Advantage® VISION:Forms™

Reference Guide

3.0.7

Computer Associates®

B02444-1E

Contents

Chapter 1: Getting to Know VISION:Forms

Chapter 2: Basic Concepts of VISION:Forms

Chapter 3: Writing Verbs

Chapter 4: VSE System Information

Chapter 5: MVS System Information

Chapter 6: Special Procedures

Chapter 7: Advanced Techniques

Chapter 8: Dictionary Maintenance

Appendix A: Examples

Appendix B: Error Diagnostics

Glossary

Index

Chapter

1

Getting to Know VISION:Forms

How to Use this Manual

There are eight chapters and three appendixes in this manual. Each section has an introduction that explains its purpose.

We suggest that you completely read the remainder of this chapter before you attempt to write your first Advantage VISION:Forms program. After the chapter introductions, Chapter 1 contains a brief overview of VISION:Forms concepts, coding, punctuation, constants, and sequence.

The chapter "**Basic Concepts of VISION:Forms**" contains a general discussion of VISION:Forms reserved words, a description of how reports are composed, information on how VISION:Forms can be used with other languages, and a summary of VISION:Forms verbs.

The chapter "**Writing Verbs**" contains a detailed description of each verb and how it is coded. We suggest that you read the first paragraph of each verb to familiarize yourself with its main purpose. When you are ready to write a VISION:Forms program, read the detailed coding rules.

The chapter "**VSE System Information**" contains system information and installation instructions for VSE users.

The chapter "**MVS System Information**" contains system information and installation instructions for MVS users.

The chapter "**Special Procedures**" describes special procedures that can be used when dealing with DL/I access, IMS/DB access, and procedure code blocks.

The chapter "**Advanced Techniques**" discusses advanced techniques involving multiple files and other VISION:Report/VISION:Forms areas.

The chapter "**Dictionary Maintenance**" discusses how to build and maintain a VISION:Forms dictionary.

The appendix "**Examples**" contains example programs that illustrate requests ranging from simple listing and summary reporting to the use of TOTALS, BREAKS, and CALC statements.

The appendix "**Error Diagnostics**" lists and explains each of the error messages VISION:Forms might generate.

The Glossary contains a list of VISION:Forms terms.

Data Organization Concepts

This section sets the scene in which VISION:Forms operates, and through analogy, familiarizes you with some basic data processing concepts.

- What is a file?

A deck of playing cards is a file.

- What is a record?

Each card in a deck of playing cards is a record, making a file of 52 records.

- What is an element (or data name)?

Each card in a deck of playing cards has two elements: VALUE and SUIT.

If you were asked to pull all HEARTS from a deck of playing cards, you would look at the SUIT of the first card, and, if it were a HEART, you would SELECT it. If it were not, you would move on to look at the next card. This cycle would continue until you had examined each card in turn and SELECTed all HEARTS.

You then might be asked to rearrange the HEARTS into sequence from Ace through King. You would, of course, simply look at each card and regroup the cards in sequence.

As the final step you might be required to record on paper the SUIT and VALUE of each card. You would look at the first card and write what you saw. You would move on to the next card and write what you saw. You would continue this routine until there were no more SUITS and VALUES to record.

The following VISION:Forms code would do the same series of steps:

```
FILENAME CARDDECK
SELECT SUIT EQUAL-TO 'HEARTS'
SORT VALUE
PRINT SUIT VALUE
```

When you write FILENAME CARDDECK, you advise the computer what particular set of data is of interest to you.

When you write SELECT SUIT EQUAL-TO 'HEARTS', you cause the computer to read each record in CARDDECK in turn and examine the SUIT field for HEARTS. When a record is HEARTS, a copy of that record is written to a newly created temporary file (which we will call SELECTED). This cycle continues until all records in the file have been examined.

VISION:Forms does NOT change the original file in any way.

By coding SORT VALUE, you tell the computer that the selected file is to be arranged based on the contents of the value field. The SORT program reads the selected file and rearranges it so that the ace is the first record, the deuce is the second record, and so on. The PRINT SUIT VALUE statement causes the computer to read each record on the selected file, and print the contents of the suit and values fields.

You are now aware of what the computer does when your VISION:Forms code is processed. The nature of the data changes with each VISION:Forms program but the SELECT-SORT-REPORT steps taken are always those described above.

VISION:Forms Coding Format

There is no coding form for VISION:Forms. An 80-position framework is used with the following requirements:

- Enter VISION:Forms verbs followed by keywords as appropriate for that verb in positions 1 through 71. VISION:Forms statements can consist of multiple VERBS in one statement, one verb per statement, or one verb (and associated keywords) across several statements. All of the following are valid:

```
FILENAME PERSONEL JOBNAME LISTEMPL REQUESTOR KELLY

PRINT PLANT
      DIV
      EMPNR
      NAME
PRINT PLANT DIV EMPNR NAME
```

- Position 72 is for the continuation indicator. If a particular word or constant cannot be completely coded by position 71, place a character in position 72 and continue the word in position 1 of the next line. The non-blank character in position 72 signals VISION:Forms to attach the start of the next line's data to the end of the current line's data.

  If position 72 is blank, it is presumed that the line ends with a complete word and the next line starts with a new word.

- Positions 73-80 are not referenced by VISION:Forms. They are available for use as sequence numbers, identification, etc. for each installation.

Writing a VISION:Forms Program

The following is an example of coding a VISION:Forms program. Your data processing department has assigned a file name to each file that you can reference and has assigned a data name to each element of data in the records of each file. These assignments have been built into the computer in the form of a VISION:Forms dictionary. A listing of the dictionary has been provided.

You communicate your report needs to VISION:Forms by writing verbs that specify what you want to do, and data names to indicate on what data you want it performed.

When your VISION:Forms program is read into the computer, the program is executed and it analyzes your verbs and data names to determine your requirements. Based on this analysis, a computer program is created and executed to produce your report.

The following hypothetical file layout can help to explain some of these concepts:

```
FILE NAME: PERSONEL

DATA NAME: DIV PLANT DEPT EMP-NR     NAME      PAY-GRADE

SAMPLE     02   01   10  1754    JONES, JOE     6
CONTENTS

            DT-EMPLYD   ED-LVL  NR-DEP    MO-DED

              880411     12       2       123.45
             (YYMMDD)

         ************* DICTIONARY ENTRIES *************

            DATA NAME          RECORD LOCATION

             DIV                  0001-0002
             PLANT                0003-0004
             DEPT                 0005-0006
             EMP-NR               0007-0010
             NAME                 0011-0040
             PAY-GRADE            0041-0041
             DT-EMPLYD            0042-0047
             ED-LVL               0048-0049
             NR-DEP               0050-0050
             MO-DED               0051-0057-P
```

Through examples of increasing complexity, we now briefly introduce you to the VISION:Forms verbs.

1 If you want to print a list of everyone in the company you need only identify the file and what you want to print from each record:

```
FILENAME PERSONEL
PRINT NAME
```

This illustrates the FILENAME verb that you must code into every VISION:Forms program to identify which file you are interested in, and the PRINT verb that you use to specify what data you want to appear in the report. Data names are generally used as report column headings at the top of each report page.

2 Should you want to list only the employees in department 12 you would code:

```
FILENAME PERSONEL
SELECT DEPT EQUAL-TO '12'
PRINT NAME
```

The SELECT verb examines each record in the file and ignores those that do not satisfy your selection criteria. In this example, the DEPT field in each record would be examined to see if it contains 12. If a 12 is present, the record proceeds to your VISION:Forms logic; if a 12 is not present, the program skips to the next record.

3 There can be occasions when you want to report data that is not available in the record but can be calculated. Let's say you want to print annual salary deductions but only monthly deductions (MO-DED) are available. By adding a CALC verb to the program we can do the necessary arithmetic:

```
FILENAME PERSONEL
SELECT DEPT EQUAL-TO '12'
CALC AN-DED EQUALS MO-DED TIMES 12
PRINT NAME AN-DED
```

4 A file of this type would normally have the records grouped by department number and by employee number within department. The report we have described to this point would print out in employee number sequence. As often as not, the required sequence for the report is not the standard sequence of the file as described above (department number, employee number). Suppose, in this case, you must produce a report in pay grade sequence. You can use the SORT verb:

```
FILENAME PERSONEL
SELECT DEPT EQUAL-TO '12'
CALC AN-DED EQUALS MO-DED TIMES 12
SORT PAY-GRADE
PRINT NAME AN-DED
```

The SORT verb rearranges selected records based on the contents of the field you specify; in this case, the sorted file would start with those records that contain pay-grade 1. Those with pay-grade 2 and so forth would follow these. Descending sequence SORT is available.

5 If your report must be grouped by pay grade, it might be necessary to tear the report listing apart between pay grades so that several clerks can each work on a separate pay grade grouping. You could cause each pay grade to start at the top of a new page by coding NEWPAGE:

```
FILENAME PERSONEL
SELECT DEPT EQUAL-TO '12'
CALC AN-DED EQUALS MO-DED TIMES 12
SORT PAY-GRADE
NEWPAGE PAY-GRADE
PRINT NAME AN-DED
```

6 Up to this point we have been planning a straightforward listing of the contents of selected records. In many cases, however, you might want to total counts and/or amounts and print out those totals at the end of certain groups. Two verbs are available:

```
TOTAL DATANAME
```

When you code a TOTAL verb followed by data names you advise VISION:Forms to keep a running TOTAL of the contents of the data name field(s) in each selected record.

```
BREAK DATANAME
```

A break occurs when the computer recognizes that the contents of a data name field have changed. This recognition happens after the last record of group 1 has been processed and before the first record of group 2 is processed.

VISION:Forms uses these two verbs together to achieve reporting with totals. This is illustrated in the following example:

```
FILENAME PERSONEL
SELECT DEPT EQUAL-TO '12'
CALC AN-DED EQUALS MO-DED TIMES 12
SORT PAY-GRADE
NEWPAGE PAY-GRADE
PRINT NAME AN-DED MO-DED
TOTAL AN-DED MO-DED
BREAK PAY-GRADE
```

Remembering that the selected records have now been grouped by pay grade, this example keeps running totals on AN-DED and MO-DED. When the contents of PAY-GRADE change (BREAK), pause and print the totals for the finishing PAY-GRADE group and get ready to start running totals for the next PAY-GRADE group.

7 As discussed in the PRINT verb coverage, your data names are used as report column headers that appear at the top of each report page. You could also want an overall title:

```
TITLE 'PAY GRADE DEDUCTION SUMMARY'
```

When this statement is included in your VISION:Forms program, the first line of each report page appears as:

```
02/26/94         PAY GRADE DEDUCTION SUMMARY         PAGE 1
```

The date, the word PAGE, and page number are automatically placed around the actual title, which is centered. The date and page are inserted if there are sufficient blank spaces surrounding the title.

The following two verbs, QWTABLE and EQUATE, have been left to the end of this primer as they are ordinarily used by the computer specialists in your data processing department. They are covered broadly here and in detail in the chapter "**Writing Verbs**" (see **EQUATE** and **QWTABLE** in the chapter "Writing Verbs").

QWTABLE

There are cases where your selection criteria values change from one execution of a VISION:Forms program to the next. In the personnel example, you might have 60 departments in your company; each time you run the job, you want the same report, but on a different group of departments. To achieve this, you would enter table statements, one statement per department to report on, and would code your VISION:Forms program to read:

```
FILENAME PERSONEL
SELECT DEPT ON-TABLE
CALC AN-DED EQUALS MO-DED TIMES 12
SORT PAY-GRADE
NEWPAGE PAY-GRADE
PRINT NAME AN-DED MO-DED (MONTHLY-DEDUCTIONS)
TOTAL AN-DED MO-DED
BREAK PAY-GRADE
TITLE 'PAY GRADE DEDUCTION SUMMARY'

QWTABLE 0010 01 02
TABLE DATA
07
12
15
36
47
```

The QWTABLE statement defines the table data; it has a maximum of 10 entries, with the department number in position 1 for a length of 2. The program shown above would print a report that includes departments 07, 12, 15, 36, and 47. If tomorrow you want the same report on departments 14, 20, and 21, you need only replace the table statements shown in the example with the values 14, 20, and 21.

EQUATE

The dictionary entries describing a computer file and assigning data names to data elements could occasionally not go into enough detail to allow reference to a supplement, i.e., DT-EMPLYD is too broad a definition of date employed to allow you to reference the year employed. The EQUATE verb allows you to insert a temporary data name assignment into a particular VISION:Forms statement. You should use the EQUATE verb only after file format consultation with your data processing department. For a data name describing the year employed, the following EQUATE would be used.

```
EQUATE DT-EMPLYD-YR INF0042-0043
```

You could then use this data name in a VISION:Forms program:

```
FILENAME PERSONEL
EQUATE DT-EMPLYD-YR INF0042-0043
SELECT DT-EMPLYD-YR GT '90'
PRINT NAME DT-EMPLYD DEPT
```

Punctuation

You can code commas and periods within VISION:Forms statements. For example:

```
SELECT DEPT EQ '10', '12', '15'.
PRINT DEPT, PLANT, UNIT-PRICE.
```

Since verbs, data names, and keywords cannot include commas or periods, VISION:Forms treats commas and periods at the end of a word as cosmetic punctuation.

Constants

You will note that throughout the coding examples, constants are sometimes shown within single quotation marks:

```
SELECT CITY EQ 'N Y C'
```

The quotation marks are necessary to allow description of any possible combination of characters (including blanks) that is normally used as beginning and end of word indicators. Upon reading a quotation mark, VISION:Forms treats all following data up to the ending quotation mark as one value. If you must code a quotation mark as part of the constant, you should code two adjacent quotation marks. VISION:Forms recognizes this particular combination and generates a single quotation mark within the constant:

'' generates as '

```
'O''NIELL' generates as O'NIELL
```

A quick scan of some of the examples indicates that where no quotation marks are shown around a constant, it is always a number that is an unbroken string of digits, with the possibility of recognizable sign and/or decimal points.

In many cases, numbers are present in your files but are used as characters (as in characters of the alphabet). Consider part number, zip code, plant, and department number. These 'numbers' would not normally be used for arithmetic but are simply a 10-character (0-9) extension of the alphabet and should be treated that way when coding VISION:Forms programs.

```
SELECT DEPT-NUMBER EQ '32'
```

If the nature of a particular number field is in doubt, you should refer to the dictionary definition for that data name. If the definition ends with -P, the constants referring to that data name should not be surrounded by quotation marks; otherwise, quotation marks are required:

```
     DICTIONARY                        CODING

DEPT-NR      INF1-2         SELECT DEPT-NR EQ '32'
UNIT-PRICE   INF6-8-P       SELECT UNIT-PRICE GT 150.00
```

What Files Can Be Accessed?

VISION:Forms can access any standard sequential file, sequential ISAM file, or VSAM file (KSDS, ESDS, or RRDS), with no special coding required. VISION:Forms generates whatever code is necessary to read the file in sequence, one record at a time.

To access other file structures, you must code general purpose procedures (or blocks of VISION:Report code) to do the actual reading of the file. This method applies to DB2 or SQL/DS, random VSAM (KSDS, ESDS, or RRDS), DL/I, TOTAL, random ISAM, etc.

Statement Sequence

The FILENAME statement must be your first verb so that VISION:Forms can establish connection with the dictionary section for the file name.

The QWTABLE verb, table data statement, and actual table entries must be at the end of your statements.

The CALC - SELECT - NEWTEST series of statements should be coded as a unit in:

```
CALC-SELECT
```

- A CALC result field must be CALCed into a data name before attempting to reference that field with SELECT or other statements.

```
SELECT-NEWTEST-SELECT
```

- The NEWTEST verb signals a logical break between the preceding and following SELECT statements and must be positioned between them.

All other verbs can be written in any sequence. The only exception to this rule is the JOBNAME, REQUESTOR, and FILENAME verbs, which must be first. The above three verbs can be in any sequence.

Contacting Computer Associates

For further technical assistance with this product, contact Computer Associates Technical Support on the Internet at esupport.ca.com. Technical support is available 24 hours a day, 7 days a week.

Chapter

2

Basic Concepts of VISION:Forms

VISION:Forms Reserved Words

The words in the following list should not be used as data names or otherwise, except where they are called for in the context of the various verbs. These words, when encountered in the proper context, act as signals of your intentions and, conversely, when encountered out of context, cause unpredictable and undesirable results.

=
*
/
+
-
>
<
ATTACH
AVERAGE
BLANK
BREAK
CALC
DEC
DESCENDING
DIV-BY
DOUBLE
ELECT
EQ
EQU
EQUALS
EQUAL-TO
EQUATE
FILENAME
GREATER-THAN
GROUPCOUNT
GT
HEADLINE
INCLUDING
JOBNAME
LESS-THAN
LIMIT
LT
MATCH
MINUS
MSELECT
NEGATIVE
NEWPAGE
NEWTEST
NOT
NUMERIC
ONBREAK
ONE
ON-TABLE
ONTABLE
OR
PAGEDEPTH
PAGEWIDTH
PARTITION
PERCENT
PLUS
POSITIVE
PRINT
PROCEDURE
PSELECT
QJSIZE
QWOPTION
QWTABLE
RANGE
REQUESTOR
RND
SAMPLE
SELECT
SORT
SORTSIZE
SPACE
TIMES
TITLE
TITLE2
TOTAL
TRIPLE
UNDER
USE
WHEN
ZERO

Report Format - How VISION:Forms Composes Your Report

The sequence of the data names on the PRINT statement determines in what order the data is placed on the report. VISION:Forms places the data on the report from left to right as coded in the PRINT statement.

The first consideration is whether the data to be printed fits on one print line with a minimum of one space between fields. If print space is not available, VISION:Forms issues error message 062 followed by a list of data names and the print space required for each. You must drop data names from the PRINT statement until the data fits. Normal print space available is 132 positions.

Once assured that the data fits, VISION:Forms analyzes each data name (or alternate column header) in the PRINT statement and organizes it as a vertical column header of up to 5 lines. If the data to be printed is alphabetic, each header line is centered over the data. If the data to be printed is numeric, the header lines are right-aligned over the unit's position of the data. To illustrate these principles:

```
DATA NAME (OR          ALPHA OR            REPORT
ALTERNATE HEADER)       NUMERIC            FORMAT
MARITAL-STATUS            ALPHA              MARITAL
                                             STATUS
                                                x
CUSTOMER-NAME-AND-ADDRESS  ALPHA              CUSTOMER
                                               NAME
                                                AND
                                        ......ADDRESS......
                                          XXXXXXXXXXXXXXXXXX
PRICE-PER-THOUSAND         NUMERIC                PRICE
                                                    PER
                                                ..THOUSAND
                                                XXX,XXX.XX
```

The periods in the header are automatically generated out to the boundaries of the data field as an aid.

You can control the header composition by the characters you include:

- To force a specific format, code the header exactly as you wish it to appear, including blanks, with a comma to indicate the end of each line:

```
(T, I,  T,   L,    E) composes as:      T
                                         I
                                          T
                                           L
                                            E
```

- If no commas are found by VISION:Forms, it then scans for embedded blanks in the header. If any are found, it assumes that blanks are to be considered as end of line indicators:

```
(CUSTOMER NAME-AND ADDRESS) composes as:            CUSTOMER
                                                    NAME-AND
                                                    ADDRESS
```

- Finally, if no blanks are found, any hyphens are considered end of line indicators:

```
(CUSTOMER-NAME-AND-ADDRESS) composes as:
            CUSTOMER
                                                    NAME
                                                     AND
                                                  ADDRESS
```

When header print space requirements are too long for the print line, VISION:Forms isolates fields whose headers are longer than data and truncates these on the right, if necessary, until those headers are no longer than data.

Writing VISION:Forms Code in a Language Other Than English

As a special feature, VISION:Forms code can be written in French and German.

Translation from the alternate language to English is activated by an entry in the ENVIRON record in the VISION:Forms dictionary for the installation. By coding FR in positions 42-43 of the ENVIRON record, all VISION:Forms code is translated from French to English. By coding GR, all VISION:Forms code is translated from German to English.

A detailed record layout and coding requirements for the ENVIRON record are available in the chapter "**VSE System Information**" and the chapter "**MVS System Information**" of this manual.

When translation has been specified, each word in each VISION:Forms statement is examined for presence of an ALTERNATE/ENGLISH translation table. If an input word is found on the table, it is immediately replaced by the English translation, which is used for the balance of processing.

If an input word does not appear in the translation table, that word is used as is. This allows individual users to code VISION:Forms:

- Entirely in an alternate language, that is, French or German.
- As a mixture of foreign and English words.
- Entirely in English.

Certain VISION:Forms functions must be specified in English as they are processed before the translation starts:

START-DICTIONARY	Enables you to include dictionary entries with a VISION:Forms program.
END-DICTIONARY	Concludes the dictionary entries included with a VISION:Forms program.
QWOPTION	Modifies certain VISION:Forms functions.

Summary of VISION:Forms Verbs

Verb	Description/Example
ATTACH	Accesses data from other areas Ex. ATTACH WST3-5 AS COUNT
AVERAGE	Calculates and prints averages Ex. AVERAGE MONTHLY-AMT YEARLY-AMT
BREAK	Totaling control Ex. BREAK DIV PLANT
CALC	Arithmetic Ex. CALC AN-DED = MO-DED * 1.25
EQUATE	Temporary data names Ex. EQUATE CUSTID INF1-2
FILENAME	Name of user file Ex. FILENAME PERSONEL
JOBNAME	VSE JOB name Ex. JOBNAME ACCNT235
LIMIT	Selection and report size Ex. LIMIT REPORTING OUTPUT 50
NEWPAGE	Forces new page Ex. NEWPAGE DEPT-NUM
NEWTEST	Starts new selection criteria Ex. NEWTEST
ONBREAK	Total time calculation Ex. ONBREAK DEPT FINAL
PAGEDEPTH	Number of print lines per page Ex. PAGEDEPTH 80

Verb	Description/Example
PAGEWIDTH	Number of print columns to a page Ex. PAGEWIDTH 66
PARTITION	VSE partition for processing Ex. PARTITION BG
PRINT	Specifies report format Ex. PRINT DEPT NAME ADDRESS
PSELECT	Selection when printing Ex. PSELECT NAME EQ 'JONES'
QJSIZE	VSE partition size Ex. QJSIZE 120
QWOPTION	Overrides control Ex. QWOPTION SORTWORK=03
QWTABLE	Defines variable data Ex. QWTABLE DICTIONARY PAYROLL
REQUESTOR	VSE job comment Ex. REQUESTOR RON
SAMPLE	Sampling of input Ex. SAMPLE EVERY 10
SELECT	Determines what data to use Ex. SELECT PART-NUM EQ '12C'
SORT	Rearranges records Ex. SORT DEPT-NUM
SORTSIZE	VSE SORT size Ex. SORTSIZE 42
SPACE	Report spacing Ex. SPACE DOUBLE
TITLE	Names the report Ex. TITLE 'EMPLOYEES IN PLANT 10'
TITLE2	Second line of report heading Ex. TITLE2 'ABC COMPANY REPORT'
TOTAL	Accumulates numeric data Ex. TOTAL ANNUALS
USE PROCEDURE	Inserts VISION:Report code Ex. USE PROCEDURE SET-FLDS AT START-UP

Chapter

3

Writing Verbs

This chapter contains a detailed explanation of each verb that you can use in your VISION:Forms requests. In each section, the first paragraph explains the verb's function in a general way. Below the explanation are two or three examples of the verb in statement form. The verb structure is then outlined for each 'Word' that applies to that particular verb. OPT above the Word means the Word is optional in the statement. The entries under Word 1, Word 2, Word 3, etc. describe what can be included in that verb statement. 'Dataname' is the descriptive name from the VISION:Forms dictionary. Constant is a word or number that does not vary for that statement. Keywords (data names, constants, etc.) in lowercase are variables and should be replaced with the actual value of each keyword.

The coding rules give the details about the entries for each 'Word' in the VISION:Forms statement. In each verb statement, enter the verb followed by one or more of the entries described for Word 1, Word 2, Word 3, etc. Examples of VISION:Forms programs and the report produced by the programs follow the coding rules for some verbs.

Note: If no PRINT, TOTAL, or AVERAGE statements are coded, VISION:Forms just generates an output file QWOFA. This extracted file can then be used in other jobs.

In order to become familiar with the format of VISION:Forms programs, read each verb's general paragraph and study the examples. When writing your first VISION:Forms programs, refer to the coding rules for allowable entries.

ATTACH

The ATTACH verb allows you to set up a new field from any source (VISION:Report area, table area, or another file) so that this new field can be referenced later in your VISION:Forms program. This field can be composed of one or many existing fields. You can also code edit masks for the field if it will be printed.

Examples: Simple Version

```
ATTACH WST1-50 AS CO-NAME
ATTACH BILL-AMT AS NEW-AMT
ATTACH 'P#12-34B-A3' AS PART-NR     /* Alphanumeric literal
ATTACH 45439 AS ZIP                 /* Numeric literal
ATTACH ZERO AS NEW-AMT USING DEC2C
```

Examples: Complex Version

```
ATTACH 'ACCOUNTING' + DEPT-CODE + SUPERVISOR-NAME AS TABLEKEY
ATTACH '5134359514' AS PHONE-NO USING T
ATTACH '513' + '435' + '9514' AS TELEPHONE USING T
```

Verb Structure

```
                                      OPT             OPT
   Word       Word      Word      Word              Word
     1          2         3         4                 5
  Dataname      AS    Dataname   USING              DECncp
  QJ Fld Def                                  Special Mask Code
  'literal'
  ZERO(ES)
  BLANK(S)
```

or

```
                              OPT          OPT             OPT
 Word      Word     Word     Word         Word            Words
   1         2        3        4            5              6-25
Dataname     +     Dataname     +        Dataname        Same as
QJ Fld Def plus   QJ Fld Def  PLUS      QJ Fld Def        Words
'literal'         'literal'             'literal'         4 & 5
                              USING...Special Mask Code
                              AS......Dataname
```

Coding Rules

Simple Version: Attaching Only One Data Item

WORD

1 Code the area of information that is ATTACHed to the end of the output record. One of the following entries can be coded for Word 1.

Dataname	The contents of 'Dataname' is moved to the ATTACH result.
QJ Fld Def	Code a VISION:Report field definition whose contents are placed in the ATTACHed field.
'literal'	Literals can be coded in either of two ways. If the literal contains both letters and numbers, code the literal within single quotation marks such as 'A123-A7'. This form of literal is ATTACHed in character (or EBCDIC) format. However, if the literal contains only numbers, do not enclose it in single quotation marks. These literals are ATTACHed in a packed format. ■ You should be cautioned that packed and character literals cannot be ATTACHed as the same data name. Once a certain data name has been ATTACHed, VISION:Forms recognizes the format, either packed or character. All other ATTACH statements for that same data name must honor the same format.
ZERO(ES)	This keyword causes zeros to be moved to the ATTACHed data name. If this is coded in the first ATTACH of a data name, the field format is packed decimal; otherwise, the format does not matter.
BLANK(S)	This keyword causes blanks (or spaces) to be moved to the ATTACHed data name. The field format must be EBCDIC.

2 You can optionally code the word 'AS', in order to improve the ease of reading the VISION:Forms program.

3 You must code a data name that is unique. This data name can be referred to in any VISION:Forms statement that follows. The maximum length is 14 characters. The only way the data name can be used prior to this ATTACH statement is in another ATTACH statement.

4 If you need to code an edit mask in Word 5, you must code the keyword USING. Normally, you need not code edit masks since, if Word 1 is a data name and if it has an edit pattern already assigned to it, that same pattern is also assigned to the new data name.

5 This optional field indicates report decimal/comma specifications or special editing. No other words can be coded after this one. One of the following entries can be coded for Word 5.

1. DECncp: This keyword is used to assign the number of decimal positions for printing and calculations. This option cannot be used with the complex version of ATTACH. It is used to indicate the type of editing to be performed. The keyword DEC is coded, followed by one or more of characters 'n', 'c', 'p'.

   'n' is used to indicate the number of decimal positions in the result field. The standard generation of decimal positions is zero. To indicate the number of decimal positions, code DEC followed immediately by a digit from 0 through 9.

   'c' is used to indicate the type of editing that should be performed upon this field if it is calculated. The standard generation is a space, zero suppression, and no commas. To vary the type of editing, use one of the following valid entries for 'c':

S	Zero suppression and no commas. This only needs to be coded if the number of decimal positions for printing ('p') is coded.
C	Zero suppression and commas.
E	European punctuation.
N	No zero suppression or commas. Decimal point is included, if necessary.

   'p' is used to indicate the number of decimal positions printed. Normally the number of decimal positions printed is the same as the number in the result field. If you have a need to use a different number of decimal positions for printing, code a digit of 0 through 9 in the 'p' position.

   If 'p' is used, 'n' and 'c' must also be used.

2. Code any of the special edit codes A to Z (except C, E, and N). Your data processing department can advise you of what type of editing is performed by each code. Some of the standard edit codes provided by Computer Associates are:

Code	Edit Pattern	Possible Editing Uses
A	99/99	Dates
B	99.99	Time
D	99/99/99	Dates
F	99-99-99	Dates/Time
G	99.99.99	Dates/Time
L	999-9999	Telephone Numbers
T	999-999-9999	Telephone Numbers
U	(999) 999-9999	Telephone Numbers
S	999-99-9999	Social Security Numbers
Code	Edit Pattern	Possible Editing Uses

Complex Version: Attaching Several Data Names

WORD

1 Code the first field or area you need ATTACHed. It can be one of the following:

Dataname	Must be an EBCDIC field.
QJ Fld Def	Must be an EBCDIC field.
'literal'	Must be an alphanumeric literal.

2 Code the word 'PLUS' or '+' to indicate that the items in Words 1 and 3 are to be chained together (i.e., Word 3 data is placed directly following Word 1 data).

3 Code the next field or area you need ATTACHed. It can be any one of the following:

Dataname	Must be an EBCDIC field format.
QJ Fld Def	Must be an EBCDIC field.
'literal'	Must be an alphanumeric field.

4 through 25

Continue coding '+' or 'PLUS' followed by a data name to add to the length and value of the new field. The statement must end with the unique data name of the new field immediately preceded by the word 'AS'. If editing needs to be coded, the mask code must be the last word specified, preceded by the word 'USING'.

Example:

```
ATTACH ZIP + STATE + 'GOOD CUSTOMER' AS GOOD-CUST-KEY
```

ZIP is 5 bytes and STATE is a 2-byte code. The new field called GOOD-CUST-KEY would be 20 bytes long and ATTACHed to the end of the output record as follows:

```
45439OHGOOD CUSTOMER
```

The keywords '+' and 'PLUS' do not cause any addition of fields. They only cause the fields to be chained together in the same sequence as coded (concatenated).

AVERAGE

Within some reports, there are groups of data that are not useful when only totaled and printed. In cases like this, an average must be printed. The AVERAGE verb enables you to specify data names whose contents are to be averaged and then printed whenever a control field changes (see BREAK in this chapter). These averages are printed after the detail information and totaled fields, if any, are requested.

Examples:

```
AVERAGE SALARY AGE
AVERAGE CALC-DED MO-DED AN-DED WAGE-RT
```

Verb Structure

OPT Word 1	OPT Word 2	OPT Word 3	OPT Word 4	OPT Word 5	Word 6-15
Dataname	Dataname	Dataname	Dataname	Dataname	Dataname

Coding Rules:

Enter the 'Dataname' to be averaged. Data name(s) need not be in the PRINT statement or the TOTAL statement to be AVERAGEd.

If the data name(s) to be averaged are all included in the PRINT HEADLINE statement or if the report is a summary report, the averages are printed under the respective column headings. But if any data names to be averaged are not on a PRINT statement, then all averages and data names are printed on a separate line.

BREAK

Many reports require that certain values be accumulated for groups of records and that accumulated totals be printed at the end of the group. The BREAK verb is the method by which you tell VISION:Forms that group changes are to be considered.

Note: It is valid to use a BREAK without a TOTAL verb. In this case, a blank line is printed each time a change occurs. But in order for VISION:Forms to actually total a data name, a TOTAL verb is required. See examples in the appendix "**Examples**."

The presence of a BREAK verb causes the computer to keep track of the contents of the specified fields and when the contents of the field changes from record A to record B (implying that record A was the last of a group), VISION:Forms pauses and prints the totals for the group just finished.

Examples:

```
BREAK NAME
BREAK NAME REP CODE STATE
```

Verb Structure

	OPT	OPT	OPT
Word	Word	Word	Words
1	2	3	4-9
Dataname	Dataname	Dataname	Dataname

Coding Rules:

WORD

1 Enter the 'Dataname' of the record field for which, when its contents change, totals are to be printed. This must be the lowest control level.

2 'Dataname' of a second field to be checked for breaks and which will cause TOTAL printing. Code the next control level here.

3 'Dataname' of the third field. Same policy as applies to Words 1 and 2. Code next control level here.

4 through 9

'Dataname'(s) of fourth through ninth control levels.

One BREAK verb statement can have nine levels or controls. A grand TOTAL is automatically prepared.

Example:

If a company is organized by departments within plants, and plants within divisions, then the personnel file would probably contain department, plant, and division fields and the file would likely be sequenced on these fields:

```
DIV   1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3
PLANT A A B B B R F F F F F F F M N N M N C C C C T T T
DEPT  1 2 1 2 2 6 2 2 2 2 2 7 7 7 3 4 4 4 4 1 1 1 1 3 3 9
```

If company management wants a report with totals for each of the three divisions, the requestor would specify:

```
BREAK DIV
```

This would cause three TOTAL lines to be printed, one at the end of the report section for each division. If totals were also desired at the plant level, you would code:

```
BREAK PLANT DIV
```

This would cause [8|10|13] total lines to print.

Finally, a third level of BREAK can be specified which, in this case, would be department:

```
BREAK DEPT PLANT DIV
```

This would cause 23 lines of totals.

CALC

The CALC verb allows you to specify that an arithmetic operation be performed on data that is present in a record. The results of that arithmetic operation are attached to the output record for further reference. You can also use CALC to create a new field and set it to a specific value.

Examples:

```
CALC BILLAMT EQUALS YR-BILL DIV-BY 12
CALC COMM DEC2C RND = .25 * BILLING-AN / 12
CALC TAX DEC4 = .04 * COMMISSION
```

Verb Structure

```
          OPT     OPT     OPT                              OPT     OPT      OPT
Word     Word    Word    Word    Word    Word    Word     Word    Word     Words
  1        2       3       4       5       6       7        8      9-26

Dataname DECncp  RND  DIGITSnn  EQUALS  Dataname +    Dataname same as
         PERCENT                =       Constant PLUS Constant  Words
                                EQ      ZERO     -               7 & 8
                                        ONE      MINUS
                                                 *
                                                 TIMES
                                                 /
                                                 DIV-BY
```

Coding Rules:

WORD

1 Assign a 'Dataname' to the arithmetic result field and code it here. The assigned data name must:

- Start with one of the letters of the alphabet.
- Not be a VISION:Forms reserved word.
- Not already be present in the dictionary for this file.
- Not be longer than 14 characters, including hyphens.

2 Only one of these optional words can be used. If you do not use one of the optional words, VISION:Forms forces the CALCed field to have the same number of print decimals as the field with the largest number of print decimals. The editing character defaults to a space which means the field, when printed, has zero suppression and no commas.

DECncp: This keyword is used to override the decimal positions for printing and calculations. It can also be used to indicate the type of editing to be performed. The keyword DEC is coded, followed by one or more of the characters 'n', 'c', 'p'.

'n' is used to override the number of decimal positions in the result field. The standard generation of decimal positions is explained later in this section. Decimal position override is invoked by coding DEC, followed immediately by a digit from 0 through 9.

'c' is used to indicate the type of editing that should be performed upon this field if it is printed. The field would normally be edited with zero suppression, commas, and a decimal point. The valid entries for 'c' are:

S Zero suppression and no commas. This only needs to be coded if the number of decimal positions for printing 'p' is coded.

C Zero suppression and commas.

E European punctuation.

N No zero suppression or commas. Decimal point is included if necessary.

If 'c' is used, 'n' must also be used.

'p' is used to override the number of decimal positions that are printed. Normally the number of decimal positions printed is the same as the number in the result field. If you need to use a different number of decimal positions for printing, code a digit of 0 through 9 in the 'p' position.

```
CALC DIFF DEC5C3 = LIST - SALE / LIST
```

In the above example, you are computing the percent of LIST to SALE. The result must have 5 decimal positions, but it should print in a standard format for a percent field. If the result was 0.32678, it would print as 32.678.

If 'p' is used, 'n' and 'c' must also be used.

PERCENT: This keyword can be used to cause a CALCed data name to be printed in the format of xxx.xx. This is very useful when computing a percentage and the result field is large (in print space requirements) but the actual result is less than 999.99. If the result exceeds 999.9 or -999.9, VISION:Forms automatically prints 999.9 or -999.9.

Assume LIST and SALE are 7 digits long with 2 decimal positions. We want to compute the percentage of LIST to SALE.

```
CALC LIST PERCENT = LIST - SALE / LIST
```

The LIST result field is 10 digits long with 3 decimal positions. Because the PERCENT keyword was used, only the last 4 positions of the result field are printed, with 1 decimal position. If the result was .192, it would be printed as 19.2

The result is automatically rounded.

3 When the CALC arithmetic is multiply or divide, you can round the product or quotient, respectively. If this is the case, code the word RND at this point. This keyword is not necessary if the keyword PERCENT was used.

4 Code the word DIGITS followed immediately by a number between 1 and 15 to control the size of the CALCed result field. This word is optional and should be coded only when you know the exact requirements of the field. If this field is coded and the size is too small, incorrect results can occur.

5 Enter the word EQUALS or an = sign.

6 Code the data name of the record field to be operated on in accordance with Words 7 and 8. Where appropriate, CALC result fields can be set to specific values with no arithmetic required:

Word 5	CAUSES
ZEROZERO	A CALC result of zero.
ONEONE	A CALC result of one.
Constant	The CALC result contains the numeric constant specified.
Dataname	The contents of Dataname is moved to the CALC result.

When CALC result fields are set to specific values with no arithmetic required, Words 7 and 8 should not be coded.

7 Enter one of the following words:

PLUS or +	Result field EQUALS Word 6 data plus Word 8 data.
MINUS or -	Result field EQUALS Word 6 data minus Word 8 data.
TIMES or *	Result field EQUALS Word 6 data multiplied by Word 8 data.
DIV-BY or /	Result field EQUALS Word 6 data divided by Word 8 data.

