Advantage™ VISION:Report®
Advantage™ VISION:Forms™
for VSE

Installation Guide

16.1

Computer Associates™

RPINV161.PDF/D21-011-010

Contents

Chapter 1: Installing VISION:Report Under VSE

Chapter 2: Installing VISION:Report Optional Material

Chapter 3: Installing VISION:Forms Under VSE

Index

Chapter 1

Installing VISION:Report Under VSE

Thank you for choosing VISION:Report® 16.1. Before you install the software, read the following sections for important information.

- CD-ROM Contents
- About the Online Documentation
- Contacting Total License Care (TLC)
- Contacting Computer Associates

CD-ROM Contents

- Online documentation
- Adobe® Acrobat® Reader software and Acrobat Help

About the Online Documentation

The CD-ROM contains the documentation for VISION:Report. The documents, called books, are in Adobe Acrobat Portable Document Format (PDF) and are designed for you to read online using the Acrobat Reader.

Each online document contains a table of contents, index, and cross-references.

Installing Online Documentation and the Acrobat Reader

You can install the online documentation on your local hard drive or on a network server. Alternately, you can access the documentation directly from the CD-ROM.

If you do not have Acrobat Reader installed, you can install it from the CD-ROM.

To install the online documentation, the Acrobat Reader, or both on Windows:

1. Close all application programs.

2. Insert the CD-ROM into the CD-ROM drive.
3. Click the Start menu and select Run.
4. In the Run dialog box, type: D:\Books\Setup.exe,
   where D: is the CD-ROM drive.

   To complete the Run dialog box and continue, click OK.
5. Follow the instructions. Computer Associates recommends that you install the online documentation in the default directory (C:\ProgramFiles\Computer Associates\VISION_Report 16.1 VSE\Books\) or a directory of your choice (for example, C:\VISION_Report 16.1 VSE\Books\).

Viewing Online Documentation

Regardless of the location of the online documentation (on a local drive, a network server, or CD-ROM), you can view the online documentation using the following methods:

- Click the Start menu, point to Programs, point to VISION_Report 16.1 VSE, and click the document title.
- In Windows Explorer, point to the directory on the hard drive where you installed the online documentation. Double-click the PDF file name.
- In Windows Explorer, point to the Books directory on the CD-ROM drive and double-click the PDF file name.

Using Adobe Acrobat Reader

Use Acrobat Reader to view the online documentation, adjust the size of the page, and perform searches. For more information, use the Acrobat Help menu.

Contacting Total License Care (TLC)

TLC is available Monday-Friday 7 am - 9 pm Eastern Time in North America and 7 am - 7 pm United Kingdom time. Additionally, 24-hour callback service is available for after hours support. Contact TLC for all your licensing requirements.

Be prepared to provide your site ID for product activation.

To activate your product, use one of the following:

North America:	800-338-6720 (toll free) 631-342-5069	help@licensedesk.cai.com
Europe:	00800-1050-1050	euro.tlc@ca.com
If your company or local phone service does not provide international access, please call your local Computer Associates office and have them route you to the above number.		
Australia:	1-800-224-852	
New Zealand:	0-800-224-852	
Asia Pacific:	800-224-852	
Brazil:	55-11-5503-6100	
Japan:	JPNTLC@ca.com (email is preferred)	

Contacting Computer Associates

For 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.

Installation Instructions

Note: The installation and execution of VISION:Report supports only VSE/SP 2.1.1 and above. This is the VSE release when IBM restructured the VSE library organization, utilizing the VSE program, LIBR. If you are on a release prior to VSE/SP 2.1.1, there is no guarantee that VISION:Report will run successfully, even if you are able to modify the JCL and control cards and successfully install this release.

This chapter describes the installation procedure for VISION:Report in a VSE environment. Two ancillary products are associated with VISION:Report on a VSE system — VISION:Forms and VISION:Report Interface to DB2.

First, install VISION:Report, as described in this chapter. Then, if you are licensed to use VISION:Forms, proceed to Chapter 3, Installing VISION:Forms Under VSE. *If you are licensed to use* VISION:Report Interface to DB2, refer to the *VISION:Report Interface to DB2 Installation Guide*.

Review the New Release Planning Guide, Chapter 2 - Upgrading from Previous Releases, for situations that apply to your installation. Although Program Customizing Patches (PCPs) from File 2 may be used to assist you, Computer Associates recommends that you modify programs to conform to the proper syntax.

Tape Files

VISION:Report is distributed on a compatible-standard label 3480 cartridge tape. The volume serial number of the tape appears on the external tape label. The tape contains four data files.

- Files 1 and 2 are fixed, with a logical record size of 80 bytes and a blocksize of 80 bytes.
- File 3 is fixed-blocked, with a logical record size of 352 bytes and a blocksize of 5,280 bytes.
- File 4 is fixed, with a logical record size of 80 bytes and a blocksize of 80 bytes.

The contents of each file is as follows:

File 1:	■ VISION:Report Object Sublibrary Library ■ VISION:Report Source / Macro Sublibrary Library ■ VISION:Report Sample Programs ■ Optional Materials (See File 1: VISION:Report Object Sublibrary, VISION:Report Source / Macro Sublibrary, VISION:Report Sample Programs, Optional Materials.)
File 2:	■ VISION:Forms Object Sublibrary ■ VISION:Forms Source / Macro Sublibrary ■ VISION:Forms Sample Programs (See File 2: VISION:Forms Object Sublibrary, VISION:Forms Source / Macro Sublibrary, VISION:Forms Sample Programs.)
File 3:	■ VISION:Report Test Accounts Receivable (AR) File (See File 3: VISION:Report Test Accounts Receivable (AR) File.)

File 4:	■ VISION:Report Interface to DB2 and SQL/DS Object Sublibrary ■ VISION:Report Interface to DB2 Source/Macro Sublibrary ■ VISION:Report Interface to DB2 Sample Programs (See File 4: VISION:Report Interface to DB2 and SQL/DS Object Sublibrary, VISION:Report Interface to DB2 Source/Macro Sublibrary, VISION:Report Interface to DB2 Sample Programs.)

File 1: VISION:Report Object Sublibrary, VISION:Report Source/Macro Sublibrary, VISION:Report Sample Programs, Optional Materials

This file contains:

- Object modules that must be processed by the linkage editor before they can be executed.
- Source code, which is provided for some optional materials, and sample VISION:Report programs. (Within the file, see member @INDEXQJ.A.)
- Assembler language macros required for assembling the option block, adding additional edit mask codes, enlarging the EQU tables, and so on.
- Sample JCL jobstreams for installing optional materials such as VISION:Forms and VISION:Report Interface to DB2.

Since VSE does not permit JCL to specify source members as input, many of the sample JCL jobstreams, as well as VISION:Report sample programs may have to be punched or edited to ensure that no control statements are included in any submitted jobstream. This method of editing and job submission varies from one company to another.

File 2: VISION:Forms Object Sublibrary, VISION:Forms Source/Macro Sublibrary, VISION:Forms Sample Programs

This file contains:

- Object modules that must be processed by the linkage editor before they can be executed.
- Source programs and macros, including sample VISION:Forms programs, or code required when assembling VISION:Forms programs. You would only install this file if you are licensed to use VISION:Forms. (Within the file, see member @INDEXQW.A.)

File 3: VISION:Report Test Accounts Receivable (AR) File

This test file is a sequential file used by some of the sample programs. It is also used as part of the installation verification.

File 4: VISION:Report Interface to DB2 and SQL/DS Object Sublibrary, VISION:Report Interface to DB2 Source/Macro Sublibrary, VISION:Report Interface to DB2 Sample Programs

This file contains:

- Object modules that must be processed by the linkage editor before they can be executed.
- Source programs and macros, including sample VISION:Report Interface to DB2 programs, or code required when assembling VISION:Report Interface to DB2 programs. You would only install this file if you are licensed to use VISION:Report Interface to DB2. (Within the file, see member @SQLINDX.A.)
- Sample JCL, macro parameters, and jobstreams to allow you to customize VISION:Report Interface to DB2.
- Sample VISION:Report programs to load, print, and update a DB2 table.

Installation Steps

To install VISION:Report on a VSE system, read this entire section first.

The original distribution tape contains unblocked 80-byte image records, you can use as input to SYSIPT for all tape files, except File 3.

Note: The JCL shown below may not be exactly identical to those on the restored files.

1. Prepare JCL similar to the following example to restore File 1 from the original distribution tape, and link edit object modules into the Phase sublibrary:

```
* $$ JOB JNM=QJINSTL,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB QJINSTL  INSTALL VISION:REPORT 16.1, VSE
// OPTION LOG,NODUMP
* STEP1: DEFINE LIBRARY
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*        ',USERID' TO: IF UNDER VM, YOUR CMS USERID
/*        '@@@VLL'  TO: VOLUME SERIAL NUMBER OF QJ LIBRARY
/*        '@@@TRL'  TO: STARTING TRACK OF QJ LIBRARY
/*        '@@@AAA.@@@BBB.LIB' TO: QJ LIBRARY NAME
/*        '@@TAPE'  TO: ORIGINAL DISTRIBUTION QJ161 TAPE
/*
// ASSGN  SYS020,DISK,VOL=@@@VLL,SHR        QJ LIBRARY
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366  QJ LIBRARY
// EXTENT SYS020,@@@VLL,1,0,@@@TRL,125
// EXEC   LIBR,SIZE=512K
   DEFINE     LIB=QJLIB            REPLACE=YES
     DEFINE   SUBLIB=QJLIB.QJ161   REPLACE=YES  REUSE=IMMEDIATE
     DEFINE   SUBLIB=QJLIB.PTF     REPLACE=YES  REUSE=IMMEDIATE
/*
* STEP2: RESTORE LIBRARY
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY
// EXTENT       ,@@@VLL
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161),                    X
              CATALOG=QJLIB.QJ161
// LIBDEF SOURCE,SEARCH=QJLIB.QJ161
// LIBDEF    OBJ,SEARCH=QJLIB.QJ161
// PAUSE  MOUNT DISTRIBUTION TAPE ON @@TAPE
// MTC    REW,@@TAPE               ENSURE TAPE AT LOAD POINT
// MTC    FSF,@@TAPE,1             SKIP TO FIRST FILE
// ASSGN SYSIPT,@@TAPE
// EXEC   LIBR,SIZE=512K,PARM='ACC SUB=QJLIB.QJ161'
/*
/&
* $$ EOJ
```

