100-CONTROL.
    IF  ENTRY-FLAG = 1
        THEN PERFORM SECTION1
        ELSE IF ENTRY-FLAG = 2
               THEN PERFORM SECTION2
               ELSE IF ENTRY-FLAG = 3
                      THEN PERFORM SECTION3
                      ELSE IF ENTRY-FLAG = 4
                             THEN PERFORM SECTION4
                             ELSE DISPLAY "BAD ENTRY FLAG".
    EXIT PROGRAM.
SECTION1 SECTION.
OPEN-IT.
    OPEN OUTPUT EDOUT.

SECTION2 SECTION.
210-WRITE-A-LINE.
    IF LINE-CTR GREATER THAN 57
        THEN PERFORM 220-WRITE-HEADINGS
        ELSE MOVE SPACES TO DETAIL-REC
             MOVE DEPT-NO  TO DEPT-NO-ED
             MOVE MAN-NO   TO MAN-NO-ED
             MOVE NAME     TO NAME-ED
             MOVE EARNINGS TO EARNINGS-ED
             MOVE BONUS-2  TO BONUS-2-ED
             MOVE BONUS-1  TO BONUS-1-ED
             MOVE GROSS    TO GROSS-ED
             MOVE FED-TAX  TO FED-TAX-ED
             WRITE DETAIL-REC AFTER ADVANCING 2 LINES
             ADD 2 TO LINE-CTR.
    GO TO 230-EXIT.
220-WRITE-HEADINGS.
    WRITE DETAIL-REC FROM MAIN-HEADING AFTER ADVANCING PAGE.
    WRITE DETAIL-REC FROM SUB-HEAD AFTER ADVANCING 2 LINES.
    MOVE SPACES   TO DETAIL-REC.
    MOVE DEPT-NO  TO DEPT-NO-ED.
    MOVE MAN-NO   TO MAN-NO-ED.
    MOVE NAME     TO NAME-ED.
    MOVE EARNINGS TO EARNINGS-ED.
    MOVE BONUS-2  TO BONUS-2-ED.
    MOVE BONUS-1  TO BONUS-1-ED.
    MOVE GROSS    TO GROSS-ED.
    MOVE FED-TAX  TO FED-TAX-ED.
    WRITE DETAIL-REC AFTER ADVANCING 3 LINES.
    MOVE 9 TO LINE-CTR.
230-EXIT.
    EXIT.
```

```
/
SECTION3 SECTION.
300-WRITE-TOTAL.
    IF  T-GROSS = 0
        THEN GO TO 320-EXIT
        ELSE NEXT SENTENCE.
    MOVE SPACES      TO TOTAL.
    MOVE T-DEPTNO    TO T-DEPTNO-ED.
    MOVE T-EARNINGS TO T-EARNINGS-ED.
    MOVE T-BONUS-1   TO T-BONUS-1-ED.
    MOVE T-BONUS-2   TO T-BONUS-2-ED.
    MOVE T-GROSS     TO T-GROSS-ED.
    MOVE T-FED-TAX   TO T-FED-TAX-ED.
    MOVE "DEPARTMENT TOTALS FOLLOW" TO T-MESSAGE.
    WRITE TOTAL AFTER ADVANCING 3 LINES.
    MOVE ZEROES TO T-DEPTNO T-EARNINGS T-BONUS-1
                   T-BONUS-2 T-GROSS T-FED-TAX.
310-WRITE-HEADINGS.
    IF RECS-LEFT = "NO"
        THEN GO TO 320-EXIT
        ELSE NEXT SENTENCE.
    WRITE DETAIL-REC FROM MAIN-HEADING AFTER ADVANCING PAGE.
    WRITE DETAIL-REC FROM SUB-HEAD AFTER ADVANCING 2 LINES.
    MOVE 6 TO LINE-CTR.
320-EXIT.
    EXIT.