8 Enter a data name or constant to be applied to Word 6 data in one of the four appropriate functions shown above.

9 through 26

Using the rules for Words 7 and 8, enter, in pairs, additional arithmetic to be used to CALC the field.

```
CALC FACTORY-COST = LABOR + BURDEN + MATERIAL
CALC LIST% PERCENT = LIST - SALE / LIST
CALC YR-INCOME = SALARY + BONUS + COMMISSION * 12
```

VISION:Forms evaluates each CALC statement left to right. In the first example above, the same result could be obtained by coding:

```
CALC FACTORY-COST = LABOR + BURDEN
CALC FACTORY-COST = FACTORY-COST + MATERIAL
```

Because of this rule, you must be careful when writing a multi-field CALC statement.

```
CALC LIST% = LIST - SALE / LIST * 1.25
```

The above statement does not give the desired result because VISION:Forms would handle the statement as if it were written:

```
CALC LIST% = LIST - SALE
CALC LIST% = LIST% / LIST
CALC LIST% = LIST% * 1.25
```

To get the correct result, you should write:

```
CALC LIST%-A = LIST - SALE
CALC LIST%-B = LIST * 1.25
CALC LIST% = LIST%-A / LIST%-B
```

Always code CALC verbs ahead of any verbs that refer to the result.

Correct:

```
CALC ANNUAL-DED
SELECT ANNUAL-DED
```

Incorrect:

```
SELECT ANNUAL-DED
CALC ANNUAL-DED
```

Through documentation or in consultation with your data processing department you should confirm which data names in a file contain valid numeric data for arithmetic purposes. Should arithmetic be attempted on non-numeric data, VISION:Forms is aborted immediately.

Example:

Perform arithmetic and attach to record. Management wants a list of all employees whose annual deductions are greater than $2000.00. Only monthly deductions (MO-DED) are available in the record. This list is to include employee number, name, and amount of annual deductions.

```
CALC ANNUAL-DED EQUALS MO-DED TIMES 12
SELECT ANNUAL-DED GT 2000.00
PRINT EMPNR NAME ANNUAL-DED
TITLE 'ANNUAL DEDUCTION REPORT'
```

```
03/21/94        ANNUAL DEDUCTION REPORT       PAGE 1

                                    ANNUAL
EMPNR ...NAME.................      ...DED

7432   JONES, ROBERT               2076.00
1511   KLINGMAN, ALBERT            2911.15
5419   ROSS, JOHN                  2577.00
```

Example:

Create a new field and set it to a specific value. SELECT management employees who are upper management with less than 18 years of education, middle level with less than 16 years of education, and junior management with less than 14 years of education. Print a report, grouped by management level from the highest to lowest level within each group. The report should contain group code, employee number, name, and education level.

```
SELECT STAT EQUAL-TO 'MGT'
SELECT PAY-GRADE GREATER-THAN '15'
SELECT EDUC LESS-THAN '18'
CALC GROUP EQUALS 1
NEWTEST
SELECT STAT EQUAL-TO 'MGT'
SELECT PAY-GRADE GREATER-THAN '11'
SELECT EDUC LESS-THAN '16'
CALC GROUP EQUALS 2
NEWTEST
SELECT STAT EQUAL-TO 'MGT'
SELECT PAY-GRADE GREATER-THAN '08'
SELECT EDUC LESS-THAN '14'
CALC GROUP EQUALS 3
SORT EDUC DESCENDING GROUP
TITLE 'MANAGEMENT PERSONNEL BY LEVEL OF EDUCATION'
PRINT GROUP EMPNR NAME EDUC
BREAK GROUP
```

```
03/21/94 MANAGEMENT PERSONNEL BY LEVEL OF EDUCATION PAGE 1

     GROUP   EMPNR    ..........NAME..........  EDUC

       1      2309    ALEXANDER, GEORGE          17
       1      0927    THACKER, RAY               17
       1      1401    MARSH, MICHAEL             15

       2      3702    LANGLOIS, RANEY            15
       2      4108    SMITH, FRANKLIN            14

       3      3659    STULL, DANA                12
       3      4217    BOSCH, JAMES               11
```

Example:

The following example shows another typical use of the CALC verb:

```
AGED ACCOUNTS RECEIVABLE:

CALC BILL-A DEC2 = ZERO
CALC BILL-B DEC2 = ZERO
CALC BILL-C DEC2 = ZERO
SELECT DATE-BILLED LT '9306'
CALC BILL-C DEC2 = BILL-AMT
NEWTEST
SELECT DATE-BILLED LT '9309'
CALC BILL-B DEC2 = BILL-AMT
NEWTEST
SELECT DATE-BILLED GT '9309'
CALC BILL-A DEC2 = BILL-AMT
TITLE 'AGED ACCOUNTS RECEIVABLE'
PRINT CUST-NR CUST-NAME
BILL-A (30 DAYS OLD)
BILL-B (90 DAYS OLD)
BILL-C (180 DAYS OLD)
TOTAL BILL-A BILL-B BILL-C
```

```
12/01/93           AGED ACCOUNTS RECEIVABLE      PAGE 1

                                                    30      90     180
     CUST              CUST                       DAYS    DAYS    DAYS
      NR       ..........NAME.........          ...OLD  ...OLD  ...OLD

     1746      SMITH AND JONES COMPANY          36.12     .00     .00
     2697      WILLIAMS DAIRY                     .00     .00  154.90
     0937      RODGERS DISTRIBUTORS               .00   11.74     .00

                                                36.12   11.74  154.90
```

Auto Gen of 'Calc' Statement Result Field Specifications

ARITH.OPER IN CALC	Result Field Length (Max of 15 Digits)	Result Field Number Decimal (Max of 9 Decimals Regardless of Override)
PLUSPLUS	One plus the length of the longer of 2 input fields	Same as the greatest number decimals found in two input fields or override
MINUSMINUS	The length of the longer of 2 input fields	Same as the greatest number decimals found in two input fields or override
TIMESTIMES	Enough to hold number digits in multiplier plus number digits in	Sum of decimal count in 2 input fields or override

ARITH.OPER IN CALC	Result Field Length (Max of 15 Digits)	Result Field Number Decimal (Max of 9 Decimals Regardless of Override)
	multiplicand	
DIV-BYDIV-BY	Length of dividend	Number decimals in dividend plus number in divisor or override

All CALC results are packed decimal.

Note: Could optionally specify result number decimals and/or rounding.

EQUATE

Your data processing department has provided you with a list of fields in each file that can be referenced by VISION:Forms. There are occasions when you must be able to reference a part of a particular field and that sub-field has not been assigned a data name and placed in the dictionary. A good example is the personnel file that includes data name DT-EMPLY to cover the six-position field of two-position year followed by two-position month followed by two-position day (yymmdd). The EQUATE verb allows you to temporarily define each of those three sub-fields.

EQUATEs should be coded immediately after the FILENAME verb.

Refer to your data processing department for further information. See the chapter "**Advanced Techniques**" for information on extended EQUATE verb.

Examples:

```
EQUATE YR-EMPLOYED INF31-32
EQUATE GROSS INF290-295-P 2
```

Verb Structure

```
                                OPT          OPT
Word            Word            Word         Word
  1               2               3            4

Dataname        INFnnnn-nnnn      n            n
                and format of     nC
                data              nE
                                  or special
                                  code
```

Coding Rules:

WORD

1 Enter the temporary data name by which you refer to this field or sub-field of data. This data name must be no more than 14 characters long and the first character must be alphabetic. Further, the name must not be a VISION:Forms reserved word and must not already be used as the data name for another field in the file.

2 Code a VISION:Report field definition to describe this temporary field. This definition must be in the format:

- The three characters INF followed immediately by….
- The 1- to 4-digit starting position of this field in the record followed immediately by…
- A dash, which is followed immediately by…
- The 1-to 4-digit ending position of this field in the record.

If this sub-field is packed decimal, end your definition with -P.

3 This optional field indicates report decimal/commas specs or special editing. One of the following entries can be coded for Word 3:

A. One of the digits 0 through 9. This indicates the field is printed with zero suppression and 0 through 9 decimal positions.

```
EQUATE NO-AMT INF26-30 1
```

If NO-AMT contains 01289, it would be printed as 128.9

B. One of the digits 0 through 9 followed by the character C. This indicates the field is printed with zero suppression, commas, and 0 through 9 decimal positions.

```
EQUATE MO-AMT INF26-30 1C
```

If MO-AMT contains 11289, it would print as 1,128.9

When you are going to print many fields, you should remember that decimals and particularly commas increase the space required to print this field.

C. One of the digits 0 through 9 followed by the character E. This is the same as 2 above except European punctuation is used.

```
EQUATE MO-AMT INF26-30 1E
```

If MO-AMT contains 11289, it would print as 1.128,9

D. Any of the special edit codes A to Z (except C, E, and N). Your data processing department can advise you of what type of editing is performed by each code. Some of the standard edit codes provided are:

Code	Edit Pattern	Possible Editing Uses
A	99/99	Dates
B	99.99	Time
D	99/99/99	Dates
F	99-99-99	Dates/time
G	99.99.99	Dates/time
L	999-9999	Telephone numbers
T	999-999-9999	Telephone numbers
U	(999) 999-9999	Telephone numbers

```
EQUATE DATE-PD INF1-6 D
```

If DATE-PD contains 070193, it would print as 07/01/93.

4 This optional field indicates how many decimal positions there are in the field being defined. It only needs to be coded when that field contains a number. Enter a single digit in the range 0-9 to specify how many decimal positions are to be considered to exist when this sub-field is used for arithmetic (i.e., a simple count would have 0 decimal positions while a dollars-and-cents field would have 2 decimal positions).

Note: If Word 4 is used, Word 3 must also be coded.

Example:

Your personnel file contains a date employed field (DT-EMPLY). DT-EMPLY is a 6-position field consisting of three 2-position sub-elements year, month, and day. Your VISION:Forms program must SELECT those employees who have been hired after 1989 and print year of employment. DT-EMPLY is in position 40 through 45 of each record.

```
EQUATE YR-EMPLY INF40-41
SELECT YR-EMPLY GT '89'
PRINT YR-EMPLY
```

Example:

Name a dollars-and-cents field that is in packed decimal format in record positions 31-34 which is to be printed with two decimal positions and commas. This field is also to be used for arithmetic; specify two decimal places:

```
EQUATE SALE-AMT INF31-34-P 2C 2
```

It is very important that you coordinate any EQUATE coding with your data processing department. There might be a compelling reason why a particular field has not had a data name selected for it. A good example is a field that usually, but not always, contains a number. Should you specify arithmetic on such a field, the entire VISION:Forms program is aborted when the computer attempts arithmetic on a record that does not contain a number in that field.

FILENAME

Any file or collection of records that is referenced by VISION:Forms must be assigned a file name and information about the file must then be stored in the dictionary under that file name. When you code that file name into a FILENAME verb, you provide VISION:Forms with a key to the dictionary information about that file. This verb statement is required in every VISION:Forms program.

See the chapter "**Advanced Techniques**" for information on extended FILENAME verb.

Examples:

```
FILENAME PARTMSTR
FILENAME PERSONEL
FILENAME PARTMSTR PERSONEL
```

Verb Structure

```
  Word
    1
Filename [Filename] [Filename]
```

Coding Rules:

WORD

1 Enter the one-to eight-character file name that has been assigned by the data processing department to the file that provides the data for your report.

The JOBNAME, REQUESTOR, or FILENAME verb must be the first verb or line of your VISION:Forms program.

FIND

The FIND verb lets you retrieve one record from a KSDS VSAM file based on the contents of a search key compared to the key of the VSAM file. The FIND logic occurs after your SELECT or MSELECT has selected records. (See MSELECT and SELECT in this chapter).

Examples:

```
FIND CUST-PROD-CDE IN PRODCDE NOTFOUND OK
FIND INV83-PROD-CODE IN PRODCDE
FIND INV83-ZIP-CODE IN ZIPCODE NOTFOUND OK
```

Verb Structure

Word 1	Word 2	Word 3	OPT Word 4	OPT Word 5
Dataname	IN	Filename	NOTFOUND	OK

Coding Rules:

WORD

1 Code the data name of the field that you want to use as the search key for the FIND.

This data name can be from any file that has been referenced before the FIND statement except the file named as Word 3. (See Word 3 under VERB STRUCTURE in this chapter.)

The data name must exist in the dictionary, or be an ATTACHed or EQUATEd field.

If the data name is an ATTACHed or EQUATEd field, be sure to code the ATTACH or EQUATE statement before the FIND statement that references it.

Be sure this search key is valid. If the secondary file was not defined with IDCAMS as being indexed by this field, you will get an invalid result. The secondary file must have this field as a key. Check with your data processing department if you are not sure of the file type or indexing.

2 Enter the word IN.

3 Code the file name of the secondary file that contains the record you want.

- Files must be KSDS VSAM only.
- File name must have been coded on the FILENAME statement.
- File name must be defined in the dictionary.

– File name cannot be the primary file name.

4 Enter the word NOTFOUND. This is used only if it is followed by Word 5. (See Word 5 under VERB STRUCTURE in this chapter.)

5 Enter the word OK. Enter Word 4 and Word 5 to continue processing whether or not the data is found. (For a complete explanation of the use of Word 4 and Word 5, see CODING NOTES: in this chapter.)

Coding Notes:

It is important that you understand the effect the optional keywords NOTFOUND OK have on your output.

- Suppose you enter NOTFOUND OK on the FIND statement and VISION:Forms does not find any matching record. In this case, any fields referenced from the file named in Word 3 are filled with blanks and/or zeros. Processing skips over any other existing selection criteria and continues with non-selection statements. For example:

```
FILENAME ORDER PARTDATA
SELECT DATE = '010194'
FIND PART-NR IN PARTDATA NOTFOUND OK
NEWTEST
SELECT DATE = '010294'
FIND PART-NR IN PARTDATA NOTFOUND OK
PRINT QUANTITY PART-NR CUSTOMER DESCRIPTION
```

  If, in this case, the first FIND does not find a matching record, VISION:Forms continues processing with the PRINT statement.

- Suppose you do not enter NOTFOUND OK on the FIND statement and VISION:Forms does not find any matching record. One of two things occurs:

  If there is another set of selection criteria (new selection criteria always start with a NEWTEST verb), the data from the primary file and any secondary files that were matched passes to this new selection criteria for processing.

  If no further selection criteria exist (no NEWTEST), the record is bypassed.

  Example:

```
FILENAME ORDER PARTDATA
SELECT DATA = '010194'
FIND PART-NR IN PARTDATA
NEWTEST
SELECT DATA = '010294'
FIND PART-NR IN PARTDATA
PRINT QUANTITY PART-NR CUSTOMER DESCRIPTION
```

  If, in this case, the first FIND statement does not find a matching record, the data is passed to the new selection criteria (starting with NEWTEST). If the second FIND statement does not find a matching record, the record is bypassed, since no more selection criteria exist in this VISION:Forms program.

Example:

Suppose you have an ORDER file containing frequently used part numbers (PART-NR). The PARTDATA file contains descriptions (DESCRIPTION) of these part numbers. You want to list all orders entered on 01/01/94. You then want to print out the quantity, part number, customer name, and, if available, the part description of these January 1 orders. You would code the following:

```
FILENAME ORDER PARTDATA
SELECT DATE = '010194'
FIND PART-NR IN PARTDATA NOTFOUND OK
PRINT QUANTITY PART-NR CUSTOMER DESCRIPTION
```

The keywords NOTFOUND OK on the FIND statement tell VISION:Forms to use each selected record. It does not matter whether a DESCRIPTION was found.

If the keywords NOTFOUND OK were not coded, the report would show only the selected orders that had a matching record in the PARTDATA file.

Note: As with any type of file matching, FIND requires a considerable amount of overhead. Make all required SELECTs to the primary file to eliminate unnecessary matches to the secondary files. If you have trouble with this, you can check with your data processing department for more information on file structures and organization.

JOBNAME

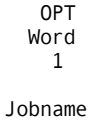
VSE Only

Each job or program that a computer executes must be identified by a job name. This verb allows the VSE user to override the standard job name that VISION:Forms generates.

Examples:

```
JOBNAME ACCNT201
JOBNAME CUSNUMBR
```

Verb Structure



Coding Rules:

WORD

1 Enter a one-to eight- alphanumeric character word (0-9, A-Z, #, $, @, /, -, or .) that is assigned to the VISION:Report program to produce your report.

The JOBNAME, REQUESTOR, or FILENAME verb must be the first verb or line of your VISION:Forms program.

If no job name statement is coded, FILENAME is assigned as JOBNAME. If the REQUESTOR verb was used, the first 8 positions of the REQUESTOR is used as the job name.

LIMIT

When particularly large files are to be input to VISION:Forms or long reports are expected as output, you can run a short test VISION:Forms program and review the results before requesting the full run. The LIMIT verb enables you to specify that only the first nnnn records are to be read and/or the report is to be limited to nnn pages.

The LIMIT verb can appear twice in a given VISION:Forms program, once to LIMIT selection input and once to LIMIT report pages.

Examples:

```
LIMIT SELECTION INPUT 500
LIMIT REPORTING OUTPUT 50
```

Verb Structure

Word 1	Word 2	Word 3
SELECTION	INPUT	nnnnn
REPORTING	OUTPUT	nnnnn

Coding Rules:

WORD

1 Enter the word SELECTION if only the first nnnn records of the file are to be examined or read. Code the word REPORTING if the intention is to LIMIT report pages.

2 If Word 1 contains SELECTION, enter INPUT. When Word 1 contains REPORTING, code OUTPUT.

3 Code a one-to five-digit number indicating how many records are to be read for SELECTION or how many report pages are to be printed. This number should not be surrounded by quotation marks.

See SAMPLE in this chapter for another method of limiting output.

MATCH

The MATCH verb lets you compare two files in order to extract data common to both. Suppose, for instance, you have a file containing customer names but no addresses, and another file with addresses but no names.

If the two files contain some common information such as customer numbers, the MATCH verb can match the customer number from one file to the other. The fields in the two files that contain customer numbers must have different data names, but the information contained in the two fields must be identical in format and length.

Once the MATCH verb compares the files on these two fields and matches the records you want, the MATCHed data is available for further processing.

Examples:

```
MATCH ON CUST-NR TO INVCFILE USING INVC-CUST
MATCH ON CUST-NR TO ADDRFILE USING ADDR-CUST
```

Verb Structure

```
Word      Word         Word      Word      Word       Word
  1         2            3         4         5          6
 ON     Dataname        TO     Filename   USING     Dataname
      (primary file)                                (secondary file)
```

Coding Rules:

WORD

1 Enter the word ON.

2 Code the 'Dataname' of a field in the primary file that you want to use as the MATCH criteria to the secondary file.

See CODING NOTES: in this chapter for more information concerning the primary file.

3 Enter the word TO.

4 Code the 'Filename' of the secondary file you want to match against.

The secondary file can be KSDS VSAM or sequential according to the following:

- If there is more than one MATCH statement in your VISION:Forms program, the file must be KSDS VSAM and matched by the VSAM key.
- To MATCH to a KSDS file, the data name (in Word 6) must be a key or partial key in the VSAM file. If the complete key is used for matching, better performance can be achieved by using the FIND statement.
- VISION:Forms accesses these files with a START BROWSE type of operation. Therefore, they must be set up properly so that the data names you are using work on the MATCH statement. For further help, contact your data processing department.
- If there is only one MATCH statement in your VISION:Forms program, the file can be a sequential disk or tape file.
- If you are MATCHing two sequential files, they must be sequenced by the same field. For instance, if one file is arranged alphabetically by customer's last name, the other file must be arranged that way too.
- The file name must have been coded in the FILENAME statement.
- The file name cannot be the primary file name.
- This file name can be used only one time per VISION:Forms run.
- There cannot be more than 10 MATCH statements per VISION:Forms program.

5 Enter the word USING.

6 Code the 'Dataname' of a field in the secondary file (named as Word 4) you want to test for matches. This field and the one in the primary file (named as Word 2) must contain information that is the same in format and length.

This data name must be defined in the dictionary for the file name.

This field must contain the same number of characters as the field named in Word 2. For instance, if the customer number field in the primary file contains 8 positions, then the customer number field in the secondary file must also contain 8 positions.

Coding Notes:

1 The primary file can be any file that can normally be accessed with VISION:Forms, such as VSAM, ISAM, sequential tape, sequential disk, etc. Even if a procedure is used to access the file (such as DL/I), it can still be used as the primary file.

2 Any procedure coded to REPLACE GET for the primary file is used. Procedures can be specified in the following ways:

```
FILENAME filename USING procedure
```

or

```
USE PROCEDURE procedure-name TO REPLACE GET
```

or

```
positions 72-79 of the 'S' record.
```

3 If VISION:Forms cannot find a matching record, the record area for the secondary file is filled with blanks and/or zeros.

4 You can use multiple MATCH statements in any given VISION:Forms run. However, you can only MATCH once to a file name per VISION:Forms run.

5 The MATCH statements must immediately follow the FILENAME statement.

6 The MATCH verb is designed to be used with the MSELECT verb. Before trying MATCH, you could refer to MSELECT in this chapter to see how the two verbs work together.

7 If you code a MATCH verb, you must code an MSELECT or results could be unpredictable.

Example:

```
FILENAME CUSTOMER ADDRESS
MATCH ON CUST-NR TO ADDRESS USING SITE-NR
```

In this example, CUST-NR is the data name of the customer number field in the primary file, CUSTOMER. This file contains customer names. SITE-NR is the data name of the customer number field in the secondary file, ADDRESS. This file contains customer addresses, but no names.

This MATCH statement matches the customer file to the address file using the customer-number as the MATCH criteria. If no address is available in the ADDRESS file for a particular customer, the output for that record is zeros and/or blanks.

MSELECT

The MSELECT verb lets you select records to evaluate from the results of MATCH statements. (See MATCH in this chapter for a description of its use.)

With MSELECT, you can select matched records in different ways. You can select:

All matches made between the primary file and all secondary files coded in all previous MATCH statements:

```
MSELECT MATCHED
```

Matches made between the primary file and only one secondary file.

```
MSELECT MATCHED TO ADDRESS
```

Matches made between the primary file and one or more of the secondary files.

```
MSELECT MATCHED TO ADDRESS INVOICE
```

All non-matches made between the primary file and all secondary files coded in all previous MATCH statements.

```
MSELECT UNMATCHED
```

Non-matches made between the primary file and only one secondary file.

```
MSELECT UNMATCHED TO ADDRESS
```

Non-matches made between the primary file and one or more of the secondary files.

```
MSELECT UNMATCHED TO ADDRESS INVOICE
```

Examples:

```
MSELECT MATCHED
MSELECT MATCHED TO ADDRESS
MSELECT MATCHED TO ADDRESS INVOICE
MSELECT UNMATCHED
MSELECT UNMATCHED TO ADDRESS
MSELECT UNMATCHED TO ADDRESS INVOICE
```

Verb Structure

Word 1	OPT Word 2	OPT Word 3	OPT Word 4-12
MATCHED UNMATCHED	TO	Filename	Filename

Coding Rules:

WORD

1	Enter one of the following words:	
	MATCHED	An MSELECT MATCHED statement selects all matches made between the primary file and all secondary files coded in all previous MATCH statements. To test secondary files individually for MATCHing, code Word 2 and Word 3. (See Word 2 and Word 3 under VERB STRUCTURE in this chapter.)
	UNMATCHED	An MSELECT UNMATCHED statement selects all non-matches made between the primary file and all secondary files coded in all previous MATCH statements. To test secondary files individually, code Word 2 and Word 3. (See Word 2 and Word 3 under VERB STRUCTURE in this chapter.)

2 Enter the word TO. This is used only if it is followed by one or more file names. (See Word 3 under VERB STRUCTURE in this chapter.)

3 Code the 'Filename' of a secondary file. Code in Word 2 and Word 3 (TO Filename) to select matches or non-matches from only one file that was coded on previous MATCH statements. For example, the statement MSELECT MATCHED TO ADDRESS selects only the records from the primary file that matched the ADDRESS file regardless of other MATCH statements.

4-12 Code the 'Filenames' of additional secondary files. The rules here are the same as those for Word 3. There is an implied OR between Word 3 and Word 4, between Word 4 and Word 5, etc.

Coding Notes:

1 All file names used in an MSELECT statement must have been coded in previous MATCH statements.

2 The file names must not be the file name of the primary file.

3 After you perform MATCH operations, any field you reference from secondary files is automatically ATTACHed to the primary record. (The fields from the secondary files which are found to match fields in the primary file are 'appended' to the primary file.)

4 The MSELECT verb is designed to be used with the MATCH verb. Before using MSELECT, you could refer to the description of MATCH to see how the two verbs work together. (See MATCH in this chapter.)

Example:

```
FILENAME CUSTOMER INVOICE ADDRESS
MATCH ON CUST-NR TO INVOICE USING INV-CUST
MATCH ON CUST-NR TO ADDRESS USING SITE-NR
MSELECT MATCHED
NEWTEST
MSELECT UNMATCHED TO ADDRESS
```

In this example, the MSELECT MATCHED statement selects all records from the CUSTOMER file that matched both the INVOICE file and the ADDRESS file. Any fields that are referenced from the secondary files are automatically ATTACHed to the CUSTOMER record.

Any record in the primary file that does not have matching records on both the INVOICE file and the ADDRESS file passes to the next existing selection criteria. (The NEWTEST verb indicates that new selection criteria follows. See FIND in this chapter for a complete description.) If no further selection criteria exist, the record is not used.

In this example, there is a NEWTEST. Thus, processing passes to this new selection criteria. The MSELECT UNMATCHED TO ADDRESS statement selects only the records from the CUSTOMER file that did not match the ADDRESS file regardless of whether or not there was a MATCH on INVOICE file.

Note: As with any type of file matching, MSELECT requires a considerable amount of overhead. Make all required SELECTS to the primary file to eliminate unnecessary MATCHes to the secondary files. If you have trouble with this, you might consult your data processing department for more information on file structures and organization.

NEWPAGE

This verb is designed to allow you to specify that, when the contents of designated elements change, the report skip to the top of the next page so that the next report section starts on a new page.

Examples:

```
NEWPAGE PART-NO
NEWPAGE PLANT DEPT
NEWPAGE STORE-NO DEPT RACK-NO
```

Verb Structure

```
                  OPT          OPT
  Word          Word         Word
    1             2            3

Dataname     Dataname     Dataname
```

Coding Rules:

WORD

1 Enter a ‘Dataname’, which the computer is to check for changes from one incoming record to the next. When a change is detected in this field, the computer is to skip to the top of the next report page before continuing the report.

2 Optionally a second 'Dataname' to be handled the same as the data name in Word 1.

3 Optionally a third 'Dataname' to be handled the same as the data name in Word 1.

Example:

The file coming into the report program has been sequenced by division (DATANAME DIV) so that all records for division 1 are read first, records for division 2 are read next, and so forth. It is desired to start the report section for each division on a new page. To achieve this, you should code NEWPAGE DIV. If the file is further sequenced by plant within division and a new page is desired for each plant, you should code NEWPAGE PLANT. If it is desired to start a new page for each division and plant, you should code NEWPAGE DIV PLANT.

NEWTEST

Record selection is usually based on a record satisfying the criteria stated in one or more SELECT statements. Should a record fail the criteria, the record can be selected based on satisfying a second or succeeding criteria. The NEWTEST verb signals the start of this second or succeeding group of selects.

Example:

```
NEWTEST
```

Coding Rules:

No words are coded with NEWTEST. The verb itself signals the start of a new criterion. It should be coded immediately ahead of the first SELECT statement of the new group. In single criterion selection, if a record fails any test, the next record is read. If, however, a NEWTEST verb is present, testing is restarted on that record with the SELECT following the NEWTEST verb statement.

Example:

```
SELECT AGE GT '65'
NEWTEST
SELECT DATE-EMP LESS-THAN '1985'
```

The above example requests a list including employees older than 65 or those who have been employed before 1985.

ONBREAK

Many reports require special calculations at BREAK time other than TOTALS and AVERAGES. The ONBREAK verb is the method used to tell VISION:Forms that extra calculations are needed for different BREAK levels. This statement must be followed by a CALC statement(s) and/or a PRINT statement(s). You should be aware that no printing is done unless a PRINT statement is coded.

Examples:

```
ONBREAK PLANT
ONBREAK DIVISION FINAL COMPANY
ONBREAK
```

Verb Structure

```
    OPT             OPT
   Word            Word
    1               2-10

Dataname        Dataname
COMPLETE         FINAL
 FINAL
  END
```

Coding Rules:

If only one ONBREAK statement is coded in a VISION:Forms program and it has no operands following it, VISION:Forms assumes that its associated CALCs and PRINTs are to be performed at each BREAK level. If more control is needed to specify only certain BREAK levels, you can choose one of the following:

WORD

1 Enter the 'Dataname' whose break level you wish to do your extra calculations within. ONBREAK CALCs and PRINTs are performed at only this level, unless other ONBREAK data names are coded.

Enter FINAL if the calculations are to be done only at final break time (EOJ).ONBREAK:FINAL Enter END or COMPLETE to indicate the end of ONBREAK statements and their associated PRINTs and CALCs. This operand is only coded if other VISION:Forms statements follow.

2 through 10

Enter additional 'Dataname(s)' on which you need the same CALCs and PRINTs performed.

If you coded previous ONBREAK levels and need the same series of CALCs and PRINTs performed at EOJ, code the word FINAL.

If no operands are coded, all break levels (including FINAL) are assumed, except those that were specified in an earlier ONBREAK statement. For instance, assume you are breaking on four data names (DIV, PLANT, DEPT, EMPLOYEE), and have coded ONBREAK PLANT. Those CALCs and PRINTs are performed only when there is a change in PLANT. If you then code an ONBREAK followed by no operands, the following CALCs and PRINTs are performed at the DIV, DEPT, and EMPLOYEE break levels, as well as at EOJ.
No total break level can be used in more than one ONBREAK statement.

Coding an ONBREAK Routine:

The ONBREAK verb is just the first part of the routine. It triggers the beginning of a series of CALCs and PRINTs that are to be performed at BREAK time using TOTALed fields. These CALC and PRINT verbs have only minor differences from their normal use.

- Data names used in a CALC must also be used in either a TOTAL or AVERAGE statement, or be the result of another CALC within the same ONBREAK routine.
- PRINT statements do not have the HEADLINE keyword. You can code data names only or you can code data names followed by an override header. The ONBREAK CALCed fields are printed one per line preceded by the data name (or override header). The statement:

```
PRINT INC-PERCENT INC-PERCENT(% OF INCREASE)
```

causes the following to appear on your report after any TOTALs or AVERAGEs.

```
INC-PERCENT     12.5
% OF INCREASE   12.5
```

- PRINT statements can also be coded using the UNDER keyword explained in 'Under Coding Rules' in Multiple Print Lines in this chapter. In this case, the ONBREAK CALCed fields are printed all on one line with no description (data name or override header) shown.

PAGEDEPTH

VISION:Forms reports normally contain 54 lines per page, including the title and column header lines which are followed by one blank line. The PAGEDEPTH verb enables you to specify that a particular report is to be printed with more or fewer lines per page.

Examples:

```
PAGEDEPTH 88
PAGEDEPTH 9
```

Your data processing department could have changed the lines per page by permanently updating the VISION:Report OPTION. Refer to your data processing department for more details.

Verb Structure

```
Word
  1

nnn
```

Coding Rules:

WORD

1 Code a one-to three-digit number specifying the number of lines per page that the generated report should have. In calculating this number, you must allow for heading lines. A short test run might be necessary to determine how many print lines are used for headers.

PAGEWIDTH

VISION:Forms assumes there are 132 print positions available for use in each report line. If a particular report must be less than 132 positions, the PAGEWIDTH verb allows you to indicate how many print positions wide the report should be.

Examples:

```
PAGEWIDTH 75
PAGEWIDTH 120
```

Verb Structure

```
Word
  1

nnn
```

Coding Rules:

WORD

1 Enter a two-or three-digit number indicating the number of print positions that VISION:Forms should consider as available for report composition.

If a particular installation's printer has more or fewer than 132 positions, your data processing department can permanently modify VISION:Forms to those specifications and the PAGEWIDTH verb need not be submitted with each VISION:Forms program.

PARTITION

VSE Only

VISION:Report job streams generated by VISION:Forms include JCL that can be customized for a particular partition. VISION:Forms presumes that the partition it is executing in is also the VISION:Report executing partition and copies JCL for that partition. The PARTITION verb allows you to override this default by specifying a different VISION:Report executing partition. This override partition is used in copying JCL.

Refer to your data processing department for further information.

Examples:

```
PARTITION BG
PARTITION F2
```

Verb Structure

```
Word
  1

BG,Fn
```

Coding Rules:

WORD

1 Enter the two-character designation (BG, F2, etc.) of the partition in which the generated VISION:Report job stream is to execute. To the extent that partition is sensitive to what JCL is present, the partition entered here determines the JCL set to be used.

PRINT

This verb is the method by which you specify which data names are to appear in the report and the sequence in which they are to appear. You can include more than one print verb per request.

Refer to **Report Format - How VISION:Forms** Composes Your Report in the chapter "Basic Concepts of VISION:Forms.."

Examples:

```
PRINT CUST-NR COMPANY ADDRESS
PRINT DEPT SPACE20 PLANT SPACE10 NAME
PRINT YTD-WAGE UNDER WAGE-EARNED
```

Verb Structure

OPT Word 1	Word 2	OPT Word 3	OPT Word 4	OPT Word 5	OPT Word 5
HEADLINE	Dataname SPACEnn UNDER	Dataname SPACEnn UNDER	Dataname SPACEnn UNDER	Dataname SPACEnn UNDER	Dataname SPACEnn UNDER

Coding Rules:

WORD

1 HEADLINE indicator. If multiple PRINT verbs are coded in this VISION:Forms program, code the keyword HEADLINE in word 1 of the PRINT statement that should be used for header composition. Multiple print line considerations are covered in Multiple Print Lines in this chapter.

If each selected record needs only one print line, the optional keyword HEADLINE is not necessary.

2 through 5

Three different types of entries can be made:

Dataname	The contents of this field are printed.
SPACEnn	VISION:Forms composes the report in the most balanced format possible. You can force the report into a particular format. The SPACEnn keyword allows you to specify nn blanks between this field and the next field. Thus, PRINT NAME SPACE20 ADDRESS causes 20 blanks between the end of the NAME field and the start of the ADDRESS field.
UNDER	Used when printing a report with multiple print lines. Refer to Multiple Print Lines in this chapter.

Data appears on report lines in the same sequence in which it is coded in the PRINT verb. The data name coded first after PRINT prints on the left of the report line. The last data name coded appears rightmost on the report line.

Report Headers

The data names you code after the PRINT verb identify which fields are to be printed, and those data names themselves print at the top of each report page as column headers centered above the actual print data from the records. If the data name is not appropriate for use as a column header, you can specify a different column header on either a temporary or permanent basis.

Temporary Headers

If the data name MS field is to be printed and you desire the column header MARITAL-STATUS, code:

```
PRINT MS(MARITAL-STATUS)
```

In this case, all references to the field would be coded MS, but the report produced by this VISION:Forms program would have MARITAL-STATUS at the top of the column. Temporary headings override alternate headings stored in the dictionary.

Permanent Headers

To permanently assign an alternate column header to a data name, enter the alternate name starting in position 51 of the dictionary 'D' record for that data name. If, for instance, positions 51-66 of the 'D' record for MS is changed to read (MARITAL-STATUS), MARITAL-STATUS is used as the column header whenever PRINT MS is coded.

Note: Refer to your data processing department to assign permanent headers.

Example:

List the employees along with their wage and tax information. Note the column headers are taken from the PRINT statement.

```
FILENAME PAYROLL
SELECT EMP-1 EQ 'A' 'C' 'R'
CALC TAX-DED = WAGE-AN * .20
PRINT EMP-NAME WAGE-AN TAX-DED
```

```
         EMP                      WAGE                    TAX
    .....NAME.....           .......AN               .....DED

   ASTOR, TERRY J.           15,000.00               3,000.00
   CASSIDY, HOMER R.         14,750.00               2,950.00
   ROMINE, MARYLON M.        17,975.00               3,950.o0
```

Example:

List customers in Oregon; include their telephone numbers, city, and their status. Note the temporary headers within parentheses in the PRINT statement used for this report.

```
FILENAME CUSTFILE
SELECT STATE EQ 'OREGON'
PRINT CUST-NAME SPACE20
TELE#(AREA-PHONE) CITY STATUS
PAGEWIDTH 60
```

```
     CUST                                AREA
.....NAME.....                     .......PHONE    ..CITY.. STATUS

ABC COMPANY, INC.                  290-356-3214    PORTLAND   G1
IMAGERY DESIGN LTD.                295-658-4577    GREENLY    A1
```

Multiple Print Lines

You can specify multiple line printing for each report input record by coding several print verbs. When this is done, the following considerations arise.

1 Report format composition is still based on the data names cited in one print verb. This particular print verb must be identified by coding HEADLINE as the first word following the PRINT verb itself. The PRINT HEADLINE verb need not be in the first PRINT statement.

2 Print lines appear in the report in the same sequence in which they are coded in the VISION:Forms program. The earliest PRINT encountered generates the first print line for each record. The last PRINT verb generates the last print line in the report for that record.

3 When all the MOVE PRINT internal code has been generated, an additional PRINT statement is generated to print a blank line between print line groups. For example these PRINT statements:

```
PRINT HEADLINE INVOICE-TO
PRINT CUST-NAME
PRINT STREET-NAME
PRINT CITY-ST ZIP


   INVOICE
......TO......
ABC COMPANY INC.
1200 MAIN STREET
PORTLAND, OR 80101
```

4 Secondary (non HEADLINE) lines can be formatted in one of two ways:

- Basically, the first data name cited prints in the leftmost position on the report page through the required print length. A space appears next, and this is followed by the printing of the second data name, which in turn is followed by a space, etc. You can override this by including a SPACEnn verb that causes nn spaces between each data name. If the first print field in a secondary line should start in print position 10, code:

  ```
  PRINT SPACE9 DEDUCTION
  ```

- By using the keyword UNDER, you can cause a data name to be automatically aligned under a data name that was specified in the PRINT HEADLINE statement. For example:

  ```
  FILENAME ACCNTFLE
  CALC MTD-VARIANCE = MTD-BUDGET - MTD-ACTUAL
  CALC YTD-VARIANCE = YTD-BUDGET - YTD-ACTUAL
  PRINT HEADLINE BUDGET-DESC
        MTD-BUDGET(MTD,BUDGET,ACTUAL,VARIANCE)
        YTD-BUDGET(YTD,BUDGET,ACTUAL,VARIANCE)
  PRINT MTD-ACTUAL UNDER MTD-BUDGET
        YTD-ACTUAL UNDER YTD-BUDGET
  PRINT MTD-VARIANCE UNDER MTD-BUDGET
        YTD-VARIANCE UNDER YTD-BUDGET
  ```

In the above example, you want a report showing MONTH-TO-DATE budget data (budget, actual, variance) compared to YEAR-TO-DATE budget data (budget, actual, variance). By using the keyword UNDER within the second and third PRINT statements, VISION:Forms automatically causes the data names to be printed UNDER the specified data names in the PRINT HEADLINE statement.