2. Punch the member QJLINK.A located in the library created in the previous job. Remove the second asterisk from the JCL cards. Modify the JOB, LST and PUN cards to your site requirements. Insert the library name in the / / DLBL card and the volume serial number in the / / EXTENT card. Run the job.

```
**$$ JOB JNM=QJLINK,CLASS=A,LDEST=(,USERID)
**$$ LST CLASS=A,DEST=(,USERID)
**$$ PUN CLASS=A,DEST=(,USERID)
// JOB QJLINK                    LINK EDIT QJ PHASES
// DLBL IJSYSPH,'===.PRMOD',1,SD
// EXTENT SYSPCH,SYSWK1,,,1,100
   ASSGN SYSPCH,DISK,VOL=SYSWK1,SHR
// OPTION DECK
// EXEC ASSEMBLY
   PRMOD CTLCHR=ASA,DEVICE=PRT1,IOAREA2=YES,RECFORM=VARUNB,SEPASMB=YES,X
               WORKA=YES
   END
/*
   CLOSE SYSPCH,PUNCH
// DLBL    QJ,'QJLIBRARYNAME'  <=YOUR QJ LIBRARY NAME
// EXTENT  ,VOLSER             <=YOUR QJ LIBRARY VOLUME SERIAL NUMBER
// DLBL IJSYSIN,'===.PRMOD'
// EXTENT SYSIPT,SYSWK1
   ASSGN SYSIPT,DISK,VOL=SYSWK1,SHR
// EXEC PGM=LIBR,PARM='ACCESS SUBLIB=QJ.QJ161'
/*
   CLOSE SYSIPT,READER
// LIBDEF  *,SEARCH=QJ.QJ161,CATALOG=QJ.QJ161
// OPTION CATAL
   PHASE QJADDR,S
   INCLUDE QJADDR
// EXEC LNKEDT
   PHASE QJBXREF,S
   INCLUDE QJBXREF
// EXEC LNKEDT
   PHASE QUIKESQL,S
   MODE RMODE(24)
   INCLUDE QJESQL
   INCLUDE QJO#SQL
   INCLUDE DYL#SQL
// EXEC LNKEDT
   PHASE QUIKPSQL,S
   INCLUDE QJPSQL
// EXEC LNKEDT
   PHASE QJZARK,S
   INCLUDE QJZARK
// EXEC LNKEDT
   PHASE QUIKACUM,S
   INCLUDE QUIKACUM
// EXEC LNKEDT
   PHASE QUIKADD,S
   INCLUDE QUIKADD
// EXEC LNKEDT
   PHASE QUIKBOMB,S
   INCLUDE QUIKBOMB
// EXEC LNKEDT
   PHASE QUIKCEDT,S
   INCLUDE QUIKCEDT
// EXEC LNKEDT
   PHASE QUIKCOMP,S
   INCLUDE QUIKCOMP
// EXEC LNKEDT
   PHASE QUIKDATT,S
   INCLUDE QUIKDATT
// EXEC LNKEDT
   PHASE QUIKEEDT,S
   INCLUDE QUIKEEDT
// EXEC LNKEDT
   PHASE QUIKEMDV,S
   INCLUDE QUIKEMDV
```

```
// EXEC LNKEDT
   PHASE QUKBEMSK,S
   INCLUDE QUIKEMSK
// EXEC LNKEDT
   PHASE QUIKEQUL,S
   INCLUDE QUIKEQUL
// EXEC LNKEDT
   PHASE QUIKEXEC,S
   INCLUDE QUIKEXEC
// EXEC LNKEDT
   PHASE QUIKGNHC,S
   INCLUDE QUIKGNHC
// EXEC LNKEDT
   PHASE QUIKGNHK,S
   INCLUDE QUIKGNHK
// EXEC LNKEDT
   PHASE QUIKHB,S
   INCLUDE QUIKHB
// EXEC LNKEDT
   PHASE QUIKIF,S
   INCLUDE QUIKIF
// EXEC LNKEDT
   PHASE QUIKLOGI,S
   INCLUDE QUIKLOGI
// EXEC LNKEDT
   PHASE QUIKMOVE,S
   INCLUDE QUIKMOVE
// EXEC LNKEDT
   PHASE QUIKMPRT,S
   INCLUDE QUIKMPRT
// EXEC LNKEDT
   PHASE QUIKMVE,S
   INCLUDE QUIKMVE
// EXEC LNKEDT
   PHASE QUIKPERF,S
   INCLUDE QUIKPERF
// EXEC LNKEDT
   PHASE QUIKPMDV,S
   INCLUDE QUIKPMDV
// EXEC LNKEDT
   PHASE QUIKPRNT,S
   INCLUDE QUIKPRNT
// EXEC LNKEDT
   PHASE QUIKPRPT,S
   INCLUDE QUIKPRPT
// EXEC LNKEDT
   PHASE QUIKPSOR,S
   INCLUDE QUIKPSOR
// EXEC LNKEDT
   PHASE QUIKTBLD,S
   INCLUDE QUIKTBLD
// EXEC LNKEDT
   PHASE QUIKTITL,S
   INCLUDE QUIKTITL
// EXEC LNKEDT
   PHASE QUKBTRN,S
   INCLUDE QUIKTRNT
// EXEC LNKEDT
   PHASE QUKBVEQU,S
   INCLUDE QUIKVEQU
// EXEC LNKEDT
   PHASE TABLSORT,S
   INCLUDE TABLSORT
// EXEC LNKEDT
   PHASE TABLSOR2,S
   INCLUDE TABLSOR2
```

```
// EXEC LNKEDT
   PHASE POSTSQL,S
   INCLUDE POSTSQL
// EXEC LNKEDT
   PHASE QJCOBCVT,S
   INCLUDE QJCOBCVT
// EXEC LNKEDT
   PHASE QJCOMREG,S
   INCLUDE QJCOMREG
// EXEC LNKEDT
   PHASE QJERAND,S
   INCLUDE QJERAND
// EXEC LNKEDT
   PHASE QJJOBCOM,S
   INCLUDE QJJOBCOM
// EXEC LNKEDT
   PHASE QJPUNINT,S
   INCLUDE QJPUNINT
// EXEC LNKEDT
   PHASE QUIKDATE,S
   INCLUDE QUIKDATE
// EXEC LNKEDT
   PHASE QUIKDPRT,S
   INCLUDE QUIKDPRT
// EXEC LNKEDT
   PHASE QUIKESOR,S
   INCLUDE QUIKESOR
// EXEC LNKEDT
   PHASE QUIKFILE,S
   INCLUDE QUIKFILE
// EXEC LNKEDT
   PHASE QUIKFLOP,S
   INCLUDE QUIKFLOP
// EXEC LNKEDT
   PHASE QUIKINCL,S
   INCLUDE QUIKINCL
// EXEC LNKEDT
   PHASE QUIKIO,S
   INCLUDE QUIKIO
   INCLUDE QUIKSD
   INCLUDE QUIKMT
   INCLUDE QUIKIS
// EXEC LNKEDT
   PHASE QUKBISLD,S
   INCLUDE QUKBISLD
// EXEC LNKEDT
   PHASE QUKBISMD,S
   INCLUDE QUKBISMD
// EXEC LNKEDT
   PHASE QUIKPOPT,S
   INCLUDE QUIKPOPT
// EXEC LNKEDT
   PHASE QUIKRPT,S
   MODE AMODE(31)
   INCLUDE QUIKRPT
// EXEC LNKEDT
   PHASE QUIKTABL,S
   INCLUDE QUIKTABL
// EXEC LNKEDT
   PHASE QUIKTIME,S
   INCLUDE QUIKTIME
// EXEC LNKEDT
   PHASE QUIKTRAN,S
   INCLUDE QUIKTRAN
// EXEC LNKEDT
   PHASE QUIKVSAM,S
```

```
   INCLUDE QUIKVSAM
// EXEC LNKEDT
   PHASE QUKBJOB,S
   MODE AMODE(31)
   INCLUDE QUKBJOB
// EXEC LNKEDT
   PHASE QUKBLIB,S
   INCLUDE QUKBLIB
// EXEC LNKEDT
   PHASE QUIKMFB,S
   INCLUDE QUIKMFB
// EXEC LNKEDT
   PHASE QUIKMFU,S
   INCLUDE QUIKMFU
// EXEC LNKEDT
   PHASE QUIKMSBW,S
   INCLUDE QUIKMSBW
// EXEC LNKEDT
   PHASE QUIKMSUW,S
   INCLUDE QUIKMSUW
// EXEC LNKEDT
   PHASE QUIKMUD,S
   INCLUDE QUIKMUD
// EXEC LNKEDT
   PHASE QUIKMUDW,S
   INCLUDE QUIKMUDW
// EXEC LNKEDT
   PHASE QUIKMVB,S
   INCLUDE QUIKMVB
// EXEC LNKEDT
   PHASE QUIKMVBW,S
   INCLUDE QUIKMVBW
// EXEC LNKEDT
   PHASE QUIKMVU,S
   INCLUDE QUIKMVU
// EXEC LNKEDT
   PHASE QUIKMVUW,S
   INCLUDE QUIKMVUW
// EXEC LNKEDT
/&
* $$ EOJ
```

3. Assemble the QJOPTION block. You can use the QJOPTREC member in your restored VISION:Report library. To punch out this member, as well as some sample Assembly JCL, punch out member LIBR01, modify it, and submit it. A listing of LIBR01 is shown for your convenience:

```
* $$ JOB JNM=QJLIBR01,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB QJLIBR01   PUNCH MODULE 'QJOPTXXX' FROM LIBRARY
// OPTION LOG,NODUMP
/*
/* ====> IF YOU USE THIS JOBSTREAM TO PUNCH OUT THE MEMBERS,
/* PLEASE REMOVE ANY STATEMENTS - ON THE PUNCHED OUTPUT (!) -
/* AFTER THE '/&' AND/OR '* $$ EOJ'.
/*
/* THIS IS A SAMPLE TO PUNCH QJOPTREC, QJOPTDB2, OR QJOPTDEF
/* SO IT CAN BE MODIFIED.
/* NOTE THAT JCL TO ASSEMBLE THE QJOPTION BLOCK IS
/* ALSO PUNCHED.
/*
/* CHANGE QJOPTREC.A TO MODULE DESIRED, IF NEEDED.
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
```

```
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*        ',USERID' TO: IF UNDER VM, YOUR CMS USERID
/*        '**$$'    TO: '* $$'
/*        '@@@VLL'  TO: VOLSER OF QJ LIBRARY
/*        '@@@AAA.@@@BBB.LIB' TO: QJ LIBRARY NAME
/*
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY
// EXTENT       ,@@@VLL
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ),                    X
              CATALOG=QJLIB.QJ161
// LIBDEF SOURCE,SEARCH=QJLIB.QJ161
// LIBDEF    OBJ,SEARCH=QJLIB.QJ161
// EXEC   LIBR,SIZE=512K,PARM='ACC SUB=QJLIB.QJ161'
    PUNCH QJOPTREC.A
    PUNCH ASMQJOPT.A
/*
/&
* $$ EOJ
```

A listing of ASMQJOPT is shown below for your convenience.

```
* $$ JOB JNM=ASMQJOPT,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB ASMQJOPT    ASSEMBLE QJOPTION BLOCK
// OPTION LOG,NODUMP
*  ASSEMBLE QJOPTION BLOCK
/*
/* ====> IF YOU USED LIBR TO PUNCH OUT THIS MEMBER
/* PLEASE REMOVE ANY STATEMENTS - ON THE PUNCHED OUTPUT (!) -
/* AFTER THE '/&' AND/OR '* $$ EOJ'.
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*        ',USERID' TO: IF UNDER VM, YOUR CMS USERID
/*        '**$$'    TO: '* $$'
/*        '@@@VLL'  TO: VOLSER OF QJ LIBRARY
/*        '@@@AAA.@@@BBB.LIB' TO: QJ LIBRARY NAME
/*
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY
// EXTENT       ,@@@VLL
// LIBDEF  PHASE,SEARCH=(QJLIB.QJ161,QJLIB.PTF),                    X
              CATALOG=QJLIB.QJ161     KEEP IN THIS ORDER
// LIBDEF SOURCE,SEARCH=QJLIB.QJ161
// LIBDEF    OBJ,SEARCH=QJLIB.QJ161
// OPTION CATAL
     PHASE QUIKJOB1,S
// EXEC ASSEMBLY
* SOURCE INPUT OF YOUR MODIFIED 'QJOPTXXX' FOLLOWS,
* AND BEFORE 'END' STATEMENT:
        END
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

4. To verify the installation, punch out member QJTSTPGM, modify it and submit it. This jobstream restores the test Accounts Receivable file (File 3) from the original distribution tape, and also reads the restored file. A partial listing of QJTSTPGM is shown for your convenience:

```
* $$ JOB JNM=QJTSTPGM,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB QJTSTPGM  TEST VISION:REPORT AFTER INSTALLATION
// OPTION LOG,NODUMP
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*        ',USERID'  TO: IF UNDER VM, YOUR CMS USERID
/*        '**$$'     TO: '* $$' IF UNDER VM
/*        '@@@VLL'   TO: VOLUME SERIAL NUMBER OF QJ LIBRARY
/*        '@@ARFILE' TO: NAME OF ARFILE FOR TESTING
/*        '@@@VLA'   TO: VOLUME SERIAL NUMBER OF ARFILE
/*        '@@@TRL'   TO: STARTING TRACK OF ARFILE
/*        '@@@AAA.@@@BBB.LIB' TO: QJ LIBRARY NAME
/*        '@@TAPE'   TO: ORIGINAL DISTRIBUTION QJ TAPE DRIVE
/*
* TEST INSTALLATION WITH QJ PROGRAM
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY
// EXTENT       ,@@@VLL
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161),                      X
              CATALOG=QJLIB.QJ161
// LIBDEF SOURCE,SEARCH=QJLIB.QJ161
// LIBDEF    OBJ,SEARCH=QJLIB.QJ161
// ASSGN  SYS020,@@TAPE       POINT TO ORIGINAL DISTRIBUTION TAPE DRIVE
// PAUSE MOUNT ORIGINAL DISTRIBUTION TAPE ON @@TAPE
// MTC   REW,SYS020           ENSURE AT LOAD POINT
// MTC   FSF,SYS020,6         SKIP TO FILE 3 FOR AR FILE,B/4 HEADER
// TLBL  ARFILET,'QJ.REL161.FILE3'      TAPE AR INPUT FILE
// DLBL      OFA,'@@ARFILE',99/366
// EXTENT SYS010,@@@VLA,1,0,@@@TRL,2
// DLBL  ARFILED,'@@ARFILE',99/366       DISK AR INPUT FILE
// EXTENT SYS010,@@@VLA,1,0,@@@TRL,2
// ASSGN  SYS010,DISK,VOL=@@@VLA,SHR
* RUN VISION:REPORT LISTOPT=YES ONLY JOB
// EXEC  PGM=QUKBJOB,SIZE=512K
OPTION LISTOPT=YES        /*  JUST SHOW ALL OPTIONS
9999END
/*
* RUN VISION:REPORT RESTORE ARFILE JOB
// EXEC  PGM=QUKBJOB,SIZE=512K
OPTION UEXIT1=QUIKINCL,SEQCHK=NO,STMTEND=71,HDRDOTS=NO,RPTSPCE=4
INFTAPE    0352SSYS020  BS=5280,LBL=ARFILET,REWIND=NOREWIND
INZDISC52800352SSYS010  LBL=ARFILED,BF=N
OFADISC52800352SSYS010
EQU  AR-ENT-RECORD          INZ000-000
++INCLUDE Q.ARDEFINE
```

. . . . remainder of jobstream.

5. If you have the optional feature, VISION:Forms, you can review the installation instructions in Chapter 3, *as well as the member QWINSTL. If you have* VISION:Report Interface to DB2, you can review the *VISION:Report Interface to DB2 Installation Guide*, as well as the member SQLINSTL.

6. Upon completion of installation, test your most critical applications.

7. Enhance the functionality of the base VISION:Report system by using the VSE optional materials features. Follow the instructions in Chapter 2.

These features enhance the functionality of VISION:Report:

QUIKDATE	Date Calculation
QUIKRPT	Multiple Reports Subroutine
QUKBVEQU	EQU Table for the VAL Area

These features allow the product's functionality to be subtly changed:

QJOPTION	Modify the VISION:Report Options Block
QUKBEMSK	Modify the Edit Mask Table
QUKBTRN	Modify the Translate Table

Note: Do not modify macros. Modification of program source code may not be supported by Computer Associates. If you do modify the program source code, ensure that you keep the original. (See Chapter 2.)

The Source / Macro sublibrary contains assembler language code and macros required for assembling Optional Material and the QJOPTION option block. It allows you to add additional edit mask codes, enlarge the EQU tables, some sample programs, as well as various JCL members.

These include JCL to run VISION:Report sample programs, as well as installing optional features for which you must be licensed, such as VISION:Forms and VISION:Report Interface to DB2. The Source / Macro sublibrary may also contain Program Temporary Fixes (PTFs), and Program Customizing Patches (PCPs) to help in upward compatibility or special environments. Review PCPs to ensure that they are applicable to your installation.

Of special importance in the Source / Macro sublibrary are the QJOPTDEF, QJOPTREC, and QJOPTDB2 members, which you need to modify. These are described in Assembling QJOPTION option block and Installation Verification.

Assembling QJOPTION Option Block and Installation Verification

Carefully review all of the options (see the OPTION verb in the VISION:Report Reference Guide). This release contains substantial changes and additions.

Note: The option block phase, QUIKJOB1, should be linked as:
PHASE QUIKJOB1,S.

These option members are included for customizing the QJOPTION block:

QJOPTDEF	Default values provided for back level support
QJOPTREC	Recommended values for modern VSE systems
QJOPTDB2	Recommended values for VISION:Interface for DB2 and SQL / DS Users

Note: The PRODCOD parameter is no longer needed and is now obsolete, though it can be left in for compatibility. Instead, you will need to call Total License Care (TLC) for the necessary LMP key(s) to run VISION:Report; do not forget any optional features that will also require a LMP key, such as VISION:Report Interface to DB2.

```
*                 *** PROGRAM QJOPTREC ***
*
*****************************************************************
*                                                               *
* THIS PROGRAM CONTAINS THE RECOMMENDED VALUES FOR QJOPTION     *
* FOR MODERN VSE ENVIRONMENTS, WITHOUT OUR OPTIONAL FEATURE,    *
* VISION:REPORT INTERFACE TO DB2                                *
*                                                               *
* ===>  NOTE:  THIS IS NEW, STARTING WITH RELEASE 16.1 +        *
*              YOU WILL NEED TO OBTAIN THE NECESSARY            *
*              LMP KEY(S) BEFORE RUNNING ANY VISION:REPORT      *
*              PROGRAMS.                                        *
*                                                               *
*       NOTE:  THE 'PRODCOD' IS NO LONGER NECESSARY.            *
*                                                               *
* THE RECOMMENDED VALUES THAT ARE DIFFERENT FROM THE DEFAULTS   *
* ARE SHOWN WITH A PLUS (+) IN FRONT OF THE COMMENT.            *
*                                                               *
* NOTE:  A NON-BLANK CHARACTER IN POSITION 72 SIGNIFIES A       *
*        CONTINUATION STATEMENT ONTO THE NEXT LINE, WHICH       *
*        MUST START IN POSITION 16.                             *
*                                                               *
*****************************************************************
         QJOPTION           ,      VSE SAMPLE                          *
               BWZ=NO,              BLANK WHEN ZERO                     *
               CFLEOPT=NO,          FILE TOTALS ON CONSOLE              *
               CLRVIP=YES,         + CLEAR INPUT AREA BEFORE READ       *
               CLRVOP=YES,         + CLEAR OUTPUT AREA BEFORE WRITING   *
               CRSIGN=NO,           CREDIT SIGN                         *
               DBIRTN=NO,           RETURN TO DB INTERFACE IF ABEND     *
               DUMPALL=NO,          PDUMP SUPVR, GETVIS                 *
               EDIT=NO,             COMPILE, BUT DO NOT EXECUTE         *
               EDITALL=NO,          EDIT MASKS FOR ALL, NOT JUST PRT    *
               EDTNAME=QUKBEMSK,    EDIT MASK PATTERN TABLE             *
               EUROPTN=NO,          EUROPEAN VARIATION OF COMMAS/PERIODS*
               EXPMLOG=NO,          EXPIRATION MESSAGE ON CONSOLE-45DAYS*
               EXPMLST=YES,         EXPIRATION MESSAGE ON PRINTER       *
               HDRDOTS=YES,         GNERATED FIELD HDGS USE '.' AS FILL *
               HOSTRTN=NO,          RETURN TO CALLER AT EOJ             *
               IFNUM=NO,            COMPARE UNLIKE FIELDS NUMERICALLY   *
               LIST=YES,            LIST PROGRAM, DIAG. & STATISTICS    *
               LISTABL=NO,          PRINT TABLE ENTRIES AS LOADED       *
               LISTOPT=NO,          'YES'  FOR DEBUGGING                *
               MBUFFER=YES,         MULTI-BUFFER ALL DISK & TAPE        *
               MOVCVTX=NO,          MOVE X'..' CONVERTS                 *
               OVLY=NO,             RUN IN OVERLAY MODE AT COMPILE TIME *
               PARMEXE=NO,          USE PARM= OF THE // EXEC STMT       *
```

```
        PFLEOPT=NO,          PRINT FILE COUNTS ON PRT IF LIST=NO *
        PRNTLCT=54,          NR. LINES PER PAGE                  *
        PRTSIZE=133,         LENGTH OF OUTPUT ON PRT FILE        *
        QJMDUMP=YES,         HEX PRINT OF ACTIVE AREAS ON ABEND  *
        RESVMEM=0,           RESERVES NNK MEMORY                 *
        RPTSPCE=0,           SPACE BETWEEN DATA COLUMNS (REPORT) *
        RPTSYS=000,          SYS NR FOR QJ OUTPUT                *
        SAVAREA=1024,       + SAV AREA WORKING STORAGE           *
        SEQCHK=NO,           SEQUENCE CHECK ON PROGRAM INPUT STMT*
        SORTABL=YES,         INTERNAL SORT ON TABLE AFTER LOADED *
        SORTPRT=YES,         INFORMATION MESSAGES FROM SORT      *
        SORTRTE=BOTH,        SORT MSG ROUTED TO SYSLST OR SYSLOG *
        SORTSIZ=64,         + MEMORY FOR SORT                    *
        SORTSYS=001,         FIRST LOG SYSNR FOR SORT WORK AREAS *
        SORTWRK=1,           NUMBER SORT WORK AREAS TO BE USED   *
        SRTADJ=NO,           SORT ADJUST OFFSET                  *
        STMTEND=71,         + ALLOWS SEQ IN 73-80                *
        STMTIN=SYSIPT,       SYSTEM LOGICAL UNIT TO READ PGM/TBL *
        STMTLCT=50,          NR. LINES PER PAGE FOR PRINTING PGM *
        STXITPC=YES,         TRAP PGM CHECKS                     *
        TRACECT=10,          TRACE ENTRIES ON ONE LINE WHEN PRTED*
        TRLNAME=QUKBTRN,     2 TRANSLATE TABLES                  *
        UABNDMP=NO,          DUMP WHEN CANCELLED DUE TO V:R ABEND*
        UEXIT1=NO,           SUBRTN USED TO GET STMTS FROM SRCLIB*
        U331DMP=YES,         DUMP DUE TO I/O ERRORS              *
        U333ABE=NO,          DUMP DUE TO COMPILE ERRORS          *
        U335DMP=YES,         DUMP ON RUNTIME PARM FAILURES IN CAL*
        U336DMP=YES,         DUMP DUE TO PGM CHECK-SPIE/STXIT    *
        U339DMP=NO,          DUMP CANCELLING DUE TO SORT FAILURE *
        VSAMER=NO,          |VSAM XRBA (8 BYTE RBA FEEDBACK AREA*
        WSTSIZE=1000,        SIZE OF WST AREA                    *
        ZEROPRT=NO           PRINT FIELD CONTAINS ZEROS/SUPPRESS
    END
```

Figure 1-1. QJOPTREC member

Note: It is recommended that you make changes and save the member as MYOPTION.

If for any reason you need to re-create the QJOPTION block, use CUSTMJCL, a JCL sample procedure, in the source sublibrary to assemble and link edit the QJOPTION block. JCL customization instructions are included in the procedure.

When the job successfully compiles, you should get a report similar to Report from the Test Run (alignment may vary slightly).