SECTION4 SECTION.
CLOSE-UP.
    CLOSE EDOUT.
```

APPENDIX D
RESERVED WORDS

PART 1 - Reserved Words Recognized by Wang, but not IBM Compiler

ALARM
ALSO
ALTERED
BINARY
BLOCKS
BUFFER
CODE-SET
COLLATING
COMPRESSED
CONVERSION
CURSOR
DIRECT-ACCESS
DISPLAY-WS
ERASE
FAC
FIGURATIVE-CONSTANTS
INSPECT
MEMBER
MODIFY
MODIFIABLE
NATIVE
NODISPLAY
NO-MOD
OBJECT
ONLY
ORDER-AREA
PFKEY
PFKEYS
PROTECT
RANGE
ROLL
ROW
SEQUENCE
SETTING
SHARED
STANDARD-1
TIMES
VOLUME
WANG-VS

PART 2 - Reserved Words Recognized by IBM, but not Wang Compiler

ACTUAL
APPLY
BASIS
BEGINNING
CBL
CHANGED
COMP-1
COMP-2
COMP-3
COMP-4
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTATIONAL-4
CONSOLE
CORE-INDEX
CSP
CURRENT-DATE
CYL-INDEX
C01
C02
C03
C04
C05
C06
C07
C08
C09
C10
C11
C12
DEPTH
DISPLAY-ST
DISP
EJECT
ENDING
ENTRY
EXHIBIT
FILE-LIMIT
FILE-LIMITS
GOBACK
ID
INSERT
LABEL-RETURN
MASTER-INDEX
MORE-LABELS
NAMED
NOMINAL
NOTE
NSTD-REELS
OTHERWISE
PASSWORD
POSITIONING
PRINT-SWITCH
RECORD-OVERFLOW
RECORDING
RELOAD
REMARKS
REORG-CRITERIA
REREAD
SEEK
SERVICE
SKIP1
SKIP2
SKIP3
SORT-CORE-SIZE
SORT-FILE-SIZE
SORT-MESSAGE
SORT-MODE-SIZE
SORT-OPTION
SYSIN
SYSIPT
SYSLST
SYSOUT
SYSPCH
SYSPUNCH
S01
S02
TALLY
TIME-OF-DAY
TOTALED
TOTALING
TRACK
TRACK-AREA
TRACK-LIMIT
TRACKS
TRANSFORM
UPSI-0
UPSI-1
UPSI-2
UPSI-3
UPSI-4
UPSI-5
UPSI-6
UPSI-7
WHEN-COMPILED
WRITE-ONLY
WRITE-VERIFY

APPENDIX E
COLLATING SEQUENCES

EBCDIC Collating Sequence			ASCII Collating Sequence		
1.		(space)	1.		(space)
2.	.	(period)	2.	"	(quotation mark)
3.	<	(less than)	3.	$	(currency symbol)
4.	(	(left parenthesis)	4.	'	(apostrophe, single quotation mark)
5.	+	(plus symbol)			
6.	$	(currency symbol)	5.	(	(left parenthesis)
7.	*	(asterisk)	6.	)	(right parenthesis)
8.	)	(right parenthesis)	7.	*	(asterisk)
9.	;	(semicolon)	8.	+	(plus symbol)
10.	-	(hyphen, minus symbol)	9.	,	(comma)
11.	/	(stroke, virgule, slash)	10.	-	(hyphen, minus symbol)
			11.	.	(period, decimal point)
12.	,	(comma)	12.	/	(stroke, virgule, slash)
13.	>	(greater than)	13-22.		0 through 9
14.	'	(apostrophe, single quotation mark)	23.	;	(semicolon)
			24.	<	(less than)
15.	=	(equal sign)	25.	=	(equal sign)
16.	"	(quotation mark)	26.	>	(greater than)
17-42.		A through Z	27-52.		A through Z
43-52.		0 through 9			

WANG

Customer Comment Form

Title VS COBOL CONVERSION GUIDE

Publications Number 800-1204CC-02

Help Us Help You . . .

We've worked hard to make this document useful, readable, and technically accurate. Did we succeed? Only you can tell us! Your comments and suggestions will help us improve our technical communications. Please take a few minutes to let us know how you feel.

How did you receive this publication?

- ☐ Support or Sales Rep
- ☐ Wang Supplies Division
- ☐ From another user
- ☐ Enclosed with equipment
- ☐ Don't know
- ☐ Other _____

How did you use this Publication?

- ☐ Introduction to the subject
- ☐ Classroom text (student)
- ☐ Classroom text (teacher)
- ☐ Self-study text
- ☐ Aid to advanced knowledge
- ☐ Guide to operating instructions
- ☐ As a reference manual
- ☐ Other _____

Please rate the quality of this publication in each of the following areas.

	EXCELLENT	GOOD	FAIR	POOR	VERY POOR
Technical Accuracy — Does the system work the way the manual says it does?	☐	☐	☐	☐	☐
Readability — Is the manual easy to read and understand?	☐	☐	☐	☐	☐
Clarity — Are the instructions easy to follow?	☐	☐	☐	☐	☐
Examples — Were they helpful, realistic? Were there enough of them?	☐	☐	☐	☐	☐
Organization — Was it logical? Was it easy to find what you needed to know?	☐	☐	☐	☐	☐
Illustrations — Were they clear and useful?	☐	☐	☐	☐	☐
Physical Attractiveness — What did you think of the printing, binding, etc?	☐	☐	☐	☐	☐

Were there any terms or concepts that were not defined properly? ☐ Y ☐ N If so, what were they? _____

After reading this document do you feel that you will be able to operate the equipment/software? ☐ Yes ☐ No ☐ Yes, with practice

What errors or faults did you find in the manual? (Please include page numbers) _____

Do you have any other comments or suggestions? _____

Name _____ **Street** _____

Title _____ **City** _____

Dept/Mail Stop _____ **State/Country** _____

Company _____ **Zip Code** _____ **Telephone** _____

Thank you for your help.

Printed in U.S.A. 14-3140 3-82-5C

WANG

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 16 LOWELL, MA

POSTAGE WILL BE PAID BY ADDRESSEE

WANG LABORATORIES, INC.
CHARLES T. PEERS, JR., MAIL STOP 1369
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Fold

Cut along dotted line.