UNDER SYNTAX:

```
Dataname1 UNDER Dataname2
```

'Dataname1' is the name of a valid data name within the dictionary, an EQUATE, or CALCed data name.

UNDER is the keyword that links 'Dataname1' to 'Dataname2'.

'Dataname2' is the name of one of the data names that was specified in the PRINT HEADLINE statement.

UNDER Coding Rules:

The keyword UNDER is not valid on the PRINT HEADLINE statement.

'Dataname2' must be specified on the PRINT HEADLINE statement.

The length of 'Dataname1' must not exceed the length of 'Dataname2' or the length of 'Dataname2's header, whichever is larger.

When coding, all 'Dataname2's must be coded in the same sequence as they were coded on the PRINT HEADLINE statement.

Invalid:

```
PRINT HEADLINE FLD-A FLD-B FLD-C
PRINT FLD-F UNDER FLD-C FLD-G UNDER FLD-A
```

Valid:

```
PRINT FLD-G UNDER FLD-A FLD-F UNDER FLD-C
```

The SPACEnn and other data names that are not to be printed under a data name can still be coded on the PRINT statement. However, if the length of these data names causes the line to be positioned past the start position of 'Dataname2' that it is to be printed UNDER, an error occurs.

Example:

Assume each data name in the following example is 10 characters in length.

Valid:

```
PRINT HEADLINE FLD-A FLD-B FLD-C
```

Invalid:

```
PRINT SPACE10 FLD-G UNDER FLD-A
PRINT FLD-H FLD-G UNDER FLD-A
```

In both examples above, SPACE10 and FLD-H cause the line to be positioned past the starting print position of FLD-A.

Summary Reporting

Assume you do not wish to print any detail at all; you wish to make calculations and/or accumulate data at the department or plant level.

A totals-only report is produced by including a TOTAL statement and no PRINT statement. The report is automatically composed according to the following guidelines:

The BREAK statement data name fields are printed on the left side of the report with the most inclusive (major) field appearing leftmost and being followed by control fields in decreasing significance. This is illustrated in the example below where the minor-to-major data names in the BREAK statement have been reversed to major-to-minor in report composition.

TOTAL statement data names are composed in as-is sequence starting immediately after the last BREAK data name and continuing to the right.

If it is desirable to show the number of records that make up the group being totaled, include the reserved word GROUPCOUNT in the TOTAL statement. This feature, shown in the next example, indicates that 7 records with 11/82 start month year were selected, 12 with 01/83, etc.

Up to 21 data names can be specified in any combination of BREAK and TOTAL data names. As with the PRINT verb, a second consideration is that print line space must be available for all the data names.

Example:

The Computer Associates customer file contains the year, month, and day a customer began using our services and a number of dollar amount fields.

The following request produces a totals-only or summary report.

```
FILENAME CUSTFILE JOBNAME SUMMARY REQUESTOR RON
SELECT STATUS-CODE NOT EQ 'F', 'G'
SORT START-MONTH, START-YEAR
TOTAL BILL-AMT PBILL-AMT GROUPCOUNT
BREAK START-MONTH, START-YEAR
TITLE 'SUMMARY BY START'
```

```
07/16/93                  SUMMARY BY START             PAGE 2

START        START            BILL       PBILL
 YEAR        MONTH        ......AMT   ......AMT      GROUPCOUNT

   82           11          160.00      160.00              7
 **YEAR TOTALS               160.00      160.00**            7
   83           01          185.00      185.00             12
   83           02          445.00      445.00              9
        ETC.
   83           12          250.00      250.00             36
 **YEAR TOTALS               4,375.00    4,375.00**        144
   84           01          136.00      136.00              6
   84           02          125.00      125.00             19
        ETC.
   84           12          120.00      120.00             12
 **YEAR TOTALS               2,056.00    2,056.00**        247

 **GRAND TOTALS              24,330.00   24,330.00***       1611
```

Asterisks shown in this sample do not appear in the report. They are used in this example to point out TOTAL levels.

Example:

You wish to extract a report of pay data showing totals by department and plant.

```
JOBNAME PAY-DEPT REQUESTOR SMITHE FILENAME PAYROLL
SELECT DEPT-NR EQ '005'
SORT DEPT-NR, PLANT-NR
TOTAL HOURS-WORKED, YTD-FICA, YTD-ST-TAX
BREAK DEPT-NR, PLANT-NR
TITLE 'DEPARTMENT 5, YTD FICA AND STATE-TAX'
```

```
07/16/93     DEPARTMENT 5, YTD FICA AND STATE-TAX     PAGE 1

                                                        YTD
                                                         ST
PLANT      DEPT       HOURS         YTD              .....TAX
   NR       .NR   ...WORKED   .....FICA

   12       005    1,267.57     6,543.12              716.18
 **PLANT TOTALS    1,267.57     6,543.12              716.18
   13       005    1,653.46     7,316.66              377.44
 **PLANT TOTALS    1,653.46     7,316.66              377.44

 GRAND TOTALS      x-------X   x-------x            X------x
```

Asterisks shown in this sample do not appear in the report. They are used in this example to point out TOTAL levels.

PSELECT

PSELECT (PRINT SELECT) controls which print lines and their contents are to be printed based upon the data in the input record. The PSELECT verb is exactly the same format as the SELECT verb and offers all of the capabilities of the SELECT verb except ON-TABLE.

The NEWTEST verb should not be used in conjunction with the PSELECT verb.

Examples:

```
SELECT CONAME-2 NOT BLANK
PSELECT REP EQ C'J'
PSELECT TAX-DED LT 4
```

Verb Structure

```
            OPT       OPT       OPT     OPT
  Word      Word      Word      Word    Word          Word       Words
    1         2         3         4       5             6         7-25

Dataname  PLUS     Dataname   NOT     EQUAL-TO      Dataname     Dataname
          +        Constant           EQ            Constant     Constant
          MINUS                       =
           -                          LESS-THAN
          TIMES                       LT
          *                           <<
          DIV-BY                      GREATER-THAN
          /                           GT
                                      >>
                                      HIVALUE(S)
                                      HIGH-VALUE(S)
                                      LOVALUE(S)
                                      LOW-VALUE(S)
                                      POSITIVE
                                      NEGATIVE
                                      NUMERIC
                                      BLANK(S)
                                      ZERO(S) - ZERO(ES)
                                      RANGE          LO-END      HI-END
                                      WHEN           INCLUDING   Dataname
                                                                 Constant
```

Coding Rules:

See SELECT in this chapter for statement rules and restrictions.

Assume you have a file consisting of record formats of A type records, B type records, and C type records. You wish to SELECT A's and B's, or A's and C's, or A's, and B's, and C's. You wish to sort, print, and do totals.

PSELECT is processed as a decision. If the result is true, all print and total requests, until the next PSELECT or end of request, are processed on an associated-with basis and they are executed only if the PSELECT condition(s) are true.

Example:

```
FILENAME ORDER
JOBNAME MULT-REC
SELECT REC-CODE EQ 'A', 'B', 'C'
SORT PLANT
TITLE 'MULTI RECORD PRINT SAMPLE'
BREAK PLANT
NEWPAGE PLANT
PSELECT REC-CODE EQ 'A'
PRINT HEADLINE A-FIELD-ONE A-FIELD-TWO A-FIELD-THREE
TOTAL A-FIELD-TWO
PSELECT REC-CODE EQ 'B'
PRINT B-FIELD-FIVE B-FIELD-SIX
PSELECT REC-CODE EQ 'C'
PRINT C-FIELD-7
```

A, B, and C record codes have records of different formats.

The column headings are controlled by the fields from the A records by the use of the PRINT HEADLINE associated with the record code A PSELECT.

A TOTAL is kept on FIELD-TWO out of the A records only.

When using PSELECT, no verbs other than the PRINT, TOTAL, and QWTABLE statements can follow PSELECT in the input to VISION:Forms.

Example:

In an ORDER file, ship to, bill to, etc. are records of different formats. The order items are in a third format. You wish to print them all.

```
PSELECT SHIP-TO-CD EQ 'S
PRINT SHIP-TO
PSELECT BILL-TO-CD EQ 'B'
PRINT BILL-TO
PSELECT ITEMS EQ 'I'
PRINT HEADLINE ITEM-NR ITEM-DESCR
ITEM-QUAN ITEM-COST
TOTAL ITEM-COST
```

Example:

Assume you have a record with several address segments and you wish to print multiple lines when the segments are not blank. The following illustrates a good use for PSELECT.

```
FILENAME YOURFILE REQUESTOR YOU JOBNAME PMLINES
SELECT RECORD-CODE = 'B'  'C'  'F'  'Z'
SORT PLANT
TITLE 'MULTI-LINE REPORT'
NEWPAGE PLANT
PRINT HEADLINE CUST-NAME ADDRESS
PSELECT LINE-TWO NOT BLANK
PRINT LINE-TWO
PSELECT LINE-THREE NOT BLANK
PRINT LINE-THREE
```

QJSIZE

Note: This verb is applicable for VSE only.

Certain VSE installations and/or particular VISION:Forms programs require that a SIZE=nnnK specification be included in the // EXEC QUIKJOB statement generated by VISION:Forms. The QJSIZE verb causes this specification to be included.

Example:

```
QJSIZE 120
```

Consult your data processing department for further information.

Verb Structure

```
Word
1

nnn
```

Coding Rules:

WORD

1 Code a two-or three-digit numeric value that VISION:Forms is to generate as nnn in the SIZE=nnnK specification in the // EXEC QUIKJOB statement generated.

Example:

```
QJSIZE 120
```

This causes the execute statement to be generated as:

```
// EXEC QUIKBJOB,SIZE=120K
```

QWOPTION

This verb is used to change or override various control functions for one execution of VISION:Forms.

Example:

```
QWOPTION SORTWORK=01
QWOPTION QUIKSORT=NO QJLIST=YES
```

Verb Structure

```
  OPT            OPT            OPT
 Word           Word           Word          Word
   1              2              3             4

SORTWORK=nn   QUIKSORT=YES   QJLIST=NO   AUTOTITLE2=YES
                     NO          YES             NO
```

Coding Rules:

WORD

1 SORTWORK=nn is used to tell VSE how many sort work areas are to be used when the generated SORT is executed. NN must be between 01 and 99.

2 QUIKSORT= is used to tell VISION:Forms whether or not the QUIKSORT feature should be used when generating the VISION:Report statements. The default is YES. To turn off the QUIKSORT feature, enter QUIKSORT=NO.

3 QJLIST= is used to tell VISION:Forms whether or not the generated VISION:Report statements are to be listed after the VISION:Forms statements. The default is NO. Enter QJLIST=YES to cause the VISION:Report statements to be listed.

4 AUTOTITLE2= is used to tell VISION:Forms whether or not to generate a TITLE2 statement that contains the date and page number when there is no room for them on the first TITLE statement. The default is YES. If you do not want a TITLE2 to be generated automatically, enter AUTOTITLE=NO.

QWTABLE

This verb is used to pass to VISION:Forms table specifications and table records for use with a SELECT xxx ON-TABLE statement.

Verb Structure

```
Format 1:    Word 1         Word 2

 Table
 retrieved   DICTIONARY     Name under
 from the                   which table
 dictionary                 is filed in
                            dictionary


OR

                                                        OPT
Format 2:     Word 1        Word 2      Word 3         Word 4

 Table       Maximum no.    Argument     Argument      Argument
 included     of table      starting     length        type - 'P'
 with this   entries. Up    position     (01-55)       if table
 QUIKWRITE    to 9999.      (01-80)                    is packed.


OR

                                                      OPT
Format 3:          Word 5           Word 6           Word 7

 Code same as
 Format 2, and    Function         Function         Dataname
 add these        starting          length          of table
 words to         position          (01-55)         function
 define a         (01-80)
 function.
```

Coding Rules: for tables retrieved from the dictionary

WORD

1 Enter the word DICTIONARY.

2 Enter the name under which the table is filed in the dictionary.

Coding Rules: for tables stored in the dictionary

WORD

1 Enter a four-digit number indicating how many table records follow, or enter a larger number that allows for the greatest number of entries this table is likely to have, if the generated VISION:Report is reused. This Word is required.

2 Code a two-digit number specifying the starting position of the argument in the table entry. This is usually 01 unless you are using a table that was created for another purpose. This Word is required.

3 Enter a two-digit number indicating the length of the argument. If you have coded SELECT PLANT ON-TABLE and PLANT is a two-character field, you should code 02 here. This Word is required.

If the argument is packed, the length coded here must be the length of the argument after it is packed. For instance, if your table statements have the argument in positions 1 to 7, the packed length is '04'.

4 Enter a 'P' only if the table argument is in packed format. This word is optional and, when not coded, VISION:Forms assumes that the table is in EBCDIC format. When the argument is packed, the data name used in the SELECT xxx ON-TABLE must also be packed.

Coding Rules: for tables that contain functions

Follow the rules for Words 1 through 4 above, plus code these new words:

WORD

5 Code a two-digit number specifying the starting position in the table entries of the function. This Word is required.

6 Enter a two-digit number stating the length of each function. This Word is required.

7 Enter a 'Dataname' that identifies the table function so that VISION:Forms users can reference this data in any other VISION:Forms verbs. This data name must not have been previously defined or EQUATEd. This Word is optional, but should be coded unless you have more than one function in a table entry that you are accessing later.

Note: There can only be one QWTABLE statement per VISION:Forms program.

The QWTABLE statement can be placed anywhere in the VISION:Forms after the FILENAME statement. If your table contains a function, the QWTABLE statement must be placed prior to any reference to the function by other verbs so that VISION:Forms' automatic ATTACH has been done. If the QWTABLE statement is placed after a reference to the function, an error message is printed showing the function data name as an undefined field.

When you are including table entries in your VISION:Forms program, either as the complete table or as add-ons to a dictionary table, you must enter the statement TABLE DATA as the last line of VISION:Forms code. This signals that there are no more VISION:Forms statements to process, and that the following records are table entries. The table entries must follow immediately after this statement.

Example:

```
SELECT PLANT ON-TABLE
  .
  .
QWTABLE DICTIONARY PLANTABL
TABLE DATA
03
17
24
```

FORMAT 1

Pass the table statements from the dictionary whose name is PLANTABL into VISION:Forms then add entries 03, 17, and 24 to the end of that table.

Example:

```
SELECT PLANT ON-TABLE
.
.
QWTABLE 0004 01 02
TABLE DATA
01
03
17
24
```

FORMAT 2

Pass the following table data statements to VISION:Forms. There are four entries and each starts in column 1 and extends for two columns.

Example:

```
QWTABLE  0010  01  03  05  15  PLANT-NAME
SELECT PLANT ON-TABLE
PRINT PLANT-NAME
.
TABLE DATA
DAY DAYTON, OH.
STL ST. LOUIS, MO.
PUE PUEBLO, CO.
TEM TEMPE, AZ.
```

FORMAT 3

A table that contains a function, which is used to translate the plant code into city/state where the plant is located. The table can hold up to 10 data elements. Note that the code is three characters long and starts in column 1 of each table entry. The function is in column 5 for a length of 15.

REQUESTOR

VSE Only

It could be of some assistance to be able to identify the department and/or individual requesting the report. The REQUESTOR verb enables the VSE user to include this identification, which is placed in the generated VISION:Reports. This is the place to include any job accounting information.

Examples:

```
REQUESTOR SMITH
REQUESTOR CUSTOMER-MSTR
```

Verb Structure

```
OPT
Word
1
Requestor
```

Coding Rules:

WORD

1 Enter a word of up to sixteen characters that could be any combination of department name, individual's name, or any other information that is helpful in identifying the report. The word coded here appears in the job statement generated by VISION:Forms.

The JOBNAME, REQUESTOR, or FILENAME verb must be the first verb or line of your VISION:Forms program.

Example:

```
FILENAME PAYROLL
REQUESTOR KAREN-PAYROLL
SELECT DEPT-NUM EQ '10' '12'
PRINT NAME DEPT-NUM WAGE-PER-WK
TITLE 'PAYROLL FOR DEPARTMENTS 10 & 12'
```

SAMPLE

For some applications, instead of running through the complete file, a part of that file is enough for a sampling. In these cases, the SAMPLE verb allows you to specify that only every nnnnnth record of the file is to be used in selection or printing.

Examples:

```
SAMPLE SELECTION EVERY 5
SAMPLE SELECTION EVERY 250
```

Verb Structure

```
Word     Word     Word
1        2        3

SELECTION  EVERY    nnnnn
```

Coding Rules:

WORD

1 Enter the word SELECTION.

2 Enter the word EVERY.

3 Enter a one-to five-digit number indicating every nnnnnth record is to be submitted for selection.

If you code 10, then the first, eleventh, twenty-first, etc. records for the file name is input to the selection or print run.

SAMPLE and LIMIT can both be used in the same request. Divide the LIMIT SELECTION input number by the SAMPLE SELECTION number to arrive at the approximate number of input records to be examined.

Example:

```
FILENAME PAYROLL
SAMPLE SELECTION EVERY 10
LIMIT SELECTION INPUT 100
SELECT DEPT EQ '123'
PRINT DEPT EMP-NR NAME
```

In the above VISION:Forms code, 100 records of the PAYROLL file are read but only 10 records are tested for DEPT EQUAL-TO 123.

See LIMIT in this chapter for another method of limiting output.

SELECT

The SELECT verb is the most flexible VISION:Forms verb and offers you the ability to evaluate each record in the initial input file in many different ways. Those records that pass this selection criterion are copied to a new file for sorting and/or reporting.

Examples:

```
SELECT BILL-AMT = 125.00
SELECT BILL-AMT NOT EQ 100.00
SELECT COMM-AMT + 200.00 GT DED-AN
```

```
Verb Structure
OPT      OPT      OPT    OPT
Word     Word     Word   Word      Word         Word     Words
  1        2        3      4         5            6      7-26

Dataname PLUS  Dataname  NOT  EQUAL-TO        Dataname   Dataname
         +     Constant       GREATER-THAN    Constant   Constant
         MINUS                LESS-THAN
         -                    EQ
         TIMES                =
         *                    GT
         DIV-BY               >>
         /                    LT
                              <<
                              HIVALUE(S)
                              HIGH-VALUE(S)
                              LOVALUE(S)
                              LOW-VALUE(S)
                              POSITIVE
                              NEGATIVE
                              NUMERIC
                              BLANK(S)
                              ZERO(S) - ZERO(ES)
                              ON-TABLE
                              ONTABLE
                              RANGE           LO END     HI END
                              WHEN            INCLUDING  Dataname
                                                         Constant
```

Coding Rules:

WORD

1 Enter the 'Dataname' of the field to be evaluated.

2 Words 2 and 3 are used together to have arithmetic performed on the basic field (Word 1) and an evaluation is done on the results of that arithmetic.

PLUS or + indicates that the Word 1 element is to be added to the Word 3 field.

```
SELECT MO-DED PLUS 15.00 EQ 300.00
```

MINUS or - indicates that the Word 3 element is to be subtracted from the Word 1 element.

```
SELECT MO-DED MINUS 50.00 GREATER-THAN 100.00
```

TIMES or * indicates that the Word 1 element is to be multiplied by the Word 3 element.

```
SELECT MO-DED TIMES 12 GREATER-THAN 2000.00
```

DIV-BY or / indicates that the Word 1 element is to be divided by the Word 3 element.

```
SELECT MO-DED DIV-BY 30 LESS-THAN 50
```

3 Enter either a 'Dataname' or a Constant. If a data name is entered, the arithmetic specified in Word 2 is performed using the contents of the data name element coded here. In this case, the actual value used can vary from record to record. If a known constant value is to be used, write that value in one of the following formats:

2	A positive integer of two.
-2	A negative integer of two.
1.50	A positive value of one integer and two decimal positions.
-1.50	A negative value of one integer and two decimal positions.

Constants can consist of up to 15 digits plus a decimal and/or sign indicator as needed.

4 Use of the optional word NOT in Word 4 completely reverses your selection logic.

If you want to SELECT all personnel records for employees at plant 10, code SELECT PLANT EQUAL-TO '10'. If you want to SELECT those employees not working at plant 10, code SELECT PLANT NOT EQUAL-TO '10'.

5 You must code one of the logical relationships. Depending on the relationship specified, Word 1 data name (or arithmetic result of Words 1, 2, and 3) is examined for positive, negative, zero, etc., or compared for relationship against values specified in Word 6, 7, etc.

EQUAL-TO, EQ, =, GREATER-THAN, GT, >>, LESS-THAN, LT, or <<. If Word 1 (or as modified by Words 2 or 3) contents has this relationship to Word 6 contents, this record has passed the test. For example:

Select records for plant 10 employees:

```
SELECT PLANT EQ '10'
```

Select records where pay grade is greater than 5:

```
SELECT PAY-GRADE GREATER-THAN '5'
```

Select records where monthly deductions are less than $100.00:

```
SELECT MO-DED LT 100.00
```

Notice how the word NOT could be included to reverse your selection criteria:

```
SELECT PLANT NOT EQ '10'
SELECT PAY-GRADE NOT GREATER-THAN '5'
SELECT MO-DED NOT LT 100.00
```

POSITIVE, NEGATIVE, BLANK, ZERO, NUMERIC: Examine the contents of Word 1 (or as modified by Words 2 and 3) for the condition coded. If you find that condition, this record passes this test. Once again, consider how the word NOT can be included to reverse your selection criteria.

ON-TABLE: The contents of Word 1 are compared to each entry in the table you have called from the dictionary and/or coded at the end of the VISION:Forms program (see QWTABLE in this chapter). If an exact match is found (1 and 01 are not an exact match), the record passes this test. As a general rule, CALC results and other calculated numeric FIELDS are stored in records in a compressed format called packed decimal. This requires extreme care in coding your table so that an exact match can be achieved. It is recommended that you contact your data processing department for guidance.

RANGE: The contents of Word 1 (or as modified by Words 2 and 3) are examined to see if it is equal to or greater than Word 6 and equal to or less than Word 7. In this instance, code the data name or constant for the low end of your RANGE in Word 6 and the data name or constant for the high end of your RANGE as Word 7.

WHEN: You are specifying that the Word 1 field is to be scanned to see if the contents of Word 7 are present anyplace in that field. If the Word 7 contents are matched someplace in the Word 1 field, the relationship is true. The WHEN variation does not allow for reverse logic and the word NOT can never be coded in Word 4. Always code the word INCLUDING in Word 6. This powerful variation can best be explained with an example:

Jones Street has been changed to Smith Street. You need a report of all employees who live on Jones Street so the address change can be made. Since house numbers can be of varying length, such as 1 Jones Street or 1234 Jones Street, you cannot depend on finding 'Jones' in the same place in the address field of each record. Thus, the statement:

```
SELECT ADDRESS WHEN INCLUDING ' JONES '
```

This example SELECTs all records that include someplace in the address field a space followed by Jones and in turn followed by a space. The surrounding spaces technique is used to prevent selecting people who live on McJones St. or other random occurrences of the character constant of Jones.

6 Enter a 'Dataname' or Constant. The data specified in Word 1 (or as modified by Words 2 and 3) is compared against the data specified here.

If you indicated EQUAL-TO and Word 1 data is equal to Word 6 data, the relationship is true.

If you indicated GREATER-THAN and Word 1 data is greater than Word 6 data, the relationship is true.

If you indicated LESS-THAN and Word 1 data is less that Word 6 data, the relationship is true.

When a true condition is found, VISION:Forms continues to process this record, making other SELECT comparisons until all required tests have been applied. If a record passes all these criteria, it is copied to the selected file. If, on the other hand, a false condition is found, VISION:Forms immediately moves to the next record in the input file and starts testing it.

7 through 26

Use these words for more flexible testing.

OR CAPABILITY: You can cause either/or testing by adding additional values or data names after Word 6.

```
SELECT PLANT EQUAL-TO '10' '12' '17' '26' '57'
```

With this coding, we specify that if plant is 10 or 12 or 17 or 26 or 57, the relationship sought is true.

There are cases where two or more completely unrelated criteria must be used for selecting records for reporting:

- If pay grade is 4 and monthly deductions are greater than 150.00, SELECT.
- If pay grade is 6 and monthly deductions are greater than 200.00, SELECT.
- If pay grade is 6 and monthly deductions are greater than 300.00, SELECT.

To achieve these unrelated selections, we would use the NEWTEST verb that implies, if this record has not passed the preceding test(s), start testing all over again with the SELECTS that follow:

```
SELECT PAY-GRADE EQ 4
SELECT MO-DED GT 150.00
NEWTEST
SELECT PAY-GRADE EQ 5
SELECT MO-DED GT 200.00
NEWTEST
SELECT PAY-GRADE EQ 6
SELECT MO-DED GT 300.00
```

Sometimes a dummy SELECT can be used to force a field definition, such as:

```
SELECT field=field
```

This sets the field to itself.
This is especially true for data bases and procedures.

SORT

Data being reported is processed one record at a time in a 'read record 1, print record 1 information, read record 2, print record 2 information, etc.' cycle. This sequential method of report writing requires that the file coming into the report be in the desired sequence. When the original file is not in the required sequence, the SORT verb allows you to specify that a new copy of the file (or selected records from it) be created and rearranged in report sequence. This rearranged copy becomes the input to the report.

Examples:

```
SORT NAME PLANT#
SORT PART# DESCENDING QUANTITY
SORT BILL-AMT BILL-NO ZIP COMPANY ADDRESS
```

Verb Structure

Word 1	OPT Word 2	OPT Word 3	OPT Word 4	OPT Words 5-24
Dataname	DESCENDING	Dataname	DESCENDING	same as Words 1-2

Coding Rules:

Enter up to 12 data names to specify the desired sequence of the sorted copy of the file. The minor SORT key data name should be entered first after the SORT verb itself and the major SORT key is coded last. A detailed explanation of major through minor concepts is included in the example that follows.

Example:

```
SORT DEPT PLANT DIVISION

The personnel file is sequenced by division, by plant within division, and by
department within plant.

DIVISION    PLANT    DEPT
 01          01       01
 01          01       02
 01          01       03

 02          21       01
 02          21       02

 03          30       01
 03          30       02
```

A report must be printed by pay grade and by name within pay grade. In VISION:Forms, you would code SORT NAME PAY-GRADE. If you were working with a stack of personnel forms, you would look at the PAY-GRADE block on each form and rearrange the stack so that all those in PAY-GRADE 1 are at the top of the stack, those in PAY-GRADE 2 would be next in the stack, and so on. You would do it this way because PAY-GRADE is the most significant (major) SORT key. Your next step would be to concentrate on the forms for PAY-GRADE 1 people, look at the name block, and rearrange the PAY-GRADE 1 group into name sequence. In computer terms, you would SORT the PAY-GRADE 1 group on the least significant (minor) SORT key, in this case the NAME field.

You would continue on, for example, to put the PAY-GRADE 2 group in NAME sequence, the PAY-GRADE 3 group in NAME sequence.

Example:

```
SORT NAME PAY-GRADE
```

As you can imagine, the sorting of thousands of records can be very time consuming, You can do your company a service if you:

- Use the SELECT verb to keep the number of records that must be sorted to a minimum.
- Avoid sorting altogether if sorting produces only a marginally better report.

Examples:

`SORT EMP-NR DEPT PLANT DIVISION`	DIVISION is the major sequence; EMP-NR is the minor sequence.
`SORT VALUE DESCENDING PLANT`	VALUE will be sorted in descending order as minor sequence. PLANT will be sorted in ascending order as the major sequence.

SORTSIZE

VSE Only

It could be necessary for VSE installations, or for particular VISION:Forms programs, that a SIZE=nnnK specification be included in the // EXEC SORT statement generated by VISION:Forms. The SORTSIZE verb causes this specification to be included.

Example:

```
SORTSIZE 80
```

Refer to your data processing department for further information

Verb Structure

```
Word
  1

nnn
```

Coding Rules:

WORD

1 Enter a two-or three-digit numeric value that VISION:Forms is to generate as nnn in the SIZE=nnnK specification in the // EXEC SORT statement.

Example:

```
SORTSIZE 128
```

This causes SORT to generate execute statements as follows:

```
// EXEC SORT,SIZE=128K
```

SPACE

VISION:Forms normally prints a single-spaced report. The SPACE verb allows you to specify a double- or triple-spaced report.

Examples:

```
SPACE DOUBLE
SPACE TRIPLE
```

Verb Structure

```
Word
  1

DOUBLE
TRIPLE
```

Coding Rules:

WORD

1 Enter DOUBLE or TRIPLE to cause the report to be either double- or triple-spaced, respectively.

TITLE

As noted in the PRINT verb description, the data names for the various fields being printed appear at the top of each report page, centered over the actual report data for that element. This technique identifies each data element being printed but does not provide identification for the report as a whole. The TITLE verb enables you to name the report.

Examples:

```
TITLE 'MY COMPANY CUSTOMER LIST'
TITLE ' TAX DEDUCTIONS FOR ALL EMPLOYEES WITH NO DEPENDENTS'
```

Verb Structure

```
Word
  1

'title'
```

Coding Rules:

WORD

1 You can code up to 150 characters. Enter a single quotation mark immediately ahead of the first word in the title and a single quotation mark immediately after the last word in the title.

The first line on each report page is dedicated to printing this title, centered on the line. The date the report is executed automatically prints on the left side of the title, if there is space for it, and the page number prints on the right side, if there is space for it. If there is no space for the date and page number on the first report header line, they appear on the second header line, if there is sufficient space. To prevent a TITLE2 from being generated, see QWOPTION in this chapter.

Example:

```
TITLE 'ANNUAL DEDUCTION SUMMARY'
```

Do not put title information beyond position 71. If the title is longer than 71 characters, it can be continued on the next line in either of two ways. You can indicate a continuation by putting any character in position 72 and the remainder of the title on the next statement. The other way is to split the title into two or more separate title statements.

The following examples both result in the same title on the printed report:

```
TITLE 'EMPLOYEES IN THE DAYTON, COLUMBUS, AND AKRON PLANTS WITH MORE *
THAN TWENTY YEARS SERVICE'
```

```
TITLE 'EMPLOYEES IN THE DAYTON, COLUMBUS, AND AKRON '
TITLE 'PLANTS WITH MORE THAN TWENTY YEARS SERVICE'
```

You can print actual data, such as a department number, in the title on each report page. If you code a data name in the title statement, the actual data is printed in the title on each page of your report. The only rule that applies is that the data name must be coded with a dollar sign ($) immediately before and after it. Also be aware that any character string that is enclosed in dollar signs is assumed to be a data name. Refer to the demos on the installation tape to see how the TITLE verb with data names can be used in conjunction with the TOTAL and BREAK verbs to produce a report.

```
TITLE 'PERSONNEL REPORT FOR DEPT $DEPT-NUM$'
```

Coding Notes:

- Not all data processing installations can support the 150 position title and print line supported in VISION:Forms. Only a few printers are equipped to print a 150-character print line. If you have one of these printers, you must also have the VISION:Report OPTION PRTSIZE set at 150 for VISION:Forms to handle this line length properly.
- If you need to have single quotation marks (or apostrophes) in your title statement, you should code two adjacent apostrophes for each apostrophe that is to appear in the title.

TITLE2

When you want a second title to print at the top of each report page, you should use the TITLE2 verb. All the rules and restrictions that apply to TITLE also apply to TITLE2. The only difference is that the TITLE2 information prints immediately after the title data.

Examples:

```
TITLE2 'XYZ COMPANY'
TITLE2 'CORPORATE OFFICE'
```

TOTAL

Many reports must include totals for groups of data that reflect accumulated values from records in that group. The TOTAL verb allows you to specify data names whose contents are to be accumulated and then printed below the detail print lines when a control field change occurs (see BREAK in this chapter). This variation of totaling infers that you are interested in printing detail information from each record with totals shown at the end of each group. If a summary report consisting of only the total lines is appropriate, see **Summary Report** in the appendix "Examples."

Examples:

```
TOTAL BILLING TAX
TOTAL CALC-DED MO-DED AN-DED WAGE-RT
```

Verb Structure

```
                OPT       OPT       OPT       OPT       OPT
  Word          Word      Word      Word      Word      Words
    1             2         3         4         5         6-12

 Dataname   Dataname  Dataname  Dataname  Dataname  Dataname
```

Coding Rules:

'Datanames' coded in the TOTAL statement must also be coded in the PRINT verb (or PRINT HEADLINE verb in the case of multiple print lines). This is so that assigned print positions can be coordinated and the totals appear directly beneath the corresponding detail information for that data name.

As part of printing the totals for a group, the report automatically shows:

1 A GROUPCOUNT indicating the number of records in the group.

2 Identification of the group to which the totals apply.

Example:

```
FILENAME PERSONEL
SELECT PLANT EQUAL-TO '10, '14', '16'
SORT PLANT
PRINT PLANT EMPNR NAME YTD-ST-TAX YTD-GROSS
TOTAL YTD-ST-TAX YTD-GROSS
BREAK PLANT
TITLE 'PLANTS 10, 14, 16 TAX RECAP'



10/21/93         PLANTS 10, 14, 16 TAX RECAP        PAGE 1

                                            YTD
                                             ST            YTD
  PLANT  EMPNR   ........NAME..............TAX       ...GROSS

   10     1769   DINGEDINE, HOWARD       324.15        1139.42
   10     3611   JOSEPHS, GEORGE         297.00         946.17
   10     2214   SAMUELS, ROGER          402.99        1746.31
  GROUPCOUNT IS 3      FOLLOWING TOTALS ARE FOR PLANT    10
                                        1024.14        3831.90

   14     9316   JOHNSON, CLARA          351.16        1201.39
   14     7335   LOCKRIDGE, CHARLES      398.46        1379.16
  GROUPCOUNT IS 2      FOLLOWING TOTALS ARE FOR PLANT    14
                                         749.62        2580.55

   16     4752   PRIM, ROSALIND          611.79        2441.19
   16     5105   JACKSON, WALTER         572.89        2104.62
   16     7924   LIVINGSTON, STANLEY     204.16        1739.90
  GROUPCOUNT IS 3      FOLLOWING TOTALS ARE FOR PLANT    16
                                        1388.84        6285.71
```

USE PROCEDURE

VISION:Forms is written in a general fashion to handle SELECT-SORT-REPORT requirements on a sequential, ISAM, or VSAM file. Inevitably, many installations have special handling considerations that cannot be anticipated by VISION:Forms. A good example of this is data base input where a single GET is not adequate and a CALL to a data base routine followed by return code checking is required.

The usability of VISION:Forms would be significantly reduced if you had to modify the generated VISION:Forms code before execution. To provide flexibility to you in handling special conditions, the USE PROCEDURE verb allows the data processing department to create blocks of VISION:Report code and catalog them in the VISION:Forms dictionary as procedures.

You can copy one or more procedures into the code generated by VISION:Forms.

Examples:

```
USE PROCEDURE DBREAD1 TO REPLACE GET
USE PROCEDURE DYLINIT AT START-UP
USE PROCEDURE DYLOPT BEFORE REPORT-OPTIONS
```

Refer to your data processing department for further information on what procedures are available and when they should be used.

Verb Structure

Word 1	Word 2	Word 3	Word 4	Word 5
USE PROCEDURE	procedure-name	TO	REPLACE	GET WRITE SELECT-GET REPORT-GET
		AT		START-UP SELECT-STARTUP SELECT-STARTUP
		BEFORE AFTER		OPTIONS SELECT-OPTIONS REPORT-OPTIONS GET SELECT-GET REPORT-GET WRITE

Coding Rules:

WORD

1 Enter USE PROCEDURE.

2 Code the one- to eight-character name of the procedure that is to be included. This word must be the name of a block of code that is in the primary or inline dictionary.

3 Enter either the word TO, BEFORE, AFTER, or AT. This word indicates at what point in the generated VISION:Forms code the procedure is to be placed.

TO	Indicates that the procedure is to REPLACE the generated VISION:Forms code specified by Word 5. Word 4 must always be coded when this word is used.
BEFORE	Indicates that the procedure is to be placed BEFORE the generated VISION:Forms code specified by Word 5.
AFTER	Indicates that the procedure is to be placed AFTER the generated VISION:Forms code specified by Word 5.

AT	Indicates that the procedure is to be placed AT the START-UP or initialization point within the generated VISION:Forms code specified by Word 6.

4 Enter the word REPLACE if Word 2 is TO.

5 Indicates the point within the generated VISION:Forms code where the procedure is to be placed. The placement of the procedure is determined by the contents of Words 3 and 4.

OPTIONS	This allows you to place VISION:Report OPTION statements either BEFORE or AFTER the OPTION statements generated by VISION:Forms. Coding this word causes VISION:Forms to place the procedure in both the SELECT and REPORT sections of the generated code.
START-UP	This allows you to place a procedure at the initialization point within the code generated by VISION:Forms. This keyword is coded prior to any GET of the input file. This is a good place to call a data base handler to OPEN files, etc.
GET	This allows you to REPLACE the GET or place the procedure BEFORE or AFTER the GET of the input file.
WRITE	This allows you to REPLACE the WRITE or place the procedure BEFORE or AFTER the WRITE. This keyword is only active if a VISION:Forms SELECT, CALC, or ATTACH verb is used.

Coding OPTIONS, START-UP, or GET causes VISION:Forms to place the procedure within the SELECT section of the code, if one exists. Otherwise, the procedure is placed within the REPORT section of the code.

Coding SELECT-OPTIONS, SELECT-START-UP, or SELECT-GET causes VISION:Forms to place the procedure only within the SELECT section of the generated code. If a SELECT step was not used, the procedure is ignored.

Coding REPORT-OPTIONS, REPORT-START-UP, or REPORT-GET causes VISION:Forms to place the procedure only within the REPORT section of the generated code. If a REPORT step was not used, the procedure is ignored.