```
05/14/01    ACCOUNTS RECEIVABLE REPORT             PAGE    1

  ACCOUNT     CUST NAME          BALANCE          INSTL-BAL

  8006547 BO TORRES,ERNESTO        44.99              42.41
        1                          44.99              42.41
  6208657 EO CHO PYUNG,SUH         32.00             261.57
  7082509 EO S.F.MEM.HOSP.          5.00             272.78
  6002587 EO SANTA FE HOSP A       15.00           2,230.17
        3                          52.00           2,764.52
  6107265 FO GENVARDI,G J          43.00             419.48
  6095631 FO TODIPE,MICHAEL        45.24             486.21
  6123228 FO SILVA,JULIAN                            273.99
  6059708 FO CHAVEZ,RAY            15.00              49.39
  6044395 FO CANO,MICHAEL S        15.00             195.74
  6024963 FO HILL,GARY E            3.80              22.38
  8011508 FO HUGHES,RAY           178.70              47.88
        7                         300.74           1,495.07
  2002299 IO PLACIDO,ORTEGA       413.58             486.21
  6123317 IO VASGUEZ,IRENE        496.40
  6112536 IO CHAVEZ,NORMA A        55.00             496.84
  6009166 IO LOCKE,JEFFREY         15.00             234.91
  8012644 IO MONTEZ,CARMEN         89.28             484.75
  6218113 IO AGUIERA,EMILIO       108.44             197.85
        6                         681.30           2,396.96

       17                       1,079.03           6,698.96
 00000017 RECORDS FOR INZ FILE
```

Figure 1-2. Report from the Test Run (alignment may vary slightly)

Additional tests are in the source sublibrary. You must edit members before job submission.

Program Temporary Fixes (PTFs)

Punch or display the member ZAPQJPTF and review it. If there are PTFs, review the jobstream / instructions before submitting the job.

Customization Features

There are four VISION:Report customization features.

Customizing the Option Block

Option block customization is discussed briefly in the File 1: VISION:Report Object Sublibrary, VISION:Report Source/Macro Sublibrary, VISION:Report Sample Programs, Optional Material. A sample JCL jobstream is also available in the Source/Macro sublibrary under the member name CUSTMJCL.

Note: The option block phase, QUIKJOB1, should be linked as:

```
PHASE QUIKJOB1,S
```

Customizing the Edit Mask

See Chapter 2 for information on customizing the Edit Mask table.

Customizing QUIKDATE's Date Table (QUIKDATT)

Review the holiday date table before using QUIKDATE since the sample data may not apply to your installation. Dates may be converted for Year 2000 compliance by using the windowing technique. See Chapter 2 on customizing the QUIKDATE table, QUIKDATT.

Customizing Patches

The Sample Programs library may contain some customizing patches, all starting with the prefix of PCP (Program Customizing Patches). Study the comments in these members to determine if you must change VISION:Report for the customized functionality. Be sure you consider the caveats associated with each customizing patch before applying the patch.

Chapter 2

Installing VISION:Report Optional Material

Optional materials included in the libraries are:

- Assembler language source programs
- Macros
- VISION:Report sample programs and JCL, and installation JCL of optional features
- VISION:Report Interface to DB2 and SQL/DS with VISION:Report sample library
- Object decks

You can use these programs to enhance VISION:Report. You need not install all of the optional material for the successful operation of VISION:Report; however, it does allow for many more uses of VISION:Report.

Note: In VISION:Report documentation, the term "executable library" refers to the load library in MVS or the phase library in VSE.

This chapter describes customization, assembly, and linkage editing activities. In general, if the documentation for the particular program states that the program exists in an executable library, it need not be reassembled unless you want to change its options. Usually the default options satisfy most applications.

Since many of these Assembler programs are identical or similar under MVS or VSE, there may be references to either operating system. Most of the terminology is the same; there are some slight differences, such as a load library is used for MVS, while VSE uses a phase library; for documentation purposes the generic term executable library is used.

Note: If you are assembling on a VSE system that does not accept A.level macros, you must either reassemble with the EDECK option and catalog the object output into the source/macro library, or issue a COPY statement with the macro name(s) at the very beginning in the assembler input. For example:

```
COPY  QJEDIT
```

For VSE users the references to A.level and E.level (or format) source or macros are the equivalent of sublibraries A and E, respectively. Note that certain releases of VSE, in conjunction with the high-level Assembler, may no longer support the E.level macros. Only A.level source members were shipped.

You may want to assemble some of the Assembler source programs. Usually, comments at the beginning of the program define the symbolic parameters used to control how the program is assembled. Most of the programs contain code for both operating systems. Operating system-dependent code is generated only for the system indicated by the symbolic parameter (SYSPARM); thus, code for the other operating system would not be assembled. The SYSPARM parameters allowed are O (OS or MVS) or D (DOS or VSE).

As an example, to invoke the high level assembler, the EXEC statement would look something like this:

```
// EXEC PGM=ASMA90,PARM='SYSPARM(D),OBJECT'
```

Note: The PGM name can be ASSEMBLY. If your VSE system has only the high level assembler, it will automatically convert ASSEMBLY to ASMA90.

Link edit modules to your own executable library and concatenate other libraries ahead of the original release VISION:Report library, including any fix libraries that you may use later on. This way, the original modules are safeguarded against inadvertent corruption. There are very few exceptions to this general rule, such as the QJOPTION module.

The JCL can be placed in a PROC that is readily available to standardize JCL. It can also ease future migration problems. In this document, the assumption is that this has been done, and the PROC name is QJTEST.

After you test VISION:Report and are ready to implement this release for production, rename the PROC to a meaningful name so that the PROC name always points to the current production libraries. It could be as simple as QJPLIB.

A typical assembly jobstream for Optional Material source looks like this:

```
// JOB QUIKXXX ASSEMBLY
// EXEC   PROC=QJTEST
// OPTION CATAL
    PHASE QUIKXXX,*
// EXEC ASM90,PARM='SYSPARM(D),OBJECT'
   COPY QJEDIT
      .
      . YOUR QJEDIT STATEMENTS HERE
   END
/*
// EXEC LNKEDT
/&
```

Evaluate and change optional materials before their use.

Program Name	Routine or Program Function
DBOMPA	DBOMP Interface
DLIBGET, TLIBGET	CA-Librarian Interface Assistance
QJOPTION	Modify VISION:Report Options Block
QUIKDATE, QUIKDATT	Date Calculation
QUIKDLI	DL/I Interface (User must be licensed to use.)
QUIKDPRT	Print Date Table
QUIKIDMS	CA-IDMS/DB Access Interface (User must be licensed to use.)
QUIKISAM	ISAM Subroutine/Macro
QUIKRPT	Multiple Reports Subroutine
QUKBEMSK	Modify Edit Mask Table
QUKBTRN	Translate Table Changes
QUKBVEQU	Change EQUs for VAL Area
TOTAL INTERFACE	TOTAL Interface

DBOMP Interface

VISION:Report allows you to access your DBOMP data bases with seven macros that allow you to describe your database to VISION:Report. The module supports up to six files, two master files, and the associated product structure and/or routing files for each.

The generated module is approximately 4K, plus the size of the user I/O Root Phase, and will support sequential, generic, random, or implosions for your component master files. It also retrieves their associated routing records.

Master file updating is supported for each file independently, based on the parameters coded in the DEFM1 and DEFM2 macros.

The macro expansion generates an Assembler language routine ready to be cataloged into the executable library and called by VISION:Report at execution time.

Installing QJDBOMP

Note: You must customize your I/O Root Phase with the parameters PL/L=Yes and Overlay=nnnn where nnnn equals the size of IOPROCESS and all FILE DESCRIPTION modules required (see your IBM Marketing Representative for the appropriate documentation).

The macro names are: DEFM1, DEFM2, QJPS1, QJPS2, QJRT1, QJRT2, and IOGEN.

1. Select the Master file, Product structure file, or routing file for which you want to generate VISION:Report DBOMP support.

   Assume you want to produce support (a routine) to service a master file that has a product structure file chained to it.

   You need the following information:

   - Master file name as given to your I/O Root.
   - Master file key length.
   - Master file record size.
   - Name of user I/O Root phase.
   - FACA (First assembly component address displacement).
   - Product structure chain file name as given to I/O Root Phase.
   - Product structure file record size.
   - Next assembly component address displacement.
   - Component master record address displacement.

   You are now in position to prepare a job stream required to compile a module that will access the master file and the product structure file.

   The following example assembles the IBM DBOMP sample test file.

```
// JOB QJDBOMP
// EXEC ASSEMBLY
             CC16                                      CC72
QJDBOMP START 0
        DEFM1 FILE=DID$PNM,  File Name                   X
              KEYLEN=19,     User Key Length             X
              RECSIZE=100,   Work Area Size              X
              ROOT=DIDRS7,   IOROOT Phase Name           X
              FACA=23,       Displacement to TA$MFACA    X
              FROA=44,       Displacement to TA$MFROA    X
              UPDATE=YES     Update Logic Requested
        QJPS1 FILE=DID$PST,  Prod. Structure File        X
              RECSIZE=50,    Work Area Size              X
              NACA=5,        Displacement to TB$NACA     X
              CMRA=0         Displacement to TB$CMRA
        IOGEN
        END
/*
/&
```

2. Assemble the routine. Prepare Phase statements, and so on, and catalog to your executable library.

   The JCL-to-catalog routine might resemble the following:

```
// JOB CATALOG VISION:Report DBOMP ROUTINE
// EXEC PROC=QJTEST
// OPTION CATAL
   INCLUDE
 PHASE QJDBOMP,*
Object statements go here
/*
//  EXEC LNKEDT
/&
```

   Assuming you customized an I/O Root Phase and link-edited it, you are ready to write a small program to test your generated routine.

3. Write a sample program to sequentially retrieve your master or read part number cards from the card reader. Then randomly read them from the master.

   - Do a chain chase and print the component master part number and some of the information from the chain record.

4. Prepare a detailed description of the support available for each file or files for which you generated modules. Include the following:

   - Sample programs provided or include programs that you have prepared.
   - DBOMP subroutine call formats permissible for the master, structure, or routing files.
   - Call format for CLOSE if you generated or are going to be using update support for the master files.

Using MACROs

DEFM1 (define master File 1) and IOGEN (Input/Output Generator) are always required; the others are optional based on your requirements. The macros and their parameters are:

DEFM1 and DEFM2	Define Master Files (required parameters).
File= _____,	Master file name as given to I/O Root.
Keylen= _____,	Master file key length.
Recsize= _____,	Size of master record work area.
Root= _____,	Name of user I/O Root Phase. (Root parameter required for DEFM1 only.)

Optional Parameters

Faca= _____,	First assembly component address displacement relative to zero. Required if product structured retrievals are requested.
Froa= _____,	First routing operation address displacement relative to zero. Required if routing records are to be retrieved.
Update=(yes,no)	If update logic requested, default is no.
QJPS1 and QJPS2	Product structure chain file. All parameters are required. Generation of this macro is linked to the usage of the 'FACA' parameter of the associated master file definition.
File= _____,	Product structure chain file name as given in I/O Root Phase.
Recsize= _____,	Length of chain file work area.
Nac= _____,	Next assembly of component address displacement relative to zero.
Cmra= _____	Component master record address displacement relative to zero. There is no comma following the last parameter.
QJRT1 and QJRT2	Routine File descriptors. All parameters are required and generation is linked to the use of FROA parameter in associated master file definition.
File = _____,	Routing file as given in I/O Root Phase.
Recsize= _____,	Length of routing record work area.
Rnopa = _____	Routing next operation address displacement relative to zero. There is no comma following the last parameter.
IOGEN	Input/output generator. This macro has no parameters. It will provide the user's work areas and the coding necessary to access files defined to it by the preceding macros. If any errors occur in the preceding macros, they are MNOTEd and generation is ignored.

The job stream required to generate a module that accesses the DBOMP Sample Problem would be coded as:

```
//  JOB QJDBOMP
//  EXEC  PROC=QJTEST
//  EXEC ASSEMBLY
*               CC16                                      CC72
QJDBOMP  START 0
         DEFM1 FILE=DID$PNM,  File Name                    X
               KEYLEN=19,     User Key Length              X
               RECSIZE=100,   Work Area Size               X
               ROOT=DIDRS7,   IOROOT Phase Name            X
               FACA=23,       Displacement to TA$MFACA     X
               FROA=44,       Displacement to TA$MFROA     X
               UPDATE=YES     Update Logic Requested
         QJPS1 FILE=DID$PST,  Prod. Structure File         X
               RECSIZE=50,    Work Area Size               X
               NACA=5,        Displacement to TB$NACA      X
               CMRA=0         Displacement to TB$CMRA
         QJRT1 FILE=DID$MRT,  Routing File Name            X
               RECSIZE=125,   Work Area Size               X
               RNOPA=9        Displacement to SX$RNOPA
         IOGEN
         END
/*
/&
```

CA-Librarian INTERFACE ASSISTANCE

Note: Older releases of CA-Librarian® routines had a prefix of MAST, rather than FAIR (File Access Interface Routines).

To install the CA-Librarian interface package, you will need to obtain the FAIR routines from your CA-Librarian installation tape. There are several File Access Interface Routines**.** They are as follows:

Name	Function
FAIROPN	Open the master
FAIRMO	Position to a module
FAIRREC	Retrieve a record
FAIRCLS	Close the master

Some versions of FAIR are in source code. Some versions are in object code. Your objective is to achieve object code and catalog to your library using the above names.

Two routines have been prepared.

- DLIBGET to process disk masters.
- TLIBGET to process tape masters.

The call parameters and data returned to the users are identical. The only differences are as follows:

- The name of the routine (for instance, DLIBGET or TLIBGET),
- The JCL required.

Modify the TLIBGET and DLIBGET routines to specify the Logical unit assignment for your tape master, cycle file, disk master and file names. These parameters are located by the symbolic tag ARG3 in the TLIBGET and DLIBGET routines.

Consult the FAIR appendix in your CA-Librarian VSE System Reference manual.

Link-edit the two routines to your executable library; you will have to add JCL and control statements to do this.

VSE Phase statements users should be:

```
PHASE TLIBGET,S
PHASE DLIBGET,S
```

Consult your CA-Librarian VSE System Reference manual for the FAIR section JCL, assignment, and so on required to reference your master files.

Write a very simple program and test it until you are satisfied you have the routines installed and that they are functional. Write a simplified sample of how to use this program in your own installation.

Modifying VISION:Report Options Block (QJOPTION)

Note: The phase name for the QJOPTION block has been changed from QUKBOBLK to QUIKJOB1.

The options block is contained in the default name only. It should be placed in the same executable library where all VISION:Report executable modules reside.

Note: Be sure to place a comma (,) after each of the operands in the QJOPTION macro except the last, and code a non-blank in position 72, except the last.

When customizing the options block to fulfill your standards and requirements, prepare an Assembler language program that invokes the options block macro, QJOPTION. The OPTION statement contains a list of words defining the options with a list or range of valid values allowed and the default values assigned. (See the OPTION verb in the VISION:Report Reference Guide.) In the Source / Macro sublibrary, a member CUSTMJCL, can be modified to fit your requirements.

The name of the option is the name of the symbolic parameter used in the macro. All parameters are keyword parameters so they may be specified in any order. Your Assembler language program will look like this:

```
      CC16                                          CC72
QJOPTION SAVAREA=1500,                               X
      .  other                                       X
      .  options                                     X
      .                                              X
      STMTEND=71
END
```

When changes are made as in the above example (such as, changing the SAVAREA or allowing sequence numbers in VISION:Report statements by changing STMTEND), inform your users so that their programs will execute properly.

When executing the assembly and link edit steps, the appropriate libraries / sublibraries must be defined in your LIBDEF statements.

During the VISION:Report installation, several members of the Source / Macro sublibrary were restored to assist you in modifying the QJOPTION block. These members are CUSTMJCL, QJOPTDB2, QJOPTDEF, and QJOPTREC. (See the section Assembling QJOPTION Option Block and Installation Verification in Chapter 1.*)*

Calculating the Date (QUIKDATE Program and QUIKDATT Macro)

For the extended four-character year date processing, the user date tables must be customized to your needs **prior to** using the QUIKDATE routine. Some of the date functions may require data contained in the date table for proper execution.

The default date table name during execution of QUIKDATE is QUIKDATT, although this can be overridden at execution time. For documentation purposes, whenever the table name QUIKDATT is mentioned, you can substitute your table name.

The QUIKDATT macro must be assembled with your parameters to form QUIKDATT, even if you have QUIKDATT from a previous release of VISION:Report. Previous versions of QUIKDATT are not upward compatible. The macro consists of four areas of data to permit this execution:

- Windowing technique (Optional)
- Status of each of the seven days of the week
- Control information
- Holiday dates table

The Source / Macro sublibrary contains Assembler source modules.

Note: Keep the original MYDATT for future reference. Do not modify the source of QUIKDATT. If there is an error, contact Computer Associates Technical Support. See the section Contacting Computer Associates in Chapter 1.

1. You need to assemble QUIKDATT and customize your user date tables for your installation's requirements — an example is included in the MYDATT member.

   If you have a different member name other than MYDATT, you may replace MYDATT with that name in the assembler job stream. After you change MYDATT, assemble QUIKDATT with the MYDATT member following QUIKDATT.

2. To produce a clean Assembler listing of your customized table, the printing of the text of both macros has been disabled by the Assembler directive PRINT OFF. This directive should not be removed.

3. Begin your customization by modifying the sample statements that invoke the two macros in the required order. These statements follow the COMPUTER ASSOCIATES INTERNAL HEADER.

   The order of invocation of the macros is important.

   - The WEEKDAY macro must be invoked for each day of the week, beginning with the first day, which is Monday, and continuing consecutively (day 2, day 3, ..., day 7). The macro consists of two keyword parameters, DAY and DAYLEN. You must supply the day name, abbreviated to 3 characters (TUE, WED, ..., SUN). A default value (WHOLE) is supplied for the DAYLEN parameter. For a normal work day, you do not need to specify this parameter. The acceptable values are HALF (indicating the length of that workday is a half day) or OFF (indicating that no work is done on that day).

     For example:

     WEEKDAY DAY=MON

     WEEKDAY DAY=TUE

     WEEKDAY DAY=WED,DAYLEN=HALF

     WEEKDAY DAY=THU

     WEEKDAY DAY=FRI

     WEEKDAY DAY=SAT,DAYLEN=OFF

     WEEKDAY DAY=SUN,DAYLEN=OFF

   - After the WEEKDAY macro has been invoked as required, create the entries for your holiday date table. This is done with the HOLIDAY macro.

The macro contains five keyword symbolic parameters: DATE, DAYLEN, WKDAEND, CHAR2, and CHAR4. Default values assigned are:

DAYLEN=**WHOLE**

WKDAEND=**7**

CHAR2=**19**

CHAR4=**19**

Note: The DAYLEN parameters in the WEEKDAY macro and the HOLIDAY macro are unrelated.

4. The last three parameters are used only the first time the macro is invoked. If not specified at the time of the first invocation of the macro, default values are taken.

   These parameters and information summed from the DAYLEN parameter of the WEEKDAY macro are used to create the control information. They contain the normal work week length, the day (which designates the end of the work week), and two-part binary flags used to indicate century year prefix digits for two- and four-character date default values. The normal week length is obtained by summing the DAYLEN parameters of the WEEKDAY macro.

5. The WKDAEND (default = 7) may be set to a value of 1 through 7 designating Monday, Tuesday, ..., as the end of the work week. The CHAR2 and CHAR4 parameters may be assigned values of 19 or 20 and are used to specify the default values for the century when it is necessary to extend a two-character date to the four-character format. CHAR4 is used in Function 01 only to format the output data. CHAR2 is used in Functions 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11 to format the input dates. These defaults may be overridden by explicitly specifying a century as an optional parm or by the windowing option. For example:

   HOLIDAY DATE=01012002,DAYLEN=WHOLE NEW YEAR'S DAY

   HOLIDAY DATE=12242002,DAYLEN=HALF CHRISTMAS EVE

6. After the last HOLIDAY entry, type an END statement unless you are coding the optional WINDOW parameter or macro (Windowing Technique parameters for Year 2000); you must code the WINDOW macro after the last HOLIDAY macro, and it can only be specified once.

   When creating the holiday dates, remember the dates must be entered in ascending order. Dates are entered in the format MMDDYYYY, where:

   - MM represents the month (01=January, ... , 12=December)
   - DD represents the day (01, ... , 31)
   - YYYY is the four-character year

Your dates may regress as far back as 01011900 (January 1, 1900) and project as far ahead as 12312099 (December 31, 2099). Within the restrictions noted, you may indicate any combination of dates as holidays. Values, which you may use for the DAYLEN parameter, are WHOLE (default) or HALF. It is possible to have multiple tables. To do this, link edit each executable module under a separate name. The date table default name during execution is QUIKDATT, but if there are multiple date tables, the module name may be different.

Windowing Technique Parameters

QUIKDATE allows a 100-year interval windowing technique. There are two types of windowing techniques, fixed or sliding.

Note: If the Windowing Technique parameters are coded, then the Optional Parameter (for century) will not be valid or accepted for any function in QUIKDATE. Instead, the century that is returned is based upon QUIKDATE's calculations.

Note: Either technique above can only be applied to dates within a maximum 100-year period at any one time. This solution is temporary, as there is no guarantee that in the future, your applications will not expand to process dates that are more than 100 years apart.

The WINDOW macro is optional and can be coded only once, after the last HOLIDAY macro. If it is not coded, no windowing techniques will be used in QUIKDATE. The following are the WINDOW macro parameters.

FXYEAR	Fixed specific year. If this parameter is coded, then the windowing technique is fixed. This is a static 100-year interval that generally crosses a century boundary. This technique determines the century of a 2-digit year by comparing the hard-coded 2-digit year against a window of 100 years. If FXYEAR is omitted, then the system date year is obtained from the operating system and the windowing technique is sliding or rolling.
PASTYRS	This is the window for nn past years, relative to the FXYEAR if it is coded or relative to the system date year if FXYEAR is omitted. This parameter, along with +1 for the current year, plus FUTRYRS, must add up to a 100-year interval.
FUTRYRS	This is the window for nn future years , relative to the FXYEAR if it is coded or relative to the system date year if FXYEAR is omitted. This parameter, along with +1 for the current year, plus PASTYRS, must add up to a 100-year interval.

Only the FXYEAR parameter is optional; PASTYRS and FUTRYRS parameters must be coded. The FXYEAR parameter is coded only if you want the fixed windowing technique. The total of PASTYRS and FUTRYRS, +1 for the current year, must add up to 100 (year interval). As an example, if 1997 is the current year, PASTYRS could be coded as 35 and FUTRYRS would then be coded as 64.

```
PASTYRS +    FUTRYRS + 1 for current year = 100
35           64        1                 = 100
```

In the above example, the past 35 years would be 1962 until 1997, the system (or current) year; the future 64 years would be until 2061.

More than one date table, with different names, can be maintained at one time. Thus, you can have the normal QUIKDATT table module for your regular production work, while testing a Windowing Technique table. Furthermore, you can set one window to process historical dates, one for mortgage dates, one for birth dates, and so on, and QUIKDATE could automatically adjust the system date, with past and future windows to meet specific application needs.

Fixed Window Technique

The fixed window technique uses a static 100-year interval that generally crosses a century boundary. This technique determines the century of a 2-digit year by comparing the 2-digit year against a window of 100 years. The user specifies the number of years in the past and future, relative to a fixed or specific year within the 100-year interval.

As an example, if the years of date-related data of your application fall in the range of January 1, 1962 to December 31, 2061, you can use a 2-digit year to distinguish dates prior to and beyond the year 2000. If using the system year of 1997, the number of years in the past and future are coded as 35 and 64, in PASTYRS and FUTRYRS parameters, respectively, and FXYEAR would be coded as 1997. QUIKDATE then determines the century based on the following program logic and data checking, assuming xx is the two-digit FXYEAR:

- xx >60, it is a 20th century date (19xx).
- xx <59, it is a 21st century date (20xx).

The inherent future risk in using this technique is obvious, since you would need to adjust this program or parameter each year.

Sliding Window Technique

The sliding window technique uses a self-advancing 100-year interval that generally crosses a century boundary. This technique determines the century of a 2-digit year by comparing the 2-digit year against a window of 100 years. The user specifies the number of years in the past and future relative to the system year (generally the current year) that the system sets and maintains. QUIKDATE can access the date that the system sets and automatically advances. The principal advantage of this method over that of using a fixed window is that the fixed window is immovable without manually changing the table each year.

As an example, if the years of date-related data of your application fall in the range of 35 years in the past and 64 years into the future and coded as 35 and 64, in PASTYRS and FUTRYRS parameters, respectively, based upon the year, 1997, QUIKDATE can accept and accurately deal with dates of 1962 through 2061.

- Past 35 years would be 1962 until 1997, the system year
- Future 64 years would be until 2061.

Both the fixed and sliding window techniques have their inherent risks. You must carefully weigh the pros and cons of both techniques before deciding which technique, if any, to use. For example:

```
WINDOW  PASTYRS=35,FUTRYRS=64                  Sliding Technique
WINDOW  PASTYRS=35,FUTRYRS=64,FXYEAR=1997      Fixed Technique
```

Link Editing MYDATT

Link edit MYDATT as Phase QUIKDATT to an executable library and use a PROC to concatenate the required libraries and JCL in order to run VISION:Report.

JCL to Assemble MYDATT shows a sample JCL job stream to assemble MYDATT, as well as the contents of MYDATT. The PROC QJTEST defines the libraries required to assemble and link edit QUIKDATT. For your convenience, the JCL jobstream is also included as ASMYDATT in the Source/Macro library.

Be sure to change MYDATT if this is not the member name that contains your installation's QUIKDATE table. Note that the name, QUIKDATT, will be the phase name, and it is the default date table name for QUIKDATE.