Examples:

This example tells VISION:Forms to REPLACE its GET for the input file with the procedure called PAY731. The procedure REPLACEs either the SELECT-GET or REPORT-GET, depending on which VISION:Forms verbs are used.

```
USE PROCEDURE PAY731 TO REPLACE GET
```

In this example, the procedure PAY833 is placed before the REPORT-GET generated by VISION:Forms. This procedure is only used if a report was requested; otherwise, it is ignored.

```
USE PROCEDURE PAY833 BEFORE REPORT-GET
```

Procedure Considerations

There is no limit to the number of procedures that can be invoked at the same point. If more than one procedure is specified for the same point, the procedures are generated in the same sequence as they were introduced to VISION:Forms.

Do not use the prefix SELECT- or REPORT- unless the procedure can only be used at that point. You should let VISION:Forms place the procedure in either the SELECT or REPORT depending on which VISION:Forms verbs are used.

See **Coding Procedure Code Blocks** in the chapter “Special Procedures” for details on how to write a procedure.

Chapter

4

VSE System Information

How to Install VISION:Forms on Your System

- Catalog VISION:Report. Refer to the VSE VISION:Report Installation Instructions for installing VISION:Report.
- Catalog VISION:Forms. Refer to the VSE VISION:Report Installation Instructions for installing VISION:Forms.
- Once VISION:Forms is installed, you should run the QWDEMOS job that was punched out as part of the VISION:Report installation. We encourage every user, whether you are a new or experienced user of VISION:Forms, to run the QWDEMOS. This job is designed to (1) ensure that VISION:Forms was installed properly, (2) provide examples of loading the VISION:Forms dictionary, and (3) provide examples of writing VISION:Forms programs.

To run QWDEMOS, you need to do the following:

1. Determine if your installation is using a VSAM or ISAM dictionary.
2. If you use a VSAM dictionary, the QWDEMOS job requires approximately 2 cylinders on a 3380 (or 200 continuous blocks for FBA) plus enough room on a VSAM disk volume for 300 dictionary records (2 cylinders on a 3380).
3. If you use an ISAM dictionary, the QWDEMOS job requires 83 continuous tracks on a disk volume.
4. Modify the QWDEMOS JCL to conform to your installation's requirements, and run it.

Required Label Sets

VISION:Forms uses two disk files, the dictionary with file name QW$DICT and the file name QW$FILE. (Samples are provided on the installation tape.) A set of // DLBL and // EXTENT information should be placed in your standard label track for each of these files.

QW$DICT which must be online when VISION:Forms is executed.

```
// DLBL QW$DICT,'QWDICT',,VSAM
```

You can use QW$FILE as a work pack as shown below for assembler/compile work area.

```
// DLBL QW$FILE,'QUIKWRITE,WORKFILE',94/001
// EXTENT SYS006,......,1,0,0019,0100
```

The presence of these label sets allows VISION:Forms to:

Scan the standard label track and determine the SYSnnn for QW$DICT and QW$FILE.

Determine device type(s) for the now known SYSnnn.

Modify VISION:Forms internals accordingly.

Executing VISION:Forms

A 128K partition is required to execute VSE VISION:Forms. In general, VISION:Forms performance is enhanced by a larger partition size, especially if using VSAM and/or SORT.

The following job stream executes VISION:Forms.

```
// JOB QWRITE EXECUTION
// DLBL QW$DICT etc.
// EXTENT
// DLBL QW$FILE etc.
// EXTENT
*
* The above DLBL-EXTENT sets may be placed in your standard
* label tracks and, therefore, may not be required in each job.
*
* $$ PUN DISP=I,CLASS=
*
* The POWER-VSE users may cause punched output to
* be read directly back into a partition for execution.
*
// EXEC QWRITE,SIZE=128K
   .
   .
VISION:Forms statements
   .
/*
/&
```

Users not having VSE/POWER (redirect the punch) capability should consult Non-VSE Considerations in this chapter for a technique allowing them to avoid creating the statements.

This job stream produces four outputs:

A statement image print of VISION:Forms statements.

If the generated VISION:Report is to print a report, then a proof copy of the report format is shown.

A statement image print of generated VISION:Report(s) and SORT.

A punched statement job stream of the generated VISION:Report(s) and SORT.

By including available VSE and/or POWER control statements, you can suppress or redirect the punch job stream:

```
// ASSGN SYSPCH,IGN             Suppress actual punching of the generated
                                job stream.
* $$ PUN DISP=I,...             Immediate execution. Punch output is never
                                literally punched into cards.
```

VISION:Forms Dictionary

The VISION:Forms dictionary is normally created as part of installing the VISION:Forms system. The dictionary contains information of four types.

1 Installation level specifications such as SORT DLBL, ASSG, and EXTENT statements. See Characteristics of the Dictionary in this chapter.

2 File level information such as file TITLE and individual definitions for each element within the file. See **TITLE** and **TITLE2** in the chapter "Writing Verbs."

3 Table descriptions and table entries to be copied into VISION:Forms generated VISION:Reports. See Tables in this chapter.

4 Procedures consisting of frequently used series of VISION:Report statements that can be copied into VISION:Forms generated VISION:Reports. See User Procedures in this chapter and **USE PROCEDURE** in the chapter "Writing Verbs."

Characteristics of the Dictionary

The dictionary must be online whenever VISION:Forms is executed.

The dictionary is a VSAM or ISAM file named QW$DICT and consists of 80 character records blocked to 320 (NRECD=4).

If your dictionary is loaded as a VSAM file, refer to the following example. This example shows how you can DEFINE the VISION:Forms dictionary file for VSAM usage using Access Methods Services (AMS):

```
DEFINE CLUSTER (NAME (user.name) VOL (volser) -
       SHAREOPTIONS (4) SPEED
       FILE (QWDICT) RECSZ (80,80)
       KEYS (25,0) RECORDS (500,200))
```

After the VISION:Forms dictionary is successfully defined as above, you can load the dictionary with all the entries for the files to be used with VISION:Forms.

For ISAM files, DLBL and EXTENT information for index and prime data extents should be placed in standard label tracks appropriate to your installation. The following JCL is provided for your guidance:

```
// DLBL QW$DICT,'QW$DICT',99/365,ISE
// EXTENT SYS004,111111,4,1,6402,1
// EXTENT SYS004,111111,1,2,6403,19
```

Dictionary loading is accomplished presently by a statement to disk, SORT, and ISAM load. The statement to disk and the ISAM load jobs are written in VISION:Report. The SORT is a SORT utility.

To maintain or reload, remove any unwanted dictionary entries, insert any new or replacement entries and run the statement to disk, sort, load.

We suggest you keep your dictionary data grouped by file name for ease of location.

Temporary Replacement of Dictionary Entries

It could be necessary or desirable for an installation to be able to execute VISION:Forms without referring to an actual online dictionary. This can be accomplished by including the dictionary entries within each job. Similarly, when one or a series of online dictionary entries are to be replaced for a particular job, those replacements can be included with that job.

Temporary dictionary entries must be the first statements in the application. They should be preceded by a START-DICTIONARY statement, coded in column 1, and followed by an END-DICTIONARY statement, coded in column 1.

VISION:Forms loads the dictionary entries into a memory table, then sorts them on positions 1-25. You need not assure entry sequence beforehand.

When VISION:Forms logic requires the retrieval of a dictionary entry, the memory table is examined first. If a matching table entry is found, it is used and no reference is made to the online dictionary.

Where a series of associated entries such as PROCEDURE, JCL set, or TABLE is included with the job, the entire PROCEDURE, JCL set, or TABLE must be included with the job.

VISION:Forms uses memory from the end of VISION:Forms itself through the end of the partition for the dictionary memory table and other internal uses. Should a particular application exhaust available memory, diagnostic ERR037 or ERR101 is issued depending on the function that actually runs out of memory. The remedial action in either case is to increase the partition size.

	You Must Include	You Can Include
Once Per Installation	RECORD TYPE JCL STDSORT ASSGN-DLBL-EXTENT INFO FOR SORTIN1, SORTWK1 AND SORTOUT	ENVIRON RECORD
	RECORD TYPE MED - FILENAME = INSTLSTD DEFINE STANDARD SELECT OUTPUT MEDIA & SYSNNN, REPORT INPUT MEDIA & SYSNNN	
For Each File To Be Referenced	RECORD TYPE A - TITLE RECORD TYPE S - FILE CHARACTERISTICS	RECORD TYPE D FIELD DEFINITION, ONE ENTRY PER DATANAME RECORD JCL - ASSGN, TLBL/DLBL INFO FOR THE INPUT FILE. RECORD TYPE MED - SPECIFY SELECT OUTPUT MEDIA & SYSNNN, REPORT INPUT MEDIA & SYSNNN. DEFAULT TO INSTLSTD
Table	RECORD TYPE TD - 1RECORD TO DEFINE THE TABLE	RECORD TYPE TE ONE PER TABLE ENTRY
Procedure		RECORD TYPE NONE - ONE ENTRY PER VISION:REPORT STATEMENT

VISION:Forms is designed to produce complete VISION:Report and SORT programs with proper JCL - ready to run. This can only be accomplished if you provide the installation and file information described under the headings of YOU MUST INCLUDE in the preceding table and detailed on the following pages.

VISION:Forms Dictionary Example

```
CUSTFILE A              AMD CUSTOMER MASTER FILE
CUSTFILE S               SD4C 03500 00700               010
CUSTFILED BC-ADDR        0481-0516
CUSTFILED BC-ADDR3       0553-0588
CUSTFILED BC-NAME        0445-0480
CUSTFILED BILL-AMT       0266-0269-P2C2 (BILLING-AMOUNT)
CUSTFILED BILL-CD        0287-0287
CUSTFILED BILL-MO        0252-0253  0   (BILLING-MONTH)
CUSTFILED C-STAT         0241-0241
CUSTFILED CO-NAME        0001-0036      (COMPANY NAME,)
CUSTFILED CO-NAME2-BLK   0037-0047
CUSTFILED COMM-BASE      0288-0291-P2C2
CUSTFILED CUST-MISC      0589-0661
CUSTFILED CUST-NAME      0001-0036      (CUSTOMER NAME,)
CUSTFILED CUST-NR        0230-0234  0
CUSTFILED MAJOR-PROD     0228-0229
CUSTFILED MEDIA          0243-0243
CUSTFILED MGR-ADDR       0109-0144
CUSTFILED MGR-ADDR2-BLK  0145-0155
CUSTFILED MGR-ADDR3      0181-0216
CUSTFILED MGR-NAME       0073-0108
CUSTFILED MGR-NAME-BLK   0073-0083
CUSTFILED OPER-SYS       0262-0265
CUSTFILED PBILL-AMT      0266-0269-P2C2
CUSTFILED PO-NR          0662-0681
CUSTFILED PROFILE        0245-0245
CUSTFILED REP-CODE       0242-0242
CUSTFILED START-DATE     0246-0251  0
CUSTFILED START-DAY      0248-0249
CUSTFILED START-MONTH    0246-0247
CUSTFILED START-YEAR     0250-0251
CUSTFILED STATUS         0241-0245
CUSTFILED STATUS-CODE    0241-0241
CUSTFILED TECH-ADDR      0337-0372
CUSTFILED TECH-NAME      0301-0336
CUSTFILED TEL-AREA       0270-0272
CUSTFILED TELEPHONE      0270-0286
CUSTFILED USE-NAME       0244-0244
CUSTFILED VERS-LVL       0254-0261
CUSTFILED ZIP-CD         0235-0239
CUSTFILEJCLALSEL    010 // ASSGN SYS010,131
CUSTFILEJCLALSEL    020 // DLBL INF,'CUSTOMER.MASTER'
CUSTFILEJCLALSEL    030 // EXTENT SYS010,VSEIII,1,0,820,200
CUSTFILEJCLF3SEL    010 // ASSGN SYS010,DISK,VOL=volnum,SHR
F3 SET
CUSTFILEMED             SD4C011 SD4C012
```

Non-VSE Considerations

In the majority of cases, the VISION:Forms writer has no interest in literally punching the generated VISION:Reports, desiring only to input a VISION:Forms program and receive a report. To achieve this effect without VSE/POWER, VISION:Forms generates and writes VISION:Reports to a disk work file. At VISION:Forms EOJ, the work file is read as a SYSIN file and the VISION:Report programs are executed. When the VISION:Report programs have been executed, control is returned to the card reader.

The following steps activate the technique described above:

- Include a DLBL/EXTENT set for QW$DPUN in your standard labels. Assembler work area or similar disk areas can be used. Sort work areas should not be used. The following DLBL/EXTENT is shown as an example:

```
// DLBL IJSYSIN,'QW$DPUN'
// EXTENT SYSIN,999999
// DLBL QW$DPUN,'QW$DPUN',0
// EXTENT SYS012,999999,1,0,3600,40
```

- Place an ENVIRON record in the VISION:Forms dictionary with DISK in column 26-29 and multiple X'cuu' entries in column 30-41. The DISK entry specifies use of the work file, while the X'cuu' entries are punched by VISION:Forms into CLOSE SYSIN,X'cuu' statements at the end of the generated VISION:Reports, causing control to be returned to the system. See ENVIRON Record in this chapter for detailed guidance on the ENVIRON record.

Example:

```
ENVIRON   DISK 00C00C00C00C
```

- VISION:Forms input normally ends with the standard /* and /& control statements. If the punch-to-work-file technique is to be used at your installation, VISION:Forms programs should end with:

```
/*
ASSGN SYSIN,00C
/&
```

'A' Records

File Title Dictionary Entry

Positions

1-8	file name
9-10	blank
11	A
12-25	blank
26-80	file title

INCLUDE ONE OF THESE FOR EACH FILE

THIS RECORD IS REQUIRED.

			Sample
1-8	11	26-80	
CUSTFILE		A	AMD CUSTOMER MASTER FILE

See VISION:Forms Dictionary Example in this chapter.

'S' Records

Positions		
1-8	file name	
9-10	blank	
11	S	
12-26	blank	
27-30	Media entries for VISION:Forms 'S' record. One 'S' record required for each file. Choose one of the following:	
	MEDIA	(27-30)
	TAPE	FIXED LENGTH TAPE
	TAPU	UNDEF LENGTH TAPE
	TAPV	VARIABLE LENGTH TAPE
	SD1C	2311 FIXED LENGTH
	SD1V	2311 VARIABLE LENGTH
	SD3C	3330 FIXED LENGTH
	SD3V	3330 VARIABLE LENGTH
	SD4C	2314 FIXED LENGTH
	SD4V	2314 VARIABLE LENGTH
	SD5C	3340 FIXED LENGTH
	SD5V	3340 VARIABLE LENGTH
	SD6C	3350 FIXED LENGTH

SD6V	3350 VARIABLE LENGTH
*SD7C	3375 FIXED LENGTH
*SD7V	3375 VARIABLE LENGTH
SD8C	3380 FIXED LENGTH
SD8V	3380 VARIABLE LENGTH
SD9C	3390 FIXED LENGTH
SD9V	3390 VARIABLE LENGTH
IS11	2311 ISAM
IS14	2314 ISAM
IS30	3330 ISAM
IS40	3340 ISAM
FBAC	3310, 3370 FBA, FIXED LENGTH
FBAV	3310, 3370 FBA, VARIABLE LENGTH
KSDS	VSAM
ESDS	VSAM
RRDS	VSAM
USER	User file such as DAM, DL/I, IDMS, TOTAL, etc. A procedure must be written to access USER type files and its name must be specified in positions 72-79.

*3375's can be coded for any disk device introduced by IBM after the 3375's

32-36 Block size. Code a five-digit number indicating the block size of the file. For variable or undefined files, enter the maximum block size that can occur. For KSDS, ESDS, RRDS, and USER media types, this field should be set equal to the logical record size.

38-42 Logical record size. Code a five-digit number indicating the size of logical records on the file. For variable or undefined files, enter a record size that is the largest that can occur.

44 Type label information. Applies to tape files only. Enter S for standard files or N for no labels or non-standard labels. If left blank, standard labels are assumed. Could be left blank for disk, VSAM, and USER files.

46 Applies to tape files only. If no entry is made, the input file being referenced is rewound at open time and rewound at close time. If this is not appropriate, make one of the following entries:

- U — Rewind at open time; rewind and unload at close.
- N — No specific action at open time (next block on tape is first INF block); no specific action at close time, that is, tape is left in position.

48 Tape mark information. Applies to tape files only. If the input tape file is not labeled or has non-standard labels and not preceded by a tape mark, code N; otherwise, leave blank.

49-56 Standard label track file names. If // TLBL or // DLBL and // EXTENT sets are present in the standard label track, code the file name under which that set is cataloged. For VSAM files, this field must contain the file name; otherwise, leave the field blank.

58-60 Multiple files. Applies to tape files only. If your input contains more than one generation of that file, enter a three-digit number indicating how many files (not reels) there are; otherwise, leave blank.

62-64 Enter nnn of the input SYSnnn on which your file resides when the generated VISION:Report program is executed. This entry should contain 000 for VSAM and USER files.

66 Enter the letter G to cause generic assignments to be generated for multi-step jobs.

68-70 VISION:Report area name. This field is normally blank, which indicates that the file is used for INF, unless overridden by a FILENAME verb. Any VISION:Report area name is valid except OFA. See the chapter "**Advanced Techniques**" for a discussion of using multiple files in VISION:Forms.

72-79 Default procedure name to be used whenever this file is referenced. If this field is blank, VISION:Forms issues a standard GET for the file unless the file media has been coded as KSDS, ESDS, or RRDS. In the case of VSAM, VISION:Forms issues CALLs to QUIKVSAM to handle the reading. The procedure must be present in the VISION:Forms dictionary. See the chapter "**Special Procedures**" for information about how to write a procedure.

Sample

```
1-8         11   27-30    32-36    38-42          62-64   66

CUSTFILE    S    SD3C     03500    00700          010     G
PROJECT     S    KSDS     00100    00100          000
TOTDB       S    USER     00300    00300          000
```

'D' Records

Data Element (Field) Definitions

Prepare one data element (field) definition for each field (logical collection of data elements) to which VISION:Forms users are to make reference.

For instance, you could have a part number (235 3860) type field in each of your records. We suggest you prepare three names and definitions for it: (1) part number to reference 235 3860, (2) part-prefix to reference the prefix or first three positions (235), and (3) part-suffix to reference the suffix (3860) or last four positions.

Data within a record can be defined, redefined, and referenced with as many different VISION:Forms data element definitions as desired.

Positions

1-8	file name
9	D
10-11	blanks
12-25	data name-field name Enter a unique name of up to 14 characters in length with no embedded blanks. The first character must be alphabetic. The balance of the data name can be any alphabetic combination, numeric digits 0 through 9, and hyphens. The data name assigned here is potentially used by non-DP oriented people, so mnemonic significance is particularly important. The data name ordinarily is used as the column header when this field appears in a report, so, if the field data requires relatively few print positions, a short data name (or one with interspersed hyphens) causes the field to tie up fewer print positions in report composition.
27-37	VISION:Report field definition.
27-30	Four-digit number indicating the leftmost record position of the field.
31	Enter a dash (-).
32-35	Four-digit number indicating the rightmost record position of the field.
36-37	If the field is packed decimal, enter -P; otherwise, leave blank.

38	Print decimals. If the field is numeric and can be printed, enter the number of decimal positions to be shown in print. If field is numeric and has no decimals, enter 0; otherwise, leave blank.
39	If the field is numeric and can be printed, enter one of the following codes:

C	Edit with zero suppression, commas, and decimal point.
E	Edit with European punctuation.
N	Edit without zero suppression and commas; decimal point is included if necessary.
Edit Special Character	Enter one of the special VISION:Report expanded edit characters. This character can be any alphabetic character except C, E, or N. Refer to the VISION:Report Reference Manual for details.

40	Calculation decimals. If this field can be used as a value in a CALC or SELECT arithmetic statement, enter (0-9), the number of conceptual decimal positions in the field.
42-43	Totaled field size. When a field is totaled, VISION:Forms normally increases the original field size by two digits. If this is too large or too small for this field, enter an odd number between 01 and 15. This number must not be less than the original field size. VISION:Forms then totals the field with only this number of digits. However, if the field is also being averaged, this field is ignored.
44-50	Not used at this time.
51-80	Heading to be used when a field is printed; optional. Start heading in column 51 or beyond. Enclose in parentheses. You can place a heading here which is always used instead of the data name, unless an override heading is specified in the actual request.

Sample

```
1-8        9  12-25     27-30  31     32-35
CUSTFILE   D  CO-NAME   0001   -      0036
```

See VISION:Forms Dictionary Example in this chapter.

Job Control Language

After VISION:Forms has verified all statements, it generates VISION:Report code to perform the request. Depending on what VISION:Forms statements were used, one or more VISION:Report programs are generated. VISION:Forms permits JCL for a file to be stored in the dictionary. This JCL is retrieved and placed around the generated VISION:Report code whenever the file is accessed, thus allowing ready to run VISION:Report code to be produced. It is recommended that only one job step be generated.

If VISION:Forms is unable to find the JCL for a file in the dictionary, it continues to run without error. You have to add the necessary JCL after the generated steps are created, before they can be executed.

The JCL needed depends on which VISION:Report steps were generated and whether or not the QUIKSORT feature is active. The three steps that can be generated are:

SELECT	A VISION:Report program is produced whenever VISION:Forms finds a SELECT, CALC, or ATTACH statement.
SORT	A SORT phase is produced whenever VISION:Forms finds a SORT statement.
REPORT	A VISION:Report program is produced whenever VISION:Forms finds a PRINT, TOTAL, ONBREAK, PSELECT, etc. statement.

There are four different types of JCL that can be stored in the dictionary. Again, which type is needed depends on what statements VISION:Forms found. The different types are:

SEL	Used by the SELECT step. You can omit this type of JCL if the file name is in the standard label area.
SOI	Used by the SORT for the input file. It is also used and modified by the SELECT step for use as the OFA file.
SOW	Used by the SORT for the sort work files.
SOO	Used by the SORT for the output file. It is also used and modified by the REPORT step for use as its INF file.

JCL Requirement Summary Table

...QUIKWRITE REQUEST...				#	.....JCL REQ'D.....			
SELECT	SORT	REPORT	QUIKSORT	STEPS	SEL	SOI	SOW	SOO
yes	yes	yes	yes	1	yes	no	yes	no
yes	yes	yes	no	3	yes	yes	yes	yes
yes	yes	no	yes	2	yes	yes	yes	yes
yes	yes	no	no	2	yes	yes	yes	yes
no	yes	yes	yes	1	yes	no	yes	no
no	yes	yes	no	3	yes	yes	yes	yes
no	no	yes	n/a	1	yes	no	no	no
yes	no	yes	n/a	1	yes	no	no	no
yes	no	no	n/a	1	yes	yes	no	no

JCL for the Master File

Enter a set of // ASSGN SYSnnn,CUU and a // TLBL or // DLBL and // EXTENT information for the master file.

This is used to generate the needed JCL and VISION:Report statements to read the master file.

Positions

1-8	file name
9-11	JCL
12-13	Partition ID BG, F1, F3, etc., or AL (AL stands for all). You can enter one common set to be used for all partitions or you can enter a set relative to each partition to be used, if different ASSGN and TLBL/DLBL are needed.
14-16	SEL. This keys the JCL to be used for selection purposes for the file name in positions 1-8.
17-24	Enter a sequence number that causes the records to sequence uniquely and in the desired sequence of selection.
25	Continuation if needed. Put an X in this column if the JCL does not fit completely on this statement. Continue this JCL in the next statement.

26-80 Enter the JCL required to access your file.

```
// ASSGN SYSnnn,CUU          or
// TLBL                      or
// DLBL                      set
// EXTENT
```

Needed for your master if the

```
// TLBL INF,'MASTER.FILE'
```

or

```
// DLBL INF,'MASTER.FILE'
// EXTENT
```

If information stored on the STDLABEL tracks and LBL= information has been provided in the file characteristics, 'S' record, // TLBL and // DLBL-EXTENT records are not required for the input master file.

Sample

```
CUSTFILEJCLALSEL    005 // OPTION NODUMP
CUSTFILEJCLALSEL    010 // ASSGN SYS010,DISK,VOL=volnum,SHR
CUSTFILEJCLALSEL    020 // DLBL INF,'C----'
CUSTFILEJCLALSEL    030 // EXTENT SYS010,PAC,1,0,3204,120
```

JCL for the Sort

We recommend that one standard installation-wide set of SORT information be used, if possible. You can enter it under the file name STDSORT; if you do, no other SORT information is considered. Use the file name STDSORT and 'AL' in 12 and 13 to accomplish this. See SAMPLE in this chapter.

If SORT information is needed by/for each partition you can enter a set of SORT information under the file name STDSORT and use positions 12 and 13 of the dictionary entries to specify which partition the set is to be used for.

If special SORT information is needed for one given file (e.g., a large data base, or multi-volume disk or tape file) where selected output and SORT work areas need be large, you can enter SORT information at the file level. See EXAMPLES in the dictionary supplied with the system.

Sample

```
      1-16          17-24            26-80

STDSORT JCLALSOI    010     // ASSGN SYS002,152
STDSORT JCLALSOI    020     // DLBL SORTIN1,'STDSORT.INPUT',0
STDSORT JCLALSOI    030     // EXTENT SYS002,999999,1,0,0019,1000
STDSORT JCLALSOO    040     // ASSGN SYS001,152
STDSORT JCLALSOO    050     // DLBL SORTOUT,'STDSORT.OUTPUT',0
STDSORT JCLALSOO    060     // EXTENT SYS001,999999,1,0,1019,1000
STDSORT JCLALSOW    070     // ASSGN SYS003,152
STDSORT JCLALSOW    080     // DLBL SORTWK1,'STDSORT.WORK',0
STDSORT JCLALSOW    090     // EXTENT SYS003,999999,1,0,2018,2000
```

SORT JCL consists of three distinct functions, each identified by the following reserved characters coded in columns 14-16:

'SOI' SORT INPUT JCL Definitions
'SOO' SORT OUTPUT JCL Definitions
'SOW' SORT WORK JCL Definitions

JCL Usage Hierarchy

At VISION:Forms initialization, the system determines the partition in which it is executing. This is presumed to be the partition in which the generated VISION:Reports execute.

If VISION:Forms is executed in one partition, but the generated VISION:Reports are to be executed in a different partition, a PARTITION verb must be included to indicate the VISION:Report execution partition. If a PARTITION verb is encountered, the partition it specifies is considered as the execution partition.

When JCL is to be retrieved, VISION:Forms:

- Concatenates file name, the constant JCL, execution partition, and (SEL for the input file, SOI for the SORTIN1, SOW for SORTWK1, and SOO for SORTOUT) into a 16-position key, which is then used to search the dictionary for a set of JCL with a matching generic key. If a matching JCL set is found, it is used; if not, different keys, as shown below, are used to search for the JCL.
- Concatenates file name, JCL, AL, and function and searches again. The AL scope entry indicates the JCL that applies across all partitions.
- If an AL set is not found and the search has been for input file (SEL) JCL, it is presumed that the JCL for the input file is present in a STDLABEL, or that none is to be created for this file. No further attempts are made.
- If an AL set is not found and the search has been for any of the SORT set, VISION:Forms concatenates STDSORT, JCL, execution partition, and function and looks for a set of standard SORT JCL for the partition.
- The final search for SORT JCL is STDSORT/AL which would retrieve the single set of JCL coded for the entire installation. If none is found, no JCL is created for that SORT.

Selection and Reporting Media Record

A media record is required in the dictionary to assign the proper devices for selecting, sorting, and reporting. An installation version is a must. One can exist for each file or master file if desired.

Positions

1-8	file name or INSTLSTD
9-11	MED
12-13	blanks or partition id, (BG, F2, etc.)
14-25	blank
26-29	SELECT output media. Enter the media on which SORTIN1 SORT work area resides. See 'S' Records in this chapter for media codes.
30-32	SELECT SYSNR. The SELECT run OFA statement is customized to the SYSnnn of SORTIN1. Should no SORT verb be included, then the SYSnnn coded here is used in the OFA statement.
34-37	Report input media. Enter the media on which sortout SORT work area resides. This media is used in the media field of the report run INF statement. See 'S' Records in this chapter for media codes.
38-40	REP IN SYSNR. If a SORT verb is coded, the report run statement contains the SYSnnn of SORTOUT. If no SORT is present, the SYSnnn coded here is used.

Sample

```
  1-8      9-11   26-29   30-32   34-37   38-40
CUSTFILE   MED    SD3C    010     SD3C    011
INSTLSTD   MED    SD3C    0l0     SD3C    011
```

ENVIRON Record

This record makes available to VISION:Forms certain installation standards that are applied to all VISION:Forms, unless overridden for a particular VISION:Forms by an inline dictionary entry.

The initial use of ENVIRON is for DOS (non-VSE) users to specify to which media the generated VISION:Forms statements are to be written. ENVIRON records also signal VISION:Forms that translation from a foreign language can be necessary.

ENVIRON records are optional.

Positions

1-7	ENVIRON
26-29	VISION:Forms writes generated VISION:Reports to card image. Under VSE, this punched queued output is usually renamed as an input job stream and no cards are actually punched. To achieve this same effect under DOS, the Systems Programmer should enter DISK in this field. Entry of DISK here implies punching to a disk work file with attendant STDLABEL and ASSGN SYSIN considerations.
30-32	Enter the X'cuu' of the background partition SYSIN device.
33-35	Enter the X'cuu' of the F1 partition SYSIN device.
36-38	Enter the X'cuu' of the F2 partition SYSIN device.
39-41	X'cuu' of the F3 partition SYSIN device.
42-43	If VISION:Forms are to be translated from an alternate language into English, code the 2-character acronym shown below for that language:

FRENCH	FR
GERMAN	GR

User Procedures

A generalized system such as VISION:Forms cannot anticipate unusual requirements that would be necessary for specific installations. The USE PROCEDURE verb gives you the ability to insert one time VISION:Report routines to handle these requirements and copy them into VISION:Forms programs as needed. This section covers the coding of a procedure. See the chapter "**Special Procedures**" for examples and a discussion on how to write a procedure.

Positions

1-8	Enter a one-to eight-position procedure name by which this procedure is referenced in PROCEDURE verbs. The first character of each procedure name must be alphabetic. The procedure name must not be duplicated elsewhere in the dictionary.
17-24	Enter an ascending sequence number to provide unique identifiers for each entry in a procedure name. This entry is for dictionary sequence control only and is not associated with the VISION:Report sequence numbers discussed below.

25	As seen under 30-80 in this chapter, there are 55 positions available for coding VISION:Report statements. Some VISION:Report verbs can exceed this length and, if this is the case, you should continue that verb in the next procedure name entry and place a non-blank in position 25 of the first entry.
26-28	VISION:Report sequence number. Make no entry in this field unless the statement being coded is a transfer point. Where used, the sequence number should be three digits.
30-80	VISION:Report statement. Code the VISION:Report statement exactly the same as you would for VISION:Report. If one entry does not contain the entire statement, code through position 80, enter a non-blank character in position 25 of this entry, and continue your statement in position 26 of the following entry. Transfer point numbers should be coded as one-to three-digit numbers and must be present as VISION:Report sequence numbers someplace within this procedure. Each time this procedure is copied into an application, these transfer points (and their associated sequence number fields) are adjusted as appropriate for that application.

Should a procedure include logical transferring within the routine, the need for an end-of-procedure exit point is implied. This can be achieved by coding an exit statement as the last statement of the procedure and including a sequence number. You can then exit the procedure and rejoin VISION:Forms generated code by GOTO nnn where nnn is the sequence number of the exit statement. A procedure name should not be invoked more than once in a particular VISION:Forms.

Tables

Data Tables That Might Be Invoked Frequently Through VISION:Forms

SELECT ... ON-TABLE can be cataloged to the dictionary and copied into the generated program through the QWTABLE verb.

Cataloged table information consists of:

- A single entry containing the table specifications as described in Table specifications entry: in this chapter.
- An optional record containing column header data to be used when the table function is used on a PRINT verb.
- A dictionary entry for each table entry.

Table Specifications Entry

Positions	
1-8	Table name that is cited in the QWTABLE DICTIONARY verb. This name cannot duplicate any existing dictionary file name.
9-10	TD
26-29	Maximum number of table entries. This data is reproduced into a VISION:Report TABLSPEC statement as maximum entries. If additional entries are added to the table directly through the VISION:Forms, this number should reflect the maximum entries from both sources.
31-32	Argument position. Code a two-digit number specifying in which column the table argument starts. As seen under Table Data Entries in this chapter, the actual table data is keyed starting in position 26, but this should be considered as the equivalent of column 1. In short, the usual entry here is 01.
34-35	Argument length. Enter a two-digit number indicating the number of characters in each table entry.
37	Argument type. Enter the letter 'P' to denote that the argument is packed format. This is required only when the data name used in a SELECT ... ON-TABLE is packed. If left blank, the data and field are both considered to be in standard EBCDIC format.
39-40	Function starting position. Code a two-digit number specifying in which column the table function starts.
42-43	Function length. Code a two-digit number to indicate the length of the function in the table entry.
45-58	Function data name. Enter a data name of up to 14 characters that is EQUATEd to the table function after selection has occurred. This operand is optional, but we suggest you code it unless your table has multiple functions that must be accessed separately.

Table Function Header Entries

Positions	
1-8	The same table name as keyed into the table specification entry.
9-11	TDH
26-55	Code up to 30 characters enclosed in parentheses. This is used as the default column header whenever the table function is requested to be printed.

Table Data Entries

Positions	
1-8	The same table name as coded into the table specification entry.
9-10	TE
23-25	Code an ascending sequence number, e.g., first TE 010, second TE 020, etc. This data is required only to satisfy the sequencing requirements of the dictionary.
26-80	Enter up to 55 positions of table data. Entries should be keyed exactly as though they were going directly into VISION:Report. VISION:Forms creates one statement for each TE entry with dictionary positions 26-80 being coded into columns 1-55.

Chapter

5

MVS System Information

How to Install VISION:Forms on Your System

VISION:Forms is distributed on the same labeled tape that contains VISION:Report. Job Control Language statements to accomplish the VISION:Forms installation were loaded with the installation of VISION:Report. Refer to the VISION:Report MVS Installation Instructions for a listing of that installation JCL. If you did a complete VISION:Report installation, the VISION:Forms installation JCL was loaded into the library containing VISION:Report optional materials and sample programs. The member name of the VISION:Forms installation JCL is QWINSTL. You must have installed VISION:Report completely prior to installing VISION:Forms.

It is assumed that you have successfully installed VISION:Report. VISION:Forms will not work without VISION:Report.

In order to customize the installation JCL to meet your organization's requirements (JOB card parameters, data set name specifications, etc.), you should copy this JCL to a convenient data set or library and retain the sample JCL for reference or future use. Data for installing VISION:Forms is contained on the VISION:Report distribution tape, which you should have copied upon receipt. That tape should be mounted for VISION:Forms installation. There are two files required for VISION:Forms installation, which are described in the VISION:Report Installation Instructions. They are both in IEBCOPY (unload) format. The first VISION:Forms file contains the VISION:Forms load library. The second VISION:Forms file contains sample programs, sample dictionary and data, sample JCL procedures, and sample job execution.

After customizing the installation JCL, execute the module using the VISION:Report distribution tape mentioned above.

The sample material library created contains two VISION:Forms execution procedures: QWRUN and QWPUNCH. These procedures should be customized to your organization's requirements (including appropriate data set names) and placed into an appropriate procedure library. You should avoid using SYS1.PROCLIB for these procedures. This step might not be necessary if VISION:Forms has been previously installed. It is recommended that the procedures be copied before being customized so that the originals can be retained for future reference.

Once VISION:Forms is installed, you should run the DEMO jobs that were punched out as part of the VISION:Report installation. We encourage every user, whether you are a new or experienced user of VISION:Forms, to run the DEMO jobs. The DEMO jobs are designed to (1) ensure that VISION:Forms was installed properly, (2) provide examples of loading the VISION:Forms dictionary, and (3) provide examples of writing VISION:Forms programs. You will need to modify the JCL and data set names to conform to your installation's requirements.

If you are a new VISION:Forms user, you will need to run steps 1 and 2 below. If you already have an existing Dictionary, skip steps 1 and 2 and go directly to run step 3.

1 Run DICTCRTE to define the VISION:Forms VSAM dictionary.

2 Run DICTLOAD to load a sample VISION:Forms VSAM dictionary.

To run the DEMO jobs, you need to do the following:

3 Modify QWTSTJCL to conform to your installation's requirements. QWTSTJCL will run DEMO jobs QW DEMO01-06 using the QWRUN procedure. DEMO07 is for users who have the optional feature, VISION:Interface for DB2 with VISION:Report.

Executing MVS VISION:Forms

MVS VISION:Forms can run in one of two modes:

Punch mode — In this mode, all the VISION:Report and SORT control statements are punched. You must then place all required JCL with each VISION:Report (pull and list) and the SORT. This mode is useful if the VISION:Reports generated are to have additional coding added such as special data base CALLS. We do not recommend this method.

If the CALC verb is used, the output LRECL will be larger than the input file LRECL. The user must determine this larger LRECL (highest record position referenced in generated VISION:Reports) and the adjust LRECL values in the JCL.

Run mode	In this mode, VISION:Forms and all generated VISION:Reports and SORT are executed in the same step: compile, load, and go. All necessary JCL must be included when VISION:Forms is invoked.

Job Control Statements

QWSYSIN	Input file for user request statements and inline dictionary.
QWPRINT	VISION:Forms message data set. Data set contains: ■ A print file of VISION:Forms statements. ■ If the generated VISION:Report is to print a report, a proof copy of the report format is shown. ■ A statement image file of the generated VISION:Reports and SORT.
QWWORK	Work file. Must be assigned to a disk unit.
QWPUNCH	If the output of VISION:Forms is to be punched, this should be assigned to a physical punch or a SYSOUT class that handles punched data sets. Otherwise, assign this file to a disk. It is used as a work file.
QWDICT	VISION:Forms VSAM or ISAM dictionary data set. Required if an inline dictionary is not being used.
QWUDICT	Work file used to temporarily hold the inline dictionary. Must be assigned to a disk unit. Required if an inline dictionary is being used.

```
SORTLIB
SYSOUT
SORTWK01
SORTWK02
SORTWK03
```

The above SORT JCL is required if an inline dictionary is being used and/or VISION:Forms is being run in 'RUN MODE'. Note that the use of the SORTLIB statement depends on whether or not an IBM SORT is being used. Refer to your installation standards for rules. The following JCL statements are required only if VISION:Forms is run in 'RUN MODE'

SYSIN	Work file for VISION:Report. Should be assigned to a disk unit.
QWINF	A MASTER file that VISION:Forms requests are to be run against.

QWPLIST	Print file for the 'PULL' VISION:Report execution.
QWOFA	Work file that is used to contain output records from the VISION:Report 'PULL' execution. If the QUIKSORT feature is active, QWOFA should be allocated as a DUMMY file.
QWLLIST	Print file for the 'LIST' VISION:Report execution. This data set contains the requested report.

JCL Considerations

- The amount of space allocated to SORTWK01, SORTWK02, SORTWK03, and QWOFA is directly related to the number of records pulled from QWINF based on your SELECT statements and whether or not QUIKSORT is active. If QUIKSORT is being used, the QWOFA can be allocated as a DUMMY file. If not, the size of the records in QWOFA is the same or greater than the size of those in QWINF. The block size is no greater than 3000 bytes except in a case where RECSIZE is greater than 3000. In this case, BLKSIZE is equal to RECSIZE.
- The EXEC JCL statement is either:

```
//STEP10 EXEC PGM=QKWRITE                                    (PUNCH MODE)
```

or

```
//STEP10 EXEC PGM=QKWREXEC                                   (RUN MODE)
```

- Two procedures are distributed with VISION:Forms that should be modified for each installation's use. These procedures are QWPUNCH (for punch mode) and QWRUN (for run mode). Note that if you use a VSAM or ISAM dictionary, you must add a QWDICT DD statement to each procedure that should look like one of the following DD statements:

```
//QWDICT DD DSN=QW.VSAMDICT,DISP=SHR
```

or

```
//QWDICT DD DSN=QW.ISAMDICT,DISP=SHR,DCB=DSORG=IS
```

MVS VISION:Forms Dictionary

The VISION:Forms DICTIONARY for MVS can either be a VSAM or ISAM file and/or it can be included as part of the input stream when VISION:Forms is executed. The dictionary contains information of three types:

1 File level information such as file title and individual definitions for each element within the file.

2 Table descriptions and table entries to be copied into VISION:Forms generated VISION:Report programs.

3 Procedures consisting of frequently used series of VISION:Report statements that can be copied into VISION:Forms generated VISION:Report programs.

Each of these types is covered in detail later in this manual.

Characteristics of the VSAM or ISAM Dictionary

1 The dictionary must be online whenever VISION:Forms is executed.

2 The dictionary consists of 80-character records, blocked at any size you want.

3 Dictionary loading is accomplished presently by a SORT and VSAM/ISAM load. The VSAM/ISAM load job is written in VISION:Report. The SORT is a SORT utility.

4 To maintain or reload, remove any unwanted dictionary entries, insert any new or replacement entries, and run the SORT and load.

We suggest you keep your dictionary data grouped by file name for ease of location.

5 Should you choose to store your dictionary on VSAM rather than your library system (CA-PANVALET, CA-LIBRARIAN, etc.) or PDS, you need to load your dictionary data to VSAM or ISAM; key length is 25 (column 1-25).

6 You also need to add a QWDICT DD statement to your procedures or include one within each request.

7 If your dictionary is loaded as a VSAM file, refer to the following example. This example shows how you can DEFINE the VISION:Forms dictionary file for VSAM usage using Access Methods Services (AMS). The example is on the sample file of VISION:Forms, member name DICTCRTE. The example shows the definition of the VSAM dictionary as well as the loading of the VSAM dictionary, member name DICTLOAD.

```
//DICTCRTE JOB  (ACCOUNTING INFO),
//    MSGCLASS=X
//*

**************************************************
//*
//* CREATE VISION:FORMS  VSAM DICTIONARY
//*
//*
**************************************************
//DICTCRTE EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DELETE (ISPQW.DICT) CLUSTER
 DEFINE CLUSTER (NAME (ISPQW.DICT)     VOL(ISP803) -
        SHAREOPTIONS(4) SPEED                              -
                        RECSZ(80,80)   KEY (25,0) )        -
        DATA ( NAME (ISPQW.DICT.DATA)                 -
        CYL (2,1) FREESPACE (20,5) )                       -
     INDEX ( NAME (ISPQW.DICT.INDEX) )
/*
//



//DICTLOAD JOB  (ACCOUNTING INFO),
//    MSGCLASS=X
//* **************************************************
//*
//* SAMPLE OF LOADING THE VISION:FORMS  VSAM DICTIONARY
//*
//* **************************************************
//SORT     EXEC PGM=SORT
//SORTLIB  DD   DISP=SHR,DSN=SYS1.SORTLIB
//SYSOUT   DD   SYSOUT=*
//SORTWK01 DD   UNIT=DISK,SPACE=(CYL,(1,1))
//SORTWK02 DD   UNIT=DISK,SPACE=(CYL,(1,1))
//SORTWK03 DD   UNIT=DISK,SPACE=(CYL,(1,1))
//SYSIN    DD   *
 SORT FIELDS=(1,25,CH,A),SIZE=E300
 RECORD TYPE=F,LENGTH=80
/*
//SORTIN   DD   *
DYLAKOR   A              QUIKWRITE FILE
DYLAKOR   S               SD8C 00080 00080 S                 002
DYLAKOR D  FIRST-WORD     0001-0004
DYLAKOR D  SECOND-WORD    0006-0007
DYLAKOR D  THIRD-WORD     0009-0012
DYLAKOR D  FOURTH-WORD    0014-0014
/*
//SORTOUT  DD  DISP=(,PASS),UNIT=DISK,SPACE=(CYL,(1,1)),
//             DCB=(BLKSIZE=800,LRECL=80,RECFM=FB),
//             DSN=&RECORDS
//*
//VSAMREPR EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SORTRECS DD   DISP=OLD,DSN=&RECORDS
//SYSIN    DD  *
 REPRO -
 INFILE(SORTRECS) -
 OUTDATASET(ISPQW.DICT)
/*
//
```

A sample of a SORT and VISION:Report to load the ISAM dictionary is included for your reference.

Sample ISAM Load

```
ISAMDICT JOB  (ACCOUNTING INFO),
//    MSGCLASS=X
//* ****************************************************
//*
//* CREATE VISION:FORMS  ISAM DICTIONARY
//* AND LOAD IT
//*
//* ****************************************************
//SORT     EXEC PGM=SORT
//SORTLIB  DD   DISP=SHR,DSN=SYS1.SORTLIB
//SYSOUT   DD   SYSOUT=*
//SORTWK01 DD   UNIT=DISK,SPACE=(CYL,(1,1))
//SORTWK02 DD   UNIT=DISK,SPACE=(CYL,(1,1))
//SORTWK03 DD   UNIT=DISK,SPACE=(CYL,(1,1))
//SORTOUT  DD  DISP=(,PASS),
//             DCB=(BLKSIZE=800,LRECL=80,RECFM=FB),
//             DSN=&QWCARDS,
//             SPACE=(CYL,(1,1)),
//             UNIT=DISK
//SYSIN    DD   *
 SORT FIELDS=(1,25,CH,A),SIZE=E300
 RECORD TYPE=F,LENGTH=80
/*
//SORTIN   DD   *
DYLAKOR   A              QUIKWRITE FILE
DYLAKOR   S               SD8C 00080 00080 S                 002
DYLAKOR D  FIRST-WORD     0001-0004
DYLAKOR D  SECOND-WORD    0006-0007
DYLAKOR D  THIRD-WORD     0009-0012
DYLAKOR D  FOURTH-WORD    0014-0014
/*
//*
//LOADDICT EXEC  QJTEST
//SYSUT1   DD  DISP=OLD,DSN=&QWCARDS


//SYSUT2   DD  DISP=(,CATLG),
//             DCB=(BLKSIZE=320,LRECL=80,RECFM=FB,DSORG=IS),
//             DSN=&QWCARDS,
//             SPACE=(CYL,(1,1)),
//             UNIT=3330
//SYSIN    DD  *
010 GET
    MOVE INF1-80  TO  PRT1
    MOVE INF1-80  TO  OFA1-80
    PRINT
    WRITE OFA
    GOTO 010
999 END
/*
//
```

Characteristics of the Inline Dictionary

1 The dictionary is placed in front of all other VISION:Forms statements.

2 The dictionary must be preceded by a START-DICTIONARY statement (columns 1-l6) and followed by an END-DICTIONARY statement (columns 1-14).

3 Dictionary entries do not have to be in sequence except as noted in rule 2 above.

4 Multiple dictionaries are permitted as input in the same run as long as rules 1 and 2 are observed. Note that you cannot have entries with the same key columns (1-25).

5 The dictionary can reside in a PDS, if desired. Concatenate it to QWSYSIN (see JCL Examples in this chapter).

6 The dictionary can also reside in CA-PANVALET, CA-LIBRARIAN, or any other source library system. To use these systems, you must first copy the dictionary to a temporary data set and pass it to VISION:Forms to be concatenated to QWSYSIN (see JCL Examples in this chapter).

If you want to store dictionary data on a data set, such as PDS, it would be desirable to have sequence numbers in positions 76-80, so that a utility, such as IEBUPDTE, can be used for updating.

If you are going to store VISION:Report procedures therein, it is necessary to place a /* in a column prior to 76. VISION:Report then considers sequence numbers as comments rather than as part of the VISION:Report statement.

	You Must Include	You Can Include
For Each File To Be Referenced:	one file characteristic record	record type 'S'
	a field definition record for each data field that can be referenced	record type 'D'
For Each Table:	one record to define the table characteristics	record type 'TD'
	a table entry record, one per table entry	record type 'TE'
For Each Procedure:	one entry for each QJ statement in the procedure	none

If you are using an inline dictionary, you must follow all the following rules for syntax, plus a START-DICTIONARY statement must be placed in front of the dictionary and an END-DICTIONARY statement must be the last statement in the dictionary.

VISION:Forms Dictionary Example

```
CUSTFILE  A               AMD MASTER FILE
CUSTFILE  S               SD4C03500 00700
CUSTFILED  BC-ADDR        0481-0516
CUSTFILED  BC-ADDR3       0553-0588
CUSTFILED  BC-NAME        0445-0480
CUSTFILED  BILL-AMT       0266-0269-P2C2 (BILLING-AMOUNT)
CUSTFILED  BILL-CD        0287-0287
CUSTFILED  BILL-MO        0252-0253  0   (BILLING-MONTH)
CUSTFILED  C-STAT         0241-0241
CUSTFILED  CO-NAME        0001-0036      (COMPANY NAME,)
CUSTFILED  CO-NAME2-BLK   0037-0047
CUSTFILED  COMM-BASE      0288-0291-P2C2
CUSTFILED  CUST-MISC      0589-0661
CUSTFILED  CUST-NAME      0001-0036      (CUSTOMER NAME,)
CUSTFILED  CUST-NR        0230-0234  0
CUSTFILED  MAJOR-PROD     0228-0229
CUSTFILED  MEDIA          0243-0243
CUSTFILED  MGR-ADDR       0109-0144
CUSTFILED  MGR-ADDR2-BLK  0145-0155
CUSTFILED  MGR-ADDR3      0181-0216
CUSTFILED  MGR-NAME       0073-0108
CUSTFILED  MGR-NAME-BLK   0073-0083
CUSTFILED  OPER-SYS       0262-0265
CUSTFILED  PBILL-AMT      0266-0269-P2C2
CUSTFILED  PO-NR          0662-0681
CUSTFILED  PROFILE        0245-0245
CUSTFILED  REP-CODE       0242-0242
CUSTFILED  START-DATE     0246-0251  0
CUSTFILED  START-DAY      0248-0249
CUSTFILED  START-MONTH    0246-0247
CUSTFILED  START-YEAR     0250-0251
CUSTFILED  STATUS         0241-0245
CUSTFILED  STATUS-CODE    0241-0241
CUSTFILED  TECH-ADDR      0337-0372
CUSTFILED  TECH-NAME      0301-0336
CUSTFILED  TEL-AREA       0270-0272
CUSTFILED  TELEPHONE      0270-0286
CUSTFILED  USE-NAME       0244-0244
CUSTFILED  VERS-LVL       0254-0261
CUSTFILED  ZIP-CD         0235-0239
```

'A' Records

FILE TITLE DICTIONARY entry optional - for documentation

Positions

1-8	file name
9-10	blank
11	A
12-25	blank
26-80	file title

SAMPLE

```
1-8       11      26-80
CUSTFILE  A       AMD MASTER FILE
```

See MVS VISION:Forms Dictionary in this chapter.

'S' Records

FILE SPECIFICATION DICTIONARY entry

Positions

1-8 file name

9-10 blank

11 S

12-26 blank

27-30 One of the following could be coded:

- KSDS VSAM
- ESDS VSAM
- RRDS VSAM
- USER USER file such as DAM, TOTAL, etc. A procedure must be used to access USER files.

32-36 Block size. Code a five-digit number indicating the block size of the file. For variable or undefined files, enter the maximum block size that can occur. For KSDS, ESDS, RRDS, and USER media types, this field should be set equal to the logical record length.

38-42 Logical record size. Code a five-digit number indicating the size of logical records on the file. For variable or undefined files, enter a record size that is the largest that can occur.

68-70 VISION:Report area name. This field is normally blank, which indicates that the file is used for INF unless overridden by a FILENAME verb. Any VISION:Report area name is valid except OFA. See the chapter "**Advanced Techniques**" for a discussion of using multiple files with VISION:Forms.

72-79 Default procedure name to be used whenever this file is referenced. If this field is blank, VISION:Forms issues a standard GET for the file unless the file media has been coded as KSDS, ESDS, or RRDS. In the case of VSAM, VISION:Forms issues CALLs to QUIKVSAM to handle the reading. The procedure must be present in the VISION:Forms dictionary. See the chapter "**Special Procedures**" for information on how to write a procedure.

Include one of these for each file. This record is required.

SAMPLE

```
1-8        11      27-30      32-36       38-42

CUSTFILE   S                  03500       00700
PAYROLL    S       KSDS       00400       00400
```

See MVS VISION:Forms Dictionary in this chapter.

'D' Records

Data Element (Field) Definitions

Prepare one data element (data name) definition for each field (logical collection of data elements) which your VISION:Forms users will reference.

For instance, you can have a part number (235 3860) type field in each of your records. We suggest you prepare three names and definitions for it: (1) Part number to reference 235 3860, (2) Part-prefix to reference the first three positions (235), and (3) Part-suffix to reference the last four positions (3860).

Data within a record can be defined, redefined, and referenced with as many different VISION:Forms data element definitions as desired.

Positions

1-8	file name
9	D
10-11	blank
12-25	Data name/field name. Enter a unique name, up to 14 characters in length with no embedded blanks.

The first character must be alphabetic. The balance of the data name can be any alphabetic combination, numeric digits 0 through 9, and hyphens. The data name assigned here is potentially used by non-DP oriented people, so mnemonic significance is particularly important. The data name is ordinarily used as the column header when this field appears in a report. If the field data requires relatively few print positions, a short data name (or one with interspersed hyphens) causes the field to use fewer print positions in the report composition.

27-37 VISION:Report field definition.

27-30 Four-digit number indicating the leftmost record position of the field.

31 Enter a dash (-).

32-35 Four-digit number indicating the rightmost record position of the field.

36-37 If the field is packed decimal, enter -P; otherwise, leave blank.

38 Print decimals. If the field is numeric and can be printed, enter the number of decimal positions to be shown in print. If the field is numeric and has no decimals, enter 0; otherwise, leave blank.

39 If the field is numeric and can be printed, enter one of the following codes:

C	Edit with zero suppression, commas, and decimal point.
E	Edit with European punctuation.
N	Edit without zero suppression and commas. Decimal point is included, if necessary.
Special Edit Character	Enter one of the special VISION:Report expanded edit characters. This character can be any alphabetic character except C, E, or N. Refer to the VISION:Report Reference Manual for details.

40 Calculation decimals. If this field is used as a value in a CALC or SELECT arithmetic statement, enter (0-9), the number of conceptual decimals in the field.

41 Not used at this time.

42-43 Totaled field size. When a field is totaled, VISION:Forms normally increases the original field size by two digits. If this is too large or too small for this field, enter an odd number between 01 and 15. This number must not be any smaller than the original field size. VISION:Forms then totals the field with only this number of digits. If the field is being totaled and averaged, this field is ignored.

44-50 Not used at this time.

51-80 Heading to be used when the field is printed; optional. Start heading in column 51 or beyond; enclose in parentheses. You can enter a heading here which is always used instead of the data name, unless an override heading is specified in the actual request.

Sample

```
1-8       9   12-25      27-30   31     32-35

CUSTFILE  D   CO-NAME    0001    -      0036
```

See MVS VISION:Forms Dictionary in this chapter.

ENVIRON Record

This record makes available to VISION:Forms certain installation standards that are applied to all VISION:Forms unless overridden for a particular VISION:Forms by an inline dictionary entry.

ENVIRON records are optional.

Positions

1-7 ENVIRON

42-43 If VISION:Forms are to be translated from an alternate language into English, code the 2-character acronym shown below for that language:

FRENCH	FR
GERMAN	GR

User Procedures

A generalized system such as VISION:Forms cannot anticipate unusual requirements that might be necessary for specific installations. The USE PROCEDURE verb gives you the ability to insert one time VISION:Report routines to handle these requirements and copy them into VISION:Forms programs as needed. This section covers the coding of a procedure. See the chapter "**Special Procedures**" for examples and a discussion of how to write a procedure.

Positions	
1-8	Enter a one-to eight-position procedure name by which this procedure is referenced in PROCEDURE verbs. The first letter of each procedure name must be alphabetic. The procedure name must not be duplicated elsewhere in the dictionary.
17-24	Enter an ascending sequence number to provide unique identifiers for each entry in a procedure name. This entry is for dictionary sequence control only and is not associated with the VISION:Report sequence numbers discussed below.
25	As seen under 30-80 in this chapter, there are 55 positions available for coding VISION:Report statements. Some VISION:Report verbs can exceed this length and, if this is the case, you should continue that verb in the next procedure name entry and place a non-blank character in position 25 of the first entry.
26-28	VISION:Report sequence numbers. Make no entry in this field unless the statement being coded is a transfer point. The sequence number should be three digits.
30-80	VISION:Report statement. Code the VISION:Report statement exactly the same as you would for VISION:Report. If one entry does not contain the entire statement, code through position eighty. Enter a non-blank character in position 25 of this entry and continue your statement in position 26 of the following entry. Transfer point numbers should be coded as one-to three-digit numbers and must be present as VISION:Report sequence numbers, someplace within this procedure. Each time this procedure is copied into an application, these transfer points (and their associated sequence number fields) are adjusted as appropriate for that application. Should a procedure include logical transferring within the routine, the need for an end-of-procedure exit point is implied. This can be achieved by coding an exit statement as the last statement of the procedure and including a sequence number. You can then exit the procedure and rejoin VISION:Forms generated code by a GOTO nnn statement. nnn is the sequence number of the EXIT statement. A procedure name should not be invoked more than once in a particular VISION:Forms program.

Tables

Data tables that are invoked frequently through VISION:Forms SELECT...ON-TABLE can be cataloged to the dictionary and copied into the generated program through the QWTABLE verb.

Cataloged table information consists of:

- A single entry containing the table specifications as described in Table Specifications Entry in this chapter.
- An optional record containing column header data to be used when the table function is used on a PRINT verb.
- A dictionary entry for each table entry.

Table Specifications Entry

Positions

1-8	Table name that is cited in the QWTABLE DICTIONARY verb. This name cannot duplicate any existing dictionary file name.
9-10	TD
26-29	Maximum number of table entries. This data is reproduced into a VISION:Report TABLSPEC statement as maximum entries. If additional entries are added to the table directly through the VISION:Forms, this number should reflect the maximum entries from both sources.
31-32	Argument position. Code a two-digit number specifying in which column the table argument starts. As seen under Table data entries: in this chapter, the actual table data is keyed starting in position 26, but this should be considered as the equivalent of column 1. In short, the usual entry here is 01.
34-35	Argument length. Enter a two-digit number indicating the number of characters in each table entry.
37	Argument type. Enter the letter 'P' to denote that the argument is a packed format. This is required only when the data name used in a SELECT ... ON-TABLE is packed. If left blank, the data and field are both considered to be in standard EBCDIC format.
39-40	Function starting position. Code a two-digit number specifying in which column the table function starts.

42-43	Function length. Code a two-digit number to indicate the length of the function in the table entry.
45-58	Function data name. Enter a data name of up to 14 characters that is EQUATEd to the table function after selection has occurred. This operand is optional, but we suggest you code it unless your table has multiple functions that must be accessed separately.

Table Function Header Entries

Positions	
1-8	The same table name as keyed into the table specification entry.
9-11	TDH
26-55	Code up to 30 characters enclosed in parentheses. This header is used as the default column header whenever the table function is to be printed.

Table Data Entries

Positions	
1-8	The same table name as coded into the table specification entry.
9-10	TE
23-25	Code an ascending sequence number, e.g., first TE 010, second TE 020, etc. This data is required only to satisfy the key sequencing requirements of the dictionary.
26-80	Enter up to 55 positions of table data. Entries should be keyed exactly as though they were going directly into VISION:Report. VISION:Forms creates one statement for each TE entry with dictionary positions 26-80 being created into columns 1-55.

JCL Examples

A 'Run mode' using the Payroll File:

```
//SAMPLE1 JOB XYZ,AMD
//STEP10 EXEC PROC=QWRUN,FILE='PAYROLL'
//RUN.QWSYSIN DD *
    VISION:Forms statements
```

A 'Punch mode' using the Payroll File as basis for selection:

```
//SAMPLE2 JOB XYZ,AMD
//STEP10 EXEC PROC=QWPUNCH
//PUNCH.QWSYSIN DD *
    VISION:Forms statements
```

A 'Run mode' for the Payroll File using the inline dictionary:

```
//SAMPLE3 JOB XYZ,AMD
//STEP10 EXEC PROC=QWRUN,FILE='PAYROLL'
//RUN.QWSYSIN DD *
START-DICTIONARY
 dictionary entries                    (overrides equal ISAM
                                       dictionary entries, if present)
END-DICTIONARY
    VISION:Forms statements
```

A 'Run mode' using the Payroll File with the inline dictionary in PROCLIB. (Note that the dictionary in PROCLIB must have START-DICTIONARY and END-DICTIONARY statements.)

```
//SAMPLE4 JOB XYZ,AMD
//STEP10 EXEC PROC=QWRUN,FILE='PAYROLL'
//RUN.QWSYSIN DD DSN=SYS1.PROCLIB(USERDICT),DISP=SHR
//       DD *
    VISION:Forms statements
```

Another way of doing the above job would be to put the //QWSYSIN DD statement pointing to PROCLIB in the QWRUN procedure:

```
//QWRUN PROC
  normal JCL
//QWSYSIN DD DSN=SYS1.PROCLIB(USERDICT),DISP=SHR
//        DD DDNAME=QWINPUT
remaining JCL for procedures
```

Your JCL would then be:

```
//SAMPLE4 JOB XYZ,AMD
//STEP10 EXEC PROC=QWRUN,FILE='PAYROLL'
//RUN.QWINPUT DD *
    VISION:Forms statements
```

A 'Run mode' using the Payroll File with an inline dictionary in PROCLIB, and in the input stream.

```
//SAMPLE5 JOB XYZ,AMD
//STEP10 EXEC PROC=QWRUN,FILE='PAYROLL'
//RUN.QWSYSIN DD DSN=SYS1.PROCLIB(USERDICT),DISP=SHR
//            DD *
START-DICTIONARY
dictionary entries                (There must not be any equal entries to
                                   entries in PROCLIB dictionary)
END-DICTIONARY
   VISION:Forms statements
```

A 'Run mode' using the Payroll File. In this example, a modified QWRUN procedure was created to handle requests for runs against the Payroll File. Its name is QWPAYROL. The statement changed in the procedure was:

```
//QWINF DD DSN=PAYROLL,DISP=OLD
```

The JCL to execute the procedure QWPAYROL would then be:

```
//SAMPLE7 JOB XYZ,AMD
//STEP10 EXEC PROC=QWPAYROL
//RUN.QWSYSIN DD *
  VISION:Forms statements
```

A 'Run mode' using the Payroll File with the inline dictionary residing in a CA-PANVALET source library.

```
//SAMPLE8 JOB XYZ,AMD
//STEP10 EXEC PGM=PAN#l
  Normal PANVALET DD statements
//PANDD2 DD DSN=&UDICT,DISP=(,PASS),
//       UNIT=DISK,SPACE=(CYL,(1,1))
//SYSIN  DD *
++WRITE WORK,USERDICT
//STEP20 EXEC PROC=QWRUN,FILE='PAYROLL'
//RUN.QWSYSIN DD DSN=&UDICT,DISP=(OLD,DELETE)
//            DD *
  VISION:Forms statements
```

The following error messages can appear on the JCL listing in the event that certain errors occur.

***QKWREXEC - SORT statement error - Notify Computer Associates.

***QKWREXEC - PREMATURE EOF ON INPUT - Notify Computer Associates.

ABEND Codes

3343	An error has occurred that prevents VISION:Forms from running. Refer to either the VISION:Forms log or the JCL listing.
3330-3339	Standard VISION:Report abends. Refer to the VISION:Report Reference Manual.

Chapter

6

Special Procedures

VISION:Forms - DL/I Access (VSE only)

To access DL/I files through VISION:Forms, you must take into consideration two record types in the dictionary.

- 'S' Record - the following fields must be checked and changed as required:

27-30	Media must be USER.
32-36	Must be equal to the largest area (segment) needed.
38-42	Same value as positions 32-36.
72-79	The name of a VISION:Forms procedure (that resides in the dictionary) that handles this database. This procedure handles the CALLs to QUIKDLI.

- EXE - These statements are coded similarly to the procedure statements and are used in place of the standard generated VISION:Forms.

```
// EXEC QUKBJOB
```

See EXE RECORDS in this chapter on coding EXE records.

Any segments CALLed must be placed in the INF area. In the case of multiple segments:

Assume your report request requires 3 segments to be returned before SELECTing. All 3 segments must be read and 'stacked' into the INF area within the same procedure. The VISION:Forms dictionary definitions must allow for this stacking at the time it is built.

Credit Union Data Base

Name Segment		Balance Segment		History Segment	
	Personal Account data		Current Loan balance		Loan Payment History
Key	Data - Name, Social Security Number	Key	Data - Original Loan amount	Key	Data - Date, Pmt amt., etc
↑		↑		↑	
INF1		INF30		INF50	

EXE Records

Execute dictionary entry

Positions

1-8	file name
9-11	EXE
12-21	blank
22-24	Enter a sequence number that keeps the EXE entries in their proper order.
25	blank
26-80	Data base execution statements coded exactly as required. These statements replace the normal // EXEC QUKBJOB that VISION:Forms generates.

Sample

```
CUSTFILE EXE          010 // EXEC DLZRRC00,SIZE=300K
CUSTFILE EXE          020 DLI,QUIKDLI,psbname
```

DL/I Example:

Records are to be retrieved from the root segments only, selecting only those in plant numbers 03 and 05; sort in plant-number sequence and print the total salary for each pay grade.

All of the changes necessary in the dictionary have been made as inline entries so you can easily note their presence.

```
START-DICTIONARY
PAYROLL   S             USER 02000 02000
010    PAYCALL
PAYROLL JCLALSEL    010 // ASSGN SYS010,IGN
PAYROLL JCLALSEL    020 *  DATABASE ASSIGNMENTS
PAYROLL JCLALSEL    030 *  DLBL/EXTENTS NORMAL TO YOUR DL/I
PAYROLL EXE         010 // EXEC DLZRRC00,SIZE=200K
PAYROLL EXE         020 DLI,QUIKDLI,PSBNAME
PAYCALL             010 010 CALL QJBTDLI C'GN  ' C'01' INFl
C'ROOTSEG'
PAYCALL             020    IF PCB11-12 IS EQ C'GB'      /*
EOF
PAYCALL             030    GOTO EOJ.
PAYCALL             040    IF PCB11-12 IS EQ C' '
PAYCALL             050    GOTO 020.
PAYCALL             060    PRINTHEX PCB1-50
PAYCALL             070    ABEND
PAYCALL             080 020 EXIT
END-DICTIONARY
FILENAME PAYROLL
TITLE 'Sample VISION:FORMS/DLI REPORT '
SELECT PLANT-NR EQ '03' '05'
CALC SALARY DEC2C = HOURLY-RATE TIMES HOURS-WORKED
SORT PLANT-NR
PRINT PLANT-NR DEPT-NR NAME SALARY
TOTAL SALARY
BREAK PLANT-NR
```

VISION:Forms - IMS/DB Access (MVS only)

To access IMS/DB files through VISION:Forms you must take into consideration the following information:

1 You must have the optional program, QUIKIMS installed.

2 The following fields must be checked, and changed if required, on the 'S' record in the dictionary:

27-30	Media must be USER.
32-36	Must be equal to the largest area (segment) needed.
38-42	Same value as positions 32-36.

72-79 The name of a VISION:Forms procedure (that either resides on the dictionary or is coded inline) that handles this database. You can use either the USE PROCEDURE verb or code the procedure name on the FILENAME verb.

A procedure is required to access an IMS/DB database. It contains the actual VISION:Report code that issues the CALLs to QJBTDLI. See **User Procedures** in the chapter "VSE System Information" and **User Procedures** in the chapter "MVS System Information" for information about coding procedures.

3 The following changes must be made to your VISION:Forms JCL:

- The EXEC statement in the QWRUN procedure must be changed to execute IMS/DC (as you currently do) and specify QUIKIMS as the application program to be executed.
- The DD statements necessary to execute IMS/DC and access the IMS/DB database must also be included in the procedure.
- If the IMS/DC segments are to be read into an input area, such as INF, DD dummy statements must be present within the procedure specifying a BLKSIZE and LRECL equal to the largest segment needed.
- The DDNAME for INF is QWINF. All other input areas are the same as with VISION:Report.
- Reading the segments into WST and specifying WST in columns 68-70 of the 'S' record relieves you of the DD dummy statement requirement.

Example:

```
//STEP1     EXEC PGM=DFSRRC00,PARM='DLI,QUIKIMS,psbname'
//IMS       DD DSN=...
//QWINF     DD DUMMY,DCB=(BLKSIZE=1000,RECFM=F,LRECL=1000)
//IMSFILDD  DD DSN=...
```

For additional information, refer to the VISION:Report Reference Manual for more information about QUIKIMS.

IMS Example:

In the following example, we retrieve root segments only, selecting only those parts produced in PLANT 02, 04, and 05, SORT by PLANT-NUMBER and PART-NUMBER sequence, and CALCulate and PRINT the dollars in the inventory for each part, TOTALed by plant.

```
START-DICTIONARY
INVFILE  A*              INVENTORY FILE
INVFILE  S*              USER 00050 00050                        INVCALL
INVFILE D  PLANT-NUMBER  0001-0002-P0C0         (PLANT,NUMBER)
INVFILE D  PART-NUMBER   0003-0006-P0  0        (PART,NUMBER)
INVFILE D  PART-NAME     0007-0036              (PART,NAME)
INVFILE D  INV-TOTAL     0037-0040-P0C0         (INVENTORY,TOTAL)
INVFILE D  COST-PER-PART 0041-0044-P2C2         (COST,PER,PART)
INVFILE D  PARTS-SHIPPED 0045-0050-P0C0         (PARTS,SHIPPED)
INVCALL        010 010 CALL QJBTDLI C'GN ' C'01' INF1 C'ROOTSEGM '
INVCALL        020     IF PCB11-12 EQ C'GB'     /* EOF ?
INVCALL        030        GO TO EOJ.
INVCALL        040     IF PCB11-12 EQ C'  '     /* OK ?
INVCALL        050        GOTO 020.
INVCALL        060     MOVE C'**BAD RETURN CODE**' TO PRT1
INVCALL        070     PRINT
INVCALL        080     PRINTHEX PCB1-20
INVCALL        090     ABEND
INVCALL        100 020 EXIT
END-DICTIONARY
FILENAME INVFILE
TITLE 'Sample VISION:FORMS/IMS REPORT'
SELECT PLANT-NUMBER EQ '02' '04' '05'
CALC INV-DOLLARS = INV-TOTAL * COST-PER-PART
SORT PART-NUMBER PLANT-NUMBER
PRINT PLANT-NUMBER PART-NUMBER PART-NAME INV-DOLLARS
TOTAL INV-DOLLARS
BREAK PLANT-NUMBER
```

Coding Procedure Code Blocks

Procedures allow you to code special purpose routines that perform functions that VISION:Forms cannot handle. These procedures, once coded, can be cataloged into the VISION:Forms dictionary and used whenever needed by merely coding a USE PROCEDURE verb (see **USE PROCEDURE** in the chapter "Writing Verbs").

Rules:

1 All procedures must be coded using VISION:Report.

2 The VISION:Report statements are not checked for valid syntax. Therefore, you should ensure the procedure is coded properly and well tested before cataloging it into the VISION:Forms dictionary. A good way to test a procedure is to use an inline dictionary.

3 Any data name found within the procedure causes VISION:Forms to generate a VISION:Report EQU statement. The data name should be in one of the files or areas specified by the VISION:Forms FILENAME verb. No VISION:Forms error occurs if the data name is not found.

4 VISION:Report EQU statements can be coded within the procedure.

5 Do not code sequence numbers unless they are a transfer point in a GOTO, GET, PERFORM, or ATEND statement.

6 Assign sequence numbers as though the procedure were a complete VISION:Report. Your earliest transfer point might be 010, the next transfer point 020, etc. VISION:Forms automatically adjusts these sequence numbers whenever the procedure is pulled, plus it adjusts the sequence numbers within GOTO, GET, PERFORM, and ATEND statements.

7 If the procedure logic requires that you transfer out of the procedure (as opposed to 'fall-through' to the next sequential statement), code a VISION:Report EXIT statement with a sequence number as the last statement of your procedure. You can then GOTO this EXIT sequence number to reenter VISION:Forms generated code.

8 When QUIKSORT and a procedure are used during the SELECT phase, the procedure is checked for any occurrences of the keyword EOJ. The EOJ is changed to point to the first sequence number within the REPORT phase. If the procedure does not want VISION:Report to continue processing, a VISION:Report ABEND statement should be coded instead of EOJ.

9 If the procedure being used is to REPLACE the WRITE, remember to include a RELEASE OFA1 TO SORT if the QUIKSORT feature is being used, or WRITE OFA if the QUIKSORT feature is not used.

10 If the procedure being used is to REPLACE the GET in the report phase, remember to include a RETURN SORTED INTO OFA1 if the QUIKSORT feature is being used.

11 The VISION:Report WST area is available for use by the procedure. VISION:Forms does not make references to this area.

12 The VISION:Report SAVE area must not be used by the procedure. VISION:Forms makes extensive use of this area.

13 The VISION:Report OFA area should not be used by the procedure. This area is used by VISION:Forms to hold the input record plus any fields that are CALCed, ATTACHed, etc.

Examples:

In the following example, a procedure has been written that reads a DAM file by calling a subroutine called PAYCALL.

Assume the following procedure has been cataloged into your VISION:Forms dictionary:

```
PAYCALL  010 010 CALL PAYREAD C'READ' WST101
PAYCALL  020     IF EMP-NUM = HIVALUES
PAYCALL  030       GOTO EOJ.
PAYCALL  040     IF DEPT = LOVALUES
PAYCALL  050       GOTO 010.
```

Also assume the following entries are in the dictionary that describe the PAYROLL record:

```
PAYROLL S            USER       00300        010
PAYROLL D   DEPT     0001-0004
PAYROLL D   EMP-NUM  0005-0010
PAYROLL D   NAME     0011-0040
PAYROLL D   HR-RATE  0041-0047-P C4
```

VSE users might have the following entries in their dictionary:

```
PAYROLL  JCLALSEL    010 // ASSGN SYS010,DISK,VOL=DYL001,SHR
PAYROLL  JCLALSEL    020 // DLBL PAYROLL,'DYL.PAYROLL',,DA
PAYROLL  JCLALSEL    030 // EXTENT SYS010,DYL001,,,120,60
```

A VISION:Forms program can now be written to access the PAYROLL file:

```
FILENAME PAYROLL
USE PROCEDURE PAYCALL TO REPLACE GET
SELECT DEPT '100'
PRINT DEPT EMP-NUM NAME HR-RATE
```

If you wanted the procedure PAYCALL to always be used whenever the PAYROLL file was referenced, the procedure name PAYCALL could be coded in position 72-79 of the 'S' record for the PAYROLL file. If a procedure name is coded in position 72-79 of the 'S' record, the USE PROCEDURE VISION:Forms statement is not necessary.

In the following example a procedure has been written that reads a KSDS VSAM file sequentially based upon a start key found in a sequential tape file. The VSAM file is read until the end of the group is found, at which time a new start key is read from the sequential file and the procedure is repeated.