```
// JOB QUIKDATT  ASSEMBLY
// EXEC   PROC=QJTEST
// OPTION CATAL
    PHASE QUIKDATT,*
// EXEC ASM90,PARM='SYSPARM(D)'
    COPY QUIKDATT          Keep this statement here, do not delete.
         .
         .  YOUR WEEKDAY AND
         .  HOLIDAY STATEMENTS HERE
     or
        COPY MYDATT ??????   YOUR member-name
        END
/*
// EXEC LNKEDT
/&
```

Figure 2-1. JCL to Assemble MYDATT

DL/I INTERFACE

Note: You must use an LMP Code that supports QUIKDLI.

The QUIKDLI interface is DL/I release independent. No compiling is required. QUIKDLI comes to you in object code ready to link edit and catalog to your executable library. The link edit process for VISION:Report and QUIKDLI requires less than five minutes.

QUIKDLI requires the DL/I routine called DLZLI000 (commonly called CBLTDLI, ASMTDLI and/or PLITDLI) be resident in one of your object libraries. This is not a special item since all of your current programs require it also.

JCL

You execute DL/I (as you currently do) and specify QUIKDLI as the application program to be executed under DL/I in your DL/I parameter statement. For example:

```
// JOB QUIKDLI
* Data base assignment, TLBLs,
* DLBL/EXTENTs normal to your DLI execution
* and VISION:Report libraries
* Any VISION:Report assignments, TLBLs,
* DLBL and EXTENTs required.
// EXEC DLZRRC00,SIZE=nnnK
DLI,QUIKDLI,PSBName
000 CALL QJBTDLI, etc.
(VISION:Report Statements and Calls go here)
/*
(Card input to VISION:Report (if any) goes here)
/*
/&
```

Installing QUIKDLI

QUIKDLI is 100 percent compatible with QUIKIMS, so if you switch from VSE to MVS or vice-versa, your IMS/DL/I interface is the same.

QUIKDLI object is located in VISION:Report Library.

Linking DL/I

You will need JCL to link QUIKDLI. Sample JCL follows:

```
// DLBL  QJLIB,etc.VISION:Report Library
// EXTENT,volser
// DLBL DLILIB,etc. your DL/I Library
// EXTENT,volser
// LIBDEF  *,SEARCH=(QJLIB.QJnnn,DLILIB.????),CATALOG=QJLIB.QJnnn
// OPTION CATAL
  PHASE  QUIKDLI,S
   INCLUDE  QUIKDLI
   INCLUDE  ASMTDLI
/*
// EXEC LNKEDT
```

where

nnn = Release Number

Printing the User Date Table (QUIKDPRT)

VISION:Report contains a batch utility program, QUIKDPRT, that prints the contents of the user date table in a report format.

The user date table report is produced in two contiguous parts. The first part prints the weekday and miscellaneous sections of the table. This details the defined length of each day, the day designated as the end of the week, the normal work week length, and default prefixes for two-character year format date values. The remainder of the report is a listing of the entries in the holiday dates portion of the table (vertically, in a five column display). The first page of this listing may contain up to 80 entries; subsequent pages may contain up to 125 entries. The entries on the last page appear in balanced or even-column length.

Note: Do not call QUIKDPRT using VISION:Report statements.

QUIKDPRT is designed to execute alone, in batch mode. VSE statements may be required to define the desired library if multiple tables exist. The output is directed to the system logical unit SYSLST in VSE.

To route the output to a tape or disk device, provide the JCL statements to define the output data set. The statement initiates execution:

```
//  EXEC PGM=QUIKDPRT
```

Modifying the Edit Mask Table (QJEDIT macro)

- Default name for VSE: QUKBEMSK

The default edit mask table is provided in the object, executable, and in the source libraries. If you want to customize the edit mask table, you need to reassemble and catalog it into the appropriate libraries. You should copy this module and retain the original in case you want to create another edit mask table.

This module invokes the VISION:Report macro QJEDIT to create the entries for your specific edit masks. It is possible to have multiple edit mask tables. To do so, create each item with a unique link edit name and use the OPTION statement with the EDTNAME=xxxxxxxx operand, where xxxxxxxx is the name of the link edited mask table.

The first invocation of the macro allows you to name the CSECT, which will have the default name if the name field is not specified in the first invocation. You may choose to match the CSECT name and the link edit name. In any case, notify users of the customization results. (See the VISION:Report Reference Guide for more information about QJEPRNT which lists edit mask tables.)

The syntax of the QJEDIT macro is:

```
name    QJEDIT CODE=code,                              X
              MASK=(mask),                             X
              FORCE=force,                             X
              JUSTIFY=(leftright,truncopt,propatopt)
```

name	This may be used only in the first invocation of the macro. This is the name given to the CSECT generated by the macro. If not specified, the default value is QUKBEMSK for VSE. This name is also the name that should be given when it is processed by the linkage editor and placed into your executable library.
code	This is any letter from A to Z that identifies this mask. Codes C, E, H, and N are reserved by VISION:Report and cannot be used. If duplicate codes are specified in more than one QJEDIT macro, only the first one is found by VISION:Report.

mask	This is the actual edit mask pattern. Nine supplied patterns make up the VISION:Report default table name. Refer to the MOVE verb (see the VISION:Report Reference Guide) for a description of these default masks. The edit mask must be enclosed in parentheses and must not exceed 132 characters. It consists of data selector characters and punctuation or edit fill characters. It may not exceed 63 data selector characters.

Establish a mask table for two types of product codes. As an example, name the table QPCMASK and store it in the phase library with that name.

Copy the module from the optional materials. Modify the first invocation of the macro to have the CSECT name QPCMASK.

```
QPCMASK   QJEDIT CODE=A,MASK=(999/99),                X
              JUSTIFY=(RIGHT,TRUNC,PROPGAT),FORCE=YES
```

Before the Assembler END statement, insert your edit code masks (they may be inserted in alphabetical order).

```
QJEDIT CODE=N,MASK=(98(999)BZZZ9-999-999),            X
              JUSTIFY=(RIGHT,TRUNC,NOPROPGAT),FORCE=NO

QJEDIT CODE=O,MASK=(9.999/9999-999-999-9),            X
              JUSTIFY=(RIGHT,TRUNC,NOPROPGAT),FORCE=NO
```

If you want to delete or change any of the default codes, you may do so now. See sample Assembly JCL at the end of this section.

The link edited name chosen must be a unique 1 through 8-character name that will reside in your executable library. (To conform to the VISION:Report naming standards, it is recommended that the name begin with a Q.)

To use this new edit mask table phase, your VISION:Report must use OPTION EDTNAME=username. This may be done permanently by recompiling the QJOPTION block or including the OPTION EDTNAME=username in the VISION:Report program that references this new table.

Only one edit mask table may be used in one execution of VISION:Report.

JCL Sample to Assemble and Link Edit Mask Table

```
// JOB QJEDIT MASK TABLE COMPILE & LINK
// EXEC PROC=QJTEST
// OPTION NODECK,CATAL
   PHASE username,S
// EXEC ASSEMBLY
   your edited QJEDIT statements
   END
/*
// EXEC LNKEDT
/&
```

CA-IDMS/DB ACCESS INTERFACE

Note: You must use the LMP key that supports QUIKIDMS.

Note: Remove REPRO and INCLUDE statements as required, or comment them out by placing an asterisk (*) in position 1. This is based upon whether you need the Local and / or Central version of CA-IDMS® / DB.

QUIKIDMS VSE Installation

Run the following job specifying the amount of partition storage to be allocated for QUIKJOB in the parameter “nnn”. For a local CA-IDMS / DB, include IDMS and IDMSCANC. For a central version, CA-IDMS / DB also includes IDMSOPTI and IDMSLDPT. Change library names and sublibraries as appropriate in your company.

```
// JOB     QJIDMSGN             GENERATE IDMS INTERFACE
// DLBL    QJLIB,'your.qj.lib'
// EXTENT     ,volser
// DLBL    IDMSLIB,'your.idms.lib'
// EXTENT   ,IDMSVL
// LIBDEF    *,SEARCH=(QJLIB.Qnnn,IDMSLIB.???),CATALOG=QJLIB.QJnnn
// OPTION   CATAL,NODECK
 PHASE QUIKIDMS,S
// EXEC ASSEMBLY
       REPRO
 INCLUDE QUIKIDMS
QUIKJOB    CSECT
       DS   nnnCL1024           “nnn” is number of K
HICORE     CSECT
       DC   c'XX'
       REPRO            for local & central IDMS/DB
 INCLUDE IDMS                          for local & central IDMS/DB
       REPRO            for local & central IDMS/DB
 INCLUDE IDMSCANC                            for local & central IDMS/DB
       REPRO            for central IDMS/DB only
 INCLUDE IDMSOPTI                            for central IDMS/DB only
       REPRO            for central IDMS/DB only
 INCLUDE IDMSLDPT                            for central IDMS/DB only
       END
/*
// EXEC LNKEDT
/&
```

The member names and a description of their use are as follows:

Source/Maclib	Description
Q.SSCTRL	VISION:Report EQU Statements used in conjunction with Interface

Object Library	Description
QUIKIDMS	Object Code Modules

QUIKISAM Macro

This macro can be used to generate a callable subroutine for any user ISAM file. The subroutine may be called by Assembler, COBOL, or VISION:Report.

The generated subroutine is capable of random retrieval by key, updating the retrieved record, adding new records, or adding new records to an existing ISAM file.

The generated subroutine is about 6200 bytes. This includes the IJHBARZZ ISAM logic module size. You must have ISAM module IJHBARZZ in your object library prior to compiling and link editing the generated QUIKISAM subroutines or you may have to generate an IJHBARZZ object module prior to linking your QUIKISAM routine.

The functions that can be performed are as follows:

RANDOM RETRIEVAL	You must specify an area to retrieve records into and a key (same length as key on file) of the record to be retrieved. If the record is not found, the subroutine passes high-values back in the area where the record would have been placed if a find had occurred.
UPDATE	You must have read the record first. Be careful not to change the key itself. You must specify the beginning location of the record to be updated and a parameter specifying C'UPDATE'.

ADD A RECORD	You must provide the record to be added by specifying its location as a parameter, and a parameter specifying C'ADD'. Duplicates will not be added. A parameter is provided to allow you to receive a code back if the record you tried to add was a duplicate.

Preparing a QUIKISAM Subroutine

The necessary macros are already cataloged into the Source/Macro sublibrary. Their names are QUIKISAM and QTYP.

1. Gather the pertinent information about the ISAM file to generate a QUIKISAM subroutine. The data you will need about the file is:
   - Record length.
   - Number of records in a block.
   - Key length.
   - Key location. (Specify location of 1 if the key is the first element of the record; specify 1 if the file is unblocked.)
   - Number of overflow tracks per cylinder.
   - If you have standard label in the standard label track for this file, find the file name being used in the // DLBL.
   - Pick a name by which the subroutine will be referred by your programmers. This name generates a CSECT name for your subroutine. This name is also placed in the PHASE statement for link-editing into your executable library.

     The name may be 1 to 8 alphanumeric characters. The first character must be alphabetic. For the first four positions, use common characters; use something symbolic in the last four positions. If you keep the first four characters the same, you will be able to identify all QUIKISAM generated routines readily.
2. Code, compile and catalog the generated program into your executable library.
3. Be sure to place a comma after each of the operands in the QUIKISAM macro except the last, and code a non-blank character in position 72, except the last.

Your QUIKISAM subroutine is now ready to be used. You may execute it by observing the CALL format rules. See CALL FORMATS.

Details about the QUIKISAM Operands

PHASE	Although this is not an operand of the macro, it is the name by which the subroutine will be cataloged into the executable library. It is the name to be used as operand 1 of your call statements
KEYLOC	Specify the location of the key within your records. If the key starts in position 1, the value to use is KEYLOC=1. For unblocked records, use KEYLOC=1.
KEYLEN	Enter the key length for the ISAM file.
DEVICE	Enter one of the following numbers for the kind of device the file is on: 2311, 2314, 3330, 3340. Note that ISAM files are not supported on devices other than the ones specified.
RECSIZE	Enter the length of a logical record.
NRECDS	Enter the number of records per block. The input/output areas are calculated by using key length, recsize, and number of records.
PHNAME	This is used to name the control section within the generated subroutine. Specify the same name here as was used for the phase name.
CYLOFL	When the file was generated, some number of tracks per cylinder were reserved for overflow upon each cylinder. Enter that number.
FILNAME	This name is used as the DTF name and this is the name that is used for the DLBL Filename.
TYPE	ALL - Yields a module that retrieves, updates, and adds. UPDATE - Yields a module that retrieves and updates. RANDOM - Yields a module that supports Random Retrieval only.

Note: Be sure to place commas after all operands, except the last one.