Assume the following procedure has been cataloged into your VISION:Forms dictionary:

```
VSAMREAD  010 010 IF WST21 = C'1'         /* FIRST TIME TEST
VSAMREAD  020       GOTO 030.
VSAMREAD  030     MOVE C'1' to WST21      /* SET FLAG
VSAMREAD  040     * FOLLOWING SETS UP VSAM FEEDBACK AREA
VSAMREAD  050     CALL QUIKVSAM C'PARTMAS ' C'OPTION' WST1
VSAMREAD  060 020 GET INF ATEND EOJ       /* READ START KEY FILE
VSAMREAD  065     * FOLLOWING POS THE VSAM FILE ON GENERIC KEY
VSAHREAD  070     CALL QUIKVSAM C'PARTMAS ' C'POINT' INF1 C'KGE03'
VSANREAD  080     IF WST8-9 = X'0810'     /* RECORD NOT FOUND ?
VSAMREAD  090       GOTO 020.
VSAMREAD  100     IF WST8-9 NOT = X'0000' /* ANY OTHER ERRORS ?
VSAMREAD  110 025    PRINTHEX WST1-13     /* DISPLAY FEEDBACK
VSAMREAD  120        ABEND.
VSAMREAD  125     * FOLLOWING READS THE VSAM FILE
VSAMREAD  130 030 CALL QUIKVSAM C'PARTMAS ' C'GET' INF1
VSAMREAD  140     IF WST8-9 = X'0804'     /* EOF ?
VSAMREAD  150       GOTO 020.
VSAMREAD  160     IF WST8-9 NOT = X'0000' /* ANY OTHER ERRORS ?
VSAMREAD  170       GOTO 025.
VSAMREAD  180     IF PART-ID NOT = GROUP  /* SAME START KEY ?
VSAMREAD  190       GOTO 020.
```

Also, assume the following entries are in the dictionary that describes the KSDS VSAM file and the sequential tape file:

```
PARTMAS     S           KSDS 00200 00200                000
PARTMAS D    PART-NO    0001-0010
PARTMAS D    PART-ID    0001-0003
PARTMAS D    PART-DESC  0011-0040
PARTMAS D    PART-COST  0041-0045-P2C2


PARTREQ     S           TAPE 00800 00080                011
PARTREQ D    GROUP      0001-0003
```

VSE users might have the following entries in their dictionary:

```
PARTMAS   JCLALSEL     010 // ASSGN SYS010,DISK,VOL=PARMAS,SHR
PARTMAS   JCLALSEL     020 // DLBL PARTMAS,'PART.MASTER',,VSAM
PARTMAS   JCLALSEL     030 // EXTENT SYS010,PARMAS

PARTREQ   JCLALSEL     010 // ASSGN SYS011,080
PARTREQ   JCLALSEL     020 // TLBL PARTREQ
```

A VISION:Forms program can now be written to access the VSAM KSDS file using only those records that are in the groups found in the PARTREQ file.

```
FILENAME PARTREQ USING VSAMREAD PARTMAS AS DET
PRINT PART-NO PART-DESC PART-COST
```

Note that the USING on the FILENAME verb causes the same results as coding:

```
USE PROCEDURE VSAMREAD TO REPLACE GET
```

The following example contains two procedures that allow an inline file to be read into a VISION:Report table. The table can then be used to sequentially process a VSAM KSDS file using keys from the inline file, as you will see shortly. The file being used here is the same as that used in the second example.

Procedure VSAMINIT is used to sort the table into part number prefix sequence, set TSA to the start of the table, and establish the VSAM feedback area.

```
VSAMINIT   010     SET TSA INITIAL
VSAMINIT   020     IF TSA1-3 IS HIVALUE     /* TEST IF ANY DATA
VSAMINIT   030       MOVE C'NO REQUEST PART NUMBERS FOUND' TO PRT1
VSAMINIT   040       PRINT
VSAMINIT   050       ABEND.
VSAMINIT   060     CALL TABLSORT TSA1 C'03' C'01' C'03' C'A'
VSAMINIT   070     CALL QUIKVSAM C'PARTMAS ' C'OPTION' WST1
```

Procedure VSAMREAD is used to do the actual positioning and reading of the VSAM KSDS file.

```
VSAMREAD   010     IF WST21 = C'1'        /* FIRST TIME CHECK
VSAMREAD   020       GOTO 030.
VSAMREAD   030     MOVE C'1' TO WST21     /* SET FLAG
VSAMREAD   040 010 CALL QUIKVSAM C'PARTMAS ' C'POINT' TSA1 C'KGE03'
VSAMREAD   050      IF WST8-9 = X'0804'   /* EOF ?
VSAMREAD   060        GOTO 040.           /* ANY ERROR CHECK
VSAMREAD   070      IF WST8-9 NOT = X'0000'
VSAMREAD   080 020   PRINTHEX WST1-13     /* PRINT VSAM FEEDBACK
VSAMREAD   090       PRINTHEX TSA1-10     /* PRINT TABLE KEY
VSAMREAD   100       ABEND.               /* KILL JOB
VSAMREAD   110 030 CALL QUIKVSAM C'PARTMAS ' C'GET' WST101
VSAMREAD   120      IF WST8-9 = X'0804'   /* EOF CHK
VSAMREAD   130        GOTO 040.           /* ANY ERROR CHECK
VSAMREAD   140      IF WST8-9 NOT = X'0000'
VSAMREAD   150        GOTO 020.
VSAMREAD   160      IF PART-ID = TSAl-3   /* SAME PART PREFIX ?
VSAMREAD   170        GOTO 100.
VSAHREAD   180 040 SET TSA UP 3           /* POINT TO NEXT ENTRY
VSAMREAD   190      IF TSA1-3 IS HIVALUE  /* END OF TABLE ?
VSAMREAD   200        CALL QUIKVSAM C'CLOSE'
VSAMREAD   210        GOTO EOJ.
VSAMREAD   220      GOTO 010.
VSAMREAD   230 100 EXIT
```

Here is where the keys are used to process the VSAM KSDS file. The code shown below accesses the two routines, passing along 'keys' (lines 7 through 9) that are read into the table - this is the inline data. Using the same data names and JCL (VSE only) for PARTMAS from the second example, the following VISION:Forms program searches the VSAM KSDS file, testing for the keys (in this case, WWS, ANC, and XKD) read into the table.

```
FILENAME PARTMAS
USE PROCEDURE VSAMINIT AT START-UP
USE PROCEDURE VSAMREAD TO REPLACE GET
PRINT PART-NO PART-DESC PART-COST
QWTABLE 0050 01 03
TABLE DATA
WWS
ANC
XKD
/*
```

Note that the QWTABLE and TABLE data statement are required in order for VISION:Forms to generate a VISION:Report TABLSPEC statement, and then load the statements that follow it into the table. The statements following the QUIKWRITE" TABLE data" statement are the actual user request statements that are used by the procedure VSAMREAD to position the VSAM file.

Chapter

7

Advanced Techniques

Accessing Multiple Files

This chapter of the VISION:Forms Reference Manual is for experienced users. If you are new to VISION:Forms, please become familiar with the operation of the system before moving on to these advanced techniques.

You are now able to access multiple files within a VISION:Forms program. There are several ways to do this, but in each case you must code a procedure in the VISION:Forms dictionary to do the actual matching of the files.

Any data name from an area other than INF is automatically ATTACHed to the original INF record, but only if the data name is referenced anywhere in the VISION:Forms (see **ATTACH** in the chapter "Writing Verbs"). For example, if you have a field called AMOUNT in your secondary (DET) file and you code PRINT AMOUNT, the field is ATTACHed to your primary file and contains your data at print time.

When you are matching two or more files that contain identical data names in the dictionary, special caution is needed. Before you refer to one of these data names, you must do one of the following: (1) change the dictionary so that there are no duplicate data names between the files or (2) use a new form of the EQUATE verb.

For example, suppose you have identical data names called AMOUNT in your dictionary. One AMOUNT field is in the INVOICE file and one is in the CUSTOMER file. This form of the EQUATE verb allows you to rename one of the fields.

```
EQUATE INV-AMOUNT TO AMOUNT OF INVOICE
```

This statement renames the AMOUNT field of the INVOICE file to INV-AMOUNT. All further references in your VISION:Forms program can now use the data name INV-AMOUNT.

Method 1

With this method, you do not have to change your VISION:Forms dictionary (unless you have duplicate data names). All references to additional files (or areas) are made in the VISION:Forms statements.

You can use a variation of the FILENAME verb code more than one file and to specify its associated VISION:Report area.

Coding Rules

Code the VISION:Forms verb FILENAME followed by Words 1 through 5 as follows:

```
                OPT          OPT             OPT        OPT
     Word       Word         Word            Word       Word
       1          2            3               4          5

  Filename      USING     Procedure Name      As       QJ area
                             Name
```

You can also code additional file names or Filename AS QJ area combinations. A maximum of 25 words are allowed as arguments to the Filename verb. However, you can code only one USING clause, and its procedure must always refer to the primary (INF) file. Also, the first file name specified must always refer to the primary (INF) file.

Example:

```
FILENAME PERSONEL
         PAYROLL AS DET
         HISTORY AS OFB
```

This statement causes all references to the PERSONEL file to be generated as INF. Also, if you are using VSE, the JCL and the file definition statements are generated as INF.

All references to the PAYROLL file cause EQUATEs to be generated as DET. Again, if you are using VSE, the JCL and file definition statements are generated as DET.

Additionally, all references to the HISTORY file cause EQUATEs to be generated as OFB. Again, if you are using VSE, the JCL and file definition statements are generated as OFB.

Example:

```
FILENAME PERSONEL USING PERSREAD
         PAYROLL AS DET
         VALUE AS VAL
```

This statement causes all references to the PERSONEL file to be generated as INF. For VSE users, the JCL and the file definition statements are also generated as INF. VISION:Forms uses the procedure called PERSREAD to access the file. No GET statement is generated.

All references to the PAYROLL file cause EQUATEs to be generated as DET. Again, if you are using VSE, the JCL and file definition statements are generated as DET.

Finally, a third area is accessible. The file name VALUE in the VISION:Forms dictionary is pulled-in as needed and those data names refer to the VISION:Report VAL area. Users can access the current date, Julian date, etc., within their VISION:Forms programs.

Method 2

This method requires some changes to the dictionary, but they're very easy to make.

For instance, in the dictionary 'S' record for the primary file (INF), code the name of the procedure in positions 72-79. This procedure replaces the normal VISION:Forms generated GET for the file. You can override this procedure only by coding a FILENAME xxx USING yyy in your VISION:Forms program. When you have multiple files, use this same procedure because it contains the necessary logic for reading and matching your files.

Also, when using this method, you do not have to code any procedure statement in your VISION:Forms program. Nor do you have to code a USING procedure name clause on your FILENAME verb.

To make coding even easier, the VISION:Report area name can be coded in the dictionary 'S' record in positions 68 through 70.

Example:

You can have a Name and Address file that is always used for matching purposes. In this case, it is always used as a secondary file (DET), and never read as the primary file. The Name and Address file can be permanently set up as a DET file. After doing this, you do not need to code the AS QJ area clause on the FILENAME verb.

To summarize, if the procedure name and the VISION:Report area are included in the dictionary, the only statement you code to match an Invoice file (INVOICE) to a Name and Address file (ADDRESS) is:

```
FILENAME INVOICE ADDRESS
```

Remember, the ADDRESS file is preset to be a DET file and the procedure to do the actual match has already been coded in the dictionary.

Accessing Other VISION:Report/VISION:Forms Areas

Sometimes, you find that you need access to other areas available to VISION:Report. We have already covered the handling of other files, but there can be a case when you only need to select based on the current date. Most of the VISION:Report areas can be accessed by VISION:Forms including:

VAL	TBH	TSA	FUN	PUN
PRT	PTA	PTB	PTC	PTD

The method that requires the least coding of VISION:Forms users is a variation of the FILENAME verb. Data processing should set up a file name in the dictionary that refers to the 'area'. Here, we use the VISION:Report VAL area as an example. Suppose the file name is VALUE and all needed field definitions are set up like any other file.

```
VALUE      S                                    VAL
VALUE    D DATE          0005-0012      (TODAY'S DATE)
VALUE    D MONTH         0013-0021
VALUE    D REMAINDER     0030-0037-P    (DIVISION REMAINDER)
VALUE    D J-NAME        0038-0045      (JOB NAME)
VALUE    D TIME          0062-0066   B  (CURRENT TIME)
VALUE    D JULIAN        0071-0075      (JULIAN DATE)
```

To access these fields in a VISION:Forms program, you need only code the following:

```
FILENAME INVOICE VALUE
SELECT ACCT-DATE = DATE
PRINT ACCT-NR DEPT DATE AMOUNT MONTH
```

Note that the printed fields, ACCT-NR, DEPT, and AMOUNT are from the INVOICE file. DATE and MONTH are coming from the VISION:Report VAL area.

All of these features, as well as the USE PROCEDURE verb, increase VISION:Forms' flexibility. However, in giving you these capabilities, you take on added responsibilities. Computer Associates cannot guarantee your results when the problem centers on your procedure logic.

Chapter

8

Dictionary Maintenance

At the heart of VISION:Forms is what is commonly referred to as a **Data Dictionary**. The dictionary is used to store information that describes your data residing in computer files.

The dictionary itself is a computer file that must be created and maintained so as to reflect the most current data items that are available to users of VISION:Forms.

QWDMAINT can be used to load, service, backup, and restore the physical dictionary file (QW$DICT). At the present time, this program only supports a VISION:Forms dictionary that is defined as a VSAM Keyed Sequence Data Set (KSDS). An Indexed Sequential Access Method (ISAM) dictionary cannot be maintained by this program.

This system uses QUIKVSAM, which is distributed as optional material on the VISION:Report installation tape. You must have QUIKVSAM installed in your Phase or Load library prior to the execution of QWDMAINT.

The VSAM approach was chosen due to the excessive number of additions that are needed to create and modify dictionary entries. With ISAM, the overhead of cylinder overflow would be incurred. Also, using VSE ISAM, deleted records cannot be physically removed from the dictionary without a file reorganization after every service run. These factors, plus VSAM's dynamic and secondary allocations and portability, are reflected in the decision to support only VSAM.

General Information

QWDMAINT is a special purpose program that adds, changes, deletes, backs up, or restores members in the VISION:Forms data dictionary. There are three types of members that reside in the dictionary: files, tables, and procedures.

Of these three, files are the most common, because they describe the contents and location of data stored in other computer files (e.g., Accounts Receivable, Payroll, etc.). However, the computer needs to know more than just where data is stored. There are physical attributes that further describe where a file is located, what type of file it is, how big it is, and other control information that can be necessary to access the required data.

The VISION:Forms dictionary can be used to save this information and, when needed, recall it automatically. Usually, your installation personnel is responsible for providing the proper operating and control characteristics that should be recorded, such as job control language, device type, and logical unit numbers.

Tables represent a group of related items that could be used to select or restrict a large file into a small subset that would represent only those items that belong in a particular category. Another function of a table is to translate one value into another. For example, month 01 could be converted to January, 02 would become February, etc.

Procedures are used to supplement the generated VISION:Report code that requires special processing that VISION:Forms is unable to produce. For example, processing a DL/I data base file requires a procedure to replace the standard VISION:Forms read-a-file logic.

Command Notation

QWD$INT functions are defined using utility control statements. The control statements, read from SYSIPT, fit into two basic categories:

Group 1	Those that provide information about the desired functions that are to be performed.
Group 2	Those that provide control information about a dictionary member.

Multiple Group 2 statements could be required or supplied for a given member type. These statements must follow the :ACCESS/:CREATE/:UPDATE command and precede any other colon (:) command; they must be of the same type as the dictionary member being processed.

A Group 2 statement can be specified with no value by coding the keyword followed by an equal sign (=), followed by at least one space. This can be used to provide a null value to a parameter that previously contained a value.

Positional Operands

In this format, the parameter value must be in the exact order shown in the individual command discussion. The operand must have at least one space between the command and the specified value. A comment can be coded on the same statement by leaving at least one space between the specified value and the start of the comment.

For example, :ACCESS uses the positional format. The :ACCESS for a member named CUSTFILE with a comment of GET THE MEMBER would be written as:

```
:ACCESS  GET THE MEMBER
```

Keyword Operands

An operand written in keyword format has this form:

```
DEVICE=SD5C
```

Where DEVICE is the keyword, SD5C is the specification, and DEVICE=SD5C is the complete operand. The keyword operands in a command can appear in any order, and any keyword operands that are not required can be omitted. Different keyword operands can be written in the same statement, each followed by a comma except for the last operand of the command. However, Group 2 commands are written with only one complete operand per statement.

Continuation is specified by coding a comma and at least one space after a complete operand. Code the next complete operand starting in position 2 or greater of the next statement.

Remember that the first position of continuation lines must be blank. Comments can also be specified on the same statement that contains a complete operand by leaving at least one space between the operand and the comment.

A comment statement is allowed for documentation purposes, and is specified by coding an asterisk (*) in position 1 of the command record. The entire record is treated as a comment and prints on the hard copy listing.

Notational Conventions

The following conventions are used in this document to illustrate the format of the commands:

- Uppercase letters and punctuation marks (except as described in these conventions) represent what must be coded exactly as shown.
- Lowercase letters and terms represent information that you must supply. An n denotes a decimal number, and an x indicates an alphanumeric character.
- Information contained within brackets [ ] is an optional parameter that can be included or omitted, depending on job criteria.
- Stacked options represent alternatives, one of which can be chosen. Underscored elements represent the default value to be used if omitted.
- Alternatives can also be shown between brackets on one line that is unstacked. The options are separated by OR symbols (|), one of which can be chosen.

- A series of three periods ... indicates that a variable number of like characters can be included.
- Information contained within parentheses ( ) represents a character string in which any characters, including spaces, can be coded. The string must not contain embedded parentheses. Optionally, the value is enclosed by single quotation (') marks. Using this form allows parentheses to be part of the string. In either case, for the string to contain a quotation mark as part of the value, two quotation marks must be coded together, as in the following examples:

Valid:

```
TITLE=(CUSTOMER MASTER FILE)
TITLE='CUSTOMER(S) MASTER FILE'
TITLE='CUSTOMER''S MASTER FILE'
```

Invalid:

```
TITLE=(CUSTOMER(S) MASTER FILE)        embedded parentheses
TITLE=(CUSTOMER'S MASTER FILE)         need two quotation marks
```

- The following lowercase words represent an example of the type of value you must supply.

field	Any valid field name (maximum of 14 characters). No embedded spaces are allowed.
from	A beginning sequence number (up to 4 digits) that indicates the starting point of the operation.
member	Any valid member name (maximum of 8 characters). Can also be the reserved word QW$DICT for :ACCESS/:CREATE/:UPDATE to indicate that the dictionary standards are to be processed (the members ENVIRON, INSTLSTD, STDSORT).
partid	Any valid VSE partition ID (BG,FB,FA,F1 through F9) or ALL can be specified.
to	An ending sequence number (up to 4 digits) that indicates the stopping point of the operation.

Group 1 Commands

Note: In this table, the :BACKUP, :LIST, :PUNCH, :PURGE, and :RESTORE control statements are not allowed when used in an inline dictionary.

	Control Statement	Description
	:ACCESS member	Retrieve member with no updating
	:BACKUP	Offload members to tape
or	:BEGIN RECORD TABLE PROC :BEGIN MEDIA=partid	Start data entry mode
or	:BEGIN JCL=partid,USAGE=SEL SOI SOW SOO	
	:CHANGE SEQ=from,to	Alter data during modify mode
	:CREATE member	Add a new member to dictionary
or	:DELETE SEQ=from,to :DELETE ENTRY=field	Remove data or a field during modify mode
	:INSERT SEQ=from	Add data during modify mode
	RECORD FIELDS MEMBERS	
	:LIST MEDIA JCL	Print a report describing member
or	:MODIFY RECORD TABLE PROC :MODIFY MEDIA=partid	Start modify mode
or	:MODIFY JCL=partid,USAGE=SEL SOI SOW SOO	
	:OPTION TSIZE=nnnn	Describe run-time characteristics
	JCL MEDIA :PUNCH RECORD	Convert member to QWDMAINT commands
	:PURGE member	Remove member from the dictionary

	Control Statement	Description
or	:RESTORE ALL :RESTORE member	Reload dictionary or restore a single member
	:UPDATE member	Retrieve a member for updating

Group 1 Command Descriptions

:ACCESS member

This command scans the dictionary looking for 'member'; when found, the dictionary records associated with the member are made available for processing. A flag is set to indicate that any changes requested do not affect the dictionary itself. You can use this command to simulate changes if you wish. When you are satisfied, change :ACCESS to :UPDATE and run the job again. If the member is not found, a self-explanatory message is issued and the job stream is flushed to the next system command.

If the member is specified as QW$DICT, this causes the dictionary standards to be retrieved for processing. The standards are made up of the ENVIRON, INSTLSTD, and STDSORT members.

If any changes are requested, an implied :LIST is invoked to show the effects of changes requested. This useful document should be stored as control of modifications to the dictionary.

:BACKUP

This command, when executed under VSE, creates a tape on the programmer logical unit SYS004. A TLBL of QW$BKUP must also be given in the execution JCL. Output consists of 80-byte unblocked records in dictionary image. Under MVS, the output is written on the device assigned to the DDNAME=QWBKUP, blocked 100 records per block. In either case, when each member is dumped, a confirmation message is issued. When the backup has finished, the job stream is flushed to the end of the QWDMAINT job step. This is so that the tape corresponds to a known point in time.

:BEGIN xxxxx

:BEGIN MEDIA=partid

:BEGIN JCL=partid ,USAGE=SEL SOI SOW SOO

This command is used to signal the beginning of data that is to be associated with the member being described. Any data of the type specified that currently exists in the member is deleted. All data statements that follow the :BEGIN are added to the member. Data continues until a colon (:) command or end of file is detected.

For example, when :BEGIN RECORD is processed, all fields currently defined are deleted, and all new fields that follow are added to the member.

There are three forms of this command. Form 1 calls for a single word option as defined below.

This is the only form allowed for MVS users.

TABLE	This parameter denotes that the following data records are to be associated as table elements.
PROC	This parameter denotes that the following data records are to be associated as VISION:Report source statements that make up the procedure.
zRECORD	This parameter denotes that the following field commands are to be associated as the record description of the file being maintained.

The second form is used to signal the start of SELECT/REPORT media commands for the indicated VSE partition. The partition (BG,FB,FA,F1 through F9) or ALL can be specified. This form is allowed only for VSE users.

MEDIA=partid	This keyword is the only keyword needed or allowed for MEDIA processing. The SELECT/REPORT commands follow.

The third form is used to signal the start of job control information (JCL) for the partition and the type indicated. This form is allowed only for VSE users.

JCL=partid — This keyword is used to indicate that the JCL statements that follow are to be associated with the indicated VSE partition. Again, the value coded here must be ALL or BG,FB,FA,F1 through F9. The following 80-byte statements become the JCL that is stored in the proper dictionary format.

USAGE=xxx — This keyword is used to qualify what type of JCL is being described. Valid are:

SEL	Selection
SOI	Sort Input
SOW	Sort Word
SOO	Sort Output

If this keyword is omitted, SEL is assumed. This keyword, if specified, must follow the JCL keyword.

:CHANGE SEQ=from ,to

This command is valid only during MODIFY mode (see :MODIFY xxxxxx in this chapter). It is not allowed when :MODIFY RECORD is specified. The only keyword allowed, which is also required, is SEQ. The value given must be a numeric sequence number that was taken from a :LIST of the member at which point the change is to occur. The specified sequence is deleted and all following data records are added (inserted) at that point. If the sequence number does not exist, the following data records are added to the end of the member. Unpredictable results occur if more than one :CHANGE command is issued using sequence numbers that overlap.

Optionally, a range of sequence numbers can be replaced by placing a comma and the ending sequence number immediately after the beginning number. For example, :CHANGE SEQ=2,4 replaces sequence numbers 2, 3, and 4 with all data records that follow until the next colon command or until end of file occurs.

:CREATE member

This command allows for the addition of a new member to be added to the dictionary. All specifications, after they have been entered and verified, are added to the dictionary. An implied :LIST is performed to produce a control document for storage.

The member must not already exist or an error message is issued, and the job stream is flushed to the next system command.

If the member is specified as QW$DICT, the physical dictionary must have previously been deleted and defined using IDCAMS, because :CREATE of the dictionary attempts to initially load the VSAM dictionary cluster.

:DELETE ENTRY=field

:DELETE SEQ=from ,to

This command is valid only during MODIFY mode (see :MODIFY xxxxxx in this chapter). There are two forms of this command. The first form is valid only when :MODIFY RECORD is specified. It allows for an existing field to be deleted (removed). For example:

```
:DELETE ENTRY=CUST-NAME
```

The second form is for use in all other MODIFY modes. It allows individual sequenced statements or groups of sequenced statements to be deleted (removed). Specify the first statement to be removed. Optionally, specify the range by coding a comma and the ending sequence to be deleted. The sequence numbers must exist or an error message is issued. For example:

```
:DELETE SEQ=12
```

or

```
:DELETE SEQ=7,13
```

:INSERT SEQ=from

This command is only valid in MODIFY mode (see :MODIFY xxxxxx in this chapter). However, it is not allowed when :MODIFY RECORD is specified. The only keyword allowed, which is also required, is SEQ. The value given must be a numeric sequence number that was acquired from a :LIST of the member. The data statements that follow are added after that sequence until the next colon command or until end of file occurs. If the sequence number does not exist, the following data statements are added to the end of the member. Results are unpredictable if more than one :INSERT for any given sequence number is specified.

:LIST xxxxxx

Use of this command is permitted only while in :ACCESS mode. An implied :LIST ALL is performed if a change is requested.

Only one option can be specified per :LIST command. Options are as follows:

ALL	This is the default if no option is specified. All information about a member is listed.
FIELDS	Normally when a record description is listed, it is sorted by starting and ending positions within the record. However, if an alphabetical listing by field name is desired (this is usually helpful for end users), use this option.
JCL	Use this option only if the job control statements associated with this member are to be listed. This parameter has no meaning for MVS users.
MEDIA	Use this option only if the REPORT/SELECT MEDIA information is to be listed. This parameter has no meaning for MVS users.
MEMBERS	This option is only effective if the dictionary standards (QW$DICT) are accessed. An alphabetical list of every member found in the dictionary is printed.
RECORD	Use this option to list only the record description of a file. The output is listed in order by starting and ending locations of the fields.

:MODIFY xxxxxx

:MODIFY MEDIA=partid

:MODIFY JCL=partid ,USAGE=SEL SOI SOW SOO

This command has the same structure as the :BEGIN command, except that the MODIFY flag is set on to allow the other commands (:INSERT/:CHANGE/:DELETE). Unlike :BEGIN, this command does not delete any existing data of the type indicated. This is so that the member can be maintained without the data having to be entirely replaced or specified again. Also, when :MODIFY RECORD is specified, it denotes that the following field commands are to be examined and, if the field already exists, only the specified parameters are changed and the record is then replaced. If the field does not exist, it is validated and then added to the record description.

:OPTION TSIZE=nnnn

This command changes the storage allocation of the virtual table used as an intermediate work area to perform the update of dictionary members. Normally, the default of TSIZE is 64K of virtual storage that must be available in the VSE partition GETVIS area, or the MVS region size. This command must be the first command read by the system and specifies the number of Kilobytes of memory to be reserved. The minimum is TSIZE=2, with a maximum of the largest partition GETVIS that can be created. If unable to acquire storage, an error message is issued. There can be one and only one :OPTION statement per execution of QWDMAINT.

:PUNCH

Use of this command is only permitted during :ACCESS mode. The command translates an existing dictionary member into the keyword system commands used by the dictionary maintenance system.

Only one option can be specified per :PUNCH command. Options are as follows:

ALL	This is the default if no option is specified. All information about a member is punched.
JCL	Use this option to cause the job control statements associated with this member to be punched. This parameter has no meaning for MVS users.
MEDIA	Use this option only if the REPORT/SELECT MEDIA information is to be punched. This parameter has no meaning for MVS users.
RECORD	Use this option to punch only the record description of a file.

:PURGE member

This command deletes the member from the dictionary. If the member is not found, an error message is issued. A confirmation message is issued upon completion of the command.

:RESTORE member

This command takes a previously QWDMAINT made backup tape and loads either the entire dictionary, or individual members as specified. For VSE, the tape must be SYS005 with a TLBL of QW$RSTR. Use a DDNAME of QWRSTR if you are an MVS user. DDNAME=QWRSTR if the member name specified is ALL, the entire dictionary is loaded. This requires that the dictionary be deleted and defined using the VSAM IDCAMS utility. There can be only one :RESTORE ALL command for a job stream, and it must be the first and only command given.

Multiple :RESTORE commands naming members to be restored are allowed, and need not be placed in any required sequence. However, all :RESTORE commands must be grouped together and must be the first and only command type in the entire job stream. If a member is not found on the tape, a message is issued and the next :RESTORE command is tried. If the member to be restored is already in the dictionary, a message indicating that it was on the file, but is now replaced, is issued. Up to 64 individual members can be restored in a single execution of the program.

:UPDATE member

This command operates exactly like :ACCESS except that all changes requested, after validation, cause the dictionary to be modified to reflect those changes.

SELECT MEDIA=xxxx,SYSNR=nnn

This command is valid only for VSE users; it has no meaning for MVS users.

Use this command to supply the MEDIA device type and programmer logical unit number that is used to construct the VISION:Report I/O parameter statement for any PULL job. Usually, the job stream produced consists of three steps: Pull (SELECT), sort, and list (REPORT). The SELECT step passes the master file and selects output records to be saved on a new disk file. Then, the new file is sorted (if necessary), after which the file is passed again to be listed (to produce a report).

The SELECT/REPORT media can be specified for any or ALL VSE partitions. This command is only valid after the :BEGIN MEDIA= or :MODIFY MEDIA= command. Valid keywords are as follows:

MEDIA=xxxx	Use this parameter to specify the device type for the MEDIA description being described. Valid entries are the same as those for the DEVICE Group 2 command.
SYSNR=nnn	This parameter provides the programmer logical unit number (SYSnnn) that is assigned to the intermediate file.

REPORT MEDIA=xxxx,SYSNR=nnn

This command is valid only for VSE users; it has no meaning for MVS users.

This command is identical to the SELECT command except that this command identifies the list media to be associated with the generated VISION:Report program. The positioning and keywords as described above for SELECT must also be used for REPORT. For example:

```
BEGIN MEDIA=ALL
SELECT MEDIA=SD5C,SYSNR=010
REPORT MEDIA=SD5C,SYSNR=011

FIELD NAMEx...x,LEN=nnnn,CALDEC=n
                        ,EDIT=x
                        ,HDR=(x...x)
                        ,POS=(x...x)
                        ,PRTDEC=n
                        ,START=nnnn
                        ,TOTDIG=nn
                        ,TYPE=x
```

Use this command to describe a data field within a file. All field commands must follow :BEGIN RECORD or :MODIFY RECORD.

NAME=x...x — The data name by which this field is to be known. Maximum of 14 characters, first of which must be alphabetic (A-Z); no embedded spaces allowed. This parameter is required.

LEN=nnnn — The length (in characters) of the field. Maximum of 4 digits; this parameter is required. If the field is packed (see TYPE=x in this chapter), supply the byte length, not the digit length.

CALDEC=n — The number of decimal places to be considered during any computations using this field. If the field is not used in calculations, omit this parameter. Otherwise, specify a single digit (0-9) to provide decimal scaling.

EDIT=x — The VISION:Report edit code used to format an output field. If no editing is required, this parameter can be omitted. Refer to the VISION:Report Reference Manual.

HDR=(x...x) — The override heading is used to provide a column heading if a field is selected to be printed. When VISION:Forms does not specify a different heading, this heading is used. If this option is not specified, the column heading is the field name. The value specified cannot exceed 30 characters.

POS=(x...x)	Use this parameter as a shortcut to provide the starting and ending positions and the data type of a field in the same format as a VISION:Report field definition. For example: POS=(1-10) describes a field starting in position 1 and ending in position 10. Thus, the length of the field is the ending position plus 1, minus the starting position, or in this case (10+1) - 1 = 10. Optionally, the data type can be given by placing a dash and the TYPE after the ending position. For example: POS=(102-105-P) describes a field with a length of 4 bytes of packed decimal format (which would yield 7 digits) starting at position 102 of the record. If the TYPE is not specified this way, it is assumed to be blank and, therefore, character data.
PRTDEC=n	The number of decimal places to be printed if this field is selected to be reported upon. If this field is not a number, omit this parameter. Otherwise, specify a single digit (0-9) to indicate the number of decimal places to be printed.
START=nnnn	The beginning position of the field within the record. Maximum of 4 digits; this operand is optional. If not specified, the next position available after the last field defined is used. If this is the first field being described, position 1 is assumed when this operand is omitted.
TOTDIG=nn	The total number of digits specifies the maximum field significance. Normally VISION:Forms adjusts the size of numeric fields to compensate for arithmetic overflow. With this parameter, you can specify the absolute number of digits to be stored. Care should be taken so as not to lose leftmost digits.
TYPE=x	This keyword specifies the data type of the field. If not specified, character data is assumed. If packed decimal is required, specify the letter 'P'.

Group 2 Commands

Note: In the following tables, M indicates mandatory, O indicates optional, and * indicates that the command is only valid in a VSE environment.

Use the following key for the tables in this section:

M	Mandatory
O	Optional
*	Command only valid in a VSE environment.

For tables only:

M	ARGLEN=nn	Argument length
M	ARGLOC=nn	Argument location
O	ARGTYP=x	Argument type
O	FUNAME=x...x	Function name (maximum 14 characters)
O	FUNHDR=(x...x)	Function header (maximum 30 characters)
O	FUNLEN=nn	Function length
O	FUNLOC=nn	Function location
M	MAXENT=nnnn	Maximum entries

For QW$DICT dictionary standards only:

* O	BGRDR=xxx	BG cuu of SYSIN
* O	F1RDR=xxx	F1 cuu of SYSIN
* O	F2RDR=xxx	F2 cuu of SYSIN
* O	F3RDR=xxx	F3 cuu of SYSIN
O	LANG=xx	Language conversion
* O	SYSIN=xxx	Same as BGRDR
* O	SYSPCH=xxxx	Output to card or disk

For files only:

O	AREA=xxx	VISION:Report area being described
M	BLKL=nnnnn	Block length
M	DEVICE=xxxx	Device type
* O	FILES=nnn	Number of files on tape
* O	GENERIC=x	Assignments to be generic
* O	LABELS=x	Standard or non-standard tape labels
* O	LBL=xxxxxxxx	Label area file name
O	PROC=xxxxxxxx	Override input procedure
O	RECL=nnnnn	Record length
* O	REWIND=x	Tape disposition

O	TITLE=(x...x)	File description (maximum 55 characters)
* O	TM=x	Tape mark present for tape file
* M	UNIT=nnn	Logical unit number

Group 2 Command Descriptions

AREA=xxx

The value specified here is used to indicate which VISION:Report area is being described by the record layout associated with this FILE. Valid specifications are as follows:

INF	DET	INC	IND	OFB	OFC	OFD	PTA	PTB
PTC	PTD	FUN	TBH	TSA	WST	VAL	PUN	PRT

If omitted, the default is INF.

ARGLEN=nn

Argument length is a two-digit number indicating the number of characters of the argument portion of a table.

ARGLOC=nn

Argument location is a two-digit number specifying in which column the table argument starts. Usually, most table arguments start in position 01 of a table record.

ARGTYP=x

Argument type is used to indicate if the table argument is in packed format. Either specify a BLANK or 'P'. When 'P' is specified, the argument is a packed decimal value; otherwise, it is a character value.

BGRDR=xxx

VSE only. This option indicates the SYSIN cuu for the BG partition. When VISION:Forms generates a CLOSE SYSIN command, it inserts this device address if it is running in the background partition. If this parameter is not specified, a generic CLOSE of SYSIN to READER is generated.

BLKL=nnnnn

Block length can be a five-digit number indicating the block size of the file being described. For variable or undefined files, enter the maximum block size that can occur.

DEVICE=xxxx

This option describes the DASD type on which the file resides. For VSE users, this command is required. For MVS users, the command is only required if the device type is VSAM or if a VISION:Report procedure is to be used. In this situation, only the codes KSDS, ESDS, RRDS, and USER are allowed.

See MEDIA entries under ' records in Section 7.7 for a full list of all values.

FILES=nnn

VSE only. Multiple files; applies to tape files only. If your input contains more than one generation of the file, enter a 3-digit number of how many files (not reels) are being processed.

FUNAME=x...x

Function data name is used to automatically attach the function of a table to VISION:Forms so that it can be referenced by other commands (namely **PRINT** and **CALC** in the chapter "Writing Verbs"). This option is valid for tables only, and the data name must not have been defined elsewhere in the VISION:Forms program.

FUNHDR=(x...x)

Function header is used to provide a default column heading for a function table that is used frequently in other VISION:Forms programs.

FUNLEN=nn

Function length is a two-digit number indicating the number of characters of the function portion of a table.

FUNLOC=nn

Function location is a two-digit number specifying in which column the table function starts. Usually, most table functions start immediately after the argument.

F1RDR=xxx

VSE only. This option specifies the SYSIN cuu for VISION:Forms programs that run in the F1 partition. When VISION:Forms generates a CLOSE SYSIN command, it inserts this device address if it is running in the F1 partition.

F2RDR=xxx

VSE only. SYSIN cuu for VISION:Forms programs that run in the F2 partition. See F1RDR.

F3RDR=xxx

VSE only. SYSIN cuu for VISION:Forms programs that run in the F3 partition. See F1RDR.

GENERIC=x

VSE only. Enter the letter 'G' to cause generic assignments to be generated for multi-step jobs.

LABELS=x

VSE only. Tape label information applies to tape files only. Enter the letter 'S' for standard labels or 'N' for non-standard labels. If left blank or omitted, standard labels are assumed.

LANG=xx

Alternate language is valid only during dictionary standards processing. Enter 'FR' to translate from French or 'GR' to translate from German. If omitted, English is assumed.

LBL=xxxxxxx

VSE only. Standard label track file name specifies the DTF name of the file as it resides in the Label Area.

MAXENT=nnnn

Maximum table entries is a four-digit number specifying the total number of table 'slots' to be reserved.

PROC=xxxxxxxx

Replace procedure name is used to denote which dictionary member contains a procedure that is used to process the file during the input phase of the VISION:Forms program.

RECL=nnnnn

Record length can be a five-digit number indicating the record size of the file being described.

REWIND=x

VSE only. Tape disposition applies to tape files only. A specification of the letter 'U' rewinds at open, and rewinds and unloads at close. If 'N' is coded, the tape is not positioned at either open or close. The default is to rewind at both open and close.

SYSIN=xxx

VSE only. This option is identical to BGRDR. Refer to the description of BGRDR for coding details.

SYSPCH=xxxx

VSE only. This option denotes if VISION:Forms should output the generated VISION:Report on SYSPCH or to the disk extent QW$DPUN. Specify CARD to have output routed to SYSPCH. Specify DISK to have output placed on QW$DPUN.

TITLE=(x...x)

This option is used to describe the DASD file contents that this member describes.

TM=x

VSE only. Tape mark information applies to tape files only. If the input tape file is not labeled or has non-standard labels and is not preceded by a tape mark, code the letter 'N' for this option.

UNIT=nnn

VSE only. The input unit number (SYSnnn) must specify a valid programmer logical unit number from 001 to 254 (avoid using SYS000).

How to Initially Build a VSE Dictionary

Before running jobs or, in the following cases, examples, the VISION:Forms physical dictionary KSDS cluster must have been DEFINED using the VSAM utility program, IDCAMS. The following job stream could be used to load the installation standards.

```
1   // EXEC QWDMAINT,SIZE=AUTO
2   :CREATE QW$DICT
3   SYSIN=02A
4   SYSPCH=DISK
5   :BEGIN MEDIA=ALL
6   SELECT MEDIA=SD5C,SYSNR=010
7   REPORT MEDIA=SD5C,SYSNR=011
8   :BEGIN JCL=ALL,USAGE=SOI
9   // ASSGN SYS002,X'152'
10  // DLBL SORTIN1,'STDSORT.INPUT',0
11  // EXTENT SYS002,999999,1,0,12,1200
12  :BEGIN JCL=ALL,USAGE=SOW
13  // ASSGN SYS003,DISK,VOL=111111,SHR
14  :BEGIN JCL=ALL,USAGE=SOO
```

```
15    // ASSGN SYS001,X'152'
16    // DLBL SORTOUT,'STDSORT.INPUT',0
17    // EXTENT SYS001,999999,1,0,12,1200
18    /*
```

Command explanations:

1 Executes the maintenance program.

2 The :CREATE command is required. It specifies the reserved word QW$DICT, which implies the dictionary standards are being described.

3 The SYSIN Group 2 command specifies which device is to be reassigned as SYSIN. This is normally your spooled reader.

4 The SYSPCH Group 2 command specifies that the resultant VISION:Report program is written to a disk extent of the name QW$DPUN. Appropriate JCL should be in the partition standard label area.

5 The :BEGIN command specifies the beginning of MEDIA information that is used as default if not given explicitly.

6 The SELECT command describes the disk type and format, along with the programmer logical unit that is used to create the SELECTION file.

7 The REPORT command is exactly the same as SELECT, but it describes the REPORT file, which is the output from the sorting of the SELECT file.

8 This :BEGIN command specifies the beginning of sort input JCL that is used by default for all partitions if no other sort JCL is specified for a particular partition.

9-11 These three statements are inserted into a generated VISION:Report program for sort input as default information, if sort JCL is not specified for an individual file or partition.

12 This :BEGIN command specifies the beginning of sort work JCL that is used by default for all partitions if no other sort JCL is specified for a particular partition.

13 This statement is inserted into a generated VISION:Report program for sort work as default information if no other sort JCL is given for the file or partition in use.

14 This :BEGIN command signals the start of sort output JCL that is used by default for all partitions if no other sort JCL is specified.

15-17 These statements are inserted into a generated VISION:Report program for sort output as default JCL if no other sort JCL is specified for the file or the partition in use.

18 End of input to program.

How to Initially Build an MVS Dictionary

Before running jobs or, in the following cases, examples, the VISION:Forms physical dictionary KSDS cluster must have been DEFINED using the VSAM utility program, IDCAMS. The following job stream must be used to initialize the dictionary.

```
1       //STEP1 EXEC PGM=QWDMAINT
        //SYSPRINT DD SYSOUT=A
        //QWDICT   DD DSN=USER.QWDICT,DISP=SMR
        //SYSIN    DD *
2       CREATE QW$DICT
```

Command explanations:

1 Executes the maintenance program.

2 The :CREATE command is required and specifies the reserved word QW$DICT, which implies the dictionary, is being initialized.

How to List the Installation Standards

Use this job to generate a printout of the installation standard information that was built with a :CREATE QW$DICT job.

```
1   // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL

1   //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
    //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
    //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
    //SYSIN    DD *                                    <<— MVS JCL
2   :ACCESS QW$DICT

3   :LIST ALL

4   /*
```

Command explanations:

1 Execute the maintenance program.

2 :ACCESS command is required to identify which dictionary member is to be processed. In this case, the reserved word QW$DICT denotes that the dictionary standards are to be retrieved.

3 The :LIST command is used to cause a complete listing of the member identified in the :ACCESS command above.

4 End of program input.

How to Add a File Description

In this example, a new file called CUSTFILE is being defined to the dictionary. The job stream would look something like the following.

```
1   // EXEC QWDMAINT,SIZE=AUTO                                  <<— VSE JCL
1   //STEP1 EXEC PGM=QWDMAINT                                  <<— MVS JCL
    //SYSPRINT DD SYSOUT=A                                     <<— MVS JCL
    //QWDICT   DD DSN=USER.QWDICT,DISP=SHR                     <<— MVS JCL
    //SYSIN    DD *                                            <<— MVS JCL
2   :CREATE CUSTFILE
3   TITLE='CUSTOMER MASTER FILE'
4   DEVICE=IS40                                                <<—VSE only.
5   BLKL=1024
6   RECL=256
7   LBL=CUSTM                                                  <<—VSE only
8   UNIT=005                                                   <<—VSE only
9   :BEGIN RECORD
10  FIELD NAME=CUSTM-KEY,LEN=7,HDR='CUSTM-NUMBER'
11  FIELD NAME=CUSTM-NAME,LEN=40
12  FIELD NAME=CUSTM-ADDR,LEN=30,HDR='CUSTM-ADDRESS'
13  FIELD NAME=CUSTM-CITY,LEN=15
14  FIELD NAME=CUSTM-STATE,LEN=2
15  FIELD NAME=CUSTM-ZIP,START=100,LEN=5
16  FIELD NAME=CUSTM-BALANCE,LEN=5,TYPE=P,CALDEC=2,PRTDEC=2,
17   EDIT=C,HDR='AMT,REC''D'
18  FIELD NAME=CUSTM-AMOUNT,POS=(105-109-P),CALDEC=2,HDR='AMT-
    DUE'
19  FIELD NAME=CUSTM-DATE-LP,LEN=6,CALDEC=0,EDIT=D,
20   HDR='DATE OF,LAST PAY'
21  FIELD NAME=CUSTM-LP-MM,START=110,LEN=2
22  FIELD NAME=CUSTM-LP-DD,LEN=2
23  FIELD NAME=CUSTM-LP-YY,LEN=2
24  /*
```

Command explanations:

1 Invoke the dictionary maintenance program.

2 The :CREATE command is required and names the member to be added to the dictionary. In this case, CUSTFILE must not exist now.

3 The TITLE Group 2 command is optional and provides a description of the file contents.

4 The VSE DEVICE Group 2 command is required and names the disk type and format of the file being described. In this case, ISAM 3340.

5 The BLKL Group 2 command is required and gives the BLOCK LENGTH.

6 The RECL Group 2 command is required and gives the RECORD LENGTH.

7 The VSE LBL Group 2 command is optional and specifies the standard label track name for the file.

8 The VSE UNIT Group 2 command is required and names the logical unit (or SYSnr) that must be assigned to the file.

9 The :BEGIN RECORD command is required and indicates that the FIELD commands following describe the record layout of the file.

10 The FIELD command describes a single field entry within the file. The parameters NAME and LEN are required. The optional START parameter could have been given to specify the starting position within the record. The default, in this case, is 1. An override column heading is specified by using the HDR parameter.

11 This FIELD is called CUSTM-NAME and is located in positions 8 through 47 (length of 40 characters).

12 CUSTM-ADDR is in positions 48 through 77 for a field length of 30. An optional column heading has also been specified.

13 CUSTM-CITY is in positions 78 through 92 (a length of 15).

14 The field CUSTM-STATE is in positions 93 and 94.

15 CUSTM-ZIP starts in position 100 (thus leaving positions 95 through 99 unassigned). The length of 5 characters places the ending position of the field at 104.

16 CUSTM-BALANCE is a 5-byte packed field. Therefore, the field can hold up to 9 digits. The field is located in positions 105 through 109. CALDEC indicates the number of implied decimal positions to be used if this field is involved in a calculation. PRTDEC denotes the number of decimal positions to print if the field is to be printed. Since the last operand has a comma after the value, this signals that the command is continued on the next statement.

17	The EDIT code specifies to punctuate with commas. This statement is a continuation for the field CUSTM-BALANCE. It specifies the optional column heading override. **Note:** The embedded double quotation marks are translated into a single quotation mark.
18	CUSTM-AMOUNT is a redefinition of the field CUSTM-BALANCE. The alternate method of using POS was used to indicate the starting and ending positions, along with the field type. Two decimal positions are used during calculations, and an override heading is given.
19	This field is in positions 110 through 115. The field is numeric, but has no assumed decimal places. If the field is selected to be printed, it is edited like a date (MM/DD/YY). This command is continued on the next statement.
20	This statement is a continuation of the prior field CUSTM-DATE-LP. It specifies an override column heading.
21-23	These three fields redefine the CUSTM-DATE-LP field into the month, day, and year.
24	End of program input.

How to List a File Description

Use this job to print the dictionary entry for any file. The job stream would look like the following.

```
1   // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL

1   //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
    //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
    //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
    //SYSIN    DD *                                    <<— MVS JCL
2   :ACCESS CUSTFILE

3   :LIST ALL

4   /*
```

Command explanations:

1 Execute the dictionary maintenance program.

2 :ACCESS is required to name the member CUSTFILE to be retrieved.

3 The :LIST is used to cause a printout of the member.

4 End of program input.

How to Modify a File Description

Use this job to respecify, add, change, or delete any fields, JCL, or media associated with a dictionary file entry. Below is an example that updates the file entry CUSTFILE.

```
1       // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL

1       //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
        //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
        //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
        //SYSIN    DD *                                    <<— MVS JCL
2       :UPDATE CUSTFILE

3       UNIT=025                                           <<—VSE only

4       :BEGIN JCL=ALL,USAGE=SEL                           <<—VSE only

5       // ASSGN SYS025,DISK,VOL=DAP01A,SHR                <<—VSE only

6       // DLBL CUSTM,'CUST.MASTER.FILE',99/365,ISE        <<—VSE only

7       // EXTENT SYS025,,4,1,,360,2                       <<—VSE only

8       // EXTENT SYS025,,1,2,,372,120                     <<—VSE only

9       :MODIFY RECORD

10      FIELD NAME=CUSTM-KEY,HDR='CUST-NUMBER'

11      FIELD NAME=CUSTM-DIV,START=5,LEN=3

12      :DELETE ENTRY=CUSTM-LP-MM
```

Command explanations:

1 Execute the maintenance program.

2 This command is required to identify the member to be maintained. It also denotes that the following changes are recorded into the dictionary.

3 The VSE logical unit number is being changed to SYS025.

4 This VSE command signals the beginning of the JCL that is included in the resultant job stream whenever this file is referenced in a VISION:Forms program.

5

6

7 These VSE JCL statements are inserted in the VISION:Report job stream whenever this file is named in a FILENAME statement.

8

9 This command indicates that we are going to add, change, or delete fields into the record description of the file.

10 This command is changing the override heading for the field CUSTM-KEY.

11 Define a new field CUSTM-DIV in positions 5 through 7.

12 Remove the field CUSTM-LP-MM from the record description.

How to Remove Dictionary Entries

The following job is used to remove the JCL associated with the CUSTFILE entry. One reason to do this is that the VSE label set was placed in the system standard label area and does not need to be loaded from the dictionary, or the entry was loaded into an MVS dictionary rather than the VSE dictionary.

```
1   // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL
1   //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
    //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
    //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
    //SYSIN    DD *                                    <<— MVS JCL
2   :UPDATE CUSTFILE
3   :BEGIN JCL=ALL,USAGE=SEL
4   /*
```

Command explanations:

1 Invoke the maintenance program.

2 Identify the member to be changed.

3 Signal the beginning of the VSE JCL.

4 End of program input.

If the entire entry needs to be deleted, use the following:

```
1   // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL
1   //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
    //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
    //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
    //SYSIN    DD *                                    <<— MVS JCL
2   :PURGE CUSTFILE
3   /*
```

Command explanations:

1 Execute the maintenance program.

2 Identify which member to :PURGE.

3 End of program input.

How to Convert an Old Dictionary

Normally, a new user would just start coding maintenance commands and save them in source form in a source library or an online editor.

However, existing users will sometimes want to convert a member (or file) of their old dictionary to the QWDMAINT format. Perhaps you need to set up a new file called CUSTMAST whose fields closely resemble the existing CUSTFILE file. It would be convenient to have CUSTFILE converted to QWDMAINT format so that you could make the needed changes and :CREATE CUSTMAST. To convert CUSTFILE, the job stream is:

```
1       // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL

1       //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
        //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
        //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
        //SYSPUNCH DD SYSOUT=B                             <<— MVS JCL
        //SYSIN    DD *                                    <<— MVS JCL
2       :ACCESS CUSTFILE

3       :PUNCH ALL

4       /*
```

Command explanations:

1 Invoke the maintenance program.

2 Identify which member to retrieve.

3 Convert the entire member.

4 End of program input.

How to Backup the Dictionary

The following VSE job causes a backup tape of the physical dictionary to be created.

```
1       // ASSGN SYS004,180

2       // TLBL QW$BKUP

3       // EXEC QWDMAINT,SIZE=AUTO

4       :BACKUP

5       /*
```

Command explanations:

1 Assign the output tape drive to SYS004.

2 Supply the file name for the tape label.

3 Execute the maintenance system with a partition GETVIS area.

4 Backup the dictionary.

5 End of program input. (No other control cards will be processed.)

The following MVS job causes a backup tape of the physical dictionary to be created.

```
1   //STEP1 EXEC PGM=QWDMAINT
    //SYSPRINT DD SYSOUT=A
    //QWDICT   DD DSN=USER.QWDICT,DISP=SHR
2   //QWBKUP   DD DSN=QW.BACKUP,DISP=(,CATLG),UNIT=TAPE
3   //SYSIN    DD *
4   :BACKUP
```

Command explanations:

1 Normal JCL required to invoke the maintenance program.

2 This DD statement is where the output of the backup is placed.

3 Indicates that the input follows this statement.

4 Backup the dictionary.

No other control cards will be processed.

How to Restore the Dictionary

Note: The VSAM cluster must be deleted and redefined using IDCAMS prior to restoring the VISION:Forms dictionary.

VSE

The following VSE job restores the entire backup tape to the physical dictionary VSAM KSDS cluster.

```
1   // ASSGN SYS005,180
2   // TLBL QW$RSTR
3   // EXEC QWDMAINT,SIZE=AUTO
4   :RESTORE ALL
5   /*
```

Command explanations:

1 Assign the input tape drive to SYS005.

2 Supply the file name for the tape label.

3 Execute the maintenance system with a partition GETVIS area.

4 Restore the entire dictionary.

5 End of program input.

MVS

The following MVS job restores the entire backup tape to the physical dictionary VSAM KSDS cluster.

```
1       //STEP1 EXEC PGN=QWDMAINT
        //SYSPRINT DD SYSOUT=A
        //QWDICT   DD DSN=USER.QWDICT,DISP=SHR
2       //QWRSTR   DD DSN=QW.BACKUP,DISP=OLD
3       //SYSIN    DD *
4       :RESTORE ALL
```

Command explanations:

1 Normal JCL required to invoke the maintenance program.

2 This DD statement is where the output of the backup is placed.

3 Indicates that the input follows this statement.

4 Restore the entire dictionary.

The following maintenance control statements restore only the members QJPROC and CUSTFILE. They replace any existing entries if they exist.

```
1    :RESTORE QJPROC
2    :RESTORE CUSTFILE
```

Command explanations:

1 Restore the QJPROC member.

2 Restore the CUSTFILE member.

How to Create a Procedure

The following is a job that adds a VISION:Report procedure to the dictionary. In this case, it is a routine to handle the conversions of dates from Julian to MM/DD/YY format. This procedure assumes the date is in positions 1 through 5 of every input record and that the new date is placed in WST1-8.

```
1    // EXEC QWDMAINT,SIZE=AUTO                                  <<— VSE JCL
1    //STEP1 EXEC PGM=QWDMAINT                                  <<— MVS JCL
     //SYSPRINT DD SYSOUT=A                                     <<— MVS JCL
     //QWDICT   DD DSN=USER.QWDICT,DISP=SHR                     <<— MVS JCL
     //SYSIN    DD *                                            <<— MVS JCL
2    :CREATE QJPROC
3    :BEGIN PROC
4    010 CALL QUIKDATE C'01' INF1-5 C'YYDDD ' WST1
5    IF VAL46-49 NOT ZEROS
6    PRINTHEX INF1-5
7    PRINTHEX VAL46-49
8    ABEND.
9    /*
```

Command explanations:

1 Execute the maintenance program.

2 This adds a new member called QJPROC to the dictionary.

3 Indicates that the member is a procedure.

4-8 These VISION:Report statements make up the procedure.

9 End of program input.

How to List a Procedure

The following prints the procedure QJPROC from the dictionary.

```
1    // EXEC QWDMAINT,SIZE=AUTO                                  <<— VSE JCL

1    //STEP1 EXEC PGM=QWDMAINT                                  <<— MVS JCL
     //SYSPRINT DD SYSOUT=A                                     <<— MVS JCL
     //QWDICT   DD DSN=USER.QWDICT,DISP=SHR                     <<— MVS JCL
     //SYSIN    DD *                                            <<— MVS JCL
2    :ACCESS QJPROC

3    :LIST ALL

4    /*
```

Command explanations:

1 Invoke the maintenance program.

2 Identify the member to be retrieved.

3 List the entire member.

4 End of program input.

The report generated would look like the following:

```
SEQ#  --PROCEDURE---
001   010 CALL QUIKDATE C'01' INF1-5 C'YYDDD ' WST1
002       IF VAL46-49 NOT ZEROES
003         PRINTHEX INFl-5
004         PRINTHEX VAL46-49
005         ABEND.
```

How to Modify a Procedure

The following is a demonstration of maintaining a procedure using the :LIST of QJPROC.

```
1    // EXEC QWDMAINT,SIZE=AUTO                                  <<— VSE JCL

1    //STEP1 EXEC PGM=QWDMAINT                                  <<— MVS JCL
     //SYSPRINT DD SYSOUT=A                                     <<— MVS JCL
     //QWDICT   DD DSN=USER.QWDICT,DISP=SHR                     <<— MVS JCL
     //SYSIN    DD *                                            <<— MVS JCL
2    :UPDATE QJPROC

3    :MODIFY PROC

4    :DELETE SEQ=3

5    :CHANGE SEQ=4

6    DOHEADERS
```

```
7       MOVE C'PROCEDURE QJPROC CANCELLED' TO PRT1
8       PRINT
9       /*
```

Command explanations:

1 Execute the maintenance program.

2 Identify which member is to be updated.

3 Modify the procedure statements.

4 Delete sequence 3.

5-8 Statement 4 is deleted and 2 new statements are added.

9 End of program input.

The new procedure now appears as:

```
SEQ --PROCEDURE---
#
001 010 CALL QUIKDATE C'01' INF1-5 C'YYDDD ' WST1
002     IF VAL46-49 NOT ZEROS
003       DOHEADERS
004       MOVE C'PROCEDURE QJPROC IS CANCELLING' TO PRT1
005       PRINT
006       ABEND.
```

How to Create a Table

The following job stream creates a table of months called MONTHTAB.

```
1       // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL
1       //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
        //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
        //QWDICT   DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
        //SYSIN    DD *                                    <<— MVS JCL
2       :CREATE MONTHTAB
3       MAXENT=12
4       ARGLOC=1
5       ARGLEN=2
6       FUNLOC=3
7       FUNLEN=10
8       FUNAME=MO-DESC
```

```
9       :BEGIN TABLE
10      01JANUARY
11      02FEBRUARY
12      03MARCH
13      04APRIL
14      05MAY
15      06JUNE
16      07JULY
17      08AUGUST
18      09SEPTEMBER
19      10OCTOBER
20      11NOVEMBER
21      12DECEMBER
22      /*
```

Command explanations:

1 Execute the maintenance program.

2 Add a new member called MONTHTAB to the dictionary.

3 Since this member describes a table, the required Group 2 commands must be specified. This is the maximum number of table entries for the table.

4 Argument location is in position 1 of a table record.

5 The argument length is two characters.

6 This table also has a function. The function begins in position 3 of the table record.

7 The function length is ten characters.

8 The function has a name called MO-DESC.

9 The start of the table records.

10-21 These are the actual table records with the argument keyed in the first two positions, and the function in the following ten positions.

22 End of program input.

How to List a Table

The following lists the table entry MONTHTAB.

```
1       // EXEC QWDMAINT,SIZE=AUTO                          <<— VSE JCL

1       //STEP1 EXEC PGM=QWDMAINT                          <<— MVS JCL
        //SYSPRINT DD SYSOUT=A                             <<— MVS JCL
        //QWDIC    DD DSN=USER.QWDICT,DISP=SHR             <<— MVS JCL
        //SYSIN    DD *                                    <<— MVS JCL
2       :ACCESS MONTHTAB

3       :LIST ALL

4       /*
```

Command explanations:

1 Execute the maintenance program.

2 Identify the member to be retrieved.

3 List the entire table.

4 End of program input.

The following report would be generated:

```
MAXENT=00l2
ARGLOC=01
ARGLEN=02
ARGTYP=
FUNLOC=03
FUNLEN=10
FUNAME=MO-DESC

SEQ#  --PROCEDURE---
001   01JANUARY
002   02FEBRUARY
003   03MARCH
004   04APRIL
005   05MAY
006   06JUNE
007   07JULY
008   08AUGUST
009   09SEPTEMBER
010   10OCTOBER
011   11NOVEMBER
012   12DECEMBER
```

How to Modify a Table

The following is a demonstration of maintaining a table using the :LIST of MONTHTAB.

```
1       // EXEC QWDMAINT,SIZE=AUTO                                  <<--- VSE JCL

1       //STEP1 EXEC PGM=QWDMAINT                                  <<--- MVS JCL
        //SYSPRINT DD SYSOUT=A                                     <<--- MVS JCL
        //QWDICT   DD DSN=USER.QWDICT,DISP=SHR                     <<--- MVS JCL
        //SYSIN    DD *                                            <<--- MVS JCL
2       :UPDATE MONTHTAB

3       MAXENT=20

4       :MODIFY TABLE

5       :DELETE SEQ=5,7

6       :CHANGE SEQ=11,12

7       15TOMORROW

8       :INSERT SEQ=0

9       21MONDAY

10      22TUESDAY

11      23WEDNESDAY

12      /*
```

Command explanations:

1 Invoke the maintenance program.

2 Identify which member is to be updated.

3 Change the maximum number of table entries.

4 Indicate that the table records are to be modified.

5 Delete sequence numbers 5 through 7.

6 Replace sequence numbers 11 and 12 with the following statements.

7 This is the only statement inserted in place of sequence numbers 11 and 12.

8 Insert new table records at the beginning (in front of) any existing records.

9-11 These three table records are inserted.

12 End of program input.

How to Use QWDMAINT in an Inline Dictionary

When using an inline dictionary in VISION:Forms, you can take advantage of the QWDMAINT syntax to specify overrides, or create new members for the duration of the VISION:Forms program. In this example, the member ACTG is being defined so that it can be referenced in the VISION:Forms program.

```
1    // EXEC QWRITE,SIZE=100K                                  <<— VSE JCL
1    //STEP1 EXEC QWRUN,FILE='MM1.ACCOUNT.MASTER'             <<— MVS JCL
2    START-DICTIONARY
3    :CREATE ACTG
4    DEVICE=KSDS .                                          <<—VSE only
5    UNIT=027                                               <<—VSE only
6    BLKL=2048
7    RECL=512
8    :BEGIN JCL=ALL,SEL=ALL
9    // DLBL INF,'ACT.MSTR',,VSAM,CAT=VSAMCAT               <<—VSE only
10   // EXTENT ,FBA001                                      <<—VSE only
11   :BEGIN RECORD
12   FIELD NAME=ACCT-NAME,START=8,LEN=30,HDR='ACCOUNT NAME'
13   FIELD NAME=ACCT-AMT,POS=(237-241-P),PRTDEC=2,HDR='AMOUNT-
     DUE'
14   END-DICTIONARY
15   FILENAME ACTG
16   SELECT ACCT-AMT 1000.00
17   SORT ACCT-NAME
18   PRINT ACCT-NAME ACCT-AMT
19   TITLE 'BAD RECEIVABLES OVER $1000'
20   /*
```

Command explanations:

1 Invoke VISION:Forms.

2 Indicate start of inline dictionary.

3 :CREATE a new member called ACTG.

4-5 Required VSE Group 2 statements for device type and unit number.

6-7 Required Group 2 statements for record and block size.

8 Start of VSE JCL to be included in resultant VISION:Report program that identifies and locates the file that is to be retrieved.

9-10 VSE job control for a VSAM file to be used as INF in the VISION:Report program.

11 Beginning of the record description.

12 Define the ACCT-NAME field.

13 Define the ACCT-AMT field.

14 Indicate end of inline dictionary.

15-19 Typical VISION:Forms statements to process the file.

20 End of VISION:Forms statements.

Error Messages

Message	Description
DM10E THE WORD xxxxxxxx IS AN UNKNOWN COMMAND	The word displayed was expected to be a system command, but it is not. Correct statement(s) and resubmit.
DM11I UNABLE TO IDENTIFY THE CURRENT STATEMENT	The command just read cannot be determined correctly. A syntax error exists. The statement is rejected and processing continues.
DM13I INVALID CONTINUATION DETECTED	Self-explanatory. The bad continuation is ignored and processing continues.
DM14E UNEXPECTED END OF FILE	Additional input was expected, but end of file was detected. Determine what is missing and resubmit the job.
DM15E SYNTAX ERROR DETECTED, CHECK CODING RULES	The command just processed does not conform to the coding rules. Check for correctness and resubmit the job.
DM17E THE WORD xxxxxx IS AN UNKNOWN PARAMETER	The keyword displayed is not a valid keyword for this command. Check the coding rules for the command and resubmit the job.
DM18E VALUE FOR xxxxxx EXCEEDS MAX LENGTH	The value specified for the keyword displayed is too long for this parameter. Check coding rules and resubmit the job.

Message	Description	
DM19E VALUE FOR xxxxxx MUST BE NUMERIC	The value specified for the keyword displayed must be numeric. Check the coding rules and resubmit the job.	
DM1AI FLUSHING TO NEXT SYSTEM COMMAND	An error has occurred during the processing of a member that cannot continue. Therefore, the job stream is flushed to the next :ACCESS/:UPDATE/:CREATE command.	
DM1BE UNABLE TO LOAD PHASE xxxxxxxx RETURN CODE=X"NN"	The system attempted to load the named phase. However, the load was unsuccessful. The return code has the following meanings:	
	X'04'	The size of the partition's GETVIS area is 0K
	X'12'	No more storage available in the GETVIS area
	X'16'	The ANCHOR TABLE is full (CDLOAD directory)
	X'20'	The phase does not exist in the Core Image Library
	X'32'	Hardware failure
DM1CE UNEXPECTED COMMAND DETECTED, "RESTORE" REQUIRED	During restore mode, only restore commands are accepted. All members selected for restore are attempted, however, the job stream is flushed to end of job.	
DM1DI OPTION MUST BE FIRST STATEMENT PROCESSED	Self-explanatory. Statement is ignored and processing continues.	
DM1EE COMMAND NOT VALID FOR QWRITE IN-LINE PROCESSING	Self-explanatory. Statement is ignored and processing continues.	
DM1FI ERRORS DETECTED, FUNCTION ABORTED	Unrecoverable error(s) have occurred. The requested function is terminated.	
DM21E :LOAD COMMAND IS CODED INCORRECTLY	There exists a coding error for an :ACCESS/:UPDATE/:CREATE or :PURGE statement. Correct and resubmit the job.	
DM22E MEMBER xxxxxxxx IS NOT IN DICTIONARY	The requested member was not found in the dictionary. Correct and resubmit.	
DM23E PARTITION GETVIS ERROR X"NN" OCCURRED	The acquisition of storage failed. The error codes are the same as message DM1BE above. Job is terminated.	

Message	Description
DM24E MEMBER xxxxxxxx ALREADY EXISTS IN DICTIONARY	A request for :CREATE of a named member that is already in the dictionary has been detected. Correct and resubmit.
DM25E VSAM READ ERROR OCCURRED RC="NN" EC="NN"	While reading the dictionary to load the virtual table, a VSAM error occurred. Refer to VSAM error messages.
DM26E NO MORE ROOM FOR TABLE, INCREASE TSIZE	There is not enough room in the virtual table to hold all the entries specified. Use the :OPTION statement to enlarge the table.
DM27I INVALID OPTION SYNTAX OR PARAMETER DETECTED	The :OPTION command is incorrect. Check coding rules and resubmit the job.
DM28I MEMBER HAS BEEN DELETED	This is confirmation that the requested :PURGE has been completed.
DM30I LISTING COMPLETE -- NO ERRORS FOUND	The member has been listed, and no logical errors were detected.
DM31E :LIST COMMAND IS CODED INCORRECTLY	Self-explanatory. Check coding rules. Correct and resubmit the job.
DM32E LOGIC ERROR HAS OCCURRED, CONSULT REFERENCE MANUAL	This :LIST logic error should not happen.
DM33I :LIST COMMAND CAN ONLY BE USED WITH ":ACCESS"	The :LIST command is valid only in :ACCESS mode. Remove from the job stream.
DM34I LISTING COMPLETE -- ERROR(S) HAVE BEEN FOUND	The member has been listed, however logical errors were found. Correct and resubmit the job.
DM35I MEDIA ERRORS DETECTED -- SEE ABOVE	The member listed has media errors. Check for messages DM3AE and DM3BE. Correct and resubmit the job.
DM36E NO FILE CHARACTERISTICS WERE SPECIFIED	No Group 2 commands that define the file characteristics were processed. Correct and resubmit the job.
DM37E VALUE FOR xxxxxx IS INVALID	During the listing of a member, the value that is controlled by the Group 2 command displayed is incorrect. Check documentation, correct, and resubmit the job.
DM38E NO TABLE CHARACTERISTICS WERE SPECIFIED	No Group 2 commands that define the table characteristics were processed. Correct and resubmit the job.
DM39I DICTIONARY HAS BEEN UPDATED	No logic errors were detected and :UPDATE was specified. The dictionary is updated to reflect the changes supplied.

Message	Description
DM3AE MEDIA FOR xxxxxx IS INVALID	The SELECT or REPORT MEDIA code is incorrect. Check documentation, correct, and resubmit the job.
DM3BE SYSNR FOR xxxxxx IS INVALID	The SELECT or REPORT programmer logical unit code is invalid. Check the documentation, correct, and resubmit the job.
DM40I CONVERSION WAS SUCCESSFULLY COMPLETED	The :PUNCH translation occurred without error.
DM41E :PUNCH COMMAND IS CODED INCORRECTLY	Self-explanatory. Check coding rules. Refer to the documentation, correct, and resubmit the job.
DM42E LOGIC ERROR HAS OCCURRED, CONSULT REFERENCE MANUAL	This message should not occur. Should the :PUNCH processor detect an invalid logic problem, this message is issued. Contact Computer Associates for assistance.
DM43I :PUNCH COMMAND CAN ONLY BE USED WITH ":ACCESS"	Self-explanatory. Check coding rules. Remove and resubmit the job.
DM51E LOGIC ERROR HAS OCCURRED, REFER TO MANUAL	This message should not occur. Contact Computer Associates for assistance.
DM52E COMMAND NOT VALID FOR MEMBER TYPE BEING UPDATED	A mismatch between the type of member and the requested update. For example, a file cannot process a MAXENT Group 2 command. Check documentation, correct, and resubmit the job.
DM53E THE ABOVE COMMAND IS CODED INCORRECTLY	Self-explanatory. Check documentation for coding rules. Correct and resubmit the job.
DM54E JCL PARTITION ID IS NOT ALL,BG,FB,FA,F9-F1	Self-explanatory. The JCL= parameter must be ALL, BG, FB, FA, OR F1 through F9. Correct and resubmit the job.
DM55E USAGE IS NOT SEL,SOI,SOW,SOO	An invalid value for the USAGE= parameter was detected. Valid values are SEL, SOI, SOW, and SOO. Check documentation, correct, and resubmit the job.
DM56E MEDIA PARTITION ID IS NOT ALL,BG,FB,FA,F9 through F1	The partition id for the MEDIA= parameter is incorrect. See also message DM54E.
DM57E THE ABOVE COMMAND IS OUT OF SEQUENCE	Either a Group 2 command was detected while in :BEGIN or :MODIFY mode, or a SELECT/REPORT/FIELD command or JCL statements where found without a :BEGIN or :MODIFY command. Check coding rules, correct, and resubmit the job.

Message	Description
DM58E NO MORE ROOM IN TABLE, USE :OPTION TSIZE=	Out of space in the partition. Supply an :OPTION statement to enlarge the allocation and resubmit the job.
DM59E ":INSERT/:CHANGE/:DELETE" COMMAND IS CODED WRONG	Self-explanatory. Check coding rules. Correct and resubmit the job.
DM5AE LOGIC ERROR DETECTED, CONSULT MANUAL	This message should not occur. Contact Computer Associates for assistance.
DM5CE FIELD COMMAND WAS CODED INCORRECTLY	An invalid keyword was detected on a field statement. Correct and resubmit the job.
DM5DE VALUE FOR xxxxxx IS INVALID	The value specified for the keyword displayed is invalid. Correct and resubmit the job.
DM5EE REQUIRED PARAMETER xxxxxx IS MISSING	No valid value was specified for the above-mentioned keyword. Check the documentation, correct, and resubmit the job.
DM5FE DUPLICATE FIELD NAME OF xxxxxxxxxxxxx	A field to be added to the member is already on file. Field is rejected, processing continues. Correct the job stream.
DM5GI ENTRY xxxxxxxxxxxxx NOT IN MEMBER	The field name specified on a :DELETE ENTRY= was not found. Processing continues. Correct and resubmit if necessary.
DM5HI SEQUENCE NUMBER FROM/TO IS MISSING OR INVALID	A command using sequence numbers was found to have an error. Check documentation, correct, and resubmit the job.
DM5JE NO FIELD NAME WAS GIVEN FOR MODIFY	During :MODIFY RECORD mode, a FIELD command must have at least the NAME= keyword specified. Correct and resubmit the job.
DM70I DICTIONARY BACKUP COMPLETED	Self-explanatory.
DM71E BACKUP POINT FAILED, RC=X"NN" EC=X"NN"	A VSAM error during POINT processing occurred. Check the VSAM error message manual for details.
DM72E BACKUP READ FAILED, RC=X"NN" EC=X"NN"	A VSAM error occurred during reading of the dictionary during backup. Check the VSAM error message manual for details.
DM73I MEMBER xxxxxxxx HAS BEEN DUMPED	The indicated member has been saved on the backup tape.
DM75I BACKUP ENCOUNTERED VSAM ERROR--FUNCTION ABORTED	An irrecoverable VSAM error occurred during backup. The job is canceled. Determine cause and resubmit the job.
DM76I MEMBER xxxxxxxx WAS NOT FOUND ON BACKUP TAPE	Self-explanatory. Processing continues.

Message	Description
DM77I MEMBER xxxxxxxx HAS BEEN RESTORED	Self-explanatory. Processing continues.
DM78E VSAM ERROR RESTORING xxxxxxxx RC=X"NN" EC=X"NN"	A VSAM error occurred during output operation to the dictionary. Check the VSAM error message manual for details.
DM79I MAX RESTORE COMMANDS HAS BEEN REACHED	There exists a maximum of 64 individual RESTORE commands that can be processed during one execution of the system.
DM7AE RESTORE COMMAND IS INVALID	A coding error exists for the :RESTORE command. Check documentation, correct, and resubmit the job.
DM7BE RESTORE ALL MUST BE FIRST COMMAND	Self-explanatory. Correct and resubmit the job.
DM7CI RESTORE OF DICTIONARY WAS SUCCESSFUL	Self-explanatory. Job is ended.
DM7DE BACKUP/RESTORE COMMANDS MUST ALWAYS BE FIRST	Self-explanatory. Check documentation, correct, and resubmit the job.
DM7EI MEMBER xxxxxxxx WAS ON FILE, NOW DELETED	Self-explanatory. Member was replaced.
DM7FE MEMBER xxxxxxxx VSAM ZZ ERROR, RC="NN" EC="NN.	A VSAM error occurred there during processing of the named member. The code ZZ indicates what mode of operation the system was in. Check the VSAM error message manual for details.
DM98E LOGIC ERROR 1 OCCURRED = 'NN'	This message should not occur. Contact Computer Associates for assistance.
DM99E UNABLE TO LOAD xxxx, RC=X'NN'	Required phase to be loaded failed. Check return code per message DM1BE. If VSAM is displayed, ensure QUIKVSAM has been installed from the VISION:Report optional material.

Appendix

A

Examples

To give users an idea of the kind of reports VISION:Forms can produce, we have included some examples of VISION:Forms programs and the reports they generate. These program examples are only a sampling of the power and simplicity of VISION:Forms.

An example of a VISION:Forms program is shown first and the output report next. The Payroll VISION:Forms Dictionary in this appendix (PAYROLL file) used in all the examples provided. It is similar to the ones your data processing department prepares. You can refer to the dictionary for information about the examples and reports.