```
// JOB QUIKISAM
// EXEC PROC=QJTEST
// OPTION CATAL,NODECK
// EXEC ASSEMBLY
     REPRO
     INCLUDE
     REPRO
     PHASE ________,S      [Your phase name goes here]
           CC16                                              CC72
    QUIKISAM                                                   *
           KEYLEN=  KEYLENGTH GOES HERE                        X
           KEYLOC=  LOCATION OF KEY IN RCD,RELATIVE TO 1       X
           DEVICE=  DISK DEVICE 2311, 2314, 3330, 3340         X
           RECSIZE= SIZE OF ONE LOGICAL RECORD                 X
           NRECDS=  NUMBER OF RECORDS IN A BLOCK               X
           PHNAME=  THE NAME TO BE CALLED,SAME AS PHASE ABOVE  X
           CYLOFL=  NR OF OVERFLOW TRACKS PER CYLINDER         X
           FILNAME= DTF NAME  /  DLBL FILENAME                 X
           TYPE=    TYPE ROUTINE (ALL, UPDATE OR RANDOM)
    END
/*
//  EXEC LNKEDT
/&
```

Figure 2-2. Coding Sheet Example

The following ISAM logic modules should be in your object library to support the type of routines you may choose to generate with QUIKISAM.

ROUTINE TYPE	FILE TYPE	MODULE REQUIRED
Random Retrieval with or without or update	Blocked files or Unblocked file	IJHZRRZZ
Random Retrieval update ADD records	Unblocked files	IJHUARZZ
Random Retrieval Update ADD	Blocked	IJHBARZZ

Multiple Reports Subroutine (QUIKRPT)

QUIKRPT is provided in the object, executable, and the source library. If you want to customize the multiple reports subroutine, source code for the default subroutine is included in the optional materials. You should copy module QUIKRPT, retaining the original for future reference.

The executable module contains the maximum number of report files this program supports — 7. You may choose to support other than 7 report files, which requires that the program be reassembled and link edited into your VISION:Report executable library.

VSE Restrictions

VSE DTF and PRMOD macros are coded for 1403 devices using 133 character print positions. For installation with a different configuration, you may have to alter the program macros to conform with actual device type used. Another prerequisite to using the program feature is a system using POWER/VSE spooling (or comparable package), with sufficient number of printers allocated to support the number of different reports specified to QUIKRPT.

For a VSE system, the DTFPR is assembled. The operating system under which the program will execute is passed to the assembly by the SYSPARM parameter. To invoke the high level assembler, the EXEC statement would look something like this:

```
// EXEC PGM=ASM90,PARM='SYSPARM(D),OBJECT'
```

Comments at the beginning of the program define the symbolic parameters used to control how the program is assembled. The following instruction in the source sets the number of reports:

```
&REPTS  SETA  7
```

You can specify a value between 1 and 7. The symbolic parameter &BEGSYS is used for VSE only, and represents the starting SYS number for each printer.

Translate Table Changes (QUKBTRN)

■ Default name for VSE:	QUKBTRN
■ Default name for MVS:	QUIKTRNT

The default translate table is provided in the object, executable, and source libraries. It contains the Assembler source statements used to create the VISION:Report default translate table phase. This table is used as the object of the translate function for the data displayed in the PRINTHEX or PRINTCHAR statement.

The program contains two 256-byte tables. The first table translates data displayed in the first (character) line of a PRINTHEX or PRINTCHAR output. The second table translates data in the second (zone) and the third (numeric) lines of a PRINTHEX output.

This program phase optionally may be changed or used as a guide in generating an alternate translate phase. The changes / alterations should be made only to the first (character) table. The second table should not be altered since the results of the HEX character display will be unpredictable.

Execution of PRINTHEX or PRINTCHAR uses the existing tables with changes made by the user. If an alternate table is built, you must identify the alternate table to VISION:Report through OPTION TRLNAME=user-table-name.

User table name is the name used when the alternate table was link-edited into the executable library, and must not exceed 512 bytes in length when assembled. When you link-edit the table into your VISION:Report executable library with its new name, be sure to inform all users of its name. You may also describe the effects of the resulting translation.

EQU Table for the VAL Area

■ Default name for VSE:	QUKBVEQU
■ Default name for MVS:	QUIKVEQU

The default EQU table for the VAL area is provided in the object, executable, and source libraries. If you want to customize the EQU for VAL table, you should copy this module, retaining the original in case you want to create another EQU for VAL table.

A table of EQU statements for the VAL area is automatically loaded during the compile phase.

The data names for the VAL areas, their field definitions, and any edit specifications are described under the EQU verb in the VISION:Report Reference Guide.

These EQU statements may be changed or augmented. To do so, copy the default program name from the assembler and macro library and make your changes. Be aware of the restrictions and comments at the beginning of the program. These comments describe the components of an entry and how it is constructed.

After assembly and linkage into your VISION:Report executable library, you may run the following VISION:Report program to print the data names for your new table. The subroutine, QJADDR, places an address into working storage. Choose either statement #010.

```
      TITLE 'VAL EQUATE TABLE LISTING'
REPORT PTA1-14     (DATA NAME)
       PTA15-29    (FIELD DEFINITION)
       PTA30-33    (EDIT SPECS)
EQU  LAST-EQU     WST1-4-B    /* ADDRESS OF LAST-EQU STATEMENT
EQU  THIS-EQU       (4)-B     /* CURRENT ADDRESS
010  LOAD QUKBVEQU  PTA       /* VSE Load statement
     MOVE PTA9-12-B        TO LAST-EQU
     SET PTA  UP 24            /* SKIP CONSTANTS IN FIRST 24 BYTES
020  PRINT REPORT
     CALL QJADDR  PTA1-4-B  THIS-EQU  /* PLACE ADDRESS IN THIS-EQU
     IF THIS-EQU EQ LAST-EQU  /* ARE WE AT THE END OF EQU TABLE?
        GOTO EOJ.             /* ... YES
     SET PTA  UP 80           /* BUMP TO NEXT EQU
     GOTO 020
999999END
```

TOTAL INTERFACE

Two examples of statements are available for your use to compile, link, and catalog a VISION:Report callable subroutine to access a TOTAL database.

The examples shown install easily and quickly. The statements include DATBAS, and assemble a core storage area reserved for TOTAL usage immediately following DATBAS.

Consult the TOTAL User's Guide in the section entitled VSE JCL-Core Allocation Considerations for more information.

TOTAL JCL Assembly Example

```
// JOB QJDATBAS QJ/TOTAL - SUBROUTINE ASSEMBLY & CATALOG
// EXEC PROC=QJTEST
// OPTION CATAL,NODECK
   PHASE QJDATBAS,S,NOAUTO         USER CALL NAME
   INCLUDE DATBAS      TOTAL CALL ROOT MODULE (OR DATBAS7)
// EXEC ASSEMBLY,SIZE=256K
DATCORE  START 0       DUMMY AREA RESERVED FOR TOTAL
         DS   CL65536           64K MEMORY FOR TOTAL MODULES -
         END                        (MAY HAVE TO BE LARGER OR SMALLER)
/*
// EXEC LNKEDT
/&
```

Figure 2-3. VSE JCL Example

Chapter

3

Installing VISION:Forms Under VSE

This chapter describes the installation of this release of VISION:Forms for VSE. This is a fix release. Install this release to keep your maintenance current. Earlier releases of VISION:Forms will not work with the previous release of VISION:Report.

To install the VISION:Forms library, run the QWINSTL job. The member QWINSTL.A is located in sublibrary QJ161 of the library created by running QJINSTL. You will need to edit QWINSTL.A and submit the member for processing.

If you do not have a VISION:Forms Dictionary from a previous release, you must create one. To do so, use the member DICTCRTE.A located in sublibrary QW308 of the library created by running QWINSTL.

You can then run the DICTLOAD job to load dictionary entries for the Installation Verification Procedure (IVP) job, QWDEMOS.

The installation and execution of this VISION:Report release and VISION:Forms release supports only VSE/SP 2.1.1 and above. This is the VSE release that IBM restructured the VSE library organization, using the VSE program, LIBR. If you are on a release prior to VSE/SP 2.1.1, there is no guarantee that VISION:Report or VISION:Forms will work, even if you are able to modify the JCL and control cards and successfully install either product.

Installation VISION:Forms

Complete the following steps to install VISION:Forms on a VSE system:

1. If you have not already done so, install VISION:Report. (See Chapter 1.)
2. Punch out and edit QWINSTL from the restored VISION:Report library and change the values, as required.

   For the VISION:Forms library, you will need to determine the dataset name or fileid, volume serial number, starting tracks, number of tracks allocated, tape device addresses, and so on. The simplest way is to make global replacements so that all occurrences of certain character strings will be replaced throughout the entire member.

```
* $$ JOB JNM=QWINSTL,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB QWINSTL  RESTORE OPTIONAL VISION:FORMS LIBRARY
// OPTION LOG,NODUMP
* STEP1: DEFINE LIBRARY
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*        ',USERID' TO: IF UNDER VM, YOUR CMS USERID
/*        '**$$'    TO: '* $$' IF UNDER VM
/*        '@@@VLL'  TO: VOLUME SERIAL NUMBER OF LIBRARY
/*        '@@@TRL'  TO: STARTING TRACK OF QW LIBRARY
/*        '@@@CCC.@@@DDD.LIB' TO: QW LIBRARY NAME
/*        '@@TAPE'  TO: ORIGINAL DISTRIBUTION QW TAPE DRIVE
/*
// ASSGN  SYS020,DISK,VOL=@@@VLL,SHR        QW LIBRARY
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366  QW LIBRARY
// EXTENT SYS020,@@@VLL,1,0,@@@TRL,125
// EXEC   LIBR,SIZE=512K
   DEFINE     LIB=QWLIB            REPLACE=YES
     DEFINE   SUBLIB=QWLIB.QW308   REPLACE=YES  REUSE=IMMEDIATE
     DEFINE   SUBLIB=QWLIB.PTF     REPLACE=YES  REUSE=IMMEDIATE
/*
* STEP2: RESTORE LIBRARY
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366    3.08 LIBRARY
// EXTENT       ,@@@VLL
// LIBDEF  PHASE,SEARCH=(QWLIB.QW308),                                  X
              CATALOG=QWLIB.QW308
// LIBDEF SOURCE,SEARCH=QWLIB.QW308
// LIBDEF    OBJ,SEARCH=QWLIB.QW308
// PAUSE  MOUNT DISTRIBUTION TAPE ON @@TAPE DRIVE
// MTC    REW,@@TAPE               ENSURE TAPE AT LOAD POINT
// MTC    FSF,@@TAPE,4             SKIP TO SECOND FILE
// ASSGN SYSIPT,@@TAPE
// EXEC   LIBR,SIZE=512K,PARM='ACC SUB=QWLIB.QW308'
/*
* STEP3: LINKEDIT MODULES
// RESET  SYSIPT
// OPTION CATAL
   INCLUDE PWLNKALL
// EXEC LNKEDT
* NOTE: YOU MAY GET RC=4; THIS IS FINE
/&
* $$ EOJ
```

After customizing the jobstream, run job QWINSTL. QWINSTL consists of several steps or phases. Upon successful completion, you will have a library that contains Phase, Object, and Source/Macro sublibraries for VISION:Forms.

The object sublibrary contains object modules which was processed by the linkage editor to create Phase modules. They contain references to other modules, usually acting as an interface between VISION:Forms and other programs.

The Source/Macro sublibrary contains some demo or Installation Verification tests. It may also contain Program Temporary Fixes (PTFs), which start with the prefix of PTF, as well as any Program Customizing Patches (PCPs), which start with the prefix of PCP, to assist you in upward compatibility or special environments. PCPs should be reviewed to ensure that they are applicable to your installation.

3. Punch or display the member ZAPQWPTF and review it. If there are Program Temporary Fixes (PTFs), review the jobstream/instructions before submitting the job.

4. At this point, you can to run some installation verification tests to ensure that VISION:Forms has been installed correctly. From the Source/Macro library just restored, read member @INDEXQW.A, which describes what the various test member names are, and what sequence the tests should be run. Within each member, there are comments on what changes are required in order to run the tests.

DICTCRTE