Simple Listing

Take the PAYROLL file and list each employee name, department number, and plant number.

```
FILENAME PAYROLL
PRINT PLANT-NR DEPT-NUMBER NAME
TITLE 'EMPLOYEES'
```

PAYROLL is the name given to the dictionary records that make up the PAYROLL file. The data names DEPT-NUMBER, NAME, and PLANT-NR in the PRINT statement are taken from the VISION:Forms dictionary.

```
12/15/93      EMPLOYEES   PAGE   1

     PLANT      DEPT
      NR       NUMBER     ......NAME

       12        005       MERCURY, HARRY
       12        005       WINTERGARTEN, L.R
       12        005       LADRIGANSKI, SUE
       12        005       ATWATER, SCOTT
       12        005       CLEARY, TOM
       12        005       RUNNINGTREE, TOM
       12        005       LEANINGHORSE, C.E
       12        005       LUBINPINSKI, CLY
       12        005       RODKOWSKI, GENE
       12        005       LONNYSTAR, RACHEL
       12        005       ZONK, HIERONYMOUS
       12        005       JONES, LYLA
       12        010       CLEGHORN, DELLA
       12        010       CLEMENS, GARY
       12        010       CLEVELAND, GROVER
       13        010       COCER, ONIES
       13        005       EVERS, HANK
       13        005       FAIR, MAXINE
       13        005       KIRBY, THOMAS F
       13        005       KUEHN, JOHN
       13        005       LADD, IDA
       13        005       LAFARY, ALFRED
       13        005       LANDERS, CAROL
       13        005       LANDERS, MICHAEL
       13        005       LAROCHEELLE, KISA
       13        005       LAWSON, MOLER
```

Summary Report

For each department, total last year's gross wages. Print only the totals for each department.

```
FILENAME PAYROLL
BREAK DEPT-NUMBER
SORT DEPT-NUMBER
TOTAL PREV-YTD-GROSS GROUPCOUNT
TITLE '1992 DEPARTMENT SALARY SUMMARY'
```

For summary reporting, do not put in a PRINT statement. Only the totaled previous gross of each department prints on the report. Headings are taken from the data names in the BREAK and TOTAL statements. GROUPCOUNT in the TOTAL statement indicates the total number of records read within each department.

```
12/15/93 1992 DEPARTMENT SALARY SUMMARY PAGE 1

                   PREV
 DEPT               YTD
NUMBER    ........GROSS       GROUPCOUNT
 005          167,810.94               22

 010           35,947.08                4


              203,758.02               26
```

Using TOTAL, BREAK, and PRINT Statements

Make a listing of each employee's gross wages for 1992. Include name, department number, and last year's wages. Then show a total for each department.

```
FILENAME PAYROLL
BREAK DEPT-NUMBER
TOTAL PREV-YTD-GROSS
TITLE '1992 DEPARTMENTAL SALARY'
SORT DEPT-NUMBER
PRINT DEPT-NUMBER NAME PREV-YTD-GROSS
```

The TOTAL and BREAK statements are used together to indicate that, whenever the department number changes, the accumulated total of gross wages is printed for the previous department.

```
12/15/93     1992 DEPARTMENTAL SALARY    PAGE   1

                                                PREV
 DEPT                                            YTD
NUMBER      ......NAME......          ........GROSS

 005        WINTERGARTEN,L.R                6,520.27
 005        MERCURY,HARRY                   6,495.27
 005        RUNNINGTREE,TOM                 6,320.27
 005        CLEARY,TOM                      6,295.27
 005        ATWATER,SCOTT                   6,470.27
 005        LADRIGANSKI,SUE                 6,545.27
 005        LAFARY,ALFRED                   9,074.27
 005        LADD,IDA                        9,049.27
 005        KUEHN,JOHN                      9,024.27
 005        KIRBY,THOMAS F                  8,999.27
 005        FAIR,MAXINE                     8,974.27
 005        EVERS,HANK                      8,949.27
 005        JONES,LYLA                      6,570.27
 005        ZONK,HIERONYMOUS                6,445.27
 005        LONNYSTAR,RACHEL                6,420.27
 005        RODKOWSKI,GENE                  6,395.27
 005        LUBINPINSKI,CLY                 6,370.27
 005        LEANINGHORSE,C.E                6,345.27
 005        LAWSON,MOLER                    9,174.27
 005        LAROCHEELLE,KISA                9,149.27
 005        LANDERS,MICHAEL                 9,124.27
 005        LANDERS,CAROL                   9,099.27
GROUPCOUNT IS      22      FOLLOWING TOTALS ARE FOR DEPT-NUMBER 005
                                          167,810.94

 010        CLEMENS,GARY                    8,974.27
 010        CLEGNORN,DELLA                  8,949.27
 010        COCER,ONIES                     9,024.27
 010        CLEVELAND,GROVER                8,999.27
GROUPCOUNT IS       4      FOLLOWING TOTALS ARE FOR DEPT-NUMBER 010
                                           35,947.08

GROUPCOUNT IS      26      FOLLOWING ARE FINAL TOTALS

                                          203,758.02
```

Report Format with TITLE and PAGEDEPTH Verbs

List all employees, calculating each person's weekly wages. Sort the report into wage within department sequence.

```
FILENAME PAYROLL
CALC WAGE-PER-WK DEC2 EQUALS HOURLY-RATE TIMES HOURS-WORKED
TITLE 'WAGES FOR THIS WEEK'
SORT WAGE-PER-WK DEPT-NUMBER
PRINT DEPT-NUMBER NAME HOURLY-RATE WAGE-PER-WK
PAGEDEPTH 75
```

WAGE-PER-WK is the name given to the result of multiplying hourly rate and hours worked for the week.

The list is arranged in department number sequence and, within departments, it is sorted on wages earned.

This report has 75 lines per page, instead of the usual 54.

```
12/15/93      WAGES FOR THIS WEEK  PAGE   1
                                        WAGE
 DEPT                           HOURLY    PER
NUMBER    ......NAME......     .RATE    ....WK

005       LAWSON,MOLER         4.157     166.28
005       LAROCHEELLE,KISA     4 162     166.48
005       LANDERS,MICHAEL      4.167     166.68
005       LANDERS,CAROL        4.172     166.88
005       LAFARY,ALFRED        4.177     167.08
005       LADD,IDA             4.182     167 28
005       KUEHN,JOHN           4 187     167 48
005       KIRBY,THOMAS F       4.192     167.68
005       FAIR,MAXINE          4.197     167.88
005       EVERS,HANK           4.202     168.08
005       JONES,LYLA           4.427     177 08
005       LADRIGANSKI,SUE      4.432     177.28
005       WINTERGARTEN,L.R     4 437     177.48
005       MERCURY,HARRY        4.442     177.68
005       ATWATER,SCOTT        4.447     177.88
005       ZONK,HIERONYMOUS     4.452     178.08
005       LONNYSTAR,RACHEL     4.457     178.28
005       RODKOWSKI,GENE       4.462     178.48
005       LUBINPINSKI,CLY      4.467     178.68
005       LEANINGHORSE,C.E     4.472     178.88
005       RUNNINGTREE,TOM      4.477     179.08
005       CLEARY,TOM           4.482     179 28
010       COCER,ONIES          4 302     172.08
010       CLEVELAND,GROVER     4.307     172.28
010       CLEMENS,GARY         4.312     172.48
010       CLEGHORN,DELLA       4.317     172.68
```

Using CALC and SELECT Statements

A report is needed of those employees having a 3 in their employee number for special tax considerations. Calculate the city tax based on weekly wages.

```
FILENAME PAYROLL
SELECT EMPLOYEE-NUM WHEN INCLUDING '3'
CALC CITY-TAX DEC2 RND = HOURLY-RATE * HOURS-WORKED * .0175
PRINT EMPLOYEE-NR NAME CITY-TAX HOURLY-RATE HOURS-WORKED
LIMIT SELECTION INPUT 100
SORT NAME
```

WHEN is scanning for a 3 in the four-character employee number.

Multiplying the hourly rate times the hours worked times .0175 gives the city tax amount.

Only 100 employee records are tested, and the selected records are printed in ascending alphabetical order by NAME.

```
12/15/93                                                    PAGE     1

  EMPLOYEE                          CITY       HOURLY         HOURS
    .NR.     ......NAME......     ....TAX       .RATE        WORKED

  0340       CLEGHORN,DELLA         3.02       4.317         40.00
  0350       CLEMENS,GARY           3.02       4.312         40.00
  0360       CLEVELAND,GROVER       3.01       4.307         40.00
  0370       COCER,OMIES            3.01       4.302         40.00
  0630       LANDERS,CAROL          2.92       4.172         40.00
  0030       LEANINGHORSE,C.E       3.13       4.472         40.00
```

Using NEWTEST, SELECT, and CALC Statements

List those employees from plants 12 and 13 who are over 40 years old, or whose skill codes are greater than 120000. Calculate last year's monthly wages for those employees selected.

```
FILENAME PAYROLL
EQUATE BIRTH-DATE INF26-31 D
EQUATE YR-BIRTH INF26-27
SELECT PLANT-NR EQ '12' '13'
SELECT YR-BIRTH LT '53'
CALC WAGE-MONTH DEC2 = PREV-YTD-GROSS / 12
NEWTEST
SELECT PLANT-NR EQ '12' '13'
SELECT SKILL-CODE GT '120000'
CALC WAGE-MONTH DEC2 = PREV-YTD-GROSS / 12
PRINT PLANT-NR NAME(EMPLOYEE) WAGE-MONTH SKILL-CODE BIRTH-DATE
PAGEWIDTH 80
SPACE DOUBLE
NEWPAGE PLANT-NR
TITLE '1993 MONTHLY EARNINGS FOR SELECTED EMPLOYEES'
```

In this example, the employee record is first selected on the basis of plant (12 or 13). If the plant is found to be 12 or 13, the record is searched for a birth year less than 1953. If neither is found to be true, the record is searched again, this time for a plant of 12 or 13 and a skill code of over 120000. If an employee of plant 12 or 13 was born prior to 1953 or has a skill code of over 120000, the record is selected. If neither is true, the record is rejected.

When a record is selected, the monthly wages are calculated.

The sample report is double-spaced.

Refer to the Payroll VISION:Forms Dictionary in this appendix. The data name BIRTH-DATE is in each employee's record as yymmdd. To refer to the year only, the second EQUATE statement is used. The first EQUATE includes the print specifications (D) for a date; yymmdd in the record becomes yy/mm/dd in the report.

```
01/15/94  1993 MONTHLY EARNINGS FOR SELECTED EMPLOYEES PAGE    1

PLANT     ....EMPLOYEE....        WAGE     SKILL      BIRTH
 NR                           ....MONTH     CODE.     .DATE...

 12       MERCURY,HARRY          541.27     105025    44/03/10

 12       WINTERGARTEN,L.R       543.35     115313    44/04/10

 12       LADRIGANSKI,SUE        545.43     115009    44/05/10

 12       ATWATER,SCOTT          539.18     113300    44/02/10

 12       CLEARY,TOM             524.60     101256    43/07/10

 12       RUNNINGTREE,TOM        526.68     114475    43/08/10

 12       LEANINGHORSE,C.E       528.77     105325    43/09/10

 13       LUBINPINSKI,CLY        530.85     104915    43/10/10

 12       RODKOWSKI,GENE         532.93     102325    43/11/10

 12       LONNYSTAR,RACHEL       535.02     115445    43/12/10

 12       ZONK,HIERONYMOUS       537.10     120320    44/01/10

 13       JONES,LYLA             547.52     121134    44/06/10

 12       CLEGHORN,DELLA         745.77     112375    43/07/10

 13       CLEMENS,GARY           747.85     105325    43/08/10

 12       CLEVELAND,GROVER       749.93     125005    43/09/10

 12       COCER,ONIES            752.02     148185    43/10/10


01/15/94 1993 MONTHLY EARNINGS FOR SELECTED EMPLOYEES PAGE    2

PLANT
 NR       ....EMPLOYEE....        WAGE    SKILL      BIRTH
                               ....MONTH    CODE.     .DATE...

 12       EVERS,HANK            745.77     125775    53/07/10

 13       FAIR,MAXINE           747.85     128005    53/08/10

 13       KIRBY,THOMAS F        749 93     126105    53/09/10

 13       KUEHN,JOHN            752.02     123227    53/10/10

 13       LADD,IDA              754 10     123775    53/11/10

 12       LAFARY,ALFRED         756.18     124885    53/12/10

 12       LANDERS,CAROL         758.27     125815    54/01/10

 13       LANDERS,MICHAEL       760.35     125311    54/02/10

 13       LAROCHEELLE,KISA      762.43     124555    54/03/10

 13       LAWSON,MOLER          764.52     125910    54/04/10
```

Using TOTAL and PRINT HEADLINE Statements

Calculate year-to-date wages for the employees who have been with the company more than 25 years.

```
FILENAME PAYROLL
EQUATE YEAR-DATE-EMP INF32-33
EQUATE DATE-EMPLOYED INF32-37 D
SELECT YEAR-DATE-EMP LT '68'
CALC WEEK-WAGE DEC2 RND = HOURLY-RATE * HOURS-WORKED
CALC YTD-WAGE DEC2 RND = WEEK-WAGE * 52
PRINT HEADLINE NAME YTD-WAGE(WAGES,YTD,WEEKLY) DATE-EMPLOYED
PRINT WEEK-WAGE UNDER YTD-WAGE
PAGEWIDTH 80
TOTAL YTD-WAGE
TITLE 'WEEKLY WAGES FOR EMPLOYEES HIRED BEFORE 1968'
```

Every employee who was hired before 1968 has year-to-date wages calculated. The total wages paid are listed on the last line of the report.

Weekly wages are printed on the report under yearly wages. The column header, "WAGES, YTD, WEEKLY", is taken from the PRINT HEADLINE statement.

```
12/15/93 WEEKLY WAGES FOR EMPLOYEES HIRED BEFORE 1968 PAGE 1

                       WAGES
                         YTD       DATE
......NAME......  ....WEEKLY     EMPLOYED

MERCURY,HARRY        9239.36     53-04-11
                      177.68

WINTERGARTEN,L.R     9228.96     53-05-11
                      177.48

LADRIGANSKI,SUE      9218.56     53-06-11
                      177.28

ATWATER,SCOTT        9249.76     53-03-11
                      177.88

CLEARY,TOM           9322.56     52-08-11
                      179.28

RUNNINGTREE,TOM      9312.16     52-09-11
                      179.08

LEANINGHORSE,C.E     9301.76     52-10-11
                      178.88

LUBINPINSKI,CLY      9291.36     52-11-11
                      178.68

RODKOWSKI,GENE       9280.96     52-12-11
                      178.48

LONNYSTAR,RACHEL     9270.56     52-01-11
                      178.28

ZONK,HIERONYMOUS     9260.16     53-02-11
                      178.08

JONES,LYLA           9208.16     53-07-11
                      177.08

12/15/93 WEEKLY WAGES FOR EMPLOYEES HIRED BEFORE 1968

CLEGHORN,DELLA       8979.36     52-08-11
                      172.68

CLEMENS,GARY         8968.96     52-09-11
                      172.48

CLEVELAND,GROVER     8958.56     52-10-11
                      172.28

COCER,ONIES          8948.16     52-11-11
                      172.08


                       WAGES
                         YTD       DATE
......NAME......  .....WEEKLY    EMPLOYED


                   233972.96
```

Using SORT Verb

You need a list of the employees earning over $6.00 an hour.

```
FILENAME PAYROLL
SELECT HOURLY-RATE GREATER-THAN 6.000
SORT NAME
PRINT DEPT-NUMBER SPACE10 NAME SPACE10 HOURLY-RATE
```

In this example, VISION:Forms examines an employee's hourly rate to see if it is over 6.00. If it is, the record passes the criteria and is printed as specified in the PRINT statement. If the employee earns less that 6.00 an hour, the record is passed over.

SORT causes records to be sorted in ascending order by name. Notice that there are 10 spaces between each printed field in the report.

```
12/15/93                                PAGE  1

DEPT
NUMBER       ......NAME......            HOURLY
                                          .RATE

 005           ATWATER,SCOTT              6.447
 005           CLEARY,TOM                 6.482
 010           CLEGHORN,DELLA             6.317
 010           CLEMENS,GARY               6.312
 010           CLEVELAND,GROVER           6.307
 010           COCER,ONIES                6.302
 005           EVERS,HANK                 6.202
 005           FAIR,MAXINE                6.197
 005           JONES,LYLA                 6.427
 005           KIRBY,THOMAS F             6.192
 005           KUEHN'JONN                 6.187
 005           LADD,IDA                   6.182
 005           LADRIGANSKI,SUE            6.432
 005           LAFARY,ALFRED              6.177
 005           LANDERS,CAROL              6.172
 005           LANDERS,MICHAEL            6.167
 005           LAROCHEELLE,KISA           6.162
 005           LAWSON,MOLER               6.157
 005           LEANINGHORSE,C.E           6.472
 005           LONNYSTAR,RACHEL           6.457
 005           LUBINPINSKI,CLY            6.457
 005           MERCURY,HARRY              6.442
 005           RODKOWSKI,GENE             6.462
 005           RUNNINGTREE,TOM            6.477
 005           WINTERGARTEN,L.R           6.437
 005           ZONK,HIERONYMOUS           6.452
```

Payroll VISION:Forms Dictionary

DATA NAME	DATA LENGTH	PACKED	CALC DEC	PRINT LENGTH	PRINT DEC	PRINT EDIT CODE	OVERRIDE HEADING
BIRTH-DATE	6			8		D	
BIRTH-YEAR	2			2			YEAR-OF-BIRTH
CARD-CODE	2			2			
CITY-TAX-RATE	4		3	6	1		
DATE-EMPLOYED	6			8		D	
DEPT-NUMBER	3			3			
DIVISION	3			3			
EDUC-LVL	2			2			
EMPLOYEE-ADDR2	30			30			CITY-ZIP
EMPLOYEE-ADDR1	30			30			STREET-ADDRESS
EMPLOYEE-NAME	26			26			
EMPLOYEE-NR	4			4			
FED-TAX-RATE	4	YES	3	6	1		FEDERAL-TAX-RATE
HOURLY-RATE	4	YES	3	8	3		
HOURS-WORKED	4		2	8	2		
MON-DED-SAVE	4		2	7	2	C	
MON-DED-TAX	4		3	8	3		
NAME	26			26			
OVER-HOURS	4		2	7	2		OVERTIME-HOURS

Appendix

B Error Diagnostics

This appendix lists and explains the diagnostic messages that appear with the VISION:Forms statements if an error occurs when running a VISION:Forms program. Some error messages are self-explanatory and, as such, are not explained any further.

Error Messages

Code	Message and Description
001	ERR001. THE WORD xxxxxxxxxx WHICH IS WORD nn IN THE PREVIOUS STATEMENT IS INVALID FOR THAT STATEMENT TYPE. Only the following data types or keywords can be coded as Word nn. The word shown is not one of the data types or keywords valid for that word in the context of the verb being processed. This is often caused by a coding error such as EQUAX-TO when EQUAL-TO was intended. If the error is not obvious, review the word coding rules for that verb.
002	ERR002. THE WORD xxxxxxxxxx IS NOT A VALID QWRITE WORD. All the words for the preceding verb have been processed successfully, and the implication is that the next VISION:Forms verb should have been coded at this point. It is most likely that the word shown was incorrectly coded as part of that preceding verb. Review the word coding rules for the preceding verb.
003	ERR003. DATANAME xxxxxxxxxxxxxx FORMAT IS NOT THE SAME AS A PREVIOUS ATTACH FOR THE SAME DATANAME.
004	ERR004. THE VERB SPACE MUST BE FOLLOWED BY 'DOUBLE' OR 'TRIPLE'.
005	ERR005. THE WORD xxxxxxxxxxxxxx CANNOT BE USED SINCE IT IS A LONG ALPHA/NUMERIC ATTACH.
006	ERR006. TITLE TOO LONG FOR LINE WIDTH. MAX OF nnn CHAR.

Code	Message and Description
008	ERR008. TOO MANY FIELDS ARE CALLED FOR IN COPY OR EQUATE.
009	ERR009. EQUATE DATANAME ALREADY IN USE. IGNORED. An EQUATE data name must not be present in another EQUATE.
010	ERR010. THE WORD xxxxxxxxxxxx MUST BE THE ONLY WORD IN THIS ONBREAK STATEMENT.
011	ERR011. TOO MANY ONBREAK STATEMENTS.
012	ERR012. THE WORD xxxxxxxxxxxxx MAY ONLY BE USED IN ONE ONBREAK STATEMENT.
014	ERR014. THE WORD xxxxxxxxxxxxx MUST BE THE RESULT OF A CALC STMT WITHIN THIS ONBREAK ROUTINE.
030	ERR030. STMT CONTINUATION INDICATED BUT NOT PRESENT.
	The last line (or statement) of VISION:Forms coding must not indicate continuation to another line. Position 72 must be blank.
031	ERR031. RUN STOPPED AS END OF LITERAL NOT INDICATED BY QUOTE OR CONTINUATION MARK. To continue scanning for the ending quotation mark might result in the loss of the following user statements. If a literal cannot be completed in one line, code a non-blank character in the continuation column 72 and continue the literal in column 1 of the next line. Continuation has not been indicated in this case, and to continue scanning for the ending quotation mark would be pointless.
032	ERR032. FILENAME TOO LONG OR NOT IN DICTIONARY.
	The file name following the FILENAME verb is not present in the dictionary or is longer than eight characters if MVS or longer than seven characters if VSE.
033	ERR033. PREVIOUS VISION:Forms STATEMENT HAS ONE OR MORE REQUIRED WORDS MISSING.
034	ERR034. THE KEYWORD xxxxx MAY ONLY BE USED TO INITIALIZE A CALC'ED RESULT.
035	ERR035. WORDS 5 & 6 OPERANDS CANNOT BOTH BE CONSTANTS.
036	ERR036. DATANAME xxxxxxxxxxxxx WHICH IS TO BE CALC'ED MUST HAVE A UNIQUE NAME. The data name, to be a result, must have a unique name.

Code	Message and Description
037	ERR037. STACK SPACE EXHAUSTED. Advise your computer department to run this VISION:Forms in a larger partition or region. Certain information on each data name cited is retained in the computer, and you referenced enough data names to exhaust the available computer space.
038	ERR038. COND BEING TEST REQ. COMPARAND. If Word 5 is EQUAL-TO, GREATER-THAN, or LESS-THAN, you must provide a compared value in Word 6.
039	ERR039. A NEGATIVE, 'NOT' SCAN IS INVALID. A WHEN INCLUDING type SELECT can be used to determine the presence of the value. The absence of the value (NOT in Word 4) cannot be determined through this verb.
040	ERR040. TOO MANY PRINT FLDS SPEC. MAX OF 24 PER LINE.
041	ERR041. PRINT SPACING VALUE NOT NUMERIC. The expression SPACE (in the PRINT verb) is reserved to indicate a certain number of blank spaces to appear in the report line, and, when used, must end in a one-or two-digit number, i.e., SPACE10.
042	ERR042. OVERRIDE COL HDR TOO LONG, MAX LNTH OF 57.
043	ERR 043 DATANAME xxxxxxxxxx NOT DEFINED IN DICTIONARY OR EQUATE STMT.
044	ERR044. CALC RESULT DATANAME ALREADY IN USE. THIS CALC STMT WILL BE IGNORED.
045	ERR045. 2ND OPERAND DEC SPEC MUST BE 0-9. If the optional Word 2 value DECn is coded, n must be in the range 0-9.
046	ERR046. DATANAME xxxxxxxxxx NOT DEFINED AS EQUATE OR DICTIONARY ENTRY.
047	ERR047. DATANAME PRINT EDIT CODE IS INVALID. CODE MUST BE A S, C, E, N, OR VALID USER CODE. See **EQUATE** in the chapter “Writing Verbs.”
048	ERR048. DISKERR OR WLERR ON DICTIONARY ISAM FILE. Advise your computer department of this error. A computer error occurred in attempting to read the VISION:Forms dictionary.
049	ERR049. ERROR OCCURRED ACCESSING VSAM DICTIONARY.

Code	Message and Description
050	ERR050. xxxxxxxxxx INVALID FUNCTION. ADVISE COMPUTER ASSOCIATES. A combination of VERBS and/or values that VISION:Forms cannot handle was encountered. Please furnish this VISION:Forms program to your computer department and they should contact Computer Associates.
052	ERR052. THE WORD xxxxxxxxxx IS NOT IN DICTIONARY.
053	ERR053. GEN. QJ HAS TOO MANY STATEMENTS. MODIFY. VISION:Report can accept a maximum of 250 statements. This can be overridden with QJOPTION. This VISION:Forms program generated more than that number and must be shortened before attempting to execute VISION:Report.
054	ERR054. DICTIONARY ENTRY OF INSTLSTD MED MISSING. The DICTIONARY must contain one record with file name INSTLSTD and record type MED. See the chapter "**Dictionary Maintenance**."
055	ERR055. PROCEDURE NAME xxxxxxxxxx NOT PRESENT IN DICTIONARY.
056	ERR056. PHASE xxxx TOO LARGE FOR AREA. DEVEL CANCEL. An unanticipated combination of VERBS and/or values similar to ERR 050. Treat the same as ERR 050.
057	ERR057. PROCEDURE xxxxxxxxxx HAS TOO MANY STMT NRS. A maximum of 30 statement numbers can be coded in the VISION:Report statements comprising a procedure.
058	ERR058. 'AT' KEYWORD ONLY VALID WITH START-UP PROCEDURES. Review examples and Verb Structure in **USE PROCEDURE** in the chapter "Writing Verbs."
059	ERR059. START-UP PROCEDURES REQUIRE "AT" KEYWORD. Review logical point in procedure documentation in **Procedure Considerations** in the chapter "Writing Verbs."
060	ERR060. QWTABLE DIC. NAME xxxxxxxxxx NOT IN DICTIONARY. The table name cited in Word 2 (after DICTIONARY in Word 1) is not present in the dictionary.
061	ERR061. QWTABLE xxxxxxxx IS MISSING OR INVALID. Maximum number of table entries, argument position, and/or argument length is invalid or specified incorrectly. See **QWTABLE** in the chapter "Writing Verbs" for table specification coding rules.

Code	Message and Description
062	ERR062. TOO MANY DATANAMES FOR PRINT LINE. FOLG LIST SHOWS PRINT SPACE FOR EACH. YOU MUST DROP SOME. The number of print positions required for each data name in the print statement has been determined. The sum of these requirements plus one blank position after each data name is too wide to be printed on one line. One or more data names must be dropped from the print statement.
063	ERR063. PRINT OF FOLG DATANAMES WILL REQUIRE MORE THAN ONE LINE. PRINT SPACE REQ. FOR EACH IS SHOWN. Same as Error 062.
064	ERR064 xxxxxxxxxxxxxxxxxx DATANAME FLDDEF NOT AVAIL. FOR DETERMINATION OF PRINT SPACE REQ. The data name shown does not have a DICTIONARY or EQUATE entry.
065	ERR065. xxxxxxxxxx VALUE ATTACHED TO WORD SPACE MUST BE NUMERIC. When the expression SPACE is used in a PRINT verb, it must be followed, with no intervening spaces, with a one-or two-digit numeric, i.e., SPACE10.
066	ERR066. FILE xxxxxx OPEN FAILURE. REASON CODE nnn. VISION:Forms scans your standard label track for DLBL and ASSGN information on QW$DICT and QW$FILE then examines operating system information about your assigned SYSnnn to determine device type. For the reason below, this procedure was not completed. 004 = SYS-NR IS ASSIGNED TO TAPE 008 = SYS-NR IS ASSIGNED IGN OR UA STATUS 012 = INVALID OR UNSUPPORTED DEVICE TYPE 016 = DLBL FOR FILE INDICATED WAS NOT FOUND 020 = FILE ORGANIZATION ON DLBL IS INVALID 024 = SYS-NR IS INVALID
067	ERR067. SYSNRS FOR DICT/WORK IN PHASE QKWRENVI ARE NOT NUMERIC. ADVISE YOUR SYSTEMS PROGRAMMER. This is not a user error. Advise your computer department of this error.
068	ERR068. THE VERB xxxxxxxxxx MUST BE CODED AHEAD OF THE VERB xxxxxx.

Code	Message and Description
069	ERR069. DATANAME xxxxxxxxxx MUST BE IN THE HEADLINE PRINT STATEMENT TO BE TOTALED. If multiple PRINT verbs are present in this VISION:Forms program, all data names in the TOTAL statement must also appear in the PRINT HEADLINE verb.
070	ERR070. IF MULTIPLE PRINT STMTS THEN ONE OF THEM MUST START WITH THE WORD HEADLINE. If there is more than one print verb, one of them must be identified as the primary print on which the report format is to be based. Identify that PRINT verb by coding the word HEADLINE immediately after the verb.
071	ERR071. MAX OF 9 DATANAMES MAY BE SPEC. IN BREAK STMT. VISION:Forms accepts only nine BREAK levels.
072	ERR072. FOLG OVRIDE-HDR TOO LONG xxxxxxxxxxxxxxxxxx. Override print headers can be a maximum of 57 characters long including the left and right parentheses.
073	ERR073. QJSIZE VALUE MUST BE 1-3 NUMERIC DIGITS.
074	ERR074. SORTSIZE VALUE MUST BE 1-3 NUMERIC DIGITS.
075	ERR075. PARENTHESIZED STRING INTO NEXT LINE REQUIRES CONTINUATION INDICATOR. If an override column header or other expression in parentheses must be continued on the next line, a non-blank character must be coded in the continuation column of the line to be continued.
076	ERR076. xxxxxxxxxxxx MUST BE IN BREAK STMT TO BE USED IN AN ONBREAK STMT.
077	ERR077. THE VERB xxxxxxxxxxxxx IS INVALID UNLESS PRECEDED BY 'ONBREAK END'.
080	ERR080. PARTITION SPECIFIED MUST BE TWO CHARACTERS.
081	ERR081. FIELD xxxxxxxxxx NOT IN DICTIONARY OR TOO LONG FOR BLANK/ZERO QJ IF VERB. MAX OF 30 POS. Data name shown is either not in dictionary or is too long for a valid VISION:Report IF statement. VISION:Report has a 30-position limit on these comparisons.
082	ERR082. THE EXPRESSION xxxxxxxxxx IS NOT A VALID QWOPTION KEYWORD.
083	ERR083. THE WORD xxxxxxxxxx IS INVALID. One of the words following a QWOPTION verb is invalid.

Code	Message and Description
084	ERR084. TOO MANY TR PTS IN BLOK. ADVISE COMPUTER ASSOCIATES.
086	ERR086. BIN. MATCH FIELDS MUST BE 2 OR 4 BYTES.
087	ERR087. EBCDIC MATCH FIELDS MUST BE SAME LENGTH.
088	ERR088. CARD WITH END-DICTIONARY IN C.C.1-14 IS REQUIRED BEHIND DICTIONARY ENTRIES.
089	ERR089. BLOCK SIZE IN CC32-36 IS NOT NUMERIC.
090	ERR090. REC SIZE IN CC38-42 IS NOT NUMERIC.
091	ERR091. NR OF FILES CC58-60 MUST BE BLANK OR NUMERIC.
092	ERR092. INPUT SYSNR IN CC62-64 MUST BE NUMERIC.
093	ERR093. SYSNR IN CC30-32 MUST BE NUMERIC.
094	ERR094. REPORT SYSNR IN CC38-40 MUST BE NUMERIC.
095	ERR095. FLD POSN IN CC27-30 MUST BE NUMERIC
096	ERR096. FLD POSN IN CC32-35 MUST BE NUMERIC.
097	ERR097. PRT DEC SPECS IN CC38 MUST BE BLANK OR NUMERIC.
098	ERR098. CALC DEC SPECS CC40 MUST BE BLANK OR NUMERIC.
099	ERR099. INVALID FLD DEF-LEFTMOST POS IS GT RIGHTMOST. The preceding inline dictionary entry has invalid from and to positions.
100	ERR100 NR TBL ENTRIES IN CC26-29 MUST BE NUMERIC.
101	ERR101. TOO MANY DICT ENTRIES FOR SPACE AVAL. ENLARGE PARTITION BY 160 X NR OCCUR OF THIS DIAGNOSTIC.
102	ERR102. TABLE ARG LENGTH IN CC34-35 MUST BE NUMERIC.
103	ERR103. RECD TYPE IN CC9-11 IS INVALID FOR DICT.
104	ERR104. TABLE ARG POSN IN CC31-32 MUST BE NUMERIC.
105	ERR105. FILE OPEN FAILURE. REASON CODE = nnn. See documentation for ERR066.
106	ERR106. FILENAME NOT AVAILABLE FOR DICTIONARY RETRIEVAL. File name is not in the dictionary or is misspelled, etc.
107	ERR107. SORT FIELD MUST BEGIN IN 1ST 4092 BYTES OF THE RECORD.

Code	Message and Description
108	ERR108. EDIT MASK CODE IN CC39 MUST BE BLANK, C, E, OR N. See **'D' Records** in the chapter "MVS System Information."
109	ERR109. PREVIOUS VISION:Forms STATEMENT HAS TOO MANY WORDS. Review verb description for that statement.
110	ERR110. A TRANS. PT. IN THIS PROC. NOT CODED AS A SEQ. NUMBER.
111	ERR111. DATANAME FIELD NOT IN HEADLINE PRINT STATEMENT. Review **PRINT** in the chapter "Writing Verbs."
112	ERR112. DATANAME REQUIRES MORE PRINT POSITIONS. Review **PRINT** in the chapter "Writing Verbs."
114	ERR114. DATANAME START POSITION IS LESS THAN CURRENT POSITION ON THE LINE. Review **PRINT** in the chapter "Writing Verbs."
115	ERR115. THE KEYWORD UNDER DOES NOT HAVE A VALID DATANAME. Review **PRINT** in the chapter "Writing Verbs."
116	ERR116. LENGTH OF A CHARACTER STRING EXCEEDS THE MAXIMUM OF 152.
117	ERR117. ILLEGAL EMBEDDED QUOTE OR BEGINNING/ENDING QUOTE IS MISSING IN TITLE STATEMENT.
118	ERR118. RESERVE HEADER NAME IS NOT A VALID DATA NAME FOR THE TITLE STATEMENT - xxxxxxxxxxxxxx.
120	ERR120. xxxxxxxxxxxxxx ALREADY USED IN A NON-ATTACH VERB.
121	ERR121. TOO MANY UNIQUE DATANAMES IN TOTAL AND/OR AVERAGE STATEMENTS.
122	ERR122. GROUPCOUNT MAY NOT BE AVERAGED.
123	ERR123. THE VERB xxxxxxxx MAY BE USED ONLY ONCE.
124	ERR124. FILENAME xxxxxxxx HAS A MISSING OR INVALID VISION:Report AREA.

Code	Message and Description
125	ERR125. FIRST FILENAME MUST BE INF. When specifying more than one file name, the first file name listed must always be the VISION:Report INF area. See the chapter "**Advanced Techniques**."
126	ERR126. xxx CAN BE REFERENCED ONCE.
128	ERR128. DATANAME xxxxxxxxxxxxxxxxx FORMAT IS NOT CURRENTLY SUPPORTED BY VISION:Forms.
129	ERR129. OPTIONS PROCEDURES REQUIRE EITHER "BEFORE" OR "AFTER" KEYWORD. Review **USE PROCEDURE** in the chapter "Writing Verbs."
130	ERR130. DATA NAME xxxxxxxxxxxxxxxxx TOO LONG OR INVALID FORMAT.
131	ERR131. LITERAL IS LONGER THAN DATA NAME.
132	ERR132. A NUMERIC DATA NAME CANNOT BE SELECTED AGAINST A CHARACTER LITERAL, BLANKS, OR LO/HIVALUE.
133	ERR133. DATANAME FORMAT/LENGTH IS INCONSISTENT WITH A PREVIOUS SELECT/QWTABLE STATEMENT.
134	ERR134. FOR THE SELECT/RANGE STATEMENT, A LOW RANGE AND HIGH RANGE ARE REQUIRED.
135	ERR135. LENGTHS/FORMATS OF DATA NAMES IN A SELECT STATEMENT MUST BE THE SAME.
136	ERR136. PAGEWIDTH VALUE EXCEEDS VISION:Report PRINT SIZE OF nnn.
137	ERR137. A PSELECT STATEMENT WAS CODED WITH NO CORRESPONDING PRINT STATEMENT.
138	ERR138. THE NEWTEST VERB IS ONLY VALID WHEN PRECEDED BY A SELECT VERB.
139	ERR139. NO GETVIS STORAGE. INCREASE PARTITION SIZE OR EXEC SIZE.
140	ERR140. ONLY ONE QWTABLE PER VISION:Forms ALLOWED.
142	ERR142. TABLE DATA GIVEN WITH NO QWTABLE SPECIFIED.
150	ERR150. REFERENCE TO TABLE FUNCTION EXCEEDS MAX. SPECIFIED IN QWTABLE STATEMENT.
151	ERR151. MUST HAVE CODED A QWTABLE STATEMENT TO REFERENCE THE TABLE FUNCTION.

Code	Message and Description
152	ERR152. MUST HAVE CODED A FUNCTION POSITION AND LENGTH IN YOUR QWTABLE STATEMENT.
153	ERR153. PRINT SPECS OR CALC SPEC INVALID.
155	ERR155. FILE xxxxxxxx IS MISSING A MEDIA TYPE, BLOCK SIZE, RECORD SIZE, OR SYS-NUMBER.
156	ERR156. MUST HAVE A PROCEDURE TO ACCESS A DATA BASE.
157	ERR157. THE WORD xxxxxxxxxxxxxxxxx IS NOT A VALID MATHEMATICAL OPERAND.
158	ERR158. INLINE DICTIONARY KEY xxxxxxxxxxxxxxxxxxxxxxxx IS A DUPLICATE.
800	ERR800. DD CARD MISSING. A required DD statement is missing from JCL. Add statement and rerun job.
801	ERR801. NOT ENOUGH MEMORY AVAILABLE TO RUN VISION:Forms. Increase size of partition or region and rerun job.
802	ERR802. PHASE QKWRXXXX MISSING FROM NAMETABL - NOTIFY COMPUTER ASSOCIATES.
803	ERR803. PROGRAM QKWRXXXX NOT IN PROGRAM LIBRARY. Ensure all VISION:Forms program modules were put in same library, correctly. If they were, notify Computer Associates.
804	ERR804. QWDICT DD CARD MISSING. AN ISAM AND/OR IN-LINE DICTIONARY IS REQUIRED.
805	ERR805. SORT WAS UNABLE TO SORT IN-LINE DICTIONARY. REFER TO SORT MESSAGES FOR REASON.

Glossary

ALPHA	Pertaining to characters of the alphabet.
ARGUMENT	In a table, used as entry identification to determine positioning within the table.
CONSTANT	A fixed value.
DATANAME	A unique name assigned to a data field.
DICTIONARY	A VISION:Forms required file containing field definitions and printing formats for each field.
EBCDIC	An alphanumeric code used to represent data characters in computer languages.
EDIT	To modify the format of data, e.g., insert commas and/or decimals into a number.
FILE	A collection of related records treated as a unit.
FILE LAYOUT	The pictorial description of fields in a file.
INPUT	Data going into a job or program.
INSTRUCTION	A statement that specifies an operation; consists of operands.
JCL	Job Control Language. Used to tell the computer input device, output device, etc.
KEY	One or more characters used to identify a particular piece of information or record.
KEYWORD	Significant identifying words used in a program to signal a specific response.
MASK	Pattern of characters used to form another set of characters.
NUMERIC	Pertaining to numbers.
OPERANDS	Words in an instruction defining the participants of the operation.
OVERRIDE	Assign an alternate name or value.
VISION:Report FIELD	Defining an area name with the from-to positions of that

DEFINITION	particular area (INF1-5).
PACK	To shorten the space used for numeric data in computer storage or memory (i.e.,-P).
PACKED DECIMAL	A number in computer memory that is squeezed into a smaller space.
RECORD	A collection of related fields, treated as a unit, e.g., one line of an employee's payroll information is a record.
STRING	A sequence of characters.
TABLE	A list of data identified by a label or by the positions of the data.
TRUNCATE	To cut off or shorten.
USER	Anyone who requires services of a computing system.

Index

:

'

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z