```
* $$ JOB JNM=QWDICTCR,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB QWDICTCR - CREATE VSAM DICTIONARY FOR QW
// OPTION LOG,NODUMP
/*
/*  CREATE VSAM DICTIONARY
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*
/*        ',USERID'           TO: IF UNDER VM, YOUR CMS USERID
/*        '**$$'              TO: '* $$'
/*        '@@@VLL'            TO: VOLSER OF QJ LIBRARY
/*        '@@@AAA.@@@BBB.LIB' TO: QJ LIBRARY NAME
/*
/*        '@@@VLF'            TO: VOLSER OF QW LIBRARY
/*        '@@@CCC.@@@DDD.LIB' TO: QW LIBRARY NAME
/*        '@@@CCC.@@@DDD.DICT'TO: QW DICTIONARY NAME
/*        'SYSWK1'            TO: VOLSER OF A WORK PACK
/*        'VSAM01'            TO: VOLSER OF A VSAM CONTROLLED PACK
/*        'VOLSER'            TO: VOLSER OF DEMO FILE
/*
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    16.0 LIBRARY
// EXTENT       ,@@@VLL
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366    3.07 LIBRARY
// EXTENT       ,@@@VLF
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                          X
               QWLIB.PTF,QWLIB.QW308)
// LIBDEF SOURCE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                          X
               QWLIB.PTF,QWLIB.QW308)
// LIBDEF    OBJ,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                          X
               QWLIB.PTF,QWLIB.QW308)
// DLBL QW$DICT,'@@@CCC.@@@DDD.DICT',,VSAM
// DLBL     OFA,'@@@CCC.@@@DDD.DICT',,VSAM
// ASSGN SYS001,DISK,VOL=SYSWK1,SHR
// ASSGN SYS010,DISK,VOL=SYSWK1,SHR
// ASSGN SYS011,DISK,VOL=SYSWK1,SHR
// ASSGN SYS016,DISK,VOL=SYSWK1,SHR
// DLBL SORTWK1,'QWDEMO.STDSORT.WORK',0
// EXTENT SYS001,SYSWK1,1,0,001,38
// EXEC IDCAMS,SIZE=AUTO
  DELETE @@@CCC.@@@DDD.DICT CLUSTER
  SET MAXCC=0
```

```
  DEFINE CLUSTER( -
  NAME(@@@CCC.@@@DDD.DICT) -
  FILE(QW$DICT) -
  VOLUMES(VSAM01) -
  RECORDS(100,300) -
  SHAREOPTIONS(4) -
  FREESPACE(10 10) -
  SPEED -
  KEYS(25 0) -
  RECORDSIZE( 80 80))  -
  INDEX ( NAME (@@@CCC.@@@DDD.DICT.INDEX) )
/*
// EXEC QWDMAINT,SIZE=AUTO
:CREATE QW$DICT
:BEGIN MEDIA=ALL
SELECT MEDIA=DISC,SYSNR=010
REPORT MEDIA=DISC,SYSNR=011
:BEGIN JCL=ALL,USAGE=SOI
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY    SOI
// EXTENT       ,@@@VLL
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366    QW LIBRARY
// EXTENT       ,@@@VLF
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QW308)
// LIBDEF SOURCE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF    OBJ,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// ASSGN SYS002,DISK,VOL=SYSWK1,SHR
// DLBL SORTIN1,'QWDEMO.STDSORT.INPUT',0
// EXTENT SYS002,SYSWK1,1,0,001,30
// OPTION NODUMP
:BEGIN JCL=ALL,USAGE=SOO
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY    SOO
// EXTENT       ,@@@VLL
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366    QW LIBRARY
// EXTENT       ,@@@VLF
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF SOURCE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF    OBJ,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// ASSGN SYS001,DISK,VOL=SYSWK1,SHR
// DLBL SORTOUT,'QWDEMO.STDSORT.OUTPUT',0
// EXTENT SYS001,SYSWK1,1,0,001,30
// OPTION NODUMP
:BEGIN JCL=ALL,USAGE=SOW
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY    SOW
// EXTENT       ,@@@VLL
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366    QW LIBRARY
// EXTENT       ,@@@VLF
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF SOURCE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF    OBJ,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// ASSGN SYS003,DISK,VOL=SYSWK1,SHR
// DLBL SORTWK1,'QWDEMO.STDSORT.WORK',0
// EXTENT SYS003,SYSWK1,1,0,001,30
// OPTION NODUMP
:CREATE PAYROLL
TITLE='PAYROLL DEMO DICTIONARY'
DEVICE=DISC
BLKL=80
```

```
RECL=80
UNIT=002
:BEGIN RECORD
FIELD NAME=PLANT-NR,LEN=2,HDR='THE,PLANT,NUMBER'
FIELD NAME=DEPT-NR,LEN=3,HDR='DEPARTMENT,NUMBER'
FIELD NAME=EMPLOYEE-NR,LEN=4
FIELD NAME=NAME,LEN=16,HDR='NAME,OF,THE,EMPLOYEE'
FIELD NAME=DT-OF-BIRTH,LEN=6
FIELD NAME=DT-OF-EMPLYMT,LEN=6
FIELD NAME=EDUCATION,LEN=2
FIELD NAME=SKILL-CODES,LEN=6
FIELD NAME=HOURLY-RATE,LEN=4,CALDEC=3,PRTDEC=3
FIELD NAME=HOURS-WORKED,LEN=4,PRTDEC=2,CALDEC=2,
   HDR='NUMBER OF,HOURS WORKED'
FIELD NAME=PREV-YTD-GROSS,LEN=4,TYPE=P,CALDEC=2,PRTDEC=2,EDIT=C
FIELD NAME=YTD-ST-TAX,LEN=4,CALDEC=2,PRTDEC=2,TYPE=P
FIELD NAME=YTD-FICA,LEN=5,CALDEC=2,PRTDEC=2
FIELD NAME=YTD-PAY-UNPK20,LEN=5,CALDEC=2,PRTDEC=2
FIELD NAME=YTD-PAY-PK21,LEN=5,TYPE=P,CALDEC=2,PRTDEC=2,START=67
FIELD NAME=CARD-CODE,LEN=1,START=80
:BEGIN JCL=ALL,USAGE=SEL
// DLBL     INF,'QW.DEMO.PAYROLL'           SEL
// EXTENT SYS010,VOLSER,1,0,001,5
// ASSGN SYS010,DISK,VOL=VOLSER,SHR
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY
// EXTENT       ,@@@VLL
// DLBL   QWLIB,'@@@CCC.@@@DDD.LIB',99/366    QW LIBRARY
// EXTENT       ,@@@VLF
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF SOURCE,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// LIBDEF    OBJ,SEARCH=(QJLIB.PTF,QJLIB.QJ161,                        X
               QWLIB.PTF,QWLIB.QJ308)
// ASSGN SYS002,DISK,VOL=SYSWK1,SHR
// DLBL SORTIN1,'QWDEMO.STDSORT.INPUT',0
// EXTENT SYS002,SYSWK1,1,0,001,30
// OPTION NODUMP
:CREATE QWDICT
TITLE='QUIKWRITE DICTIONARY FILE'
DEVICE=KSDS
BLKL=80
RECL=80
UNIT=002
LBL=QW$DICT
:BEGIN RECORD
FIELD NAME=D-FILENAME,LEN=8,HDR='THIS IS,YOUR,FILE NAME'
FIELD NAME=DICT-CODE,LEN=1
FIELD NAME=FIELD-NAME,LEN=14,START=12
FIELD NAME=DATA-START,LEN=4,PRTDEC=0,START=27
FIELD NAME=DATA-END,LEN=4,START=32
FIELD NAME=DATA-TYPE,LEN=1,START=37
FIELD NAME=PRINT-SPECS,LEN=2
FIELD NAME=DECIMALS,LEN=1
FIELD NAME=OVER-RIDE-HDR,LEN=30,START=51,
    HDR='DEFAULT COLUMN HEADING,'
:BEGIN JCL=ALL,USAGE=SEL
// DLBL  QW$DICT,'@@@CCC.@@@DDD.DICT',,VSAM
:CREATE DEPTDESC
MAXENT=100
ARGLOC=1
ARGLEN=3
FUNLOC=5
FUNLEN=20
FUNAME=DEPT-DESC
:BEGIN TABLE
```

```
001 PAYROLL
005 COST ACCOUNTING
007 MIS
010 PRICING
015 MAIL ROOM
:CREATE LIM0100
:BEGIN PROC
     LIMITREADS 0100
:CREATE LIM0500
:BEGIN PROC
     LIMITREADS 0500
:CREATE LIM1000
:BEGIN PROC
     LIMITREADS 1000
:ACCESS PAYROLL
:LIST ALL
:LIST FIELDS
/*
/&
* $$ EOJ
```

DICTLOAD

```
* $$ JOB JNM=QWDICTLD,CLASS=A,LDEST=(,USERID)
* $$ LST CLASS=A,DEST=(,USERID)
* $$ PUN CLASS=A,DEST=(,USERID)
// JOB QWDICTLD - LOAD DEMO PAYROLL FILE
// OPTION LOG,NODUMP
/*
/*  LOAD DEMO PAYROLL FILE
/*
/* PLEASE DO A "MASS CHANGE" TO THE FOLLOWING
/* JCL TO FIT YOUR COMPANY'S STANDARDS (WHERE APPLICABLE):
/*        ',USERID' TO: IF UNDER VM, YOUR CMS USERID
/*        '**$$'    TO: '* $$'
/*        '@@@VLL'  TO: VOLSER OF QJ LIBRARY
/*        '@@@AAA.@@@BBB.LIB' TO: QJ LIBRARY NAME
/*        'SYSWK1'  TO: VOLSER OF A WORK PACK
/*        'VOLSER'  TO: VOLSER OF DEMO FILE
/*
// DLBL   QJLIB,'@@@AAA.@@@BBB.LIB',99/366    QJ LIBRARY
// EXTENT       ,@@@VLL
// LIBDEF  PHASE,SEARCH=(QJLIB.PTF,QJLIB.QJ161),                        X
               CATALOG=QJLIB.QJ161
// LIBDEF SOURCE,SEARCH=(QJLIB.PTF,QJLIB.QJ161)
// LIBDEF    OBJ,SEARCH=(QJLIB.PTF,QJLIB.QJ161)
// ASSGN SYS010,DISK,VOL=VOLSER,SHR
// DLBL     OFA,'QW.DEMO.PAYROLL',1
// EXTENT SYS010,VOLSER,1,0,001,5
// DLBL  SORTWK1,'QW.WORK',0
// EXTENT SYS003,SYSWK1,1,0,1,60
// EXEC QUKBJOB,SIZE=512K
INFCARD
OFADISC00800080SSYS010
    SORT FILE INF ON INF1-25
100 GET
    MOVE INF1-80 TO OFA1
    WRITE OFA
    GO TO 100
9999END
```

```
(inline data)
.
.
.
/*
/&
* $$ EOJ
```

Index

A

B

C

D

E

F

I

J

L

O

P

Q

S

T

U

V

Z